SunFounder ESP32 Starter Kit

www.sunfounder.com

26.03.2024

Inhaltsverzeichnis

Danke, dass Sie sich für unser ESP32 Starter Kit entschieden haben.

Bemerkung: Dieses Dokument ist in den folgenden Sprachen verfügbar.

-
-
-
-

Bitte klicken Sie auf die jeweiligen Links, um das Dokument in Ihrer bevorzugten Sprache aufzurufen.

Willkommen beim ESP32 Lernkit! Dieses umfassende Paket ist sowohl für Anfänger als auch für erfahrene Entwickler konzipiert, um einen tiefen Einblick in die vielseitige Welt des ESP32-Mikrocontrollers zu erhalten. Mit dem ESP32 WROOM 32E als Herzstück und einer Reihe von begleitenden Komponenten wie LEDs, Sensoren, Motoren und mehr können Benutzer eine Vielzahl von Projekten erkunden.

Ob Sie sich für Grundlagen der Elektronik oder IoT-Integrationen interessieren, dieses Kit bietet alles. Für MicroPython-Begeisterte bieten wir eine strukturierte Einführung in MicroPython, komplett mit Einrichtungen der IDE und Grundlagen der Syntax. Auch Arduino-Anwender kommen nicht zu kurz, mit einem eigenen Abschnitt zum Einstieg in Arduino und einer Sammlung von Grundprojekten, um den Lernprozess zu starten.

Für Kreative gibt es einen reizvollen Abschnitt zur Integration mit Scratch, der Programmierung und Geschichtenerzählen verbindet. Jedes Projekt im Kit ist akribisch dargestellt, um sicherzustellen, dass Sie die Ziele, den Schaltungsanbau und die Programmieraspekte verstehen.

Mit einer Fülle von Spielprojekten, praktischen Anwendungen und FAQs zur Fehlersuche verspricht dieses Kit eine bereichernde Lernerfahrung für alle. Tauchen Sie ein und lassen Sie das ESP32-Abenteuer beginnen!

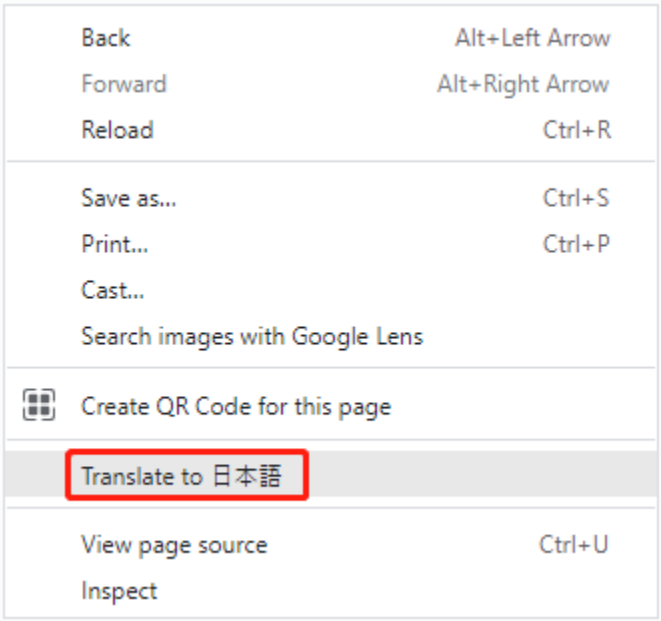
Wenn Sie Fragen haben oder andere interessante Ideen, zögern Sie nicht, eine E-Mail an service@sunfounder.com zu senden.

Über die Anzeigesprache

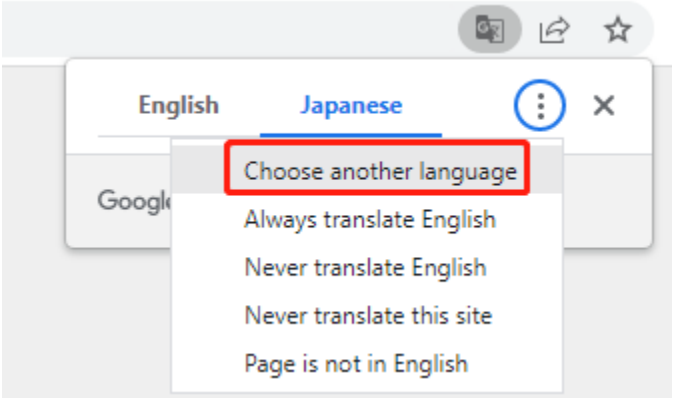
Zusätzlich zu Englisch arbeiten wir an anderen Sprachen für diesen Kurs. Bitte kontaktieren Sie service@sunfounder.com , wenn Sie helfen möchten, und wir werden Ihnen als Dank ein kostenloses Produkt geben. In der Zwischenzeit empfehlen wir, Google Translate zu verwenden, um Englisch in die Sprache zu übersetzen, die Sie sehen möchten.

Die Schritte sind wie folgt.

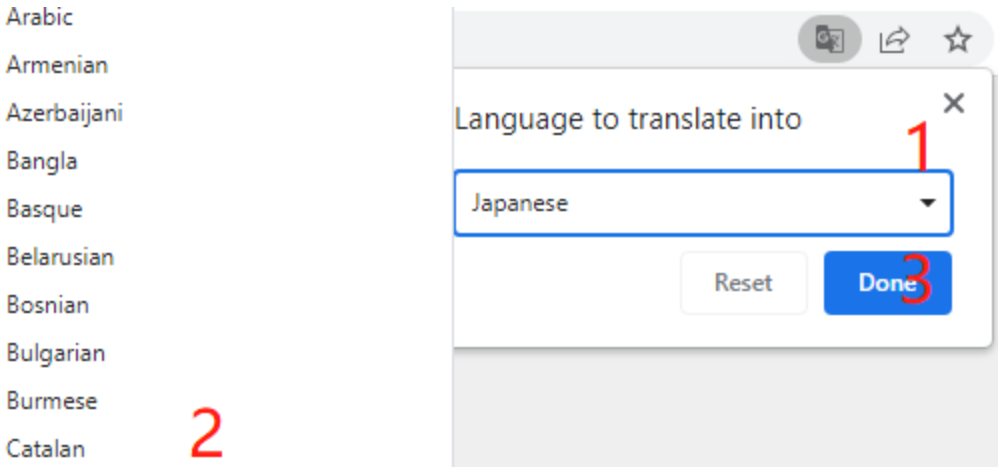
- Rechtsklicken Sie auf dieser Kursseite und wählen Sie **Translate to xx**. Wenn die aktuelle Sprache nicht die gewünschte ist, können Sie sie später ändern.



- Es erscheint ein Sprach-Popup in der oberen rechten Ecke. Klicken Sie auf das Menüsymbol, um **choose another language**.



- Wählen Sie aus dem umgekehrten Dreieckkasten die Sprache aus und klicken Sie dann auf **Done**.


Arabic
Armenian
Azerbaijani
Bangla
Basque
Belarusian
Bosnian
Bulgarian
Burmese
Catalan
2
Language to translate into
1
Japanese
Reset
Done
3

KAPITEL 1

Laden Sie den Code herunter

Hier ist das vollständige Code-Paket für das ESP32 Starter Kit. Sie können auf den folgenden Link klicken, um es herunterzuladen:

`SunFounder ESP32 Starter Kit`

Sobald der Download abgeschlossen ist, entpacken Sie die Datei und öffnen Sie den entsprechenden Beispielcode oder Projektdateien in der entsprechenden Software. Dadurch können Sie den gesamten Code und die bereitgestellten Ressourcen des Kits durchsuchen und nutzen.

KAPITEL 2

Für Arduino-Benutzer

Hier ist das vollständige Code-Paket für das ESP32 Starter Kit. Sie können auf den folgenden Link klicken, um es herunterzuladen:

- `SunFounder ESP32 Starter Kit`

Sobald der Download abgeschlossen ist, entpacken Sie die Datei und öffnen Sie den entsprechenden Beispielcode oder Projektdateien in der entsprechenden Software. Dadurch können Sie den gesamten Code und die bereitgestellten Ressourcen des Kits durchsuchen und nutzen.

1. Erste Schritte

2.1 1.1 Arduino IDE installieren (Wichtig)

Die Arduino-IDE, bekannt als Arduino Integrated Development Environment, bietet alle Softwareunterstützung, die für die Durchführung eines Arduino-Projekts erforderlich ist. Es handelt sich um eine speziell für Arduino entwickelte Programmiersoftware, die vom Arduino-Team bereitgestellt wird und es uns ermöglicht, Programme zu schreiben und sie auf das Arduino-Board hochzuladen.

Die Arduino-IDE 2.0 ist ein Open-Source-Projekt. Sie ist ein großer Schritt von ihrem robusten Vorgänger Arduino IDE 1.x entfernt und bietet eine überarbeitete Benutzeroberfläche, einen verbesserten Board- und Bibliotheksmanager, einen Debugger, eine Autovervollständigungsfunktion und vieles mehr.

In diesem Tutorial zeigen wir, wie Sie die Arduino-IDE 2.0 auf Ihrem Windows-, Mac- oder Linux-Computer herunterladen und installieren können.

2.1.1 Anforderungen

- Windows - Win 10 und neuer, 64 Bit
- Linux - 64 Bit
- Mac OS X - Version 10.14: „Mojave“ oder neuer, 64 Bit

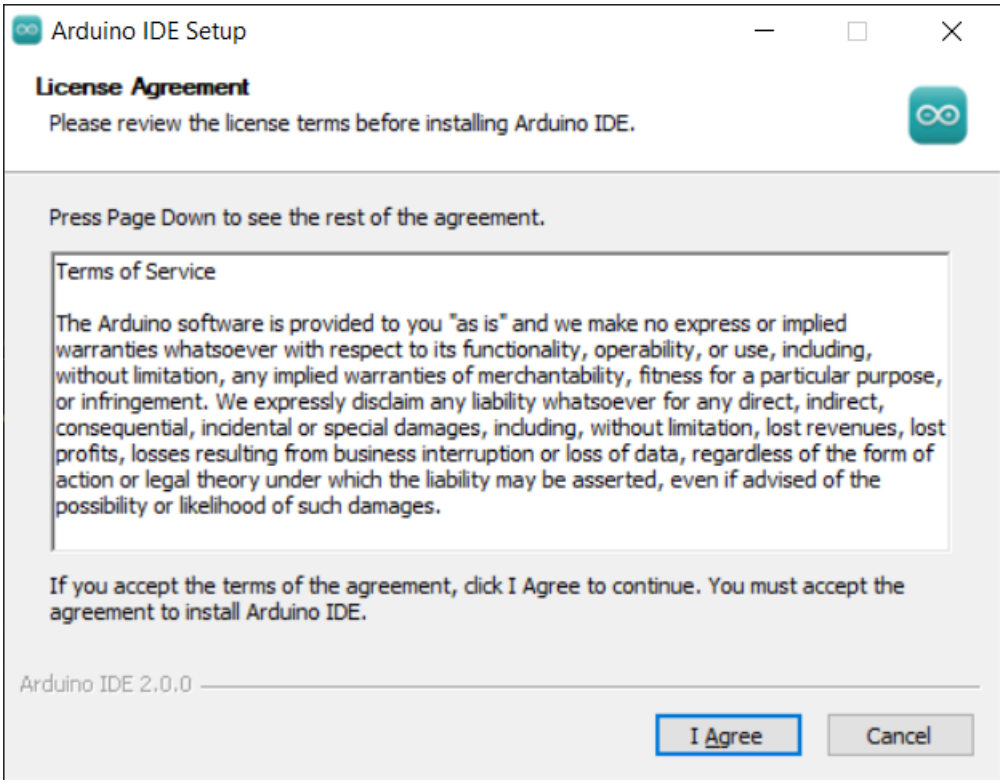
2.1.2 Arduino-IDE 2.0 herunterladen

1. Besuchen Sie .
2. Laden Sie die IDE für Ihre Betriebssystemversion herunter.

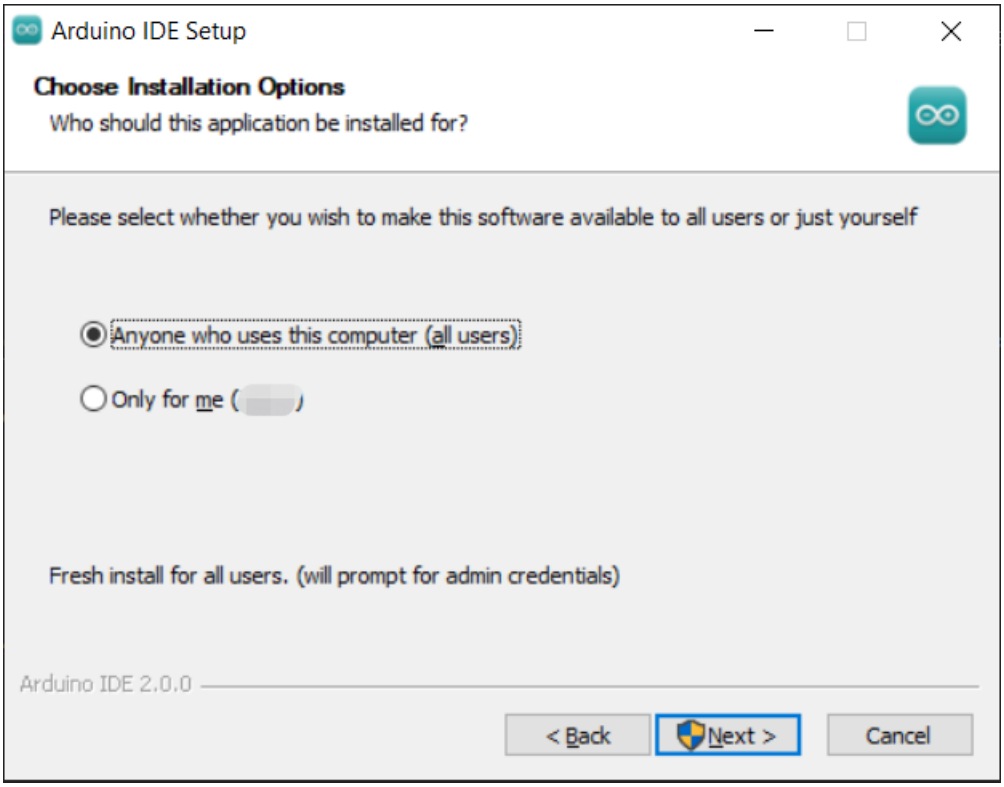
2.1.3 Installation

Windows

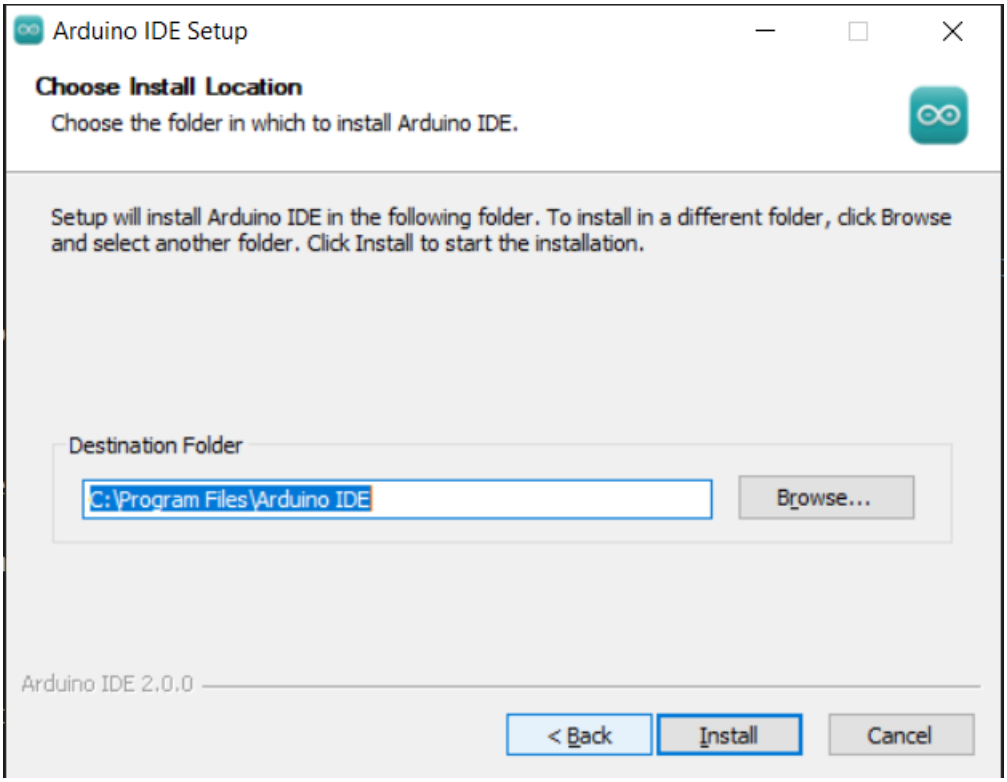
1. Doppelklicken Sie auf die Datei `arduino-ide_xxxx.exe`, um die heruntergeladene Datei auszuführen.
2. Lesen Sie die Lizenzvereinbarung und stimmen Sie ihr zu.



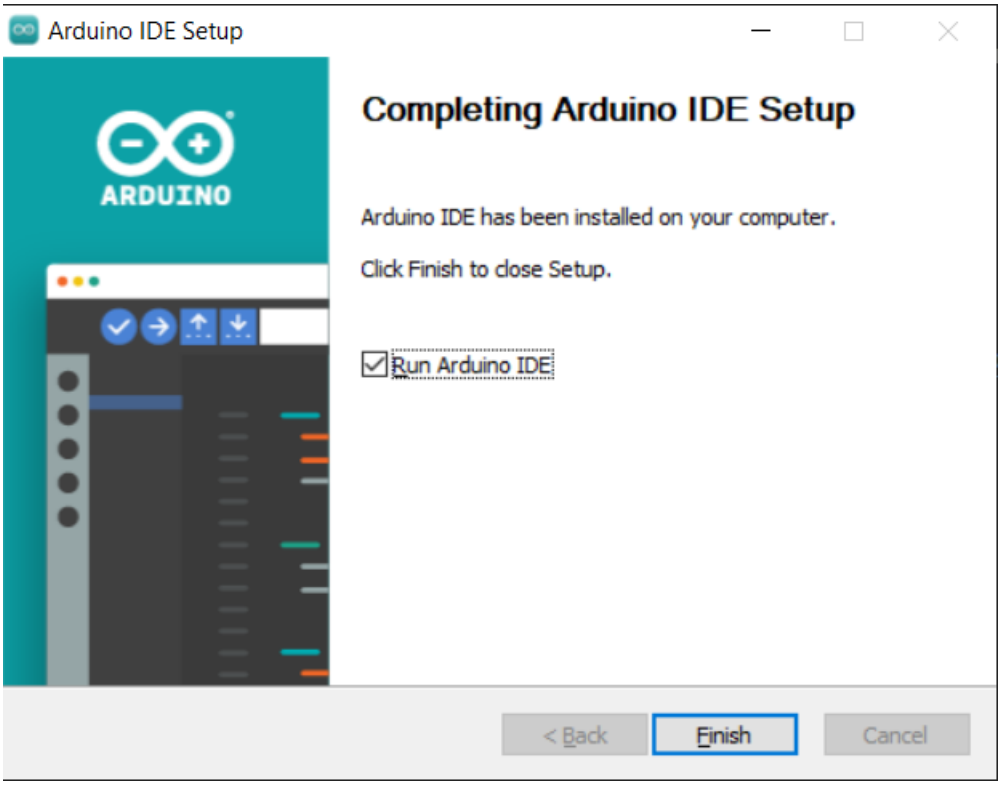
3. Wählen Sie die Installationsoptionen aus.



4. Wählen Sie den Installationsort. Es wird empfohlen, die Software auf einem Laufwerk außerhalb des Systemlaufwerks zu installieren.

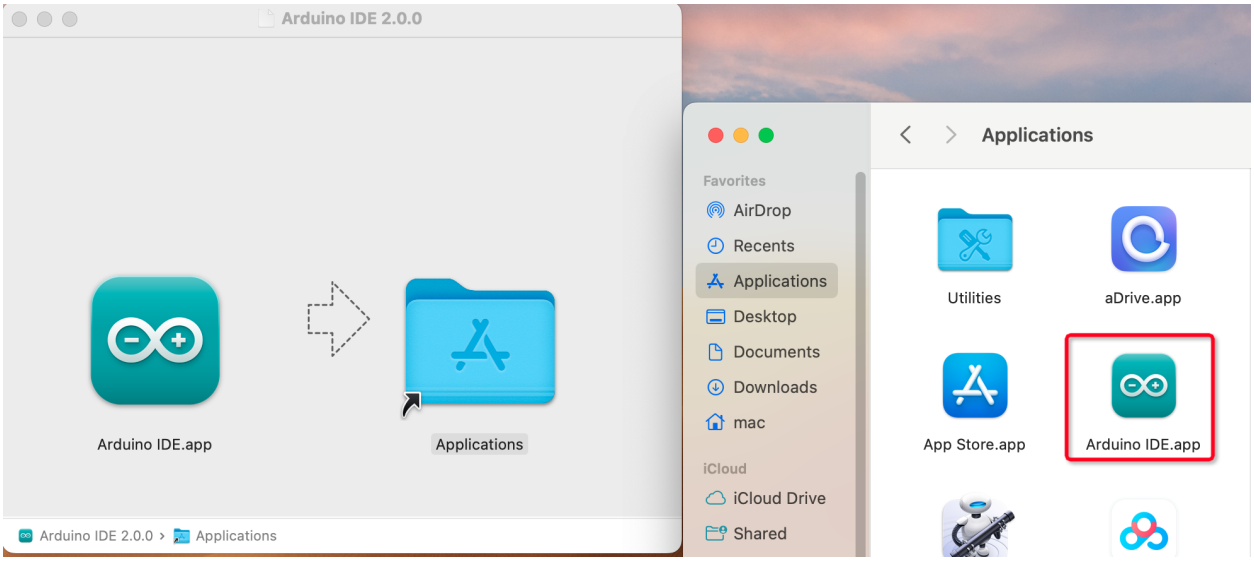


5. Klicken Sie dann auf „Fertig“.



macOS

Doppelklicken Sie auf die heruntergeladene Datei `arduino_ide_xxxx.dmg` und folgen Sie den Anweisungen, um die **Arduino IDE.app** in den **Applications**-Ordner zu kopieren. Nach einigen Sekunden wird die Arduino-IDE erfolgreich installiert.

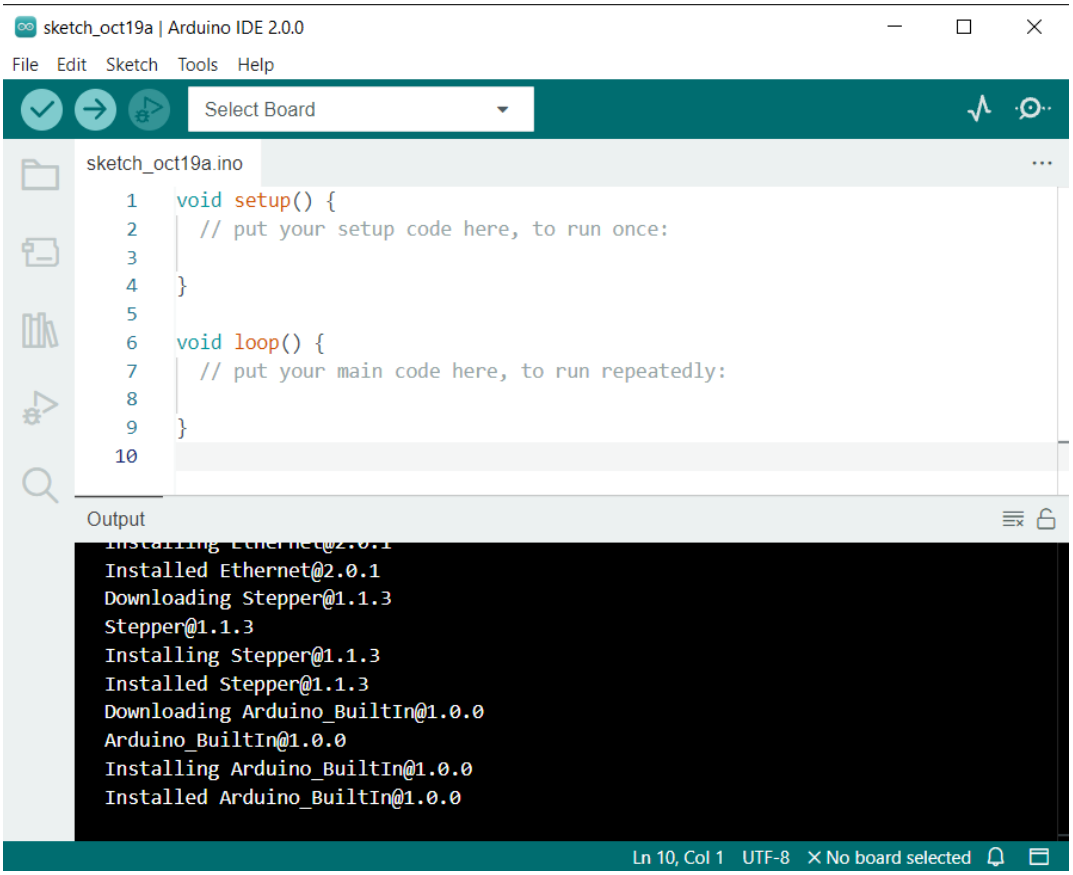


Linux

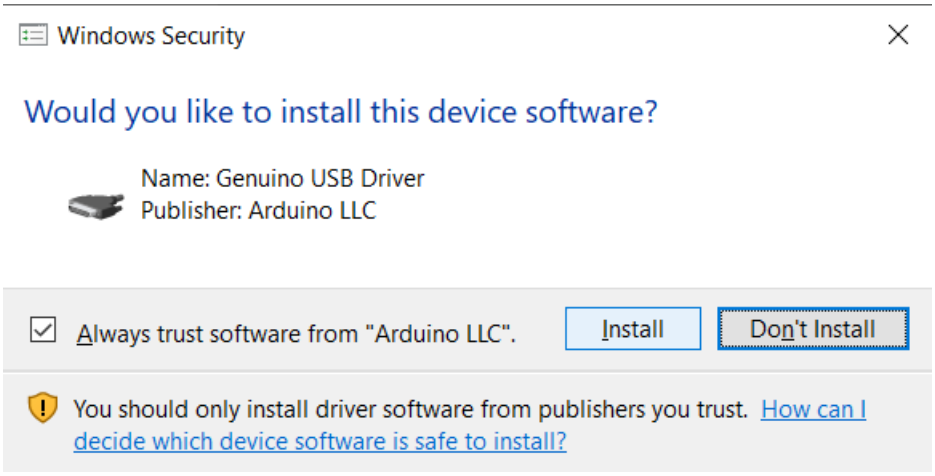
Für das Tutorial zur Installation der Arduino-IDE 2.0 auf einem Linux-System verweisen Sie bitte auf: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing#linux

2.1.4 Öffnen Sie die IDE

1. Wenn Sie die Arduino-IDE 2.0 zum ersten Mal öffnen, werden automatisch die Arduino AVR Boards, integrierte Bibliotheken und andere erforderliche Dateien installiert.



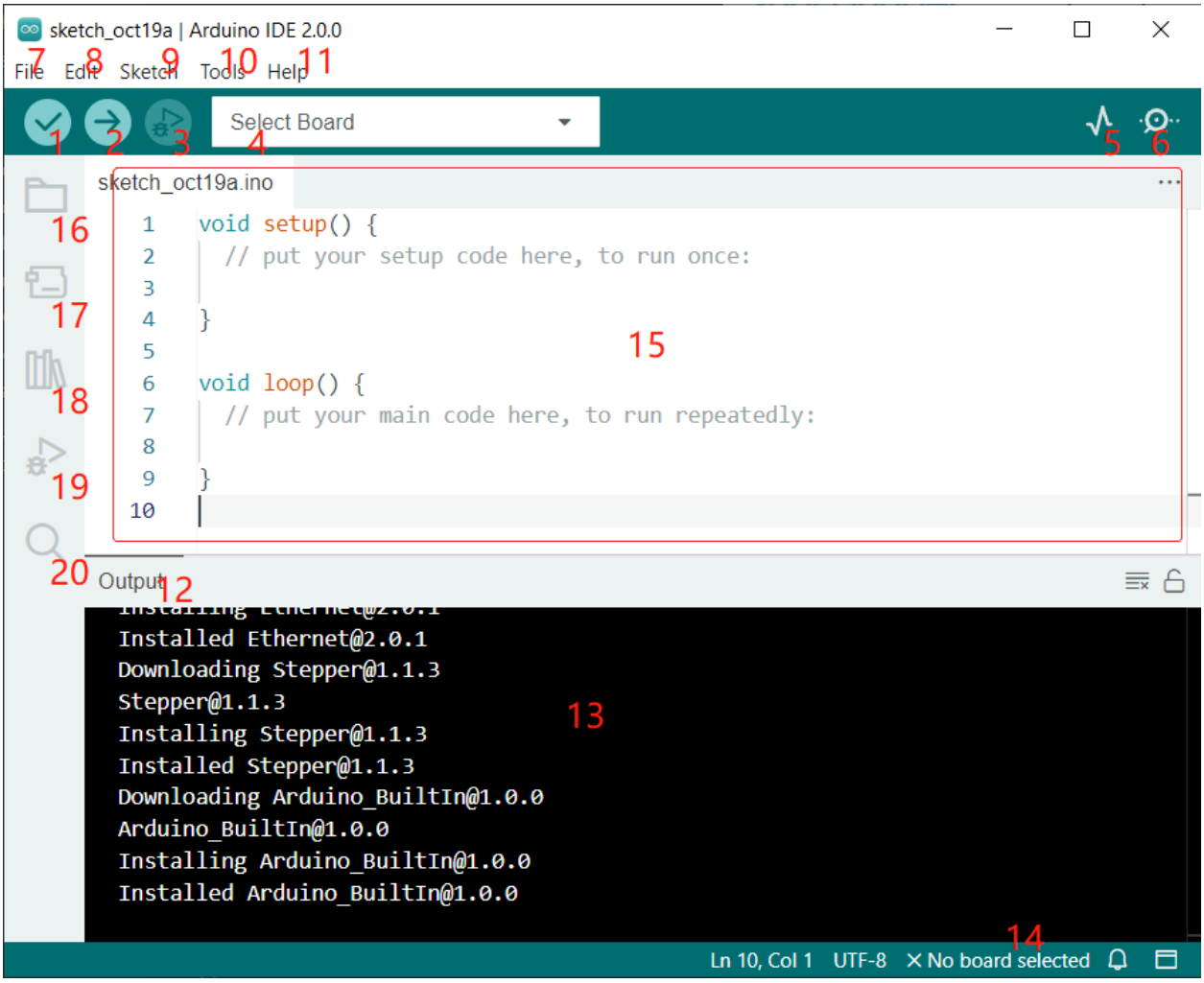
2. Darüber hinaus kann Ihre Firewall oder Ihr Sicherheitscenter mehrmals Pop-ups anzeigen und Sie fragen, ob Sie einige Gerätetreiber installieren möchten. Installieren Sie bitte alle.



3. Jetzt ist Ihre Arduino-IDE bereit!

Bemerkung: Falls einige Installationen aufgrund von Netzwerkproblemen oder anderen Gründen nicht funktionierten, können Sie die Arduino-IDE erneut öffnen und sie wird den Rest der Installation abschließen. Das Ausgabefenster wird sich erst öffnen, nachdem alle Installationen abgeschlossen sind, es sei denn, Sie klicken auf „Verify“ oder „Upload“.

2.2 1.2 Einführung in die Arduino-IDE



1. **Verify**: Kompilieren Sie Ihren Code. Etwaige Syntaxprobleme werden mit Fehlern angezeigt.
2. **Upload**: Laden Sie den Code auf Ihr Board hoch. Wenn Sie auf die Schaltfläche klicken, blinken die RX- und TX-LEDs auf dem Board schnell und hören erst auf, wenn der Upload abgeschlossen ist.
3. **Debug**: Für eine Fehlerprüfung Zeile für Zeile.
4. **Select Board**: Schnelle Einrichtung von Board und Port.
5. **Serial Plotter**: Überprüfen Sie die Änderung des Messwerts.
6. **Serial Monitor**: Klicken Sie auf die Schaltfläche, und ein Fenster wird angezeigt. Es empfängt die Daten, die von Ihrem Steuerungsboard gesendet werden. Es ist sehr nützlich für die Fehlersuche.
7. **File**: Klicken Sie auf das Menü, und es wird eine Dropdown-Liste angezeigt, einschließlich Dateierstellung, Öffnen, Speichern, Schließen, einige Parameterkonfigurationen usw.
8. **Edit**: Klicken Sie auf das Menü. In der Dropdown-Liste gibt es einige Bearbeitungsvorgänge wie **Cut**, **Copy**, **Paste**, **Find** usw., mit ihren entsprechenden Tastenkombinationen.
9. **Sketch**: Enthält Operationen wie **Verify**, **Upload**, **Add** usw. Eine wichtige Funktion ist **Include Library**, wo Sie Bibliotheken hinzufügen können.

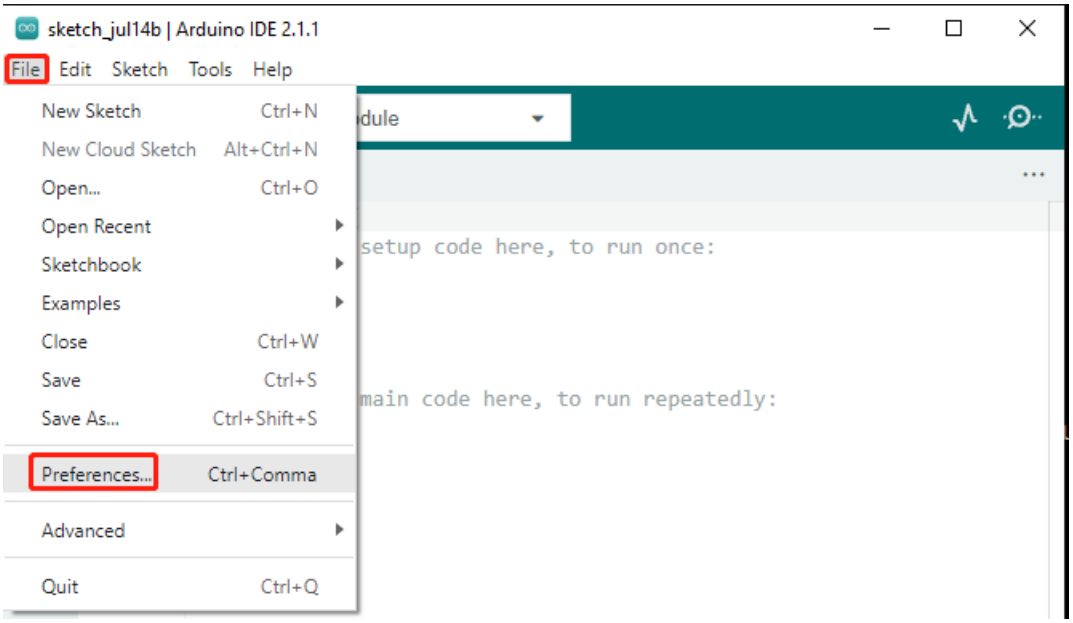
10. **Tool**: Enthält einige Werkzeuge - das am häufigsten verwendete Board (das Board, das Sie verwenden) und Port (der Port, an dem sich Ihr Board befindet). Jedes Mal, wenn Sie den Code hochladen möchten, müssen Sie sie auswählen oder überprüfen.
11. **Help**: Wenn Sie Anfänger sind, können Sie die Optionen im Menü überprüfen und die benötigte Hilfe erhalten, einschließlich Bedienungshinweise zur IDE, Einführungsinformationen, Fehlerbehebung, Codeerklärungen usw.
12. **Output Bar**: Hier können Sie zwischen den Ausgabetabs wechseln.
13. **Output Window**: Hier werden Informationen ausgegeben.
14. **Board and Port**: Hier können Sie das für den Code-Upload ausgewählte Board und den Port anzeigen. Wenn etwas nicht korrekt ist, können Sie sie erneut über **Tools** -> **Board / Port** auswählen.
15. Der Bearbeitungsbereich der IDE. Hier können Sie Code schreiben.
16. **Sketchbook**: Zur Verwaltung von Sketch-Dateien.
17. **Board Manager**: Zur Verwaltung des Board-Treibers.
18. **Library Manager**: Zur Verwaltung Ihrer Bibliotheksdateien.
19. **Debug**: Hilft bei der Fehlerbehebung von Code.
20. **Search**: Durchsuchen Sie Ihren Sketch-Code.

2.3 1.3 ESP32-Board installieren (Wichtig)

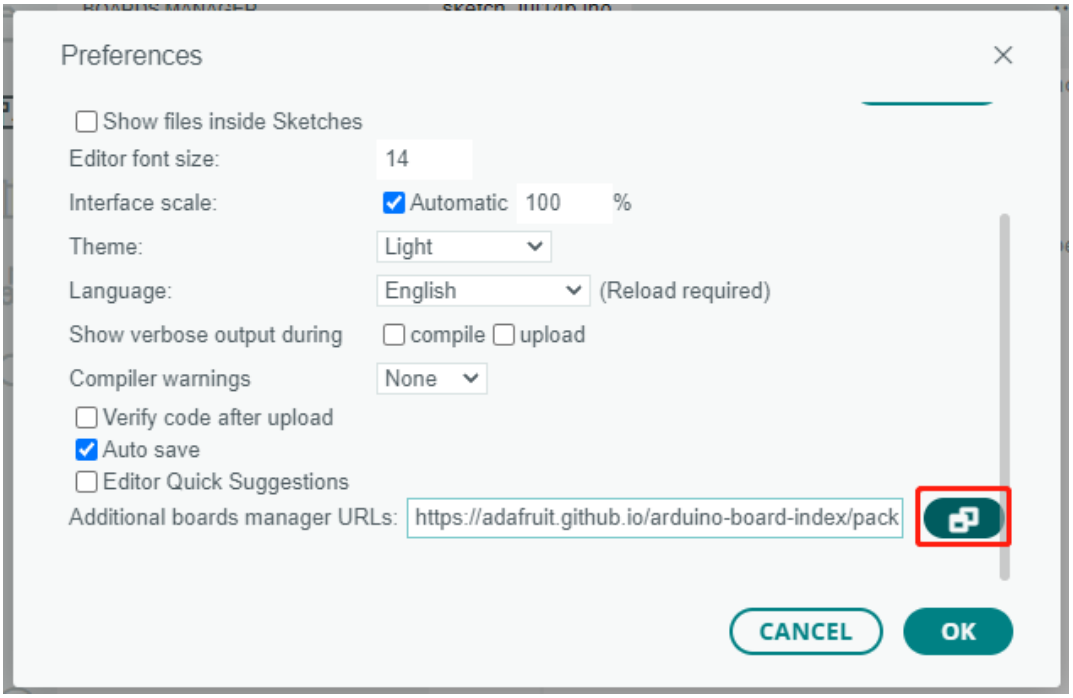
Um den ESP32-Mikrocontroller zu programmieren, müssen wir das ESP32-Board-Paket in der Arduino-IDE installieren. Befolgen Sie die folgende Schritt-für-Schritt-Anleitung:

ESP32-Board installieren

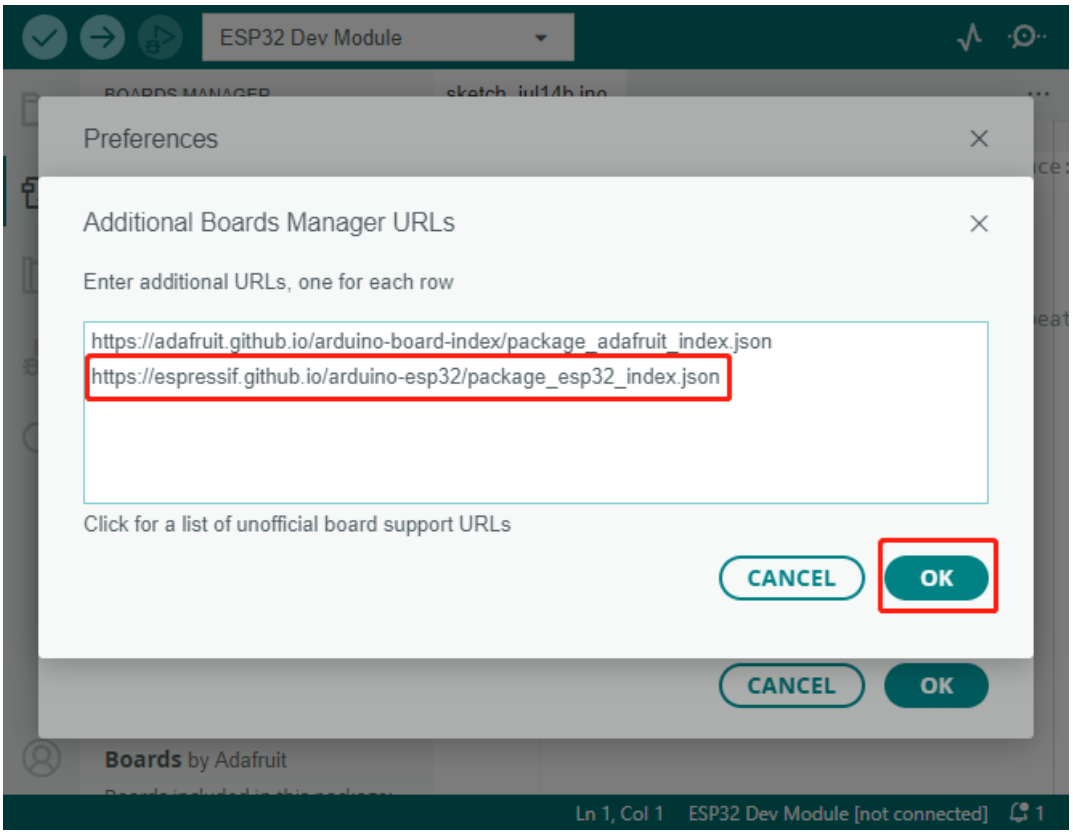
1. Öffnen Sie die Arduino-IDE. Gehen Sie zu **File** und wählen Sie **Preferences** aus dem Dropdown-Menü.



2. In dem Einstellungen-Fenster finden Sie das Feld **Additional Board Manager URLs**. Klicken Sie darauf, um das Textfeld zu aktivieren.

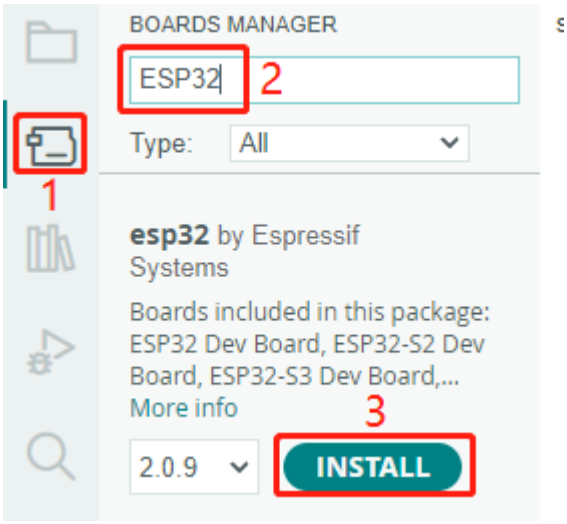


3. Fügen Sie die folgende URL in das Feld **Additional Board Manager URLs** ein: https://espressif.github.io/arduino-esp32/package_esp32_index.json. Diese URL verweist auf die Paketindexdatei für die ESP32-Boards. Klicken Sie auf die Schaltfläche **OK**, um die Änderungen zu speichern.



4. Geben Sie im **Boards Manager**-Fenster **ESP32** in die Suchleiste ein. Klicken Sie auf die Schaltfläche **Install**,

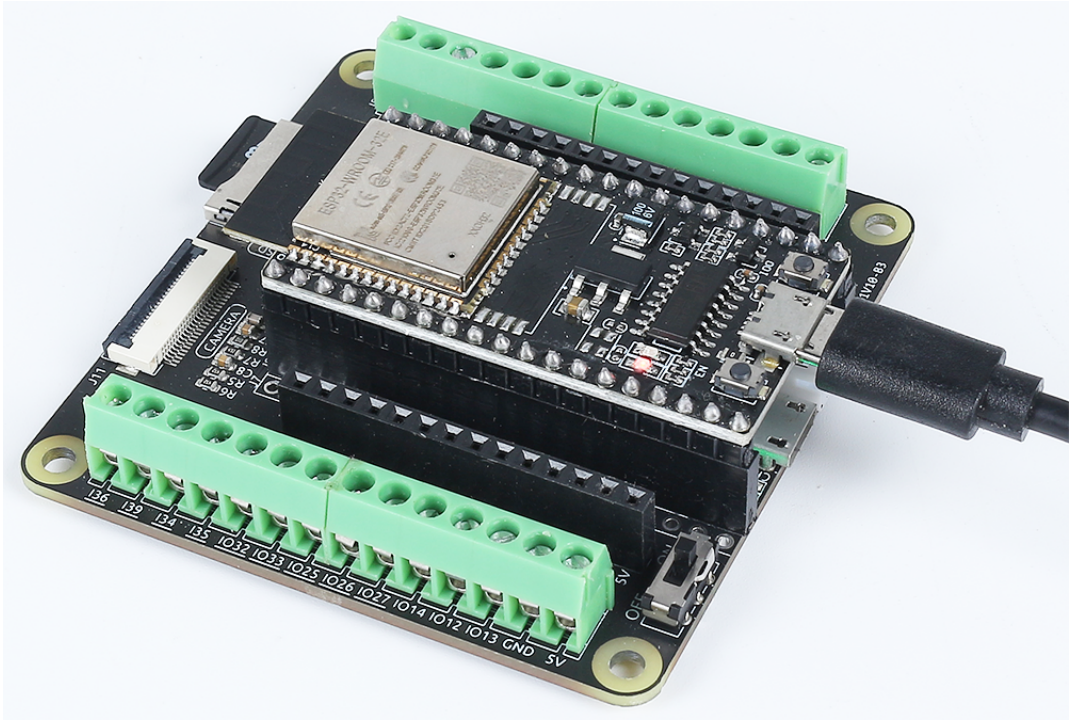
um den Installationsprozess zu starten. Dadurch wird das ESP32-Board-Paket heruntergeladen und installiert.



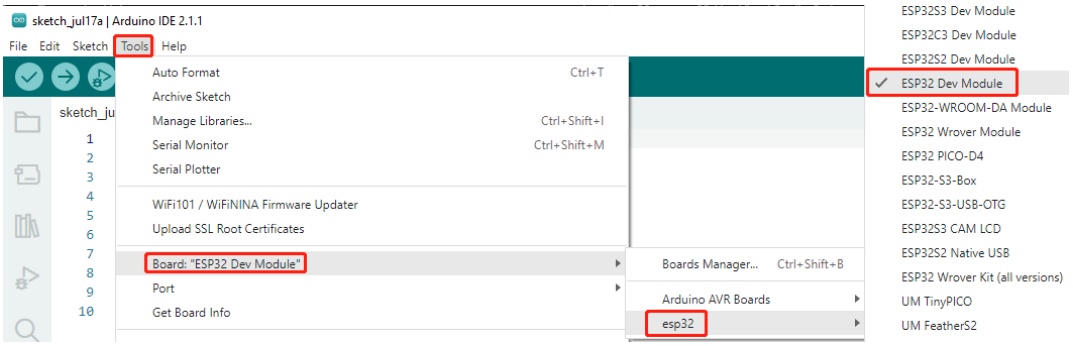
5. Herzlichen Glückwunsch! Sie haben das ESP32-Board-Paket erfolgreich in der Arduino-IDE installiert.

Laden Sie den Code hoch

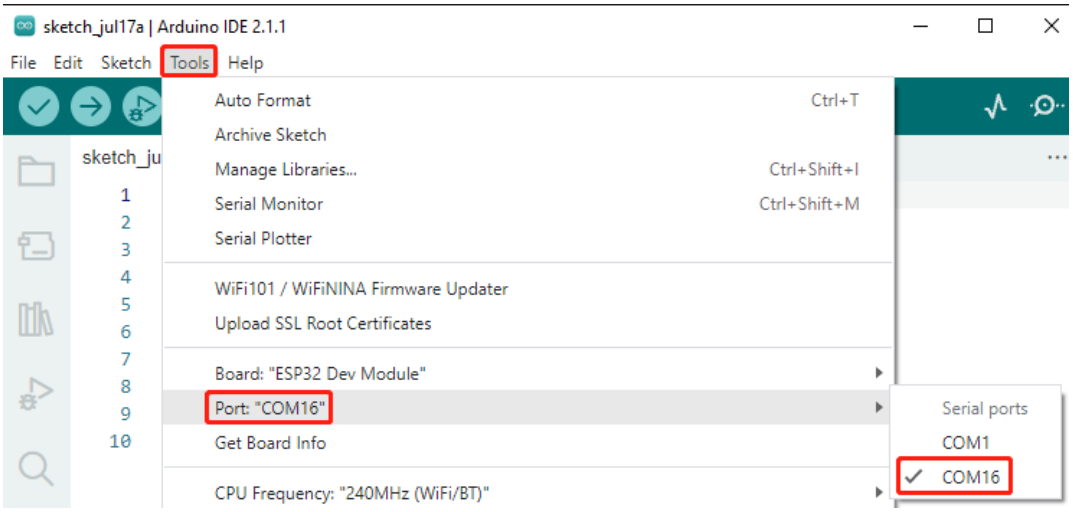
1. Verbinden Sie jetzt das ESP32 WROOM 32E mit Ihrem Computer über ein Micro-USB-Kabel.



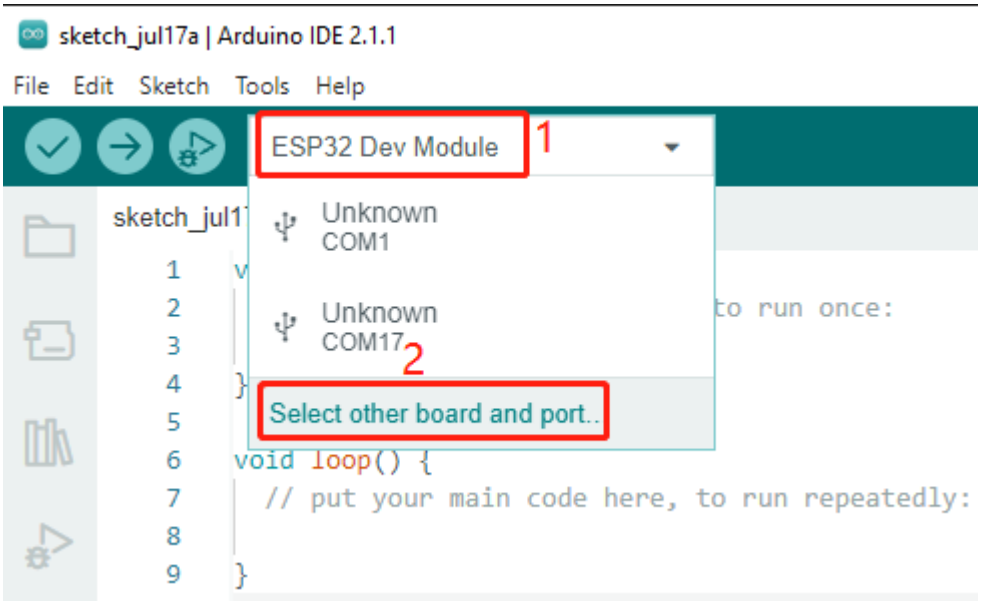
2. Wählen Sie dann das richtige Board, ESP32 Dev Module, aus, indem Sie auf Werkzeuge -> Board -> esp32 klicken.



3. Wenn Ihr ESP32 mit dem Computer verbunden ist, können Sie den richtigen Port auswählen, indem Sie auf **Tools** -> **Port** klicken.

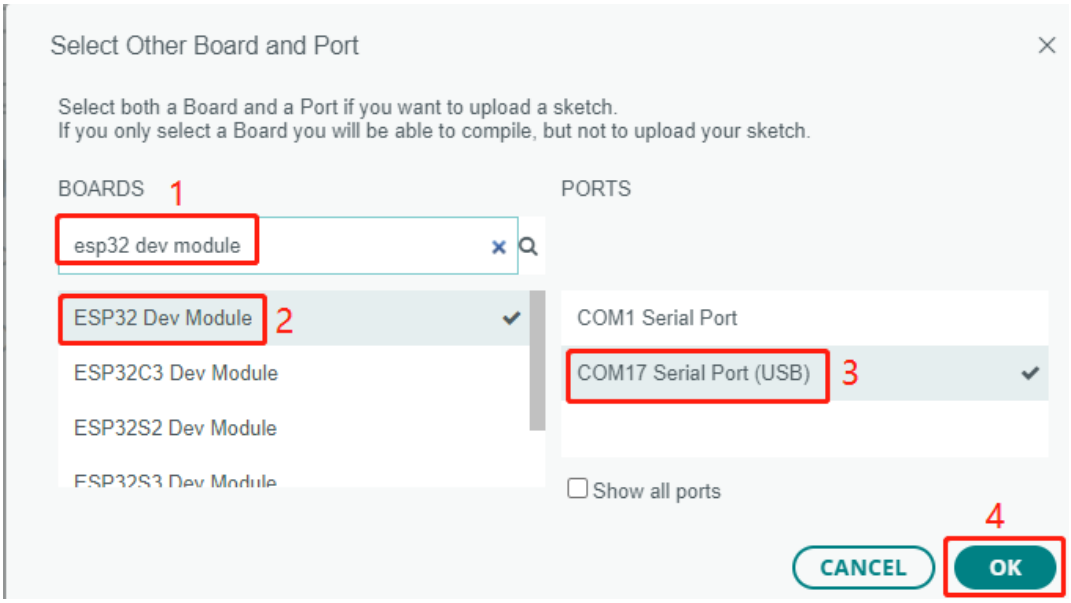


4. Zusätzlich wurde in Arduino 2.0 eine neue Möglichkeit eingeführt, das Board und den Port schnell auszuwählen. Für ESP32 wird es normalerweise nicht automatisch erkannt, daher müssen Sie auf **Select other board and port** klicken.



5. Geben Sie im Suchfeld **ESP32 Dev Module** ein und wählen Sie es aus, wenn es angezeigt wird. Wählen Sie

dann den richtigen Port und klicken Sie auf **OK**.



6. Danach können Sie es über dieses Schnellzugriffsfenster auswählen. Beachten Sie, dass es bei der späteren Verwendung Zeiten geben kann, in denen ESP32 im Schnellzugriffsfenster nicht verfügbar ist und Sie die oben genannten beiden Schritte wiederholen müssen.

7. Beide Methoden ermöglichen Ihnen die Auswahl des richtigen Boards und Ports. Wählen Sie diejenige, die Ihnen am besten gefällt. Jetzt ist alles bereit, um den Code auf den ESP32 hochzuladen.

2.4 1.4 Bibliotheken installieren (Wichtig)

Eine Bibliothek ist eine Sammlung von vorab geschriebenem Code oder Funktionen, die die Möglichkeiten der Arduino-IDE erweitern. Bibliotheken stellen fertigen Code für verschiedene Funktionen bereit und ermöglichen es Ihnen, Zeit und Mühe bei der Programmierung komplexer Features zu sparen.

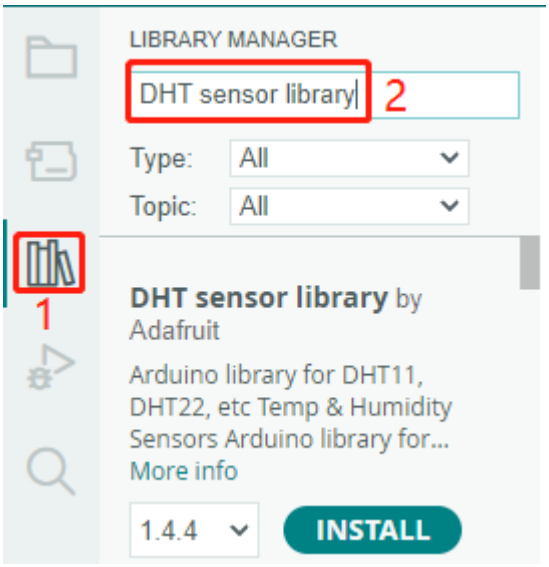
Es gibt zwei Hauptmethoden, um Bibliotheken zu installieren:

2.4.1 Installation über die Bibliotheksverwaltung

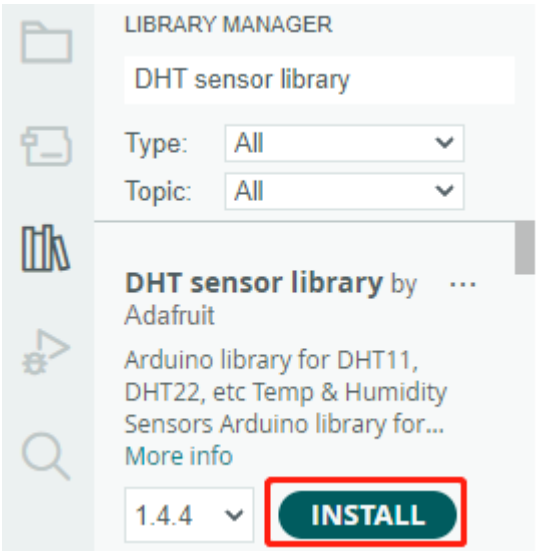
Viele Bibliotheken können direkt über die Arduino-Bibliotheksverwaltung installiert werden. Sie können die Bibliotheksverwaltung über die folgenden Schritte aufrufen:

1. In der **Library Manager** können Sie nach der gewünschten Bibliothek suchen oder verschiedene Kategorien durchsuchen.

Bemerkung: Bei Projekten, bei denen eine Bibliotheksinstallation erforderlich ist, gibt es Hinweise, welche Bibliotheken installiert werden müssen. Befolgen Sie die bereitgestellten Anweisungen wie z. B. „Die DHT-Sensorbibliothek wird hier verwendet. Sie können sie über die Bibliotheksverwaltung installieren.“ Installieren Sie einfach die empfohlenen Bibliotheken entsprechend den Hinweisen.



2. Sobald Sie die Bibliothek, die Sie installieren möchten, gefunden haben, klicken Sie darauf und anschließend auf die Schaltfläche **Install**.

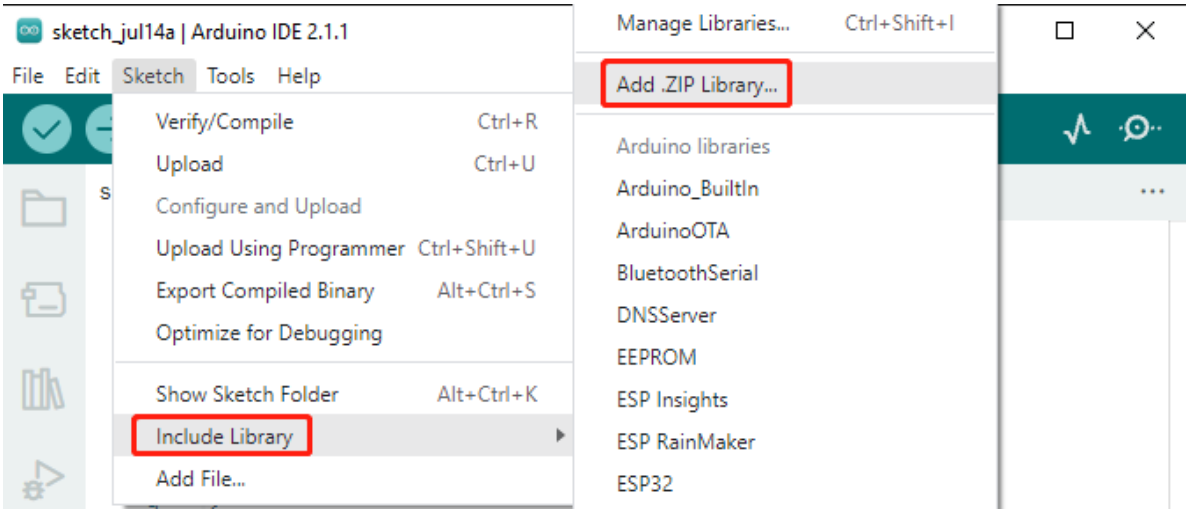


3. Die Arduino-IDE wird die Bibliothek automatisch für Sie herunterladen und installieren.

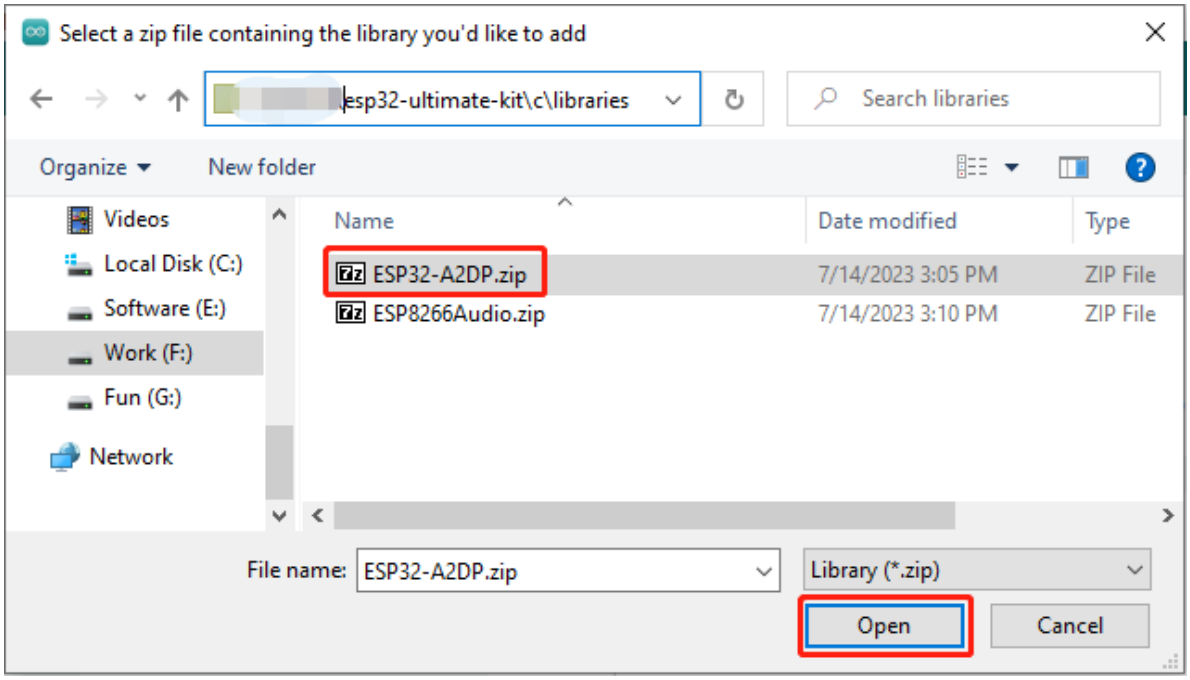
2.4.2 Manuelle Installation

Einige Bibliotheken sind nicht über die **Library Manager** verfügbar und müssen manuell installiert werden. Befolgen Sie diese Schritte, um diese Bibliotheken zu installieren:

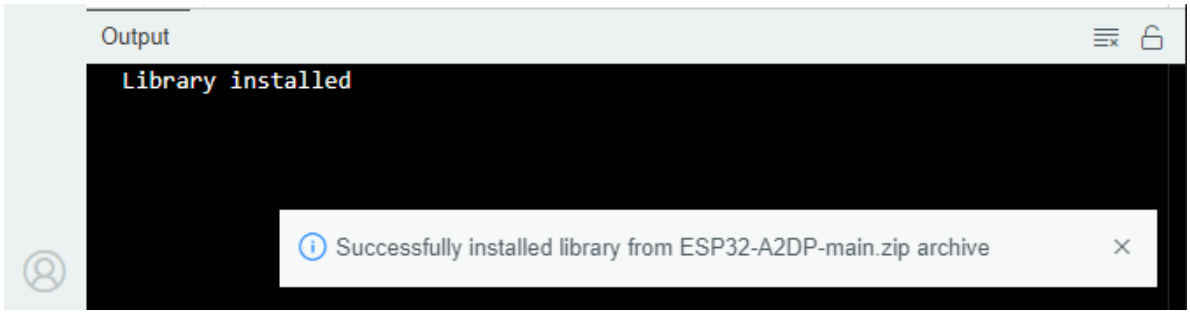
1. Öffnen Sie die Arduino-IDE und gehen Sie zu **Sketch** -> **Include Library** -> **Add .ZIP Library**.



2. Navigieren Sie zum Verzeichnis, in dem sich die Bibliotheksdateien befinden, z. B. zum Ordner `esp32-starter-kit\c\libraries`, und wählen Sie die gewünschte Bibliotheksdatei, z. B. `ESP32-A2DP.zip`, aus. Klicken Sie dann auf **Open**.



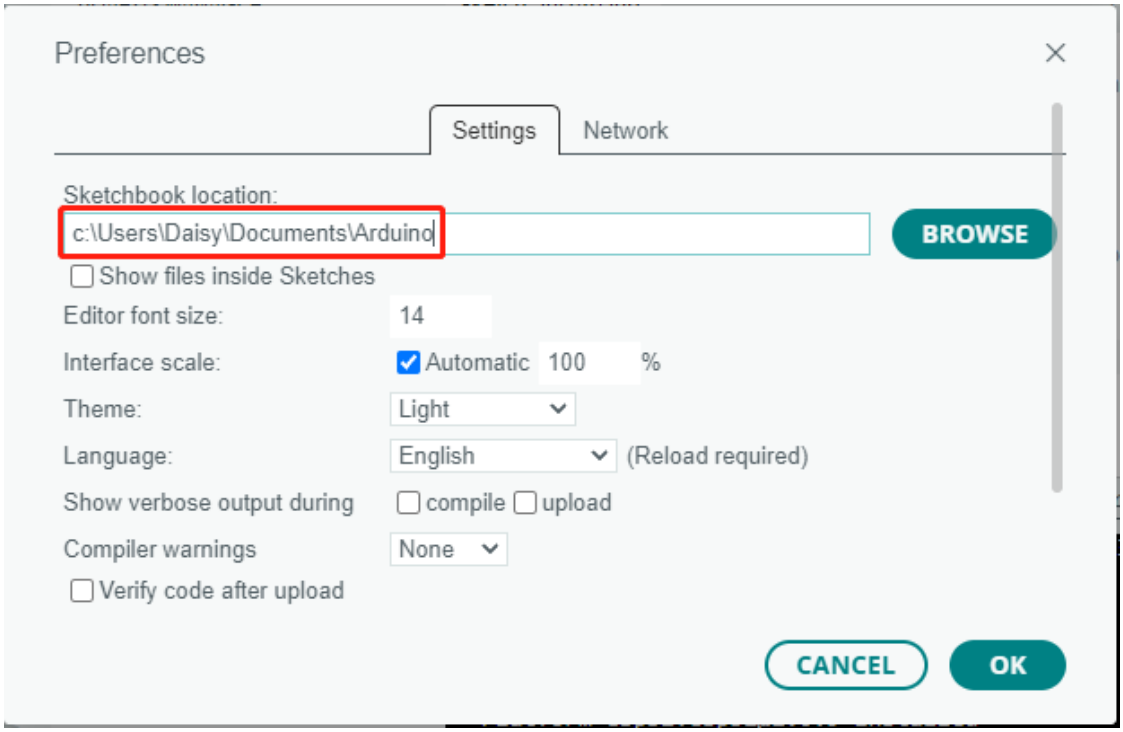
3. Nach kurzer Zeit erhalten Sie eine Benachrichtigung über eine erfolgreiche Installation.



4. Wiederholen Sie den gleichen Vorgang, um die Bibliothek `ESP8266Audio.zip` hinzuzufügen.

Bemerkung: Die über eine der oben genannten Methoden installierten Bibliotheken finden Sie im Standardbibliotheksverzeichnis der Arduino-IDE, das sich normalerweise unter `C:\Users\xxx\Documents\Arduino\libraries` befindet.

Wenn sich Ihr Bibliotheksverzeichnis an einem anderen Ort befindet, können Sie es über **File** -> **Preferences** überprüfen.



2. Displays

2.5 2.1 Hallo, LED!

Genau wie das Drucken von „Hallo, Welt!“ der erste Schritt beim Erlernen des Programmierens ist, ist die Verwendung eines Programms zur Steuerung einer LED die traditionelle Einführung in das Erlernen der physischen Programmierung.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Du kannst sie auch einzeln über die folgenden Links kaufen.

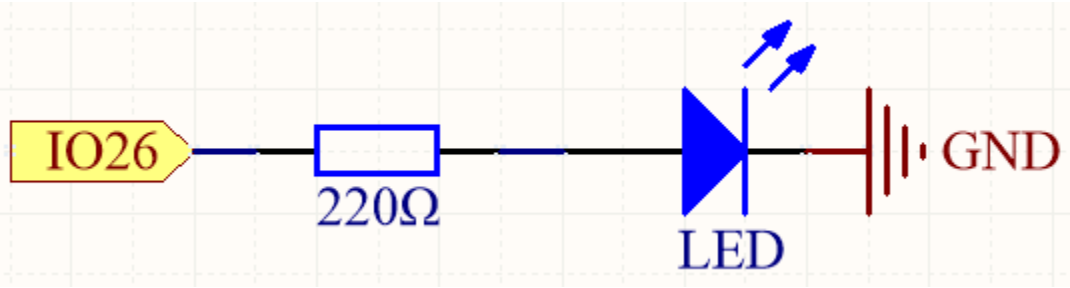
KOMPONENTENEINFÜHRUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

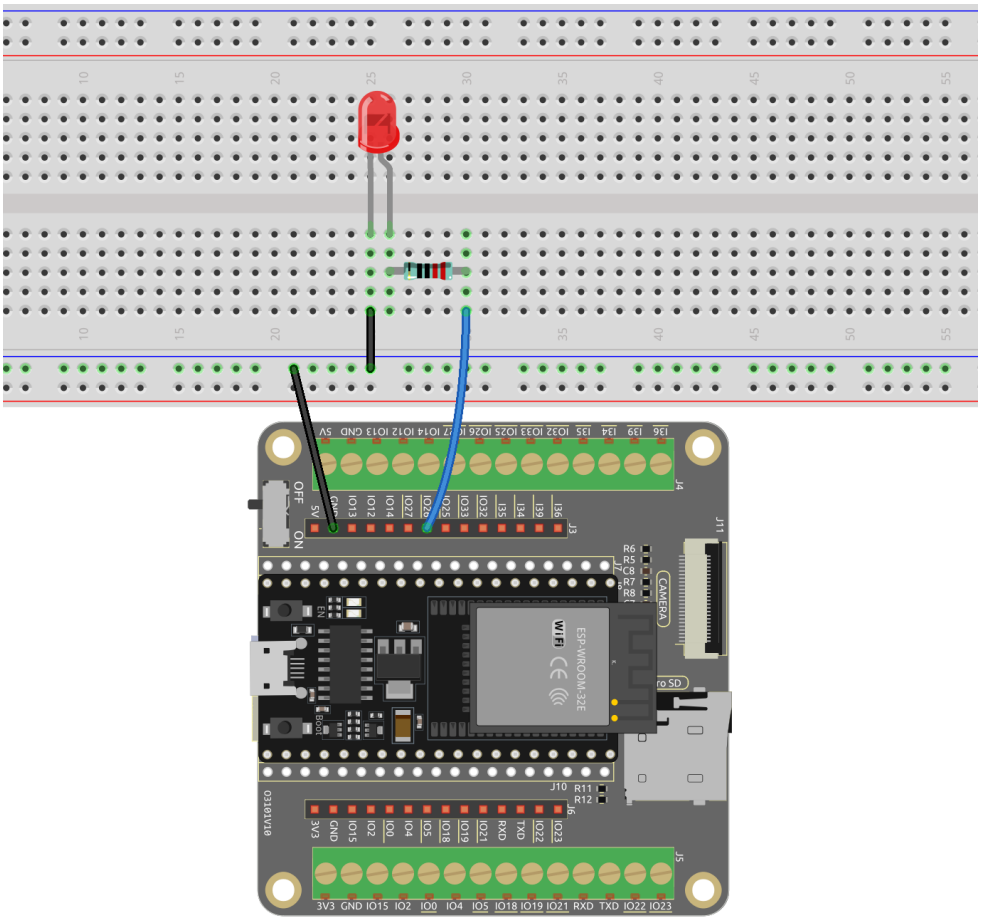
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Dieser Schaltkreis funktioniert nach einem einfachen Prinzip, und die Stromrichtung ist in der Abbildung dargestellt. Die LED leuchtet auf, nachdem der 220-Ohm-Strombegrenzungswiderstand, wenn Pin26 ein hohes Pegel ausgibt. Die LED schaltet sich aus, wenn Pin26 ein niedriges Pegel ausgibt.

Verdrahtung

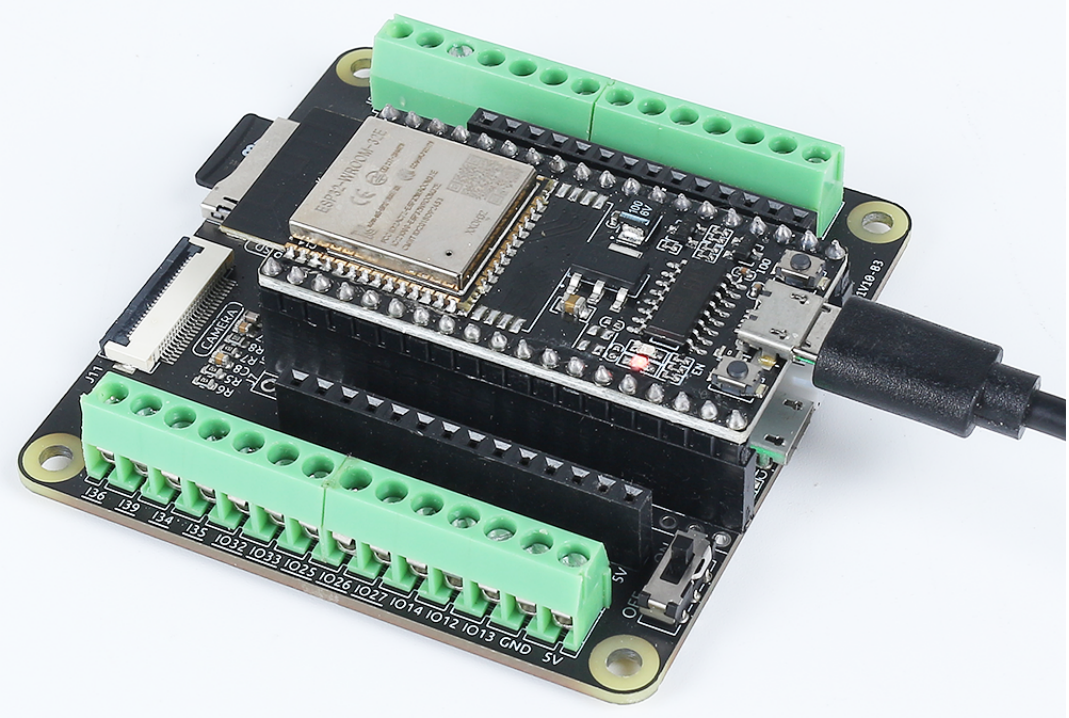


Code hochladen

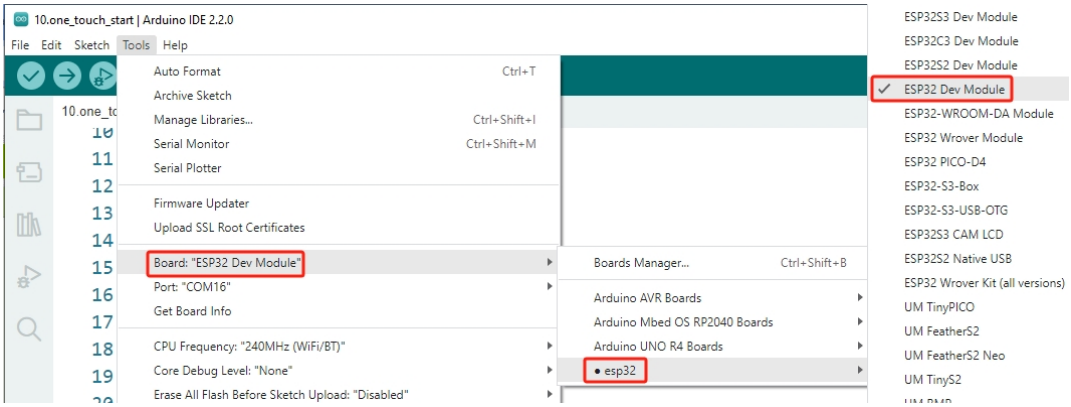
1. Sie können die Datei `2.1_hello_led.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\2.1_hello_led` öffnen. Oder kopieren Sie diesen Code direkt in die Arduino IDE.

2. Verbinden Sie dann das ESP32 WROOM 32E mit Ihrem Computer über ein Micro-USB-Kabel.

    - *„Unbekanntes COMxx“ wird immer angezeigt?*



3. Wählen Sie das Board (ESP32 Dev Module) und den entsprechenden Port aus.



4. Klicken Sie jetzt auf den **Hochladen**-Knopf, um den Code auf das ESP32-Board zu laden.

5. Nachdem der Code erfolgreich hochgeladen wurde, sehen Sie das LED-Blinken.

Wie funktioniert das?

1. Deklariere eine Integer-Konstante mit dem Namen `ledPin` und weise ihr den Wert 26 zu.

```
const int ledPin = 26;  // The GPIO pin for the LED
```

2. Initialisiere den Pin in der `setup()`-Funktion, in der du den Pin im `OUTPUT`-Modus initialisieren musst.

```
void setup() {
    pinMode(ledPin, OUTPUT);
}
```

   - `void pinMode(uint8_t pin, uint8_t mode);`: Diese Funktion wird verwendet, um den GPIO-Betriebsmodus für einen bestimmten Pin festzulegen.
     - `pin` definiert die GPIO-Pinnummer.
     - `mode` legt den Betriebsmodus fest.

     Folgende Modi werden für die grundlegende Ein- und Ausgabe unterstützt:
     - `INPUT` setzt den GPIO als Eingang ohne Pull-up oder Pull-down (hochohmig).
     - `OUTPUT` setzt den GPIO als Ausgabe-/Lesemodus.
     - `INPUT_PULLDOWN` setzt den GPIO als Eingang mit internem Pull-down.
     - `INPUT_PULLUP` setzt den GPIO als Eingang mit internem Pull-up.

3. Die `loop()`-Funktion enthält die Hauptlogik des Programms und läuft kontinuierlich. Sie wechselt zwischen dem Setzen des Pins auf High und Low, wobei zwischen den Änderungen ein Intervall von einer Sekunde liegt.

```
void loop() {
    digitalWrite(ledPin, HIGH);   // turn the LED on (HIGH is the voltage␣
↪level)
    delay(1000);                       // wait for a second
    digitalWrite(ledPin, LOW);    // turn the LED off by making the voltage␣
↪LOW
    delay(1000);                       // wait for a second
}
```

   - `void digitalWrite(uint8_t pin, uint8_t val);`: Diese Funktion setzt den Zustand des ausgewählten GPIO auf `HIGH` oder `LOW`. Diese Funktion wird nur verwendet, wenn der `pinMode` als `OUTPUT` konfiguriert wurde.
     - `pin` definiert die GPIO-Pinnummer.
     - `val` setzt den Ausgangszustand auf `HIGH` oder `LOW`.

2.6 2.2 Verblassen

Im vorherigen Projekt haben wir die LED durch Ein- und Ausschalten mit digitaler Ausgabe gesteuert. In diesem Projekt werden wir einen Atemeffekt auf der LED erzeugen, indem wir Pulsweitenmodulation (PWM) verwenden. PWM ist eine Technik, die es uns ermöglicht, die Helligkeit einer LED oder die Geschwindigkeit eines Motors zu steuern, indem wir den Tastgrad eines Rechteckwellensignals variieren.

Mit PWM werden wir anstelle des einfachen Ein- oder Ausschaltens der LED die Dauer einstellen, während der die LED pro Zyklus eingeschaltet ist im Vergleich zur Dauer, während der sie ausgeschaltet ist. Durch schnelles Umschalten der LED ein und aus in variablen Intervallen können wir den Eindruck erwecken, dass die LED allmählich heller und dunkler wird und einen Atemeffekt simuliert.

Durch die Verwendung der PWM-Funktionen des ESP32 WROOM 32E können wir eine sanfte und präzise Kontrolle über die Helligkeit der LED erreichen. Dieser Atemeffekt verleiht Ihren Projekten eine dynamische und visuell ansprechende Komponente und schafft eine auffällige Anzeige oder Atmosphäre.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die folgenden Links kaufen.

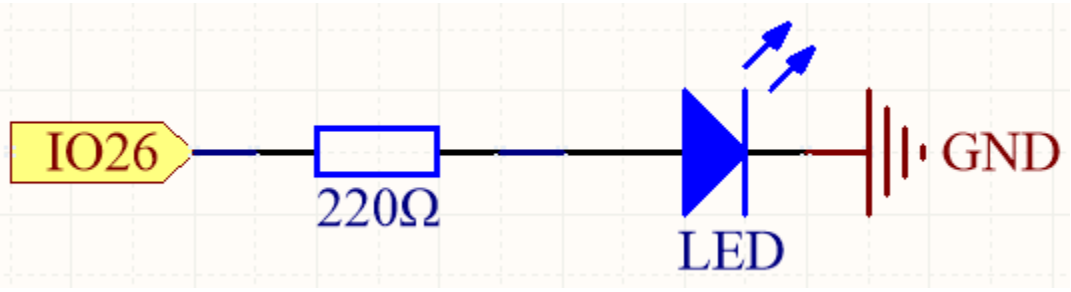
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

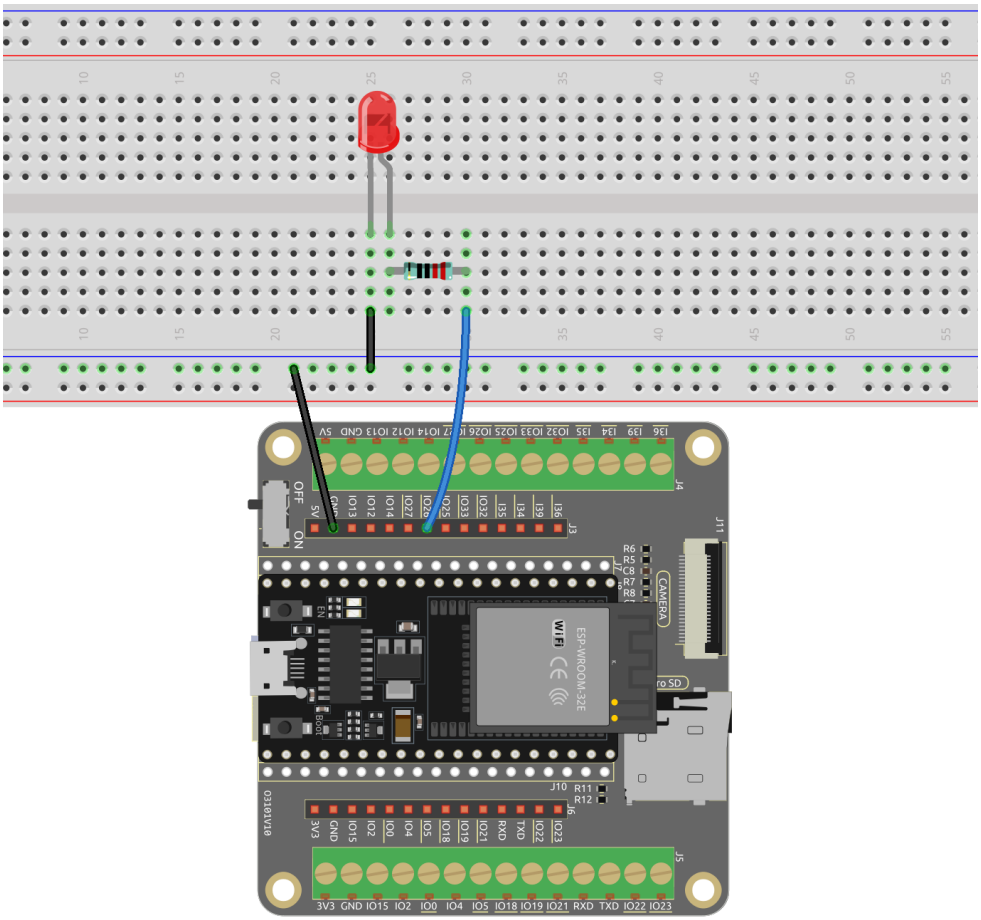
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Dieses Projekt ist der gleiche Schaltkreis wie das erste Projekt *2.1 Hallo, LED!*, aber der Signaltyp ist unterschiedlich. Das erste Projekt gibt digitale High- und Low-Pegel (0&1) direkt von Pin26 aus, um die LED zum Leuchten oder Ausschalten zu bringen. Dieses Projekt gibt ein PWM-Signal von Pin26 aus, um die Helligkeit der LED zu steuern.

Verdrahtung



Code

Bemerkung:

- Sie können die Datei `2.2_fading_led.ino` im Pfad `esp32-starter-kit-main\c\codes\2.2_fading_led` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf die **Upload**-Schaltfläche.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, können Sie die LED atmen sehen.

Wie funktioniert das?

1. Definition von Konstanten und Variablen.

```
const int ledPin = 26; // The GPIO pin for the LED
int brightness = 0;
int fadeAmount = 5;
```

- `ledPin`: Die GPIO-Pin-Nummer, an die die LED angeschlossen ist (in diesem Fall GPIO 26).
- `brightness`: Der aktuelle Helligkeitswert der LED (initial auf 0 gesetzt).

- `fadeAmount`: Die Menge, um die sich die Helligkeit der LED in jedem Schritt ändert (auf 5 gesetzt).

2. Initialisiert den PWM-Kanal und konfiguriert den LED-Pin.

```
void setup() {
    ledcSetup(0, 5000, 8); // Configure the PWM channel (0) with 5000Hz␣
↪frequency and 8-bit resolution
    ledcAttachPin(ledPin, 0); // Attach the LED pin to the PWM channel
}
```

Hier verwenden wir das (LED-Steuerung) Peripheriegerät, das primär zur Steuerung der Intensität von LEDs entwickelt wurde, aber auch zur Erzeugung von PWM-Signalen für andere Zwecke verwendet werden kann.

- `uint32_t ledcSetup(uint8_t channel, uint32_t freq, uint8_t resolution_bits);`: Diese Funktion wird verwendet, um die Frequenz und Auflösung des LEDC-Kanals einzustellen. Sie gibt die konfigurierte `frequency` für den LEDC-Kanal zurück. Wenn 0 zurückgegeben wird, ist ein Fehler aufgetreten und der LEDC-Kanal wurde nicht konfiguriert.
  - `channel`: Wählt den LEDC-Kanal zur Konfiguration aus.
  - `freq`: Wählt die PWM-Frequenz aus.
  - `resolution_bits`: Wählt die Auflösung für den LEDC-Kanal aus. Der Wertebereich liegt bei 1 bis 14 Bits (1 bis 20 Bits für ESP32).
- `void ledcAttachPin(uint8_t pin, uint8_t chan);`: Diese Funktion dient zum Anschließen des Pins an den LEDC-Kanal.
  - `pin`: Wählt den GPIO-Pin aus.
  - `chan`: Wählt den LEDC-Kanal aus.

3. Die Funktion `loop()` enthält die Hauptlogik des Programms und läuft kontinuierlich. Sie aktualisiert die Helligkeit der LED, invertiert die Helligkeitsänderung, wenn die Helligkeit den minimalen oder maximalen Wert erreicht, und führt eine Verzögerung durch.

```
void loop() {
    ledcWrite(0, brightness); // Write the new brightness value to the PWM␣
↪channel
    brightness = brightness + fadeAmount;

    if (brightness <= 0 || brightness >= 255) {
        fadeAmount = -fadeAmount;
    }

    delay(50); // Wait for 20 milliseconds
    }
```

- `void ledcWrite(uint8_t chan, uint32_t duty);`: Diese Funktion wird verwendet, um die Duty-Zyklus für den LEDC-Kanal festzulegen.
  - `chan`: Wählt den LEDC-Kanal für das Schreiben des Duty-Zyklus aus.
  - `duty`: Wählt den Duty-Zyklus für den ausgewählten Kanal aus.

2.7 2.3 Bunte Beleuchtung

In diesem Projekt tauchen wir in die faszinierende Welt der additiven Farbmischung mit Hilfe einer RGB-LED ein.

Die RGB-LED kombiniert drei Primärfarben, nämlich Rot, Grün und Blau, in einem einzigen Gehäuse. Diese drei LEDs teilen sich einen gemeinsamen Kathoden-Pin, während jeder Anoden-Pin die Intensität der entsprechenden Farbe steuert.

Durch Variation der elektrischen Signalstärke, die auf jede Anode angewendet wird, können wir eine Vielzahl von Farben erzeugen. Zum Beispiel führt die Mischung von hochintensivem rotem und grünem Licht zu gelbem Licht, während die Kombination von blauem und grünem Licht Cyan erzeugt.

Durch dieses Projekt werden wir die Prinzipien der additiven Farbmischung erforschen und unsere Kreativität entfesseln, indem wir die RGB-LED manipulieren, um faszinierende und lebendige Farben darzustellen.

Benötigte Komponenten

In diesem Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die unten stehenden Links kaufen.

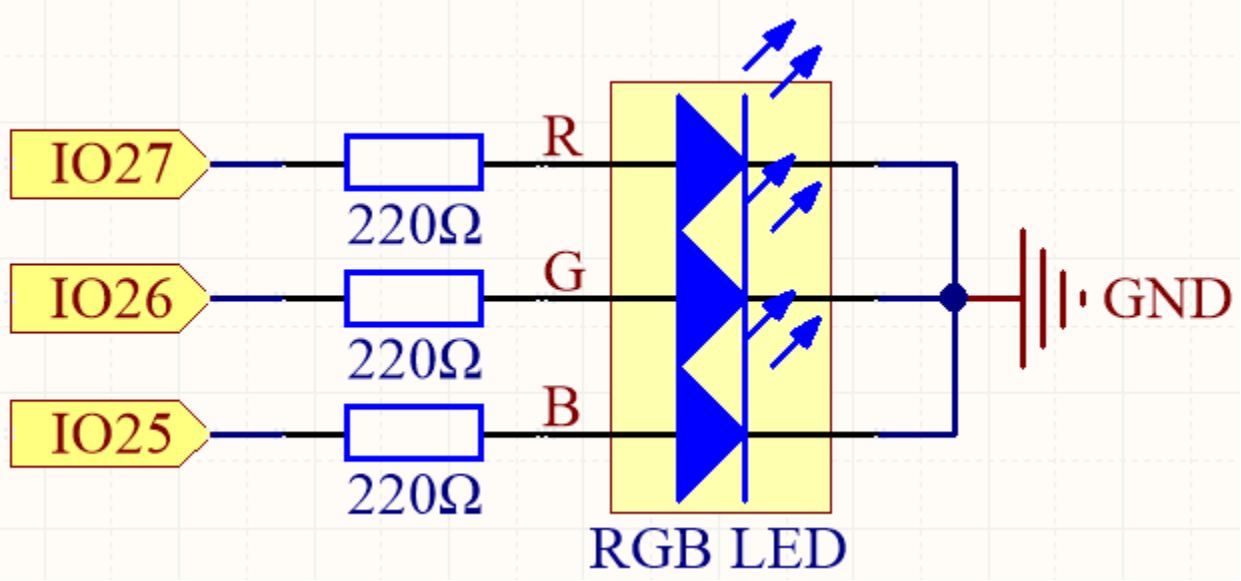
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
RGB LED	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

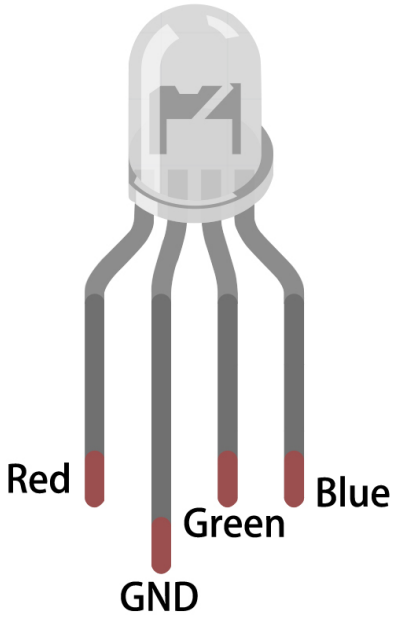
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan

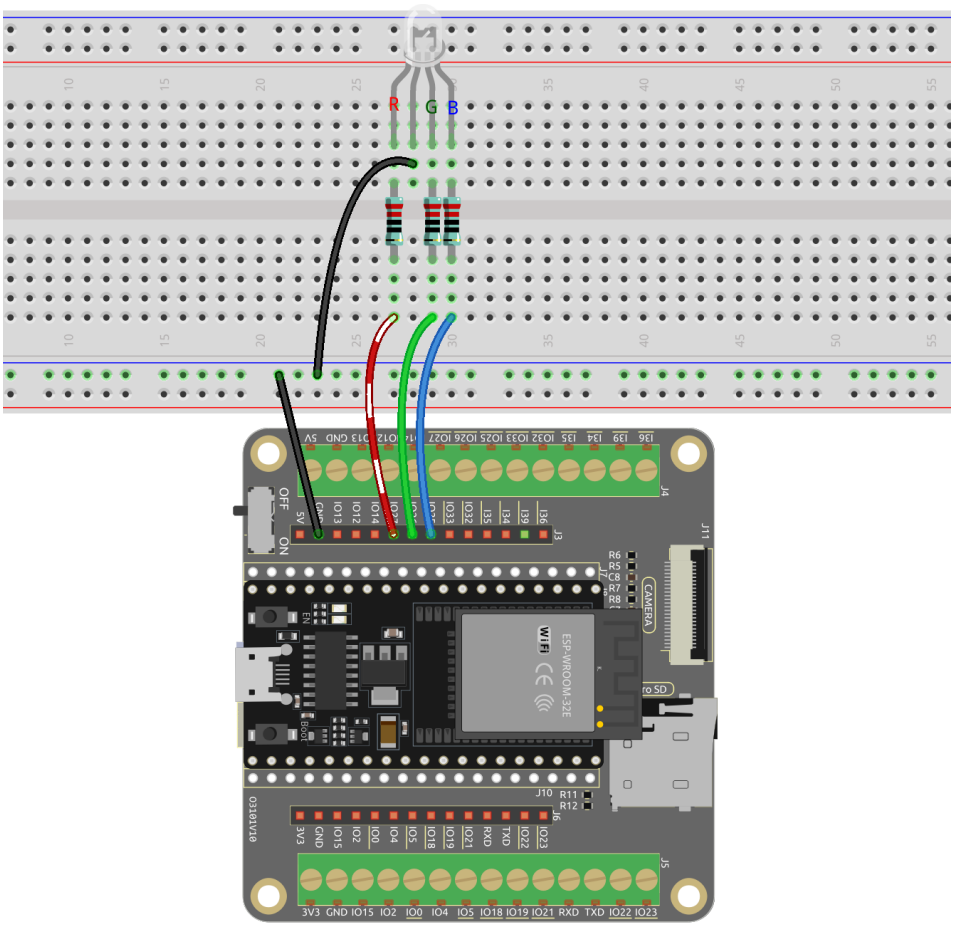


Die PWM-Pins Pin27, Pin26 und Pin25 steuern die Rot-, Grün- und Blau-Pins der RGB-LED und verbinden den gemeinsamen Kathoden-Pin mit GND. Dadurch kann die RGB-LED eine bestimmte Farbe anzeigen, indem sie Licht mit unterschiedlichen PWM-Werten auf diese Pins überlagert.

Verdrahtung



Die RGB-LED hat 4 Pins: der lange Pin ist der gemeinsame Kathoden-Pin, der normalerweise mit GND verbunden ist; der linke Pin neben dem längsten Pin ist Rot; und die beiden Pins rechts sind Grün und Blau.

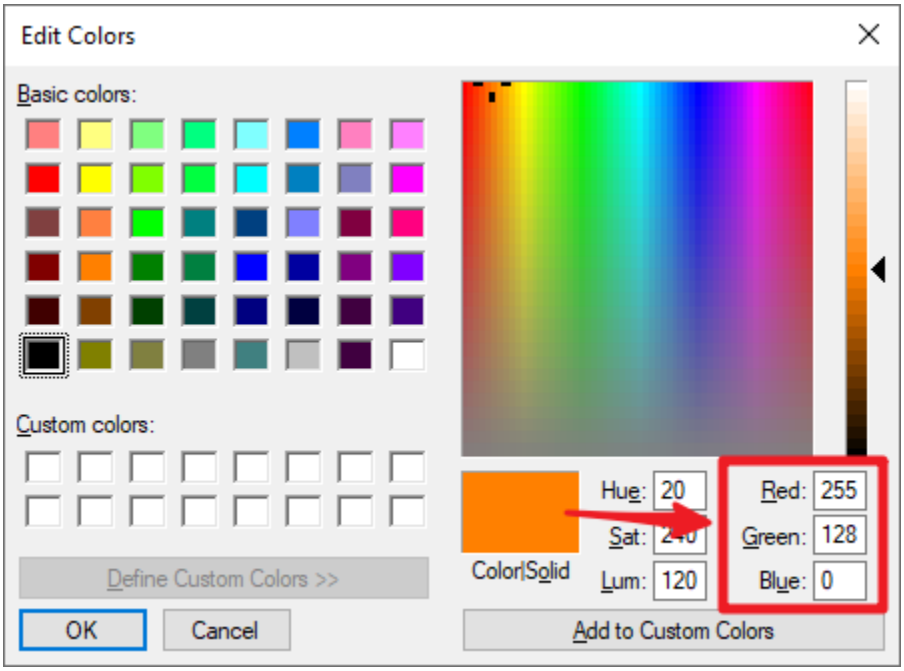


Code

Hier können wir unsere Lieblingsfarbe in einer Zeichensoftware (wie z.B. Paint) auswählen und sie mit der RGB-LED anzeigen.

Bemerkung:

- Sie können die Datei `2.3_rgb_led.ino` im Pfad `esp32-starter-kit-main\c\codes\2.3_rgb_led` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf die **Upload**-Schaltfläche.
- *„Unbekanntes COMxx“ wird immer angezeigt?*



Schreiben Sie den RGB-Wert in `color_set()` und Sie werden sehen, wie die RGB-LED die gewünschten Farben aufleuchten lässt.

Wie funktioniert das?

1. Definiere die GPIO-Pins, die PWM-Kanäle sowie die Frequenz (in Hz) und Auflösung (in Bits).

```
// Define RGB LED pins
const int redPin = 27;
const int greenPin = 26;
const int bluePin = 25;

// Define PWM channels
const int redChannel = 0;
const int greenChannel = 1;
const int blueChannel = 2;

// Define PWM frequency and resolution
const int freq = 5000;
const int resolution = 8;
```

2. Die Funktion `setup()` initialisiert die PWM-Kanäle mit der angegebenen Frequenz und Auflösung und weist dann den LED-Pins ihre entsprechenden PWM-Kanäle zu.

```
void setup() {
    // Set up PWM channels
    ledcSetup(redChannel, freq, resolution);
    ledcSetup(greenChannel, freq, resolution);
    ledcSetup(blueChannel, freq, resolution);

    // Attach pins to corresponding PWM channels
    ledcAttachPin(redPin, redChannel);
    ledcAttachPin(greenPin, greenChannel);
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    ledcAttachPin(bluePin, blueChannel);
}
```

Hier verwenden wir die (LED-Steuerung), die hauptsächlich zur Steuerung der Intensität von LEDs entwickelt wurde, aber auch verwendet werden kann, um PWM-Signale für andere Zwecke zu erzeugen.

- `uint32_t ledcSetup(uint8_t channel, uint32_t freq, uint8_t resolution_bits);`: Diese Funktion wird verwendet, um die Frequenz und Auflösung des LEDC-Kanals einzustellen. Sie gibt die konfigurierte `Frequenz` für den LEDC-Kanal zurück. Wenn 0 zurückgegeben wird, ist ein Fehler aufgetreten und der LEDC-Kanal wurde nicht konfiguriert.
  - `channel` wählt den LEDC-Kanal zur Konfiguration aus.
  - `freq` wählt die PWM-Frequenz aus.
  - `resolution_bits` wählt die Auflösung für den LEDC-Kanal aus. Der Bereich beträgt 1-14 Bits (1-20 Bits für ESP32).
- `void ledcAttachPin(uint8_t pin, uint8_t chan);`: Diese Funktion wird verwendet, um den Pin dem LEDC-Kanal zuzuordnen.
  - `pin` wählt den GPIO-Pin aus.
  - `chan` wählt den LEDC-Kanal aus.

3. Die Funktion `loop()` wechselt mit einer Sekunde Verzögerung zwischen verschiedenen Farben (Rot, Grün, Blau, Gelb, Lila und Cyan).

```
void loop() {
    setColor(255, 0, 0); // Red
    delay(1000);
    setColor(0, 255, 0); // Green
    delay(1000);
    setColor(0, 0, 255); // Blue
    delay(1000);
    setColor(255, 255, 0); // Yellow
    delay(1000);
    setColor(80, 0, 80); // Purple
    delay(1000);
    setColor(0, 255, 255); // Cyan
    delay(1000);
}
```

4. Die Funktion `setColor()` setzt die gewünschte Farbe, indem sie die entsprechenden Tastverhältniswerte für jeden PWM-Kanal schreibt. Die Funktion erhält drei Ganzzahlargumente für die Rot-, Grün- und Blau-Farbintensität.

```
void setColor(int red, int green, int blue) {
     // For common-anode RGB LEDs, use 255 minus the color value
    ledcWrite(redChannel, red);
    ledcWrite(greenChannel, green);
    ledcWrite(blueChannel, blue);
}
```

- `void ledcWrite(uint8_t chan, uint32_t duty);`: Diese Funktion wird verwendet, um das Tastverhältnis für den LEDC-Kanal festzulegen.
  - `chan` wählt den LEDC-Kanal zum Schreiben des Tastverhältnisses aus.
  - `duty` wählt das Tastverhältnis, das für den ausgewählten Kanal festgelegt werden soll.

2.8 2.4 Microchip - 74HC595

Willkommen bei diesem aufregenden Projekt! In diesem Projekt verwenden wir den 74HC595-Chip, um eine fließende Anzeige von 8 LEDs zu steuern.

Stellen Sie sich vor, Sie starten dieses Projekt und beobachten einen faszinierenden Lichtfluss, als ob ein funkelnder Regenbogen zwischen den 8 LEDs springt. Jede LED leuchtet nacheinander auf und verblasst schnell, während die nächste LED weiterhin leuchtet und einen wunderschönen und dynamischen Effekt erzeugt.

Durch geschickte Nutzung des 74HC595-Chips können wir die Ein- und Ausschaltzustände mehrerer LEDs steuern, um den fließenden Effekt zu erzielen. Dieser Chip verfügt über mehrere Ausgangspins, die in Serie geschaltet werden können, um die Reihenfolge der LED-Beleuchtung zu steuern. Darüber hinaus können wir dank der Erweiterbarkeit des Chips problemlos weitere LEDs zur fließenden Anzeige hinzufügen und noch spektakulärere Effekte erzielen.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die unten stehenden Links kaufen.

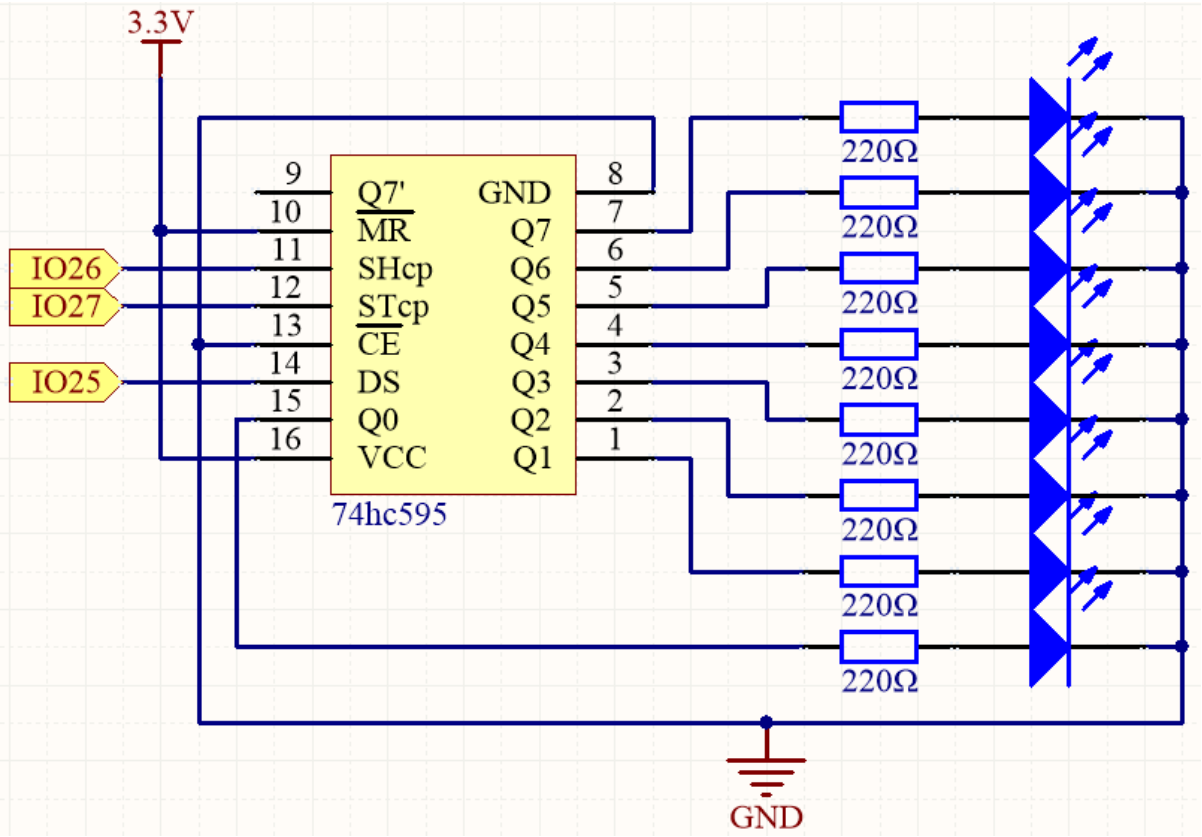
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
74HC595	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

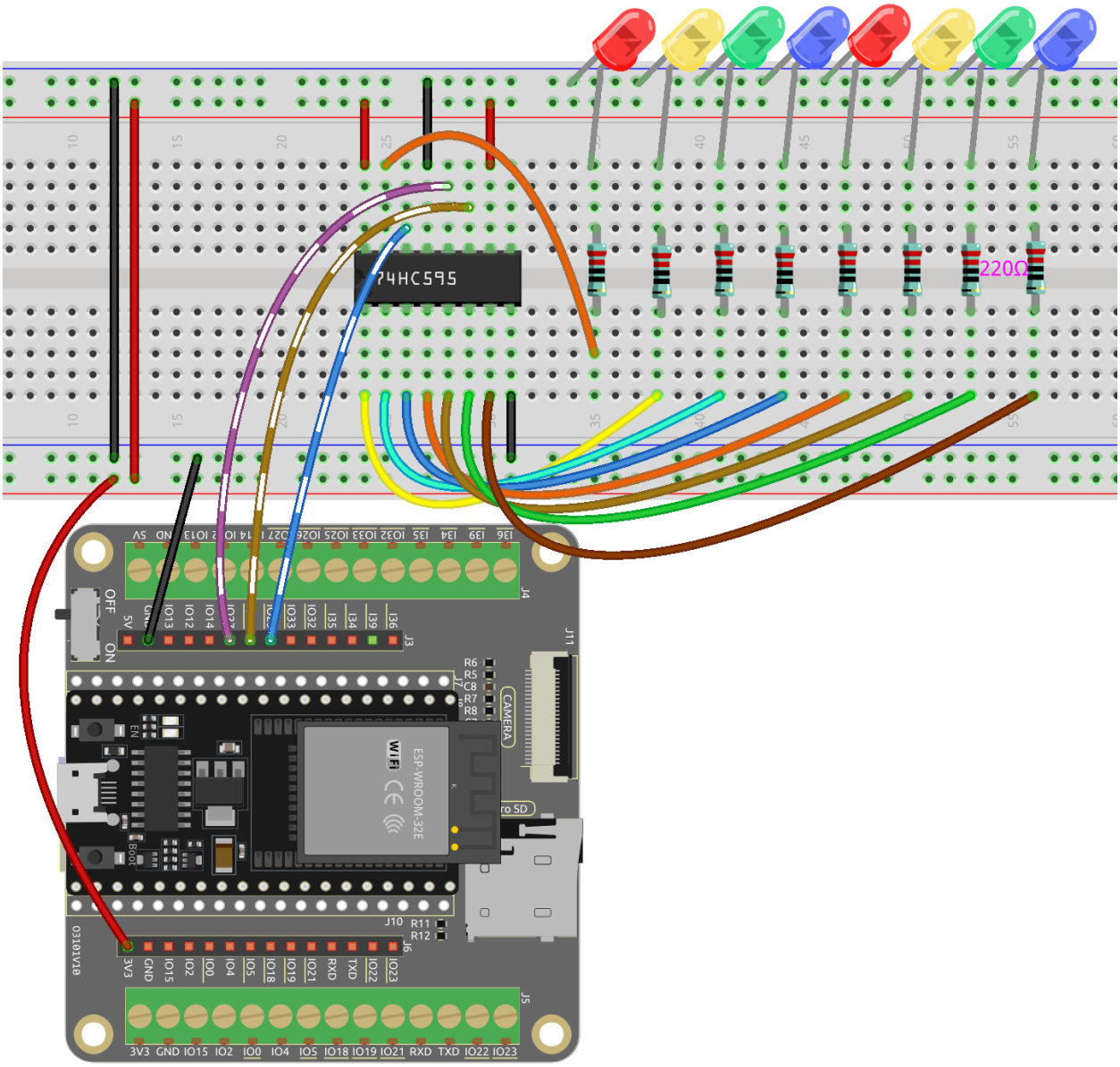
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



- Wenn MR (Pin10) auf hohem Pegel ist und CE (Pin13) auf niedrigem Pegel ist, wird die Daten beim Anstieg der SHcp eingegeben und geht durch den Anstieg der SHcp in das Speicherregister.
- Wenn die beiden Takte miteinander verbunden sind, ist der Schieberegister immer einen Takt früher als das Speicherregister.
- Im Speicherregister befinden sich ein serieller Schiebeeingangspin (DS), ein serieller Ausgangspin (Q7‘) und ein asynchroner Rücksetzknopf (niedriger Pegel).
- Das Speicherregister gibt einen Bus mit parallel 8 Bit und in drei Zuständen aus.
- Wenn OE aktiviert ist (niedriger Pegel), werden die Daten im Speicherregister auf den Bus (Q0 ~ Q7) ausgegeben.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `2.4_74hc595.ino` im Pfad `esp32-starter-kit-main\c\codes\2.4_74hc595`.
- Wählen Sie das Board (ESP32 Dev Module) und den entsprechenden Port aus, klicken Sie dann auf die Schaltfläche **Upload**.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem Sie den Code auf das ESP32-Board hochgeladen haben, können Sie sehen, wie die LEDs nacheinander eingeschaltet werden.

Wie funktioniert das?

1. Deklarieren Sie ein Array, um mehrere 8-Bit-Binärzahlen zu speichern, die verwendet werden, um den Arbeitszustand der acht LEDs zu ändern, die von 74HC595 gesteuert werden.

```
int datArray[] = {B00000000, B00000001, B00000011, B00000111, B00001111,␣
↪B00011111, B00111111, B01111111, B11111111};
```

2. `loop()` Funktion.

```
void loop()
    {
        for(int num = 0; num <10; num++)
        {
            digitalWrite(STcp,LOW); //Set ST_CP and hold low for as long as␣
↪you are transmitting
            shiftOut(DS,SHcp,MSBFIRST,datArray[num]);
            digitalWrite(STcp,HIGH); //pull the ST_CPST_CP to save the data
            delay(1000);
        }
    }
```

- Durchläuft das Array `datArray[]` und sendet nacheinander die binären Werte an das Schieberegister.
- Die Befehle `digitalWrite(STcp, LOW)` und `digitalWrite(STcp, HIGH)` speichern die Daten im Speicherregister.
- Die Funktion `shiftOut()` sendet die binären Werte aus `datArray[]` an das Schieberegister unter Verwendung des Datenpins (DS) und des Schieberegister-Takt-Pins (SHcp). `MSBFIRST` bedeutet, dass die Übertragung von den höchsten Bits erfolgt.
- Erzeugt dann eine 1-sekündige Pause zwischen jedem Aktualisieren des LED-Musters.

2.9 2.5 7-Segment-Anzeige

Willkommen bei diesem faszinierenden Projekt! In diesem Projekt werden wir die bezaubernde Welt der Anzeige von Zahlen von 0 bis 9 auf einer Siebensegmentanzeige erkunden.

Stellen Sie sich vor, dieses Projekt zu starten und zu beobachten, wie ein kleines, kompaktes Display hell mit jeder Zahl von 0 bis 9 leuchtet. Es ist, als hätte man einen Miniaturbildschirm, der die Ziffern auf fesselnde Weise präsentiert. Durch die Steuerung der Signalpins können Sie mühelos die angezeigte Zahl ändern und verschiedene ansprechende Effekte erzeugen.

Durch einfache Schaltungsverbindungen und Programmierung lernen Sie, wie Sie mit der Siebensegmentanzeige interagieren und Ihre gewünschten Zahlen zum Leben erwecken können. Ob als Zähler, Uhr oder jede andere faszinierende Anwendung, die Siebensegmentanzeige wird Ihr zuverlässiger Begleiter sein und Ihren Projekten einen Hauch von Brillanz verleihen.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

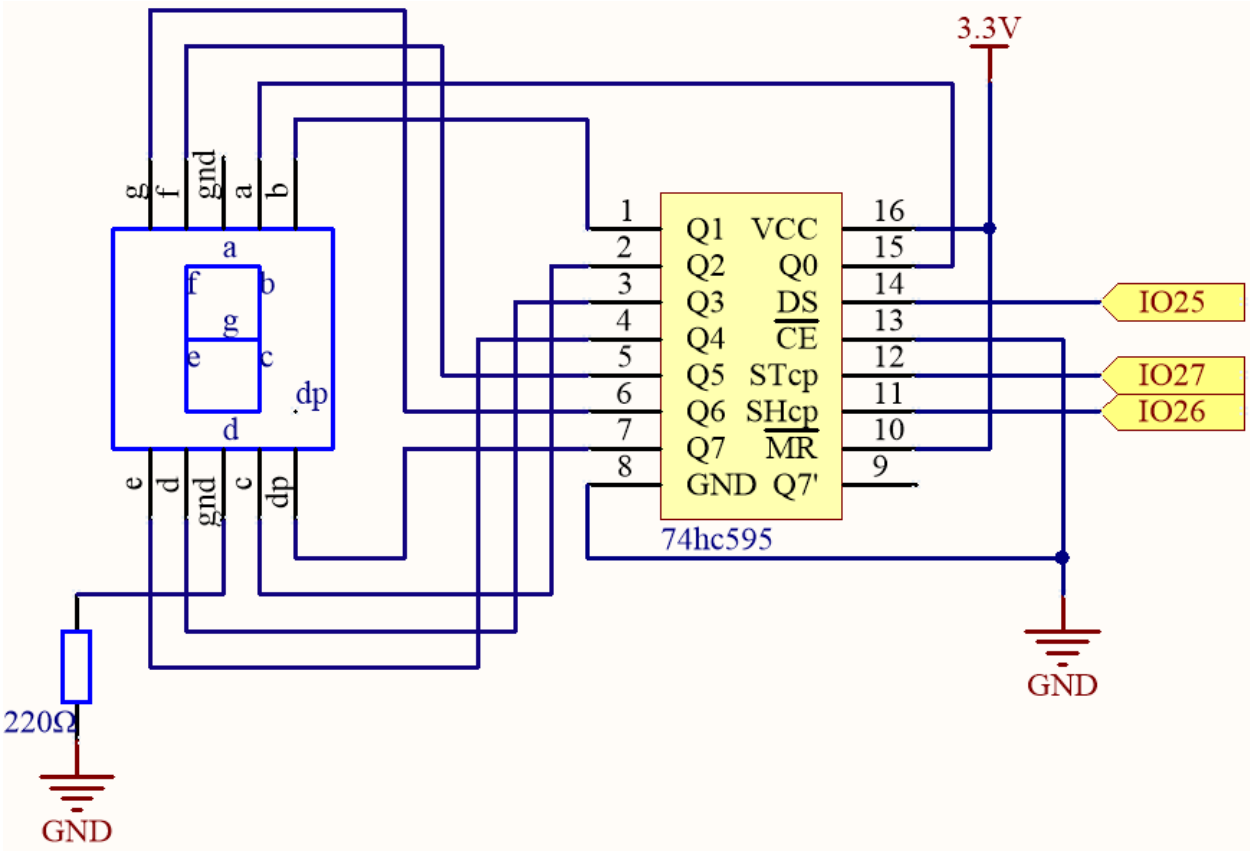
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
7-Segment-Anzeige	
74HC595	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan

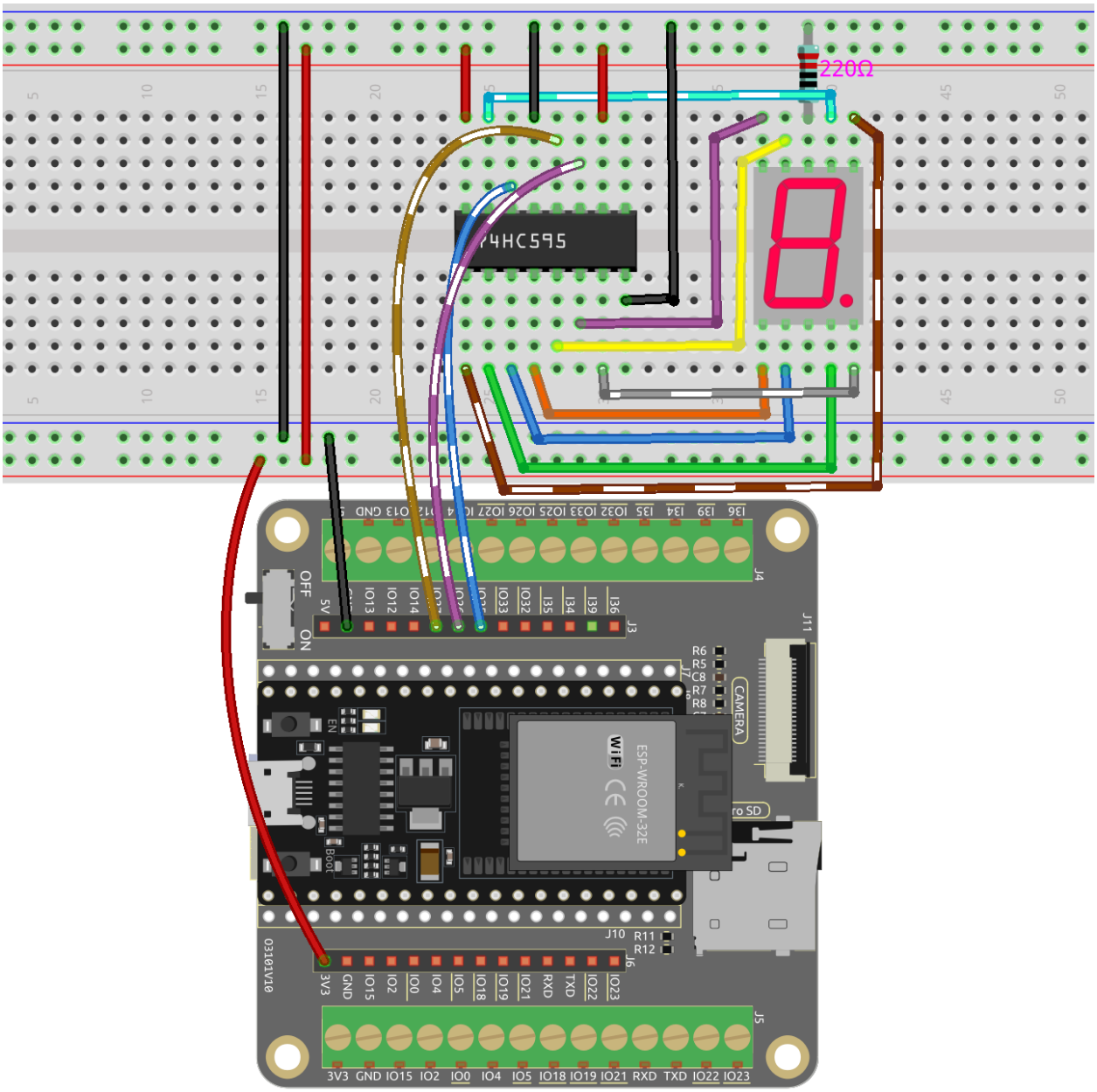


Hier ist das Verkabelungsprinzip im Wesentlichen das gleiche wie bei *2.4 Microchip - 74HC595*, der einzige Unterschied ist, dass Q0-Q7 mit den a ~ g Pins der 7-Segment-Anzeige verbunden sind.

Tab. 1: Verkabelung

74HC595	LED-Segmentanzeige
Q0	a
Q1	b
Q2	c
Q3	d
Q4	e
Q5	f
Q6	g
Q7	dp

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `2.5_7segment.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\2.5_7segment`.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, können Sie sehen, wie das LED-Segmentdisplay die Zahlen 0~9 nacheinander anzeigt.

Wie funktioniert das?

In diesem Projekt verwenden wir die Funktion `shiftOut()`, um die Binärzahl in das Schieberegister zu schreiben.

Angenommen, das 7-Segment-Display zeigt die Zahl „2“ an. Dieses Bitmuster entspricht den Segmenten **f**, **c** und **dp**, die ausgeschaltet (niedrig) sind, während die Segmente **a**, **b**, **d**, **e** und **g** eingeschaltet (hoch) sind. Das ist „01011011“ in Binär und „0x5b“ in hexadezimaler Schreibweise.

Daher müssten Sie `shiftOut(DS,SHcp,MSBFIRST,0x5b)` aufrufen, um die Zahl „2“ auf dem 7-Segment-Display anzuzeigen.

- Hexadezimal
- Binär-Hex-Konverter

Die folgende Tabelle zeigt die hexadezimalen Muster, die in das Schieberegister geschrieben werden müssen, um die Zahlen 0 bis 9 auf einem 7-Segment-Display anzuzeigen.

Tab. 2: Glyphen-Code

Zahlen	Binärcode	Hex-Code
0	00111111	0x3f
1	00000110	0x06
2	01011011	0x5b
3	01001111	0x4f
4	01100110	0x66
5	01101101	0x6d
6	01111101	0x7d
7	00000111	0x07
8	01111111	0x7f
9	01101111	0x6f

Schreiben Sie diese Codes in `shiftOut()`, um das LED-Segmentdisplay die entsprechenden Zahlen anzeigen zu lassen.

2.10 2.6 Zeichenanzeige

Nun tauchen wir ein in die faszinierende Welt der Zeichenanzeige mit dem I2C LCD1602-Modul.

In diesem Projekt lernen wir, wie man das LCD-Modul initialisiert, die gewünschten Anzeigeparameter festlegt und Zeichendaten zur Anzeige auf dem Bildschirm sendet. Wir können individuelle Nachrichten darstellen, Sensormessungen anzeigen oder interaktive Menüs erstellen. Die Möglichkeiten sind grenzenlos!

Durch das Beherrschen der Kunst der Zeichenanzeige auf dem I2C LCD1602 erschließen wir neue Wege für Kommunikation und Informationsdarstellung in unseren Projekten. Lassen Sie uns in diese aufregende Reise eintauchen und unsere Zeichen auf dem LCD-Bildschirm zum Leben erwecken.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

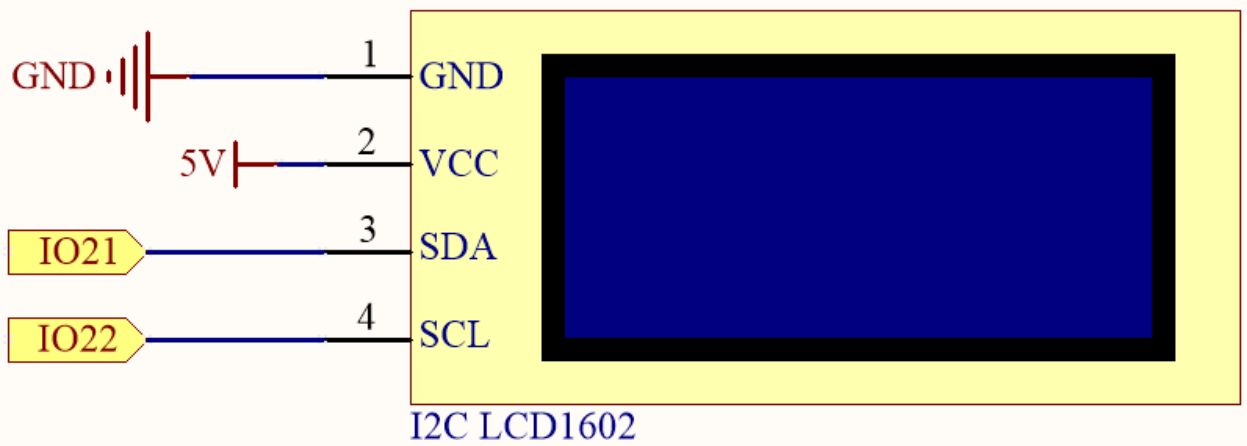
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
I2C LCD1602	

Verfügbare Pins

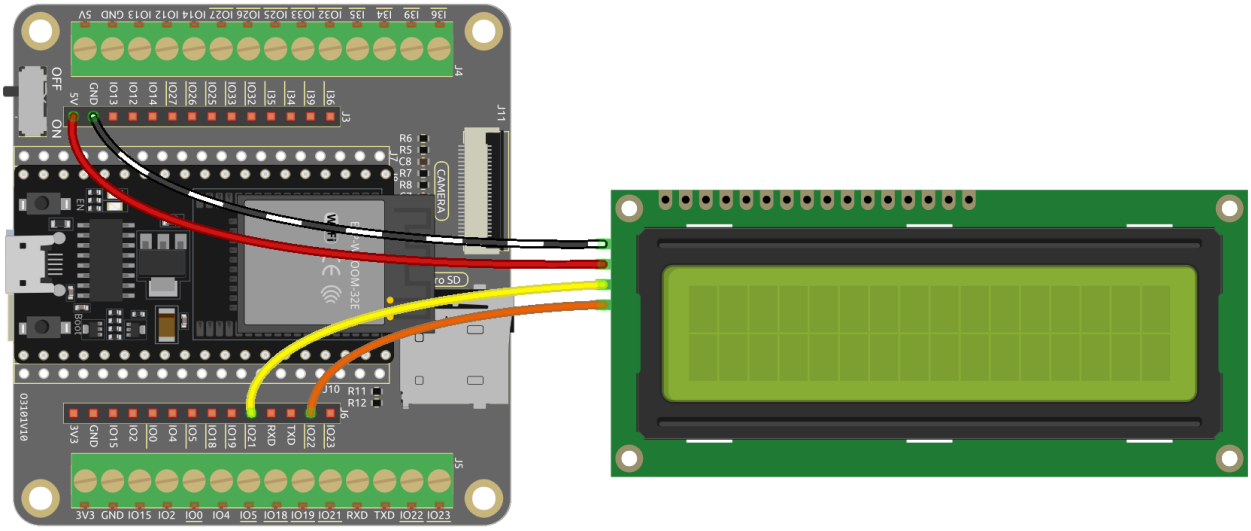
Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	Verwendungszweck
IO21	SDA
IO22	SCL

Schaltplan



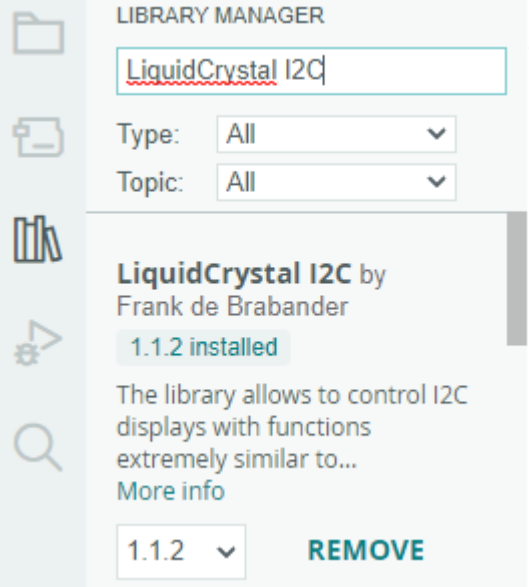
Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `2.6_lcd1602.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\2.6_lcd1602`.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*
- Hier wird die Bibliothek `LiquidCrystal I2C` verwendet, die Sie über den **Library Manager** installieren können.



Nach dem Hochladen dieses Programms wird das I2C LCD1602 für 3 Sekunden die Begrüßungsnachricht „Hallo, Sunfounder!“ anzeigen. Danach zeigt der Bildschirm ein „ZÄHLER:“-Label und den Zählwert an, der sich jede Sekunde erhöht.

Bemerkung: Wenn der Code und die Verkabelung korrekt sind, das LCD aber immer noch keinen Inhalt anzeigt, können Sie das Potentiometer auf der Rückseite justieren, um den Kontrast zu erhöhen.

Wie funktioniert das?

Durch den Aufruf der Bibliothek `LiquidCrystal_I2C.h` können Sie das LCD leicht steuern.

```
#include <LiquidCrystal_I2C.h>
```

Bibliotheksfunktionen:

- Erstellt eine neue Instanz der Klasse `LiquidCrystal_I2C`, die ein bestimmtes LCD darstellt, das an Ihr Arduino-Board angeschlossen ist.

  ```
  LiquidCrystal_I2C(uint8_t lcd_Addr,uint8_t lcd_cols,uint8_t lcd_rows)
  ```

  - `lcd_Addr`: Die Adresse des LCDs, standardmäßig auf 0x27 eingestellt.
  - `lcd_cols`: Das LCD1602 hat 16 Spalten.
  - `lcd_rows`: Das LCD1602 hat 2 Reihen.

- Initialisiert das LCD.

  ```
  void init()
  ```

- Schaltet die (optional) Hintergrundbeleuchtung ein.

  ```
  void backlight()
  ```

- Schaltet die (optional) Hintergrundbeleuchtung aus.

```
void nobacklight()
```

- Schaltet das LCD-Display ein.

```
void display()
```

- Schaltet das LCD-Display schnell aus.

```
void nodisplay()
```

- Löscht das Display und setzt die Cursorposition auf Null.

```
void clear()
```

- Setzt die Cursorposition auf col,row.

```
void setCursor(uint8_t col,uint8_t row)
```

- Druckt Text auf dem LCD.

```
void print(data,BASE)
```

  - `data`: Die zu druckenden Daten (char, byte, int, long oder string).
  - `BASE (optional)`: Die Basis, in der Zahlen gedruckt werden sollen.
    - `BIN` für Binär (Basis 2)
    - `DEC` für Dezimal (Basis 10)
    - `OCT` für Oktal (Basis 8)
    - `HEX` für Hexadezimal (Basis 16).

2.11 2.7 RGB-LED-Streifen

In diesem Projekt tauchen wir ein in die faszinierende Welt der Steuerung von WS2812-LED-Streifen und erwecken eine lebendige Farbenvielfalt zum Leben. Mit der Fähigkeit, jede LED auf dem Streifen einzeln zu steuern, können wir fesselnde Lichteffekte kreieren, die die Sinne bezaubern.

Darüber hinaus haben wir eine spannende Erweiterung zu diesem Projekt hinzugefügt, in der wir das Reich der Zufälligkeit erkunden. Durch die Einführung zufälliger Farben und die Implementierung eines fließenden Lichteffekts können wir ein hypnotisierendes visuelles Erlebnis schaffen, das fasziniert und verzaubert.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

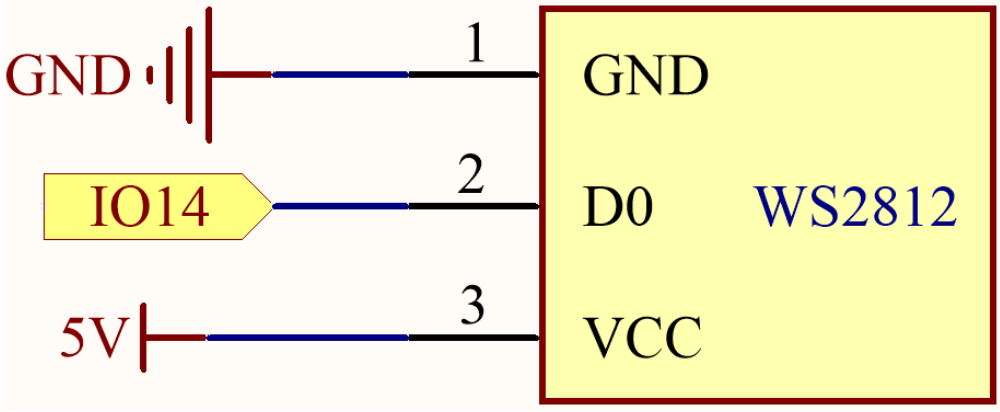
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
WS2812 RGB 8 LEDs Leiste	

Schaltplan



Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

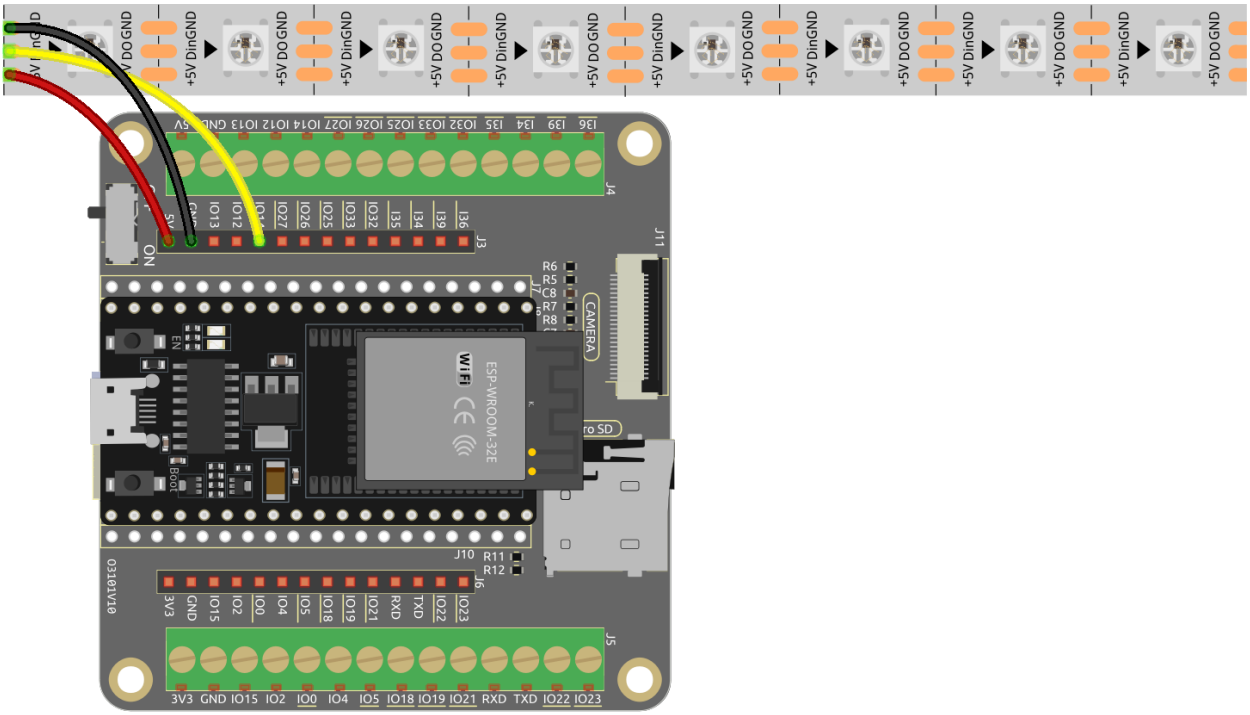
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Bemerkung: IO33 ist für dieses Projekt nicht verfügbar.

Der WS2812-LED-Streifen ist eine Art von LED-Streifen, der ein präzises Pulsweitenmodulationssignal (PWM) benötigt. Das PWM-Signal hat genaue Anforderungen in Bezug auf Zeit und Spannung. Zum Beispiel entspricht ein „0“-Bit für den WS2812 einem High-Level-Impuls von etwa 0,4 Mikrosekunden, während ein „1“-Bit einem High-Level-Impuls von etwa 0,8 Mikrosekunden entspricht. Das bedeutet, dass der Streifen hochfrequente Spannungsänderungen empfangen muss.

Jedoch wird mit einem 4,7K-Pull-Up-Widerstand und einem 100nf-Pull-Down-Kondensator an IO33 ein einfacher Tiefpassfilter erstellt. Diese Art von Schaltung „glättet“ hochfrequente Signale, da der Kondensator einige Zeit zum Aufladen und Entladen benötigt, wenn er Spannungsänderungen erhält. Wenn sich das Signal zu schnell ändert (d.h. hochfrequent ist), kann der Kondensator nicht mithalten. Dies führt dazu, dass das Ausgangssignal verschwommen und für den Streifen unkenntlich wird.

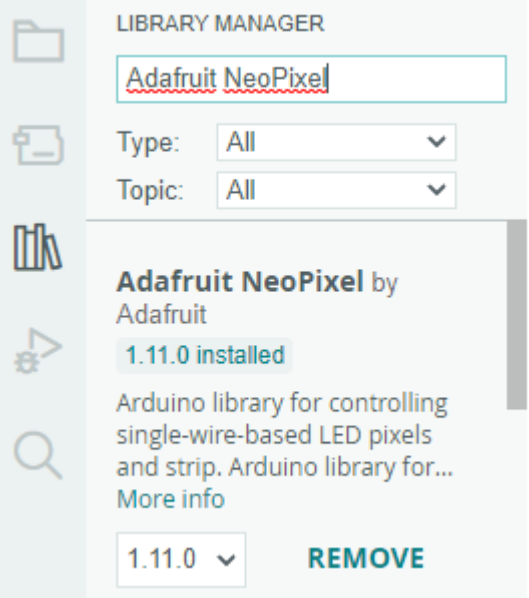
Verdrahtung



Code

Bemerkung:

- Sie können die Datei `2.7_rgb_strip.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\2.7_rgb_strip` öffnen. Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*
- Hier wird die `Adafruit NeoPixel`-Bibliothek verwendet, die Sie über den **Library Manager** installieren können.



Nachdem der Code erfolgreich hochgeladen wurde, werden die LEDs auf dem Streifen nacheinander mit einer gelben Farbe aufleuchten und dann ausgehen, wodurch ein einfacher Verfolgungseffekt entsteht.

Wie funktioniert das?

1. Die Adafruit NeoPixel-Bibliothek einbinden: Diese Zeile importiert die Adafruit NeoPixel-Bibliothek, damit das Sketch ihre Funktionen und Klassen zur Steuerung des LED-Streifens nutzen kann.

```
#include <Adafruit_NeoPixel.h> // Include the Adafruit NeoPixel library
```

2. Konstanten für den LED-Streifen definieren.

```
#define LED_PIN 13 // NeoPixel LED strip
#define NUM_LEDS 8 // Number of LEDs
```

3. Eine Instanz der Adafruit_NeoPixel-Klasse erstellen.

```
// Create an instance of the Adafruit_NeoPixel class
 Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_LEDS, LED_PIN, NEO_GRB +␣
↪NEO_KHZ800);
```

Diese Zeile erstellt eine Instanz der Klasse `Adafruit_NeoPixel` namens `strip` und konfiguriert sie mit der Anzahl der LEDs, dem mit dem LED-Streifen verbundenen Pin und den Signalparametern (GRB-Farbreihenfolge und 800 kHz Datenrate).

- `Adafruit_NeoPixel (uint16_t n, int16_t p = 6, neoPixelType t = NEO_GRB + NEO_KHZ800)`

NeoPixel-Konstruktor, wenn Länge, Pin und Pixeltyp zur Kompilierzeit bekannt sind. Gibt ein Adafruit_NeoPixel-Objekt zurück. Vor der Verwendung die Funktion `begin()` aufrufen.

- `n`: Anzahl der NeoPixels im Strang.
- `p`: Arduino-Pinnummer, die das NeoPixel-Datensignal steuert.
- `t`: Pixeltyp - Kombinieren Sie `NEO_*` Konstanten definiert in `Adafruit_NeoPixel.h`, zum Beispiel `NEO_GRB+NEO_KHZ800` für NeoPixels, die einen 800 KHz (statt 400 KHz) Datenstrom erwarten mit Farbbytes in grün, rot, blau Reihenfolge pro Pixel.

4. Den WS2812 RGB-Streifen initialisieren und die Anfangsfarbe des Streifens auf Schwarz (aus) setzen.

```
void setup() {
    strip.begin(); // Initialize the NeoPixel strip
    strip.show(); // Set initial color to black
}
```

- `void begin (void)`: Konfigurieren des NeoPixel-Pins für den Ausgang.
- `void show (void)`: Übertragen von Pixeldaten im RAM zu NeoPixels.

5. In der Funktion `loop()` werden die LEDs auf dem Streifen nacheinander mit einer gelben Farbe eingeschaltet und dann ausgeschaltet, um einen einfachen Verfolgungseffekt zu erzeugen.

```
void loop() {
    // Turn on LEDs one by one
    for (int i = 0; i < NUM_LEDS; i++) {
        strip.setPixelColor(i, 100, 45, 0); // Set the color of the i-th␣
↪LED to red
        strip.show(); // Update the LED strip with the new colors
        delay(100); // Wait for 100 milliseconds
    }

    // Turn off LEDs one by one
    for (int i = 0; i < NUM_LEDS; i++) {
        strip.setPixelColor(i, 0, 0, 0); // Set the color of the i-th LED␣
↪to black (turn it off)
        strip.show(); // Update the LED strip with the new colors
        delay(100); // Wait for 100 milliseconds
    }
}
```

- `void setPixelColor (uint16_t n, uint8_t r, uint8_t g, uint8_t b)`

Setzt die Farbe eines Pixels mit separaten Rot-, Grün- und Blaukomponenten. Bei Verwendung von RGBW-Pixeln wird Weiß auf 0 gesetzt.

- `n`: Pixelindex, beginnend bei 0.
- `r`: Rot-Helligkeit, 0 = minimal (aus), 255 = maximal.
- `g`: Grün-Helligkeit, 0 = minimal (aus), 255 = maximal.
- `b`: Blau-Helligkeit, 0 = minimal (aus), 255 = maximal.

3. Töne

2.12 3.1 Piepser

Dies ist ein einfaches Projekt, um einen aktiven Summer jede Sekunde schnell viermal piepen zu lassen.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

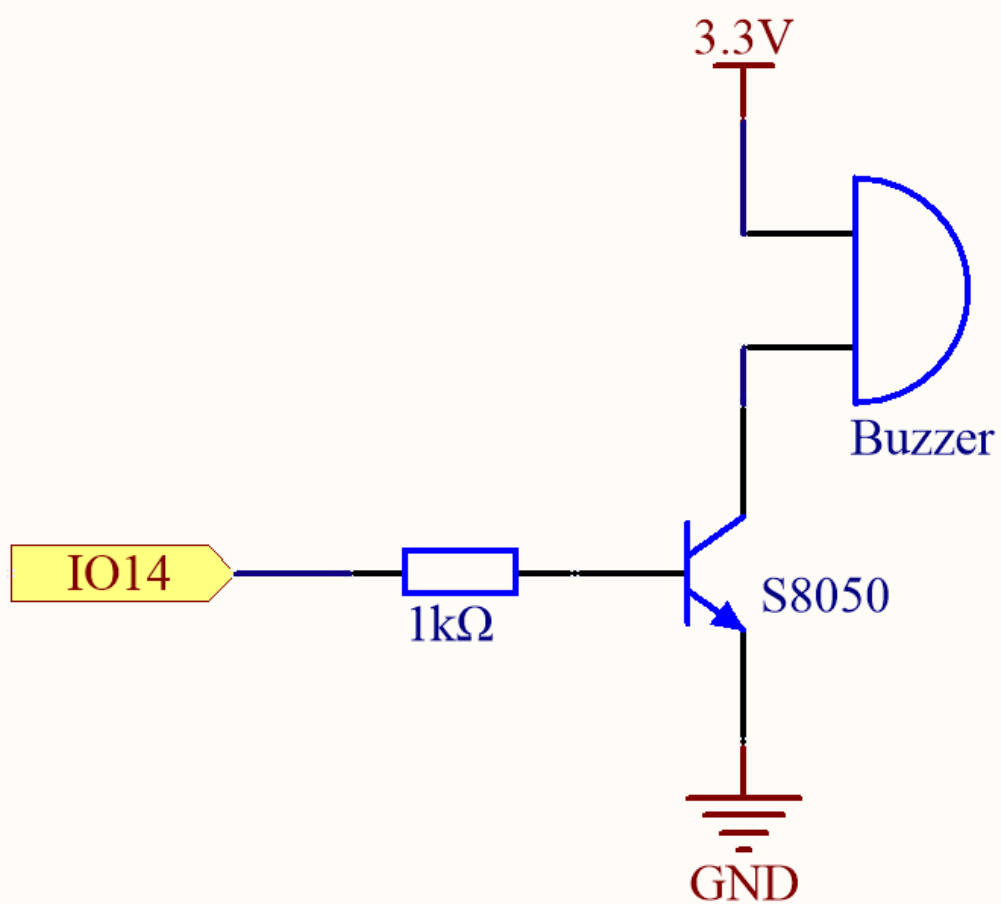
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Summer	-
Transistor	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



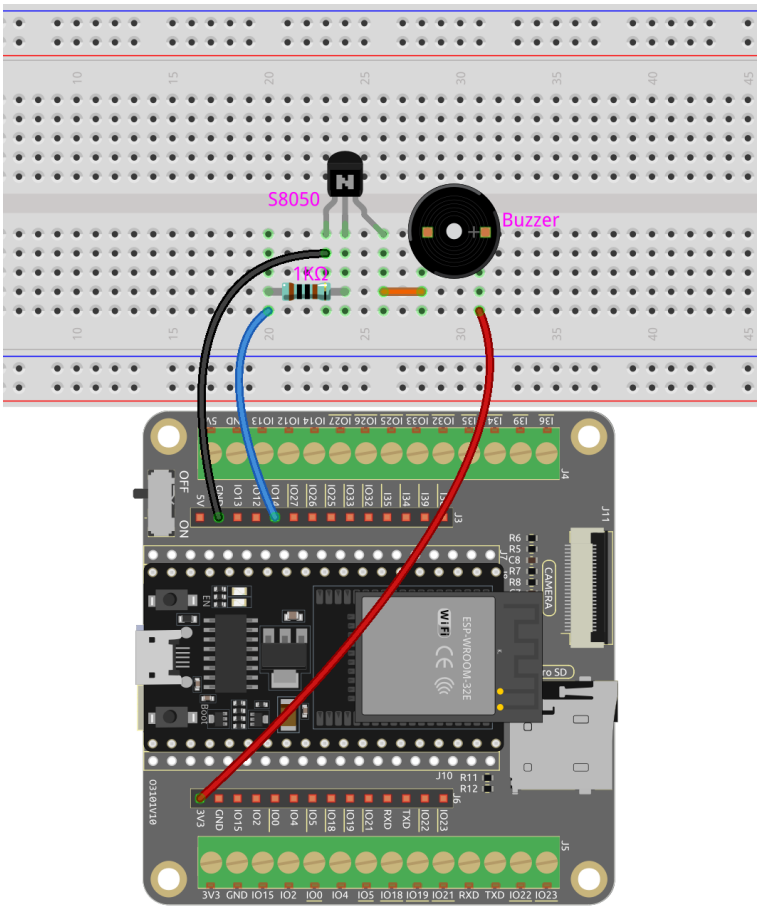
Wenn der IO14-Ausgang hoch ist, wird nach dem 1K-Strombegrenzungswiderstand (zum Schutz des Transistors) der S8050 (NPN-Transistor) leiten, so dass der Summer ertönt.

Die Rolle des S8050 (NPN-Transistor) besteht darin, den Strom zu verstärken und den Summer lauter klingen zu lassen. Tatsächlich können Sie den Summer auch direkt an IO14 anschließen, aber Sie werden feststellen, dass der Summer leiser klingt.

Verdrahtung

Im Kit sind zwei Arten von Summern enthalten. Wir müssen den aktiven Summer verwenden. Drehen Sie sie um, die versiegelte Rückseite (nicht die freiliegende PCB) ist die, die wir wollen.

Der Summer benötigt beim Arbeiten einen Transistor, hier verwenden wir S8050 (NPN-Transistor).



Code

Bemerkung:

- Sie können die Datei `3.1_beep.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\3.1_beep` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, hören Sie jede Sekunde einen Piepton.

2.13 3.2 Eigene Töne

Wir haben im vorherigen Projekt einen aktiven Summer verwendet, dieses Mal werden wir einen passiven Summer benutzen.

Wie der aktive Summer nutzt auch der passive Summer das Phänomen der elektromagnetischen Induktion, um zu funktionieren. Der Unterschied besteht darin, dass ein passiver Summer keine eigene Oszillationsquelle hat, daher wird er bei Verwendung von Gleichstromsignalen nicht piepen. Aber das ermöglicht es dem passiven Summer, seine eigene Oszillationsfrequenz anzupassen und verschiedene Noten wie „doh, re, mi, fa, sol, la, ti“ zu erzeugen.

Lassen Sie den passiven Summer eine Melodie spielen!

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

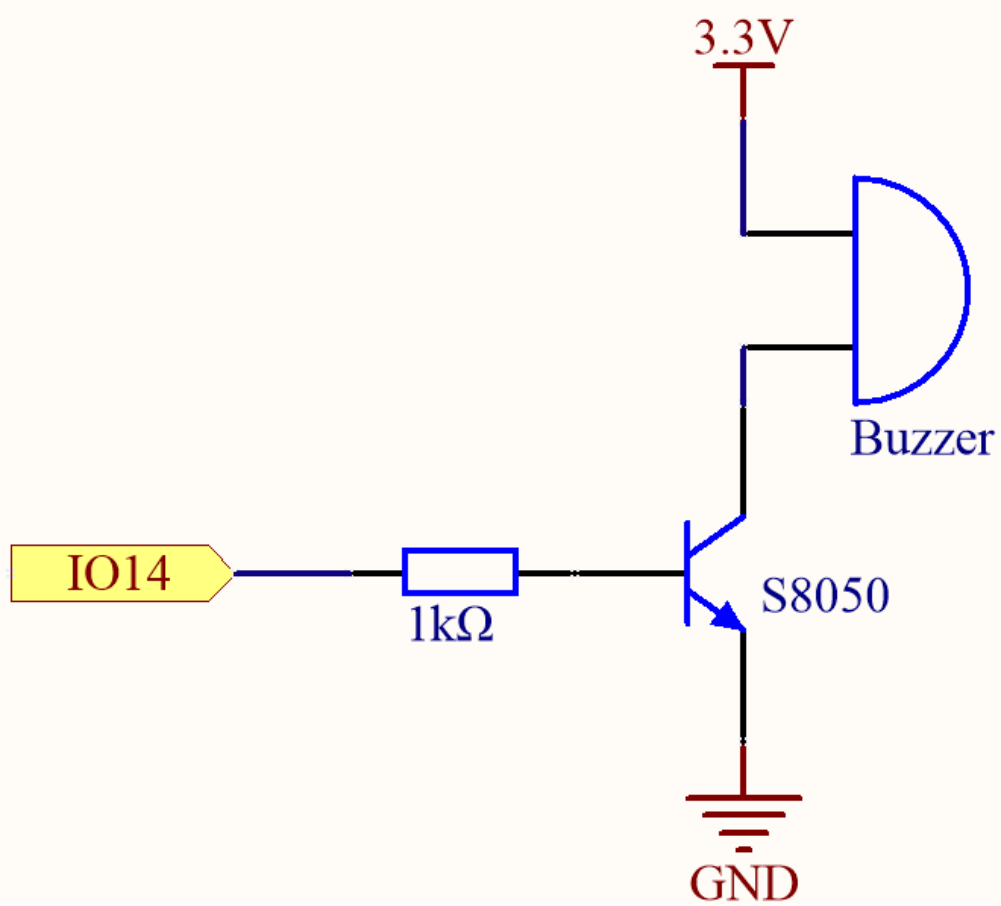
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Summer	-
Transistor	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



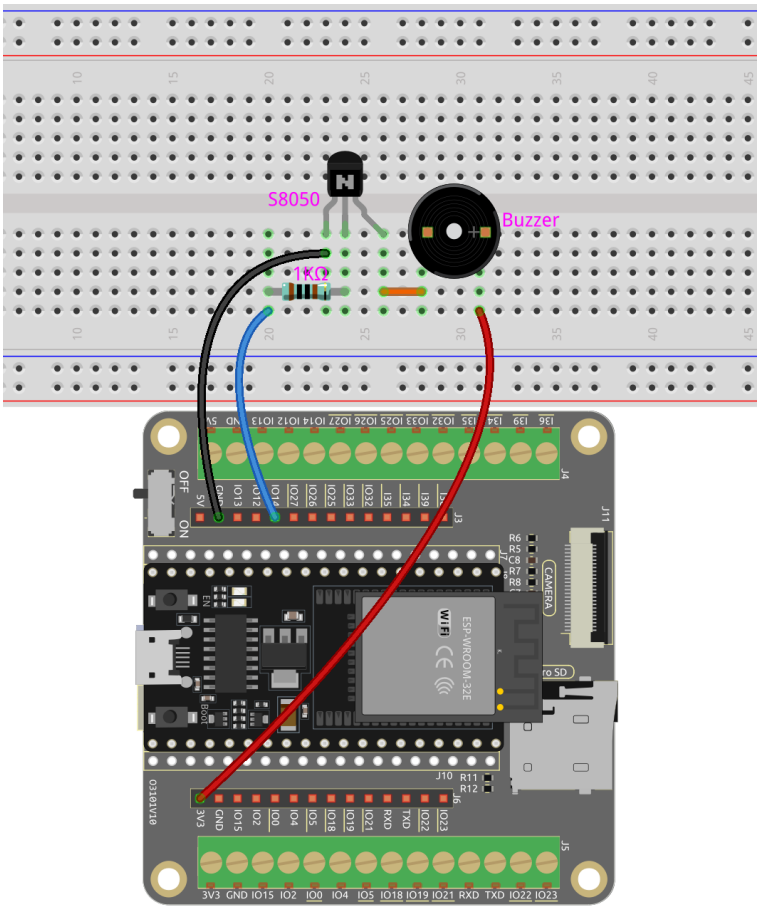
Wenn der IO14-Ausgang hoch ist, wird nach dem 1K-Strombegrenzungswiderstand (zum Schutz des Transistors) der S8050 (NPN-Transistor) leiten, so dass der Summer ertönt.

Die Rolle des S8050 (NPN-Transistor) besteht darin, den Strom zu verstärken und den Summer lauter klingen zu lassen. Tatsächlich können Sie den Summer auch direkt an IO14 anschließen, aber Sie werden feststellen, dass der Summer leiser klingt.

Verdrahtung

Im Kit sind zwei Arten von Summern enthalten. Wir müssen den passiven Summer verwenden. Drehen Sie sie um, die freiliegende PCB ist die, die wir wollen.

Der Summer benötigt beim Arbeiten einen Transistor, hier verwenden wir S8050 (NPN-Transistor).



Code

Bemerkung:

- Öffnen Sie die Datei `3.2_custom_tone.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\3.2_custom_tone`.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, hören Sie, wie der passive Summer eine Folge von 7 Musiknoten abspielt.

Wie funktioniert das?

1. Definieren Sie Konstanten für den Summer-Pin und die PWM-Auflösung.

```
const int buzzerPin = 14; //buzzer pin
const int resolution = 8;
```

2. Definieren Sie ein Array mit den Frequenzen der 7 Musiknoten in Hz.

```
int frequencies[] = {262, 294, 330, 349, 392, 440, 494};
```

3. Erstellen Sie eine Funktion, um eine gegebene Frequenz für eine bestimmte Dauer am Summer abzuspielen.

```
void playFrequency(int frequency, int duration) {
    ledcWriteTone(0, frequency); // Start the tone
    delay(duration); // Wait for the specified duration
    ledcWriteTone(0, 0); // Stop the buzzer
}
```

- `uint32_t ledcWriteTone(uint8_t chan, uint32_t freq);`: Diese Funktion wird verwendet, um den LEDC-Kanal auf 50 % PWM-Ton bei ausgewählter Frequenz einzustellen.
  - `chan` wählt LEDC-Kanal aus.
  - `freq` wählt Frequenz des PWM-Signals aus.

Diese Funktion gibt die `frequency` für den Kanal zurück. Wenn `0` zurückgegeben wird, ist ein Fehler aufgetreten und der LEDC-Kanal wurde nicht konfiguriert.

4. Konfigurieren Sie den PWM-Kanal und verbinden Sie den Summer-Pin in der Funktion `setup()`.

```
void setup() {
    ledcSetup(0, 2000, resolution); // Set up the PWM channel
    ledcAttachPin(buzzerPin, 0); // Attach the buzzer pin to the PWM channel
}
```

- `uint32_t ledcSetup(uint8_t channel, uint32_t freq, uint8_t resolution_bits);`: Diese Funktion wird verwendet, um die Frequenz und Auflösung des LEDC-Kanals einzurichten. Sie gibt die für den LEDC-Kanal konfigurierte `frequency` zurück. Wenn 0 zurückgegeben wird, ist ein Fehler aufgetreten und der LEDC-Kanal wurde nicht konfiguriert.
  - `channel` wählt LEDC-Kanal aus.
  - `freq` wählt Frequenz des PWM aus.
  - `resolution_bits` wählt Auflösung für LEDC-Kanal aus. Bereich ist 1-14 Bits (1-20 Bits für ESP32).
- `void ledcAttachPin(uint8_t pin, uint8_t chan);`: Diese Funktion wird verwendet, um den Pin an den LEDC-Kanal anzuschließen.
  - `pin` wählt GPIO-Pin aus.
  - `chan` wählt LEDC-Kanal aus.

5. In der Funktion `loop()` spielen Sie die Sequenz von 7 Noten mit einer kurzen Pause zwischen jeder Note und einer 1-sekündigen Pause vor der Wiederholung der Sequenz ab.

```
void loop() {
    for (int i = 0; i < 7; i++) {
        playFrequency(frequencies[i], 300); // Play each note for 300ms
        delay(50); // Add a brief pause between the notes
    }
    delay(1000); // Wait for 1 second before replaying the sequence
}
```

4. Aktoren

2.14 4.1 Motor

In diesem spannenden Projekt werden wir erforschen, wie man einen Motor mit dem L293D antreibt.

Der L293D ist ein vielseitiger integrierter Schaltkreis (IC), der häufig für die Motorsteuerung in Elektronik- und Robotikprojekten verwendet wird. Er kann zwei Motoren sowohl vorwärts als auch rückwärts antreiben, was ihn zu einer beliebten Wahl für Anwendungen macht, die eine präzise Motorsteuerung erfordern.

Am Ende dieses fesselnden Projekts werden Sie ein gründliches Verständnis dafür haben, wie digitale Signale und PWM-Signale effektiv zur Steuerung von Motoren eingesetzt werden können. Dieses unschätzbare Wissen wird eine solide Grundlage für Ihre zukünftigen Unternehmungen in Robotik und Mechatronik sein. Schnallen Sie sich an und machen Sie sich bereit, in die aufregende Welt der Motorsteuerung mit dem L293D einzutauchen!

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

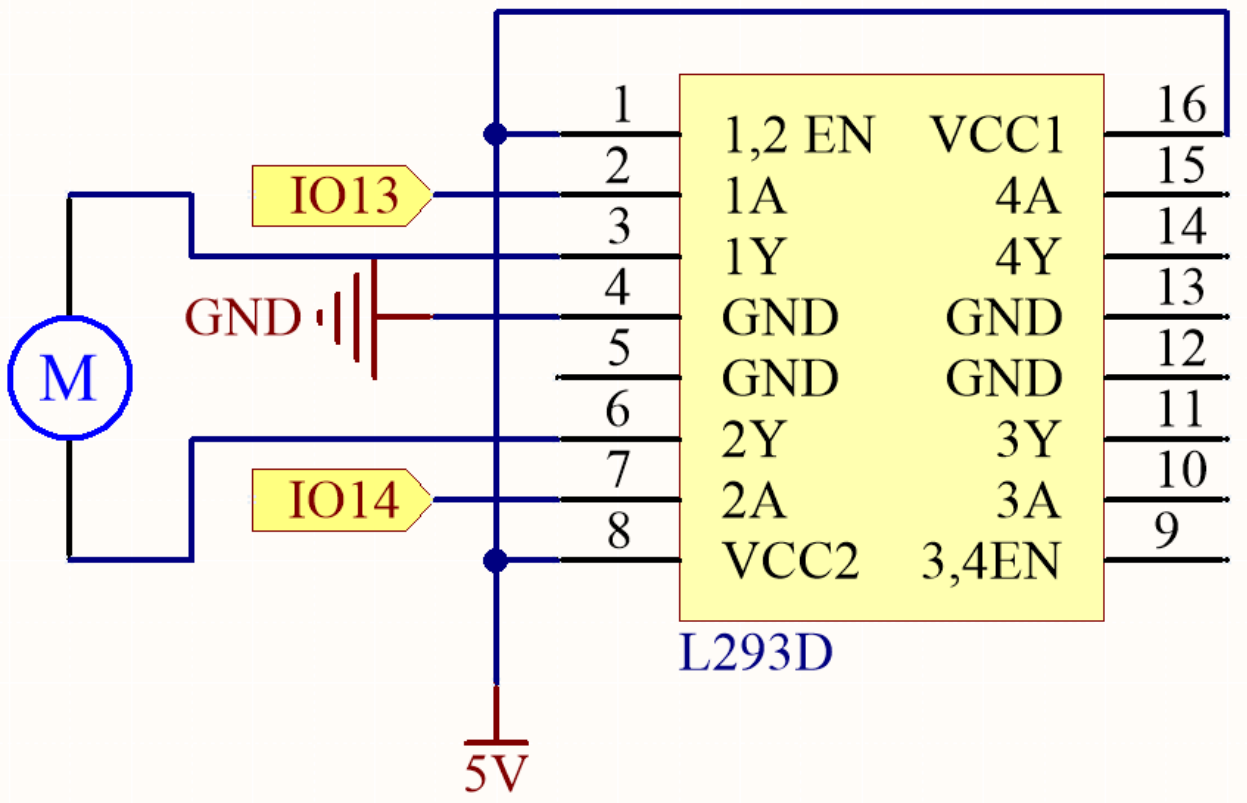
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Gleichstrommotor	
L293D	-

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

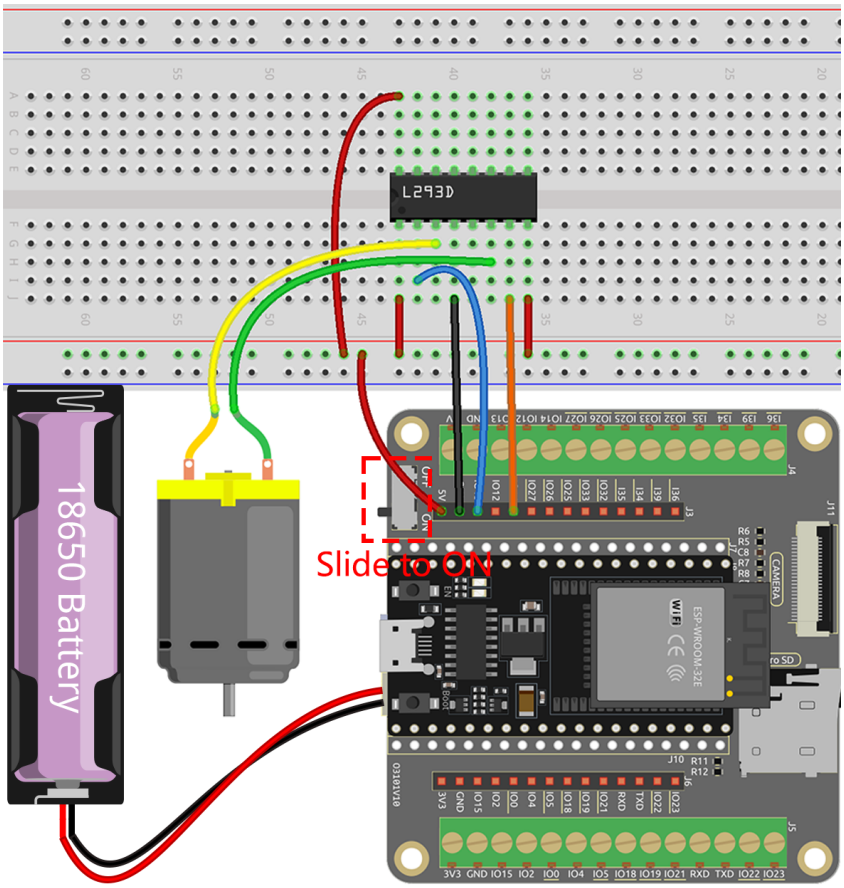
Verfügbare Pins	IO13, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Verdrahtung

Bemerkung: Da der Motor einen relativ hohen Strom benötigt, ist es notwendig, zuerst die Batterie einzulegen und dann den Schalter auf dem Erweiterungsboard auf die Position ON zu schieben, um die Batterieversorgung zu aktivieren.



Code

Bemerkung:

- Öffnen Sie die Datei `4.1_motor.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\4.1_motor`.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, werden Sie beobachten, dass der Motor sich eine Sekunde lang im Uhrzeigersinn dreht, dann eine Sekunde lang gegen den Uhrzeigersinn, gefolgt von einer zweisekündigen Pause. Diese Sequenz von Aktionen wird in einer endlosen Schleife fortgesetzt.

Mehr erfahren

Zusätzlich zum einfachen Drehen des Motors im und gegen den Uhrzeigersinn können Sie auch die Geschwindigkeit der Motordrehung steuern, indem Sie auf dem Steuerpin eine Pulsweitenmodulation (PWM) verwenden, wie unten gezeigt.

Bemerkung:

- Öffnen Sie die Datei `4.1_motor_pwm.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\4.1_motor_pwm`.

- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Der vorherige Code setzt direkt die beiden Pins des Motors auf hohe oder niedrige Spannungspegel, um die Drehung und das Anhalten des Motors zu steuern.

Hier verwenden wir das (LED-Steuerung) Peripheriegerät, um PWM-Signale zu erzeugen, um die Motorgeschwindigkeit zu steuern. Durch zwei `for` Schleifen wird der Tastgrad von Kanal A von 0 auf 255 erhöht oder verringert, während Kanal B bei 0 bleibt.

Auf diese Weise können Sie beobachten, wie der Motor seine Geschwindigkeit allmählich auf 255 erhöht und dann auf 0 verringert, unendlich so weiterlaufend.

Wenn Sie möchten, dass der Motor sich in die entgegengesetzte Richtung dreht, tauschen Sie einfach die Werte von Kanal A und Kanal B.

2.15 4.2 Pumpen

In diesem faszinierenden Projekt werden wir uns mit der Steuerung einer Wasserpumpe mit dem L293D befassen.

Bei der Steuerung von Wasserpumpen ist es etwas einfacher als bei anderen Motoren. Die Schönheit dieses Projekts liegt in seiner Einfachheit - es besteht keine Notwendigkeit, sich um die Drehrichtung zu kümmern. Unser Hauptziel ist es, die Wasserpumpe erfolgreich zu aktivieren und in Betrieb zu halten.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

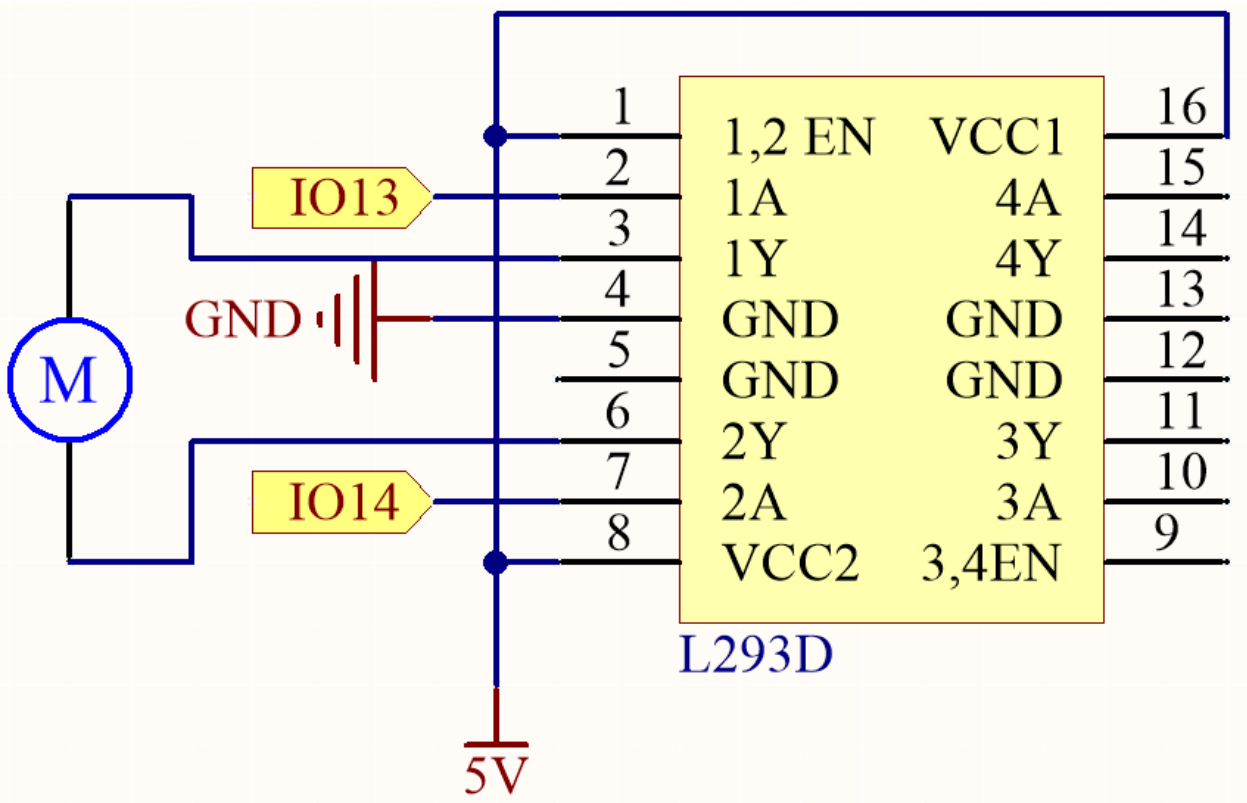
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Zentrifugalpumpe	-
L293D	-

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

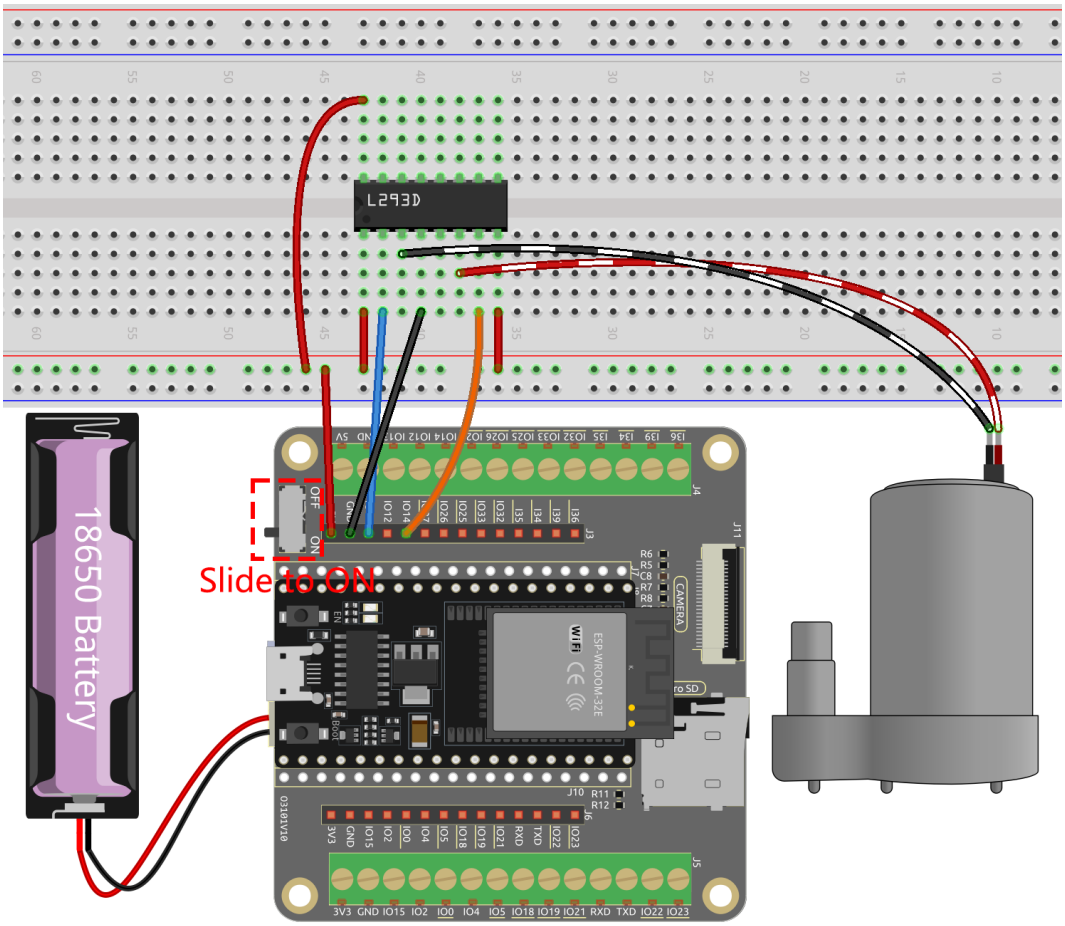
Verfügbare Pins	IO13, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Verdrahtung

Bemerkung: Hier wird empfohlen, zuerst die Batterie einzulegen und dann den Schalter auf dem Erweiterungsboard auf die Position ON zu schieben, um die Batterieversorgung zu aktivieren.



Code

Bemerkung:

- Sie können die Datei `4.2_pump.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\4.2_pump` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Verbinden Sie den Schlauch mit der Pumpe und platzieren Sie ihn im wassergefüllten Behälter. Sobald der Code erfolgreich hochgeladen wurde, werden Sie beobachten, wie das Wasser im Behälter allmählich abgepumpt wird. Während dieses Experiments stellen Sie bitte sicher, dass der elektrische Stromkreis von Wasser ferngehalten wird, um einen Kurzschluss zu verhindern!

2.16 4.3 Schwingender Servo

Ein Servo ist eine Art positionsbasiertes Gerät, das für seine Fähigkeit bekannt ist, spezifische Winkel zu halten und präzise Drehungen zu liefern. Dies macht es besonders wünschenswert für Steuerungssysteme, die eine konstante Winkelverstellung erfordern. Es ist nicht verwunderlich, dass Servos in hochwertigem ferngesteuertem Spielzeug weit verbreitet sind, von Flugzeugmodellen bis hin zu U-Boot-Repliken und anspruchsvollen ferngesteuerten Robotern.

In diesem faszinierenden Abenteuer werden wir uns der Herausforderung stellen, den Servo auf eine einzigartige Weise zu manipulieren - indem wir ihn schwingen lassen! Dieses Projekt bietet eine hervorragende Gelegenheit, tiefer in die Dynamik der Servos einzutauchen, Ihre Fähigkeiten in präzisen Steuerungssystemen zu schärfen und ein tieferes Verständnis für ihre Funktionsweise zu erlangen.

Sind Sie bereit, den Servo nach Ihrer Melodie tanzen zu lassen? Dann begeben wir uns auf diese spannende Reise!

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

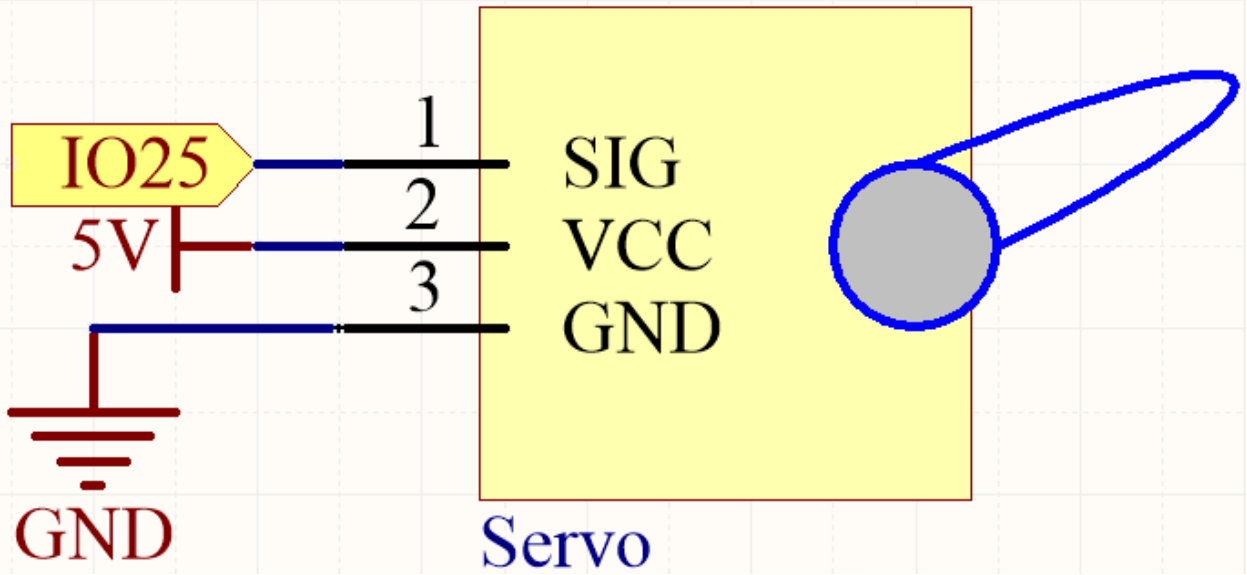
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Servo	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

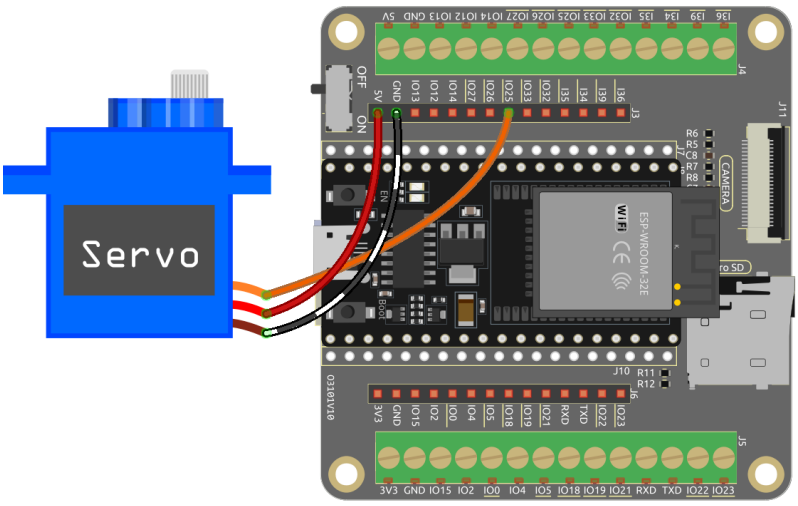
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Verdrahtung

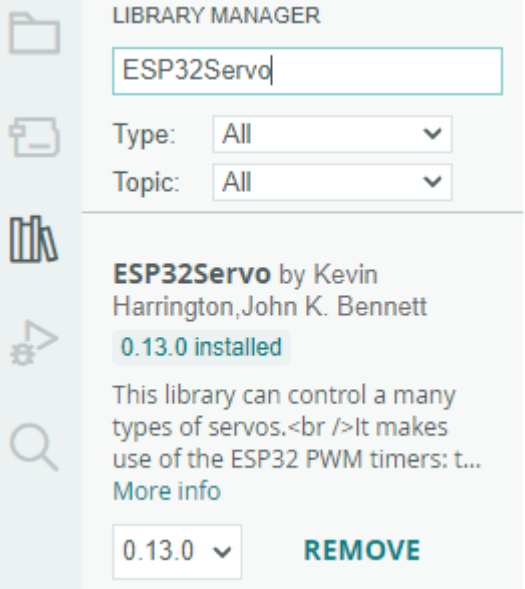
- Orangefarbenes Kabel ist das Signal und an IO25 angeschlossen.
- Rotes Kabel ist VCC und an 5V angeschlossen.
- Braunes Kabel ist GND und an GND angeschlossen.



Code

Bemerkung:

- Öffnen Sie die Datei `4.3_servo.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\4.3_servo`. Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*
- Hier wird die `ESP32Servo`-Bibliothek verwendet, die Sie über den **Library Manager** installieren können.



Nachdem Sie den Code erfolgreich hochgeladen haben, können Sie sehen, wie der Servoarm im Bereich von 0°~180° rotiert.

Wie funktioniert das?

1. Die Bibliothek einbinden: Diese Zeile importiert die ESP32Servo-Bibliothek, die erforderlich ist, um den Servomotor zu steuern.

```
#include <ESP32Servo.h>
```

2. Den Servo und den Pin, an den er angeschlossen ist, definieren: Dieser Abschnitt deklariert ein Servo-Objekt (`myServo`) und eine konstante Ganzzahl (`servoPin`), um den Pin darzustellen, an den der Servomotor angeschlossen ist (Pin 25).

```
// Define the servo and the pin it is connected to
Servo myServo;
const int servoPin = 25;
```

3. Die minimalen und maximalen Pulsbreiten für den Servo definieren: Dieser Abschnitt legt die minimalen und maximalen Pulsbreiten für den Servomotor fest (0,5 ms und 2,5 ms).

```
// Define the minimum and maximum pulse widths for the servo
const int minPulseWidth = 500; // 0.5 ms
const int maxPulseWidth = 2500; // 2.5 ms
```

4. Die `setup` Funktion initialisiert den Servomotor, indem sie ihn an den angegebenen Pin anhängt und seinen Pulsbreitenbereich festlegt. Sie stellt auch die PWM-Frequenz für den Servo auf die Standardfrequenz von 50 Hz ein.

```
void setup() {
    // Attach the servo to the specified pin and set its pulse width range
    myServo.attach(servoPin, minPulseWidth, maxPulseWidth);

    // Set the PWM frequency for the servo
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    myServo.setPeriodHertz(50); // Standard 50Hz servo
}
```

- `attach (int pin, int min, int max)`: Diese Funktion hängt den Servomotor an den angegebenen GPIO-Pin und legt die minimalen und maximalen Pulsbreiten für den Servo fest.
  - `pin`: Die GPIO-Pinnummer, an die der Servo angeschlossen ist.
  - `min` und `max`: Die minimalen und maximalen Pulsbreiten in Mikrosekunden. Diese Werte definieren den Bewegungsbereich des Servomotors.
- `setPeriodHertz(int hertz)`: Diese Funktion legt die PWM-Frequenz für den Servomotor in Hertz fest.
  - `hertz`: Die gewünschte PWM-Frequenz in Hertz. Die Standard-PWM-Frequenz für Servos beträgt 50Hz, was für die meisten Anwendungen geeignet ist.

5. Die `loop` Funktion ist der Hauptteil des Codes, der kontinuierlich läuft. Sie dreht den Servomotor von 0 bis 180 Grad und dann wieder zurück auf 0 Grad. Dies geschieht, indem der Winkel in die entsprechende Pulsbreite umgerechnet und der Servomotor mit dem neuen Pulsbreitenwert aktualisiert wird.

```
void loop() {
    // Rotate the servo from 0 to 180 degrees
    for (int angle = 0; angle <= 180; angle++) {
        int pulseWidth = map(angle, 0, 180, minPulseWidth, maxPulseWidth);
        myServo.writeMicroseconds(pulseWidth);
        delay(15);
    }

    // Rotate the servo from 180 to 0 degrees
    for (int angle = 180; angle >= 0; angle--) {
        int pulseWidth = map(angle, 0, 180, minPulseWidth, maxPulseWidth);
        myServo.writeMicroseconds(pulseWidth);
        delay(15);
    }
}
```

- `writeMicroseconds(int value)`: Diese Funktion setzt die Pulsbreite des Servomotors in Mikrosekunden.
  - `value`: Die gewünschte Pulsbreite in Mikrosekunden.

  Die Funktion `writeMicroseconds(int value)` nimmt einen Ganzzahlwert als Argument, der die gewünschte Pulsbreite in Mikrosekunden darstellt. Dieser Wert sollte typischerweise innerhalb des Bereichs liegen, der durch die zuvor im Code definierten minimalen und maximalen Pulsbreiten (`minPulseWidth` und `maxPulseWidth`) festgelegt wurde. Die Funktion stellt dann die Pulsbreite für den Servomotor ein, wodurch dieser sich in die entsprechende Position bewegt.

5. Sensoren

2.17 5.1 Lesen des Tasterwerts

In diesem interaktiven Projekt werden wir uns in die Welt der Tastersteuerung und LED-Manipulation begeben.

Das Konzept ist einfach, aber effektiv. Wir werden den Zustand eines Tasters lesen. Wenn der Taster gedrückt wird, registriert er ein hohes Spannungsniveau oder einen ‚hohen Zustand'. Diese Aktion löst dann das Aufleuchten einer LED aus.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
Taste	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Für Eingang	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Für Ausgang	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Bedingte Verwendung Pins (Eingang)**

  Die folgenden Pins haben eingebaute Pull-up- oder Pull-down-Widerstände, sodass externe Widerstände nicht erforderlich sind, wenn **sie als Eingangspins verwendet werden**:

Bedingte Verwendung Pins	Beschreibung
IO13, IO15, IO2, IO4	Hochziehen mit einem 47K-Widerstand setzt den Wert standardmäßig auf hoch.
IO27, IO26, IO33	Hochziehen mit einem 4.7K-Widerstand setzt den Wert standardmäßig auf hoch.
IO32	Herunterziehen mit einem 1K-Widerstand setzt den Wert standardmäßig auf niedrig.

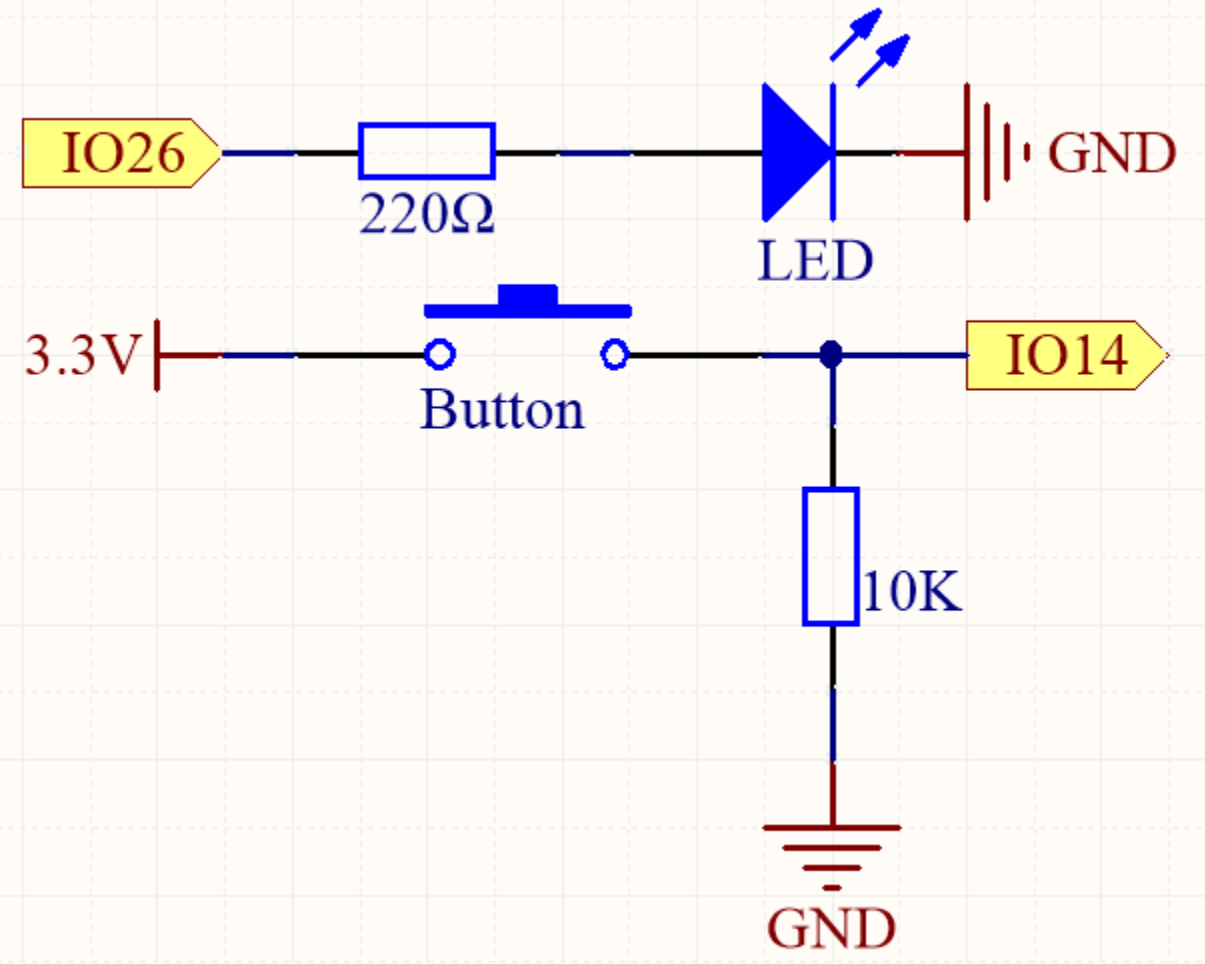
- **Strapping Pins (Eingang)**

  Strapping-Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um bestimmte Boot-Modi während des Gerätestarts (d.h., Einschalt-Reset) zu bestimmen.

Strapping Pins	IO5, IO0, IO2, IO12, IO15

  Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, berücksichtigen Sie den potenziellen Einfluss auf den Boot-Vorgang. Weitere Details finden Sie im Abschnitt *Strapping-Pins*.

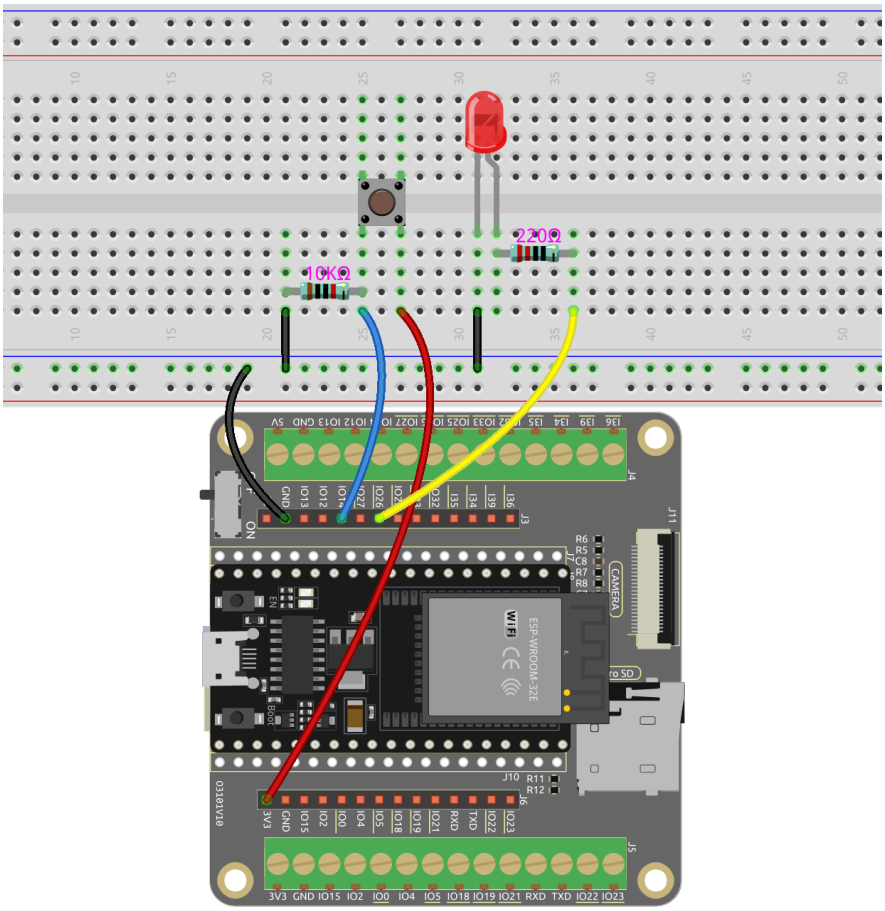
Schaltplan



Um eine ordnungsgemäße Funktionalität zu gewährleisten, verbinden Sie eine Seite des Taster-Pins mit 3,3V und die

andere Seite mit IO14. Wenn der Taster gedrückt wird, wird IO14 auf hoch gesetzt, was dazu führt, dass die LED aufleuchtet. Wird der Taster losgelassen, kehrt IO14 in seinen schwebenden Zustand zurück, der entweder hoch oder niedrig sein kann. Um ein stabiles niedriges Niveau zu gewährleisten, wenn der Taster nicht gedrückt ist, sollte IO14 über einen 10K-Pull-Down-Widerstand mit GND verbunden werden.

Verdrahtung



Bemerkung: Ein Vier-Pin-Taster ist in H-Form gestaltet. Wenn der Taster nicht gedrückt ist, sind die linke und rechte Pins getrennt, und es kann kein Strom zwischen ihnen fließen. Wenn der Taster jedoch gedrückt wird, werden die linke und rechte Pins verbunden, wodurch ein Stromweg entsteht.

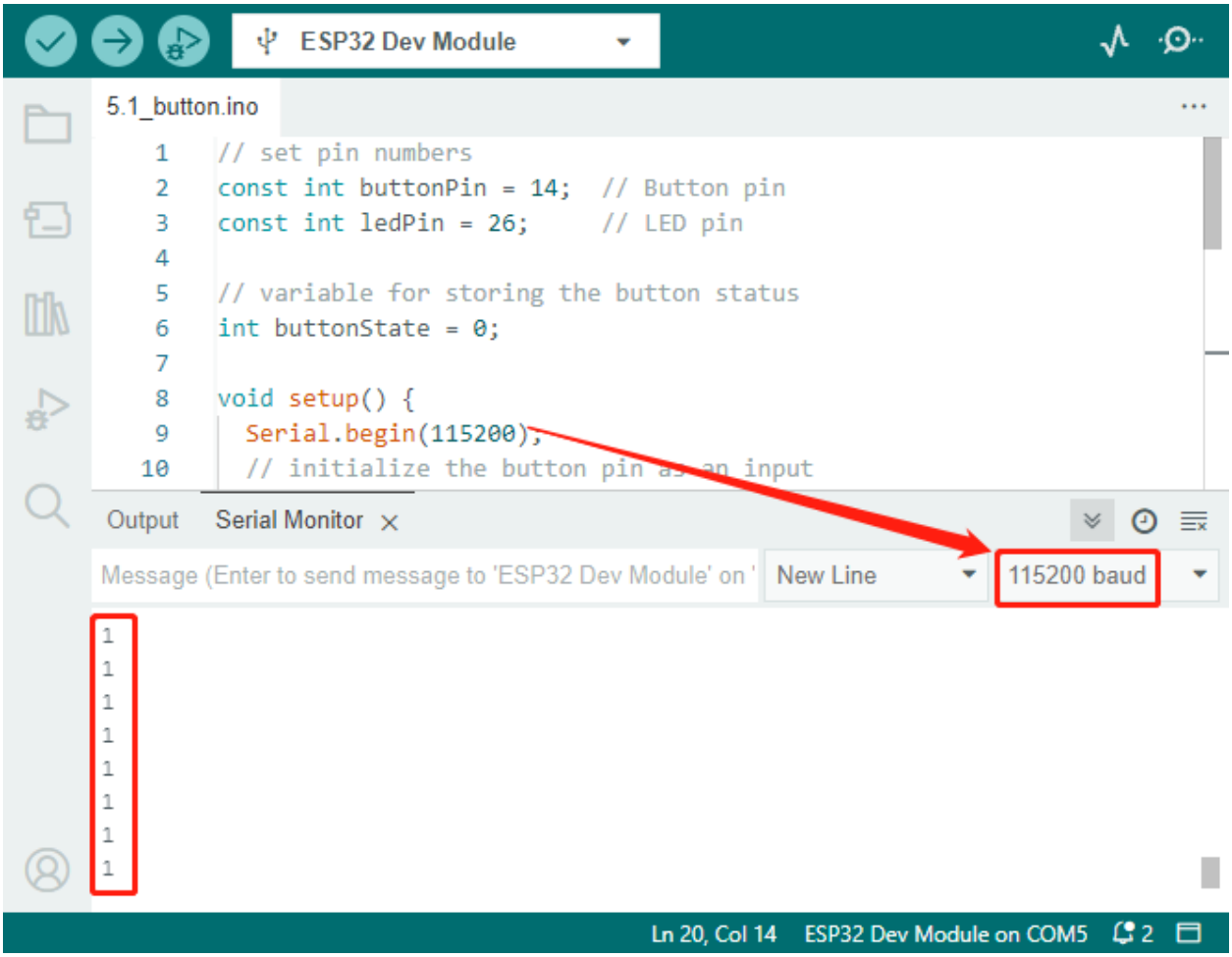
Code

Bemerkung:

- Sie können die Datei `5.1_button.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.1_button` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, leuchtet die LED auf, wenn Sie den Taster drücken, und geht aus, wenn Sie ihn loslassen.

Gleichzeitig können Sie den Seriellen Monitor in der oberen rechten Ecke öffnen, um den Wert des Tasters zu beobachten. Wenn der Taster gedrückt wird, wird „1“ gedruckt, ansonsten „0“.



Wie funktioniert das?

Die vorherigen Projekte beinhalteten alle das Ausgeben von Signalen, entweder in Form von digitalen oder PWM-Signalen.

Dieses Projekt beinhaltet das Empfangen von Eingangssignalen von externen Komponenten zum ESP32-Board. Sie können das Eingangssignal über den Seriellen Monitor in der Arduino IDE einsehen.

1. In der `setup()` Funktion wird der Taster-Pin als `input` und der LED-Pin als `output` initialisiert. Die serielle Kommunikation wird ebenfalls mit einer Baudrate von 115200 gestartet.

```
void setup() {
    Serial.begin(115200);
    // initialize the button pin as an input
    pinMode(buttonPin, INPUT);
    // initialize the LED pin as an output
    pinMode(ledPin, OUTPUT);
}
```

   - `Serial.begin(speed)`: Legt die Datenrate in Bits pro Sekunde (Baud) für die serielle Datenübertragung fest.
     - `speed`: in Bits pro Sekunde (Baud). Erlaubte Datentypen: `long`.

2. In der `loop()` Funktion wird der Zustand des Tasters gelesen und in der Variablen `buttonState` gespeichert. Der Wert von `buttonState` wird mit `Serial.println()` in den Seriellen Monitor gedruckt.

```
void loop() {
    // read the state of the button value
    buttonState = digitalRead(buttonPin);
    Serial.println(buttonState);
    delay(100);
    // if the button is pressed, the buttonState is HIGH
    if (buttonState == HIGH) {
        // turn LED on
        digitalWrite(ledPin, HIGH);

    } else {
        // turn LED off
        digitalWrite(ledPin, LOW);
    }
}
```

Wenn der Taster gedrückt und der `buttonState` HIGH ist, wird die LED eingeschaltet, indem der `ledPin` auf `HIGH` gesetzt wird. Andernfalls wird die LED ausgeschaltet.

- `int digitalRead(uint8_t pin);`: Um den Zustand eines als EINGANG konfigurierten Pins zu lesen, wird die Funktion digitalRead verwendet. Diese Funktion gibt den logischen Zustand des ausgewählten Pins als `HIGH` oder `LOW` zurück.
  - `pin` auswählen GPIO
- `Serial.println()`: Druckt Daten an den seriellen Port als lesbaren ASCII-Text, gefolgt von einem Wagenrücklaufzeichen (ASCII 13 oder ‚r') und einem Zeilenumbruchzeichen (ASCII 10 oder ‚n').

2.18 5.2 Kippen!

Der Kippschalter ist ein einfaches, aber effektives 2-Pin-Gerät, das eine Metallkugel in seiner Mitte enthält. Wenn der Schalter in einer aufrechten Position ist, sind die beiden Pins elektrisch verbunden, was den Stromfluss ermöglicht. Wenn der Schalter jedoch gekippt oder in einem bestimmten Winkel geneigt wird, bewegt sich die Metallkugel und unterbricht die elektrische Verbindung zwischen den Pins.

In diesem Projekt werden wir den Kippschalter nutzen, um die Beleuchtung einer LED zu steuern. Indem wir den Schalter so positionieren, dass die Kippaktion ausgelöst wird, können wir die LED basierend auf der Orientierung des Schalters ein- und ausschalten.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
Neigungsschalter	-

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Für Eingang	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Für Ausgang	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Bedingte Verwendung Pins (Eingang)**

  Die folgenden Pins haben eingebaute Pull-up- oder Pull-down-Widerstände, sodass externe Widerstände nicht erforderlich sind, wenn **sie als Eingangspins verwendet werden**:

Bedingte Verwendung Pins	Beschreibung
IO13, IO15, IO2, IO4	Hochziehen mit einem 47K-Widerstand setzt den Wert standardmäßig auf hoch.
IO27, IO26, IO33	Hochziehen mit einem 4.7K-Widerstand setzt den Wert standardmäßig auf hoch.
IO32	Herunterziehen mit einem 1K-Widerstand setzt den Wert standardmäßig auf niedrig.

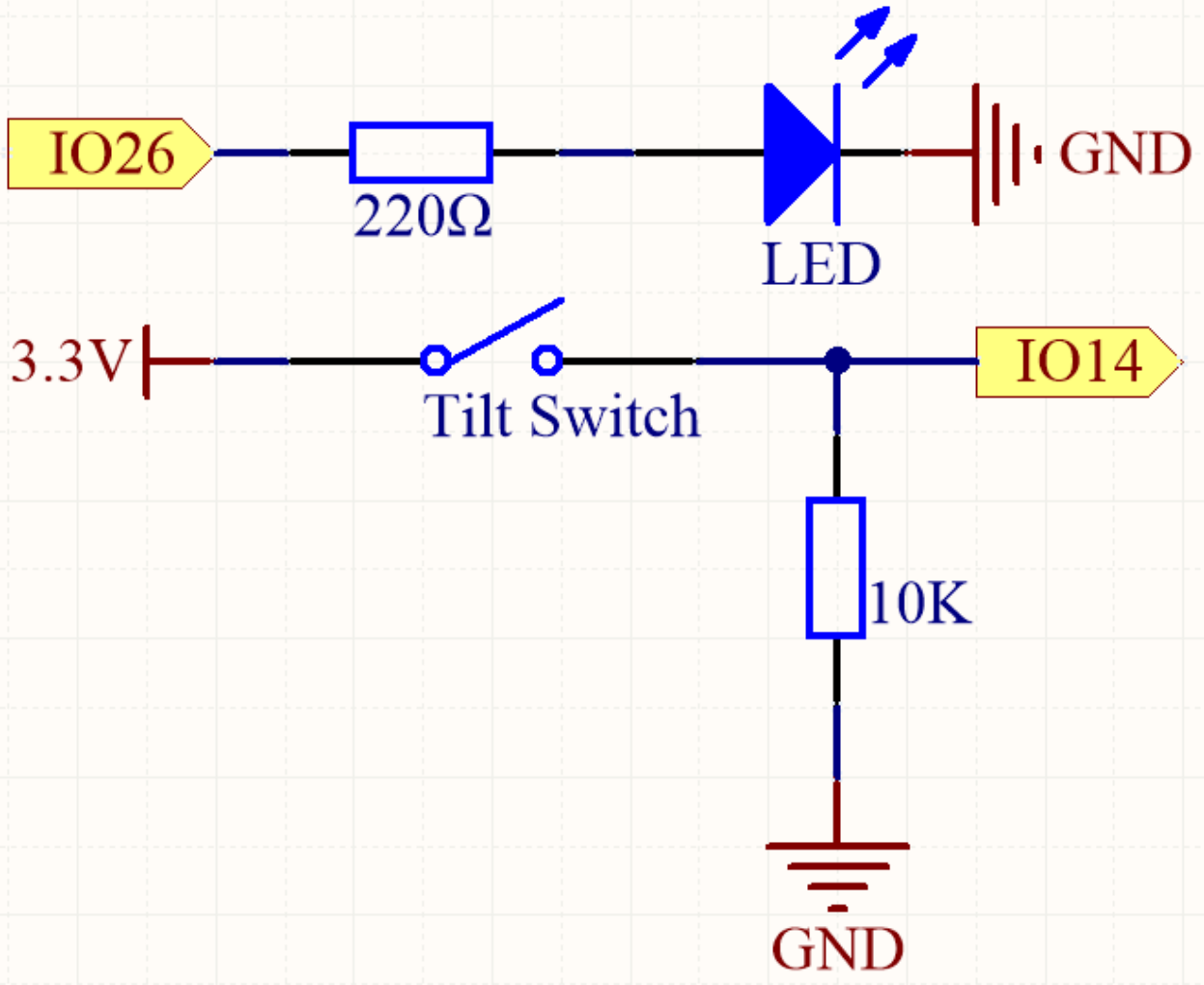
- **Strapping Pins (Eingang)**

  Strapping-Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um bestimmte Boot-Modi während des Gerätestarts (d.h., Einschalt-Reset) zu bestimmen.

Strapping Pins	IO5, IO0, IO2, IO12, IO15

  Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, berücksichtigen Sie den potenziellen Einfluss auf den Boot-Vorgang. Weitere Details finden Sie im Abschnitt *Strapping-Pins*.

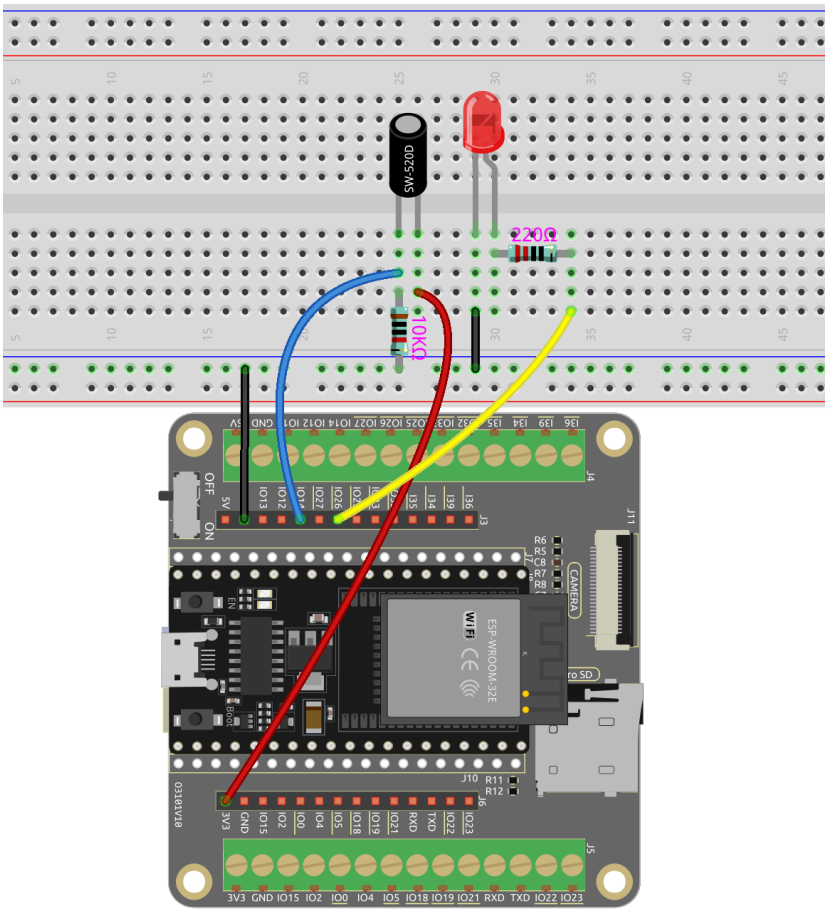
Schaltplan



Wenn der Kippschalter in einer aufrechten Position ist, wird IO14 auf hoch gesetzt, was dazu führt, dass die LED aufleuchtet. Umgekehrt wird IO14 auf niedrig gesetzt, wenn der Kippschalter geneigt ist, wodurch die LED ausgeschaltet wird.

Der Zweck des 10K-Widerstands besteht darin, einen stabilen niedrigen Zustand für IO14 aufrechtzuerhalten, wenn der Kippschalter geneigt ist.

Verdrahtung



Code

Bemerkung:

- Sie können die Datei `5.2_tilt_switch.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.2_tilt_switch` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, leuchtet die LED auf, wenn der Schalter aufrecht steht, und geht aus, wenn der Schalter geneigt wird.

2.19 5.3 Hindernis Erkennen

Dieses Modul wird häufig auf Autos und Robotern installiert, um die Existenz von Hindernissen voraus zu beurteilen. Es wird auch weit verbreitet in Handheld-Geräten, Wasserhähnen und so weiter eingesetzt.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Modul zur Hindernisvermeidung	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23

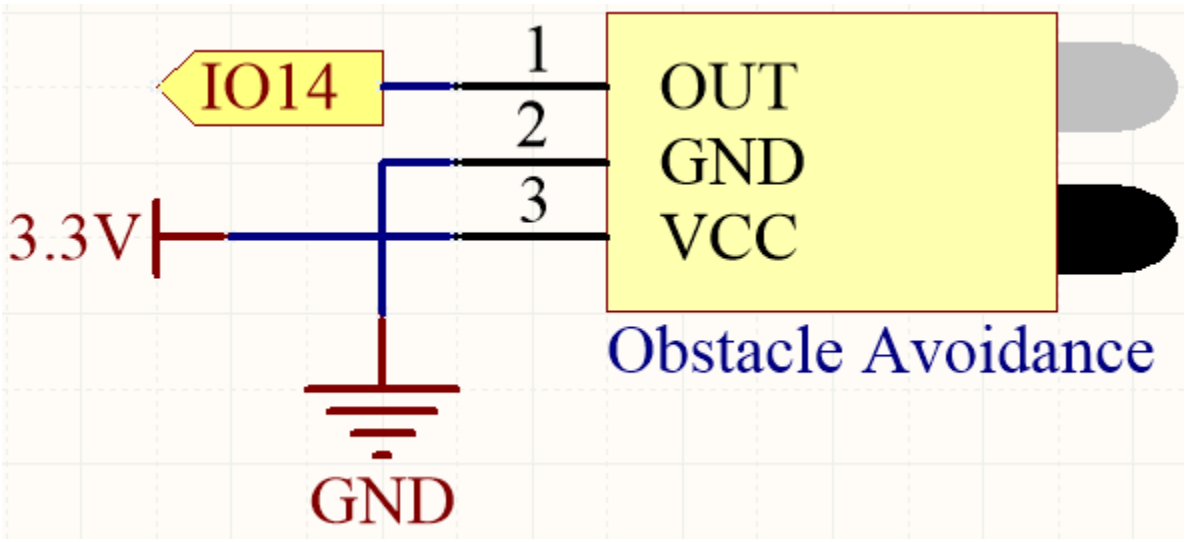
- **Strapping Pins (Eingang)**

  Strapping-Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um bestimmte Boot-Modi während des Gerätestarts (d.h., Einschalt-Reset) zu bestimmen.

Strapping Pins	IO5, IO0, IO2, IO12, IO15

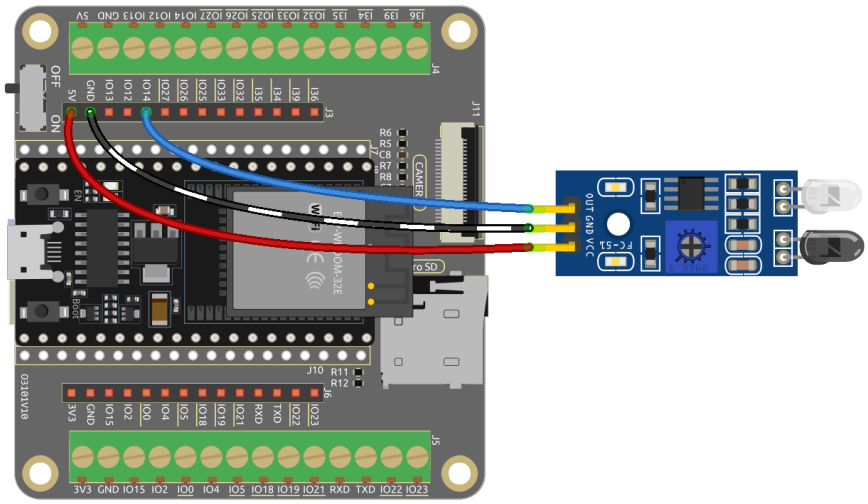
  Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, berücksichtigen Sie den potenziellen Einfluss auf den Boot-Vorgang. Weitere Details finden Sie im Abschnitt *Strapping-Pins*.

Schaltplan



Wenn das Hindernisvermeidungsmodul keine Hindernisse erkennt, gibt IO14 ein hohes Niveau zurück. Wenn es jedoch ein Hindernis erkennt, gibt es ein niedriges Niveau zurück. Sie können das blaue Potentiometer einstellen, um die Erkennungsdistanz dieses Moduls zu ändern.

Verdrahtung



Code

Bemerkung:

- Sie können die Datei `5.3.detect_the_obstacle.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.3.detect_the_obstacle` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, wird auf dem seriellen Monitor „0“ angezeigt, wenn das IR-Hindernisvermeidungsmodul etwas in seinem Weg erkennt, andernfalls wird „1“ angezeigt.

2.20 5.4 Linie Erkennen

Das Linienverfolgungsmodul wird verwendet, um das Vorhandensein von schwarzen Flächen auf dem Boden zu erkennen, wie zum Beispiel schwarze Linien, die mit Isolierband geklebt sind.

Sein Emitter sendet geeignetes Infrarotlicht in den Boden, das von schwarzen Oberflächen relativ absorbiert und schwach reflektiert wird. Das Gegenteil ist der Fall bei weißen Oberflächen. Wenn reflektiertes Licht erkannt wird, wird der Boden momentan als weiß angezeigt. Wird es nicht erkannt, wird er als schwarz angezeigt.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Linienverfolgungsmodul	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23

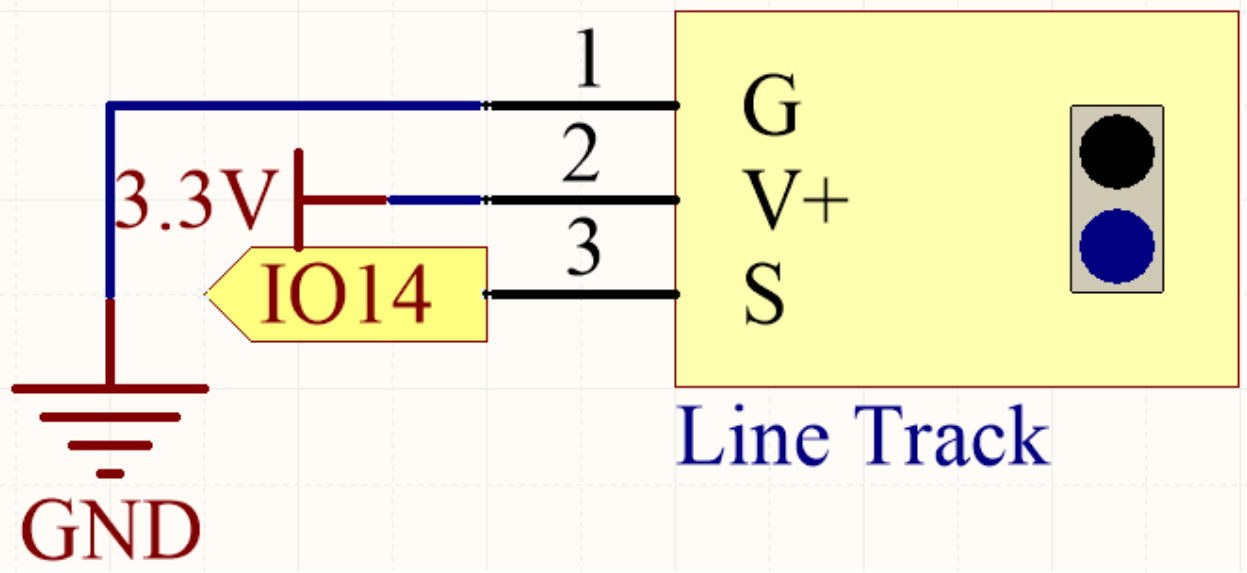
- **Strapping Pins (Eingang)**

  Strapping-Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um bestimmte Boot-Modi während des Gerätestarts (d.h., Einschalt-Reset) zu bestimmen.

Strapping Pins	IO5, IO0, IO2, IO12, IO15

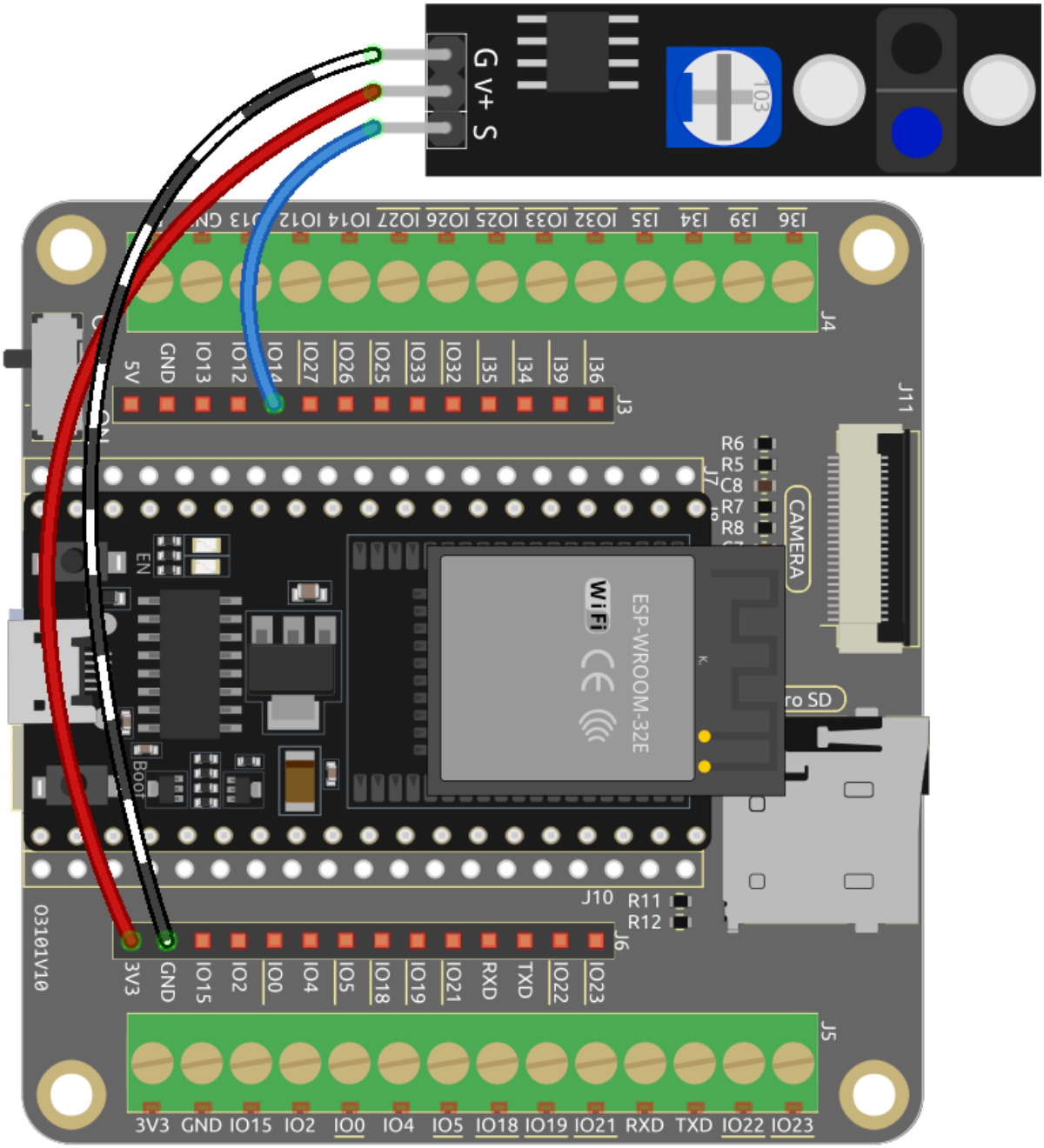
  Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, berücksichtigen Sie den potenziellen Einfluss auf den Boot-Vorgang. Weitere Details finden Sie im Abschnitt *Strapping-Pins*.

Schaltplan



Wenn das Linienverfolgungsmodul eine schwarze Linie erkennt, gibt IO14 ein hohes Niveau zurück. Andererseits gibt es ein niedriges Niveau zurück, wenn es eine weiße Linie erkennt. Sie können das blaue Potentiometer einstellen, um die Empfindlichkeit der Erkennung dieses Moduls zu ändern.

Verdrahtung



Code

Bemerkung:

- Sie können die Datei `5.4_detect_the_line.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.4_detect_the_line` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Wenn das Linienverfolgungsmodul eine schwarze Linie erkennt, nachdem der Code erfolgreich hochgeladen wurde,

wird „Schwarz“ im Seriellen Monitor angezeigt. Andernfalls wird „Weiß“ gedruckt.

2.21 5.5 Menschliche Bewegung Erkennen

Der passive Infrarotsensor (PIR-Sensor) ist ein gängiger Sensor, der Infrarotstrahlung (IR), die von Objekten in seinem Sichtfeld ausgestrahlt wird, messen kann. Einfach ausgedrückt, empfängt er Infrarotstrahlung, die vom Körper ausgestrahlt wird, und erkennt so die Bewegung von Menschen und anderen Tieren. Konkreter gesagt, teilt er der Hauptsteuerplatine mit, dass jemand Ihr Zimmer betreten hat.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
PIR-Bewegungssensormodul	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23
Für Ausgang	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Bemerkung: IO32 kann **nicht als Eingangspin** in diesem Projekt verwendet werden, da er intern mit einem 1K-Pull-Down-Widerstand verbunden ist, der seinen Standardwert auf 0 setzt.

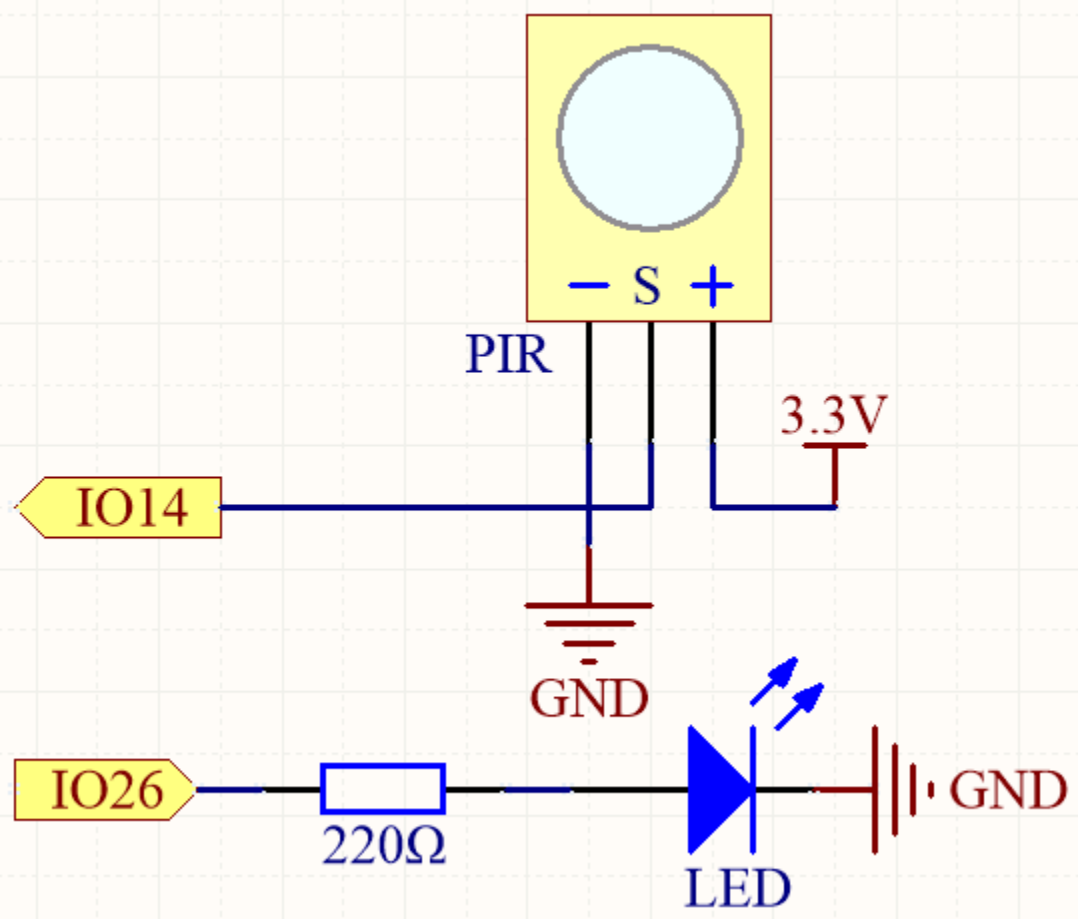
- **Strapping Pins (Eingang)**

Strapping-Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um bestimmte Boot-Modi während des Gerätestarts (d.h., Einschalt-Reset) zu bestimmen.

Strapping Pins	IO5, IO0, IO2, IO12, IO15

Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, berücksichtigen Sie den potenziellen Einfluss auf den Boot-Vorgang. Weitere Details finden Sie im Abschnitt *Strapping-Pins*.

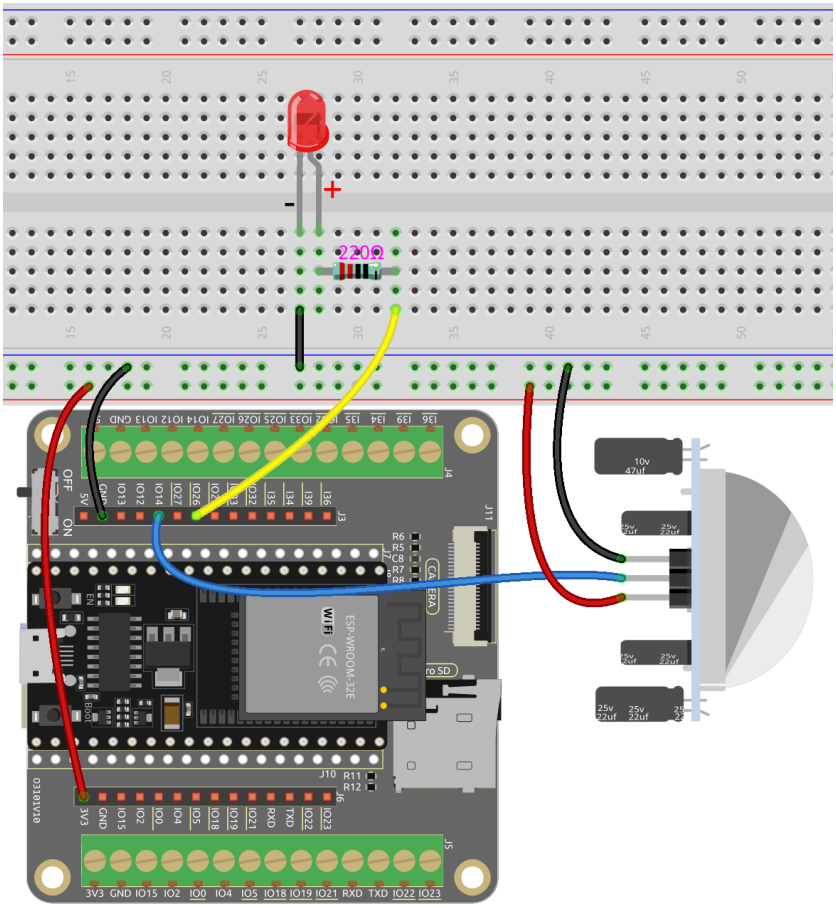
Schaltplan



Wenn das PIR-Modul eine Bewegung erkennt, wird IO14 hoch gesetzt und die LED leuchtet auf. Andernfalls, wenn keine Bewegung erkannt wird, wird IO14 niedrig gesetzt und die LED schaltet sich aus.

Bemerkung: Das PIR-Modul hat zwei Potentiometer: eines zur Einstellung der Empfindlichkeit, das andere zur Einstellung der Erkennungsdistanz. Um das PIR-Modul besser arbeiten zu lassen, müssen Sie beide gegen den Uhrzeigersinn bis zum Ende drehen.

Verdrahtung



Code

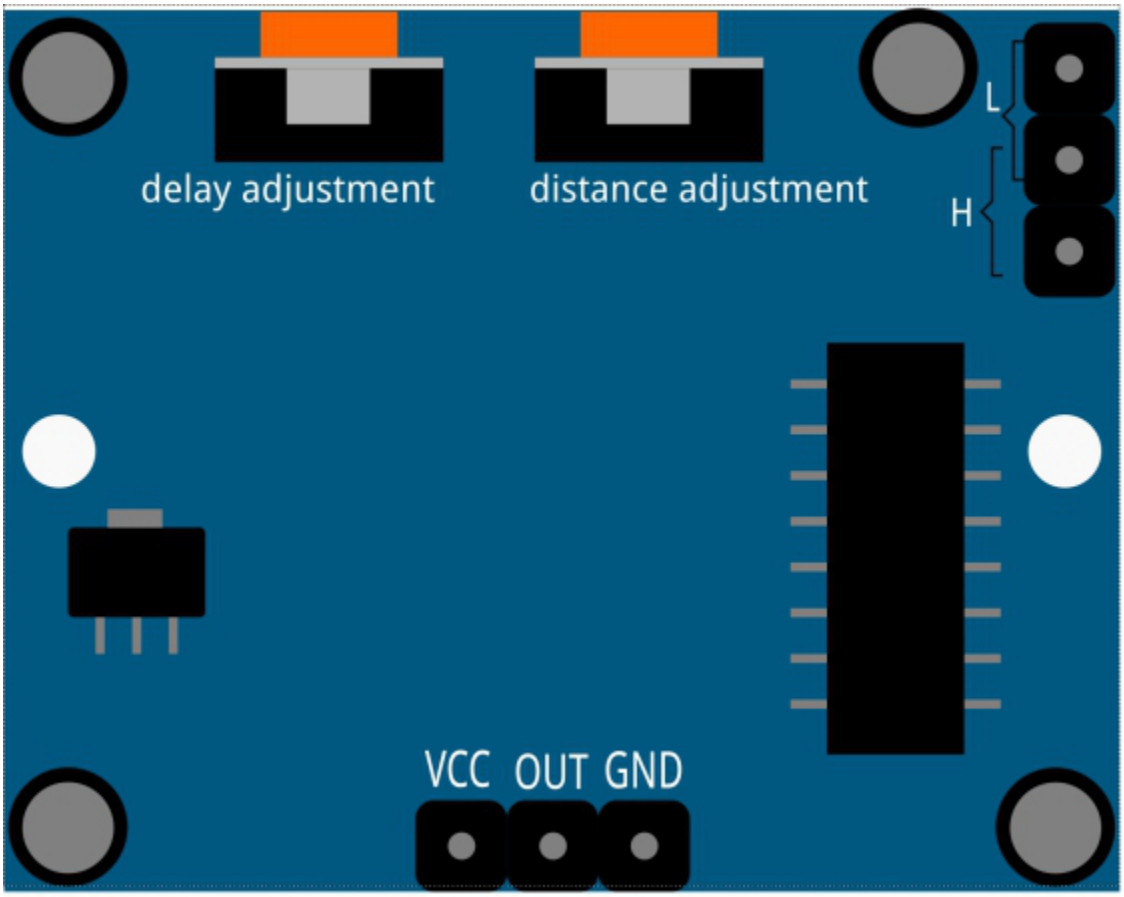
Bemerkung:

- Sie können die Datei `5.5_pir.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.5_pir` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.

- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, leuchtet die LED auf und geht aus, wenn das PIR-Modul eine vorbeigehende Person erkennt.

Bemerkung: Das PIR-Modul hat zwei Potentiometer: eines zur Einstellung der Empfindlichkeit, das andere zur Einstellung der Erkennungsdistanz. Um das PIR-Modul besser arbeiten zu lassen, müssen Sie beide gegen den Uhrzeigersinn bis zum Ende drehen.



2.22 5.6 Zwei Arten von Transistoren

Dieses Kit ist mit zwei Arten von Transistoren ausgestattet, S8550 und S8050, wobei ersterer ein PNP- und letzterer ein NPN-Transistor ist. Sie sehen sehr ähnlich aus, und wir müssen sorgfältig prüfen, um ihre Beschriftungen zu sehen. Wenn ein High-Level-Signal durch einen NPN-Transistor fließt, wird er aktiviert. Aber ein PNP-Transistor benötigt ein Low-Level-Signal, um ihn zu steuern. Beide Arten von Transistoren werden häufig für kontaktlose Schalter verwendet, genau wie in diesem Experiment.

Lassen Sie uns LED und Taster verwenden, um zu verstehen, wie man Transistoren einsetzt!

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
Taste	
Transistor	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Für Eingang	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Für Ausgang	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Bedingte Verwendung Pins (Eingang)**

  Die folgenden Pins haben eingebaute Pull-up- oder Pull-down-Widerstände, sodass externe Widerstände nicht erforderlich sind, wenn **sie als Eingangspins verwendet werden**:

Bedingte Verwendung Pins	Beschreibung
IO13, IO15, IO2, IO4	Hochziehen mit einem 47K-Widerstand setzt den Wert standardmäßig auf hoch.
IO27, IO26, IO33	Hochziehen mit einem 4.7K-Widerstand setzt den Wert standardmäßig auf hoch.
IO32	Herunterziehen mit einem 1K-Widerstand setzt den Wert standardmäßig auf niedrig.

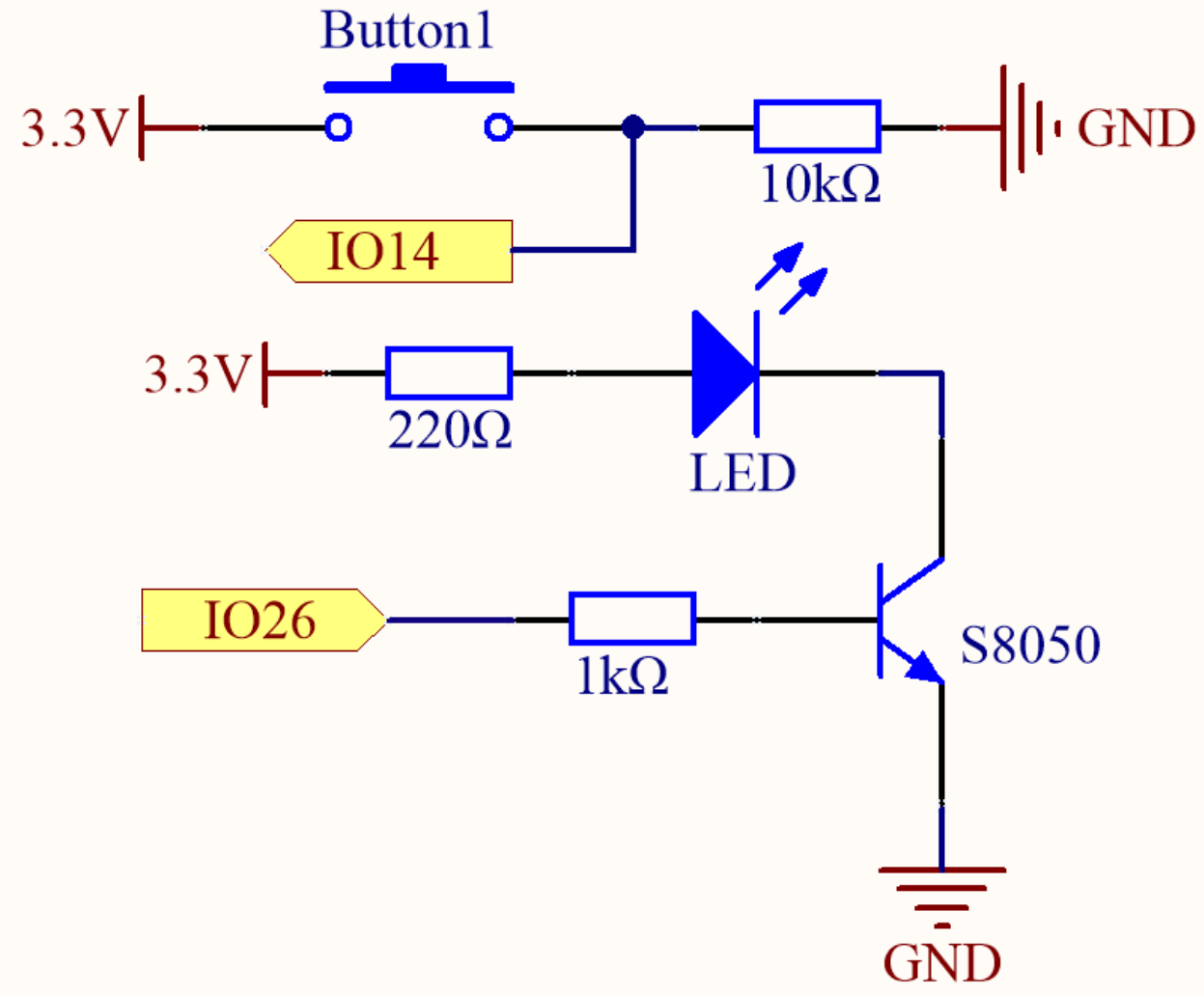
- **Strapping Pins (Eingang)**

  Strapping-Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um bestimmte Boot-Modi während des Gerätestarts (d.h., Einschalt-Reset) zu bestimmen.

Strapping Pins	IO5, IO0, IO2, IO12, IO15

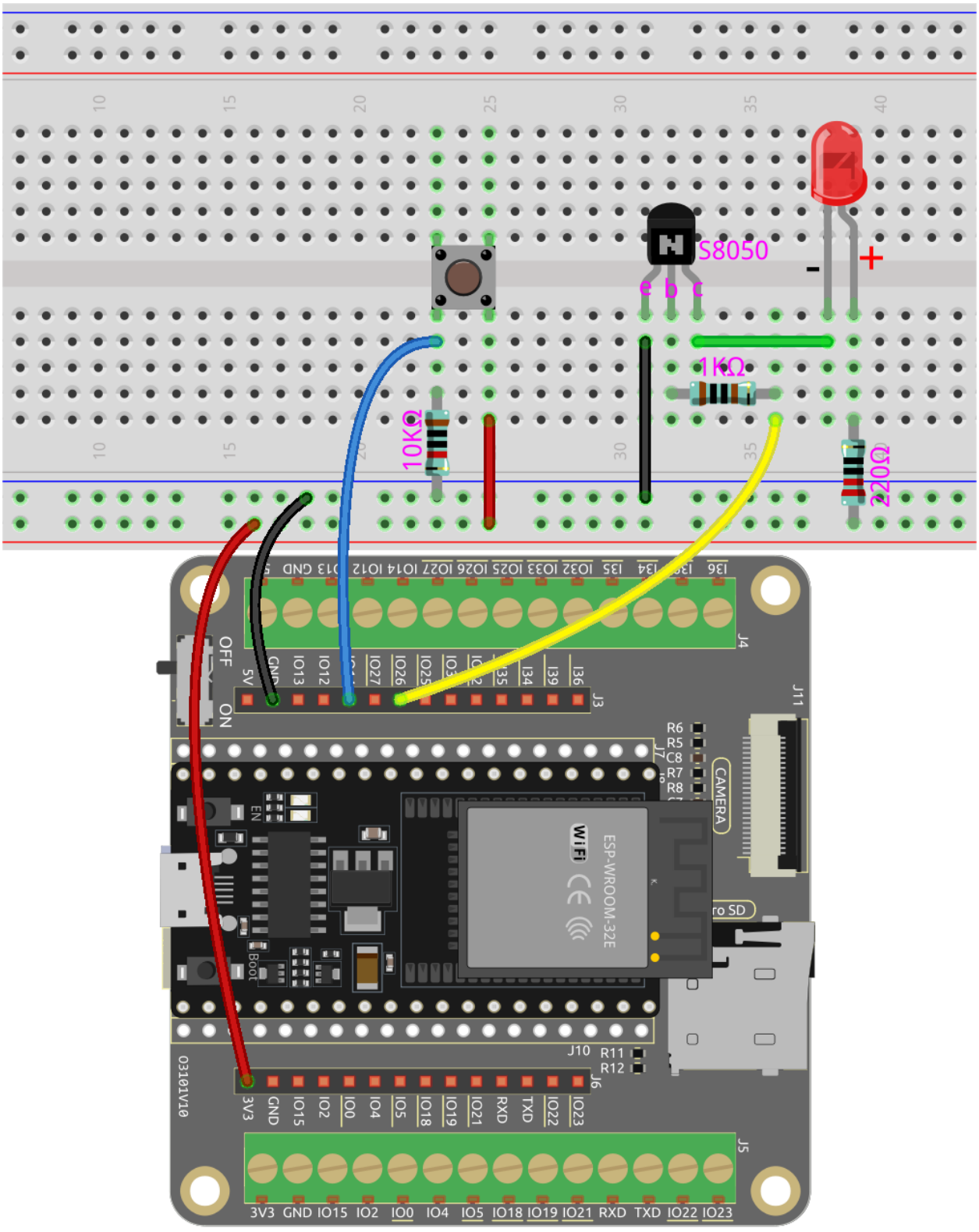
Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, berücksichtigen Sie den potenziellen Einfluss auf den Boot-Vorgang. Weitere Details finden Sie im Abschnitt *Strapping-Pins*.

Verbindungsmöglichkeit des NPN (S8050) Transistors

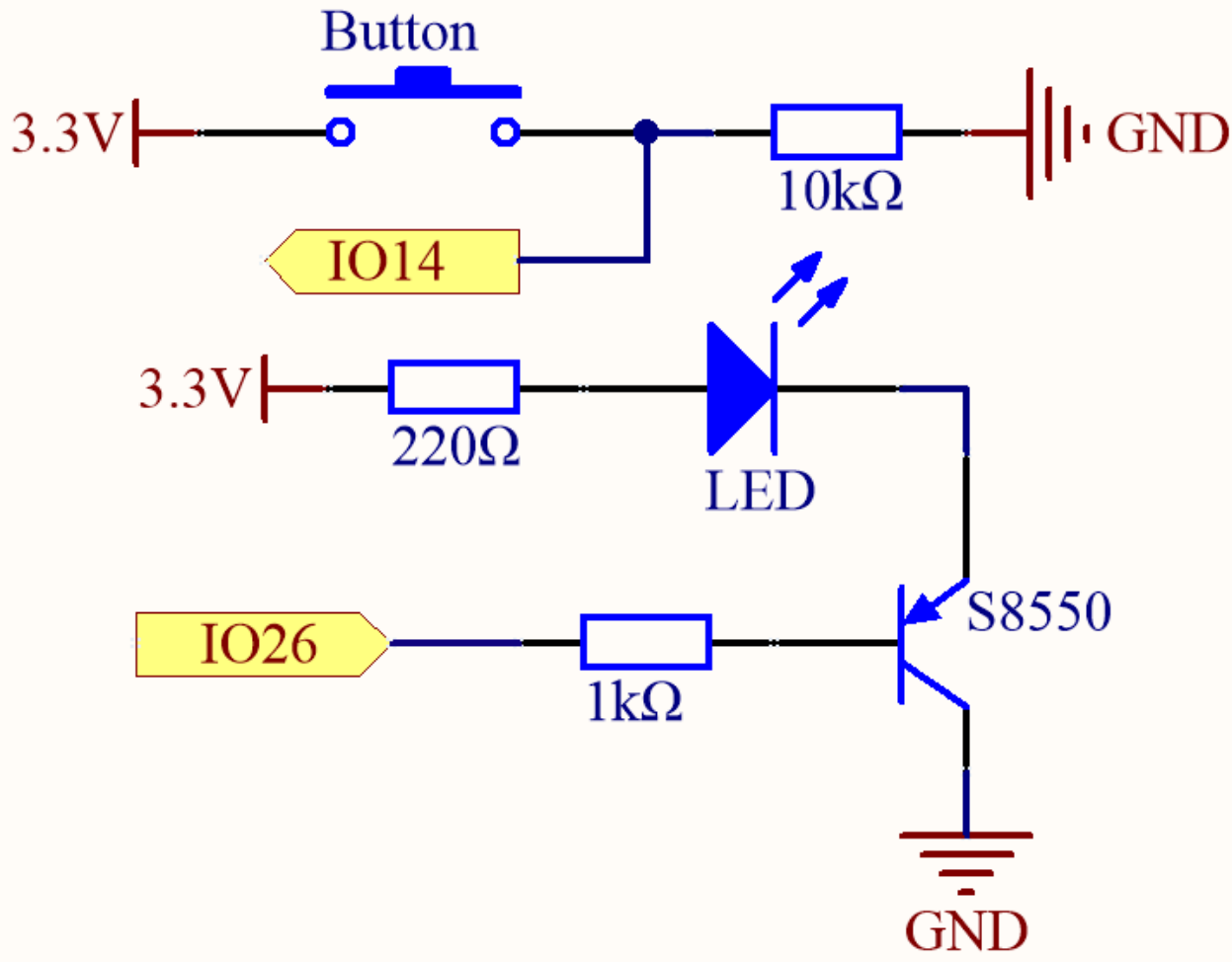


In diesem Schaltkreis wird IO14 hoch, wenn der Taster gedrückt wird.

Indem IO26 programmiert wird, um **high** auszugeben, nach einem 1k-Strombegrenzungswiderstand (um den Transistor zu schützen), wird der S8050 (NPN-Transistor) leitfähig, wodurch die LED aufleuchten kann.



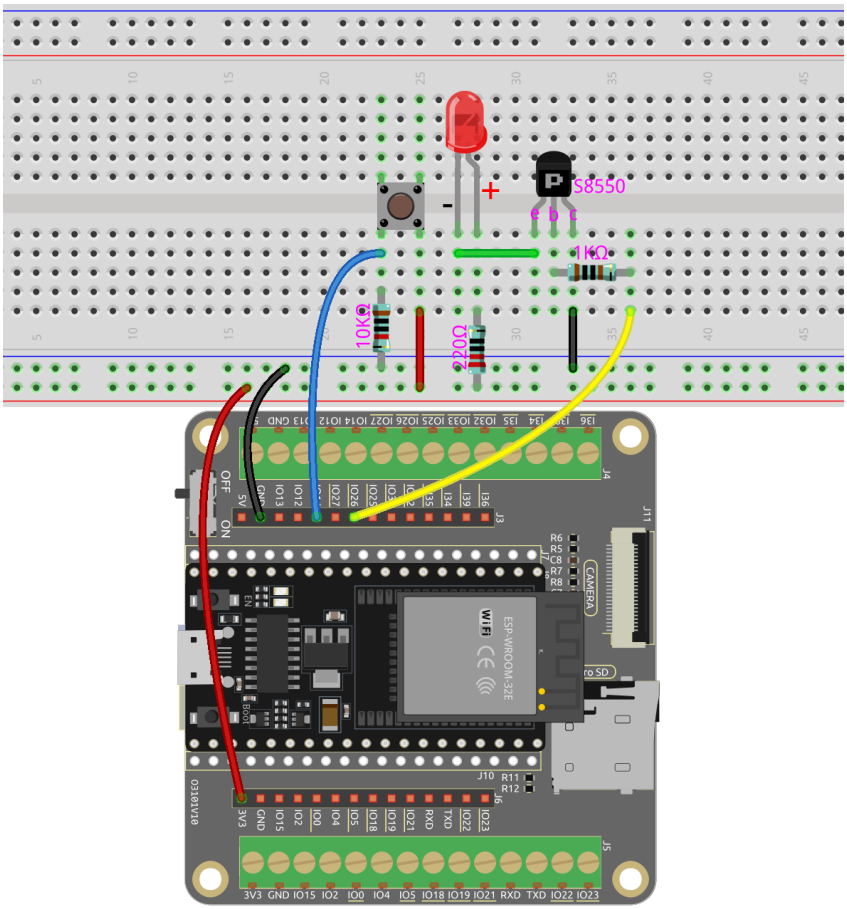
Verbindungsmöglichkeit des PNP(S8550) Transistors



In diesem Schaltkreis ist IO14 standardmäßig niedrig und wird hoch, wenn der Taster gedrückt wird.

Indem IO26 programmiert wird, um **low** auszugeben, nach einem 1k-Strombegrenzungswiderstand (um den Transistor zu schützen), wird der S8550 (PNP-Transistor) leitfähig, wodurch die LED aufleuchten kann.

Der einzige Unterschied, den Sie zwischen diesem Schaltkreis und dem vorherigen feststellen werden, ist, dass in dem vorherigen Schaltkreis die Kathode der LED mit dem **collector** des **S8050 (NPN transistor)** verbunden ist, während sie in diesem mit dem **emitter** des **S8550 (PNP transistor)** verbunden ist.



Code

Bemerkung:

- Sie können die Datei `5.6_transistor.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.6_transistor` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Beide Arten von Transistoren können mit demselben Code gesteuert werden. Wenn wir den Taster drücken, sendet der ESP32 ein Hochpegel-Signal an den Transistor; wenn wir ihn loslassen, sendet er ein Niedrigpegel-Signal.

- Der Schaltkreis mit dem S8050 (NPN-Transistor) leuchtet auf, wenn der Taster gedrückt wird, was darauf hindeutet, dass er sich in einem Hochpegel-Leitzustand befindet;
- Der Schaltkreis mit dem S8550 (PNP-Transistor) leuchtet auf, wenn der Taster losgelassen wird, was darauf hindeutet, dass er sich in einem Niedrigpegel-Leitzustand befindet.

2.23 5.7 Das Licht Fühlen

Der Fotowiderstand ist ein häufig verwendetes Gerät für analoge Eingaben, ähnlich einem Potentiometer. Sein Widerstandswert ändert sich abhängig von der Intensität des Lichts, das er empfängt. Bei starker Lichtexposition verringert sich der Widerstand des Fotowiderstands, und wenn die Lichtintensität abnimmt, steigt der Widerstand.

Indem wir den Wert des Fotowiderstands auslesen, können wir Informationen über die Umgebungslichtverhältnisse sammeln. Diese Informationen können für Aufgaben wie die Steuerung der Helligkeit einer LED, die Anpassung der Empfindlichkeit eines Sensors oder die Implementierung lichtabhängiger Aktionen in einem Projekt verwendet werden.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Fotowiderstand	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

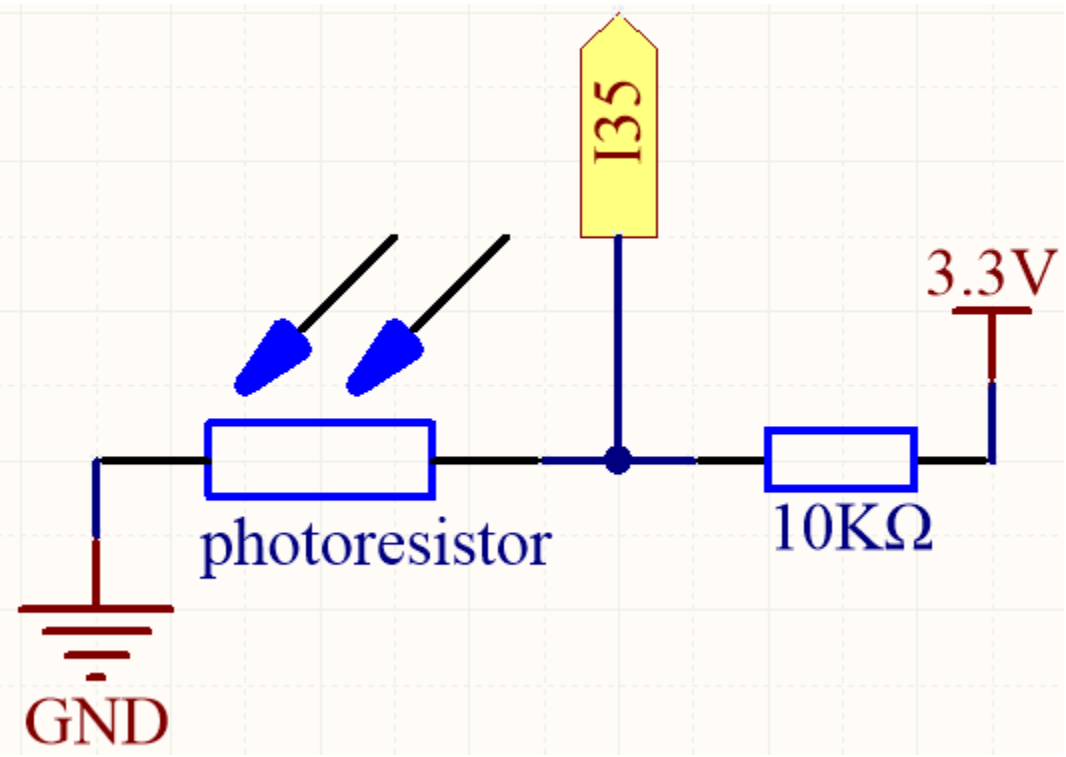
Verfügbare Pins	IO14, IO25, I35, I34, I39, I36

- **Strapping Pins**

  Die folgenden Pins sind Strapping-Pins, die den Startprozess des ESP32 während des Einschaltens oder Resets beeinflussen. Sobald der ESP32 jedoch erfolgreich gestartet ist, können sie als normale Pins verwendet werden.

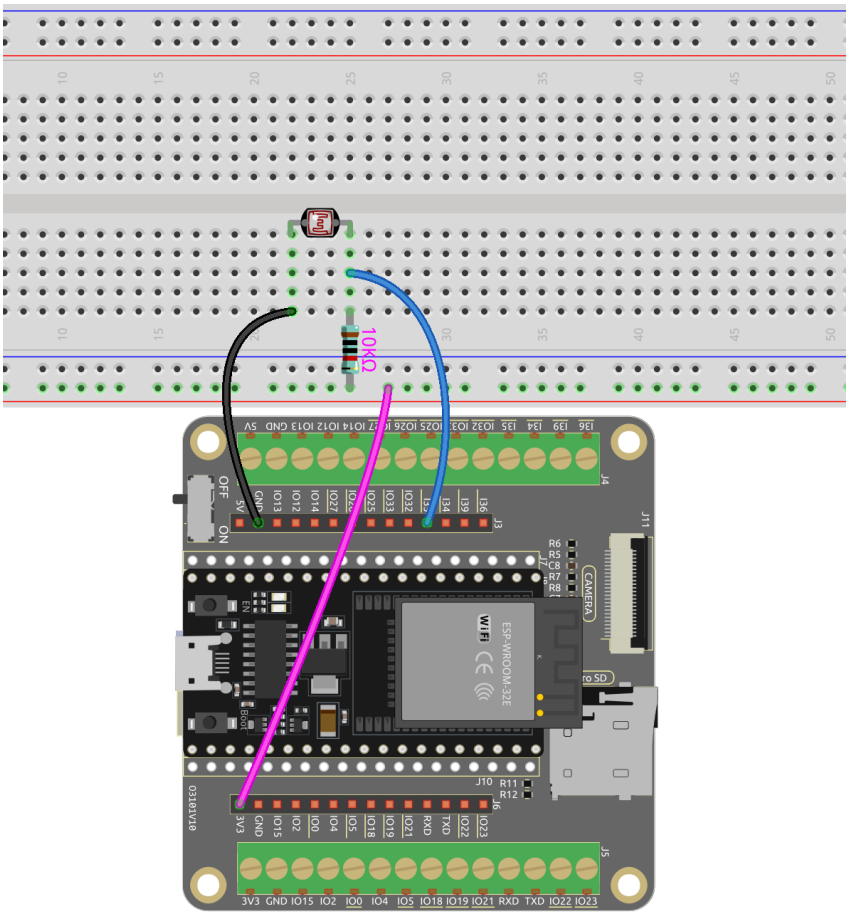
Strapping Pins	IO0, IO12

Schaltplan



Mit zunehmender Lichtintensität nimmt der Widerstand des lichtabhängigen Widerstands (LDR) ab, was zu einer Verringerung des auf I35 ausgelesenen Werts führt.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.7_feel_the_light.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.7_feel_the_light`.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, druckt der Serielle Monitor die Fotowiderstandswerte von 0 bis 4095 aus. Je stärker die aktuelle Umgebungshelligkeit, desto größer der Wert, der im seriellen Monitor angezeigt wird.

Bemerkung: Für den ESP32 liegt die Auflösung zwischen 9 und 12 und es wird die ADC-Hardwareauflösung geändert. Andernfalls wird der Wert verschoben.

Standardmäßig ist sie 12 Bit (Bereich von 0 bis 4096) für alle Chips außer ESP32S3, wo der Standard 13 Bit (Bereich von 0 bis 8192) ist.

Sie können `analogReadResolution(10);` zur `setup()`-Funktion hinzufügen, um eine andere Auflösung festzulegen, wie z.B. `20`.

2.24 5.8 Den Knopf Drehen

Ein Potentiometer ist ein dreipoliges Gerät, das häufig verwendet wird, um den Widerstand in einem Schaltkreis anzupassen. Es verfügt über einen Drehknopf oder einen Schieberegler, mit dem der Widerstandswert des Potentiometers verändert werden kann. In diesem Projekt werden wir es nutzen, um die Helligkeit einer LED zu steuern, ähnlich wie bei einer Schreibtischlampe im täglichen Leben. Durch Anpassen der Position des Potentiometers können wir den Widerstand im Schaltkreis verändern, wodurch der Stromfluss durch die LED reguliert und deren Helligkeit entsprechend angepasst wird. Dies ermöglicht es uns, ein anpassbares und verstellbares Beleuchtungserlebnis zu schaffen, ähnlich dem einer Schreibtischlampe.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
Potentiometer	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

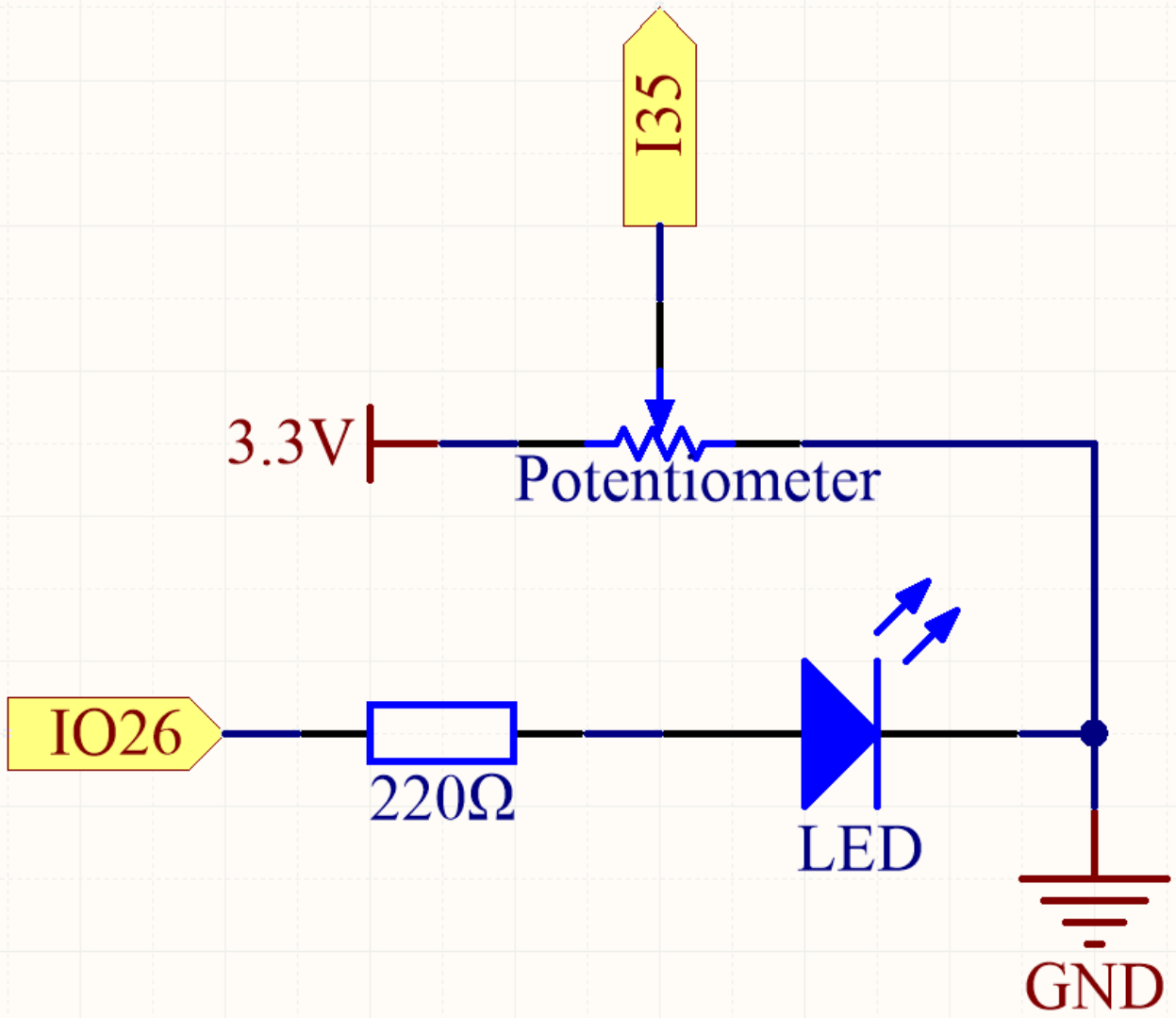
Verfügbare Pins	IO14, IO25, I35, I34, I39, I36

- **Strapping Pins**

  Die folgenden Pins sind Strapping-Pins, die den Startprozess des ESP32 während des Einschaltens oder Resets beeinflussen. Sobald der ESP32 jedoch erfolgreich gestartet ist, können sie als normale Pins verwendet werden.

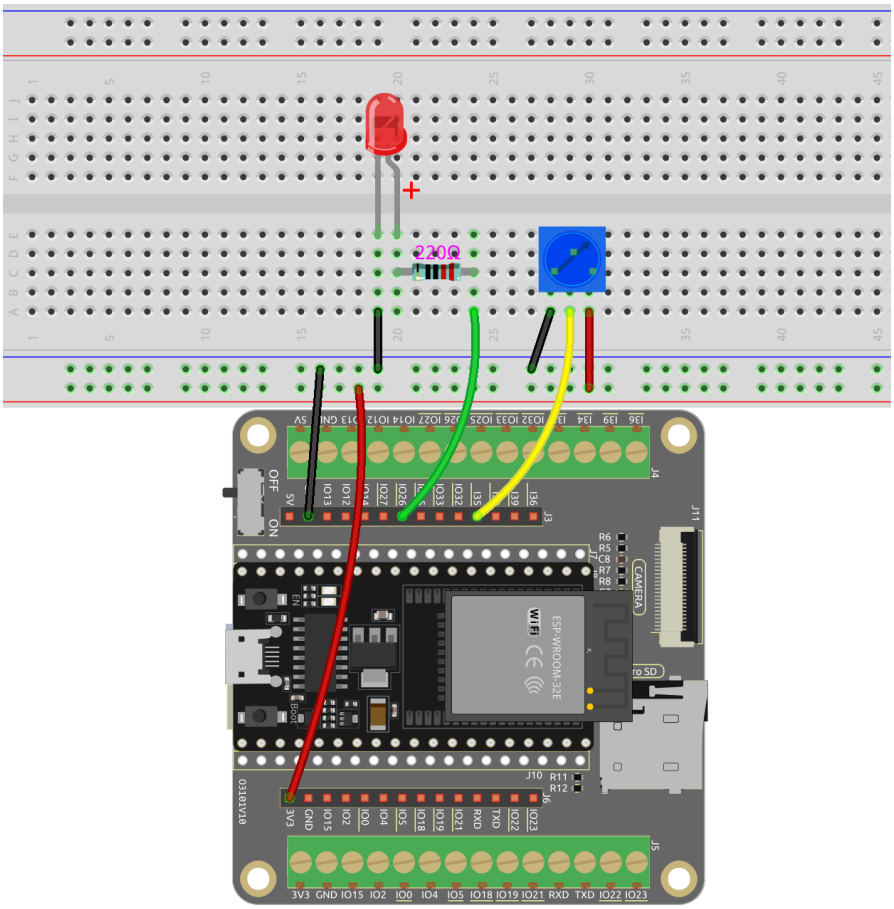
Strapping Pins	IO0, IO12

Schaltplan



Wenn Sie das Potentiometer drehen, ändert sich der Wert von I35. Durch Programmierung können Sie den Wert von I35 verwenden, um die Helligkeit der LED zu steuern. Daher ändert sich die Helligkeit der LED entsprechend, wenn Sie das Potentiometer drehen.

Verdrahtung



Code

Bemerkung:

- Die Datei `5.8_pot.ino` kann unter dem Pfad `esp32-starter-kit-main\c\codes\5.8_pot` geöffnet werden.
- Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Button.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, drehen Sie am Potentiometer und Sie werden sehen, wie sich die Helligkeit der LED entsprechend ändert. Gleichzeitig können Sie die analogen und Spannungswerte des Potentiometers im seriellen Monitor sehen.

Wie funktioniert das?

1. Definition der Konstanten für Pin-Verbindungen und PWM-Einstellungen.

```
const int potPin = 14; // Potentiometer connected to GPIO14
const int ledPin = 26; // LED connected to GPIO26

// PWM settings
const int freq = 5000; // PWM frequency
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
const int resolution = 12; // PWM resolution (bits)
const int channel = 0; // PWM channel
```

Hier ist die PWM-Auflösung auf 12 Bit eingestellt und der Bereich ist 0-4095.

2. Konfiguration des Systems in der Funktion `setup()`.

```
void setup() {
    Serial.begin(115200);

    // Configure PWM
    ledcSetup(channel, freq, resolution);
    ledcAttachPin(ledPin, channel);
}
```

- In der Funktion `setup()` wird die serielle Kommunikation mit einer Baudrate von 115200 gestartet.
- Die Funktion `ledcSetup()` wird aufgerufen, um den PWM-Kanal mit der angegebenen Frequenz und Auflösung einzurichten, und die Funktion `ledcAttachPin()` wird aufgerufen, um den angegebenen LED-Pin mit dem PWM-Kanal zu verknüpfen.

3. Hauptloop (wird wiederholt ausgeführt) in der Funktion `loop()`.

```
void loop() {

    int potValue = analogRead(potPin); // read the value of the␣
↪potentiometer
    uint32_t voltage_mV = analogReadMilliVolts(potPin); // Read the voltage␣
↪in millivolts

    ledcWrite(channel, potValue);

    Serial.print("Potentiometer Value: ");
    Serial.print(potValue);
    Serial.print(", Voltage: ");
    Serial.print(voltage_mV / 1000.0); // Convert millivolts to volts
    Serial.println(" V");

    delay(100);
}
```

- `uint32_t analogReadMilliVolts(uint8_t pin);`: Diese Funktion wird verwendet, um den ADC-Wert für einen gegebenen Pin/ADC-Kanal in Millivolt zu erhalten.
  - `pin` GPIO-Pin, um den analogen Wert zu lesen.

Der Potentiometerwert wird direkt als PWM-Tastverhältnis für die Steuerung der LED-Helligkeit über die Funktion `ledcWrite()` verwendet, da der Wertebereich ebenfalls von 0 bis 4095 reicht.

2.25 5.9 Bodenfeuchtigkeit messen

Dieser kapazitive Bodenfeuchtigkeitssensor unterscheidet sich von den meisten resistiven Sensoren auf dem Markt, indem er das Prinzip der kapazitiven Induktion zur Messung der Bodenfeuchtigkeit verwendet.

Durch das visuelle Ablesen der Werte vom Bodenfeuchtigkeitssensor können wir Informationen über den Feuchtigkeitsgehalt im Boden sammeln. Diese Informationen sind nützlich für verschiedene Anwendungen, wie automatische Bewässerungssysteme, Pflanzengesundheitsüberwachung oder Umweltsensorikprojekte.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv bequem, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Bodenfeuchtigkeitsmodul	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

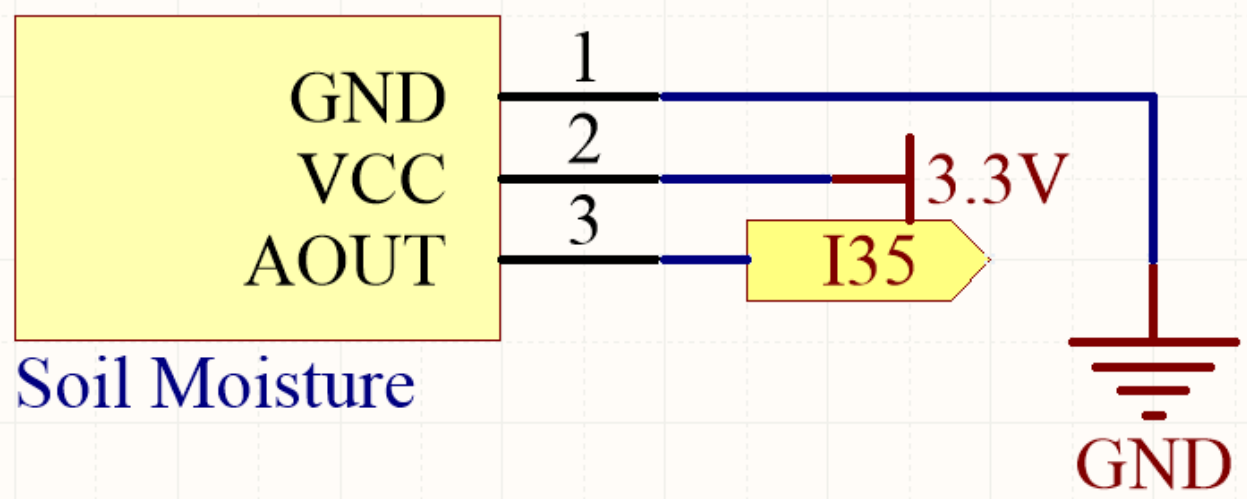
Verfügbare Pins	IO14, IO25, I35, I34, I39, I36

- **Strapping Pins**

  Die folgenden Pins sind Strapping-Pins, die den Startprozess des ESP32 während des Einschaltens oder Zurücksetzens beeinflussen. Sobald der ESP32 jedoch erfolgreich gestartet ist, können sie als reguläre Pins verwendet werden.

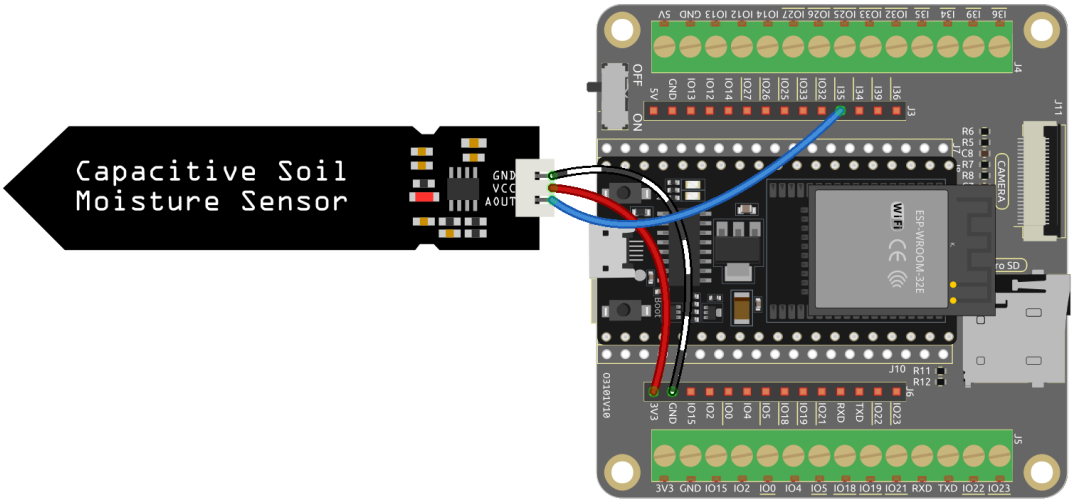
Strapping Pins	IO0, IO12

Schaltplan



Durch das Einstecken des Moduls in den Boden und das Bewässern wird der auf I35 gelesene Wert sinken.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.9_moisture.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.9_moisture`.
- Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Button.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Sobald der Code erfolgreich hochgeladen wurde, gibt der serielle Monitor den Bodenfeuchtigkeitswert aus.

Durch das Einstecken des Moduls in den Boden und das Bewässern wird der Wert des Bodenfeuchtigkeitssensors kleiner.

2.26 5.10 Thermometer

Ein Thermistor ist ein Temperatursensor, der eine starke Temperaturabhängigkeit aufweist und in zwei Typen eingeteilt werden kann: Negative Temperature Coefficient (NTC) und Positive Temperature Coefficient (PTC). Der Widerstand eines NTC-Thermistors verringert sich mit steigender Temperatur, während der Widerstand eines PTC-Thermistors mit steigender Temperatur zunimmt.

In diesem Projekt werden wir einen NTC-Thermistor verwenden. Durch das Verbinden des NTC-Thermistors mit einem analogen Eingangspin des ESP32-Mikrocontrollers können wir seinen Widerstand messen, der direkt proportional zur Temperatur ist.

Indem wir den NTC-Thermistor einbeziehen und die notwendigen Berechnungen durchführen, können wir die Temperatur genau messen und auf dem I2C LCD1602-Modul anzeigen. Dieses Projekt ermöglicht eine Echtzeit-Temperaturüberwachung und bietet eine visuelle Schnittstelle zur Temperaturanzeige.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Thermistor	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

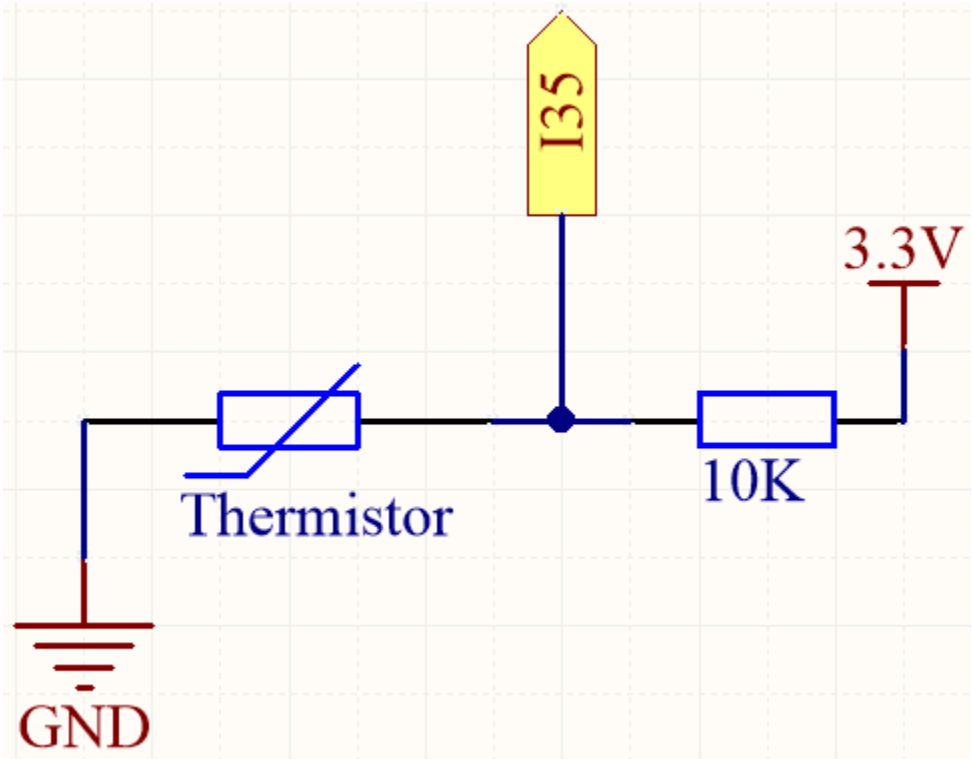
Verfügbare Pins	IO14, IO25, I35, I34, I39, I36

- **Strapping Pins**

  Die folgenden Pins sind Strapping-Pins, die den Startprozess des ESP32 während des Einschaltens oder Zurücksetzens beeinflussen. Sobald der ESP32 jedoch erfolgreich gestartet ist, können sie als reguläre Pins verwendet werden.

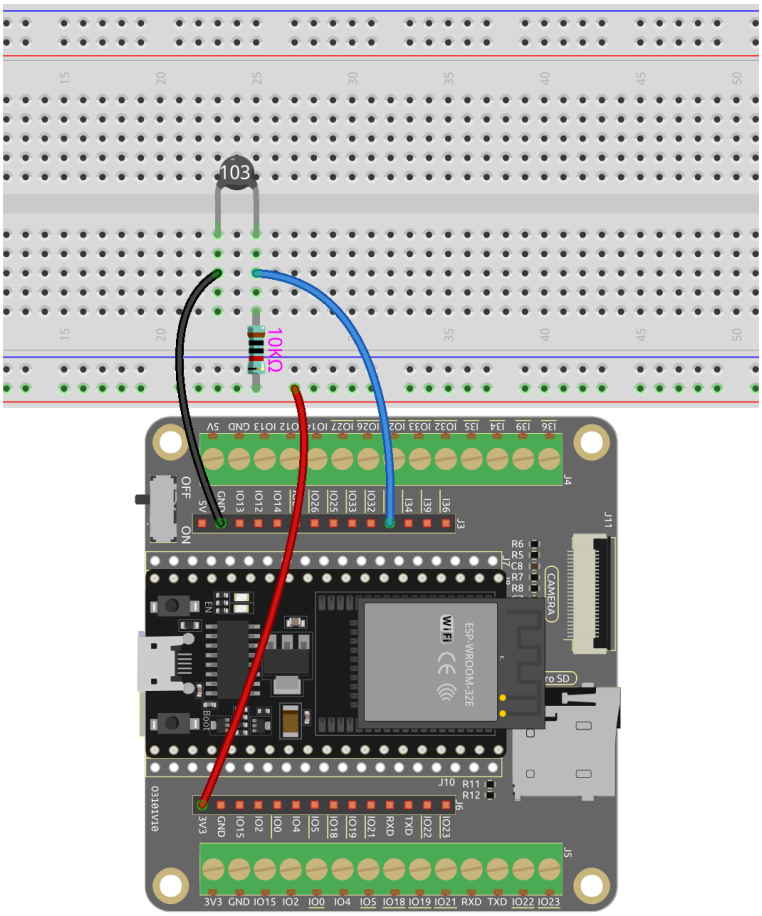
Strapping Pins	IO0, IO12

Schaltplan



Wenn die Temperatur steigt, verringert sich der Widerstand des Thermistors, was dazu führt, dass der auf I35 gelesene Wert sinkt. Zusätzlich kann durch die Verwendung einer Formel der analoge Wert in eine Temperatur umgerechnet und dann ausgegeben werden.

Verdrahtung



Bemerkung:

- Der Thermistor ist schwarz und mit 103 gekennzeichnet.
- Der Farbring des 10K-Ohm-Widerstands ist rot, schwarz, schwarz, rot und braun.

Code

Bemerkung:

- Öffnen Sie die Datei `5.10_thermistor.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.10_thermistor`.
- Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Button.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, gibt der Serielle Monitor die Temperaturen in Celsius und Fahrenheit aus.

Wie funktioniert das?

Jeder Thermistor hat einen normalen Widerstand. Hier beträgt er 10k Ohm, gemessen bei 25 Grad Celsius.

Wenn die Temperatur steigt, verringert sich der Widerstand des Thermistors. Dann werden die Spannungsdaten durch den A/D-Adapter in digitale Daten umgewandelt.

Die Temperatur in Celsius oder Fahrenheit wird über die Programmierung ausgegeben.

Hier ist der Zusammenhang zwischen Widerstand und Temperatur:

RT =RN expB(1/TK - 1/TN)

- **RT** ist der Widerstand des NTC-Thermistors bei der Temperatur **TK**.
- **RN** ist der Widerstand des NTC-Thermistors unter der Nenntemperatur TN. Hier beträgt der numerische Wert von RN 10k.
- **TK** ist eine Kelvintemperatur und die Einheit ist K. Hier beträgt der numerische Wert von **TK** `373.15 + Grad Celsius`.
- **TN** ist eine Nenntemperatur in Kelvin; die Einheit ist auch K. Hier beträgt der numerische Wert von TN `373.15+25`.
- Und **B(beta)**, die Materialkonstante des NTC-Thermistors, wird auch als Wärmeempfindlichkeitsindex bezeichnet und hat einen numerischen Wert `4950`.
- **exp** ist die Abkürzung von Exponentialfunktion, und die Basiszahl `e` ist eine natürliche Zahl und beträgt ungefähr 2,7.

Umrechnen dieser Formel `TK=1/(ln(RT/RN)/B+1/TN)` um die Kelvintemperatur zu erhalten, die minus 273,15 Grad Celsius entspricht.

Diese Beziehung ist eine empirische Formel. Sie ist nur genau, wenn die Temperatur und der Widerstand innerhalb des wirksamen Bereichs liegen.

Mehr erfahren

Sie können auch die berechneten Temperaturen in Celsius und Fahrenheit auf dem I2C LCD1602 anzeigen.

Bemerkung:

- Sie können die Datei `5.10_thermistor_lcd.ino` unter dem Pfad `euler-kit/arduino/5.10_thermistor_lcd` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Button.
- *„Unbekanntes COMxx“ wird immer angezeigt?*
- Hier wird die `LiquidCrystal I2C`-Bibliothek verwendet, die Sie aus dem **Library Manager** installieren können.

2.27 5.11 Den Joystick umschalten

Wenn Sie viele Videospiele spielen, sollten Sie mit dem Joystick sehr vertraut sein. Er wird üblicherweise verwendet, um die Spielfigur zu bewegen, den Bildschirm zu drehen usw.

Das Prinzip hinter der Fähigkeit des Joysticks, dem Computer unsere Aktionen mitzuteilen, ist sehr einfach. Man kann sich diesen als aus zwei Potentiometern bestehend vorstellen, die senkrecht zueinander stehen. Diese beiden Potentiometer messen den analogen Wert des Joysticks vertikal und horizontal, was in einem Wert (x,y) in einem ebenen rechtwinkligen Koordinatensystem resultiert.

Der Joystick dieses Kits hat auch einen digitalen Eingang, der aktiviert wird, wenn der Joystick gedrückt wird.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

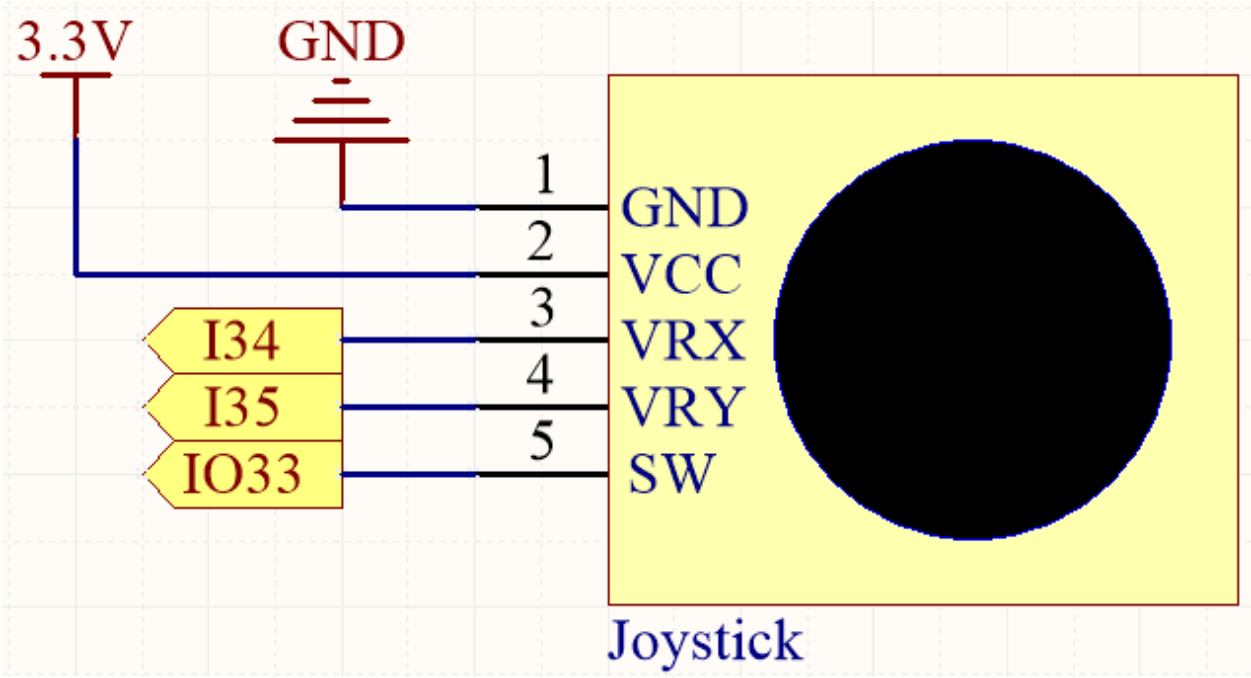
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Joystick-Modul	

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Für analogen Eingang	IO14, IO25, I35, I34, I39, I36
Für digitalen Eingang	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan

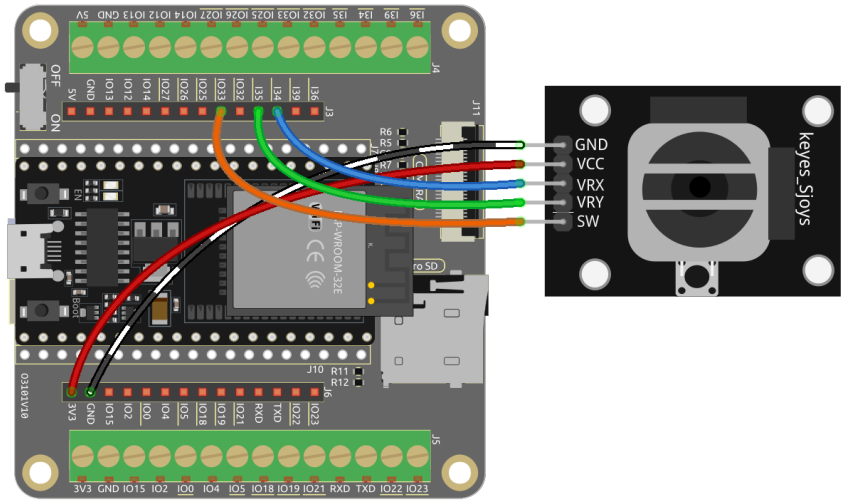


Der SW (Z-Achse)-Pin ist mit IO33 verbunden, der einen eingebauten 4,7K-Pull-up-Widerstand hat. Daher wird, wenn der SW-Knopf nicht gedrückt ist, ein hoher Pegel ausgegeben. Wenn der Knopf gedrückt wird, wird ein niedriger Pegel

ausgegeben.

I34 und I35 ändern ihre Werte, wenn Sie den Joystick bedienen. Der Wertebereich reicht von 0 bis 4095.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.11_joystick.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.11_joystick`.
- Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Button.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Öffnen Sie den seriellen Monitor, nachdem der Code erfolgreich hochgeladen wurde, um die x-, y- und z-Werte des Joysticks zu sehen.

- Die x- und y-Achsenwerte sind analoge Werte, die von 0 bis 4095 variieren.
- Die Z-Achse ist ein digitaler Wert mit einem Status von 1 oder 0 (wenn gedrückt, ist er 0).

2.28 5.12 Entfernung messen

Das Ultraschallmodul wird zur Entfernungsmessung oder Objekterkennung verwendet. In diesem Projekt programmieren wir das Modul, um Hindernisentfernungen zu ermitteln. Indem wir Ultraschallimpulse senden und die Zeit messen, die sie zum Zurückprallen benötigen, können wir Entfernungen berechnen. Dies ermöglicht es uns, distanzbasierte Aktionen oder Hindernisvermeidungsverhalten zu implementieren.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Ultraschall-Modul	

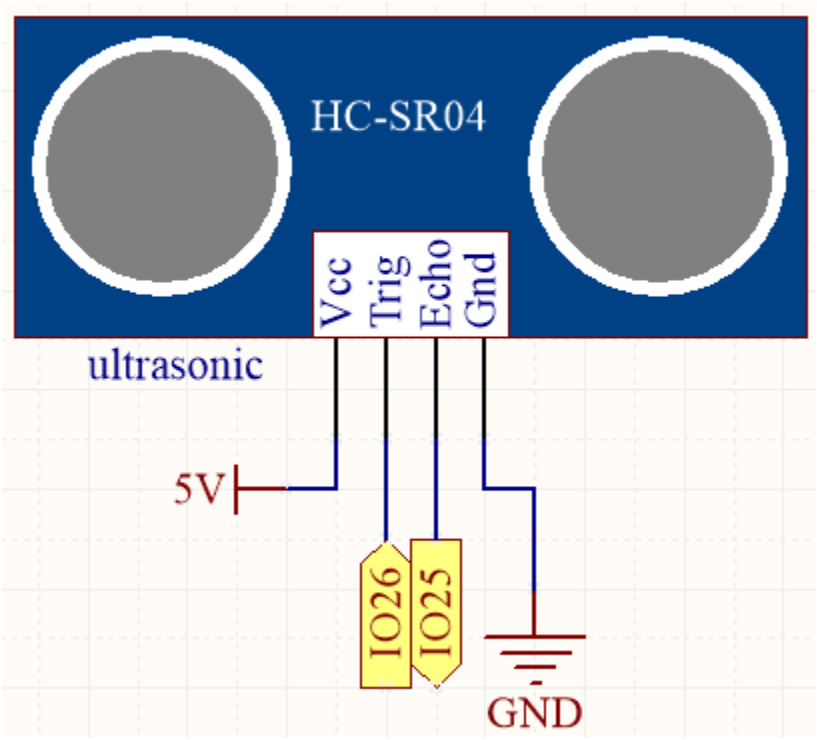
Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Für Eingang	IO13, IO14, IO27, IO26, IO25, IO33, IO32, I35, I34, I39, I36
Für Ausgang	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

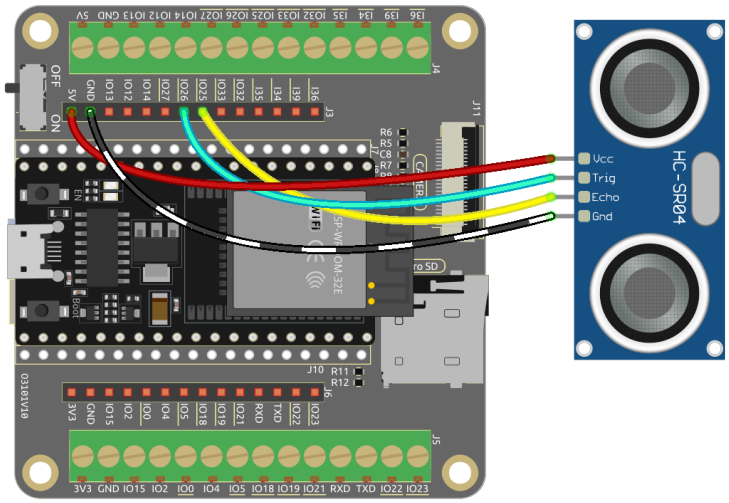
Schaltplan



Der ESP32 sendet alle 10 Sekunden einen Satz von Rechteckwellensignalen an den Trig-Pin des Ultraschallsensors. Dies veranlasst den Ultraschallsensor, ein 40kHz Ultraschallsignal nach außen zu senden. Wenn sich ein Hindernis vorne befindet, werden die Ultraschallwellen zurückreflektiert.

Durch die Aufzeichnung der Zeit, die vom Senden bis zum Empfangen des Signals vergeht, diese durch 2 teilen und mit der Schallgeschwindigkeit multiplizieren, können Sie die Entfernung zum Hindernis bestimmen.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.12_ultrasonic.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.12_ultrasonic`.
- Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Button.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Nachdem der Code erfolgreich hochgeladen wurde, gibt der serielle Monitor die Entfernung zwischen dem Ultraschallsensor und dem Hindernis vorne aus.

Wie funktioniert das?

Über die Anwendung des Ultraschallsensors können wir direkt die Unterfunktion überprüfen.

```
float readSensorData(){// ...}
```

- Der `trigPin` des Ultraschallmoduls sendet alle 2us ein 10us-Rechteckwellensignal.

```
// Trigger a low signal before sending a high signal
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Send a 10-microsecond high signal to the trigPin
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
// Return to low signal
digitalWrite(trigPin, LOW);
```

- Der `echoPin` empfängt ein Hochpegeleingangssignal, wenn sich ein Hindernis im Bereich befindet, und verwendet die Funktion `pulseIn()`, um die Zeit vom Senden bis zum Empfangen aufzuzeichnen.

```
unsigned long microsecond = pulseIn(echoPin, HIGH);
```

- Die Schallgeschwindigkeit beträgt 340 Meter pro Sekunde, was 29 Mikrosekunden pro Zentimeter entspricht. Indem wir die Zeit messen, die eine Rechteckwelle benötigt, um zu einem Hindernis zu gelangen und zurückzukehren, können wir die zurückgelegte Entfernung berechnen, indem wir die Gesamtzeit durch 2 teilen. Dies ergibt die Entfernung des Hindernisses von der Schallquelle.

  ```
  float distance = microsecond / 29.00 / 2;
  ```

Beachten Sie, dass der Ultraschallsensor das Programm während der Arbeit anhält, was bei der Erstellung komplexer Projekte zu Verzögerungen führen kann.

2.29 5.13 Temperatur - Feuchtigkeit

Der DHT11 ist ein häufig für Umweltmessungen verwendeter Temperatur- und Feuchtigkeitssensor. Es handelt sich um einen digitalen Sensor, der mit einem Mikrocontroller kommuniziert, um Temperatur- und Feuchtigkeitswerte bereitzustellen.

In diesem Projekt werden wir den DHT11-Sensor auslesen und die von ihm erfassten Temperatur- und Feuchtigkeitswerte ausgeben.

Durch das Auslesen der vom Sensor bereitgestellten Daten können wir die aktuellen Temperatur- und Feuchtigkeitswerte in der Umgebung ermitteln. Diese Werte können für die Echtzeitüberwachung von Umweltbedingungen, Wetterbeobachtungen, Raumklimakontrolle, Feuchtigkeitsberichte und mehr verwendet werden.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

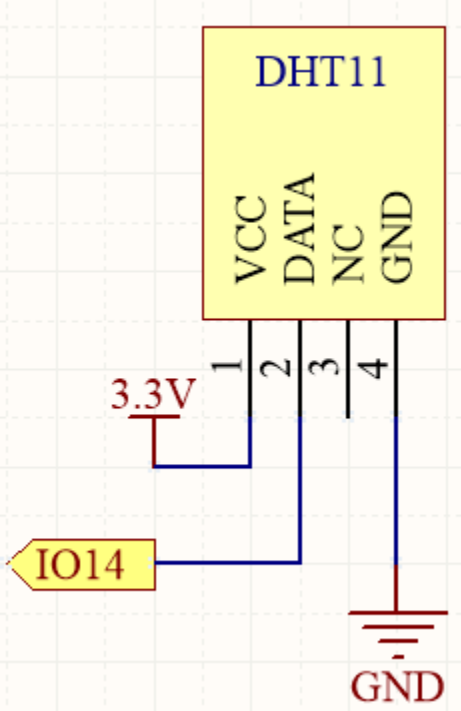
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
DHT11 Feuchtigkeits- und Temperatursensor	

- **Verfügbare Pins**

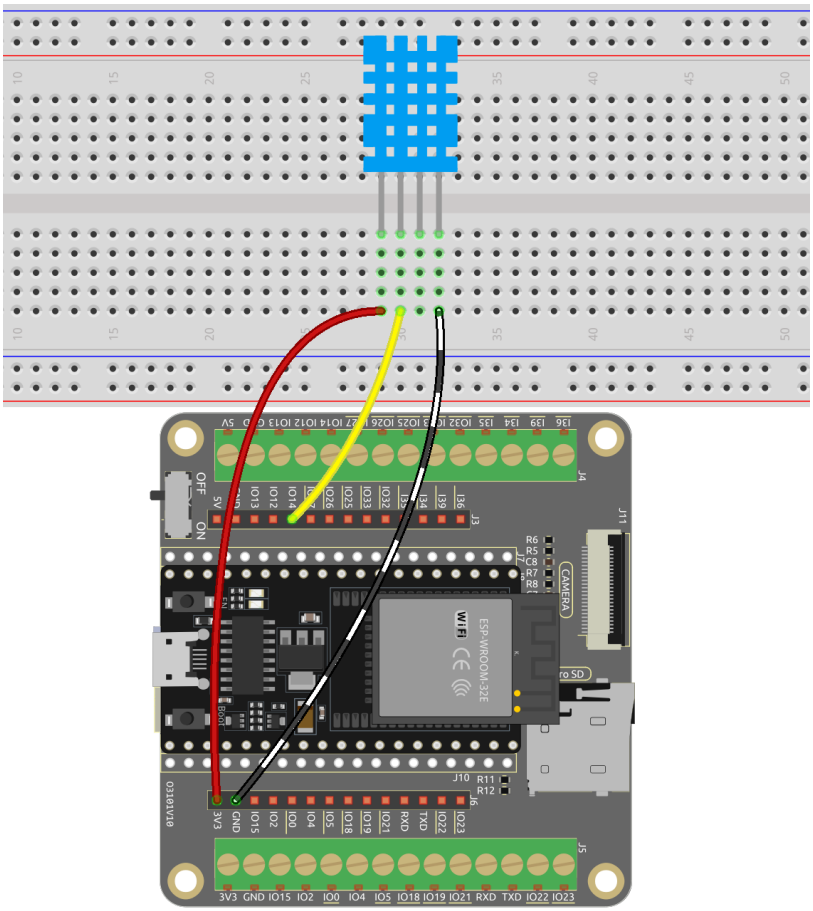
  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



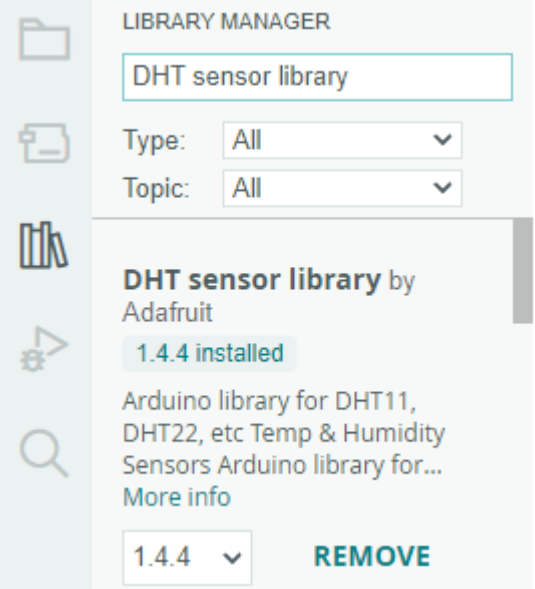
Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.13_dht11.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.13_dht11`.
- Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Button.
- *„Unbekanntes COMxx“ wird immer angezeigt?*
- Hier wird die Bibliothek „DHT-Sensorbibliothek“ verwendet, die Sie aus dem **Library Manager** installieren können.



Nachdem der Code erfolgreich hochgeladen wurde, wird der Serielle Monitor kontinuierlich die Temperatur und Feuchtigkeit ausgeben, und während das Programm stetig läuft, werden diese beiden Werte immer genauer.

Wie funktioniert das?

1. Beinhaltet die `DHT.h`-Bibliothek, die Funktionen zur Interaktion mit den DHT-Sensoren bereitstellt. Anschließend den Pin und Typ für den DHT-Sensor festlegen.

```
#include "DHT.h"

#define DHTPIN 14  // Set the pin connected to the DHT11 data pin
#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE);
```

2. Initialisiert die serielle Kommunikation mit einer Baudrate von 115200 und initialisiert den DHT-Sensor.

```
void setup() {
    Serial.begin(115200);
    Serial.println("DHT11 test!");
    dht.begin();
}
```

3. In der Funktion `loop()`, die Temperatur- und Feuchtigkeitswerte vom DHT11-Sensor lesen und sie auf dem seriellen Monitor ausgeben.

```
void loop() {
    // Wait a few seconds between measurements.
    delay(2000);

    // Reading temperature or humidity takes about 250 milliseconds!
    // Sensor readings may also be up to 2 seconds 'old' (it's a very slow sensor)
    float humidity = dht.readHumidity();
    // Read temperature as Celsius (the default)
    float temperture = dht.readTemperature();
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    // Check if any reads failed and exit early (to try again).
    if (isnan(humidity) || isnan(temperture)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }
    // Print the humidity and temperature
    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.print(" %\t");
    Serial.print("Temperature: ");
    Serial.print(temperture);
    Serial.println(" *C");
}
```

- Die Funktion `dht.readHumidity()` wird aufgerufen, um den Feuchtigkeitswert vom DHT-Sensor zu lesen.
- Die Funktion `dht.readTemperature()` wird aufgerufen, um den Temperaturwert vom DHT-Sensor zu lesen.
- Die Funktion `isnan()` wird verwendet, um zu überprüfen, ob die Messwerte gültig sind. Wenn entweder der Feuchtigkeits- oder der Temperaturwert NaN (keine Zahl) ist, deutet dies auf eine fehlgeschlagene Messung vom Sensor hin, und eine Fehlermeldung wird ausgegeben.

Mehr erfahren

Sie können auch die Temperatur und Feuchtigkeit auf dem I2C LCD1602 anzeigen.

Bemerkung:

- Sie können die Datei `5.10_thermistor_lcd.ino` unter dem Pfad `euler-kit/arduino/5.10_thermistor_lcd` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Button.
- *„Unbekanntes COMxx“ wird immer angezeigt?*
- Die Bibliotheken `LiquidCrystal_I2C` und `DHT-Sensorbibliothek` werden hier verwendet, Sie können sie aus dem **Library Manager** installieren.

2.30 5.14 IR-Empfänger

Ein Infrarotempfänger ist eine Komponente, die Infrarotsignale empfängt und in der Lage ist, unabhängig Signale zu erkennen und auszugeben, die mit dem TTL-Pegel kompatibel sind. Er ist ähnlich groß wie ein regulärer kunststoffverpackter Transistor und wird häufig in verschiedenen Anwendungen wie Infrarotfernsteuerung und Infrarotübertragung verwendet.

In diesem Projekt werden wir einen Infrarotempfänger verwenden, um Signale von einer Fernbedienung zu erkennen. Wenn ein Knopf auf der Fernbedienung gedrückt wird und der Infrarotempfänger das entsprechende Signal empfängt, kann er das Signal dekodieren, um zu bestimmen, welcher Knopf gedrückt wurde. Durch das Dekodieren des empfangenen Signals können wir den spezifischen Schlüssel oder Befehl identifizieren, der damit verbunden ist.

Der Infrarotempfänger ermöglicht es uns, Fernsteuerungsfunktionalität in unser Projekt zu integrieren, sodass wir Geräte mit Infrarotsignalen interagieren und steuern können.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

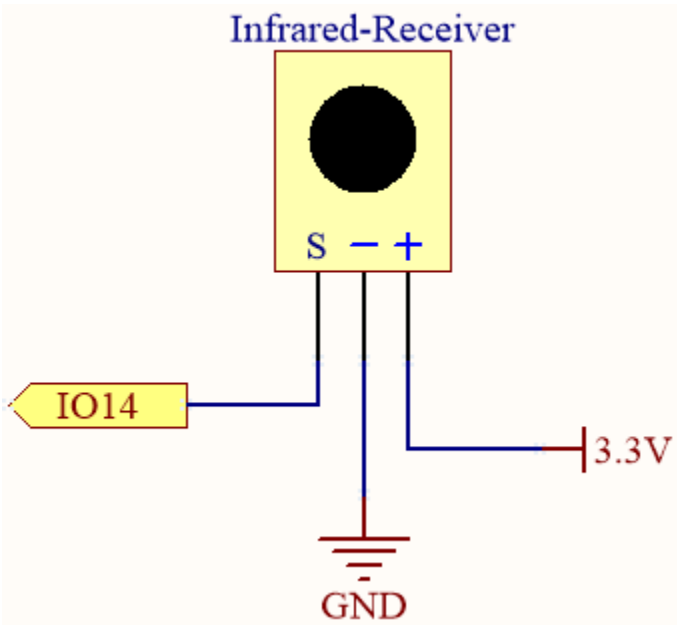
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
IR-Empfänger	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

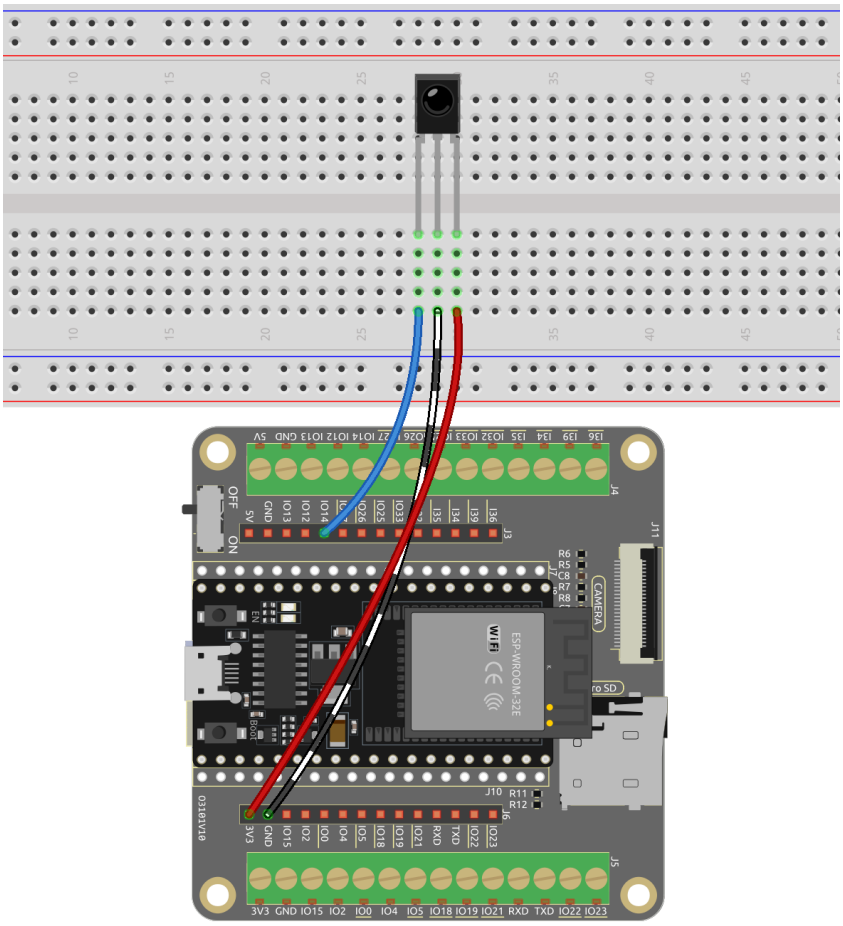
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO15, IO0, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Wenn Sie einen Knopf auf der Fernbedienung drücken, erkennt der Infrarotempfänger das Signal, und Sie können eine Infrarotbibliothek verwenden, um es zu dekodieren. Dieser Dekodierungsprozess ermöglicht es Ihnen, den Schlüsselwert zu erhalten, der mit dem Tastendruck verbunden ist.

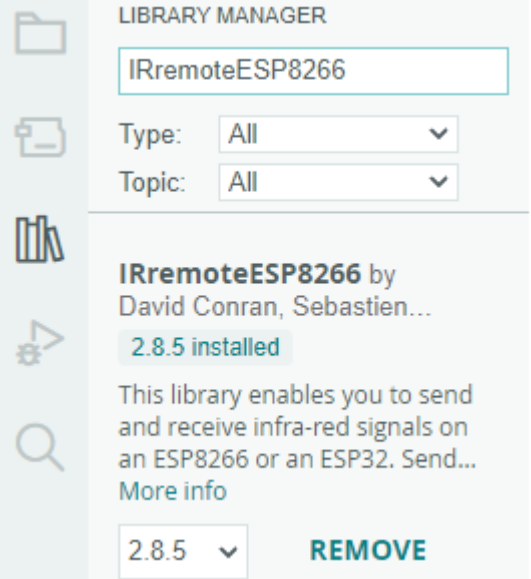
Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.14_ir_receiver.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\5.14_ir_receiver`.
- Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Button.
- *„Unbekanntes COMxx“ wird immer angezeigt?*
- Hier wird die Bibliothek `IRremoteESP8266` verwendet, die Sie aus dem **Library Manager** installieren können.



Nachdem der Code erfolgreich hochgeladen wurde, drücken Sie die verschiedenen Tasten auf der Fernbedienung und Sie werden die Namen dieser Tasten im seriellen Monitor sehen.

Bemerkung:

- Die Bibliothek `IRremoteESP8266` umfasst Implementierungen für viele verschiedene Infrarotprotokolle und -geräte, daher ist die Größe der Bibliothek relativ groß. Wenn der Compiler mehr Code verarbeiten muss, wird auch die Kompilierungszeit entsprechend zunehmen. Bitte haben Sie Geduld und warten Sie, bis die Kompilierung abgeschlossen ist.
- Die neue Fernbedienung verfügt über eine Plastiklasche am Ende, um die Batterie im Inneren zu isolieren. Um die Fernbedienung bei Gebrauch mit Strom zu versorgen, entfernen Sie einfach dieses Plastikstück.

Wie funktioniert das?

1. Dieser Code verwendet die Bibliothek `IRremoteESP8266`, um Infrarot (IR) Signale mit einem IR-Empfängermodul zu empfangen.

```
#include <IRremoteESP8266.h>
#include <IRrecv.h>

// Define the IR receiver pin
const uint16_t IR_RECEIVE_PIN = 14;

// Create an IRrecv object
IRrecv irrecv(IR_RECEIVE_PIN);

// Create a decode_results object
decode_results results;
```

2. In der Funktion `setup()` wird die serielle Kommunikation mit einer Baudrate von 115200 gestartet und der IR-Empfänger mit `irrecv.enableIRIn()` aktiviert.

```
void setup() {
    // Start serial communication
    Serial.begin(115200);

    // Start the IR receiver
    irrecv.enableIRIn();
}
```

3. Wenn Sie eine Taste auf der Fernbedienung drücken, wird der Tastenname im seriellen Monitor ausgegeben, falls er vom IR-Empfänger empfangen wird.

```
void loop() {
    // If an IR signal is received
    if (irrecv.decode(&results)) {
        String key = decodeKeyValue(results.value);
        if (key != "ERROR") {
            // Print the value of the signal to the serial monitor
            Serial.println(key);
        }
        irrecv.resume(); // Continue to receive the next signal
    }
}
```

   - Überprüfen Sie zunächst, ob ein IR-Signal mit der Funktion `irrecv.decode()` empfangen wurde.
   - Wenn ein Signal empfangen wird, rufen Sie die Funktion `decodeKeyValue()` auf, um den Wert des Signals zu dekodieren.
   - Wenn das Signal erfolgreich dekodiert wird, wird der dekodierte Wert mit `Serial.println()` auf dem seriellen Monitor ausgegeben.
   - Schließlich wird `irrecv.resume()` aufgerufen, um das nächste Signal weiterhin zu empfangen.

4. Die Funktion `decodeKeyValue()` nimmt den dekodierten Wert des IR-Signals als Argument und gibt einen String zurück, der den auf der Fernbedienung gedrückten Schlüssel repräsentiert.

```
String decodeKeyValue(long result)
{
    switch(result){
        case 0xFF6897:
        return "0";
        case 0xFF30CF:
        return "1";
        case 0xFF18E7:
        return "2";
        case 0xFF7A85:
        ...
```

   - Die Funktion verwendet eine switch-Anweisung, um den dekodierten Wert mit dem entsprechenden Schlüssel abzugleichen und gibt die String-Darstellung des Schlüssels zurück.
   - Wenn der dekodierte Wert keinem bekannten Schlüssel entspricht, gibt die Funktion den String „ERROR“ zurück.

6. Lustige Projekte

2.31 6.1 Fruchtpiano

Haben Sie jemals davon geträumt, Klavier zu spielen, konnten sich aber keines leisten? Oder möchten Sie einfach nur Spaß haben und ein Fruchtpiano basteln? Dann ist dieses Projekt genau das Richtige für Sie!

Mit nur wenigen Berührungssensoren auf dem ESP32-Board können Sie jetzt Ihre Lieblingsmelodien spielen und das Klavierspielen genießen, ohne viel Geld ausgeben zu müssen.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Widerstand	
Summer	
Transistor	

Über die Touch-Pins

Der ESP32-Mikrocontroller verfügt über eine integrierte Touchsensorfunktionalität, die es Ihnen ermöglicht, bestimmte Pins auf dem Board als berührungsempfindliche Eingänge zu verwenden. Der Touchsensor funktioniert, indem er Veränderungen in der Kapazität an den Touch-Pins misst, die durch die elektrischen Eigenschaften des menschlichen Körpers verursacht werden.

Hier sind einige wichtige Merkmale des Touchsensors am ESP32:

- **Anzahl der Touch-Pins**

  Der ESP32 hat bis zu 10 Touch-Pins, abhängig vom spezifischen Board. Die Touch-Pins sind typischerweise mit einem „T“ gefolgt von einer Zahl gekennzeichnet.

  - GPIO4: TOUCH0
  - GPIO0TOUCH1
  - GPIO2: TOUCH2
  - GPIO15: TOUCH3
  - GPIO13: TOUCH4
  - GPIO12: TOUCH5
  - GPIO14: TOUCH6

  - GPIO27: TOUCH7
  - GPIO33: TOUCH8
  - GPIO32: TOUCH9

Bemerkung: Die Pins GPIO0 und GPIO2 werden zum Booten und Flashen der Firmware auf dem ESP32 verwendet. Diese Pins sind auch mit der integrierten LED und dem Knopf verbunden. Daher wird generell nicht empfohlen, diese Pins für andere Zwecke zu verwenden, da dies die normale Funktion des Boards stören könnte.

- **Empfindlichkeit**

  Der Touchsensor auf dem ESP32 ist sehr empfindlich und kann sogar kleine Änderungen in der Kapazität erkennen. Die Empfindlichkeit kann über Softwareeinstellungen angepasst werden.

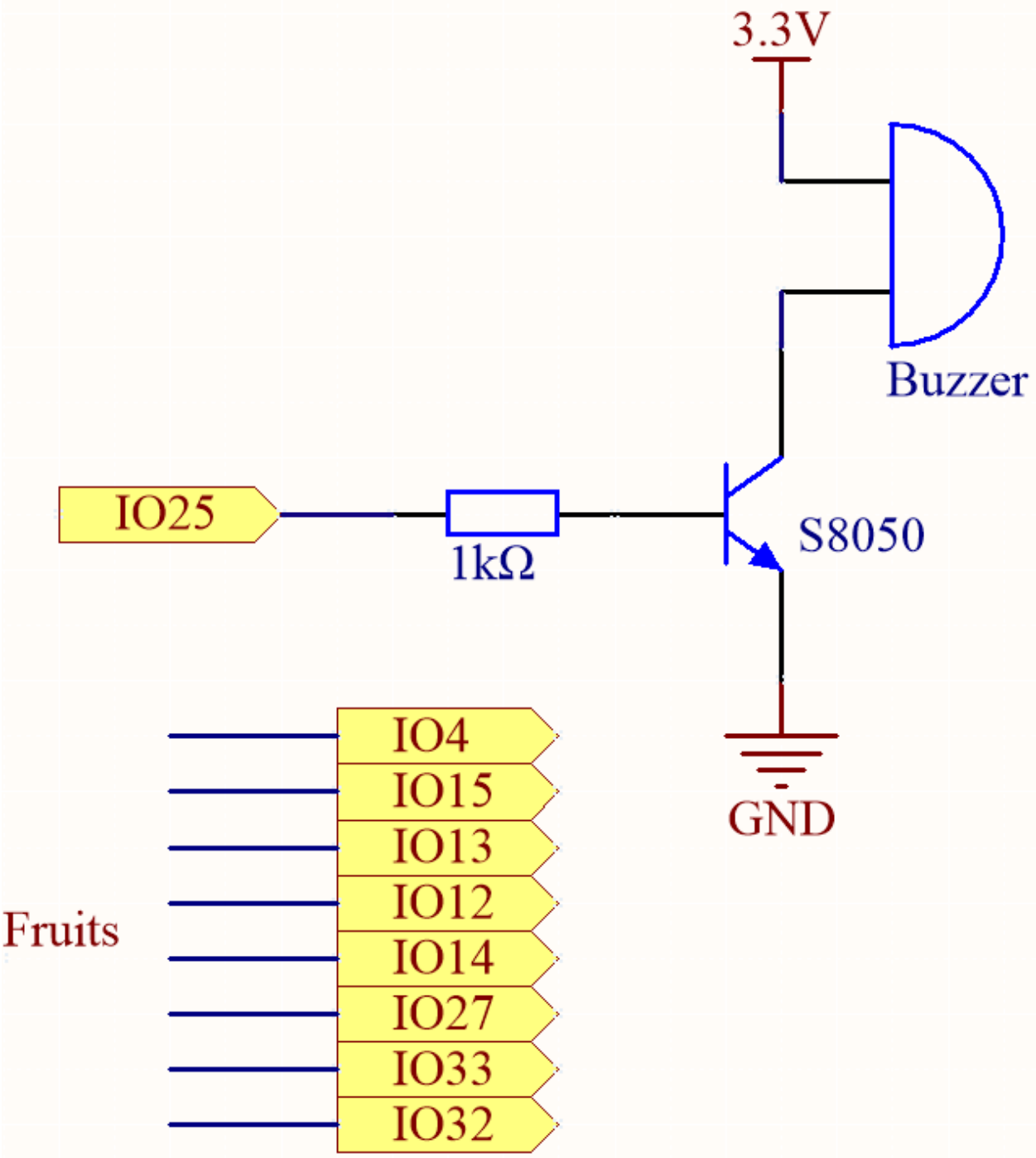
- **ESD-Schutz**

  Die Touch-Pins auf dem ESP32 verfügen über einen eingebauten ESD (Elektrostatische Entladung)-Schutz, der hilft, Schäden am Board durch statische Elektrizität zu verhindern.

- **Multitouch**

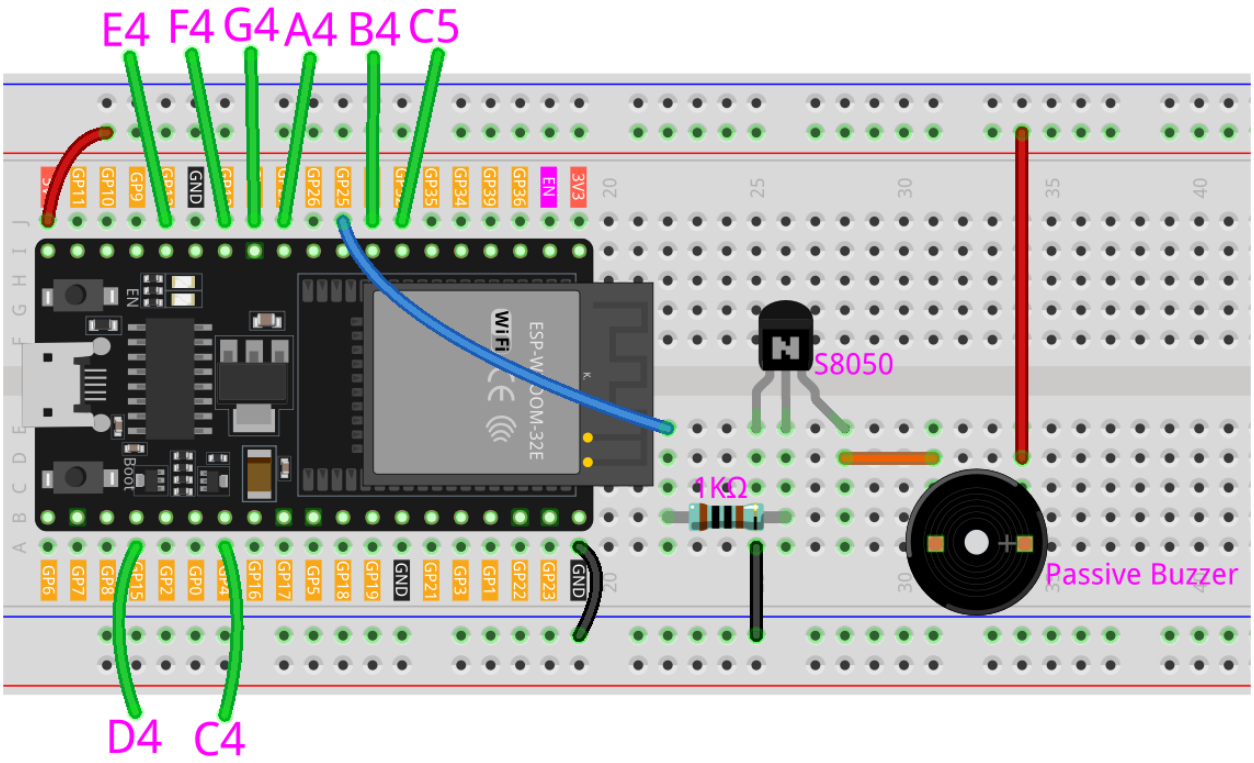
  Der Touchsensor auf dem ESP32 unterstützt Multitouch, was bedeutet, dass Sie mehrere Berührungsereignisse gleichzeitig erkennen können.

Schaltplan



Die Idee hinter diesem Projekt besteht darin, Touchsensoren zu verwenden, um zu erkennen, wenn ein Benutzer einen bestimmten Pin berührt. Jeder Touch-Pin ist mit einer bestimmten Note verbunden, und wenn der Benutzer einen Pin berührt, wird die entsprechende Note auf dem passiven Summer gespielt. Das Ergebnis ist eine einfache und erschwingliche Möglichkeit, das Klavierspielen zu genießen.

Verdrahtung



In diesem Projekt müssen Sie das ESP32 WROOM 32E von der Erweiterungsplatine entfernen und dann in das Steckbrett einsetzen. Dies liegt daran, dass einige Pins auf der Erweiterungsplatine mit Widerständen verbunden sind, was die Kapazität der Pins beeinflusst.

Code

Bemerkung:

- Sie können die Datei `6.1_fruit_piano.ino` direkt unter dem Pfad `esp32-starter-kit-main\c\codes\6.1_fruit_piano` öffnen.
- Oder kopieren Sie diesen Code in die Arduino IDE.

Sie können Früchte mit diesen ESP32-Pins verbinden: 4, 15, 13, 12, 14, 27, 33, 32.

Wenn das Skript läuft, werden durch Berühren dieser Früchte die Noten C, D, E, F, G, A, B und C5 gespielt.

Wie funktioniert das?

- `touchRead(uint8_t pin);`

  Diese Funktion erhält die Daten des Touchsensors. Jeder Touchsensor hat einen Zähler, der die Anzahl der Lade-/Entladezyklen zählt. Wenn das Pad **touched** wird, ändert sich der Wert im Zähler aufgrund der größeren äquivalenten Kapazität. Die Änderung der Daten bestimmt, ob das Pad berührt wurde oder nicht.

  – `pin` GPIO-Pin, um TOUCH-Wert zu lesen

  Diese Funktion gibt einen Wert zwischen 0 und 4095 zurück, wobei ein niedrigerer Wert eine stärkere Berührungseingabe anzeigt.

Bemerkung: `threshold` muss basierend auf der Leitfähigkeit verschiedener Früchte angepasst werden.

Sie können das Skript zuerst ausführen, um die von der Shell gedruckten Werte zu sehen.

```
0: 60
1: 62
2: 71
3: 74
4: 73
5: 78
6: 80
7: 82
```

Nach dem Berühren der Früchte an den Pins 12, 14 und 27 sehen die gedruckten Werte wie folgt aus. Daher habe ich den `threshold` auf 30 gesetzt, was bedeutet, dass, wenn ein Wert unter 30 erkannt wird, er als berührt gilt und der Summer verschiedene Noten abgibt.

```
0: 60
1: 62
2: 71
3: 9
4: 12
5: 14
6: 75
7: 78
```

2.32 6.2 Fließendes Licht

Haben Sie schon einmal daran gedacht, Ihrer Wohnfläche ein unterhaltsames und interaktives Element hinzuzufügen? Dieses Projekt beinhaltet die Erstellung eines fließenden Lichts mit einem WS2812 LED-Streifen und einem Hindernisvermeidungsmodul. Das fließende Licht ändert die Richtung, sobald ein Hindernis erkannt wird, was es zu einer spannenden Ergänzung für Ihre Wohn- oder Bürodekoration macht.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

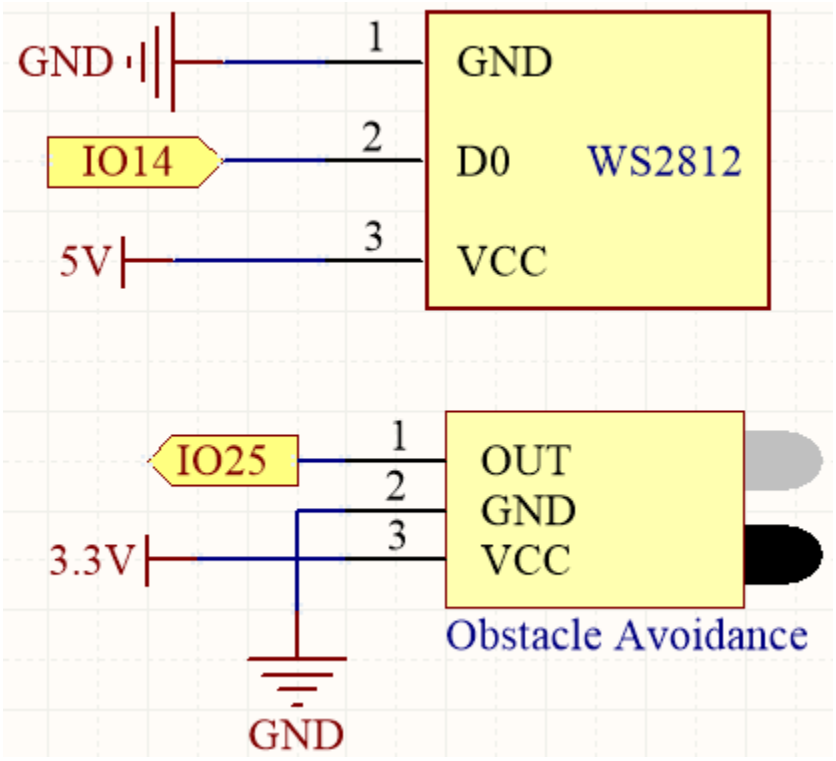
Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

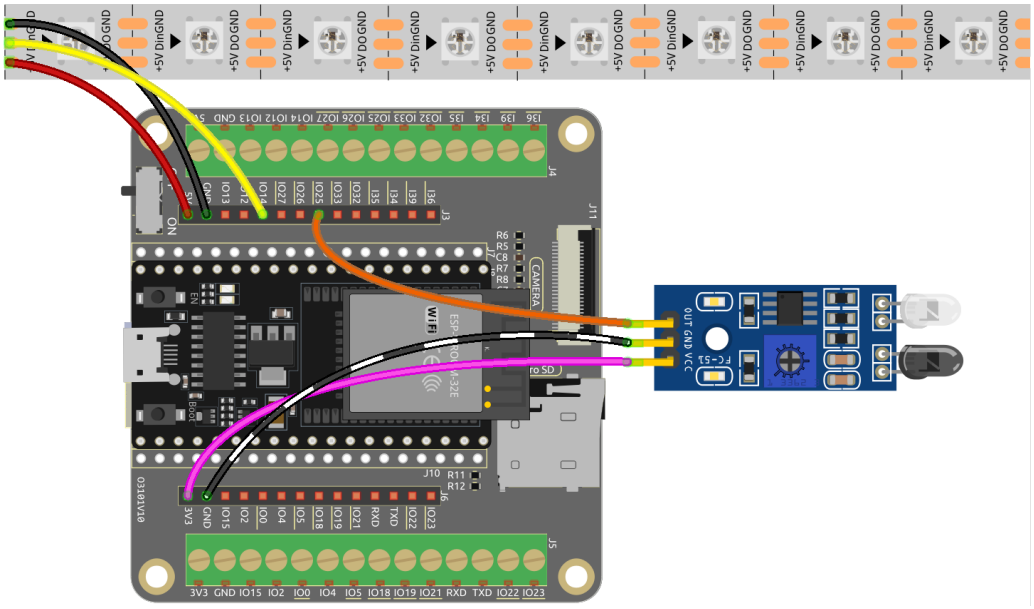
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Modul zur Hindernisvermeidung	
WS2812 RGB 8 LEDs Leiste	

Schaltplan



Der WS2812 LED-Streifen besteht aus einer Reihe von einzelnen LEDs, die so programmiert werden können, dass sie verschiedene Farben und Muster anzeigen. In diesem Projekt ist der Streifen so eingestellt, dass er ein fließendes Licht zeigt, das in eine bestimmte Richtung läuft und die Richtung ändert, wenn ein Hindernis vom Hindernisvermeidungsmodul erkannt wird.

Verdrahtung



Code

Bemerkung:

- Sie können die Datei `6.2_flowing_led.ino` direkt unter dem Pfad `esp32-starter-kit-main\c\codes\6.2_flowing_led` öffnen.
- Oder kopieren Sie diesen Code in die Arduino IDE.

Dieses Projekt erweitert die Funktionalität des *2.7 RGB-LED-Streifen* Projekts, indem es die Möglichkeit hinzufügt, zufällige Farben auf dem LED-Streifen anzuzeigen. Zusätzlich wurde ein Hindernisvermeidungsmodul integriert, um die Laufrichtung des fließenden Lichts dynamisch zu ändern.

2.33 6.3 Einparkhilfe

Stellen Sie sich Folgendes vor: Sie sitzen in Ihrem Auto und möchten rückwärts in eine enge Parklücke fahren. Mit unserem Projekt haben Sie ein Ultraschallmodul am Heck Ihres Fahrzeugs montiert, das als digitales Auge fungiert. Wenn Sie den Rückwärtsgang einlegen, wird das Modul aktiviert und sendet Ultraschallimpulse aus, die von Hindernissen hinter Ihnen reflektiert werden.

Das Besondere geschieht, wenn diese Impulse zum Modul zurückkehren. Es berechnet blitzschnell die Entfernung zwischen Ihrem Auto und den Objekten und verwandelt diese Daten in eine Echtzeit-Visuelle-Rückmeldung, die auf einem lebendigen LCD-Bildschirm angezeigt wird. Sie erleben dynamische, farbkodierte Indikatoren, die die Nähe zu Hindernissen anzeigen und Ihnen so ein kristallklares Verständnis der Umgebung ermöglichen.

Aber wir haben noch mehr getan. Um Sie noch tiefer in dieses Fahrerlebnis einzutauchen, haben wir einen lebhaften Summer integriert. Nähert sich Ihr Auto einem Hindernis, intensiviert sich das Tempo des Summers und erzeugt eine akustische Symphonie der Warnungen. Es ist, als hätten Sie ein persönliches Orchester, das Sie durch die Komplexität des Rückwärtsparkens leitet.

Dieses innovative Projekt verbindet Spitzentechnologie mit einer interaktiven Benutzeroberfläche und macht Ihr Rückwärtserlebnis sicher und stressfrei. Mit dem Ultraschallmodul, dem LCD-Display und dem lebhaften Summer, die harmonisch zusammenarbeiten, fühlen Sie sich beim Manövrieren in engen Räumen ermächtigt und selbstbewusst, sodass Sie sich auf die Freude am Fahren konzentrieren können.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

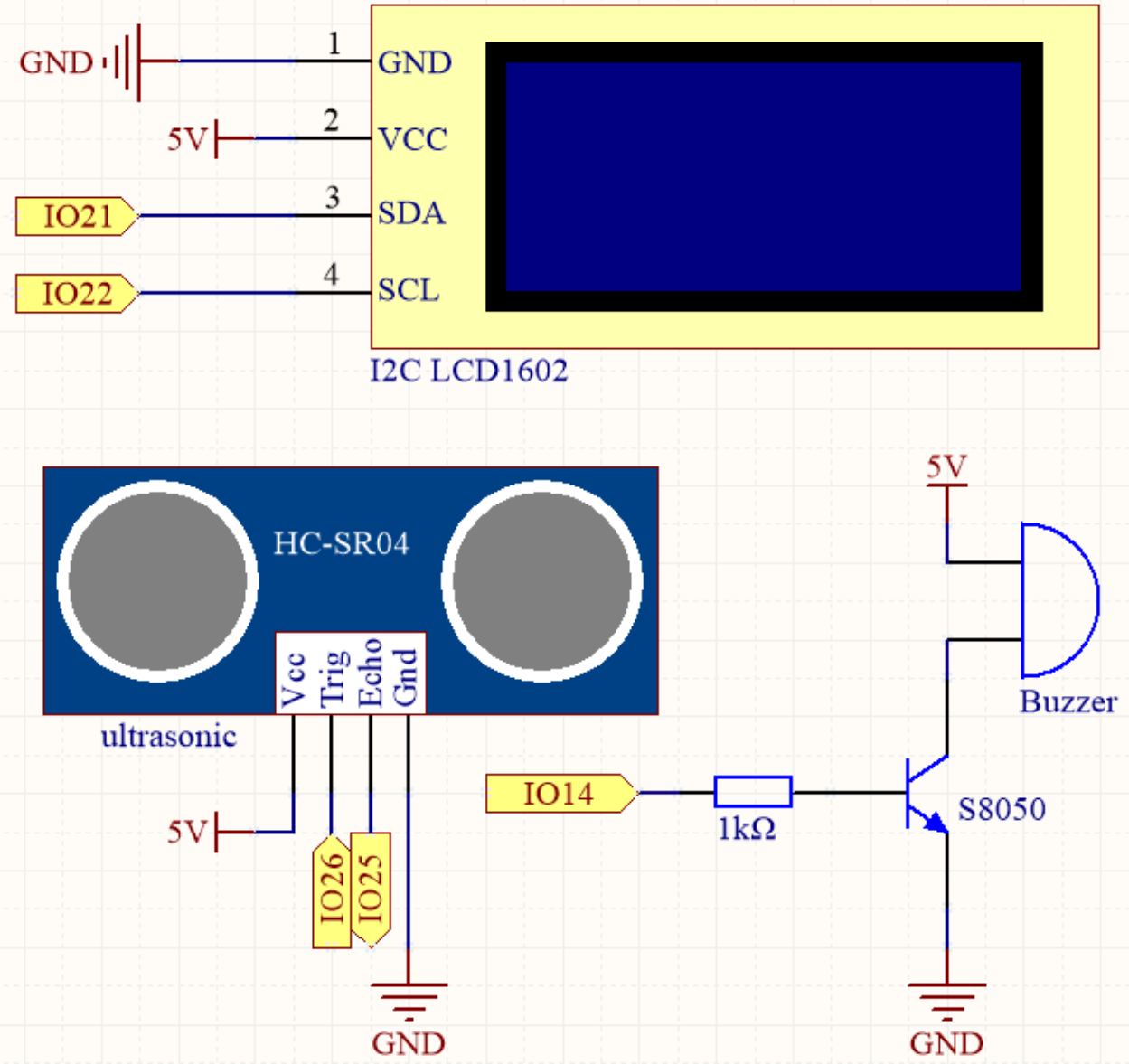
Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

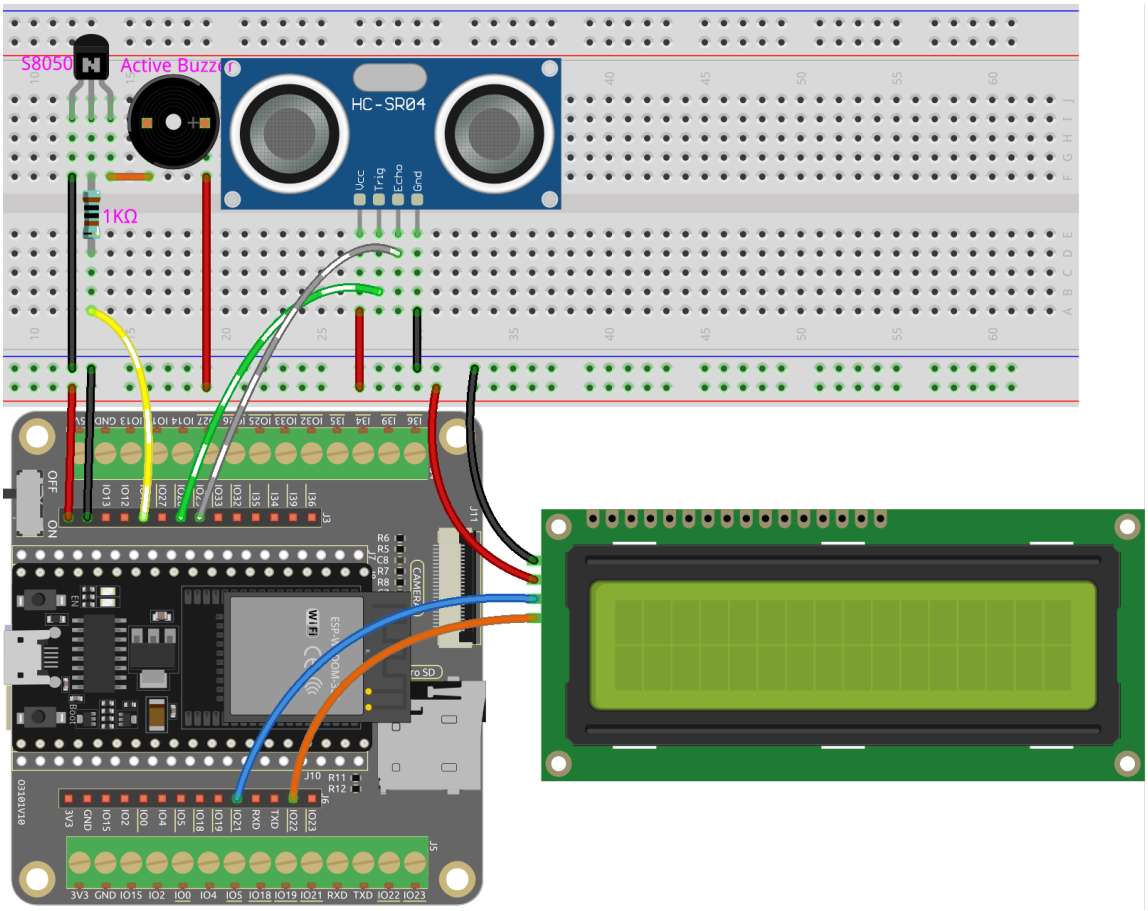
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Ultraschall-Modul	
Summer	-
Transistor	
I2C LCD1602	

Schaltplan



Der Ultraschallsensor im Projekt sendet hochfrequente Schallwellen aus und misst die Zeit, die die Wellen benötigen, um nach dem Aufprall auf ein Objekt zurückzukehren. Durch die Analyse dieser Daten kann die Entfernung zwischen dem Sensor und dem Objekt berechnet werden. Um eine Warnung zu geben, wenn das Objekt zu nah ist, wird ein Summer verwendet, um ein hörbares Signal zu erzeugen. Zusätzlich wird die gemessene Entfernung auf einem LCD-Bildschirm zur einfachen Visualisierung angezeigt.

Verdrahtung



Code

Bemerkung:

- Sie können die Datei `6.3_reversing_aid.ino` direkt unter dem Pfad `esp32-starter-kit-main\c\codes\6.3_reversing_aid` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*
- Hier wird die Bibliothek `LiquidCrystal I2C` verwendet, die Sie aus dem **Library Manager** installieren können.

Nachdem der Code erfolgreich hochgeladen wurde, wird die aktuell erkannte Entfernung auf dem LCD angezeigt. Dann ändert der Summer die Klangfrequenz je nach Entfernung.

Bemerkung: Wenn der Code und die Verdrahtung korrekt sind, das LCD jedoch weiterhin keinen Inhalt anzeigt, können Sie den Potentiometer auf der Rückseite verstellen, um den Kontrast zu erhöhen.

Wie funktioniert das?

Dieser Code hilft uns, ein einfaches Entfernungsmessgerät zu erstellen, das die Entfernung zwischen Objekten messen und Feedback über ein LCD-Display und einen Summer geben kann.

Die Funktion `loop()` enthält die Hauptlogik des Programms und läuft kontinuierlich. Lassen Sie uns die Funktion `loop()` genauer betrachten.

1. Schleife zum Lesen der Entfernung und Aktualisieren der Parameter

   In der `loop` liest der Code zunächst die vom Ultraschallmodul gemessene Entfernung und aktualisiert den Intervallparameter basierend auf der Entfernung.

   ```
   // Update the distance
   distance = readDistance();

   // Update intervals based on distance
   if (distance <= 10) {
       intervals = 300;
   } else if (distance <= 20) {
       intervals = 500;
   } else if (distance <= 50) {
       intervals = 1000;
   } else {
       intervals = 2000;
   }
   ```

2. Überprüfen, ob es Zeit zum Piepen ist

   Der Code berechnet die Differenz zwischen der aktuellen Zeit und der vorherigen Piepzeit, und wenn die Differenz größer oder gleich der Intervallzeit ist, löst er den Summer aus und aktualisiert die vorherige Piepzeit.

   ```
   unsigned long currentMillis = millis();
   if (currentMillis - previousMillis >= intervals) {
       Serial.println("Beeping!");
       beep();
       previousMillis = currentMillis;
   }
   ```

3. LCD-Display aktualisieren

   Der Code löscht das LCD-Display und zeigt dann „Dis:“ und die aktuelle Entfernung in Zentimetern in der ersten Zeile an.

   ```
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Dis: ");
   lcd.print(distance);
   lcd.print(" cm");

   delay(100);
   ```

2.34 6.4 Digitaler Würfel

Dieses Projekt baut auf dem *2.5 Ziffernanzeige* Projekt auf, indem ein Knopf hinzugefügt wird, um die auf dem Siebensegment-Display angezeigte Ziffer zu steuern.

In diesem Projekt wird eine zufällige Zahl erzeugt und auf dem Siebensegment-Display angezeigt, um einen Würfelwurf zu simulieren. Wenn der Knopf gedrückt wird, wird eine stabile Zahl (zufällig ausgewählt von 1 bis 6) auf dem Siebensegment-Display angezeigt. Ein erneutes Drücken des Knopfes startet die Simulation eines Würfelwurfs, wobei wieder zufällige Zahlen generiert werden. Dieser Zyklus setzt sich bei jedem Drücken des Knopfes fort.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

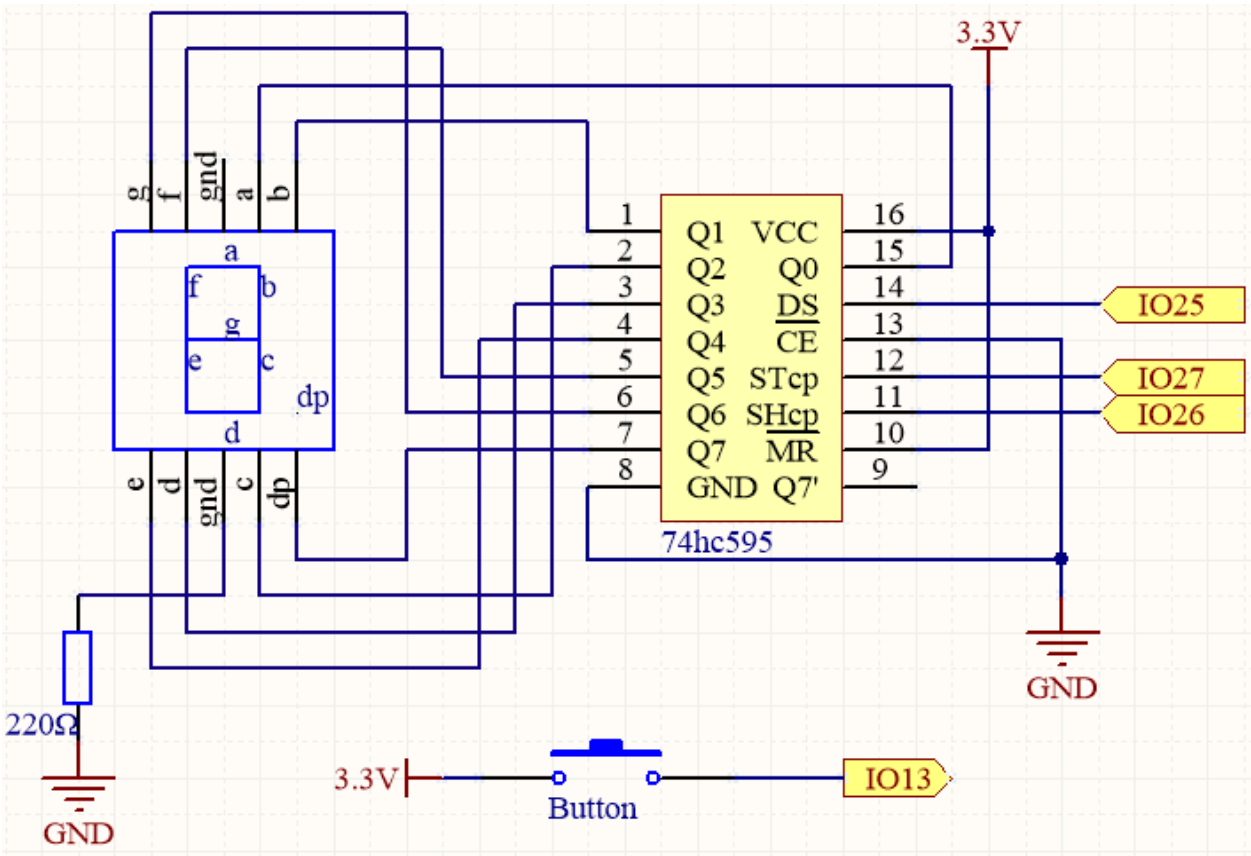
Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
74HC595	
7-Segment-Anzeige	
Taste	

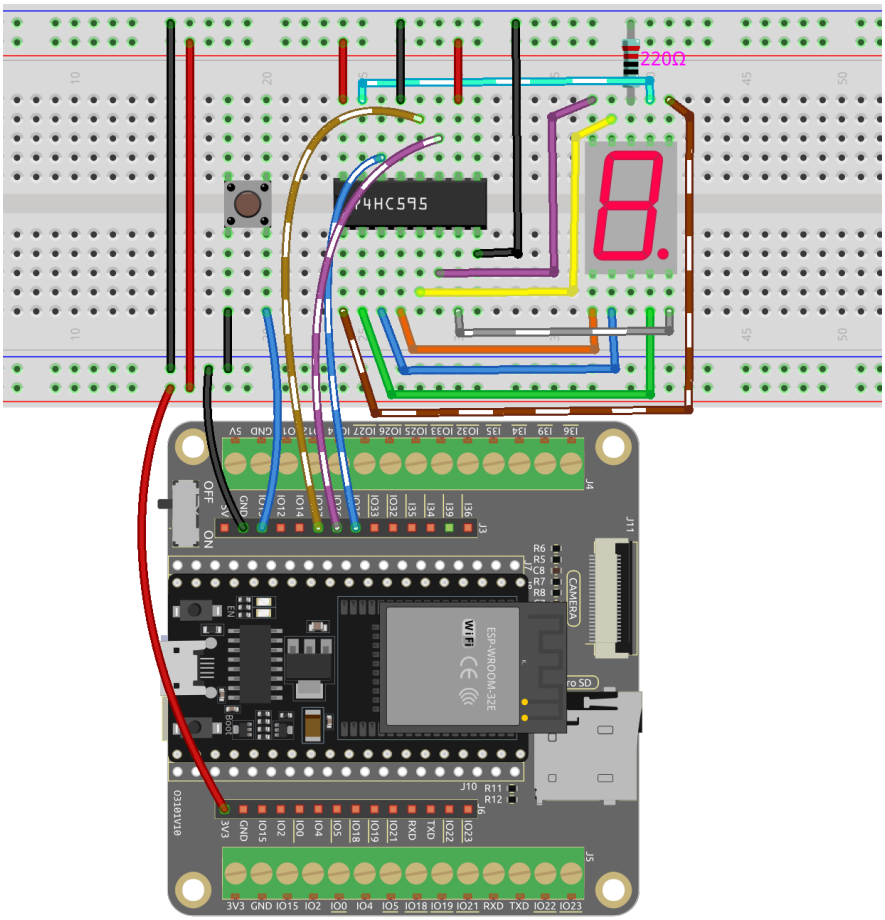
Schaltplan



Dieses Projekt baut auf dem *2.5 7-Segment-Anzeige* Projekt auf, indem ein Knopf hinzugefügt wird, um die auf dem Siebensegment-Display angezeigte Ziffer zu steuern.

Der Knopf ist direkt an IO13 angeschlossen, ohne einen externen Pull-up- oder Pull-down-Widerstand, da IO13 einen internen Pull-up-Widerstand von 47K besitzt, wodurch ein zusätzlicher externer Widerstand nicht notwendig ist.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `6.4_digital_dice.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\6.4_digital_dice`.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Dieses Projekt basiert auf *2.5 7-Segment-Anzeige* mit einem Knopf, um die scrollende Anzeige auf dem 7-Segment-Display zu starten/pausieren.

Wenn der Knopf gedrückt wird, scrollt das 7-Segment-Display durch die Zahlen 1-6, und wenn der Knopf losgelassen wird, zeigt es eine zufällige Zahl an.

2.35 6.5 Farbverlauf

Sind Sie bereit, eine Welt voller Farben zu erleben? Dieses Projekt nimmt Sie mit auf eine magische Reise, auf der Sie eine RGB-LED steuern und sanfte Farbübergänge erzielen können. Egal, ob Sie Ihrer Wohnkultur etwas Farbe verleihen oder ein unterhaltsames Programmierprojekt suchen, dieses Projekt bietet Ihnen genau das. Tauchen wir gemeinsam in diese farbenfrohe Welt ein!

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

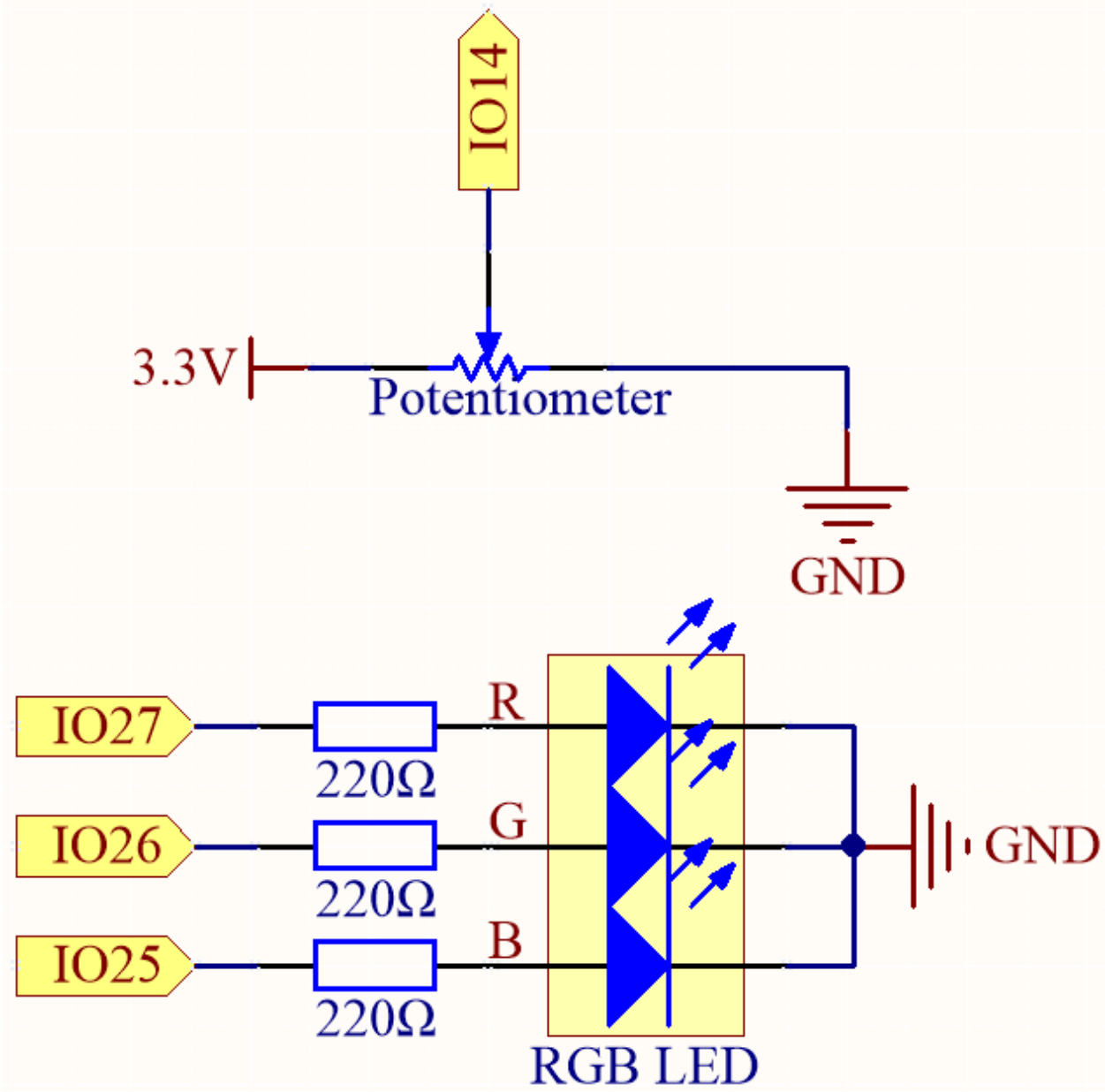
Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

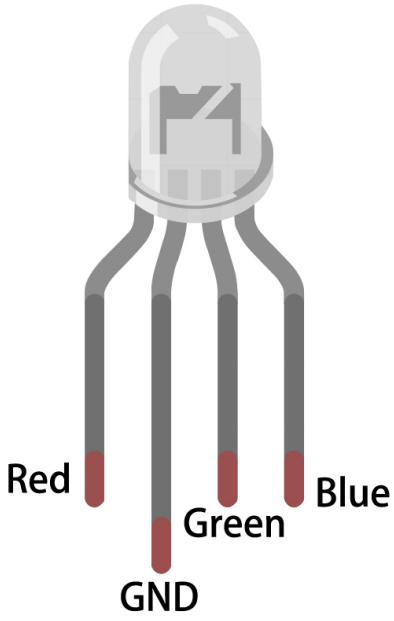
Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Potentiometer	
RGB LED	

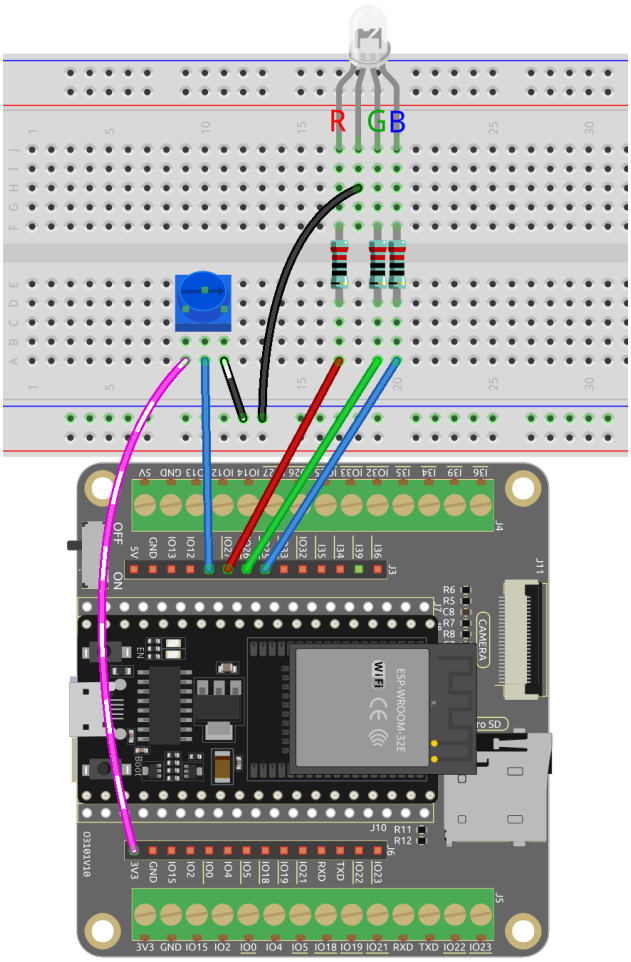
Schaltplan



Verdrahtung



Die RGB-LED hat 4 Pins: Der lange Pin ist der gemeinsame Kathodenpin, der normalerweise mit GND verbunden wird; der linke Pin neben dem längsten Pin ist Rot; und die beiden Pins rechts sind Grün und Blau.



Code

Bemerkung:

- Sie können die Datei `6.5_color_gradient.ino` direkt unter dem Pfad `esp32-starter-kit-main\c\codes\6.5_color_gradient` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

Dieses Projekt verwendet eine RGB-LED und ein Potentiometer, um einen Farbmischeffekt zu erzeugen. Das Potentiometer wird verwendet, um den Farbtonwert der LED anzupassen, der dann mit einer Farbkonvertierungsfunktion in RGB-Werte umgewandelt wird. Die RGB-Werte werden dann verwendet, um die Farbe der LED zu aktualisieren.

Wie funktioniert das?

Dieses Projekt baut auf dem *2.3 Bunte Beleuchtung* Projekt auf, indem ein Potentiometer hinzugefügt wird, um den Farbtonwert der LED anzupassen. Der Farbtonwert wird dann mit einer Farbkonvertierungsfunktion in RGB-Werte umgewandelt.

1. In der Schleifenfunktion wird der Wert des Potentiometers gelesen und in einen Farbtonwert (0-360) umgewandelt.

```
int knobValue = analogRead(KNOB_PIN);
float hueValue = (float) knobValue / 4095.0;
int hue = (int) (hueValue * 360);
```

2. Der Farbtonwert wird mit der Funktion `HUEtoRGB()` in RGB-Werte umgewandelt und die LED mit den neuen Farbwerten aktualisiert.

```
int red, green, blue;
HUEtoRGB(hue, &red, &green, &blue);
setColor(red, green, blue);
```

3. Die Funktion `setColor()` setzt den Wert der roten, grünen und blauen Kanäle mit der Bibliothek `LEDC`.

```
void setColor(int red, int green, int blue) {
    ledcWrite(redChannel, red);
    ledcWrite(greenChannel, green);
    ledcWrite(blueChannel, blue);
}
```

4. Die Funktion `HUEtoRGB` wandelt einen Farbtonwert mit dem HSL-Farbmodell in RGB-Werte um.

```
void HUEtoRGB(int hue, int* red, int* green, int* blue) {
    float h = (float) hue / 60.0;
    float c = 1.0;
    float x = c * (1.0 - fabs(fmod(h, 2.0) - 1.0));
    float r, g, b;
    if (h < 1.0) {
        r = c;
        g = x;
        b = 0;
...
```

2.36 6.6 Pflanzenüberwachung

Willkommen beim Projekt Pflanzenüberwachung!

In diesem Projekt werden wir ein ESP32-Board verwenden, um ein System zu erstellen, das uns hilft, unsere Pflanzen zu pflegen. Mit diesem System können wir die Temperatur, die Luftfeuchtigkeit, die Bodenfeuchtigkeit und den Lichtpegel unserer Pflanzen überwachen und sicherstellen, dass sie die Pflege und Aufmerksamkeit erhalten, die sie zum Gedeihen benötigen.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

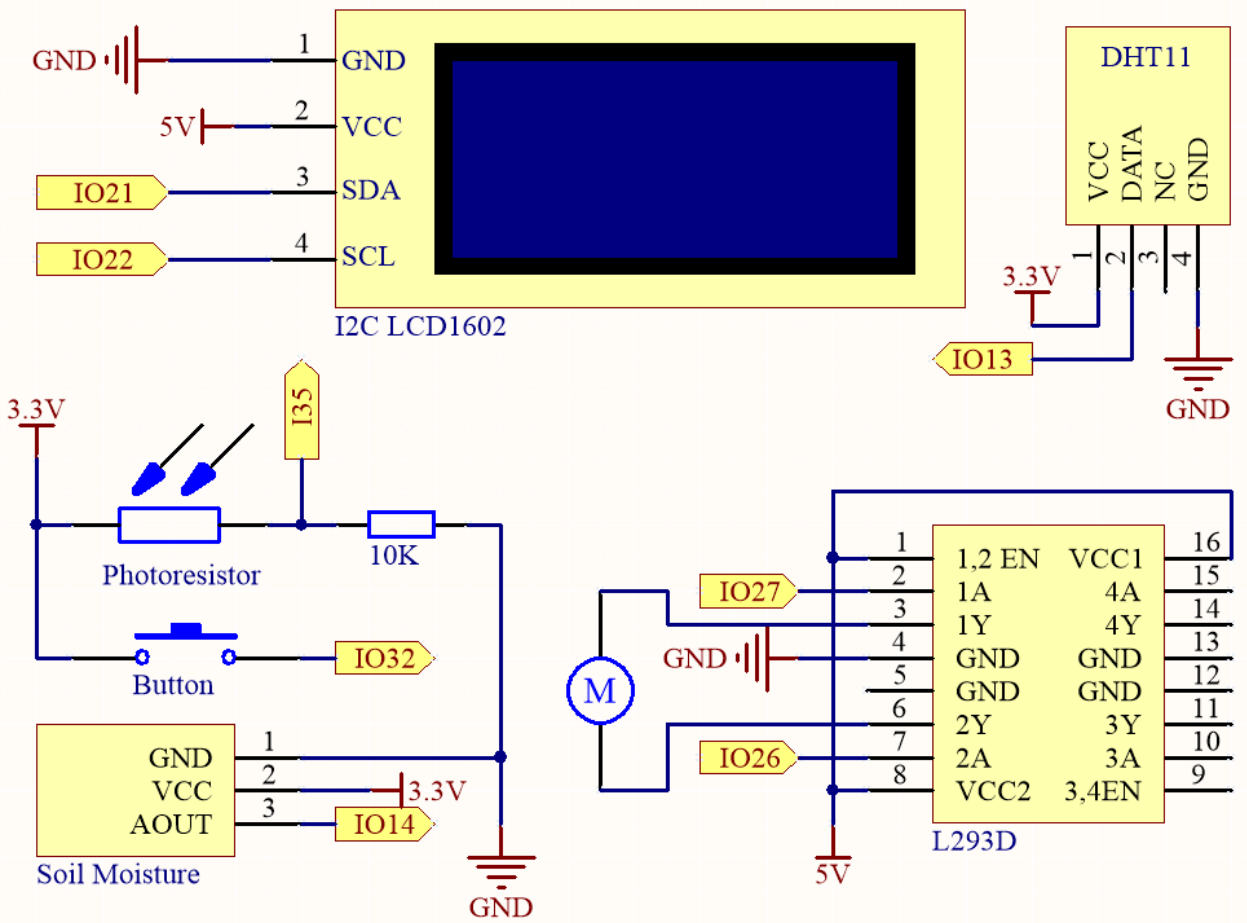
Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
DHT11 Feuchtigkeits- und Temperatursensor	
I2C LCD1602	
Zentrifugalpumpe	-
L293D	-
Taste	
Fotowiderstand	
Widerstand	
Bodenfeuchtigkeitsmodul	

Schaltplan

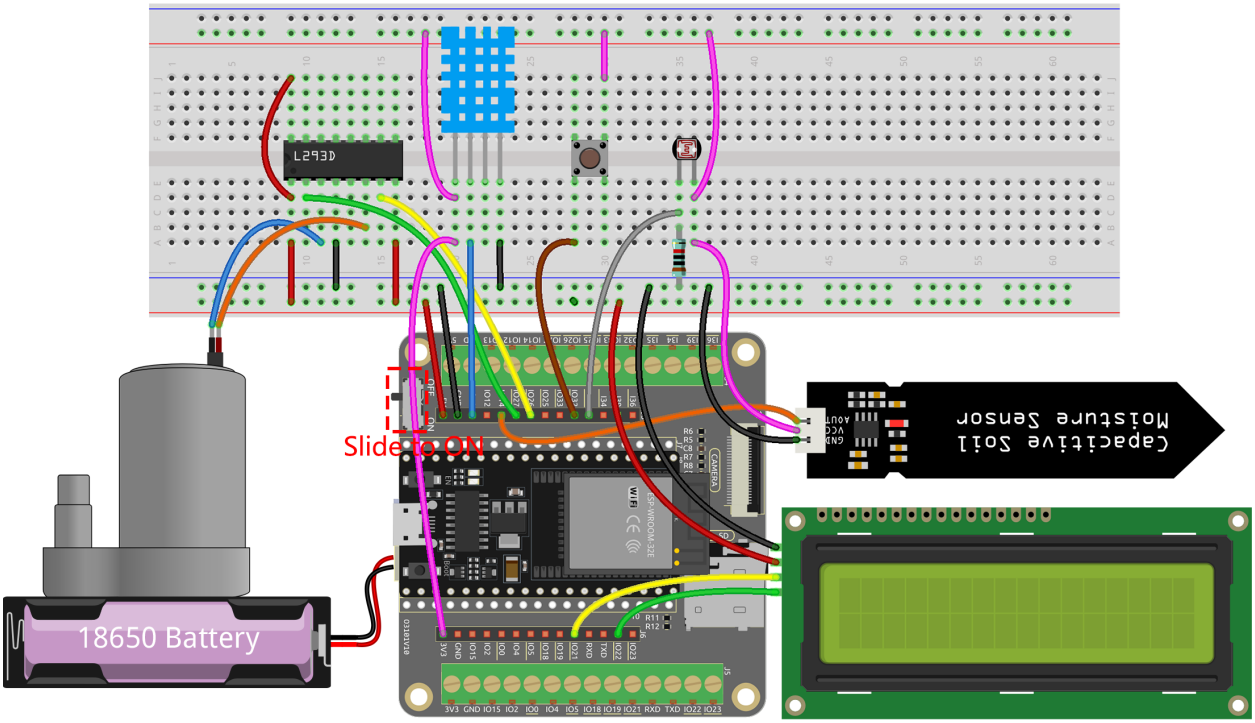


Das System verwendet einen DHT11-Sensor, um die Temperatur- und Feuchtigkeitswerte der Umgebung zu messen. In der Zwischenzeit wird ein Bodenfeuchtigkeitsmodul verwendet, um den Feuchtigkeitsgehalt des Bodens zu messen, und ein Fotowiderstand, um das Lichtniveau zu messen. Die Messwerte dieser Sensoren werden auf einem LCD-Bildschirm angezeigt, und eine Wasserpumpe kann gesteuert werden mit einem Knopf, um die Pflanze bei Bedarf zu bewässern.

IO32 hat einen internen Pull-Down-Widerstand von 1K und ist standardmäßig auf einem niedrigen Logikniveau. Wenn der Knopf gedrückt wird, stellt er eine Verbindung zu VCC (Hochspannung) her, was zu einem hohen Logikniveau auf IO32 führt.

Verdrahtung

Bemerkung: Es wird empfohlen, hier die Batterie einzulegen und dann den Schalter auf dem Erweiterungsboard auf die ON-Position zu schieben, um die Batterieversorgung zu aktivieren.



Code

Bemerkung:

- Sie können die Datei `6.6_plant_monitor.ino` direkt unter dem Pfad `esp32-starter-kit-main\c\codes\6.6_plant_monitor` öffnen.
- Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*
- Die Bibliotheken `LiquidCrystal_I2C` und `DHT sensor library` werden hier verwendet, Sie können sie aus dem **Library Manager** installieren.

- Nach dem Hochladen des Codes zeigt das I2C LCD1602 abwechselnd Temperatur und Luftfeuchtigkeit sowie Bodenfeuchtigkeit und Lichtintensität analoge Werte an, mit einem 2-Sekunden-Intervall.
- Die Wasserpumpe wird mit einem Knopfdruck gesteuert. Um die Pflanzen zu bewässern, halten Sie den Knopf gedrückt und lassen Sie ihn los, um das Bewässern zu stoppen.

Bemerkung: Wenn der Code und die Verdrahtung korrekt sind, das LCD jedoch weiterhin keinen Inhalt anzeigt, können Sie den Potentiometer auf der Rückseite verstellen, um den Kontrast zu erhöhen.

2.37 6.7 Zahlenraten

Fühlst du dich glücklich? Möchtest du deine Intuition testen und herausfinden, ob du die richtige Zahl erraten kannst? Dann ist das Spiel „Zahlenraten“ genau das Richtige für dich!

Mit diesem Projekt kannst du ein spannendes und unterhaltsames Glücksspiel erleben.

Die Spieler geben über eine IR-Fernbedienung Zahlen zwischen 0 und 99 ein, um die zufällig generierte Glückszahl zu erraten. Das System zeigt die eingegebene Zahl des Spielers auf einem LCD-Bildschirm an, zusammen mit Tipps für die obere und untere Grenze, um den Spieler zur richtigen Antwort zu führen. Mit jedem Versuch nähern sich die Spieler der Glückszahl, bis schließlich jemand den Jackpot knackt und das Spiel gewinnt!

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

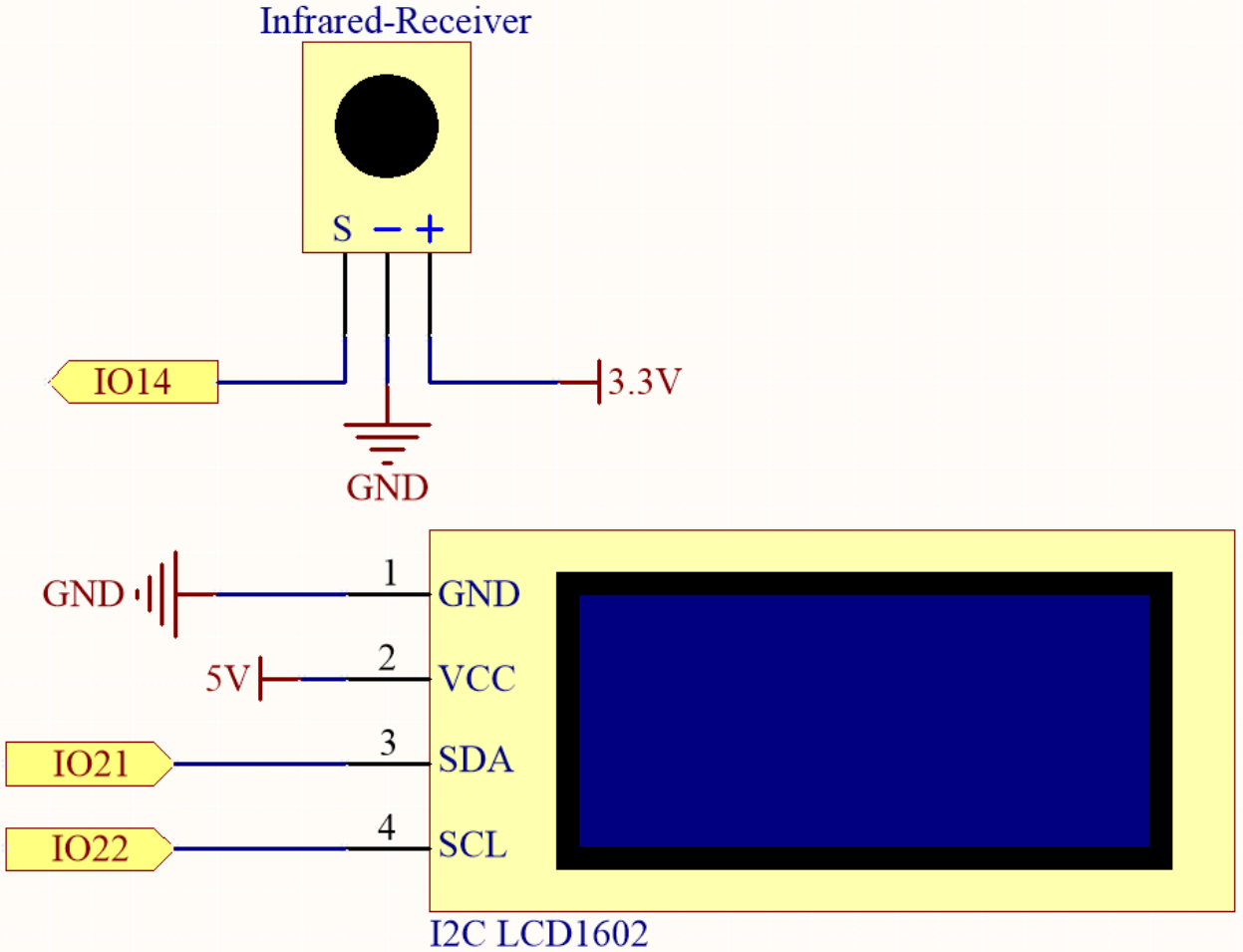
Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

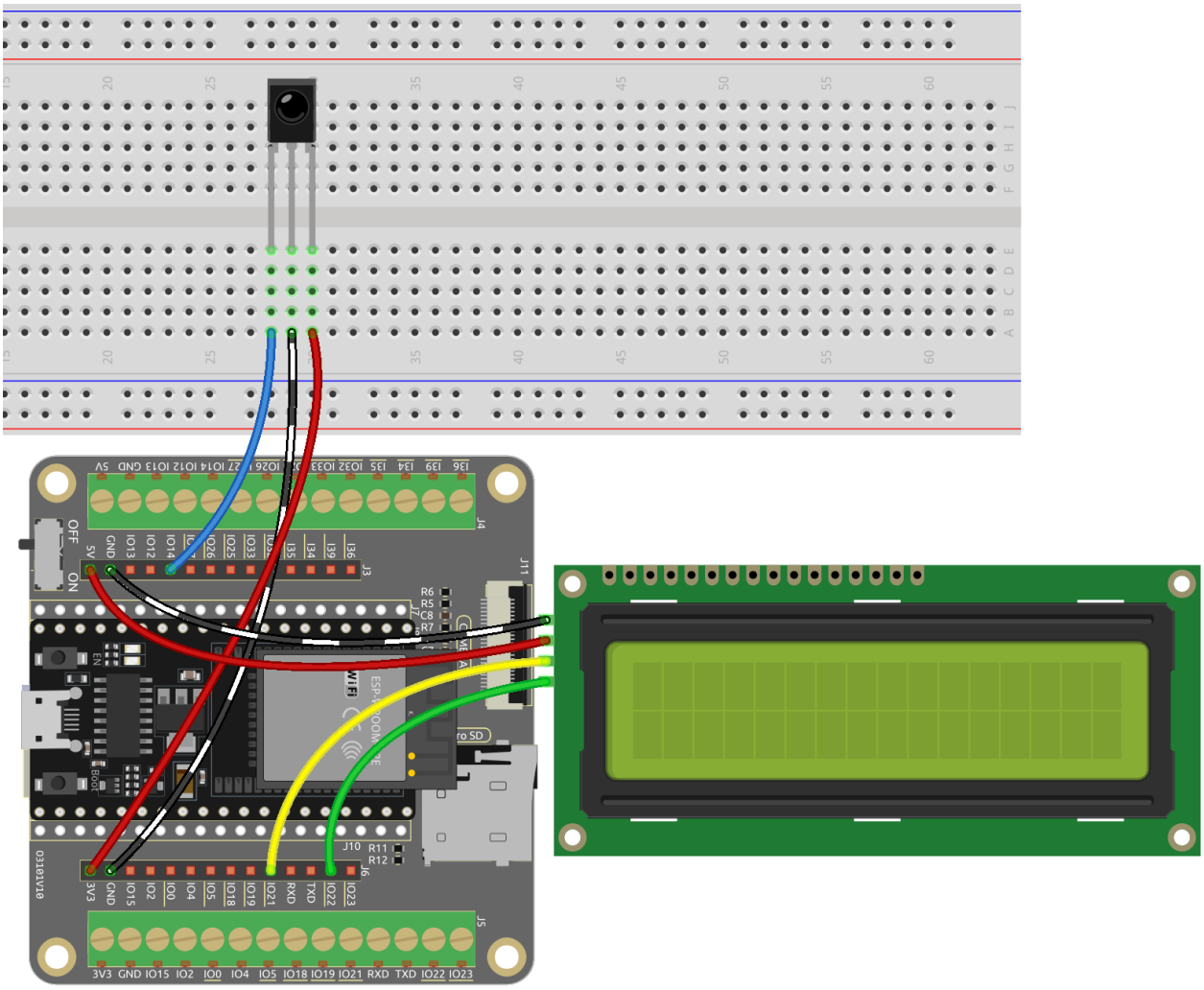
Du kannst sie auch einzeln über die unten stehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
IR-Empfänger	
I2C LCD1602	

Schaltplan



Verdrahtung



Code

Bemerkung:

- Du kannst die Datei `6.7_guess_number.ino` direkt unter dem Pfad `esp32-starter-kit-main\c\codes\6.7_guess_number` öffnen.
- Hier werden die Bibliotheken `LiquidCrystal_I2C` und `IRremoteESP8266` verwendet, siehe *Manuelle Installation* für eine Anleitung zur Installation.

- Nachdem der Code erfolgreich hochgeladen wurde, drücke irgendeine Zahlentaste auf der Fernbedienung, um das Spiel zu starten.
- Gib eine Zahl mit den Zahlentasten auf der Fernbedienung ein. Um eine einzelne Ziffer einzugeben, musst du die **cycle**-Taste zum Bestätigen drücken.
- Das System zeigt die eingegebene Zahl und die Tipps für die obere und untere Grenze auf dem LCD-Bildschirm an.
- Rate weiter, bis du die Glückszahl richtig erraten hast.
- Nach einem erfolgreichen Versuch zeigt das System eine Erfolgsmeldung an und generiert eine neue Glückszahl.

Bemerkung: Wenn der Code und die Verkabelung korrekt sind, das LCD aber dennoch keine Inhalte anzeigt, kannst du das Potentiometer auf der Rückseite justieren, um den Kontrast zu erhöhen.

Wie funktioniert das?

1. In der Funktion `setup()` werden der I2C-LCD-Bildschirm und der IR-Empfänger initialisiert. Dann wird die Funktion `initNewValue()` aufgerufen, um eine neue zufällige Glückszahl zu generieren, und eine Willkommensnachricht wird auf dem LCD-Bildschirm angezeigt.

```
void setup() {
    // Initialize the LCD screen
    lcd.init();
    lcd.backlight();

    // Start the serial communication
    Serial.begin(9600);

    // Enable the IR receiver
    irrecv.enableIRIn();

    // Initialize a new lucky point value
    initNewValue();
}
```

2. In der Funktion `loop` wartet der Code auf ein Signal vom IR-Empfänger. Wenn ein Signal empfangen wird, wird die Funktion `decodeKeyValue` aufgerufen, um das Signal zu dekodieren und den entsprechenden Tastenwert zu erhalten.

```
void loop() {
    // If a signal is received from the IR receiver
    if (irrecv.decode(&results)) {
        bool result = 0;
        String num = decodeKeyValue(results.value);

        // If the POWER button is pressed
        if (num == "POWER") {
            initNewValue(); // Initialize a new lucky point value
        }

        // If the CYCLE button is pressed
        else if (num == "CYCLE") {
            result = detectPoint(); // Detect the input number
            lcdShowInput(result); // Show the result on the LCD screen
        }

        // If a number button (0-9) is pressed,
        //add the digit to the input number
        //and detect the number if it is greater than or equal to 10
        else if (num >= "0" && num <= "9") {
            count = count * 10;
            count += num.toInt();
            if (count >= 10) {
                result = detectPoint();
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
            }
            lcdShowInput(result);
        }
        irrecv.resume();
    }
}
```

- Abhängig vom Tastenwert wird die entsprechende Funktion aufgerufen. Wenn eine Zahlentaste gedrückt wird, wird die Variable `count` aktualisiert und die Funktion `detectPoint` aufgerufen, um zu prüfen, ob die eingegebene Zahl korrekt ist. Die Funktion `lcdShowInput` wird aufgerufen, um die eingegebene Zahl und die Tipps für die obere und untere Grenze auf dem LCD-Bildschirm anzuzeigen.
- Wenn die `POWER`-Taste gedrückt wird, wird die Funktion `initNewValue` aufgerufen, um eine neue Glückspunkt-Zahl zu generieren und die Willkommensnachricht auf dem LCD-Bildschirm anzuzeigen.
- Wenn die `CYCLE`-Taste gedrückt wird, wird die Funktion `detectPoint` aufgerufen, um zu prüfen, ob die eingegebene Zahl korrekt ist. Die Funktion `lcdShowInput` wird aufgerufen, um die eingegebene Zahl und die Tipps für die obere und untere Grenze auf dem LCD-Bildschirm anzuzeigen.

7. Bluetooth&SD-Karte&Kamera&Lautsprecher

2.38 7.1 Bluetooth

Dieses Projekt bietet eine Anleitung zur Entwicklung einer einfachen Bluetooth Low Energy (BLE) Seriellen Kommunikationsanwendung mit dem Mikrocontroller ESP32. Der ESP32 ist ein leistungsstarker Mikrocontroller, der Wi-Fi und Bluetooth Konnektivität integriert, was ihn zu einem idealen Kandidaten für die Entwicklung drahtloser Anwendungen macht. BLE ist ein energiesparendes drahtloses Kommunikationsprotokoll, das für die Nahbereichskommunikation konzipiert ist. Dieses Dokument beschreibt die Schritte zur Einrichtung des ESP32 als BLE-Server, um mit einem BLE-Client über eine serielle Verbindung zu kommunizieren.

Über die Bluetooth-Funktion

Das ESP32 WROOM 32E ist ein Modul, das Wi-Fi und Bluetooth Konnektivität in einem einzigen Chip integriert. Es unterstützt sowohl das Bluetooth Low Energy (BLE) als auch das klassische Bluetooth-Protokoll.

Das Modul kann sowohl als Bluetooth-Client als auch als Server verwendet werden. Als Bluetooth-Client kann das Modul sich mit anderen Bluetooth-Geräten verbinden und Daten mit ihnen austauschen. Als Bluetooth-Server kann das Modul Dienste für andere Bluetooth-Geräte bereitstellen.

Das ESP32 WROOM 32E unterstützt verschiedene Bluetooth-Profile, einschließlich des Generic Access Profile (GAP), Generic Attribute Profile (GATT) und Serial Port Profile (SPP). Das SPP-Profil ermöglicht es dem Modul, einen seriellen Port über Bluetooth zu emulieren, wodurch eine serielle Kommunikation mit anderen Bluetooth-Geräten ermöglicht wird.

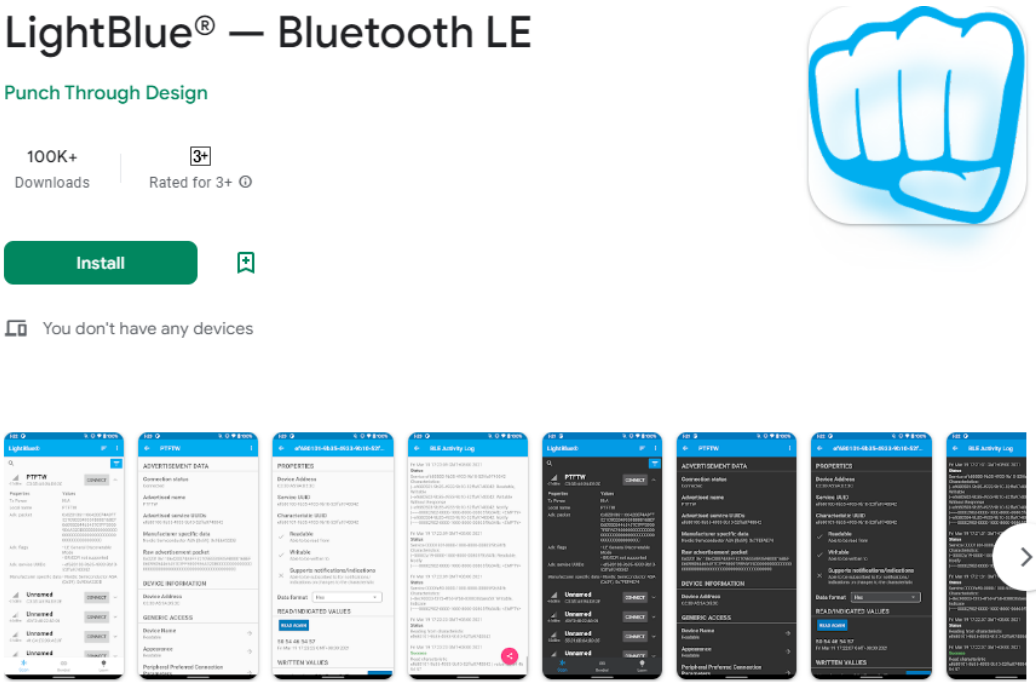
Um die Bluetooth-Funktion des ESP32 WROOM 32E zu nutzen, muss es mit einem geeigneten Software Entwicklungskit (SDK) oder mit der Arduino IDE und der ESP32 BLE-Bibliothek programmiert werden. Die ESP32 BLE-Bibliothek bietet eine hochrangige Schnittstelle für die Arbeit mit BLE. Sie enthält Beispiele, die demonstrieren, wie man das Modul als BLE-Client und -Server verwendet.

Insgesamt bietet die Bluetooth-Funktion des ESP32 WROOM 32E eine bequeme und energiesparende Möglichkeit, drahtlose Kommunikation in Ihren Projekten zu ermöglichen.

Bedienungsschritte

Hier sind die schrittweisen Anweisungen zur Einrichtung der Bluetooth-Kommunikation zwischen Ihrem ESP32 und einem mobilen Gerät mit der LightBlue-App:

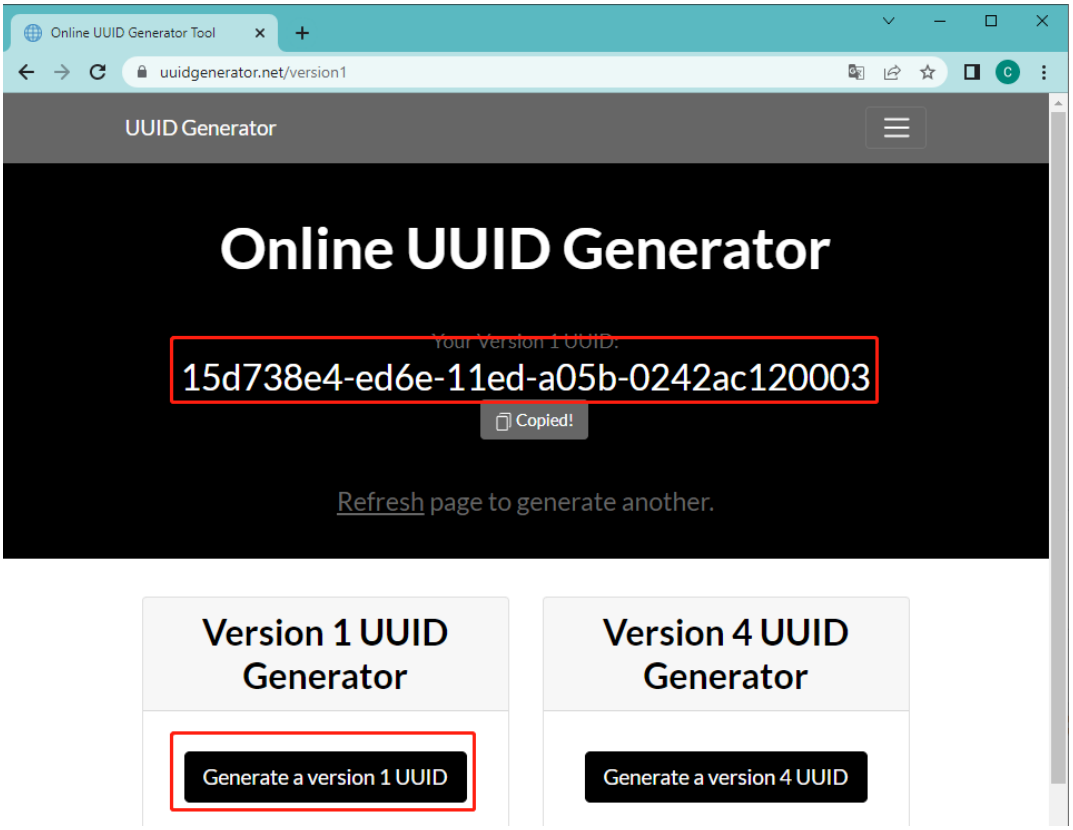
1. Laden Sie die LightBlue-App aus dem **App Store** (für iOS) oder **Google Play** (für Android) herunter.



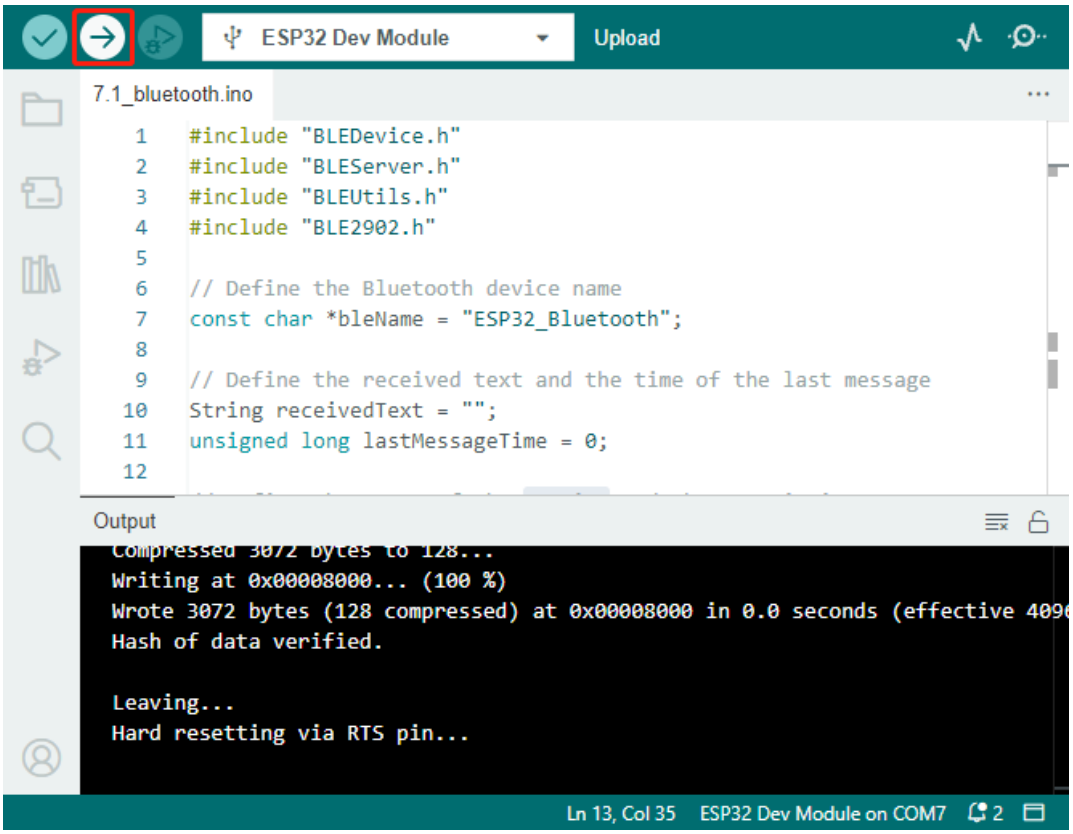
2. Öffnen Sie die Datei `7.1_bluetooth.ino` im Verzeichnis `esp32-starter-kit-main\c\codes\7.1_bluetooth`, oder kopieren Sie den Code in die Arduino IDE.

3. Um UUID-Konflikte zu vermeiden, wird empfohlen, drei neue UUIDs zufällig mit dem zu generieren und sie in den folgenden Codezeilen einzufügen.

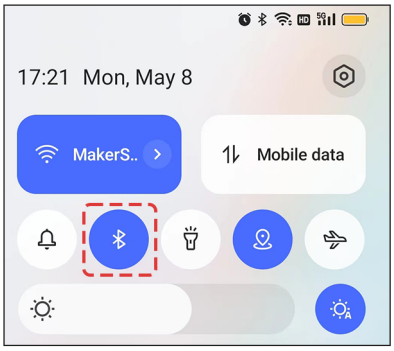
```
#define SERVICE_UUID           "your_service_uuid_here"
#define CHARACTERISTIC_UUID_RX "your_rx_characteristic_uuid_here"
#define CHARACTERISTIC_UUID_TX "your_tx_characteristic_uuid_here"
```



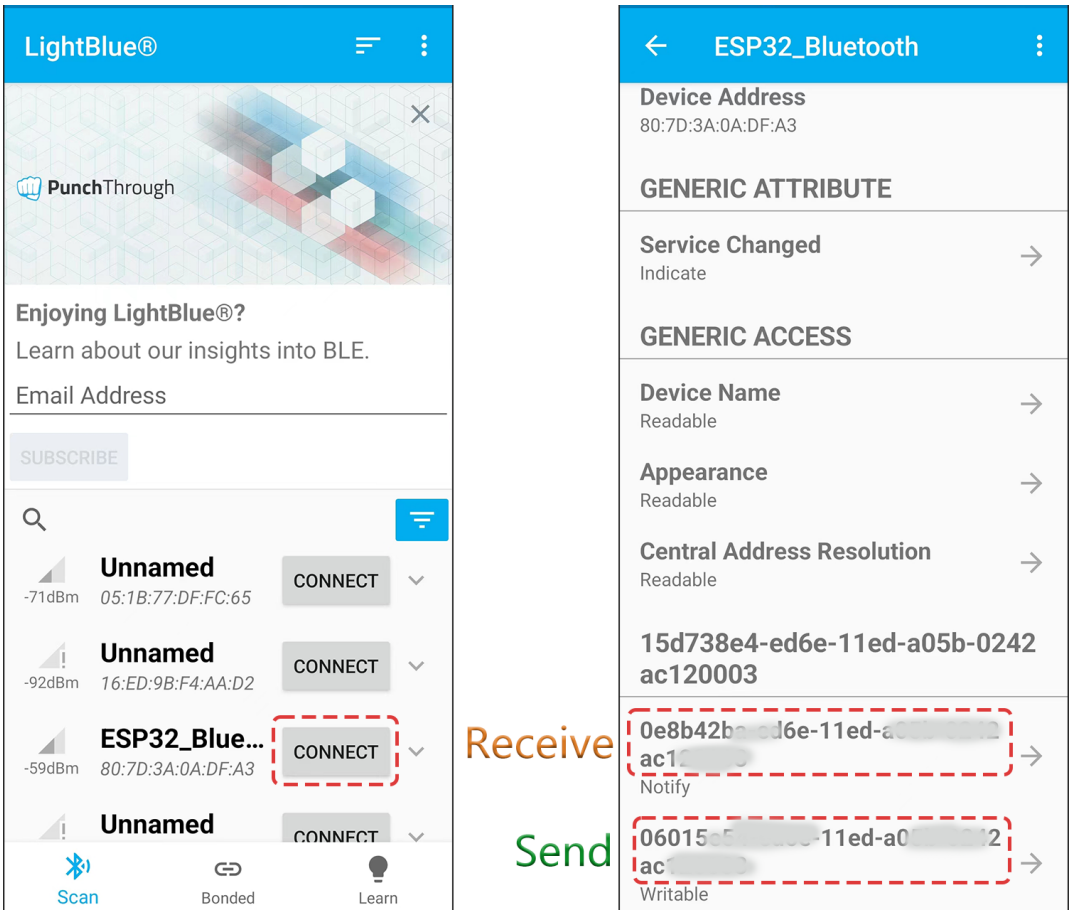
4. Wählen Sie das richtige Board und den richtigen Port aus, dann klicken Sie auf die **Upload**-Taste.



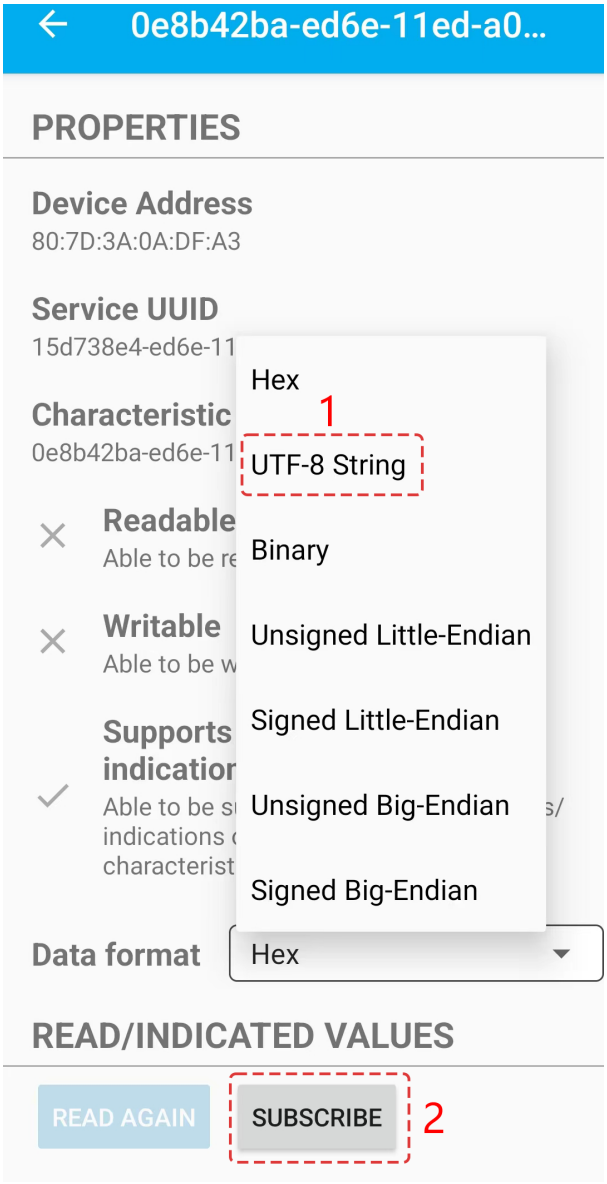
5. Nachdem der Code erfolgreich hochgeladen wurde, schalten Sie **Bluetooth** auf Ihrem mobilen Gerät ein und öffnen Sie die **LightBlue**-App.



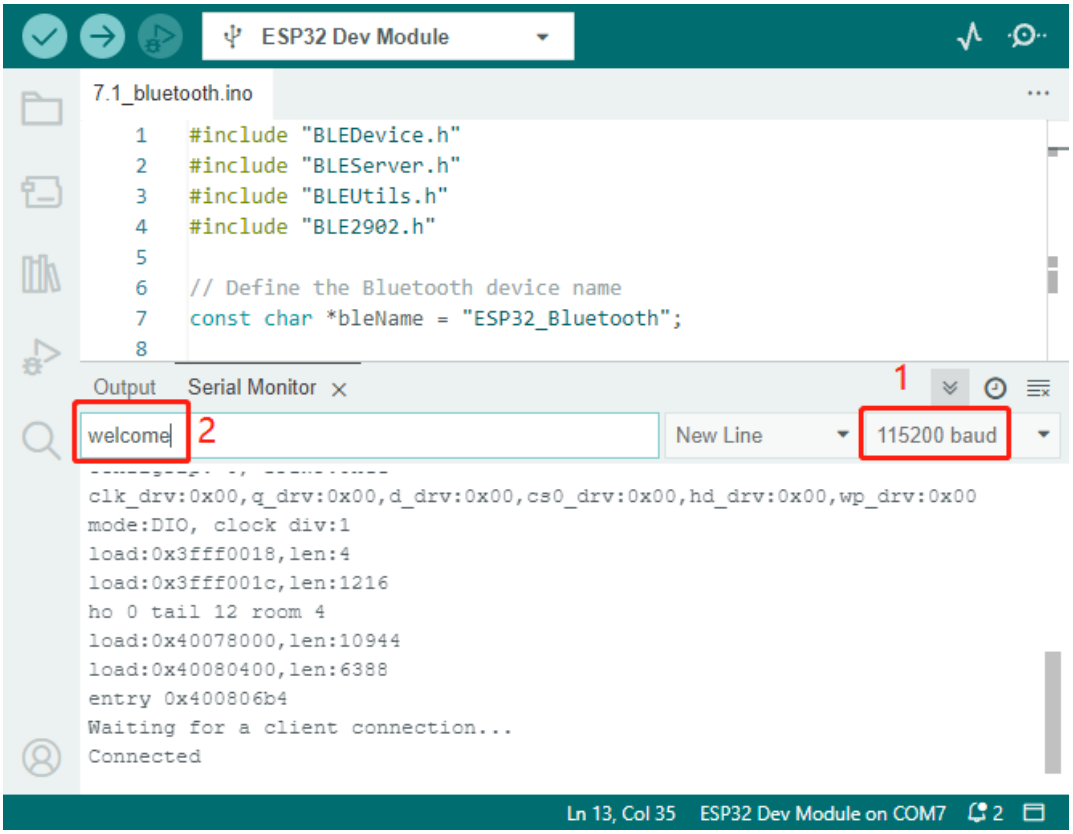
6. Auf der **Scan**-Seite finden Sie **ESP32-Bluetooth** und klicken Sie auf **CONNECT**. Wenn Sie es nicht sehen, versuchen Sie, die Seite einige Male zu aktualisieren. Wenn **„Connected to device!"** erscheint, ist die Bluetooth-Verbindung erfolgreich. Scrollen Sie nach unten, um die drei im Code eingestellten UUIDs zu sehen.



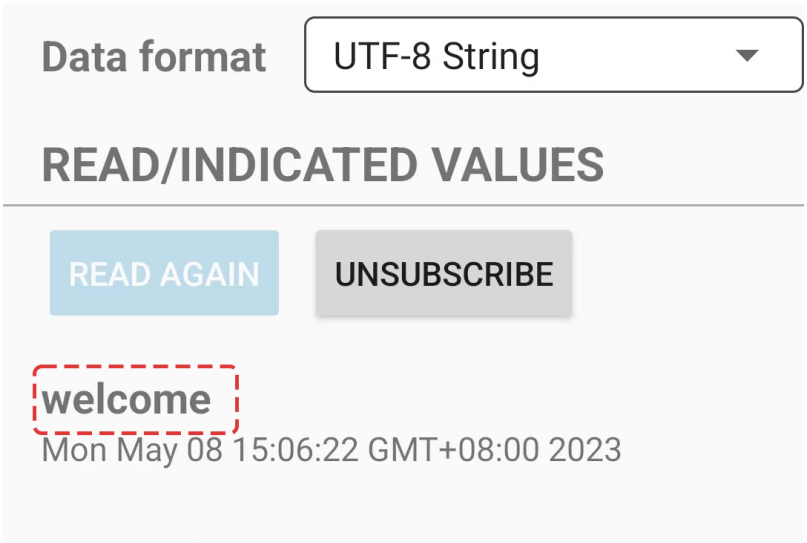
7. Klicken Sie auf die UUID **Receive**. Wählen Sie das entsprechende Datenformat im Kasten rechts neben **Data Format** aus, wie z.B. „HEX" für Hexadezimal, „UTF-8-String" für Zeichen oder „Binär" für binäre Daten usw. Dann klicken Sie auf **SUBSCRIBE**.



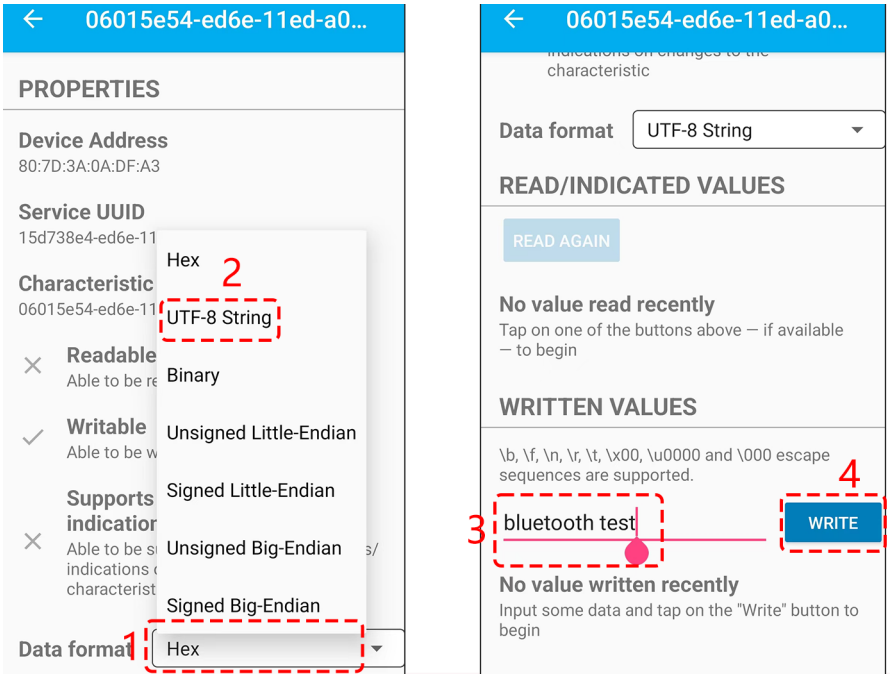
8. Gehen Sie zurück zur Arduino IDE, öffnen Sie den Seriellen Monitor, stellen Sie die Baudrate auf 115200 ein, tippen Sie „welcome“ und drücken Sie die Eingabetaste.



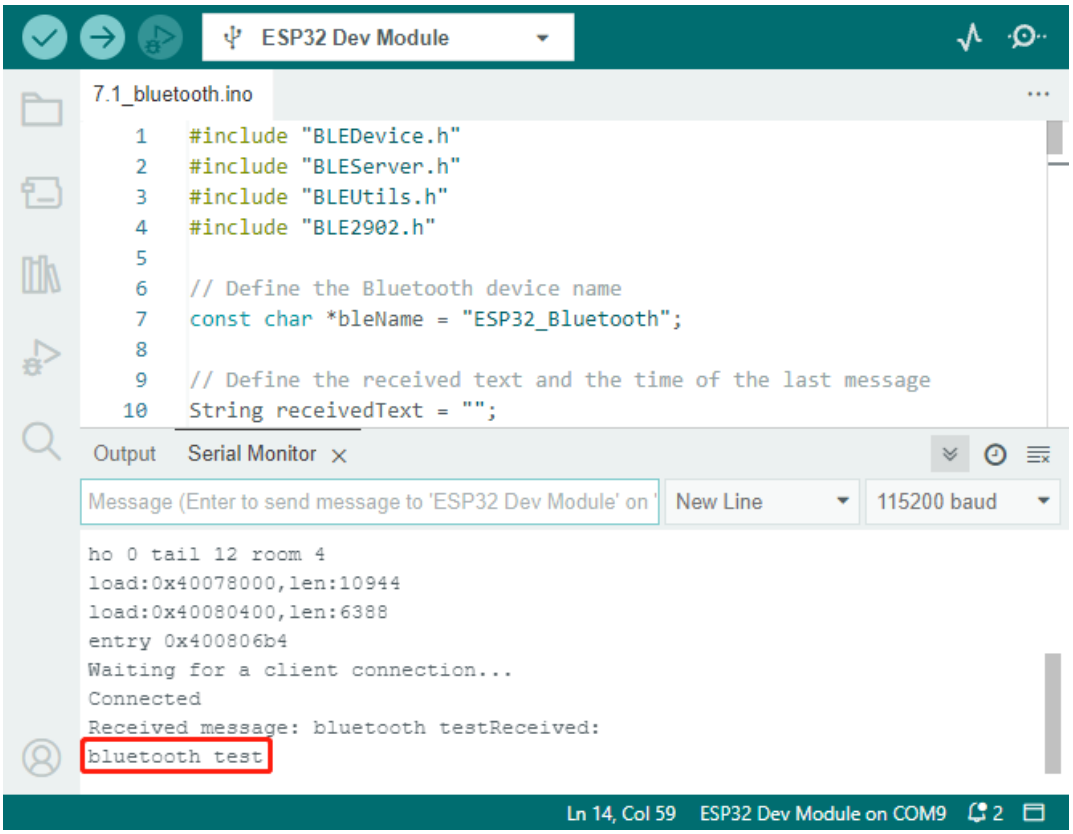
9. Sie sollten jetzt die „Willkommen“-Nachricht in der LightBlue-App sehen.



10. Um Informationen vom mobilen Gerät zum Seriellen Monitor zu senden, klicken Sie auf die UUID Senden, stellen Sie das Datenformat auf „UTF-8-String“ ein und schreiben Sie eine Nachricht.



11. Sie sollten die Nachricht im Seriellen Monitor sehen.



Wie funktioniert das?

Dieser Arduino-Code ist für den Mikrocontroller ESP32 geschrieben und richtet ihn für die Kommunikation mit einem Bluetooth Low Energy (BLE) Gerät ein.

Hier eine kurze Zusammenfassung des Codes:

- **Include necessary libraries**: Der Code beginnt mit dem Einbinden der notwendigen Bibliotheken für die Arbeit mit Bluetooth Low Energy (BLE) auf dem ESP32.

```
#include "BLEDevice.h"
#include "BLEServer.h"
#include "BLEUtils.h"
#include "BLE2902.h"
```

- **Global Variables**: Der Code definiert eine Reihe von globalen Variablen, einschließlich des Bluetooth-Gerätenamens (`bleName`), Variablen zur Überwachung empfangener Texte und der Zeit der letzten Nachricht, UUIDs für den Dienst und die Merkmale sowie ein `BLECharacteristic`-Objekt (`pCharacteristic`).

```
// Define the Bluetooth device name
const char *bleName = "ESP32_Bluetooth";

// Define the received text and the time of the last message
String receivedText = "";
unsigned long lastMessageTime = 0;

// Define the UUIDs of the service and characteristics
#define SERVICE_UUID           "your_service_uuid_here"
#define CHARACTERISTIC_UUID_RX "your_rx_characteristic_uuid_here"
#define CHARACTERISTIC_UUID_TX "your_tx_characteristic_uuid_here"

// Define the Bluetooth characteristic
BLECharacteristic *pCharacteristic;
```

- **Setup**: In der Funktion `setup()` wird der serielle Port mit einer Baudrate von 115200 initialisiert und die Funktion `setupBLE()` aufgerufen, um das Bluetooth BLE einzurichten.

```
void setup() {
   Serial.begin(115200);  // Initialize the serial port
    setupBLE();            // Initialize the Bluetooth BLE
}
```

- **Main Loop**: In der Funktion `loop()` wird, wenn ein String über BLE empfangen wurde (d.h. `receivedText` ist nicht leer) und seit der letzten Nachricht mindestens 1 Sekunde vergangen ist, der empfangene String im seriellen Monitor ausgegeben, der Merkmalswert auf den empfangenen String gesetzt, eine Benachrichtigung gesendet und dann der empfangene String gelöscht. Wenn Daten auf dem seriellen Port verfügbar sind, liest es den String bis zu einem Zeilenumbruch, setzt den Merkmalswert auf diesen String und sendet eine Benachrichtigung.

```
void loop() {
    // When the received text is not empty and the time since the last␣
↪message is over 1 second
    // Send a notification and print the received text
    if (receivedText.length() > 0 && millis() - lastMessageTime > 1000) {
        Serial.print("Received message: ");
        Serial.println(receivedText);
        pCharacteristic->setValue(receivedText.c_str());
        pCharacteristic->notify();
        receivedText = "";
    }
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    // Read data from the serial port and send it to BLE characteristic
    if (Serial.available() > 0) {
        String str = Serial.readStringUntil('\n');
        const char *newValue = str.c_str();
        pCharacteristic->setValue(newValue);
        pCharacteristic->notify();
    }
}
```

- **Callbacks**: Zwei Callback-Klassen (`MyServerCallbacks` und `MyCharacteristicCallbacks`) sind definiert, um Ereignisse im Zusammenhang mit der Bluetooth-Kommunikation zu behandeln. `MyServerCallbacks` wird verwendet, um Ereignisse im Zusammenhang mit dem Verbindungsstatus (verbunden oder getrennt) des BLE-Servers zu handhaben. `MyCharacteristicCallbacks` dient zur Behandlung von Schreibereignissen auf dem BLE-Merkmal, d.h., wenn ein verbundenes Gerät einen String über BLE an den ESP32 sendet, wird dieser erfasst und in `receivedText` gespeichert, und die aktuelle Zeit wird in `lastMessageTime` aufgezeichnet.

```
// Define the BLE server callbacks
class MyServerCallbacks : public BLEServerCallbacks {
    // Print the connection message when a client is connected
    void onConnect(BLEServer *pServer) {
    Serial.println("Connected");
    }
    // Print the disconnection message when a client is disconnected
    void onDisconnect(BLEServer *pServer) {
    Serial.println("Disconnected");
    }
};

// Define the BLE characteristic callbacks
class MyCharacteristicCallbacks : public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *pCharacteristic) {
        // When data is received, get the data and save it to receivedText,␣
↪and record the time
        std::string value = pCharacteristic->getValue();
        receivedText = String(value.c_str());
        lastMessageTime = millis();
        Serial.print("Received: ");
        Serial.println(receivedText);
    }
};
```

- **Setup BLE**: In der Funktion `setupBLE()` werden das BLE-Gerät und der Server initialisiert, die Server-Callbacks eingestellt, der BLE-Dienst mit der definierten UUID erstellt, Merkmale zum Senden von Benachrichtigungen und zum Empfangen von Daten erstellt und zum Dienst hinzugefügt, und die Merkmal-Callbacks eingestellt. Schließlich wird der Dienst gestartet und der Server beginnt mit der Werbung.

```
// Initialize the Bluetooth BLE
void setupBLE() {
    BLEDevice::init(bleName);                       // Initialize the BLE␣
↪device
    BLEServer *pServer = BLEDevice::createServer();  // Create the BLE␣
↪server
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    // Print the error message if the BLE server creation fails
    if (pServer == nullptr) {
        Serial.println("Error creating BLE server");
        return;
    }
    pServer->setCallbacks(new MyServerCallbacks());  // Set the BLE server␣
↪callbacks

    // Create the BLE service
    BLEService *pService = pServer->createService(SERVICE_UUID);
    // Print the error message if the BLE service creation fails
    if (pService == nullptr) {
        Serial.println("Error creating BLE service");
        return;
    }
    // Create the BLE characteristic for sending notifications
    pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID_TX,
↪ BLECharacteristic::PROPERTY_NOTIFY);
    pCharacteristic->addDecodeor(new BLE2902());  // Add the decodeor
    // Create the BLE characteristic for receiving data
    BLECharacteristic *pCharacteristicRX = pService->
↪createCharacteristic(CHARACTERISTIC_UUID_RX, BLECharacteristic::PROPERTY_
↪WRITE);
    pCharacteristicRX->setCallbacks(new MyCharacteristicCallbacks());  //␣
↪Set the BLE characteristic callbacks
    pService->start();                                                 //␣
↪Start the BLE service
    pServer->getAdvertising()->start();                                //␣
↪Start advertising
    Serial.println("Waiting for a client connection...");              //␣
↪Wait for a client connection
}
```

Bitte beachten Sie, dass dieser Code eine bidirektionale Kommunikation ermöglicht - er kann Daten über BLE senden und empfangen. Um jedoch mit spezifischer Hardware wie dem Ein- und Ausschalten einer LED zu interagieren, sollte zusätzlicher Code hinzugefügt werden, um die empfangenen Strings zu verarbeiten und entsprechend zu handeln.

2.39 7.2 Bluetooth-Steuerung RGB-LED

Dieses Projekt ist eine Erweiterung eines vorherigen Projekts (*7.1 Bluetooth*), bei dem RGB-LED-Konfigurationen und benutzerdefinierte Befehle wie „led_off“, „rot“, „grün“ usw. hinzugefügt wurden. Diese Befehle ermöglichen die Steuerung der RGB-LED durch Senden von Befehlen von einem mobilen Gerät über LightBlue.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

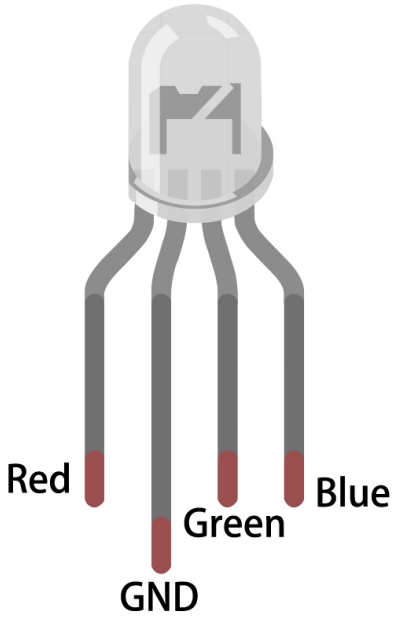
Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

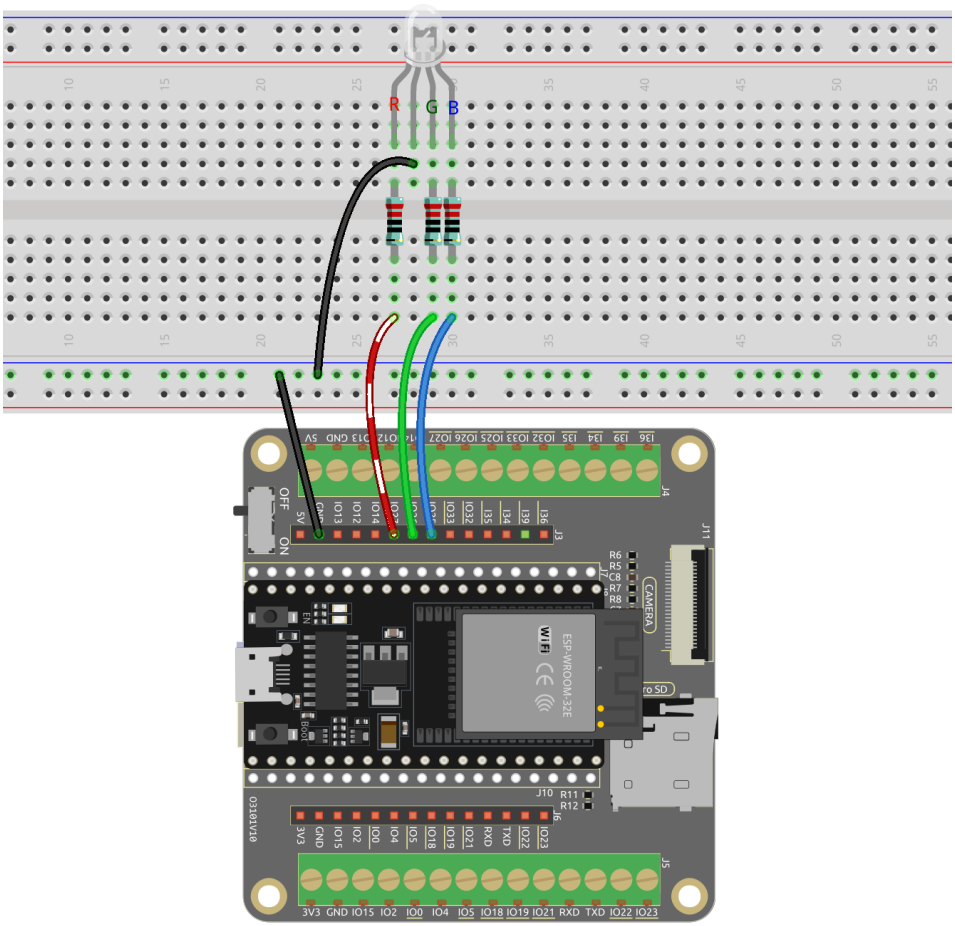
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
RGB LED	

Betriebsschritte

1. Bauen Sie den Schaltkreis auf.



Die RGB-LED hat 4 Pins: der längste Pin ist der gemeinsame Kathodenpin, der normalerweise mit GND verbunden ist; der linke Pin neben dem längsten Pin ist Rot; und die beiden Pins rechts sind Grün und Blau.

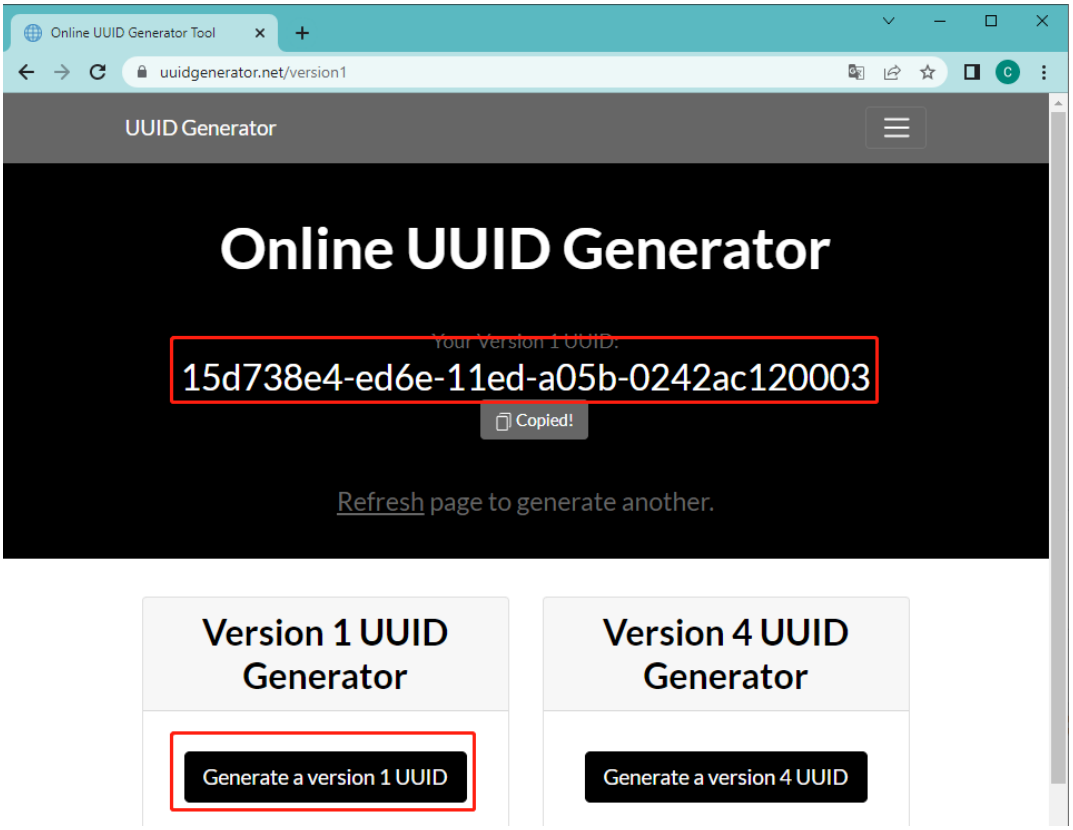


2. Öffnen Sie die Datei `7.2_bluetooth_rgb_led.ino` im Verzeichnis `esp32-starter-kit-main\c\codes\7.2_bluetooth_rgb_led` oder kopieren Sie den Code in die Arduino IDE.

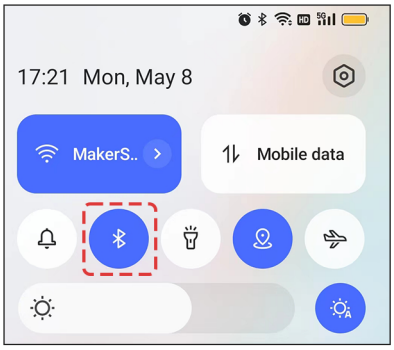
3. Um UUID-Konflikte zu vermeiden, wird empfohlen, drei neue UUIDs mit dem , bereitgestellt von der Bluetooth SIG, zufällig zu generieren und sie in den folgenden Codezeilen einzufügen.

Bemerkung: Wenn Sie bereits drei neue UUIDs im *7.1 Bluetooth* Projekt generiert haben, dann können Sie diese weiterhin verwenden.

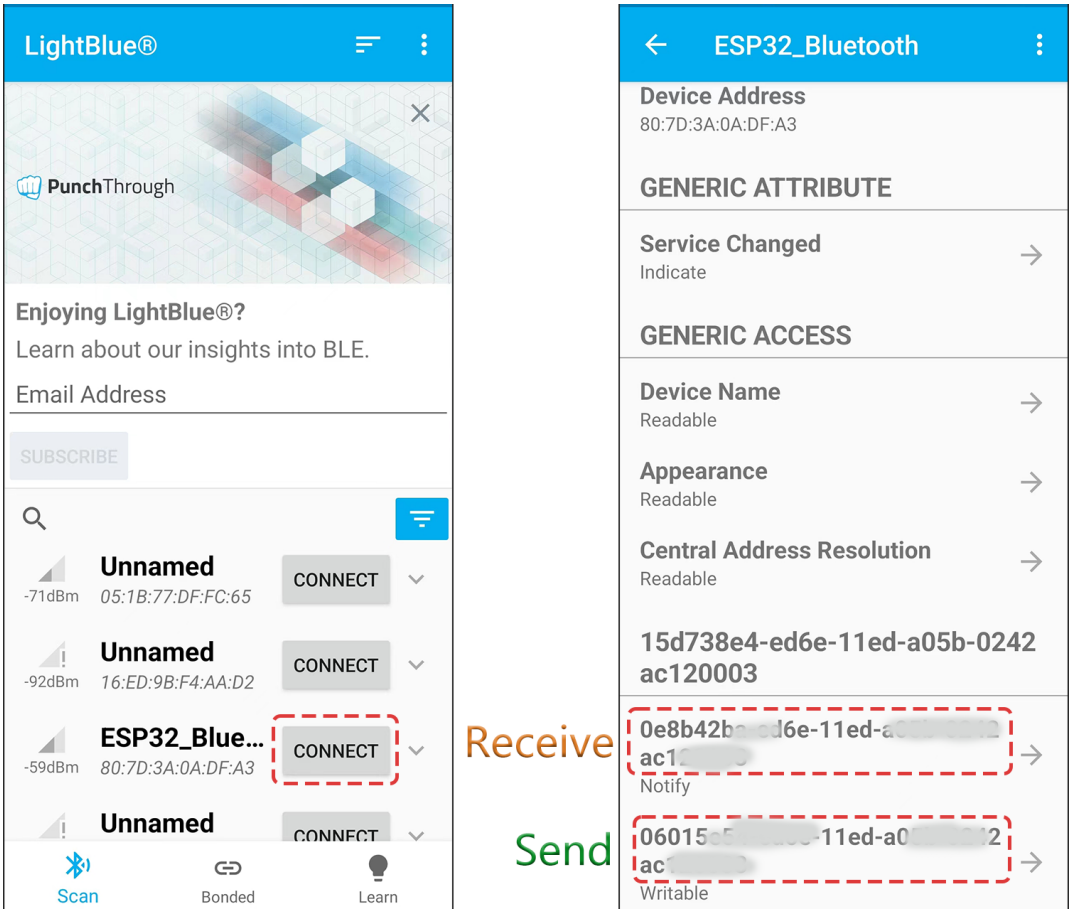
```
#define SERVICE_UUID           "your_service_uuid_here"
#define CHARACTERISTIC_UUID_RX "your_rx_characteristic_uuid_here"
#define CHARACTERISTIC_UUID_TX "your_tx_characteristic_uuid_here"
```



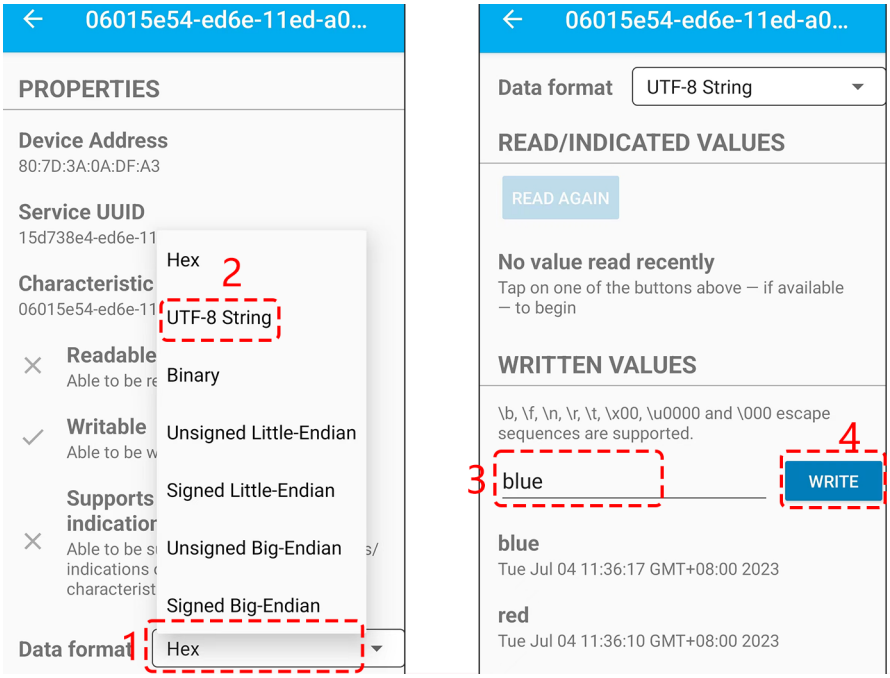
4. Wählen Sie das richtige Board und den richtigen Port aus, dann klicken Sie auf die **Upload**-Taste.
5. Nachdem der Code erfolgreich hochgeladen wurde, schalten Sie **Bluetooth** auf Ihrem mobilen Gerät ein und öffnen Sie die **LightBlue**-App.



6. Auf der **Scan**-Seite finden Sie **ESP32-Bluetooth** und klicken Sie auf **CONNECT**. Wenn Sie es nicht sehen, versuchen Sie, die Seite einige Male zu aktualisieren. Wenn **„Connected to device!“** erscheint, ist die Bluetooth-Verbindung erfolgreich. Scrollen Sie nach unten, um die drei im Code eingestellten UUIDs zu sehen.



7. Tippen Sie auf die Senden-UUID, dann stellen Sie das Datenformat auf „UTF-8-String“ ein. Jetzt können Sie diese Befehle schreiben: „led_off“, „rot“, „grün“, „blau“, „gelb“ und „lila“, um zu sehen, ob die RGB-LED auf diese Anweisungen reagiert.



Wie funktioniert das?

Dieser Code ist eine Erweiterung eines vorherigen Projekts (*7.1 Bluetooth*), bei dem RGB-LED-Konfigurationen und benutzerdefinierte Befehle wie „led_off“, „rot“, „grün“ usw. hinzugefügt wurden. Diese Befehle ermöglichen die Steuerung der RGB-LED durch Senden von Befehlen von einem mobilen Gerät über LightBlue.

Lassen Sie uns den Code Schritt für Schritt durchgehen:

- Fügen Sie neue globale Variablen für die RGB-LED-Pins, PWM-Kanäle, Frequenz und Auflösung hinzu.

```
...

// Define RGB LED pins
const int redPin = 27;
const int greenPin = 26;
const int bluePin = 25;

// Define PWM channels
const int redChannel = 0;
const int greenChannel = 1;
const int blueChannel = 2;

...
```

- Innerhalb der Funktion `setup()` werden die PWM-Kanäle mit der vordefinierten Frequenz und Auflösung initialisiert. Die RGB-LED-Pins werden dann ihren jeweiligen PWM-Kanälen zugeordnet.

```
void setup() {
    ...

    // Set up PWM channels
    ledcSetup(redChannel, freq, resolution);
    ledcSetup(greenChannel, freq, resolution);
    ledcSetup(blueChannel, freq, resolution);

    // Attach pins to corresponding PWM channels
    ledcAttachPin(redPin, redChannel);
    ledcAttachPin(greenPin, greenChannel);
    ledcAttachPin(bluePin, blueChannel);

}
```

- Modifizieren Sie die Methode `onWrite` in der Klasse `MyCharacteristicCallbacks`. Diese Funktion hört auf Daten, die von der Bluetooth-Verbindung kommen. Basierend auf dem empfangenen String (wie `"led_off"`, `"red"`, `"green"`, usw.) steuert sie die RGB-LED.

```
// Define the BLE characteristic callbacks
class MyCharacteristicCallbacks : public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *pCharacteristic) {
        std::string value = pCharacteristic->getValue();
        if (value == "led_off") {
            setColor(0, 0, 0); // turn the RGB LED off
            Serial.println("RGB LED turned off");
        } else if (value == "red") {
            setColor(255, 0, 0); // Red
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
            Serial.println("red");
        }
        else if (value == "green") {
            setColor(0, 255, 0); // green
            Serial.println("green");
        }
        else if (value == "blue") {
            setColor(0, 0, 255); // blue
            Serial.println("blue");
        }
        else if (value == "yellow") {
            setColor(255, 150, 0); // yellow
            Serial.println("yellow");
        }
        else if (value == "purple") {
            setColor(80, 0, 80); // purple
            Serial.println("purple");
        }
    }
};
```

- Schließlich wird eine Funktion hinzugefügt, um die Farbe der RGB-LED einzustellen.

```
void setColor(int red, int green, int blue) {
    // For common-anode RGB LEDs, use 255 minus the color value
    ledcWrite(redChannel, red);
    ledcWrite(greenChannel, green);
    ledcWrite(blueChannel, blue);
}
```

Zusammenfassend ermöglicht dieses Skript ein Fernsteuerungs-Interaktionsmodell, bei dem der ESP32 als Bluetooth Low Energy (BLE)-Server fungiert.

Der verbundene BLE-Client (wie ein Smartphone) kann String-Befehle senden, um die Farbe einer RGB-LED zu ändern. Der ESP32 gibt auch Feedback an den Client, indem er den empfangenen String zurücksendet, sodass der Client weiß, welche Operation durchgeführt wurde.

2.40 7.3 Bluetooth-Audio-Player

Das Ziel des Projekts ist es, eine einfache Lösung für das Abspielen von Audio von einem Bluetooth-fähigen Gerät mithilfe des eingebauten DAC des ESP32 bereitzustellen.

Das Projekt umfasst die Verwendung der `ESP32-A2DP`-Bibliothek, um Audiodaten von einem Bluetooth-fähigen Gerät zu empfangen. Die empfangenen Audiodaten werden dann an den internen DAC des ESP32 über die I2S-Schnittstelle übertragen. Die I2S-Schnittstelle ist konfiguriert, um im Master-Modus, Sendemodus und mit eingebautem DAC zu arbeiten. Die Audiodaten werden dann über den Lautsprecher, der an den DAC angeschlossen ist, wiedergegeben.

Beim Verwenden des internen DAC des ESP32 ist zu beachten, dass das Ausgangsspannungsniveau auf 1,1 V begrenzt ist. Daher wird empfohlen, einen externen Verstärker zu verwenden, um das Ausgangsspannungsniveau auf das gewünschte Niveau zu erhöhen. Es ist auch wichtig, sicherzustellen, dass die Audiodaten im richtigen Format und mit der richtigen Abtastrate vorliegen, um Verzerrungen oder Geräusche während der Wiedergabe zu vermeiden.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Audio-Modul und Lautsprecher	-

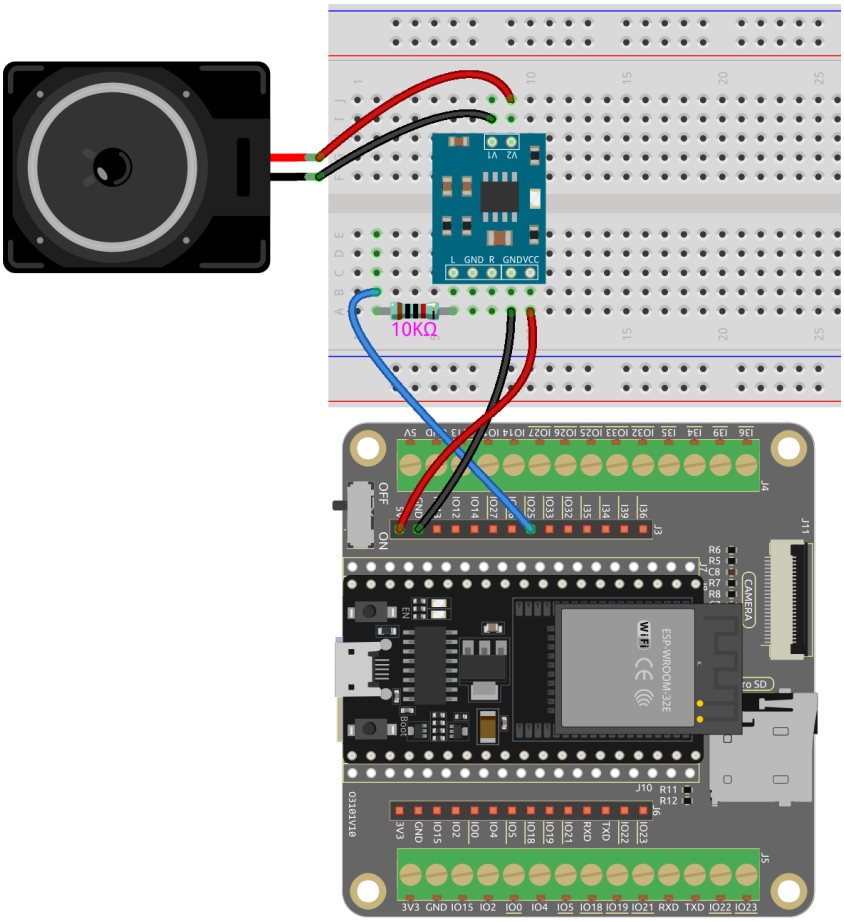
Betriebsschritte

1. Bauen Sie den Schaltkreis auf.

   Da dies ein Mono-Verstärker ist, können Sie IO25 an den L- oder R-Pin des Audioverstärkermoduls anschließen.

   Der 10K-Widerstand wird verwendet, um hochfrequentes Rauschen zu reduzieren und die Lautstärke des Audios zu verringern. Er bildet zusammen mit der parasitären Kapazität des DAC und des Audioverstärkers einen RC-Tiefpassfilter. Dieser Filter verringert die Amplitude von Hochfrequenzsignalen und reduziert so effektiv hochfrequentes Rauschen. Das Hinzufügen des 10K-Widerstands macht die Musik weicher und eliminiert unerwünschtes Hochfrequenzrauschen.

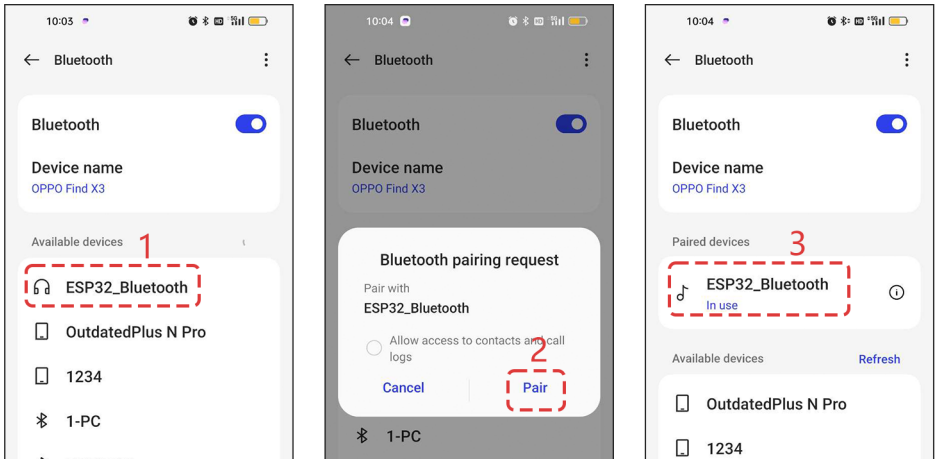
   Wenn die Musik auf Ihrer SD-Karte bereits leise ist, können Sie den Widerstand entfernen oder durch einen kleineren Wert ersetzen.



2. Öffnen Sie den Code.

    - Öffnen Sie die Datei `7.3_bluetooth_audio_player.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\7.3_bluetooth_audio_player`.
    - Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
    - *„Unbekanntes COMxx“ wird immer angezeigt?*
    - Hier wird die `ESP32-A2DP`-Bibliothek verwendet, siehe *Manuelle Installation* für eine Anleitung zur Installation.

3. Nachdem Sie das richtige Board und den Port ausgewählt haben, klicken Sie auf den Hochladen-Button.
4. Sobald der Code erfolgreich hochgeladen wurde, schalten Sie das Bluetooth-fähige Gerät ein und suchen Sie nach verfügbaren Geräten, dann verbinden Sie sich mit dem `ESP32_Bluetooth`.



5. Spielen Sie Audio auf dem Gerät ab, und der Ton sollte über den Lautsprecher, der an den ESP32 angeschlossen ist, wiedergegeben werden.

Code-Erklärung

1. Der Code beginnt mit dem Einbinden der Bibliothek `BluetoothA2DPSink.h`, die verwendet wird, um Audiodaten von dem Bluetooth-fähigen Gerät zu empfangen. Das `BluetoothA2DPSink`-Objekt wird dann erstellt und mit den Einstellungen der I2S-Schnittstelle konfiguriert.

```
#include "BluetoothA2DPSink.h"

BluetoothA2DPSink a2dp_sink;
```

2. In der Setup-Funktion initialisiert der Code eine `i2s_config_t struct` mit der gewünschten Konfiguration für die I2S (Inter-IC Sound)-Schnittstelle.

```
void setup() {
const i2s_config_t i2s_config = {
    .mode = (i2s_mode_t) (I2S_MODE_MASTER | I2S_MODE_TX | I2S_MODE_DAC_
↪BUILT_IN),
    .sample_rate = 44100, // corrected by info from bluetooth
    .bits_per_sample = (i2s_bits_per_sample_t) 16, // the DAC module will␣
↪only take the 8bits from MSB
    .channel_format =  I2S_CHANNEL_FMT_RIGHT_LEFT,
    .communication_format = (i2s_comm_format_t)I2S_COMM_FORMAT_STAND_MSB,
    .intr_alloc_flags = 0, // default interrupt priority
    .dma_buf_count = 8,
    .dma_buf_len = 64,
    .use_apll = false
};

a2dp_sink.set_i2s_config(i2s_config);
a2dp_sink.start("ESP32_Bluetooth");

}
```

- Die I2S-Schnittstelle wird verwendet, um digitale Audiodaten zwischen Geräten zu übertragen.
- Die Konfiguration umfasst den `I2S mode`, `sample rate`, `bits per sample`, `channel format`, `communication format`, `interrupt allocation flags`, `DMA buffer count`, `DMA buffer length` und ob der APLL (Audio PLL) verwendet wird oder nicht.

- Die `i2s_config_t struct` wird dann als Argument an die `set_i2s_config`-Funktion des `BluetoothA2DPSink`-Objekts übergeben, um die I2S-Schnittstelle für die Audiowiedergabe zu konfigurieren.
- Die `start`-Funktion des `BluetoothA2DPSink`-Objekts wird aufgerufen, um den Bluetooth-Audioempfänger zu starten und die Audiowiedergabe über den eingebauten DAC zu beginnen.

2.41 7.4 SD-Karten Schreiben und Lesen

Dieses Projekt demonstriert die Kernfähigkeiten der Verwendung einer SD-Karte mit dem Mikrocontroller ESP32. Es zeigt wesentliche Operationen wie das Einbinden der SD-Karte, das Erstellen einer Datei, das Schreiben von Daten in die Datei und das Auflisten aller Dateien im Root-Verzeichnis. Diese Operationen bilden die Grundlage vieler Datenprotokollierungs- und Speicheranwendungen und machen dieses Projekt zu einem entscheidenden Schritt im Verständnis und in der Nutzung des eingebauten SDMMC-Host-Peripheriegeräts des ESP32.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

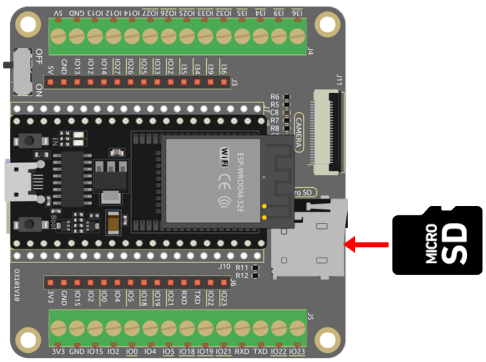
Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

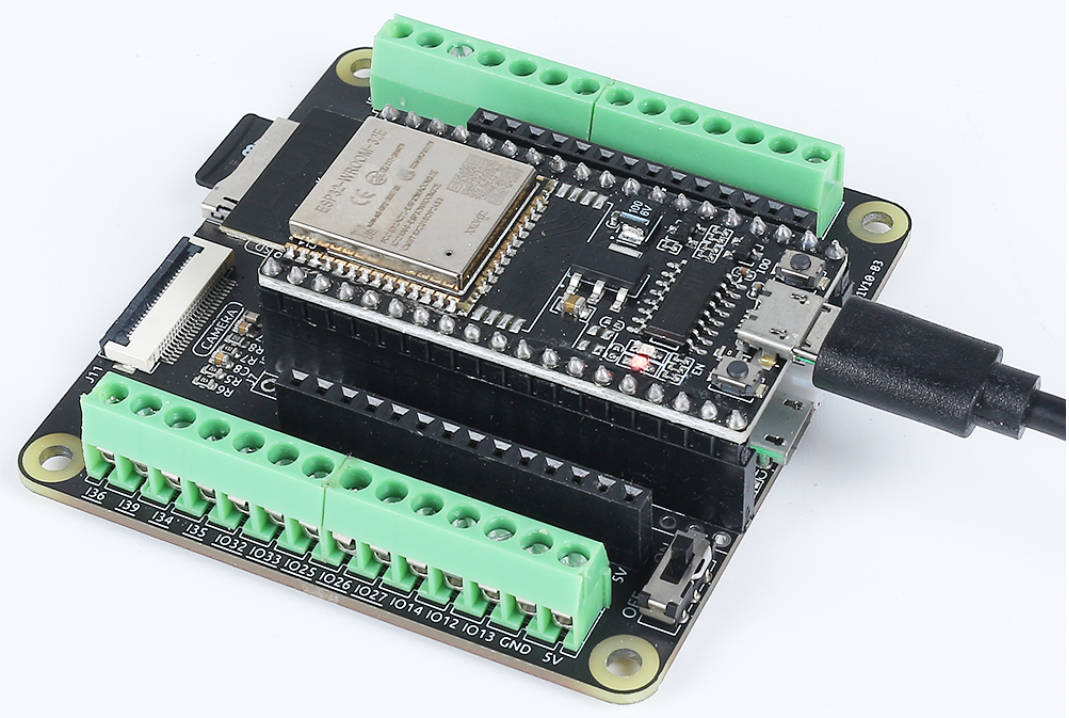
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-

Betriebsschritte

1. Bevor Sie das USB-Kabel anschließen, stecken Sie die SD-Karte in den SD-Kartenslot des Erweiterungsboards.



2. Verbinden Sie ESP32-WROOM-32E mit dem Computer über das USB-Kabel.



3. Wählen Sie den passenden Port und das Board in der Arduino IDE aus und laden Sie den Code auf Ihren ESP32.

Bemerkung:

- Öffnen Sie die Datei `7.4_sd_read_write.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\7.4_sd_read_write`.
- Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

---

4. Nachdem der Code erfolgreich hochgeladen wurde, sehen Sie eine Aufforderung, die auf den erfolgreichen Dateischreibvorgang hinweist, zusammen mit einer Liste aller Dateinamen und -größen auf der SD-Karte. Wenn Sie nach dem Öffnen des seriellen Monitors keine Ausgabe sehen, müssen Sie den EN (RST)-Knopf drücken, um den Code erneut auszuführen.

Bemerkung: Wenn Sie folgende Informationen sehen.

```
E (528) vfs_fat_sdmmc: mount_to_vfs failed (0xffffffff).
Failed to mount SD card
```

Überprüfen Sie zuerst, ob Ihre SD-Karte richtig in das Erweiterungsboard eingelegt ist.

Wenn sie richtig eingelegt ist, könnte es ein Problem mit Ihrer SD-Karte geben. Sie können versuchen, die Metallkontakte mit einem Radiergummi zu reinigen.

Wenn das Problem weiterhin besteht, wird empfohlen, die SD-Karte zu formatieren. Bitte beachten Sie *Wie formatiert man eine SD-Karte?*.

Wie funktioniert das?

Der Zweck dieses Projekts ist es, die Verwendung einer SD-Karte mit dem ESP32-Board zu demonstrieren. Das eingebaute SDMMC-Host-Peripheriegerät des ESP32 wird verwendet, um eine Verbindung mit der SD-Karte herzustellen.

Das Projekt beginnt mit der Initialisierung der seriellen Kommunikation und versucht dann, die SD-Karte einzubinden. Wenn das Einbinden der SD-Karte nicht erfolgreich ist, druckt das Programm eine Fehlermeldung und beendet die Setup-Funktion.

Sobald die SD-Karte erfolgreich eingebunden ist, erstellt das Programm eine Datei mit dem Namen „test.txt“ im Root-Verzeichnis der SD-Karte. Wenn die Datei erfolgreich im Schreibmodus geöffnet wird, schreibt das Programm eine Zeile Text - „Hello, world!“ in die Datei. Das Programm druckt eine Erfolgsmeldung, wenn der Schreibvorgang erfolgreich ist, andernfalls wird eine Fehlermeldung gedruckt.

Nachdem der Schreibvorgang abgeschlossen ist, schließt das Programm die Datei und öffnet dann das Root-Verzeichnis der SD-Karte. Anschließend beginnt es, alle Dateien im Root-Verzeichnis zu durchlaufen und druckt den Dateinamen und die Dateigröße jeder gefundenen Datei.

In der Hauptschleifenfunktion gibt es keine Operationen. Dieses Projekt konzentriert sich auf SD-Kartenoperationen wie das Einbinden der Karte, das Erstellen einer Datei, das Schreiben in eine Datei und das Lesen des Dateiverzeichnisses, die alle in der Setup-Funktion ausgeführt werden.

Dieses Projekt dient als nützliche Einführung in den Umgang mit SD-Karten mit dem ESP32, was in Anwendungen, die Datenprotokollierung oder Speicherung erfordern, entscheidend sein kann.

Hier ist eine Analyse des Codes:

1. Binden Sie die SD_MMC-Bibliothek ein, die benötigt wird, um mit SD-Karten mit dem eingebauten SDMMC-Host-Peripheriegerät des ESP32 zu arbeiten.

```
#include "SD_MMC.h"
```

2. Innerhalb der `setup()`-Funktion werden die folgenden Aufgaben ausgeführt.

    - **Initialisieren Sie die SD-Karte**

    Initialisieren und binden Sie die SD-Karte ein. Wenn das Einbinden der SD-Karte fehlschlägt, druckt sie „Failed to mount SD card“ auf den seriellen Monitor und stoppt die Ausführung.

```
if(!SD_MMC.begin()) { // Attempt to mount the SD card
    Serial.println("Failed to mount card"); // If mount fails, print to␣
↪serial and exit setup
    return;
}
```

    - **Öffnen Sie die Datei**

    Öffnen Sie eine Datei mit dem Namen `"test.txt"` im Root-Verzeichnis der SD -Karte im Schreibmodus. Wenn die Datei nicht geöffnet werden kann, druckt sie „Failed to open file for writing“ und kehrt zurück.

```
File file = SD_MMC.open("/test.txt", FILE_WRITE);
if (!file) {
    Serial.println("Failed to open file for writing"); // Print error␣
↪message if file failed to open
    return;
}
```

    - **Schreiben Sie Daten in die Datei**

    Schreiben Sie den Text „Test file write“ in die Datei. Wenn der Schreibvorgang erfolgreich ist, druckt sie „File write successful“; andernfalls druckt sie „File write failed“.

```
if(file.print("Test file write")) { // Write the message to the file
    Serial.println("File write success"); // If write succeeds, print to␣
↪serial
} else {
    Serial.println("File write failed"); // If write fails, print to serial
}
```

    - **Schließen Sie die Datei**

    Schließen Sie die geöffnete Datei. Dadurch wird sichergestellt, dass alle gepufferten Daten in die Datei geschrieben und die Datei ordnungsgemäß geschlossen wird.

```
file.close(); // Close the file
```

    - **Öffnen Sie das Root-Verzeichnis**

    Öffnen Sie das Root-Verzeichnis der SD-Karte. Wenn das Verzeichnis nicht geöffnet werden kann, druckt es „Failed to open directory“ und kehrt zurück.

```
File root = SD_MMC.open("/"); // Open the root directory of SD card
if (!root) {
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    Serial.println("Failed to open directory"); // Print error message if␣
↪directory failed to open
    return;
}
```

- **Drucken Sie den Namen und die Größe jeder Datei**

Die Schleife, die mit while (`File file = root.openNextFile()`) beginnt, iteriert über alle Dateien im Root-Verzeichnis und druckt den Namen und die Größe jeder Datei auf den seriellen Monitor.

```
Serial.println("Files found in root directory:"); // Print the list of␣
↪files found in the root directory
while (File file = root.openNextFile()) { // Loop through all the files in␣
↪the root directory
      Serial.print("  ");
      Serial.print(file.name()); // Print the filename
      Serial.print("\t");
      Serial.println(file.size()); // Print the filesize
      file.close(); // Close the file
}
```

3. Diese `loop()`-Funktion ist eine leere Schleife und tut in diesem Programm nichts. Allerdings würde in einem typischen Arduino-Programm diese Funktion kontinuierlich überlaufen und den Code in ihr ausführen. In diesem Fall sind jedoch alle erforderlichen Aufgaben in der Setup-Funktion ausgeführt worden, daher wird die Loop-Funktion nicht benötigt.

```
void loop() {} // Empty loop function, does nothing
```

2.42 7.5 MP3-Player mit SD-Kartenunterstützung

Willkommen in der aufregenden Welt der Musik mit Ihrem ESP32! Dieses Projekt bringt die Kraft der Audiobearbeitung in Ihre Hände und macht Ihren ESP32 nicht nur zu einem erstaunlichen Mikrocontroller für Computing, sondern auch zu Ihrem persönlichen Musikplayer. Stellen Sie sich vor, Sie betreten Ihr Zimmer und Ihr Lieblingslied spielt direkt von diesem kleinen Gerät. Das ist die Kraft, die wir Ihnen heute in die Hände legen.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

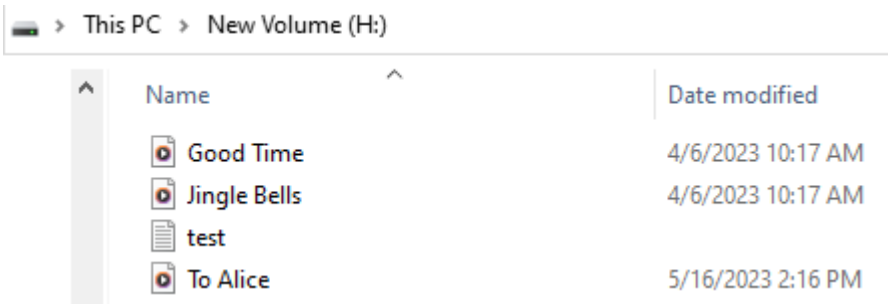
Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

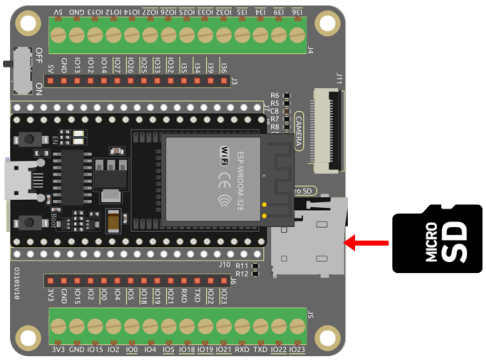
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Audio-Modul und Lautsprecher	-

Betriebsschritte

1. Stecken Sie Ihre SD-Karte in einen Kartenleser am Computer und formatieren Sie sie. Sie können sich auf das Tutorial unter *Wie formatiert man eine SD-Karte?* beziehen.
2. Kopieren Sie Ihre Lieblings-MP3-Datei auf Ihre SD-Karte.



3. Stecken Sie die SD-Karte in den SD-Kartenslot des Erweiterungsboards.

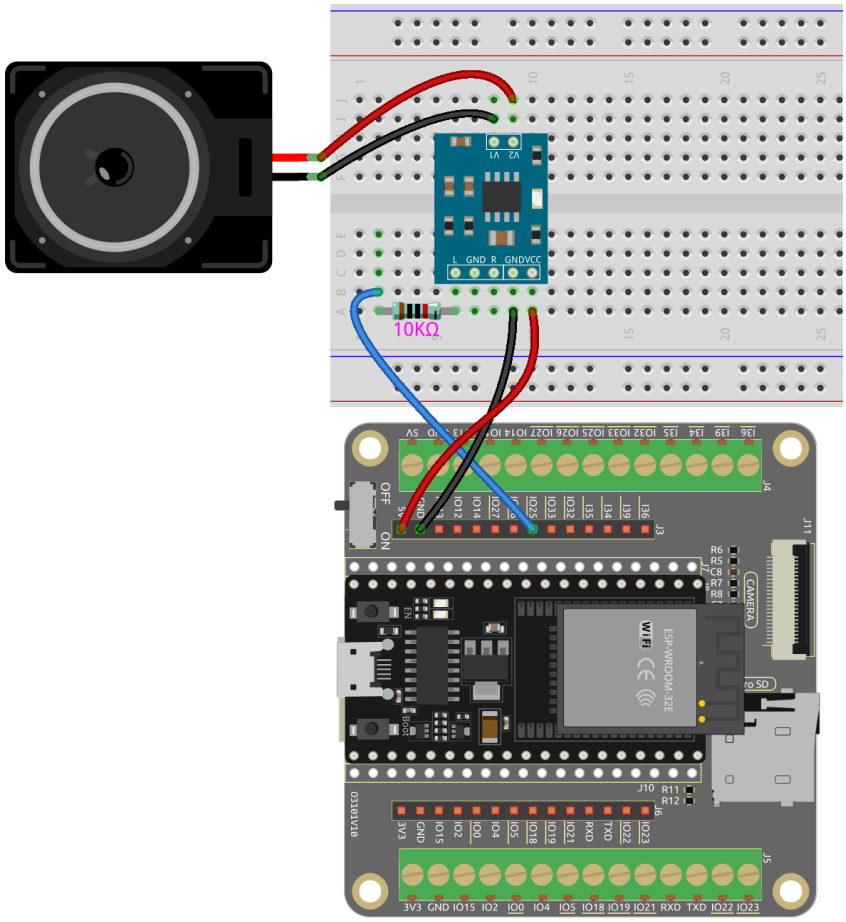


4. Bauen Sie den Schaltkreis auf.

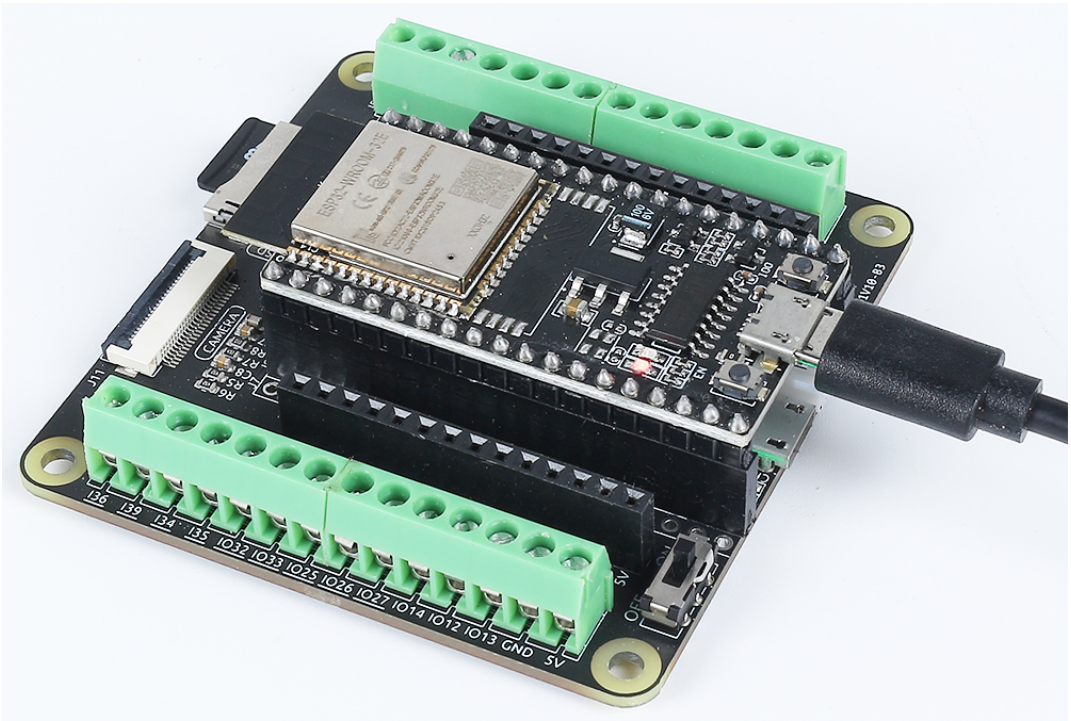
   Da dies ein Mono-Verstärker ist, können Sie IO25 an den L- oder R-Pin des Audioverstärkermoduls anschließen.

   Der 10K-Widerstand wird verwendet, um hochfrequentes Rauschen zu reduzieren und die Lautstärke des Audios zu verringern. Er bildet zusammen mit der parasitären Kapazität des DAC und des Audioverstärkers einen RC-Tiefpassfilter. Dieser Filter verringert die Amplitude von Hochfrequenzsignalen und reduziert so effektiv hochfrequentes Rauschen. Das Hinzufügen des 10K-Widerstands macht die Musik weicher und eliminiert unerwünschtes Hochfrequenzrauschen.

   Wenn die Musik auf Ihrer SD-Karte bereits leise ist, können Sie den Widerstand entfernen oder durch einen kleineren Wert ersetzen.



5. Verbinden Sie ESP32-WROOM-32E mit dem Computer über das USB-Kabel.



6. Ändern Sie den Code.

   Ändern Sie die Codezeile `file = new AudioFileSourceSD_MMC("/To Alice.mp3");` um den Namen und Pfad Ihrer Datei widerzuspiegeln.

   Bemerkung:

   - Öffnen Sie die Datei `7.5_mp3_player_sd.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\7.5_mp3_player_sd`. Oder kopieren Sie diesen Code in die **Arduino IDE**.
   - Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
   - *„Unbekanntes COMxx“ wird immer angezeigt?*
   - Hier wird die `ESP8266Audio`-Bibliothek verwendet, siehe *Manuelle Installation* für eine Anleitung zur Installation.

   ---

7. Wählen Sie den passenden Port und das Board in der Arduino IDE aus und laden Sie den Code auf Ihren ESP32.
8. Nachdem der Code erfolgreich hochgeladen wurde, hören Sie Ihre Lieblingsmusik spielen.

Wie funktioniert das?

- Der Code verwendet mehrere Klassen aus der `ESP8266Audio`-Bibliothek, um eine MP3-Datei von einer SD-Karte über I2S abzuspielen:

```
#include "AudioFileSourceSD_MMC.h"
#include "AudioOutputI2S.h"
#include "AudioGeneratorMP3.h"
#include "SD_MMC.h"
#include "FS.h"
```

  - `AudioGeneratorMP3` ist eine Klasse, die MP3-Audio dekodiert.
  - `AudioFileSourceSD_MMC` ist eine Klasse, die Audiodaten von einer SD-Karte liest.
  - `AudioOutputI2S` ist eine Klasse, die Audiodaten an die I2S-Schnittstelle sendet.

- In der `setup()`-Funktion initialisieren wir die SD-Karte, öffnen die MP3-Datei von der SD-Karte, richten den I2S-Ausgang am internen DAC des ESP32 ein, stellen den Ausgang auf Mono ein und starten den MP3-Generator.

```
void setup() {
    // Start the serial communication.
    Serial.begin(115200);
    delay(1000);

    // Initialize the SD card. If it fails, print an error message.
    if (!SD_MMC.begin()) {
        Serial.println("SD card mount failed!");
    }

    // Open the MP3 file from the SD card. Replace "/To Alice.mp3" with␣
↪your own MP3 file name.
    file = new AudioFileSourceSD_MMC("/To Alice.mp3");

    // Set up the I2S output on ESP32's internal DAC.
    out = new AudioOutputI2S(0, 1);
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    // Set the output to mono.
    out->SetOutputModeMono(true);

    // Initialize the MP3 generator with the file and output.
    mp3 = new AudioGeneratorMP3();
    mp3->begin(file, out);
}
```

- In der `loop()`-Funktion überprüfen wir, ob der MP3-Generator läuft. Wenn ja, führen wir ihn weiter aus; andernfalls stoppen wir ihn und drucken „MP3 fertig“ auf den seriellen Monitor.

```
void loop() {
    // If the MP3 is running, loop it. Otherwise, stop it.
    if (mp3->isRunning()) {
        if (!mp3->loop()) mp3->stop();
    }
    // If the MP3 is not running, print a message and wait for 1 second.
    else {
        Serial.println("MP3 done");
        delay(1000);
    }
}
```

2.43 7.6 Foto auf SD-Karte speichern

Dieses Dokument beschreibt ein Projekt, das darin besteht, ein Foto mit dem ESP32-CAM zu machen und es auf einer SD-Karte zu speichern. Das Ziel des Projekts ist es, eine einfache Lösung für das Aufnehmen von Bildern mit dem ESP32-CAM und deren Speicherung auf einer SD-Karte bereitzustellen.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-

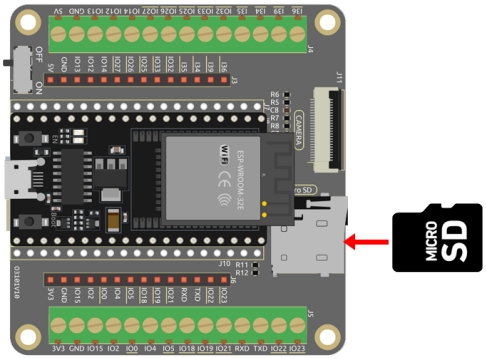
Verwandte Vorsichtsmaßnahmen

Beim Einsatz des ESP32-CAM ist zu beachten, dass der GPIO 0-Pin mit GND verbunden sein muss, um einen Sketch hochzuladen. Außerdem muss nach der Verbindung von GPIO 0 mit GND der RESET-Knopf des ESP32-CAM gedrückt

werden, um das Board in den Flash-Modus zu versetzen. Es ist auch wichtig sicherzustellen, dass die SD-Karte korrekt eingebunden ist, bevor Bilder darauf gespeichert werden.

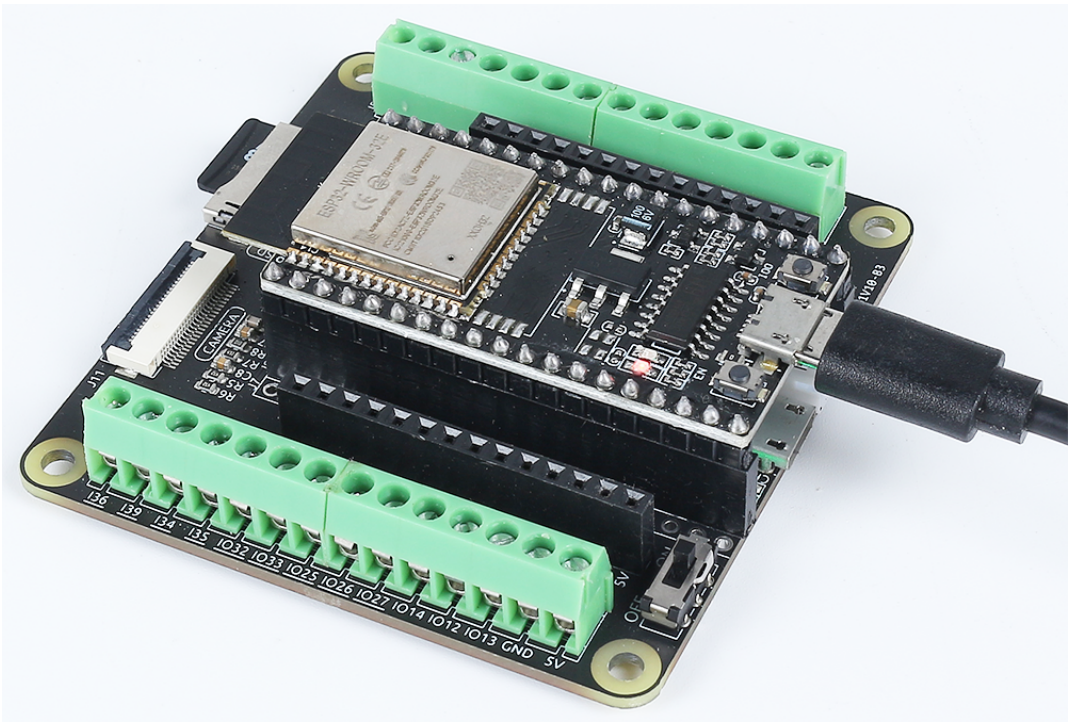
Betriebsschritte

1. Stecken Sie Ihre SD-Karte in einen Kartenleser am Computer und formatieren Sie sie. Sie können sich auf das Tutorial unter *Wie formatiert man eine SD-Karte?* beziehen.
2. Entfernen Sie dann den Kartenleser und stecken Sie die SD-Karte in das Erweiterungsboard.



3. Stecken Sie jetzt die Kamera ein.

4. Verbinden Sie ESP32-WROOM-32E mit dem Computer über das USB-Kabel.



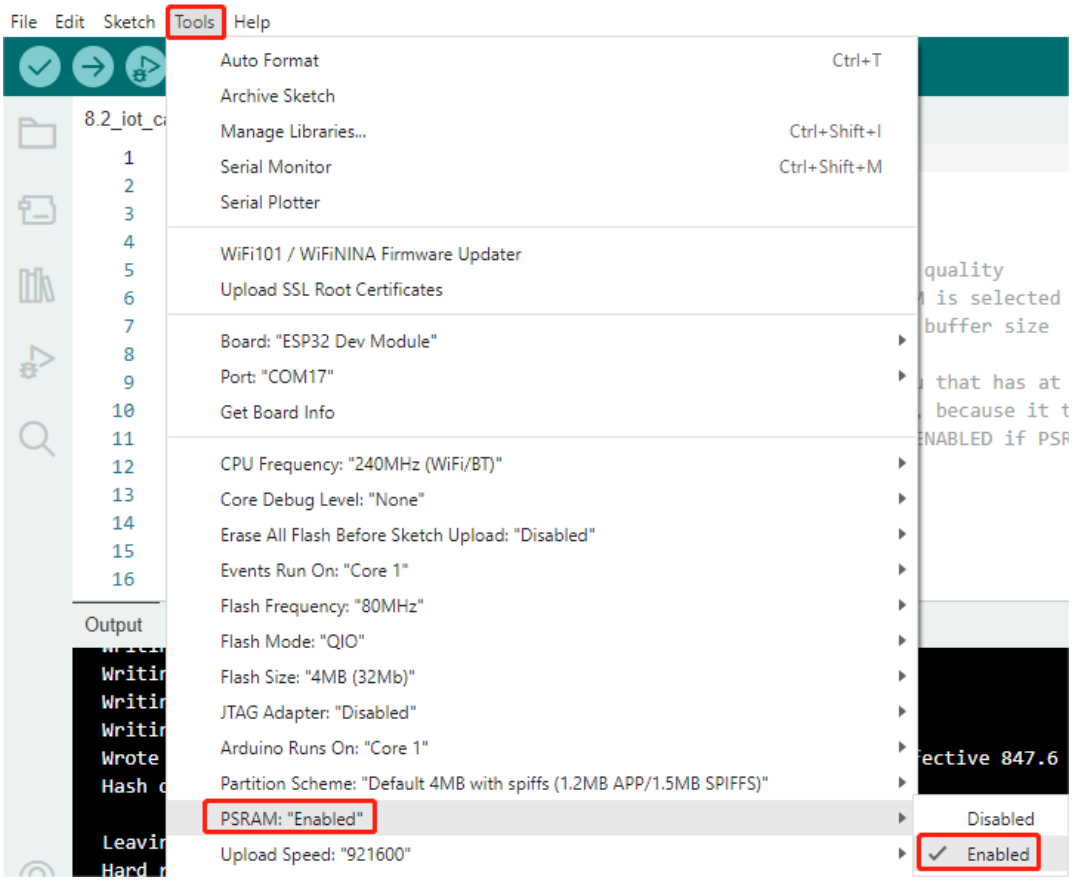
5. Öffnen Sie den Code.

Bemerkung:

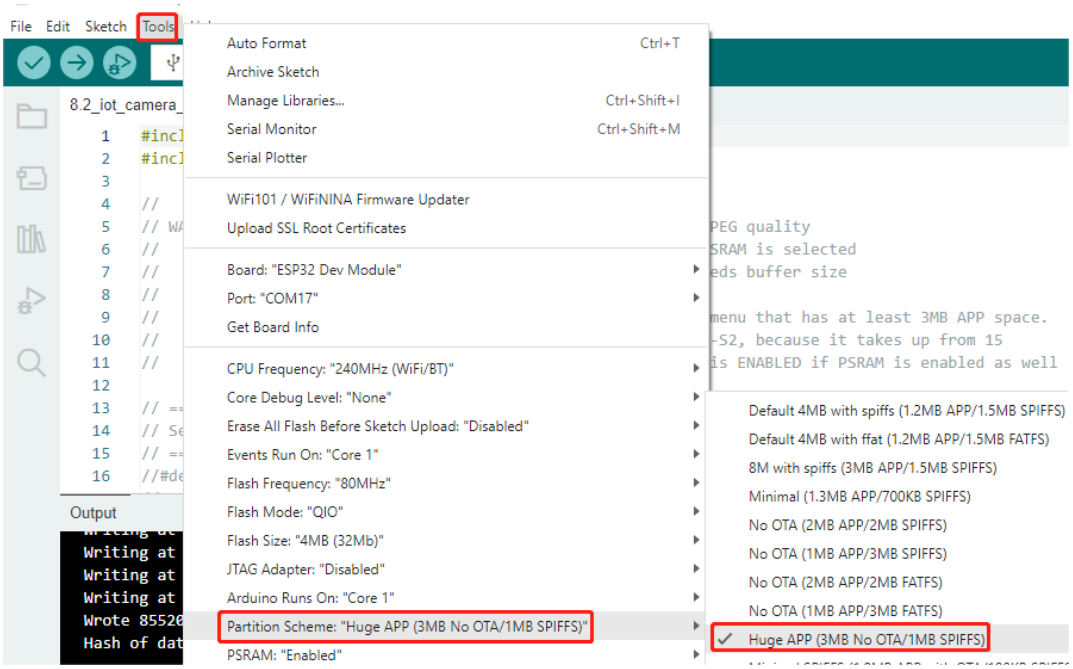
- Öffnen Sie die Datei `7.6_take_photo_sd.ino` unter dem Pfad `esp32-starter-kit-main\c\codes\7.6_take_photo_sd`.

- Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
- *„Unbekanntes COMxx“ wird immer angezeigt?*

6. Aktivieren Sie jetzt **PSRAM**.



7. Stellen Sie das Partitionsschema auf **Huge APP (3MB No OTA/1MB SPIFFS)** ein.

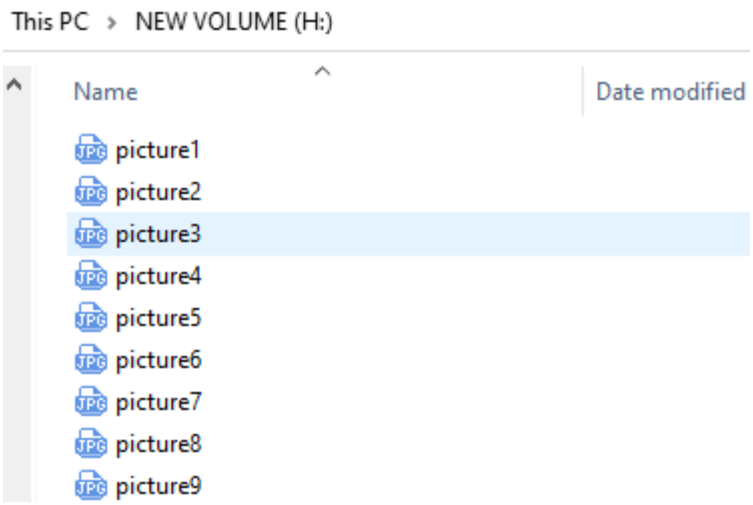


8. Wählen Sie den passenden Port und das Board in der Arduino IDE aus und laden Sie den Code auf Ihren ESP32.
9. Nach dem erfolgreichen Hochladen des Codes drücken Sie den **Reset**-Knopf, um ein Foto zu machen. Zusätzlich können Sie den seriellen Monitor überprüfen, um die folgenden Informationen über die erfolgreiche Aufnahme zu sehen.

```
Picture file name: /picture9.jpg
Saved file to path: /picture9.jpg
Going to sleep now
```

arduino/basic_projects/img/press_reset.PNG

10. Entfernen Sie jetzt die SD-Karte vom Erweiterungsboard und stecken Sie sie in Ihren Computer. Sie können die Fotos ansehen, die Sie gerade gemacht haben.



Wie funktioniert das?

Dieser Code betreibt eine AI Thinker ESP32-CAM, um ein Foto zu machen, es auf einer SD-Karte zu speichern und dann die ESP32-CAM in den Tiefschlaf zu versetzen. Hier ist eine Aufschlüsselung der wichtigsten Teile:

- **Libraries**: Der Code beginnt mit der Einbindung der notwendigen Bibliotheken für die ESP32-CAM, das Dateisystem (FS), die SD-Karte und das EEPROM (zum Speichern von Daten über Stromzyklen hinweg).

```
#include "esp_camera.h"
#include "Arduino.h"
#include "FS.h"                // SD Card ESP32
#include "SD_MMC.h"            // SD Card ESP32
#include "soc/soc.h"           // Disable brownour problems
#include "soc/rtc_cntl_reg.h"  // Disable brownour problems
#include "driver/rtc_io.h"
#include <EEPROM.h>  // read and write from flash memory
```

- **Pin Definitions**: Dieser Abschnitt richtet Konstanten ein, die die Pin-Verbindungen des ESP32-CAM zum Kameramodul darstellen.

```
#define PWDN_GPIO_NUM 32
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 0
#define SIOD_GPIO_NUM 26
#define SIOC_GPIO_NUM 27

#define Y9_GPIO_NUM 35
#define Y8_GPIO_NUM 34
#define Y7_GPIO_NUM 39
#define Y6_GPIO_NUM 36
#define Y5_GPIO_NUM 21
#define Y4_GPIO_NUM 19
#define Y3_GPIO_NUM 18
#define Y2_GPIO_NUM 5
#define VSYNC_GPIO_NUM 25
#define HREF_GPIO_NUM 23
#define PCLK_GPIO_NUM 22
```

- **Global Variables**: Eine globale Variable `pictureNumber` wird deklariert, um die Anzahl der aufgenommenen

und auf die SD-Karte gespeicherten Bilder nachzuverfolgen.

```
int pictureNumber = 0;
```

- **Setup Function**: In der Funktion `setup()` werden mehrere Aufgaben erledigt:
  - Zuerst wird der Braunout-Detektor deaktiviert, um zu verhindern, dass die ESP32-CAM während hoher Stromabnahmen (wie beim Betrieb der Kamera) zurückgesetzt wird.

    ```
    WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);  //disable brownout detector
    ```

  - Die serielle Kommunikation wird zur Fehlersuche initialisiert.

    ```
    Serial.begin(115200);
    ```

  - Die Kamerakonfiguration wird mit `camera_config_t` eingerichtet, einschließlich der GPIO-Pins, XCLK-Frequenz, Pixelformat, Bildgröße, JPEG-Qualität und Anzahl der Framebuffer.

    ```
    camera_config_t config;
    config.ledc_channel = LEDC_CHANNEL_0;
    config.ledc_timer = LEDC_TIMER_0;
    config.pin_d0 = Y2_GPIO_NUM;
    config.pin_d1 = Y3_GPIO_NUM;
    config.pin_d2 = Y4_GPIO_NUM;
    config.pin_d3 = Y5_GPIO_NUM;
    config.pin_d4 = Y6_GPIO_NUM;
    config.pin_d5 = Y7_GPIO_NUM;
    config.pin_d6 = Y8_GPIO_NUM;
    config.pin_d7 = Y9_GPIO_NUM;
    config.pin_xclk = XCLK_GPIO_NUM;
    config.pin_pclk = PCLK_GPIO_NUM;
    config.pin_vsync = VSYNC_GPIO_NUM;
    config.pin_href = HREF_GPIO_NUM;
    config.pin_sscb_sda = SIOD_GPIO_NUM;
    config.pin_sscb_scl = SIOC_GPIO_NUM;
    config.pin_pwdn = PWDN_GPIO_NUM;
    config.pin_reset = RESET_GPIO_NUM;
    config.xclk_freq_hz = 20000000;
    config.pixel_format = PIXFORMAT_JPEG;
    ```

  - Die Kamera wird dann mit der Konfiguration initialisiert und, falls dies fehlschlägt, wird eine Fehlermeldung gedruckt.

    ```
    esp_err_t err = esp_camera_init(&config);
    if (err != ESP_OK) {
        Serial.printf("Camera init failed with error 0x%x", err);
        return;
    }
    ```

  - Die SD-Karte wird initialisiert und, falls dies fehlschlägt, wird eine Fehlermeldung gedruckt.

    ```
    if (!SD_MMC.begin()) {
        Serial.println("SD Card Mount Failed");
        return;
    }
    ```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
uint8_t cardType = SD_MMC.cardType();
if (cardType == CARD_NONE) {
    Serial.println("No SD Card attached");
    return;
}
```

– Ein Foto wird mit der Kamera aufgenommen und im Framebuffer gespeichert.

```
fb = esp_camera_fb_get();
if (!fb) {
    Serial.println("Camera capture failed");
    return;
}
```

– Das EEPROM wird ausgelesen, um die Nummer des letzten Bildes abzurufen, dann wird die Bildnummer für das neue Foto erhöht.

```
EEPROM.begin(EEPROM_SIZE);
pictureNumber = EEPROM.read(0) + 1;
```

– Ein Pfad für das neue Bild wird auf der SD-Karte erstellt, mit einem Dateinamen, der der Bildnummer entspricht.

```
String path = "/picture" + String(pictureNumber) + ".jpg";

fs::FS &fs = SD_MMC;
Serial.printf("Picture file name: %s\n", path.c_str());
```

– Nach dem Speichern des Fotos wird die Bildnummer im EEPROM für das nächste Einschalten zurückgespeichert.

```
File file = fs.open(path.c_str(), FILE_WRITE);
if (!file) {
    Serial.println("Failed to open file in writing mode");
} else {
    file.write(fb->buf, fb->len);  // payload (image), payload length
    Serial.printf("Saved file to path: %s\n", path.c_str());
    EEPROM.write(0, pictureNumber);
    EEPROM.commit();
}
file.close();
esp_camera_fb_return(fb);
```

– Schließlich wird die Onboard-LED (Blitz) ausgeschaltet und die ESP32-CAM geht in den Tiefschlaf.

```
pinMode(4, OUTPUT);
digitalWrite(4, LOW);
rtc_gpio_hold_en(GPIO_NUM_4);
```

– Schlafmodus: Die ESP32-CAM geht nach jeder Fotoaufnahme in den Tiefschlaf, um Strom zu sparen. Sie kann durch einen Reset oder durch ein Signal an bestimmten Pins geweckt werden.

```
delay(2000);
Serial.println("Going to sleep now");
delay(2000);
esp_deep_sleep_start();
Serial.println("This will never be printed");
```

- Loop-Funktion: Die `loop()`-Funktion ist leer, weil die ESP32-CAM unmittelbar nach dem Setup-Prozess in den Tiefschlaf geht.

Beachten Sie, dass dieser Code nur funktioniert, wenn GPIO 0 beim Hochladen des Sketches mit GND verbunden ist, und Sie müssen möglicherweise den RESET-Knopf am Board drücken, um Ihr Board in den Flash-Modus zu versetzen. Denken Sie auch daran, „/picture“ durch Ihren eigenen Dateinamen zu ersetzen. Die Größe des EEPROMs ist auf 1 gesetzt, was bedeutet, dass es Werte von 0 bis 255 speichern kann. Wenn Sie mehr als 255 Bilder aufnehmen möchten, müssen Sie die Größe des EEPROMs erhöhen und die Art und Weise, wie Sie die pictureNumber speichern und auslesen, anpassen.

8. Bluetooth&SD-Karte&Kamera&Lautsprecher

2.44 8.1 Echtzeit-Wetter von @OpenWeatherMap

Das IoT Open Weather Display-Projekt nutzt das ESP32-Board und ein I2C LCD1602-Modul, um eine Wetterinformationsanzeige zu erstellen, die Daten von der OpenWeatherMap-API abruft.

Dieses Projekt dient als hervorragende Einführung in die Arbeit mit APIs, Wi-Fi-Konnektivität und Datenanzeige auf einem LCD-Modul mit dem ESP32-Board. Mit dem IoT Open Weather Display können Sie bequem Echtzeit-Wetteraktualisierungen auf einen Blick abrufen, was es zu einer idealen Lösung für Heim- oder Büroumgebungen macht.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

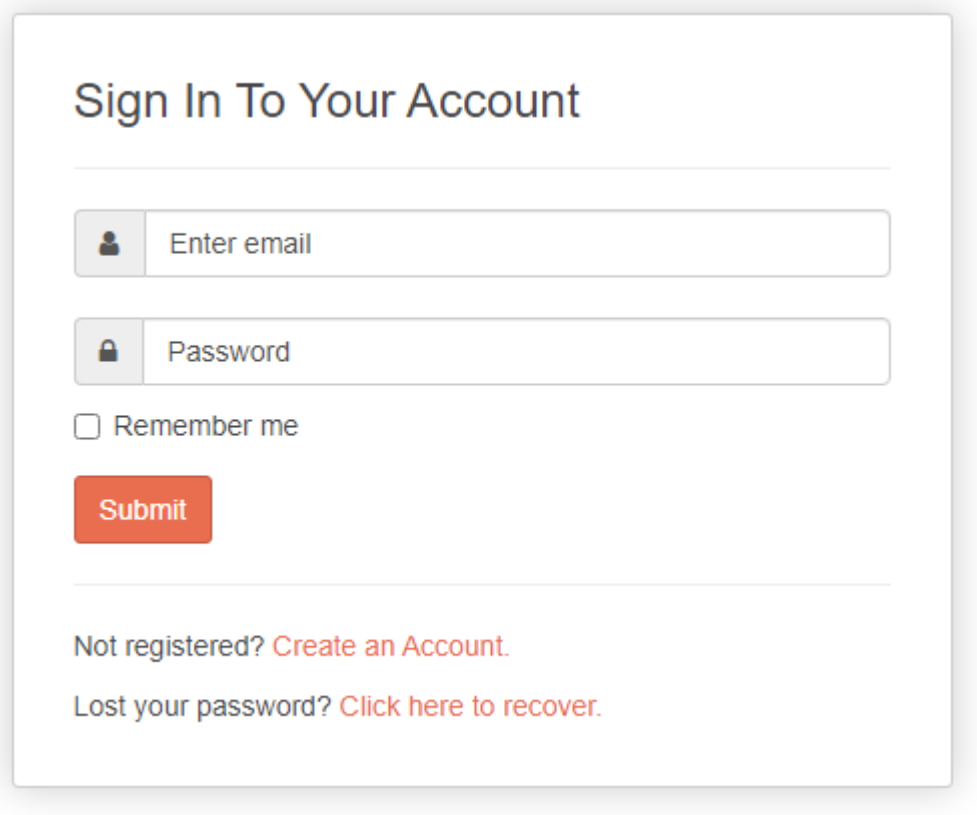
Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
I2C LCD1602	

OpenWeather API-Schlüssel abrufen

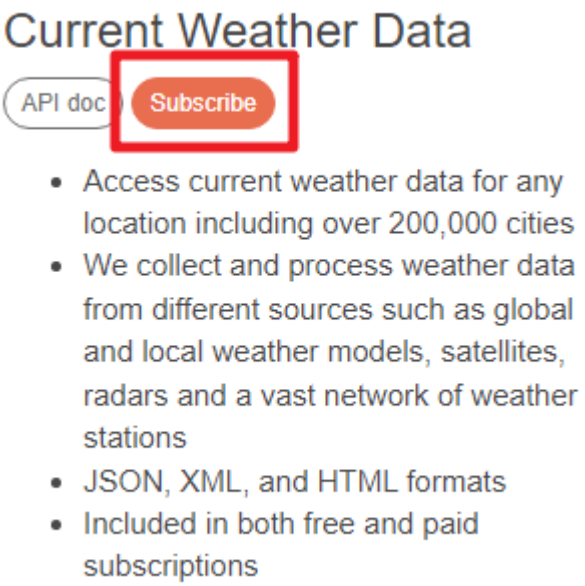
ist ein Online-Dienst von OpenWeather Ltd, der weltweite Wetterdaten über eine API bereitstellt, einschließlich aktueller Wetterdaten, Prognosen, Nowcasts und historischer Wetterdaten für jeden geografischen Standort.

1. Besuchen Sie , um sich anzumelden/ein Konto zu erstellen.



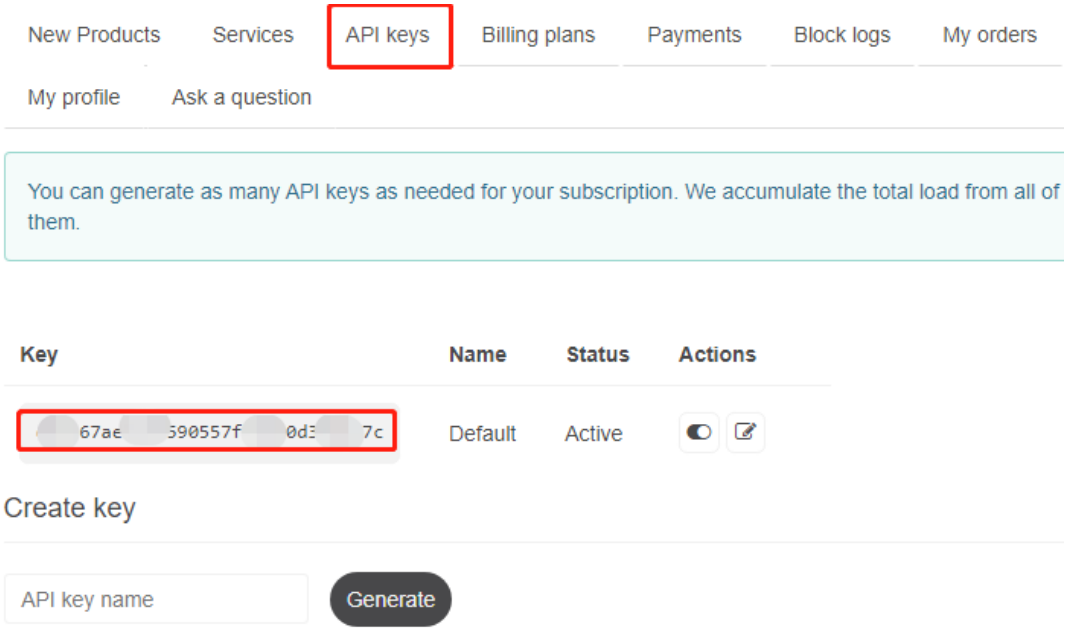
2. Klicken Sie in der Navigationsleiste auf die API-Seite.

3. Finden Sie **Current Weather Data** und klicken Sie auf Abonnieren.



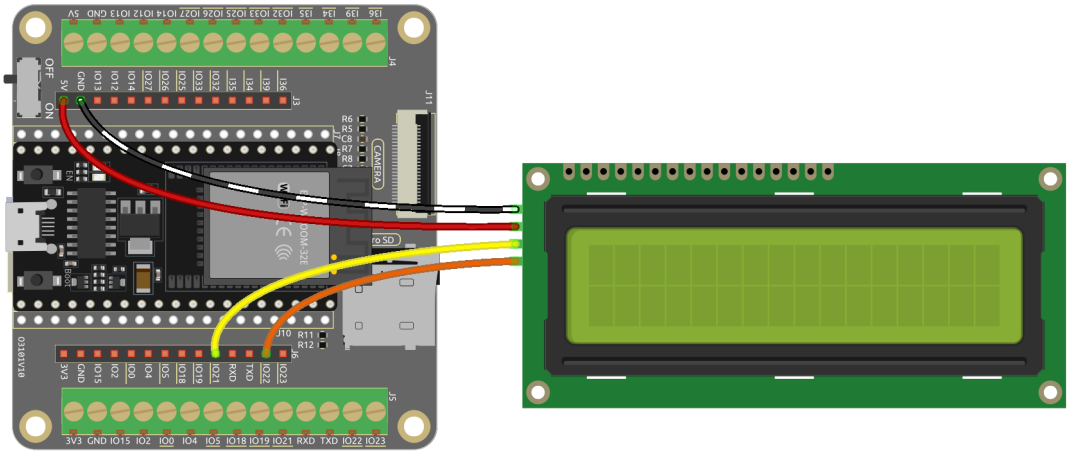
4. Abonnieren Sie unter **Current weather and forecasts collection** den entsprechenden Dienst. In unserem Projekt ist Free ausreichend.

5. Kopieren Sie den Schlüssel von der Seite **API keys**.



Vollenden Sie Ihr Gerät

1. Bauen Sie den Schaltkreis.



2. Öffnen Sie den Code.

    - Öffnen Sie die Datei `iot_1_open_weather.ino`, die sich im Verzeichnis `esp32-starter-kit-main\c\codes\iot_1_open_weather` befindet, oder kopieren Sie den Code in die Arduino IDE.
    - Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
    - *„Unbekanntes COMxx“ wird immer angezeigt?*
    - Hier werden die Bibliotheken `LiquidCrystal I2C` und `Arduino_JSON` verwendet, die Sie über den **Library Manager** installieren können.

3. Suchen Sie die folgenden Zeilen und ändern Sie sie mit Ihrem <SSID> und <PASSWORD>.

```
// Replace the next variables with your SSID/Password combination
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

4. Fügen Sie die zuvor kopierten API-Schlüssel in `openWeatherMapApiKey` ein.

```
// Your Domain name with URL path or IP address with path
String openWeatherMapApiKey = "<openWeatherMapApiKey>";
```

5. Ersetzen Sie sie mit Ihrem Ländercode und Ihrer Stadt.

```
// Replace with your country code and city
// Fine the country code by https://openweathermap.org/find
String city = "<CITY>";
String countryCode = "<COUNTRY CODE>";
```

6. Nachdem der Code ausgeführt wird, sehen Sie die Uhrzeit und Wetterinformationen Ihres Standorts auf dem I2C LCD1602.

Bemerkung: Wenn der Code läuft und der Bildschirm leer ist, können Sie das Potentiometer auf der Rückseite des Moduls drehen, um den Kontrast zu erhöhen.

2.45 8.2 Kamera-Webserver

Dieses Projekt kombiniert das ESP32-Board mit einem Kameramodul, um hochwertiges Video über ein lokales Netzwerk zu streamen. Richten Sie mühelos Ihr eigenes Kamerasystem ein und überwachen Sie jeden Ort in Echtzeit.

Mit der Web-Oberfläche des Projekts können Sie auf den Kamerastream zugreifen und ihn von jedem mit dem Netzwerk verbundenen Gerät aus steuern. Passen Sie die Kameraeinstellungen an, um das Streaming-Erlebnis zu optimieren, und stellen Sie die Einstellungen einfach über die benutzerfreundliche Oberfläche ein.

Erweitern Sie Ihre Überwachungs- oder Live-Streaming-Fähigkeiten mit dem vielseitigen ESP32 Camera Streaming-Projekt. Überwachen Sie Ihr Zuhause, Ihr Büro oder jeden gewünschten Ort mit Leichtigkeit und Zuverlässigkeit.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

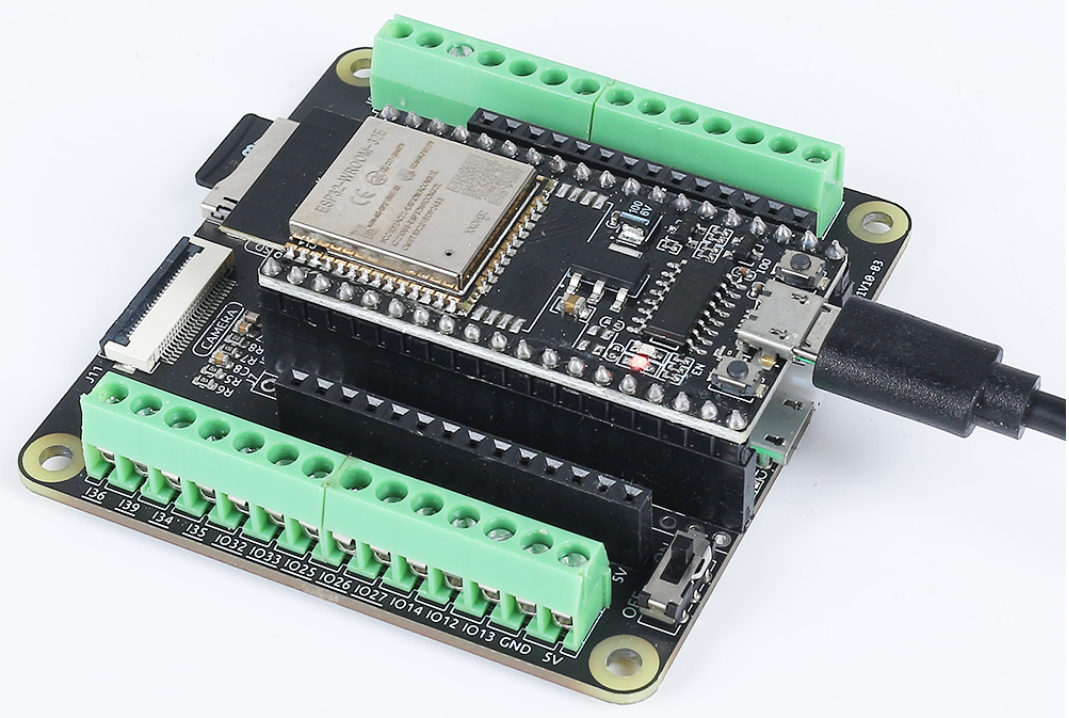
Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-

Wie macht man das?

1. Schließen Sie zuerst die Kamera an.

2. Verbinden Sie dann ESP32-WROOM-32E mit dem Computer über das USB-Kabel.



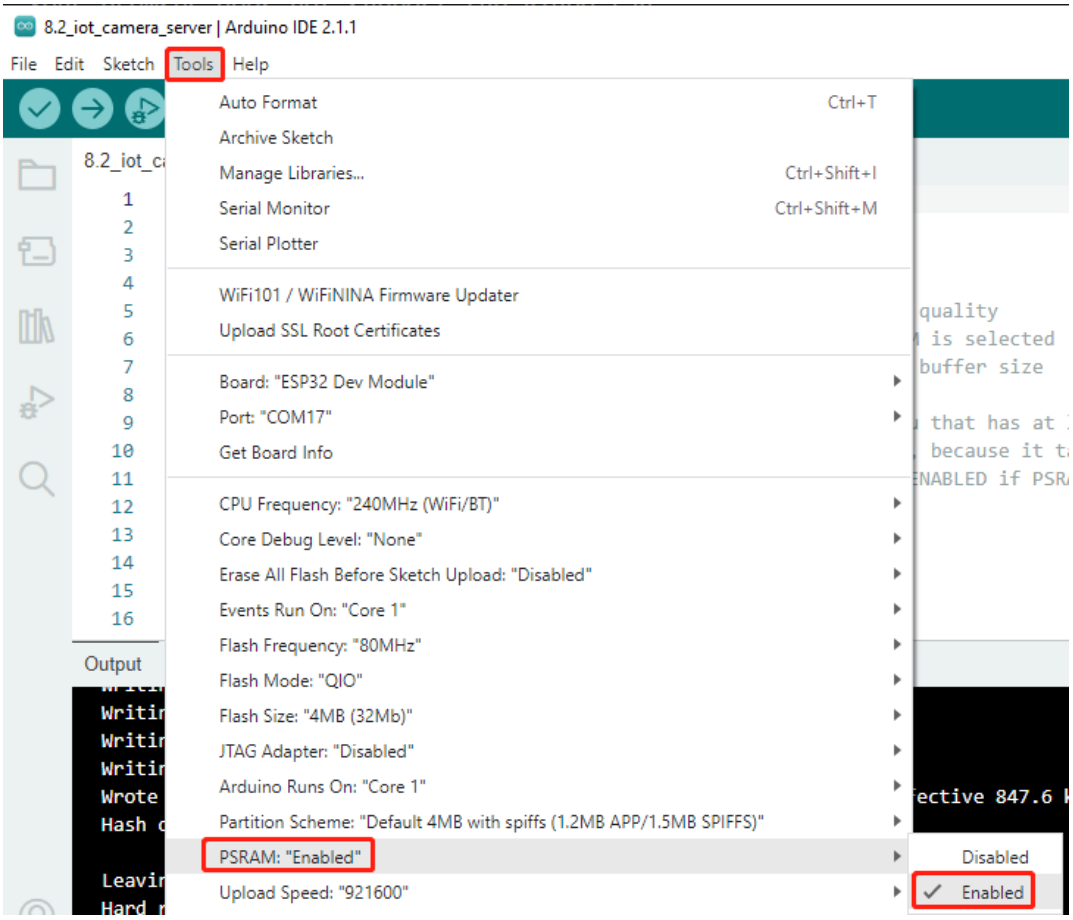
3. Öffnen Sie den Code.

    - Öffnen Sie die Datei `iot_2_camera_server.ino`, die sich im Verzeichnis `esp32-starter-kit-main\c\codes\iot_2_camera_server` befindet, oder kopieren Sie den Code in die Arduino IDE.
    - Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
    - *„Unbekanntes COMxx“ wird immer angezeigt?*

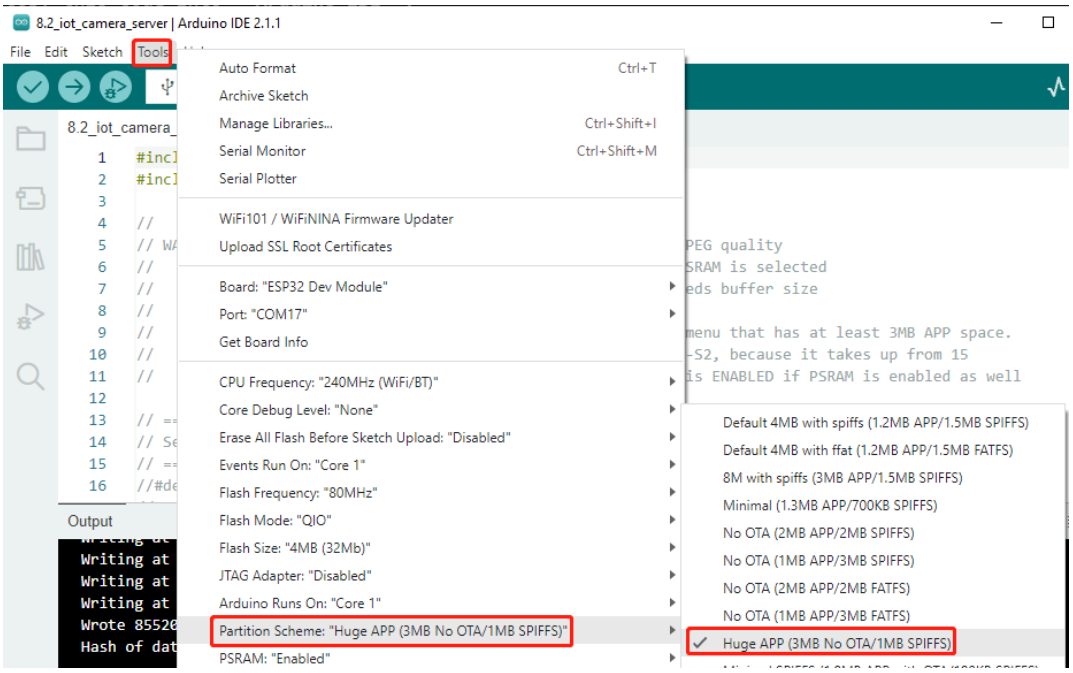
4. Suchen Sie die folgenden Zeilen und ändern Sie sie mit Ihrem <SSID> und <PASSWORD>.

```
// Ersetzen Sie die nächsten Variablen mit Ihrer SSID/Passwort-Kombination
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

5. Aktivieren Sie jetzt **PSRAM**.



6. Stellen Sie das Partitionsschema auf **Huge APP (3MB No OTA/1MB SPIFFS)** ein.

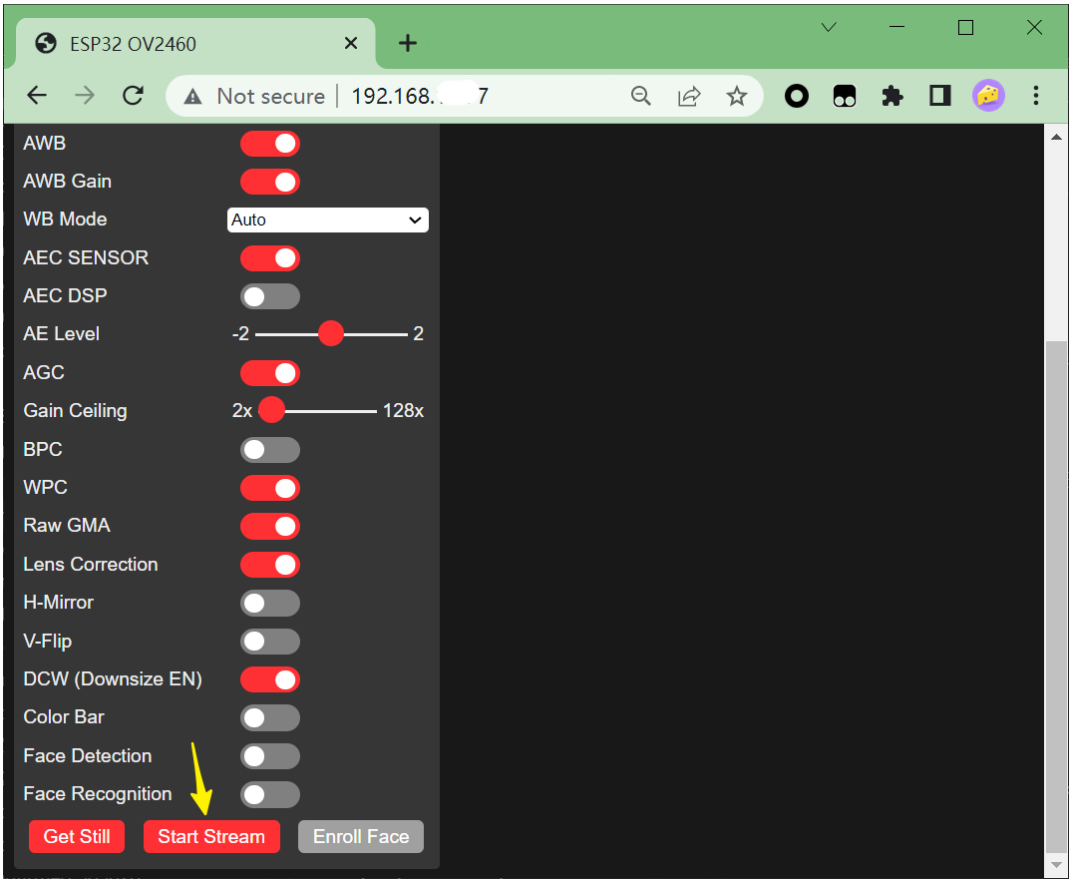


7. Nachdem Sie das richtige Board (ESP32 Dev Module) und den Port ausgewählt haben, klicken Sie auf den „Hochladen“-Knopf.

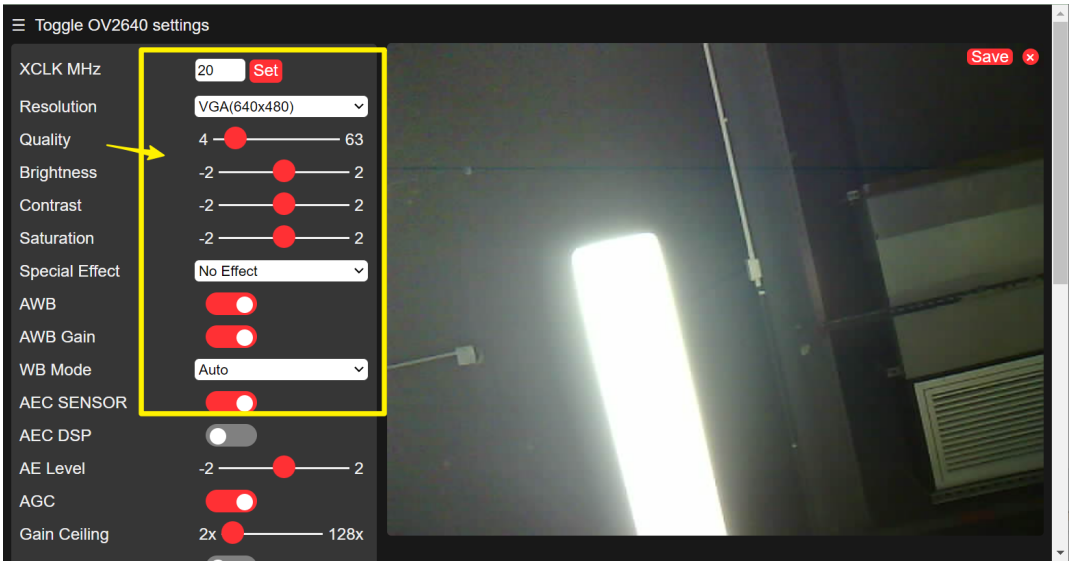
8. Im seriellen Monitor sehen Sie eine erfolgreiche WLAN-Verbindungsmeldung und die zugewiesene IP-Adresse.

```
.....
WiFi connected
Starting web server on port: '80'
Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.18.77' to connect
```

9. Geben Sie die IP-Adresse in Ihrem Webbrowser ein. Sie sehen eine Web-Oberfläche, auf der Sie **Start Stream** klicken können, um den Kamerastream anzusehen.



10. Scrollen Sie zurück nach oben auf der Seite, wo Sie den Live-Kamerastream sehen. Sie können die Einstellungen auf der linken Seite der Oberfläche anpassen.



Bemerkung:

- Dieses ESP32-Modul unterstützt Gesichtserkennung. Um es zu aktivieren, stellen Sie die Auflösung auf 240x240 ein und schalten Sie die Option für die Gesichtserkennung am unteren Rand der Oberfläche um.
- Dieses ESP32-Modul unterstützt keine Gesichtserkennung.

2.46 8.3 Benutzerdefinierter Video-Streaming-Webserver

Das Projekt „Benutzerdefinierter Video-Streaming-Webserver“ bietet die Möglichkeit, zu lernen, wie man eine Webseite von Grund auf erstellt und sie so anpasst, dass Videostreams abgespielt werden können. Zusätzlich können Sie interaktive Schaltflächen, wie EIN und AUS, integrieren, um die Helligkeit der LED zu steuern.

Durch den Aufbau dieses Projekts sammeln Sie praktische Erfahrungen in der Webentwicklung, HTML, CSS und JavaScript. Sie lernen, wie man eine responsive Webseite erstellt, die Videostreams in Echtzeit anzeigen kann. Außerdem entdecken Sie, wie man interaktive Schaltflächen integriert, um den Zustand der LED zu steuern und so ein dynamisches Benutzererlebnis zu bieten.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

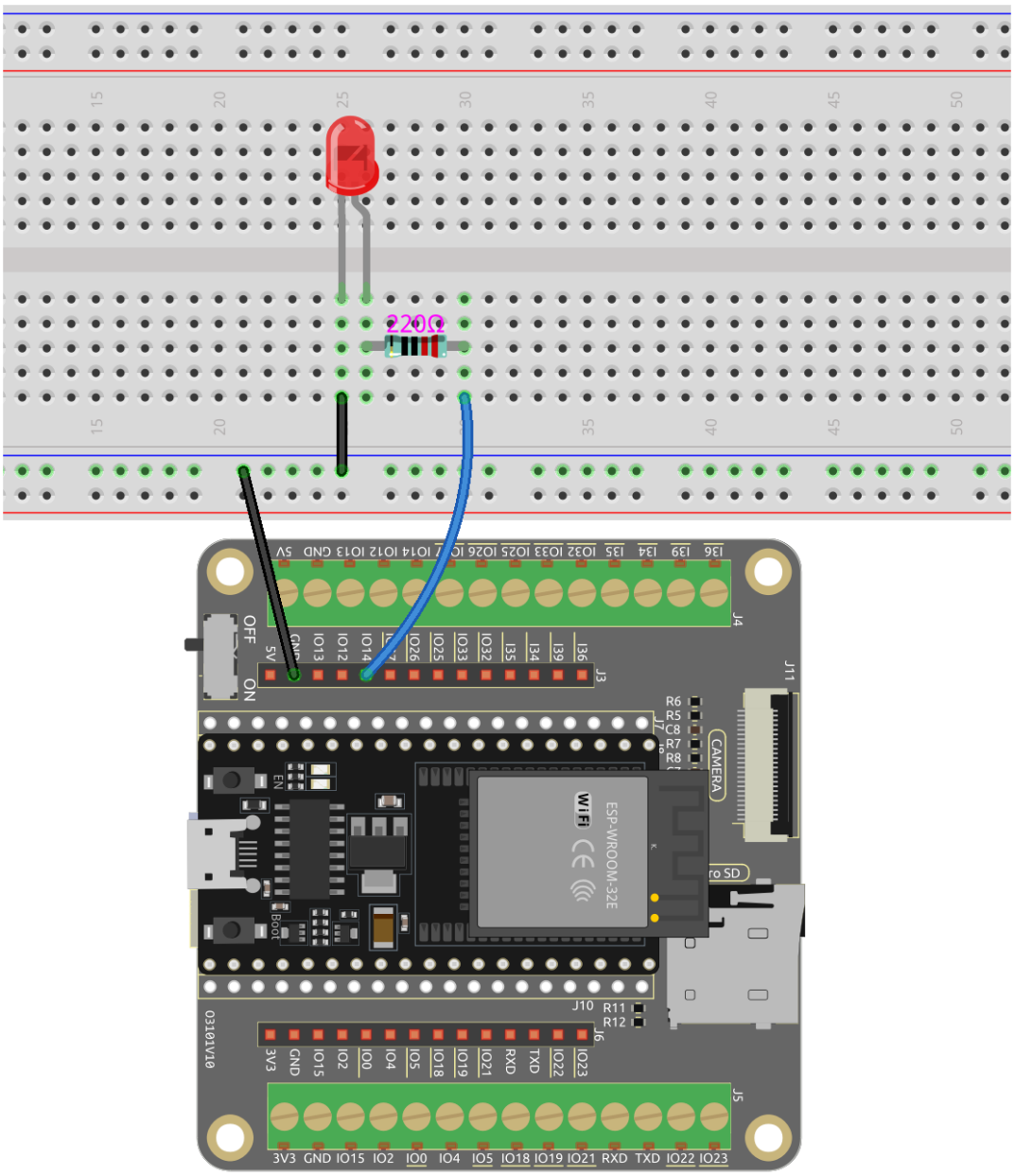
Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	

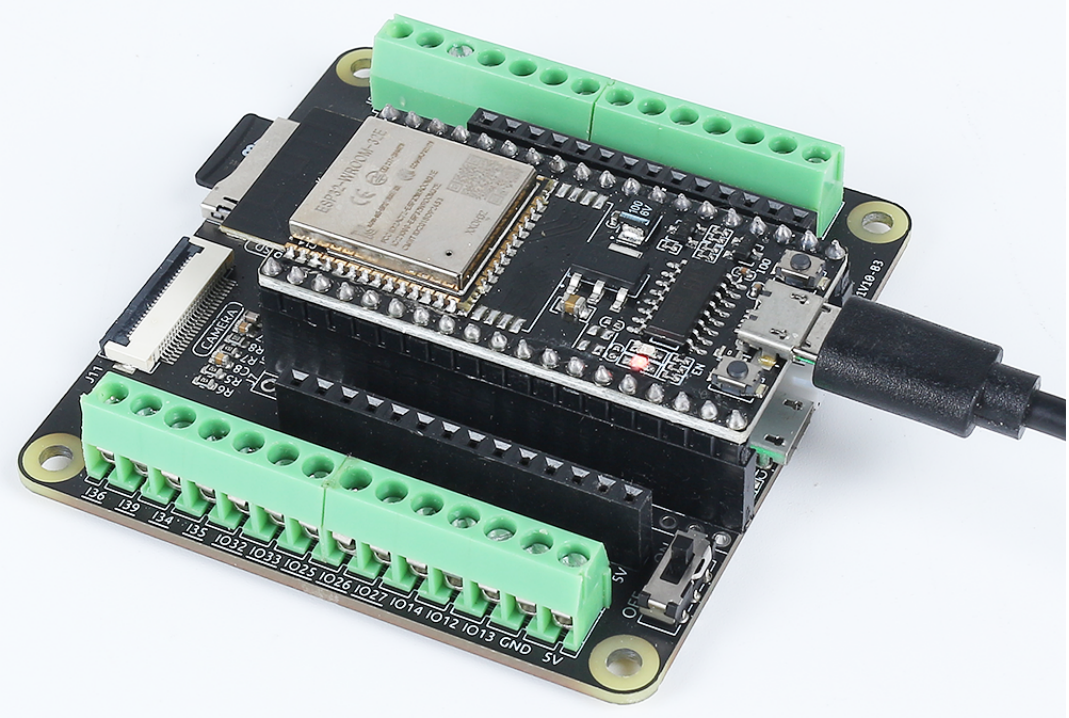
Wie macht man das?

1. Schließen Sie zuerst die Kamera an.

2. Bauen Sie den Schaltkreis.



3. Verbinden Sie dann ESP32-WROOM-32E mit dem Computer über das USB-Kabel.



4. Öffnen Sie den Code.

    - Öffnen Sie die Datei `iot_3_html_cam_led.ino`, die sich im Verzeichnis `esp32-starter-kit-main\c\codes\iot_3_html_cam_led` befindet, oder kopieren Sie den Code in die Arduino IDE.
    - Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
    - *„Unbekanntes COMxx“ wird immer angezeigt?*

5. Suchen Sie die folgenden Zeilen und ändern Sie sie mit Ihrem <SSID> und <PASSWORD>.

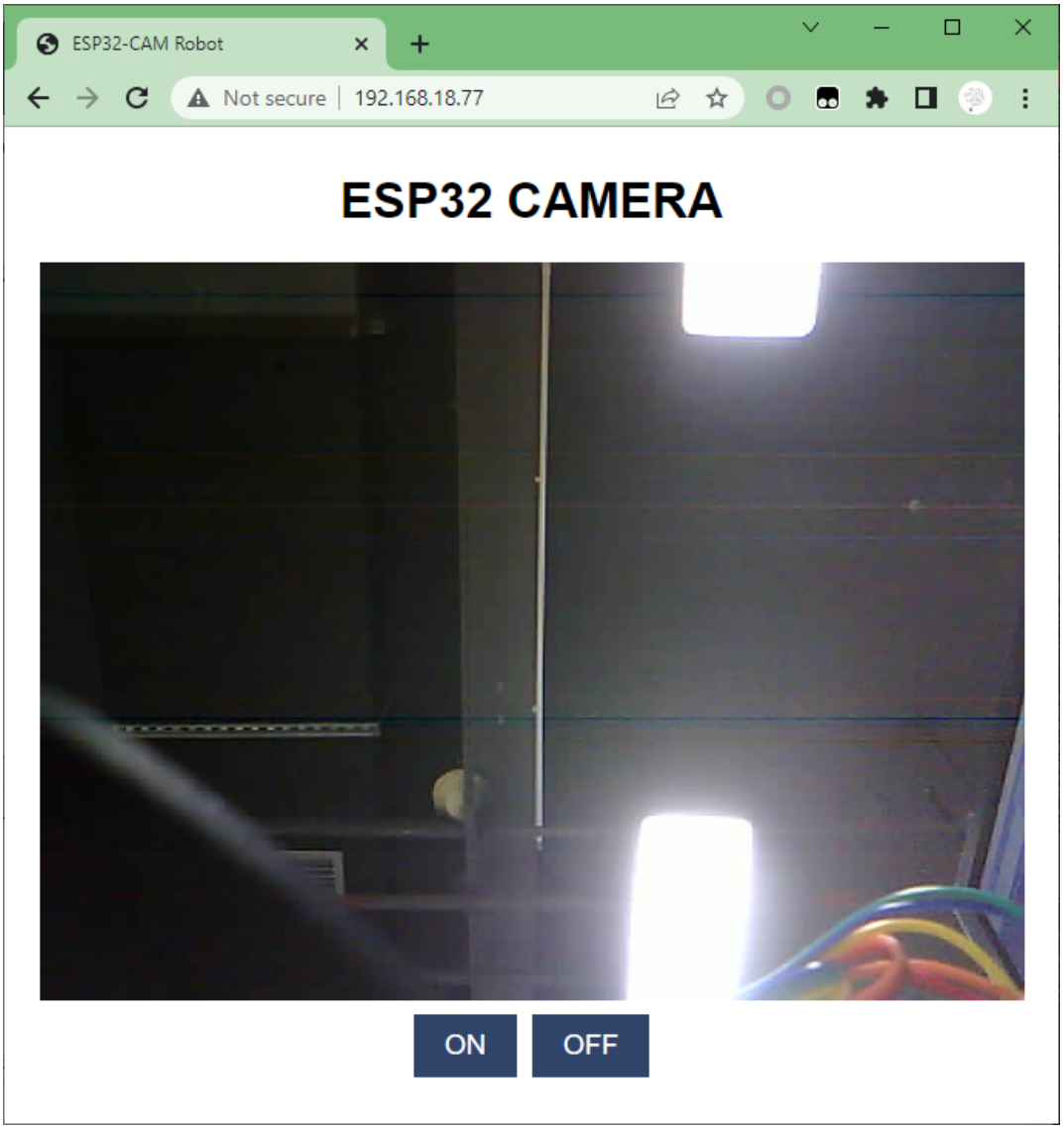
```
// Replace the next variables with your SSID/Password combination
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

6. Nachdem Sie das richtige Board (ESP32 Dev Module) und den Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.

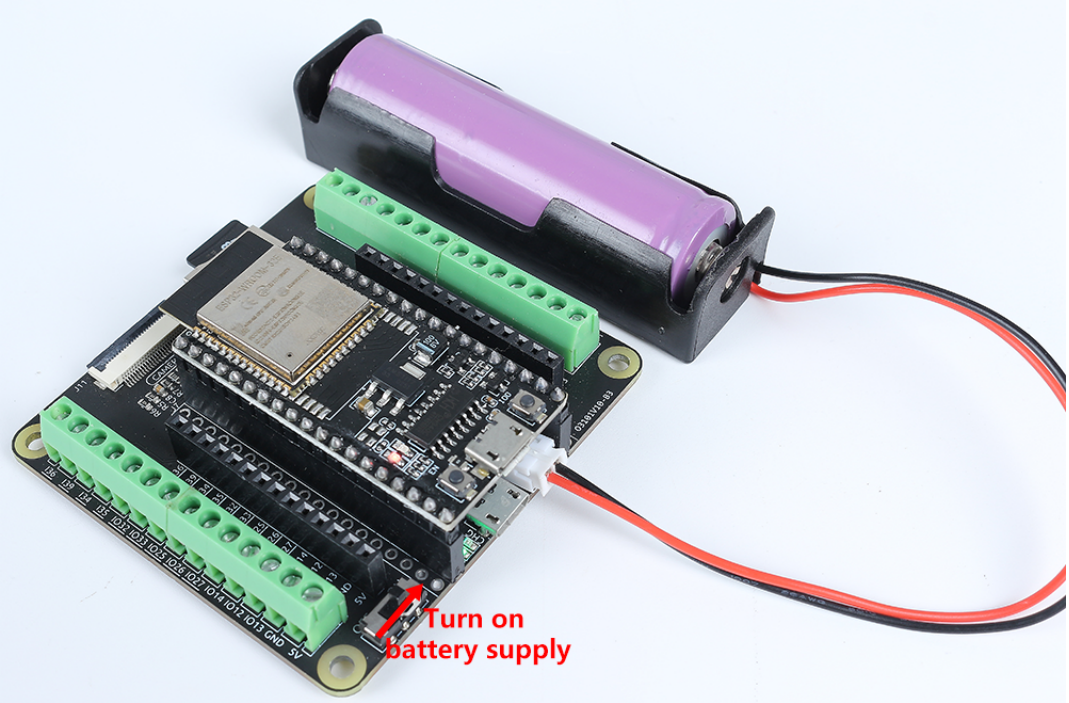
7. Im seriellen Monitor sehen Sie eine erfolgreiche WLAN-Verbindungsmeldung und die zugewiesene IP-Adresse.

```
WiFi connected
Camera Stream Ready! Go to: http://192.168.18.77
```

8. Geben Sie die IP-Adresse in Ihrem Webbrowser ein. Sie werden zu der unten gezeigten Webseite geleitet, auf der Sie die benutzerdefinierten EIN- und AUS-Tasten verwenden können, um die LED zu steuern.



9. Legen Sie einen Akku in das Erweiterungsboard ein und entfernen Sie das USB-Kabel. Jetzt können Sie das Gerät überall innerhalb der WLAN-Reichweite platzieren.



2.47 8.4 IoT-Kommunikation mit MQTT

Dieses Projekt konzentriert sich auf die Nutzung von MQTT, einem beliebten Kommunikationsprotokoll im Bereich des Internets der Dinge (IoT). MQTT ermöglicht es IoT-Geräten, Daten über ein Publish/Subscribe-Modell auszutauschen, bei dem Geräte über Themen kommunizieren.

In diesem Projekt erforschen wir die Implementierung von MQTT, indem wir einen Schaltkreis aufbauen, der eine LED, einen Knopf und einen Thermistor umfasst. Der ESP32-WROOM-32E-Mikrocontroller wird verwendet, um eine Verbindung zum WLAN herzustellen und mit einem MQTT-Broker zu kommunizieren. Der Code ermöglicht es dem Mikrocontroller, sich für bestimmte Themen zu abonnieren, Nachrichten zu empfangen und die LED basierend auf den empfangenen Informationen zu steuern. Zusätzlich demonstriert das Projekt das Veröffentlichen von Temperaturdaten vom Thermistor zu einem festgelegten Thema, wenn der Knopf gedrückt wird.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

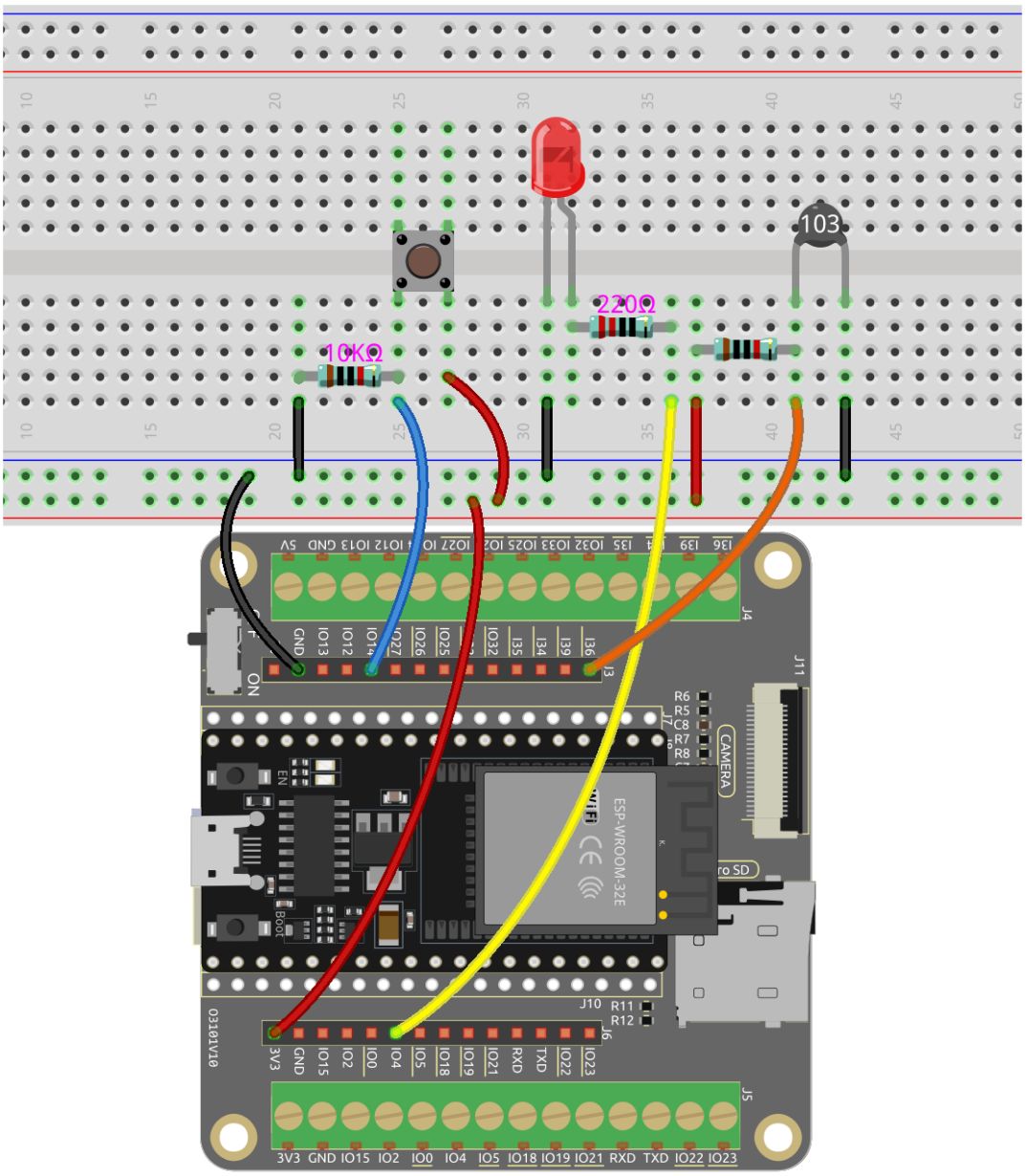
Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
Taste	
Thermistor	

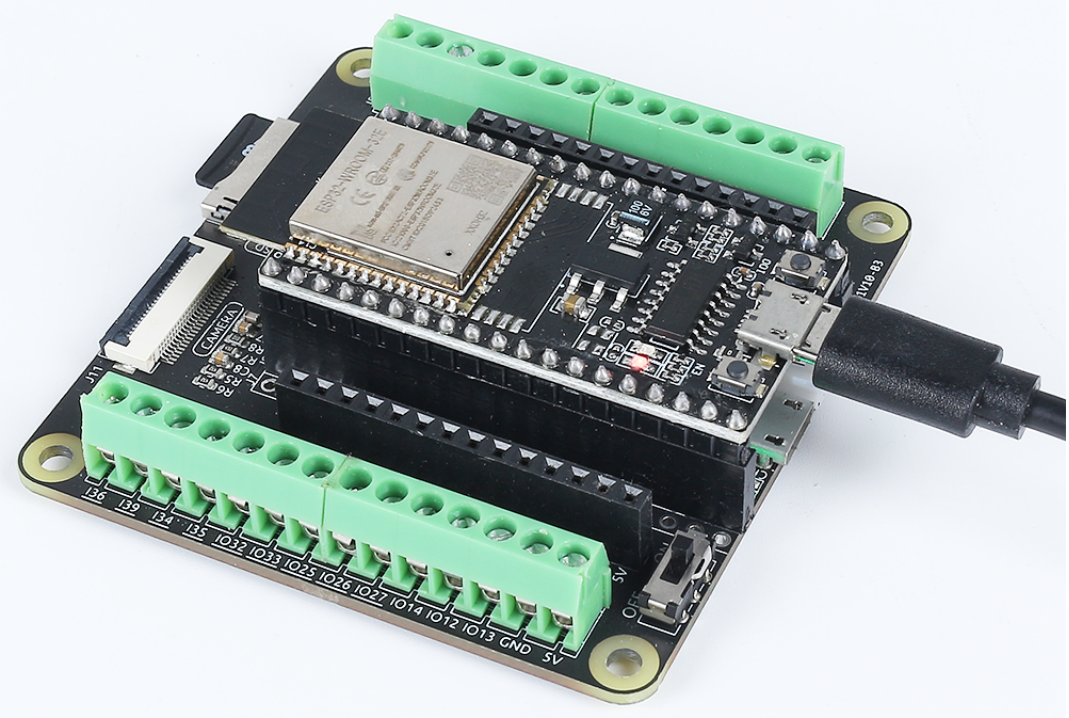
Code hochladen

1. Bauen Sie den Schaltkreis.

> **Bemerkung:** Bei der Herstellung einer Verbindung zum WLAN können nur die Pins 36, 39, 34, 35, 32, 33 für die Analogmessung verwendet werden. Bitte stellen Sie sicher, dass der Thermistor mit diesen festgelegten Pins verbunden ist.

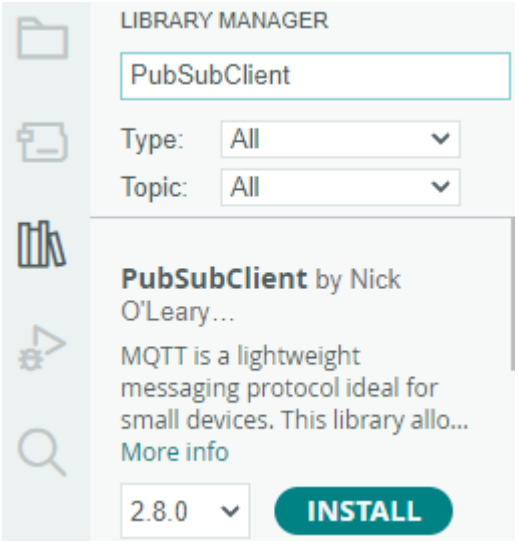


2. Verbinden Sie dann ESP32-WROOM-32E mit dem Computer über das USB-Kabel.



3. Öffnen Sie den Code.

    - Öffnen Sie die Datei `iot_4_mqtt.ino`, die sich im Verzeichnis `esp32-starter-kit-main\c\codes\iot_4_mqtt` befindet, oder kopieren Sie den Code in die Arduino IDE.
    - Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
    - *„Unbekanntes COMxx“ wird immer angezeigt?*
    - Hier wird die Bibliothek `PubSubClient` verwendet, die Sie über den **Library Manager** installieren können.



4. Suchen Sie die folgenden Zeilen und ändern Sie sie mit Ihrem <SSID> und <PASSWORD>.

```
// Replace the next variables with your SSID/Password combination
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

5. Finden Sie die nächste Zeile und ändern Sie Ihren `unique_identifier`. Stellen Sie sicher, dass Ihr `unique_identifier` wirklich einzigartig ist, da identische IDs, die versuchen, sich bei demselben MQTT-Broker anzumelden, zu einem Anmeldefehler führen können.

```
// Add your MQTT Broker address, example:
const char* mqtt_server = "broker.hivemq.com";
const char* unique_identifier = "sunfounder-client-sdgvsda";
```

Themenabonnement

1. Um Störungen durch Nachrichten anderer Teilnehmer zu vermeiden, können Sie es als obskuren oder ungewöhnlichen String setzen. Ersetzen Sie einfach das aktuelle Thema `SF/LED` mit Ihrem gewünschten Themennamen.

Bemerkung: Sie haben die Freiheit, das Thema als jeden beliebigen Charakter festzulegen. Jedes MQTT-Gerät, das das gleiche Thema abonniert hat, kann dieselbe Nachricht empfangen. Sie können auch gleichzeitig mehrere Themen abonnieren.

```
void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect(unique_identifier)) {
            Serial.println("connected");
            // Subscribe
            client.subscribe("SF/LED");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}
```

2. Ändern Sie die Funktionalität, um auf das abonnierte Thema zu reagieren. Im bereitgestellten Code wird geprüft, ob eine Nachricht zum Thema `SF/LED` empfangen wurde und ob die Nachricht `on` oder `off` ist. Abhängig von der empfangenen Nachricht ändert es den Ausgabestatus, um den LED-Zustand zu steuern.

Bemerkung: Sie können es für jedes Thema, das Sie abonniert haben, anpassen und Sie können mehrere if-Anweisungen schreiben, um auf mehrere Themen zu reagieren.

```
void callback(char* topic, byte* message, unsigned int length) {
    Serial.print("Message arrived on topic: ");
    Serial.print(topic);
    Serial.print(". Message: ");
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    String messageTemp;

    for (int i = 0; i < length; i++) {
        Serial.print((char)message[i]);
        messageTemp += (char)message[i];
    }
    Serial.println();

    // If a message is received on the topic "SF/LED", you check if the␣
↪message is either "on" or "off".
    // Changes the output state according to the message
    if (String(topic) == "SF/LED") {
        Serial.print("Changing state to ");
        if (messageTemp == "on") {
            Serial.println("on");
            digitalWrite(ledPin, HIGH);
        } else if (messageTemp == "off") {
            Serial.println("off");
            digitalWrite(ledPin, LOW);
        }
    }
}
```

3. Nachdem Sie das richtige Board (ESP32 Dev Module) und den Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
4. Öffnen Sie den seriellen Monitor und wenn die folgenden Informationen gedruckt werden, zeigt dies eine erfolgreiche Verbindung zum MQTT-Server an.

```
WiFi connected
IP address:
192.168.18.77
Attempting MQTT connection...connected
```

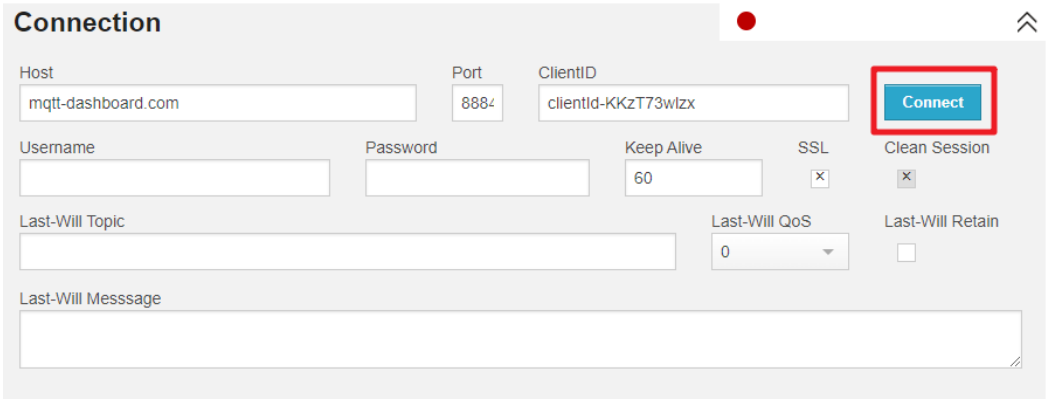
Nachrichtenpublikation über HiveMQ

HiveMQ ist eine Messaging-Plattform, die als MQTT-Broker fungiert und schnellen, effizienten und zuverlässigen Datentransfer zu IoT-Geräten ermöglicht.

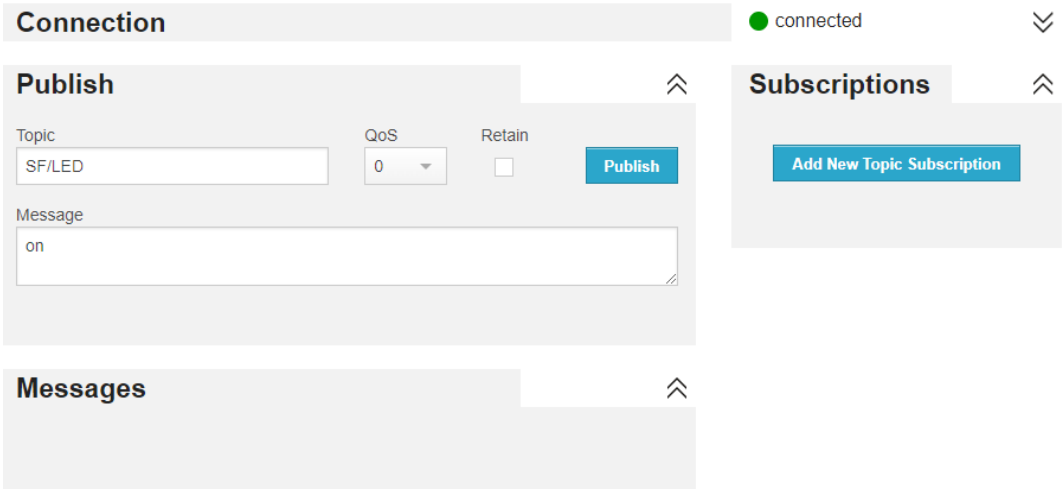
Unser Code nutzt speziell den MQTT-Broker von HiveMQ. Wir haben die Adresse des HiveMQ MQTT-Brokers im Code wie folgt aufgenommen:

```
// Add your MQTT Broker address, example:
const char* mqtt_server = "broker.hivemq.com";
```

1. Öffnen Sie jetzt in Ihrem Webbrowser.
2. Verbinden Sie den Client mit dem Standard-öffentlichen Proxy.



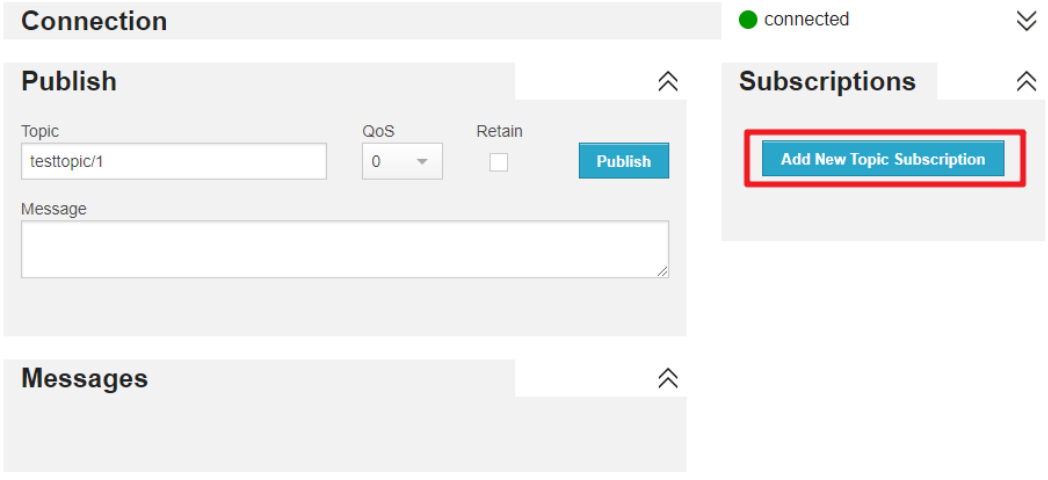
3. Veröffentlichen Sie eine Nachricht im Thema, das Sie abonniert haben. In diesem Projekt können Sie `on` oder `off` veröffentlichen, um Ihre LED zu steuern.



Nachrichtenpublikation an MQTT

Wir können den Code auch nutzen, um Informationen an das Thema zu senden. In dieser Demonstration haben wir eine Funktion codiert, die die von dem Thermistor gemessene Temperatur an das Thema sendet, wenn Sie den Knopf drücken.

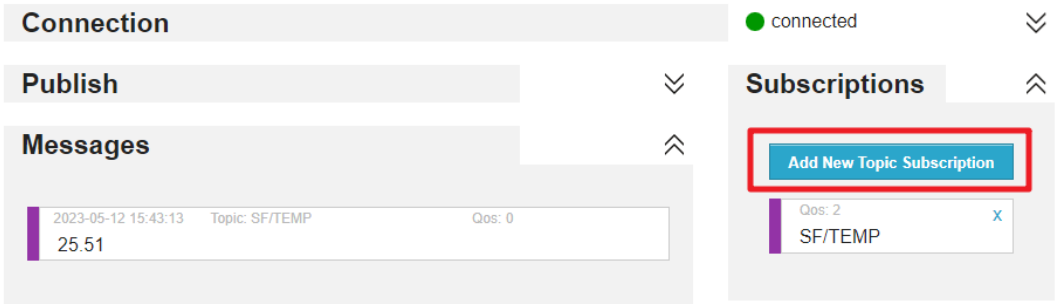
1. Klicken Sie auf **Add New Topic Subscription**.



2. Geben Sie die Themen ein, denen Sie folgen möchten, und klicken Sie auf **Subscribe**. Im Code senden wir Temperaturinformationen an das Thema `SF/TEMP`.

```
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    // if the button pressed, publish the temperature to topic "SF/TEMP"
    if (digitalRead(buttonPin)) {
            long now = millis();
            if (now - lastMsg > 5000) {
            lastMsg = now;
            char tempString[8];
            dtostrf(thermistor(), 1, 2, tempString);
            client.publish("SF/TEMP", tempString);
        }
    }
}
```

3. Daher können wir dieses Thema auf HiveMQ überwachen und die von Ihnen veröffentlichten Informationen einsehen.



2.48 8.5 CheerLights

CheerLights ist ein globales Netzwerk synchronisierter Lichter, das von jedem gesteuert werden kann.

Treten Sie der LED-Farbwechsel-Community bei, die es ermöglicht, dass LEDs auf der ganzen Welt gleichzeitig ihre Farben ändern.

Sie können Ihre LEDs in einer Ecke Ihres Büros platzieren, um sich daran zu erinnern, dass Sie nicht allein sind.

In diesem Fall nutzen wir auch MQTT, aber anstatt unsere eigenen Nachrichten zu veröffentlichen, abonnieren wir das Thema „cheerlights“. Dies ermöglicht es uns, Nachrichten, die von anderen an das Thema „cheerlights“ gesendet wurden, zu empfangen und diese Informationen zu nutzen, um die Farbe unseres LED-Streifens entsprechend zu ändern.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

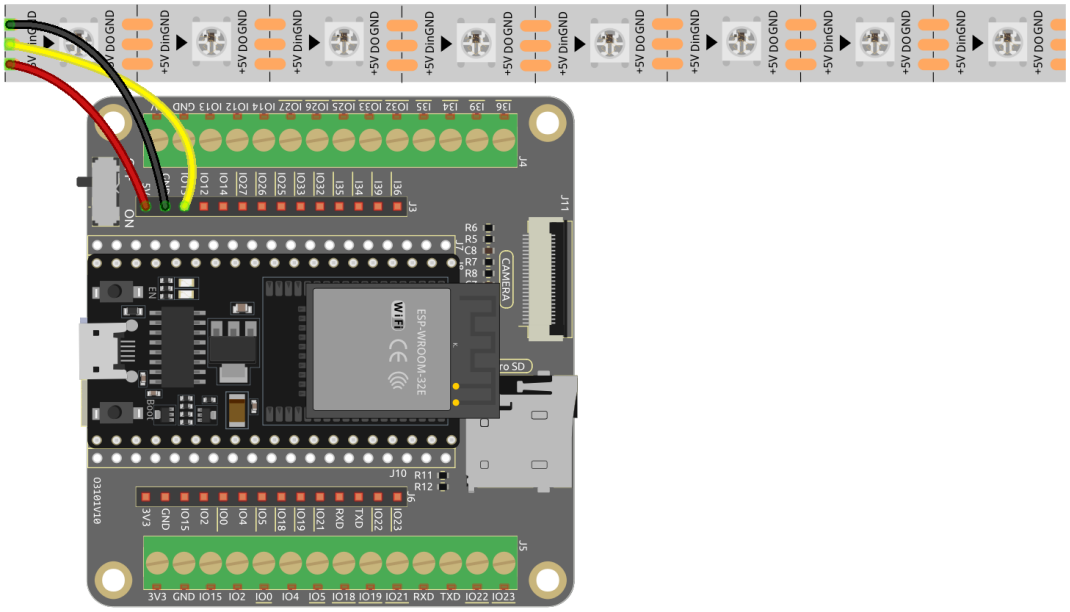
Name	ARTIKEL IN DIESEM SET	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

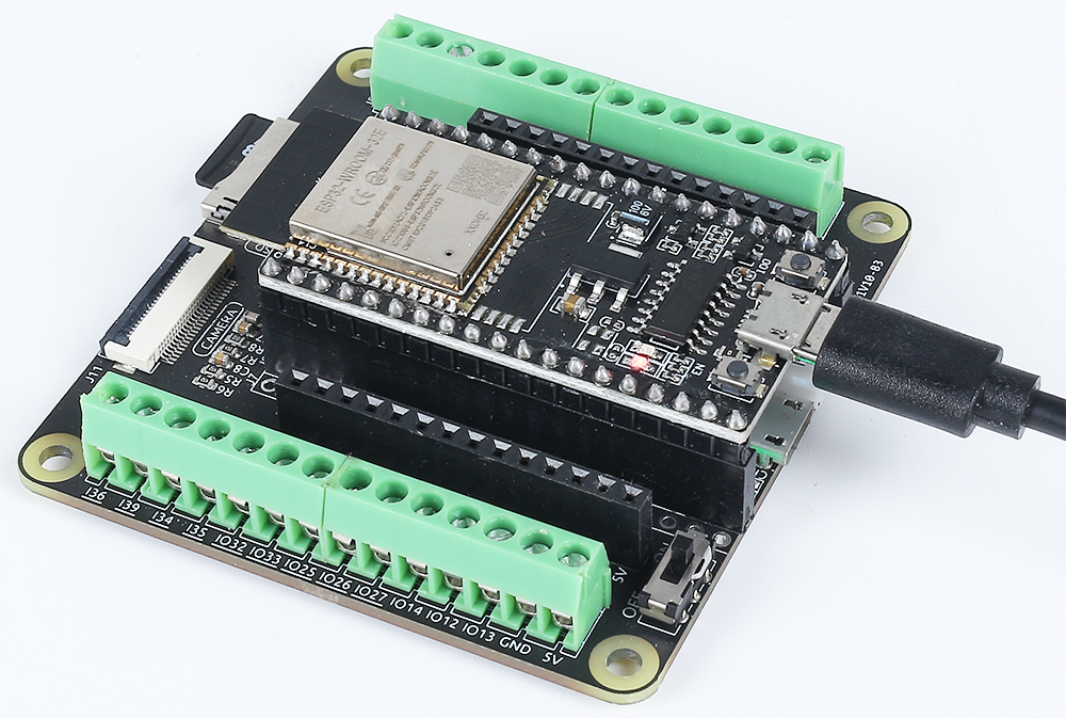
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
WS2812 RGB 8 LEDs Leiste	

Wie macht man das?

1. Bauen Sie den Schaltkreis.

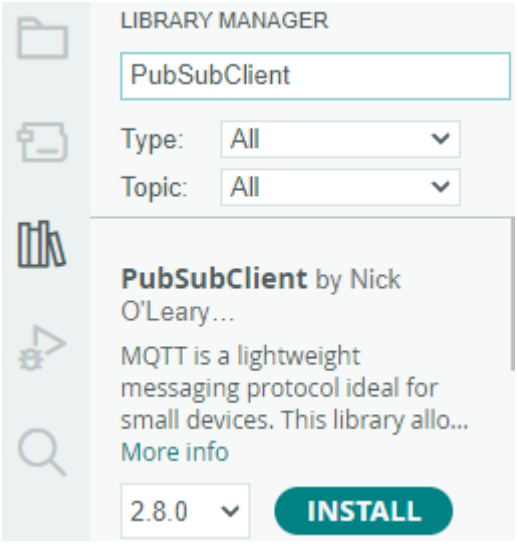


2. Verbinden Sie dann ESP32-WROOM-32E mit dem Computer über das USB-Kabel.



3. Öffnen Sie den Code.

    - Öffnen Sie die Datei `iot_5_cheerlights.ino`, die sich im Verzeichnis `esp32-starter-kit-main\c\codes\iot_5_cheerlights` befindet, oder kopieren Sie den Code in die Arduino IDE.
    - Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
    - *„Unbekanntes COMxx“ wird immer angezeigt?*
    - Hier werden die Bibliotheken `PubSubClient` und `Adafruit_NeoPixel` verwendet, die Sie über den **Library Manager** installieren können.



4. Suchen Sie die folgenden Zeilen und ändern Sie sie mit Ihrem `<SSID>` und `<PASSWORD>`.

```
// Replace the next variables with your SSID/Password combination
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

5. Finden Sie die nächste Zeile und ändern Sie Ihren `unique_identifier`. Stellen Sie sicher, dass Ihr `unique_identifier` wirklich einzigartig ist, da identische IDs, die versuchen, sich bei demselben **MQTT Broker** anzumelden, zu einem Anmeldefehler führen können.

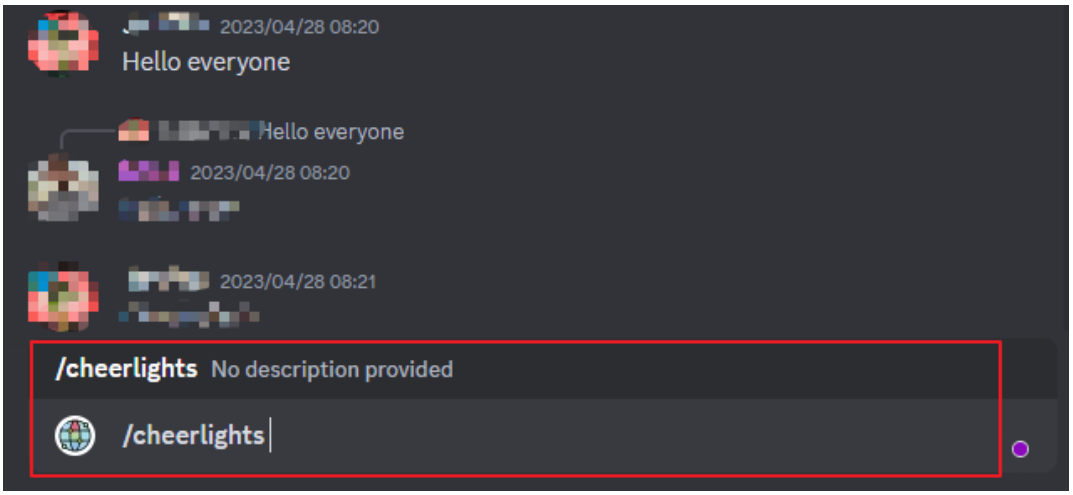
```
// Add your MQTT Broker address:
const char* mqtt_server = "mqtt.cheerlights.com";
const char* unique_identifier = "sunfounder-client-sdgvsasdda";
```

6. Nachdem Sie das richtige Board (ESP32 Dev Module) und den Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
7. An diesem Punkt können Sie sehen, dass Ihr RGB-Streifen eine bestimmte Farbe anzeigt. Platzieren Sie ihn auf Ihrem Schreibtisch und Sie werden bemerken, dass er periodisch die Farben wechselt. Das liegt daran, dass andere @CheerLights-Follower die Farbe Ihrer Lichter ändern!
8. Öffnen Sie den Seriellen Monitor. Sie werden Nachrichten ähnlich den folgenden sehen:

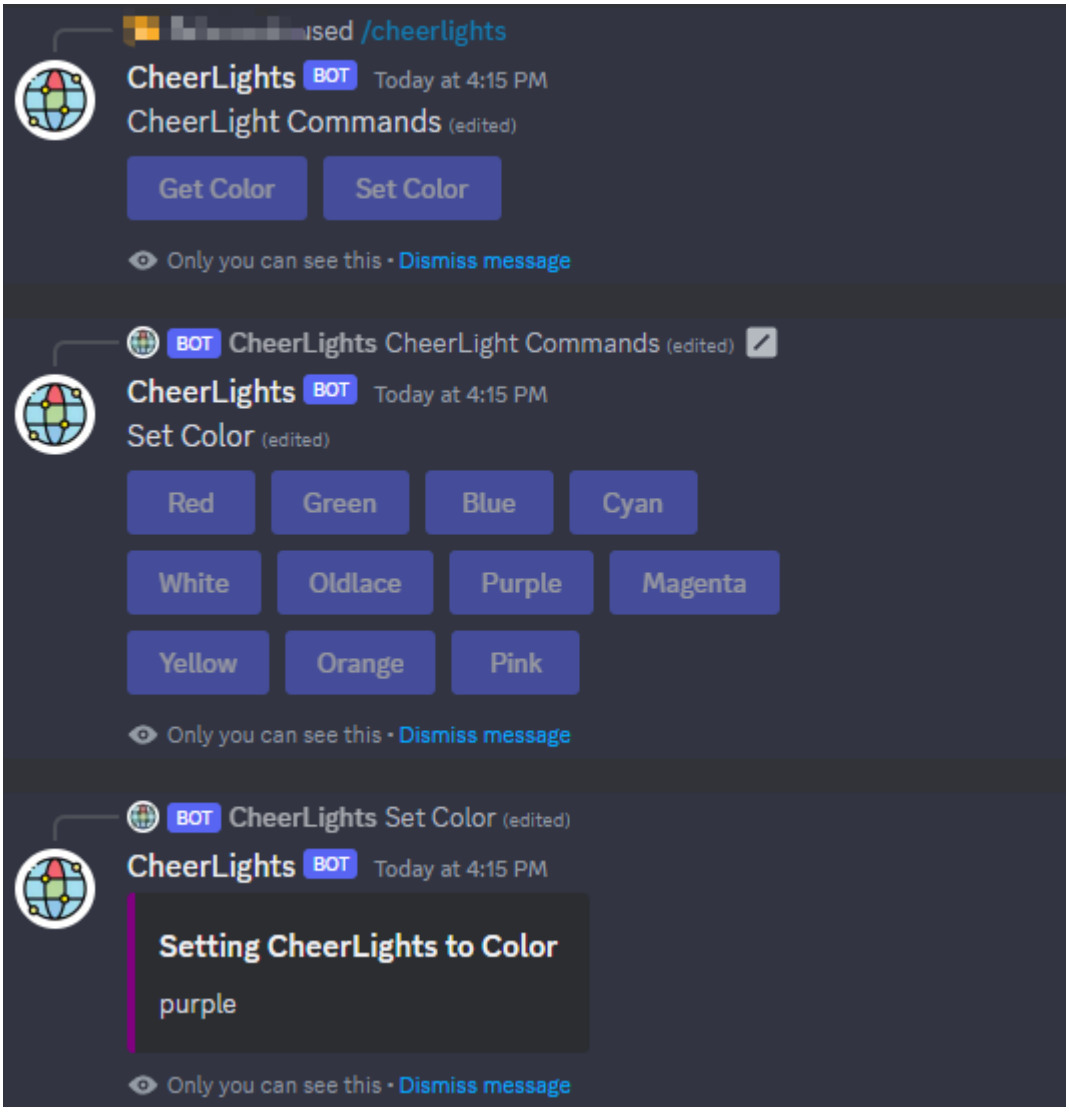
```
WiFi connected
IP address:
192.168.18.77
Attempting MQTT connection...connected
Message arrived on topic: cheerlights.
Message: oldlace
Changing color to oldlace
```

Globale @CheerLights-Geräte steuern

1. Treten Sie dem bei und nutzen Sie den CheerLights-Bot, um die Farbe festzulegen. Tippen Sie einfach `/cheerlights` in einem der Kanäle auf dem **CheerLights Discord Server**, um den Bot zu aktivieren.



2. Befolgen Sie die Anweisungen des Bots, um die Farbe festzulegen. Dadurch können Sie CheerLights-Geräte weltweit steuern.



2.49 8.6 Temperatur- und Feuchtigkeitsüberwachung mit Adafruit IO

In diesem Projekt werden wir Ihnen zeigen, wie Sie eine beliebte IoT-Plattform verwenden können. Es gibt viele kostenlose (oder kostengünstige) Plattformen online für Programmierbegeisterte. Einige Beispiele sind Adafruit IO, Blynk, Arduino Cloud, ThingSpeak und so weiter. Die Nutzung dieser Plattformen ist recht ähnlich. Hier konzentrieren wir uns auf Adafruit IO.

Wir werden ein Arduino-Programm schreiben, das den DHT11-Sensor verwendet, um Temperatur- und Feuchtigkeitsmessungen an das Dashboard von Adafruit IO zu senden. Sie können auch eine LED im Schaltkreis über einen Schalter im Dashboard steuern.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

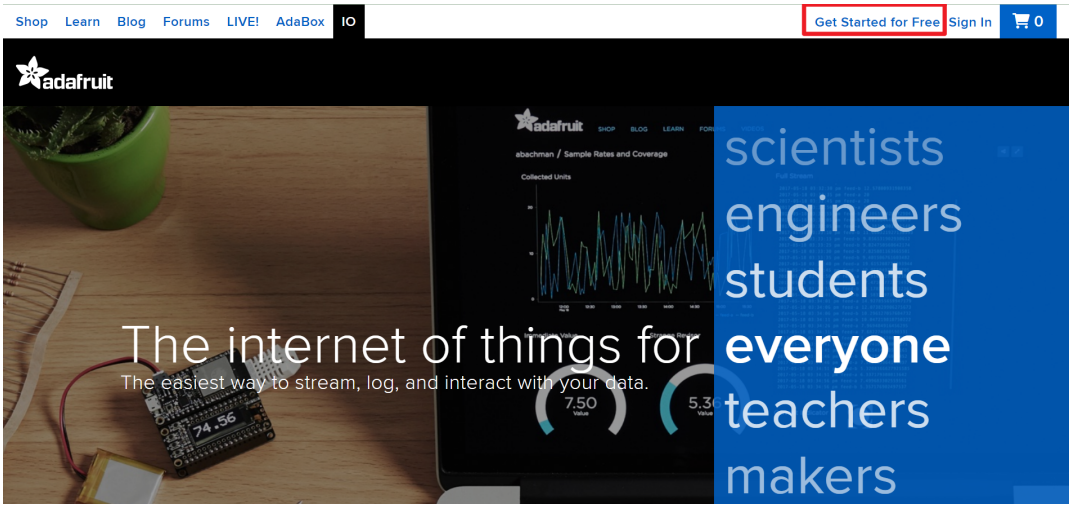
Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

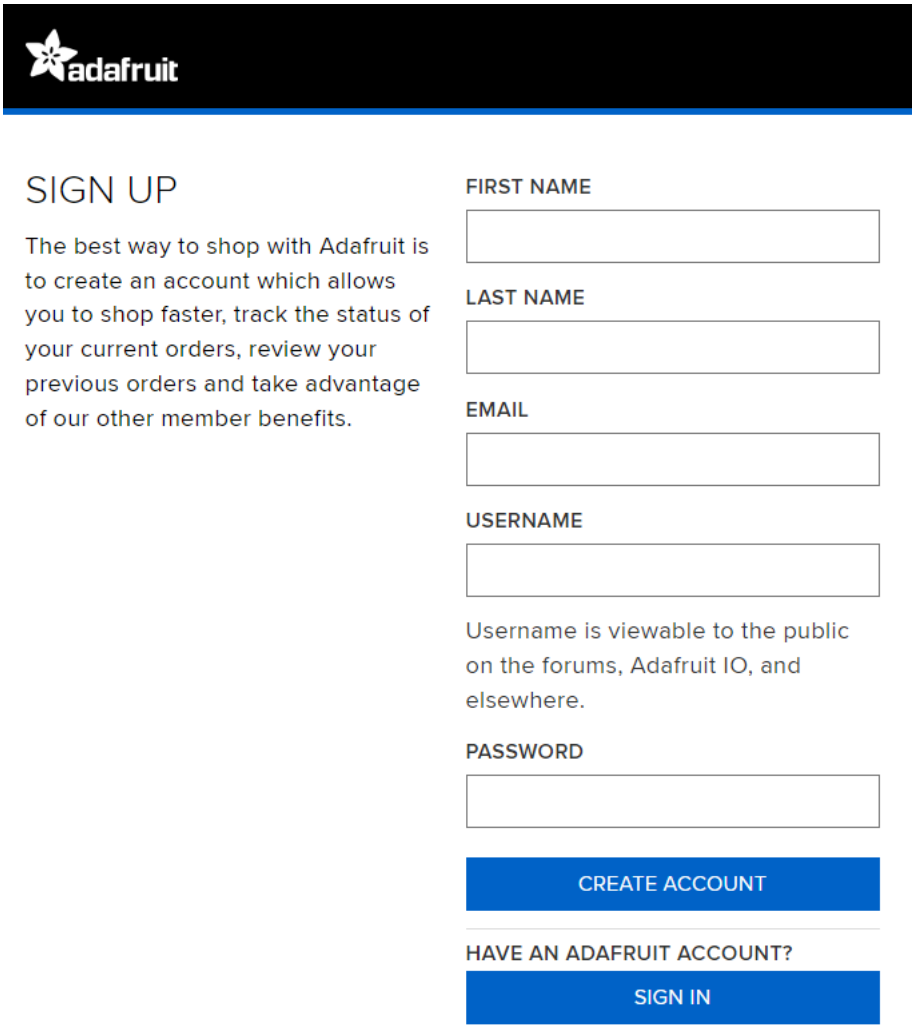
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
DHT11 Feuchtigkeits- und Temperatursensor	

Einrichten des Dashboards

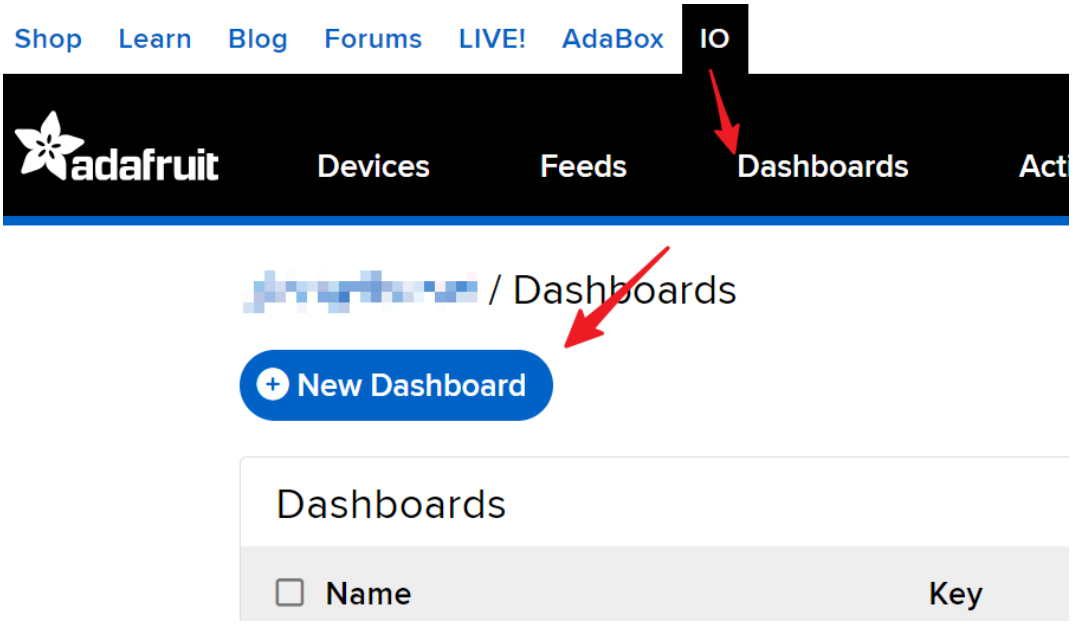
1. Besuchen Sie und klicken Sie auf **Start for free**, um ein kostenloses Konto zu erstellen.



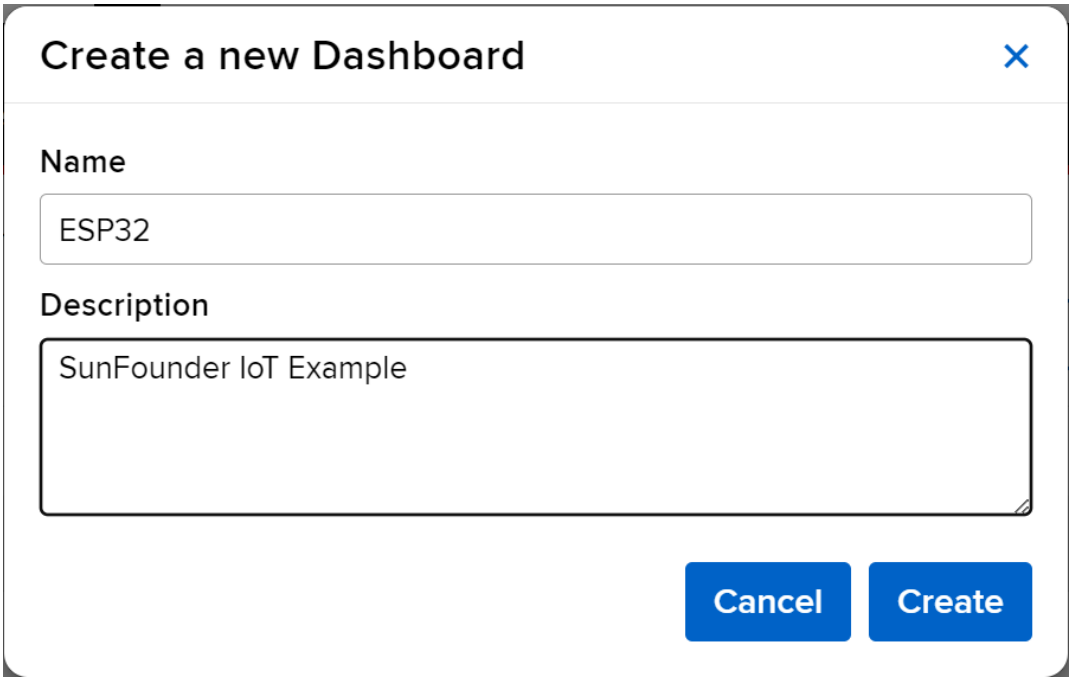
2. Füllen Sie das Formular aus, um ein Konto zu erstellen.



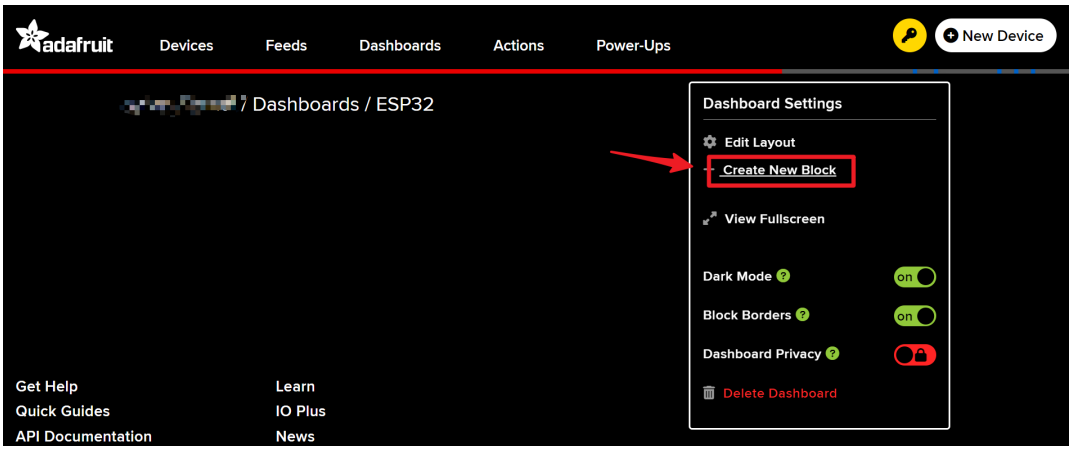
3. Nachdem Sie ein Adafruit-Konto erstellt haben, müssen Sie Adafruit io erneut öffnen. Klicken Sie auf **Dashboards** und dann auf **New Dashboard**.



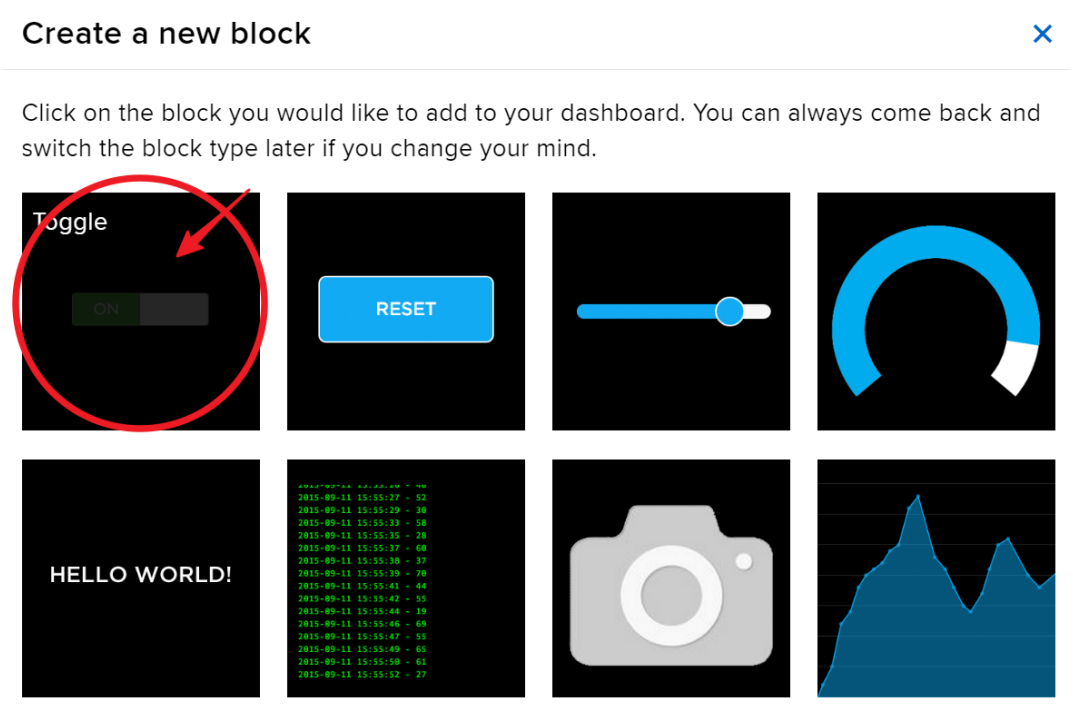
4. Erstellen Sie ein **New Dashboard**.



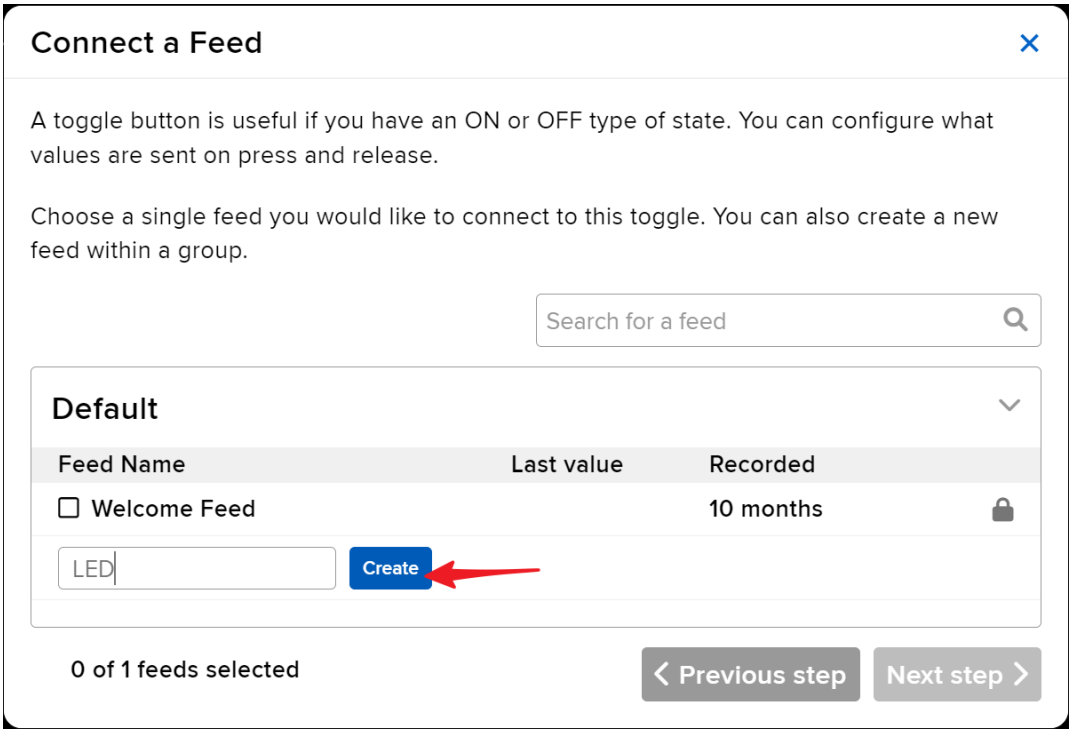
5. Betreten Sie das neu erstellte **Dashboard** und erstellen Sie einen neuen Block.



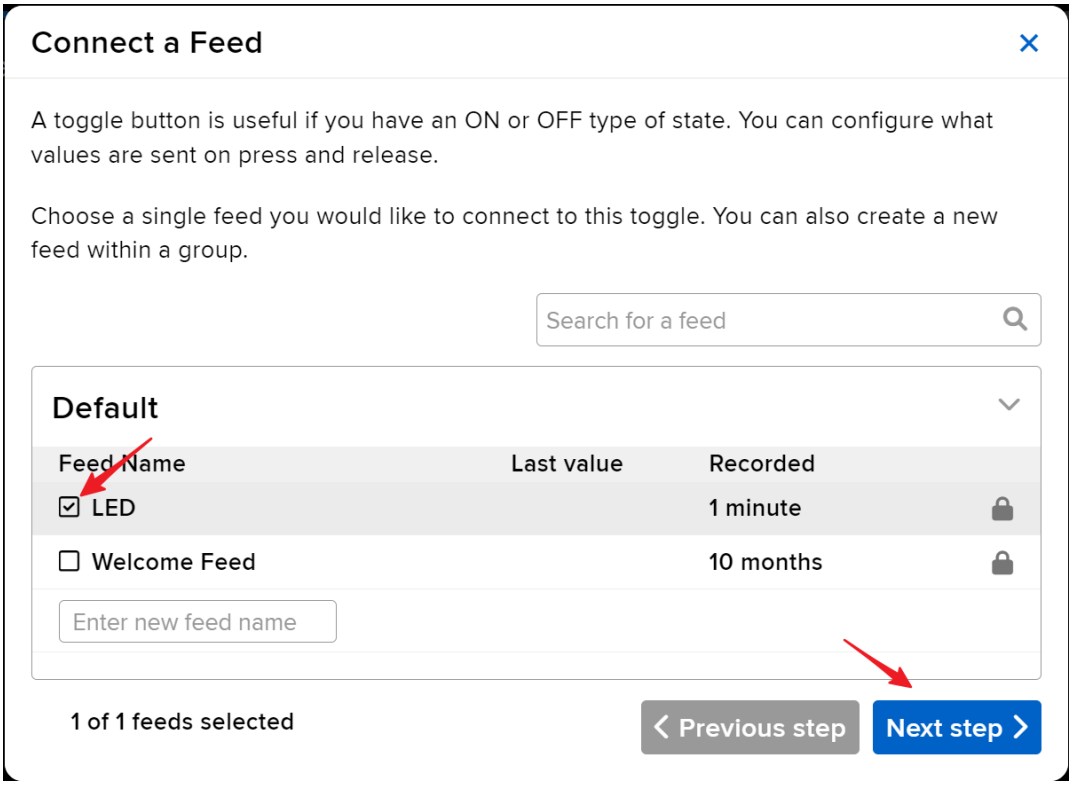
6. Erstellen Sie 1 **Toggle**-Block.



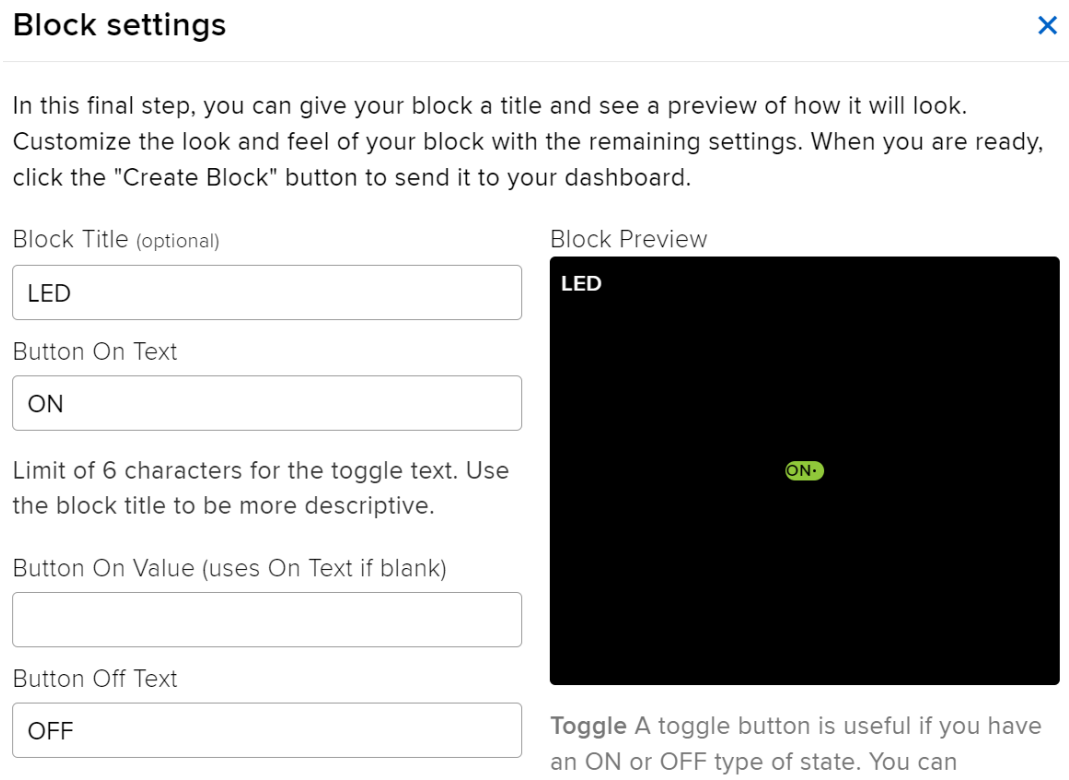
7. Als Nächstes müssen Sie hier einen neuen Feed erstellen. Dieser Toggle wird verwendet, um die LED zu steuern, und wir nennen diesen Feed „LED“.



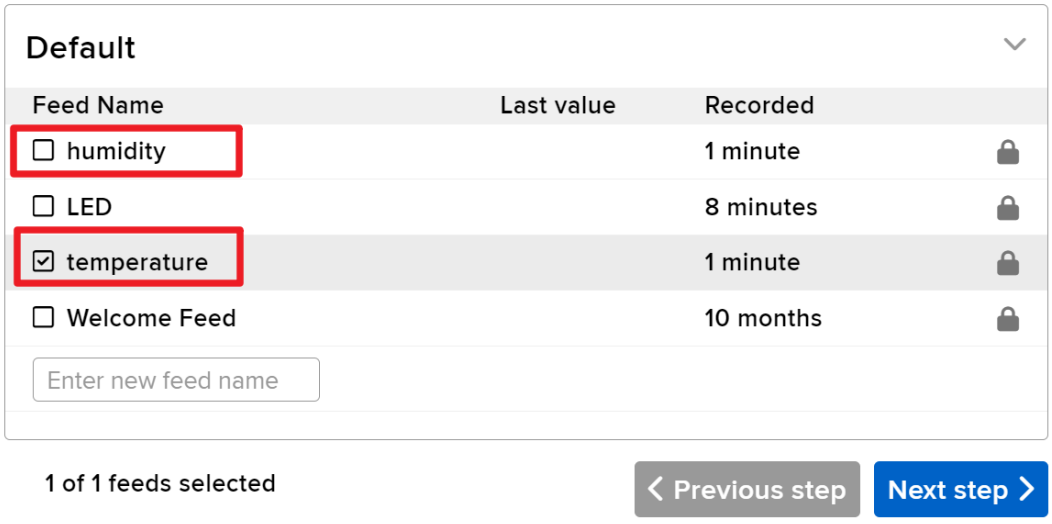
8. Überprüfen Sie den **LED**-Feed und gehen Sie dann zum nächsten Schritt über.



9. Vervollständigen Sie die Blockeinstellungen (hauptsächlich Blocktitel, On-Text und Off-Text) und klicken Sie dann unten rechts auf den Button **Create block**, um den Vorgang abzuschließen.

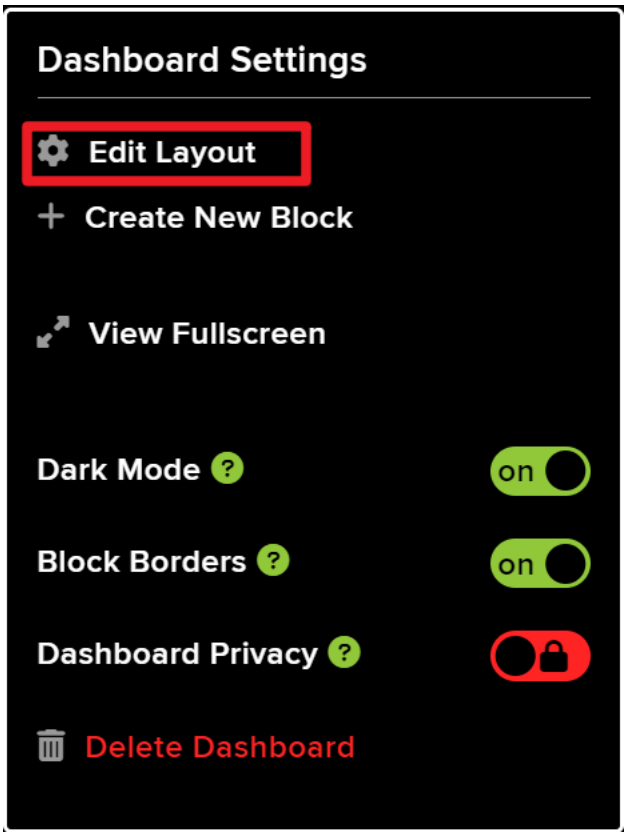


10. Als Nächstes müssen wir zwei **Text Blocks** erstellen. Sie werden verwendet, um Temperatur und Luftfeuchtigkeit anzuzeigen. Erstellen Sie also zwei Feeds mit den Namen **temperature** und **humidity**.

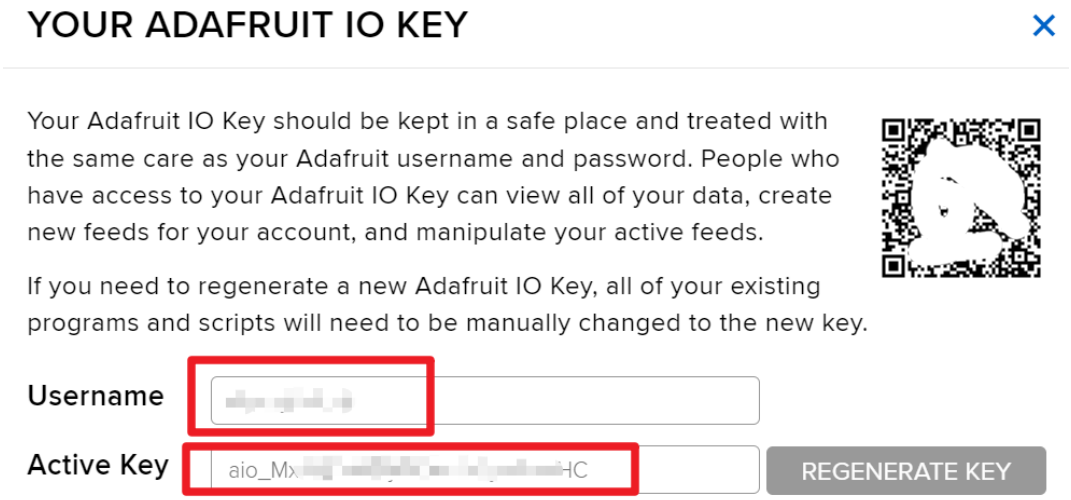


11. Nach der Erstellung sollte Ihr Dashboard ungefähr so aussehen:

12. Sie können das Layout mit der Option **Edit Layout** auf dem Dashboard anpassen.

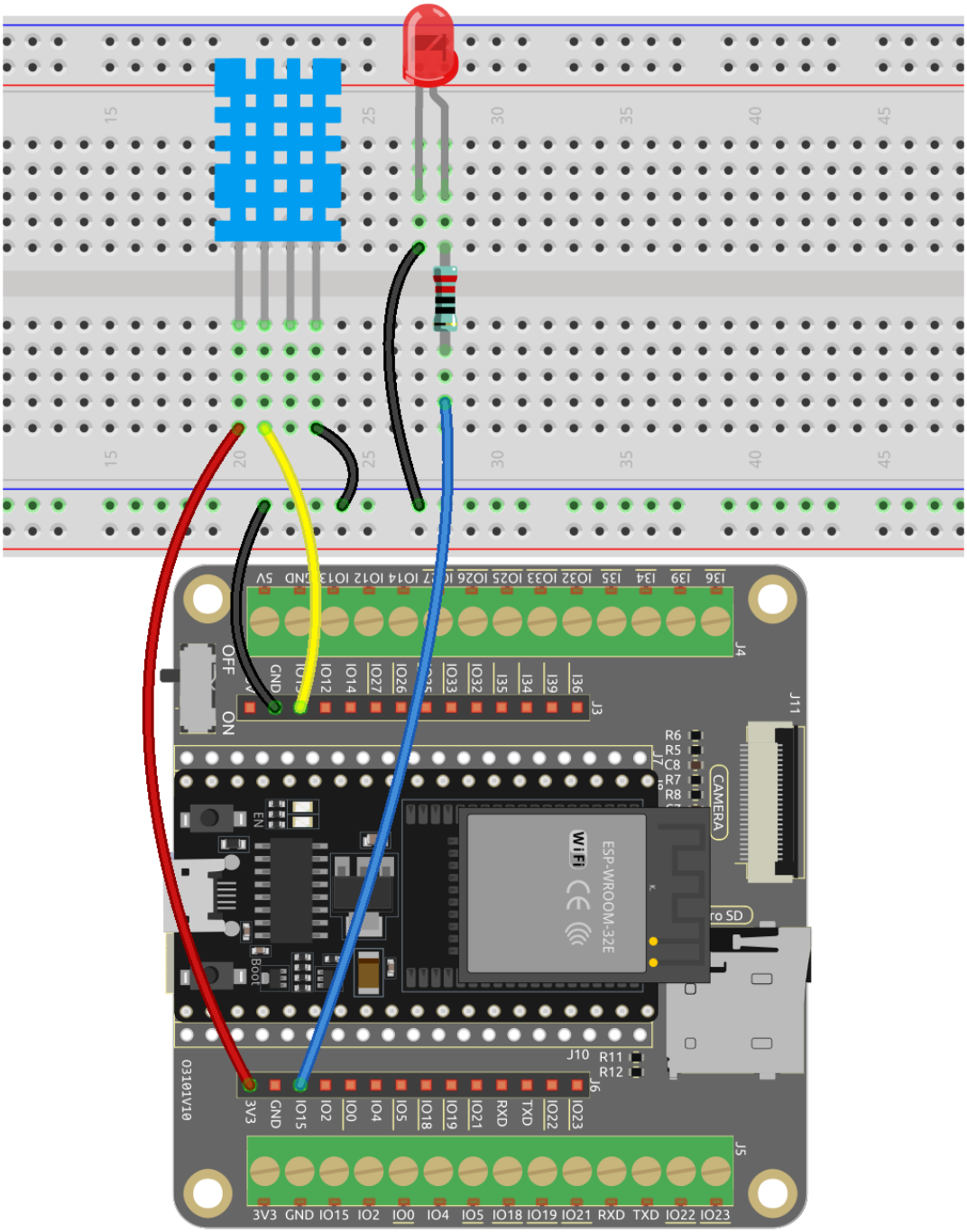


13. Klicken Sie auf **API KEY**, und Ihr Benutzername und **API KEY** werden angezeigt. Notieren Sie sich diese, da Sie sie für Ihren Code benötigen.

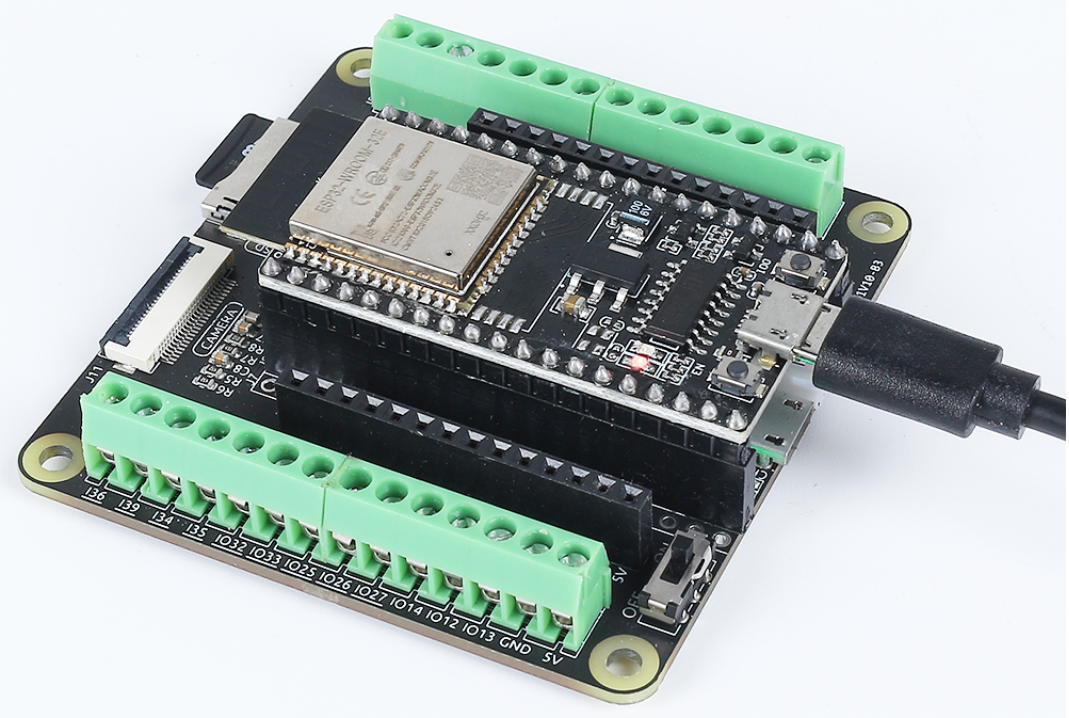


Code ausführen

1. Bauen Sie den Schaltkreis.



2. Verbinden Sie dann ESP32-WROOM-32E mit dem Computer über das USB-Kabel.



3. Öffnen Sie den Code.

    - Öffnen Sie die Datei `iot_6_adafruit_io.ino`, die sich im Verzeichnis `esp32-starter-kit-main\c\codes\iot_6_adafruit_io` befindet, oder kopieren Sie den Code in die Arduino IDE.
    - Nachdem Sie das Board (ESP32 Dev Module) und den passenden Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.
    - *„Unbekanntes COMxx“ wird immer angezeigt?*
    - Hier werden die Bibliotheken `Adafruit_MQTT Library` und `DHT sensor library` verwendet, die Sie über den **Library Manager** installieren können.

4. Finden Sie die folgenden Zeilen und ersetzen Sie <SSID> und <PASSWORD> mit den spezifischen Details Ihres WLAN-Netzwerks.

```
/************************* WiFi Access Point ␣
↪*********************************/

#define WLAN_SSID "<SSID>"
#define WLAN_PASS "<PASSWORD>"
```

5. Ersetzen Sie dann <YOUR_ADAFRUIT_IO_USERNAME> mit Ihrem Adafruit IO-Benutzernamen und <YOUR_ADAFRUIT_IO_KEY> mit dem **API KEY**, den Sie gerade kopiert haben.

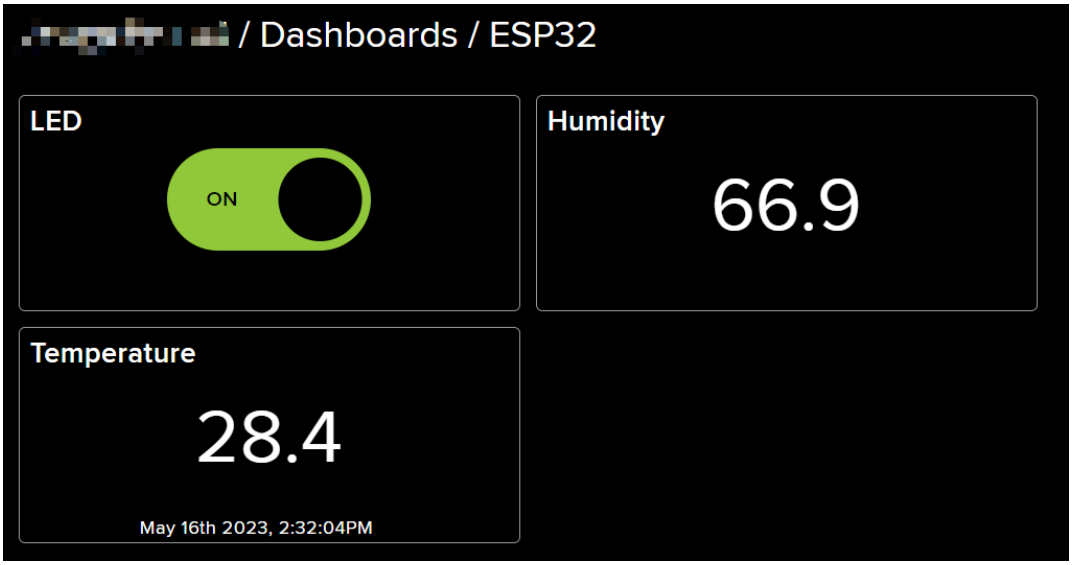
```
// Adafruit IO Account Configuration
// (to obtain these values, visit https://io.adafruit.com and click on␣
↪Active Key)
#define AIO_USERNAME "<YOUR_ADAFRUIT_IO_USERNAME>"
#define AIO_KEY      "<YOUR_ADAFRUIT_IO_KEY>"
```

6. Nachdem Sie das richtige Board (ESP32 Dev Module) und den Port ausgewählt haben, klicken Sie auf den **Upload**-Knopf.

7. Sobald der Code erfolgreich hochgeladen wurde, werden Sie die folgende Meldung im seriellen Monitor beobachten, die auf eine erfolgreiche Kommunikation mit Adafruit IO hinweist.

```
Adafruit IO MQTTS (SSL/TLS) Example


Connecting to xxxxx
WiFi connected
IP address:
192.168.18.76
Connecting to MQTT... MQTT Connected!
Temperature: 27.10
Humidity: 61.00
```

8. Navigieren Sie zurück zu Adafruit IO. Jetzt können Sie die Temperatur- und Luftfeuchtigkeitsmessungen auf dem Dashboard beobachten oder den LED-Kippschalter nutzen, um den Ein-/Ausschaltzustand der externen LED zu steuern, die mit dem Schaltkreis verbunden ist.



2.50 8.7 ESP-Kamera mit Telegram-Bot

In diesem Projekt zeigen wir, wie Sie das ESP32 mit Ihrer Lieblings-Messaging-Anwendung integrieren können. Für diese Demonstration verwenden wir Telegram.

Erstellen Sie einen Telegram-Bot, der es Ihnen ermöglicht, Ihren Schaltkreis von überall zu steuern, Fotos zu machen und den Blitz zu verwalten. Außerdem wird jedes Mal, wenn jemand an Ihrem Gerät vorbeigeht, ein neues Foto aufgenommen und eine Benachrichtigung an Ihr Telegram-Konto gesendet.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

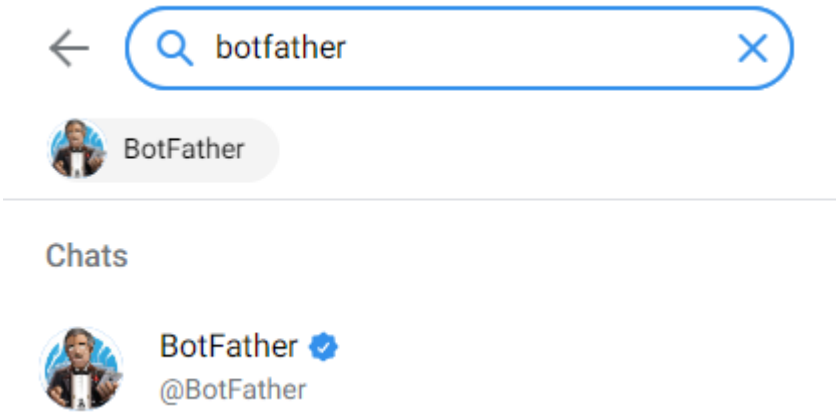
Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

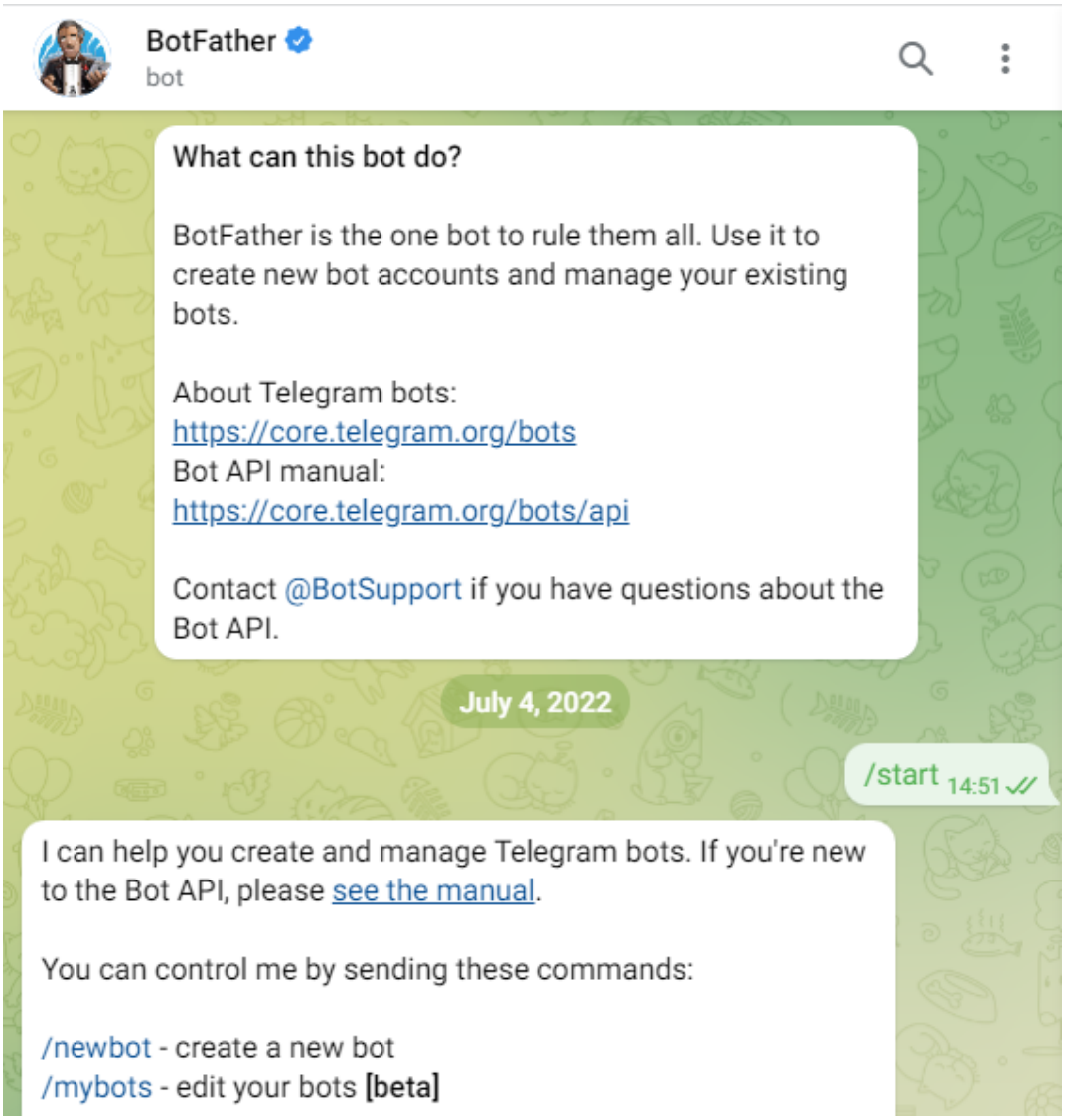
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
PIR-Bewegungssensormodul	

Einen Telegram-Bot erstellen

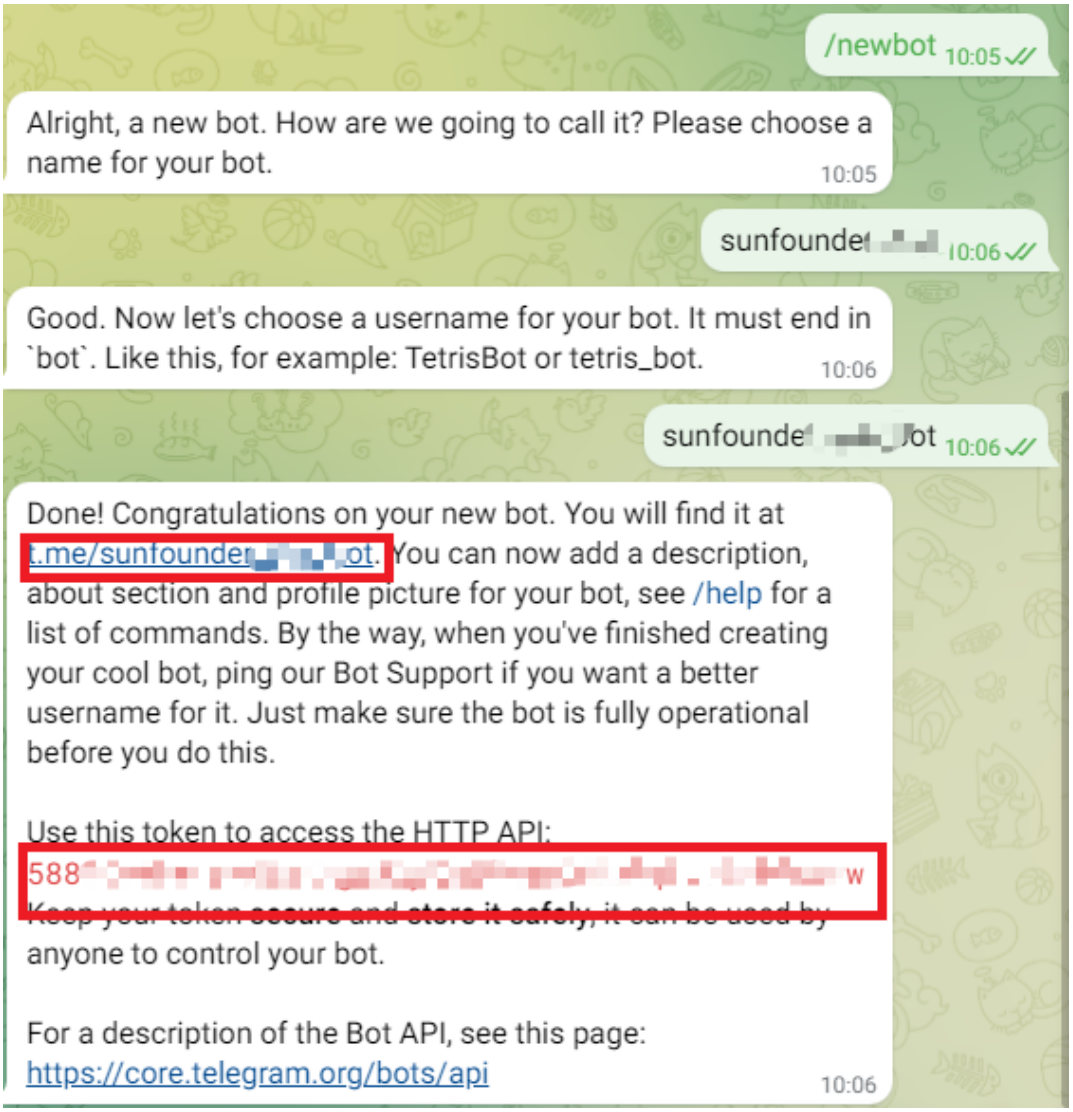
1. Gehen Sie zu **Google Play** oder dem **App Store** und laden und installieren Sie **Telegram**.
2. Suchen Sie in der **Telegram**-App nach `botfather`, und sobald er erscheint, klicken Sie darauf, um ihn zu öffnen. Alternativ können Sie direkt diesen Link verwenden: t.me/botfather.



3. Wenn Sie den Chat geöffnet haben, senden Sie den Befehl `/start`.



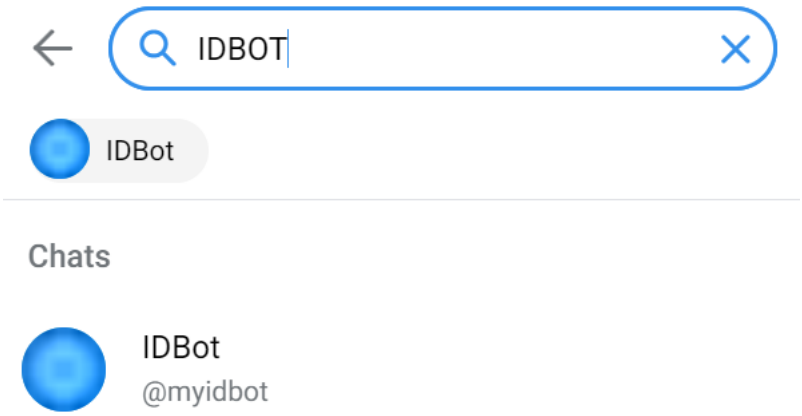
4. Geben Sie `/newbot` ein und folgen Sie den angegebenen Anweisungen, um Ihren Bot zu erstellen. Nach erfolgreichem Abschluss gibt Ihnen der BotFather den Zugangslink und die API für Ihren neuen Bot.



Autorisierung von Telegram-Benutzern

Da jeder mit dem von Ihnen erstellten Bot interagieren kann, besteht ein Risiko des Informationslecks. Um dies zu verhindern, möchten wir, dass der Bot nur auf autorisierte Benutzer reagiert.

1. Suchen Sie in Ihrem **Telegram**-Konto nach `IDBot` oder öffnen Sie den Link: t.me/myidbot.

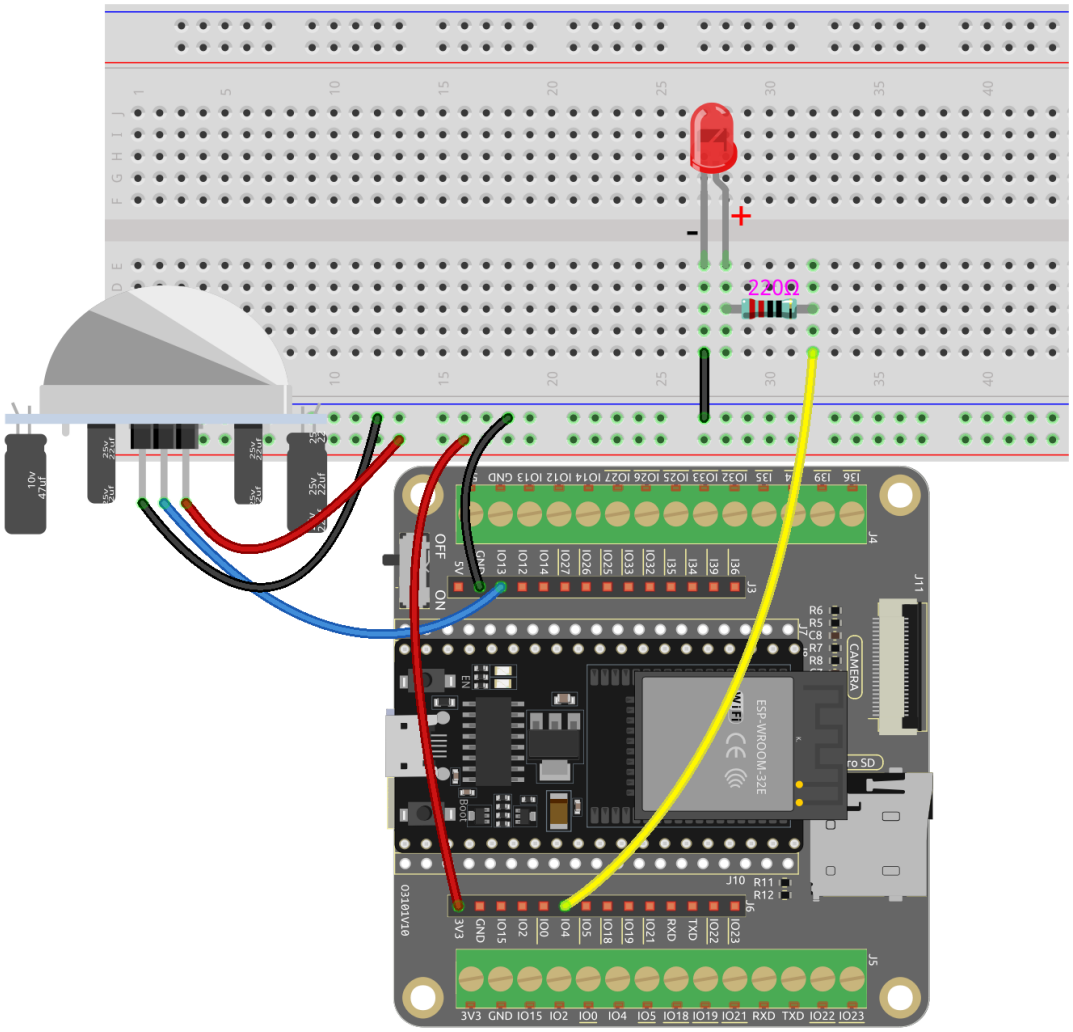


2. Senden Sie den Befehl `/getid`. Speichern Sie die bereitgestellte ID zur späteren Verwendung in unserem Programm.

Code hochladen

1. Schließen Sie zuerst die Kamera an.

2. Bauen Sie den Schaltkreis auf.



3. Öffnen Sie den Code.

    - Öffnen Sie die Datei `iot_7_cam_telegram.ino`, die sich im Verzeichnis `esp32-starter-kit-main\c\codes\iot_7_cam_telegram` befindet, oder kopieren Sie den Code in die Arduino IDE.
    - Nachdem Sie das Board (ESP32 Dev Module) und den entsprechenden Port ausgewählt haben, klicken Sie auf den **Upload**-Button.
    - *„Unbekanntes COMxx“ wird immer angezeigt?*
    - Die Bibliotheken `UniversalTelegramBot` und `ArduinoJson` werden hier verwendet. Sie können sie aus dem **Library Manager** installieren.

4. Suchen und ändern Sie die folgenden Zeilen mit Ihren WLAN-Daten und ersetzen Sie <SSID> und <PASSWORD>:

```
// Replace the next variables with your SSID/Password combination
const char* ssid = "<SSID>";
const char* password = "<PASSWORD>";
```

5. Aktualisieren Sie die nächste Zeile, indem Sie <CHATID> durch Ihre Telegram-ID ersetzen, die Sie von @IDBot erhalten haben.

```
// Use @myidbot to find out the chat ID of an individual or a group
// Also note that you need to click "start" on a bot before it can
// message you
String chatId = "<CHATID>";
```

6. Aktualisieren Sie die nächste Zeile, indem Sie <BOTTOKEN> durch den Token Ihres Telegram-Bots ersetzen, den Ihnen @BotFather bereitgestellt hat.

```
// Initialize Telegram BOT
String BOTtoken = "<BOTTOKEN>";
```

7. Nachdem Sie das richtige Board (ESP32 Dev Module) und den Port ausgewählt haben, klicken Sie auf den **Upload**-Button.
8. Öffnen Sie den Serial Monitor. Wenn eine IP-Adresse gedruckt wird, deutet dies auf eine erfolgreiche Ausführung hin.

```
Connecting to xxxx
ESP32-CAM IP Address: 192.168.18.76
Init Done!
```

9. Jetzt können Sie über Telegram mit Ihrem ESP32 interagieren.

2.51 8.8 Kamera mit Home Assistant

Dieses Projekt führt Sie durch die Einrichtung eines Video-Stream-Web-Servers für die ESP32-Kamera und dessen Integration in die beliebte Heimautomatisierungsplattform Home Assistant. Diese Integration ermöglicht Ihnen den Zugriff auf den Server von jedem Gerät in Ihrem Netzwerk aus.

Bemerkung: Bevor Sie mit diesem Projekt beginnen, benötigen Sie ein Betriebssystem mit installiertem Home Assistant.

Wir empfehlen die Installation des Home Assistant OS auf einem Raspberry Pi.

Falls Sie keinen Raspberry Pi besitzen, können Sie es auch auf einer virtuellen Maschine unter Windows, macOS oder Linux installieren.

Anweisungen zur Installation finden Sie auf der offiziellen Website: https://www.home-assistant.io/installation/

Fahren Sie mit diesem Projekt erst nach einer erfolgreichen Installation fort.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

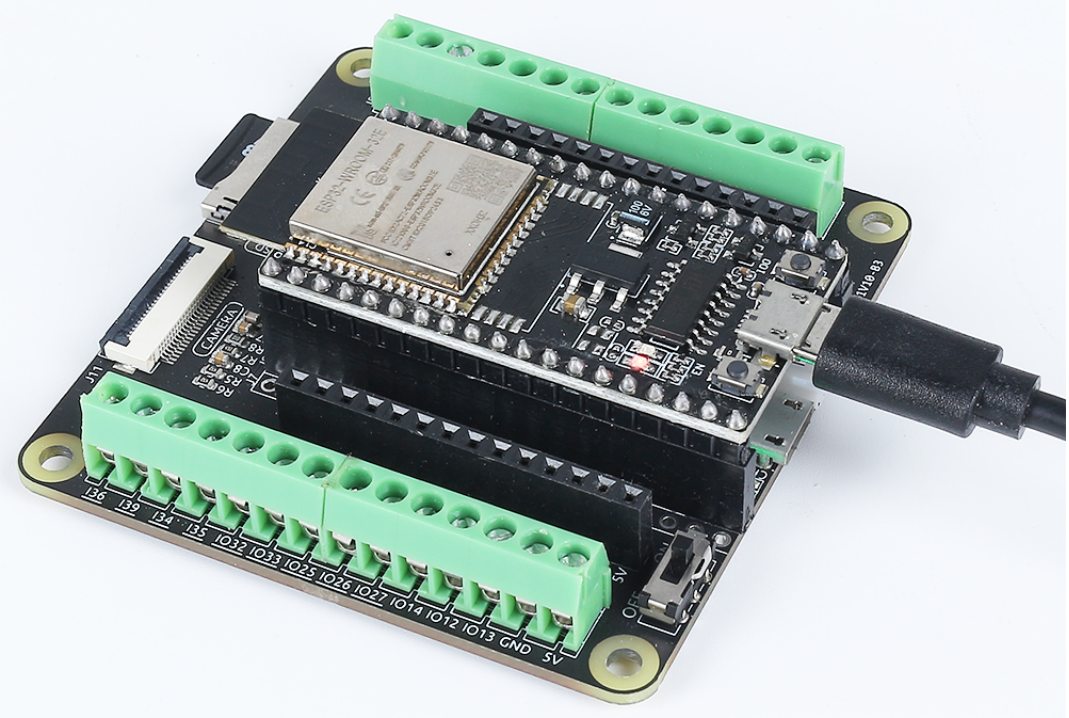
Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-

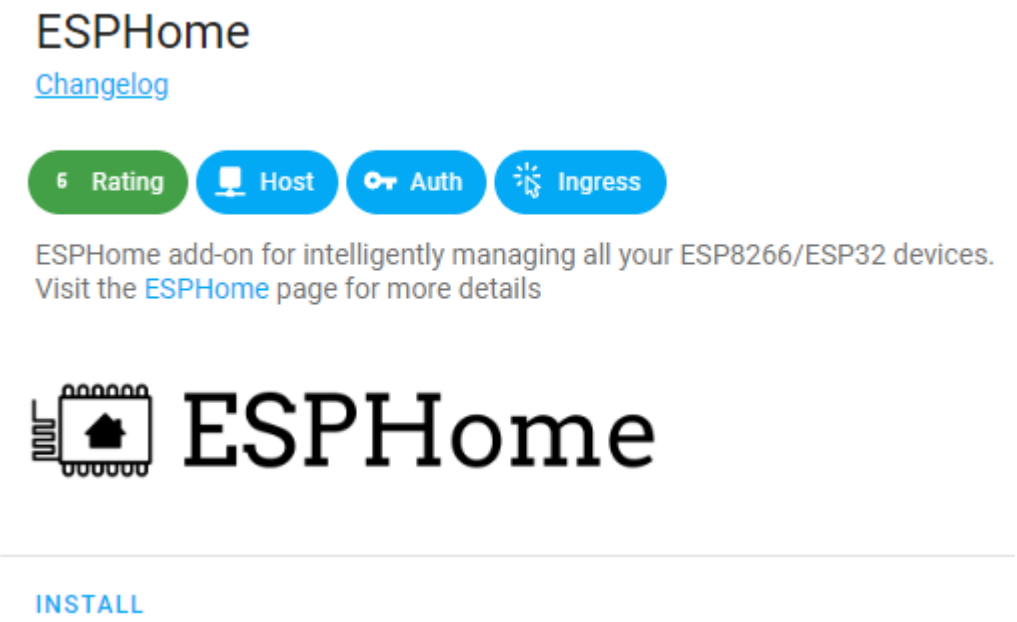
1. Konfiguration in ESPHome

1. Stecken Sie zuerst die Kamera ein.

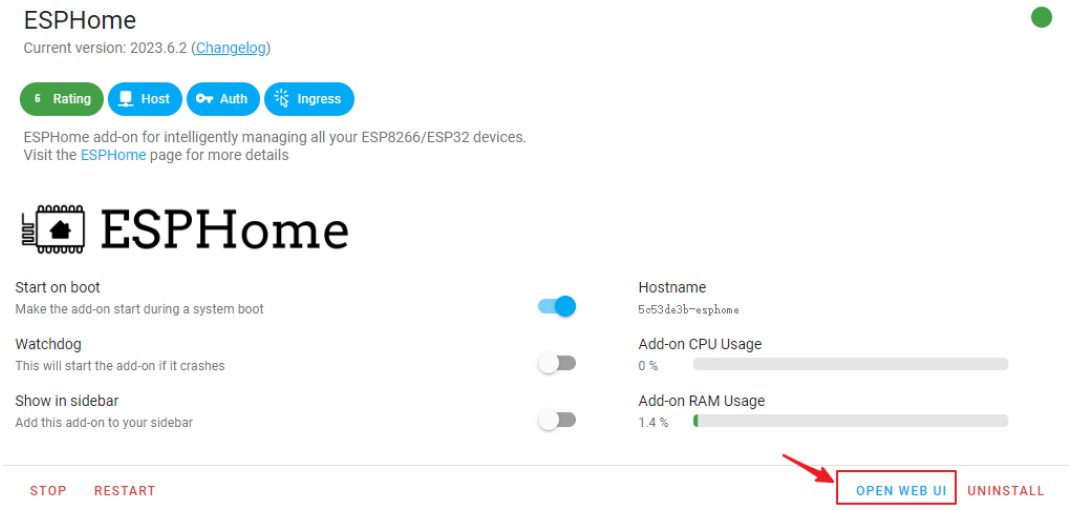
2. Verbinden Sie Ihr ESP32 mit dem Host, auf dem Sie das Home Assistant System installiert haben (z.B. wenn es auf einem Raspberry Pi installiert ist, verbinden Sie es mit dem Pi).



3. Installieren Sie das ESPHome Addon.

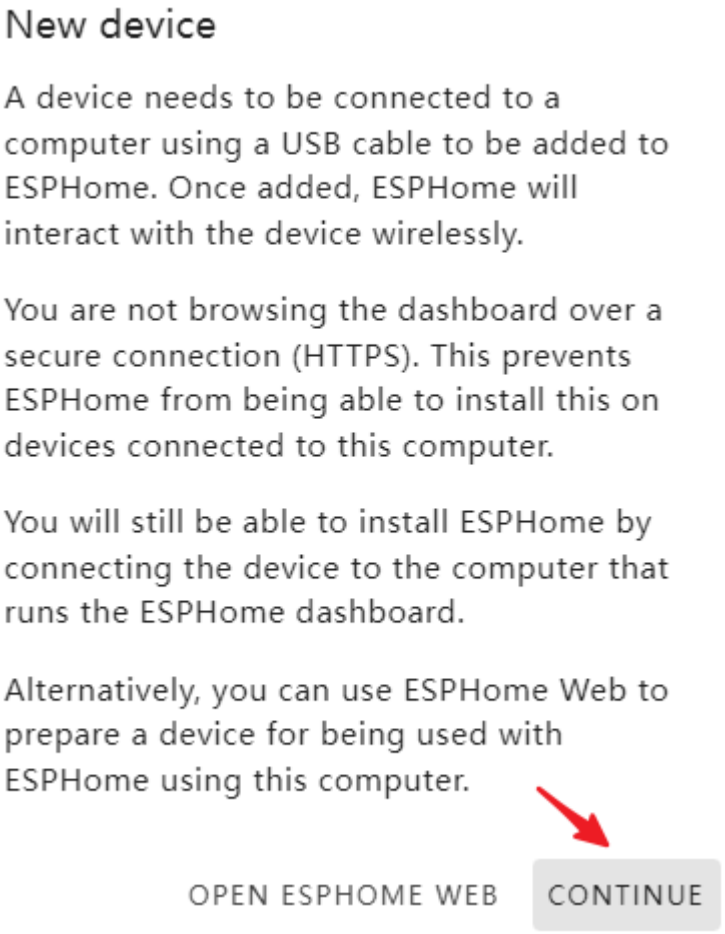


4. Klicken Sie auf **START**, dann auf **OPEN WEB UI**.

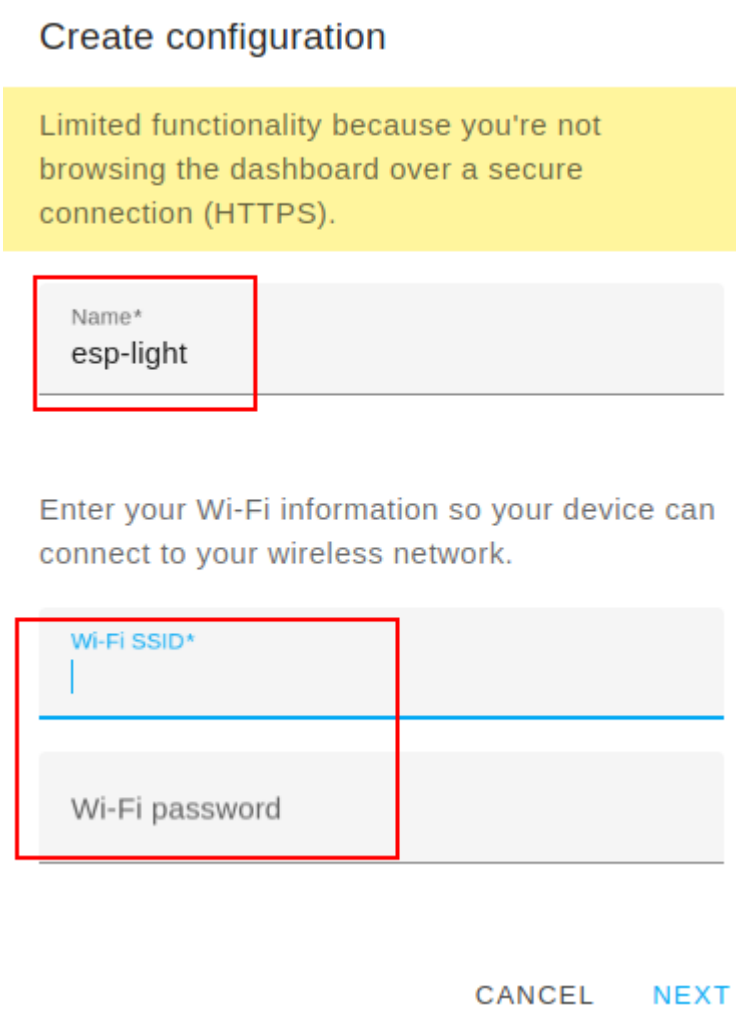


5. Fügen Sie neue Geräte hinzu.

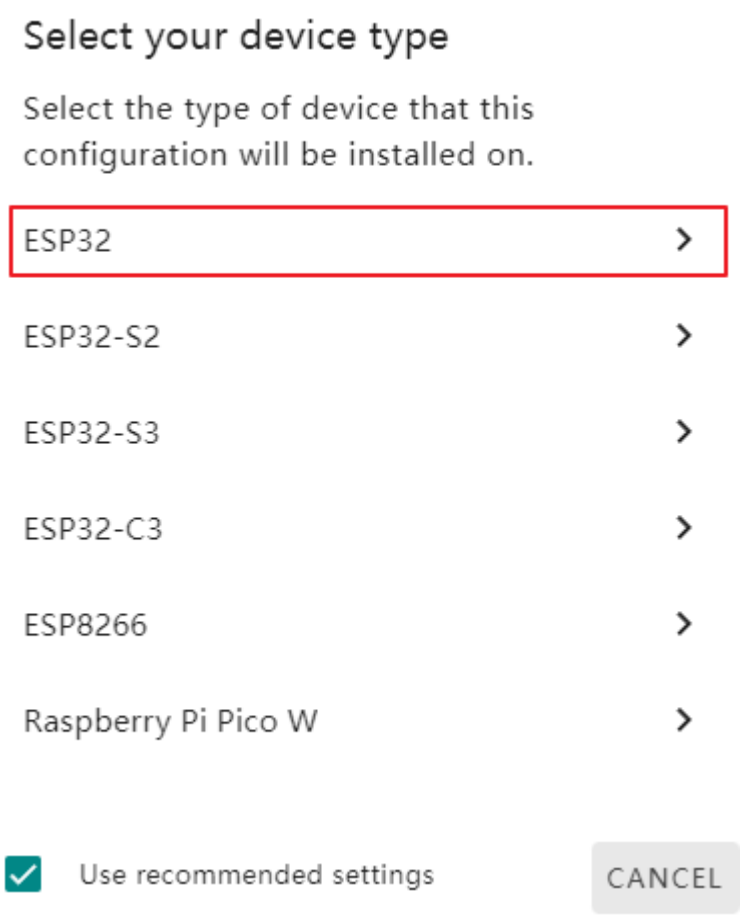
6. Es könnte eine Aufforderung erscheinen. Klicken Sie auf **CONTINUE**.



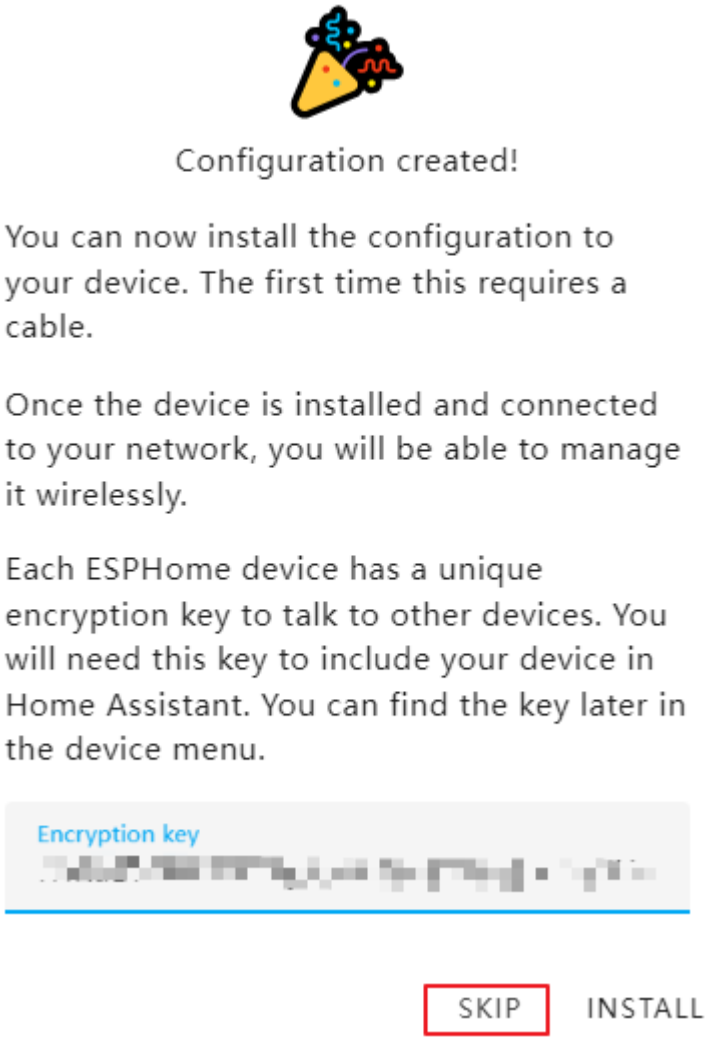
7. Erstellen Sie eine Konfiguration. Hier können Sie für **Name** einen beliebigen Namen eingeben. Für das WLAN geben Sie die Details des Netzwerks ein, in dem sich Ihr Home Assistant System befindet.



8. Wählen Sie den **ESP32** als Gerätetyp.

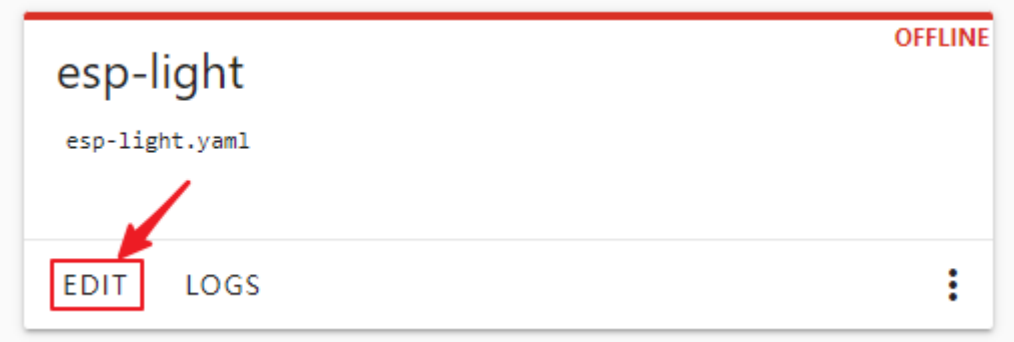


9. Wenn Sie ein Feuerwerksfeier-Symbol sehen, bedeutet dies, dass Sie das Gerät erfolgreich erstellt haben. Klicken Sie auf überspringen (NICHT auf **INSTALL** klicken).



An diesem Punkt haben Sie das Gerät nur in ESPHome hinzugefügt. Um das ESP32-Modul in Home Assistant zu integrieren, sind zusätzliche Konfigurationen notwendig:

10. Klicken Sie auf **EDIT**.



11. Nachdem Sie die `.yaml`-Schnittstelle geöffnet haben, ändern Sie `ssid` und `password` mit Ihren WLAN-Daten.

× esp-light.yaml

```yaml
esphome:
  name: esp-light
  friendly_name: esp-light

esp32:
  board: esp32dev
  framework:
    type: arduino

# Enable logging
logger:

# Enable Home Assistant API
api:
  encryption:
    key: "[redacted]"

ota:
  password: "[redacted]"

wifi:
  ssid: "[redacted]"
  password: "[redacted]"

  # Enable fallback hotspot (captive portal) in case wifi connection fails
  ap:
    ssid: "Esp-Light Fallback Hotspot"
    password: "F61FWfE2yH3f"

captive_portal:
```

12. Fügen Sie im Abschnitt `captive_portal` den folgenden Code ein:

```
# Example configuration entry
esp32_camera:
    external_clock:
        pin: GPIO0
        frequency: 20MHz
    i2c_pins:
        sda: GPIO26
        scl: GPIO27
    data_pins: [GPIO5, GPIO18, GPIO19, GPIO21, GPIO36, GPIO39, GPIO34,␣
↪GPIO35]
    vsync_pin: GPIO25
    href_pin: GPIO23
    pixel_clock_pin: GPIO22
    power_down_pin: GPIO32

    # Image settings
    name: My Camera
```

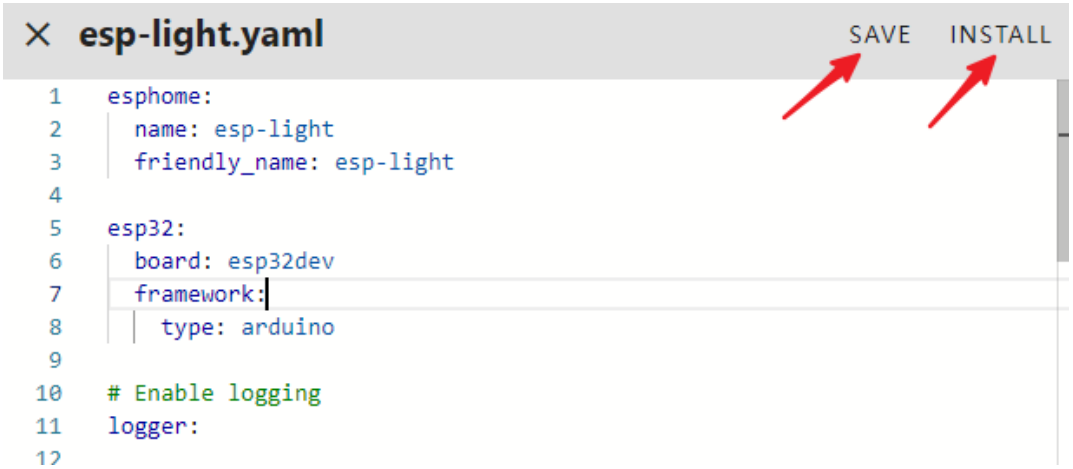
(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

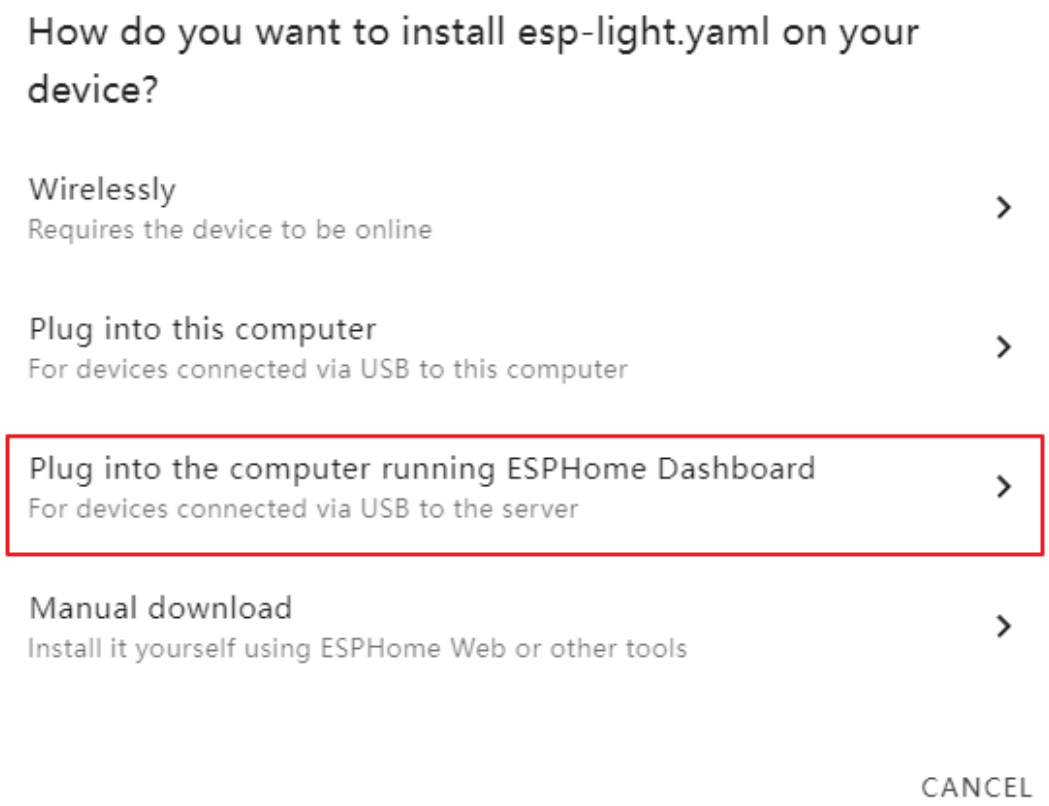
```
    # ...
```

Bemerkung: Für weitere Details zur `.yaml`-Konfiguration für ESP32 können Sie sich auf ESP32 Kamera - ESPHome beziehen.

13. **Save** Sie und klicken Sie dann auf **INSTALL**.



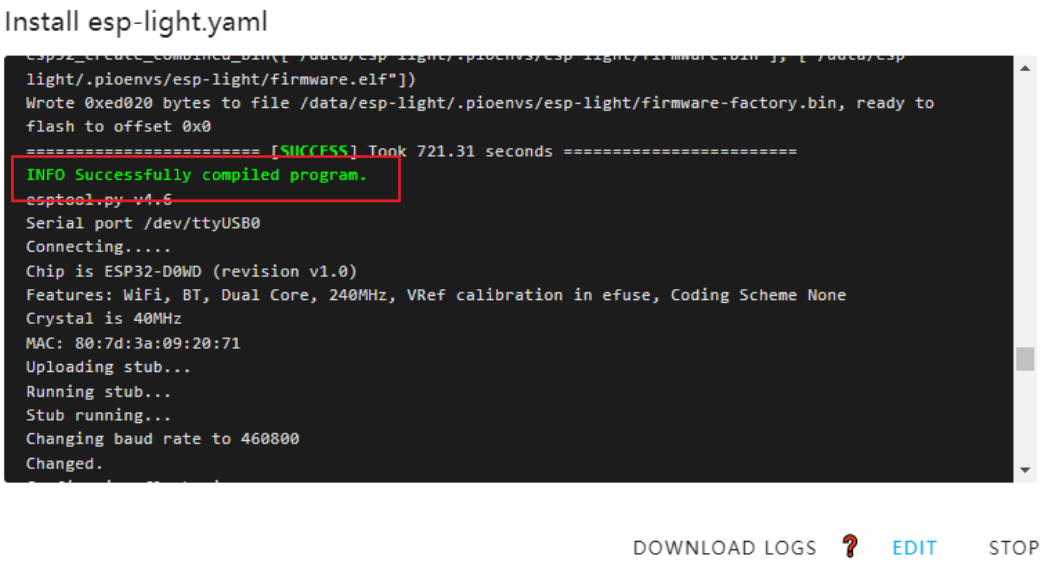
14. Wählen Sie die USB-Port-Methode zur Installation.



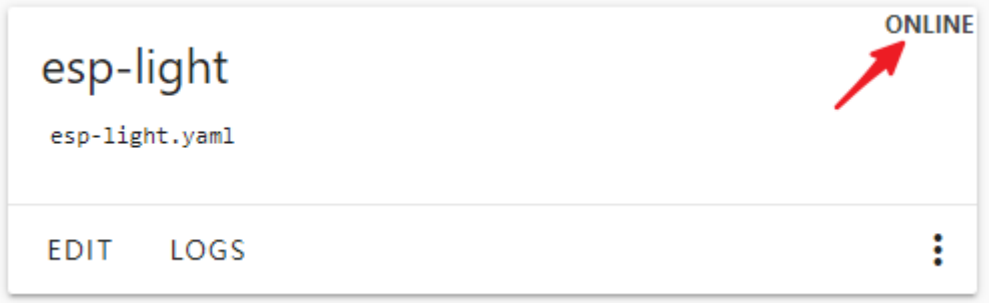
Bemerkung: Die erste Kompilierung wird Abhängigkeitspakete herunterladen, was etwa 10 Minu-

ten dauern kann. Bitte haben Sie Geduld. Wenn der Prozess lange Zeit stillsteht, überprüfen Sie, ob genügend Speicherplatz auf Ihrem System vorhanden ist.

15. Warten Sie auf die Nachricht `INFO Successfully compiled program.`, was anzeigt, dass die Firmware-Kompilierung abgeschlossen ist.



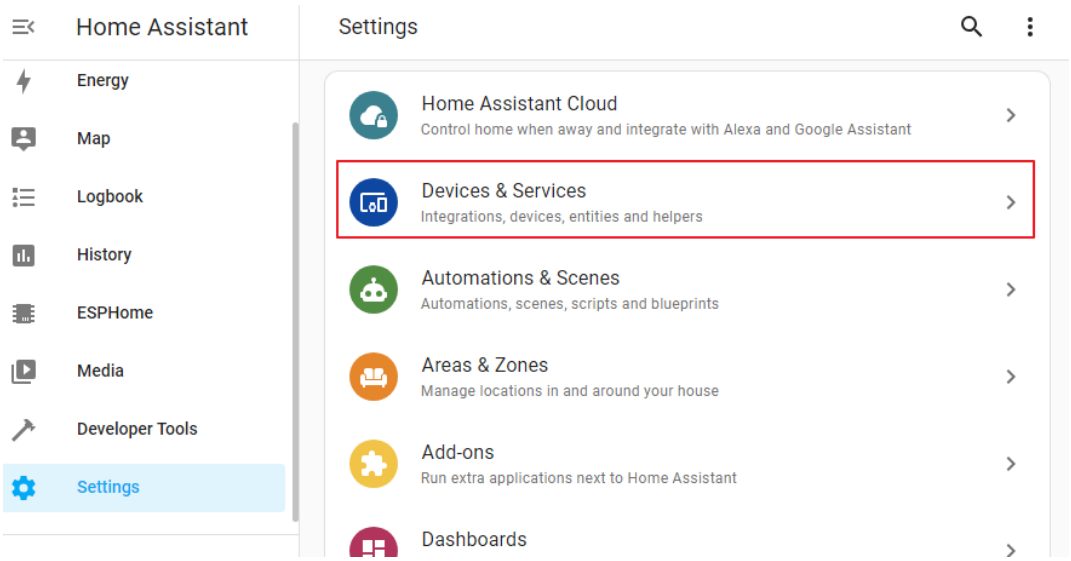
Bemerkung: An diesem Punkt sollte der Knoten als **ONLINE** angezeigt werden. Wenn nicht, stellen Sie sicher, dass Ihr ESP32 im selben Netzwerksegment ist oder versuchen Sie, das Gerät neu zu starten.



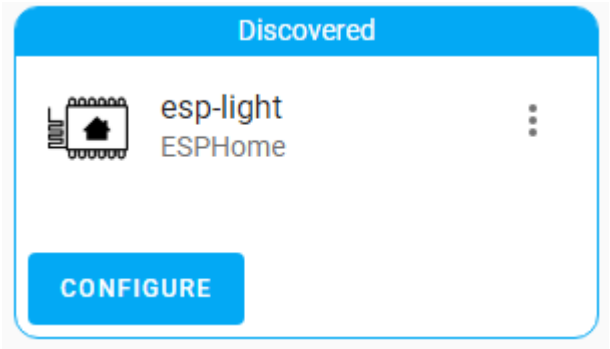
2. Konfiguration in Home Assistant

Nach der Integration mit Esphome müssen Sie die Kamera in Home Assistant noch konfigurieren.

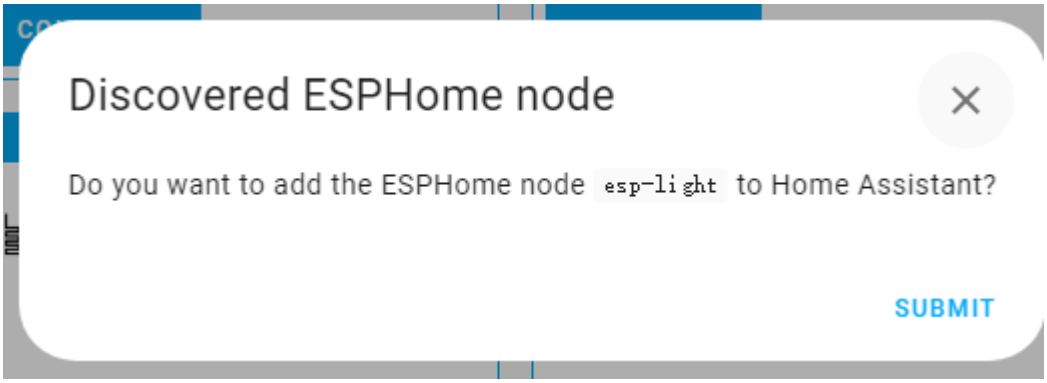
1. Gehen Sie zu **Settings** > **Devices & Services**.



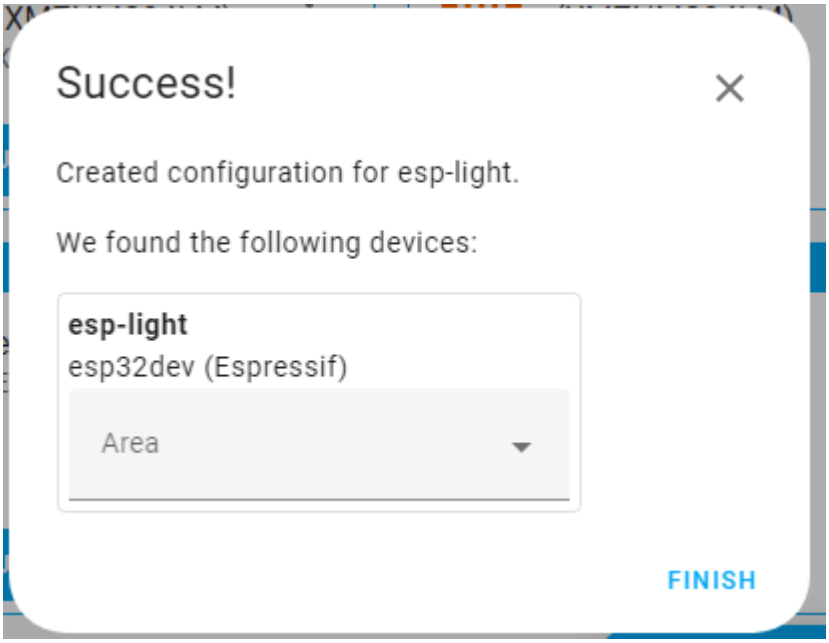
2. Jetzt sollten Sie den Reiter esphome sehen. Klicken Sie auf **CONFIGURE**.



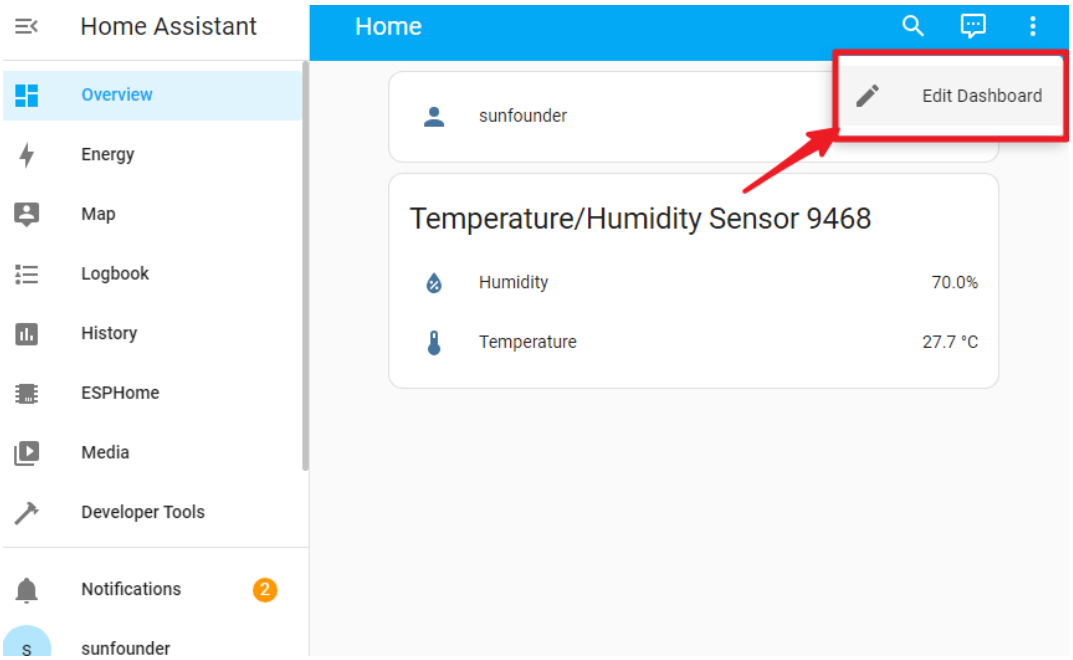
3. Klicken Sie auf **SUBMIT**.



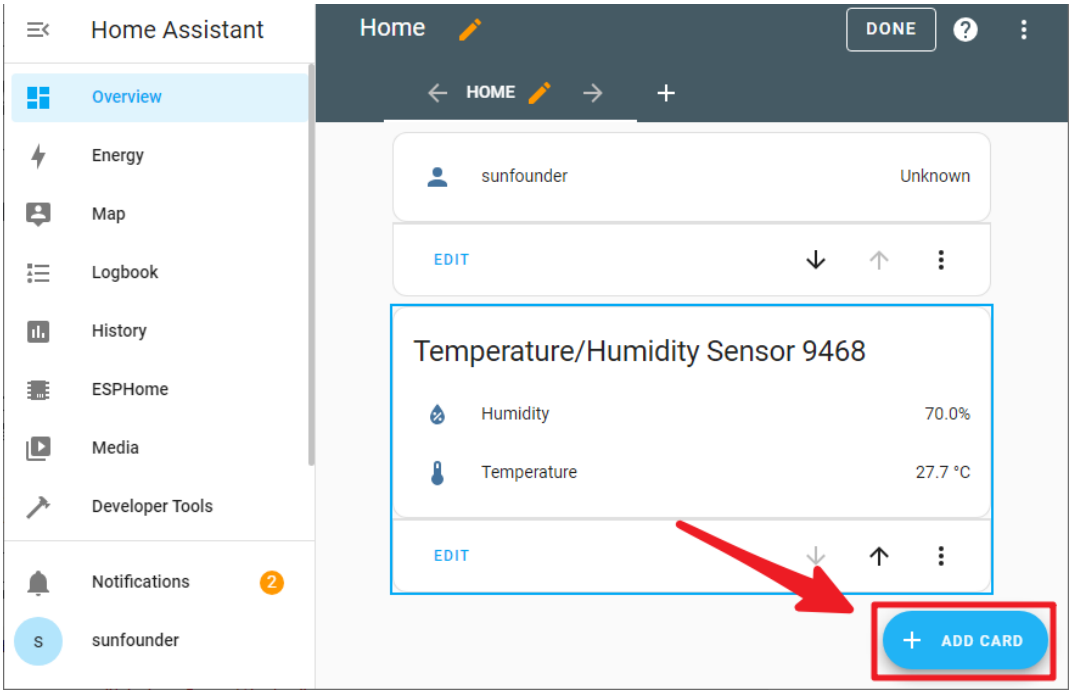
4. Warten Sie auf die **Success**-Nachricht.



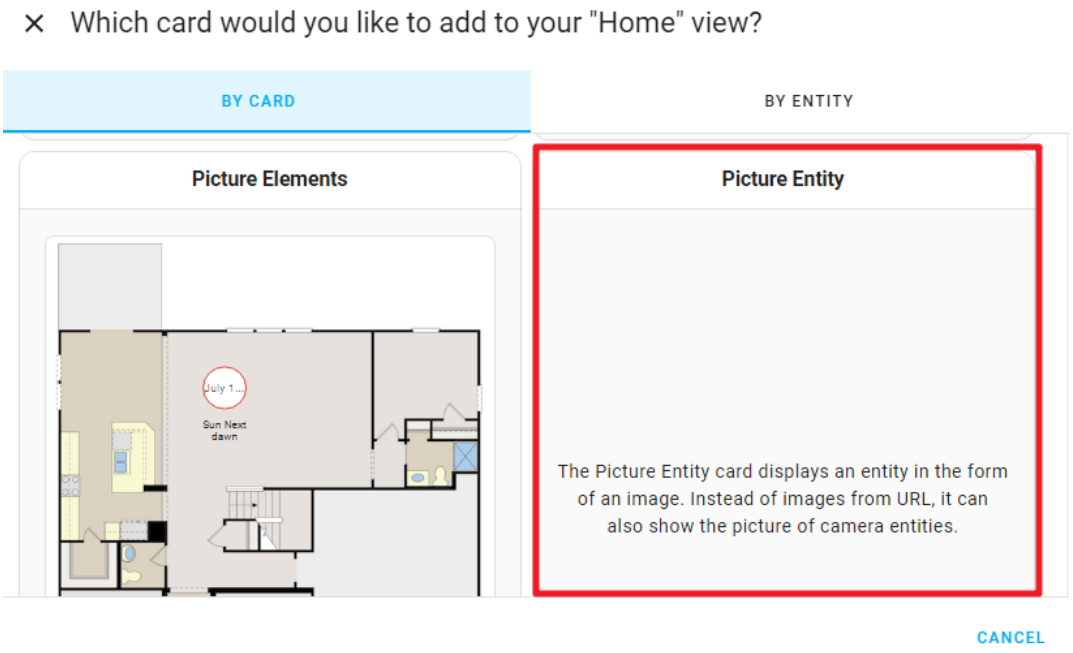
5. Klicken Sie im **Overview** oben rechts auf das Menü und wählen Sie **Edit Dashboard**.



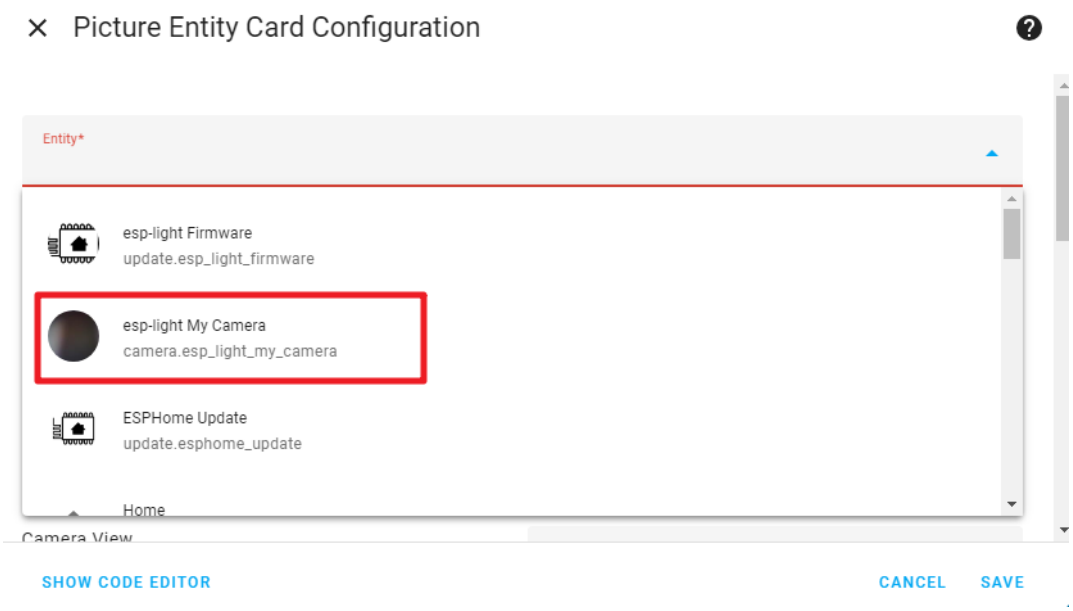
6. Klicken Sie auf **ADD CARD**.



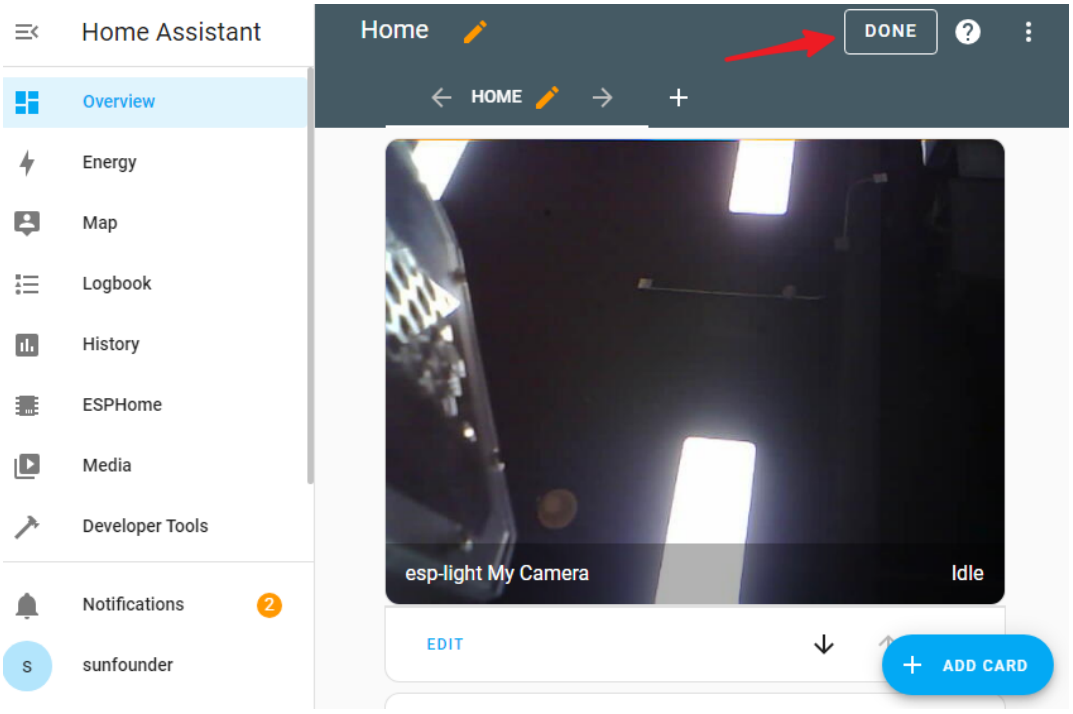
7. Wählen Sie **Picture entity**.



8. Wählen Sie im Feld Entität das ESP32 aus, das Sie gerade hinzugefügt haben. Dann **save**.



9. Klicken Sie zuletzt auf **DONE**, um die **EDIT**-Schnittstelle zu verlassen.



Nun können Sie Ihren Kamera-Feed in Home Assistant ansehen.

2.52 8.9 Blynk-basiertes Einbruchmeldesystem

Dieses Projekt demonstriert ein einfaches Einbruchmeldesystem für Zuhause, das einen PIR-Bewegungssensor (HC-SR501) nutzt. Wenn das System über die Blynk-App auf den Modus „Abwesend“ eingestellt ist, überwacht der PIR-Sensor Bewegungen. Jede erkannte Bewegung löst eine Benachrichtigung in der Blynk-App aus, die den Benutzer über einen möglichen Einbruch informiert.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

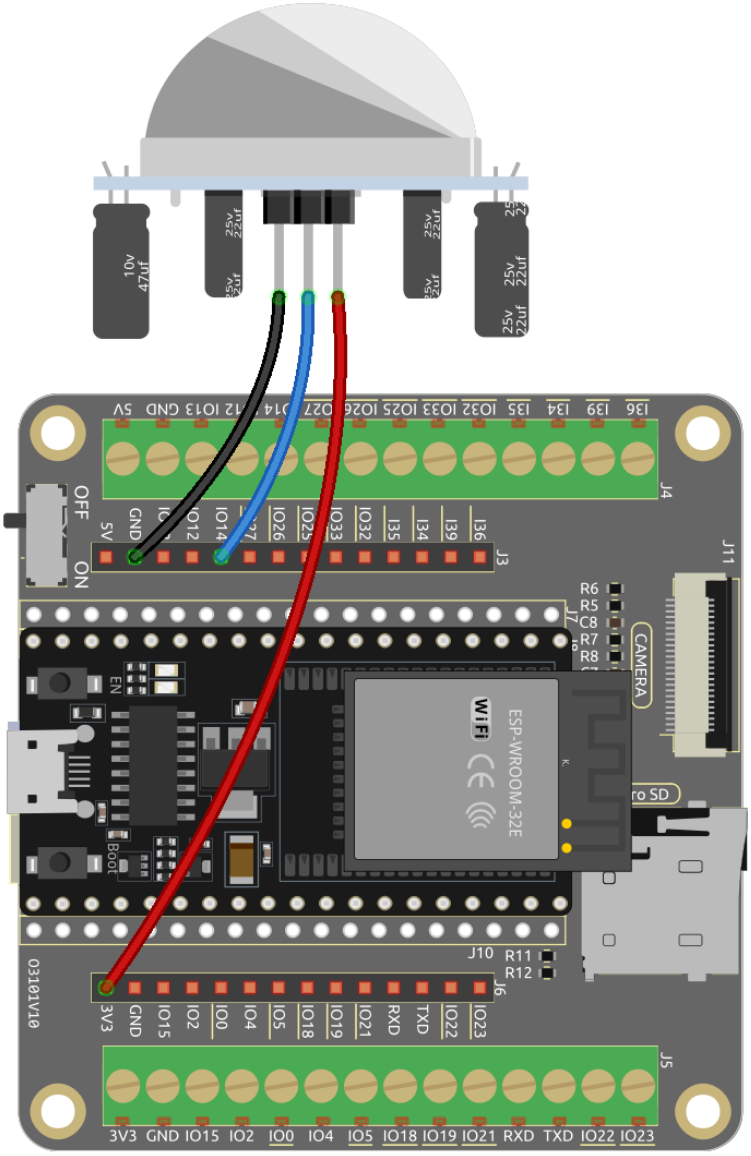
Es ist definitiv praktisch, ein ganzes Kit zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
PIR-Bewegungssensormodul	

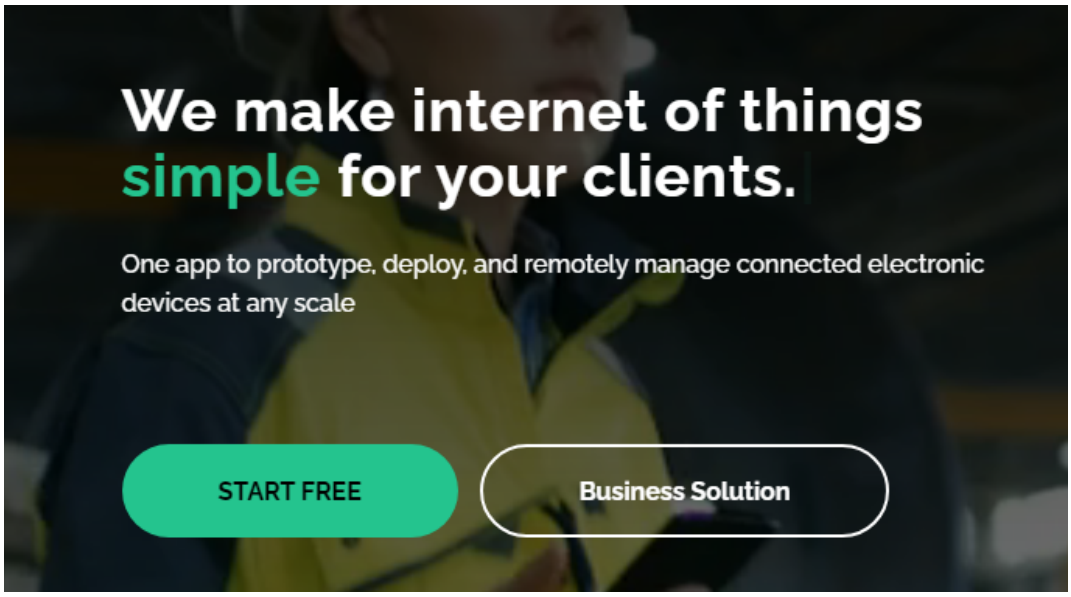
2.52.1 1. Schaltungs-Zusammenbau



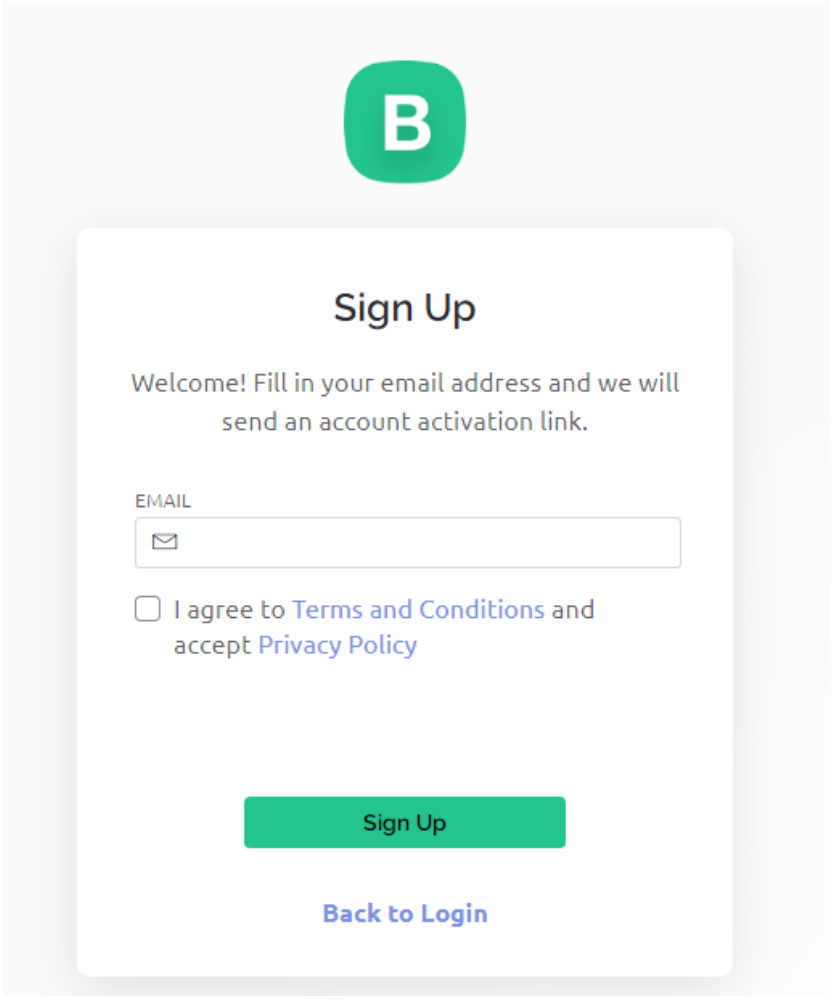
2.52.2 2. Blynk-Konfiguration

2.1 Blynk-Initialisierung

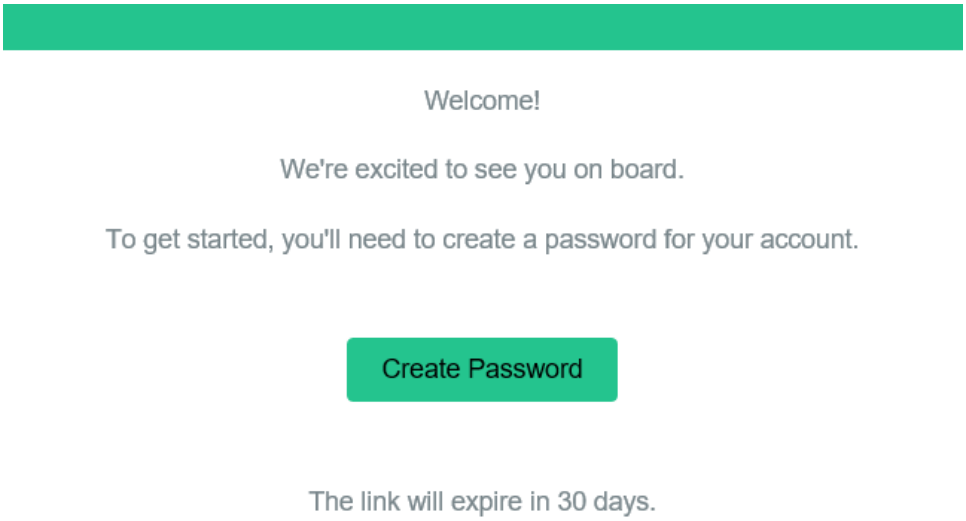
1. Navigieren Sie zur und wählen Sie **START FREE**.



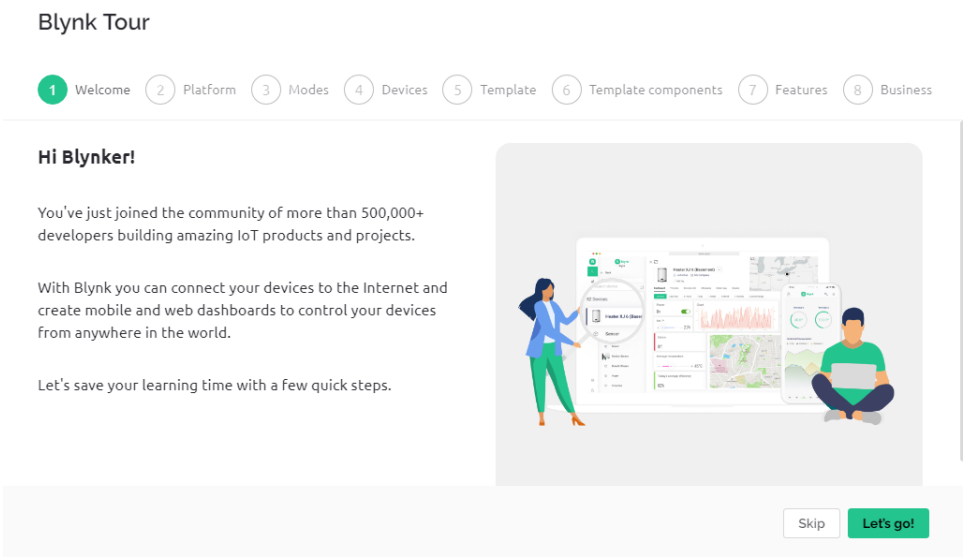
2. Geben Sie Ihre E-Mail-Adresse ein, um den Registrierungsprozess zu starten.



3. Bestätigen Sie Ihre Registrierung per E-Mail.

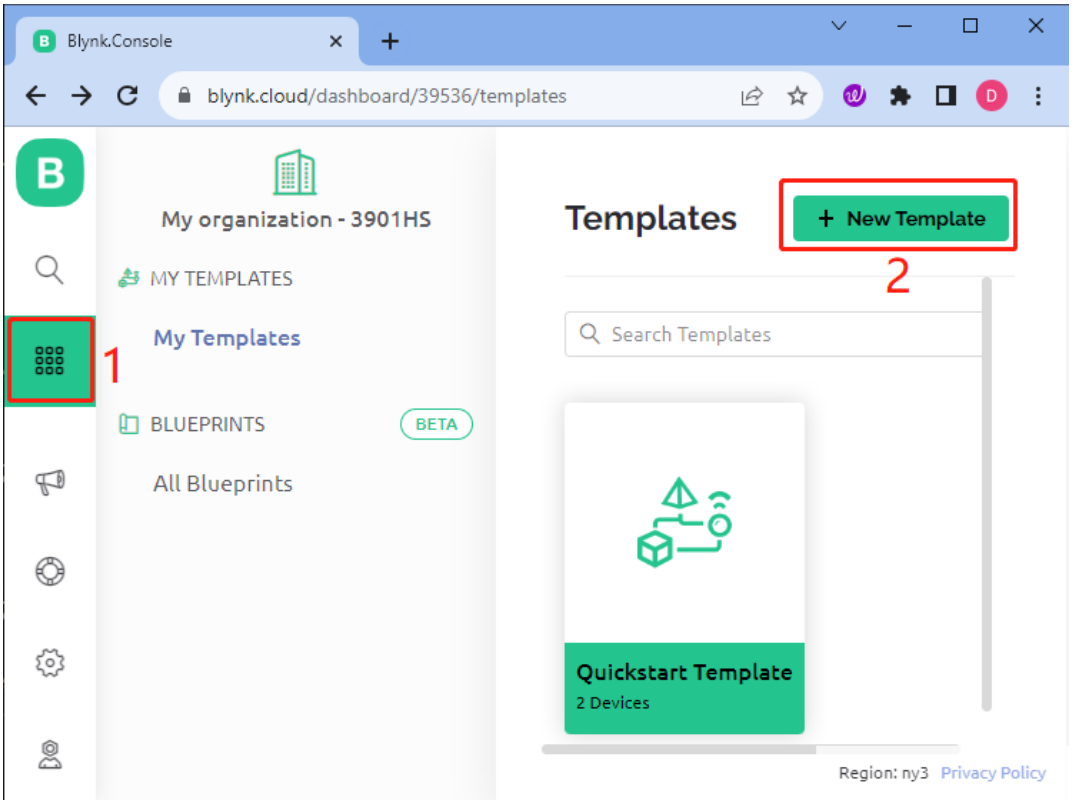


4. Nach der Bestätigung erscheint die **Blynk Tour**. Es wird empfohlen, „Überspringen“ zu wählen. Erscheint auch **Quick Start**, überlegen Sie, auch dies zu überspringen.

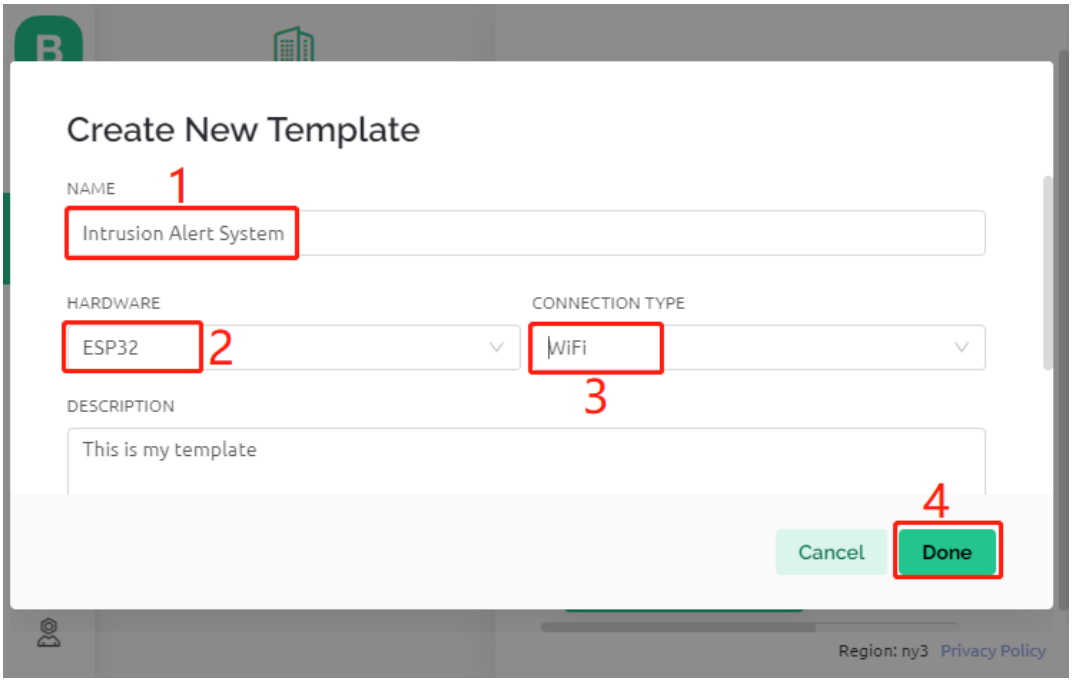


2.2 Template-Erstellung

1. Erstellen Sie zunächst ein Template in Blynk. Folgen Sie den nachfolgenden Anweisungen, um das Template **Intrusion Alert System** zu erstellen.



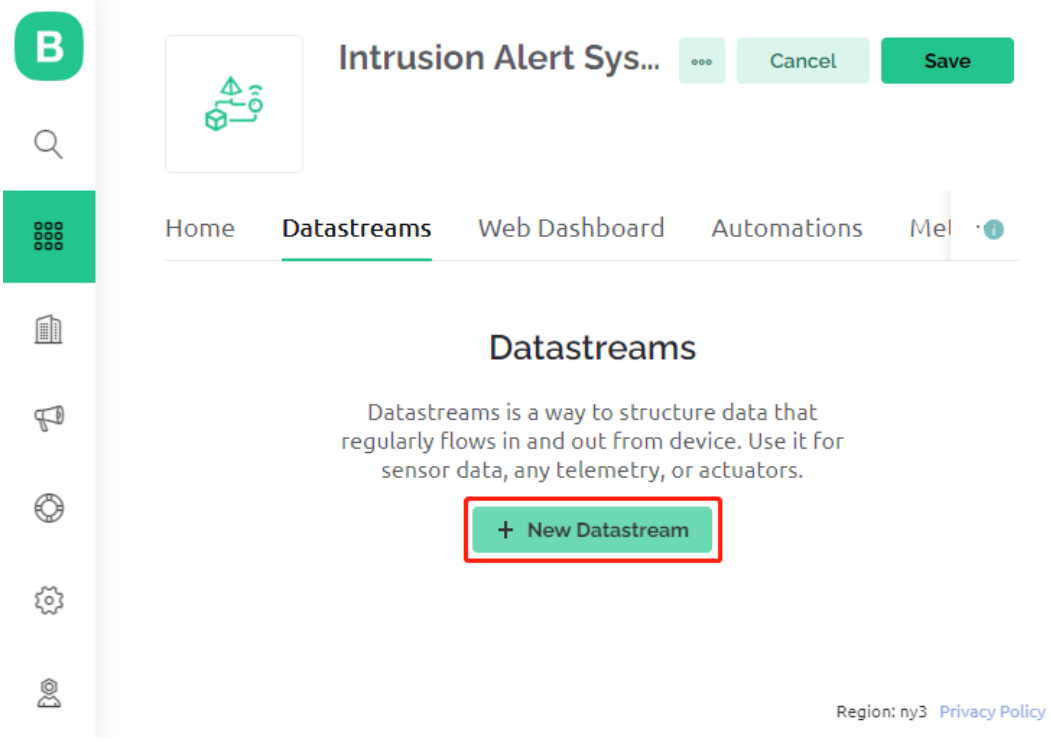
2. Geben Sie dem Template einen Namen, wählen Sie als Hardware **ESP32** und als **Connection Type WiFi**, dann wählen Sie **Done**.



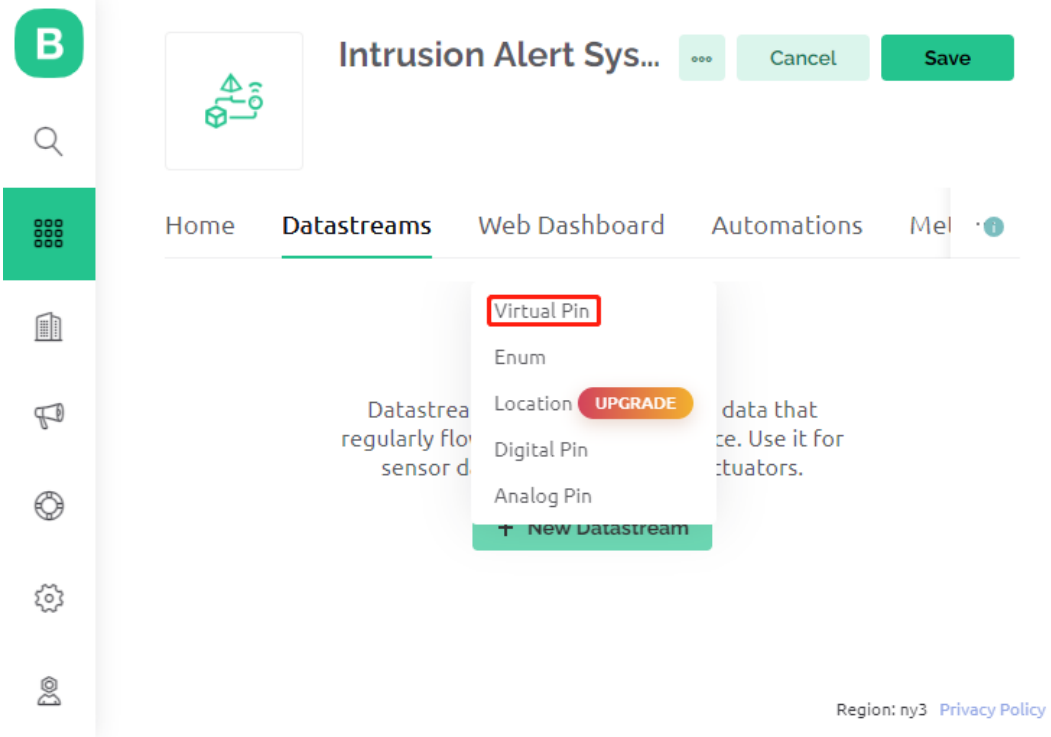
2.3 Generierung von Datenströmen

Öffnen Sie das gerade erstellte Template, um zwei Datenströme zu erstellen.

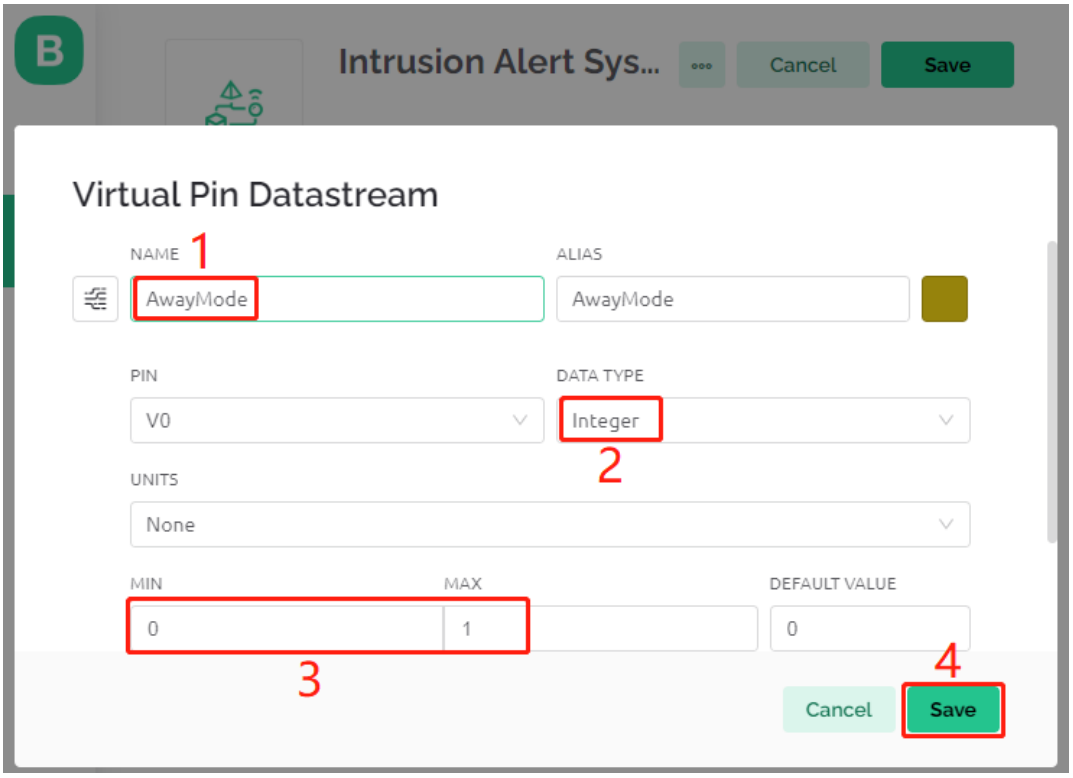
1. Klicken Sie auf **New Datastream**.



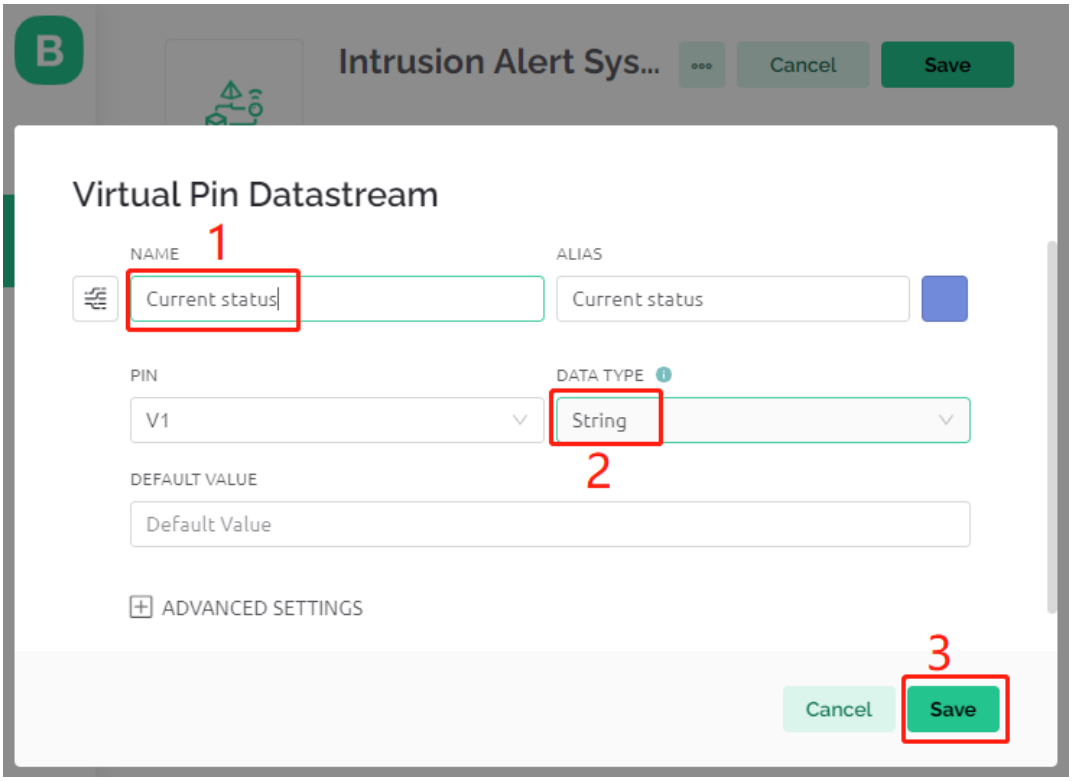
2. Wählen Sie im Popup **Virtual Pin** aus.



3. Benennen Sie den **Virtual Pin V0** als **AwayMode**. Setzen Sie den **DATA TYPE** auf **Integer** mit den **MIN**- und **MAX**-Werten **0** und **1**.



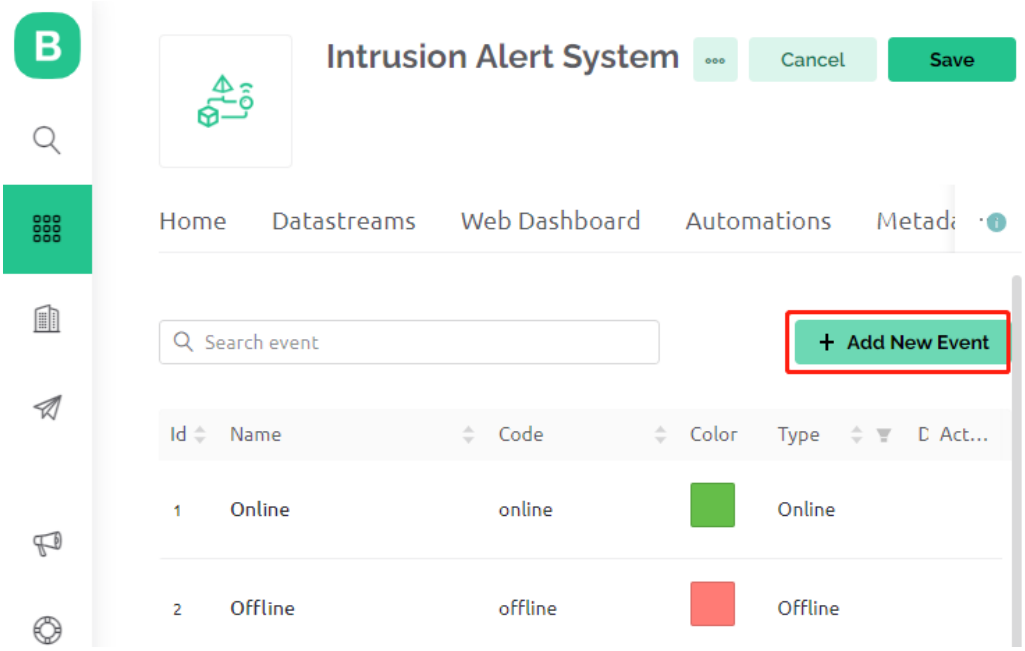
4. Erstellen Sie ähnlich einen weiteren **Virtual Pin**-Datenstrom. Benennen Sie ihn **Current Status** und setzen Sie den **DATA TYPE** auf **String**.



2.4 Einrichten eines Ereignisses

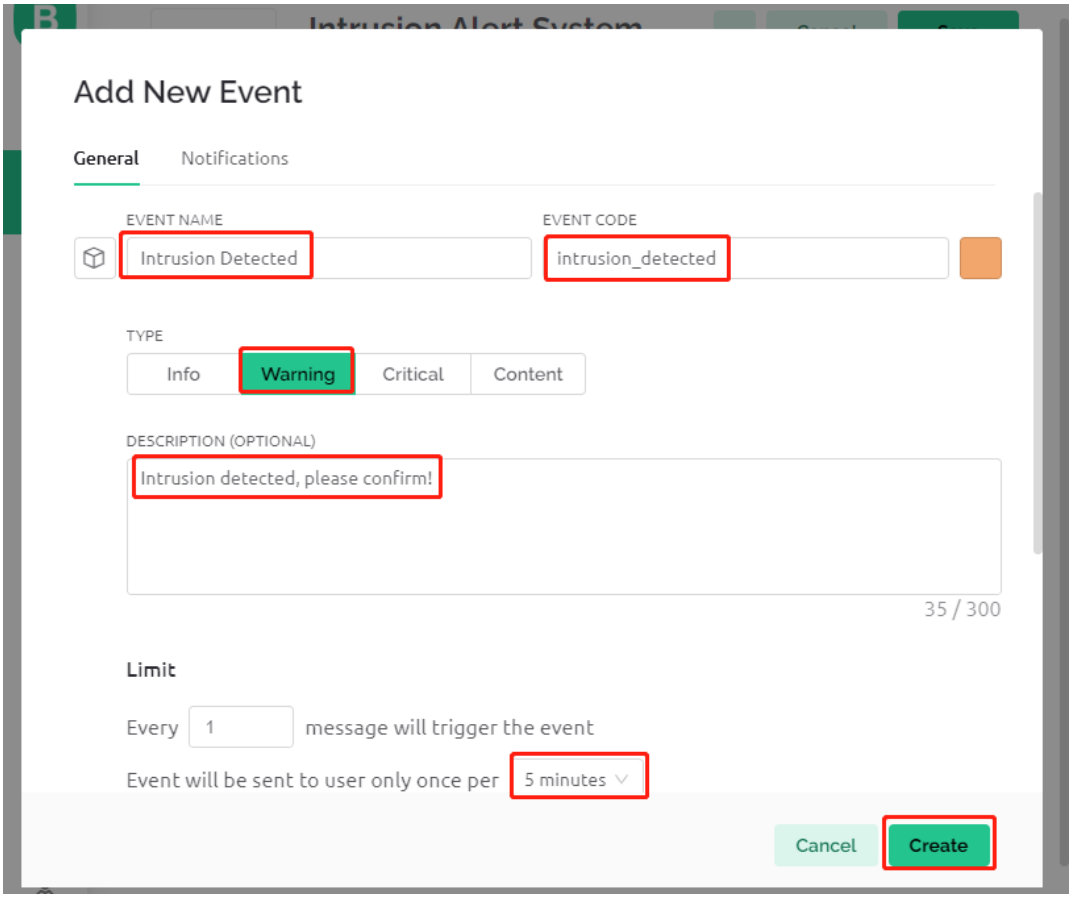
Als Nächstes richten wir ein Ereignis ein, das eine E-Mail-Benachrichtigung sendet, wenn ein Einbruch erkannt wird.

1. Klicken Sie auf **Add New Event**.

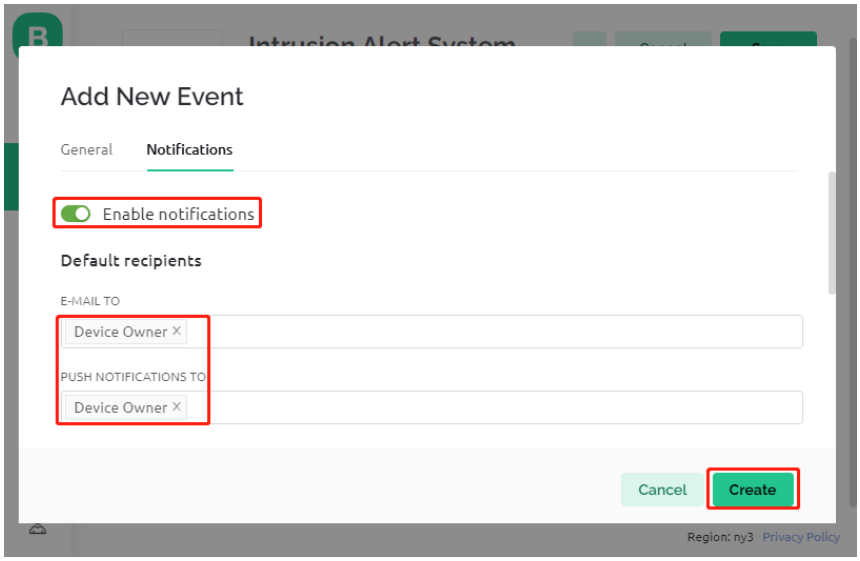


2. Definieren Sie den Namen des Ereignisses und dessen spezifischen Code. Wählen Sie für **TYPE Warning** und schreiben Sie eine kurze Beschreibung für die E-Mail, die gesendet werden soll, wenn das Ereignis eintritt. Sie können auch einstellen, wie oft Sie benachrichtigt werden.

Bemerkung: Stellen Sie sicher, dass der **EVENT CODE** als `intrusion_detected` festgelegt ist. Dies ist im Code vordefiniert, daher müssen Änderungen auch im Code vorgenommen werden.



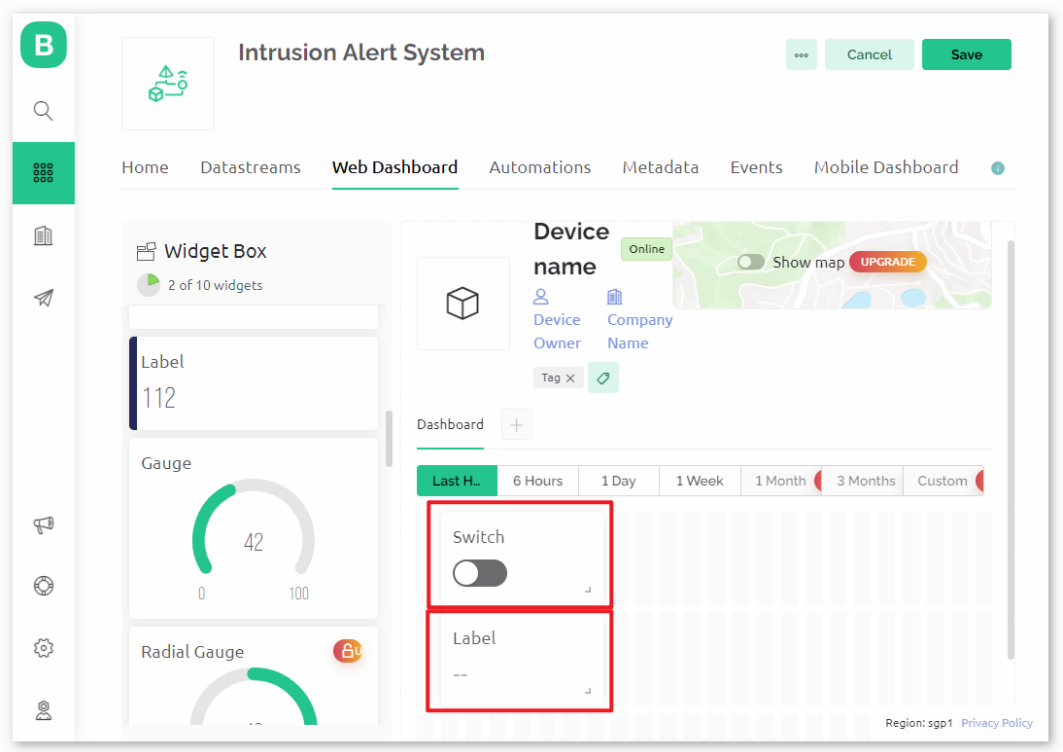
3. Gehen Sie zum Abschnitt **Notifications**, um Benachrichtigungen zu aktivieren und E-Mail-Details einzurichten.



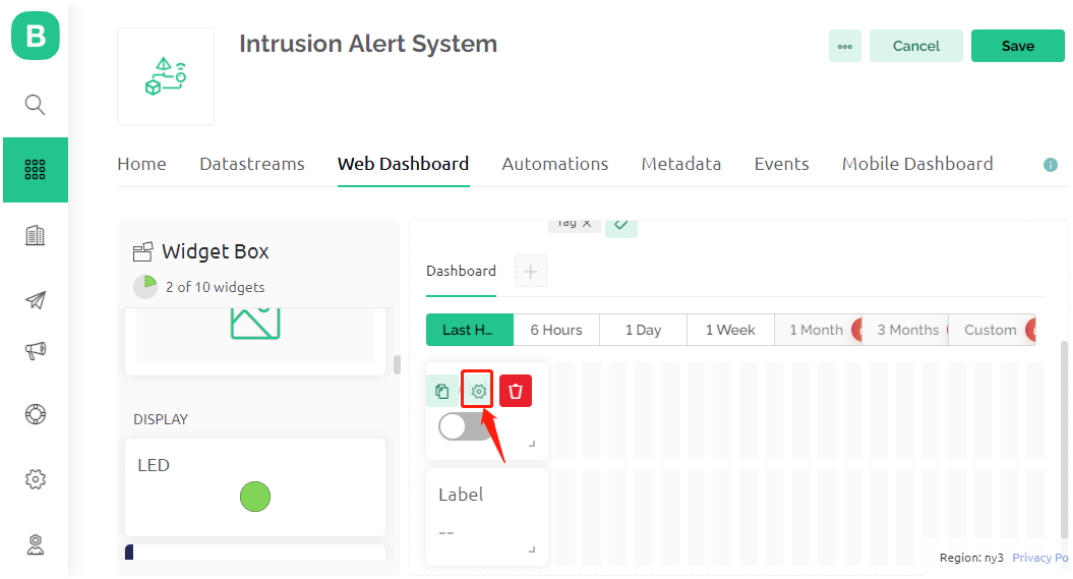
2.5 Feinabstimmung des Web-Dashboards

Es ist wichtig, dass das **Web Dashboard** perfekt mit dem Einbruchmeldesystem interagiert.

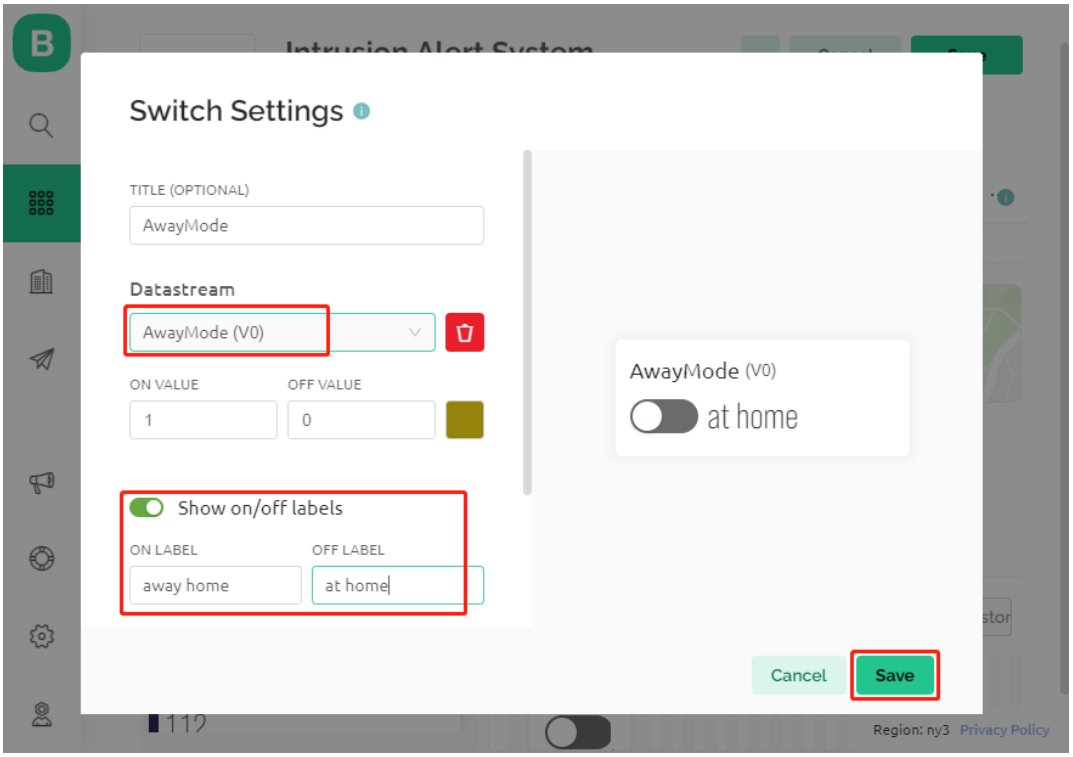
1. Ziehen Sie einfach sowohl das **Switch widget** als auch das **Label widget** auf das **Web Dashboard**.



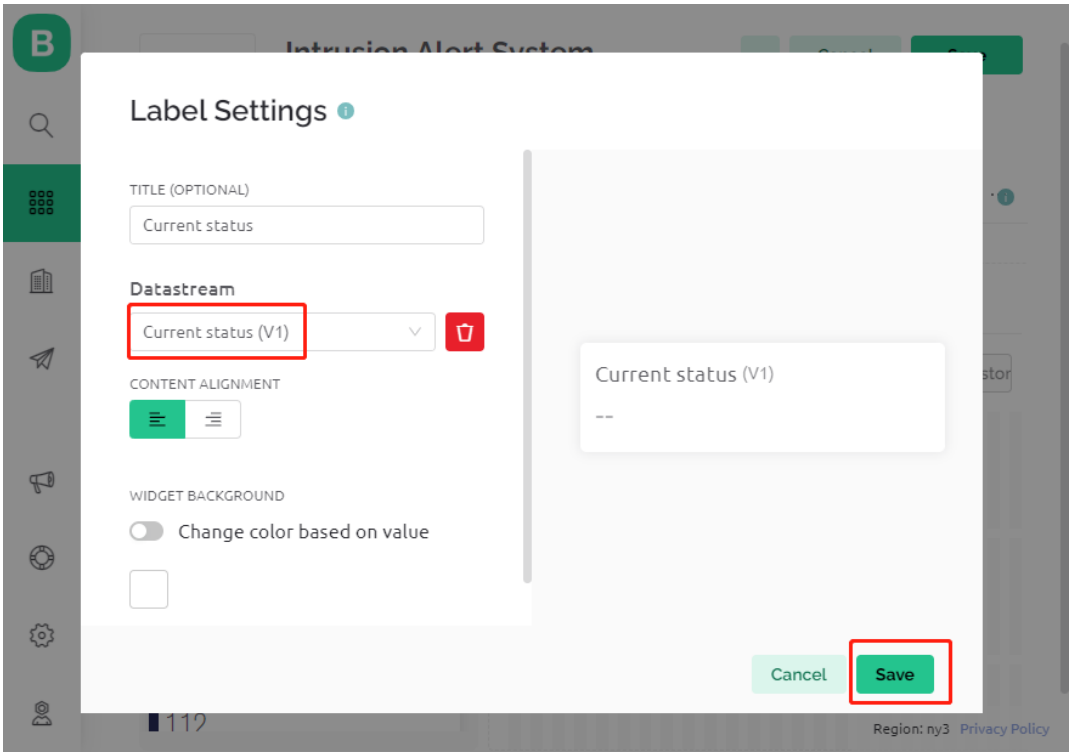
2. Wenn Sie mit der Maus über ein Widget fahren, erscheinen drei Symbole. Verwenden Sie das Einstellungssymbol, um die Eigenschaften des Widgets anzupassen.



3. In den Einstellungen des **Switch widget** wählen Sie **Datastream** als **AwayMode(V0)**. Setzen Sie **ONLABEL** und **OFFLABEL** auf **„away“** bzw. **„home“**.

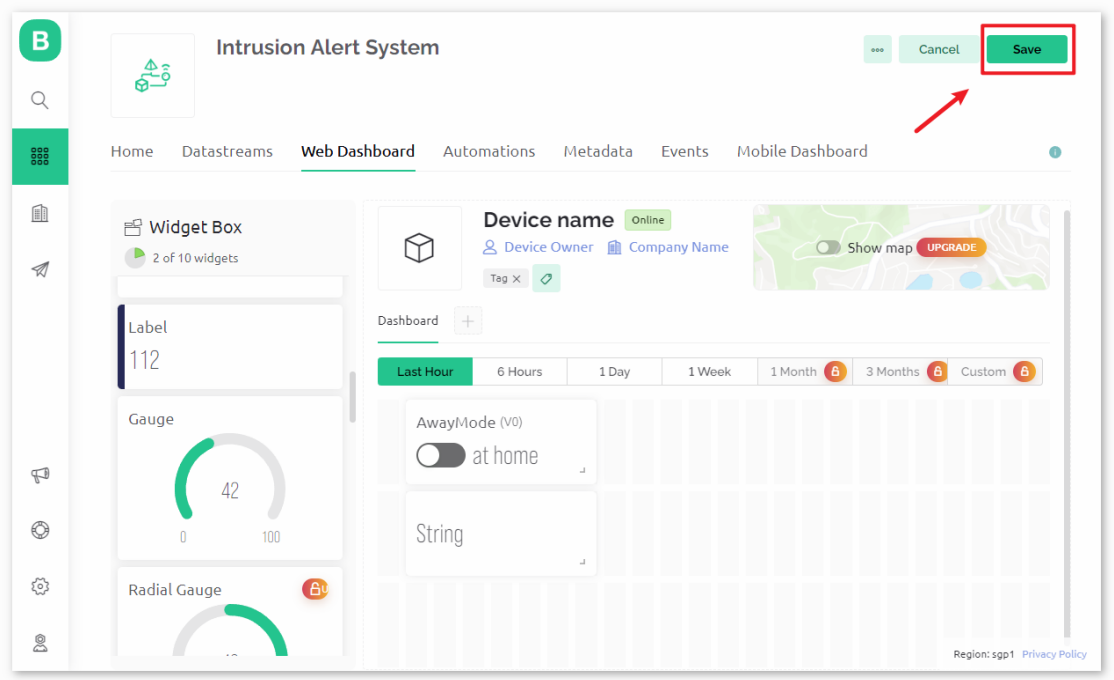


4. In den Einstellungen des **Label widget** wählen Sie **Datastream** als **Current Status(V1)**.



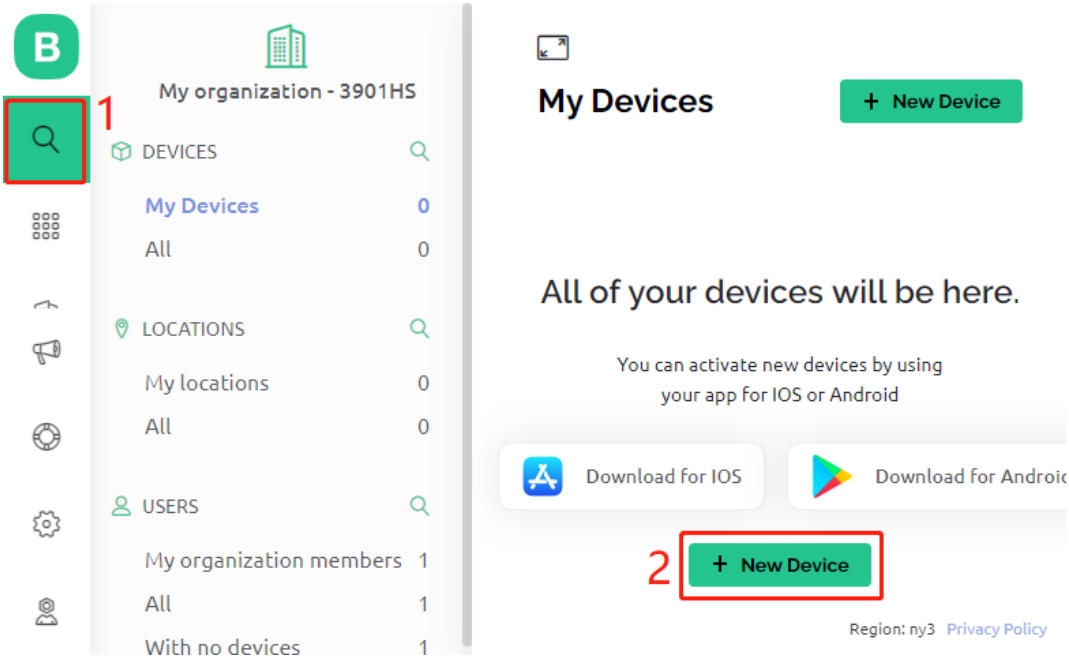
2.6 Speichern des Templates

Vergessen Sie zum Schluss nicht, Ihr Template zu speichern.

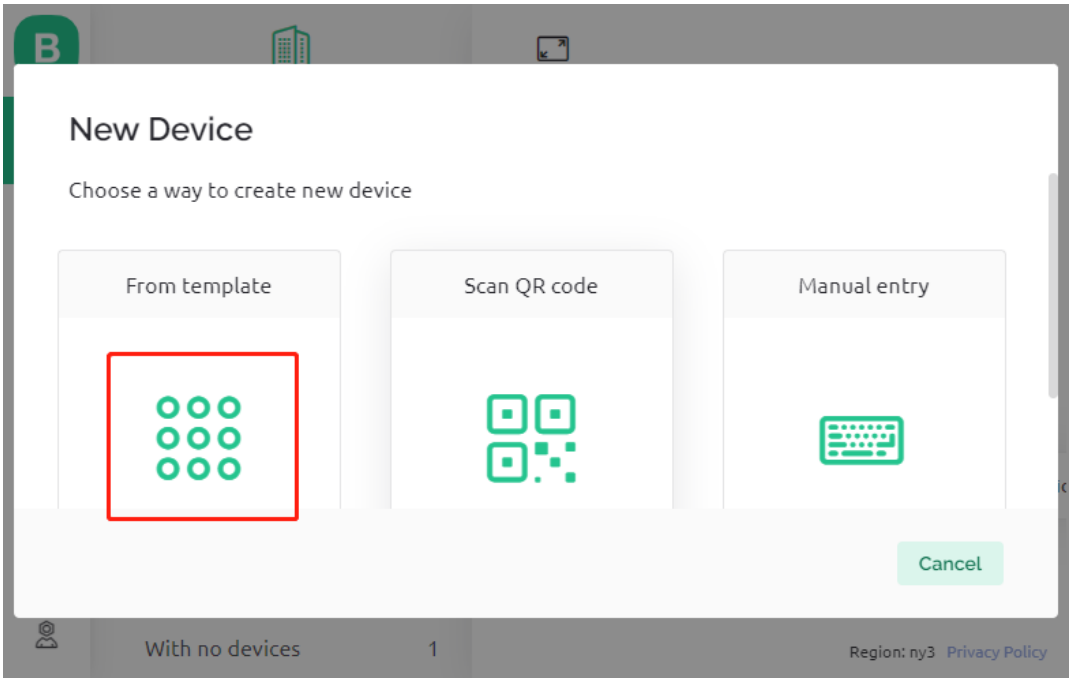


2.7 Erstellen eines Geräts

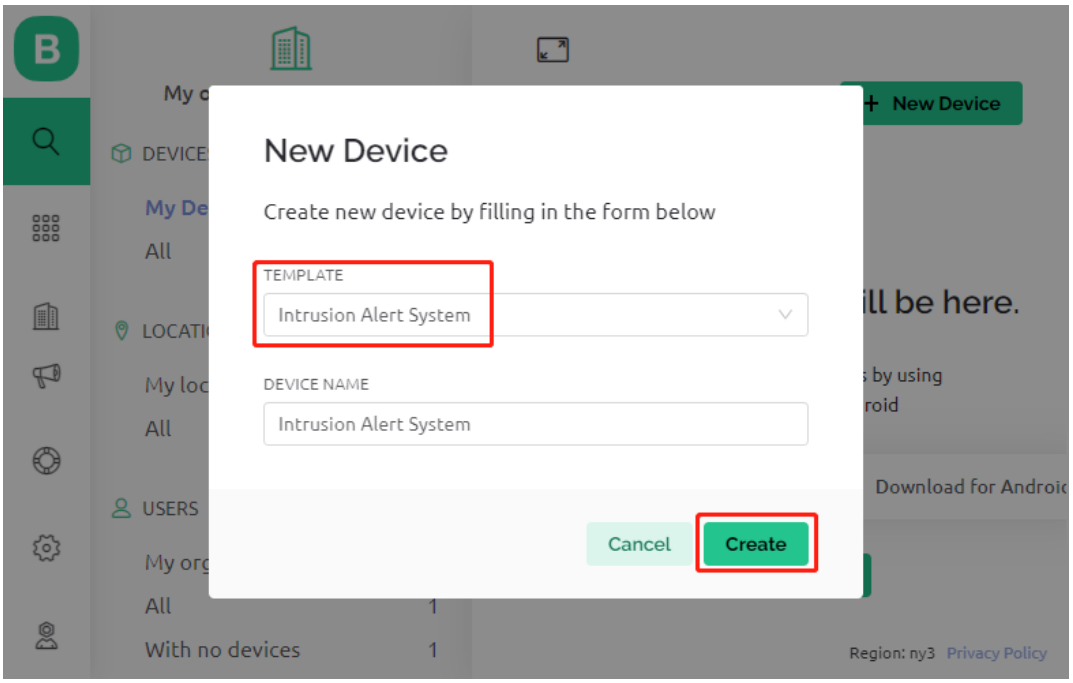
1. Jetzt ist es an der Zeit, ein neues Gerät zu erstellen.



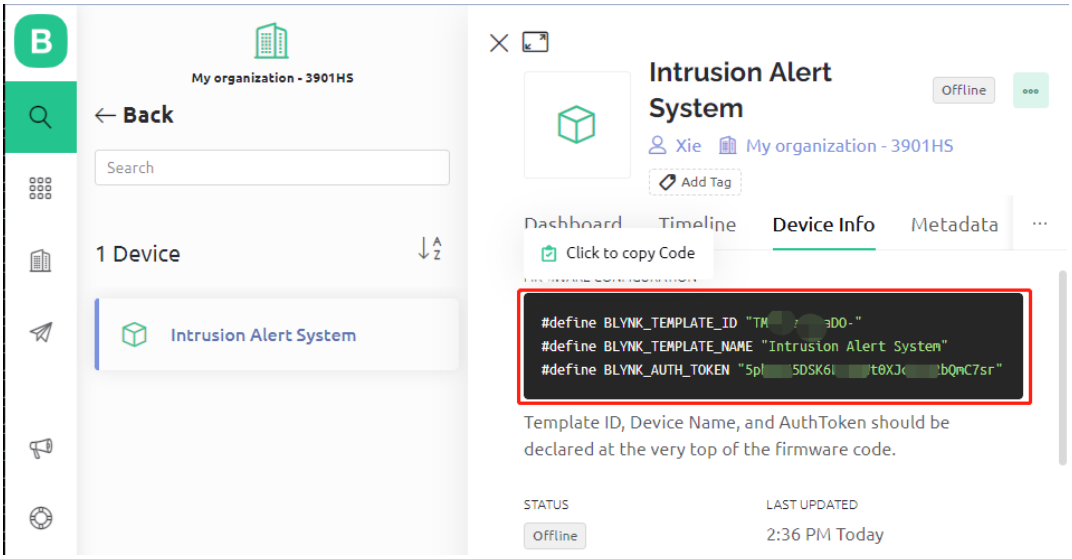
2. Klicken Sie auf **From template**, um mit einer neuen Einrichtung zu beginnen.



3. Wählen Sie dann das Template **Intrusion Alert System** und klicken Sie auf **Create**.

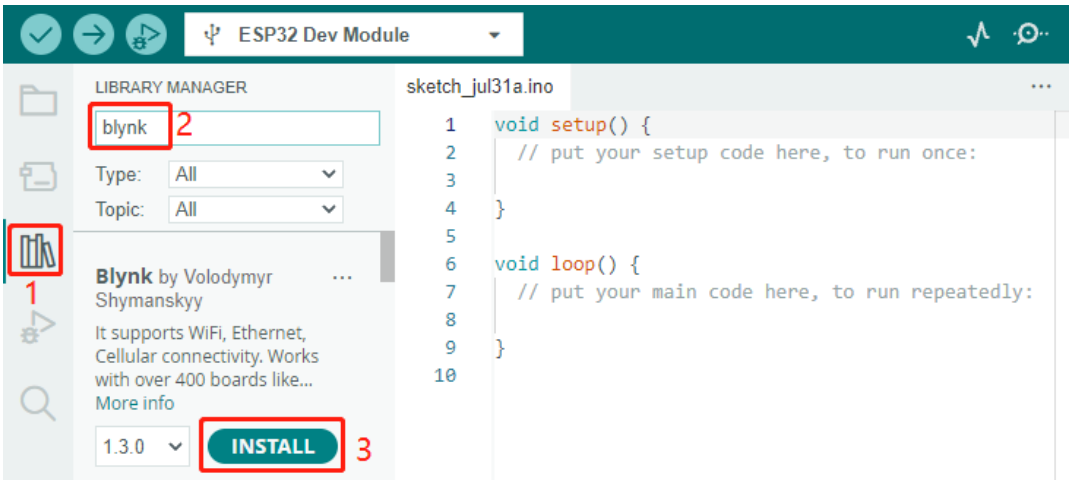


4. Hier sehen Sie die `Template ID`, den `Device Name` und den `AuthToken`. Sie müssen diese in Ihren Code kopieren, damit der ESP32 mit Blynk arbeiten kann.



2.52.3 3. Codeausführung

1. Bevor Sie den Code ausführen, stellen Sie sicher, dass Sie die `Blynk`-Bibliothek über den **Library Manager** in der Arduino IDE installiert haben.



2. Öffnen Sie die Datei `iot_9_intrusion_alert_system.ino`, die sich im Verzeichnis `esp32-starter-kit-main\c\codes\iot_9_intrusion_alert_system` befindet. Sie können auch deren Inhalt in die Arduino IDE kopieren.

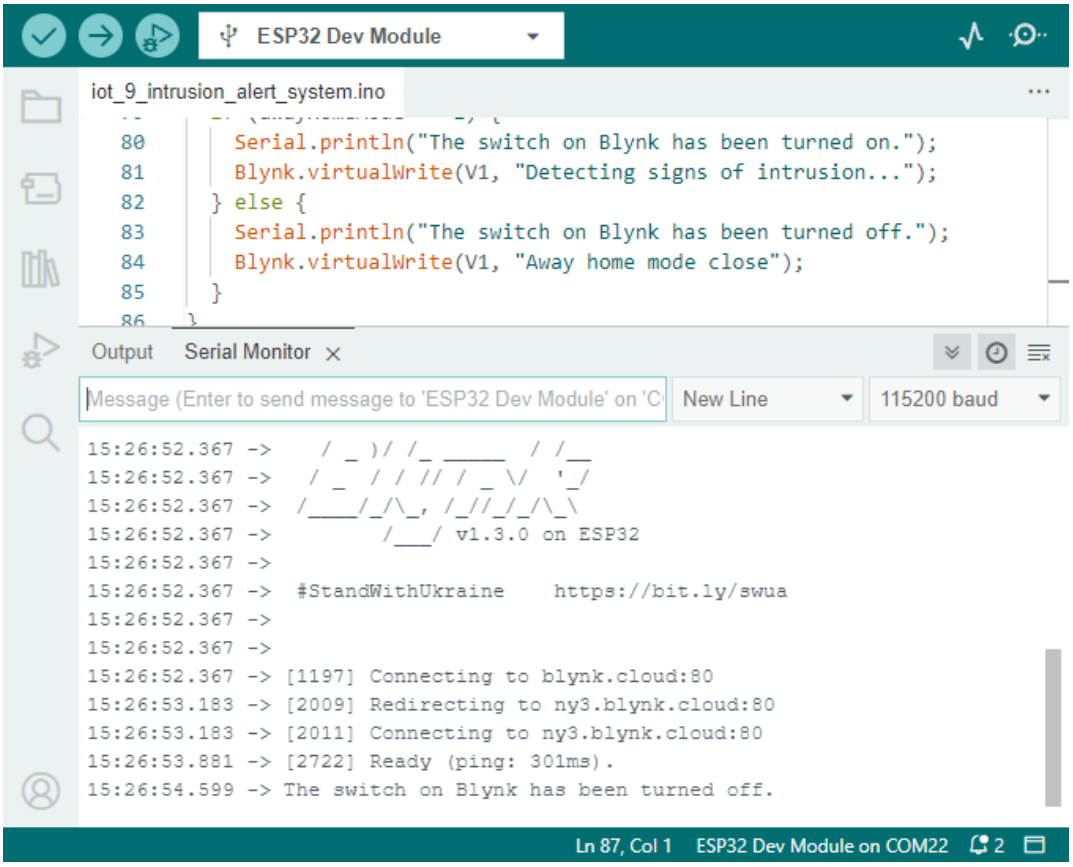
3. Ersetzen Sie die Platzhalter für `BLYNK_TEMPLATE_ID`, `BLYNK_TEMPLATE_NAME` und `BLYNK_AUTH_TOKEN` mit Ihren eigenen einzigartigen IDs.

```
#define BLYNK_TEMPLATE_ID "TMPxxxxxxx"
#define BLYNK_TEMPLATE_NAME "Intrusion Alert System"
#define BLYNK_AUTH_TOKEN "xxxxxxxxxxxx"
```

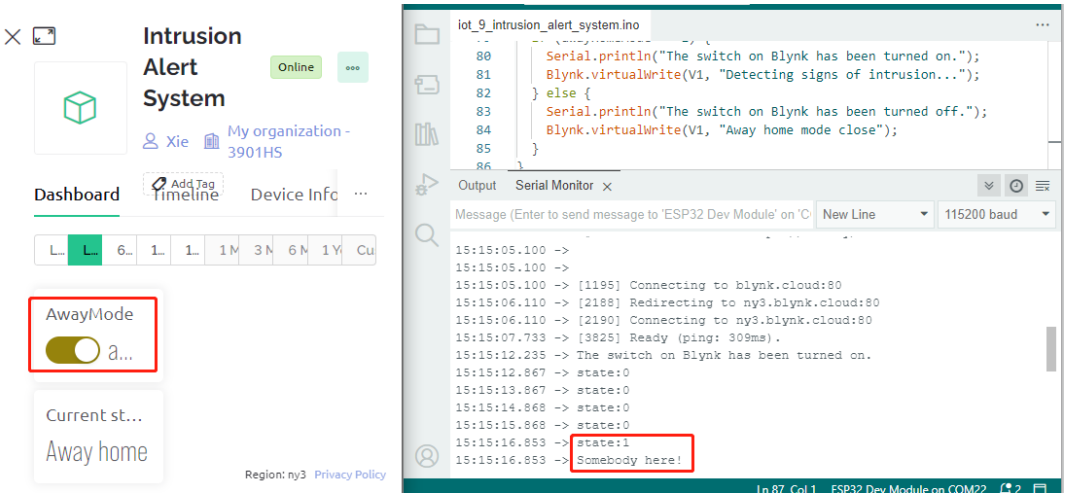
4. Geben Sie auch die `ssid` und das `password` Ihres WLAN-Netzwerks ein.

```
char ssid[] = "your_ssid";
char pass[] = "your_password";
```

5. Wählen Sie das richtige Board (**ESP32 Dev Module**) und den Port, und klicken Sie dann auf den **Upload**-Button.
6. Öffnen Sie den Seriellen Monitor (Baudrate auf 115200 einstellen) und warten Sie auf eine erfolgreiche Verbindungsmeldung.



7. Nach einer erfolgreichen Verbindung startet das Aktivieren des Schalters in Blynk die Überwachung des PIR-Moduls. Wenn eine Bewegung erkannt wird (Zustand 1), wird „Jemand ist hier!“ angezeigt und eine Warnung an Ihre E-Mail gesendet.



2.52.4 4. Code-Erklärung

1. **Konfiguration & Bibliotheken**

   Hier richten Sie die Blynk-Konstanten und Zugangsdaten ein. Sie schließen auch die notwendigen Bibliotheken für den ESP32 und Blynk ein.

   ```
   /* Comment this out to disable prints and save space */
   #define BLYNK_PRINT Serial

   #define BLYNK_TEMPLATE_ID "xxxxxxxxxx"
   #define BLYNK_TEMPLATE_NAME "Intrusion Alert System"
   #define BLYNK_AUTH_TOKEN "xxxxxxxxxxxxxxxxxxxxxxxxxx"

   #include <WiFi.h>
   #include <WiFiClient.h>
   #include <BlynkSimpleEsp32.h>
   ```

2. **WiFi-Einrichtung**

   Geben Sie Ihre WLAN-Zugangsdaten ein.

   ```
   char ssid[] = "your_ssid";
   char pass[] = "your_password";
   ```

3. **PIR-Sensor-Konfiguration**

   Legen Sie den Pin fest, an dem der PIR-Sensor angeschlossen ist, und initialisieren Sie die Zustandsvariablen.

   ```
   const int sensorPin = 14;
   int state = 0;
   int awayHomeMode = 0;
   BlynkTimer timer;
   ```

4. **setup() Funktion**

   Diese Funktion initialisiert den PIR-Sensor als Eingang, richtet die serielle Kommunikation ein, verbindet sich mit WLAN und konfiguriert Blynk.

   - Wir verwenden `timer.setInterval(1000L, myTimerEvent)` um das Timer-Intervall in `setup()` zu setzen. Hier legen wir fest, dass die Funktion `myTimerEvent()` alle **1000ms** ausgeführt wird. Sie können den ersten Parameter von `timer.setInterval(1000L, myTimerEvent)` ändern, um das Intervall zwischen den Ausführungen von `myTimerEvent` zu variieren.

   ```
   void setup() {

       pinMode(sensorPin, INPUT);  // Set PIR sensor pin as input
       Serial.begin(115200);           // Start serial communication at 115200 baud␣
   ↪rate for debugging

       // Configure Blynk and connect to WiFi
       Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

       timer.setInterval(1000L, myTimerEvent);  // Setup a function to be called every␣
   ↪second
   }
   ```

5. **loop() Funktion**

   Die loop-Funktion führt kontinuierlich Blynk und die Blynk-Timer-Funktionen aus.

```
void loop() {
   Blynk.run();
   timer.run();
}
```

6. **Blynk-App-Interaktion**

   Diese Funktionen werden aufgerufen, wenn das Gerät sich mit Blynk verbindet und wenn sich der Zustand des virtuellen Pins V0 in der Blynk-App ändert.

   - Jedes Mal, wenn das Gerät sich mit dem Blynk-Server verbindet oder aufgrund schlechter Netzwerkbedingungen erneut verbindet, wird die Funktion `BLYNK_CONNECTED()` aufgerufen. Der Befehl `Blynk.syncVirtual()` fordert einen einzelnen virtuellen Pinwert an. Der angegebene virtuelle Pin führt einen `BLYNK_WRITE()`-Aufruf durch.
   - Immer wenn sich der Wert eines virtuellen Pins auf dem BLYNK-Server ändert, wird `BLYNK_WRITE()` ausgelöst.

```
// This function is called every time the device is connected to the Blynk.Cloud
BLYNK_CONNECTED() {
    Blynk.syncVirtual(V0);
}

// This function is called every time the Virtual Pin 0 state changes
BLYNK_WRITE(V0) {
    awayHomeMode = param.asInt();
    // additional logic
}
```

7. **Datenverarbeitung**

   Jede Sekunde ruft die Funktion `myTimerEvent()` die Funktion `sendData()` auf. Wenn der Abwesenheitsmodus in Blynk aktiviert ist, überprüft sie den PIR-Sensor und sendet eine Benachrichtigung an Blynk, wenn eine Bewegung erkannt wird.

   - Wir verwenden `Blynk.virtualWrite(V1, "Jemand in Ihrem Haus! Bitte überprüfen!");` um den Text eines Labels zu ändern.
   - Verwenden Sie `Blynk.logEvent("intrusion_detected");`, um ein Ereignis in Blynk zu protokollieren.

```
void myTimerEvent() {
   sendData();
}

void sendData() {
   if (awayHomeMode == 1) {
      state = digitalRead(sensorPin);  // Read the state of the PIR sensor

      Serial.print("state:");
      Serial.println(state);

      // If the sensor detects movement, send an alert to the Blynk app
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
      if (state == HIGH) {
        Serial.println("Somebody here!");
        Blynk.virtualWrite(V1, "Somebody in your house! Please check!");
        Blynk.logEvent("intrusion_detected");
      }
   }
}
```

Referenz

-
-
-
-
-
-
-

2.53 8.10 Android-Anwendung - RGB-LED-Steuerung über Arduino und Bluetooth

Ziel dieses Projekts ist die Entwicklung einer Android-Anwendung, die in der Lage ist, den Farbton einer RGB-LED über ein Smartphone mittels Bluetooth-Technologie zu manipulieren.

Diese Android-Anwendung wird mit Hilfe einer kostenlosen webbasierten Plattform namens MIT App Inventor 2 erstellt. Das Projekt bietet eine hervorragende Gelegenheit, Erfahrungen im Umgang mit der Schnittstelle zwischen einem Arduino und einem Smartphone zu sammeln.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

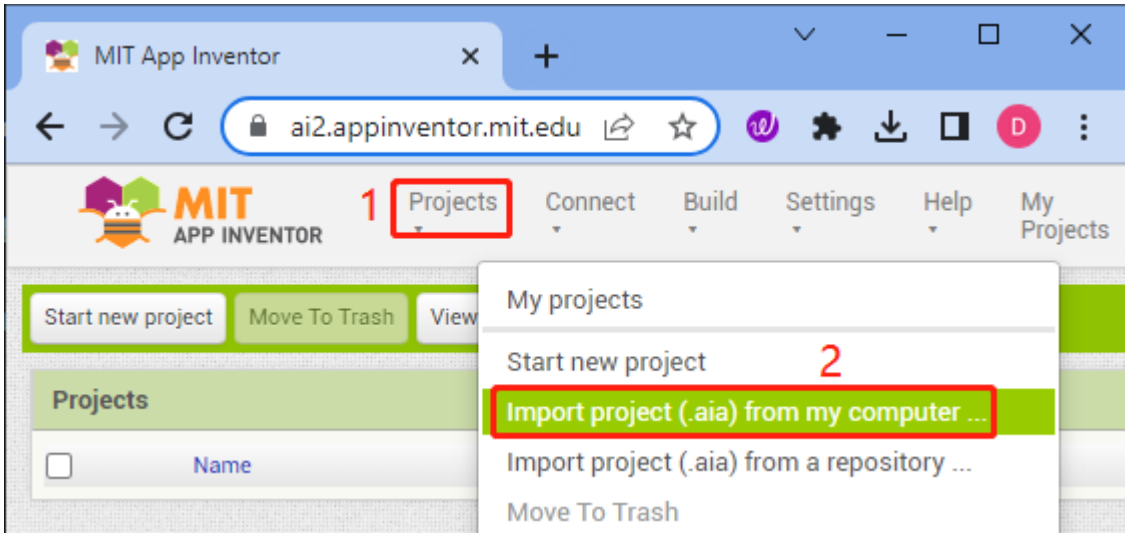
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
RGB LED	

1. Erstellung der Android-Anwendung

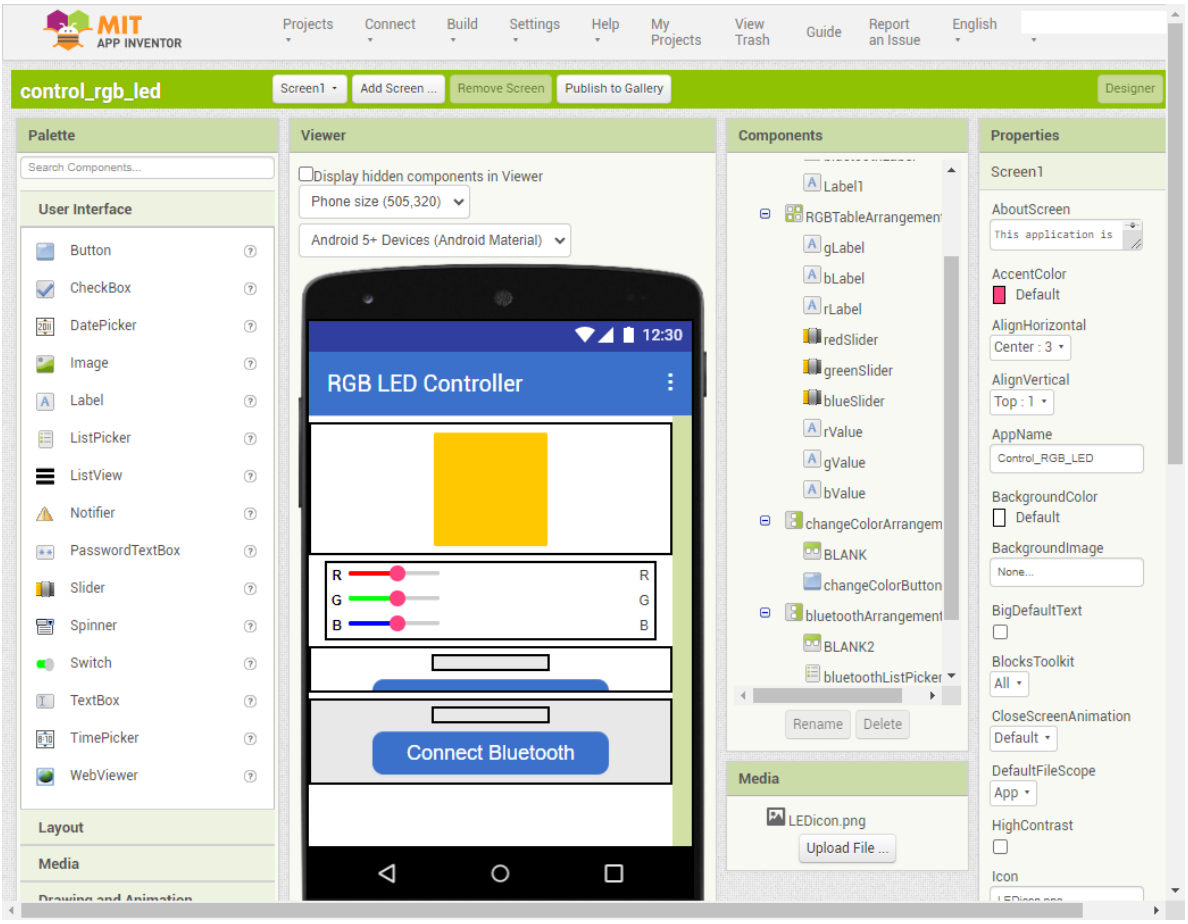
Die Android-Anwendung wird mit Hilfe einer kostenlosen Webanwendung namens erstellt. MIT App Inventor ist ein hervorragender Einstiegspunkt für die Android-Entwicklung aufgrund seiner intuitiven Drag-and-Drop-Funktionen, die die Erstellung einfacher Anwendungen ermöglichen.

Beginnen wir.

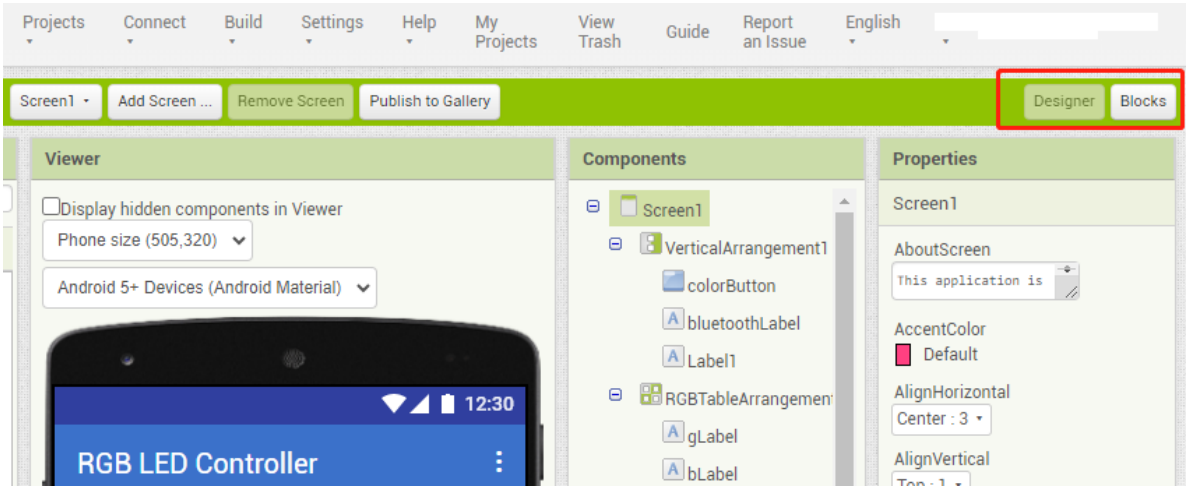
1. Hier ist die Login-Seite: http://ai2.appinventor.mit.edu. Sie benötigen ein Google-Konto, um sich bei MIT App Inventor anzumelden.
2. Nach dem Einloggen navigieren Sie zu **Projects** -> **Import project (.aia) from my computer**. Laden Sie anschließend die Datei `control_rgb_led.aia` hoch, die sich im Pfad `esp32-starter-kit-main\c\codes\iot_10_bluetooth_app_inventor` befindet.



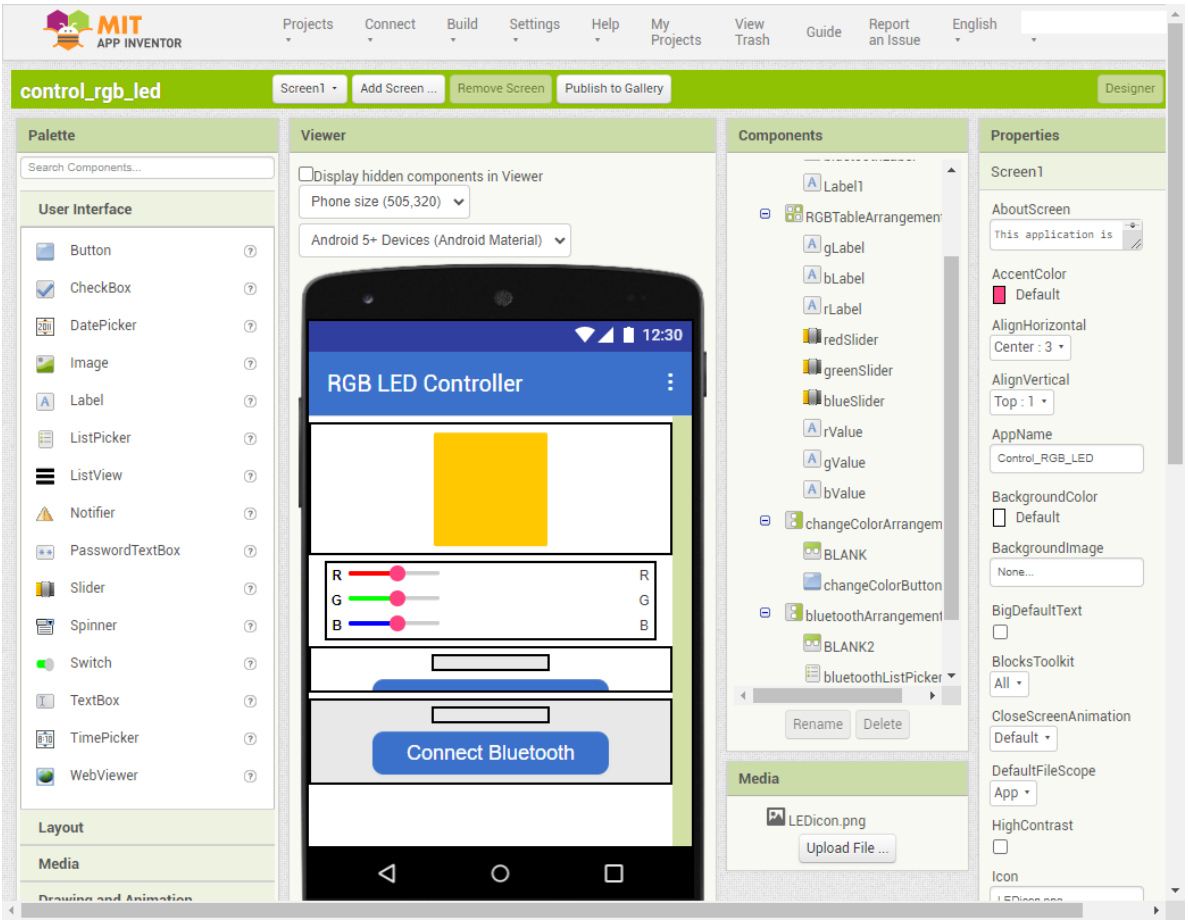
3. Nach dem Hochladen der `.aia`-Datei sehen Sie die Anwendung in der Software **MIT App Inventor**. Dies ist eine vorkonfigurierte Vorlage. Sie können diese Vorlage nach dem Kennenlernen von **MIT App Inventor** über die folgenden Schritte anpassen.



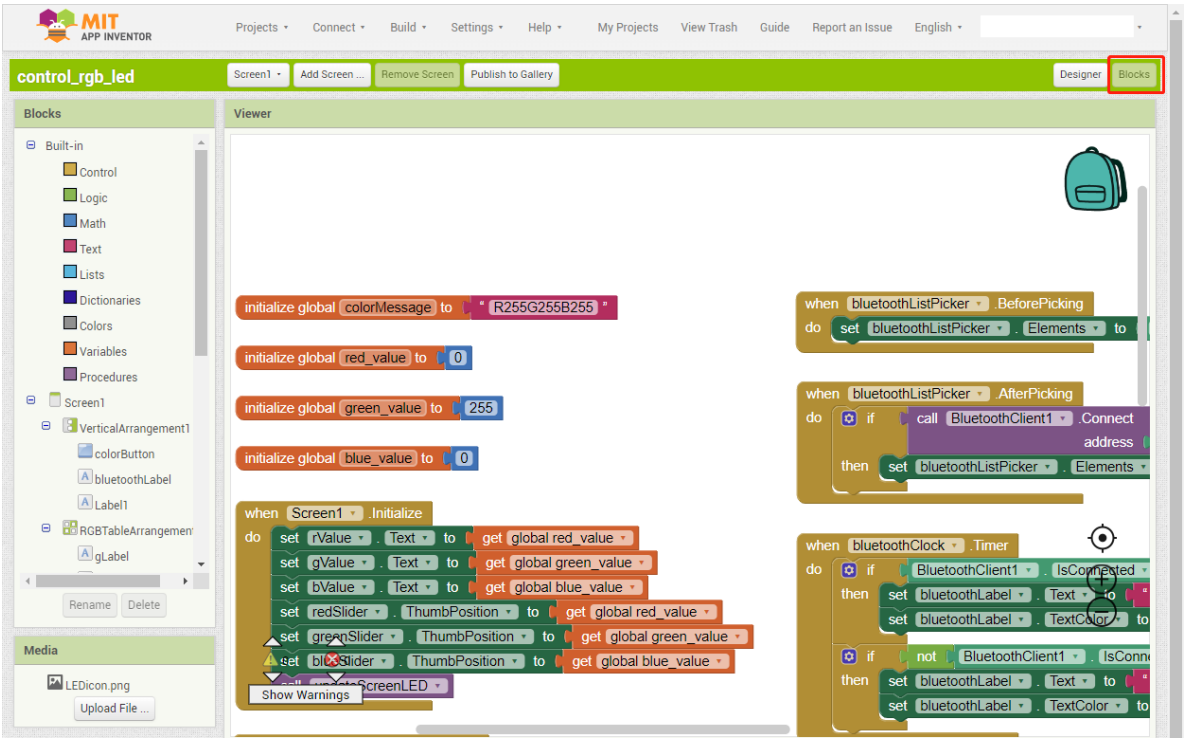
4. In **MIT App Inventor** haben Sie 2 Hauptbereiche: den **Designer** und die **Blocks**.



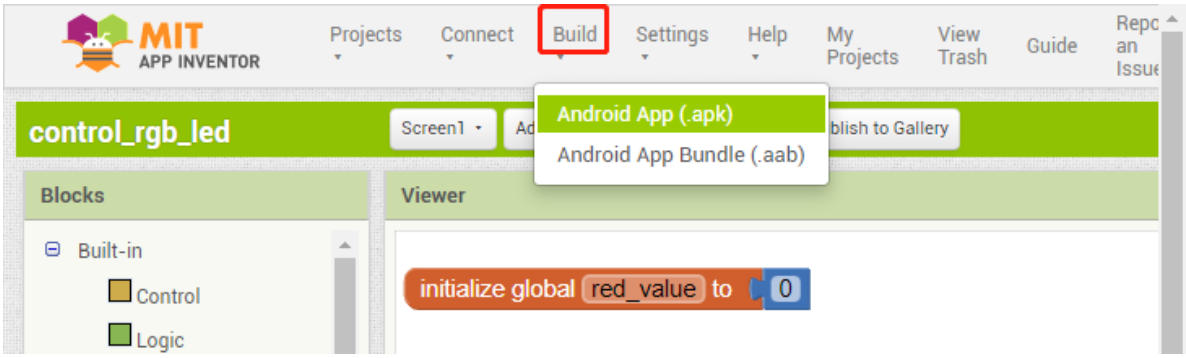
5. Der **Designer** ermöglicht es Ihnen, Schaltflächen, Texte, Bildschirme hinzuzufügen und das gesamte ästhetische Erscheinungsbild Ihrer Anwendung zu modifizieren.



6. Anschließend haben Sie den Bereich **Blocks**. Der Bereich **Blocks** erleichtert die Erstellung maßgeschneiderter Funktionen für Ihre Anwendung.



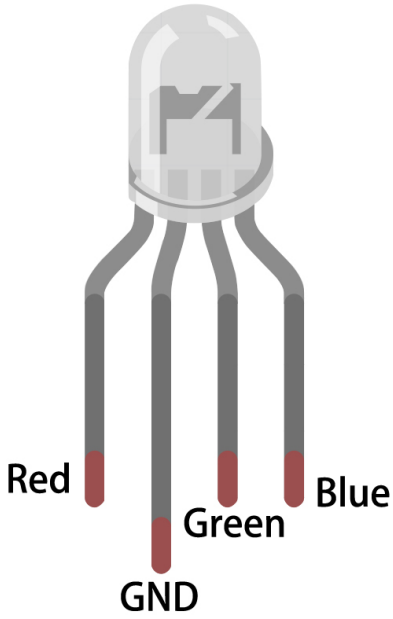
7. Um die Anwendung auf einem Smartphone zu installieren, navigieren Sie zum Tab **Build**.



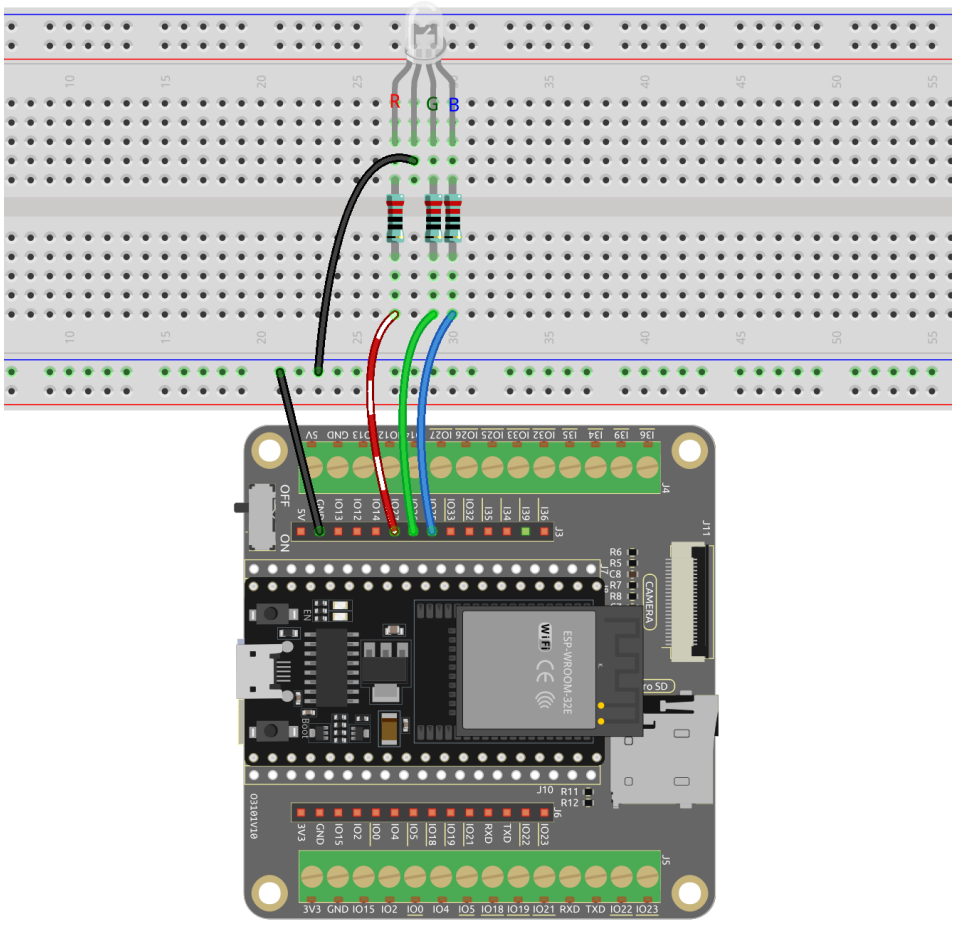
- Sie können eine `.apk`-Datei generieren. Nachdem Sie diese Option ausgewählt haben, erscheint eine Seite, auf der Sie zwischen dem Herunterladen einer `.apk`-Datei oder dem Scannen eines QR-Codes zur Installation wählen können. Befolgen Sie die Installationsanleitung, um die Installation der Anwendung abzuschließen.
- Wenn Sie diese App im **Google Play** oder einem anderen App-Marktplatz hochladen möchten, können Sie eine `.apk`-Datei generieren.

2. Hochladen des Codes

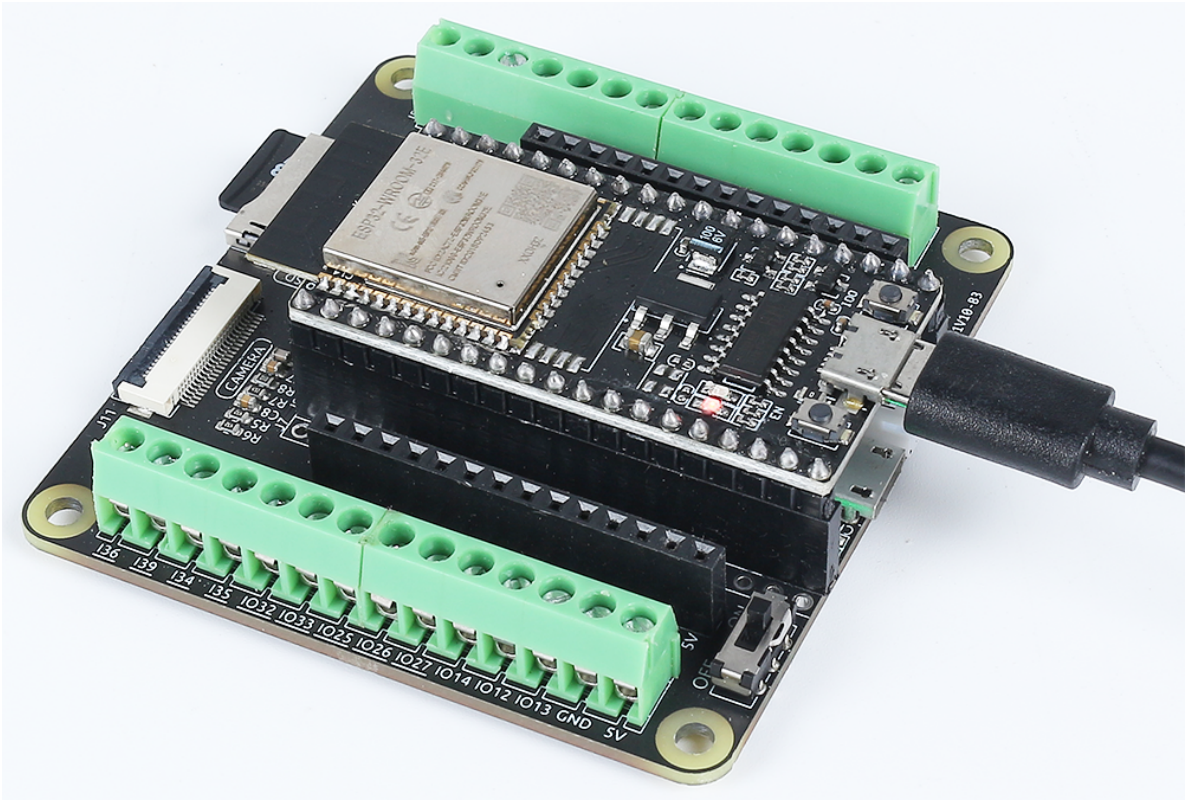
1. Bauen Sie die Schaltung auf.



Die RGB-LED besteht aus 4 Pins: Der längste Pin ist der gemeinsame Kathodenpin, üblicherweise mit GND verbunden; der Pin links vom längsten Pin steht für Rot; und die beiden Pins rechts symbolisieren Grün und Blau.



2. Verbinden Sie anschließend das ESP32-WROOM-32E mit Ihrem Computer über ein USB-Kabel.

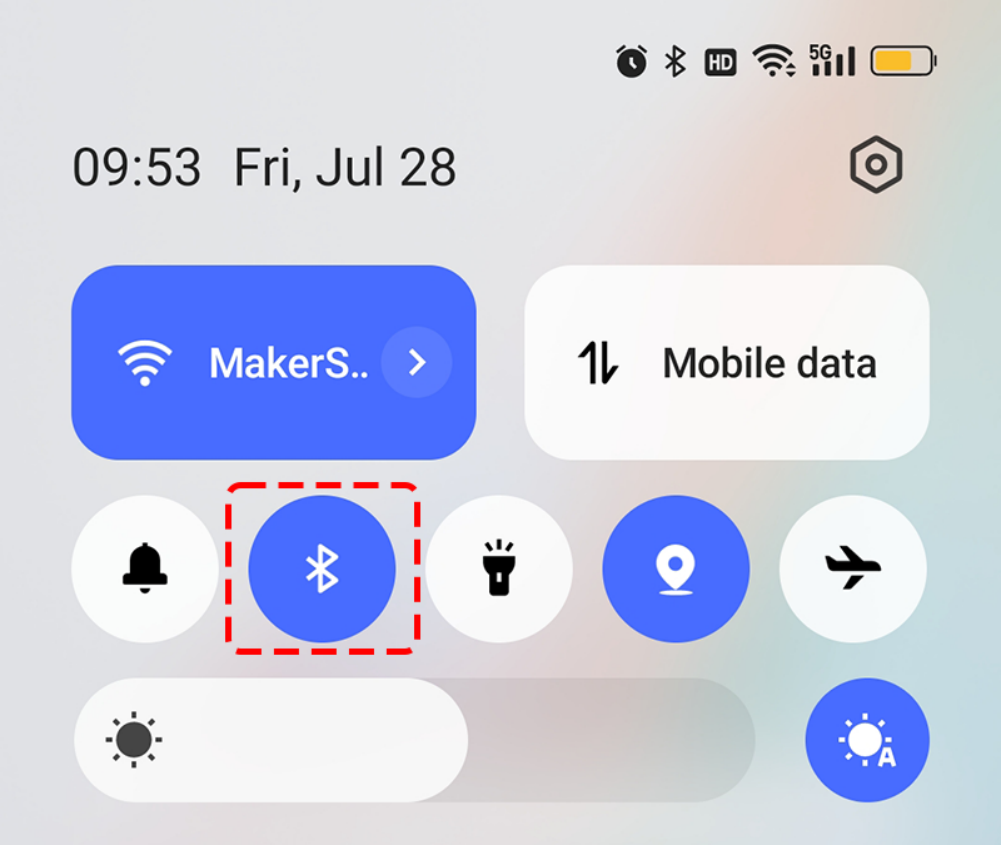


3. Öffnen Sie die Datei `iot_10_bluetooth_app_inventor.ino`, die sich im Verzeichnis `esp32-starter-kit-main\c\codes\iot_10_bluetooth_app_inventor` befindet, oder kopieren Sie den Code in die Arduino IDE.
4. Nachdem Sie das passende Board (**ESP32 Dev Module**) und den Port ausgewählt haben, klicken Sie auf den **Upload**-Button.

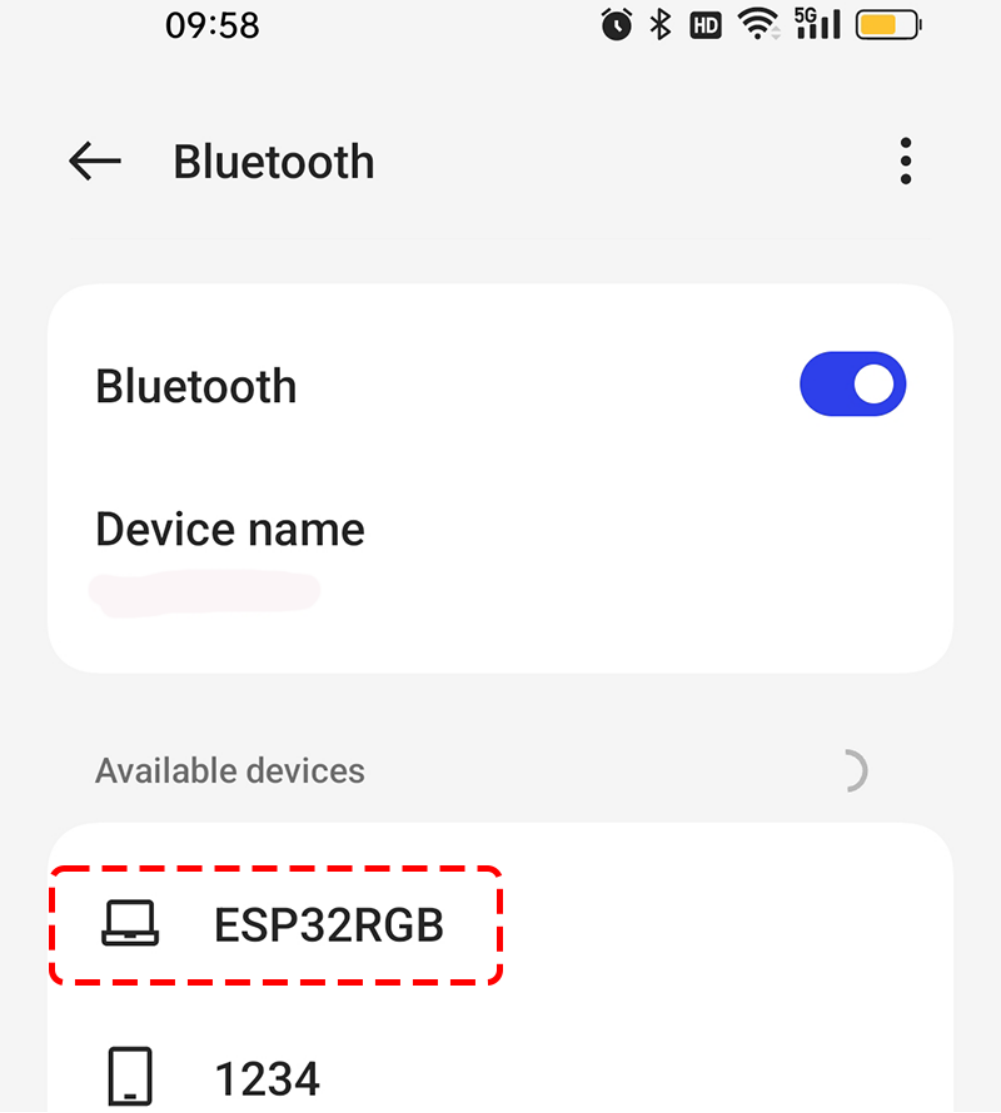
3. App- und ESP32-Verbindung

Stellen Sie sicher, dass die zuvor erstellte Anwendung auf Ihrem Smartphone installiert ist.

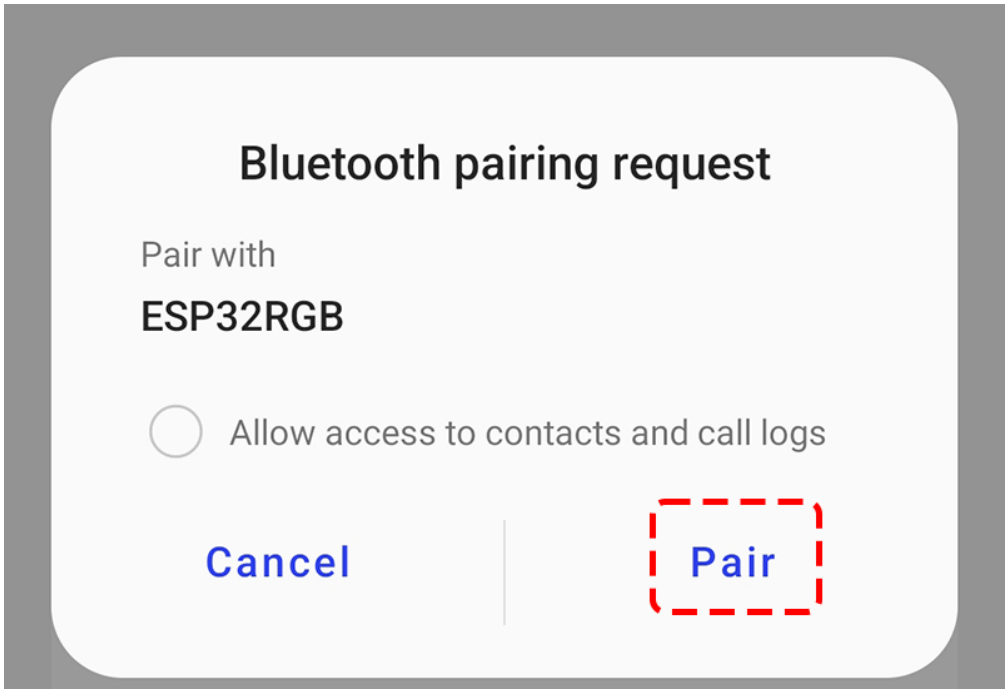
1. Aktivieren Sie zunächst **Bluetooth** auf Ihrem Smartphone.



2. Navigieren Sie zu den **Bluetooth settings** auf Ihrem Smartphone und finden Sie **ESP32RGB**.

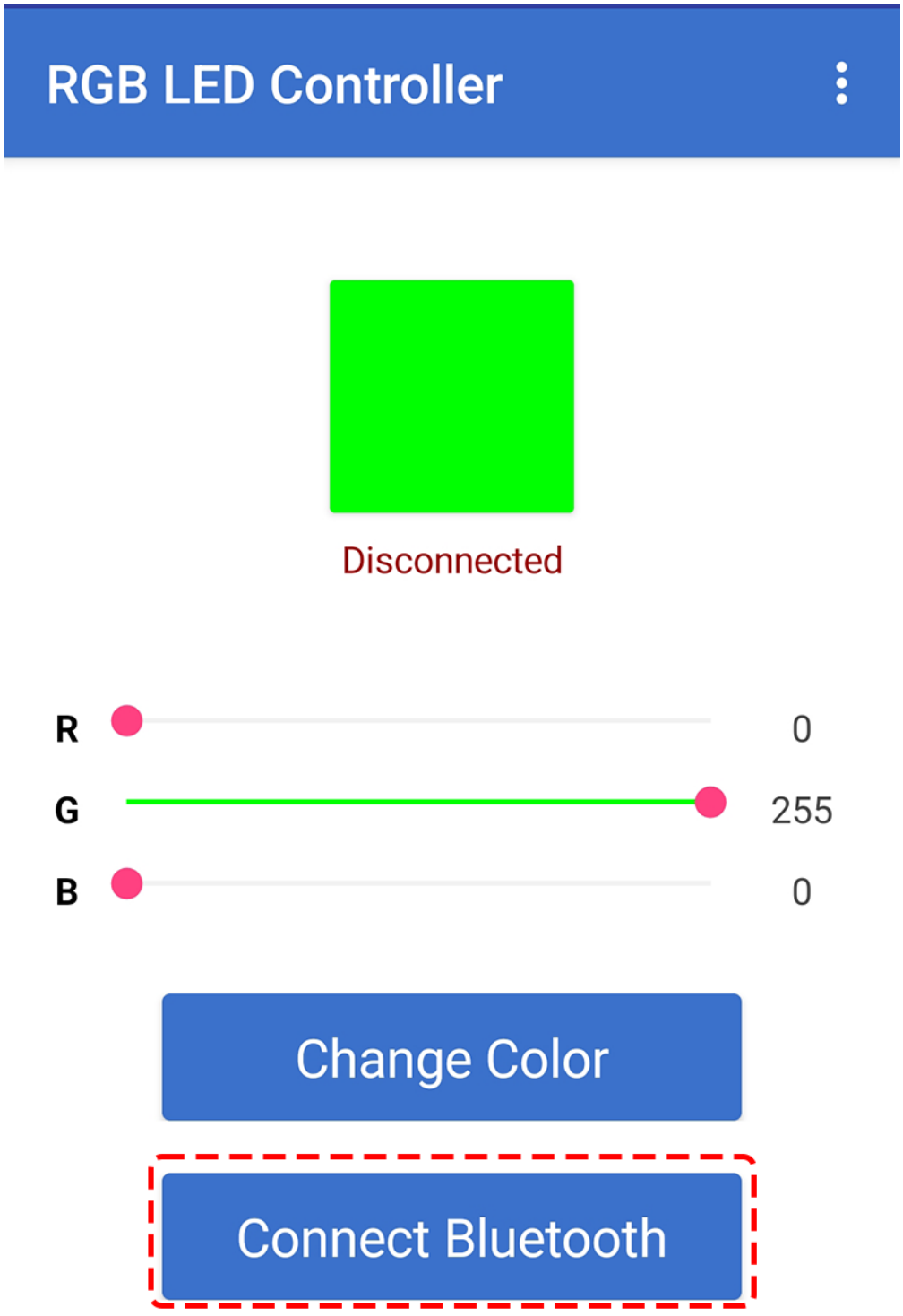


3. Nachdem Sie darauf geklickt haben, stimmen Sie der **Pair**-Anfrage im Pop-up-Fenster zu.

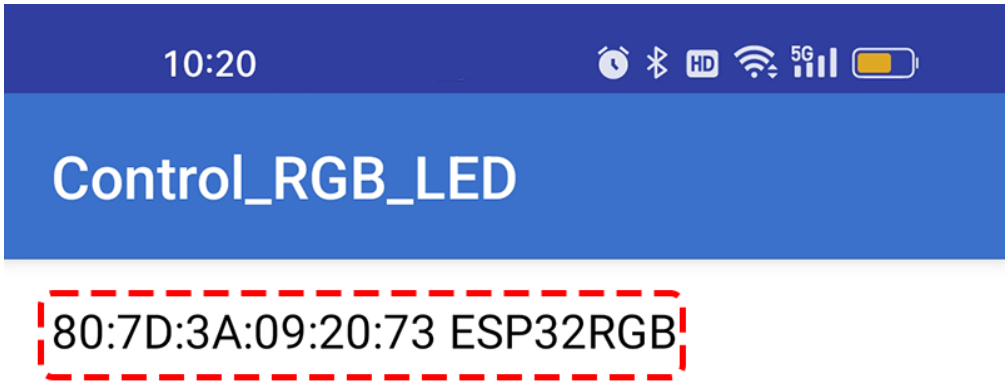


4. Öffnen Sie jetzt die kürzlich installierte **Control_RGB_LED**-APP.

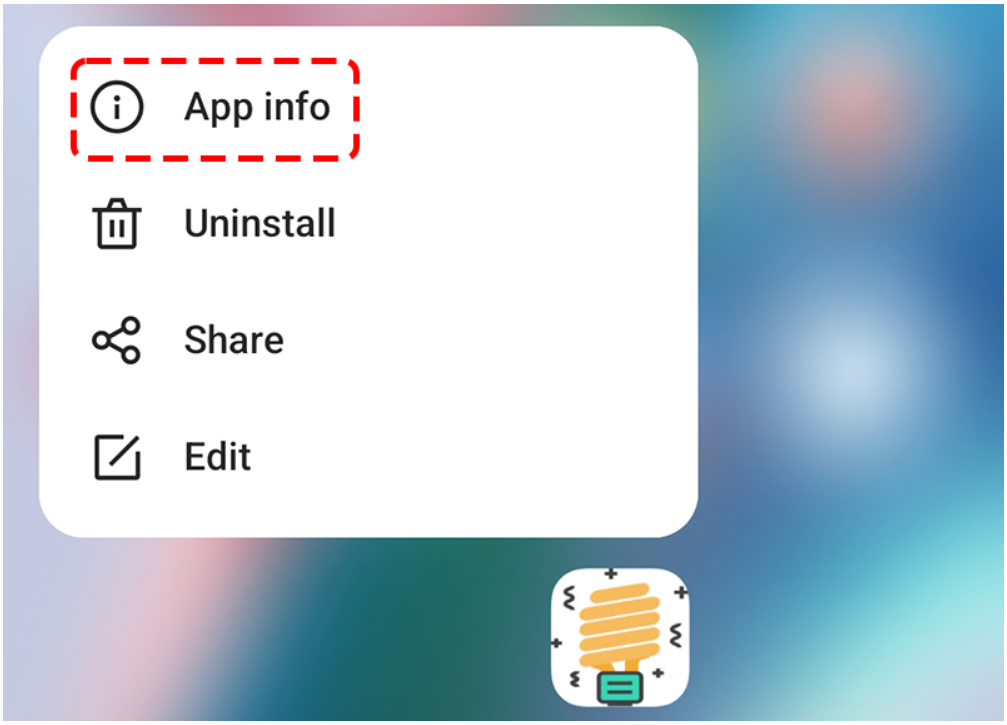
5. Klicken Sie in der APP auf **Connect Bluetooth**, um eine Verbindung zwischen der APP und dem ESP32 herzustellen.



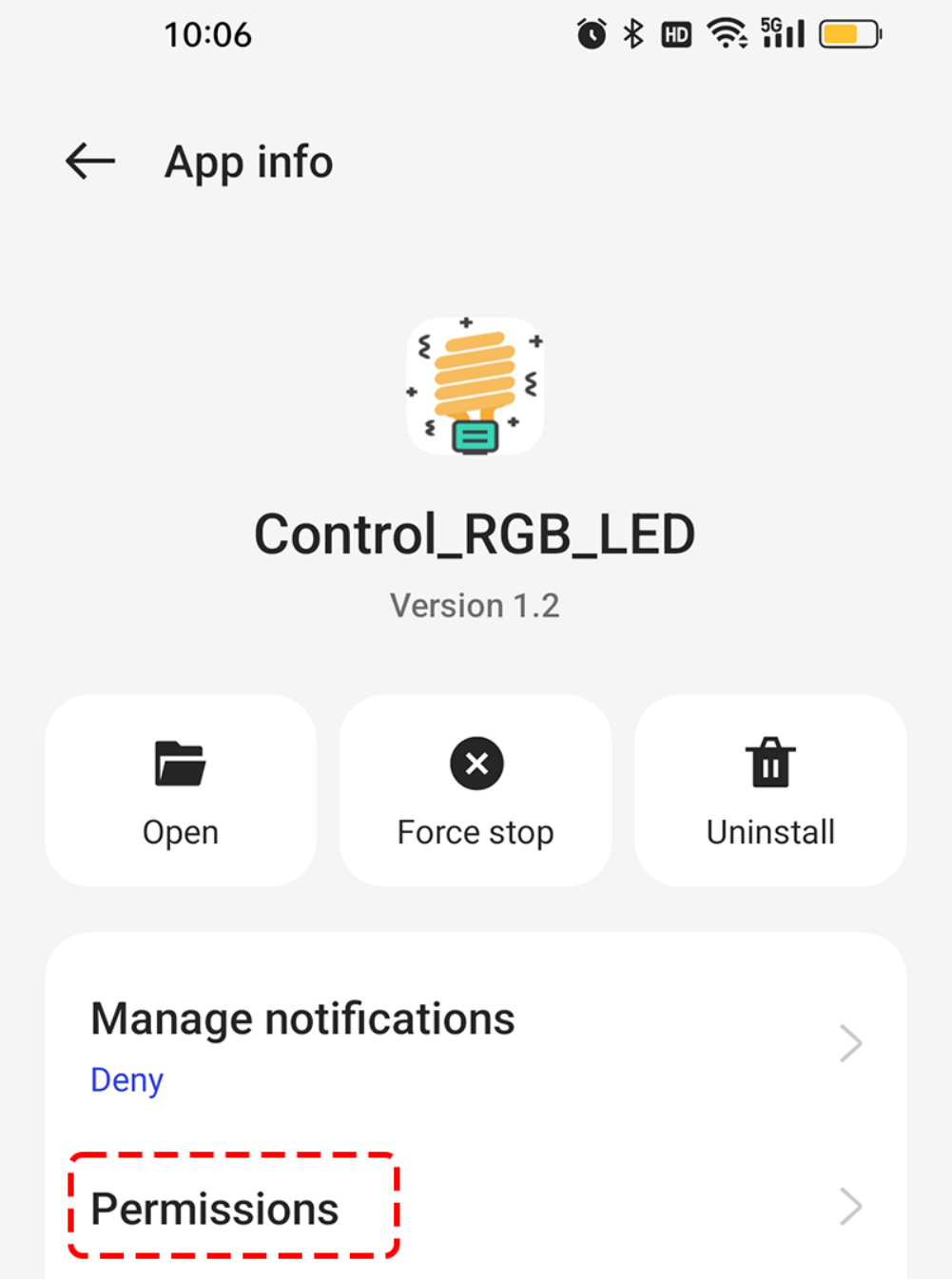
6. Wählen Sie das `xx.xx.xx.xx.xx.xx ESP32RGB`, das angezeigt wird. Wenn Sie `SerialBT.begin("ESP32RGB");` im Code geändert haben, wählen Sie einfach den Namen Ihrer Einstellung.



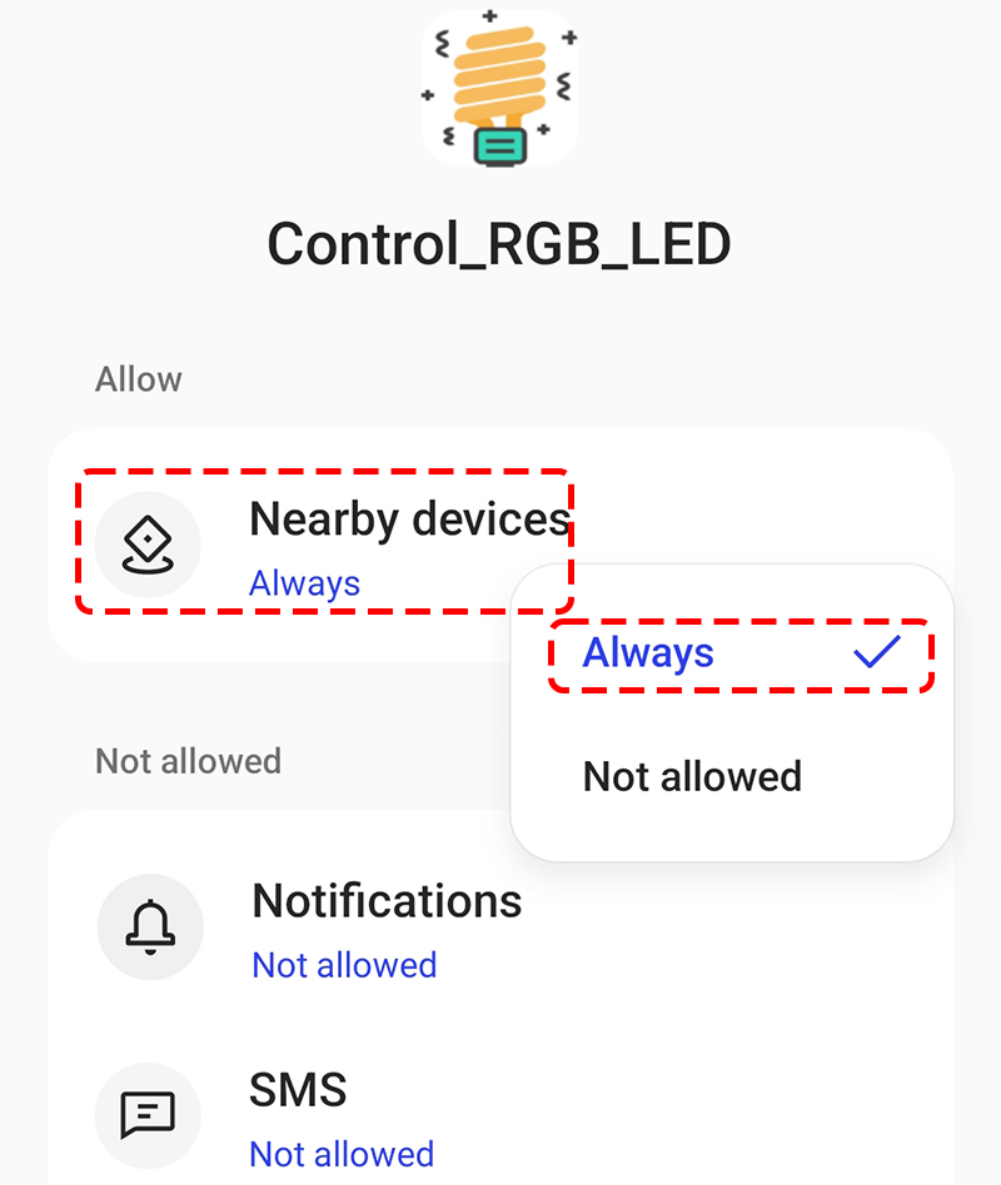
7. Wenn Sie eine Weile gewartet haben und immer noch keine Gerätenamen sehen, kann es sein, dass diese APP nicht erlaubt ist, umliegende Geräte zu scannen. In diesem Fall müssen Sie die Einstellungen manuell anpassen.
    - Halten Sie das APP-Symbol lange gedrückt und klicken Sie auf die sich daraus ergebende **APP Info**. Wenn Sie einen anderen Weg haben, um auf diese Seite zuzugreifen, folgen Sie diesem.



    - Navigieren Sie zur Seite **Permissions**.

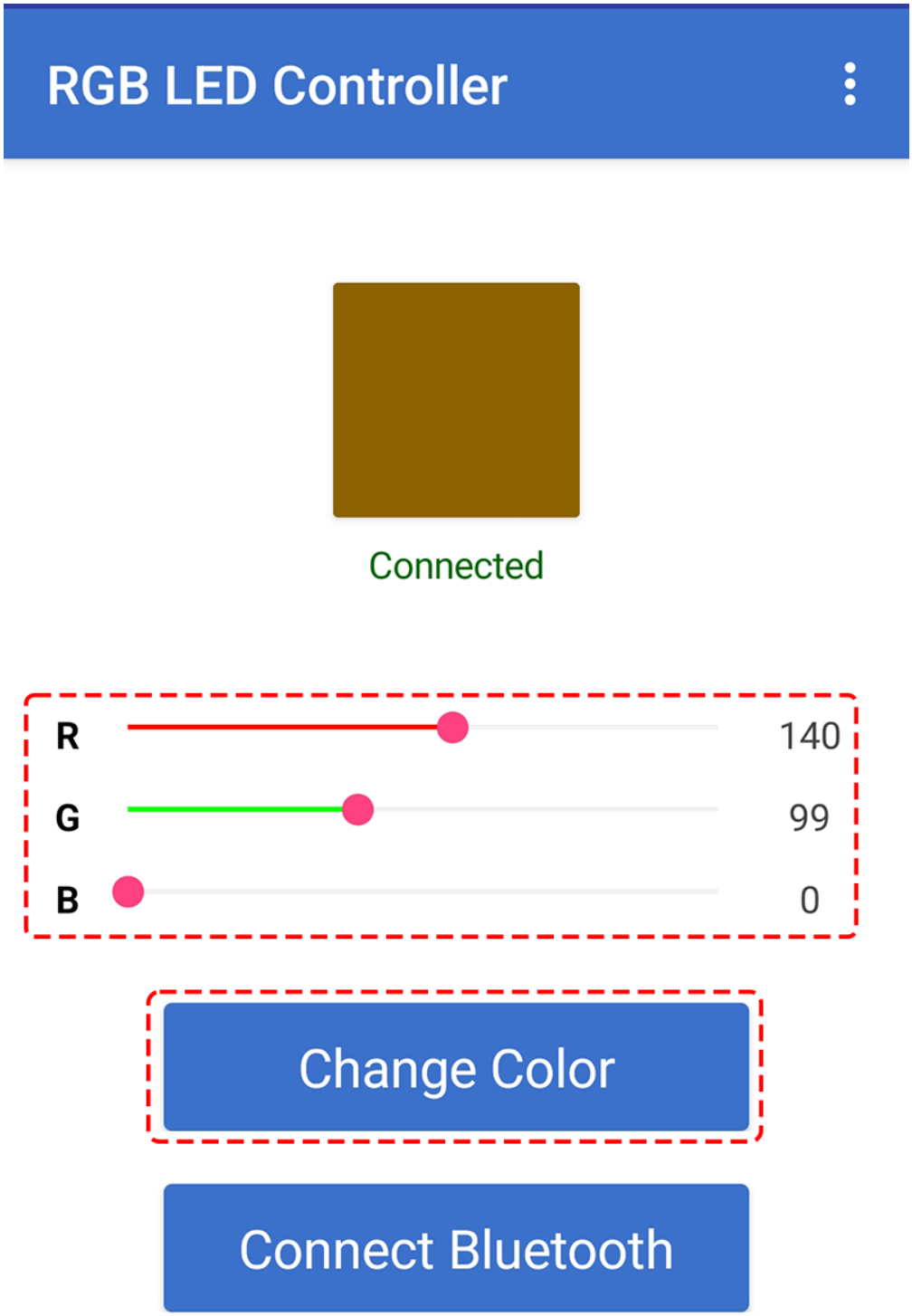


- Suchen Sie nach **Nearby devices** und wählen Sie **Always**, um dieser APP das Scannen von Geräten in der Nähe zu erlauben.



- Starten Sie nun die APP neu und wiederholen Sie die Schritte 5 und 6, um erfolgreich eine Bluetooth-Verbindung herzustellen.

8. Nach erfolgreicher Verbindung werden Sie automatisch zur Hauptseite zurückgeführt, wo es als verbunden angezeigt wird. Jetzt können Sie die RGB-Werte anpassen und die Farbe der RGB-Anzeige ändern, indem Sie auf den Button **Change Color** drücken.


RGB LED Controller
Connected
R
140
G
99
B
0
Change Color
Connect Bluetooth

KAPITEL 3

Für MicroPython-Anwender

Dieses Kapitel ist ein umfassender Leitfaden, der speziell für Anwender konzipiert ist, die mit MicroPython arbeiten möchten. Es behandelt verschiedene Themen, darunter den Einstieg in MicroPython, die Arbeit mit Displays, die Erzeugung von Tönen, die Steuerung von Aktuatoren, den Einsatz von Sensoren und das Erkunden von spannenden Projekten. Dieses Kapitel versorgt MicroPython-Anwender mit dem notwendigen Wissen und den Ressourcen, um dieses Kit effektiv zu nutzen und ihre Kreativität beim Bau aufregender Projekte zu entfalten.

Hier ist das komplette Code-Paket für das ESP32 Starter Kit. Sie können auf den folgenden Link klicken, um es herunterzuladen:

- `SunFounder ESP32 Starter Kit`

Nachdem der Download abgeschlossen ist, entpacken Sie die Datei und öffnen Sie die entsprechenden Beispielcodes oder Projektdateien in der entsprechenden Software. Dies ermöglicht es Ihnen, alle vom Kit bereitgestellten Codes und Ressourcen zu durchsuchen und zu nutzen.

1. Erste Schritte

3.1 1.1 Einführung in MicroPython

MicroPython ist eine Softwareimplementierung einer Programmiersprache, die weitgehend mit Python 3 kompatibel ist, in C geschrieben und optimiert, um auf einem Mikrocontroller zu laufen.[3][4]

MicroPython besteht aus einem Python-Compiler zu Bytecode und einem Laufzeitinterpreter dieses Bytecodes. Dem Benutzer wird eine interaktive Aufforderung (das REPL) präsentiert, um unterstützte Befehle sofort auszuführen. Enthalten ist eine Auswahl an Kern-Python-Bibliotheken; MicroPython umfasst Module, die dem Programmierer Zugriff auf hardwarenahe Funktionen ermöglichen.

- Referenz: MicroPython - Wikipedia

3.1.1 Hier beginnt die Geschichte

2013 änderte sich alles, als Damien George eine Crowdfunding-Kampagne (Kickstarter) startete.

Damien war ein Student an der Universität Cambridge und ein begeisterter Robotik-Programmierer. Er wollte die Welt von Python von einer Gigabyte-Maschine auf ein Kilobyte reduzieren. Seine Kickstarter-Kampagne sollte seine Entwicklung unterstützen, während er seinen Proof of Concept in eine fertige Implementierung umwandelte.

MicroPython wird von einer vielfältigen Pythonista-Community unterstützt, die großes Interesse am Erfolg des Projekts hat.

Neben dem Testen und Unterstützen der Codebasis lieferten die Entwickler Tutorials, Codebibliotheken und Hardware-Portierungen, sodass sich Damien auf andere Aspekte des Projekts konzentrieren konnte.

- Referenz: realpython

3.1.2 Warum MicroPython

Obwohl die ursprüngliche Kickstarter-Kampagne MicroPython als Entwicklungsboard „pyboard“ mit STM32F4 veröffentlichte, unterstützt MicroPython viele auf ARM basierende Produktarchitekturen. Die wichtigsten unterstützten Ports sind ARM Cortex-M (viele STM32-Boards, TI CC3200/WiPy, Teensy-Boards, Nordic nRF-Serie, SAMD21 und SAMD51), ESP8266, ESP32, 16-Bit-PIC, Unix, Windows, Zephyr und JavaScript. Zweitens ermöglicht MicroPython schnelles Feedback. Dies liegt daran, dass Sie REPL verwenden können, um Befehle interaktiv einzugeben und Antworten zu erhalten. Sie können sogar Code anpassen und sofort ausführen, anstatt den Zyklus Code-Kompilieren-Upload-Ausführen zu durchlaufen.

Während Python dieselben Vorteile bietet, sind einige Mikrocontroller-Boards wie das ESP32 klein, einfach und haben zu wenig Speicher, um überhaupt die Python-Sprache auszuführen. Deshalb hat sich MicroPython entwickelt, wobei die Hauptmerkmale von Python beibehalten und eine Reihe neuer Funktionen hinzugefügt wurden, um mit diesen Mikrocontroller-Boards zu arbeiten.

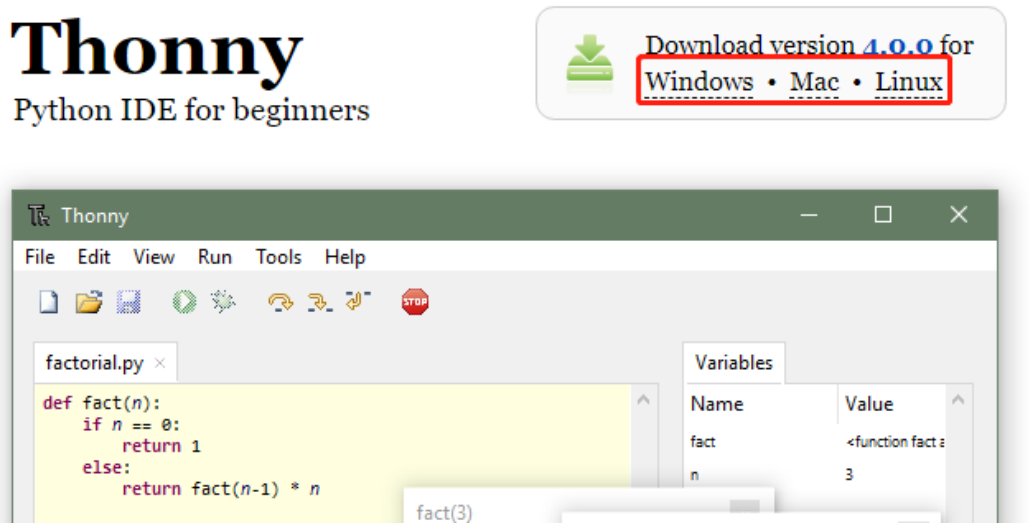
Als Nächstes lernen Sie, MicroPython auf das ESP32 zu installieren.

- Referenz: MicroPython - Wikipedia
- Referenz: realpython

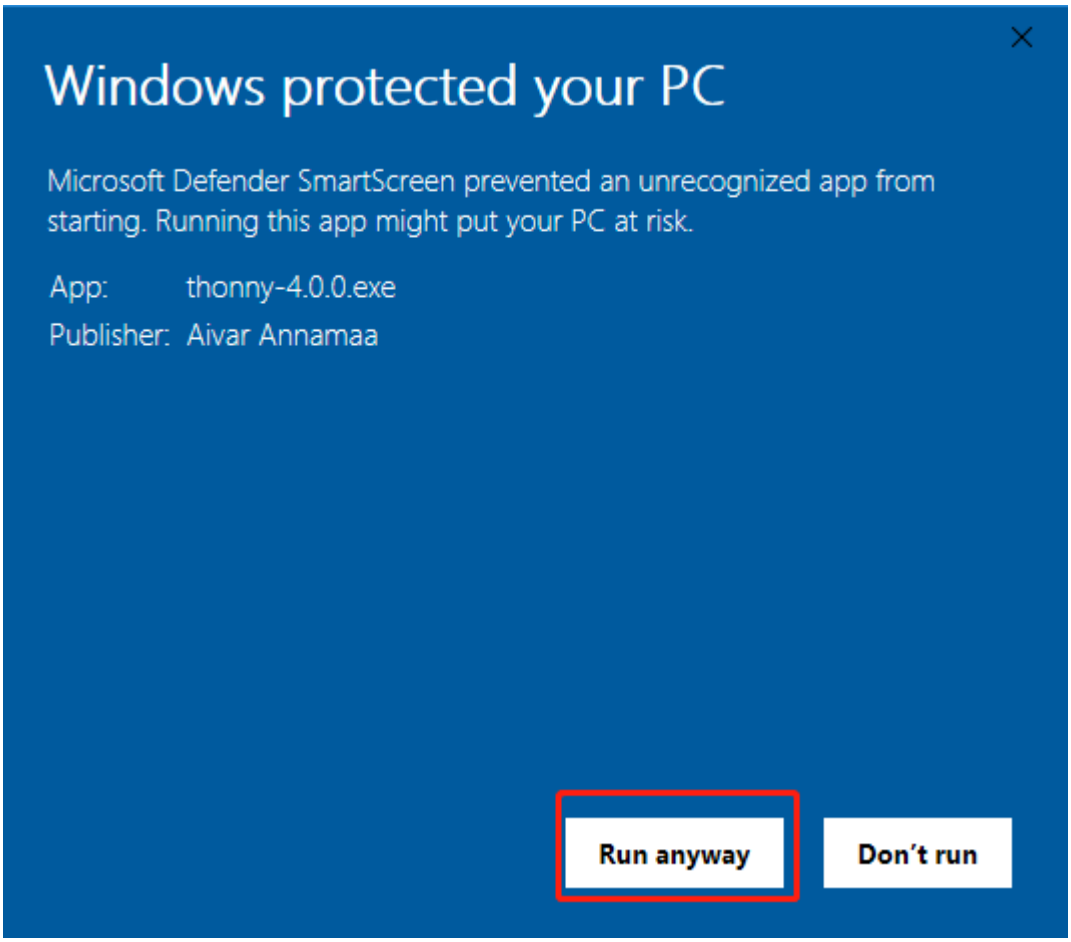
3.2 1.2 Installation der Thonny IDE

Bevor Sie anfangen können, den ESP32 mit MicroPython zu programmieren, benötigen Sie eine integrierte Entwicklungsumgebung (IDE). Hier empfehlen wir Thonny. Thonny kommt mit Python 3.7 eingebaut, Sie benötigen nur einen einfachen Installer, und schon können Sie mit dem Programmieren beginnen.

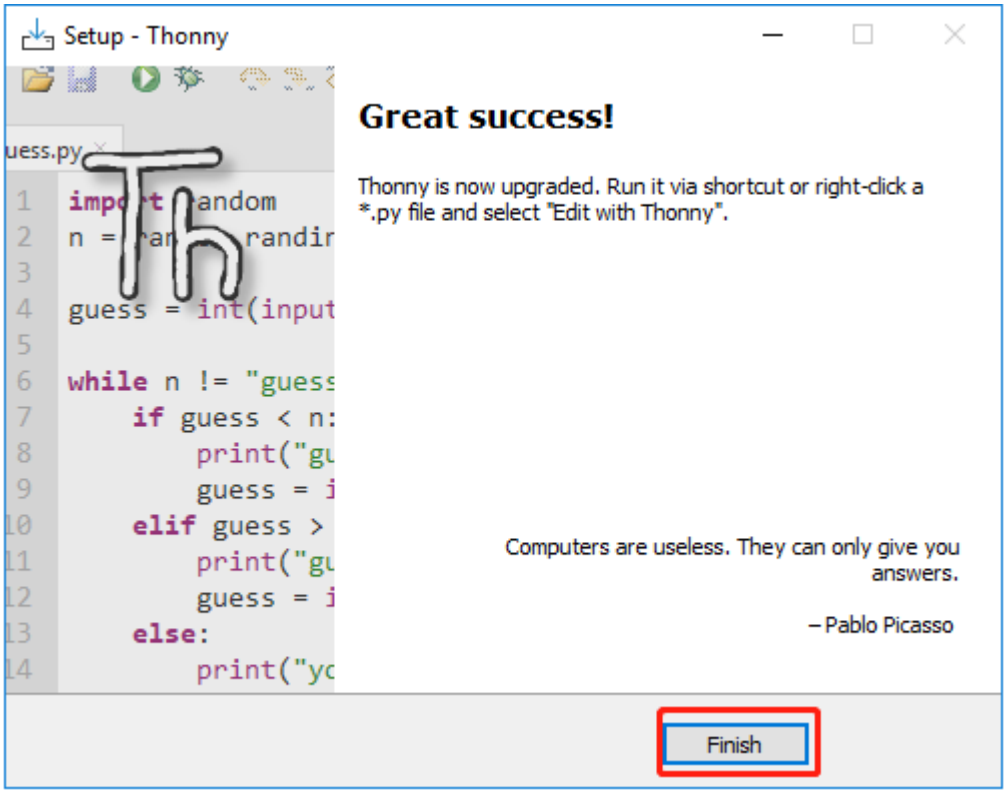
1. Sie können Thonny herunterladen, indem Sie die Website besuchen. Sobald Sie die Seite öffnen, sehen Sie oben rechts ein hellgraues Feld, klicken Sie auf den Link, der zu Ihrem Betriebssystem passt.



2. Die Installationsprogramme wurden mit einem neuen Zertifikat signiert, das noch keinen Ruf aufgebaut hat. Möglicherweise müssen Sie eine Warnung Ihres Browsers umgehen (z.B. in Chrome „Behalten“ anstatt „Verwerfen“ wählen) und eine Warnung von Windows Defender (**More info Run anyway**).

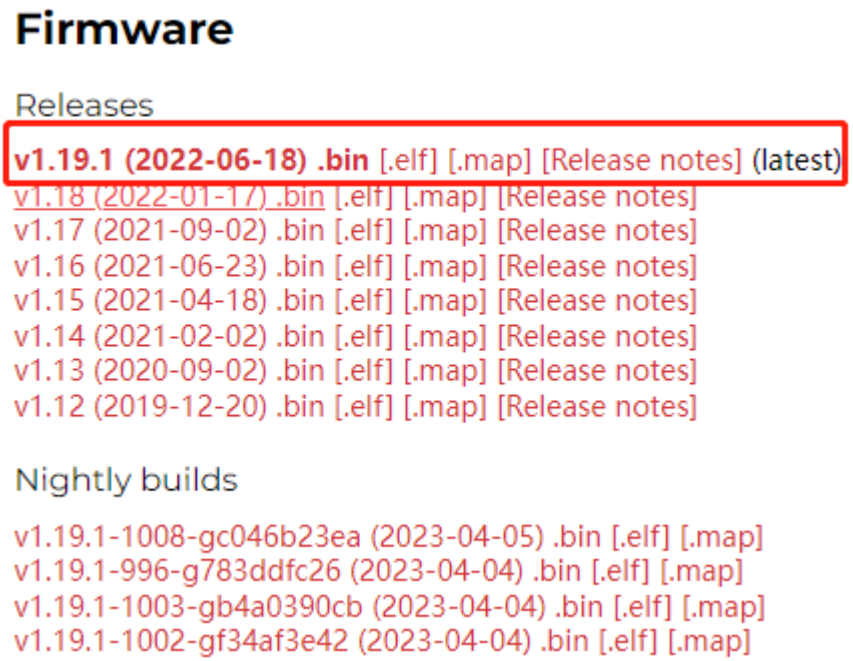


3. Klicken Sie anschließend auf **Next** und **Install**, um die Installation von Thonny abzuschließen.

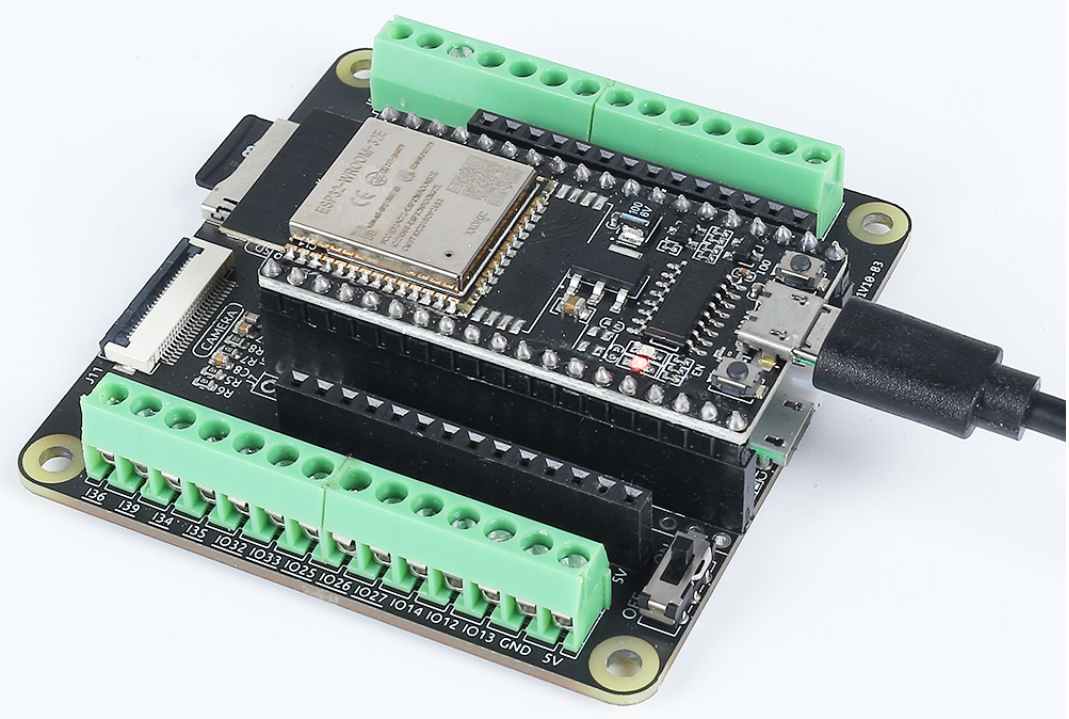


3.3 1.3 Installation von MicroPython auf dem ESP32(Wichtig)

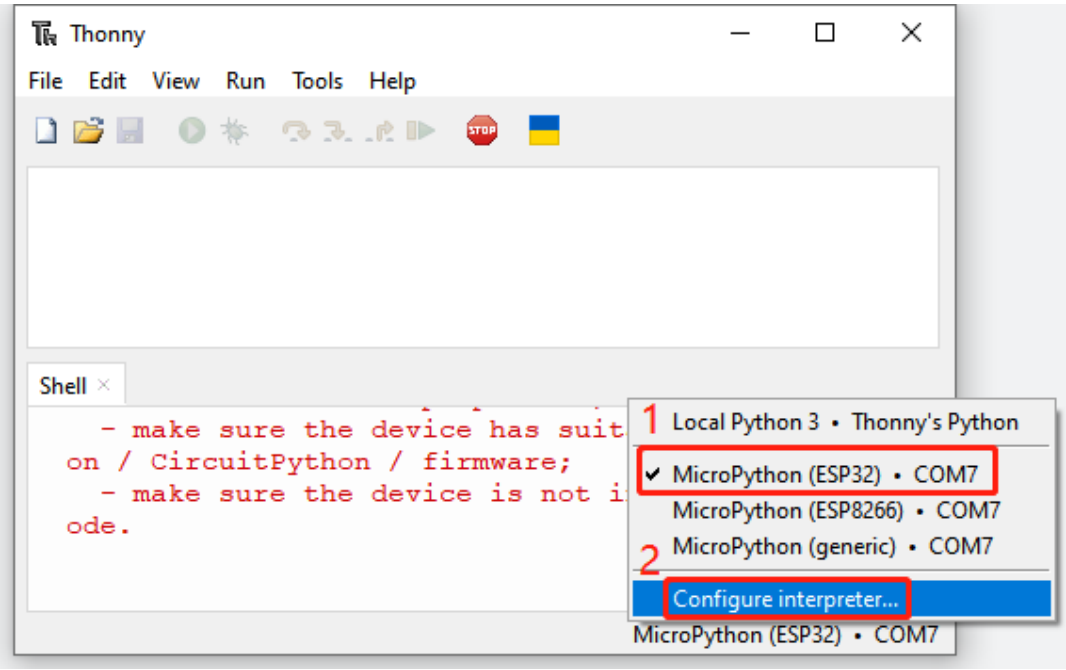
1. Laden Sie die von der offiziellen MicroPython-Website herunter und anschließend die neueste Version der Firmware.



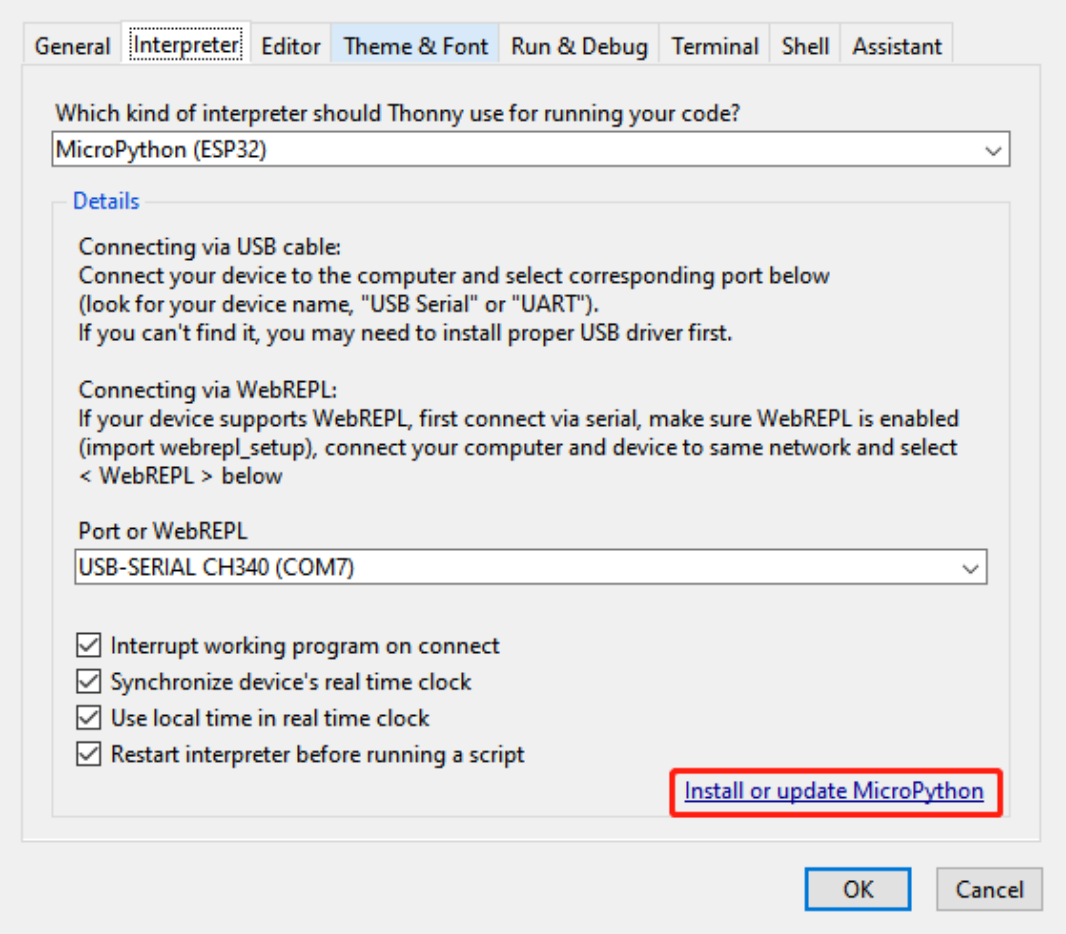
2. Verbinden Sie das ESP32 WROOM 32E mit Ihrem Computer über ein Micro-USB-Kabel.



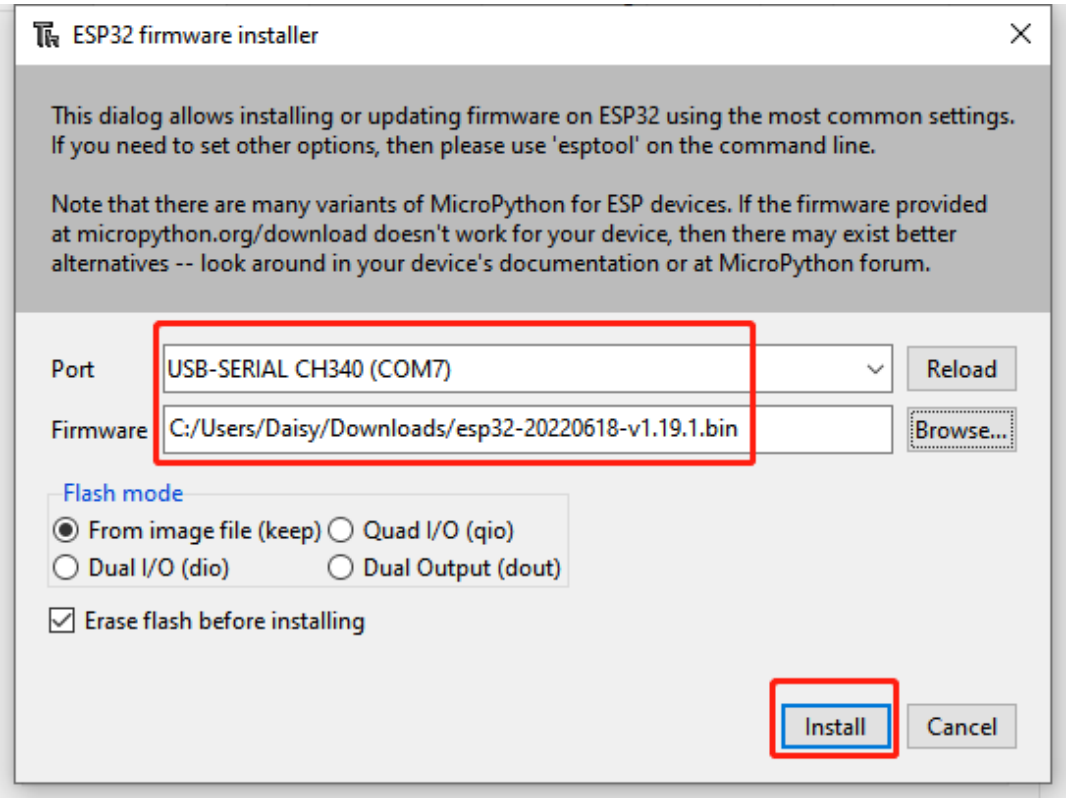
3. Klicken Sie in der unteren rechten Ecke der Thonny IDE, wählen Sie im aufklappenden Menü **„MicroPython(ESP32).COMXX“** aus und dann **„Configure interpreter“**.



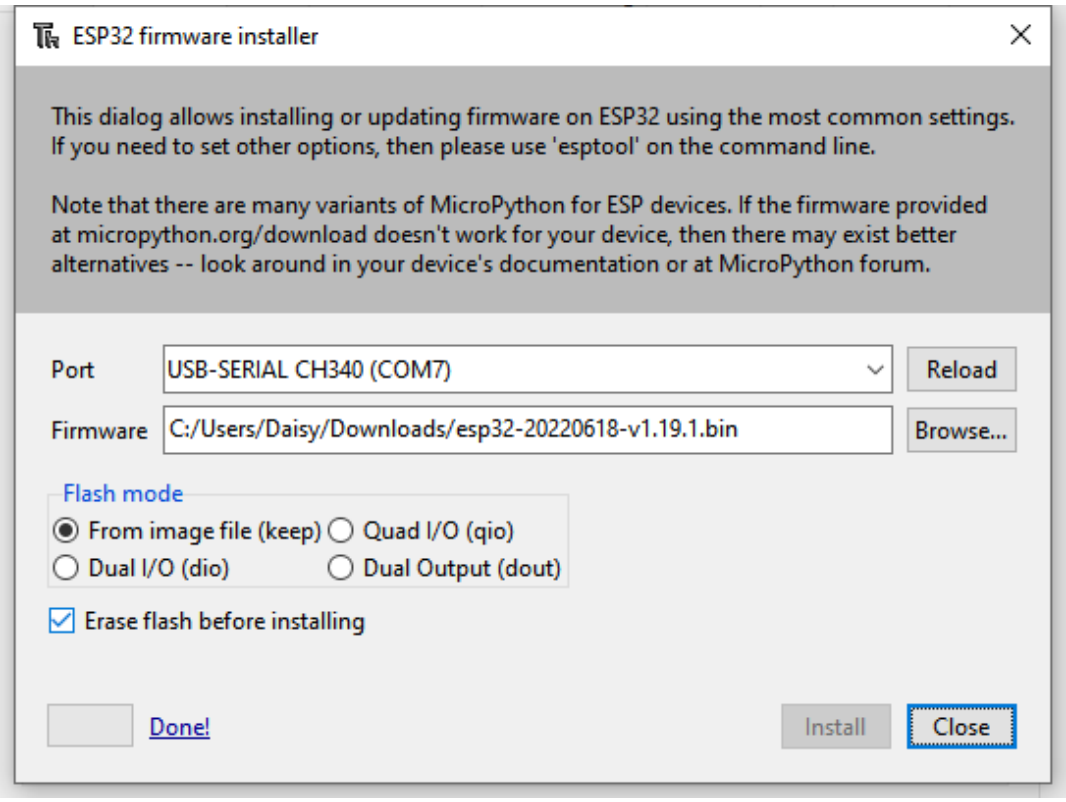
4. Klicken Sie im neuen Popup-Fenster auf **„Install or Update MicroPython“**.



5. Wählen Sie den korrekten Port und die zuvor heruntergeladene Firmware aus und klicken Sie auf **„Install“**.

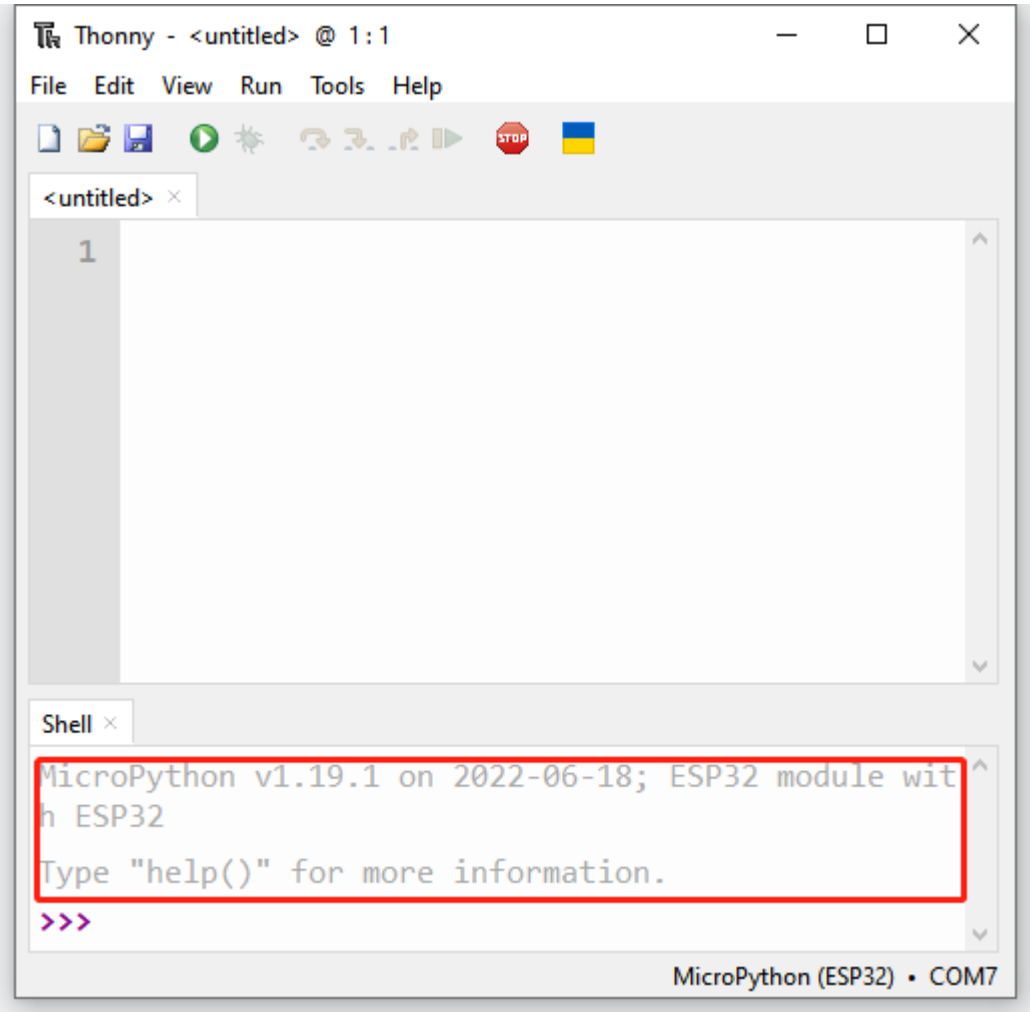


6. Nach einer erfolgreichen Installation können Sie diese Seite schließen.



7. Wenn Sie zur Thonny-Startseite zurückkehren, sehen Sie die MicroPython-Version und ESP32-bezogene Hin-

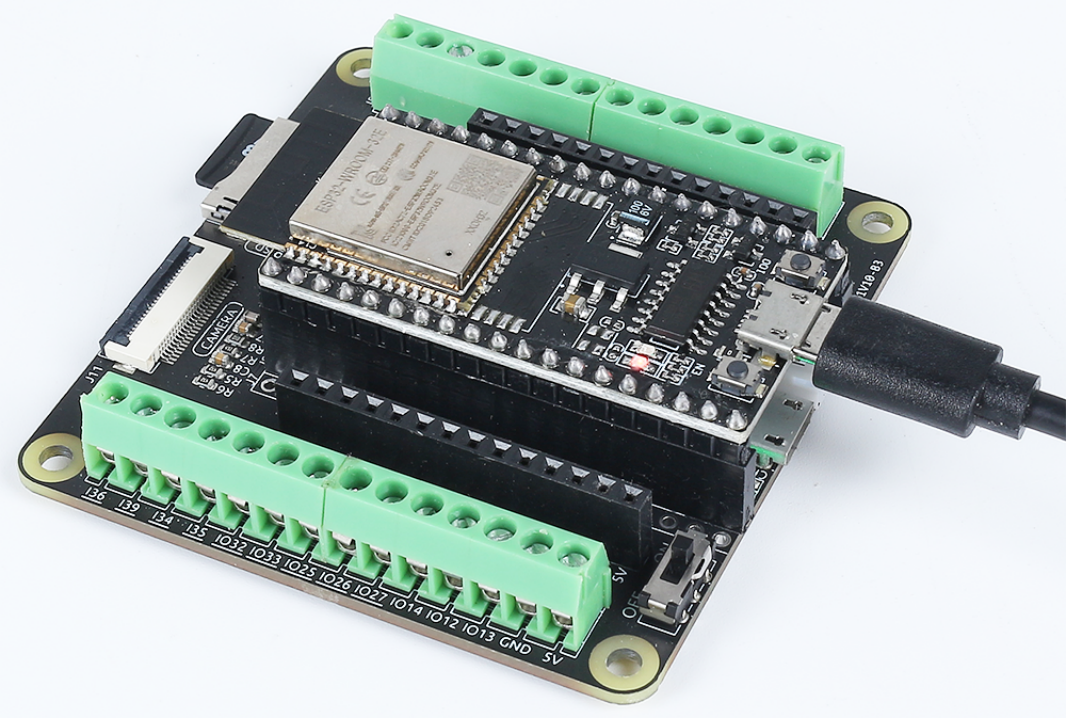
weise anstelle von roten Fehlermeldungen.



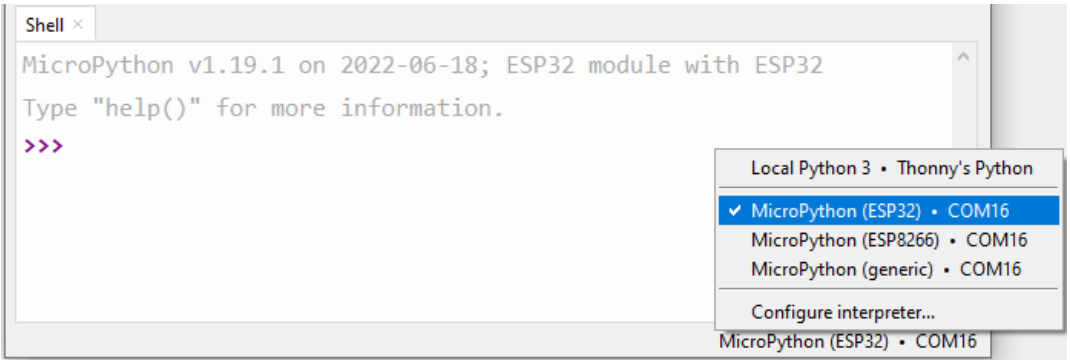
3.4 1.4 Bibliotheken Hochladen (Wichtig)

In manchen Projekten benötigt man zusätzliche Bibliotheken. Daher laden wir diese zuerst auf den ESP32 hoch, sodass wir den Code später direkt ausführen können.

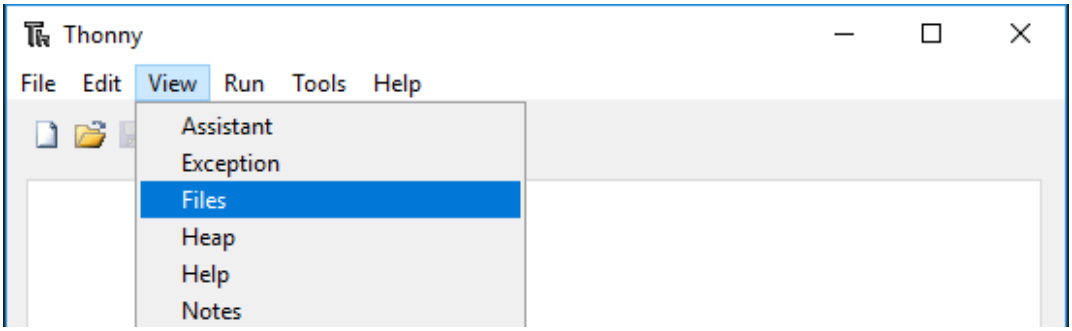
1. Laden Sie den relevanten Code von dem folgenden Link herunter.
    - `SunFounder ESP32 Starter Kit`
2. Verbinden Sie das ESP32 WROOM 32E mit Ihrem Computer über ein Micro-USB-Kabel.



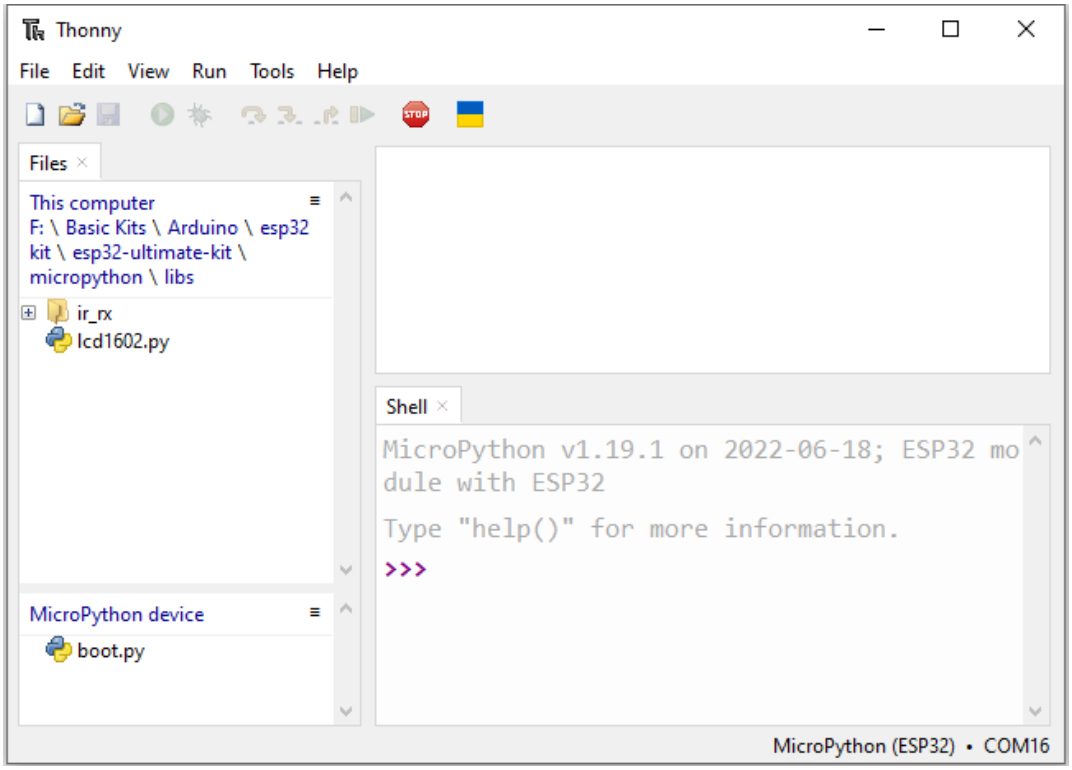
3. Öffnen Sie die Thonny IDE und klicken Sie in der unteren rechten Ecke auf den Interpreter **„MicroPython (ESP32).COMXX“**.



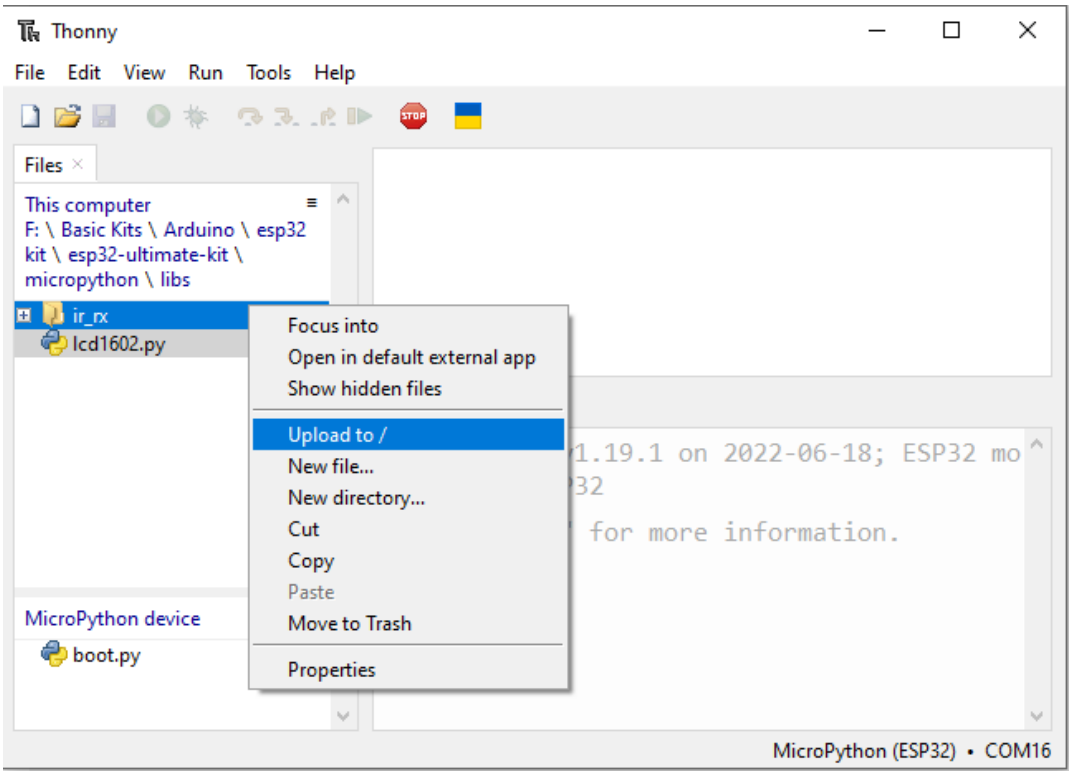
4. Klicken Sie in der oberen Navigationsleiste auf **View** -> **Files**.



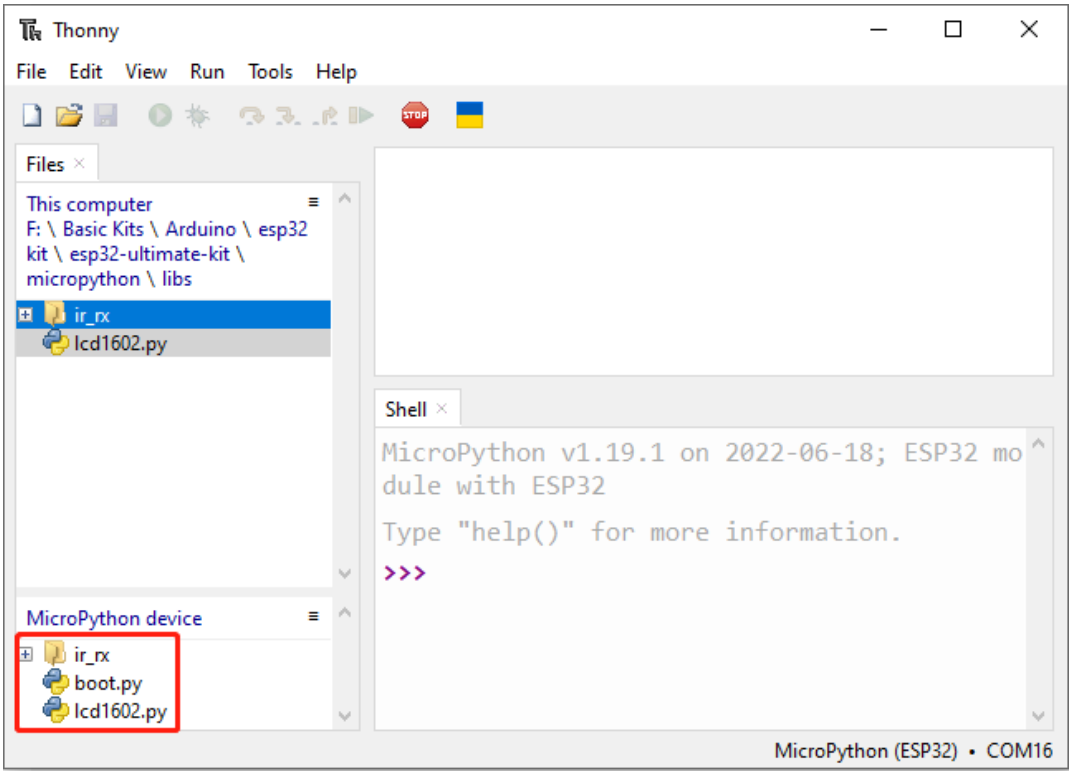
5. Wechseln Sie den Pfad zum Ordner, in dem Sie zuvor das Codepaket heruntergeladen haben, und gehen Sie dann zum Ordner `esp32-starter-kit-main\micropython\libs`.



6. Wählen Sie alle Dateien oder Ordner im Ordner `libs/`, klicken Sie mit der rechten Maustaste und wählen Sie **Upload to**, das Hochladen dauert einen Moment.



7. Jetzt sehen Sie die Dateien, die Sie gerade in Ihrem Laufwerk `MicroPython device` hochgeladen haben.



3.5 1.5 Schnelle Anleitung zu Thonny

3.5.1 Code Direkt Öffnen und Ausführen

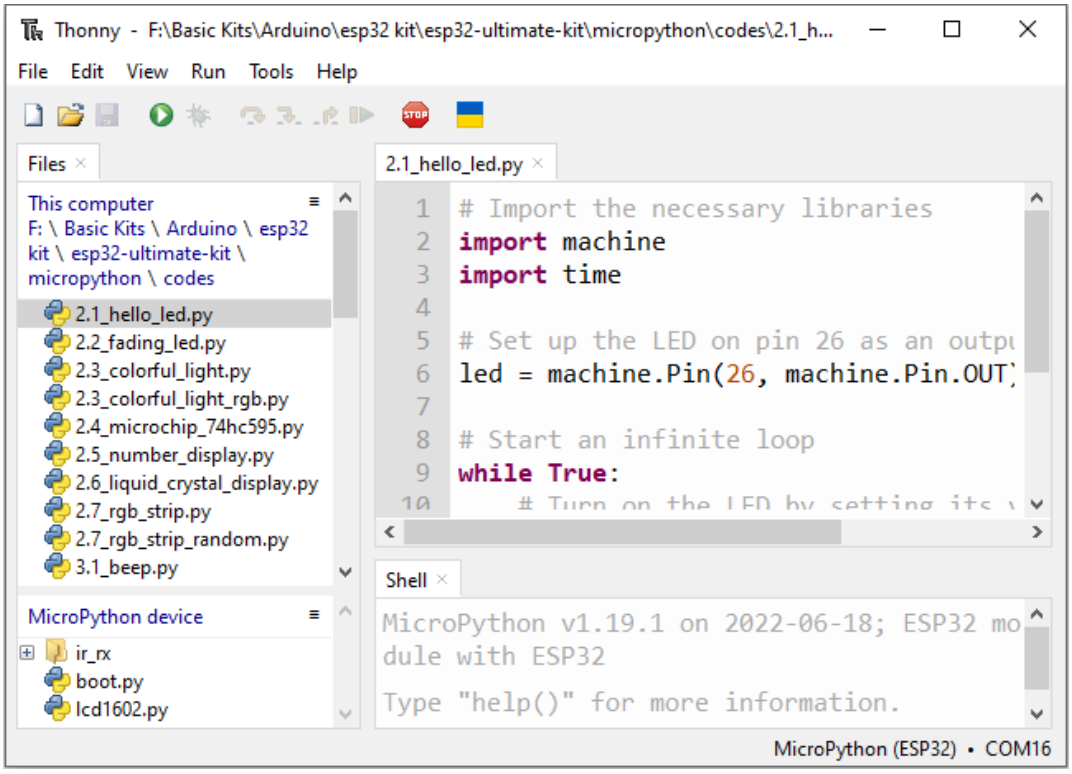
Der Codeabschnitt in den Projekten zeigt genau an, welcher Code verwendet wird. Doppelklicken Sie daher auf die Datei `.py` mit der Seriennummer im Pfad `esp32-starter-kit-main\micropython\codes\`, um sie zu öffnen.

Zuvor müssen Sie jedoch das Paket herunterladen und die Bibliotheken hochladen, wie in *1.4 Bibliotheken Hochladen (Wichtig)* beschrieben.

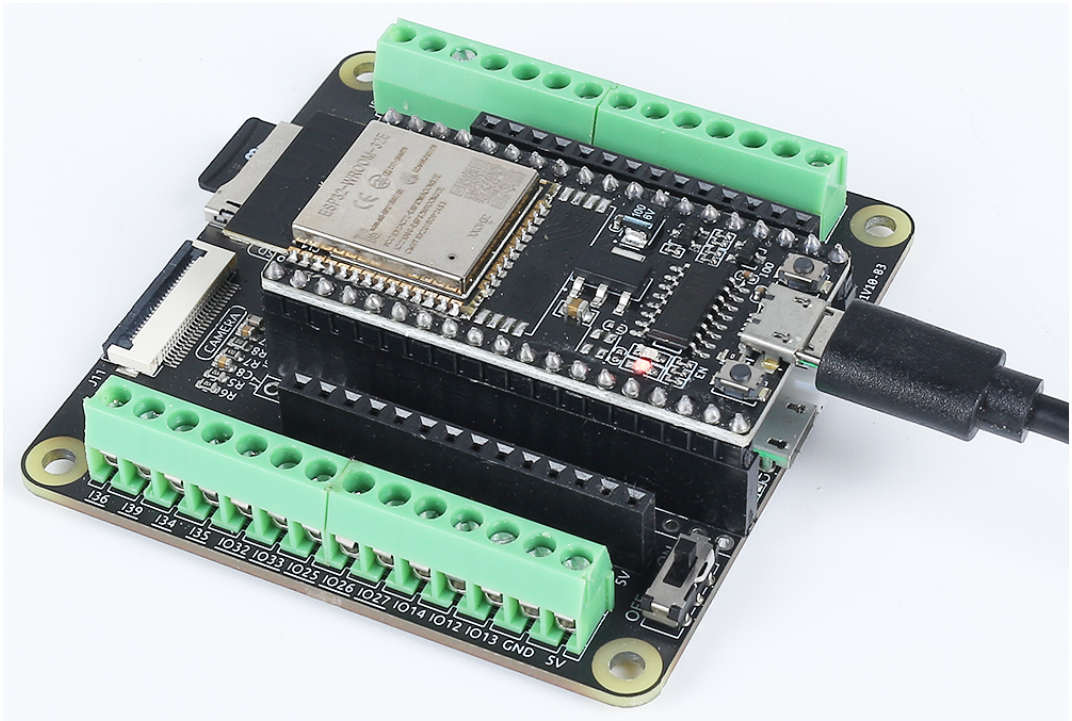
1. Code öffnen.

   Zum Beispiel `1.1_hello_led.py`.

   Wenn Sie darauf doppelklicken, öffnet sich ein neues Fenster auf der rechten Seite. Sie können gleichzeitig mehr als einen Code öffnen.

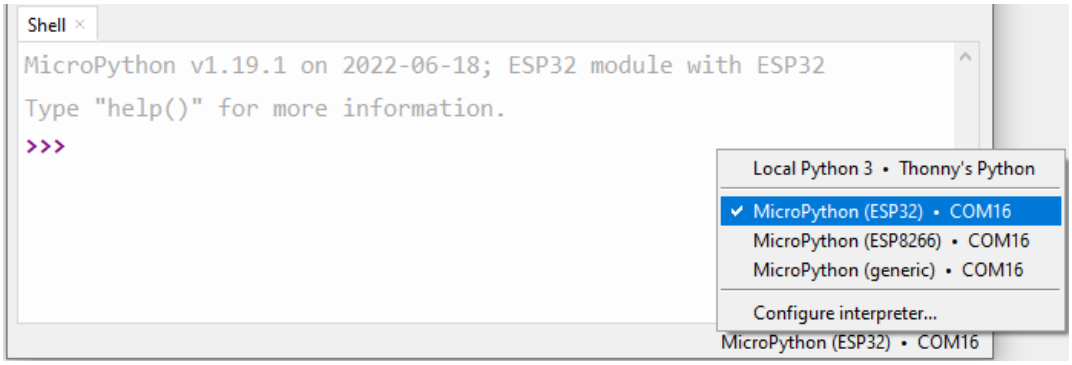


2. Stecken Sie den esp32 mit einem Mikro-USB-Kabel in Ihren Computer.



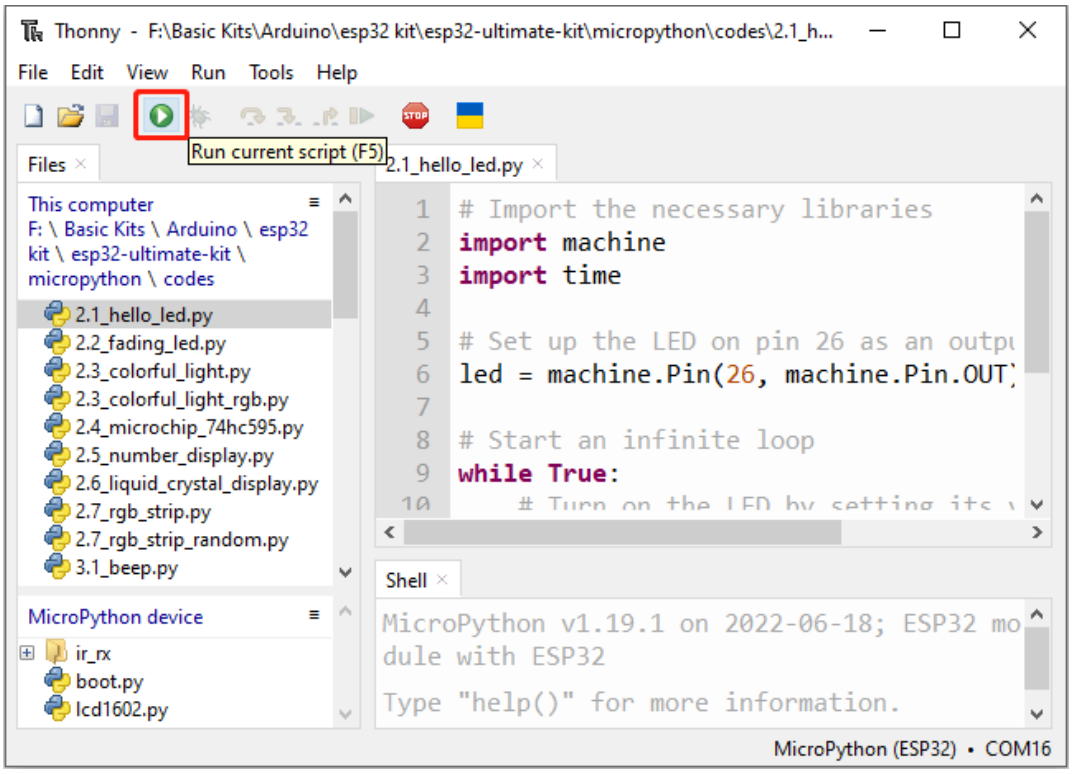
3. Richtigen Interpreter auswählen

   Wählen Sie den Interpreter „MicroPython (ESP32).COMxx“.



4. Den Code ausführen

   Um das Skript auszuführen, klicken Sie auf die Schaltfläche **Run current script** oder drücken Sie F5.



   Wenn der Code Informationen enthält, die gedruckt werden müssen, erscheinen sie in der Shell; ansonsten erscheint nur die folgende Information.

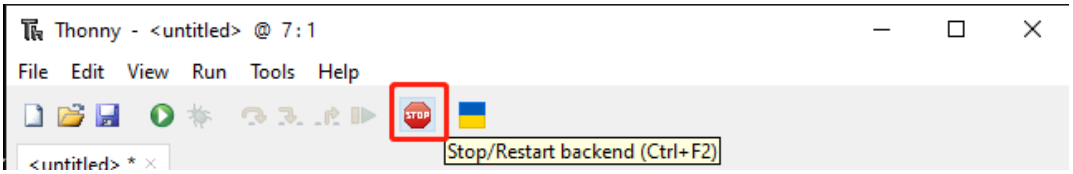
   Klicken Sie auf **View** -> **Edit**, um das Shell-Fenster zu öffnen, wenn es in Ihrem Thonny nicht erscheint.

```
MicroPython v1.19.1 on 2022-06-18; ESP32 module with ESP32

Type "help()" for more information.

>>> %Run -c $EDITOR_CONTENT
```

   - Die erste Zeile zeigt die Version von MicroPython, das Datum und Informationen zu Ihrem Gerät.
   - Die zweite Zeile fordert Sie auf, „help()“ einzugeben, um Hilfe zu erhalten.

- Die dritte Zeile ist ein Befehl von Thonny, der den MicroPython-Interpreter auf Ihrem Pico W anweist, den Inhalt des Skriptbereichs - „EDITOR_CONTENT“ - auszuführen.
- Wenn nach der dritten Zeile eine Nachricht erscheint, handelt es sich normalerweise um eine Nachricht, die Sie MicroPython ausdrucken lassen, oder um eine Fehlermeldung für den Code.

5. Ausführung stoppen

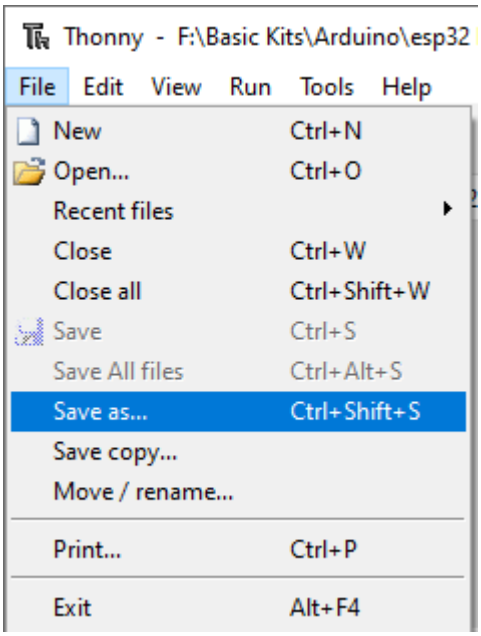


Um den laufenden Code zu stoppen, klicken Sie auf die Schaltfläche **Stop/Restart backend**. Der Befehl **%RUN -c $EDITOR_CONTENT** verschwindet nach dem Stoppen.

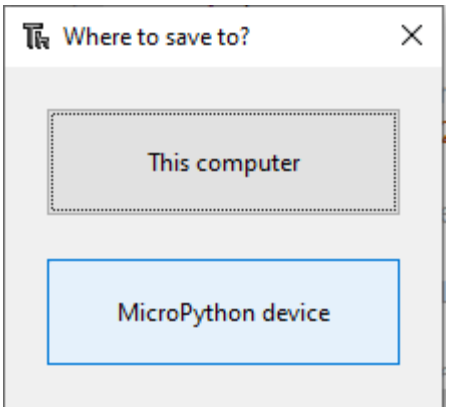
6. Speichern oder Speichern unter

Sie können Änderungen, die Sie am geöffneten Beispiel vorgenommen haben, speichern, indem Sie **Ctrl+S** drücken oder auf die Schaltfläche **Save** in Thonny klicken.

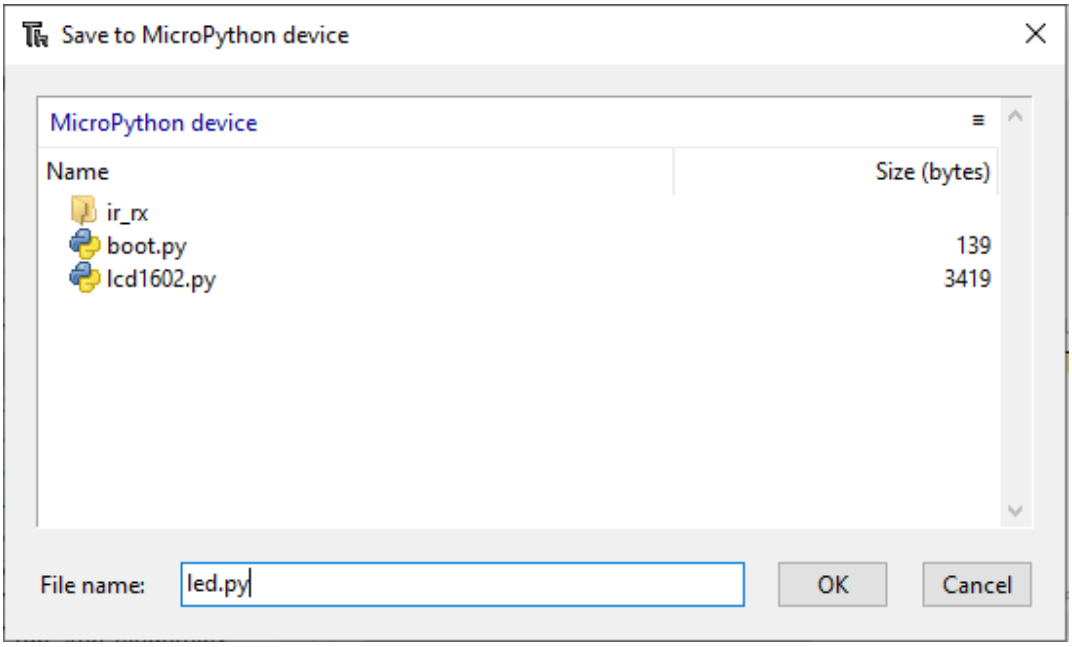
Der Code kann als separate Datei innerhalb des **MicroPython drive(ESP32)** gespeichert werden, indem Sie auf **File** -> **Save As** klicken.



Wählen Sie **MicroPython drive**.



Klicken Sie dann auf **OK**, nachdem Sie den Dateinamen und die Erweiterung **.py** eingegeben haben. Auf dem MicroPython-Laufwerk sehen Sie Ihre gespeicherte Datei.



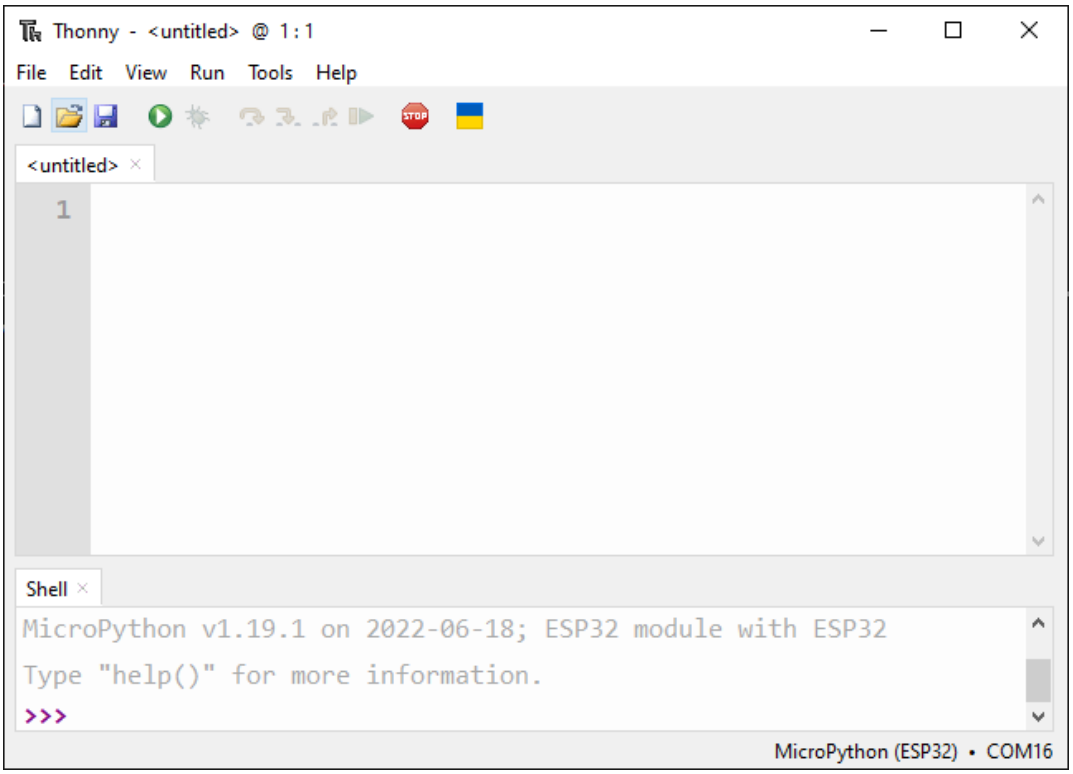
Bemerkung: Unabhängig davon, welchen Namen Sie Ihrem Code geben, ist es am besten, zu beschreiben, um welche Art von Code es sich handelt, und ihm keinen bedeutungslosen Namen wie `abc.py` zu geben. Wenn Sie den Code als `main.py` speichern, wird er automatisch ausgeführt, wenn der Strom eingeschaltet wird.

3.5.2 Datei Erstellen und Ausführen

Der Code wird direkt im Codeabschnitt angezeigt. Sie können ihn in Thonny kopieren und wie folgt ausführen.

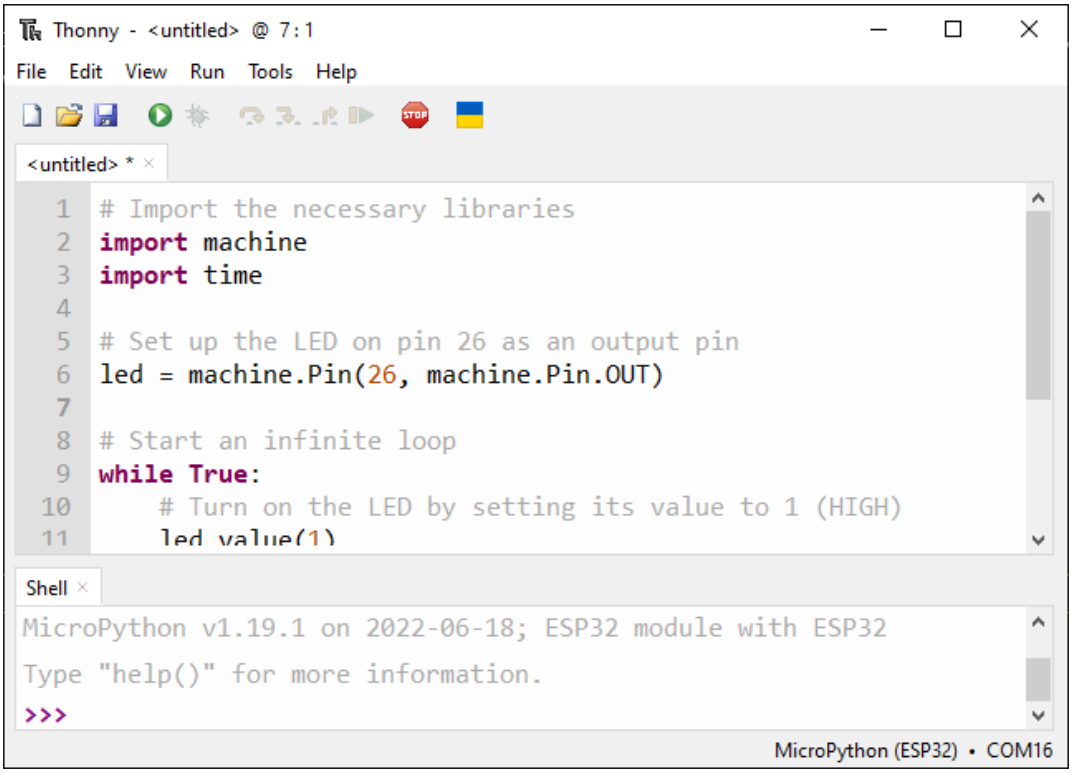
1. Eine neue Datei erstellen

   Öffnen Sie die Thonny IDE, klicken Sie auf die Schaltfläche **New**, um eine neue leere Datei zu erstellen.

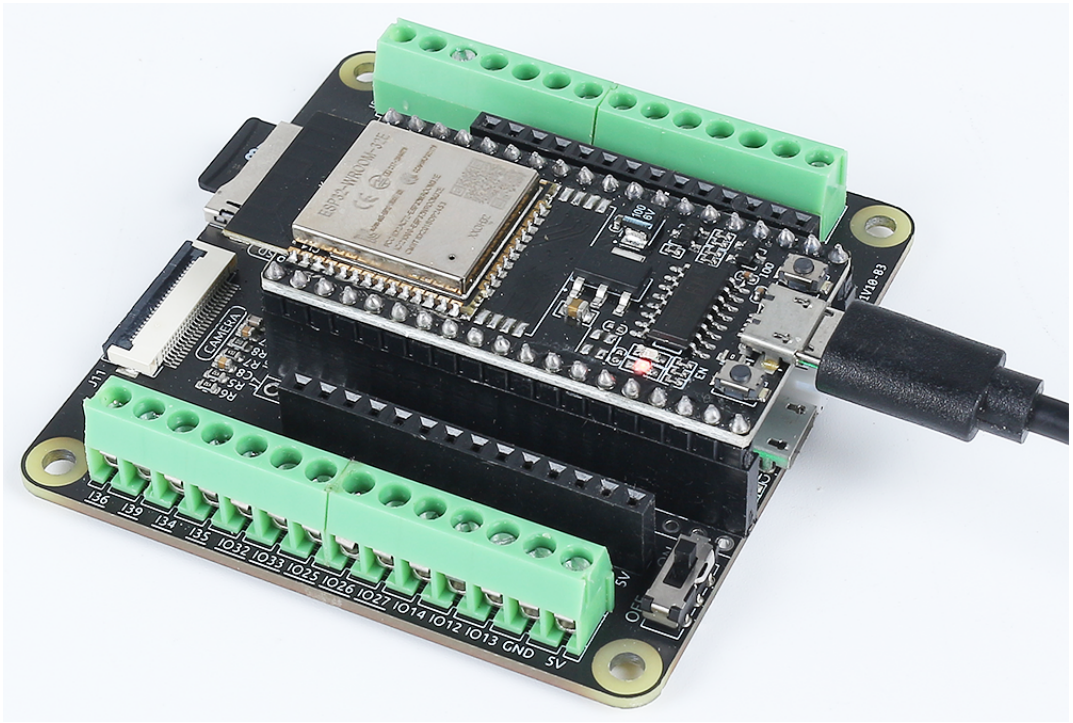


2. Code kopieren

   Kopieren Sie den Code aus dem Projekt in die Thonny IDE.

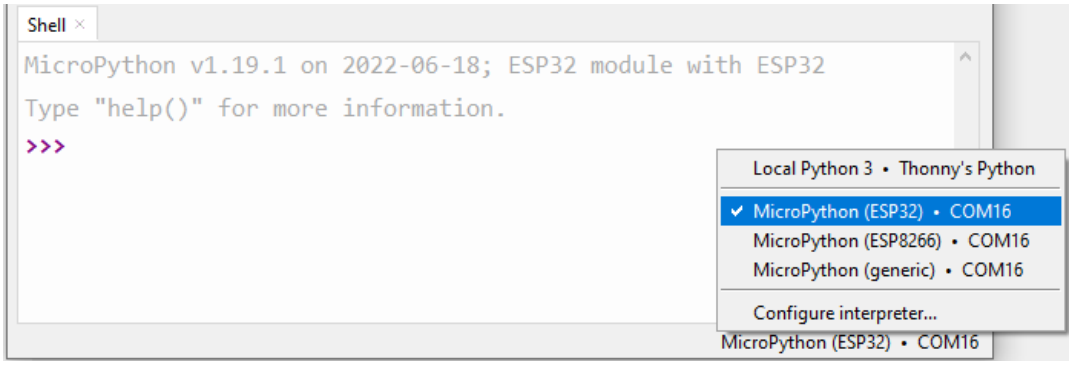


3. Stecken Sie den esp32 mit einem Mikro-USB-Kabel in Ihren Computer.



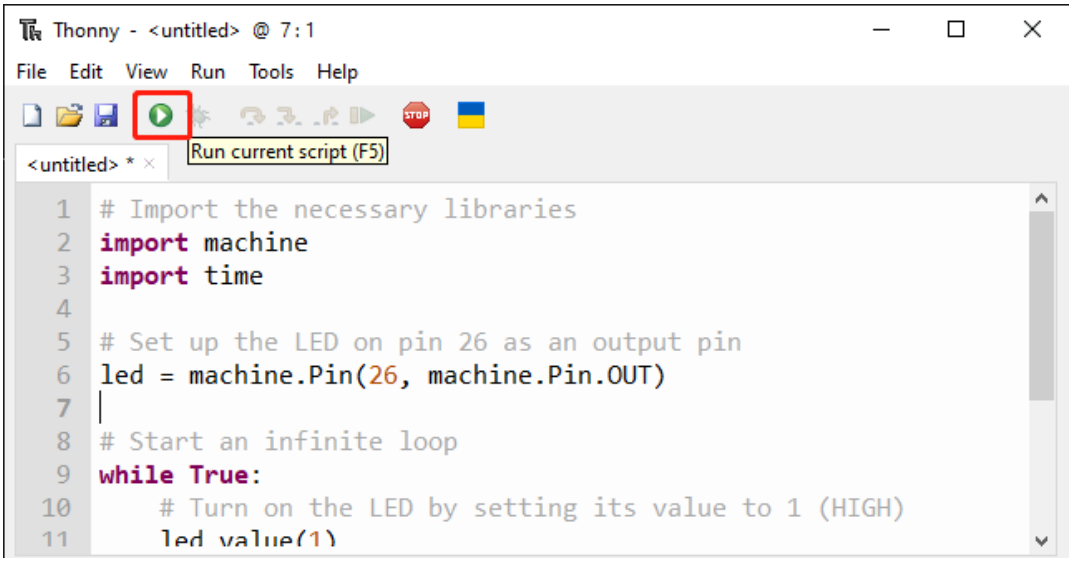
4. Richtigen Interpreter auswählen

   Wählen Sie in der unteren rechten Ecke den Interpreter „MicroPython (ESP32).COMxx“.



5. Den Code ausführen

   Sie müssen auf **Run Current Script** klicken oder einfach „F5“ drücken, um es auszuführen.



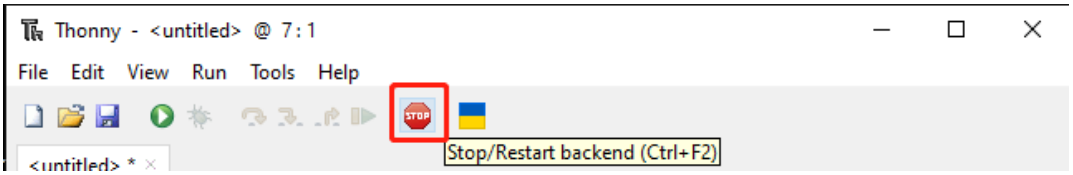
   Wenn der Code Informationen enthält, die gedruckt werden müssen, erscheinen sie in der Shell; ansonsten erscheint nur die folgende Information.

   Klicken Sie auf **View** -> **Edit**, um das Shell-Fenster zu öffnen, wenn es in Ihrem Thonny nicht erscheint.

```
MicroPython v1.19.1 on 2022-06-18; ESP32 module with ESP32

Type "help()" for more information.

>>> %Run -c $EDITOR_CONTENT
```

   - Die erste Zeile zeigt die Version von MicroPython, das Datum und Informationen zu Ihrem Gerät.
   - Die zweite Zeile fordert Sie auf, „help()“ einzugeben, um Hilfe zu erhalten.
   - Die dritte Zeile ist ein Befehl von Thonny, der den MicroPython-Interpreter auf Ihrem Pico W anweist, den Inhalt des Skriptbereichs - „EDITOR_CONTENT“ - auszuführen.
   - Wenn nach der dritten Zeile eine Nachricht erscheint, handelt es sich normalerweise um eine Nachricht, die Sie MicroPython ausdrucken lassen, oder um eine Fehlermeldung für den Code.

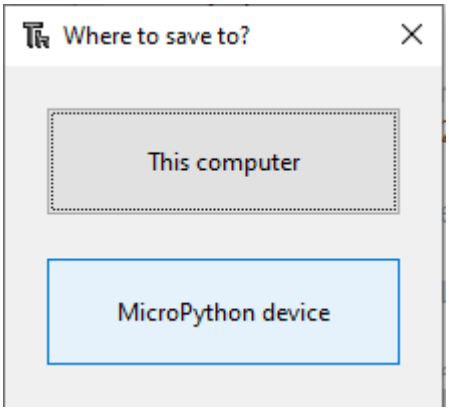
6. Ausführung stoppen



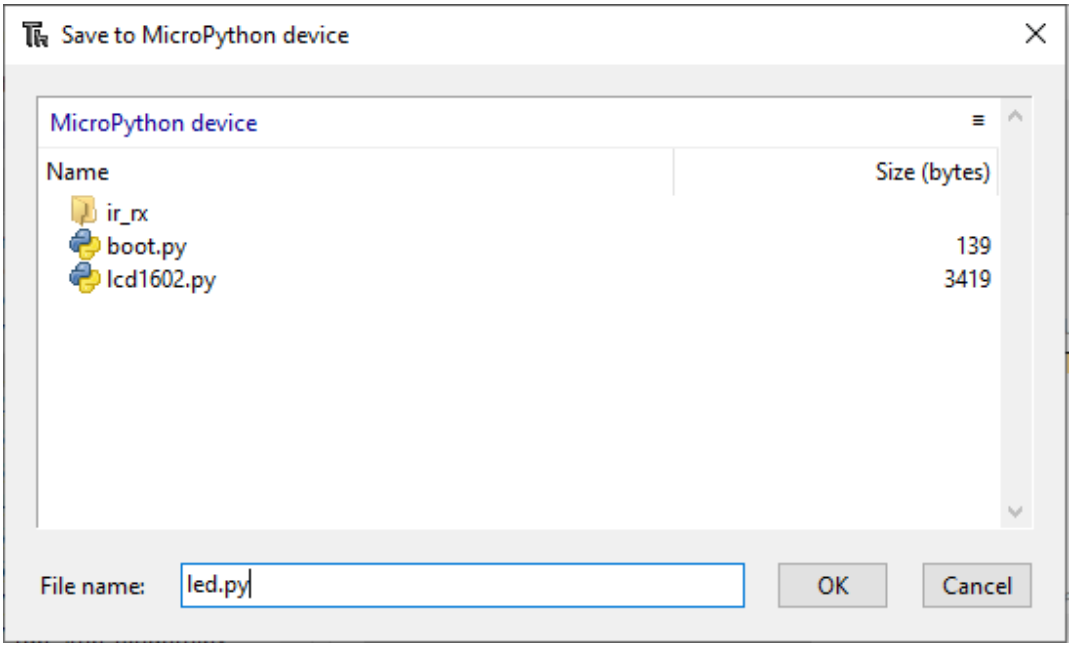
Um den laufenden Code zu stoppen, klicken Sie auf die Schaltfläche **Stop/Restart backend**. Der Befehl **%RUN -c $EDITOR_CONTENT** verschwindet nach dem Stoppen.

7. Speichern oder Speichern unter

   Sie können den Code speichern, indem Sie **Ctrl+S** drücken oder auf die Schaltfläche **Save** in Thonny klicken. Im aufpoppenden Fenster wählen Sie den Ort, an dem Sie die Datei speichern möchten.



Klicken Sie dann nach Eingabe des Dateinamens und der Erweiterung **.py** auf **OK** oder **Save**.



Bemerkung: Unabhängig davon, welchen Namen Sie Ihrem Code geben, ist es am besten, zu beschreiben, um welche Art von Code es sich handelt, und ihm keinen bedeutungslosen Namen wie `abc.py` zu geben. Wenn Sie den Code als `main.py` speichern, wird er automatisch ausgeführt, wenn der Strom eingeschaltet wird.

8. Datei öffnen

   Hier sind zwei Möglichkeiten, eine gespeicherte Code-Datei zu öffnen.

   - Die erste Methode besteht darin, auf das Öffnen-Symbol in der Thonny-Toolbar zu klicken, genau wie beim Speichern eines Programms. Sie werden gefragt, ob Sie es von **this computer** oder **MicroPython device** öffnen möchten, zum Beispiel klicken Sie auf **MicroPython device** und Sie sehen eine Liste aller Programme, die Sie auf dem ESP32 gespeichert haben.
   - Die zweite besteht darin, die Dateivorschau direkt zu öffnen, indem Sie auf **View** -> **Files** klicken und dann auf die entsprechende `.py`-Datei doppelklicken, um sie zu öffnen.

3.6 1.6 (Optional) MicroPython-Grundsyntax

3.6.1 Einrückung

Einrückung bezieht sich auf die Leerzeichen am Anfang einer Codezeile. Wie bei Standard-Python-Programmen laufen MicroPython-Programme normalerweise von oben nach unten: Das Programm durchläuft jede Zeile der Reihe nach, führt sie im Interpreter aus und geht dann zur nächsten Zeile über, genau so, als ob Sie sie Zeile für Zeile in der Shell eintippen. Ein Programm, das einfach die Anweisungsliste Zeile für Zeile durchblättert, ist jedoch nicht sehr intelligent - deshalb hat MicroPython, genau wie Python, seine eigene Methode, um die Ausführungsreihenfolge seines Programms zu steuern: Einrückung.

Sie müssen mindestens ein Leerzeichen vor print() setzen, sonst erscheint eine Fehlermeldung „Ungültige Syntax“. Es wird normalerweise empfohlen, die Leerzeichen durch gleichmäßiges Drücken der Tab-Taste zu standardisieren.

```
if 8 > 5:
print("Eight is greater than Five!")
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 2
SyntaxError: invalid syntax
```

Sie müssen die gleiche Anzahl von Leerzeichen im gleichen Codeblock verwenden, sonst wird Python einen Fehler melden.

```
if 8 > 5:
print("Eight is greater than Five!")
        print("Eight is greater than Five")
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 2
SyntaxError: invalid syntax
```

3.6.2 Kommentare

Die Kommentare im Code helfen uns, den Code zu verstehen, machen den gesamten Code lesbarer und kommentieren während des Testens Teile des Codes aus, so dass dieser Teil des Codes nicht ausgeführt wird.

Einzellinienkommentar

Einzellinienkommentare in MicroPython beginnen mit #, und der nachfolgende Text wird bis zum Ende der Zeile als Kommentar betrachtet. Kommentare können vor oder nach dem Code platziert werden.

```
print("hello world") #This is a annotationhello world
```

```
>>> %Run -c $EDITOR_CONTENT
hello world
```

Kommentare müssen nicht unbedingt Text sein, der den Code erklärt. Sie können auch einen Teil des Codes auskommentieren, um zu verhindern, dass MicroPython den Code ausführt.

```
#print("Can't run it")
print("hello world") #This is a annotationhello world
```

```
>>> %Run -c $EDITOR_CONTENT
hello world
```

Mehrzeiliger Kommentar

Wenn Sie mehrere Zeilen kommentieren möchten, können Sie mehrere # Zeichen verwenden.

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello, World!
```

Oder Sie können mehrzeilige Zeichenketten verwenden, wenn erwartet.

Da MicroPython Zeichenkettenliterale ignoriert, die nicht Variablen zugewiesen werden, können Sie dem Code mehrzeilige Zeichenketten (dreifache Anführungszeichen) hinzufügen und Kommentare darin platzieren:

```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello, World!
```

Solange die Zeichenkette keiner Variablen zugewiesen wird, ignoriert MicroPython sie nach dem Lesen des Codes und behandelt sie, als hätten Sie einen mehrzeiligen Kommentar gemacht.

3.6.3 Print()

Die Funktion `print()` gibt die angegebene Nachricht auf dem Bildschirm oder einem anderen Standardausgabegerät aus. Die Nachricht kann eine Zeichenkette oder ein anderes Objekt sein; das Objekt wird vor dem Schreiben auf dem Bildschirm in eine Zeichenkette umgewandelt.

Mehrere Objekte drucken:

```
print("Welcome!", "Enjoy yourself!")
```

```
>>> %Run -c $EDITOR_CONTENT
Welcome! Enjoy yourself!
```

Tupel drucken:

```
x = ("pear", "apple", "grape")
print(x)
```

```
>>> %Run -c $EDITOR_CONTENT
('pear', 'apple', 'grape')
```

Zwei Nachrichten drucken und das Trennzeichen angeben:

```
print("Hello", "how are you?", sep="---")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello---how are you?
```

3.6.4 Variablen

Variablen sind Behälter, die verwendet werden, um Datenwerte zu speichern.

Das Erstellen einer Variablen ist sehr einfach. Sie müssen ihr nur einen Namen geben und ihr einen Wert zuweisen. Sie müssen den Datentyp der Variablen bei der Zuweisung nicht angeben, denn die Variable ist eine Referenz, und sie greift durch Zuweisung auf Objekte verschiedener Datentypen zu.

Die Benennung von Variablen muss folgenden Regeln folgen:

- Variablennamen dürfen nur Zahlen, Buchstaben und Unterstriche enthalten
- Das erste Zeichen des Variablennamens muss ein Buchstabe oder Unterstrich sein
- Variablennamen sind groß- und kleinschreibungsempfindlich

Variable Erstellen

Es gibt keinen Befehl zum Deklarieren von Variablen in MicroPython. Variablen werden erstellt, wenn Sie ihnen zum ersten Mal einen Wert zuweisen. Es ist keine spezifische Typdeklaration erforderlich, und Sie können sogar den Typ nach dem Setzen der Variablen ändern.

```
x = 8       # x is of type int
x = "lily" # x is now of type str
print(x)
```

```
>>> %Run -c $EDITOR_CONTENT
lily
```

Casting

Wenn Sie den Datentyp für die Variable festlegen möchten, können Sie dies durch Casting tun.

```
x = int(5)    # y will be 5
y = str(5)    # x will be '5'
z = float(5)  # z will be 5.0
print(x,y,z)
```

```
>>> %Run -c $EDITOR_CONTENT
5 5 5.0
```

Den Typ Erhalten

Sie können den Datentyp einer Variablen mit der Funktion *type()* erhalten.

```
x = 5
y = "hello"
z = 5.0
print(type(x),type(y),type(z))
```

```
>>> %Run -c $EDITOR_CONTENT
<class 'int'> <class 'str'> <class 'float'>
```

Einfache oder Doppelte Anführungszeichen?

In MicroPython können einfache oder doppelte Anführungszeichen verwendet werden, um String-Variablen zu definieren.

```
x = "hello"
# is the same as
x = 'hello'
```

Groß- und Kleinschreibung

Variablennamen sind groß- und kleinschreibungsempfindlich.

```
a = 5
A = "lily"
#A will not overwrite a
print(a, A)
```

```
>>> %Run -c $EDITOR_CONTENT
5 lily
```

3.6.5 If Else

Entscheidungsfindung ist erforderlich, wenn wir einen Code nur dann ausführen möchten, wenn eine bestimmte Bedingung erfüllt ist.

if

```
if test expression:
    statement(s)
```

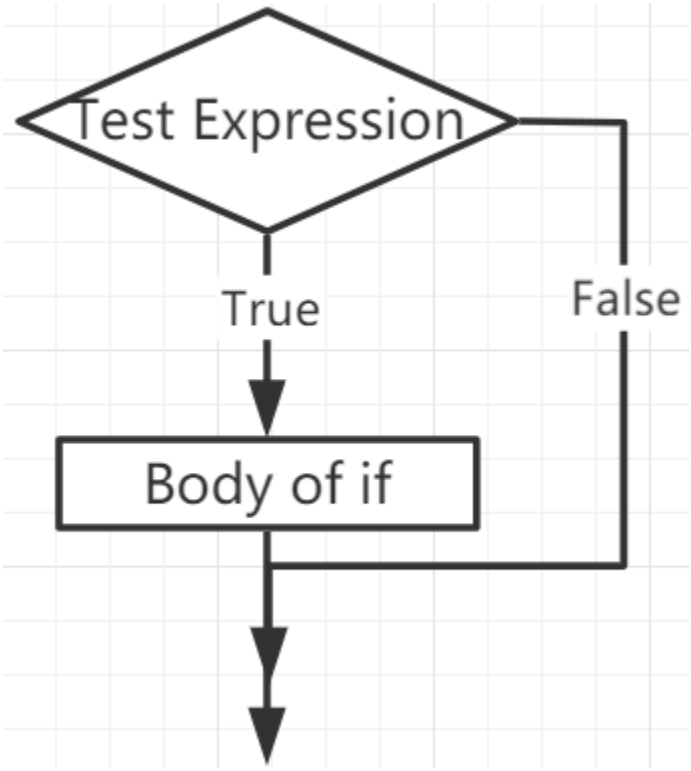
Hier bewertet das Programm den *test expression* und führt die *statement* nur aus, wenn der *test expression* True ist.

Ist der *test expression* False, werden die *statement(s)* nicht ausgeführt.

In MicroPython bedeutet Einrückung den Körper der *if*-Anweisung. Der Körper beginnt mit einer Einrückung und endet mit der ersten nicht eingerückten Zeile.

Python interpretiert Nicht-Null-Werte als „True“. None und 0 werden als „False“ interpretiert.

Flussdiagramm des if-Statements



Beispiel

```
num = 8
if num > 0:
    print(num, "is a positive number.")
print("End with this line")
```

```
>>> %Run -c $EDITOR_CONTENT
8 is a positive number.
End with this line
```

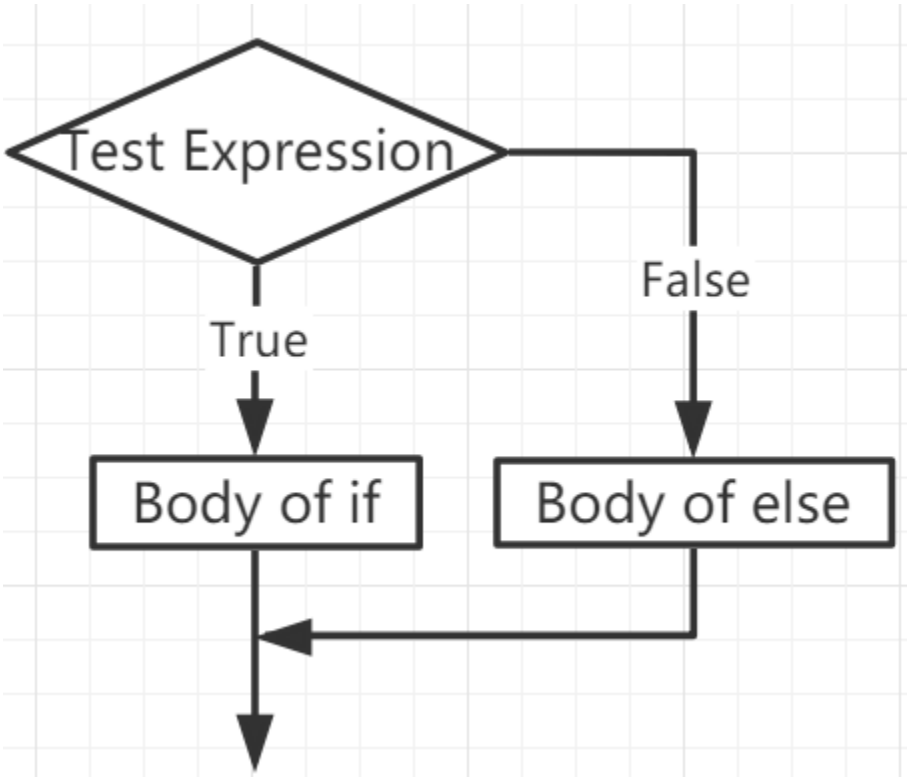
if...else

```
if test expression:
    Body of if
else:
    Body of else
```

Die *if..else*-Anweisung bewertet *test expression* und führt den Körper von *if* nur aus, wenn die Testbedingung *True* ist.

Ist die Bedingung *False*, wird der Körper von *else* ausgeführt. Einrückungen werden verwendet, um die Blöcke zu trennen.

Flussdiagramm des if...else-Statements



Beispiel

```
num = -8
if num > 0:
    print(num, "is a positive number.")
else:
    print(num, "is a negative number.")
```

```
>>> %Run -c $EDITOR_CONTENT
-8 is a negative number.
```

if...elif...else

```
if test expression:
    Body of if
elif test expression:
    Body of elif
else:
    Body of else
```

Elif steht für *else if*. Es ermöglicht uns, mehrere Ausdrücke zu überprüfen.

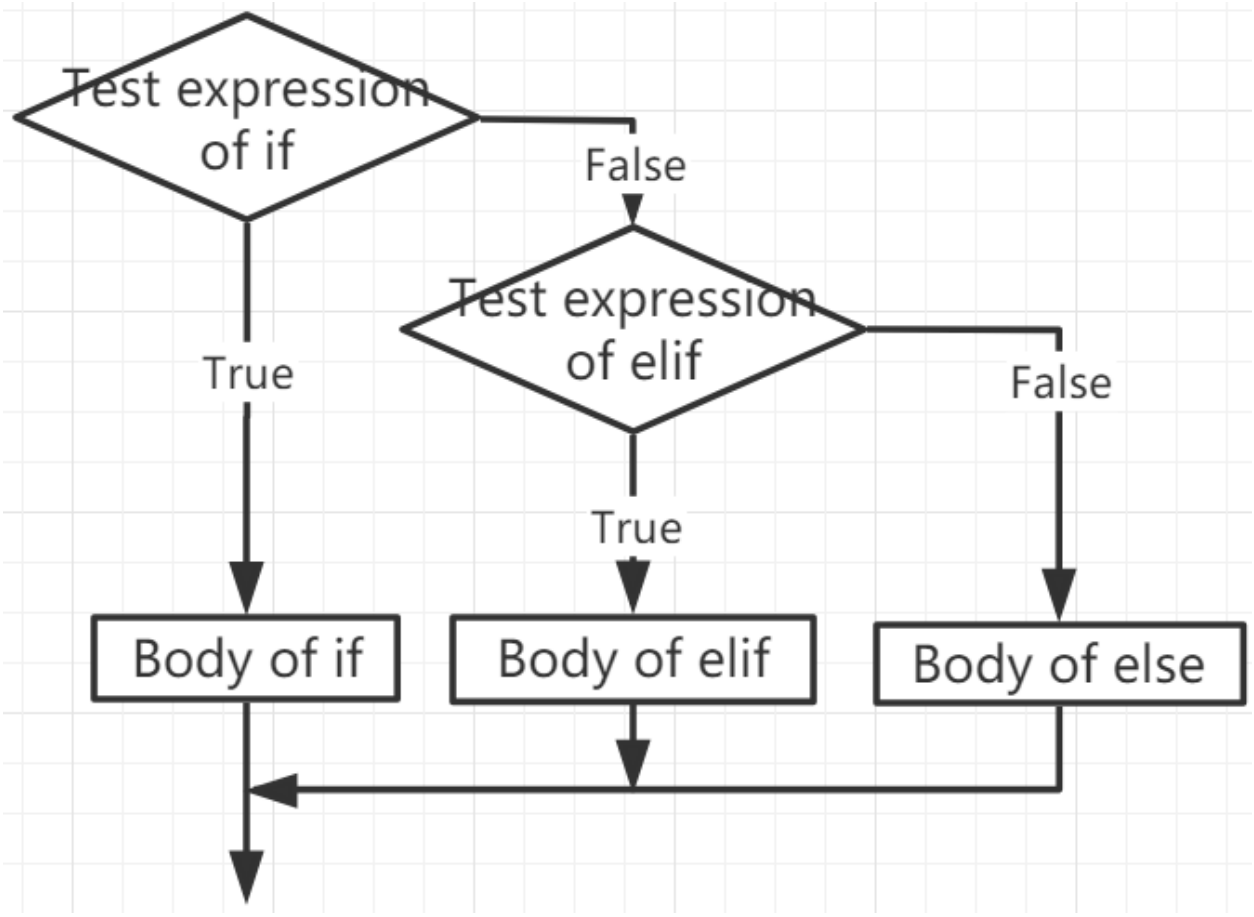
Ist die Bedingung des *if* False, wird die Bedingung des nächsten elif-Blocks überprüft, und so weiter.

Sind alle Bedingungen *False*, wird der Körper von *else* ausgeführt.

Nur einer von mehreren *if*...*elif*...*else*-Blöcken wird entsprechend den Bedingungen ausgeführt.

Der *if*-Block kann nur einen *else*-Block haben. Aber er kann mehrere *elif*-Blöcke haben.

Flussdiagramm des if...elif...else-Statements



Beispiel

```
x = 10
y = 9

if x > y:
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    print("x is greater than y")
elif x == y:
    print("x and y are equal")
else:
    print("x is greater than y")
```

```
>>> %Run -c $EDITOR_CONTENT
x is greater than y
```

Verschachteltes if

Wir können ein if-Statement in ein anderes if-Statement einbetten, das dann als verschachteltes if-Statement bezeichnet wird.

Beispiel

```
x = 67

if x > 10:
    print("Above ten,")
    if x > 20:
        print("and also above 20!")
    else:
        print("but not above 20.")
```

```
>>> %Run -c $EDITOR_CONTENT
Above ten,
and also above 20!
```

3.6.6 While-Schleifen

Das `while`-Statement wird verwendet, um ein Programm in einer Schleife auszuführen, d.h., es ermöglicht die wiederholte Ausführung einer Aufgabe unter bestimmten Bedingungen.

Seine grundlegende Form lautet:

```
while test expression:
    Body of while
```

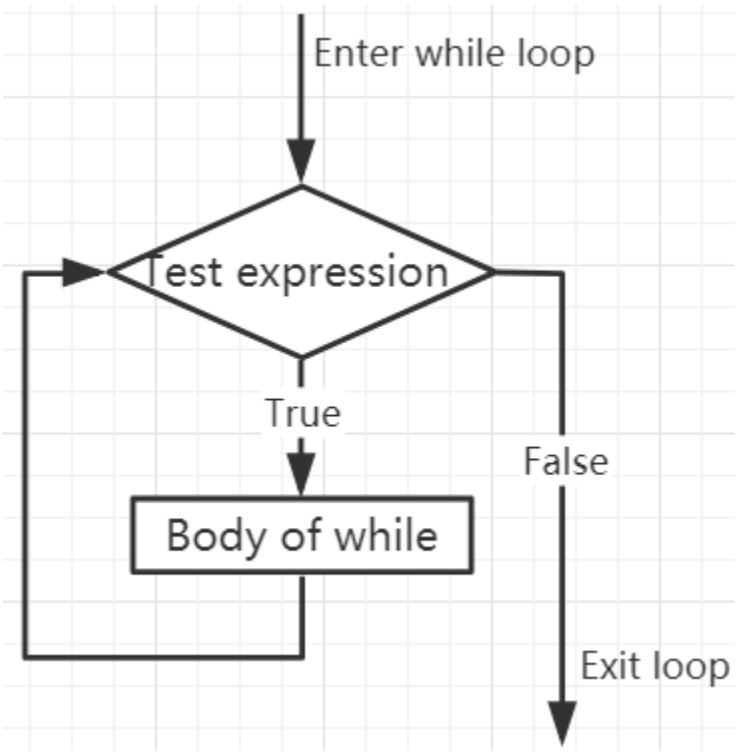
In der `while`-Schleife wird zuerst der `test expression` geprüft. Nur wenn der `test expression` als `True` bewertet wird, tritt man in den Körper der Schleife ein. Nach einer Iteration wird der `test expression` erneut geprüft. Dieser Prozess setzt sich fort, bis der `test expression` als `False` bewertet wird.

In MicroPython wird der Körper der `while`-Schleife durch Einrückung bestimmt.

Der Körper beginnt mit einer Einrückung und endet mit der ersten nicht eingerückten Zeile.

Python interpretiert jeden Nicht-Null-Wert als `True`. None und 0 werden als `False` interpretiert.

Flussdiagramm der while-Schleife




```
x = 10

while x > 0:
    print(x)
    x -= 1
```

```
>>> %Run -c $EDITOR_CONTENT
10
9
8
7
6
5
4
3
2
1
```

Break-Statement

Mit dem Break-Statement können wir die Schleife stoppen, auch wenn die Bedingung der while-Schleife wahr ist:

```
x = 10

while x > 0:
    print(x)
    if x == 6:
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        break
    x -= 1
```

```
>>> %Run -c $EDITOR_CONTENT
10
9
8
7
6
```

While-Schleife mit Else

Ähnlich wie die *if*-Schleife kann auch die *while*-Schleife einen optionalen *else*-Block haben.

Wenn die Bedingung in der *while*-Schleife als *False* bewertet wird, wird der *else*-Teil ausgeführt.

```
x = 10

while x > 0:
    print(x)
    x -= 1
else:
    print("Game Over")
```

```
>>> %Run -c $EDITOR_CONTENT
10
9
8
7
6
5
4
3
2
1
Game Over
```

3.6.7 For-Schleifen

Die *for*-Schleife kann jede Art von Elementsequenz durchlaufen, wie beispielsweise eine Liste oder einen String.

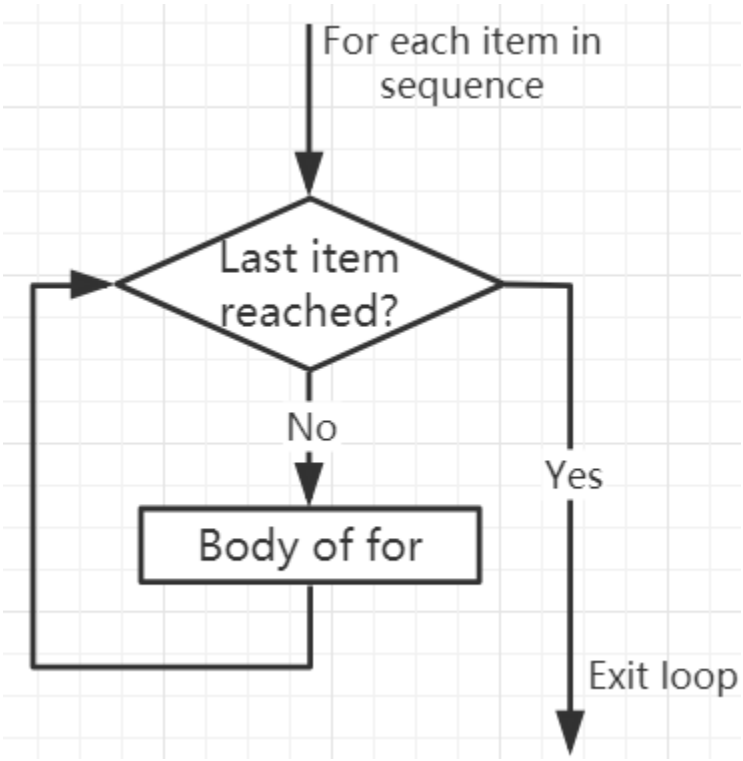
Die Syntax der For-Schleife ist wie folgt:

```
for val in sequence:
    Body of for
```

Hier ist *val* eine Variable, die in jeder Iteration den Wert des Elements in der Sequenz annimmt.

Die Schleife setzt sich fort, bis wir das letzte Element in der Sequenz erreicht haben. Um den Körper der *for*-Schleife vom restlichen Code abzugrenzen, verwenden wir Einrückungen.

Flussdiagramm der for-Schleife




```
numbers = [1, 2, 3, 4]
sum = 0

for val in numbers:
    sum = sum+val

print("The sum is", sum)
```

```
>>> %Run -c $EDITOR_CONTENT
The sum is 10
```

Das break-Statement

Mit dem break-Statement können wir die Schleife stoppen, bevor sie alle Elemente durchlaufen hat:

```
numbers = [1, 2, 3, 4]
sum = 0

for val in numbers:
    sum = sum+val
    if sum == 6:
        break
print("The sum is", sum)
```

```
>>> %Run -c $EDITOR_CONTENT
The sum is 6
```

Das continue-Statement

Mit dem *continue*-Statement können wir die aktuelle Iteration der Schleife beenden und mit der nächsten fortfahren:

```
numbers = [1, 2, 3, 4]

for val in numbers:
    if val == 3:
        continue
    print(val)
```

```
>>> %Run -c $EDITOR_CONTENT
1
2
4
```

Die range()-Funktion

Wir können die range()-Funktion verwenden, um eine Zahlenfolge zu generieren. range(6) erzeugt Zahlen zwischen 0 und 5 (6 Zahlen).

Wir können auch Start, Stopp und Schrittgröße als range(start, stop, step_size) definieren. Wird nichts angegeben, ist die Standard-Schrittgröße 1.

Im Sinne von range ist das Objekt „lazy“, da es bei der Erstellung nicht jede Zahl generiert, die es „enthält“. Es ist jedoch kein Iterator, da es Operationen wie in, len und __getitem__ unterstützt.

Diese Funktion speichert nicht alle Werte im Speicher; das wäre ineffizient. Sie merkt sich Start, Stopp und Schrittgröße und generiert während der Iteration die nächste Zahl.

Um diese Funktion zu zwingen, alle Elemente auszugeben, können wir die Funktion list() verwenden.

```
print(range(6))

print(list(range(6)))

print(list(range(2, 6)))

print(list(range(2, 10, 2)))
```

```
>>> %Run -c $EDITOR_CONTENT
range(0, 6)
[0, 1, 2, 3, 4, 5]
[2, 3, 4, 5]
[2, 4, 6, 8]
```

Wir können *range()* in einer *for*-Schleife verwenden, um über eine Zahlenfolge zu iterieren. Es kann mit der len()-Funktion kombiniert werden, um den Index zum Durchlaufen der Sequenz zu verwenden.

```
fruits = ['pear', 'apple', 'grape']

for i in range(len(fruits)):
    print("I like", fruits[i])
```

```
>>> %Run -c $EDITOR_CONTENT
I like pear
I like apple
I like grape
```

Else in For-Schleife

Die *for*-Schleife kann auch einen optionalen *else*-Block haben. Wenn die Elemente in der Sequenz, die für die Schleife verwendet werden, erschöpft sind, wird der *else*-Teil ausgeführt.

Das Schlüsselwort *break* kann verwendet werden, um die *for*-Schleife zu stoppen. In diesem Fall wird der *else*-Teil ignoriert.

Wenn also keine Unterbrechung auftritt, wird der *else*-Teil der *for*-Schleife ausgeführt.

```
for val in range(5):
    print(val)
else:
    print("Finished")
```

```
>>> %Run -c $EDITOR_CONTENT
0
1
2
3
4
Finished
```

Der else-Block wird NICHT ausgeführt, wenn die Schleife durch eine break-Anweisung gestoppt wird.

```
for val in range(5):
    if val == 2: break
    print(val)
else:
    print("Finished")
```

```
>>> %Run -c $EDITOR_CONTENT
0
1
```

3.6.8 Funktionen

In MicroPython ist eine Funktion eine Gruppe von zusammenhängenden Anweisungen, die eine spezifische Aufgabe ausführen.

Funktionen helfen dabei, unser Programm in kleinere, modulare Blöcke zu unterteilen. Je größer unser Projekt wird, desto organisierter und handhabbarer machen es Funktionen.

Außerdem vermeiden sie Duplikation und machen den Code wiederverwendbar.

Erstellen einer Funktion

```
def function_nameparameters)
    """docstring"""
    statement(s)
```

- Eine Funktion wird mit dem Schlüsselwort `def` definiert.
- Ein Funktionsname zur eindeutigen Identifikation der Funktion. Die Benennung von Funktionen entspricht der von Variablen und folgt folgenden Regeln:
  - Darf nur Zahlen, Buchstaben und Unterstriche enthalten.
  - Das erste Zeichen muss ein Buchstabe oder ein Unterstrich sein.
  - Groß- und Kleinschreibung wird unterschieden.
- Parameter (Argumente), durch die wir Werte an eine Funktion übergeben. Sie sind optional.
- Der Doppelpunkt (:) markiert das Ende des Funktionskopfs.
- Ein optionaler Docstring, der verwendet wird, um die Funktion der Funktion zu beschreiben. Wir verwenden normalerweise dreifache Anführungszeichen, damit der Docstring über mehrere Zeilen erweitert werden kann.
- Eine oder mehrere gültige MicroPython-Anweisungen, die den Funktionskörper bilden. Die Anweisungen müssen das gleiche Einrückungsniveau haben (üblicherweise 4 Leerzeichen).
- Jede Funktion benötigt mindestens eine Anweisung, aber wenn es aus irgendeinem Grund eine Funktion gibt, die keine Anweisung enthält, sollte ein pass-Statement eingefügt werden, um Fehler zu vermeiden.
- Ein optionales `return`-Statement, um einen Wert aus der Funktion zurückzugeben.

Aufrufen einer Funktion

Um eine Funktion aufzurufen, fügen Sie Klammern nach dem Funktionsnamen hinzu.

```
def my_function():
    print("Your first function")

my_function()
```

```
>>> %Run -c $EDITOR_CONTENT
Your first function
```

Das return-Statement

Das return-Statement wird verwendet, um eine Funktion zu beenden und an den Ort zurückzukehren, an dem sie aufgerufen wurde.

Syntax von return

```
return [expression_list]
```

Das Statement kann einen Ausdruck enthalten, der ausgewertet wird und einen Wert zurückgibt. Wenn kein Ausdruck im Statement vorhanden ist oder das `return`-Statement selbst in der Funktion nicht existiert, gibt die Funktion ein `None`-Objekt zurück.

```
def my_function():
        print("Your first function")

print(my_function())
```

```
>>> %Run -c $EDITOR_CONTENT
Your first function
None
```

Hier ist None der Rückgabewert, da das return-Statement nicht verwendet wird.

Argumente

Informationen können als Argumente an die Funktion übergeben werden.

Spezifizieren Sie Argumente in Klammern nach dem Funktionsnamen. Sie können so viele Argumente hinzufügen, wie Sie benötigen, trennen Sie sie einfach durch Kommata.

```
def welcome(name, msg):
    """This is a welcome function for
    the person with the provided message"""
    print("Hello", name + ', ' + msg)

welcome("Lily", "Welcome to China!")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello Lily, Welcome to China!
```

Anzahl der Argumente

Standardmäßig muss eine Funktion mit der korrekten Anzahl von Argumenten aufgerufen werden. Das bedeutet, dass, wenn Ihre Funktion 2 Parameter erwartet, Sie die Funktion auch mit 2 Argumenten aufrufen müssen, nicht mehr und nicht weniger.

```
def welcome(name, msg):
    """This is a welcome function for
    the person with the provided message"""
    print("Hello", name + ', ' + msg)

welcome("Lily", "Welcome to China!")
```

Hier hat die Funktion willkommen() 2 Parameter.

Da wir diese Funktion mit zwei Argumenten aufgerufen haben, läuft sie reibungslos und ohne Fehler.

Wird sie mit einer anderen Anzahl von Argumenten aufgerufen, zeigt der Interpreter eine Fehlermeldung an.

Folgendes ist der Aufruf dieser Funktion mit einem und keinem Argument sowie deren jeweilige Fehlermeldungen.

```
welcome("Lily")Only one argument
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 6, in <module>
TypeError: function takes 2 positional arguments but 1 were given
```

```
welcome()No arguments
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 6, in <module>
TypeError: function takes 2 positional arguments but 0 were given
```

Standardargumente

In MicroPython können wir den Zuweisungsoperator (=) verwenden, um einen Standardwert für den Parameter anzugeben.

Wenn wir die Funktion ohne Argument aufrufen, verwendet sie den Standardwert.

```
def welcome(name, msg = "Welcome to China!"):
    """This is a welcome function for
    the person with the provided message"""
    print("Hello", name + ', ' + msg)
welcome("Lily")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello Lily, Welcome to China!
```

In dieser Funktion hat der Parameter `name` keinen Standardwert und ist beim Aufruf erforderlich (obligatorisch).

Andererseits ist der Standardwert des Parameters `msg` „Willkommen in China!“. Daher ist er beim Aufruf optional. Wird ein Wert bereitgestellt, überschreibt er den Standardwert.

Jedes Argument in der Funktion kann einen Standardwert haben. Sobald jedoch ein Standardargument vorhanden ist, müssen auch alle Argumente rechts davon Standardwerte haben.

Das bedeutet, dass nicht-standardmäßige Argumente nicht auf Standardargumente folgen können.

Zum Beispiel, wenn wir den obigen Funktionskopf wie folgt definieren:

```
def welcome(name = "Lily", msg):
```

Erhalten wir folgende Fehlermeldung:

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
SyntaxError: non-default argument follows default argument
```

Schlüsselwortargumente

Wenn wir eine Funktion mit bestimmten Werten aufrufen, werden diese Werte basierend auf ihrer Position den Argumenten zugewiesen.

Zum Beispiel, in der oben genannten Funktion willkommen(), als wir sie mit willkommen(„Lily“, „Willkommen in China“) aufgerufen haben, wird der Wert „Lily“ dem `name` und entsprechend „Willkommen in China“ dem Parameter `msg` zugewiesen.

MicroPython erlaubt das Aufrufen von Funktionen mit Schlüsselwortargumenten. Wenn wir die Funktion auf diese Weise aufrufen, kann die Reihenfolge (Position) der Argumente geändert werden.

```
# keyword arguments
welcome(name = "Lily",msg = "Welcome to China!")

# keyword arguments (out of order)
welcome(msg = "Welcome to China",name = "Lily")

#1 positional, 1 keyword argument
welcome("Lily", msg = "Welcome to China!")
```

Wie wir sehen können, können wir positionelle Argumente und Schlüsselwortargumente bei Funktionsaufrufen mischen. Aber wir müssen uns daran erinnern, dass die Schlüsselwortargumente nach den positionellen Argumenten kommen müssen.

Ein positionelles Argument nach einem Schlüsselwortargument führt zu einem Fehler.

Zum Beispiel, wenn der Funktionsaufruf wie folgt ist:

```
welcome(name="Lily","Welcome to China!")
```

Führt dies zu einem Fehler:

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 5, in <module>
SyntaxError: non-keyword arg after keyword arg
```

Beliebige Argumente

Manchmal, wenn Sie nicht wissen, wie viele Argumente an die Funktion übergeben werden.

In der Funktionsdefinition können wir einen Stern (*) vor dem Parameternamen hinzufügen.

```
def welcome(*names):
    """This function welcomes all the person
    in the name tuple"""
    #names is a tuple with arguments
    for name in names:
        print("Welcome to China!", name)

welcome("Lily","John","Wendy")
```

```
>>> %Run -c $EDITOR_CONTENT
Welcome to China! Lily
Welcome to China! John
Welcome to China! Wendy
```

Hier haben wir die Funktion mit mehreren Argumenten aufgerufen. Diese Argumente werden in ein Tupel gepackt, bevor sie an die Funktion übergeben werden.

Innerhalb der Funktion verwenden wir eine Schleife, um alle Argumente abzurufen.

Rekursion

In Python wissen wir, dass eine Funktion andere Funktionen aufrufen kann. Es ist sogar möglich, dass die Funktion sich selbst aufruft. Solche Konstrukte werden als rekursive Funktionen bezeichnet.

Dies hat den Vorteil, dass man Daten durchlaufen kann, um ein Ergebnis zu erreichen.

Entwickler sollten bei der Rekursion sehr vorsichtig sein, da es leicht passieren kann, dass man eine Funktion schreibt, die niemals endet, oder eine, die übermäßig viel Speicher oder Prozessorleistung verbraucht. Wenn sie jedoch korrekt geschrieben ist, kann Rekursion eine sehr effiziente und mathematisch-elegante Herangehensweise an die Programmierung sein.

```
def rec_func(i):
    if(i > 0):
        result = i + rec_func(i - 1)
        print(result)
    else:
        result = 0
    return result

rec_func(6)
```

```
>>> %Run -c $EDITOR_CONTENT
1
3
6
10
15
21
```

In diesem Beispiel ist rek_funktion() eine Funktion, die wir definiert haben, um sich selbst aufzurufen („Rekursion“). Wir verwenden die Variable `i` als Daten, und sie wird bei jeder Rekursion um eins verringert (-1). Wenn die Bedingung nicht größer als 0 ist (also 0), endet die Rekursion.

Für neue Entwickler kann es einige Zeit dauern, um zu verstehen, wie es funktioniert, und der beste Weg, es zu testen, ist es zu testen und zu modifizieren.

Vorteile der Rekursion

- Rekursive Funktionen machen den Code sauber und elegant.
- Eine komplexe Aufgabe kann mit Rekursion in einfachere Teilprobleme zerlegt werden.
- Die Generierung von Sequenzen ist mit Rekursion einfacher als mit verschachtelten Iterationen.

Nachteile der Rekursion

- Manchmal ist die Logik hinter der Rekursion schwer nachzuvollziehen.

- Rekursive Aufrufe sind teuer (ineffizient), da sie viel Speicher und Zeit in Anspruch nehmen.
- Rekursive Funktionen sind schwer zu debuggen.

3.6.9 Datentypen

Eingebaute Datentypen

MicroPython hat die folgenden Datentypen:

- Texttyp: str
- Numerische Typen: int, float, complex
- Sequenztypen: list, tuple, range
- Zuordnungstyp: dict
- Mengentypen: set, frozenset
- Boolescher Typ: bool
- Binäre Typen: bytes, bytearray, memoryview

Ermitteln des Datentyps

Sie können den Datentyp eines beliebigen Objekts mit der Funktion *type()* ermitteln:

```
a = 6.8
print(type(a))
```

```
>>> %Run -c $EDITOR_CONTENT
<class 'float'>
```

Festlegen des Datentyps

MicroPython benötigt kein spezifisches Festlegen des Datentyps, dieser wird bestimmt, wenn Sie einer Variablen einen Wert zuweisen.

```
x = "welcome"
y = 45
z = ["apple", "banana", "cherry"]

print(type(x))
print(type(y))
print(type(z))
```

```
>>> %Run -c $EDITOR_CONTENT
<class 'str'>
<class 'int'>
<class 'list'>
>>>
```

Festlegen des spezifischen Datentyps

Wenn Sie den Datentyp spezifizieren möchten, können Sie die folgenden Konstruktorfunktionen verwenden:

Beispiel	Datentyp
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = str(„Hello World“)	str
x = list((„apple“, „banana“, „cherry“))	list
x = tuple((„apple“, „banana“, „cherry“))	tuple
x = range(6)	range
x = dict(name=“John“, age=36)	dict
x = set((„apple“, „banana“, „cherry“))	set
x = frozenset((„apple“, „banana“, „cherry“))	frozenset
x = bool(5)	bool
x = bytes(5)	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

Einige davon können Sie ausdrucken, um das Ergebnis zu sehen.

```
a = float(20.5)
b = list(("apple", "banana", "cherry"))
c = bool(5)

print(a)
print(b)
print(c)
```

```
>>> %Run -c $EDITOR_CONTENT
20.5
['apple', 'banana', 'cherry']
True
>>>
```

Typumwandlung

Sie können mit den Methoden int(), float() und complex() von einem Typ in einen anderen konvertieren: Casting in Python erfolgt daher mit Konstruktorfunktionen:

- int() - konstruiert eine ganze Zahl aus einer Ganzzahl-, Fließkommazahl- oder Stringliteral (indem alle Dezimalstellen entfernt werden)
- float() - konstruiert eine Fließkommazahl aus einer Ganzzahl-, Fließkommazahl- oder Stringliteral (vorausgesetzt, der String stellt eine Fließkommazahl oder eine Ganzzahl dar)
- str() - konstruiert einen String aus einer Vielzahl von Datentypen, einschließlich Strings, Ganzzahl- und Fließkommazahlen

```
a = float("5")
b = int(3.7)
c = str(6.0)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
print(a)
print(b)
print(c)
```

Hinweis: Sie können komplexe Zahlen nicht in einen anderen Zahlentyp konvertieren.

3.6.10 Operatoren

Operatoren werden verwendet, um Operationen mit Variablen und Werten durchzuführen.

- *Arithmetische Operatoren*
- *Zuweisungsoperatoren*
- *Vergleichsoperatoren*
- *Logische Operatoren*
- *Identitätsoperatoren*
- *Mitgliedschaftsoperatoren*
- *Bitweise Operatoren*

Arithmetische Operatoren

Sie können arithmetische Operatoren verwenden, um einige gängige mathematische Operationen durchzuführen.

Operator	Name
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	Modulo
**	Potenzierung
//	Ganzzahlige Division

```
x = 5
y = 3

a = x + y
b = x - y
c = x * y
d = x / y
e = x % y
f = x ** y
g = x // y

print(a)
print(b)
print(c)
print(d)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
print(e)
print(f)
print(g)
```

```
>>> %Run -c $EDITOR_CONTENT
8
2
15
1.66667
2
125
1
8
2
15
>>>
```

Zuweisungsoperatoren

Zuweisungsoperatoren können verwendet werden, um Werte an Variablen zuzuweisen.

Operator	Example	Same As
=	a = 6	a =6
+=	a += 6	a = a + 6
-=	a -= 6	a = a - 6
*=	a *= 6	a = a * 6
/=	a /= 6	a = a / 6
%=	a %= 6	a = a % 6
**=	a **= 6	a = a ** 6
//=	a //= 6	a = a // 6
&=	a &= 6	a = a & 6
|=	a |= 6	a = a | 6
^=	a ^= 6	a = a ^ 6
>>=	a >>= 6	a = a >> 6
<<=	a <<= 6	a = a << 6

```
a = 6

a *= 6
print(a)
```

```
>>> %Run test.py
36
>>>
```

Vergleichsoperatoren

Vergleichsoperatoren werden verwendet, um zwei Werte zu vergleichen.

Operator	Name
==	Gleich
!=	Ungleich
<	Kleiner als
>	Größer als
>=	Größer als oder gleich
<=	Kleiner als oder gleich

```
a = 6
b = 8

print(a>b)
```

```
>>> %Run test.py
False
>>>
```

Gibt **False** zurück, weil **a** kleiner als **b** ist.

Logische Operatoren

Logische Operatoren werden verwendet, um bedingte Aussagen zu kombinieren.

Operator	Beschreibung
and	Gibt True zurück, wenn beide Aussagen wahr sind
or	Gibt True zurück, wenn eine der Aussagen wahr ist
not	Kehrt das Ergebnis um, gibt False zurück, wenn das Ergebnis wahr ist

```
a = 6
print(a > 2 and a < 8)
```

```
>>> %Run -c $EDITOR_CONTENT
True
>>>
```

Identitätsoperatoren

Identitätsoperatoren werden verwendet, um zu vergleichen, ob Objekte gleich sind, nicht ob sie gleich sind, sondern ob sie tatsächlich dasselbe Objekt mit demselben Speicherort sind.

Operator	Beschreibung
is	Gibt True zurück, wenn beide Variablen dasselbe Objekt sind
is not	Gibt True zurück, wenn beide Variablen nicht dasselbe Objekt sind

```
a = ["hello", "welcome"]
b = ["hello", "welcome"]
c = a

print(a is c)
# returns True because z is the same object as x

print(a is b)
# returns False because x is not the same object as y, even if they have the same content

print(a == b)
# returns True because x is equal to y
```

```
>>> %Run -c $EDITOR_CONTENT
True
False
True
>>>
```

Mitgliedschaftsoperatoren

Mitgliedschaftsoperatoren werden verwendet, um zu testen, ob eine Sequenz in einem Objekt vorhanden ist.

Operator	Beschreibung
in	Gibt True zurück, wenn eine Sequenz mit dem angegebenen Wert im Objekt vorhanden ist
not in	Gibt True zurück, wenn eine Sequenz mit dem angegebenen Wert nicht im Objekt vorhanden ist

```
a = ["hello", "welcome", "Goodmorning"]

print("welcome" in a)
```

```
>>> %Run -c $EDITOR_CONTENT
True
>>>
```

Bitweise Operatoren

Bitweise Operatoren werden verwendet, um (binäre) Zahlen zu vergleichen.

Operator	Name	Beschreibung
&	AND	Setzt jedes Bit auf 1, wenn beide Bits 1 sind
|	OR	Setzt jedes Bit auf 1, wenn eines von zwei Bits 1 ist
^	XOR	Setzt jedes Bit auf 1, wenn nur eines von zwei Bits 1 ist
~	NOT	Invertiert alle Bits
<<	Zero-Fill-Linksverschiebung	Verschiebt nach links, indem Nullen von rechts eingeschoben werden und die linken Bits herausfallen
>>	Signierte Rechtsverschiebung	Verschiebt nach rechts, indem Kopien des linken Bits von links eingeschoben werden und die rechten Bits herausfallen

```
num = 2

print(num & 1)
print(num | 1)
print(num << 1)
```

```
>>> %Run -c $EDITOR_CONTENT
0
3
4
>>>
```

3.6.11 Listen

Listen werden verwendet, um mehrere Elemente in einer einzigen Variablen zu speichern, und werden mit eckigen Klammern erstellt:

```
B_list = ["Blossom", "Bubbles","Buttercup"]
print(B_list)
```

Listenelemente sind veränderbar, geordnet und erlauben doppelte Werte. Die Listenelemente sind indiziert, wobei das erste Element den Index [0], das zweite Element den Index [1] usw. hat.

```
C_list = ["Red", "Blue", "Green", "Blue"]
print(C_list)            # duplicate
print(C_list[0])
print(C_list[1])         # ordered
C_list[2] = "Purple"     # changeable
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Red', 'Blue', 'Green', 'Blue']
Red
Blue
['Red', 'Blue', 'Purple', 'Blue']
```

Eine Liste kann verschiedene Datentypen enthalten:

```
A_list = ["Banana", 255, False, 3.14]
print(A_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Banana', 255, False, 3.14]
```

Listenlänge

Um festzustellen, wie viele Elemente in der Liste sind, verwenden Sie die Funktion len().

```
A_list = ["Banana", 255, False, 3.14]
print(len(A_list))
```

```
>>> %Run -c $EDITOR_CONTENT
4
```

Listenelemente prüfen

Drucken Sie das zweite Element der Liste aus:

```
A_list = ["Banana", 255, False, 3.14]
print(A_list[1])
```

```
>>> %Run -c $EDITOR_CONTENT
[255]
```

Drucken Sie das letzte Element der Liste aus:

```
A_list = ["Banana", 255, False, 3.14]
print(A_list[-1])
```

```
>>> %Run -c $EDITOR_CONTENT
[3.14]
```

Drucken Sie das zweite und dritte Element aus:

```
A_list = ["Banana", 255, False, 3.14]
print(A_list[1:3])
```

```
>>> %Run -c $EDITOR_CONTENT
[255, False]
```

Listenelemente ändern

Ändern Sie das zweite und dritte Element:

```
A_list = ["Banana", 255, False, 3.14]
A_list[1:3] = [True,"Orange"]
print(A_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Banana', True, 'Orange', 3.14]
```

Ändern Sie das zweite Element, indem Sie es durch zwei Werte ersetzen:

```
A_list = ["Banana", 255, False, 3.14]
A_list[1:2] = [True,"Orange"]
print(A_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Banana', True, 'Orange', False, 3.14]
```

Listenelemente hinzufügen

Verwendung der Methode append(), um ein Element hinzuzufügen:

```
C_list = ["Red", "Blue", "Green"]
C_list.append("Orange")
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Red', 'Blue', 'Green', 'Orange']
```

Fügen Sie ein Element an der zweiten Position ein:

```
C_list = ["Red", "Blue", "Green"]
C_list.insert(1, "Orange")
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Red', 'Orange', 'Blue', 'Green']
```

Listenelemente entfernen

Die Methode remove() entfernt das angegebene Element.

```
C_list = ["Red", "Blue", "Green"]
C_list.remove("Blue")
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Red', 'Green']
```

Die Methode pop() entfernt den angegebenen Index. Wenn Sie den Index nicht angeben, entfernt die Methode pop() das letzte Element.

```
A_list = ["Banana", 255, False, 3.14, True,"Orange"]
A_list.pop(1)
print(A_list)
A_list.pop()
print(A_list)
```

```
>>> %Run -c $EDITOR_CONTENT
255
['Banana', False, 3.14, True, 'Orange']
'Orange'
['Banana', False, 3.14, True]
```

Das Schlüsselwort `del` entfernt ebenfalls den angegebenen Index:

```
C_list = ["Red", "Blue", "Green"]
del C_list[1]
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
['Red', 'Green']
```

Die Methode clear() leert die Liste. Die Liste bleibt bestehen, hat aber keinen Inhalt.

```
C_list = ["Red", "Blue", "Green"]
C_list.clear()
print(C_list)
```

```
>>> %Run -c $EDITOR_CONTENT
[]
```

2. Displays

3.7 2.1 Hallo, LED!

Genau wie das Ausdrucken von „Hallo, Welt!“ der erste Schritt beim Erlernen der Programmierung ist, so ist die Verwendung eines Programms zum Ansteuern einer LED die traditionelle Einführung in das Erlernen der physischen Programmierung.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die unten stehenden Links kaufen.

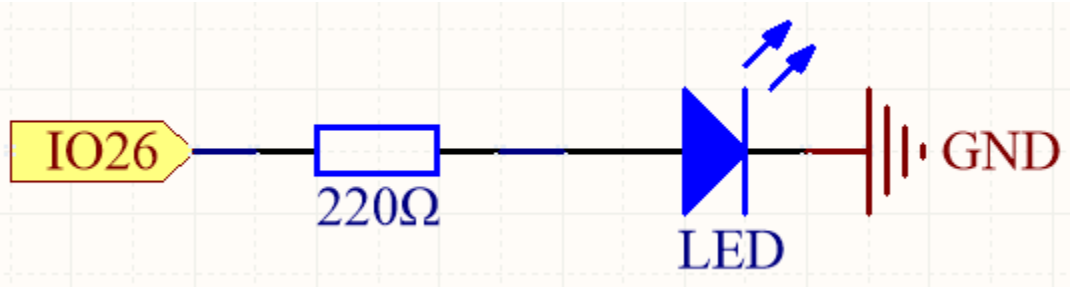
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

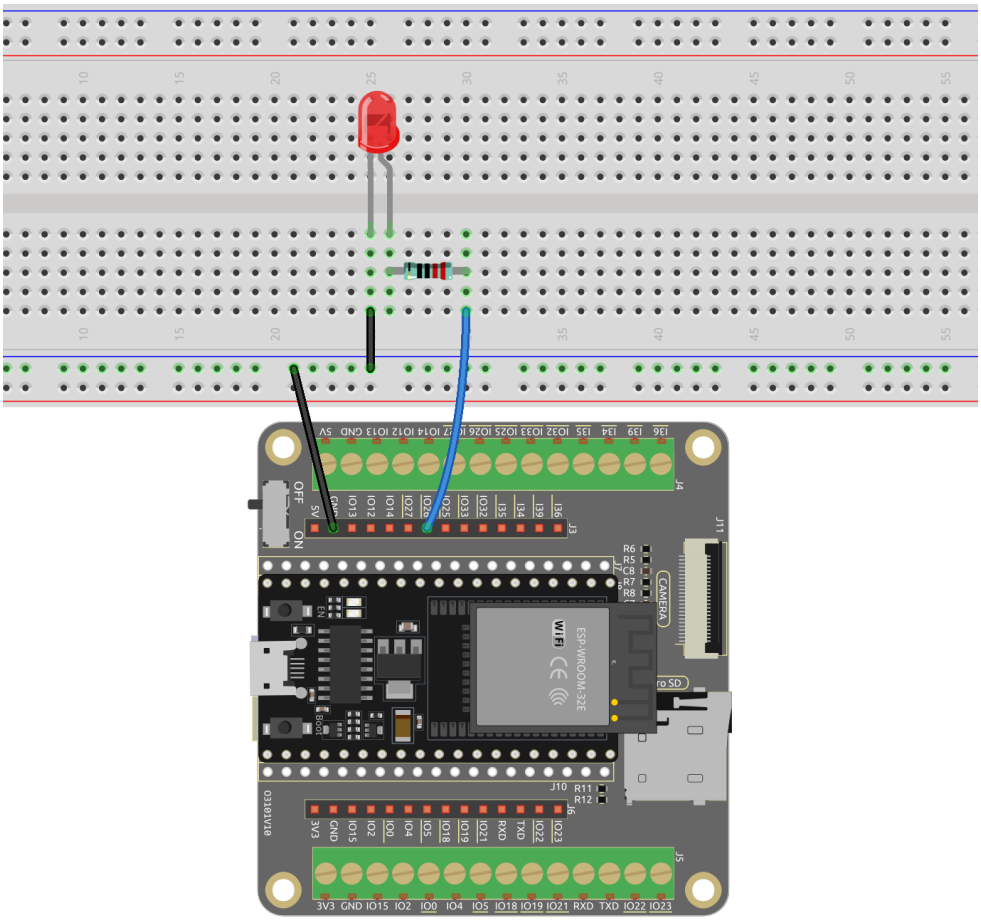
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Dieser Schaltkreis funktioniert nach einem einfachen Prinzip, und die Stromrichtung ist in der Abbildung dargestellt. Die LED leuchtet auf, nachdem der 220-Ohm-Strombegrenzungswiderstand eingeschaltet wurde, wenn Pin26 High-Level ausgibt. Die LED schaltet sich aus, wenn Pin26 Low-Level ausgibt.

Verdrahtung

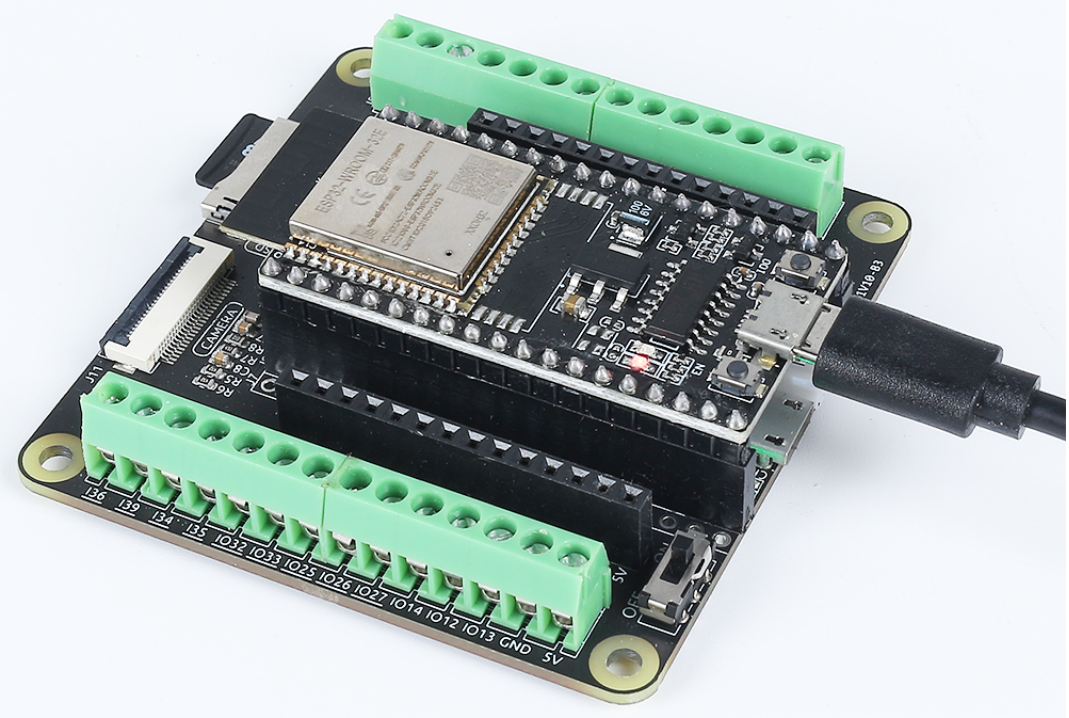


Führen Sie den Code aus

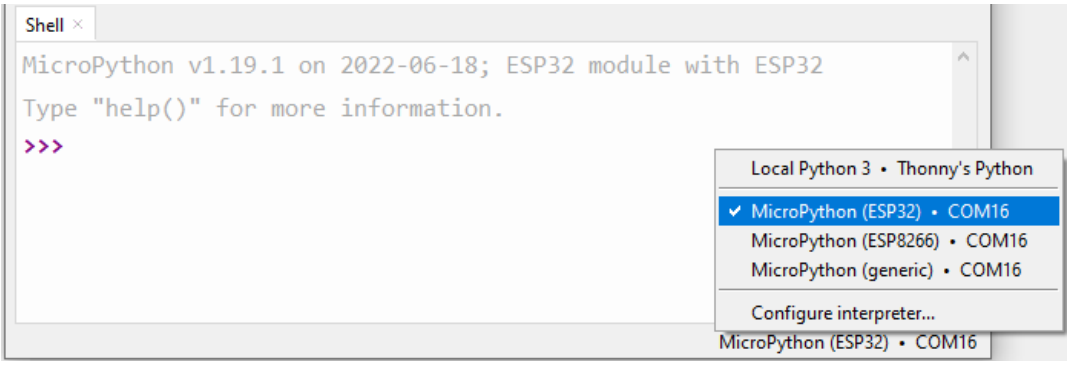
1. Öffnen Sie die Datei `2.1_hello_led.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein.

```
# Import the necessary libraries
import machine
import time

# Set up the LED on pin 26 as an output pin
led = machine.Pin(26, machine.Pin.OUT)

# Start an infinite loop
while True:
    # Turn on the LED by setting its value to 1 (HIGH)
    led.value(1)
    # Wait for 1 second (1000 milliseconds) while the LED is on
    time.sleep(1)

    # Turn off the LED by setting its value to 0 (LOW)
    led.value(0)
    # Wait for 0.5 seconds (500 milliseconds) while the LED is off
    time.sleep(0.5)
```

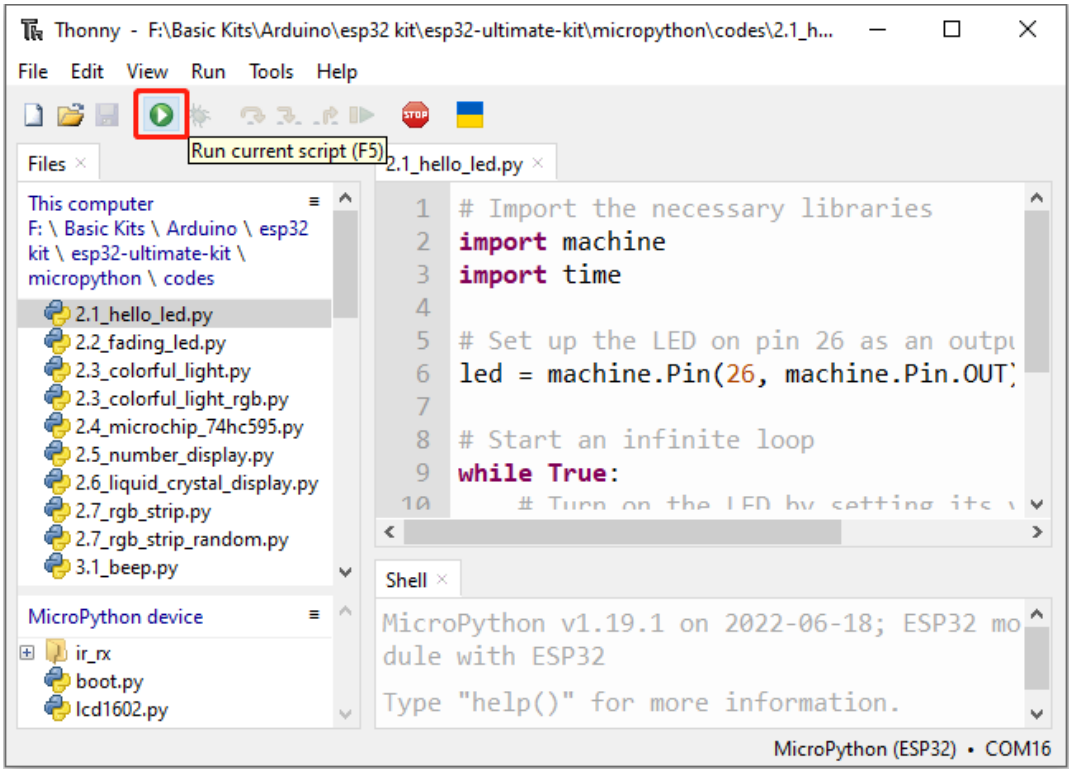
2. Verbinden Sie das ESP32 WROOM 32E mit Ihrem Computer über ein Micro-USB-Kabel.



3. Klicken Sie dann in der unteren rechten Ecke auf den Interpreter „MicroPython (ESP32).COMXX“.



4. Klicken Sie schließlich auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.



5. Nachdem der Code ausgeführt wurde, sehen Sie das Blinken der LED.

Wie funktioniert das?

1. Es importiert zwei Module, `machine` und `time`. Das Modul `machine` bietet Zugriff auf die Hardware des Mikrocontrollers auf niedriger Ebene, während das Modul `time` Funktionen für zeitbezogene Operationen bereitstellt.

```
import machine
import time
```

2. Dann wird Pin26 als Ausgangspin mit der Funktion `machine.Pin()` und dem Argument `machine.Pin.OUT` eingerichtet.

```
led = machine.Pin(26, machine.Pin.OUT)
```

3. In der Schleife `While True` wird die LED für eine Sekunde eingeschaltet, indem der Wert des Pin26 auf 1 gesetzt wird (`led.value(1)`), dann wird er auf 0 gesetzt (`led.value(0)`), um sie für eine Sekunde auszuschalten, und so weiter in einer unendlichen Schleife.

```
while True:
    # Turn on the LED by setting its value to 1 (HIGH)
    led.value(1)
    # Wait for 1 second (1000 milliseconds) while the LED is on
    time.sleep(1)

    # Turn off the LED by setting its value to 0 (LOW)
    led.value(0)
    # Wait for 0.5 seconds (500 milliseconds) while the LED is off
    time.sleep(0.5)
```

Mehr erfahren

In diesem Projekt haben wir die Module `machine` und `time` von MicroPython verwendet, wir können hier mehr Möglichkeiten finden, sie zu verwenden.

- machine.Pin
- time

3.8 2.2 Abblendende LED

Im vorherigen Projekt steuerten wir die LED, indem wir sie durch digitalen Ausgang ein- und ausschalteten. In diesem Projekt werden wir einen Atmungseffekt an der LED erzeugen, indem wir die Pulsweitenmodulation (PWM) nutzen. PWM ist eine Technik, die es uns ermöglicht, die Helligkeit einer LED oder die Geschwindigkeit eines Motors zu steuern, indem wir den Tastgrad eines Rechtecksignals variieren.

Mit PWM wird die LED nicht einfach ein- oder ausgeschaltet, sondern wir passen die Zeit an, in der die LED eingeschaltet ist, im Vergleich zur Zeit, in der sie ausgeschaltet ist, innerhalb jedes Zyklus an. Durch schnelles Ein- und Ausschalten der LED in unterschiedlichen Intervallen können wir die Illusion erzeugen, dass die LED allmählich heller und dunkler wird, was einen Atmungseffekt simuliert.

Durch die Nutzung der PWM-Fähigkeiten des ESP32 WROOM 32E können wir eine reibungslose und präzise Steuerung der LED-Helligkeit erreichen. Dieser Atmungseffekt fügt Ihren Projekten ein dynamisches und visuell ansprechendes Element hinzu und schafft ein auffälliges Display oder Ambiente.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die unten stehenden Links kaufen.

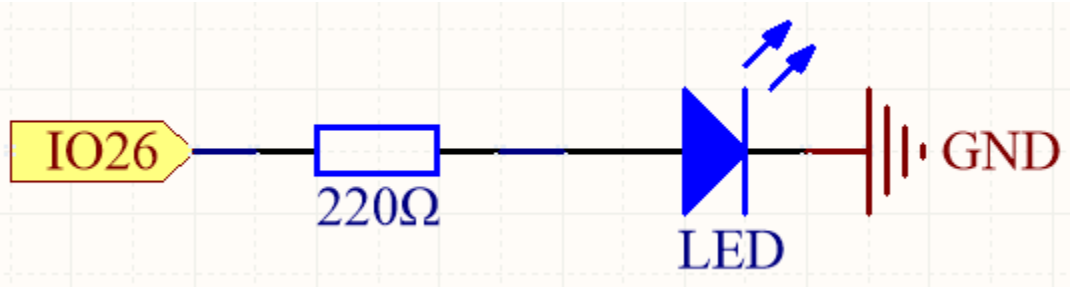
KOMPONENTENVORSTELLUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

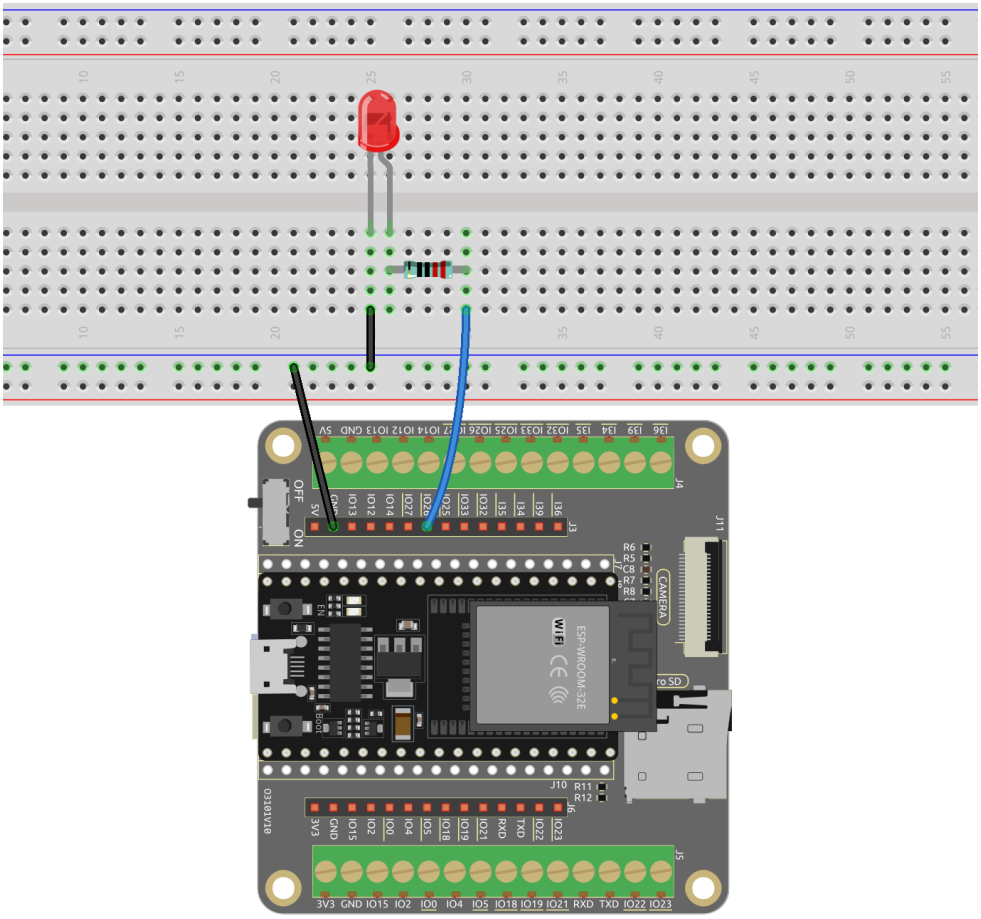
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Dieses Projekt ist der gleiche Schaltkreis wie das erste Projekt *2.1 Hallo, LED!*, aber der Signaltyp ist unterschiedlich. Im ersten Projekt wird digitaler High- und Low-Level (0&1) direkt von Pin26 ausgegeben, um die LED aufleuchten oder ausschalten zu lassen, dieses Projekt gibt ein PWM-Signal von Pin26 aus, um die Helligkeit der LED zu steuern.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `2.2_fading_led.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.

- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
# Import the necessary libraries
from machine import Pin, PWM
import time

# Create a PWM object
led = PWM(Pin(26), freq=1000)

while True:
    # Gradually increase brightness
    for duty_cycle in range(0, 1024, 1):
        led.duty(duty_cycle)
        time.sleep(0.01)

    # Gradually decrease brightness
    for duty_cycle in range(1023, -1, -1):
        led.duty(duty_cycle)
        time.sleep(0.01)
```

Die LED wird allmählich heller, während der Code läuft.

Wie funktioniert das?

Insgesamt demonstriert dieser Code, wie man PWM-Signale verwendet, um die Helligkeit einer LED zu steuern.

1. Es importiert zwei Module, `machine` und `time`. Das Modul `machine` bietet Zugriff auf die Hardware des Mikrocontrollers auf niedriger Ebene, während das Modul `time` Funktionen für zeitbezogene Operationen bereitstellt.

   ```
   import machine
   import time
   ```

2. Dann wird ein `PWM`-Objekt zur Steuerung der an Pin 26 angeschlossenen LED initialisiert und die Frequenz des PWM-Signals auf 1000 Hz eingestellt.

   ```
   led = PWM(Pin(26), freq=1000)
   ```

3. Die LED wird mit einer Schleife ein- und ausgeblendet: Die äußere `while True`-Schleife läuft unendlich. Zwei verschachtelte `for`-Schleifen werden verwendet, um die Helligkeit der LED allmählich zu erhöhen und zu verringern. Der Tastgrad reicht von 0 bis 1023, was einen Tastgrad von 0% bis 100% darstellt.

   ```
   # Import the necessary libraries
   from machine import Pin, PWM
   import time

   # Create a PWM object
   led = PWM(Pin(26), freq=1000)

   while True:
       # Gradually increase brightness
       for duty_cycle in range(0, 1024, 2):
           led.duty(duty_cycle)
           time.sleep(0.01)
   ```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
# Gradually decrease brightness
for duty_cycle in range(1023, -1, -2):
    led.duty(duty_cycle)
    time.sleep(0.01)
```

- `range()`: Erstellt eine Folge von ganzen Zahlen von 0 bis 1023.
- Der Tastgrad des PWM-Signals wird für jeden Wert in der Sequenz mit der `duty()`-Methode des `PWM`-Objekts eingestellt.
- `time.sleep()`: Pausiert die Ausführung des Programms für 10 Millisekunden zwischen jeder Iteration der Schleife und erzeugt so eine allmähliche Erhöhung der Helligkeit über die Zeit.

3.9 2.3 Farbiges Licht

In diesem Projekt werden wir in die faszinierende Welt der additiven Farbmischung mit einer RGB-LED eintauchen.

Eine RGB-LED kombiniert die drei Grundfarben Rot, Grün und Blau in einem einzigen Gehäuse. Diese drei LEDs teilen sich eine gemeinsame Kathoden-Pin, während jeder Anoden-Pin die Intensität der entsprechenden Farbe steuert.

Durch Variieren der elektrischen Signalintensität, die an jede Anode angelegt wird, können wir eine breite Palette von Farben erzeugen. Zum Beispiel ergibt die Mischung von hochintensivem rotem und grünem Licht gelbes Licht, während die Kombination von blauem und grünem Licht Cyan erzeugt.

Durch dieses Projekt werden wir die Prinzipien der additiven Farbmischung erkunden und unsere Kreativität entfesseln, indem wir die RGB-LED manipulieren, um fesselnde und lebendige Farben anzuzeigen.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die unten stehenden Links kaufen.

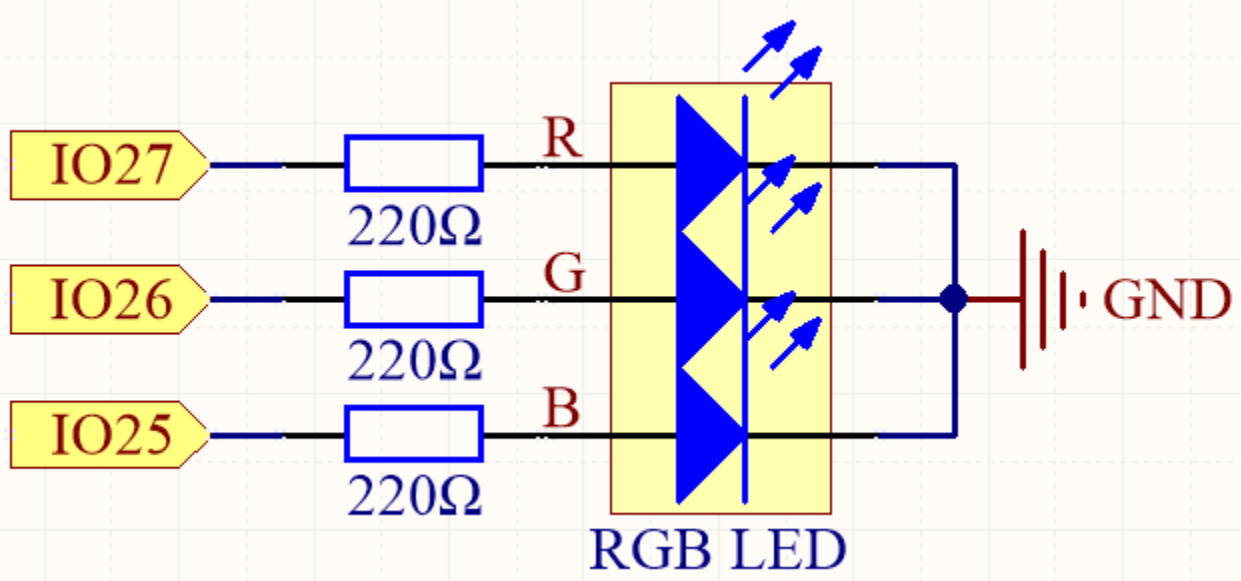
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
RGB LED	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

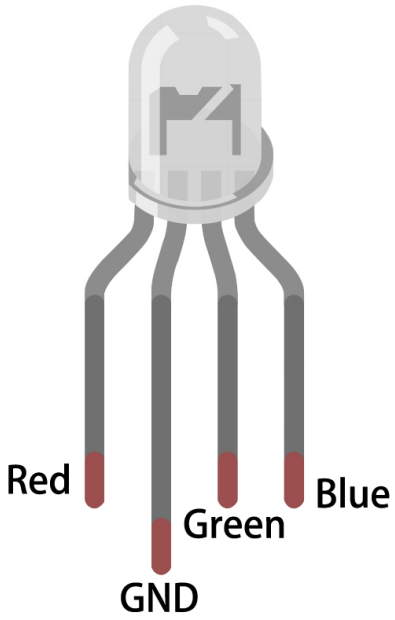
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan

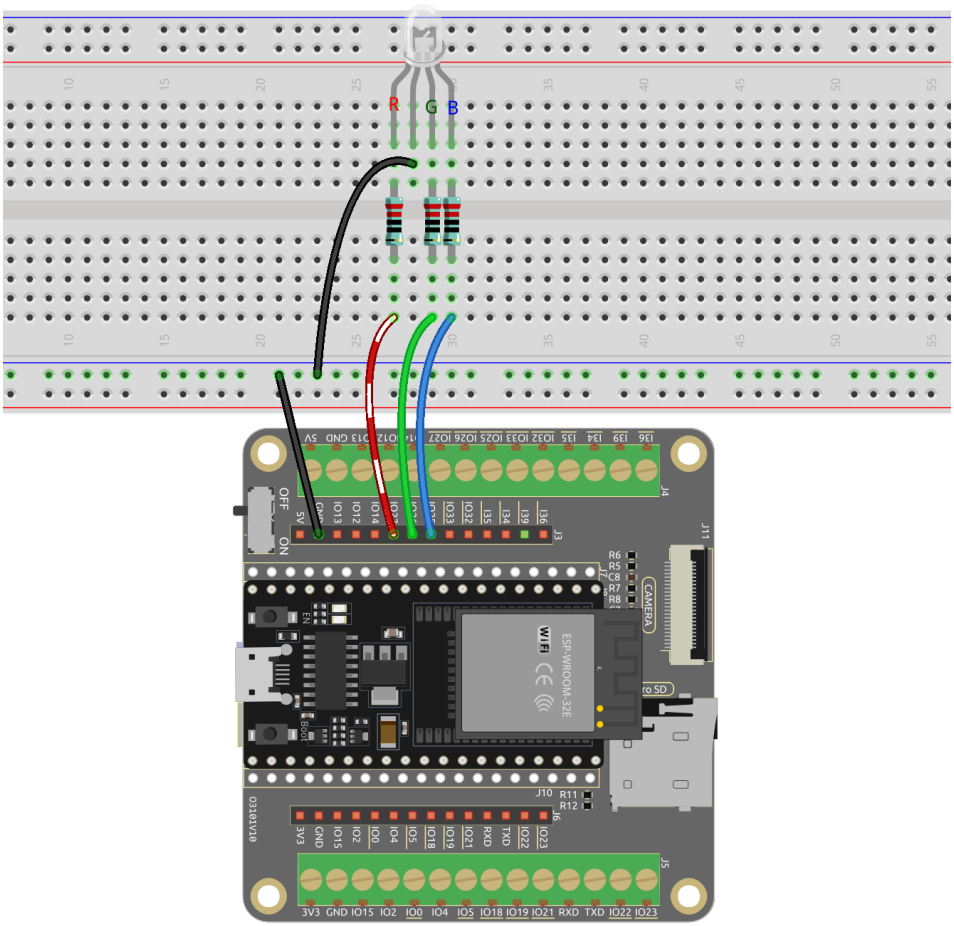


Die PWM-Pins Pin27, Pin26 und Pin25 steuern die Rot-, Grün- und Blau-Pins der RGB-LED und verbinden den gemeinsamen Kathoden-Pin mit GND. Dies ermöglicht es der RGB-LED, eine bestimmte Farbe anzuzeigen, indem Licht auf diesen Pins mit unterschiedlichen PWM-Werten überlagert wird.

Verdrahtung



Die RGB-LED hat 4 Pins: der längste Pin ist der gemeinsame Kathoden-Pin, der normalerweise mit GND verbunden wird; der linke Pin neben dem längsten Pin ist Rot; und die beiden Pins auf der rechten Seite sind Grün und Blau.



Code

Bemerkung:

- Öffnen Sie die Datei `2.3_colorful_light.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
from machine import Pin, PWM
import time

# Define the GPIO pins for the RGB LED
RED_PIN = 27
GREEN_PIN = 26
BLUE_PIN = 25


# Set up the PWM channels
red = PWM(Pin(RED_PIN))
green = PWM(Pin(GREEN_PIN))
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
blue = PWM(Pin(BLUE_PIN))

# Set the PWM frequency
red.freq(1000)
green.freq(1000)
blue.freq(1000)

def set_color(r, g, b):
    red.duty(r)
    green.duty(g)
    blue.duty(b)

while True:
    # Set different colors and wait for a while
    set_color(1023, 0, 0) # Red
    time.sleep(1)
    set_color(0, 1023, 0) # Green
    time.sleep(1)
    set_color(0, 0, 1023) # Blue
    time.sleep(1)
    set_color(1023, 0, 1023) # purple
    time.sleep(1)
```

Wenn das Skript ausgeführt wird, sehen Sie, wie die RGB-LEDs Rot, Grün, Blau und Lila sowie andere Farben anzeigen.

Mehr erfahren

Sie können auch die gewünschte Farbe mit dem folgenden Code und den bekannten Farbwerten von 0 bis 255 einstellen.

Bemerkung:

- Öffnen Sie die Datei `2.3_colorful_light_rgb.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
from machine import Pin, PWM
import time

# Define the GPIO pins for the RGB LED
RED_PIN = 27
GREEN_PIN = 26
BLUE_PIN = 25

# Set up the PWM channels
red = PWM(Pin(RED_PIN))
green = PWM(Pin(GREEN_PIN))
blue = PWM(Pin(BLUE_PIN))

# Set the PWM frequency
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
red.freq(1000)
green.freq(1000)
blue.freq(1000)

# Map input values from one range to another
def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

# Convert color values (0-255) to duty cycle values (0-1023)
def color_to_duty(rgb_value):
    rgb_value = int(interval_mapping(rgb_value,0,255,0,1023))
    return rgb_value

def set_color(red_value,green_value,blue_value):
    red.duty(color_to_duty(red_value))
    green.duty(color_to_duty(green_value))
    blue.duty(color_to_duty(blue_value))

while True:
    # Set different colors and wait for a while
    set_color(255, 0, 0) # Red
    time.sleep(1)
    set_color(0, 255, 0) # Green
    time.sleep(1)
    set_color(0, 0, 255) # Blue
    time.sleep(1)
    set_color(255, 0, 255) # purple
    time.sleep(1)
```

Dieser Code basiert auf dem vorherigen Beispiel, bildet jedoch Farbwerte von 0 bis 255 auf einen Tastgradbereich von 0 bis 1023 ab.

- Die Funktion `interval_mapping` ist eine Hilfsfunktion, die einen Wert von einem Bereich in einen anderen abbildet. Sie nimmt fünf Argumente an: den Eingabewert, die minimalen und maximalen Werte des Eingabebereichs und die minimalen und maximalen Werte des Ausgabebereichs. Sie gibt den Eingabewert, abgebildet auf den Ausgabebereich, zurück.

  ```
  def color_to_duty(rgb_value):
      rgb_value = int(interval_mapping(rgb_value,0,255,0,1023))
      return rgb_value
  ```

- Die Funktion `color_to_duty` nimmt einen ganzzahligen RGB-Wert (z.B. 255,0,255) entgegen und bildet ihn auf einen Tastgradwert, der für die PWM-Pins geeignet ist. Der Eingabe-RGB-Wert wird zunächst von 0-255 auf 0-1023 mit der Funktion `interval_mapping` abgebildet. Der Ausgabewert von `interval_mapping` wird dann als Tastgradwert zurückgegeben.

  ```
  def farbe_zu_tastgrad(rgb_wert):
      rgb_wert = int(bereichsabbildung(rgb_wert,0,255,0,1023))
      return rgb_wert
  ```

- Die Funktion `color_set` nimmt drei ganzzahlige Argumente an: die Rot-, Grün- und Blauwerte für die LED. Diese Werte werden an `color_to_duty` übergeben, um die Tastgradwerte für die PWM-Pins zu erhalten. Die Tastgradwerte werden dann mit der Methode `duty` für die entsprechenden Pins eingestellt.

```
def set_color(red_value,green_value,blue_value):
    red.duty(color_to_duty(red_value))
    green.duty(color_to_duty(green_value))
    blue.duty(color_to_duty(blue_value))
```

3.10 2.4 Mikrochip - 74HC595

Willkommen zu diesem spannenden Projekt! In diesem Projekt werden wir den 74HC595-Chip verwenden, um eine fließende Anzeige von 8 LEDs zu steuern.

Stellen Sie sich vor, dieses Projekt auszulösen und einen faszinierenden Lichtfluss zu beobachten, als ob ein funkelnder Regenbogen zwischen den 8 LEDs hin und her springt. Jede LED wird nacheinander aufleuchten und schnell verblassen, während die nächste LED weiterhin strahlt und einen wunderschönen und dynamischen Effekt erzeugt.

Durch die clevere Nutzung des 74HC595-Chips können wir die Ein- und Ausschaltzustände mehrerer LEDs steuern, um den fließenden Effekt zu erreichen. Dieser Chip verfügt über mehrere Ausgangspins, die in Serie geschaltet werden können, um die Reihenfolge der LED-Beleuchtung zu steuern. Darüber hinaus ermöglicht die Erweiterbarkeit des Chips, problemlos mehr LEDs zur fließenden Anzeige hinzuzufügen und noch spektakulärere Effekte zu erzeugen.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die unten stehenden Links kaufen.

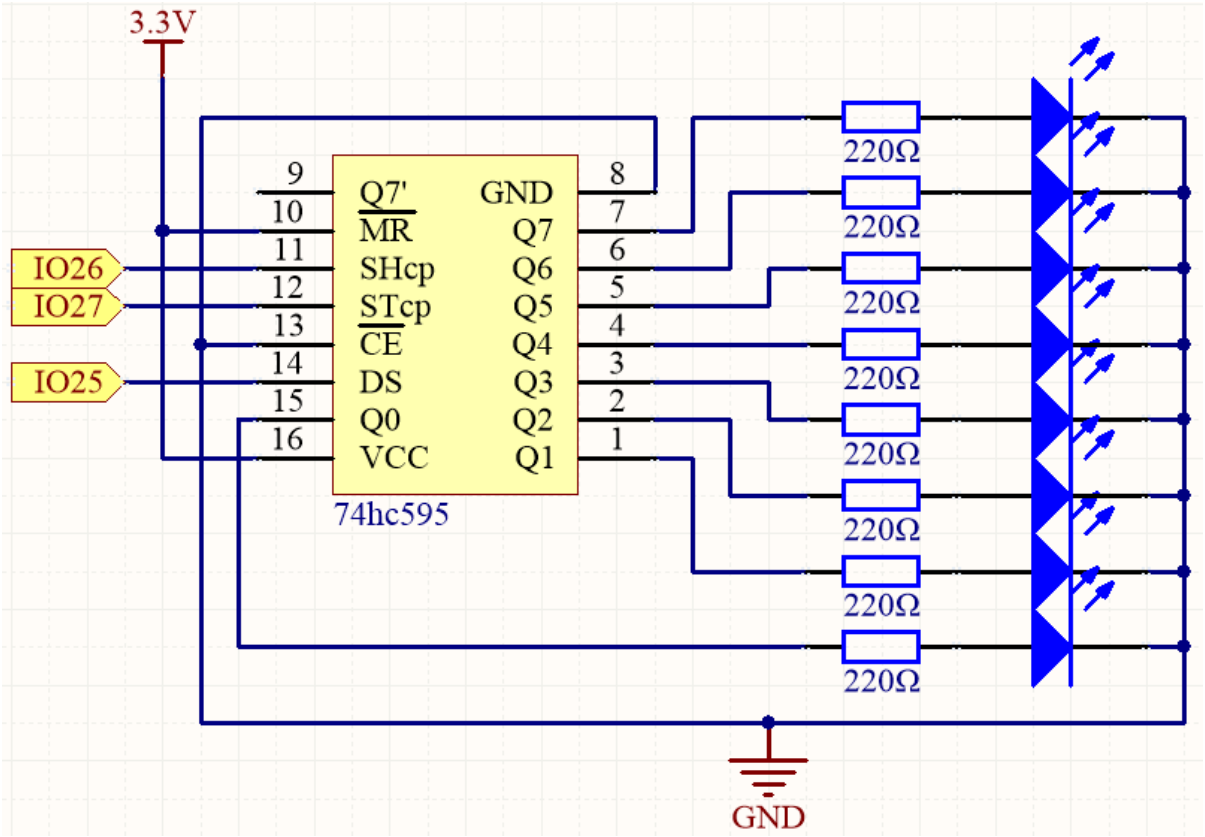
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
74HC595	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

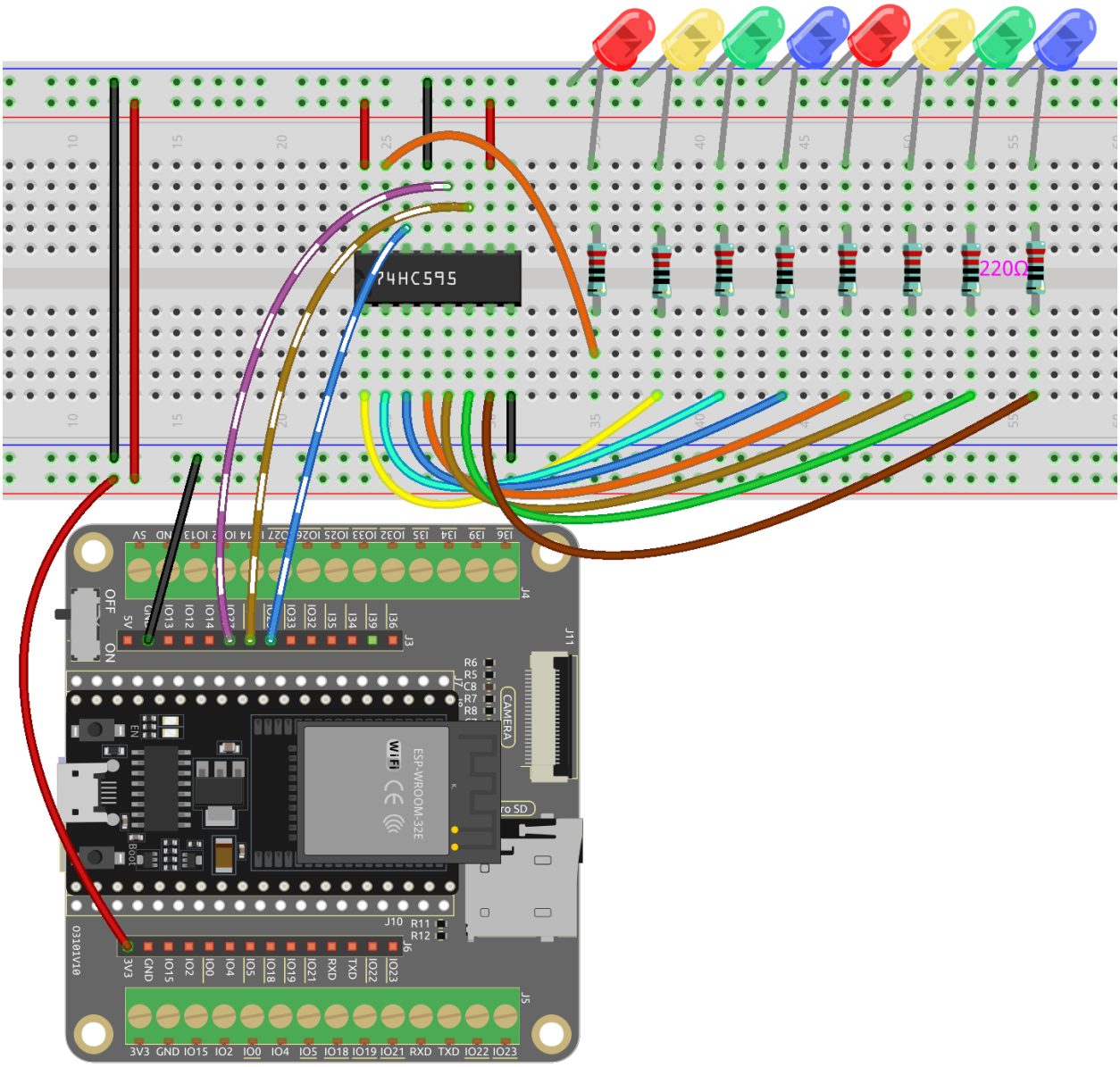
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



- Wenn MR (Pin10) High-Level und CE (Pin13) Low-Level ist, werden Daten im ansteigenden Flanken von SHcp eingegeben und gehen durch die ansteigende Flanke von SHcp ins Speicherregister.
- Wenn die beiden Uhren miteinander verbunden sind, ist das Schieberegister immer einen Puls früher als das Speicherregister.
- Es gibt einen seriellen Schiebeeingang (DS), einen seriellen Ausgang (Q7‘) und einen asynchronen Reset-Knopf (niedriges Niveau) im Speicherregister.
- Das Speicherregister gibt einen Bus mit einem parallelen 8-Bit und in drei Zuständen aus.
- Wenn OE aktiviert ist (niedriges Niveau), werden die Daten im Speicherregister auf den Bus(Q0 ~ Q7) ausgegeben.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `2.4_microchip_74hc595.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
# Initialize the pins for the 74HC595 shift register
sdi = machine.Pin(25, machine.Pin.OUT)  # DS
rclk = machine.Pin(27, machine.Pin.OUT)  # STcp
srclk = machine.Pin(26, machine.Pin.OUT)  # SHcp

# Define the hc595_shift function to shift data into the 74HC595 shift register
def hc595_shift(dat):
    # Set the RCLK pin to low
    rclk.off()

    # Iterate through each bit (from 7 to 0)
    for bit in range(7, -1, -1):
        # Extract the current bit from the input data
        value = 1 & (dat >> bit)

        # Set the SRCLK pin to low
        srclk.off()

        # Set the value of the SDI pin
        sdi.value(value)

        # Clock the current bit into the shift register by setting the SRCLK pin to high
        srclk.on()

    # Latch the data into the storage register by setting the RCLK pin to high
    rclk.on()

num = 0

# Shift data into the 74HC595 to create a moving LED pattern
for i in range(16):
    if i < 8:
        num = (num << 1) + 1  # Shift left and set the least significant bit to 1
    elif i >= 8:
        num = (num & 0b01111111) << 1  # Mask the most significant bit and shift left
    hc595_shift(num)  # Shift the current value into the 74HC595
    print("{:0>8b}".format(num))  # Print the current value in binary format
    time.sleep_ms(200)  # Wait 200 milliseconds before shifting the next value
```

Während der Ausführung des Skripts sehen Sie, wie die LEDs nacheinander aufleuchten und dann in der ursprünglichen Reihenfolge ausschalten.

Wie funktioniert das?

Dieser Code wird verwendet, um ein 8-Bit-Schieberegister (74595) zu steuern und verschiedene binäre Werte an das Schieberegister auszugeben, wobei jeder Wert für eine bestimmte Zeit auf einer LED angezeigt wird.

1. Der Code importiert die Module `machine` und `time`, wobei das Modul `machine` zur Steuerung der Hardware-I/O und das Modul `time` für die Implementierung von Zeitverzögerungen und anderen Funktionen verwendet wird.

   ```
   import machine
   import time
   ```

2. Drei Ausgangsports werden mit der Funktion `machine.Pin()` initialisiert und entsprechen dem Datenport (SDI), Speichertaktport (RCLK) und Schieberegistertaktport (SRCLK) des Schieberegisters.

```
# Initialize the pins for the 74HC595 shift register
sdi = machine.Pin(25, machine.Pin.OUT)  # DS
rclk = machine.Pin(27, machine.Pin.OUT)  # STcp
srclk = machine.Pin(26, machine.Pin.OUT)  # SHcp
```

3. Eine Funktion namens `hc595_shift()` wird definiert, um ein 8-Bit-Daten an das Schieberegister zu schreiben.

```
def hc595_shift(dat):
    # Set the RCLK pin to low
    rclk.off()

    # Iterate through each bit (from 7 to 0)
    for bit in range(7, -1, -1):
        # Extract the current bit from the input data
        value = 1 & (dat >> bit)

        # Set the SRCLK pin to low
        srclk.off()

        # Set the value of the SDI pin
        sdi.value(value)

        # Clock the current bit into the shift register by setting the␣
↪SRCLK pin to high
        srclk.on()

    # Latch the data into the storage register by setting the RCLK pin to␣
↪high
    rclk.on()
```

4. Über die `for`-Schleife.

```
for i in range(16):
        if i < 8:
            num = (num << 1) + 1  # Shift left and set the least significant␣
↪bit to 1
        elif i >= 8:
            num = (num & 0b01111111) << 1  # Mask the most significant bit␣
↪and shift left
        hc595_shift(num)  # Shift the current value into the 74HC595
        print("{:0>8b}".format(num))  # Print the current value in binary␣
↪format
        time.sleep_ms(200)  # Wait 200 milliseconds before shifting the␣
↪next value
```

- Die Variable `i` wird verwendet, um den Ausgabebinärwert zu steuern. In den ersten 8 Iterationen wird der Wert von num sukzessive 00000001, 00000011, 00000111, ..., 11111111 sein, der um ein Bit nach links verschoben und dann um 1 erhöht wird.
- In den 9. bis 16. Iterationen wird das höchste Bit von 1 zuerst in 0 geändert und dann um ein Bit nach links verschoben, was zu den Ausgabewerten 00000010, 00000100, 00001000, ..., 10000000 führt.

- In jeder Iteration wird der Wert von `num` an die Funktion `hc595_shift()` übergeben, um das Schieberegister zu steuern, um den entsprechenden Binärwert auszugeben.
- Gleichzeitig mit dem Ausgeben des Binärwerts gibt die Funktion `print()` den Binärwert als Zeichenkette an das Terminal aus.
- Nach dem Ausgeben des Binärwerts pausiert das Programm 200 Millisekunden mit der Funktion `time.sleep_ms()`, damit der Wert auf der LED für eine bestimmte Zeit angezeigt bleibt.

3.11 2.5 Ziffernanzeige

Willkommen zu diesem faszinierenden Projekt! In diesem Projekt werden wir die zauberhafte Welt der Anzeige von Zahlen von 0 bis 9 auf einem Siebensegment-Display erkunden.

Stellen Sie sich vor, dieses Projekt auszulösen und zu beobachten, wie ein kleines, kompaktes Display mit jeder Zahl von 0 bis 9 hell leuchtet. Es ist, als hätte man einen Miniaturbildschirm, der die Ziffern auf fesselnde Weise präsentiert. Durch Steuerung der Signalleitungen können Sie die angezeigte Zahl mühelos ändern und verschiedene ansprechende Effekte erzeugen.

Durch einfache Schaltungsverbindungen und Programmierung lernen Sie, wie Sie mit dem Siebensegment-Display interagieren und Ihre gewünschten Zahlen zum Leben erwecken. Ob es sich um einen Zähler, eine Uhr oder eine andere faszinierende Anwendung handelt, das Siebensegment-Display wird Ihr zuverlässiger Begleiter sein und Ihren Projekten einen Hauch von Brillanz verleihen.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die unten stehenden Links kaufen.

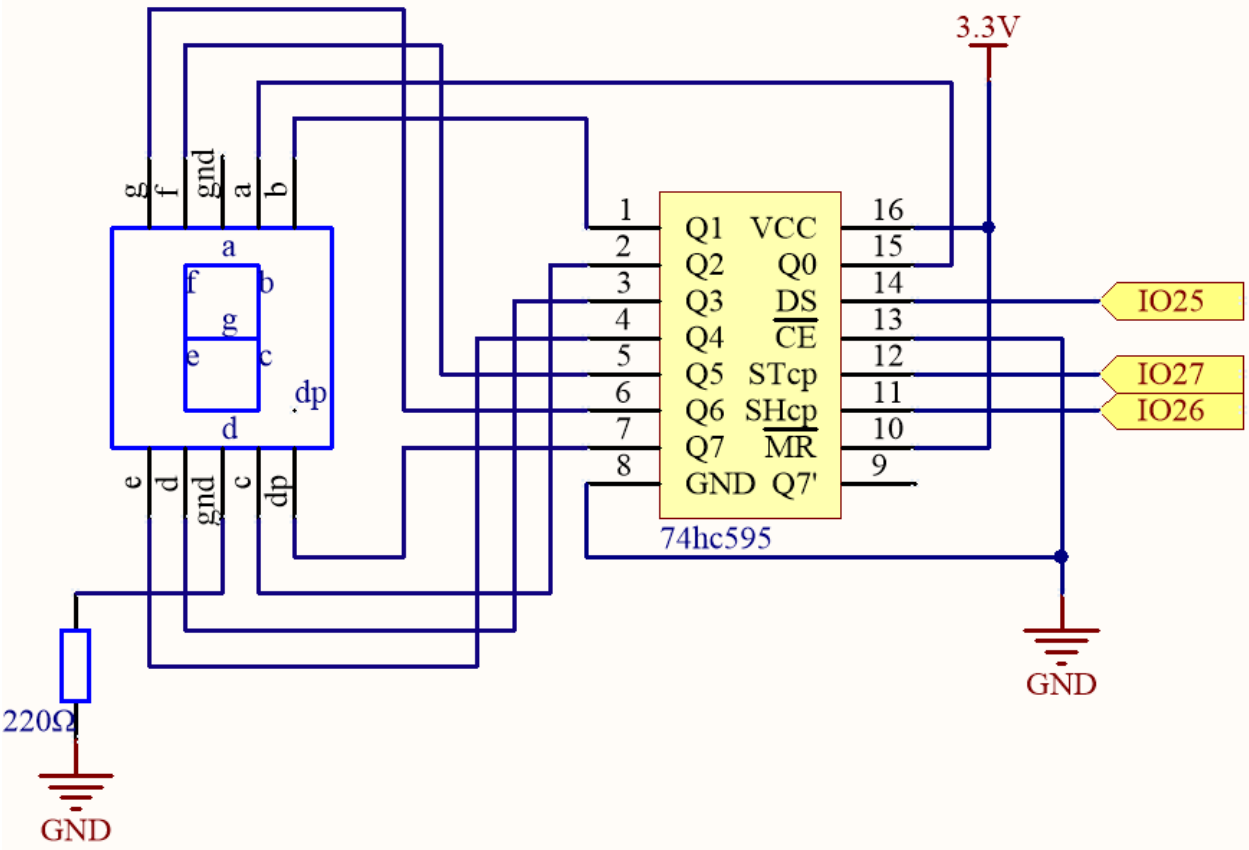
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
7-Segment-Anzeige	
74HC595	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan

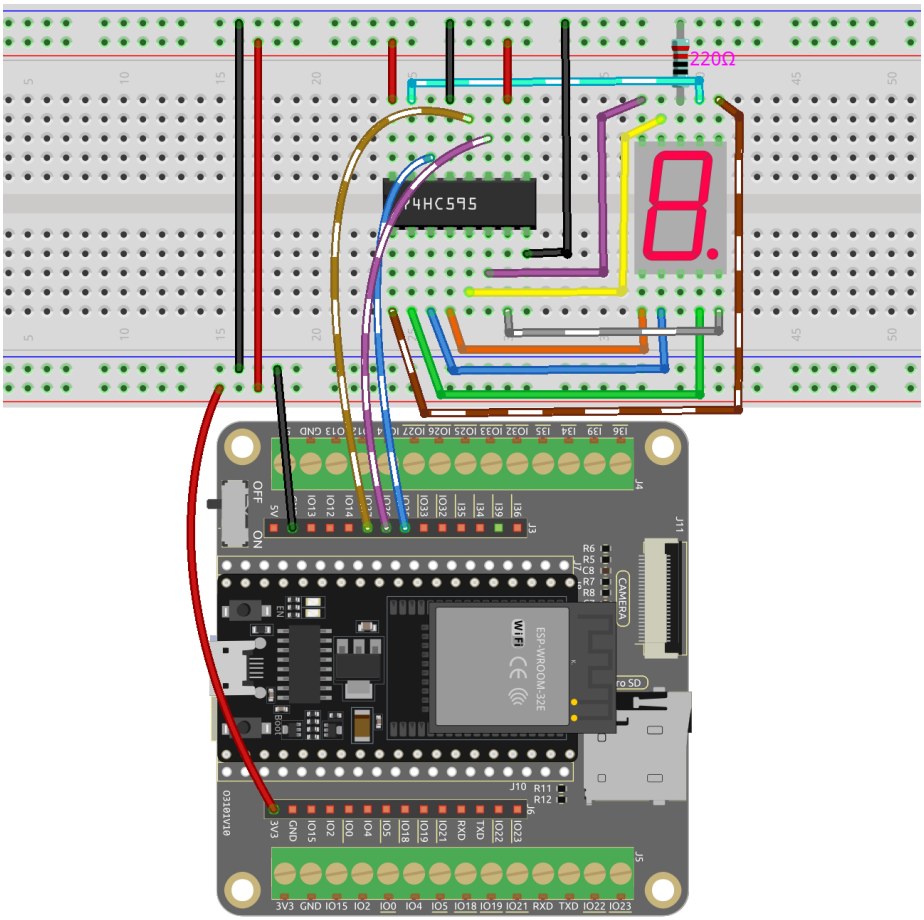


Hier ist das Verdrahtungsprinzip im Grunde das gleiche wie bei *2.4 Mikrochip - 74HC595*, der einzige Unterschied ist, dass Q0-Q7 mit den a ~ g Pins des 7-Segment-Displays verbunden sind.

Tab. 1: Verdrahtung

74HC595	LED-Segmentanzeige
Q0	a
Q1	b
Q2	c
Q3	d
Q4	e
Q5	f
Q6	g
Q7	dp

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `2.5_number_display.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time

# Define the segment code for a common anode 7-segment display
SEGCODE = [0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f]

# Initialize the pins for the 74HC595 shift register
sdi = machine.Pin(25, machine.Pin.OUT)  # DS
rclk = machine.Pin(27, machine.Pin.OUT)  # STcp
srclk = machine.Pin(26, machine.Pin.OUT)  # SHcp

# Define the hc595_shift function to shift data into the 74HC595 shift register
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
def hc595_shift(dat):
    # Set the RCLK pin to low
    rclk.off()

    # Iterate through each bit (from 7 to 0)
    for bit in range(7, -1, -1):
        # Extract the current bit from the input data
        value = 1 & (dat >> bit)

        # Set the SRCLK pin to low
        srclk.off()

        # Set the value of the SDI pin
        sdi.value(value)

        # Clock the current bit into the shift register by setting the SRCLK pin to high
        srclk.on()

    # Latch the data into the storage register by setting the RCLK pin to high
    rclk.on()

# Continuously loop through the numbers 0 to 9 and display them on the 7-segment display
while True:
    for num in range(10):
        hc595_shift(SEGCODE[num])  # Shift the segment code for the current number into␣
↪the 74HC595
        time.sleep_ms(500)  # Wait 500 milliseconds before displaying the next number
```

Wenn das Skript läuft, können Sie sehen, wie das LED-Segmentdisplay nacheinander die Zahlen 0 bis 9 anzeigt.

Wie funktioniert das?

In diesem Projekt verwenden wir die Funktion `hc595_shift()`, um die Binärzahl in das Schieberegister zu schreiben.

Angenommen, das 7-Segment-Display soll die Zahl „2“ anzeigen. Dieses Bitmuster entspricht den Segmenten **f**, **c** und **dp**, die ausgeschaltet (niedrig) sind, während die Segmente **a**, **b**, **d**, **e** und **g** eingeschaltet (hoch) sind. Das entspricht „01011011“ in Binär- und „0x5b“ in hexadezimaler Schreibweise.

Daher müssten Sie **hc595_shift(0x5b)** aufrufen, um die Zahl „2“ auf dem 7-Segment-Display anzuzeigen.

- Hexadezimal
- Binär-Hex-Konverter

Die folgende Tabelle zeigt die hexadezimalen Muster, die in das Schieberegister geschrieben werden müssen, um die Zahlen 0 bis 9 auf einem 7-Segment-Display anzuzeigen.

Tab. 2: Glyph-Code

Zahlen	Binärcode	Hexcode
0	00111111	0x3f
1	00000110	0x06
2	01011011	0x5b
3	01001111	0x4f
4	01100110	0x66
5	01101101	0x6d
6	01111101	0x7d
7	00000111	0x07
8	01111111	0x7f
9	01101111	0x6f

Schreiben Sie diese Codes in `hc595_shift()`, damit das LED-Segmentdisplay die entsprechenden Zahlen anzeigt.

3.12 2.6 Zeichenanzeige

Jetzt werden wir die faszinierende Welt der Zeichenanzeige mit dem I2C LCD1602-Modul erkunden.

In diesem Projekt lernen wir, wie man das LCD-Modul initialisiert, die gewünschten Anzeigeparameter einstellt und Zeichendaten zur Anzeige auf dem Bildschirm sendet. Wir können benutzerdefinierte Nachrichten präsentieren, Sensordaten anzeigen oder interaktive Menüs erstellen. Die Möglichkeiten sind endlos!

Indem wir die Kunst der Zeichenanzeige auf dem I2C LCD1602 meistern, erschließen wir neue Wege für Kommunikation und Informationsanzeige in unseren Projekten. Lassen Sie uns in diese spannende Reise eintauchen und unsere

Zeichen auf dem LCD-Bildschirm zum Leben erwecken.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die unten stehenden Links kaufen.

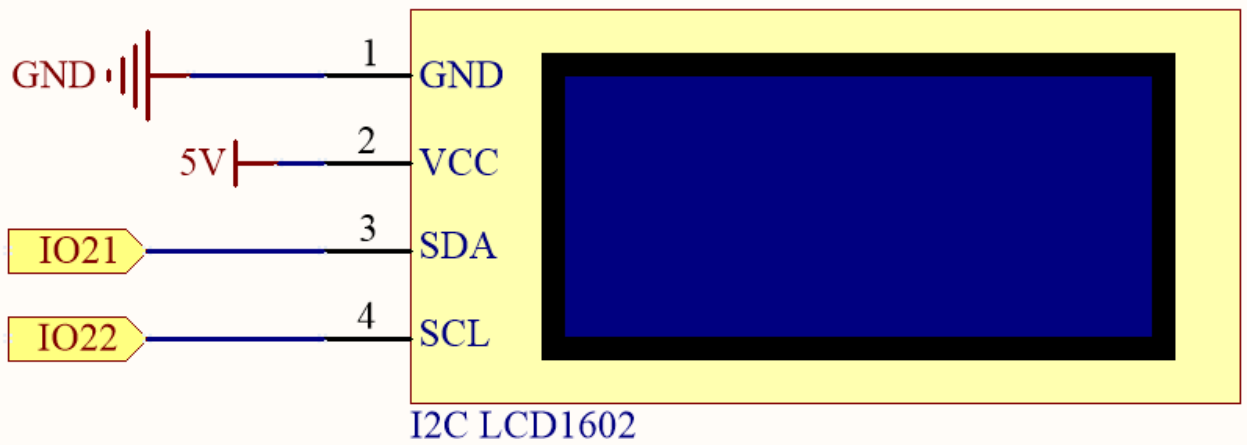
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
I2C LCD1602	

Verfügbare Pins

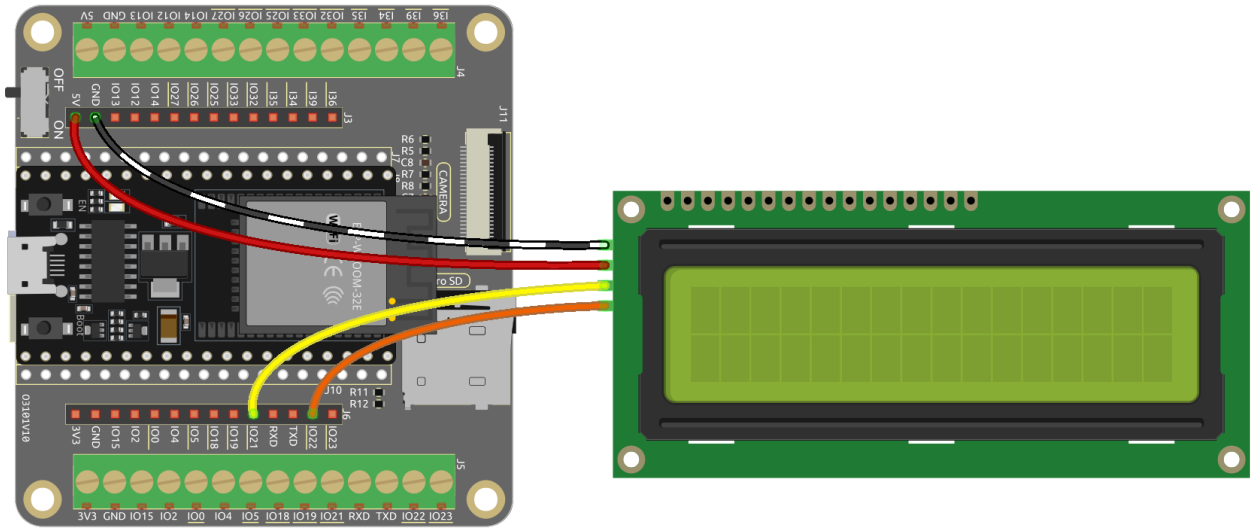
Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	Verwendungszweck
IO21	SDA
IO22	SCL

Schaltplan



Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `2.6_liquid_crystal_display.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.
- Die Bibliothek `lcd1602.py` wird hier verwendet. Überprüfen Sie, ob sie auf ESP32 hochgeladen ist. Siehe *1.4 Bibliotheken Hochladen (Wichtig)* für ein Tutorial.

```
# Import the LCD class from the lcd1602 module
from lcd1602 import LCD

import time

# Create an instance of the LCD class and assign it to the lcd variable
lcd = LCD()
# Set the string " Hello!\n"
string = " Hello!\n"
# Display the string on the LCD screen
lcd.message(string)

time.sleep(2)
# Set the string "    Sunfounder!"
string = "    Sunfounder!"
# Display the string on the LCD screen
lcd.message(string)

time.sleep(2)
# Clear the LCD screen
lcd.clear()
```

Nachdem das Skript ausgeführt wurde, werden Sie zwei Textzeilen auf dem LCD-Bildschirm nacheinander erscheinen

und dann verschwinden sehen.

Bemerkung: Wenn der Code und die Verdrahtung korrekt sind, das LCD aber trotzdem keinen Inhalt anzeigt, können Sie das Potentiometer auf der Rückseite justieren, um den Kontrast zu erhöhen.

Wie funktioniert das?

In der Bibliothek `lcd1602` integrieren wir die relevanten Funktionen von lcd1602 in die LCD-Klasse.

1. Importieren des `lcd1602`-Moduls.

```
from lcd1602 import LCD
```

2. Deklarieren eines Objekts der Klasse `LCD` und Benennen es als `lcd`.

```
lcd = LCD()
```

3. Diese Anweisung wird den Text auf dem LCD anzeigen. Es sollte beachtet werden, dass das Argument ein Stringtyp sein muss. Wenn wir eine Ganzzahl oder Float übergeben wollen, müssen wir die erzwungene Konvertierungsanweisung `str()` verwenden.

```
lcd.message(string)
```

4. Wenn Sie diese Anweisung mehrmals aufrufen, wird das lcd die Texte überlagern. Dies erfordert die Verwendung der folgenden Anweisung, um die Anzeige zu löschen.

```
lcd.clear()
```

3.13 2.7 RGB-LED-Streifen

In diesem Projekt werden wir uns in die faszinierende Welt der Steuerung von WS2812-LED-Streifen vertiefen und eine lebendige Farbdisplay zum Leben erwecken. Mit der Möglichkeit, jede LED auf dem Streifen einzeln zu steuern, können wir fesselnde Beleuchtungseffekte erzeugen, die die Sinne verzaubern.

Darüber hinaus haben wir eine spannende Erweiterung zu diesem Projekt hinzugefügt, in der wir das Reich der Zufälligkeit erkunden werden. Indem wir zufällige Farben einführen und einen fließenden Lichteffekt implementieren, können wir ein faszinierendes visuelles Erlebnis schaffen, das fesselt und verzaubert.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

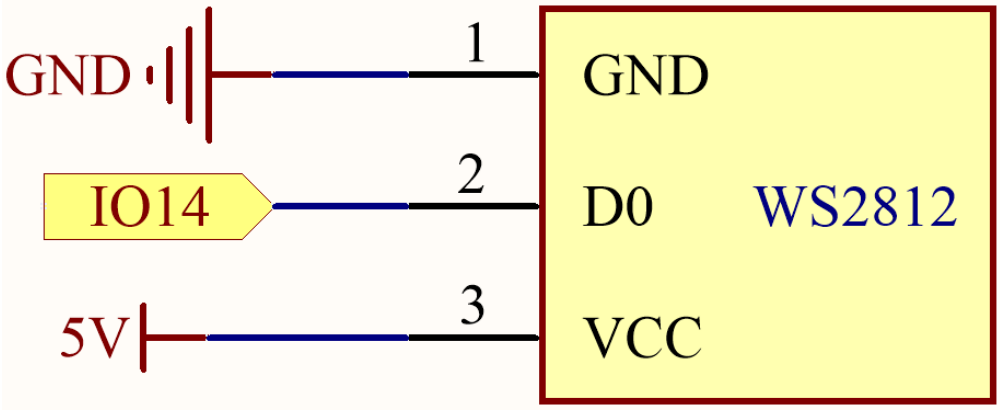
Es ist definitiv praktisch, ein ganzes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die unten stehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
WS2812 RGB 8 LEDs Leiste	

Schaltplan



Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

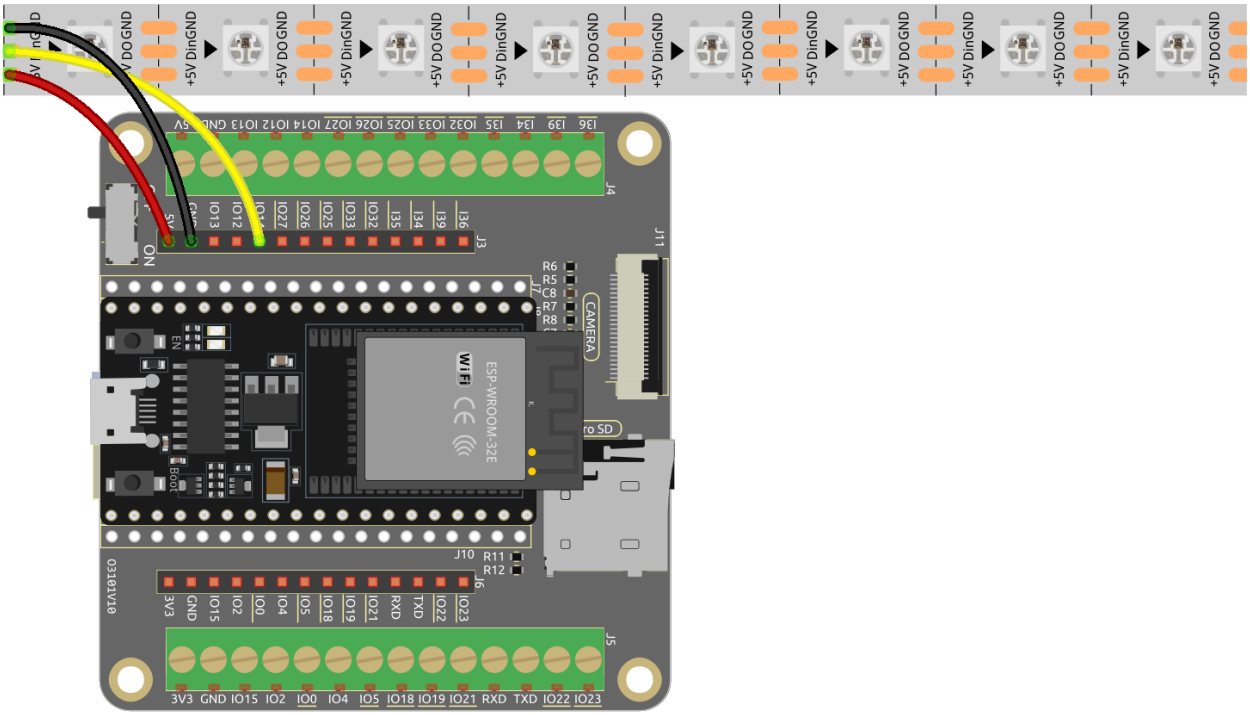
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Bemerkung: IO33 ist für dieses Projekt nicht verfügbar.

Der WS2812-LED-Streifen ist eine Art LED-Streifen, der ein präzises Pulsbreitenmodulations-(PWM-)Signal benötigt. Das PWM-Signal hat genaue Anforderungen sowohl in der Zeit als auch in der Spannung. Zum Beispiel entspricht ein „0"-Bit für den WS2812 einem High-Level-Puls von etwa 0,4 Mikrosekunden, während ein „1"-Bit einem High-Level-Puls von etwa 0,8 Mikrosekunden entspricht. Das bedeutet, dass der Streifen Hochfrequenzspannungsänderungen empfangen muss.

Jedoch wird mit einem 4,7K-Pull-up-Widerstand und einem 100nf-Pull-down-Kondensator an IO33 ein einfacher Tiefpassfilter erstellt. Diese Art von Schaltung „glättet" Hochfrequenzsignale, da der Kondensator einige Zeit benötigt, um sich aufzuladen und zu entladen, wenn er Spannungsänderungen erhält. Wenn das Signal also zu schnell wechselt (d.h. hochfrequent ist), kann der Kondensator nicht mithalten. Dies führt dazu, dass das Ausgangssignal verwischt und für den Streifen unerkennbar wird.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `2.7_rgb_strip.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
from machine import Pin
from neopixel import NeoPixel

pin = Pin(14, Pin.OUT)    # set a pin to output to drive NeoPixels
pixels = NeoPixel(pin, 8)   # create NeoPixel driver on pin for 8 pixels

pixels[0] = [64,154,227]    # set the pixel
pixels[1] = [128,0,128]
pixels[2] = [50,150,50]
pixels[3] = [255,30,30]
pixels[4] = [0,128,255]
pixels[5] = [99,199,0]
pixels[6] = [128,128,128]
pixels[7] = [255,100,0]

pixels.write()              # write data to all pixels
```

Lassen Sie uns einige Lieblingsfarben auswählen und sie auf dem RGB-LED-Streifen anzeigen!

Wie funktioniert das?

1. Im Modul `neopixel` haben wir verwandte Funktionen in die Klasse `NeoPixel` integriert.

```
from neopixel import NeoPixel
```

2. Verwenden Sie die Klasse `NeoPixel` aus dem Modul `neopixel`, um das Objekt `pixels` zu initialisieren, wobei Sie den Datenpin und die Anzahl der LEDs angeben.

```
pixels = NeoPixel(pin, 8)    # create NeoPixel driver on pin for 8 pixels
```

3. Stellen Sie die Farbe jeder LED ein und verwenden Sie die Methode `write()`, um die Daten an den WS2812-LED zu senden und seine Anzeige zu aktualisieren.

```
pixels[0] = [64,154,227]    # set the pixel
pixels[1] = [128,0,128]
pixels[2] = [50,150,50]
pixels[3] = [255,30,30]
pixels[4] = [0,128,255]
pixels[5] = [99,199,0]
pixels[6] = [128,128,128]
pixels[7] = [255,100,0]

pixels.write()              # write data to all pixels
```

Mehr erfahren

Wir können zufällig Farben generieren und ein buntes fließendes Licht machen.

Bemerkung:

- Öffnen Sie die Datei `2.7_rgb_strip_random.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen. * Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
from machine import Pin
import neopixel
import time
import random

# Set the number of pixels for the running light
num_pixels = 8

# Set the data pin for the RGB LED strip
data_pin = Pin(14, Pin.OUT)

# Initialize the RGB LED strip object
pixels = neopixel.NeoPixel(data_pin, num_pixels)

# Continuously loop the running light
while True:
    for i in range(num_pixels):
        # Generate a random color for the current pixel
        color = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        # Turn on the current pixel with the random color
        pixels[i] = color

        # Update the RGB LED strip display
        pixels.write()

        # Turn off the current pixel
        pixels[i] = (0, 0, 0)

        # Wait for a period of time to control the speed of the running light
        time.sleep_ms(100)
```

- In der `while`-Schleife verwenden wir eine `for`-Schleife, um jedes Pixel des RGB-LED-Streifens nacheinander einzuschalten.
- Zuerst verwenden wir die Funktion `random.randint()`, um eine zufällige Farbe für das aktuelle Pixel zu generieren.
- Dann schalten wir das aktuelle Pixel mit der zufälligen Farbe ein, verwenden die Methode `write()` des `NeoPixel`-Objekts, um die Farbdaten an den RGB-LED-Streifen zu senden und seine Anzeige zu aktualisieren.
- Schließlich schalten wir das aktuelle Pixel aus, indem wir seine Farbe auf (0, 0, 0) einstellen, und warten eine Zeit lang, um die Geschwindigkeit des Lauflichts zu steuern.

3. Töne

3.14 3.1 Beep

Dies ist ein einfaches Projekt, um einen aktiven Summer viermal pro Sekunde schnell piepsen zu lassen.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch einzeln über die untenstehenden Links kaufen.

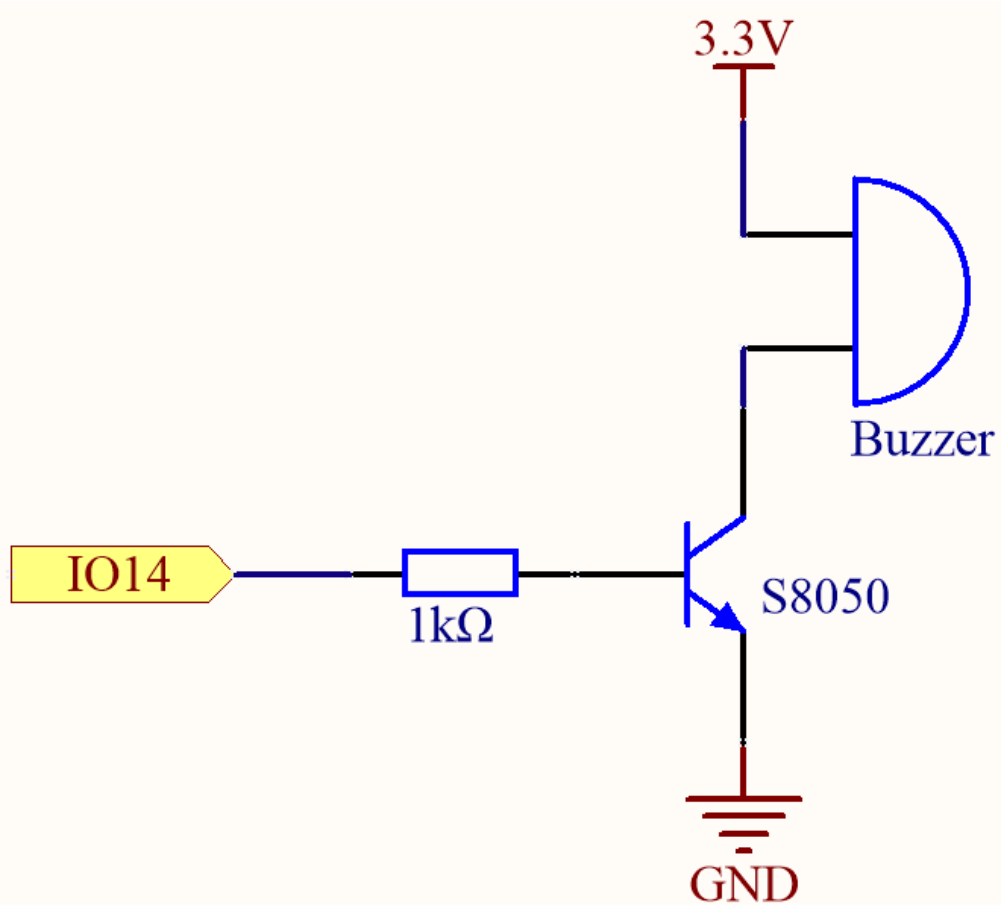
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Summer	-
Transistor	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Wenn der IO14-Ausgang hoch ist, leitet der S8050 (NPN-Transistor) nach dem 1K-Strombegrenzungswiderstand (zum

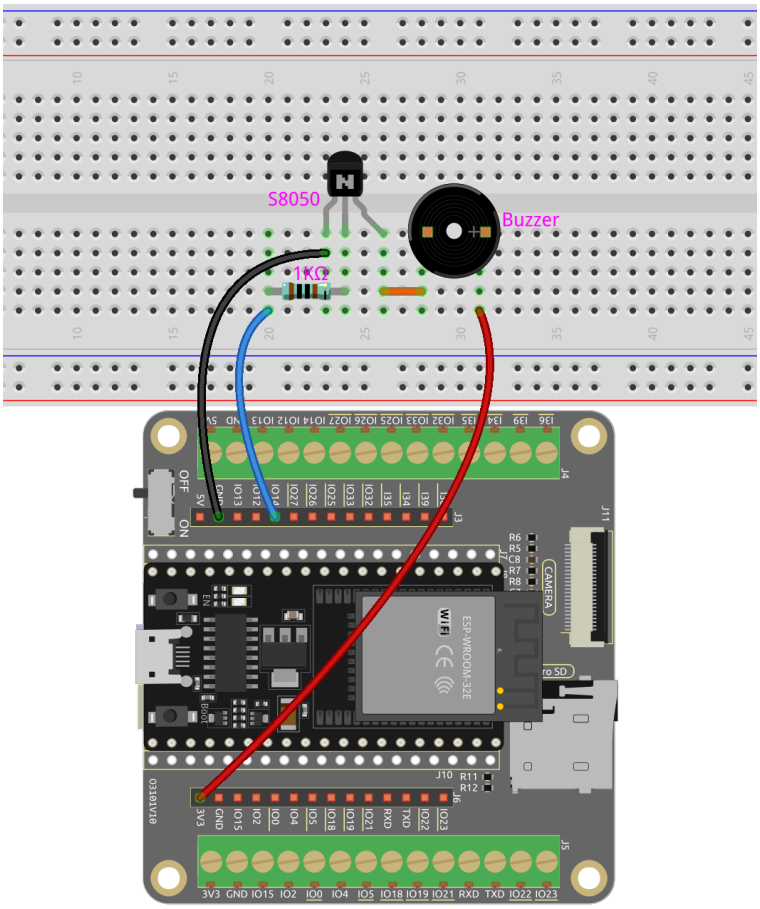
Schutz des Transistors), sodass der Summer ertönt.

Die Rolle des S8050 (NPN-Transistor) ist es, den Strom zu verstärken und den Summer lauter klingen zu lassen. Tatsächlich können Sie den Summer auch direkt an IO14 anschließen, aber dann wird der Summer leiser klingen.

Verdrahtung

Im Set sind zwei Arten von Summern enthalten. Wir benötigen den aktiven Summer. Drehen Sie ihn um, die versiegelte Rückseite (nicht die freiliegende Leiterplatte) ist die, die wir verwenden wollen.

Der Summer benötigt beim Betrieb einen Transistor, hier verwenden wir S8050 (NPN-Transistor).



Code

Bemerkung:

- Öffnen Sie die Datei `3.1_beep.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time

# Create a Pin object representing pin 14 and set it to output mode
buzzer = machine.Pin(14, machine.Pin.OUT)

# Enter an infinite loop
while True:
    # Iterate over the values 0 to 3 using a for loop
    for i in range(4):
        # Turn on the buzzer by setting its value to 1
        buzzer.value(1)
        # Pause for 0.2 seconds
        time.sleep(0.2)
        # Turn off the buzzer
        buzzer.value(0)
        # Pause for 0.2 seconds
        time.sleep(0.2)
    # Pause for 1 second before restarting the for loop
    time.sleep(1)
```

Wenn das Skript ausgeführt wird, piept der Summer jede Sekunde schnell viermal.

3.15 3.2 Eigener Klang

Im vorherigen Projekt haben wir einen aktiven Summer verwendet, dieses Mal benutzen wir einen passiven Summer.

Wie der aktive Summer nutzt auch der passive Summer das Phänomen der elektromagnetischen Induktion. Der Unterschied besteht darin, dass ein passiver Summer keine eigene Oszillationsquelle hat und daher nicht piept, wenn Gleichstromsignale verwendet werden. Dies ermöglicht es dem passiven Summer jedoch, seine eigene Oszillationsfrequenz anzupassen und unterschiedliche Töne wie „do, re, mi, fa, sol, la, ti“ zu erzeugen.

Lassen Sie den passiven Summer eine Melodie abspielen!

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch einzeln über die untenstehenden Links kaufen.

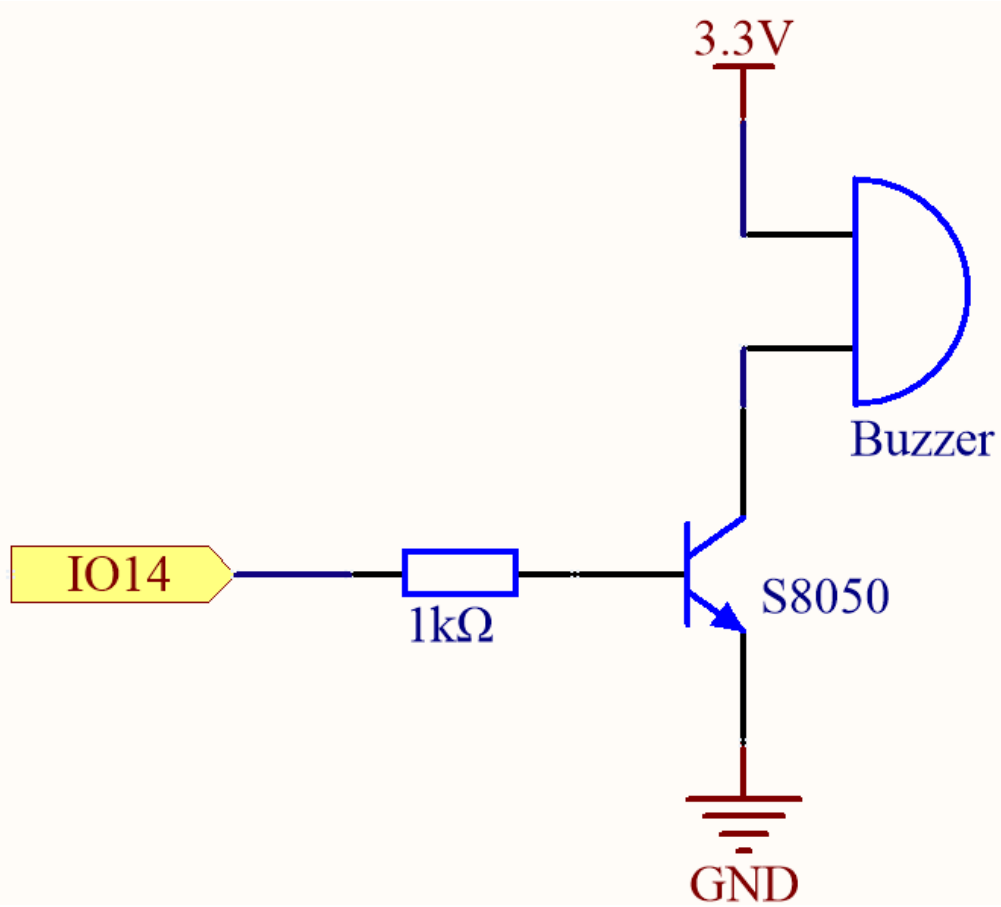
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Summer	-
Transistor	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Wenn der IO14-Ausgang hoch ist, leitet der S8050 (NPN-Transistor) nach dem 1K-Strombegrenzungswiderstand (zum

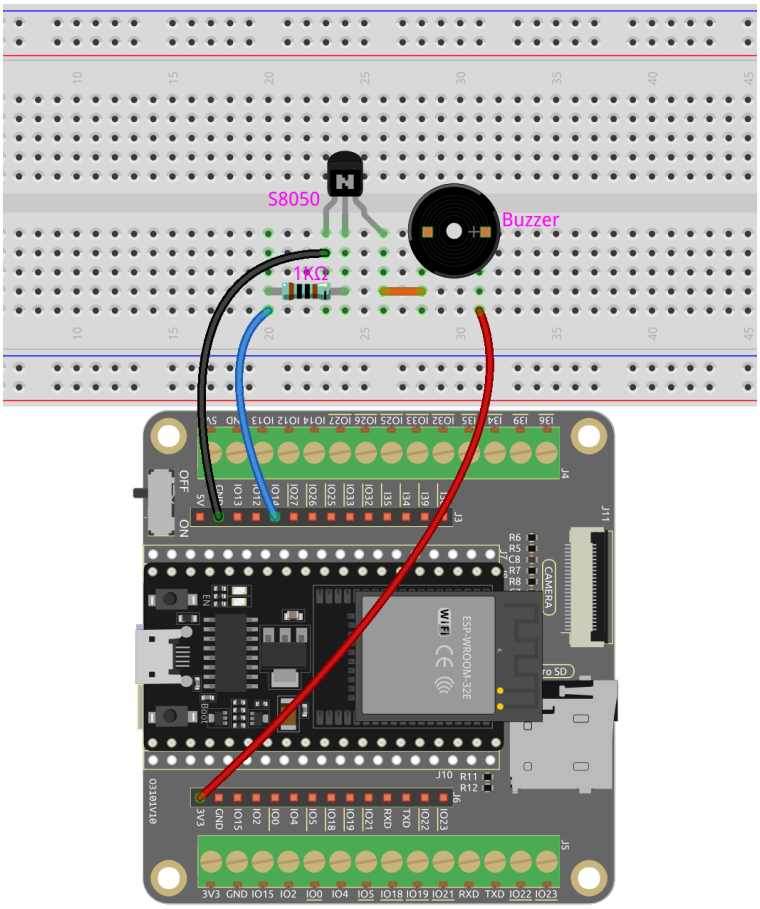
Schutz des Transistors), sodass der Summer ertönt.

Die Rolle des S8050 (NPN-Transistor) ist es, den Strom zu verstärken und den Summer lauter klingen zu lassen. Tatsächlich können Sie den Summer auch direkt an IO14 anschließen, aber dann wird der Summer leiser klingen.

Verdrahtung

Im Set sind zwei Arten von Summern enthalten. Wir benötigen den passiven Summer. Drehen Sie ihn um, die versiegelte Rückseite (nicht die freiliegende Leiterplatte) ist die, die wir verwenden wollen.

Der Summer benötigt beim Betrieb einen Transistor, hier verwenden wir S8050 (NPN-Transistor).



Code

Bemerkung:

- Öffnen Sie die Datei `3.2_custom_tone.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time

# Define the frequencies of several musical notes in Hz
C4 = 262
D4 = 294
E4 = 330
F4 = 349
G4 = 392
A4 = 440
B4 = 494

# Create a PWM object representing pin 14 and assign it to the buzzer variable
buzzer = machine.PWM(machine.Pin(14))

# Define a tone function that takes as input a Pin object representing the buzzer, a␣
↪frequency in Hz, and a duration in milliseconds
def tone(pin, frequency, duration):
    pin.freq(frequency) # Set the frequency
    pin.duty(512) # Set the duty cycle
    time.sleep_ms(duration) # Pause for the duration in milliseconds
    pin.duty(0) # Set the duty cycle to 0 to stop the tone

# Play a sequence of notes with different frequency inputs and durations
tone(buzzer, C4, 250)
time.sleep_ms(500)
tone(buzzer, D4, 250)
time.sleep_ms(500)
tone(buzzer, E4, 250)
time.sleep_ms(500)
tone(buzzer, F4, 250)
time.sleep_ms(500)
tone(buzzer, G4, 250)
time.sleep_ms(500)
tone(buzzer, A4, 250)
time.sleep_ms(500)
tone(buzzer, B4, 250)
```

Wie funktioniert das?

Wenn der passive Summer ein digitales Signal erhält, kann er nur das Zwerchfell bewegen, ohne einen Ton zu erzeugen.

Deshalb verwenden wir die Funktion `tone()`, um das PWM-Signal zu erzeugen, damit der passive Summer klingt.

Diese Funktion hat drei Parameter:

- `pin`: Der Pin, der den Summer steuert.
- `frequency`: Die Tonhöhe des Summers wird durch die Frequenz bestimmt, je höher die Frequenz, desto höher

die Tonhöhe.

- `Duration`: Die Dauer des Tons.

Wir verwenden die Funktion `duty()` um den Tastgrad auf 512 (etwa 50%) einzustellen. Es können auch andere Zahlen sein, es muss nur ein diskontinuierliches elektrisches Signal erzeugt werden, um zu oszillieren.

Mehr erfahren

Wir können bestimmte Tonhöhen simulieren und so ein komplettes Musikstück spielen.

Bemerkung:

- Öffnen Sie die Datei `3.2_custom_tone_music.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time

# Define the GPIO pin that is connected to the buzzer
buzzer = machine.PWM(machine.Pin(14))

# Define the frequencies of the notes in Hz
C5 = 523
D5 = 587
E5 = 659
F5 = 698
G5 = 784
A5 = 880
B5 = 988

# Define the durations of the notes in milliseconds
quarter_note = 250
half_note = 300
whole_note = 1000

# Define the melody as a list of tuples (note, duration)
melody = [
    (E5, quarter_note),
    (E5, quarter_note),
    (F5, quarter_note),
    (G5, half_note),
    (G5, quarter_note),
    (F5, quarter_note),
    (E5, quarter_note),
    (D5, half_note),
    (C5, quarter_note),
    (C5, quarter_note),
    (D5, quarter_note),
    (E5, half_note),
    (E5, quarter_note),
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    (D5, quarter_note),
    (D5, half_note),
    (E5, quarter_note),
    (E5, quarter_note),
    (F5, quarter_note),
    (G5, half_note),
    (G5, quarter_note),
    (F5, quarter_note),
    (E5, quarter_note),
    (D5, half_note),
    (C5, quarter_note),
    (C5, quarter_note),
    (D5, quarter_note),
    (E5, half_note),
    (D5, quarter_note),
    (C5, quarter_note),
    (C5, half_note),

]

# Define a function to play a note with the given frequency and duration
def tone(pin,frequency,duration):
    pin.freq(frequency)
    pin.duty(512)
    time.sleep_ms(duration)
    pin.duty(0)

# Play the melody
for note in melody:
    tone(buzzer, note[0], note[1])
    time.sleep_ms(50)
```

- Die Funktion `tone` setzt die Frequenz des Pins auf den Wert von `frequency` unter Verwendung der `freq`-Methode des `pin`-Objekts.
- Anschließend setzt sie den Tastgrad des Pins auf 512 unter Verwendung der `duty`-Methode des `pin`-Objekts.
- Dadurch erzeugt der Pin einen Ton mit der angegebenen Frequenz und Lautstärke für die Dauer von `duration` in Millisekunden unter Verwendung der `sleep_ms`-Methode des Zeitmoduls.
- Der Code spielt dann eine Melodie ab, indem er durch eine Sequenz namens `melody` iteriert und für jede Note in der Melodie die Funktion `tone` mit der Frequenz und Dauer der Note aufruft.
- Zwischen jeder Note wird auch eine kurze Pause von 50 Millisekunden unter Verwendung der `sleep_ms`-Methode des Zeitmoduls eingefügt.

4. Aktoren

3.16 4.1 Kleiner Ventilator

In diesem spannenden Projekt werden wir erforschen, wie man einen Motor mit dem L293D steuert.

Der L293D ist ein vielseitiger integrierter Schaltkreis (IC), der häufig für die Motorsteuerung in Elektronik- und Robotikprojekten verwendet wird. Er kann zwei Motoren in Vorwärts- und Rückwärtsrichtung antreiben, was ihn zu einer beliebten Wahl für Anwendungen macht, die eine präzise Motorsteuerung erfordern.

Am Ende dieses fesselnden Projekts werden Sie ein gründliches Verständnis dafür haben, wie digitale Signale und PWM-Signale effektiv genutzt werden können, um Motoren zu steuern. Dieses wertvolle Wissen wird eine solide Grundlage für Ihre zukünftigen Unternehmungen in der Robotik und Mechatronik sein. Schnallen Sie sich an und machen Sie sich bereit, in die aufregende Welt der Motorsteuerung mit dem L293D einzutauchen!

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch einzeln über die untenstehenden Links kaufen.

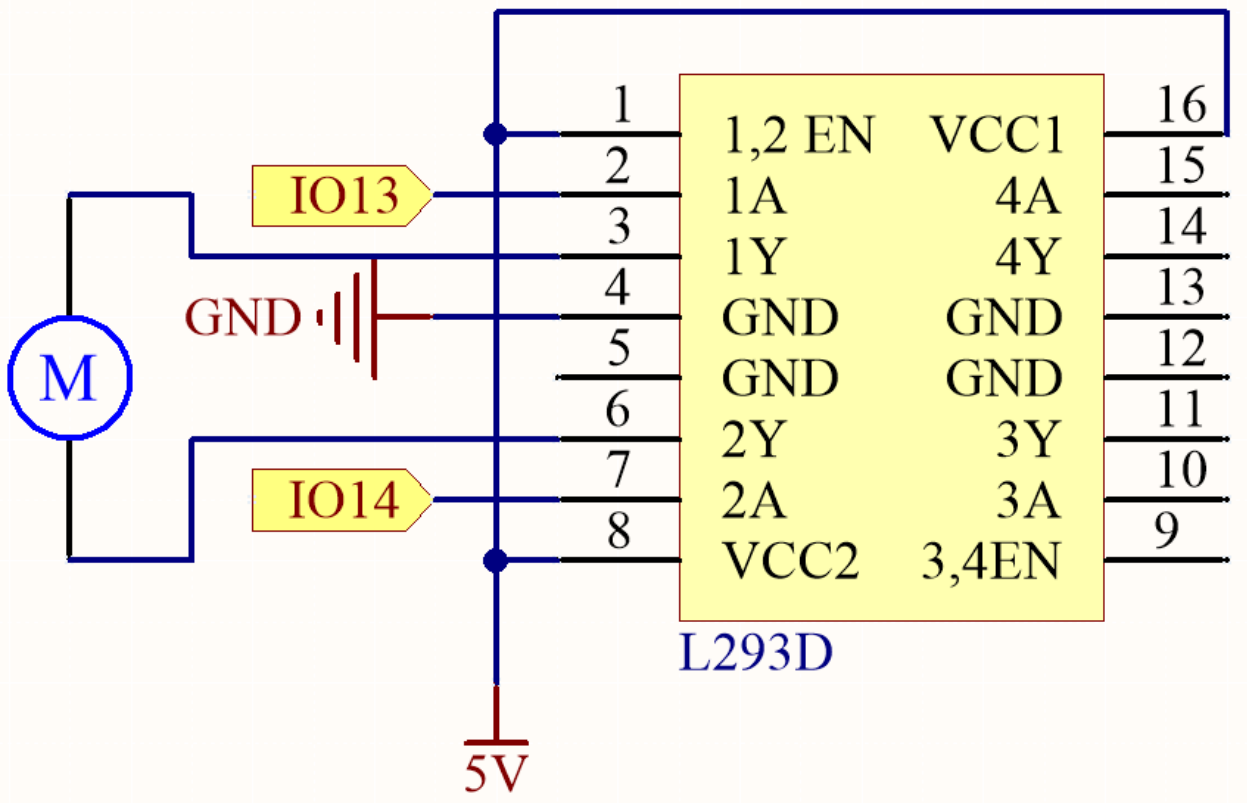
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Gleichstrommotor	
L293D	-

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

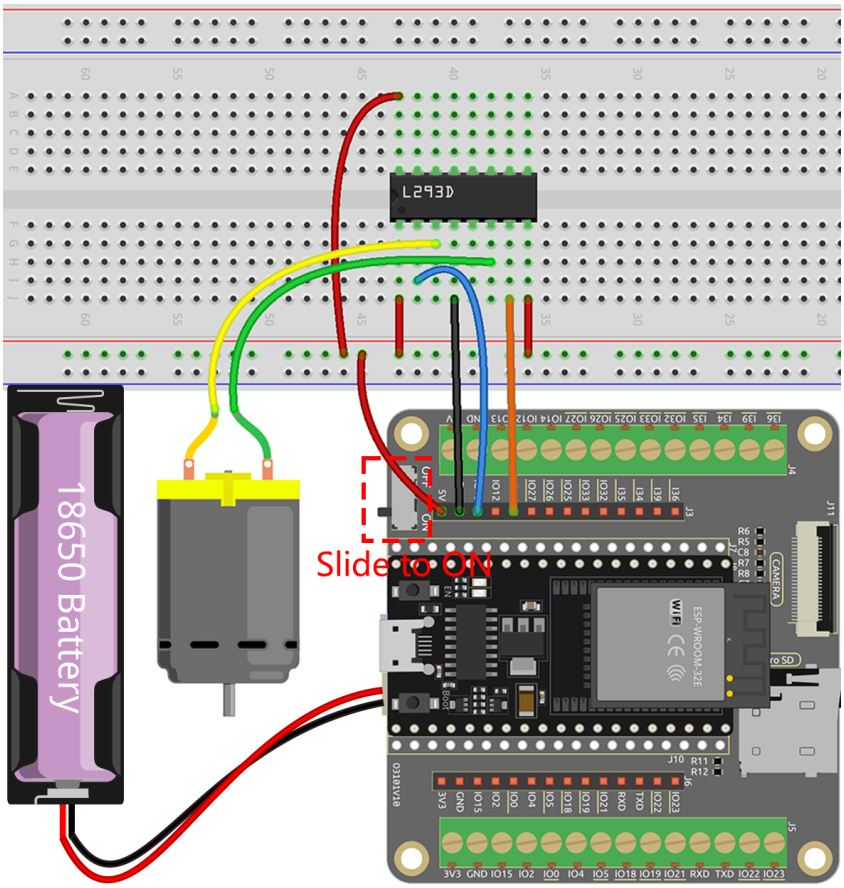
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Verdrahtung

Bemerkung: Da der Motor einen relativ hohen Strom benötigt, ist es notwendig, zuerst die Batterie einzulegen und dann den Schalter auf dem Erweiterungsboard auf die Position ON zu schieben, um die Batterieversorgung zu aktivieren.



Code

Bemerkung:

- Öffnen Sie die Datei `4.1_motor_turn.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time

# Create Pin objects representing the motor control pins and set them to output mode
motor1A = machine.Pin(13, machine.Pin.OUT)
motor2A = machine.Pin(14, machine.Pin.OUT)

# Define a function to rotate the motor clockwise
def clockwise():
    motor1A.value(1)
    motor2A.value(0)

# Define a function to rotate the motor anticlockwise
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
def anticlockwise():
    motor1A.value(0)
    motor2A.value(1)

# Define a function to stop the motor
def stop():
    motor1A.value(0)
    motor2A.value(0)

# Enter an infinite loop

try:
    while True:
        clockwise() # Rotate the motor clockwise
        time.sleep(1) # Pause for 1 second
        anticlockwise() # Rotate the motor anticlockwise
        time.sleep(1)
        stop() # Stop the motor
        time.sleep(2)

except KeyboardInterrupt:
    stop()  # Stop the motor when KeyboardInterrupt is caught
```

Während der Skriptausführung werden Sie sehen, wie der Motor abwechselnd jede Sekunde im Uhrzeigersinn und gegen den Uhrzeigersinn dreht.

Mehr erfahren

Zusätzlich zum einfachen Drehen des Motors im Uhrzeigersinn und gegen den Uhrzeigersinn können Sie auch die Geschwindigkeit der Motorrotation steuern, indem Sie die Pulsbreitenmodulation (PWM) am Steuerpin verwenden, wie unten gezeigt.

Bemerkung:

- Öffnen Sie die Datei `4.1_motor_turn_pwm.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
from machine import Pin, PWM
import time

# Create PWM objects representing the motor control pins and set their frequency to 1000␣
↪Hz
motor1A = PWM(Pin(13, Pin.OUT))
motor2A = PWM(Pin(14, Pin.OUT))
motor1A.freq(500)
motor2A.freq(500)

# Enter an infinite loop
while True:
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    # Rotate the motor forward by gradually increasing the power on the motor1A pin
    for power in range(0, 1023, 20):
        motor1A.duty(power)
        motor2A.duty(0)
        time.sleep(0.1)
    # Decreasing the power on the motor1A pin
    for power in range(1023, 0, -20):
        motor1A.duty(power)
        motor2A.duty(0)
        time.sleep(0.1)
    # Rotate the motor in the opposite direction by gradually increasing the power on␣
↪the motor2A pin
    for power in range(0, 1023, 20):
        motor1A.duty(0)
        motor2A.duty(power)
        time.sleep(0.1)
    # Decreasing the power on the motor2A pin
    for power in range(1023, 0, -20):
        motor1A.duty(0)
        motor2A.duty(power)
        time.sleep(0.1)
```

Im Gegensatz zum vorherigen Skript wird hier der Motor durch PWM-Signale mit einer Frequenz von 1000 Hz gesteuert, die die Geschwindigkeit des Motors bestimmt.

- Der Code verwendet eine `while True`-Schleife, um kontinuierlich zu laufen. Innerhalb der Schleife gibt es vier `for`-Schleifen, die die Motoren in einer Sequenz steuern.
- Die ersten beiden `for`-Schleifen erhöhen und verringern die Geschwindigkeit von IN1, während IN2 auf 0 Geschwindigkeit gehalten wird.
- Die nächsten beiden `for`-Schleifen erhöhen und verringern die Geschwindigkeit von IN2, während IN1 auf 0 Geschwindigkeit gehalten wird.
- Die `range`-Funktion in jeder `for`-Schleife erzeugt eine Reihe von Zahlen, die als Tastgrad des PWM-Signals dient. Dies wird dann über die `duty`-Methode an IN1 oder IN2 ausgegeben. Der Tastgrad bestimmt den Prozentsatz der Zeit, in der das PWM-Signal hoch ist, was wiederum die durchschnittliche Spannung bestimmt, die am Motor anliegt, und damit die Motorgeschwindigkeit.
- Die Funktion `time.sleep` wird verwendet, um eine Verzögerung von 0,1 Sekunden zwischen jedem Schritt in der Sequenz einzuführen, was es dem Motor ermöglicht, die Geschwindigkeit allmählich zu ändern, anstatt sofort von einer Geschwindigkeit zur anderen zu springen.

3.17 4.2 Pumpen

In diesem faszinierenden Projekt werden wir uns damit beschäftigen, eine Wasserpumpe mit dem L293D zu steuern.

Bei der Steuerung von Wasserpumpen sind die Dinge im Vergleich zur Steuerung anderer Motoren etwas einfacher. Die Schönheit dieses Projekts liegt in seiner Einfachheit - es besteht keine Notwendigkeit, sich um die Drehrichtung zu kümmern. Unser Hauptziel ist es, die Wasserpumpe erfolgreich zu aktivieren und in Betrieb zu halten.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch einzeln über die untenstehenden Links kaufen.

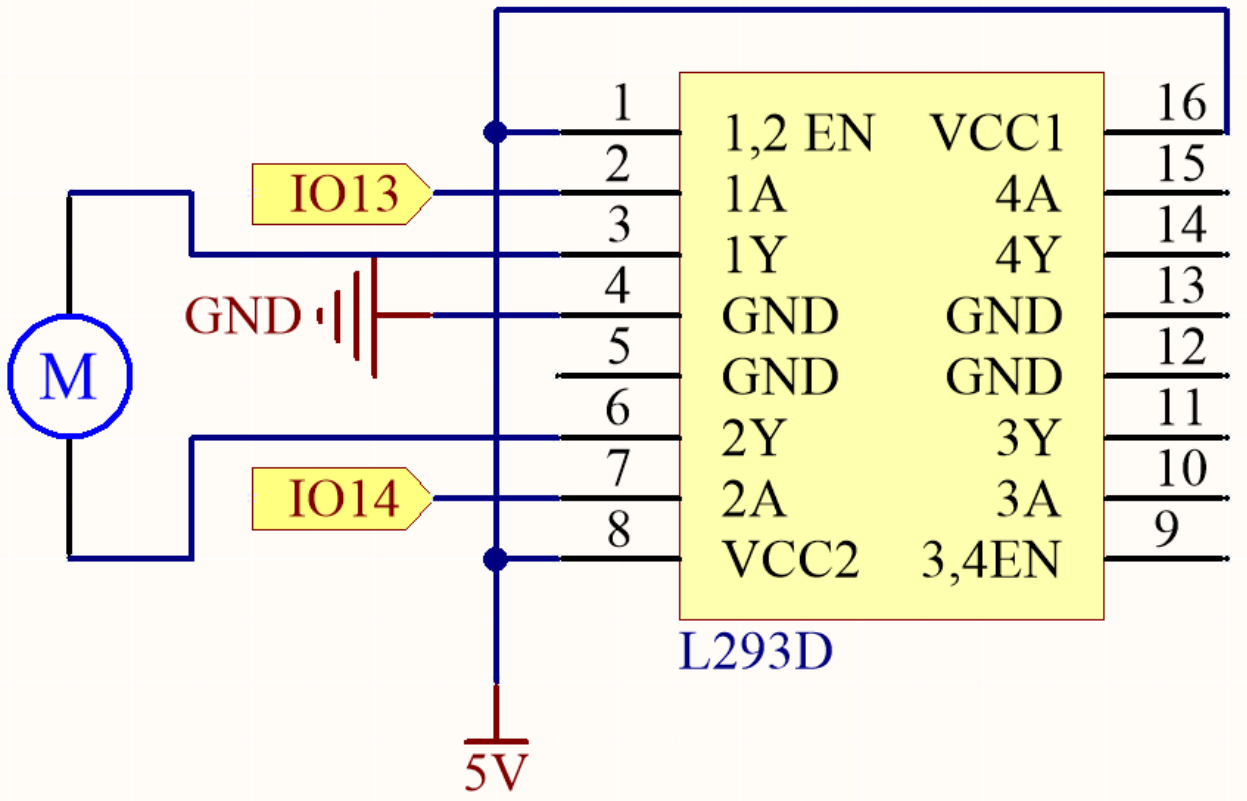
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Zentrifugalpumpe	-
L293D	-

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

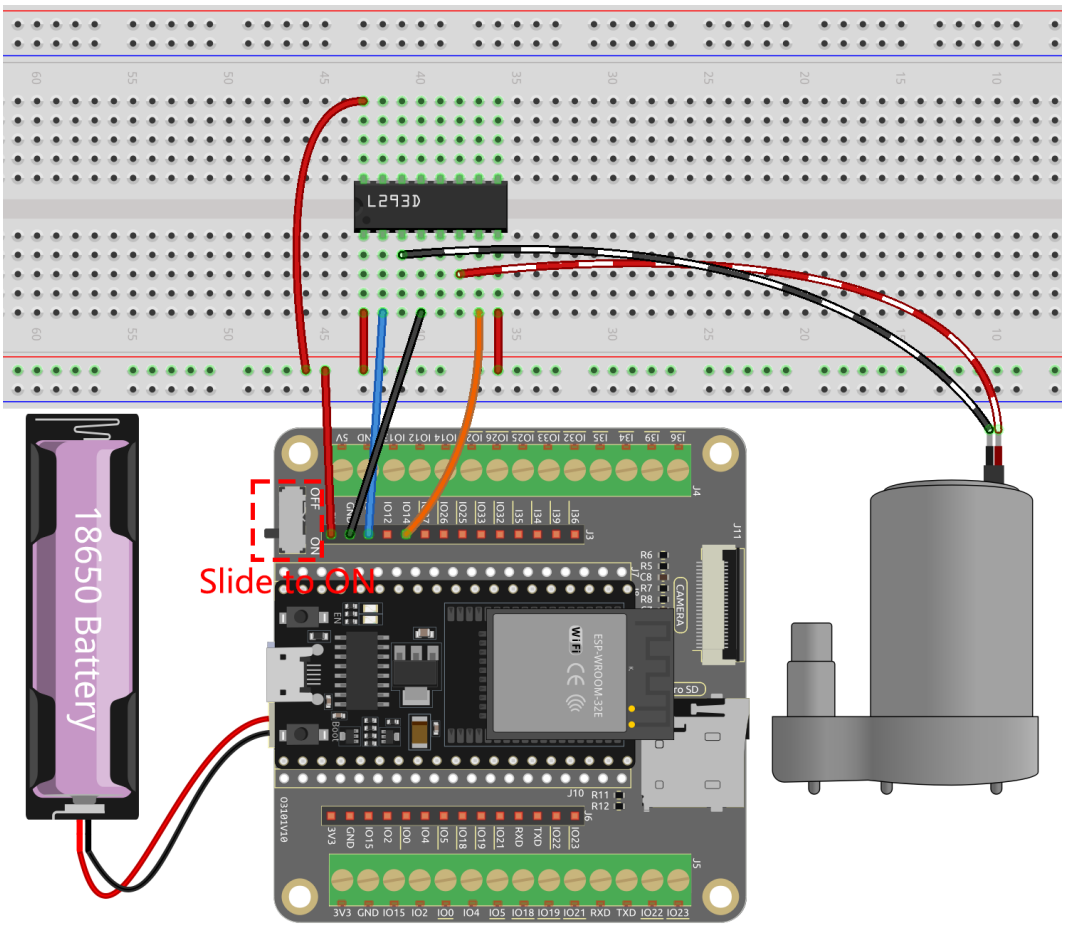
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Verdrahtung

Bemerkung: Hier wird empfohlen, zuerst die Batterie einzulegen und dann den Schalter auf dem Erweiterungsboard auf die Position ON zu schieben, um die Batterieversorgung zu aktivieren.



Code

Bemerkung:

- Öffnen Sie die Datei `4.2_pumping.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time

# Create Pin objects representing the motor control pins and set them to output mode
motor1A = machine.Pin(13, machine.Pin.OUT)
motor2A = machine.Pin(14, machine.Pin.OUT)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
# Define a function to rotate the pump
def rotate():
    motor1A.value(1)
    motor2A.value(0)

# Define a function to stop the pump
def stop():
    motor1A.value(0)
    motor2A.value(0)

try:
    while True:
        rotate()  # Rotate the motor clockwise
        time.sleep(5)  # Pause for 5 seconds
        stop()  # Stop the motor
        time.sleep(2)

except KeyboardInterrupt:
    stop()  # Stop the motor when KeyboardInterrupt is caught
```

Während der Skriptausführung werden Sie sehen, wie die Pumpe arbeitet und Wasser aus dem Schlauch kommt, dann für 2 Sekunden stoppt, bevor sie wieder zu arbeiten beginnt.

3.18 4.3 Schwingender Servomotor

Ein Servomotor ist eine Art positionsbasiertes Gerät, das für seine Fähigkeit bekannt ist, spezifische Winkel zu halten und präzise Drehungen zu liefern. Dies macht ihn besonders wünschenswert für Steuerungssysteme, die konstante Winkelverstellungen verlangen. Es ist nicht überraschend, dass Servomotoren in hochwertigen ferngesteuerten Spielzeugen weit verbreitet sind, von Flugzeugmodellen bis hin zu U-Boot-Repliken und ausgeklügelten ferngesteuerten Robotern.

In diesem faszinierenden Abenteuer werden wir uns selbst herausfordern, den Servomotor auf eine einzigartige Weise zu manipulieren - indem wir ihn schwingen lassen! Dieses Projekt bietet eine hervorragende Gelegenheit, tiefer in die Dynamik von Servomotoren einzutauchen, Ihre Fähigkeiten in präzisen Steuerungssystemen zu schärfen und ein tieferes Verständnis für ihre Funktionsweise zu erlangen.

Sind Sie bereit, den Servomotor nach Ihren Melodien tanzen zu lassen? Lassen Sie uns auf diese spannende Reise gehen!

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch einzeln über die untenstehenden Links kaufen.

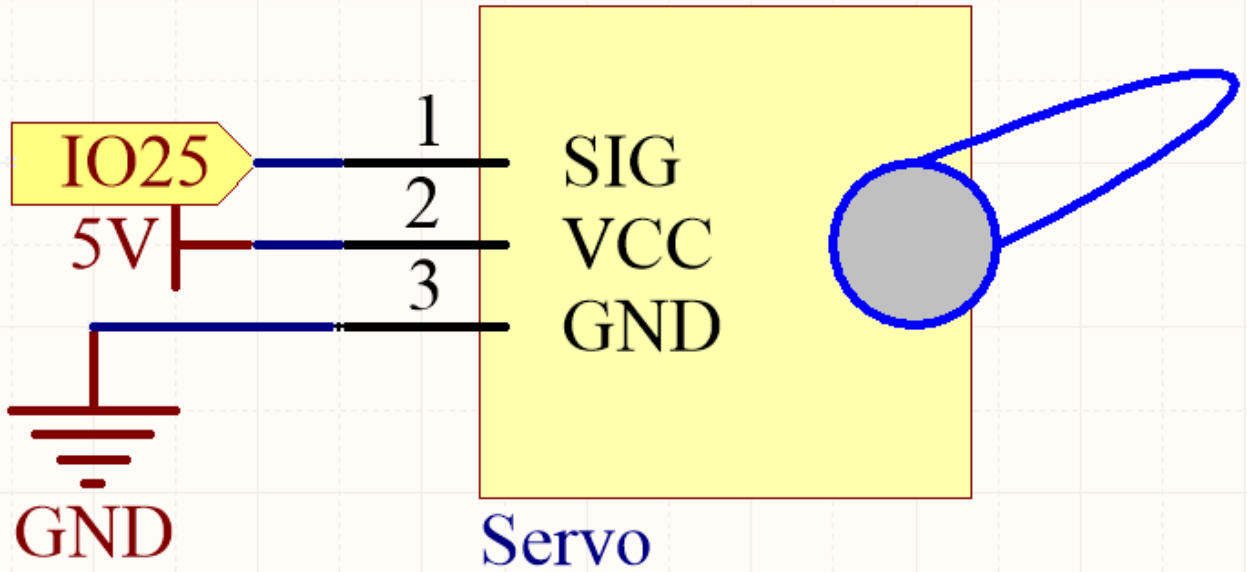
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Servo	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

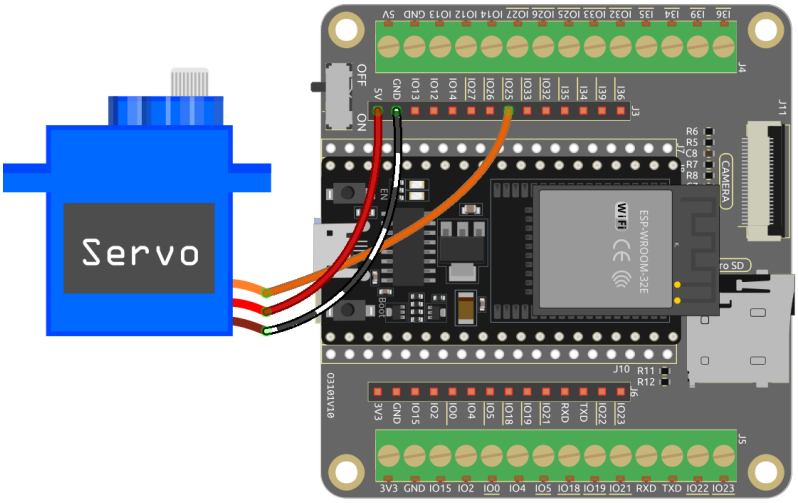
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Verdrahtung

- Das orangefarbene Kabel ist das Signal und wird an IO25 angeschlossen.
- Das rote Kabel ist VCC und wird an 5V angeschlossen.
- Das braune Kabel ist GND und wird an GND angeschlossen.



Code

Bemerkung:

- Öffnen Sie die Datei `4.3_swinging_servo.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time

# Create a PWM (Pulse Width Modulation) object on Pin 25
servo = machine.PWM(machine.Pin(25))

# Set the frequency of the PWM signal to 50 Hz, common for servos
servo.freq(50)

# Define a function for interval mapping
def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

# Define a function to write an angle to the servo
def servo_write(pin, angle):

    pulse_width = interval_mapping(angle, 0, 180, 0.5, 2.5) # Calculate the pulse width
    duty = int(interval_mapping(pulse_width, 0, 20, 0, 1023))     # Calculate the duty␣
↪cycle
    pin.duty(duty) # Set the duty cycle of the PWM signal

# Create an infinite loop
while True:
    # Loop through angles from 0 to 180 degrees
    for angle in range(180):
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        servo_write(servo, angle)
        time.sleep_ms(20)

    # Loop through angles from 180 to 0 degrees in reverse
    for angle in range(180, -1, -1):
        servo_write(servo, angle)
        time.sleep_ms(20)
```

Wenn Sie diesen Code ausführen, wird der Servomotor kontinuierlich zwischen 0 und 180 Grad hin und her schwenken.

Wie funktioniert das?

1. Importieren Sie die notwendigen Bibliotheken: `machine` zur Steuerung der Hardware des Mikrocontrollers und `time` für das Hinzufügen von Verzögerungen.

   ```
   import machine
   import time
   ```

2. Erstellen Sie ein PWM-Objekt (Pulsweitenmodulation) am Pin 25 und setzen Sie dessen Frequenz auf 50 Hz, was für Servos üblich ist.

   ```
   # Create a PWM (Pulse Width Modulation) object on Pin 25
   servo = machine.PWM(machine.Pin(25))

   # Set the frequency of the PWM signal to 50 Hz, common for servos
   servo.freq(50)
   ```

3. Definieren Sie eine `interval_mapping`-Funktion, um Werte von einem Bereich in einen anderen zu übertragen. Diese wird verwendet, um den Winkel in die entsprechende Impulsbreite und den Tastgrad umzurechnen.

   ```
   def interval_mapping(x, in_min, in_max, out_min, out_max):
       return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
   ```

4. Definieren Sie eine `servo_write`-Funktion, die ein PWM-Objekt und einen Winkel als Eingaben nimmt. Sie berechnet die Impulsbreite und den Tastgrad basierend auf dem gegebenen Winkel und setzt dann die PWM-Ausgabe entsprechend.

   ```
   def servo_write(pin, angle):

       pulse_width = interval_mapping(angle, 0, 180, 0.5, 2.5) # Calculate the␣
   ↪pulse width
       duty = int(interval_mapping(pulse_width, 0, 20, 0, 1023))     #␣
   ↪Calculate the duty cycle
       pin.duty(duty) # Set the duty cycle of the PWM signal
   ```

   - In dieser Funktion wird `interval_mapping()` aufgerufen, um den Winkelbereich 0 ~ 180 auf den Impulsbreitenbereich 0,5 ~ 2,5 ms abzubilden.
   - Warum ist es 0,5~2,5? Dies wird durch den Arbeitsmodus des *Servo* bestimmt.
   - Als nächstes wird die Impulsbreite von der Periode in den Tastgrad umgewandelt.
   - Da `duty()` bei Verwendung keine Dezimalzahlen haben darf (der Wert darf kein Float-Typ sein), haben wir `int()` verwendet, um den Tastgrad in einen Int-Typ zu konvertieren.

5. Erstellen Sie eine unendliche Schleife mit zwei verschachtelten Schleifen.

```
while True:
    # Loop through angles from 0 to 180 degrees
    for angle in range(180):
        servo_write(servo, angle)
        time.sleep_ms(20)

    # Loop through angles from 180 to 0 degrees in reverse
    for angle in range(180, -1, -1):
        servo_write(servo, angle)
        time.sleep_ms(20)
```

- Die erste verschachtelte Schleife iteriert durch Winkel von 0 bis 180 Grad, und die zweite verschachtelte Schleife iteriert durch Winkel von 180 bis 0 Grad in umgekehrter Reihenfolge.
- In jeder Iteration wird die Funktion `servo_write` mit dem aktuellen Winkel aufgerufen, und es wird eine Verzögerung von 20 Millisekunden hinzugefügt.

5. Sensoren

3.19 5.1 Lesen des Tastenwertes

In diesem interaktiven Projekt wagen wir uns in die Welt der Tastensteuerung und LED-Manipulation.

Das Konzept ist unkompliziert, aber effektiv. Wir werden den Zustand einer Taste lesen. Wenn die Taste gedrückt wird, registriert sie ein hohes Spannungsniveau, oder einen ‚hohen Zustand'. Diese Aktion wird dann dazu führen, dass eine LED aufleuchtet.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
Taste	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Für Eingang	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Für Ausgang	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Bedingte Nutzung Pins (Eingang)**

  Die folgenden Pins haben eingebaute Pull-up- oder Pull-down-Widerstände, daher sind externe Widerstände nicht erforderlich, wenn **sie als Eingangspins verwendet werden**:

Bedingte Nutzung Pins	Beschreibung
IO13, IO15, IO2, IO4	Pull-up mit einem 47K-Widerstand setzt den Wert standardmäßig auf hoch.
IO27, IO26, IO33	Pull-up mit einem 4,7K-Widerstand setzt den Wert standardmäßig auf hoch.
IO32	Pull-down mit einem 1K-Widerstand setzt den Wert standardmäßig auf niedrig.

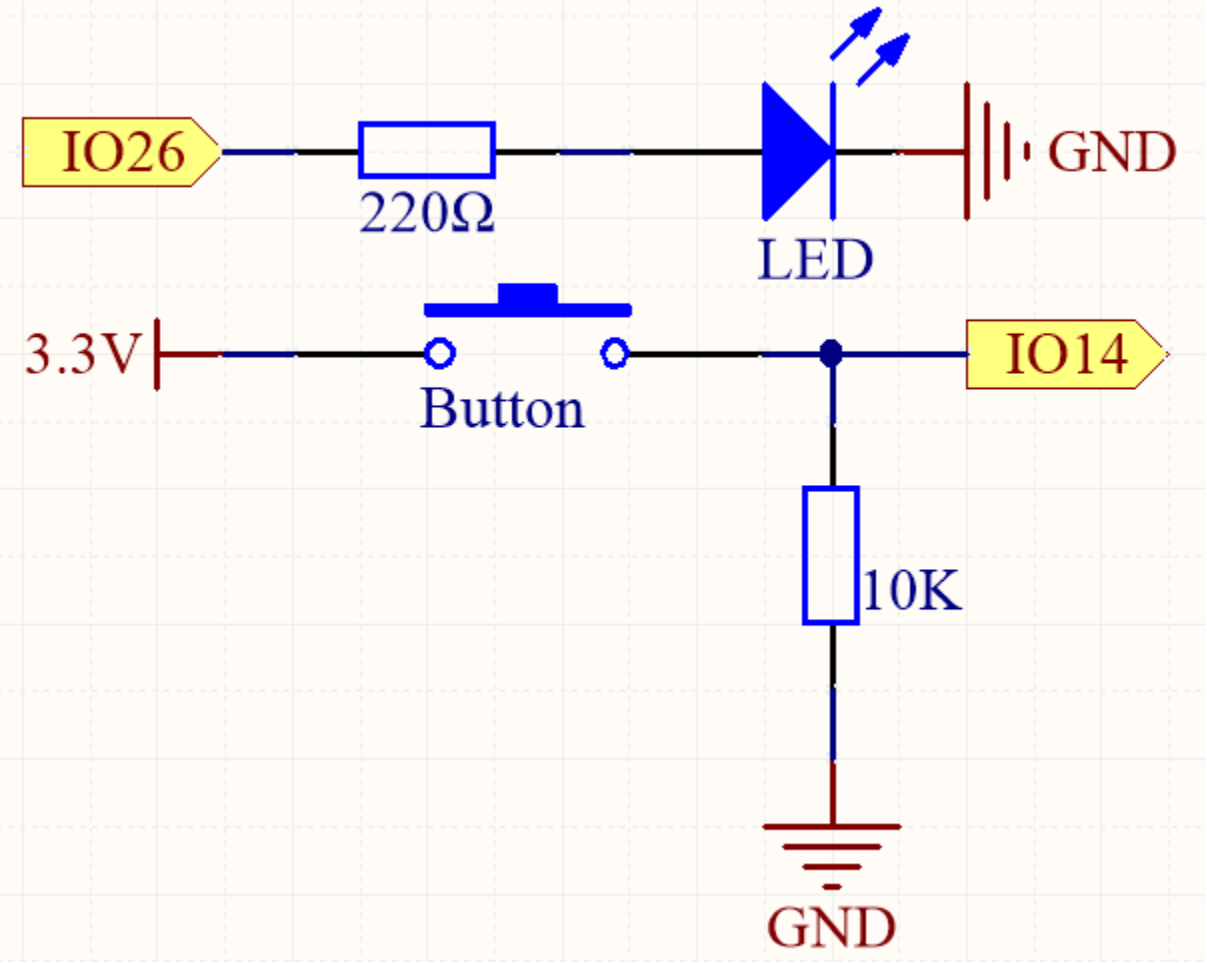
- **Strapping Pins (Eingang)**

  Strapping Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um bestimmte Boot-Modi während des Startvorgangs des Geräts (d. h. beim Einschalten) festzulegen.

Strapping Pins	IO5, IO0, IO2, IO12, IO15

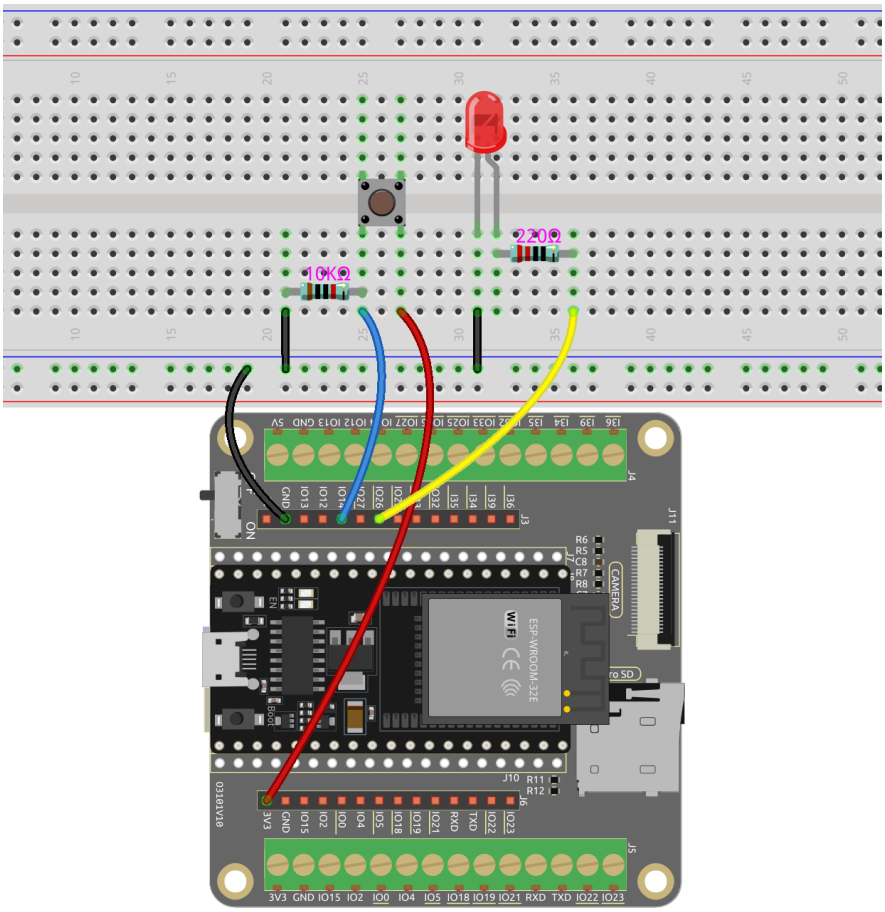
  Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, bedenken Sie die möglichen Auswirkungen auf den Bootvorgang. Für weitere Details siehe den Abschnitt *Strapping-Pins*.

Schaltplan



Um eine korrekte Funktionalität zu gewährleisten, verbinden Sie eine Seite des Tastenpins mit 3,3V und die andere Seite mit IO14. Wenn die Taste gedrückt wird, wird IO14 auf hoch gesetzt, was dazu führt, dass die LED aufleuchtet. Wenn die Taste losgelassen wird, kehrt IO14 in seinen schwebenden Zustand zurück, der entweder hoch oder niedrig sein kann. Um ein stabiles niedriges Niveau zu gewährleisten, wenn die Taste nicht gedrückt ist, sollte IO14 über einen 10K-Pull-down-Widerstand mit GND verbunden werden.

Verdrahtung



Bemerkung: Ein vierpoliger Taster ist in H-Form gestaltet. Wenn der Taster nicht gedrückt wird, sind die links und rechts Pins getrennt und der Strom kann zwischen ihnen nicht fließen. Wenn der Taster jedoch gedrückt wird, sind die links und rechts Pins verbunden, was einen Stromweg ermöglicht.

Code

Bemerkung:

- Öffnen Sie die Datei `5.1_read_button_value.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time

button = machine.Pin(14, machine.Pin.IN) # Button pin
led = machine.Pin(26, machine.Pin.OUT) # LED pin
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
while True:
    # If the button is pressed by reading its value
    if button.value() == 1:
        # Turn on the LED by setting its value to 1
        led.value(1)
#         time.sleep(0.5)
    else:
        # Turn off the LED
        led.value(0)
```

Während der Skriptausführung leuchtet die LED auf, wenn Sie den Taster drücken, und erlischt, wenn Sie ihn loslassen.

3.20 5.2 Kippen Sie es!

Der Kippschalter ist ein einfaches, aber effektives 2-Pin-Gerät, das eine Metallkugel in seinem Zentrum enthält. Wenn der Schalter in einer aufrechten Position ist, sind die beiden Pins elektrisch verbunden, was den Stromfluss ermöglicht. Kippt der Schalter jedoch oder wird in einem bestimmten Winkel geneigt, bewegt sich die Metallkugel und unterbricht die elektrische Verbindung zwischen den Pins.

In diesem Projekt werden wir den Kippschalter nutzen, um die Beleuchtung einer LED zu steuern. Indem wir den Schalter so positionieren, dass die Kippaktion ausgelöst wird, können wir die LED basierend auf der Ausrichtung des Schalters ein- und ausschalten.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	ITEMS IN THIS KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
Neigungsschalter	-

Verfügbare Pins

- **Verfügbare Pins**

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Für Eingang	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Für Ausgang	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Bedingte Nutzung Pins (Eingang)**

  Die folgenden Pins haben eingebaute Pull-up- oder Pull-down-Widerstände, daher sind externe Widerstände nicht erforderlich, wenn **sie als Eingangspins verwendet werden**:

Bedingte Nutzung Pins	Beschreibung
IO13, IO15, IO2, IO4	Pull-up mit einem 47K-Widerstand setzt den Wert standardmäßig auf hoch.
IO27, IO26, IO33	Pull-up mit einem 4,7K-Widerstand setzt den Wert standardmäßig auf hoch.
IO32	Pull-down mit einem 1K-Widerstand setzt den Wert standardmäßig auf niedrig.

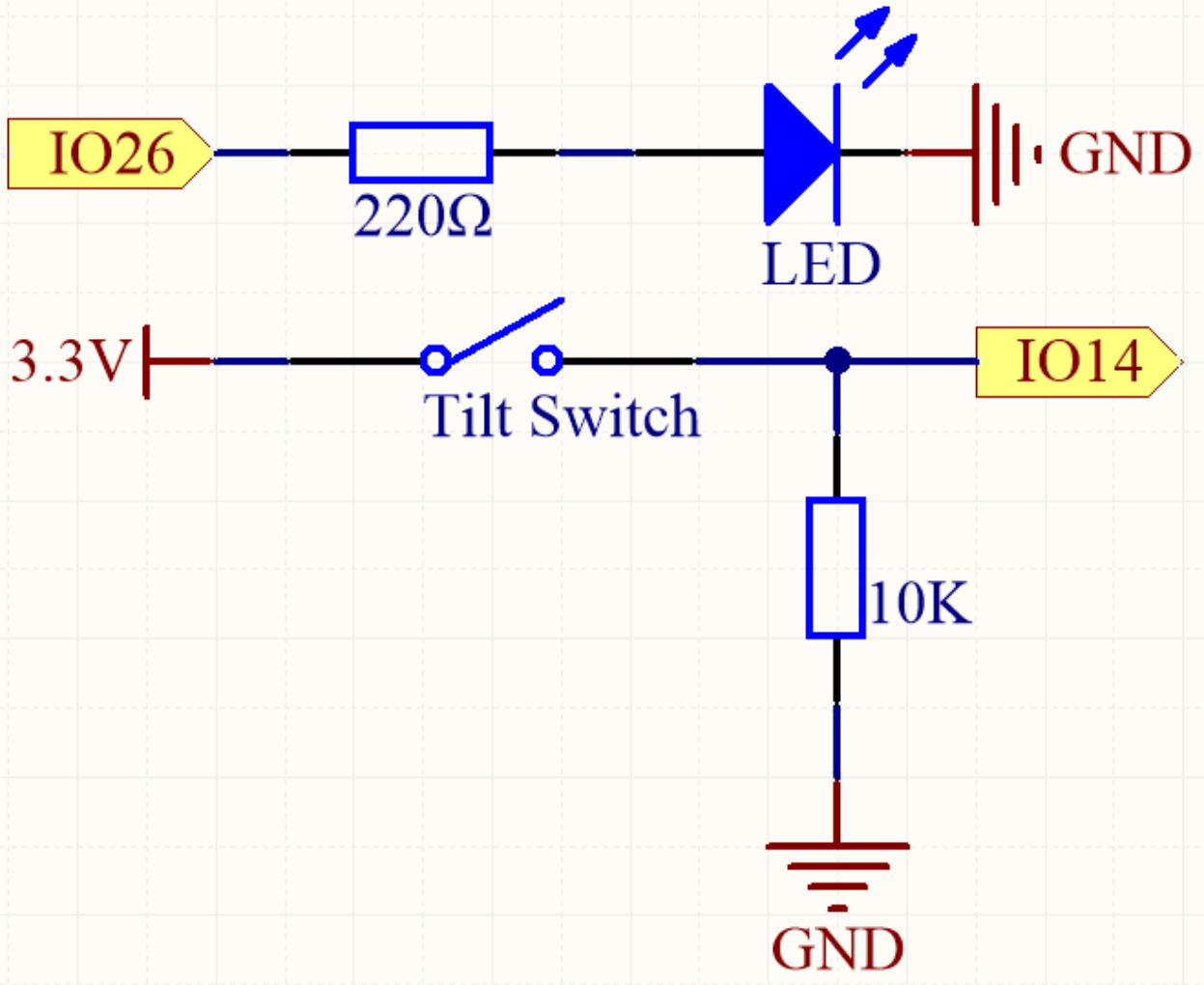
- **Strapping Pins (Eingang)**

  Strapping Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um bestimmte Boot-Modi während des Startvorgangs des Geräts (d.h. beim Einschalten) festzulegen.

Strapping Pins	IO5, IO0, IO2, IO12, IO15

  Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, bedenken Sie die möglichen Auswirkungen auf den Bootvorgang. Für weitere Details siehe den Abschnitt *Strapping-Pins*.

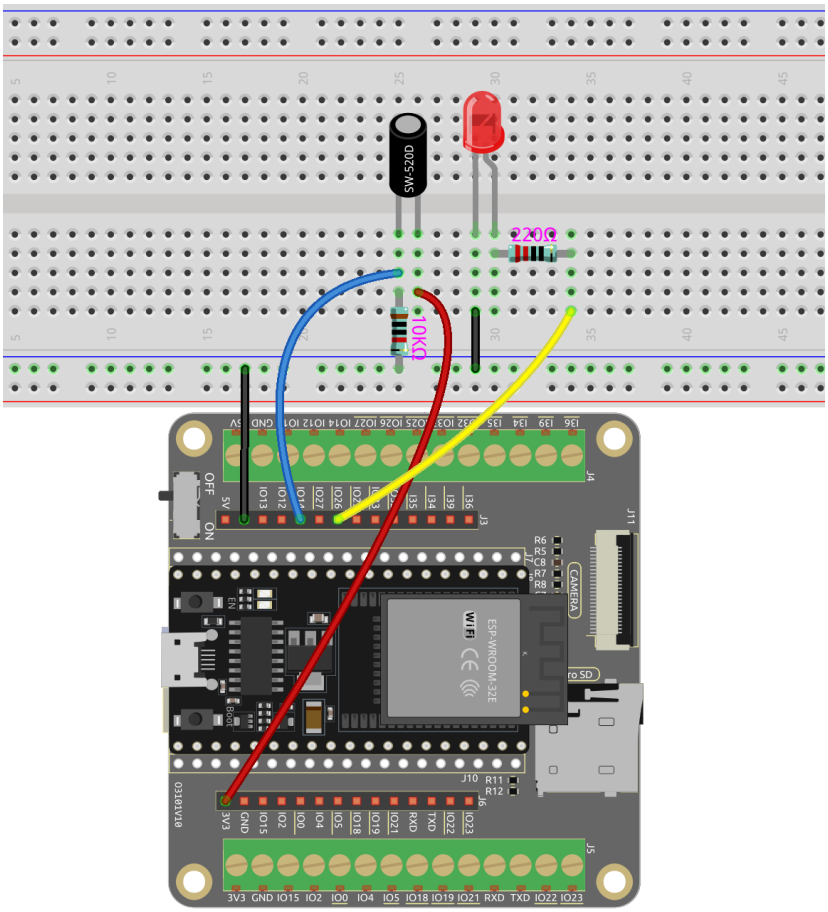
Schaltplan



Wenn der Kippschalter in einer aufrechten Position ist, wird IO14 auf hoch gesetzt, was dazu führt, dass die LED leuchtet. Umgekehrt, wenn der Kippschalter gekippt ist, wird IO14 auf niedrig gesetzt, wodurch die LED erlischt.

Der Zweck des 10K-Widerstands besteht darin, einen stabilen niedrigen Zustand für IO14 aufrechtzuerhalten, wenn der Kippschalter in einer gekippten Position ist.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.2_tilt_switch.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time

switch = machine.Pin(14, machine.Pin.IN) # Tilt switch pin
led = machine.Pin(26, machine.Pin.OUT) # LED pin

while True:
   # Check if the switch is tilted by reading its value
    if switch.value() == 1:
        # Turn on the LED by setting its value to 1
        led.value(1)
    else:
        # Turn off the LED
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        led.value(0)
```

Wenn das Skript läuft, wird die LED eingeschaltet, wenn der Schalter aufrecht ist, und ausgeschaltet, wenn der Schalter gekippt ist.

3.21 5.3 Hinderniserkennung

Dieses Modul wird üblicherweise in Autos und Robotern installiert, um die Existenz von Hindernissen voraus zu erkennen. Es wird auch häufig in Handgeräten, Wasserhähnen und ähnlichem verwendet.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Modul zur Hindernisvermeidung	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23

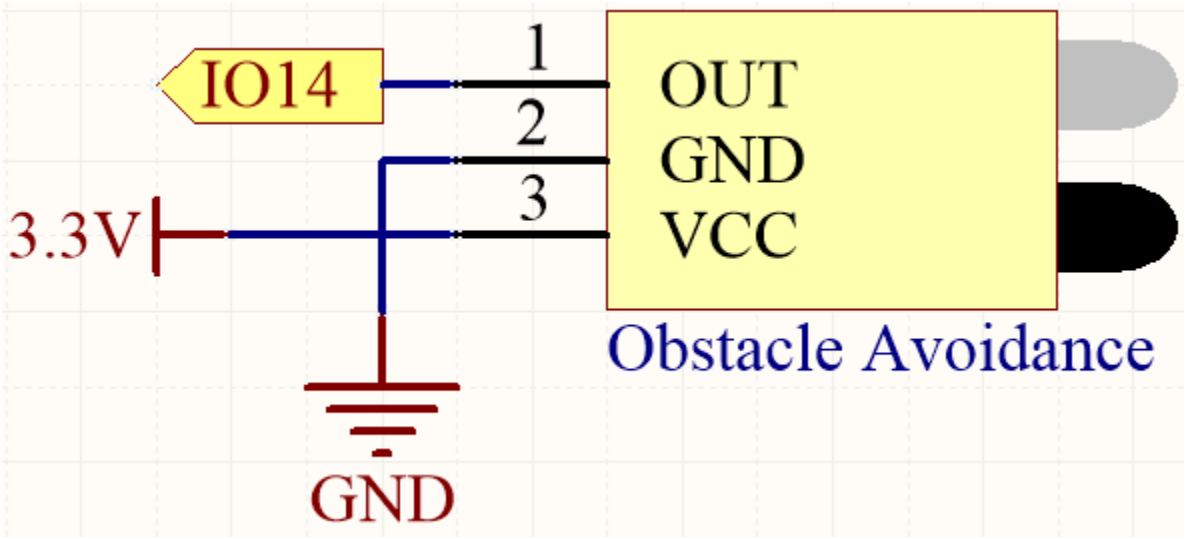
- **Strapping Pins (Eingang)**

  Strapping Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um bestimmte Boot-Modi während des Startvorgangs des Geräts (d.h., beim Einschalten) festzulegen.

Strapping Pins	IO5, IO0, IO2, IO12, IO15

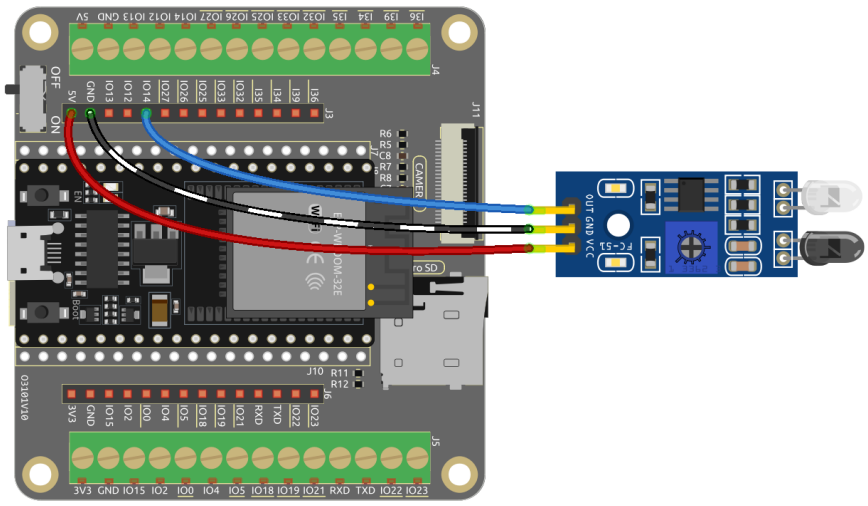
  Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, bedenken Sie die möglichen Auswirkungen auf den Bootvorgang. Für weitere Details siehe den Abschnitt *Strapping-Pins*.

Schaltplan



Wenn das Hindernisvermeidungsmodul keine Hindernisse erkennt, gibt IO14 ein hohes Signal zurück. Erkennt es jedoch ein Hindernis, wird ein niedriges Signal zurückgegeben. Sie können das blaue Potentiometer anpassen, um die Erkennungsdistanz dieses Moduls zu modifizieren.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.3_avoid.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time

ir_avoid = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_UP) # avoid module pin

while True:

    # Print values of the obstacle avoidance module
    print(ir_avoid.value())
    time.sleep(0.1)
```

Während das Programm läuft, wird auf dem Serial Monitor der Wert „0“ angezeigt, wenn das IR-Hindernisvermeidungsmodul ein Hindernis vor sich erkennt, andernfalls wird der Wert „1“ angezeigt.

3.22 5.4 Linie erkennen

Das Linienverfolgungsmodul wird verwendet, um das Vorhandensein von schwarzen Flächen auf dem Boden zu erkennen, wie beispielsweise schwarze Linien, die mit Isolierband geklebt wurden.

Sein Sender sendet geeignetes Infrarotlicht auf den Boden, das von schwarzen Oberflächen relativ absorbiert und schwach reflektiert wird. Das Gegenteil ist der Fall bei weißen Oberflächen. Wenn reflektiertes Licht erkannt wird, wird der Boden derzeit als weiß angezeigt. Wenn es nicht erkannt wird, wird es als schwarz angezeigt.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Linienverfolgungsmodul	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23

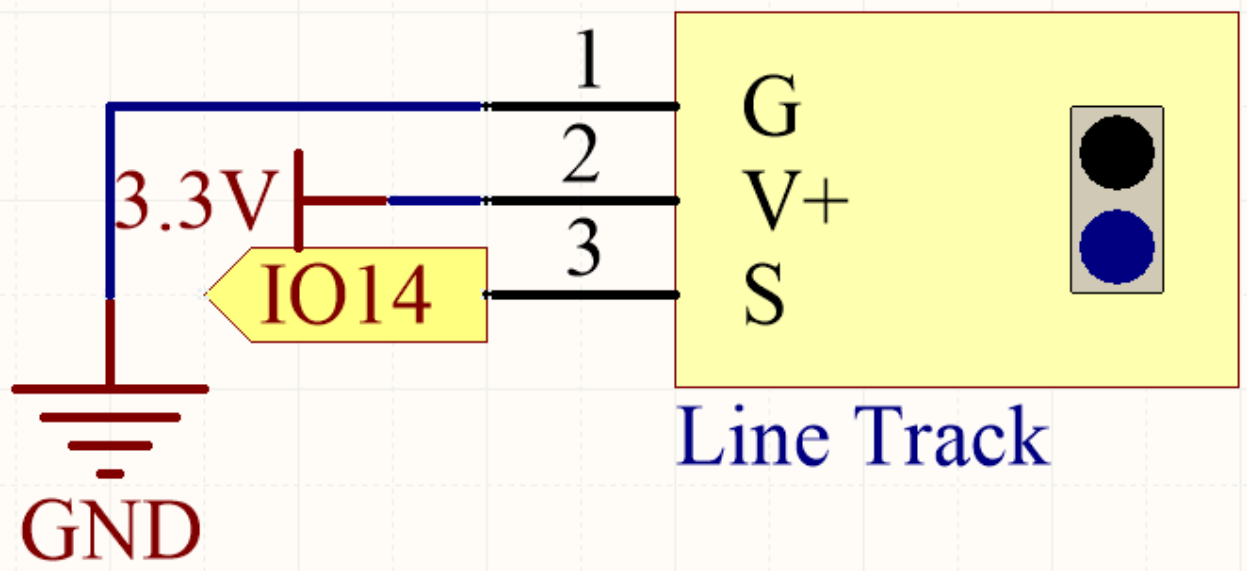
- **Strapping Pins (Eingang)**

  Strapping Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um bestimmte Boot-Modi während des Startvorgangs des Geräts (d.h., beim Einschalten) festzulegen.

Strapping Pins	IO5, IO0, IO2, IO12, IO15

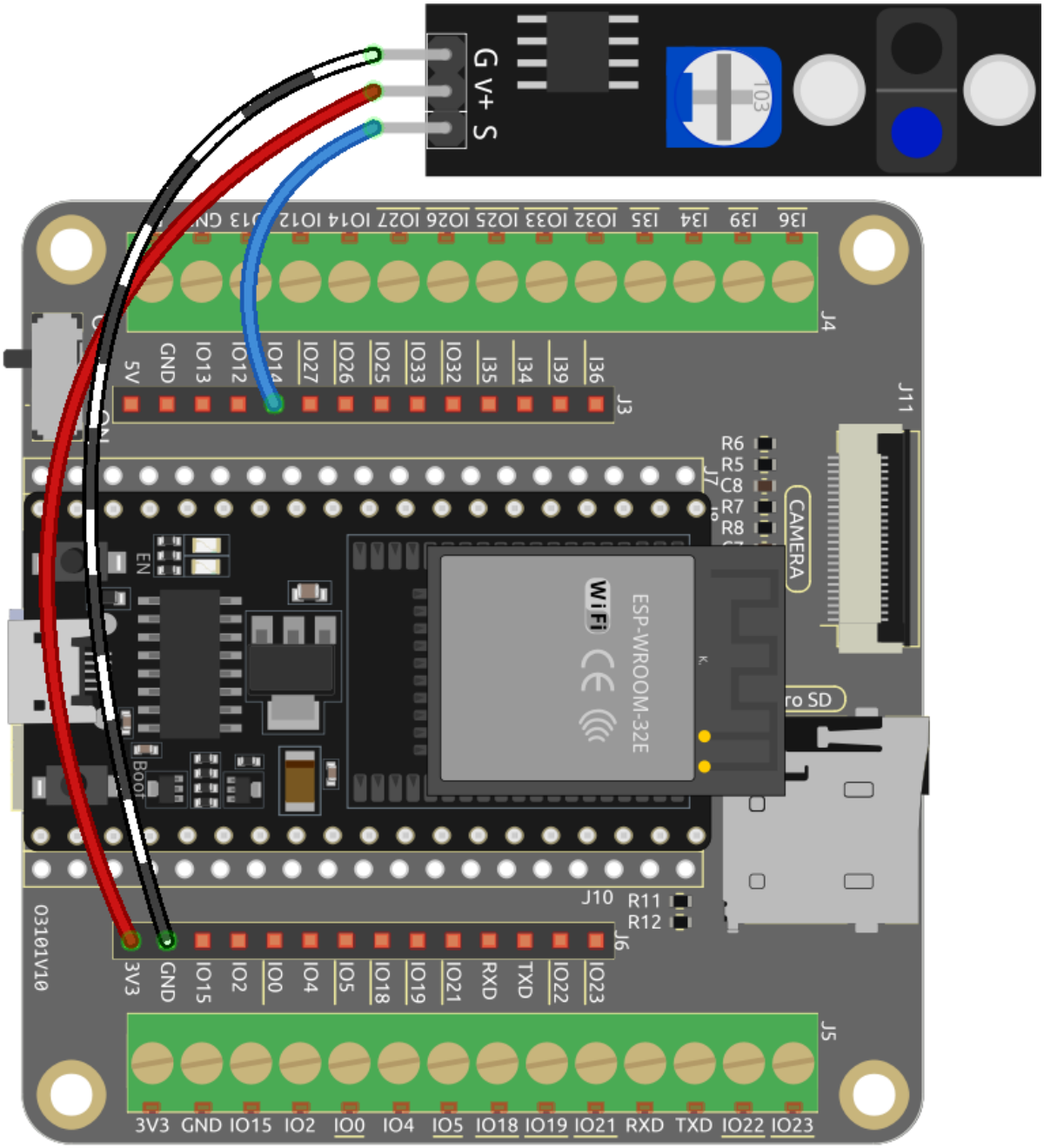
  Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, bedenken Sie die möglichen Auswirkungen auf den Bootvorgang. Für weitere Details siehe den Abschnitt *Strapping-Pins*.

Schaltplan



Wenn das Linienverfolgungsmodul eine schwarze Linie erkennt, gibt IO14 ein hohes Signal zurück. Wenn es jedoch eine weiße Linie erkennt, gibt IO14 ein niedriges Signal zurück. Sie können das blaue Potentiometer anpassen, um die Empfindlichkeit der Erkennung dieses Moduls zu modifizieren.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.4_detect_the_line.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt haben.

```
import machine
import time

# Create a pin object named greyscale, set pin number 14 as input
line = machine.Pin(14, machine.Pin.IN)


while True:
    # Check if the value is 1 (black)
    if line.value() == 1:
        # Print "black"
        print("black")
        time.sleep(0.5)
    # If the value is not 1 (it's 0, which means white)
    else :
        # Print "white"
        print("white")
        time.sleep(0.5)
```

Wenn das Linienverfolgungsmodul erkennt, dass eine schwarze Linie vorhanden ist, erscheint „schwarz“ in der Shell; andernfalls wird „weiß“ angezeigt.

3.23 5.5 Menschliche Bewegungen erkennen

Ein Passiv-Infrarotsensor (PIR-Sensor) ist ein verbreiteter Sensor, der Infrarotstrahlung (IR), die von Objekten in seinem Sichtfeld emittiert wird, messen kann. Einfach ausgedrückt empfängt er die von menschlichen Körpern ausgehende Infrarotstrahlung und erkennt somit die Bewegung von Personen und anderen Lebewesen. Genauer gesagt signalisiert er der Hauptsteuerplatine, dass jemand Ihren Raum betreten hat.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
PIR-Bewegungssensormodul	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Für Eingang	IO13, IO14, IO27, IO26, IO25, IO33, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23
Für Ausgang	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Bemerkung: IO32 kann in diesem Projekt **nicht als Eingangspin verwendet werden**, da er intern mit einem 1K-Pulldown-Widerstand verbunden ist, was seinen Standardwert auf 0 setzt.

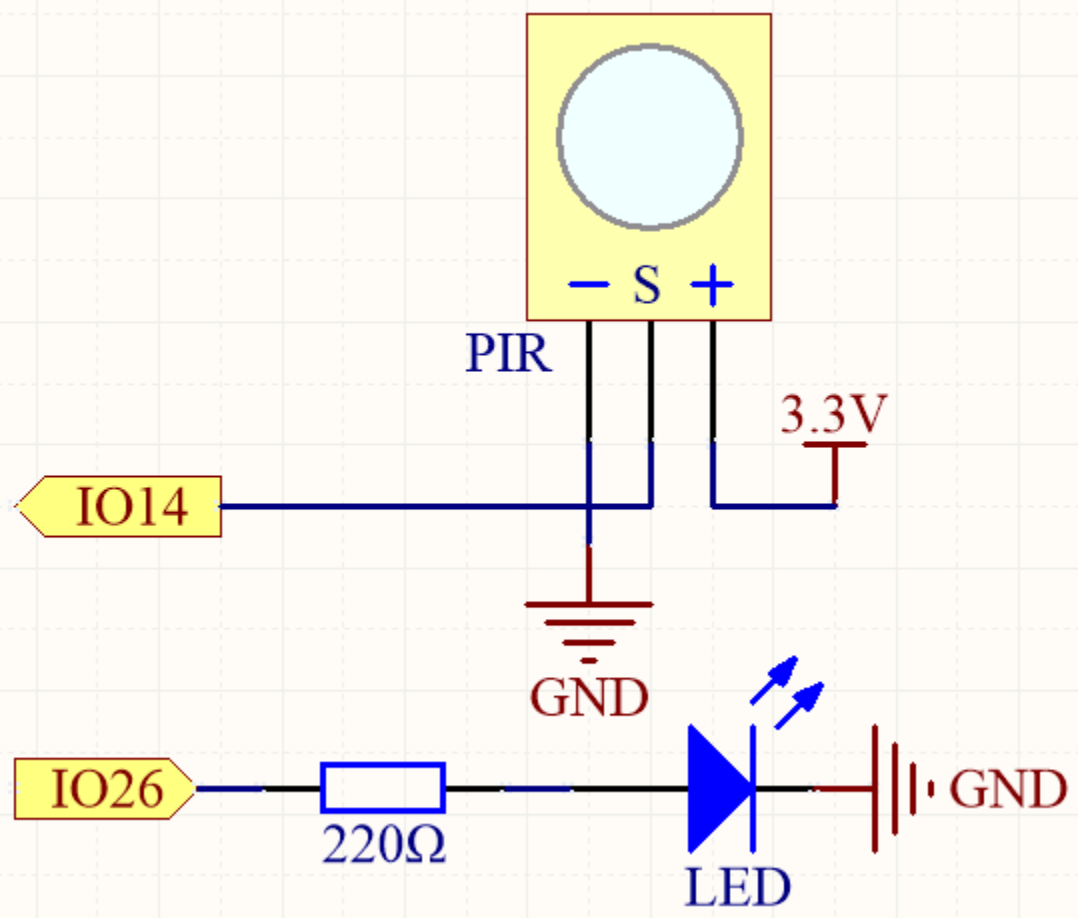
- **Strapping Pins (Eingang)**

  Strapping-Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um spezifische Boot-Modi während des Gerätestarts zu bestimmen (d.h. Power-On-Reset).

Strapping Pins	IO5, IO0, IO2, IO12, IO15

  Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, bedenken Sie die potenziellen Auswirkungen auf den Bootvorgang. Für weitere Details, siehe Abschnitt *Strapping-Pins*.

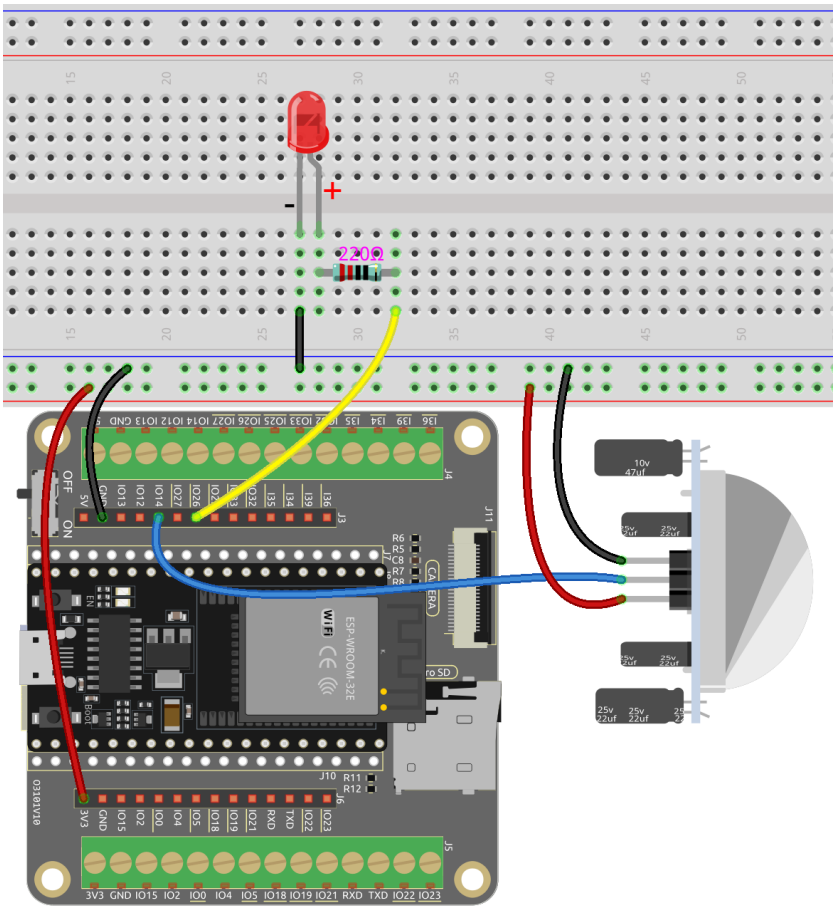
Schaltplan



Wenn das PIR-Modul eine Bewegung erkennt, wird IO14 hochschalten und die LED leuchtet auf. Andernfalls, wenn keine Bewegung erkannt wird, bleibt IO14 niedrig und die LED erlischt.

Bemerkung: Das PIR-Modul verfügt über zwei Potentiometer: eines zur Einstellung der Empfindlichkeit und das andere zur Einstellung der Erfassungsdistanz. Um das PIR-Modul besser zu nutzen, sollten Sie beide gegen den Uhrzeigersinn bis zum Anschlag drehen.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.5_detect_human_movement.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren Sie den Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke auswählen.

```
import machine
import time

# Define pins
PIR_PIN = 14    # PIR sensor
LED_PIN = 26    # LED

# Initialize the PIR sensor pin as an input pin
pir_sensor = machine.Pin(PIR_PIN, machine.Pin.IN, machine.Pin.PULL_DOWN)
# Initialize the LED pin as an output pin
led = machine.Pin(LED_PIN, machine.Pin.OUT)

# Global flag to indicate motion detected
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
motion_detected_flag = False

# Function to handle the interrupt
def motion_detected(pin):
    global motion_detected_flag
    print("Motion detected!")
    motion_detected_flag = True

# Attach the interrupt to the PIR sensor pin
pir_sensor.irq(trigger=machine.Pin.IRQ_RISING, handler=motion_detected)

# Main loop
while True:
    if motion_detected_flag:
        led.value(1)  # Turn on the LED
        time.sleep(5)  # Keep the LED on for 5 seconds
        led.value(0)  # Turn off the LED
        motion_detected_flag = False
```

Wenn das Skript läuft, leuchtet die LED für 5 Sekunden auf und geht dann aus, wenn das PIR-Modul jemanden erfasst, der vorbeigeht.

Bemerkung: Das PIR-Modul verfügt über zwei Potentiometer: eines zur Einstellung der Empfindlichkeit und das andere zur Einstellung der Erfassungsdistanz. Um das PIR-Modul besser zu nutzen, sollten Sie beide gegen den Uhrzeigersinn bis zum Anschlag drehen.

Wie funktioniert das?

Dieser Code richtet ein einfaches Bewegungserkennungssystem mit einem PIR-Sensor und einer LED ein. Wenn eine Bewegung erkannt wird, leuchtet die LED für 5 Sekunden auf.

Hier ist eine Aufschlüsselung des Codes:

1. Definieren Sie die Interrupt-Handler-Funktion, die ausgeführt wird, wenn eine Bewegung erkannt wird:

```
def motion_detected(pin):
    global motion_detected_flag
    print("Motion detected!")
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    motion_detected_flag = True
```

2. Verknüpfen Sie den Interrupt mit dem PIR-Sensor-Pin, wobei der Auslöser auf „steigend“ eingestellt ist (d.h., wenn der Pin von niedriger auf hohe Spannung wechselt):

```
pir_sensor.irq(trigger=machine.Pin.IRQ_RISING, handler=motion_detected)
```

   Dies richtet einen Interrupt am `pir_sensor`-Pin ein, der mit dem PIR-Bewegungssensor verbunden ist.

   Hier ist eine detaillierte Erklärung der Parameter:

   - `trigger=machine.Pin.IRQ_RISING`: Dieser Parameter legt die Auslösebedingung für den Interrupt fest. In diesem Fall wird der Interrupt bei einer steigenden Flanke ausgelöst. Eine steigende Flanke tritt auf, wenn sich die Spannung am Pin von einem niedrigen Zustand (0V) auf einen hohen Zustand (typischerweise 3,3V oder 5V, abhängig von Ihrer Hardware) ändert. Bei einem PIR-Bewegungssensor wechselt der Ausgangspin normalerweise bei einer erkannten Bewegung von niedrig nach hoch, was die steigende Flanke zu einer geeigneten Auslösebedingung macht.
   - `handler=motion_detected`: Dieser Parameter gibt die Funktion an, die ausgeführt wird, wenn der Interrupt ausgelöst wird. In diesem Fall wird die Funktion `motion_detected` als Interrupt-Handler bereitgestellt. Diese Funktion wird automatisch aufgerufen, wenn die Interruptbedingung (steigende Flanke) am `pir_sensor`-Pin erkannt wird.

   Mit dieser Codezeile wird der PIR-Sensor so konfiguriert, dass er die Funktion `motion_detected` aufruft, wann immer Bewegung durch den Sensor erkannt wird, da der Ausgangspin von einem niedrigen zu einem hohen Zustand wechselt.

3. In der Hauptschleife, wenn das `motion_detected_flag` auf `True` gesetzt ist, wird die LED für 5 Sekunden eingeschaltet und dann ausgeschaltet. Das Flag wird dann auf `False` zurückgesetzt, um auf das nächste Bewegungsereignis zu warten.

```
while True:
    if motion_detected_flag:
        led.value(1)  # Turn on the LED
        time.sleep(5)  # Keep the LED on for 5 seconds
        led.value(0)  # Turn off the LED
        motion_detected_flag = False
```

3.24 5.6 Zwei Arten von Transistoren

Dieses Kit enthält zwei Arten von Transistoren, S8550 und S8050, wobei der erstere ein PNP- und der letztere ein NPN-Transistor ist. Sie sehen sich sehr ähnlich, und es bedarf sorgfältiger Überprüfung, um ihre Beschriftungen zu erkennen. Wenn ein High-Level-Signal durch einen NPN-Transistor fließt, wird dieser aktiviert. Ein PNP-Transistor hingegen benötigt ein Low-Level-Signal für seine Steuerung. Beide Transistorarten werden häufig für kontaktlose Schalter verwendet, genau wie in diesem Experiment.

Lassen Sie uns LED und Taster verwenden, um den Einsatz von Transistoren zu verstehen!

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
Taste	
Transistor	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Für Eingang	IO14, IO25, I35, I34, I39, I36, IO18, IO19, IO21, IO22, IO23
Für Ausgang	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

- **Bedingungsabhängige Nutzungspins (Eingang)**

  Die folgenden Pins verfügen über eingebaute Pull-up- oder Pull-down-Widerstände, sodass externe Widerstände nicht erforderlich sind, wenn **sie als Eingangspins verwendet werden**:

Bedingungsabhängige Nutzungspins	Beschreibung
IO13, IO15, IO2, IO4	Hochziehen mit einem 47K-Widerstand setzt den Standardwert auf hoch.
IO27, IO26, IO33	Hochziehen mit einem 4,7K-Widerstand setzt den Standardwert auf hoch.
IO32	Runterziehen mit einem 1K-Widerstand setzt den Standardwert auf niedrig.

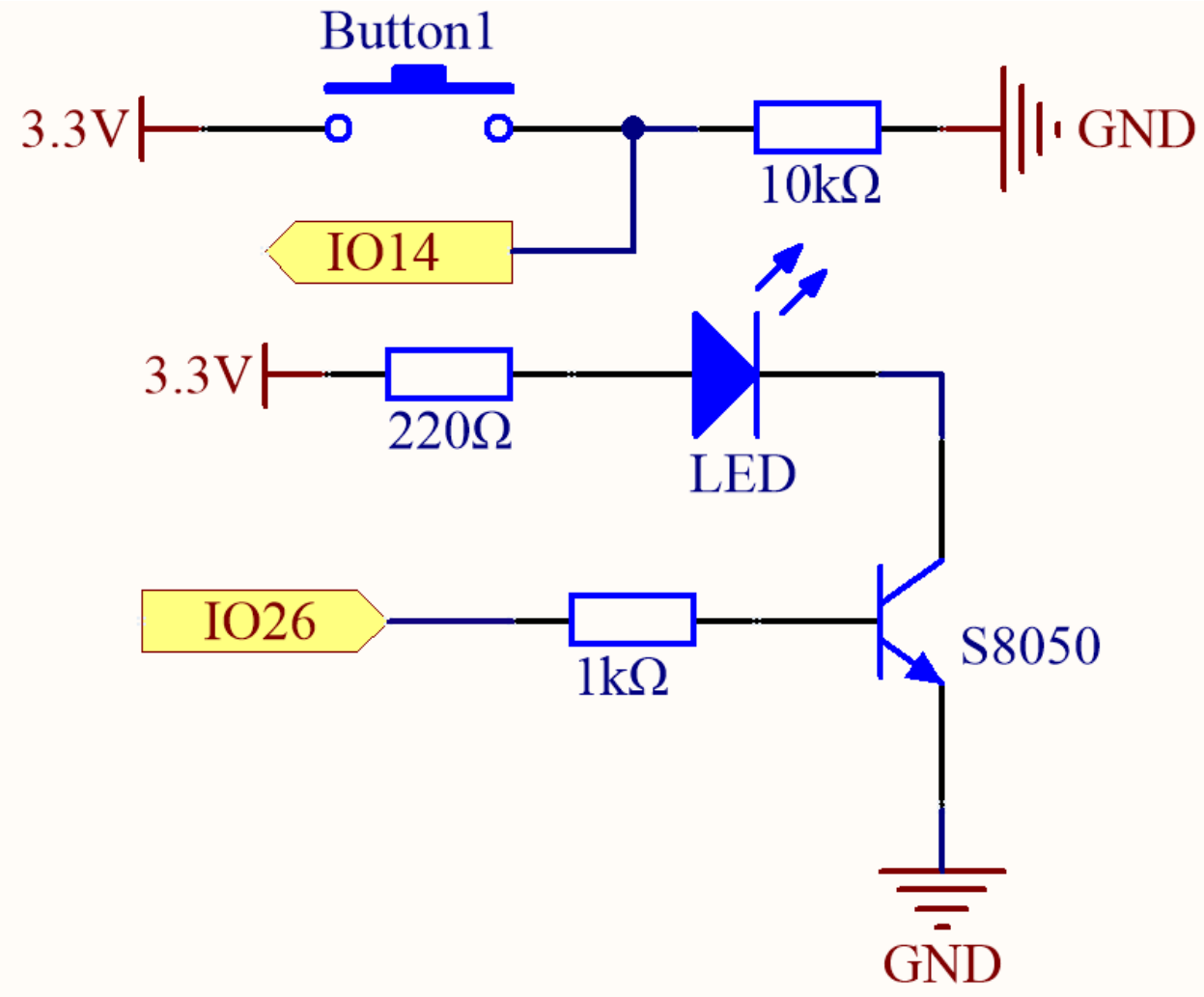
- **Strapping-Pins (Eingang)**

  Strapping-Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um bestimmte Boot-Modi während des Gerätestarts zu bestimmen (d.h. Power-On-Reset).

Strapping-Pins	IO5, IO0, IO2, IO12, IO15

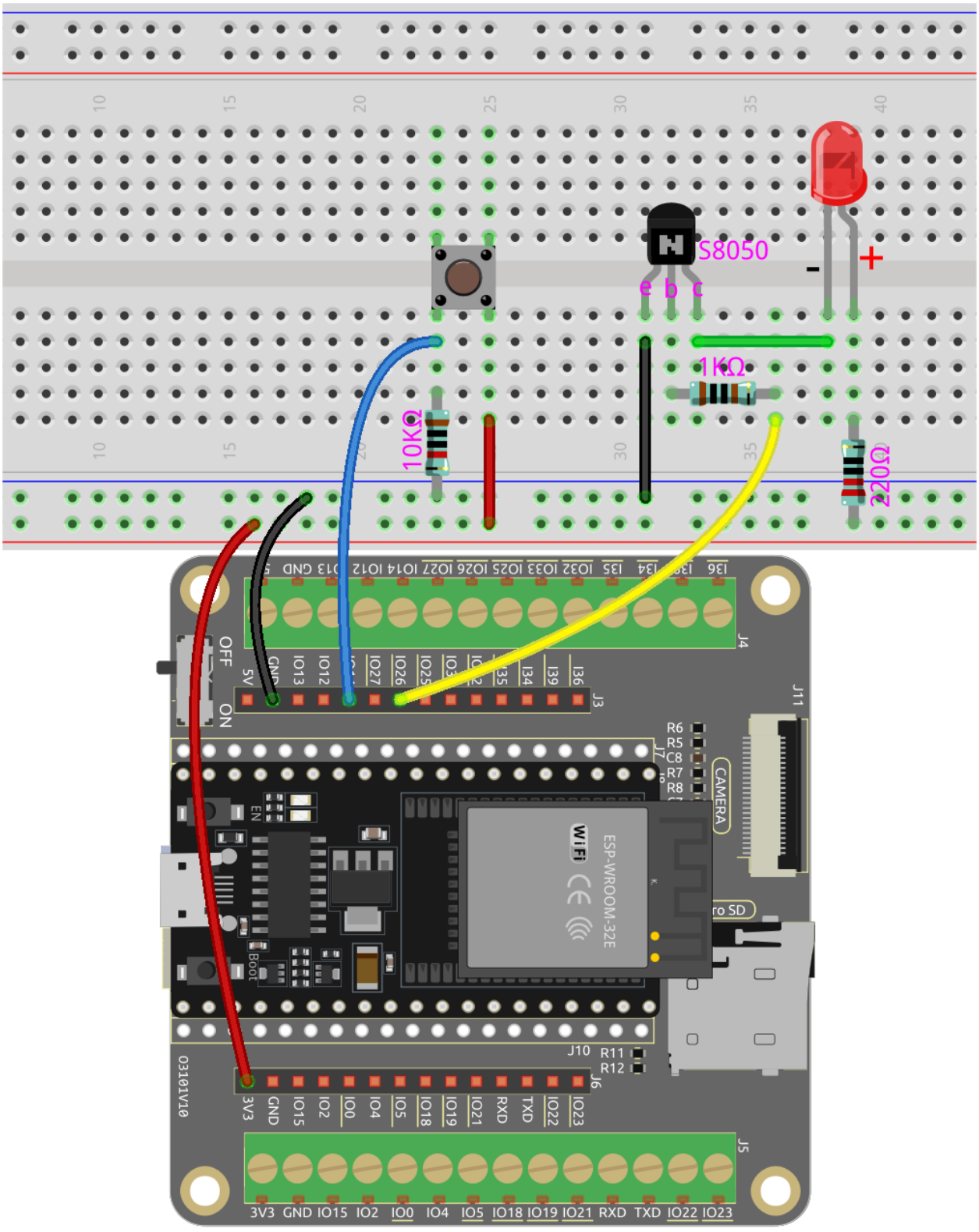
Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins nutzen möchten, bedenken Sie den potenziellen Einfluss auf den Bootvorgang. Für weitere Details siehe den Abschnitt *Strapping-Pins*.

Anschlussmethode für NPN-Transistor (S8050)

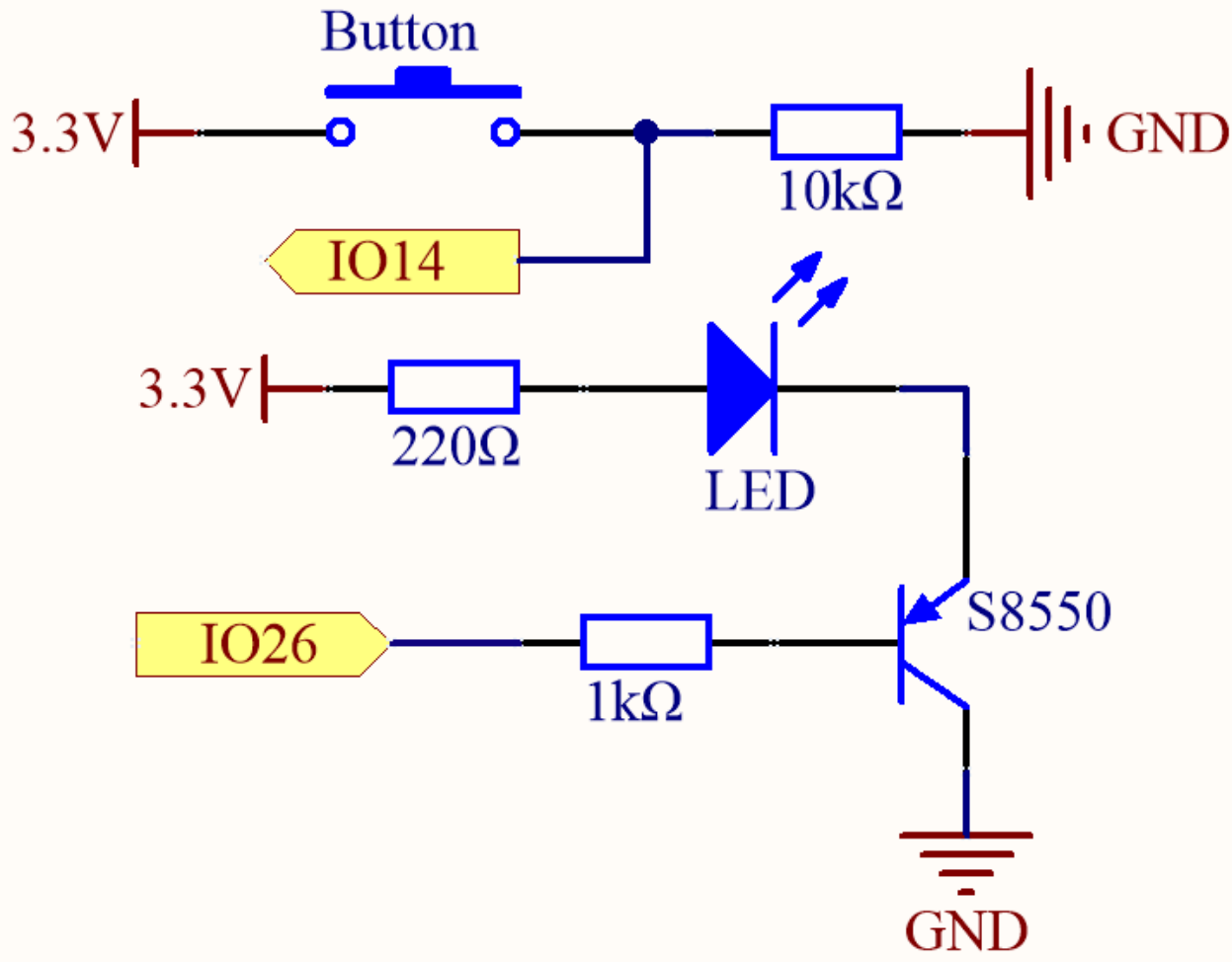


In diesem Schaltkreis leuchtet die LED auf, wenn der Knopf gedrückt wird und IO14 hoch ist.

Durch Programmierung von IO26 auf **hoch** wird nach einem 1k-Strombegrenzungswiderstand (zum Schutz des Transistors) der S8050 (NPN-Transistor) zum Leiten gebracht, sodass die LED aufleuchtet.



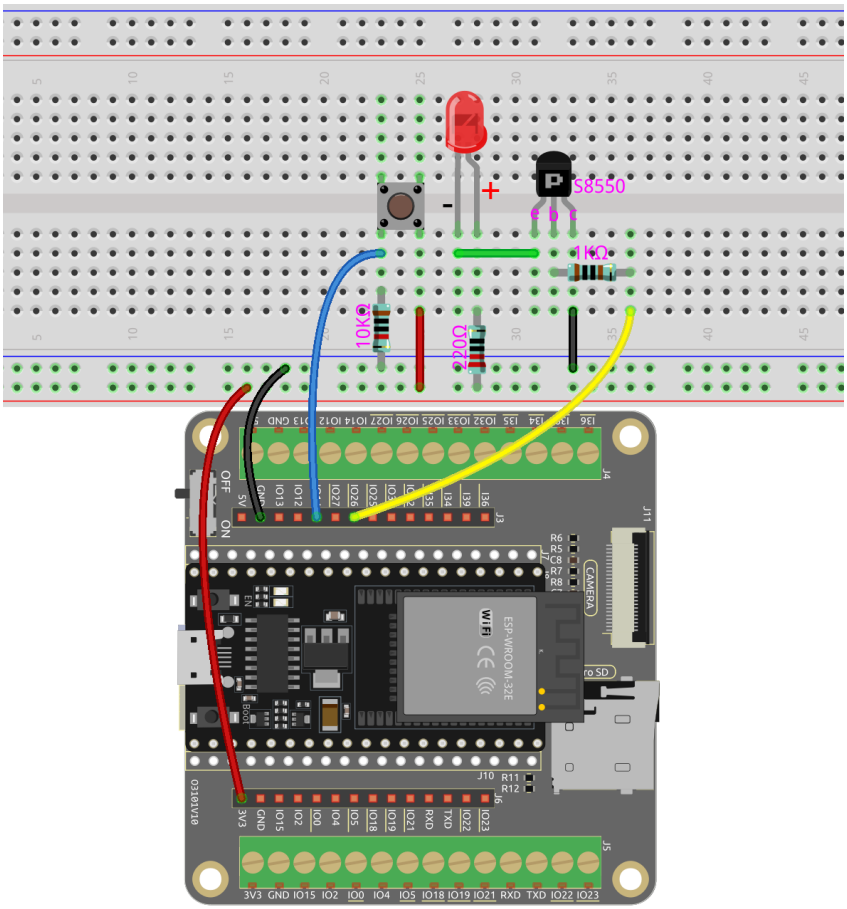
Anschlussmethode für PNP-Transistor (S8550)



In diesem Schaltkreis ist IO14 standardmäßig niedrig und wechselt auf hoch, wenn der Knopf gedrückt wird.

Durch Programmierung von IO26 auf **low** wird nach einem 1k-Strombegrenzungswiderstand (zum Schutz des Transistors) der S8550 (PNP-Transistor) zum Leiten gebracht, sodass die LED aufleuchtet.

Der einzige Unterschied, den Sie zwischen diesem und dem vorherigen Schaltkreis bemerken werden, ist, dass im vorherigen Schaltkreis die Kathode der LED an den **collector** des **S8050 (NPN-Transistor)** angeschlossen ist, während sie in diesem an den **emitter** des **S8550 (PNP-Transistor)** angeschlossen ist.



Code

Bemerkung:

- Öffnen Sie die Datei `5.6_transistor.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren Sie den Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke auswählen.

```
import machine

button = machine.Pin(14, machine.Pin.IN)    # Button
led = machine.Pin(26, machine.Pin.OUT)   # LED

# Start an infinite loop
while True:
    # Read the current value of the 'button' object (0 or 1) and store it in the 'button_
↪status' variable
    button_status = button.value()
    # If the button is pressed (value is 1)
    if button_status == 1:
        led.value(1) # Turn the LED on
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    # If the button is not pressed (value is 0)
    else:
        led.value(0)            # turn the LED off
```

Zwei Arten von Transistoren können mit demselben Code gesteuert werden. Wenn wir den Knopf drücken, sendet der ESP32 ein High-Level-Signal an den Transistor; wenn wir ihn loslassen, sendet er ein Low-Level-Signal.

- Der Schaltkreis mit dem S8050 (NPN-Transistor) leuchtet auf, wenn der Knopf gedrückt wird, was darauf hinweist, dass er sich in einem High-Level-Leitzustand befindet;
- Der Schaltkreis mit dem S8550 (PNP-Transistor) leuchtet auf, wenn der Knopf losgelassen wird, was darauf hinweist, dass er sich in einem Low-Level-Leitzustand befindet.

3.25 5.7 Das Licht erfühlen

Der Fotowiderstand ist ein häufig verwendetes Gerät für analoge Eingänge, ähnlich einem Potentiometer. Sein Widerstandswert ändert sich je nach Intensität des empfangenen Lichts. Bei starker Lichteinwirkung verringert sich der Widerstand des Fotowiderstands, und bei abnehmender Lichtintensität steigt der Widerstand.

Indem wir den Wert des Fotowiderstands auslesen, können wir Informationen über die Umgebungslichtverhältnisse sammeln. Diese Informationen können für Aufgaben wie die Steuerung der Helligkeit einer LED, die Anpassung der Empfindlichkeit eines Sensors oder die Umsetzung lichtabhängiger Aktionen in einem Projekt genutzt werden.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können diese auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Fotowiderstand	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

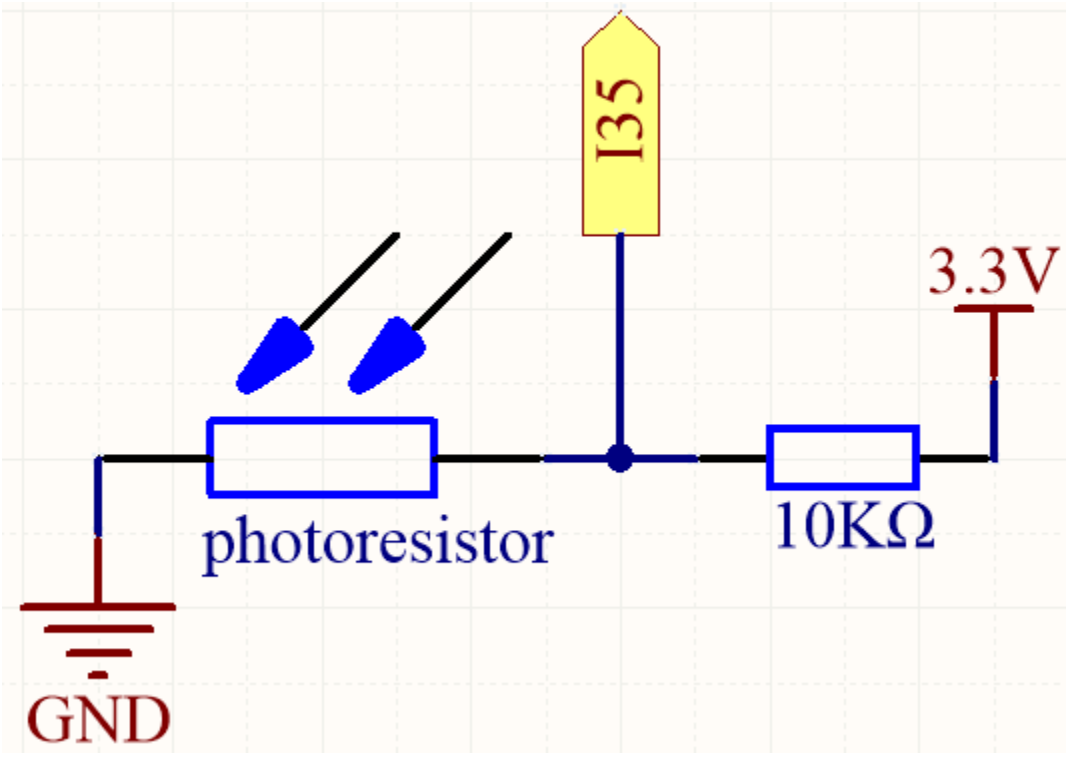
Verfügbare Pins	IO14, IO25, I35, I34, I39, I36

- **Strapping Pins**

  Die folgenden Pins sind Strapping-Pins, die den Startvorgang des ESP32 beim Einschalten oder Zurücksetzen beeinflussen. Nach erfolgreichem Booten des ESP32 können sie jedoch als reguläre Pins verwendet werden.

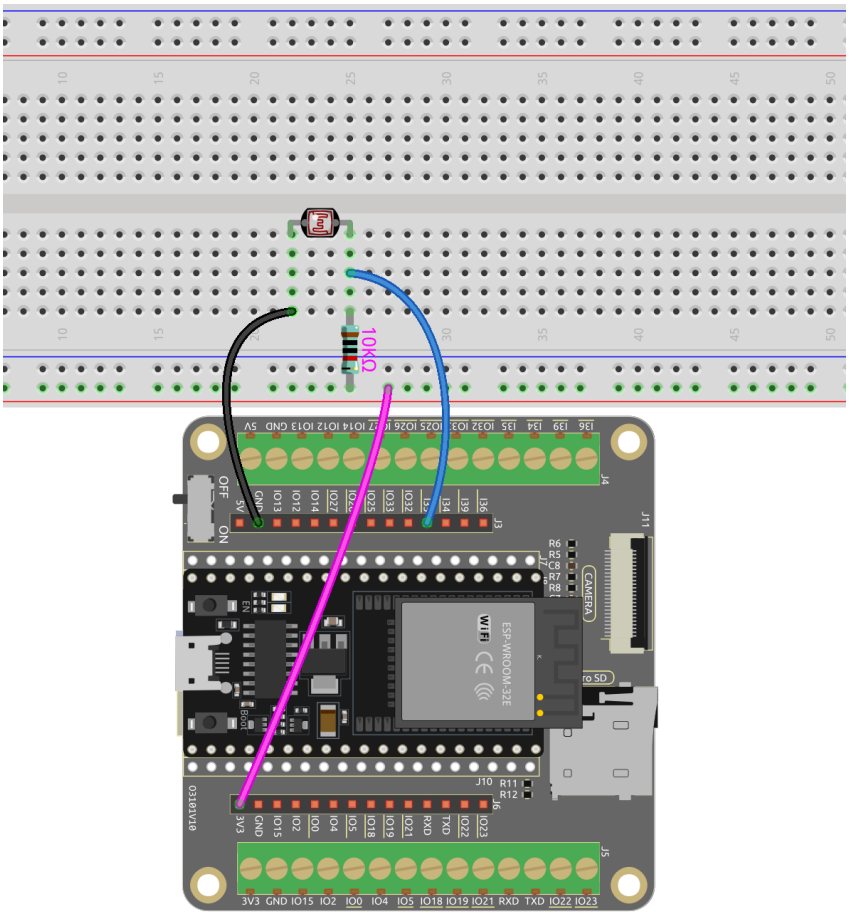
Strapping-Pins	IO0, IO12

Schaltplan



Mit zunehmender Lichtintensität verringert sich der Widerstand des lichtabhängigen Widerstands (LDR), was zu einer Abnahme des auf I35 ausgelesenen Werts führt.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.7_feel_the_light.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren Sie den Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass Sie den Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke auswählen.

```
from machine import ADC,Pin
import time

# create an ADC object acting on a pin
photoresistor = ADC(Pin(35, Pin.IN))

# Configure the ADC attenuation to 11dB for full range
photoresistor.atten(photoresistor.ATTN_11DB)

while True:

    # read a raw analog value in the range 0-4095
    value = photoresistor.read()
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    print(value)
    time.sleep(0.05)
```

Nachdem das Programm ausgeführt wurde, zeigt die Shell die Werte des Fotowiderstands an. Sie können eine Taschenlampe darauf richten oder ihn mit der Hand abdecken, um zu sehen, wie sich der Wert ändert.

- `atten(photoresistor.ATTN_11DB)`: Konfigurieren Sie die ADC-Dämpfung auf 11dB für den vollen Bereich.

  Um Spannungen über der Referenzspannung zu messen, wenden Sie eine Eingangsdämpfung mit dem Schlüsselwortargument atten an.

  Gültige Werte (und ungefähre lineare Messbereiche) sind:

  - ADC.ATTN_0DB: Keine Dämpfung (100mV - 950mV)
  - ADC.ATTN_2_5DB: 2,5dB Dämpfung (100mV - 1250mV)
  - ADC.ATTN_6DB: 6dB Dämpfung (150mV - 1750mV)
  - ADC.ATTN_11DB: 11dB Dämpfung (150mV - 2450mV)

- machine.ADC - MicroPython Dokumentation

3.26 5.8 Drehen Sie den Knopf

Ein Potentiometer ist ein dreipoliges Gerät, das häufig verwendet wird, um den Widerstand in einem Schaltkreis anzupassen. Es verfügt über einen Drehknopf oder einen Schieberegler, mit dem der Widerstandswert des Potentiometers verändert werden kann. In diesem Projekt werden wir es nutzen, um die Helligkeit einer LED zu steuern, ähnlich einer Schreibtischlampe in unserem Alltag. Durch Anpassen der Position des Potentiometers können wir den Widerstand im Stromkreis verändern und somit den durch die LED fließenden Strom regulieren, was deren Helligkeit entsprechend anpasst. Dies ermöglicht es uns, ein anpassbares und verstellbares Beleuchtungserlebnis zu schaffen, vergleichbar mit dem einer Schreibtischlampe.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
Potentiometer	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

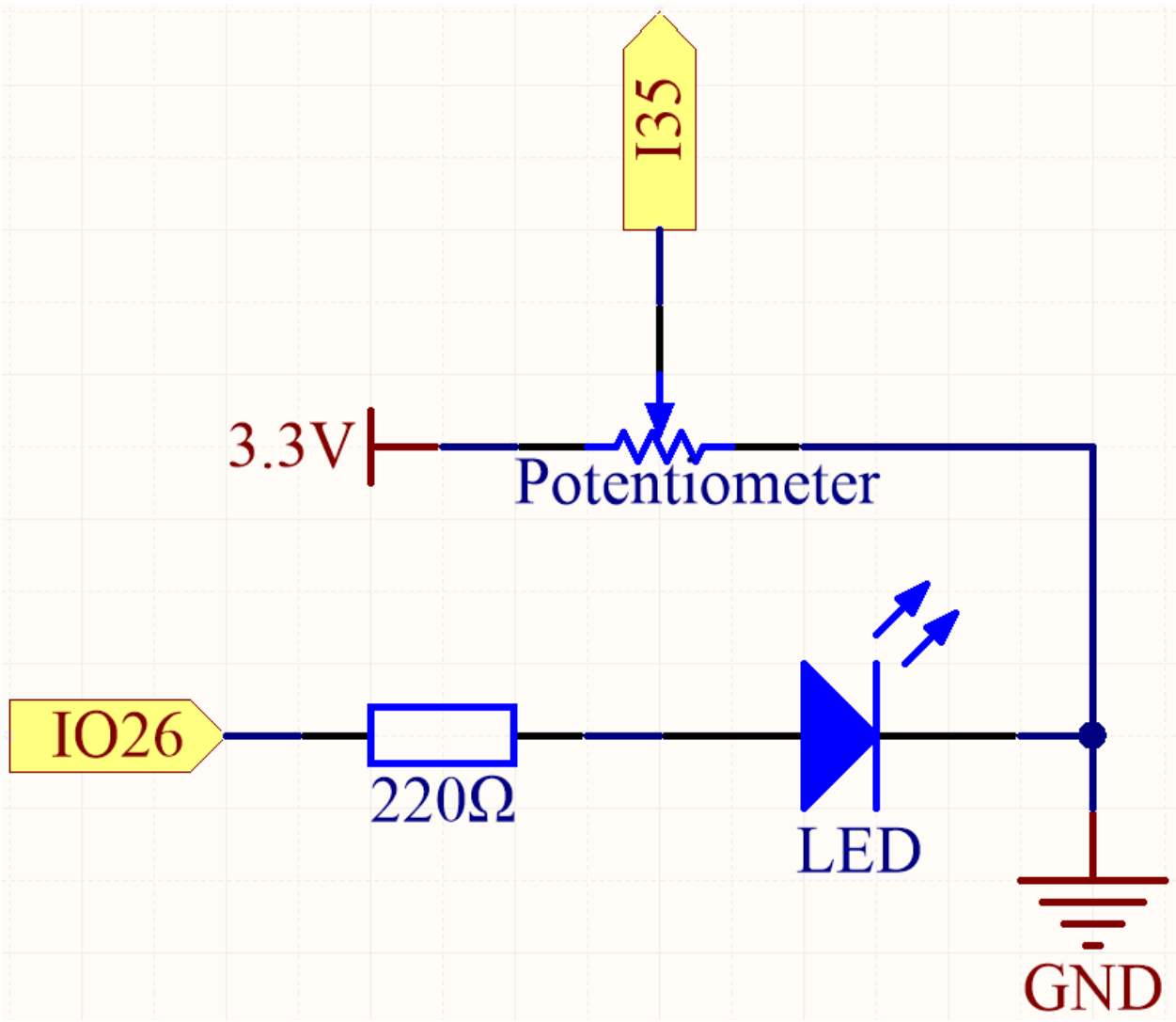
Verfügbare Pins	IO14, IO25, I35, I34, I39, I36

- **Strapping Pins**

  Die folgenden Pins sind Strapping-Pins, die den Startvorgang des ESP32 beim Einschalten oder Zurücksetzen beeinflussen. Sobald der ESP32 jedoch erfolgreich hochgefahren ist, können sie als reguläre Pins verwendet werden.

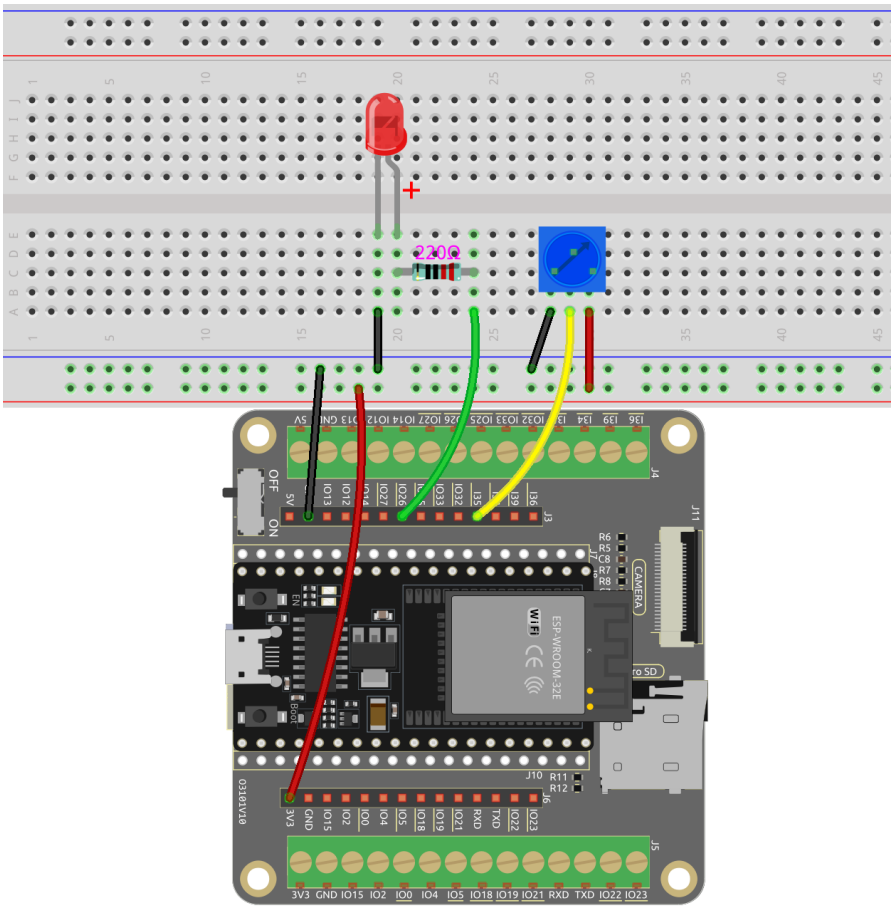
Strapping-Pins	IO0, IO12

Schaltplan



Wenn Sie das Potentiometer drehen, ändert sich der Wert von I35. Durch Programmierung können Sie den Wert von I35 nutzen, um die Helligkeit der LED zu steuern. Daher ändert sich die Helligkeit der LED entsprechend, wenn Sie am Potentiometer drehen.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.8_turn_the_knob.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
from machine import ADC, Pin, PWM
import time

pot = ADC(Pin(35, Pin.IN)) # create an ADC object acting on a pin

# Configure the ADC attenuation to 11dB for full range
pot.atten(pot.ATTN_11DB)

# Create a PWM object
led = PWM(Pin(26), freq=1000)

while True:
    # Read a raw analog value in the range of 0-4095
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    value = pot.read()

    # Scale the value to the range of 0-1023 for ESP32 PWM duty cycle
    pwm_value = int(value * 1023 / 4095)

    # Update the LED brightness based on the potentiometer value
    led.duty(pwm_value)

    # Read the voltage in microvolts and convert it to volts
    voltage = pot.read_uv() / 1000000

    # Print the raw value and the voltage
    print(f"value: {value}, Voltage: {voltage}V")

    # Wait for 0.5 seconds before taking the next reading
    time.sleep(0.5)
```

Mit diesem Skript ausgeführt, ändert sich die Helligkeit der LED, wenn das Potentiometer gedreht wird, während der analoge Wert und die Spannung an diesem Punkt im Shell angezeigt werden.

- machine.ADC - MicroPython Docs

3.27 5.9 Messung der Bodenfeuchtigkeit

Dieser kapazitive Bodenfeuchtigkeitssensor unterscheidet sich von den meisten auf dem Markt erhältlichen resistiven Sensoren, da er das Prinzip der kapazitiven Induktion zur Erfassung der Bodenfeuchtigkeit verwendet.

Durch das visuelle Ablesen der Werte vom Bodenfeuchtigkeitssensor können wir Informationen über den Feuchtigkeitsgehalt im Boden sammeln. Diese Informationen sind nützlich für verschiedene Anwendungen, wie automatische Bewässerungssysteme, Überwachung der Pflanzengesundheit oder Umweltsensorprojekte.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Bodenfeuchtigkeitsmodul	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

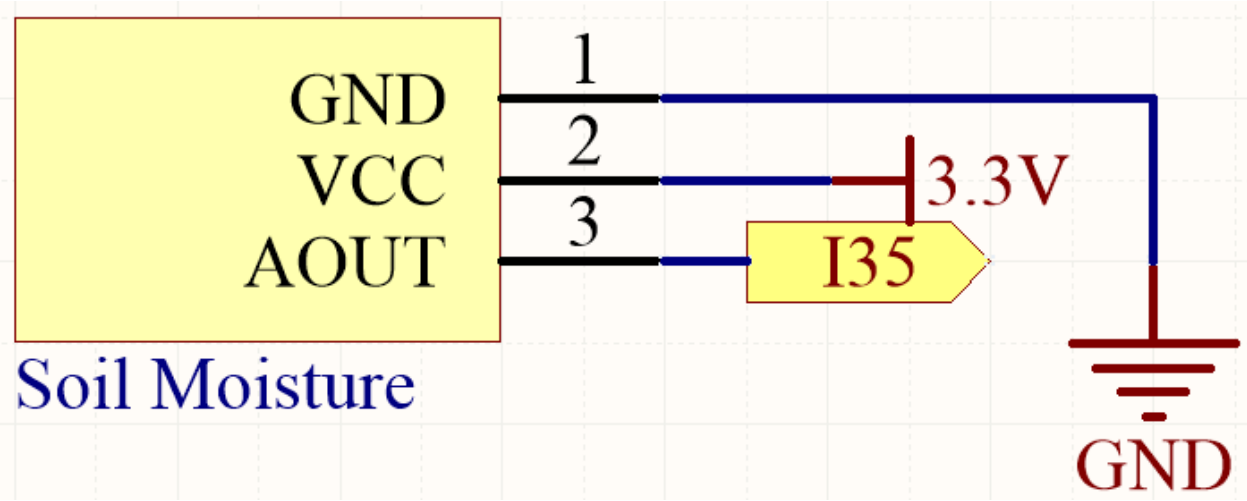
Verfügbare Pins	IO14, IO25, I35, I34, I39, I36

- **Strapping Pins**

  Die folgenden Pins sind Strapping-Pins, die den Startvorgang des ESP32 beim Einschalten oder Zurücksetzen beeinflussen. Sobald der ESP32 jedoch erfolgreich hochgefahren ist, können sie als reguläre Pins verwendet werden.

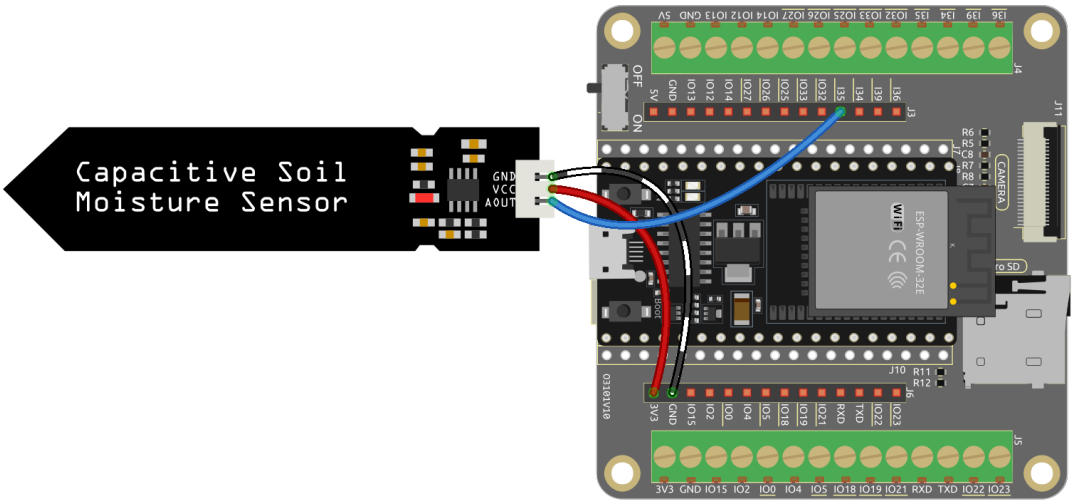
Strapping-Pins	IO0, IO12

Schaltplan



Durch Einsetzen des Moduls in den Boden und Bewässerung desselben wird der auf I35 abgelesene Wert sinken.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.9_moisture.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
from machine import ADC,Pin
import time

# create an ADC object acting on a pin
moisture = ADC(Pin(35, Pin.IN))

# Configure the ADC attenuation to 11dB for full range
moisture.atten(moisture.ATTN_11DB)

while True:

    # read a raw analog value in the range 0-4095
    value = moisture.read()
    print(value)
    time.sleep(0.05)
```

Wenn das Skript läuft, sehen Sie den Wert der Bodenfeuchtigkeit in der Shell.

Durch Einsetzen des Moduls in den Boden und Bewässerung desselben wird der Wert des Bodenfeuchtigkeitssensors kleiner werden.

3.28 5.10 Temperaturmessung

Ein Thermistor ist ein Temperatursensor, der eine starke Temperaturabhängigkeit aufweist, und er kann in zwei Typen eingeteilt werden: Negative Temperature Coefficient (NTC) und Positive Temperature Coefficient (PTC). Der Widerstand eines NTC-Thermistors nimmt mit steigender Temperatur ab, während der Widerstand eines PTC-Thermistors mit steigender Temperatur zunimmt.

In diesem Projekt werden wir einen NTC-Thermistor verwenden. Durch Anschluss des NTC-Thermistors an einen analogen Eingangspin des ESP32-Mikrocontrollers können wir seinen Widerstand messen, der direkt proportional zur Temperatur ist.

Durch die Einbindung des NTC-Thermistors und die Durchführung der notwendigen Berechnungen können wir die Temperatur genau messen und auf dem I2C LCD1602-Modul anzeigen. Dieses Projekt ermöglicht eine Echtzeit-Temperaturüberwachung und bietet eine visuelle Schnittstelle zur Temperaturanzeige.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Thermistor	

Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

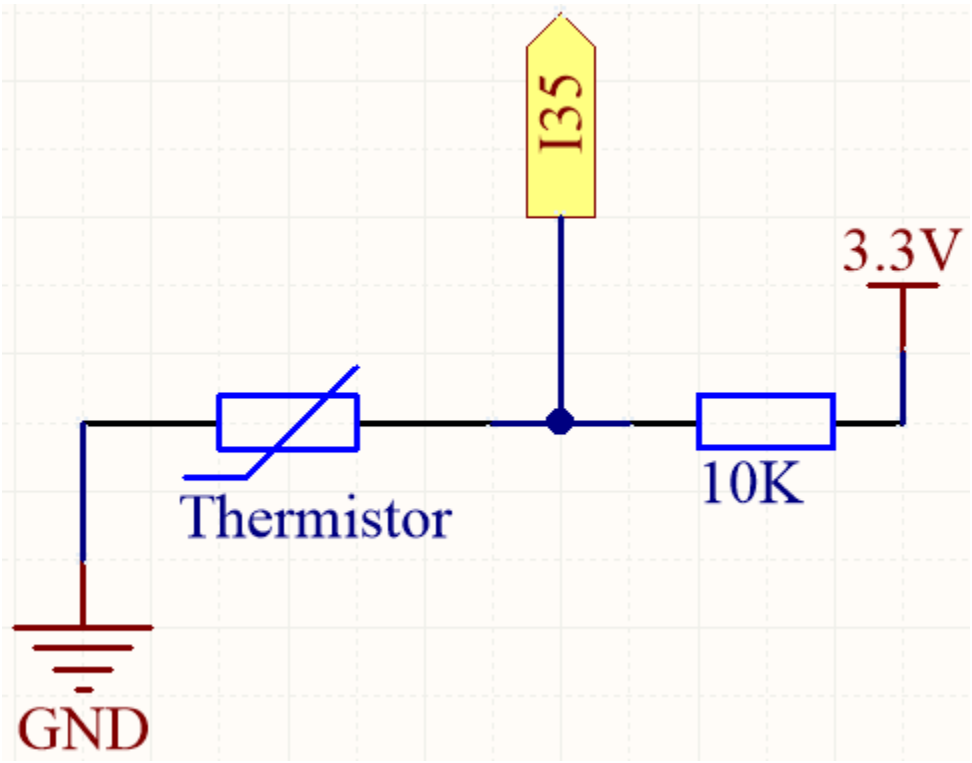
Verfügbare Pins	IO14, IO25, I35, I34, I39, I36

- **Strapping Pins**

  Die folgenden Pins sind Strapping-Pins, die den Startvorgang des ESP32 beim Einschalten oder Zurücksetzen beeinflussen. Sobald der ESP32 jedoch erfolgreich hochgefahren ist, können sie als reguläre Pins verwendet werden.

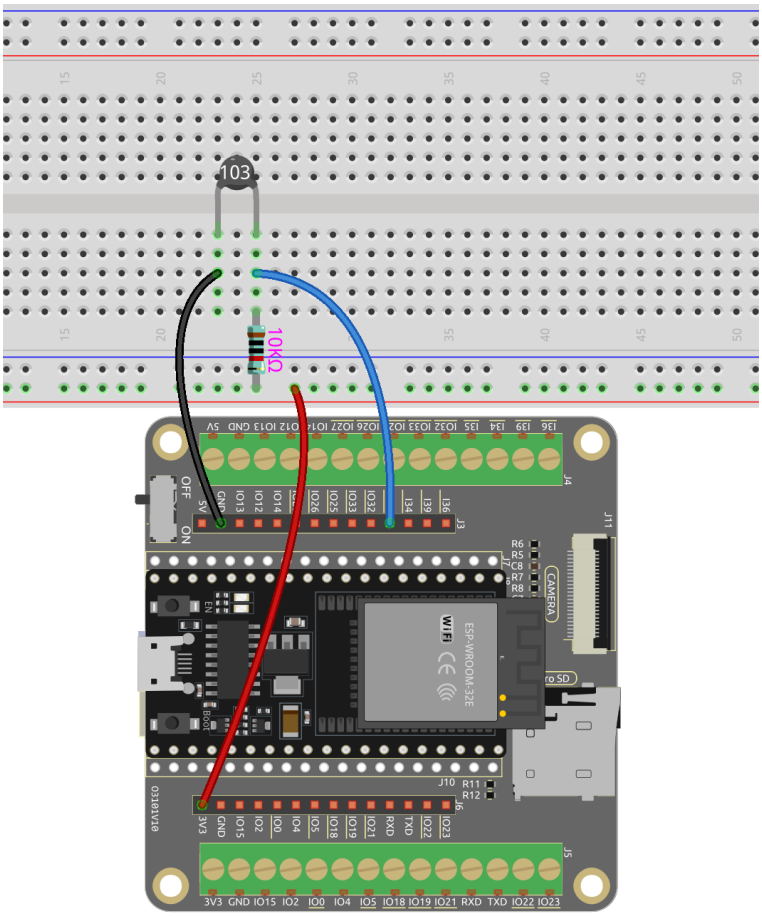
Strapping-Pins	IO0, IO12

Schaltplan



Wenn die Temperatur steigt, nimmt der Widerstand des Thermistors ab, was dazu führt, dass der auf I35 abgelesene Wert sinkt. Zusätzlich können Sie durch Verwendung einer Formel den Analogwert in Temperatur umrechnen und dann ausdrucken.

Verdrahtung



Bemerkung:

- Der Thermistor ist schwarz und mit 103 gekennzeichnet.
- Der Farbring des 10K-Ohm-Widerstands ist rot, schwarz, schwarz, rot und braun.

Code

Bemerkung:

- Öffnen Sie die Datei `5.10_thermistor.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
# Import the necessary libraries
from machine import ADC, Pin
import time
import math

# Define the beta value of the thermistor, typically provided in the datasheet
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
beta = 3950

# Create an ADC object (thermistor)
thermistor = ADC(Pin(35, Pin.IN))

# Set the attenuation
thermistor.atten(thermistor.ATTN_11DB)

# Start an infinite loop to continuously monitor the temperature
while True:
    # Read the voltage in microvolts and convert it to volts
    Vr = thermistor.read_uv() / 1000000

    # Calculate the resistance of the thermistor based on the measured voltage
    Rt = 10000 * Vr / (3.3 - Vr)

    # Use the beta parameter and resistance value to calculate the temperature in Kelvin
    temp = 1 / (((math.log(Rt / 10000)) / beta) + (1 / (273.15 + 25)))

    # Convert to Celsius
    Cel = temp - 273.15

    # Convert to Fahrenheit
    Fah = Cel * 1.8 + 32

    # Print the temperature values in both Celsius and Fahrenheit
    print('Celsius: %.2f C  Fahrenheit: %.2f F' % (Cel, Fah))
    time.sleep(0.5)
```

Wenn der Code ausgeführt wird, zeigt die Shell die Temperaturen in Celsius und Fahrenheit an.

Wie funktioniert das?

Jeder Thermistor hat einen normalen Widerstand. Hier beträgt er 10k Ohm, gemessen bei 25 Grad Celsius.

Wenn die Temperatur steigt, nimmt der Widerstand des Thermistors ab. Dann werden die Spannungsdaten durch den A/D-Adapter in digitale Werte umgewandelt.

Die Temperatur in Celsius oder Fahrenheit wird durch Programmierung ausgegeben.

Hier ist die Beziehung zwischen Widerstand und Temperatur:

RT = RN expB(1/TK - 1/TN)

- **RT** ist der Widerstand des NTC-Thermistors bei der Temperatur **TK**.
- **RN** ist der Widerstand des NTC-Thermistors unter der Nenntemperatur TN. Hier beträgt der numerische Wert von RN 10k.
- **TK** ist eine Kelvin-Temperatur und die Einheit ist K. Hier beträgt der numerische Wert von **TK** `373.15 + degree Celsius`.
- **TN** ist eine Nenntemperatur in Kelvin; die Einheit ist ebenfalls K. Hier beträgt der numerische Wert von TN `373.15+25`.
- Und **B(Beta)**, die Materialkonstante des NTC-Thermistors, wird auch als Wärmeempfindlichkeitsindex bezeichnet, mit einem numerischen Wert von `4950`.

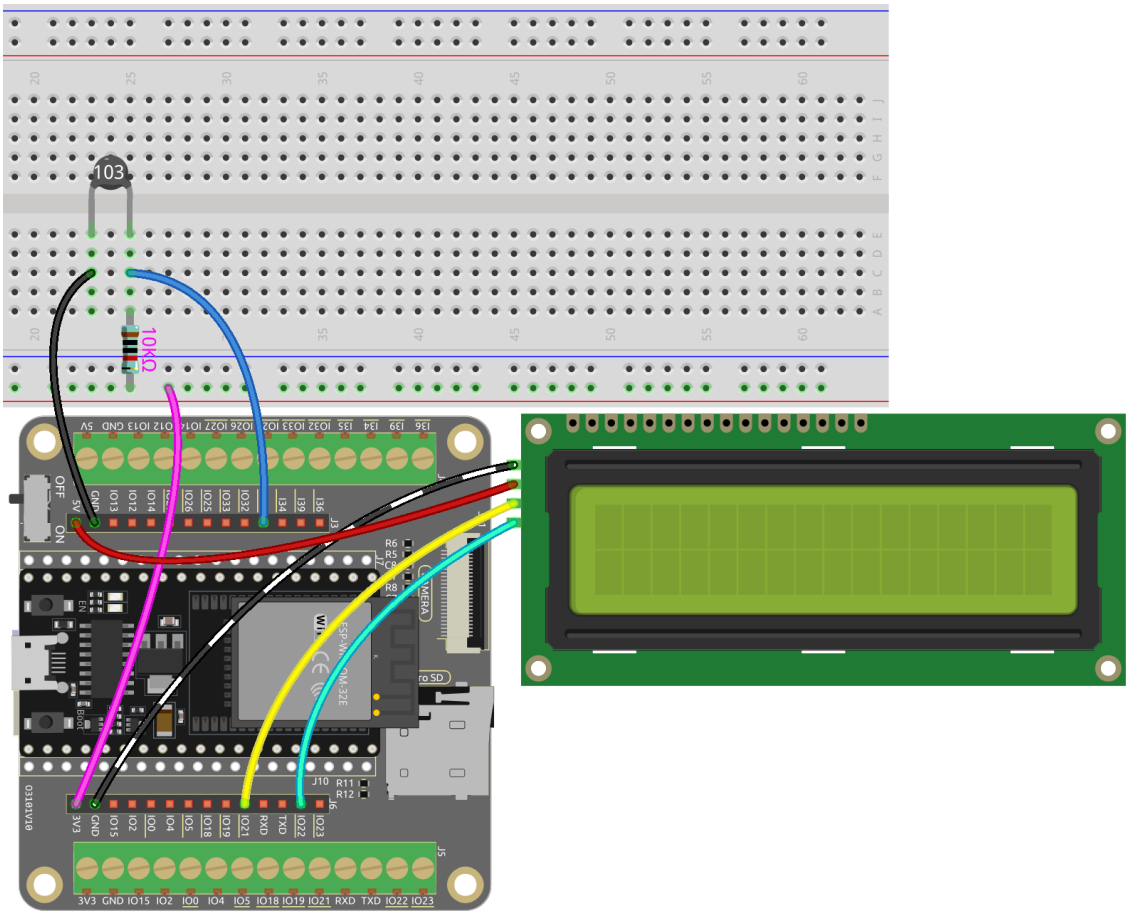
- **exp** ist die Abkürzung für Exponential, und die Basiszahl `e` ist eine natürliche Zahl und entspricht ungefähr 2,7.

Konvertieren Sie diese Formel `TK=1/(ln(RT/RN)/B+1/TN)`, um die Kelvin-Temperatur zu erhalten, die minus 273,15 Grad Celsius entspricht.

Diese Beziehung ist eine empirische Formel. Sie ist nur genau, wenn die Temperatur und der Widerstand innerhalb des effektiven Bereichs liegen.

Weitere Informationen

Sie können auch die berechneten Temperaturen in Celsius und Fahrenheit auf dem I2C LCD1602 anzeigen.



Bemerkung:

- Öffnen Sie die Datei `5.10_thermistor_lcd.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.
- Hier müssen Sie die Bibliothek `lcd1602.py` verwenden, überprüfen Sie bitte, ob sie auf den ESP32 hochgeladen wurde. Für eine detaillierte Anleitung siehe *1.4 Bibliotheken Hochladen (Wichtig)*.

```
# Import the necessary libraries
from machine import ADC, Pin
from lcd1602 import LCD
import time
import math

# Define the beta value of the thermistor, typically provided in the datasheet
beta = 3950

# Create an ADC object (thermistor)
thermistor = ADC(Pin(35, Pin.IN))

# Set the attenuation
thermistor.atten(thermistor.ATTN_11DB)

lcd = LCD()

# Start an infinite loop to continuously monitor the temperature
while True:
    # Read the voltage in microvolts and convert it to volts
    Vr = thermistor.read_uv() / 1000000

    # Calculate the resistance of the thermistor based on the measured voltage
    Rt = 10000 * Vr / (3.3 - Vr)

    # Use the beta parameter and resistance value to calculate the temperature in Kelvin
    temp = 1 / (((math.log(Rt / 10000)) / beta) + (1 / (273.15 + 25)))

    # Convert to Celsius
    Cel = temp - 273.15

    # Convert to Fahrenheit
    Fah = Cel * 1.8 + 32

    # Print the temperature values in both Celsius and Fahrenheit
    print('Celsius: %.2f C  Fahrenheit: %.2f F' % (Cel, Fah))

    # Clear the LCD screen
    lcd.clear()

    # Display the temperature values in both Celsius and Fahrenheit
    lcd.message('Cel: %.2f \xDFC \n' % Cel)
    lcd.message('Fah: %.2f \xDFF' % Fah)
    time.sleep(1)
```

3.29 5.11 Bedienung des Joysticks

Wenn Sie viele Videospiele spielen, sollten Sie mit dem Joystick sehr vertraut sein. Er wird normalerweise verwendet, um den Charakter zu bewegen, den Bildschirm zu drehen usw.

Das Prinzip hinter der Fähigkeit des Joysticks, dem Computer unsere Aktionen zu übermitteln, ist sehr einfach. Man kann ihn sich als aus zwei Potentiometern bestehend vorstellen, die senkrecht zueinander stehen. Diese beiden Potentiometer messen den analogen Wert des Joysticks vertikal und horizontal, was in einem Wert (x, y) in einem ebenen rechtwinkligen Koordinatensystem resultiert.

Der Joystick dieses Kits verfügt auch über einen digitalen Eingang, der aktiviert wird, wenn der Joystick gedrückt wird.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Joystick-Modul	

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Für Analogeingang	IO14, IO25, I35, I34, I39, I36
Für Digitaleingang	IO13, IO14, IO27, IO26, IO25, IO33, IO4, IO18, IO19, IO21, IO22, IO23

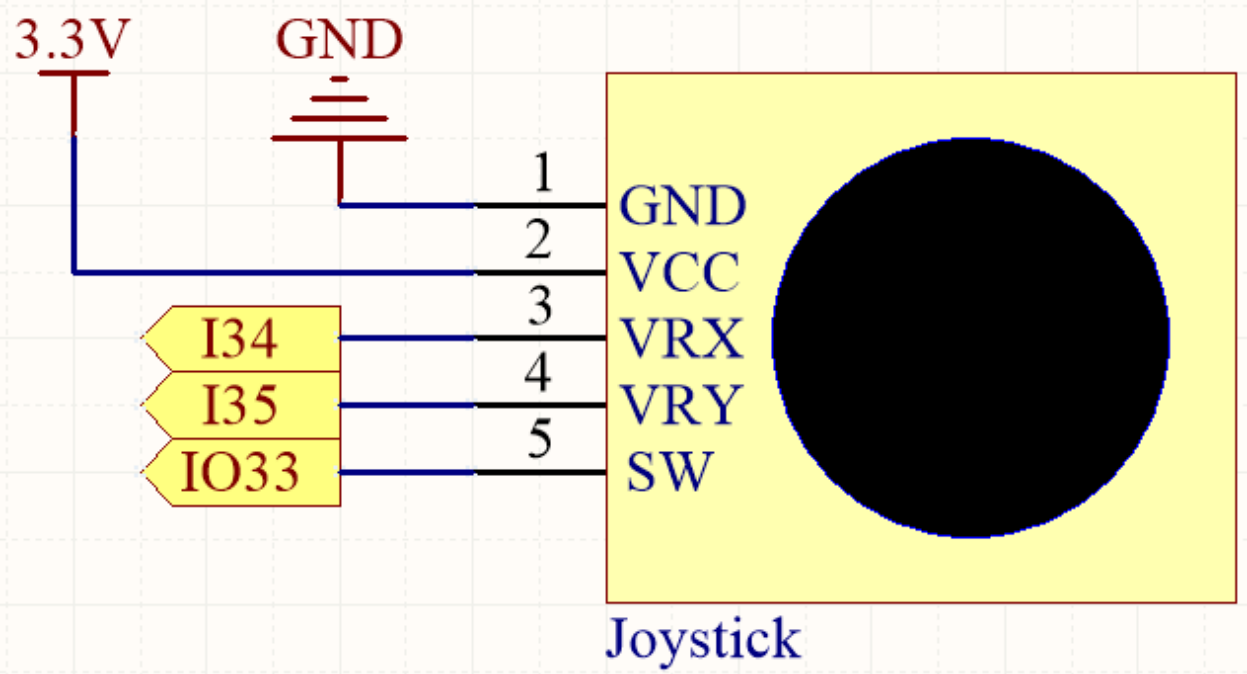
- **Strapping Pins (Eingang)**

  Strapping Pins sind eine spezielle Gruppe von Pins, die verwendet werden, um spezifische Startmodi während des Gerätestarts zu bestimmen (z. B. Power-On-Reset).

Strapping-Pins	IO5, IO0, IO2, IO12, IO15

  Generell wird **nicht empfohlen, sie als Eingangspins zu verwenden**. Wenn Sie diese Pins verwenden möchten, bedenken Sie die möglichen Auswirkungen auf den Startprozess. Für weitere Details siehe den Abschnitt *Strapping-Pins*.

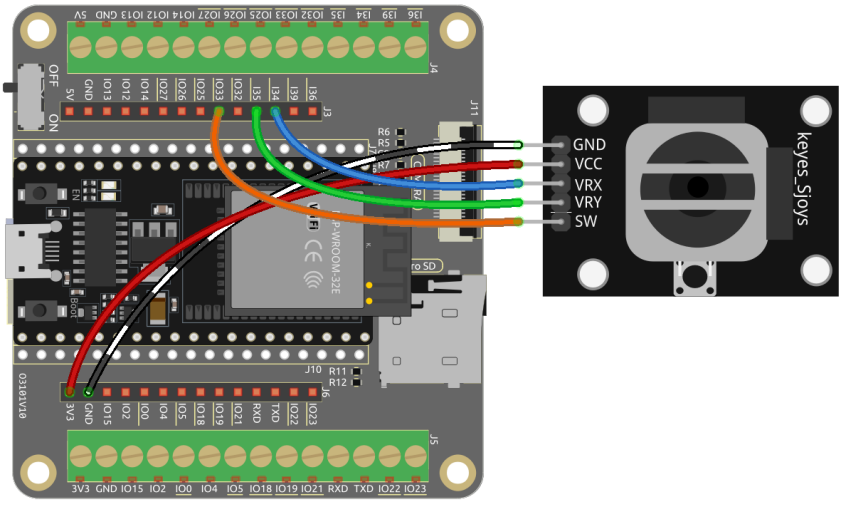
Schaltplan



Der SW (Z-Achse)-Pin ist mit IO33 verbunden, der einen eingebauten 4,7K-Pull-up-Widerstand hat. Daher gibt er, wenn der SW-Knopf nicht gedrückt ist, ein hohes Signal aus. Wird der Knopf gedrückt, wird ein niedriges Signal ausgegeben.

I34 und I35 ändern ihre Werte, wenn Sie den Joystick bewegen. Der Bereich der Werte reicht von 0 bis 4095.

Verdrahtung



Code

Bemerkung:

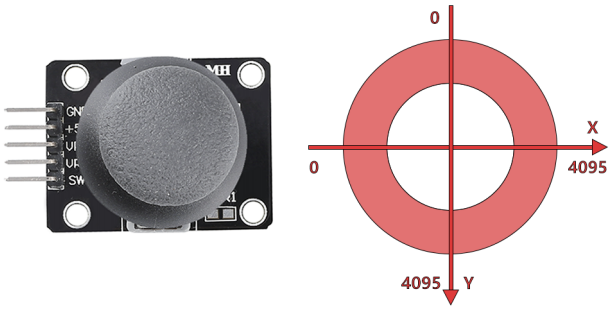
- Öffnen Sie die Datei `5.11_joystick.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
from machine import ADC,Pin
import time

xAxis = ADC(Pin(34, Pin.IN)) # create an ADC object acting on a pin
xAxis.atten(xAxis.ATTN_11DB)
yAxis = ADC(Pin(35, Pin.IN)) # create an ADC object acting on a pin
yAxis.atten(yAxis.ATTN_11DB)
button = Pin(33, Pin.IN, Pin.PULL_UP)

while True:
    xValue = xAxis.read()  # read a raw analog value in the range 0-4095
    yValue = yAxis.read()  # read a raw analog value in the range 0-4095
    btnValue = button.value()
    print(f"X:{xValue}, Y:{yValue}, Button:{btnValue}")
    time.sleep(0.1)
```

Wenn das Programm läuft, gibt die Shell die x-, y- und Knopfwerte des Joysticks aus.

```
X:1921, Y:1775, Button:0
X:1921, Y:1775, Button:0
X:1923, Y:1775, Button:0
X:1924, Y:1776, Button:0
X:1926, Y:1777, Button:0
X:1925, Y:1776, Button:0
X:1924, Y:1776, Button:0
```

- Die x- und y-Achsenwerte sind analoge Werte, die von 0 bis 4095 variieren.
- Der Knopf ist ein digitaler Wert mit einem Status von 1 (Loslassen) oder 0 (Drücken).



3.30 5.12 Distanzmessung

Das Ultraschallmodul wird zur Entfernungsmessung oder Objekterkennung verwendet. In diesem Projekt werden wir das Modul so programmieren, dass es Hindernisentfernungen erfasst. Durch das Senden von Ultraschallimpulsen und das Messen der Zeit, die sie zum Zurückprallen benötigen, können wir Entfernungen berechnen. Dies ermöglicht es uns, distanzbasierte Aktionen oder Verhaltensweisen zur Hindernisvermeidung zu implementieren.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Ultraschall-Modul	

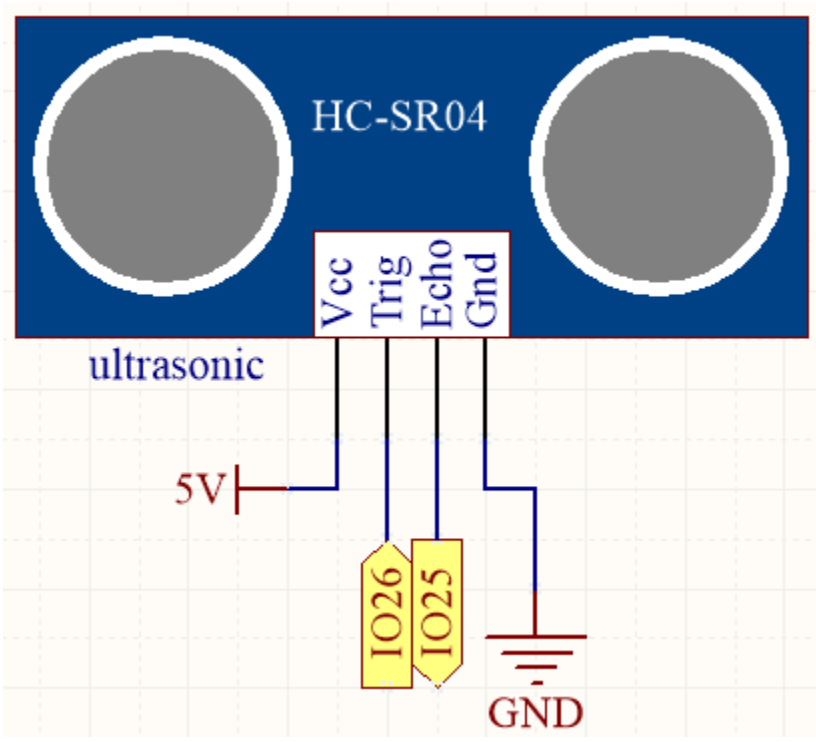
Verfügbare Pins

- **Verfügbare Pins**

  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Für Eingang	IO13, IO14, IO27, IO26, IO25, IO33, IO32, I35, I34, I39, I36, IO4, IO18, IO19, IO21, IO22, IO23
Für Ausgang	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO32, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan

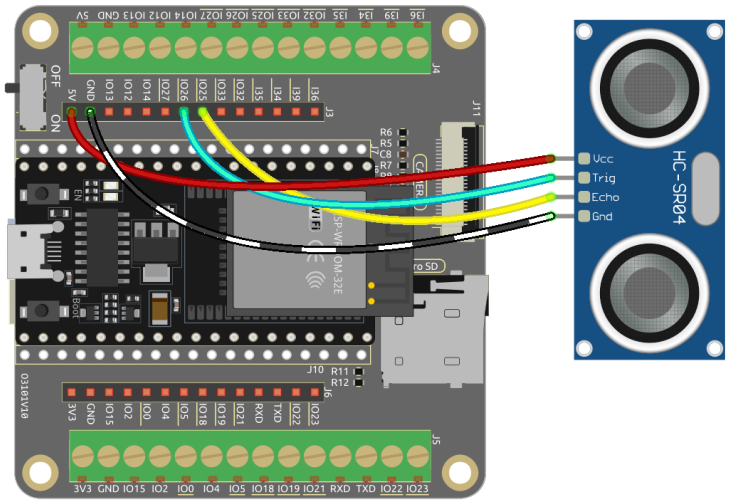


Der ESP32 sendet alle 10 Sekunden eine Reihe von Rechteckwellensignalen an den Trig-Pin des Ultraschallsensors.

Das veranlasst den Ultraschallsensor, ein 40kHz Ultraschallsignal nach außen zu senden. Gibt es ein Hindernis vorne, werden die Ultraschallwellen zurückreflektiert.

Durch Aufzeichnung der Zeit vom Senden bis zum Empfangen des Signals, Teilen durch 2 und Multiplizieren mit der Lichtgeschwindigkeit können Sie die Entfernung zum Hindernis bestimmen.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.12_ultrasonic.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
import machine
import time

# Define the trigger and echo pins for the distance sensor
TRIG = machine.Pin(26, machine.Pin.OUT)
ECHO = machine.Pin(25, machine.Pin.IN)

# Calculate the distance using the ultrasonic sensor
def distance():
    # Ensure trigger is off initially
    TRIG.off()
    time.sleep_us(2)  # Wait for 2 microseconds

    # Send a 10-microsecond pulse to the trigger pin
    TRIG.on()
    time.sleep_us(10)
    TRIG.off()

    # Wait for the echo pin to go high
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    while not ECHO.value():
        pass

    # Record the time when the echo pin goes high
    time1 = time.ticks_us()

    # Wait for the echo pin to go low
    while ECHO.value():
        pass

    # Record the time when the echo pin goes low
    time2 = time.ticks_us()

    # Calculate the time difference between the two recorded times
    during = time.ticks_diff(time2, time1)

    # Calculate and return the distance (in cm) using the speed of sound (340 m/s)
    return during * 340 / 2 / 10000

# Continuously measure and print the distance
while True:
    dis = distance()
    print('Distance: %.2f' % dis)
    time.sleep_ms(300)  # Wait for 300 milliseconds before repeating
```

Sobald das Programm läuft, wird die Shell die Entfernung des Ultraschallsensors zum Hindernis vor ihm ausgeben.

3.31 5.13 Temperatur - Feuchtigkeit

Der DHT11 ist ein Temperatur- und Feuchtigkeitssensor, der häufig für Umweltmessungen verwendet wird. Es handelt sich um einen digitalen Sensor, der mit einem Mikrocontroller kommuniziert, um Temperatur- und Feuchtigkeitswerte bereitzustellen.

In diesem Projekt werden wir den DHT11-Sensor auslesen und die von ihm erfassten Temperatur- und Feuchtigkeitswerte ausgeben.

Durch das Auslesen der vom Sensor bereitgestellten Daten können wir die aktuellen Temperatur- und Feuchtigkeitswerte in der Umgebung ermitteln. Diese Werte können für die Echtzeitüberwachung von Umweltbedingungen, Wetterbeobachtungen, Raumklimasteuerung, Feuchtigkeitsberichte und mehr verwendet werden.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

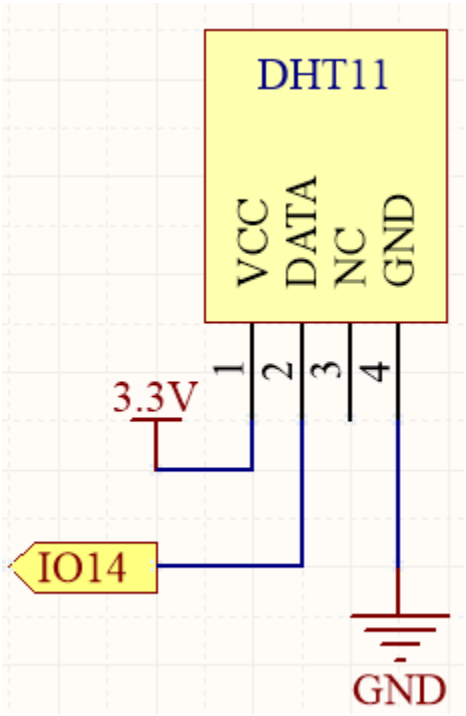
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
DHT11 Feuchtigkeits- und Temperatursensor	

- **Verfügbare Pins**

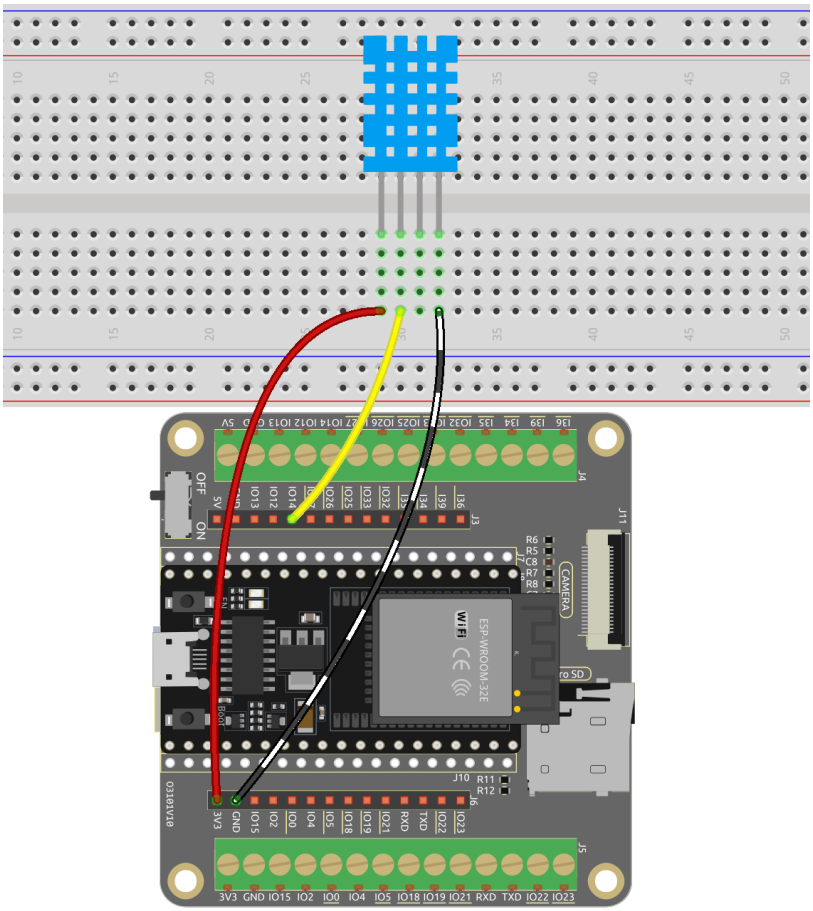
  Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO33, IO15, IO2, IO0, IO4, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.13_dht11.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
import dht
import machine
import time

# Initialize the DHT11 sensor and connect it to pin 14
sensor = dht.DHT11(machine.Pin(14))

# Loop indefinitely to continuously measure temperature and humidity
while True:
    try:
        # Measure temperature and humidity
        sensor.measure()
```

(Fortsetzung auf der nächsten Seite)

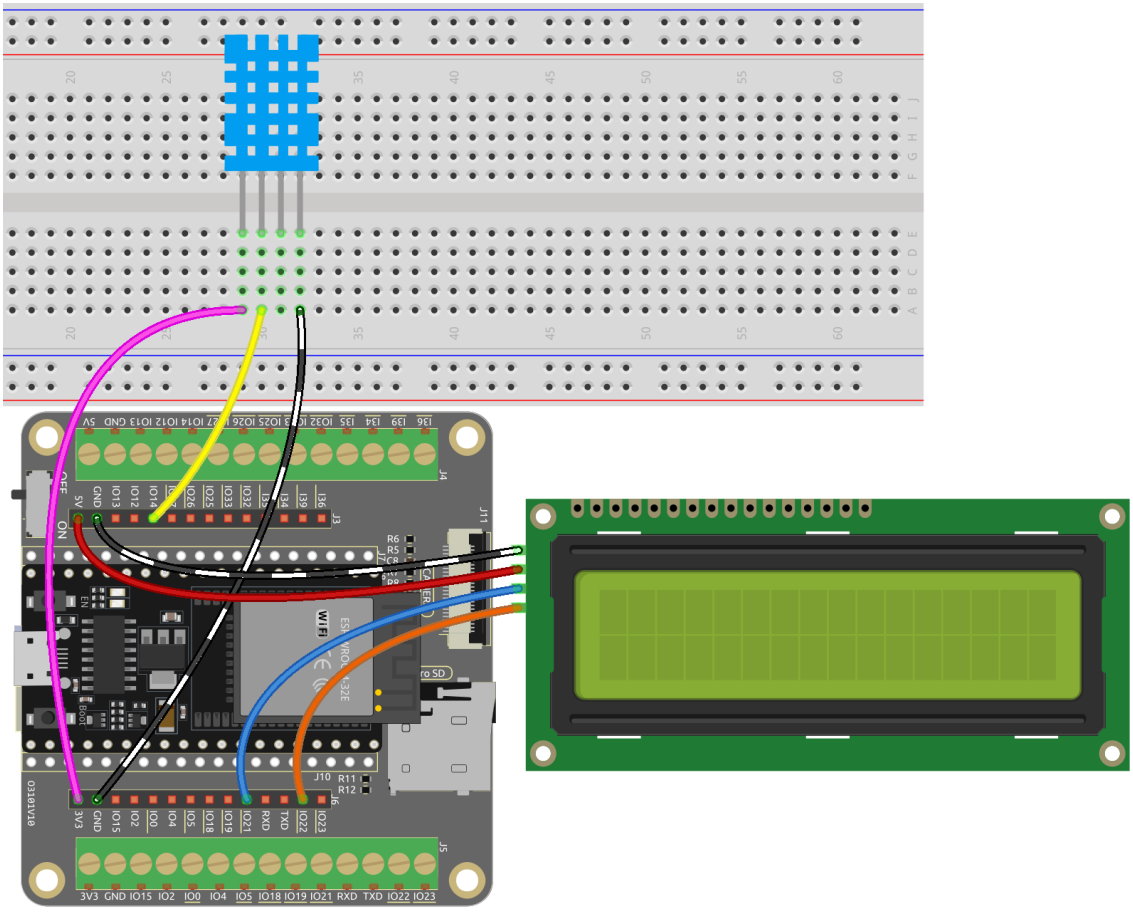
(Fortsetzung der vorherigen Seite)

```
        # Get temperature and humidity values
        temp = sensor.temperature()
        humi = sensor.humidity()

        # Print temperature and humidity
        print("Temperature: {}, Humidity: {}".format(temp, humi))

        # Wait for 1 second between measurements
        time.sleep(1)
    except Exception as e:
        print("Error: ", e)
        time.sleep(1)
```

Wenn der Code läuft, sehen Sie, wie die Shell kontinuierlich die Temperatur und Feuchtigkeit ausgibt, und während das Programm stabil läuft, werden diese beiden Werte immer genauer.

Weitere Informationen

Sie können auch die Temperatur und Feuchtigkeit auf dem I2C LCD1602 anzeigen.



Bemerkung:

- Öffnen Sie die Datei `5.13_dht11_lcd.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“

oder drücken Sie F5, um ihn auszuführen.

- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.
- Hier müssen Sie die Bibliothek `lcd1602.py` verwenden, überprüfen Sie bitte, ob sie auf den ESP32 hochgeladen wurde. Für eine detaillierte Anleitung siehe *1.4 Bibliotheken Hochladen (Wichtig)*.

```
import dht
import machine
import time
from lcd1602 import LCD

# Initialize the DHT11 sensor and connect it to pin 14
sensor = dht.DHT11(machine.Pin(14))

# Initialize the LCD1602 display
lcd = LCD()

# Loop to measure temperature and humidity
while True:
    try:
        # Measure temperature and humidity
        sensor.measure()

        # Get temperature and humidity values
        temp = sensor.temperature()
        humi = sensor.humidity()

        # Print temperature and humidity
        print("Temperature: {}, Humidity: {}".format(temp, humi))

        # Clear the LCD display
        lcd.clear()

        # Display temperature and humidity on the LCD1602 screen
        lcd.write(0, 0, "Temp: {}\xDFC".format(temp))
        lcd.write(0, 1, "Humi: {}%".format(humi))

        # Wait for 2 seconds before measuring again
        time.sleep(2)

    except Exception as e:
        print("Error: ", e)
        time.sleep(2)
```

3.32 5.14 IR-Fernbedienung

Ein Infrarotempfänger ist eine Komponente, die Infrarotsignale empfängt und Signale, die mit TTL-Pegel kompatibel sind, unabhängig erkennen und ausgeben kann. Er ist ähnlich groß wie ein regulärer, in Kunststoff verpackter Transistor und wird häufig in verschiedenen Anwendungen wie Infrarot-Fernbedienung und Infrarot-Übertragung eingesetzt.

In diesem Projekt werden wir einen Infrarotempfänger verwenden, um Signale von einer Fernbedienung zu erkennen. Wenn eine Taste auf der Fernbedienung gedrückt wird und der Infrarotempfänger das entsprechende Signal empfängt, kann er das Signal dekodieren, um zu bestimmen, welche Taste gedrückt wurde. Durch Dekodieren des empfangenen Signals können wir den spezifischen Schlüssel oder Befehl identifizieren, der damit verbunden ist.

Der Infrarotempfänger ermöglicht es uns, Fernbedienungsfunktionen in unser Projekt einzubauen, sodass wir Geräte mittels Infrarotsignalen interagieren und steuern können.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

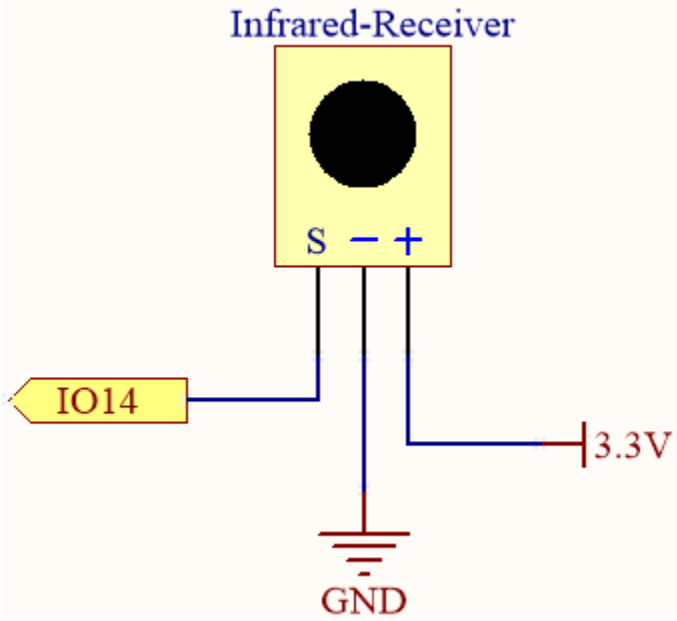
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
IR-Empfänger	

Verfügbare Pins

Hier ist eine Liste der verfügbaren Pins auf dem ESP32-Board für dieses Projekt.

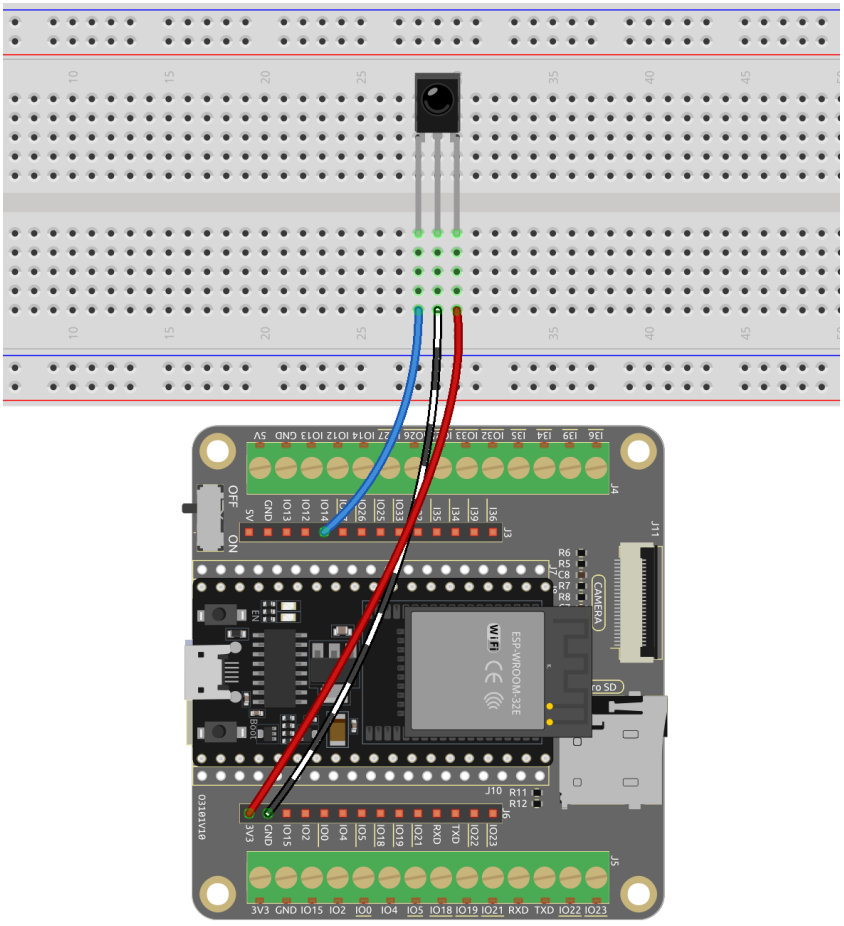
Verfügbare Pins	IO13, IO12, IO14, IO27, IO26, IO25, IO15, IO0, IO5, IO18, IO19, IO21, IO22, IO23

Schaltplan



Wenn Sie eine Taste auf der Fernbedienung drücken, erkennt der Infrarotempfänger das Signal, und Sie können eine Infrarotbibliothek verwenden, um es zu dekodieren. Dieser Dekodierungsprozess ermöglicht es Ihnen, den Schlüsselwert zu erhalten, der mit dem Tastendruck verbunden ist.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `5.14_ir_receiver.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.
- Hier müssen Sie die Bibliotheken im Ordner `ir_rx` verwenden. Stellen Sie bitte sicher, dass sie auf den ESP32 hochgeladen wurden. Für eine umfassende Anleitung siehe *1.4 Bibliotheken Hochladen (Wichtig)*.

```
import time
from machine import Pin, freq
from ir_rx.print_error import print_error
from ir_rx.nec import NEC_8

pin_ir = Pin(14, Pin.IN) # IR receiver

# Decode the received data and return the corresponding key name
def decodeKeyValue(data):
    if data == 0x16:
        return "0"
    if data == 0x0C:
        return "1"
    if data == 0x18:
        return "2"
    if data == 0x5E:
        return "3"
    if data == 0x08:
        return "4"
    if data == 0x1C:
        return "5"
    if data == 0x5A:
        return "6"
    if data == 0x42:
        return "7"
    if data == 0x52:
        return "8"
    if data == 0x4A:
        return "9"
    if data == 0x09:
        return "+"
    if data == 0x15:
        return "-"
    if data == 0x7:
        return "EQ"
    if data == 0x0D:
        return "U/SD"
    if data == 0x19:
        return "CYCLE"
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    if data == 0x44:
        return "PLAY/PAUSE"
    if data == 0x43:
        return "FORWARD"
    if data == 0x40:
        return "BACKWARD"
    if data == 0x45:
        return "POWER"
    if data == 0x47:
        return "MUTE"
    if data == 0x46:
        return "MODE"
    return "ERROR"

# User callback
def callback(data, addr, ctrl):
    if data < 0:  # NEC protocol sends repeat codes.
        pass
    else:
        print(decodeKeyValue(data))


ir = NEC_8(pin_ir, callback) # Instantiate the NEC_8 receiver

# Show debug information
ir.error_function(print_error)

# keep the script running until interrupted by a keyboard interrupt (Ctrl+C)
try:
    while True:
        pass
except KeyboardInterrupt:
    ir.close()  # Close the receiver
```

Wenn das Programm läuft, drücken Sie die Taste auf der Fernbedienung, und der Wert sowie der Name der Taste erscheinen in der Shell.

Bemerkung: Die neue Fernbedienung hat am Ende einen Kunststoffstreifen, um die Batterie im Inneren zu isolieren. Um die Fernbedienung zu aktivieren, entfernen Sie einfach dieses Kunststoffteil.

Wie funktioniert das?

1. Obwohl dieses Programm auf den ersten Blick etwas komplex erscheinen mag, erfüllt es tatsächlich die grundlegenden Funktionen des IR-Empfängers mit nur wenigen Codezeilen.

```
import time
from machine import Pin, freq
from ir_rx.nec import NEC_8

pin_ir = Pin(14, Pin.IN) # IR receiver

# User callback
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
def callback(data, addr, ctrl):
    if data < 0:  # NEC protocol sends repeat codes.
        pass
    else:
        print(decodeKeyValue(data))

ir = NEC_8(pin_ir, callback)  # Instantiate receiver
```

- In diesem Code wird ein `ir`-Objekt instanziiert, das es ihm ermöglicht, die vom IR-Empfänger erfassten Signale jederzeit zu lesen.
- Die resultierenden Informationen werden dann in der Variablen `data` innerhalb der Callback-Funktion gespeichert.
  - Callback-Funktion - Wikipedia
- Wenn der IR-Empfänger doppelte Werte empfängt (z.B. wenn eine Taste gedrückt und gehalten wird), ist das `data` kleiner als 0, und diese `data` müssen herausgefiltert werden.
- Andernfalls wäre das `data` ein nutzbarer Wert, allerdings in einem unlesbaren Code. Die Funktion `decodeKeyValue(data)` wird dann verwendet, um ihn in ein verständlicheres Format zu dekodieren.

```
def decodeKeyValue(data):
    if data == 0x16:
        return "0"
    if data == 0x0C:
        return "1"
    if data == 0x18:
        return "2"
    if data == 0x5E:
    ...
```

2. Als Nächstes integrieren wir mehrere Debug-Funktionen in das Programm. Während diese Funktionen wichtig sind, stehen sie nicht direkt mit dem gewünschten Ergebnis in Verbindung, das wir erreichen wollen.

```
from ir_rx.print_error import print_error

ir.error_function(print_error) # Show debug information
```

3. Schließlich verwenden wir eine leere Schleife für das Hauptprogramm und implementieren eine try-except-Struktur, um sicherzustellen, dass das Programm mit dem ordnungsgemäß beendeten `ir`-Objekt abgeschlossen wird.

```
try:
    while True:
        pass
except KeyboardInterrupt:
    ir.close()
```

- Try-Anweisung - Python-Dokumentation

6. Lustige Projekte

3.33 6.1 Fruchtpiano

Haben Sie schon einmal davon geträumt, Klavier zu spielen, konnten sich aber kein Klavier leisten? Oder möchten Sie einfach nur etwas Spaß mit einem selbstgebauten Fruchtpiano haben? Dann ist dieses Projekt genau das Richtige für Sie!

Mit nur wenigen Berührungssensoren auf dem ESP32-Board können Sie jetzt Ihre Lieblingsmelodien spielen und das Klavierspielen genießen, ohne dabei ein Vermögen auszugeben.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Summer	
Transistor	

Über die Touch-Pins

Der ESP32-Mikrocontroller verfügt über integrierte Touch-Sensor-Funktionalität, die es Ihnen ermöglicht, bestimmte Pins auf dem Board als berührungsempfindliche Eingänge zu verwenden. Der Touch-Sensor funktioniert, indem er Änderungen der Kapazität an den Touch-Pins misst, die durch die elektrischen Eigenschaften des menschlichen Körpers verursacht werden.

Hier sind einige Schlüsselfunktionen des Touch-Sensors auf dem ESP32:

- **Anzahl der Touch-Pins**

  Der ESP32 verfügt über bis zu 10 Touch-Pins, abhängig vom spezifischen Board. Die Touch-Pins sind typischerweise mit einem „T“ gefolgt von einer Nummer gekennzeichnet.

  - GPIO4: TOUCH0
  - GPIO0: TOUCH1
  - GPIO2: TOUCH2
  - GPIO15: TOUCH3
  - GPIO13: TOUCH4

  - GPIO12: TOUCH5
  - GPIO14: TOUCH6
  - GPIO27: TOUCH7
  - GPIO33: TOUCH8
  - GPIO32: TOUCH9

Bemerkung: Die Pins GPIO0 und GPIO2 werden für das Bootstrapping und das Flashen der Firmware auf den ESP32 verwendet. Diese Pins sind auch mit der onboard LED und dem Button verbunden. Daher wird im Allgemeinen nicht empfohlen, diese Pins für andere Zwecke zu verwenden, da dies den normalen Betrieb des Boards stören könnte.

- **Empfindlichkeit**

  Der Touch-Sensor auf dem ESP32 ist sehr empfindlich und kann selbst kleine Änderungen der Kapazität erkennen. Die Empfindlichkeit kann über Softwareeinstellungen angepasst werden.

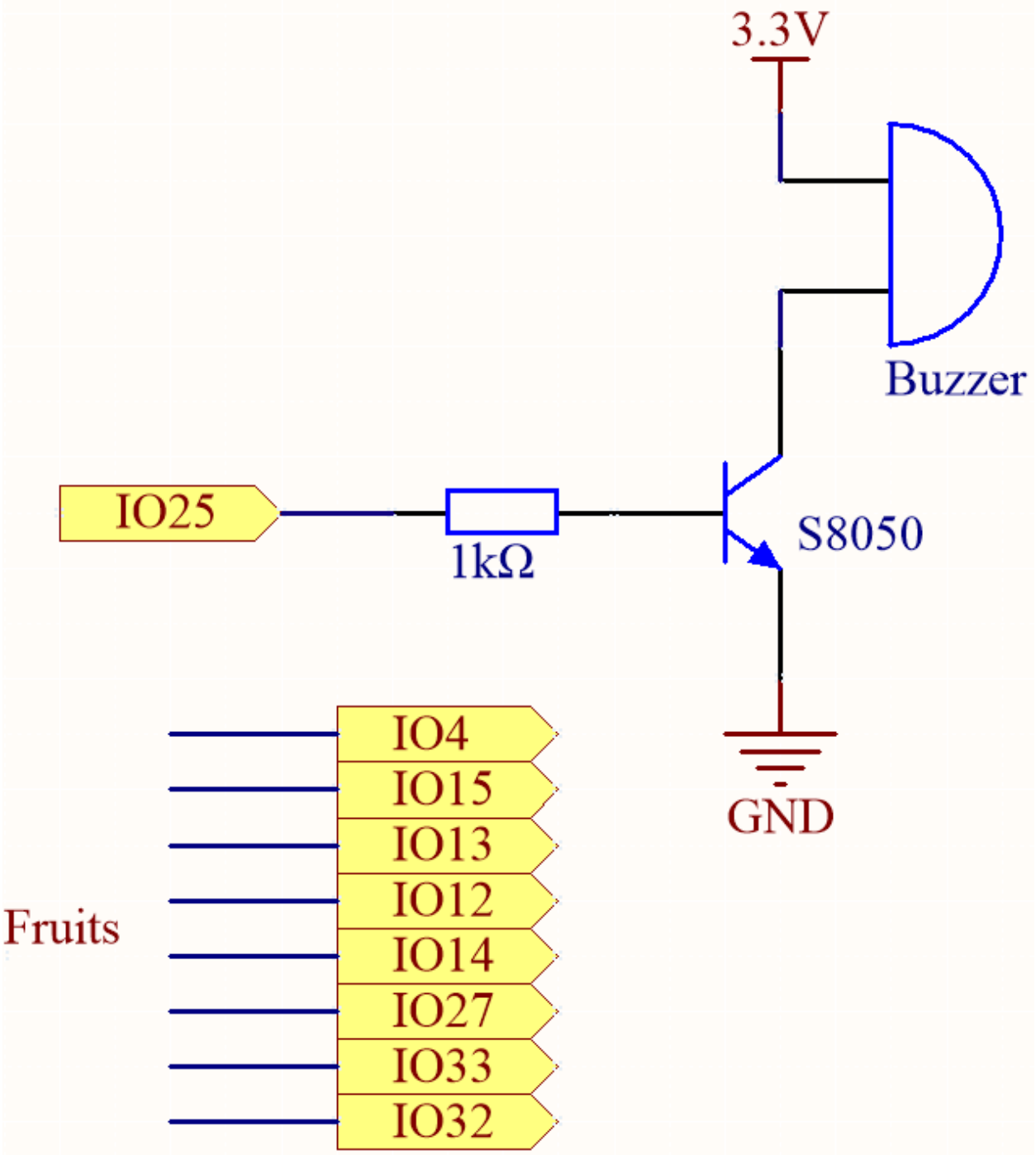
- **ESD-Schutz**

  Die Touch-Pins auf dem ESP32 verfügen über einen eingebauten ESD (Electrostatic Discharge) Schutz, der hilft, Schäden am Board durch statische Elektrizität zu verhindern.

- **Multitouch**

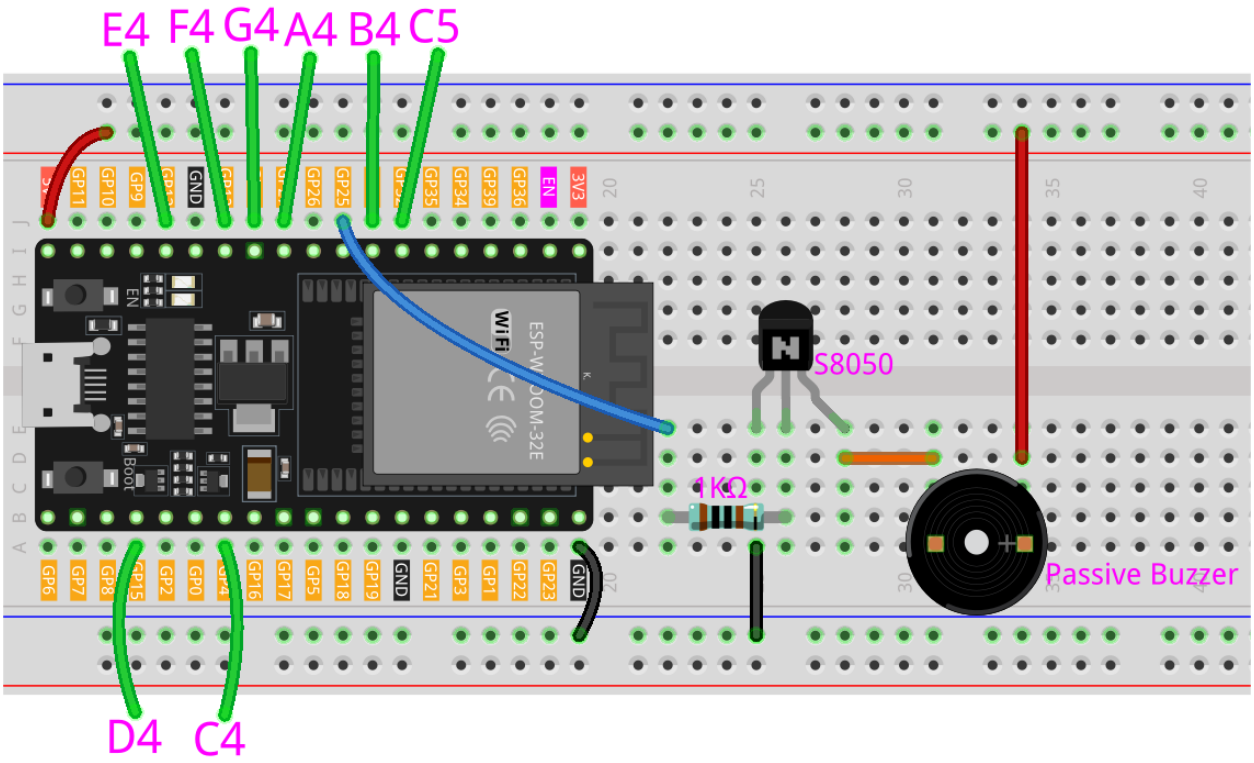
  Der Touch-Sensor auf dem ESP32 unterstützt Multitouch, was bedeutet, dass Sie mehrere Berührungsereignisse gleichzeitig erkennen können.

Schaltplan



Die Idee hinter diesem Projekt besteht darin, Berührungssensoren zu verwenden, um zu erkennen, wenn ein Benutzer einen bestimmten Pin berührt. Jeder Berührungspin ist mit einer bestimmten Note verbunden, und wenn der Benutzer einen Pin berührt, wird die entsprechende Note auf dem passiven Summer gespielt. Das Ergebnis ist eine einfache und erschwingliche Möglichkeit, das Klavierspielen zu genießen.

Verdrahtung



In diesem Projekt müssen Sie das ESP32 WROOM 32E von der Erweiterungsplatine entfernen und dann in das Steckbrett einsetzen. Dies liegt daran, dass einige Pins auf der Erweiterungsplatine mit Widerständen verbunden sind, was die Kapazität der Pins beeinflusst.

Code

Bemerkung:

- Öffnen Sie die Datei `6.1_fruit_piano.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
from machine import Pin, PWM, TouchPad
import time

# Define the touch pins and their corresponding notes
touch_pins = [4, 15, 13, 12, 14, 27, 33, 32]  # Use valid touch-capable pins
notes = [262, 294, 330, 349, 392, 440, 494, 523]

# Initialize the touch sensors
touch_sensors = [TouchPad(Pin(pin)) for pin in touch_pins]

# Initialize the buzzer
buzzer = PWM(Pin(25), duty=0)

# Function to play a tone
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
def play_tone(frequency, duration):
    buzzer.freq(frequency)
    buzzer.duty(512)
    time.sleep_ms(duration)
    buzzer.duty(0)

touch_threshold = 200

# Main loop to check for touch inputs and play the corresponding note
while True:
    for i, touch_sensor in enumerate(touch_sensors):
        value = touch_sensor.read()
        print(i,value)
        if value < touch_threshold:
            play_tone(notes[i], 100)
            time.sleep_ms(50)
        time.sleep(0.01)
```

Sie können Früchte an diese ESP32-Pins anschließen: 4, 15, 13, 12, 14, 27, 33, 32.

Wenn das Skript läuft, spielen das Berühren dieser Früchte die Noten C, D, E, F, G, A, H und C5.

Bemerkung: `Touch_threshold` muss basierend auf der Leitfähigkeit verschiedener Früchte angepasst werden.

Sie können das Skript zuerst ausführen, um die von der Shell ausgegebenen Werte zu sehen.

```
0 884
1 801
2 856
3 964
4 991
5 989
6 1072
7 1058
```

Nach dem Berühren der Früchte an den Pins 12, 14 und 27 lauten die ausgegebenen Werte wie folgt. Daher habe ich den `touch_threshold` auf 200 gesetzt, was bedeutet, dass ein Wert unter 200 als Berührung betrachtet wird und der Summer verschiedene Noten abgibt.

```
0 882
1 810
2 799
3 109
4 122
5 156
6 1068
7 1055
```

3.34 6.2 Fließendes Licht

Haben Sie schon einmal daran gedacht, Ihrer Wohnfläche ein spaßiges und interaktives Element hinzuzufügen? Dieses Projekt umfasst die Erstellung eines Lauflichts mit einem WS2812-LED-Streifen und einem Hindernisvermeidungsmodul. Das Lauflicht ändert seine Richtung, wenn ein Hindernis erkannt wird, und wird so zu einer spannenden Ergänzung für Ihre Haus- oder Bürodekoration.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

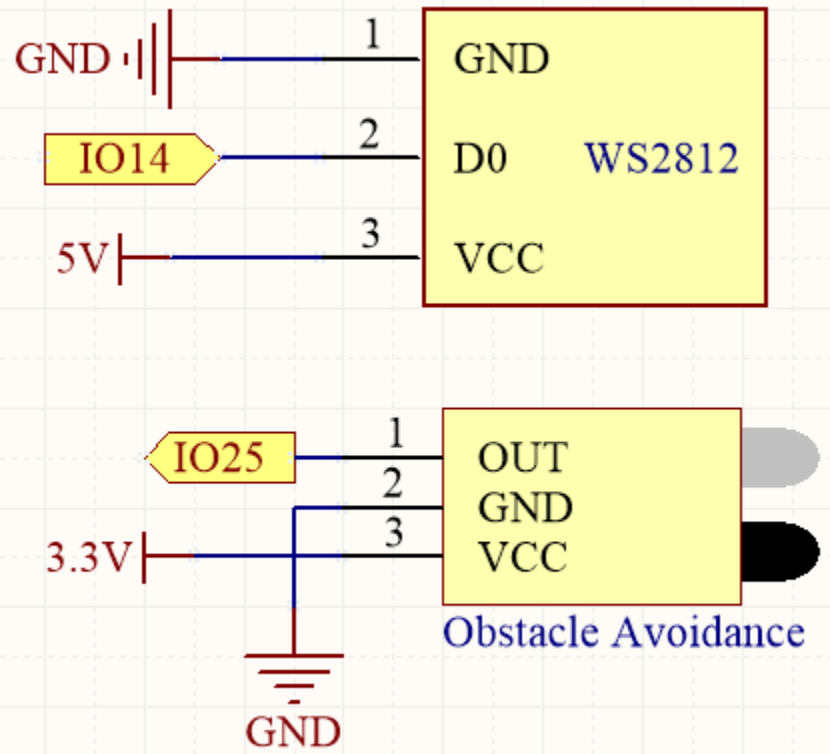
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

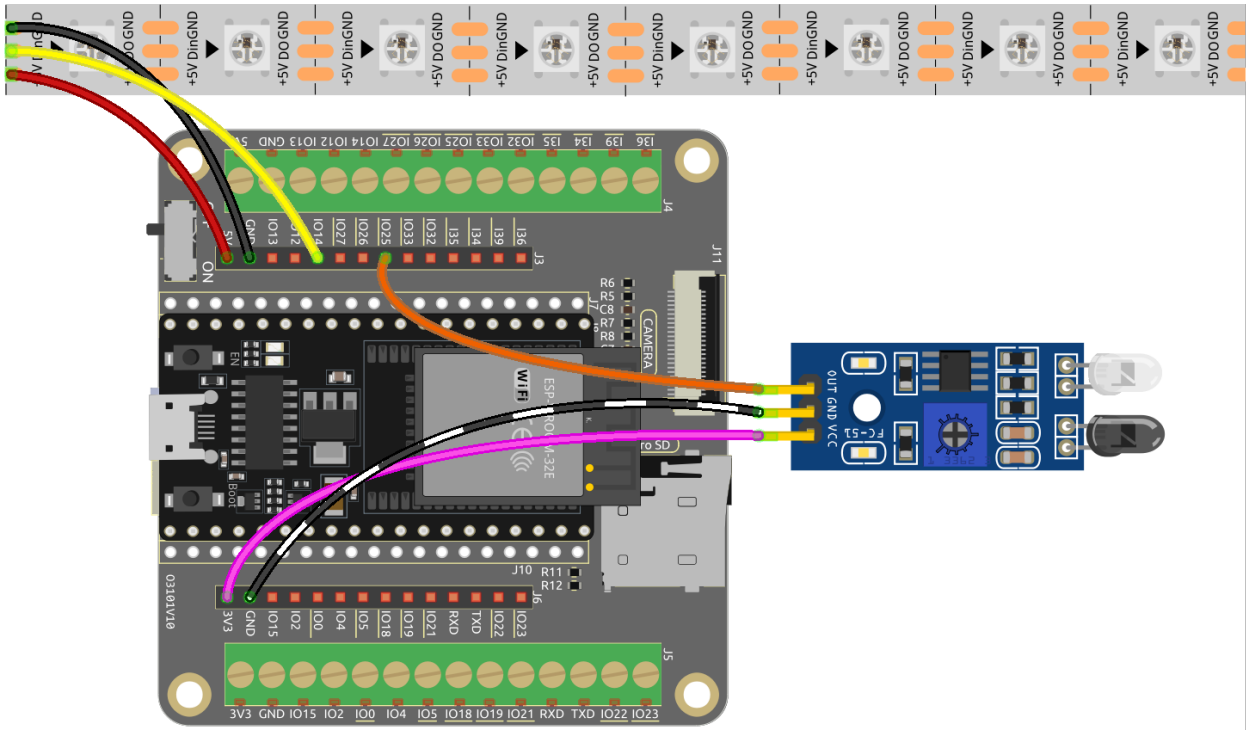
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Modul zur Hindernisvermeidung	
WS2812 RGB 8 LEDs Leiste	

Schaltplan



Der WS2812-LED-Streifen besteht aus einer Reihe von einzelnen LEDs, die programmiert werden können, um verschiedene Farben und Muster anzuzeigen. In diesem Projekt ist der Streifen so eingestellt, dass er ein Lauflicht anzeigt, das sich in eine bestimmte Richtung bewegt und seine Richtung ändert, wenn ein Hindernis vom Hindernisvermeidungsmodul erkannt wird.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `6.2_flowing_led.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
from machine import Pin
import neopixel
import time
import random

# Set the number of pixels for the running light
num_pixels = 8

# Set the data pin for the RGB LED strip
data_pin = Pin(14, Pin.OUT)

# Initialize the RGB LED strip object
pixels = neopixel.NeoPixel(data_pin, num_pixels)

# Initialize the avoid sensor
avoid = Pin(25, Pin.IN)

# Initialize the direction variable
direction_forward = True

# Initialize the reverse direction flag
reverse_direction = False

# Continuously loop the running light
while True:

# Read the input from the infrared sensor
avoid_value = avoid.value()

# Generate a random color for the current pixel
color = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))

# If no obstacle is detected
if avoid_value:
      for i in range(num_pixels):

            # Turn on the current pixel with the random color
            pixels[i] = color

            # Update the RGB LED strip display
            pixels.write()
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
            # Turn off the current pixel
            pixels[i] = (0, 0, 0)
            time.sleep_ms(100)

# If detects an obstacle, change the direction of the LED strip
else:
      for i in range(num_pixels-1, -1, -1):

            pixels[i] = color
            pixels.write()
            pixels[i] = (0, 0, 0)
            time.sleep_ms(100)
```

Die LEDs auf dem RGB-Streifen leuchten nacheinander auf, wenn das Skript läuft. Sobald sich ein Objekt vor dem Hindernisvermeidungsmodul befindet, leuchten die LEDs auf dem RGB-Streifen nacheinander in die entgegengesetzte Richtung auf.

3.35 6.3 Licht-Theremin

Das Theremin ist ein elektronisches Musikinstrument, das keinen physischen Kontakt erfordert. Je nach Position der Hand des Spielers erzeugt es verschiedene Töne.

Sein Steuerungsteil besteht normalerweise aus zwei Metallantennen, die die Position der Hände des Thereministen erfassen und Oszillatoren mit einer Hand und die Lautstärke mit der anderen steuern. Die elektrischen Signale des Theremins werden verstärkt und an einen Lautsprecher gesendet.

Wir können das gleiche Instrument nicht mit dem ESP32 reproduzieren, aber wir können einen Fotowiderstand und einen passiven Summer verwenden, um ein ähnliches Spielgefühl zu erreichen.

- Theremin - Wikipedia

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

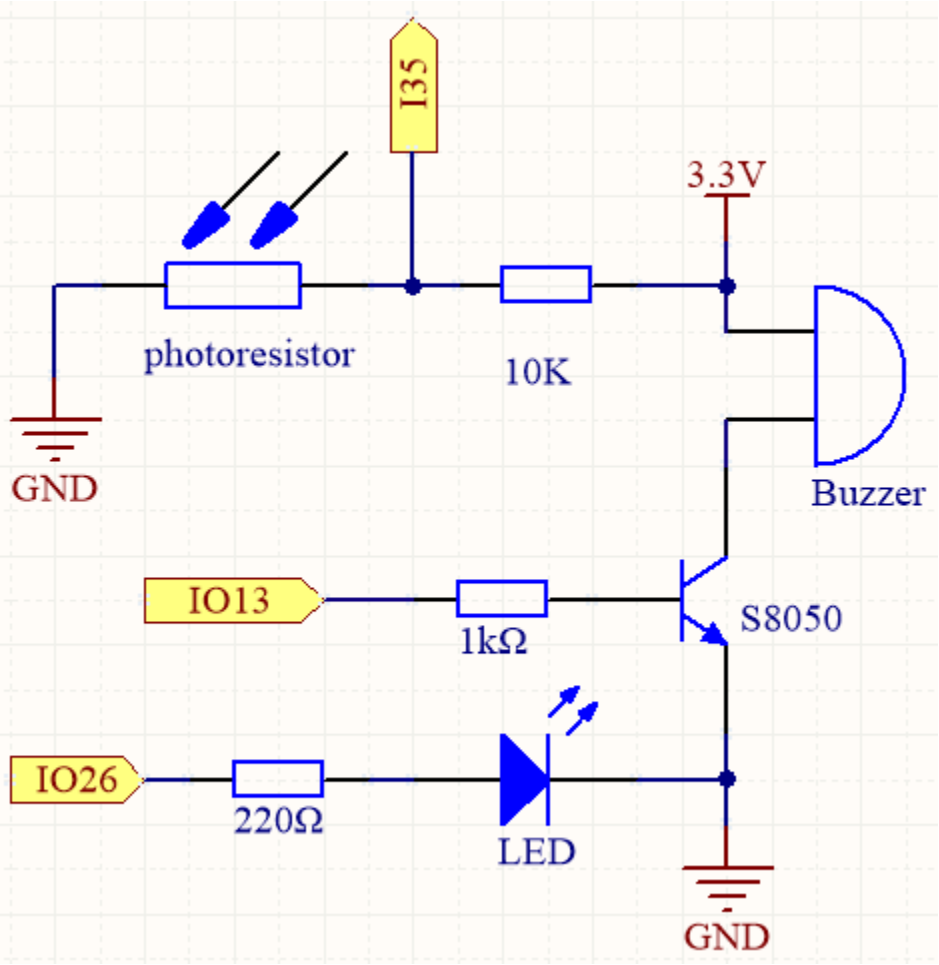
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	
Fotowiderstand	
Summer	
Transistor	

Schaltplan

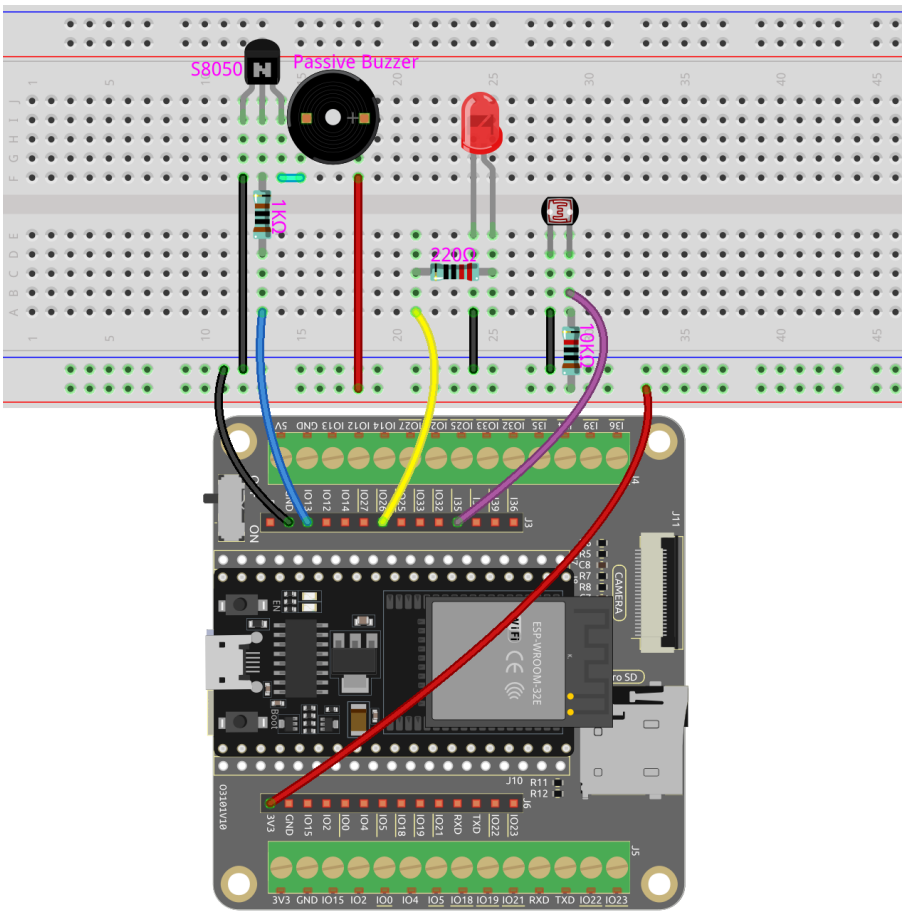


Vor dem Start des Projekts kalibrieren Sie den Bereich der Lichtintensität, indem Sie Ihre Hand über den Fotowider-

stand bewegen. Die an IO26 angeschlossene LED dient während des Kalibrierungsprozesses als Indikator. Wenn die LED leuchtet, signalisiert dies den Beginn der Kalibrierung, und wenn sie ausgeschaltet ist, zeigt sie das Ende der Kalibrierung an.

Wenn Sie Ihre Hand über den Fotowiderstand bewegen, ändert sich dessen Wert entsprechend. Nutzen Sie diese Änderung, um den Summer zu steuern und verschiedene musikalische Noten zu spielen. Jede Variation im Wert des Fotowiderstands kann einer bestimmten musikalischen Note zugeordnet werden, wodurch der Summer eine Melodie erzeugt, während Sie Ihre Hand über den Fotowiderstand bewegen.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `6.3_light_theremin.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
from machine import Pin, PWM, ADC
import time
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
# Initialize LED pin
led = Pin(26, Pin.OUT)

# Initialize light sensor
sensor = ADC(Pin(35))
sensor.atten(ADC.ATTN_11DB)

# Initialize buzzer
buzzer = PWM(Pin(13), freq=440, duty=0)

light_low=4095
light_high=0

# Map the interval of input values to output values
def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

# Create a tone using the specified pin, frequency, and duration
def tone(pin,frequency,duration):
    pin.freq(frequency)
    pin.duty(512)
    time.sleep_ms(duration)
    pin.duty(0)

# Calibrate the photoresistor's maximum and minimum values in 5 seconds.
timer_init_start = time.ticks_ms()
led.value(1) # turn on the LED
while time.ticks_diff(time.ticks_ms(), timer_init_start)<5000:
    light_value = sensor.read()
    if light_value > light_high:
        light_high = light_value
    if light_value < light_low:
        light_low = light_value
led.value(0) # turn off the LED

# Play the tones based on the light values
while True:
    light_value  = sensor.read()
    pitch = int(interval_mapping(light_value,light_low,light_high,50,6000))
    if pitch > 50 :
        tone(buzzer,pitch,20)
    time.sleep_ms(10)
```

Nach dem Start des Programms schaltet sich die LED ein und bietet uns ein fünfsekündiges Fenster, um den Erkennungsbereich des Fotowiderstands zu kalibrieren.

Die Kalibrierung ist ein entscheidender Schritt, da sie verschiedene Lichtbedingungen berücksichtigt, auf die wir bei der Verwendung des Geräts stoßen können, wie unterschiedliche Lichtintensitäten zu verschiedenen Tageszeiten. Darüber hinaus berücksichtigt der Kalibrierungsprozess die Entfernung zwischen unseren Händen und dem Fotowiderstand, die den spielbaren Bereich des Instruments bestimmt.

Sobald die Kalibrierungsphase beendet ist, schaltet sich die LED aus und signalisiert, dass wir das Instrument jetzt spielen können, indem wir unsere Hände über den Fotowiderstand bewegen. Diese Einrichtung ermöglicht es uns, Musik zu erzeugen, indem wir die Höhe unserer Hände anpassen, was ein interaktives und unterhaltsames Erlebnis

bietet.

3.36 6.4 Einparkhilfe

Stellen Sie sich Folgendes vor: Sie sitzen in Ihrem Auto und wollen in eine enge Parklücke zurücksetzen. Mit unserem Projekt haben Sie ein Ultraschallmodul am Heck Ihres Fahrzeugs montiert, das als digitales Auge fungiert. Sobald Sie den Rückwärtsgang einlegen, erwacht das Modul zum Leben, sendet Ultraschallimpulse aus, die von Hindernissen hinter Ihnen abprallen.

Das Magische passiert, wenn diese Impulse zum Modul zurückkehren. Es berechnet blitzschnell die Entfernung zwischen Ihrem Auto und den Objekten und verwandelt diese Daten in eine Echtzeit-Visuelle Rückmeldung, die auf einem lebendigen LCD-Bildschirm angezeigt wird. Sie erleben dynamische, farbcodierte Indikatoren, die die Nähe zu Hindernissen darstellen und sicherstellen, dass Sie ein kristallklares Verständnis der Umgebung haben.

Aber damit nicht genug. Um Sie noch weiter in dieses Fahrerlebnis einzutauchen, haben wir einen lebhaften Summer eingebaut. Nähert sich Ihr Auto einem Hindernis, intensiviert sich das Tempo des Summers und schafft eine akustische Symphonie von Warnungen. Es ist, als hätten Sie ein persönliches Orchester, das Sie durch die Komplexitäten des Einparkens leitet.

Dieses innovative Projekt kombiniert Spitzentechnologie mit einer interaktiven Benutzeroberfläche und macht Ihr Einparkerlebnis sicher und stressfrei. Mit dem Ultraschallmodul, dem LCD-Display und dem lebhaften Summer, die harmonisch zusammenarbeiten, fühlen Sie sich ermächtigt und sicher beim Manövrieren auf engem Raum und können sich auf die Freude am Fahren konzentrieren.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

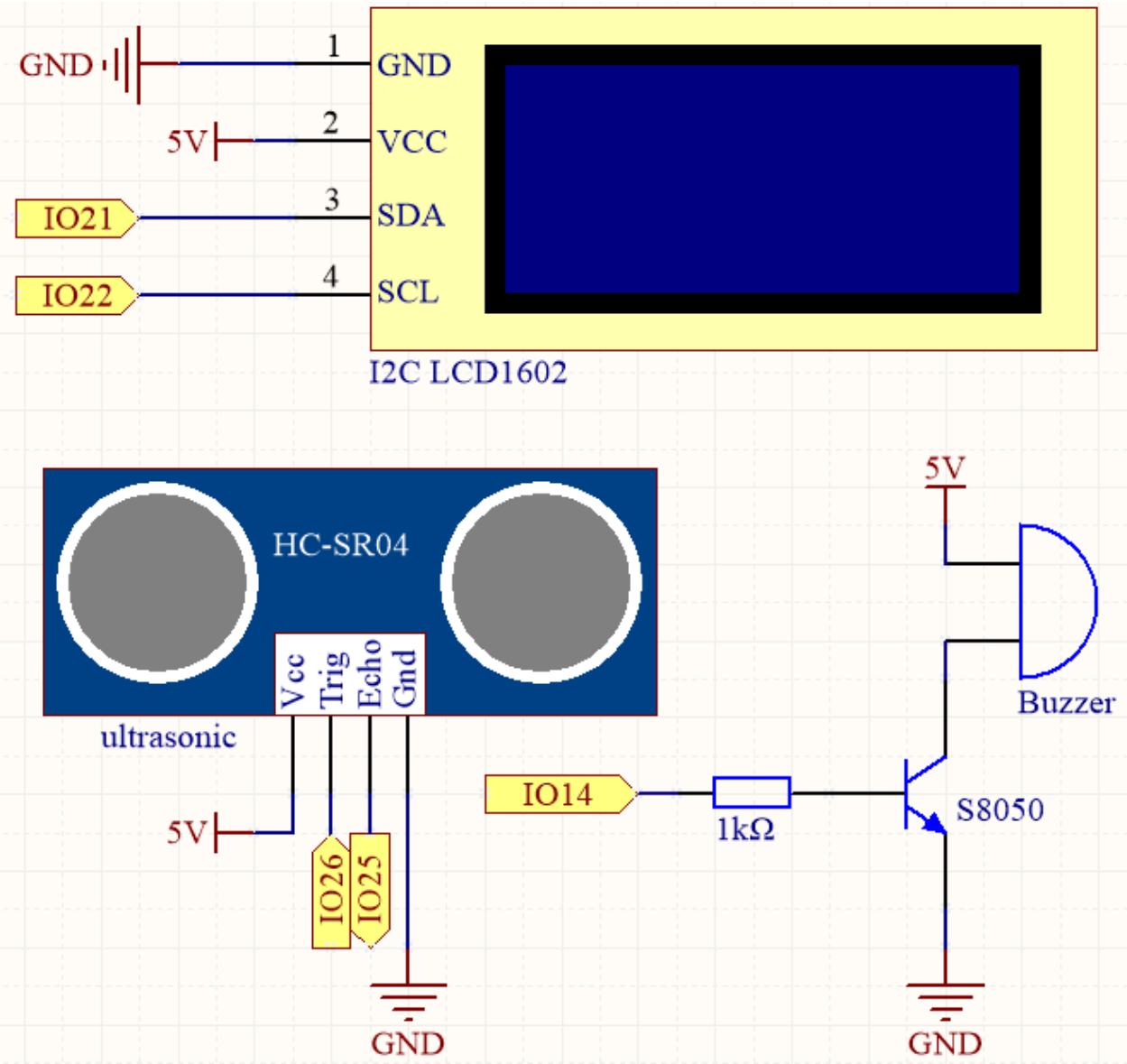
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

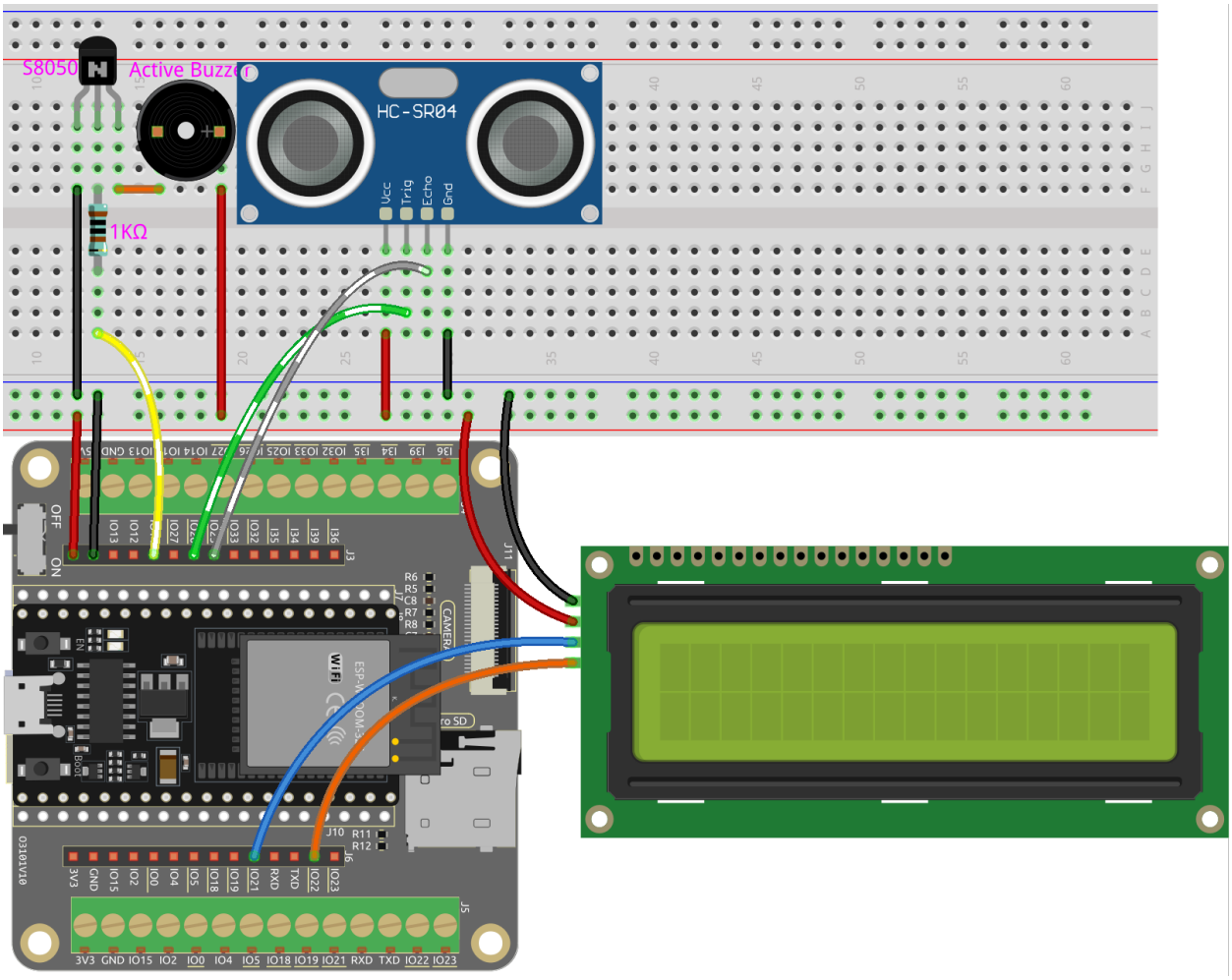
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Ultraschall-Modul	
Summer	-
Transistor	
I2C LCD1602	

Schaltplan



Der Ultraschallsensor im Projekt sendet hochfrequente Schallwellen aus und misst die Zeit, die diese Wellen benötigen, um nach dem Aufprall auf ein Objekt zurückzuprallen. Durch die Analyse dieser Daten kann die Entfernung zwischen dem Sensor und dem Objekt berechnet werden. Um eine Warnung zu geben, wenn das Objekt zu nahe ist, wird ein Summer verwendet, um ein hörbares Signal zu erzeugen. Zusätzlich wird die gemessene Entfernung auf einem LCD-Bildschirm zur einfachen Visualisierung angezeigt.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `6.4_reversing_aid.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
# Import required libraries
from machine import Pin
import time
from lcd1602 import LCD
import _thread

# Initialize the buzzer
buzzer = Pin(14, Pin.OUT)

# Initialize the ultrasonic module
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
TRIG = Pin(26, Pin.OUT)
ECHO = Pin(25, Pin.IN)

# Initialize the LCD1602 display
lcd = LCD()

dis = 100

# Calculate the distance
def distance():
    # Ensure trigger is off initially
    TRIG.off()
    time.sleep_us(2)  # Wait for 2 microseconds

    # Send a 10-microsecond pulse to the trigger pin
    TRIG.on()
    time.sleep_us(10)
    TRIG.off()

    # Wait for the echo pin to go high
    while not ECHO.value():
        pass

    # Record the time when the echo pin goes high
    time1 = time.ticks_us()

    # Wait for the echo pin to go low
    while ECHO.value():
        pass

    # Record the time when the echo pin goes low
    time2 = time.ticks_us()

    # Calculate the time difference between the two recorded times
    during = time.ticks_diff(time2, time1)

    # Calculate and return the distance (in cm) using the speed of sound (340 m/s)
    return during * 340 / 2 / 10000

# Thread to continuously update the ultrasonic sensor reading
def ultrasonic_thread():
    global dis
    while True:
        dis = distance()

        # Clear the LCD screen
        lcd.clear()

        # Display the distance
        lcd.write(0, 0, 'Dis: %.2f cm' % dis)
        time.sleep(0.5)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
# Start the ultrasonic sensor reading thread
_thread.start_new_thread(ultrasonic_thread, ())

# Beep function for the buzzer
def beep():
    buzzer.value(1)
    time.sleep(0.1)
    buzzer.value(0)
    time.sleep(0.1)

# Initialize the intervals variable
intervals = 10000000
previousMills = time.ticks_ms()
time.sleep(1)

# Main loop
while True:
    # Update intervals based on distance
    if dis < 0 and dis > 500:
        pass
    elif dis <= 10:
        intervals = 300
    elif dis <= 20:
        intervals = 500
    elif dis <= 50:
        intervals = 1000
    else:
        intervals = 2000

    # Print the distance if it's not -1
    if dis != -1:
        print('Distance: %.2f' % dis)
    time.sleep_ms(100)

    # Check if it's time to beep
    currentMills = time.ticks_ms()
    if time.ticks_diff(currentMills, previousMills) >= intervals:
        beep()
        previousMills = currentMills
```

- Wenn das Skript läuft, wird das Ultraschallmodul kontinuierlich die Entfernung von Hindernissen vor ihm erkennen und die Entfernung sowohl auf der Shell als auch auf dem I2C LCD1602 anzeigen.
- Je näher das Hindernis kommt, desto schneller wird die Piepfrequenz des Summers.
- Die Funktion `ultrasonic_thread()` läuft in einem separaten Thread, damit sie die Entfernungsmessung kontinuierlich aktualisieren kann, ohne die Hauptloop zu blockieren.

Bemerkung: Wenn der Code und die Verdrahtung korrekt sind, das LCD jedoch immer noch keinen Inhalt anzeigt, können Sie das Potentiometer auf der Rückseite justieren, um den Kontrast zu erhöhen.

3.37 6.5 Farbverlauf

Sind Sie bereit, eine Welt voller Farben zu erleben? Dieses Projekt nimmt Sie mit auf eine magische Reise, bei der Sie einen LED-Streifen steuern und sanfte Farbübergänge erzielen können. Egal, ob Sie etwas Farbe in Ihre Raumdekoration bringen oder ein spannendes Programmierprojekt suchen – dieses Projekt hat für jeden etwas zu bieten. Lassen Sie uns gemeinsam in diese farbenfrohe Welt eintauchen!

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

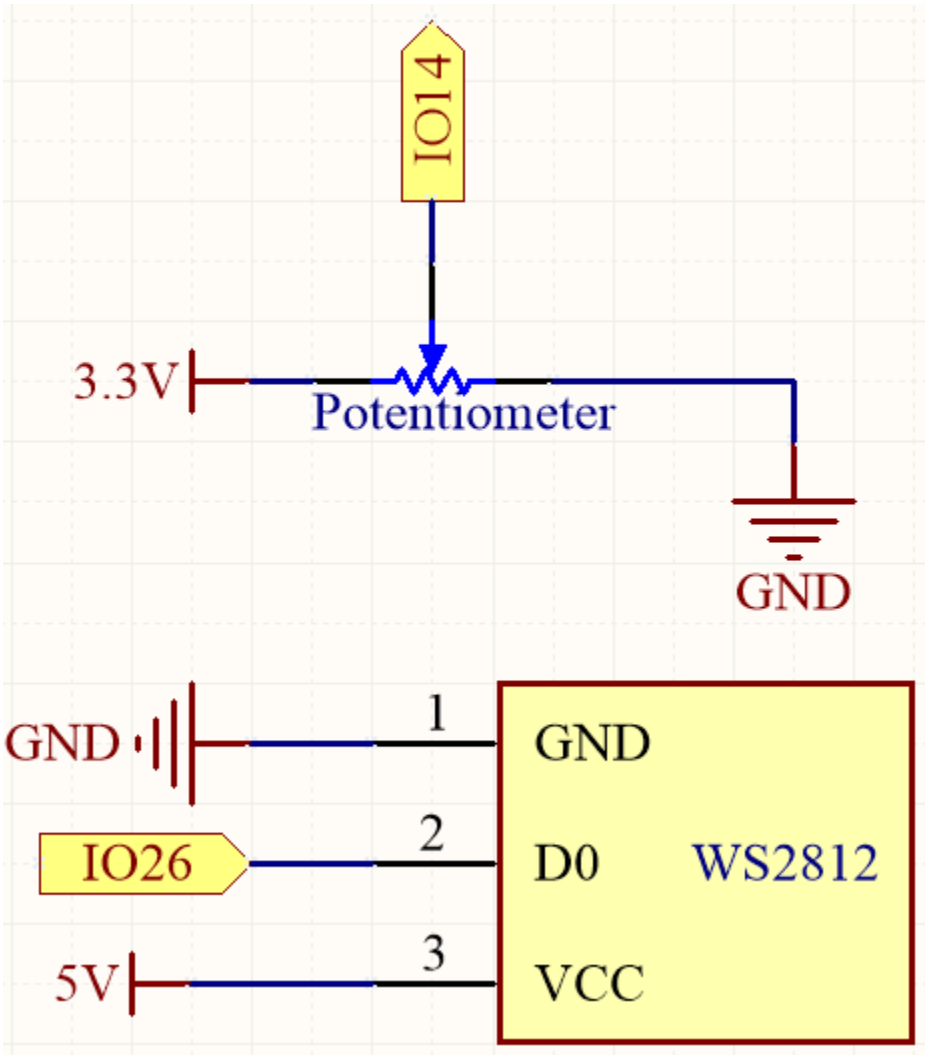
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

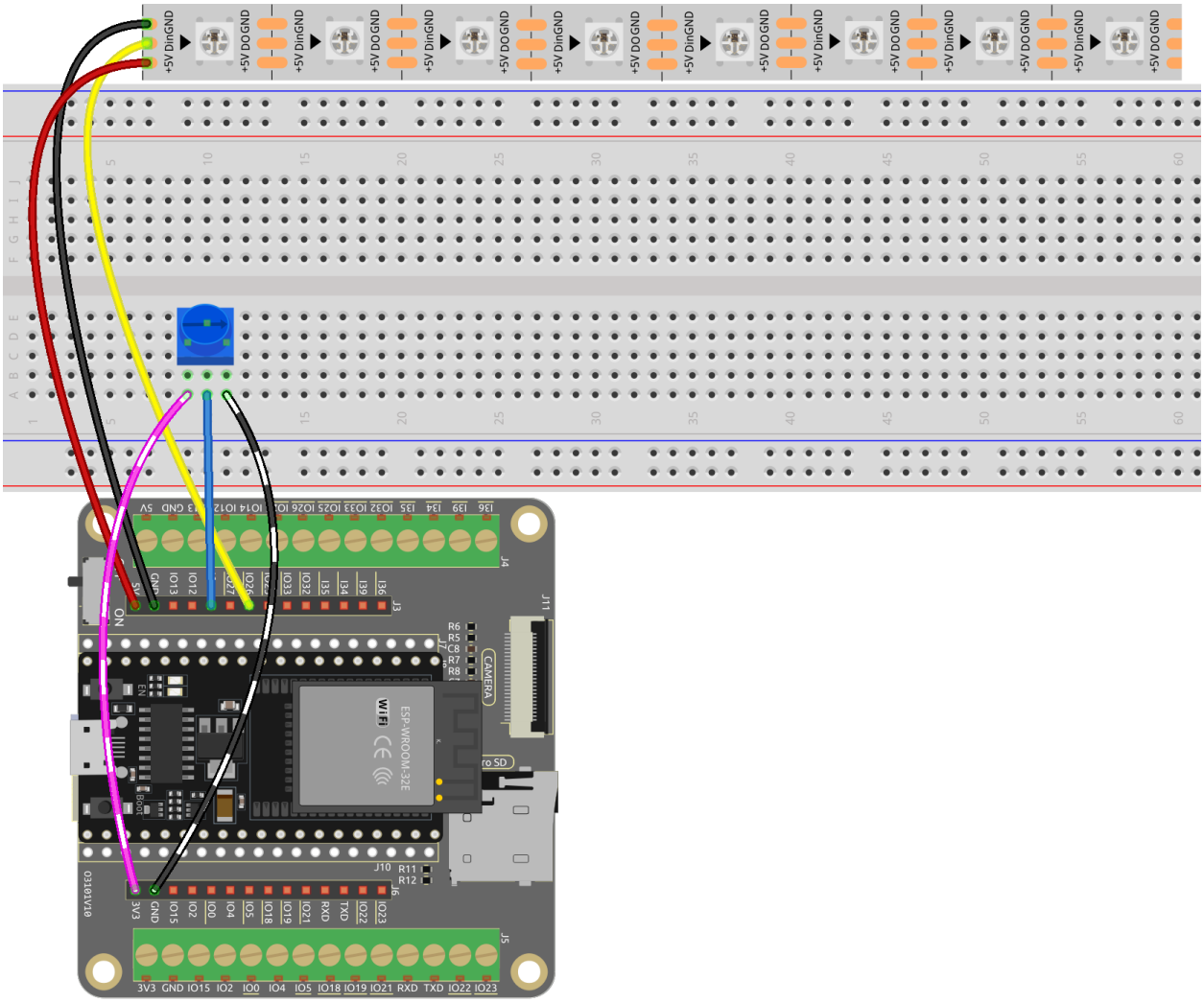
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Potentiometer	
WS2812 RGB 8 LEDs Leiste	

Schaltplan



Dieses Projekt verwendet einen LED-Streifen und ein Potentiometer, um einen Farbmischeffekt zu erzeugen. Das Potentiometer wird verwendet, um den Farbwert der LED anzupassen, der dann mit einer Farbumwandlungsfunktion in RGB-Werte umgewandelt wird. Die RGB-Werte werden dann verwendet, um die Farbe der LED zu aktualisieren.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `6.5_color_gradient.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
from machine import Pin, ADC, PWM
import neopixel
import time

NUM_LEDS = 8  # Number of LEDs in the strip
PIN_NUM = 26  # LED strip
POT_PIN = 14  # Potentiometer
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
# Initialize the potentiometer
potentiometer = ADC(Pin(POT_PIN))
potentiometer.atten(ADC.ATTN_11DB)

# Initialize the NeoPixel LED strip
np = neopixel.NeoPixel(Pin(PIN_NUM), NUM_LEDS)

# Function to convert HSL color space to RGB color space
def hsl_to_rgb(h, s, l):
    # Helper function to convert hue to RGB
    def hue_to_rgb(p, q, t):
        if t < 0:
            t += 1
        if t > 1:
            t -= 1
        if t < 1/6:
            return p + (q - p) * 6 * t
        if t < 1/2:
            return q
        if t < 2/3:
            return p + (q - p) * (2/3 - t) * 6
        return p

    if s == 0:
        r = g = b = l
    else:
        q = l * (1 + s) if l < 0.5 else l + s - l * s
        p = 2 * l - q
        r = hue_to_rgb(p, q, h + 1/3)
        g = hue_to_rgb(p, q, h)
        b = hue_to_rgb(p, q, h - 1/3)

    return (int(r * 255), int(g * 255), int(b * 255))

# Function to set the color of all LEDs in the strip
def set_color(np, color):
    for i in range(NUM_LEDS):
        np[i] = color
    np.write()

# Main loop
while True:
    # Read the potentiometer value and normalize it to the range [0, 1]
    pot_value = potentiometer.read() / 4095.0
    hue = pot_value  # Set hue value based on the potentiometer's position
    saturation = 1  # Set saturation to 1 (fully saturated)
    lightness = 0.5  # Set lightness to 0.5 (halfway between black and white)

    # Convert the HSL color to RGB
    current_color = hsl_to_rgb(hue, saturation, lightness)

    # Set the LED strip color based on the converted RGB value
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    set_color(np, current_color)

    # Sleep for a short period to allow for smooth transitions
    time.sleep(0.1)
```

Während der Code läuft, drehen Sie langsam das Potentiometer und Sie werden sehen, wie die Farbe des RGB-Streifens von Rot zu Lila übergeht.

3.38 6.6 Digitaler Würfel

Dieses Projekt baut auf dem *2.5 Ziffernanzeige*-Projekt auf, indem ein Knopf hinzugefügt wird, um die auf dem Siebensegment-Display angezeigte Zahl zu steuern.

Wenn der Knopf gedrückt wird, scrollt das Siebensegment-Display durch die Zahlen 1-6, und wenn der Knopf losgelassen wird, zeigt es eine zufällige Zahl an.

Dieser Zyklus setzt sich jedes Mal fort, wenn der Knopf gedrückt wird.

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

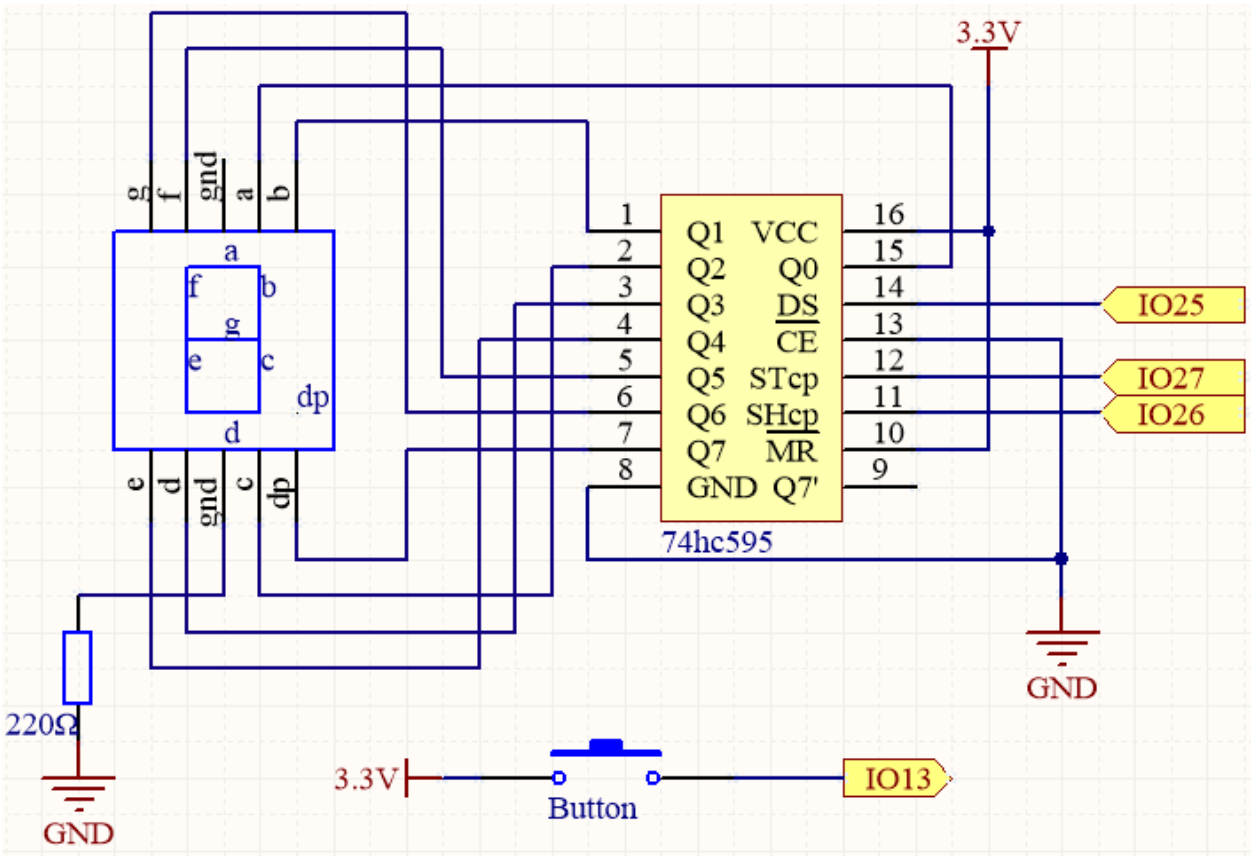
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
74HC595	
7-Segment-Anzeige	
Taste	

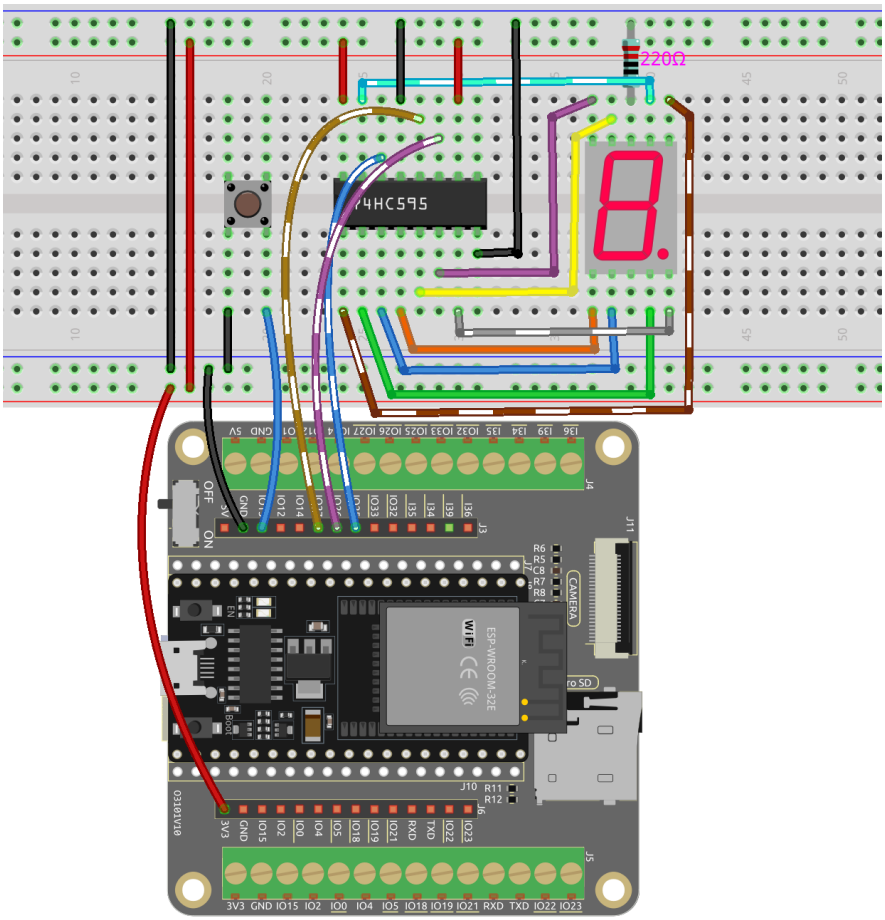
Schaltplan



Dieses Projekt baut auf dem *2.5 Ziffernanzeige*-Projekt auf, indem ein Knopf hinzugefügt wird, um die auf dem Siebensegment-Display angezeigte Zahl zu steuern.

Der Knopf ist direkt mit IO13 verbunden, ohne einen externen Pull-Up- oder Pull-Down-Widerstand, da IO13 einen internen Pull-Up-Widerstand von 47K hat, was den Bedarf an einem zusätzlichen externen Widerstand überflüssig macht.

Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `6.6_digital_dice.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.

```
import machine
import time
import random

# Define the segment code for a common anode 7-segment display
SEGCODE = [0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f]

# Initialize the pins for the 74HC595 shift register
sdi = machine.Pin(25, machine.Pin.OUT)  # DS
rclk = machine.Pin(27, machine.Pin.OUT)  # STcp
srclk = machine.Pin(26, machine.Pin.OUT)  # SHcp
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
button = machine.Pin(13, machine.Pin.IN) # Button pin

# Define the hc595_shift function to shift data into the 74HC595 shift register
def hc595_shift(dat):
    # Set the RCLK pin to low
    rclk.off()

    # Iterate through each bit (from 7 to 0)
    for bit in range(7, -1, -1):
        # Extract the current bit from the input data
        value = 1 & (dat >> bit)

        # Set the SRCLK pin to low
        srclk.off()

        # Set the value of the SDI pin
        sdi.value(value)

        # Clock the current bit into the shift register by setting the SRCLK pin to high
        srclk.on()

    # Latch the data into the storage register by setting the RCLK pin to high
    rclk.on()

# Initialize the random seed
random.seed(time.ticks_us())


num = 1
button_state = False

# Define the button callback function to toggle the button state
def button_callback(pin):
    global button_state
    button_state = not button_state

# Attach the button callback function to the falling edge of the button pin
button.irq(trigger=machine.Pin.IRQ_FALLING, handler=button_callback)

# Continuously display the current digit on the 7-segment display, scrolling if button␣
↪is not pressed
while True:

    # Display the current digit on the 7-segment display
    hc595_shift(SEGCODE[num])

    # If the button is pressed and button state is True
    if button_state:
        pass

    # If the button is pressed again and button state is False, generate a new random␣
↪digit
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    if not button_state:
        num = random.randint(1, 6)
        time.sleep_ms(10) # Adjust this value to control the display refresh rate
```

Während das Programm läuft, wird durch das Drücken des Knopfes das Siebensegment-Display scrollen und zufällig eine Zahl zwischen 1 und 6 anzeigen.

Beim erneuten Drücken des Knopfes stoppt das Siebensegment-Display und zeigt eine bestimmte Zahl an. Drücken Sie den Knopf noch einmal, und das Siebensegment-Display wird das Durchlaufen der Ziffern fortsetzen.

3.39 6.7 Zahlenraten

Fühlen Sie sich glücklich? Wollen Sie Ihre Intuition testen und sehen, ob Sie die richtige Zahl erraten können? Dann ist das Zahlenraten-Spiel genau das Richtige für Sie!

Mit diesem Projekt können Sie ein spannendes und unterhaltsames Glücksspiel spielen.

Mit einer IR-Fernbedienung geben die Spieler Zahlen zwischen 0 und 99 ein, um zu versuchen, die zufällig generierte Glückszahl zu erraten. Das System zeigt die vom Spieler eingegebene Zahl auf einem LCD-Bildschirm an, zusammen mit Tipps für die obere und untere Grenze, um den Spieler zur richtigen Antwort zu leiten. Mit jedem Versuch kommen die Spieler der Glückszahl näher, bis schließlich jemand den Jackpot knackt und das Spiel gewinnt!

Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

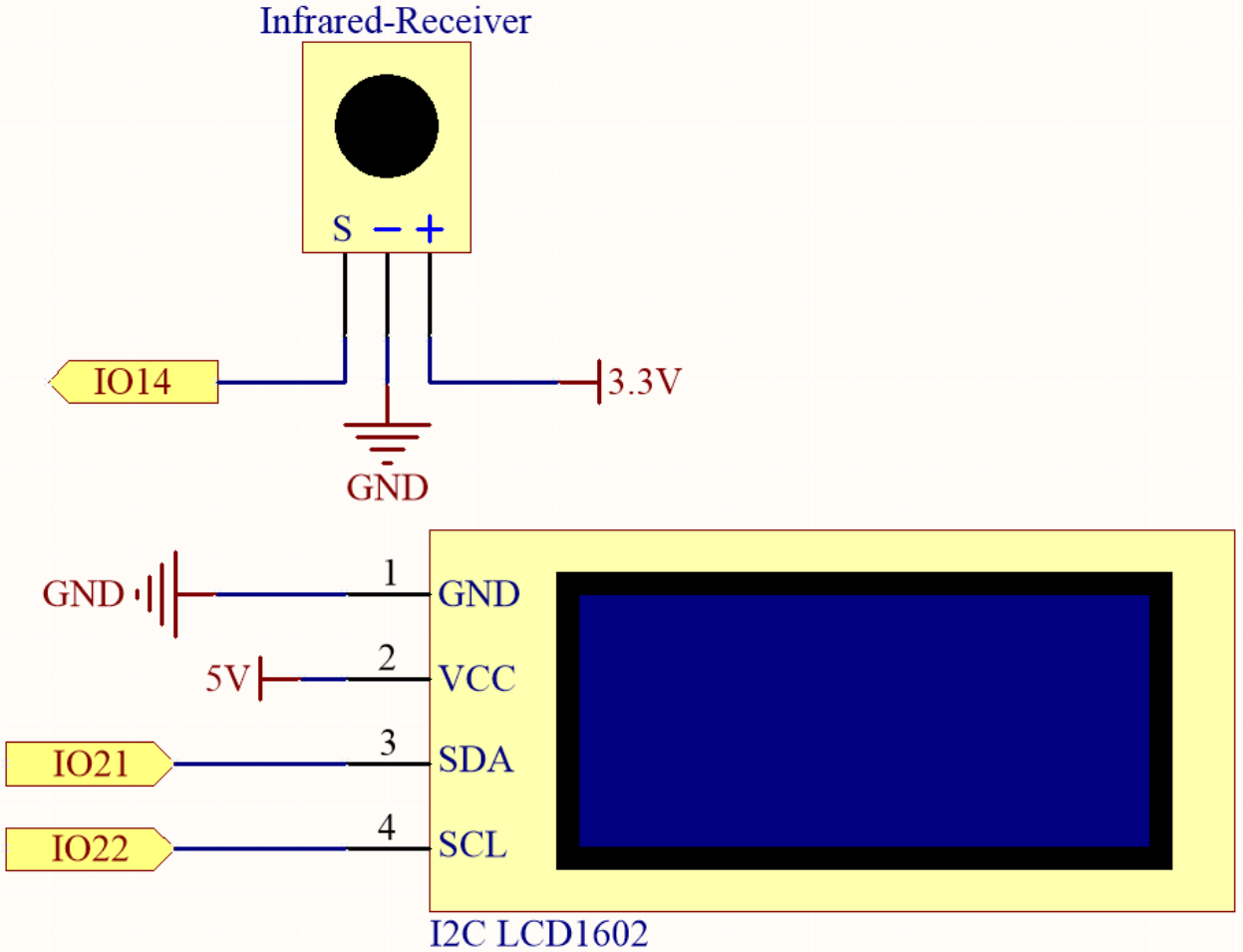
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

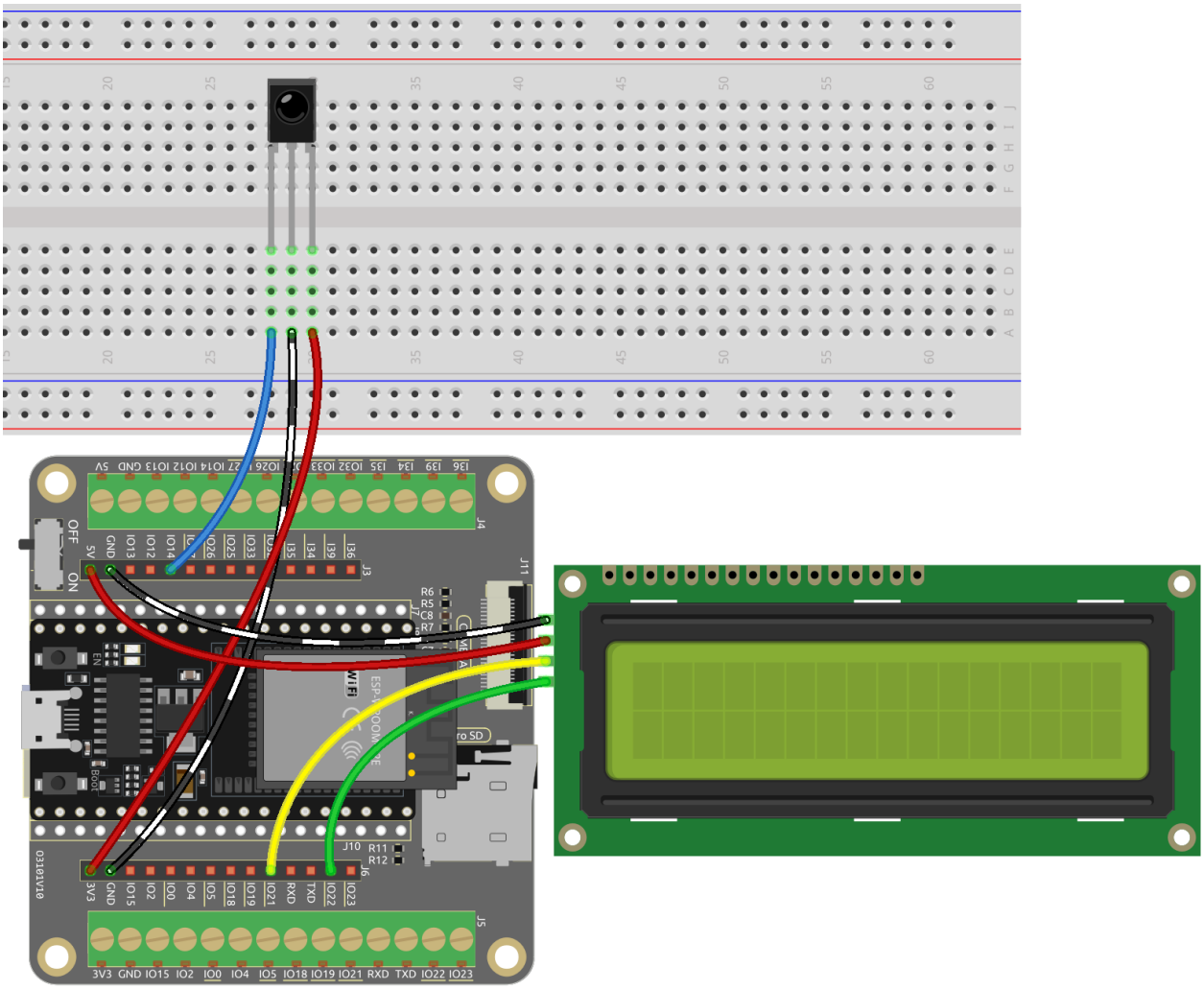
Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
IR-Empfänger	
I2C LCD1602	

Schaltplan



Verdrahtung



Code

Bemerkung:

- Öffnen Sie die Datei `6.7_game_guess_number.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren und fügen Sie den Code in Thonny ein. Klicken Sie dann auf „Run Current Script“ oder drücken Sie F5, um ihn auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der unteren rechten Ecke ausgewählt ist.
- Die Bibliotheken `lcd1602.py` und `ir_rx` werden hier verwendet. Überprüfen Sie, ob sie auf den ESP32 hochgeladen wurden. Siehe *1.4 Bibliotheken Hochladen (Wichtig)* für eine Anleitung.

```
from lcd1602 import LCD
import machine
import time
import urandom
from machine import Pin
from ir_rx.print_error import print_error
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
from ir_rx.nec import NEC_8

# IR receiver configuration
pin_ir = Pin(14, Pin.IN)

# Initialize the guessing game variables
lower = 0
upper = 99
pointValue = int(urandom.uniform(lower, upper))
count = 0

# Initialize the LCD1602 display
lcd = LCD()

# Initialize a new random value for the game
def init_new_value():
    global pointValue, upper, lower, count
    pointValue = int(urandom.uniform(lower, upper))
    print(pointValue)
    upper = 99
    lower = 0
    count = 0
    return False

# Display messages on the LCD based on the game state
def lcd_show(result):
    global count
    lcd.clear()
    if result == True:
        string = "GAME OVER!\n"
        string += "Point is " + str(pointValue)
    else:
        string = "Enter number: " + str(count) + "\n"
        string += str(lower) + " < Point < " + str(upper)
    lcd.message(string)
    return

# Process the entered number and update the game state
def number_processing():
    global upper, count, lower
    if count > pointValue:
        if count < upper:
            upper = count
    elif count < pointValue:
        if count > lower:
            lower = count
    elif count == pointValue:
        return True
    count = 0
    return False

# Process the key inputs from the IR remote control
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
def process_key(key):
    global count, lower, upper, pointValue, result
    if key == "Power":
        init_new_value()
        lcd_show(False)
    elif key == "+":
        result = number_processing()
        lcd_show(result)
        if result:
            time.sleep(5)
            init_new_value()
            lcd_show(False)
        else:
            lcd_show(False)
    elif key.isdigit():
        count = count * 10 + int(key) if count * 10 + int(key) <= 99 else count
        lcd_show(False)

# Decode the received data and return the corresponding key name
def decodeKeyValue(data):
    if data == 0x16:
        return "0"
    if data == 0x0C:
        return "1"
    if data == 0x18:
        return "2"
    if data == 0x5E:
        return "3"
    if data == 0x08:
        return "4"
    if data == 0x1C:
        return "5"
    if data == 0x5A:
        return "6"
    if data == 0x42:
        return "7"
    if data == 0x52:
        return "8"
    if data == 0x4A:
        return "9"
    if data == 0x09:
        return "+"
    if data == 0x15:
        return "-"
    if data == 0x7:
        return "EQ"
    if data == 0x0D:
        return "U/SD"
    if data == 0x19:
        return "CYCLE"
    if data == 0x44:
        return "PLAY/PAUSE"
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    if data == 0x43:
        return "FORWARD"
    if data == 0x40:
        return "BACKWARD"
    if data == 0x45:
        return "POWER"
    if data == 0x47:
        return "MUTE"
    if data == 0x46:
        return "MODE"
    return "ERROR"

def callback(data, addr, ctrl):
    if data < 0:
        pass
    else:
        key = decodeKeyValue(data)
        if key != "ERROR":
            process_key(key)

# Initialize the IR receiver object with the callback function
ir = NEC_8(pin_ir, callback)

# ir.error_function(print_error)

# Initialize the game with a new random value
init_new_value()

# Show the initial game state on the LCD
lcd_show(False)

try:
    while True:
        pass
except KeyboardInterrupt:
    ir.close()
```

- Wenn das Programm ausgeführt wird, wird eine geheime Zahl erzeugt, die jedoch nicht auf dem LCD angezeigt wird. Ihre Aufgabe ist es, diese Zahl zu erraten.
- Drücken Sie die von Ihnen vermutete Zahl auf der Fernbedienung und bestätigen Sie mit der + Taste.
- Gleichzeitig wird der auf dem I2C LCD1602 angezeigte Bereich verkleinert, und Sie müssen die entsprechende Zahl basierend auf diesem neuen Bereich eingeben.
- Sollten Sie die Glückszahl zufälligerweise erraten oder nicht erraten, erscheint `GAME OVER!`.

Bemerkung: Falls der Code und die Verkabelung korrekt sind, aber das LCD dennoch keinen Inhalt anzeigt, können Sie das Potentiometer auf der Rückseite justieren, um den Kontrast zu erhöhen.

Wie funktioniert das?

Im Folgenden eine detaillierte Analyse eines Teils des Codes.

1. Initialisierung der Variablen für das Ratespiel.

```
lower = 0
upper = 99
pointValue = int(urandom.uniform(lower, upper))
count = 0
```

   - `lower` und `upper` begrenzen die geheime Zahl.
   - Die geheime Zahl (`pointValue`) wird zufällig zwischen `lower` und `upper` generiert.
   - Der aktuelle Tipp des Benutzers (`count`).

2. Diese Funktion setzt die Werte des Ratespiels zurück und generiert eine neue Geheimzahl.

```
def init_new_value():
    global pointValue, upper, lower, count
    pointValue = int(urandom.uniform(lower, upper))
    print(pointValue)
    upper = 99
    lower = 0
    count = 0
    return False
```

3. Diese Funktion zeigt den aktuellen Spielstatus auf dem LCD-Bildschirm an.

```
def lcd_show(result):
    global count
    lcd.clear()
    if result == True:
        string = "GAME OVER!\n"
        string += "Point is " + str(pointValue)
    else:
        string = "Enter number: " + str(count) + "\n"
        string += str(lower) + " < Point < " + str(upper)
    lcd.message(string)
    return
```

   - Wenn das Spiel vorbei ist (`result=True`), wird `GAME OVER!` und die Geheimzahl angezeigt.
   - Andernfalls wird der aktuelle Tipp (`count`) und der aktuelle Ratebereich (`lower` bis `upper`) angezeigt.

4. Diese Funktion verarbeitet den aktuellen Tipp des Benutzers (`count`) und aktualisiert den Ratebereich.

```
def number_processing():
    global upper, count, lower
    if count > pointValue:
        if count < upper:
            upper = count
    elif count < pointValue:
        if count > lower:
            lower = count
    elif count == pointValue:
        return True
    count = 0
    return False
```

- Wenn der aktuelle Tipp (`count`) höher als die Geheimzahl ist, wird die obere Grenze aktualisiert.
- Wenn der aktuelle Tipp (`count`) niedriger als die Geheimzahl ist, wird die untere Grenze aktualisiert.
- Wenn der aktuelle Tipp (`count`) gleich der Geheimzahl ist, gibt die Funktion `True` zurück (Spiel vorbei).

5. Diese Funktion verarbeitet die Tastendrücke, die vom IR-Fernbedienung empfangen werden.

```
def process_key(key):
    global count, lower, upper, pointValue, result
    if key == "Power":
        init_new_value()
        lcd_show(False)
    elif key == "+":
        result = number_processing()
        lcd_show(result)
        if result:
            time.sleep(5)
            init_new_value()
            lcd_show(False)
        else:
            lcd_show(False)
    elif key.isdigit():
        count = count * 10 + int(key) if count * 10 + int(key) <= 99 else␣
↪count
        lcd_show(False)
```

- Wenn die Taste `Power` gedrückt wird, wird das Spiel zurückgesetzt.
- Wenn die Taste + gedrückt wird, wird der aktuelle Tipp (`count`) verarbeitet und der Spielstatus aktualisiert.
- Wenn eine Zifferntaste gedrückt wird, wird der aktuelle Tipp (`count`) mit der neuen Ziffer aktualisiert.

6. Diese Callback-Funktion wird ausgelöst, wenn der IR-Empfänger ein Signal empfängt.

```
def callback(data, addr, ctrl):
    if data < 0:
        pass
    else:
        key = decodeKeyValue(data)
        if key != "ERROR":
            process_key(key)
```

3.40 6.8 Pflanzenüberwachung

Willkommen beim Pflanzenüberwachungsprojekt!

In diesem Projekt werden wir ein ESP32-Board verwenden, um ein System zu entwickeln, das uns hilft, unsere Pflanzen zu pflegen. Mit diesem System können wir Temperatur, Luftfeuchtigkeit, Bodenfeuchtigkeit und Lichtverhältnisse unserer Pflanzen überwachen und sicherstellen, dass sie die notwendige Pflege und Aufmerksamkeit erhalten, um zu gedeihen.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

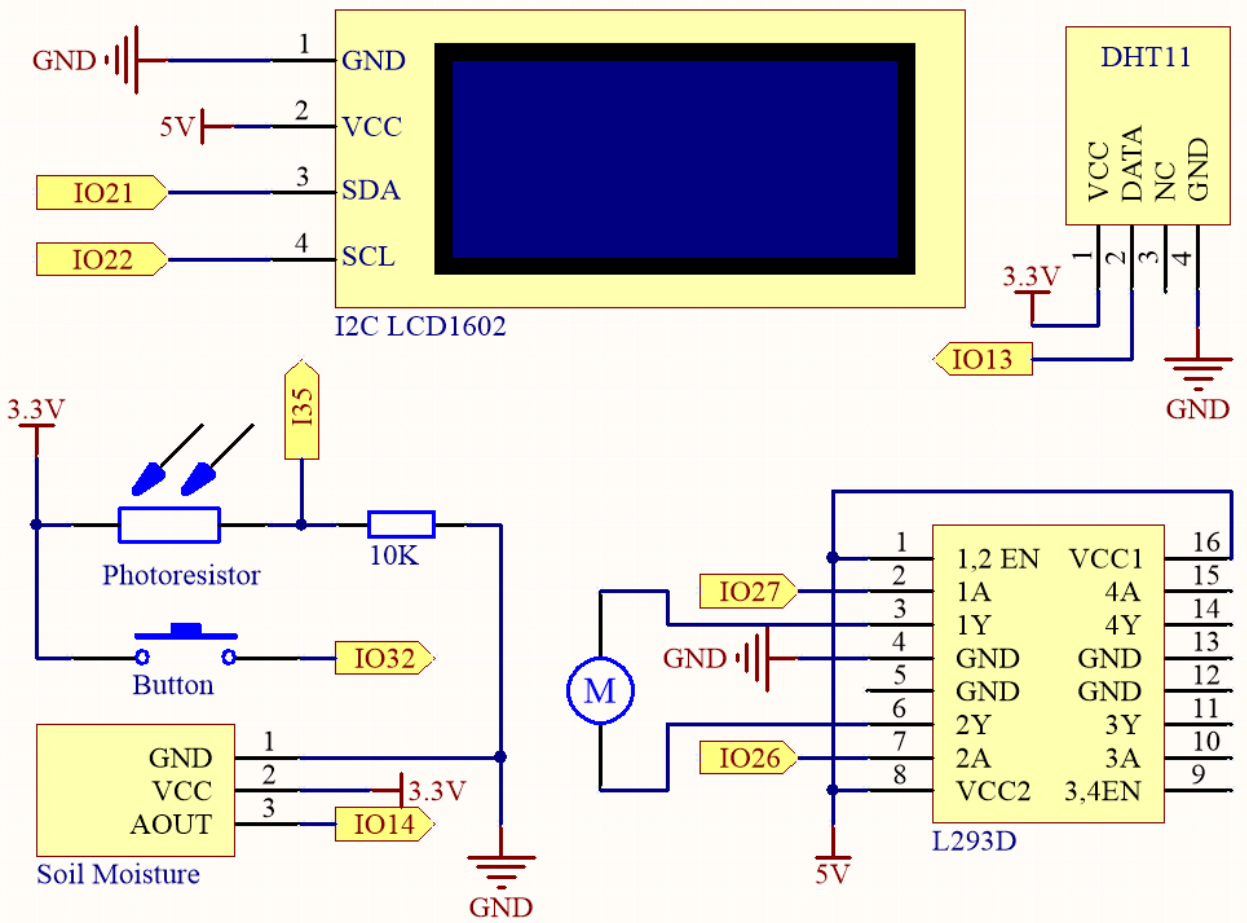
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können die Komponenten auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
DHT11 Feuchtigkeits- und Temperatursensor	
I2C LCD1602	
Zentrifugalpumpe	-
L293D	-
Taste	
Fotowiderstand	
Bodenfeuchtigkeitsmodul	

Schaltplan

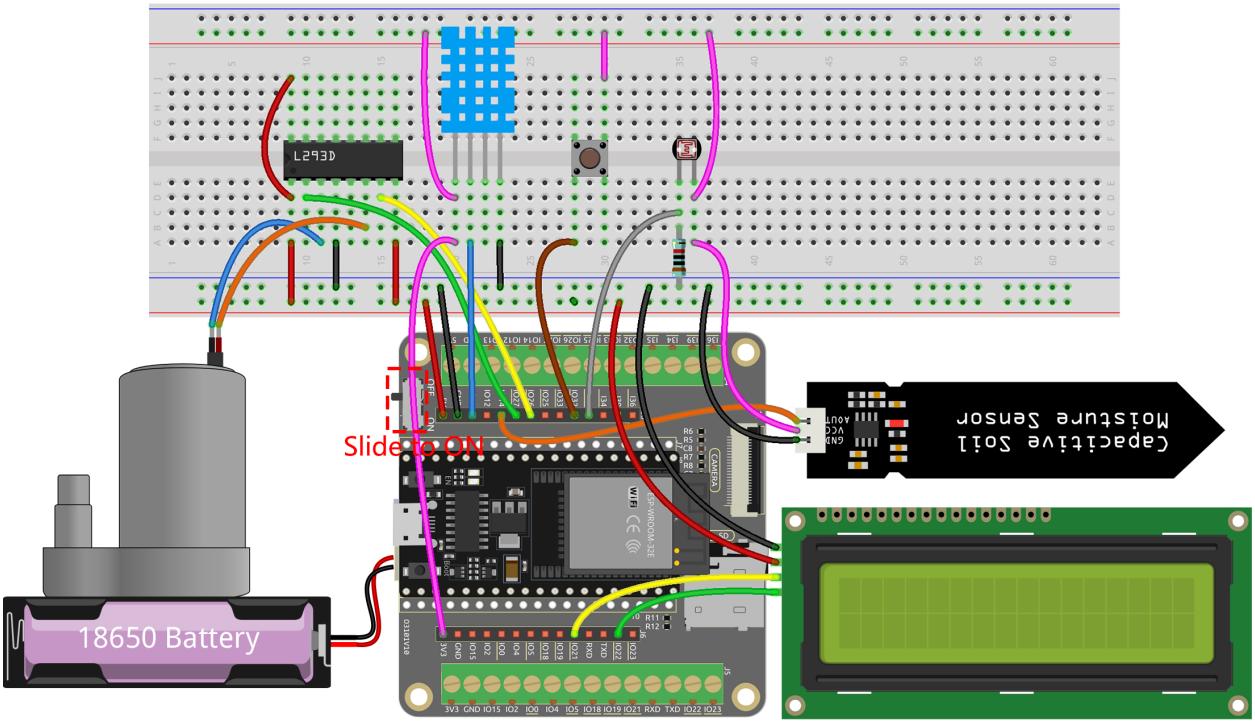


Das System verwendet einen DHT11-Sensor, um die Temperatur- und Luftfeuchtigkeitswerte der Umgebung zu messen. Gleichzeitig wird ein Bodenfeuchtigkeitsmodul verwendet, um den Feuchtigkeitsgehalt des Bodens zu messen, und ein Fotowiderstand dient zur Messung des Lichtniveaus. Die Messwerte dieser Sensoren werden auf einem LCD-Bildschirm angezeigt, und eine Wasserpumpe kann mit einem Knopf gesteuert werden, um die Pflanze bei Bedarf zu bewässern.

IO32 verfügt über einen internen Pull-Down-Widerstand von 1K und befindet sich standardmäßig auf einem niedrigen Logikniveau. Wenn der Knopf gedrückt wird, wird eine Verbindung zu VCC (Hohe Spannung) hergestellt, was zu einem hohen Logikniveau auf IO32 führt.

Verdrahtung

Bemerkung: Es wird hier empfohlen, zuerst die Batterie einzusetzen und dann den Schalter auf dem Erweiterungsboard auf die ON-Position zu schieben, um die Batterieversorgung zu aktivieren.



Code

Bemerkung:

- Öffnen Sie die Datei `6.8_plant_monitor.py`, die sich im Pfad `esp32-starter-kit-main\micropython\codes` befindet, oder kopieren Sie den Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie F5, um es auszuführen.
- Stellen Sie sicher, dass der Interpreter „MicroPython (ESP32).COMxx“ in der rechten unteren Ecke ausgewählt ist.

```
from machine import ADC, Pin
import time
import dht
from lcd1602 import LCD

# DHT11
dht11 = dht.DHT11(Pin(13))

# Soil moisture
moisture_pin = ADC(Pin(14))
moisture_pin.atten(ADC.ATTN_11DB)

# Photoresistor
photoresistor = ADC(Pin(35))
photoresistor.atten(ADC.ATTN_11DB)

# Button and pump
button = Pin(32, Pin.IN)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
motor1A = Pin(27, Pin.OUT)
motor2A = Pin(26, Pin.OUT)

# I2C LCD1602 setup
lcd = LCD()

# Rotate the pump
def rotate():
motor1A.value(1)
motor2A.value(0)

# Stop the pump
def stop():
motor1A.value(0)
motor2A.value(0)

button_state = False

# Define the button callback function to toggle the button state
def button_callback(pin):
global button_state
button_state = not button_state

# Attach the button callback function to the rising edge of the button pin
button.irq(trigger=Pin.IRQ_RISING, handler=button_callback)

page = 0
temp = 0
humi = 0

try:
while True:

      # If the button is pressed and button state is True
      if button_state:
            print("rotate")
            rotate()

      # If the button is pressed again and button state is False
      if not button_state:
            print("stop")
            stop()
      time.sleep(2)

      # Clear the LCD display
      lcd.clear()

      # Toggle the value of the page variable between 0 and 1
      page=(page+1)%2

      # When page is 1, display temperature and humidity on the LCD1602
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    if page is 1:
        try:
        # Measure temperature and humidity
        dht11.measure()

        # Get temperature and humidity values
        temp = dht11.temperature()
        humi = dht11.humidity()
        except Exception as e:
        print("Error: ", e)

        # Display temperature and humidity
        lcd.write(0, 0, "Temp: {}\xDFC".format(temp))
        lcd.write(0, 1, "Humi: {}%".format(humi))

    # If page is 0, display the soil moisture and light
    else:
        light = photoresistor.read()
        moisture = moisture_pin.read()

        # Clear the LCD display
        lcd.clear()

        # Display the value of soil moisture and light
        lcd.write(0, 0, f"Moisture: {moisture}")
        lcd.write(0, 1, f"Light: {light}")

except KeyboardInterrupt:
# Stop the motor when KeyboardInterrupt is caught
stop()
```

- Während der Code ausgeführt wird, zeigt das I2C LCD1602 abwechselnd Temperatur und Luftfeuchtigkeit sowie Analogwerte der Bodenfeuchtigkeit und Lichtintensität an, mit einem Intervall von 2 Sekunden.
- Drücken Sie den Knopf, um die Wasserpumpe zu starten, und drücken Sie ihn erneut, um die Wasserpumpe zu stoppen.

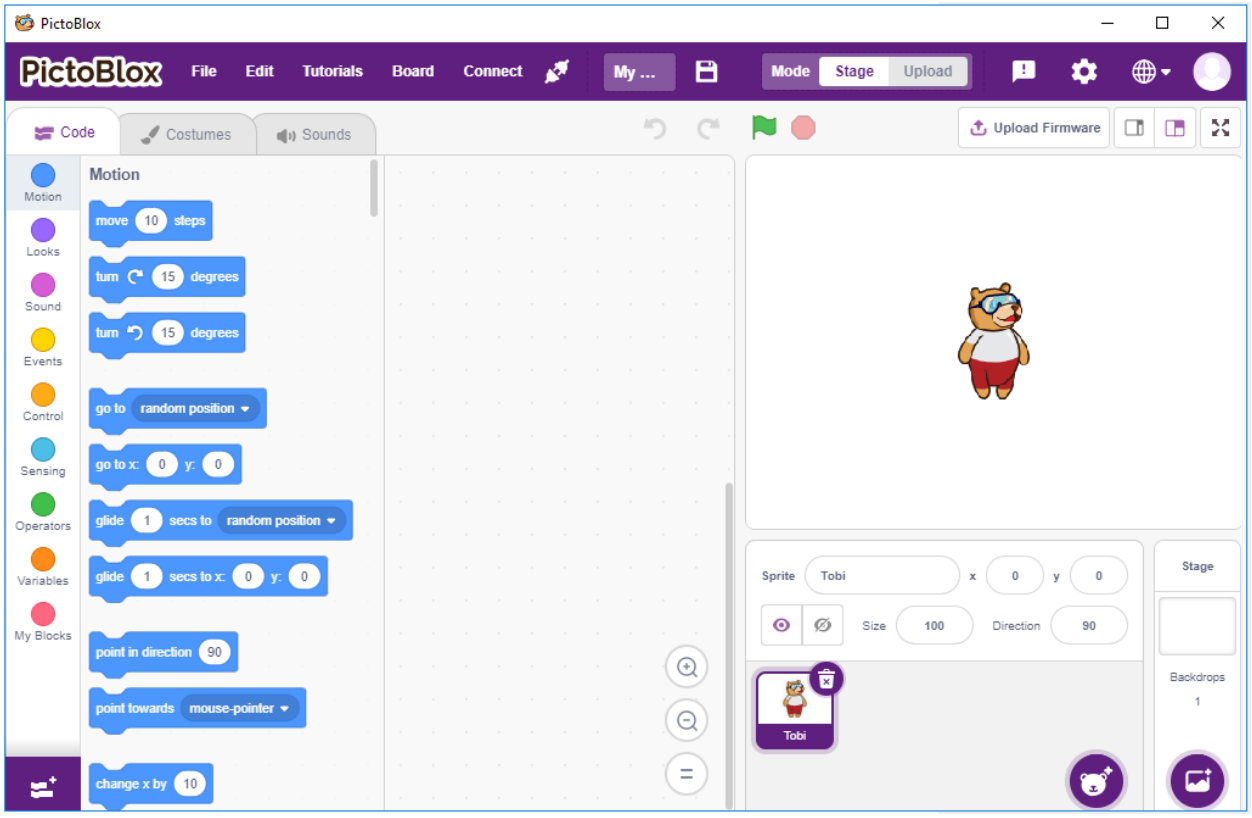
Bemerkung: Falls der Code und die Verkabelung korrekt sind, aber das LCD dennoch keinen Inhalt anzeigt, können Sie das Potentiometer auf der Rückseite justieren, um den Kontrast zu erhöhen.

KAPITEL 4

Play with Scratch

Neben der Programmierung in der Arduino IDE oder Thonny IDE können wir auch grafische Programmierung nutzen.

Hier empfehlen wir die Programmierung mit Scratch, aber das offizielle Scratch ist derzeit nur mit dem Raspberry Pi kompatibel. Deshalb haben wir uns mit einem Unternehmen, STEMPedia, zusammengetan, das eine auf Scratch 3 basierende grafische Programmiersoftware für viele Boards entwickelt hat - PictoBlox.



Es behält die grundlegenden Funktionen von Scratch 3 bei, fügt jedoch die Steuerung von Boards hinzu, wie Arduino

Uno, Mega, Nano, ESP32, Microbit und STEAMPedia-eigene Hauptplatinen, die externe Sensoren und Roboter nutzen können, um die Sprites auf der Bühne zu steuern, mit starken Hardware-Interaktionsfähigkeiten.

Darüber hinaus bietet es KI und maschinelles Lernen, sodass auch ohne umfangreiche Programmierkenntnisse diese beliebten und hochtechnologischen Werkzeuge erlernt und genutzt werden können.

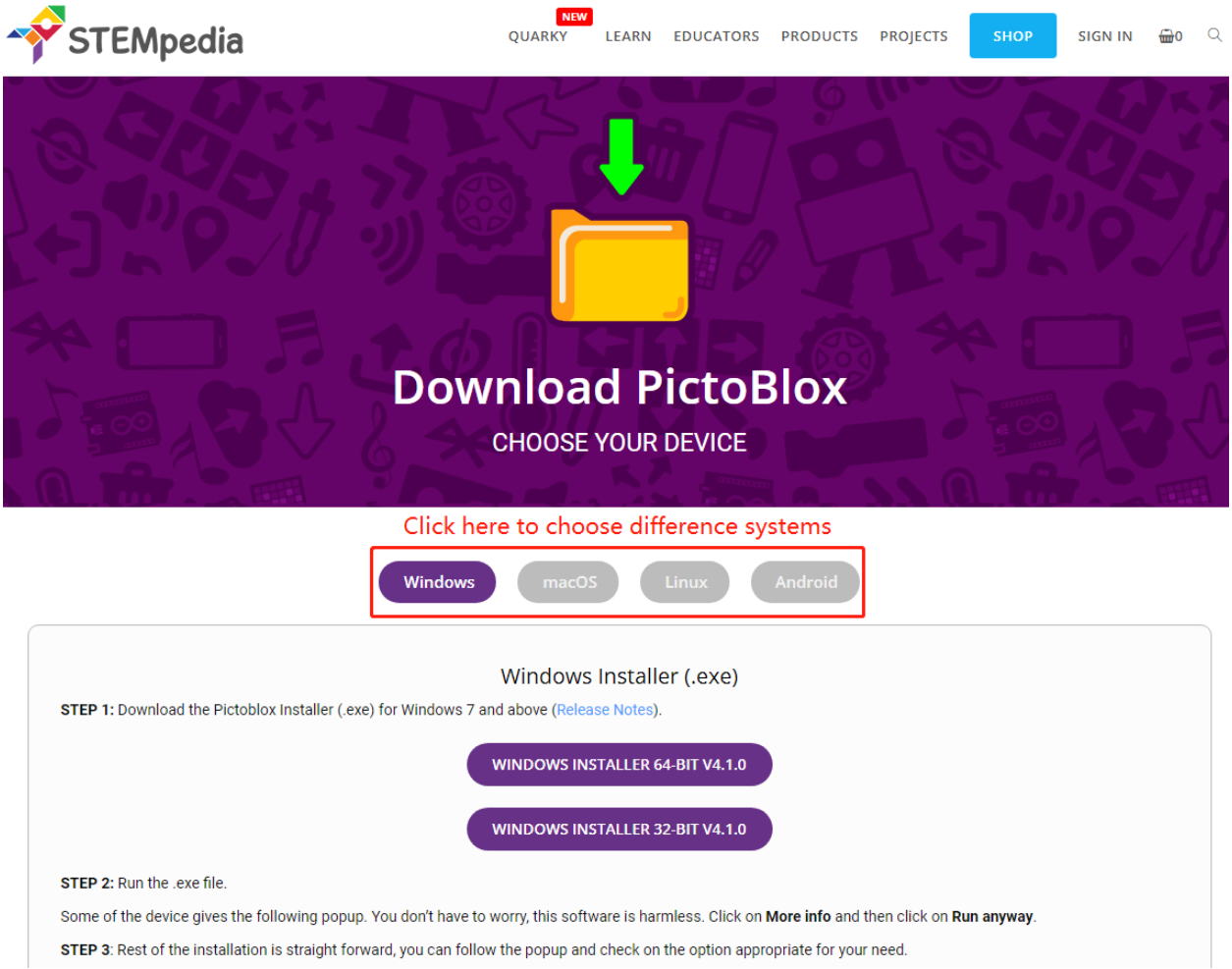
Einfach die Scratch-Programmierblöcke per Drag-and-Drop bewegen und coole Spiele, Animationen, interaktive Projekte und sogar Roboter nach Wunsch steuern!

Beginnen wir jetzt unsere Entdeckungsreise!

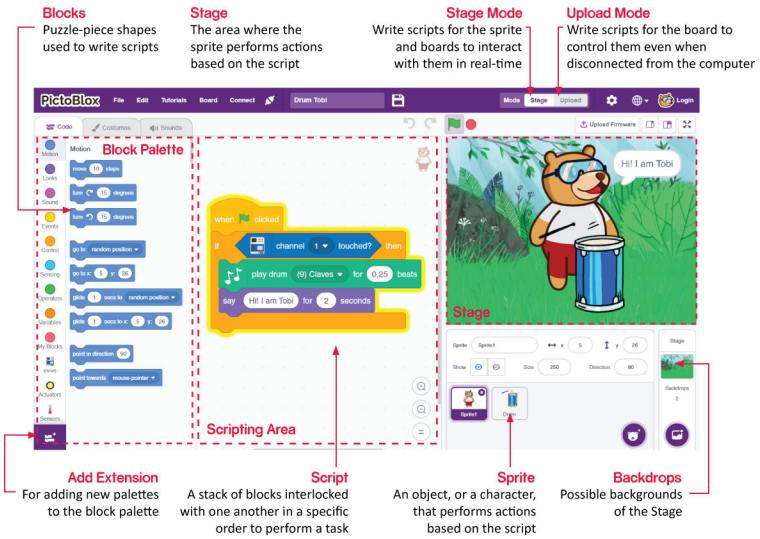
1. Erste Schritte

4.1 1.1 PictoBlox installieren

Klicken Sie auf diesen Link: https://thestempedia.com/product/pictoblox/download-pictoblox/, wählen Sie das passende Betriebssystem (Windows, macOS, Linux) aus und folgen Sie den Schritten zur Installation.



4.2 1.2 Schnittstellen-Einführung



Sprites

Ein Sprite ist ein Objekt oder Charakter, der verschiedene Aktionen in einem Projekt ausführt. Es versteht und befolgt die ihm gegebenen Befehle. Jedes Sprite hat spezifische Kostüme und Klänge, die Sie auch anpassen können.

Stage

Die Bühne ist der Bereich, in dem das Sprite Aktionen vor Kulissen gemäß Ihrem Programm ausführt.

Backdrops

Kulissen dienen dazu, die Bühne zu dekorieren. Sie können eine Kulisse aus PictoBlox auswählen, selbst eine zeichnen oder ein Bild von Ihrem Computer hochladen.

Script Area

Ein Skript ist ein Programm oder Code in PictoBlox/Scratch-Sprache. Es ist eine Reihe von „Blöcken“, die in einer bestimmten Reihenfolge angeordnet sind, um eine Aufgabe oder eine Reihe von Aufgaben auszuführen. Sie können mehrere Skripte schreiben, die alle gleichzeitig ausgeführt werden können. Skripte können Sie nur im Skriptbereich in der Mitte des Bildschirms schreiben.

Blocks

Blöcke sind wie Puzzleteile, die verwendet werden, um Programme zu schreiben, indem sie einfach im Skriptbereich gestapelt werden. Die Verwendung von Blöcken zum Schreiben von Code kann die Programmierung erleichtern und die Wahrscheinlichkeit von Fehlern verringern.

Block Palette

Die Blockpaletten befinden sich im linken Bereich und sind nach ihren Funktionen benannt, wie Bewegung, Klang und Steuerung. Jede Palette hat verschiedene Blöcke, zum Beispiel werden die Blöcke in der Bewegungspalette die Bewegung der Sprites steuern und die Blöcke in der Steuerungspalette werden die Arbeit des Skripts auf der Grundlage spezifischer Bedingungen steuern.

Es gibt andere Arten von Blockpaletten, die über den Button **Add Extension** unten links geladen werden können.

Modes

Im Gegensatz zu Scratch hat PictoBlox zwei Modi:

- *Bühnenmodus*: In diesem Modus können Sie Skripte für das Sprite und die Boards schreiben, um in Echtzeit mit Sprites zu interagieren. Wenn Sie das Board von PictoBlox trennen, ist keine Interaktion mehr möglich.
- *Upload-Modus*: Dieser Modus ermöglicht es Ihnen, Skripte zu schreiben und auf das Board hochzuladen, sodass Sie es auch verwenden können, wenn es nicht mit Ihrem Computer verbunden ist, zum Beispiel müssen Sie ein Skript für bewegende Roboter hochladen.

Für weitere Informationen besuchen Sie bitte: https://thestempedia.com/tutorials/getting-started-pictoblox

4.3 1.3 Schnellanleitung für PictoBlox

4.3.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

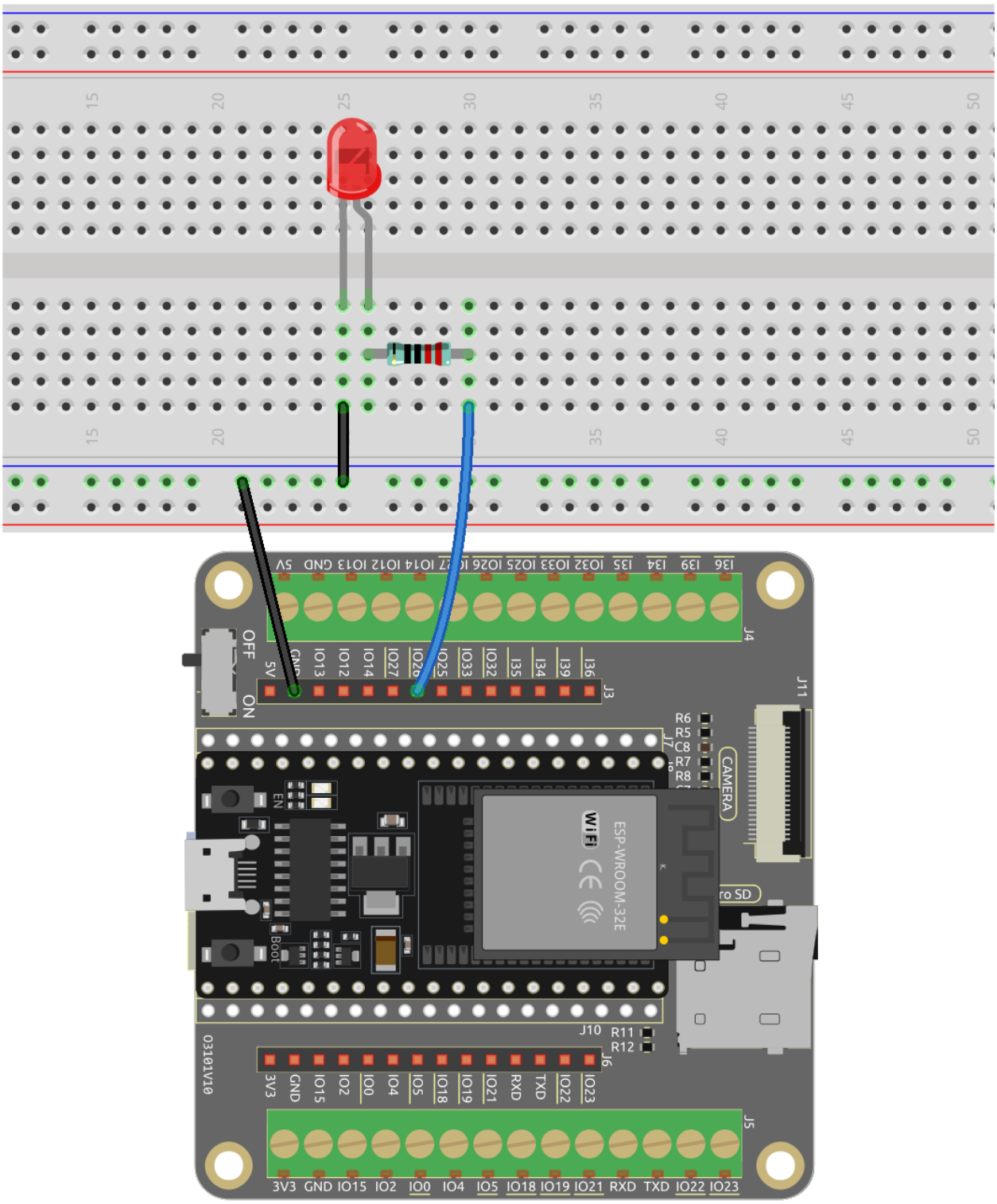
Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können die Komponenten auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	

Lernen wir nun, wie man PictoBlox in zwei Modi verwendet.

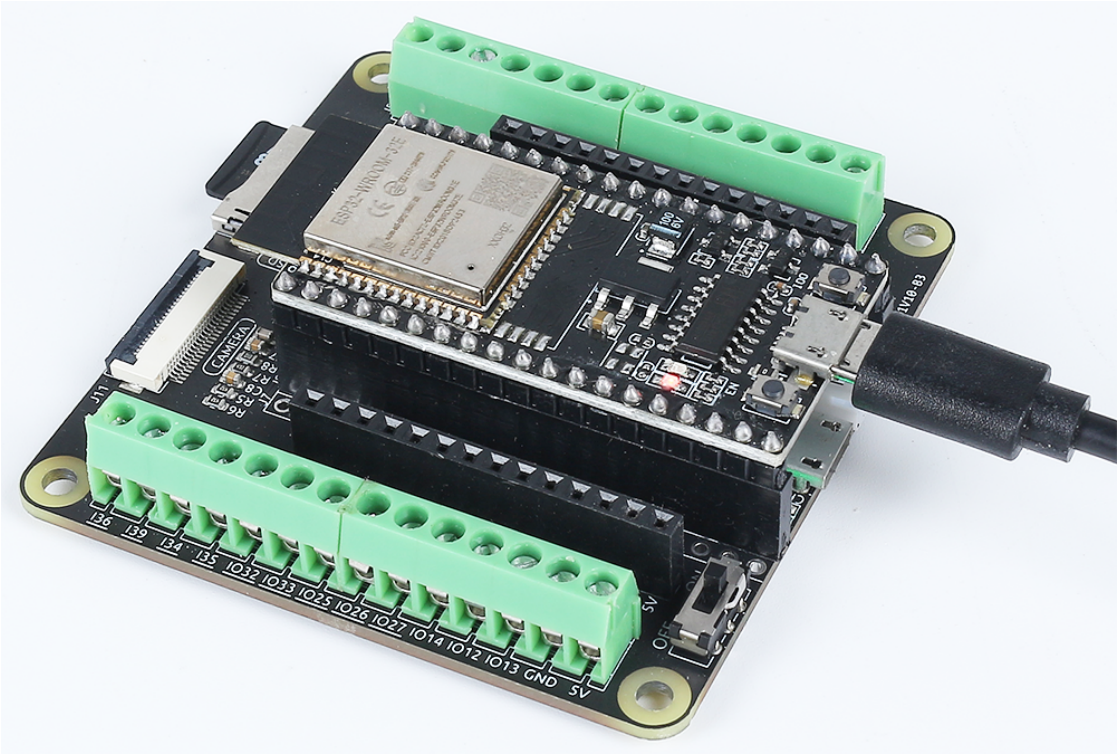
Außerdem bauen wir eine einfache Schaltung, um diese LED in 2 verschiedenen Modi blinken zu lassen.


OFF
ON
ESP-WROOM-32E
CAMERA
J3
J4
J5
J6
J10
J11
R11
R12
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

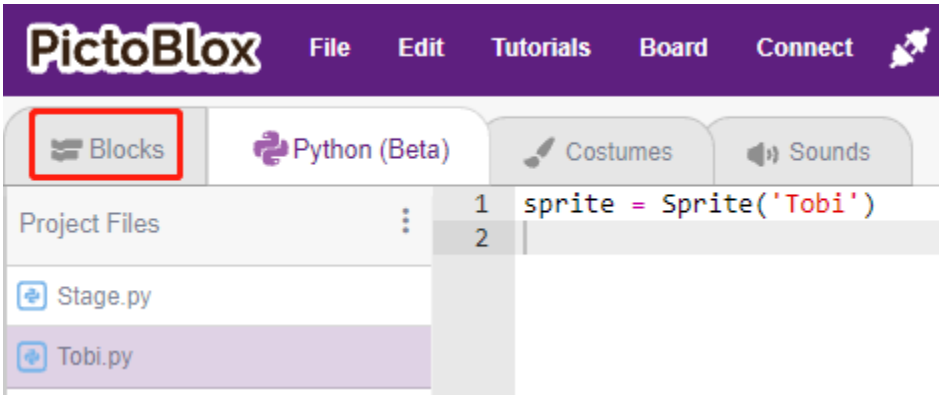
4.3.2 Bühnenmodus

1. Verbindung mit dem ESP32-Board herstellen

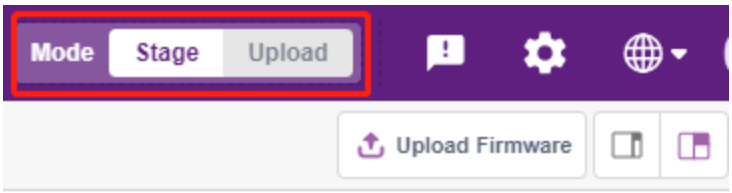
Verbinden Sie Ihr ESP32-Board mit einem USB-Kabel mit dem Computer, normalerweise wird der Computer Ihr Board automatisch erkennen und schließlich einen COM-Port zuweisen.



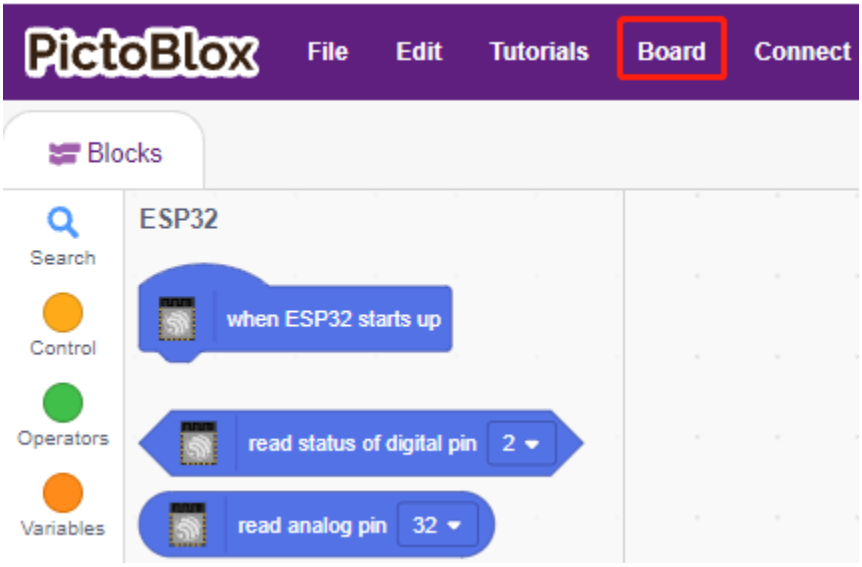
Öffnen Sie PictoBlox, standardmäßig öffnet sich die Python-Programmierschnittstelle. Wir müssen jedoch zur Block-Schnittstelle wechseln.



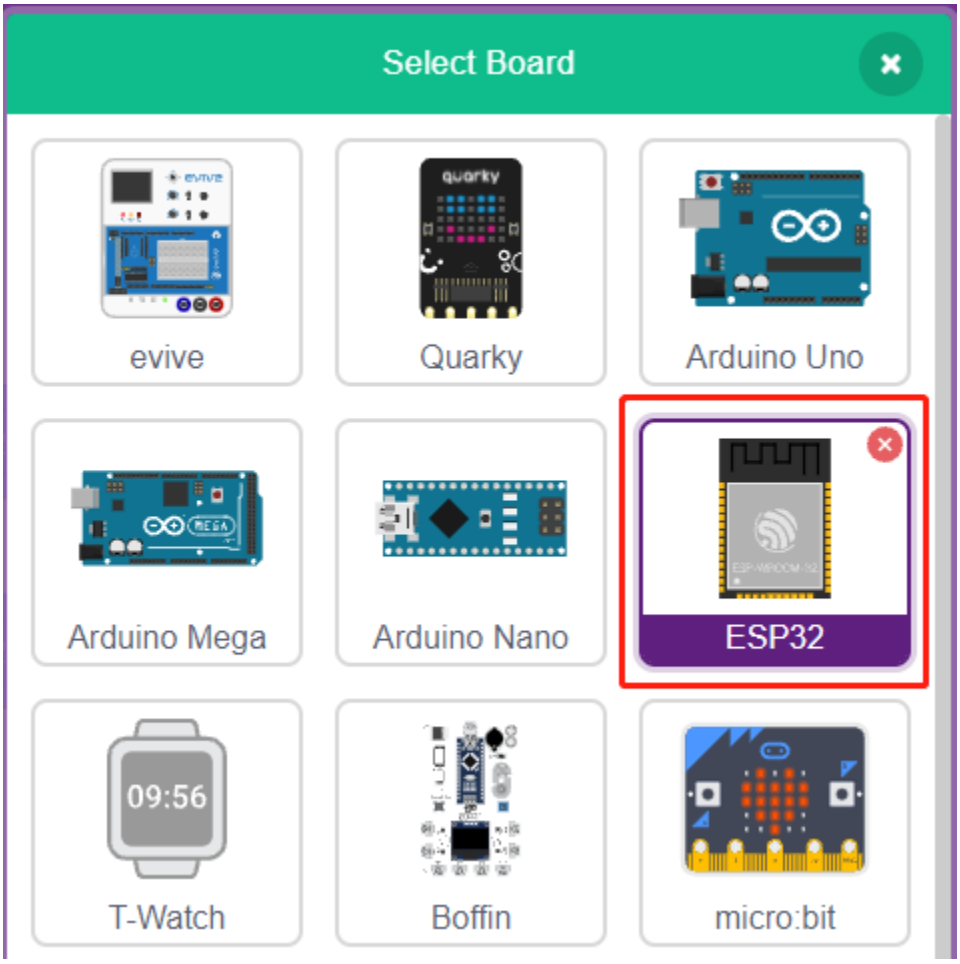
Dann sehen Sie in der oberen rechten Ecke den Moduswechsel. Standardmäßig ist der Bühnenmodus, wo Tobi auf der Bühne steht.



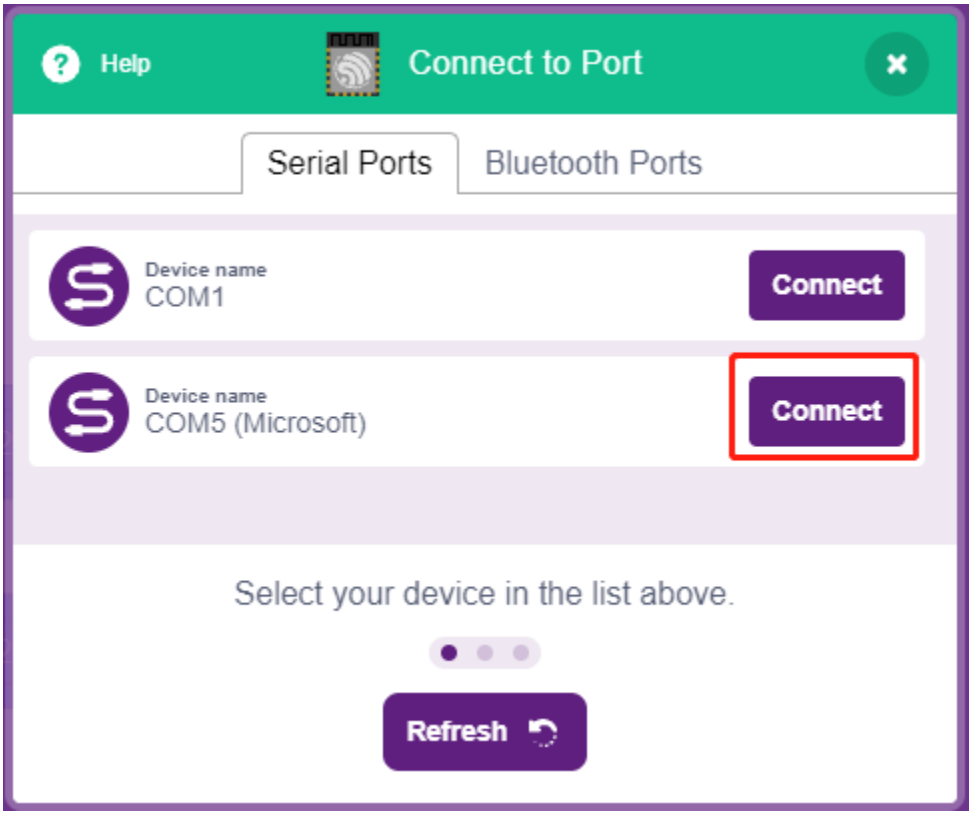
Klicken Sie in der oberen rechten Navigationsleiste auf **Board**, um das Board auszuwählen.



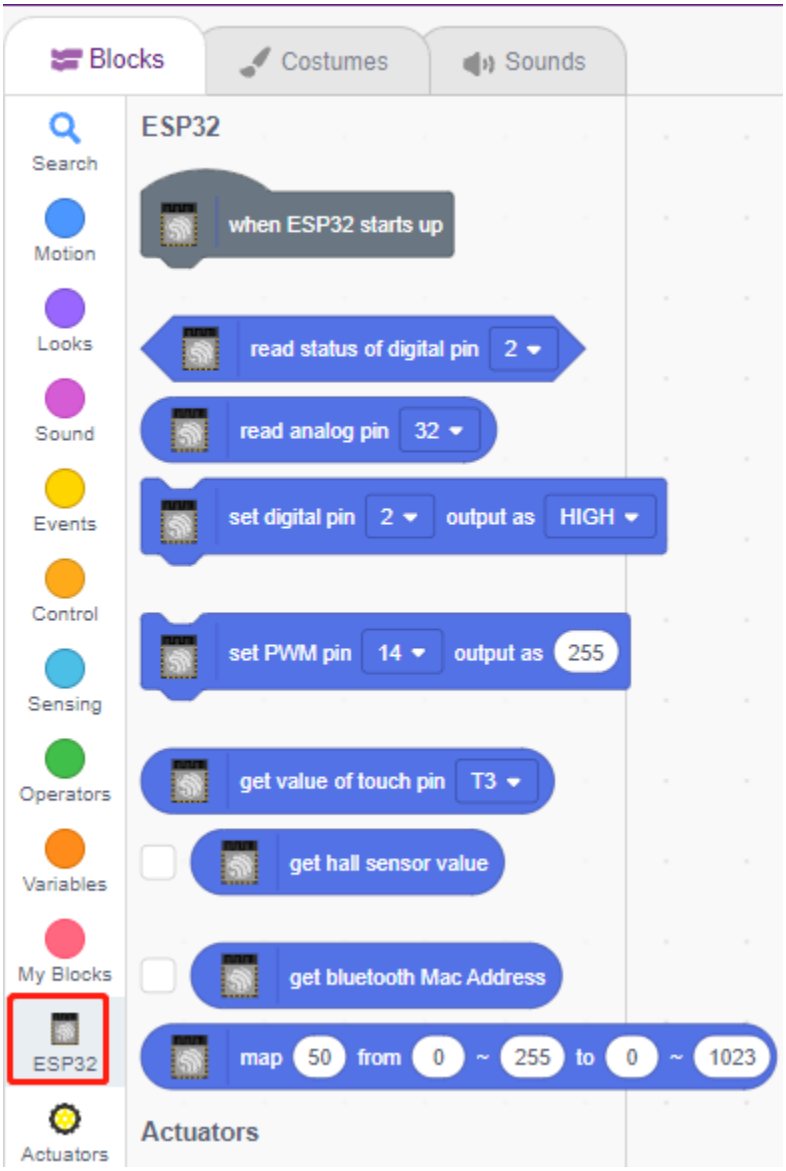
Wählen Sie zum Beispiel **ESP32**.



Ein Verbindungsfenster wird dann erscheinen, um den Port für die Verbindung auszuwählen, und zurück zur Startseite gehen, wenn die Verbindung abgeschlossen ist. Wenn Sie während der Nutzung die Verbindung trennen, können Sie auch auf **Connect** klicken, um erneut zu verbinden.



Gleichzeitig erscheinen ESP32-bezogene Paletten, wie ESP32, Aktuatoren usw., in der **Block Palette**.

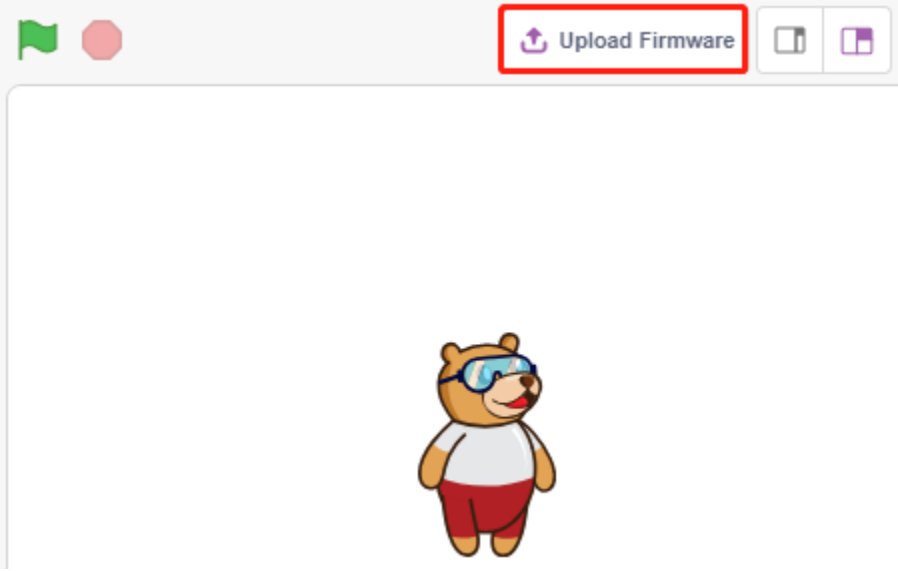


2. Firmware hochladen

Da wir im Bühnenmodus arbeiten werden, müssen wir die Firmware auf das Board hochladen. Dies stellt die Echtzeitkommunikation zwischen dem Board und dem Computer sicher. Das Hochladen der Firmware ist ein einmaliger Prozess. Klicken Sie dazu auf den Button Firmware hochladen.

Nach einer Weile erscheint die Erfolgsmeldung des Uploads.

Bemerkung: Wenn Sie dieses Board zum ersten Mal in PictoBlox verwenden oder wenn dieses Board zuvor mit der Arduino IDE hochgeladen wurde, müssen Sie **Upload Firmware** anklicken, bevor Sie es verwenden können.

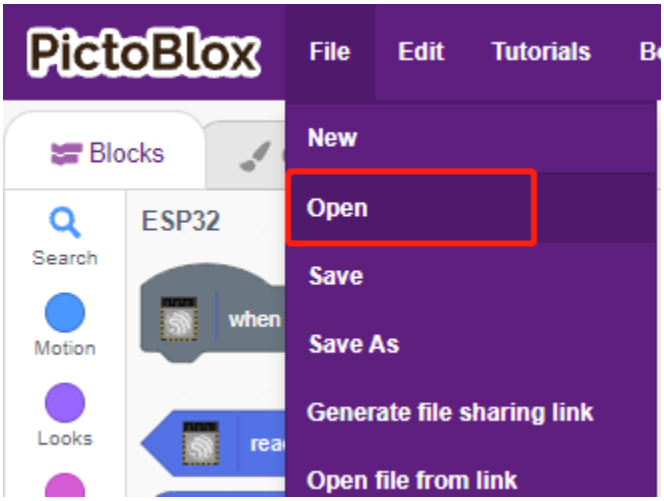


3. Programmierung

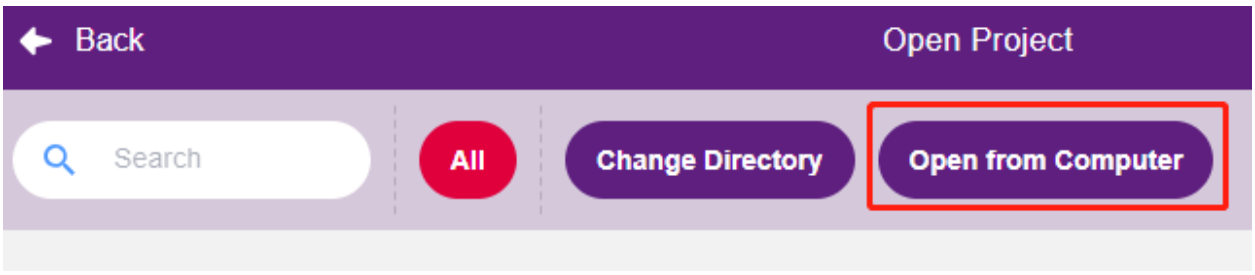
- Skript direkt öffnen und ausführen

Natürlich können Sie die Skripte direkt öffnen und ausführen, laden Sie sie aber bitte zuerst von GitHub herunter.

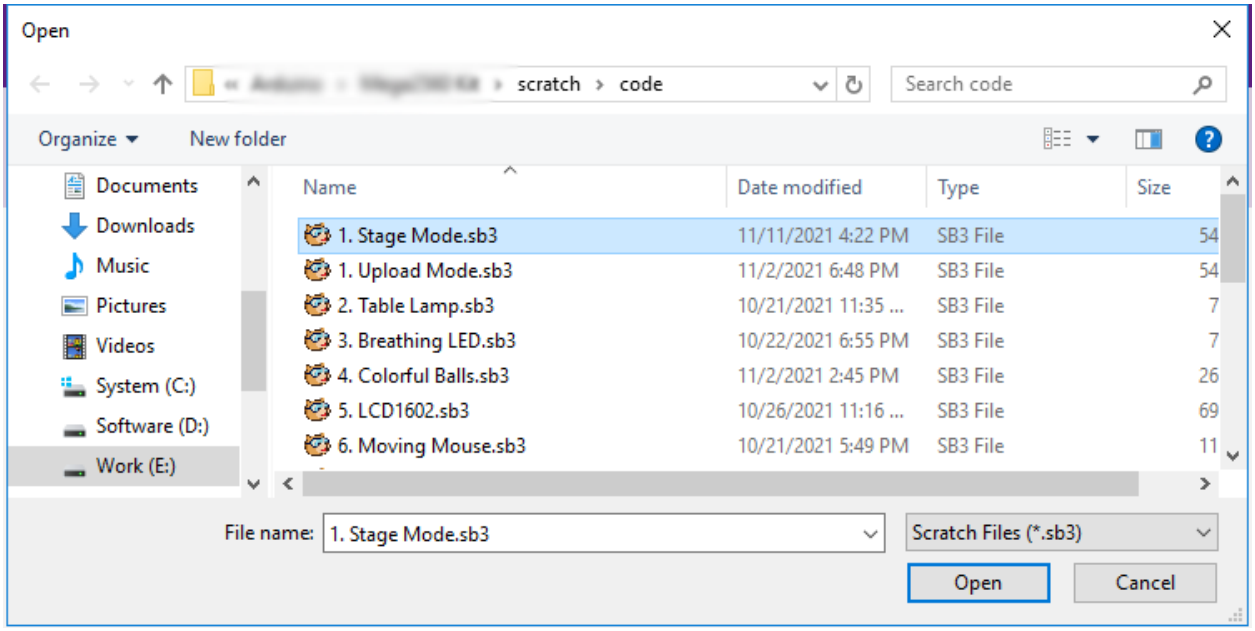
Klicken Sie oben rechts auf **File** und dann auf **Open**.



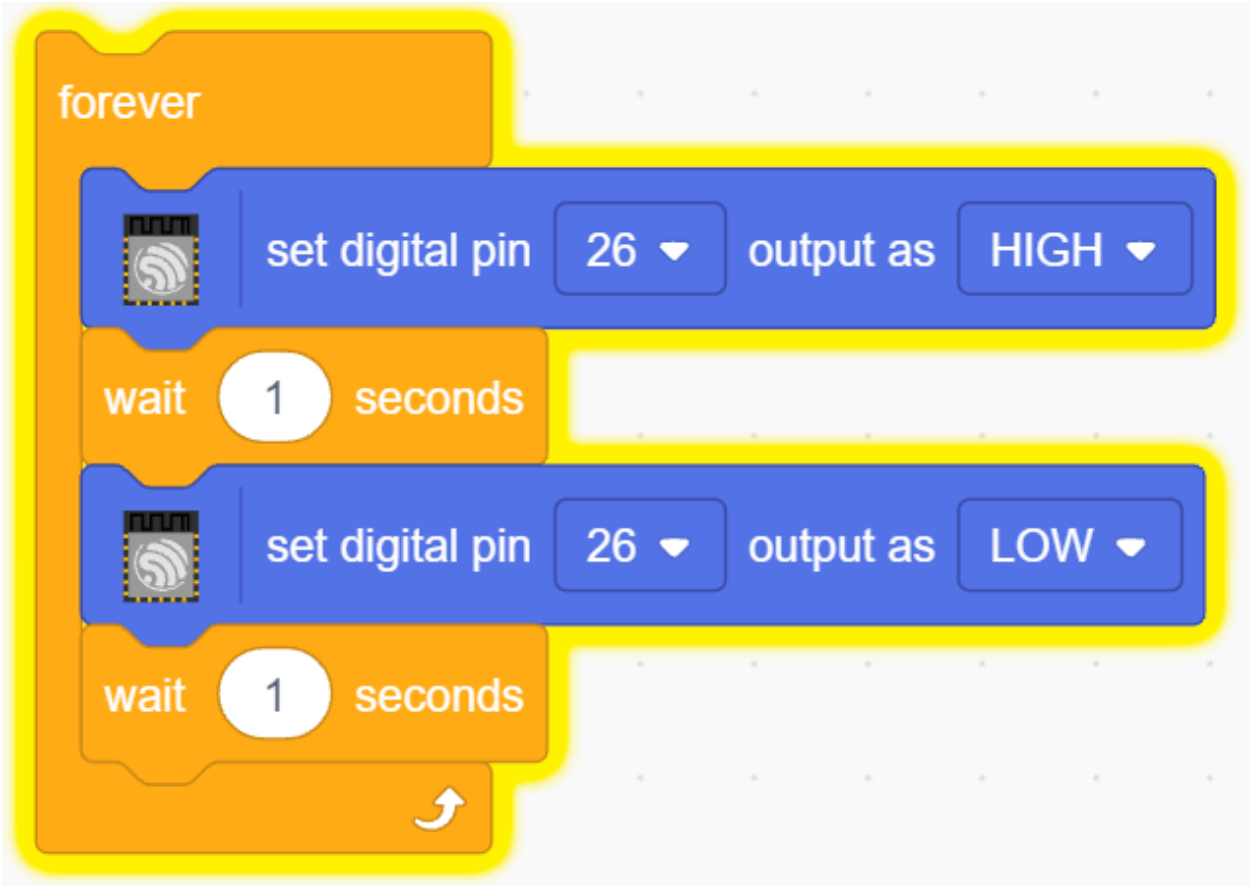
Wählen Sie **Open from Computer**.



Gehen Sie dann zum Pfad `esp32-starter-kit-main\scratch` und öffnen Sie **1. Stage Mode.sb3**. Stellen Sie sicher, dass Sie den benötigten Code von GitHub heruntergeladen haben.



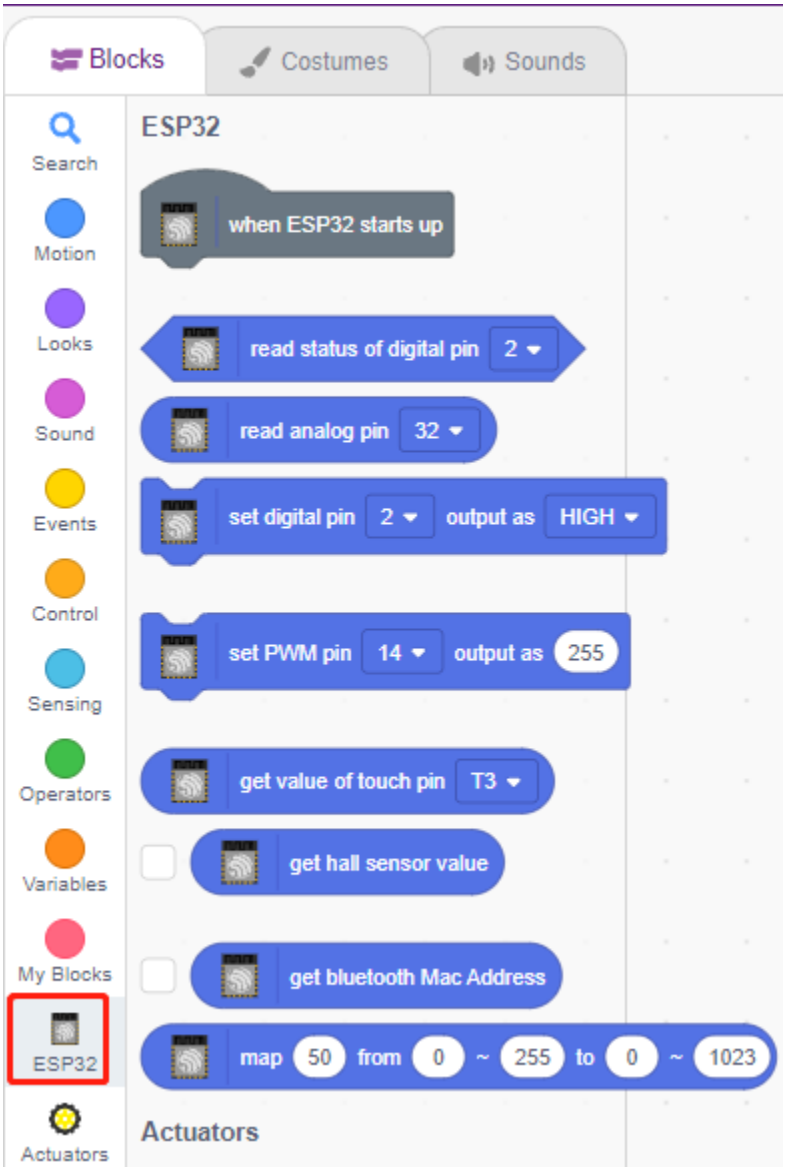
Klicken Sie direkt auf das Skript, um es auszuführen, bei einigen Projekten klicken Sie auf die grüne Fahne oder auf das Sprite.



- Schritt-für-Schritt programmieren

Sie können das Skript auch Schritt für Schritt nach diesen Schritten schreiben.

Klicken Sie auf die **ESP32**-Palette.



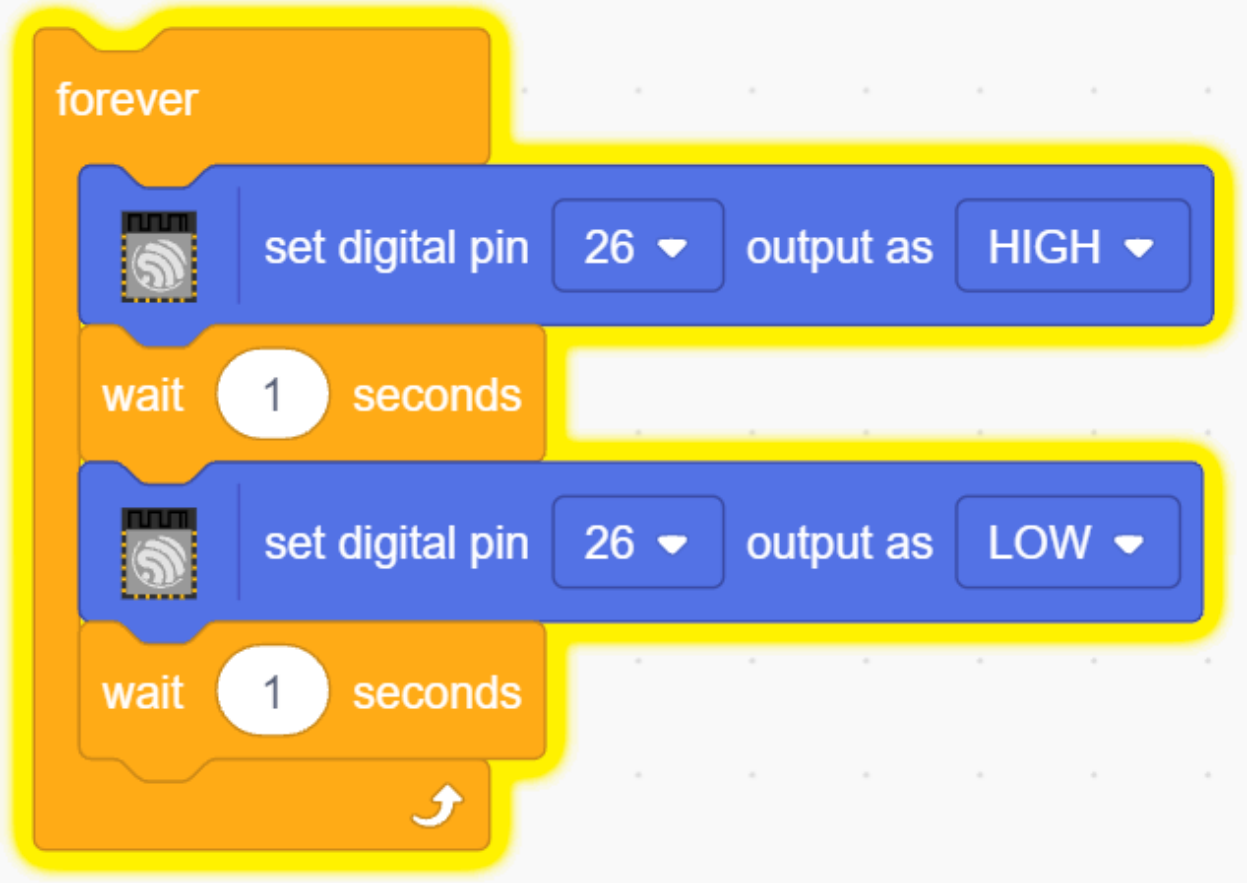
Die LED wird durch den digitalen Pin 26 gesteuert (nur 2 Zustände, HIGH oder LOW), ziehen Sie also den Block [set digital pin out as] in den Skriptbereich.

Da der Standardzustand der LED beleuchtet ist, stellen Sie nun Pin 23 auf LOW und klicken Sie auf diesen Block und Sie werden sehen, wie die LED ausgeht.

- [set digital pin out as]: Stellen Sie den digitalen Pin auf (HIGH/LOW) ein.

Um den Effekt einer kontinuierlich blinkenden LED zu sehen, benötigen Sie die Blöcke [Wait 1 seconds] und [forever] aus der **Control**-Palette. Klicken Sie nach dem Schreiben auf diese Blöcke, ein gelber Halo bedeutet, dass es läuft.

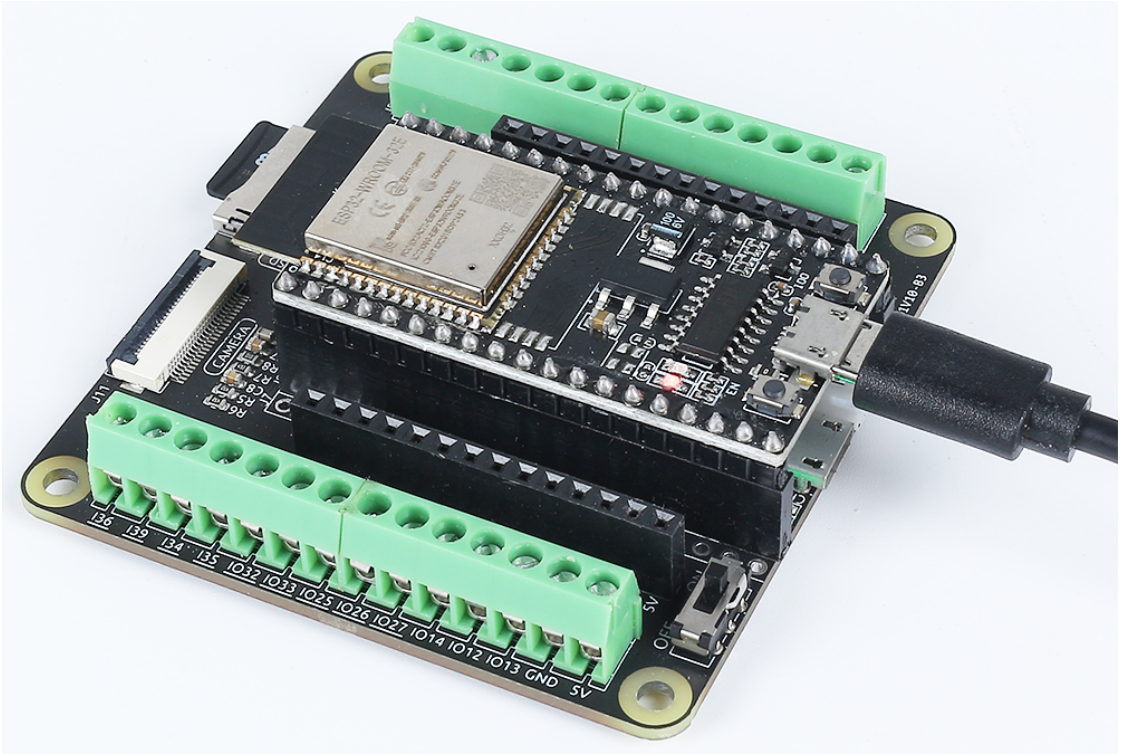
- [Wait 1 seconds]: aus der **Control**-Palette, verwendet, um das Zeitintervall zwischen 2 Blöcken einzustellen.
- [forever]: aus der **Control**-Palette, ermöglicht es dem Skript, weiterzulaufen, es sei denn, es wird manuell angehalten.



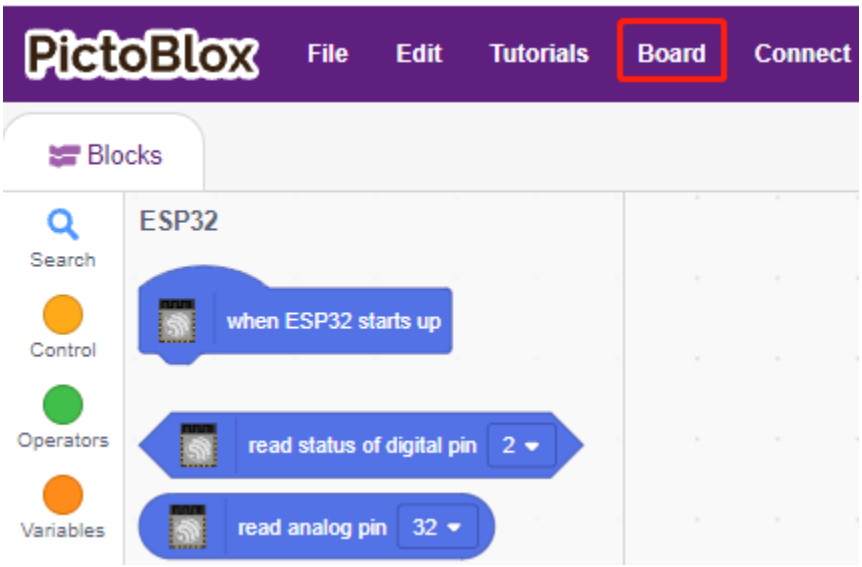
4.3.3 Upload-Modus

1. Verbindung mit dem ESP32-Board herstellen

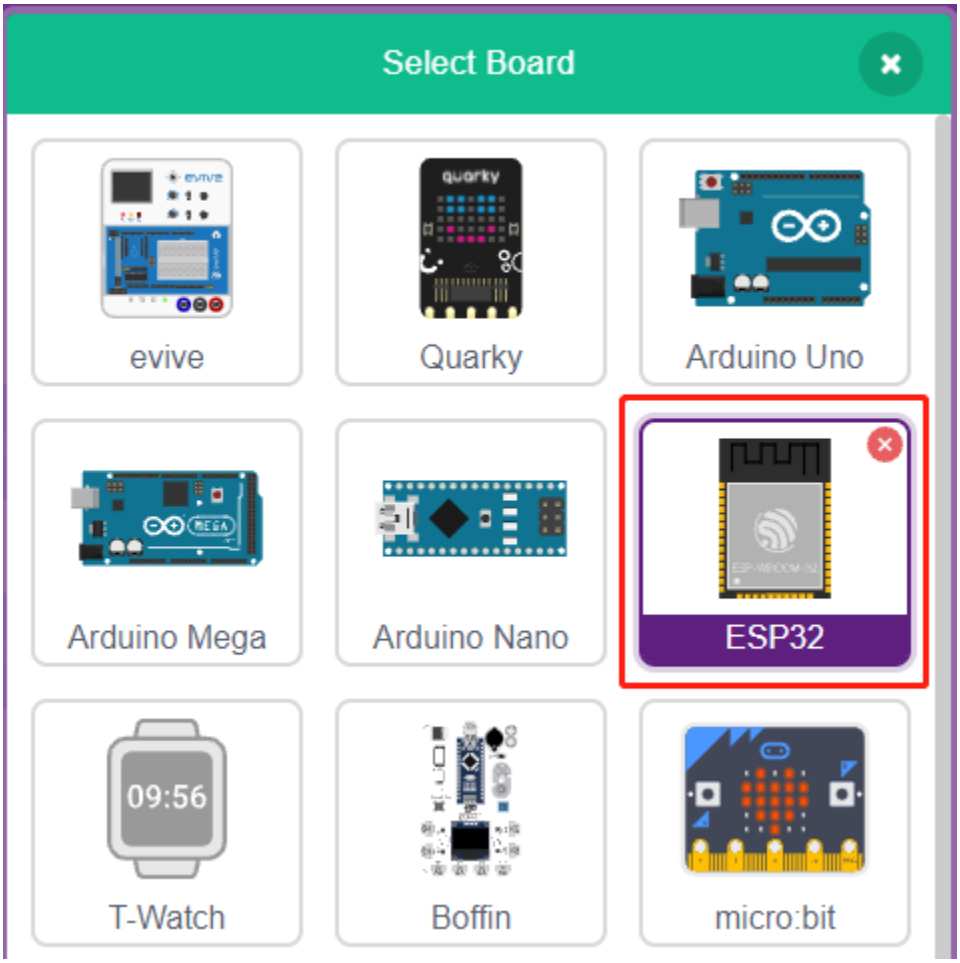
Verbinden Sie Ihr ESP32-Board mit einem USB-Kabel mit dem Computer, normalerweise wird der Computer Ihr Board automatisch erkennen und schließlich einen COM-Port zuweisen.



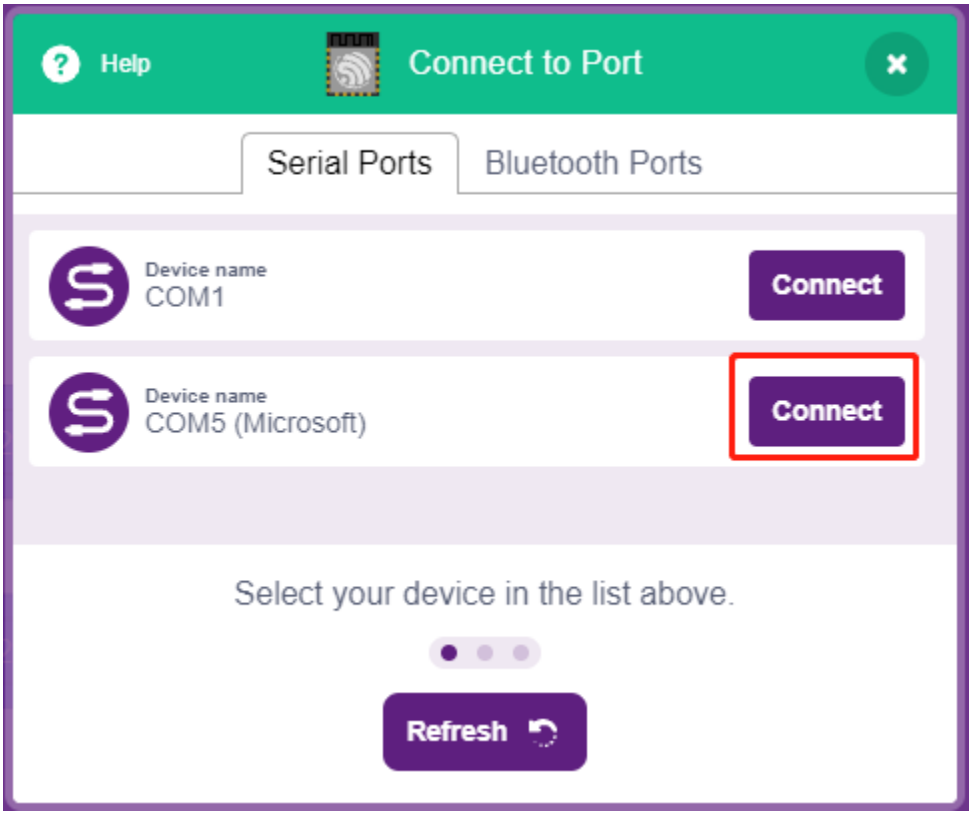
Öffnen Sie PictoBlox und klicken Sie in der oberen rechten Navigationsleiste auf **Board**, um das Board auszuwählen.



Wählen Sie zum Beispiel **ESP32**.

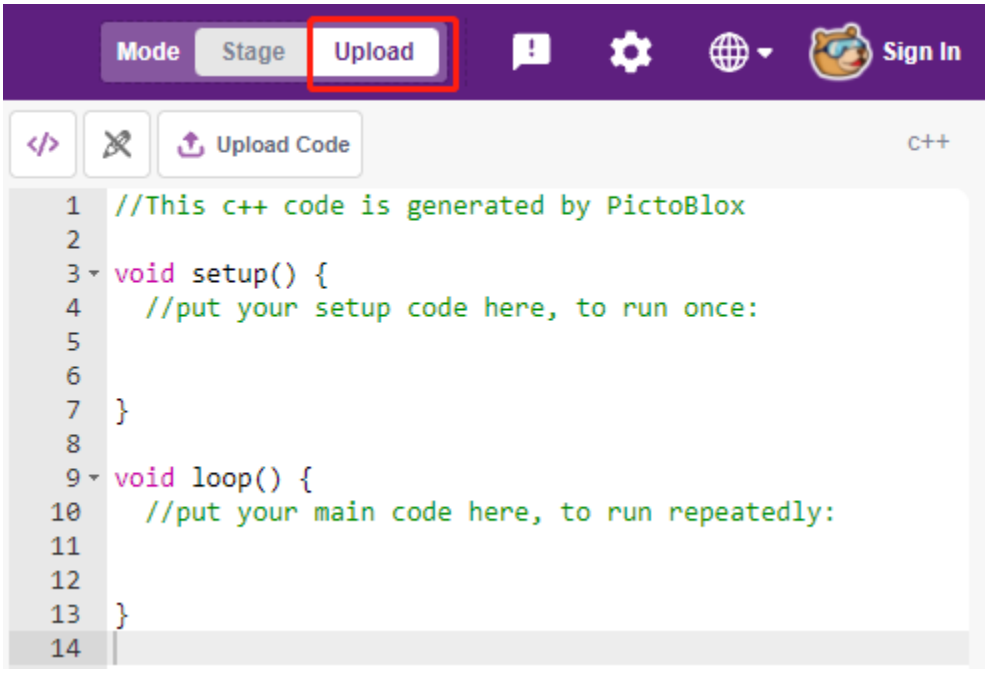


Ein Verbindungsfenster wird dann erscheinen, um den Port für die Verbindung auszuwählen, und zurück zur Startseite gehen, wenn die Verbindung abgeschlossen ist. Wenn Sie während der Nutzung die Verbindung trennen, können Sie auch auf **Connect** klicken, um erneut zu verbinden.



Gleichzeitig erscheinen ESP32-bezogene Paletten, wie ESP32, Aktuatoren usw., in der **Block Palette**.

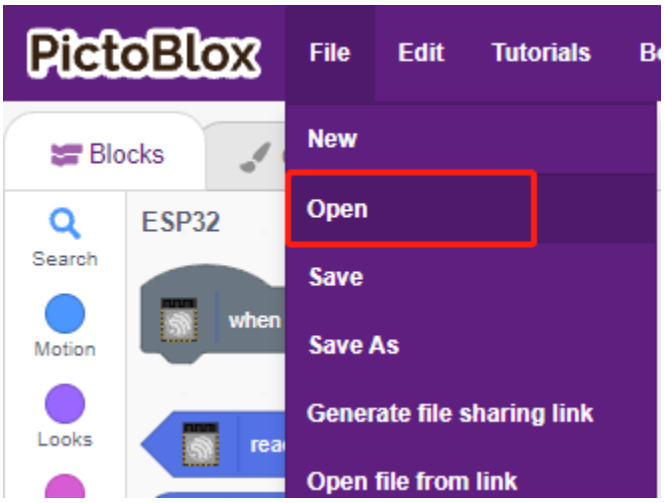
Nachdem Sie den Upload-Modus ausgewählt haben, wechselt die Bühne zum ursprünglichen Codebereich.



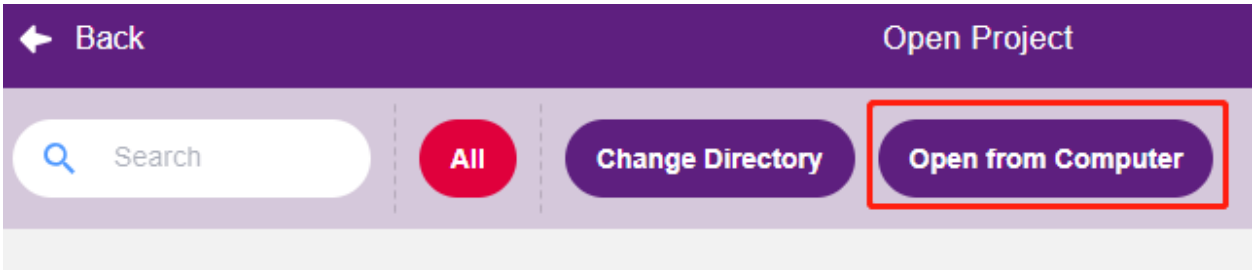
2. Programmierung

- Skript direkt öffnen und ausführen

Klicken Sie oben rechts auf **File**.

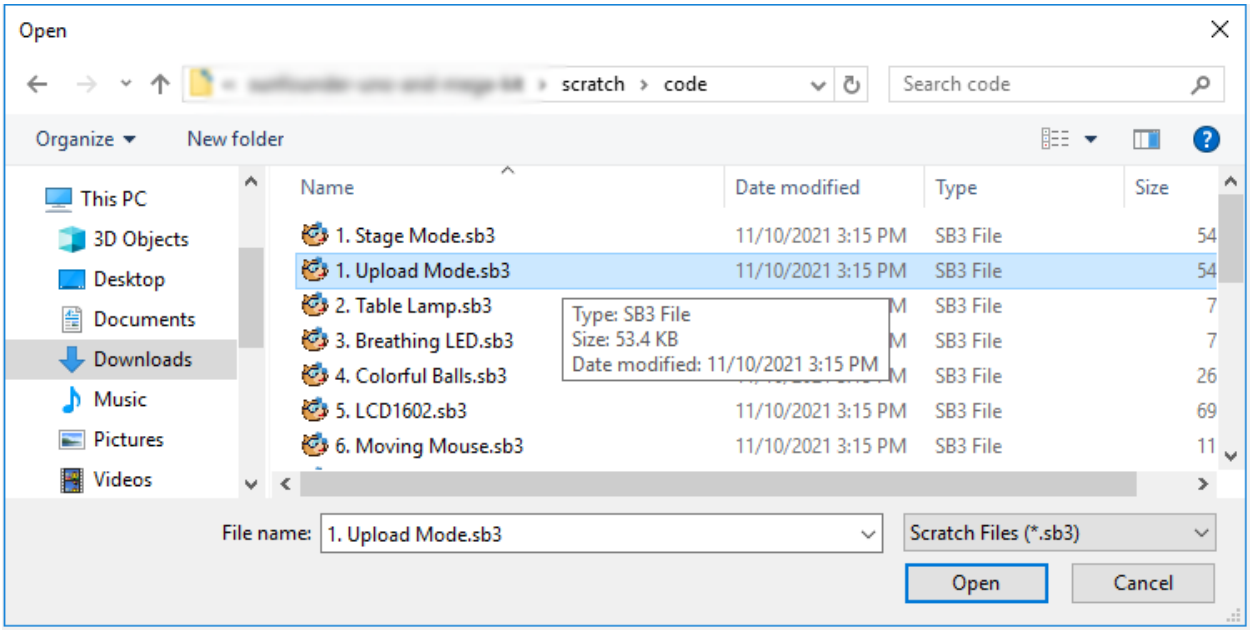


Wählen Sie **Open from Computer**.

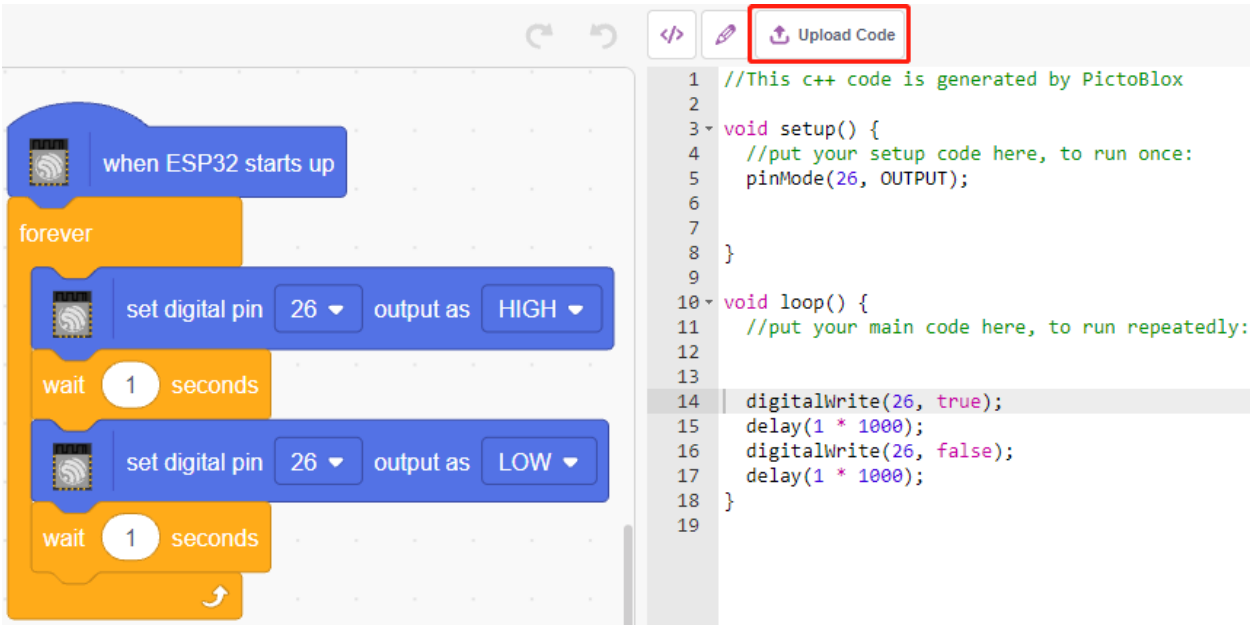


Gehen Sie dann zum Pfad `esp32-starter-kit-main\scratch` und öffnen Sie **1. Upload Mode.sb3**. Stellen Sie

sicher, dass Sie den benötigten Code von GitHub heruntergeladen haben.



Klicken Sie schließlich auf den Button **Upload Code**.

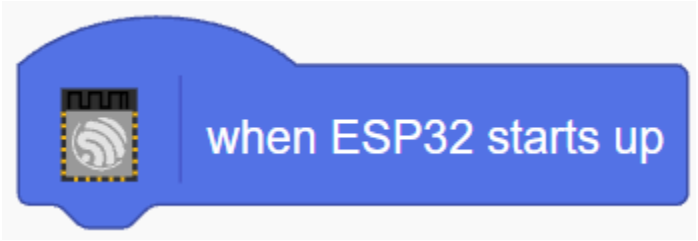


- Schritt-für-Schritt programmieren

Sie können das Skript auch Schritt für Schritt nach diesen Schritten schreiben.

Klicken Sie auf die **ESP32**-Palette.

Ziehen Sie [when ESP32 starts up] in den Skriptbereich, was für jedes Skript erforderlich ist.

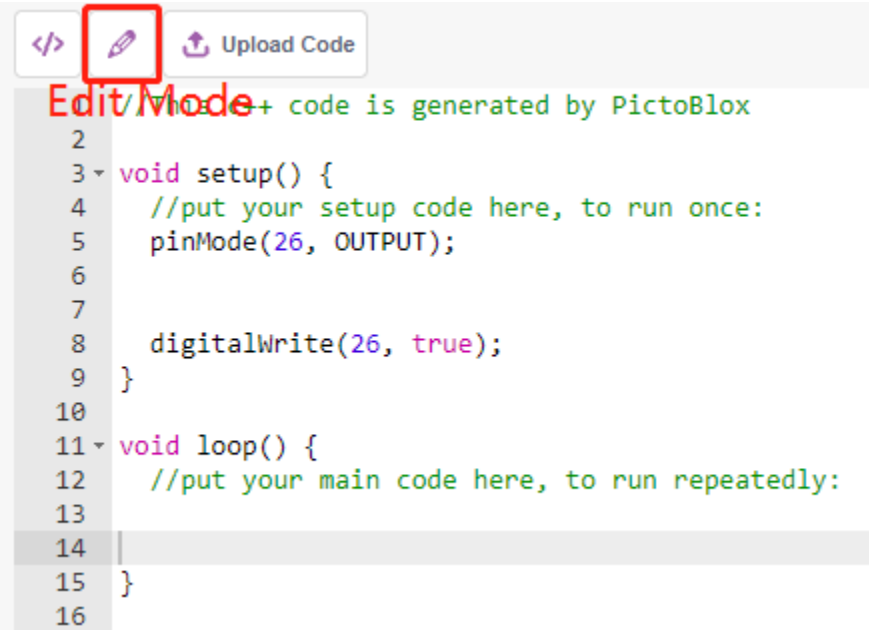


Die LED wird durch den digitalen Pin26 gesteuert (nur 2 Zustände HIGH oder LOW), ziehen Sie also den Block [set digital pin out as] in den Skriptbereich.

Da der Standardzustand der LED beleuchtet ist, stellen Sie nun Pin26 auf LOW und klicken Sie auf diesen Block und Sie werden sehen, wie die LED ausgeht.

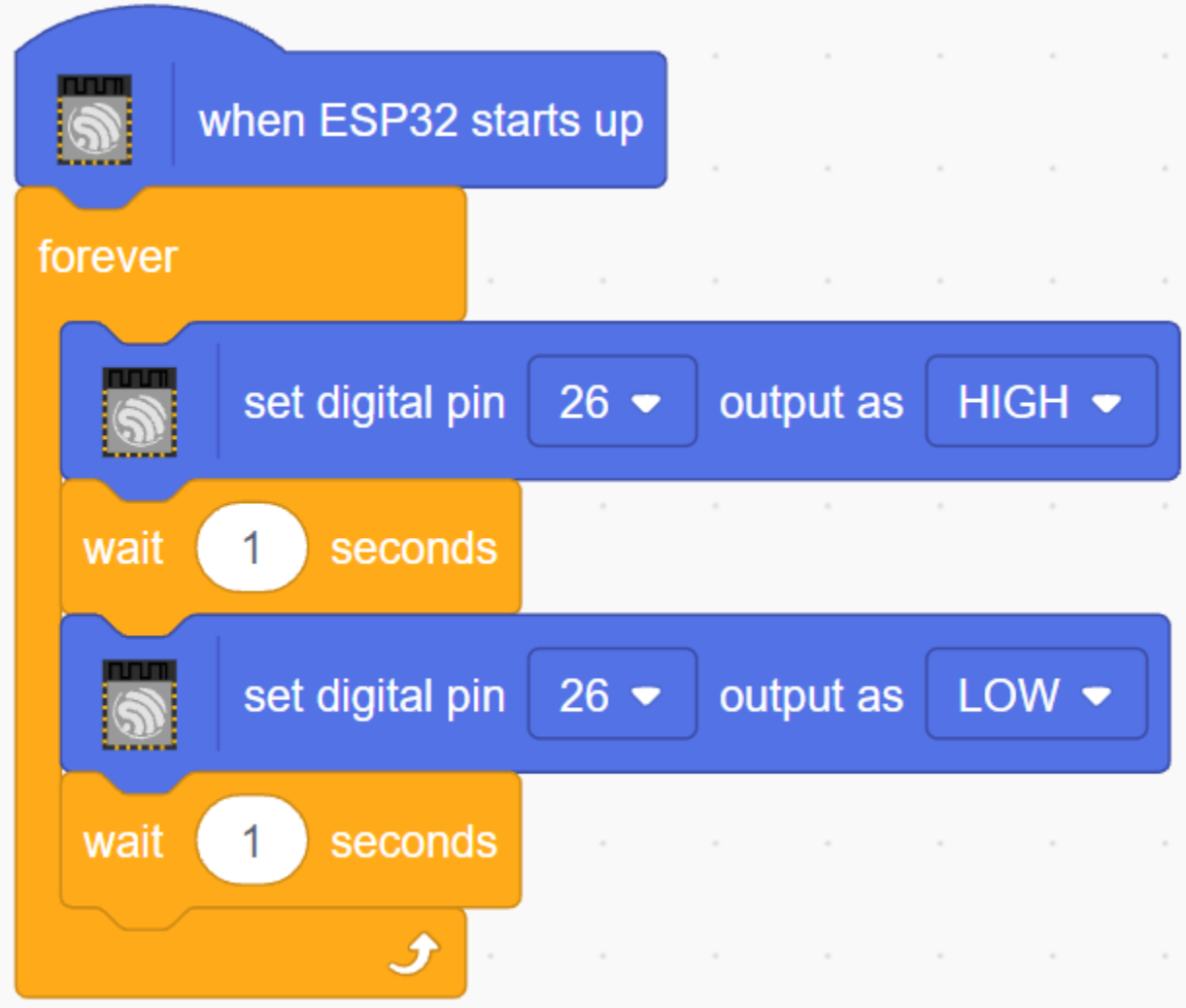
- [set digital pin out as]: Stellen Sie den digitalen Pin auf (HIGH/LOW) ein.

An dieser Stelle sehen Sie den Code auf der rechten Seite, wenn Sie diesen Code bearbeiten möchten, können Sie den Bearbeitungsmodus einschalten.

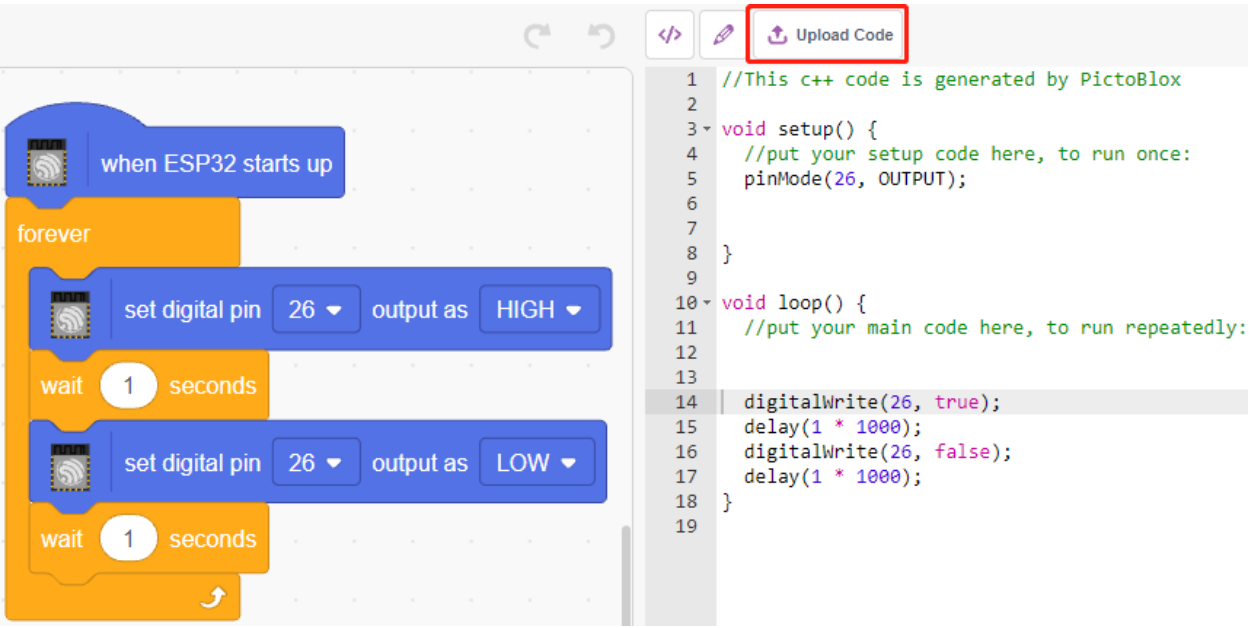


Um den Effekt einer kontinuierlich blinkenden LED zu sehen, benötigen Sie die Blöcke [Wait 1 seconds] und [forever] aus der **Control**-Palette. Klicken Sie nach dem Schreiben auf diese Blöcke, ein gelber Halo bedeutet, dass es läuft.

- [Wait 1 seconds]: aus der **Control**-Palette, verwendet, um das Zeitintervall zwischen 2 Blöcken einzustellen.
- [forever]: aus der **Control**-Palette, ermöglicht es dem Skript, weiterzulaufen, es sei denn, die Stromversorgung wird unterbrochen.



Klicken Sie schließlich auf den Button **Upload Code**.



2. Projekte

Die folgenden Projekte sind nach Programmierschwierigkeit geordnet. Es wird empfohlen, diese Bücher der Reihe nach zu lesen.

In jedem Projekt gibt es sehr detaillierte Schritte, die Ihnen beibringen, wie Sie die Schaltung aufbauen und Schritt für Schritt programmieren, um das endgültige Ergebnis zu erreichen.

Natürlich können Sie auch direkt das Skript öffnen und ausführen, aber Sie müssen sicherstellen, dass Sie das relevante Material von GitHub heruntergeladen haben.

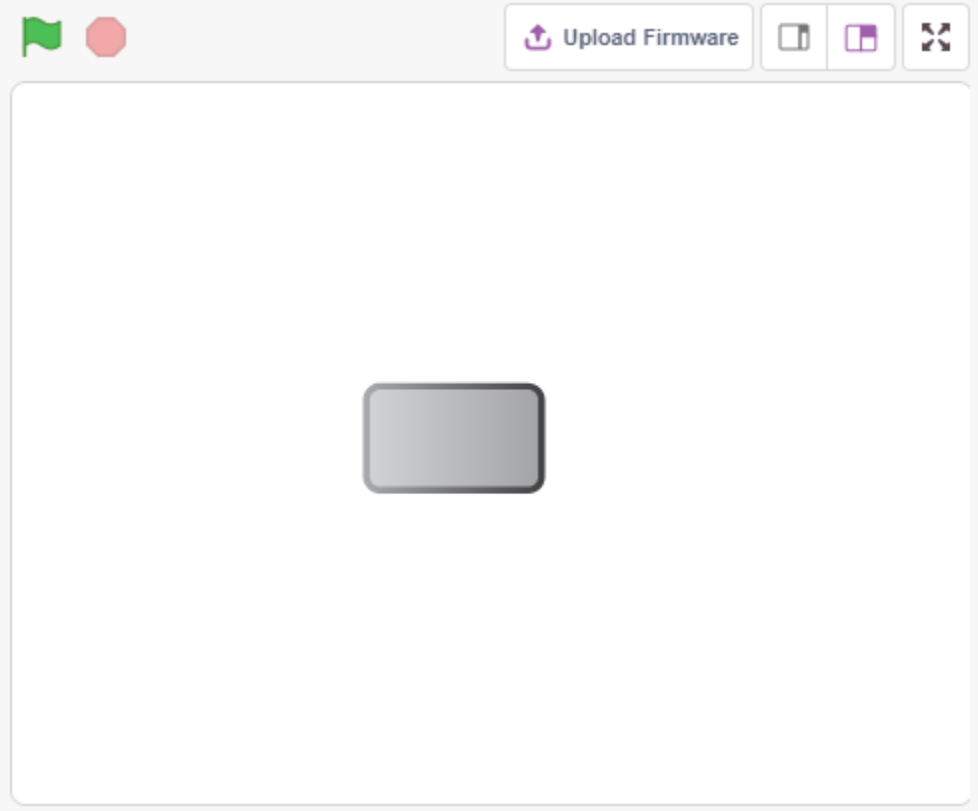
Nachdem der Download abgeschlossen ist, entpacken Sie ihn. Beziehen Sie sich auf *Bühnenmodus*, um einzelne Skripte direkt auszuführen.

Aber das *2.8 Temperatur und Luftfeuchtigkeit lesen* verwendet den *Upload-Modus*.

4.4 2.1 Tischlampe

Hier verbinden wir eine LED auf dem Steckbrett und lassen den Sprite die Blinkfrequenz dieser LED steuern.

Wenn der Button-Sprite auf der Bühne angeklickt wird, blinkt die LED 5 Mal und stoppt dann.



4.4.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können die Komponenten auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	

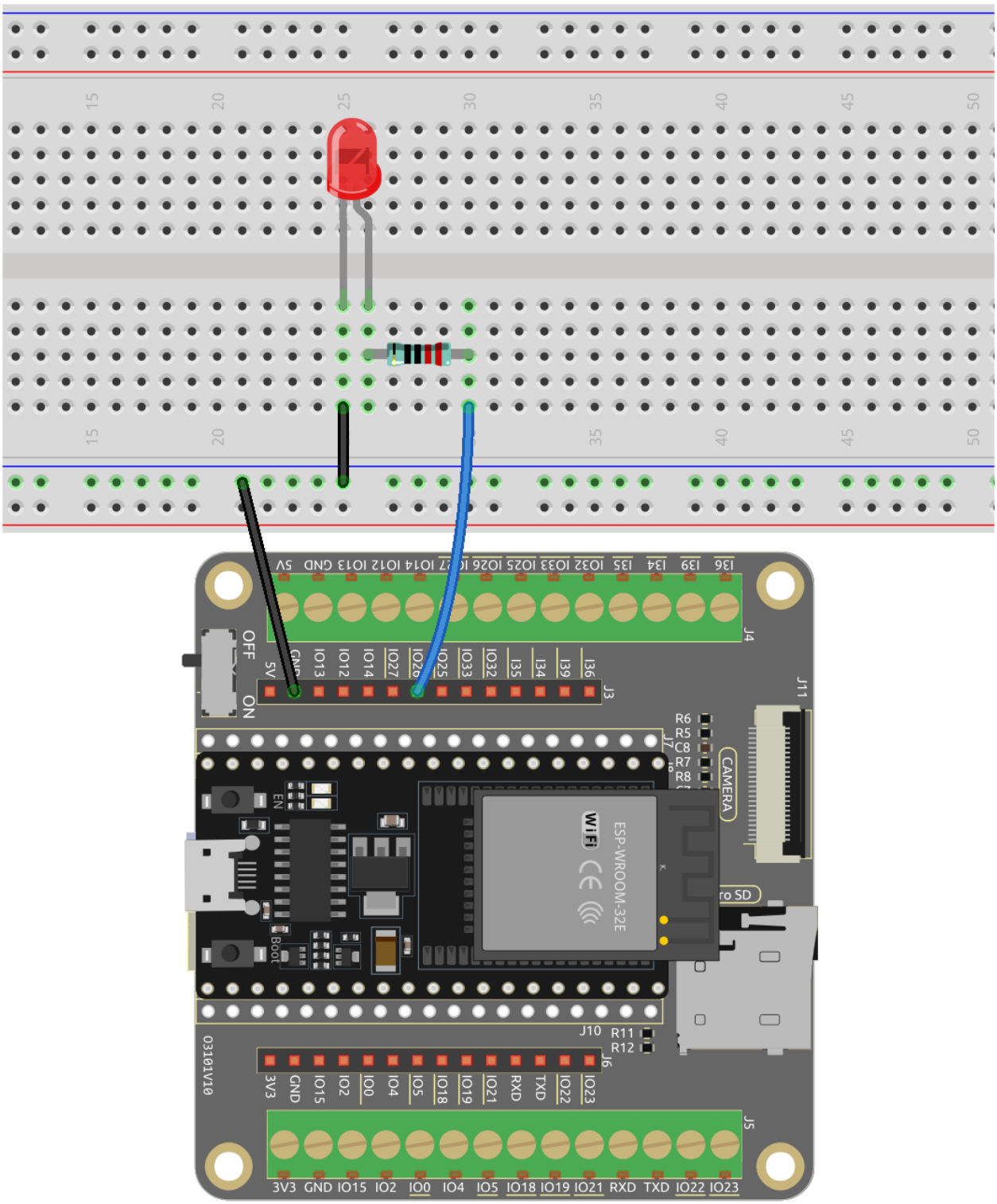
4.4.2 Was Sie Lernen Werden

- Steckbrett, LEDs und Widerstände
- Aufbau eines Stromkreises auf einem Steckbrett
- Löschen und Auswählen von Sprites
- Wechseln von Kostümen
- Festlegen einer begrenzten Anzahl von Wiederholungsschleifen

4.4.3 Schaltung Aufbauen

Folgen Sie dem untenstehenden Diagramm, um die Schaltung auf dem Steckbrett aufzubauen.

Da die Anode der LED (der längere Pin) über einen 220-Widerstand mit Pin 26 verbunden ist und die Kathode der LED mit GND verbunden ist, können Sie die LED zum Leuchten bringen, indem Sie Pin 9 auf ein hohes Level setzen.

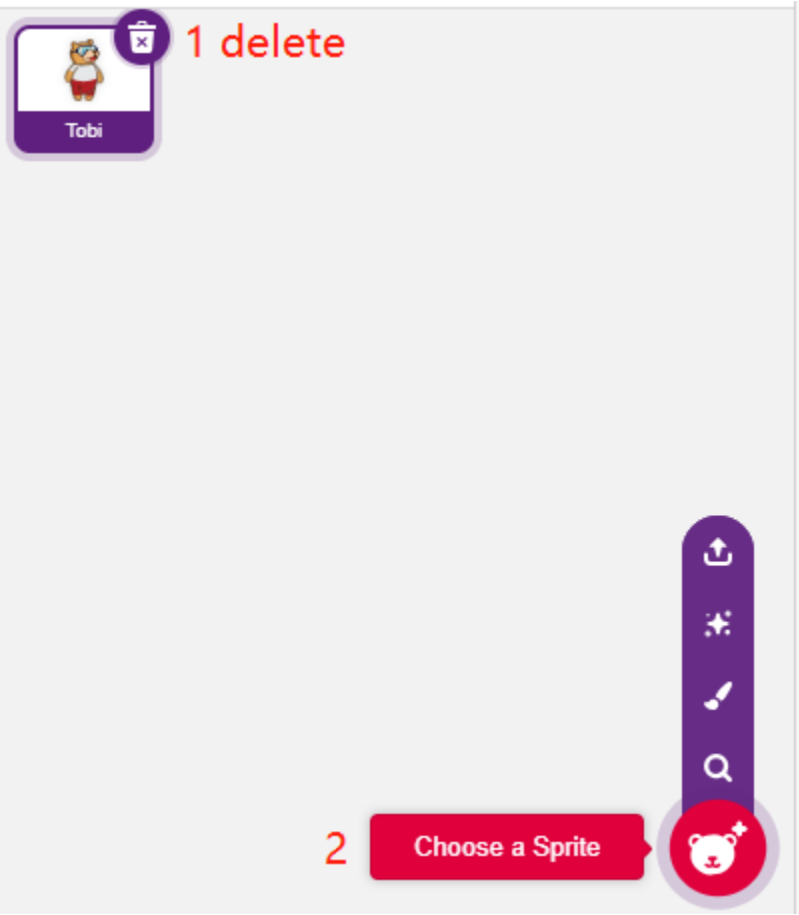

OFF
ON
ESP-WROOM-32E
CAMERA
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

4.4.4 Programmierung

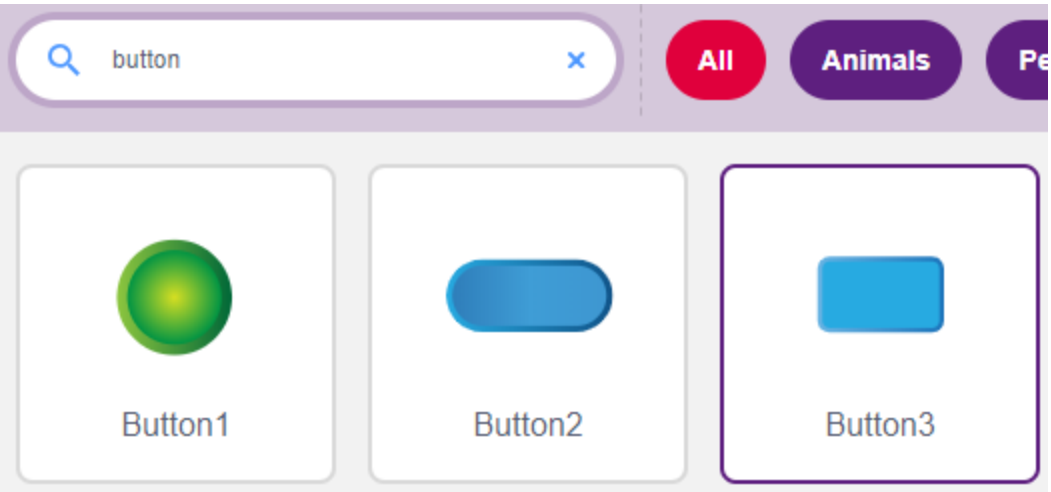
Die gesamte Programmierung ist in 3 Teile gegliedert: Der erste Teil ist die Auswahl des gewünschten Sprites, der zweite Teil ist das Wechseln des Kostüms für den Sprite, damit er klickbar erscheint, und der dritte Teil ist das Blinken der LED.

1. Button3-Sprite auswählen

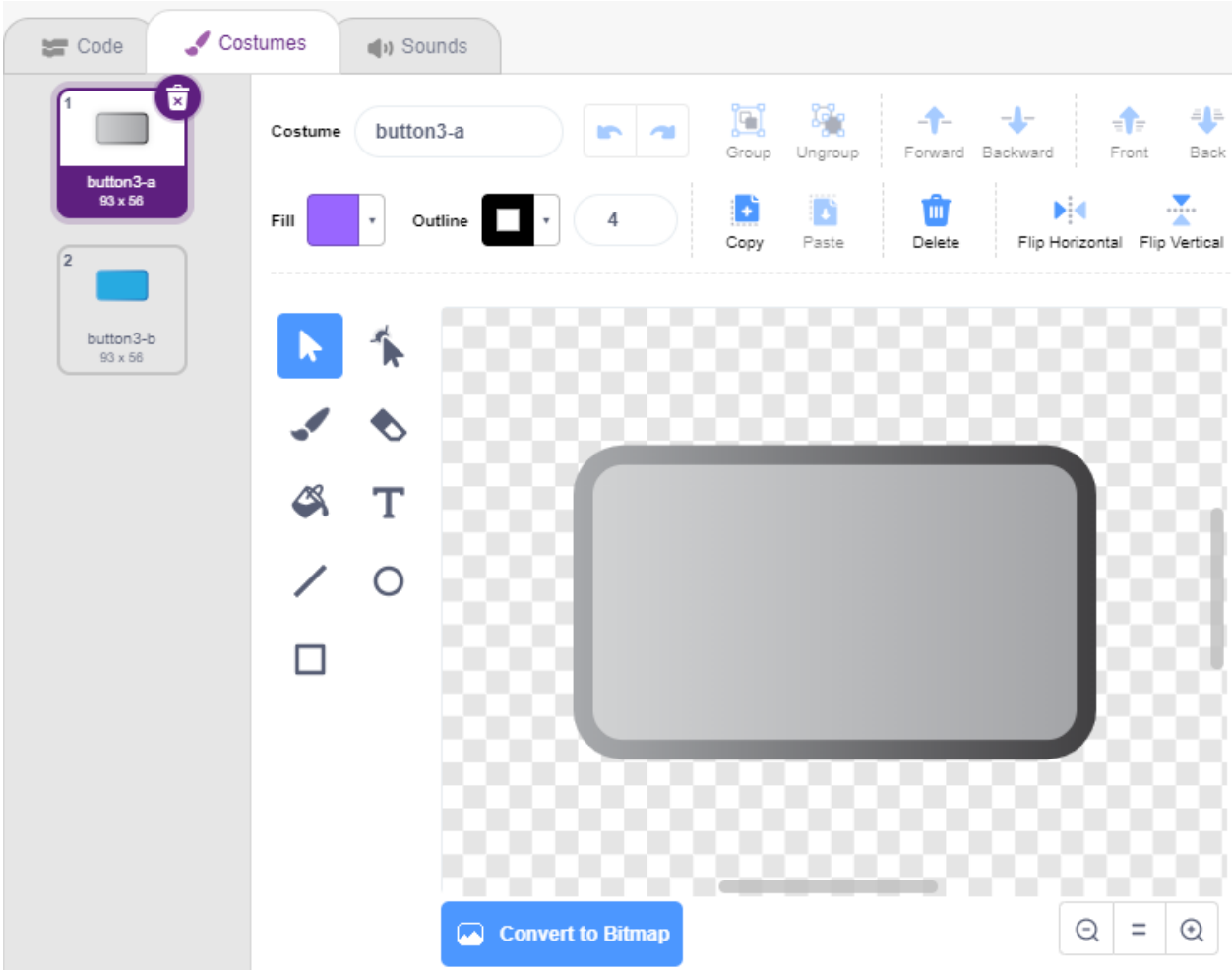
Löschen Sie das bestehende Tobi-Sprite mit dem Löschen-Knopf in der oberen rechten Ecke und wählen Sie ein Sprite erneut aus.



Hier wählen wir das **Button3**-Sprite.

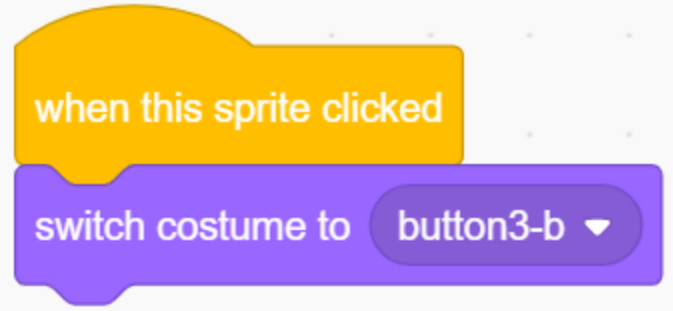


Klicken Sie in der oberen rechten Ecke auf Kostüme und Sie werden sehen, dass das Button3-Sprite 2 Kostüme hat. Wir setzen **button3-a** als freigegeben und **button3-b** als gedrückt.



2. Kostüme wechseln.

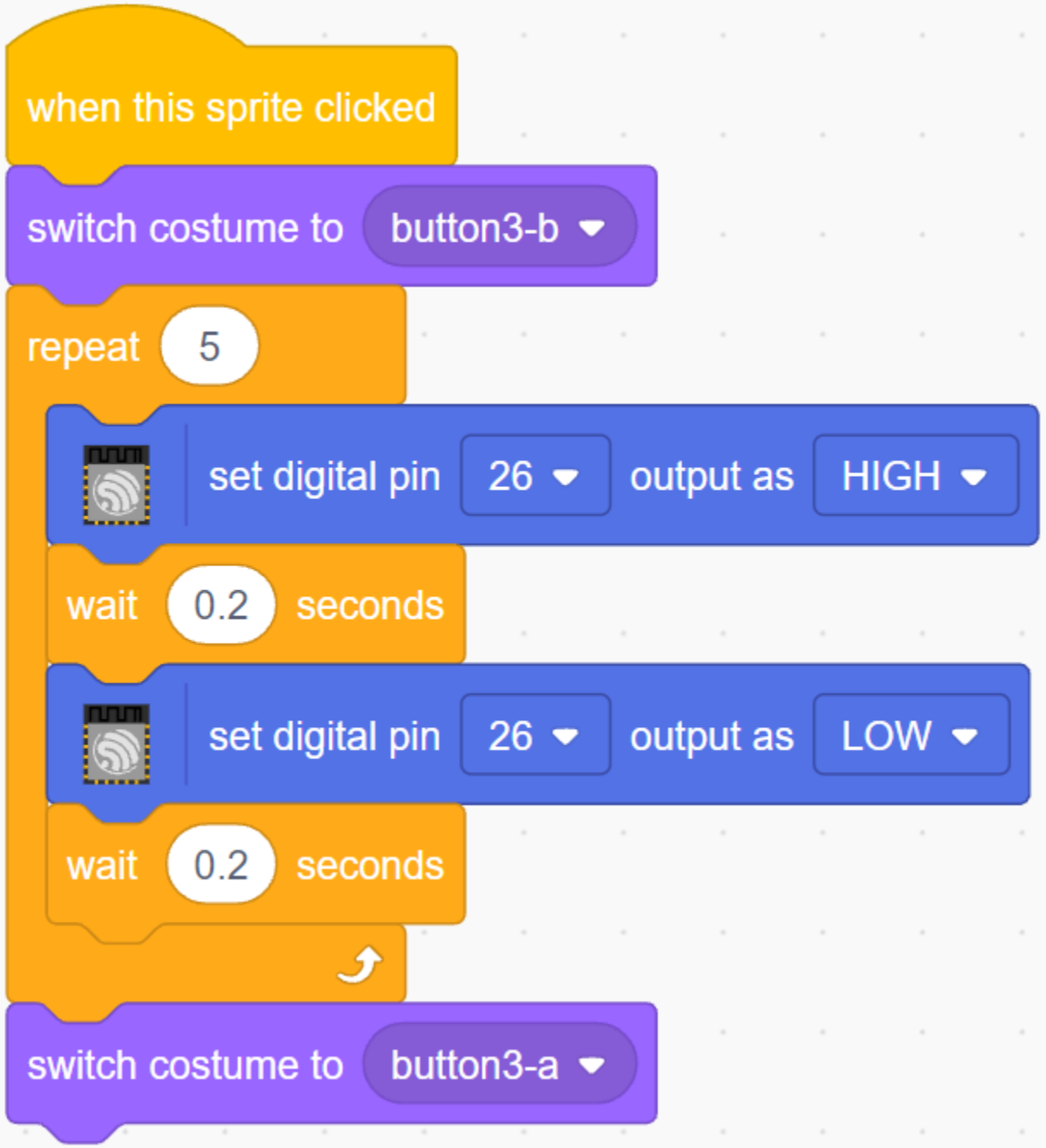
Wenn das Sprite angeklickt wird (**Events**-Palette), wechselt es zum Kostüm für **button3-b** (**looks**-Palette).



3. Die LED 5 Mal blinken lassen

Verwenden Sie den [Repeat]-Block, um die LED 5 Mal blinken zu lassen (High-> LOW Zyklus) und schließlich das Kostüm zurück auf **button3-a** zu wechseln.

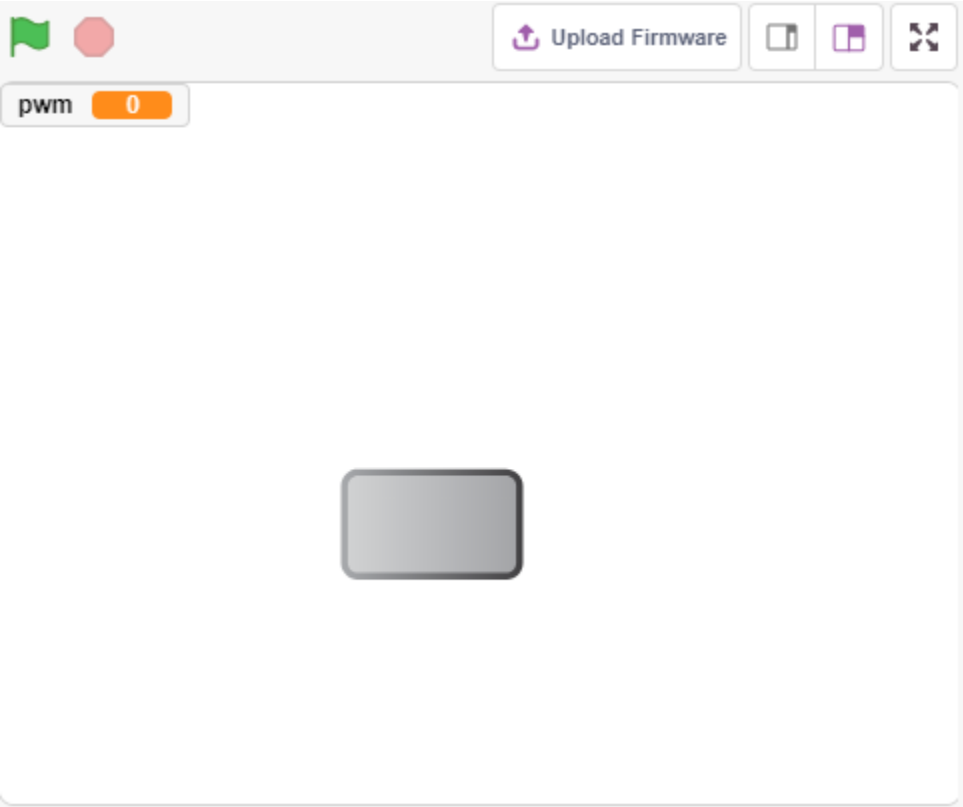
- [Repeat 10]: Begrenzte Anzahl von Wiederholungsschleifen, Sie können die Anzahl der Wiederholungen selbst festlegen, aus der **Control**-Palette.


when this sprite clicked
switch costume to button3-b
repeat 5
set digital pin 26 output as HIGH
wait 0.2 seconds
set digital pin 26 output as LOW
wait 0.2 seconds
switch costume to button3-a

4.5 2.2 Atmende LED

Nun verwenden wir eine andere Methode, um die Helligkeit der LED zu steuern. Im Gegensatz zum vorherigen Projekt wird hier die Helligkeit der LED langsam verringert, bis sie verschwindet.

Wenn das Sprite auf der Bühne angeklickt wird, nimmt die Helligkeit der LED langsam zu und erlischt dann sofort.



4.5.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können die Komponenten auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
LED	

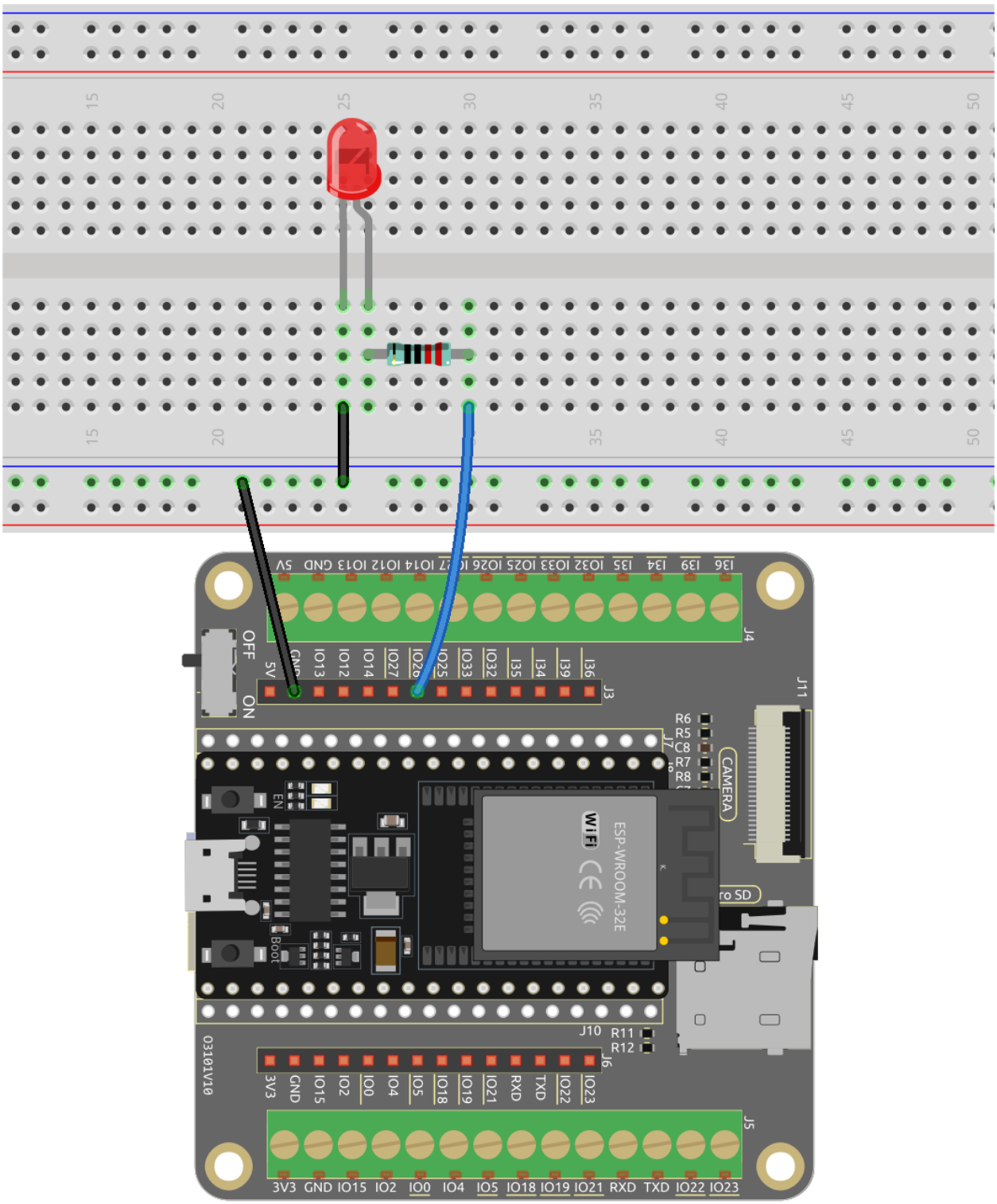
4.5.2 Was Sie Lernen Werden

- Setzen des Ausgabewerts des PWM-Pins
- Erstellen von Variablen
- Ändern der Helligkeit des Sprites

4.5.3 Schaltung Aufbauen

Dieses Projekt verwendet dieselbe Schaltung wie das vorherige Projekt *2.1 Tischlampe*, verwendet jedoch statt HIGH/LOW, um die LEDs zum Leuchten oder Ausschalten zu bringen, das PWM-Signal - Wikipedia, um die LED langsam aufleuchten oder dimmen zu lassen.

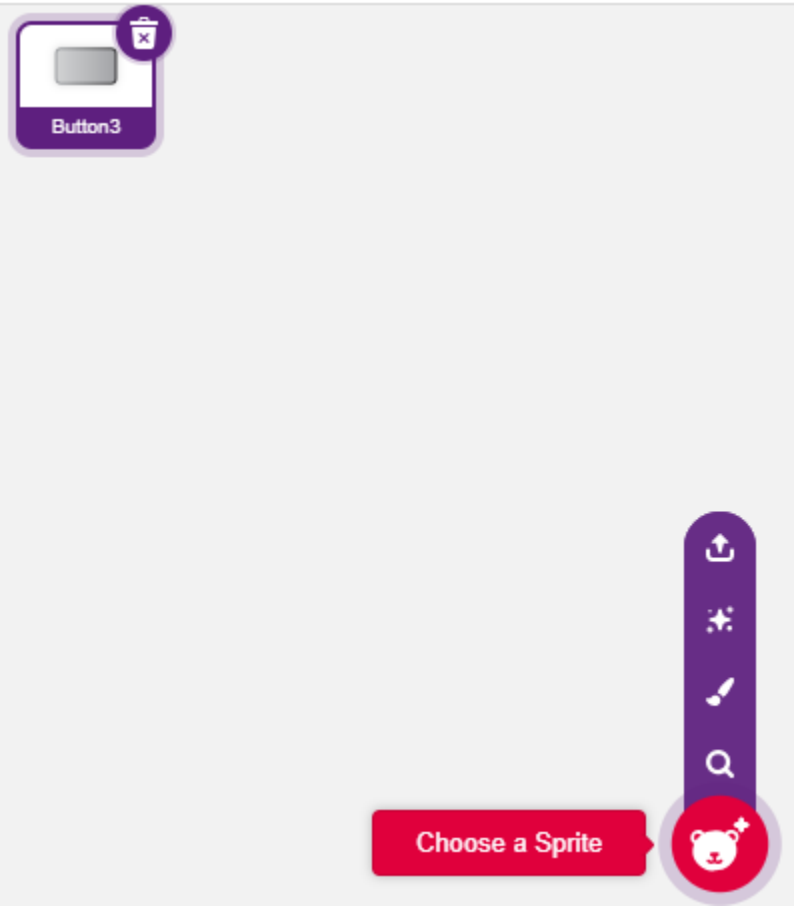
Der PWM-Signalbereich ist 0-255, auf dem ESP32-Board können die Pins 2, 5, 12~15, 18, 19, 21, 22, 25, 26 und 27 ein PWM-Signal ausgeben.


OFF
ON
ESP-WROOM-32E
CAMERA
J3
J4
J5
J6
J10
J11
R11
R12
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

4.5.4 Programmierung

1. Ein Sprite auswählen

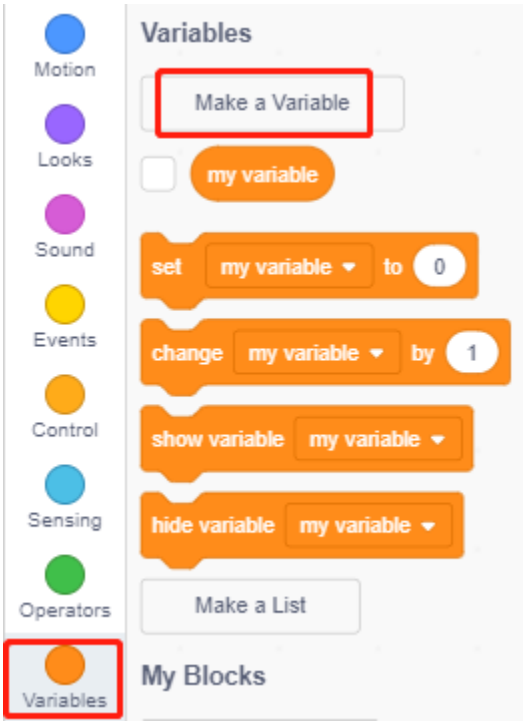
Löschen Sie das Standard-Sprite, klicken Sie auf den Button **Choose a Sprite** in der unteren rechten Ecke des Sprite-Bereichs, geben Sie **button3** in das Suchfeld ein und klicken Sie dann darauf, um es hinzuzufügen.



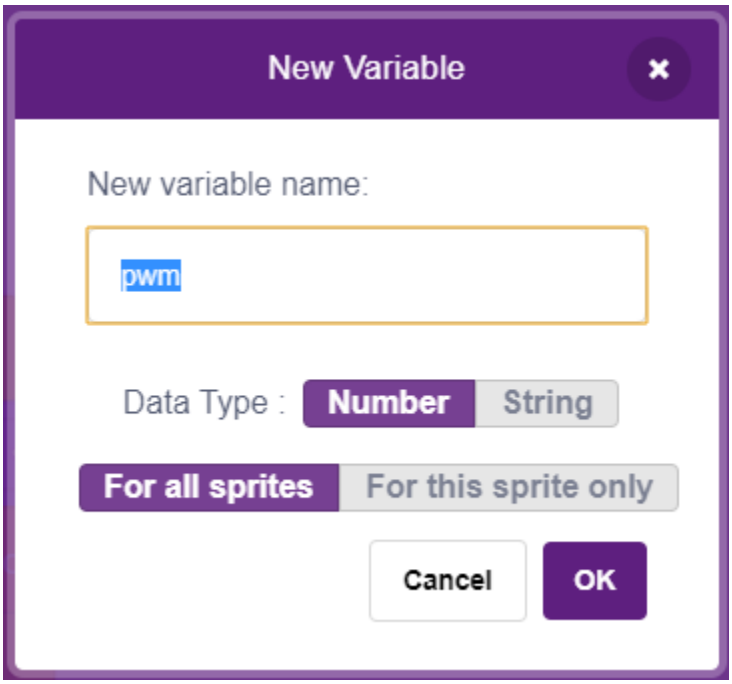
2. Eine Variable erstellen.

Erstellen Sie eine Variable namens **pwm**, um den Wert der PWM-Änderung zu speichern.

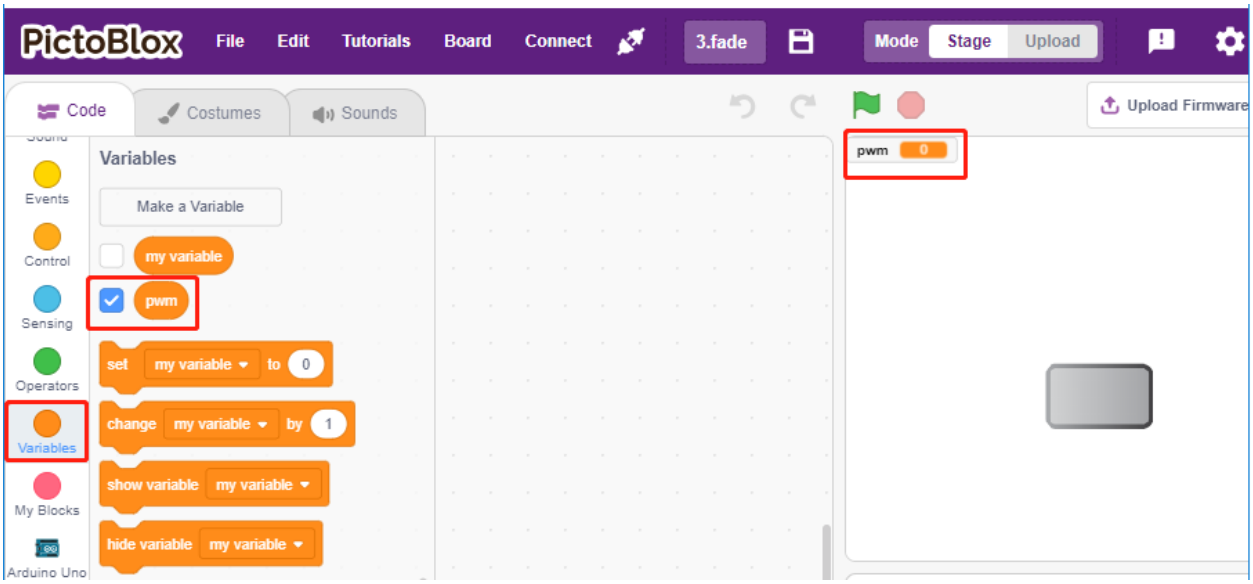
Klicken Sie auf die **Variables**-Palette und wählen Sie **Make a Variable**.



Geben Sie den Namen der Variablen ein, es kann jeder Name sein, aber es wird empfohlen, seine Funktion zu beschreiben. Der Datentyp ist Zahl und Für alle Sprites.



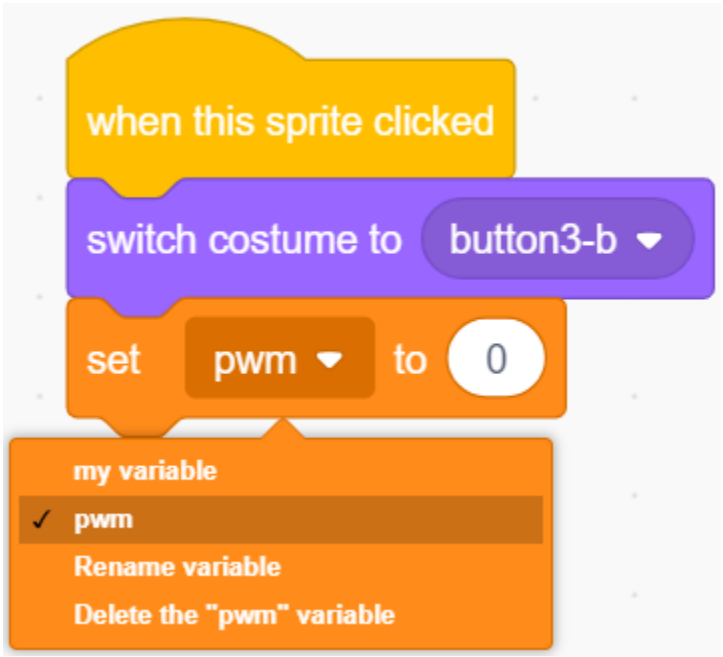
Sobald erstellt, sehen Sie **pwm** in der **Variables**-Palette und im angekreuzten Zustand, was bedeutet, dass diese Variable auf der Bühne erscheinen wird. Sie können versuchen, es abzuwählen, um zu sehen, ob pwm noch auf der Bühne vorhanden ist.



3. Den Anfangszustand festlegen

Wenn das **button3**-Sprite angeklickt wird, wechseln Sie das Kostüm zu **button-b** (angeklickter Zustand) und setzen Sie den Anfangswert der Variablen **pwm** auf 0.

- [set pwm to 0]: aus der **Variables**-Palette, verwendet, um den Wert der Variablen festzulegen.

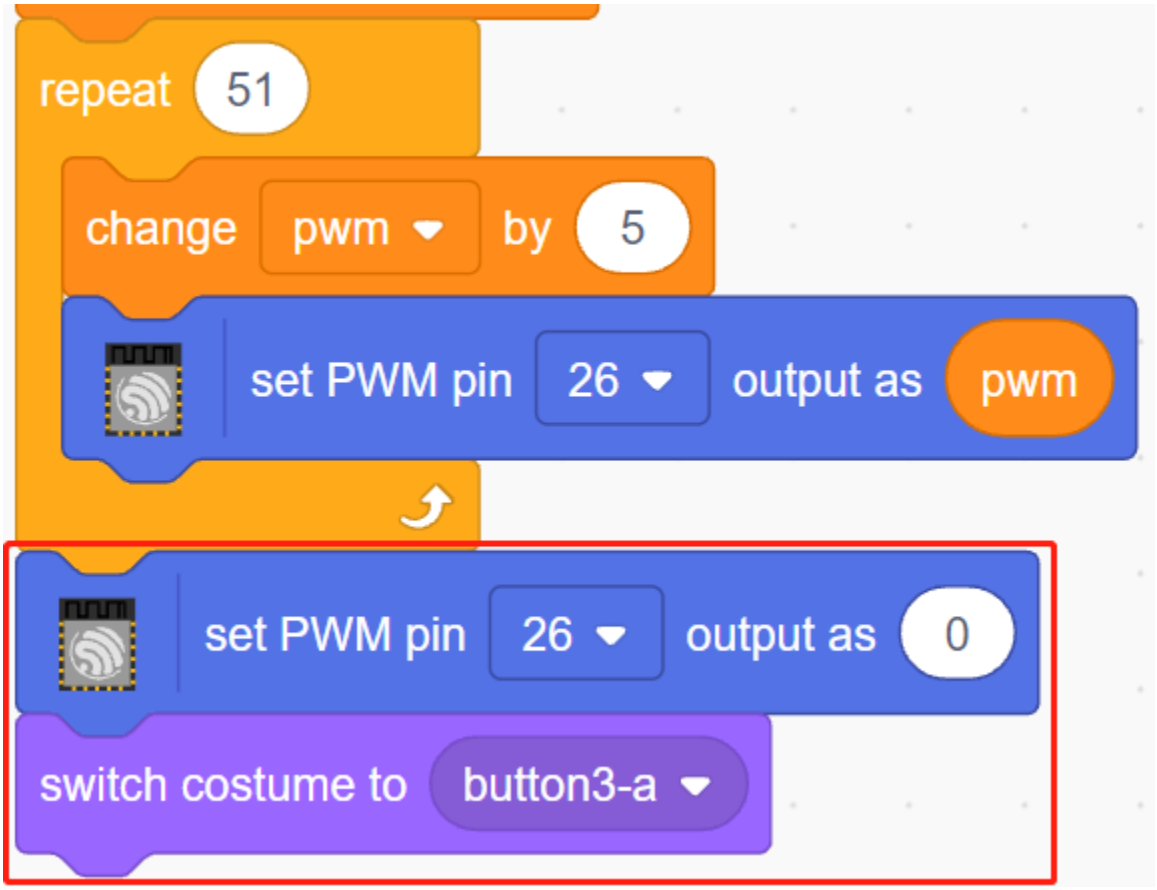


4. Die LED immer heller machen

Da der Bereich von pwm 255 ist, wird durch den [repeat]-Block die Variable **pwm** um 5 auf 255 erhöht und dann in den Block [set PWM pin] eingesetzt, sodass Sie sehen können, wie die LED langsam aufleuchtet.

- [change pwm by 5]: aus der **Variables**-Palette, lässt die Variable jedes Mal eine bestimmte Zahl ändern. Es kann eine positive oder negative Zahl sein, positiv bedeutet jedes Mal eine Zunahme, negativ bedeutet jedes Mal eine Abnahme, zum Beispiel wird hier die Variable pwm jedes Mal um 5 erhöht.
- [set PWM pin]: aus der **ESP32**-Palette, verwendet, um den Ausgabewert des PWM-Pins festzulegen.

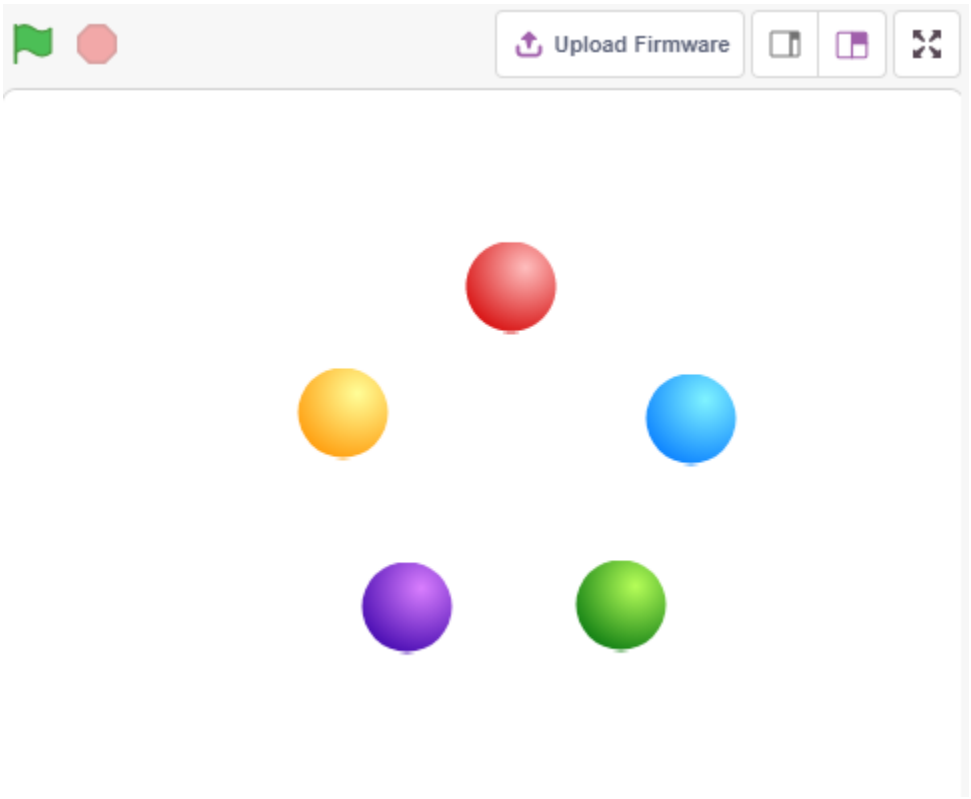
Schließlich wechseln Sie das Kostüm von button3 zurück zu **button-a** und machen den PWM-Pin-Wert 0, sodass die LED langsam aufleuchtet und dann wieder erlischt.



4.6 2.3 Farbenfrohe Bälle

In diesem Projekt werden wir die RGB-LEDs dazu bringen, verschiedene Farben anzuzeigen.

Durch Klicken auf unterschiedlich gefärbte Bälle im Bühnenbereich wird die RGB-LED in verschiedenen Farben leuchten.



4.6.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

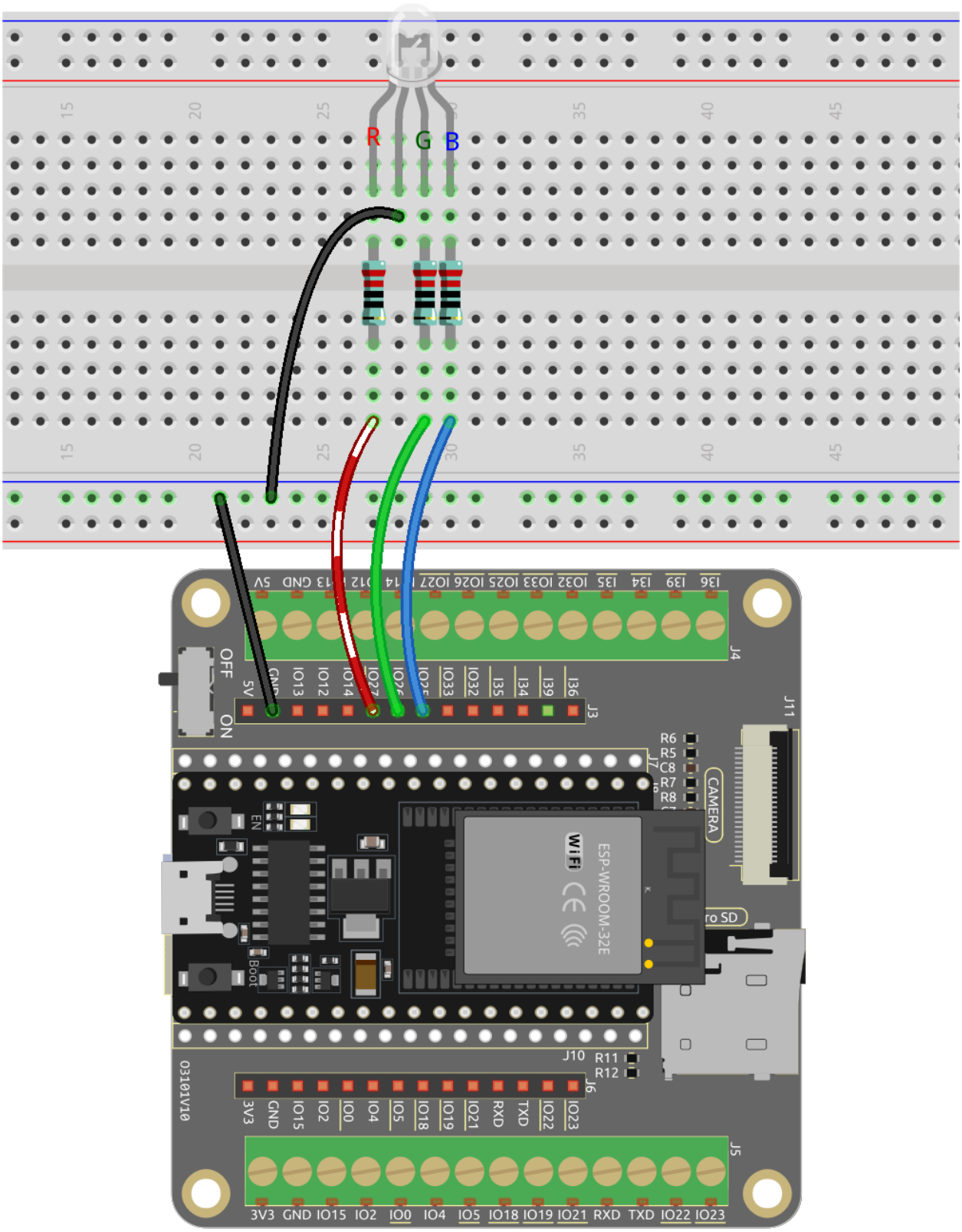
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
RGB LED	

4.6.2 Was Sie Lernen Werden

- Das Prinzip der RGB-LED
- Kopieren von Sprites und Auswahl verschiedener Kostüme
- Überlagerung der drei Grundfarben

4.6.3 Schaltung Aufbauen

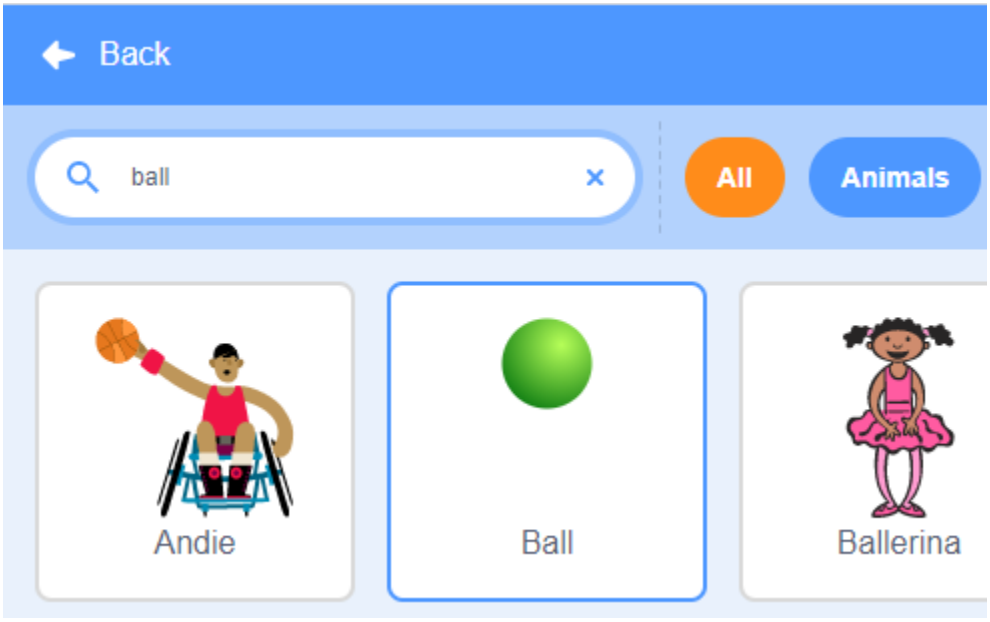
Eine RGB-LED beinhaltet drei LEDs in den Farben Rot, Grün und Blau in einer transparenten oder halbtransparenten Kunststoffhülle. Sie kann verschiedene Farben anzeigen, indem die Eingangsspannung an den drei Pins geändert und diese überlagert werden, was laut Statistik 16.777.216 verschiedene Farben erzeugen kann.


R
G
B
ESP-WROOM-32E
CAMERA

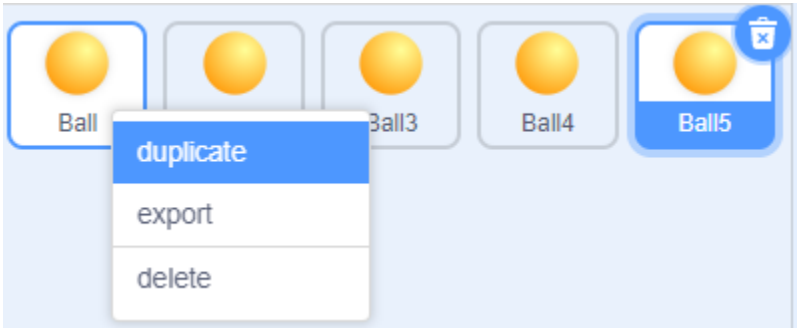
4.6.4 Programmierung

1. Wähle ein Sprite aus

Lösche das Standard-Sprite und wähle das **Ball**-Sprite aus.

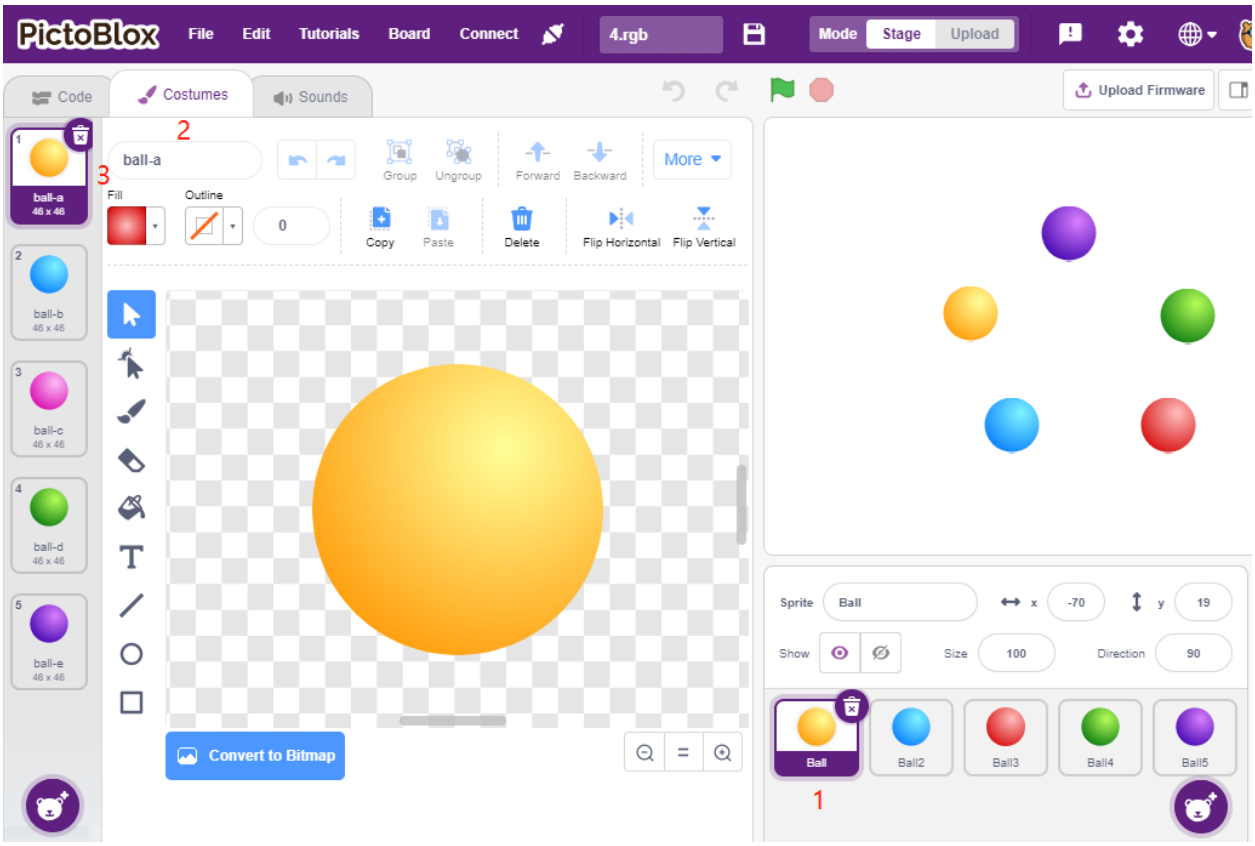


Und dupliziere es 5 Mal.



Wähle unterschiedliche Kostüme für diese 5 **Ball**-Sprites aus und verschiebe sie an die entsprechenden Positionen.

Bemerkung: Die Kostümfarbe des **Ball3**-Sprites muss manuell in Rot geändert werden.

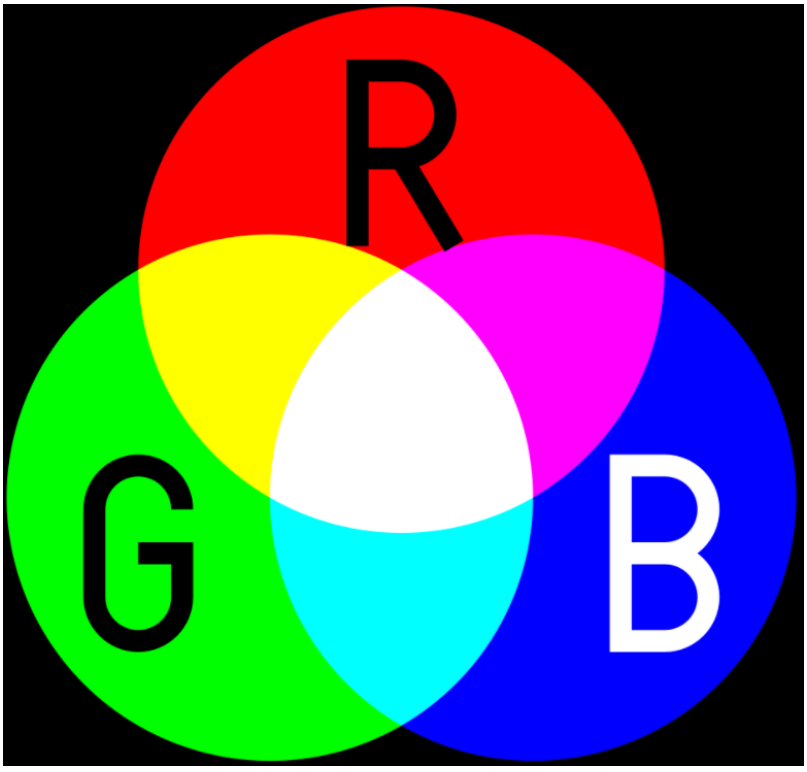


2. RGB-LEDs in der entsprechenden Farbe leuchten lassen

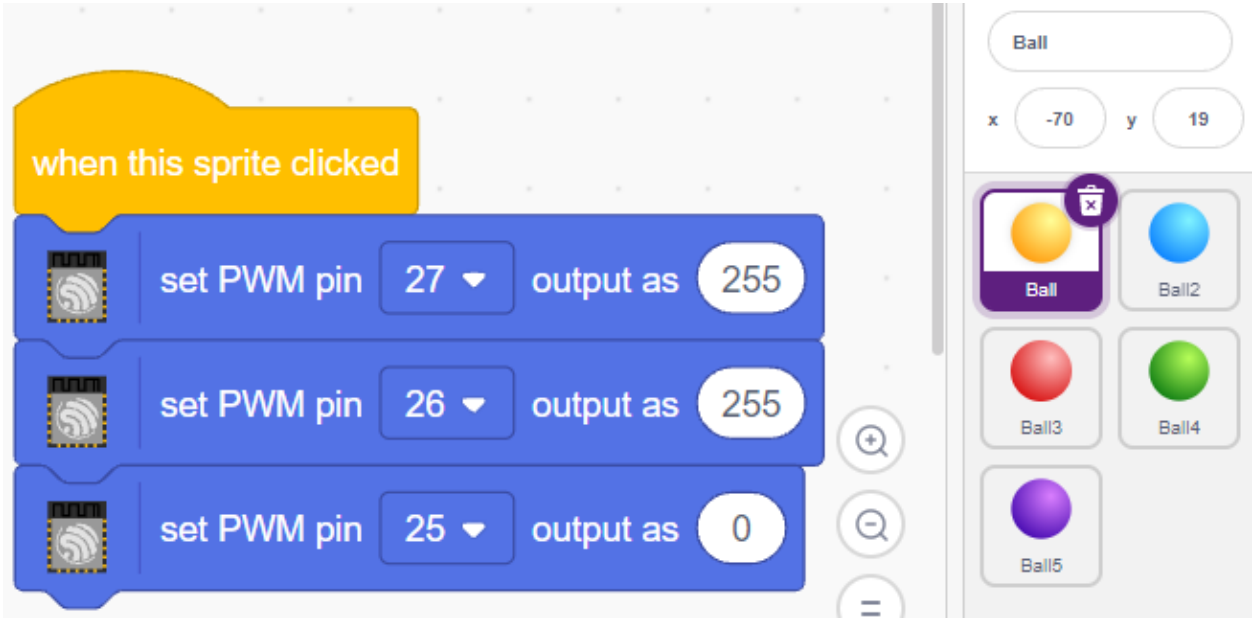
Bevor wir den Code verstehen, müssen wir das RGB-Farbmodell verstehen.

Das RGB-Farbmodell ist ein additives Farbmodell, bei dem Rot-, Grün- und Blaulicht auf verschiedene Weise zusammengefügt werden, um ein breites Spektrum an Farben zu erzeugen.

Additive Farbmischung: Rot und Grün ergeben Gelb; Grün und Blau ergeben Cyan; Blau und Rot ergeben Magenta; alle drei Grundfarben zusammen ergeben Weiß.



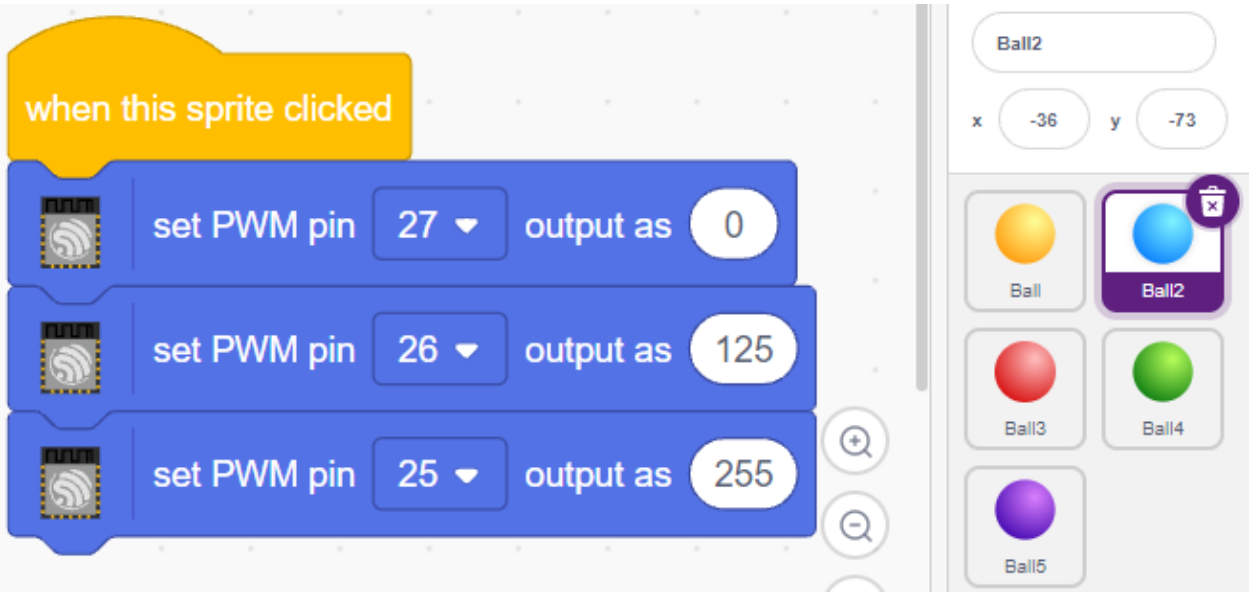
Der Code, um die RGB-LED gelb leuchten zu lassen, lautet daher wie folgt.



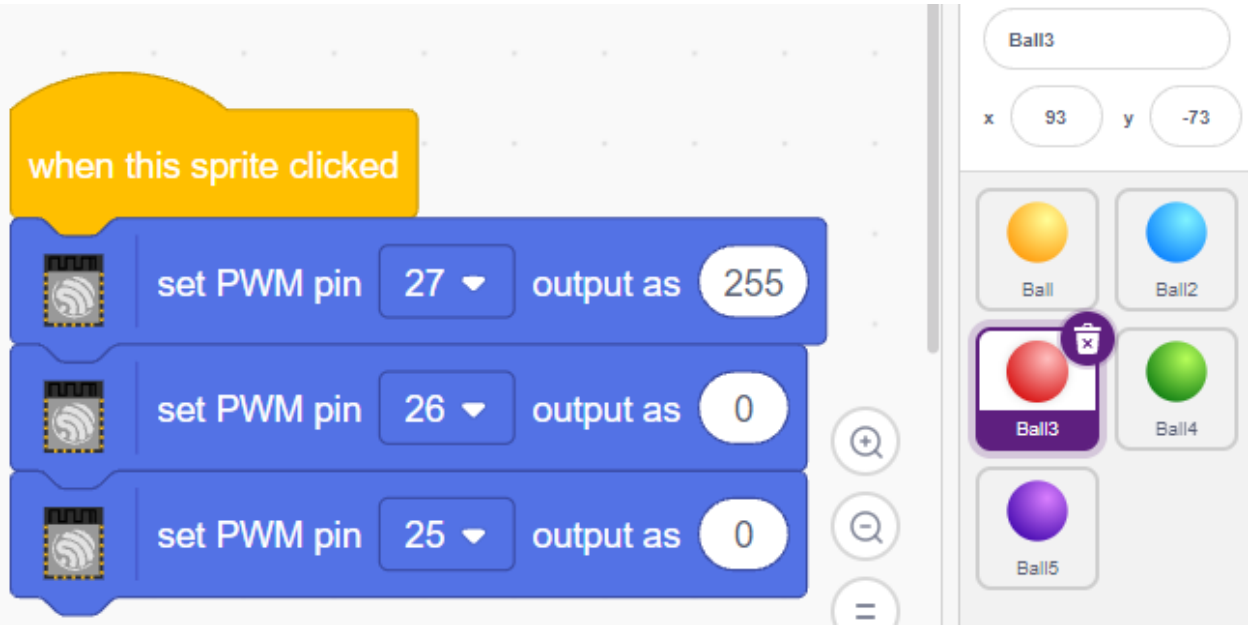
Wenn das Ball-Sprite (gelber Ball) angeklickt wird, setzen wir Pin 27 auf hoch (rote LED an), Pin 26 auf hoch (grüne LED an) und Pin 25 auf niedrig (blaue LED aus), sodass die RGB-LED gelb leuchtet.

Sie können für andere Sprites auf die gleiche Weise Codes schreiben, um die RGB-LEDs in den entsprechenden Farben leuchten zu lassen.

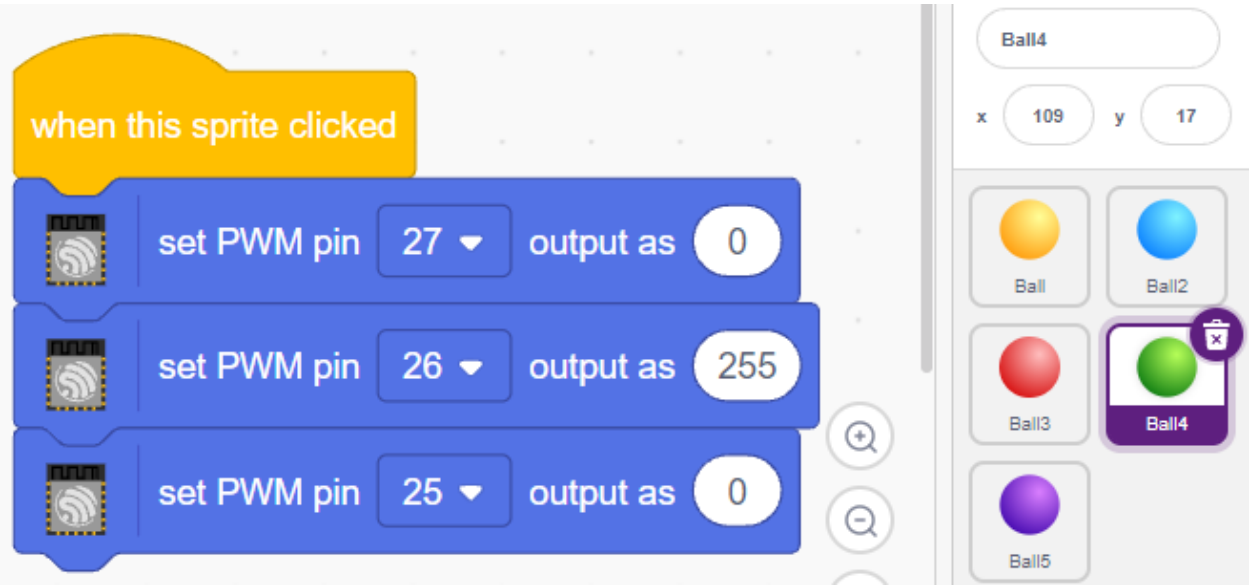
3. Ball2-Sprite (hellblau)



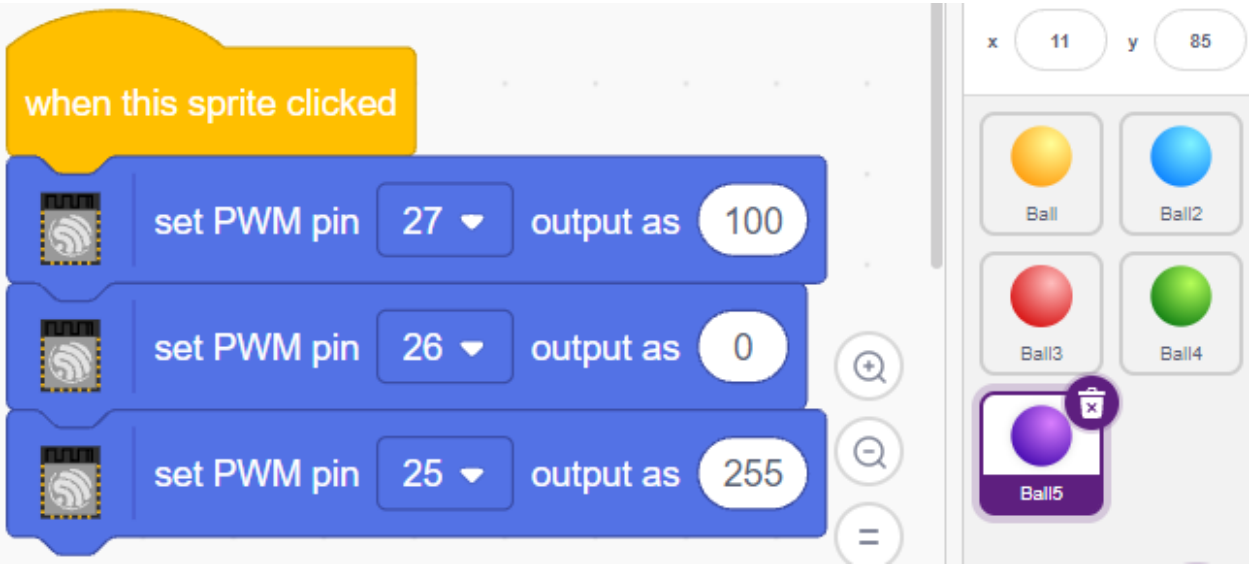
4. Ball3-Sprite (rot)



5. Ball4-Sprite (grün)



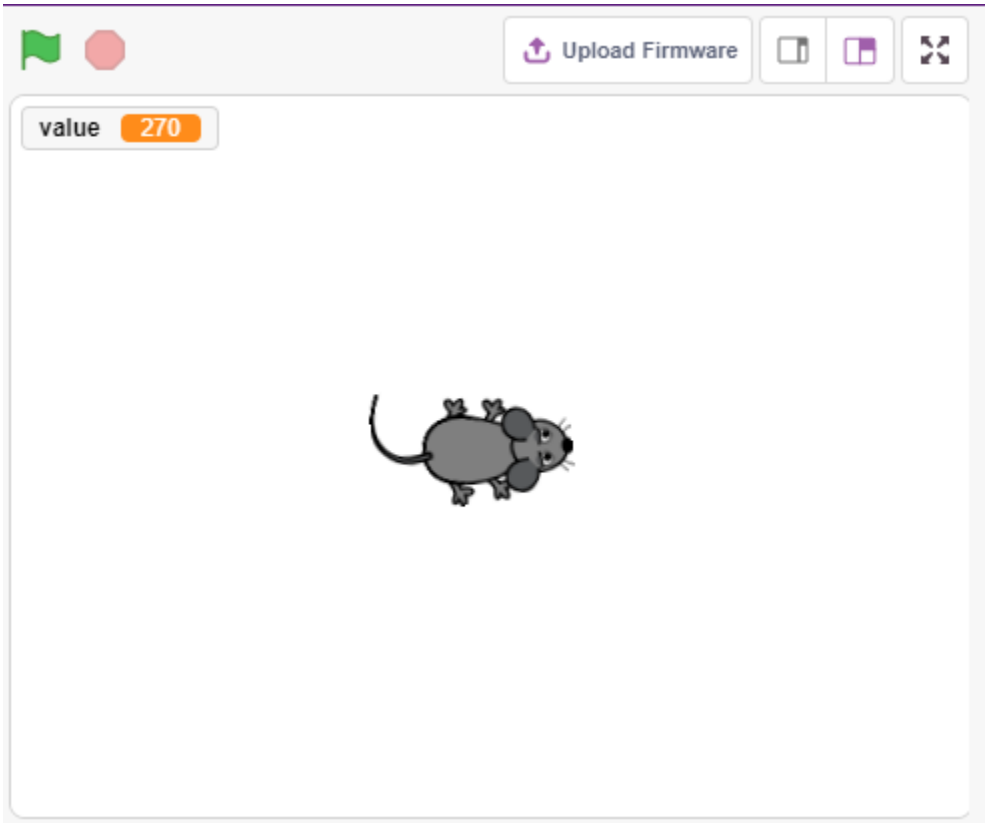
6. Ball5-Sprite (lila)



4.7 2.4 Bewegliche Maus

Heute werden wir ein von einem Potentiometer gesteuertes Mausspielzeug bauen.

Wenn die grüne Flagge angeklickt wird, bewegt sich die Maus auf der Bühne vorwärts, und wenn Sie das Potentiometer drehen, ändert die Maus die Bewegungsrichtung.



4.7.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

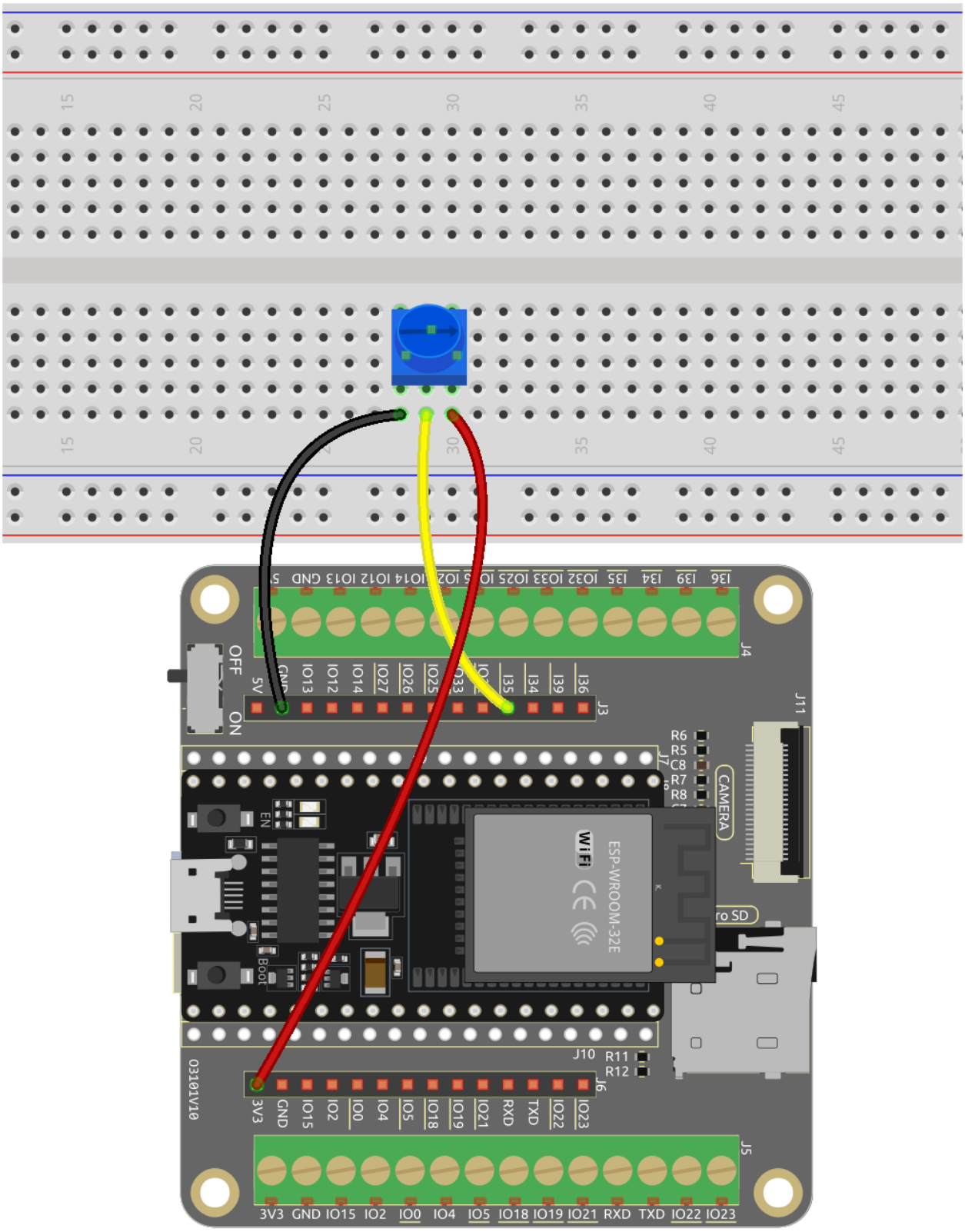
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Potentiometer	

4.7.2 Was Sie Lernen Werden

- Potentiometerprinzip
- Analogen Pin lesen und Bereich verstehen
- Einen Bereich in einen anderen abbilden
- Bewegung und Richtungsänderung des Sprites

4.7.3 Schaltung Aufbauen

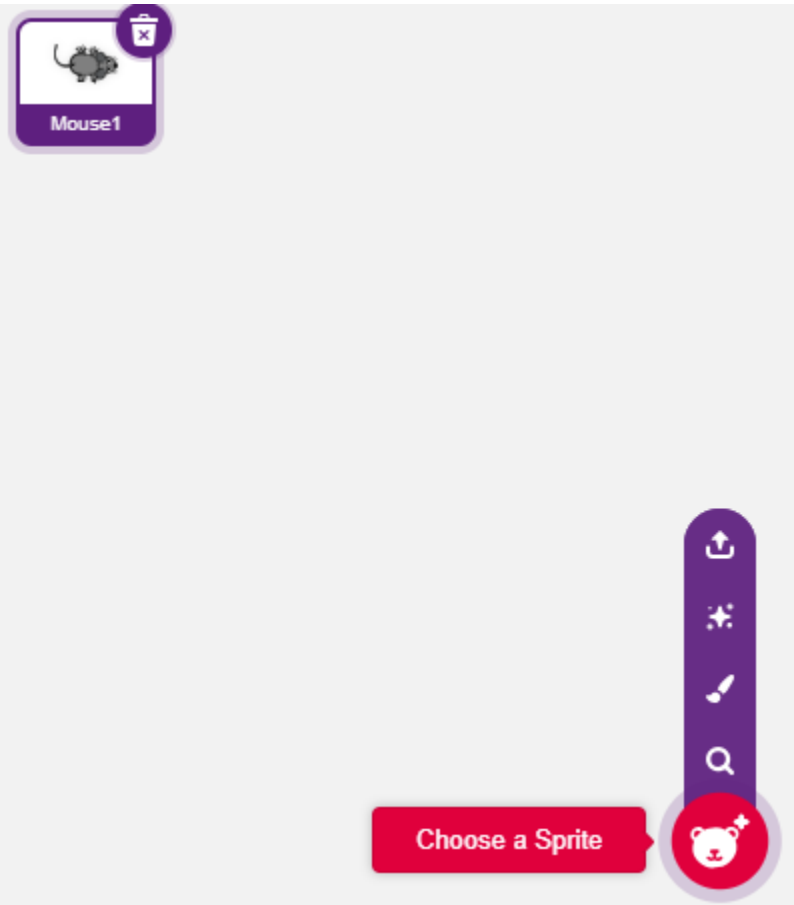
Das Potentiometer ist ein Widerstandselement mit 3 Anschlüssen, wobei die beiden seitlichen Pins mit 5V und GND verbunden sind und der mittlere Pin mit Pin35. Nach der Umwandlung durch den ADC-Wandler des ESP32 liegt der Wertebereich bei 0-4095.


OFF
ON
5V
GND
IO13
IO12
IO14
IO27
IO26
IO25
IO33
IO32
I35
I34
I39
I36
J4
J3
J11
R6
R5
C8
R7
R8
J7
CAMERA
EN
Boot
WiFi
ESP-WROOM-32E
J10
R11
R12
J6
3V3
GND
IO15
IO2
IO0
IO4
IO5
IO18
IO19
IO21
RXD
TXD
IO22
IO23
J5
03101V10

4.7.4 Programmierung

1. Wähle ein Sprite aus

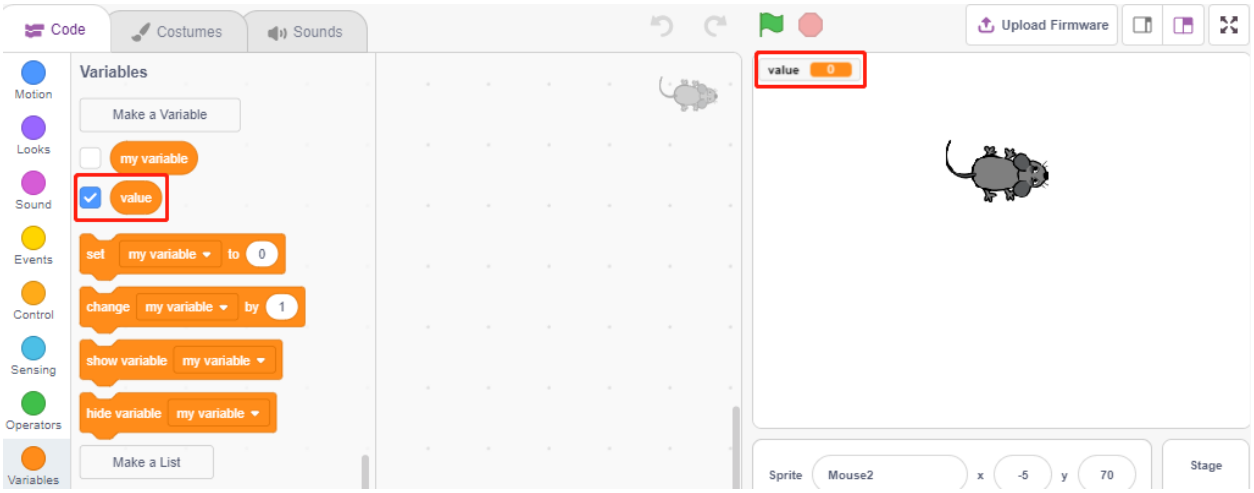
Lösche das Standard-Sprite, klicke auf den **Choose a Sprite**-Button in der unteren rechten Ecke des Sprite-Bereichs, gib **mouse** in das Suchfeld ein und klicke dann darauf, um es hinzuzufügen.



2. Eine Variable erstellen

Erstelle eine Variable namens **value** um den gelesenen Wert des Potentiometers zu speichern.

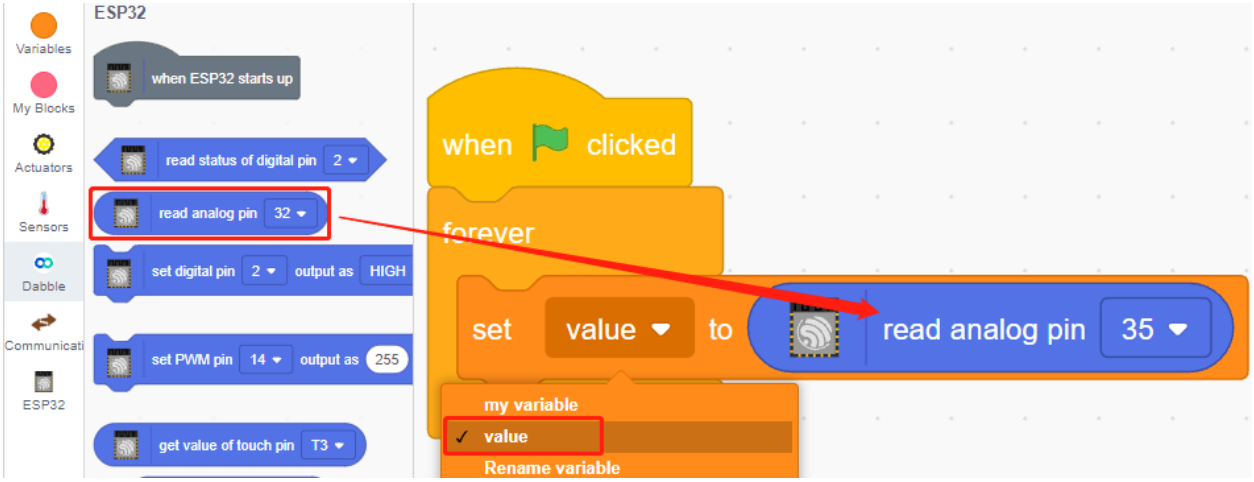
Sobald erstellt, erscheint **value** im **Variables**-Palette und im angekreuzten Zustand, was bedeutet, dass diese Variable auf der Bühne erscheint.



3. Den Wert von Pin35 lesen

Speichere den gelesenen Wert von Pin35 in die Variable **value**.

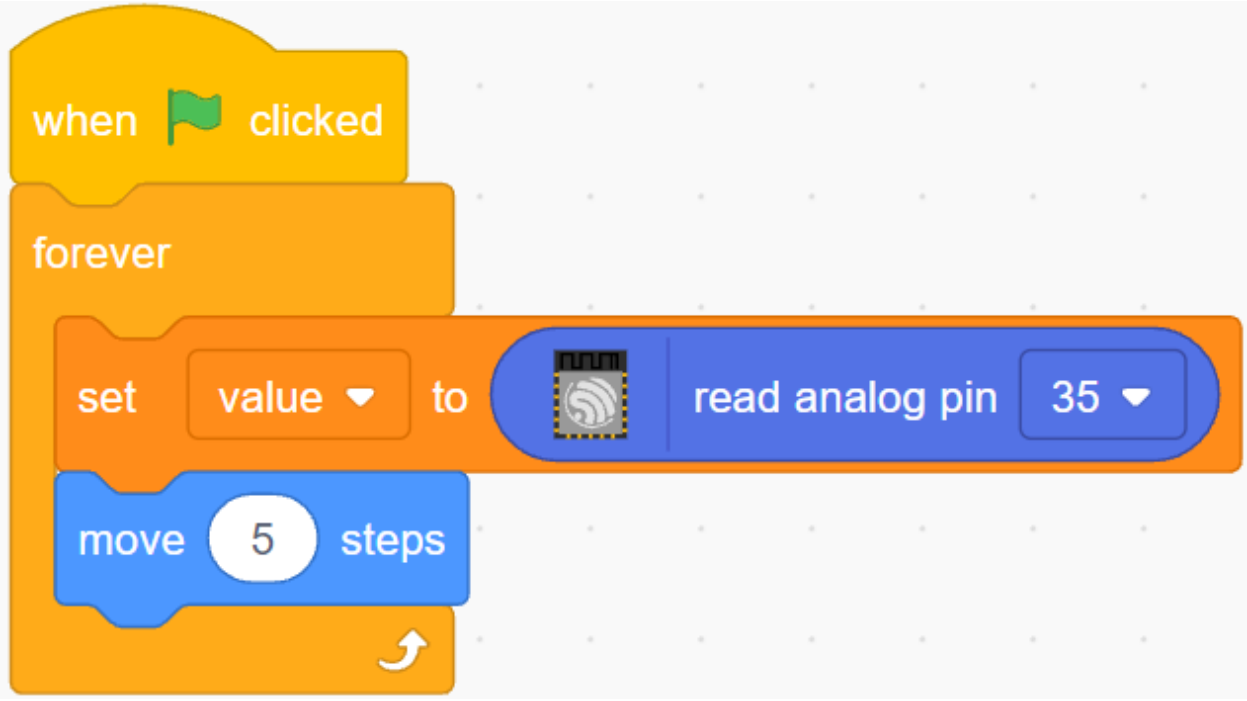
- [set my variable to 0]: Setze den Wert der Variable.
- [read analog pin ()]: Lies den Wert von Pins im Bereich von 0-4095.



Um durchgehend lesen zu können, musst du den [forever]-Block verwenden. Klicke auf dieses Skript, um es auszuführen, drehe das Potentiometer in beide Richtungen und du wirst sehen, dass der Wertebereich 0-1023 ist.

4. Bewege das Sprite

Verwende den [move steps]-Block, um das Sprite zu bewegen. Führe das Skript aus und du wirst sehen, dass sich das Sprite von der Mitte nach rechts bewegt.

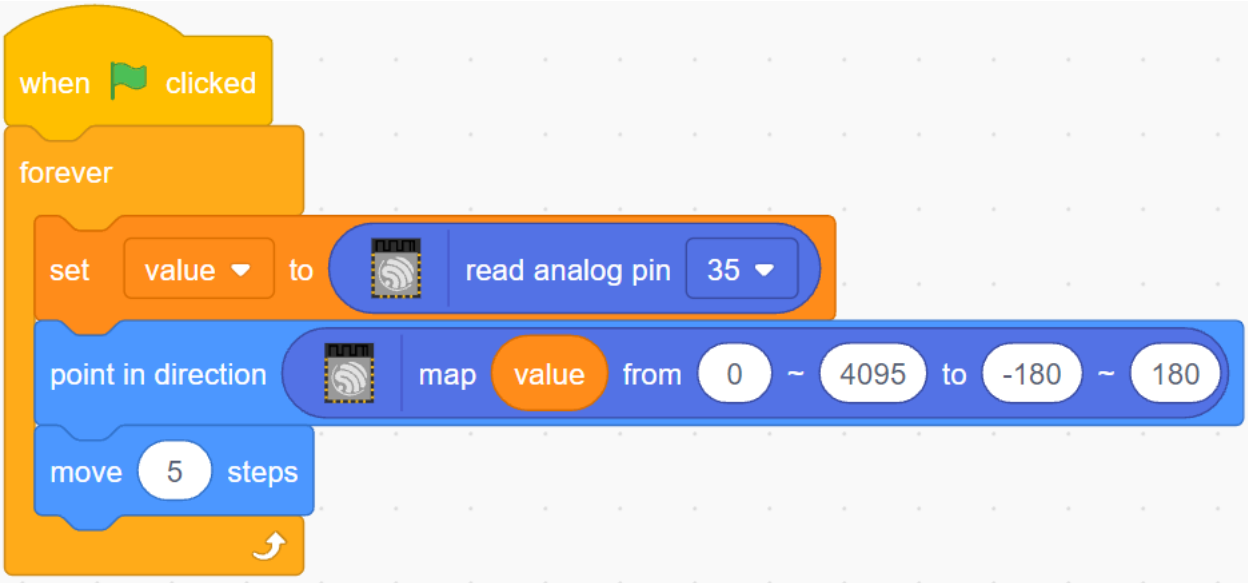


5. Die Bewegungsrichtung des Sprites ändern

Ändere jetzt die Bewegungsrichtung des Sprites durch den Wert von Pin35. Da der Wert von Pin35 von 0-4095 reicht, aber die Rotationsrichtung des Sprites -180~180 ist, muss ein [map]-Block verwendet werden.

Füge auch [when green flag clicked] am Anfang hinzu, um das Skript zu starten.

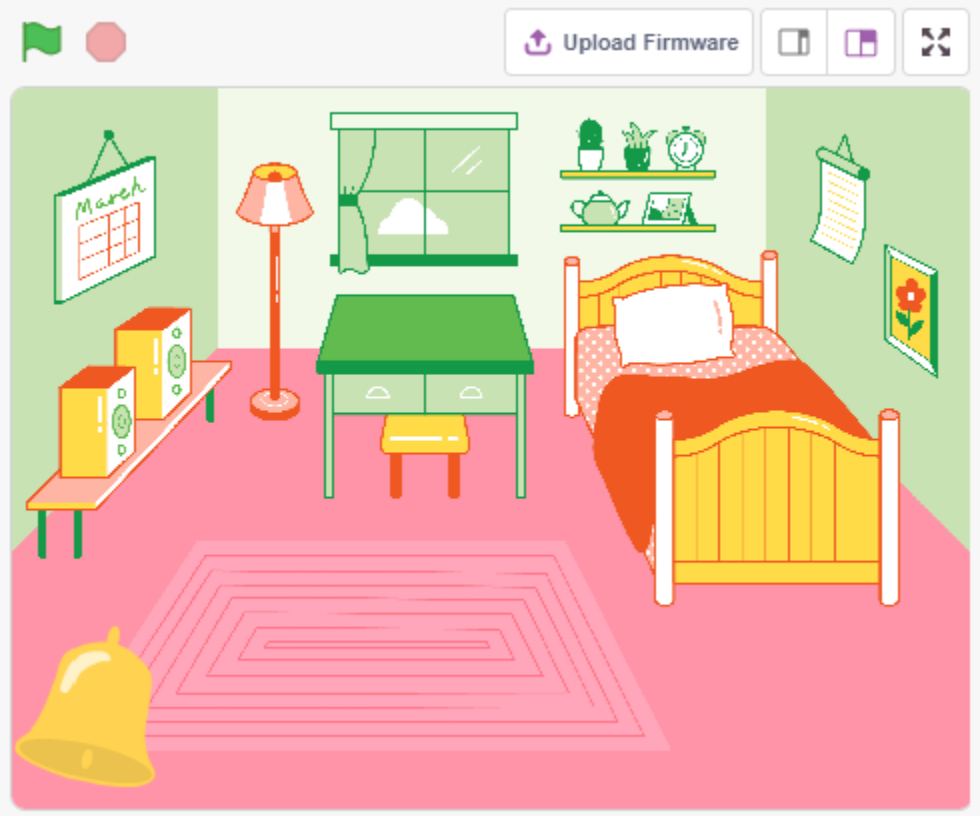
- [point in direction]: Setze den Lenkwinkel des Sprites, aus dem **Motion**-Palette.
- [map from to]: Abbilde einen Bereich auf einen anderen Bereich.



4.8 2.5 Türklingel

Hier werden wir den Knopf und die Glocke auf der Bühne verwenden, um eine Türklingel zu machen.

Nachdem die grüne Flagge angeklickt wurde, können Sie den Knopf drücken und die Glocke auf der Bühne wird einen Ton machen.



4.8.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Taste	

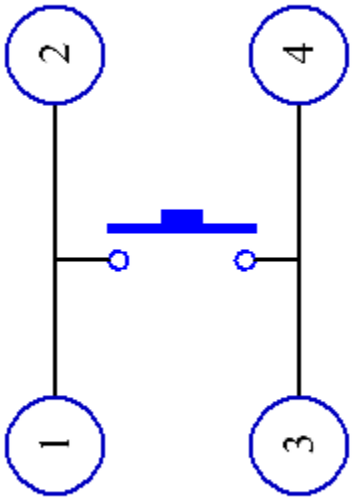
4.8.2 Was Sie Lernen Werden

- Funktionsweise des Knopfes
- Digitalen Pin lesen und Bereich verstehen
- Erstellung einer bedingten Schleife
- Hinzufügen eines Hintergrunds
- Ton abspielen

4.8.3 Schaltung Aufbauen

Der Knopf ist ein 4-poliges Gerät, da Pin 1 mit Pin 2 verbunden ist und Pin 3 mit Pin 4, wenn der Knopf gedrückt wird, sind die 4 Pins verbunden, wodurch der Stromkreis geschlossen wird.

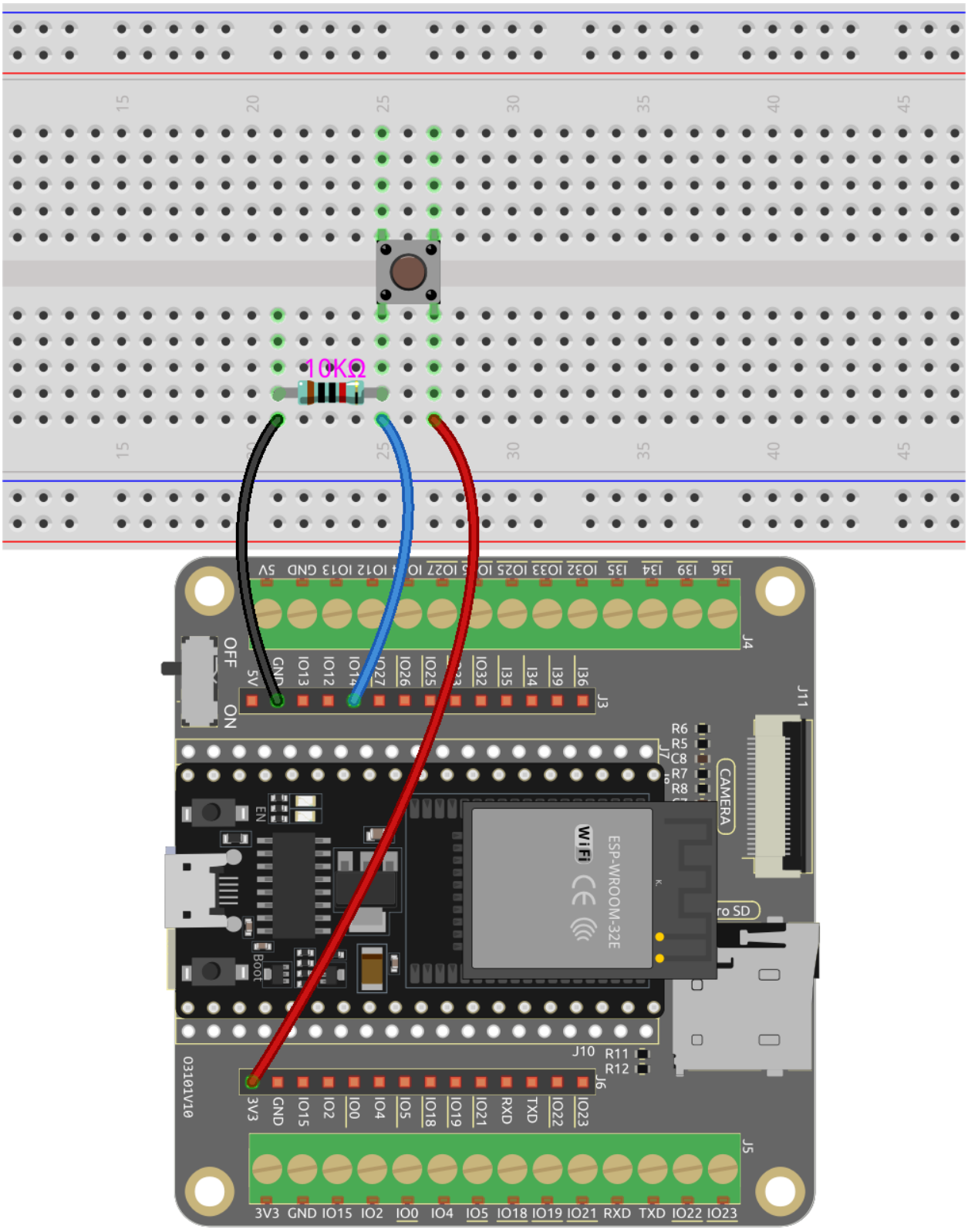
Button



Internal Structure

Bauen Sie die Schaltung gemäß dem folgenden Diagramm auf.

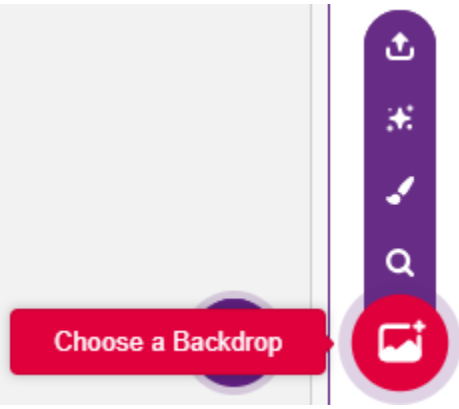
- Verbinden Sie einen der Pins auf der linken Seite des Knopfes mit Pin14, der mit einem Pull-Down-Widerstand und einem 0,1uF (104) Kondensator verbunden ist (um Schwankungen zu eliminieren und ein stabiles Level auszugeben, wenn der Knopf betätigt wird).
- Verbinden Sie das andere Ende des Widerstands und des Kondensators mit GND und einen der Pins auf der rechten Seite des Knopfes mit 5V.


10KΩ
OFF
ON
CAMERA
ESP-WROOM-32E
WiFi
Boot
EN
J3
J4
J5
J6
J10
J11
R11
R12
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

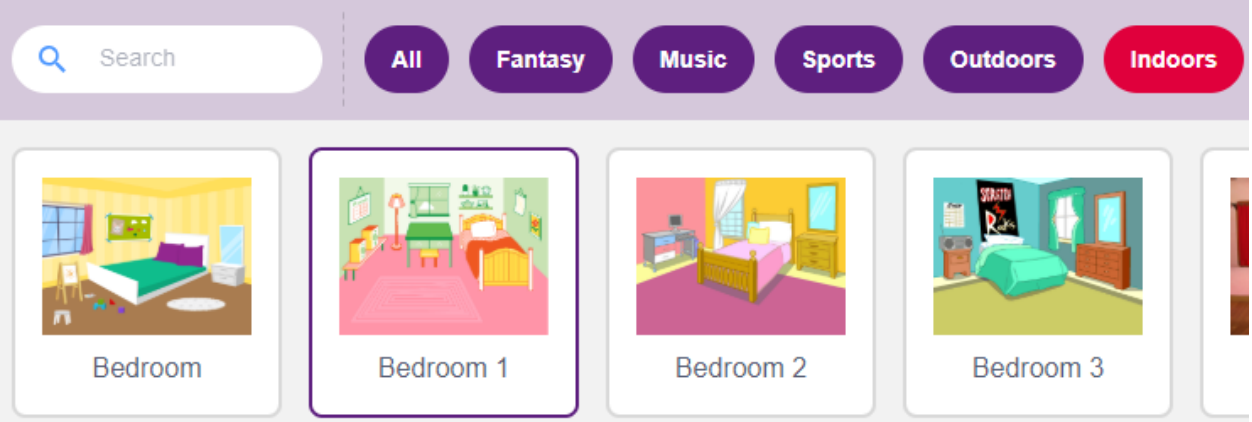
4.8.4 Programmierung

1. Hintergrund hinzufügen

Klicke auf den **Choose a Backdrop**-Button in der unteren rechten Ecke.

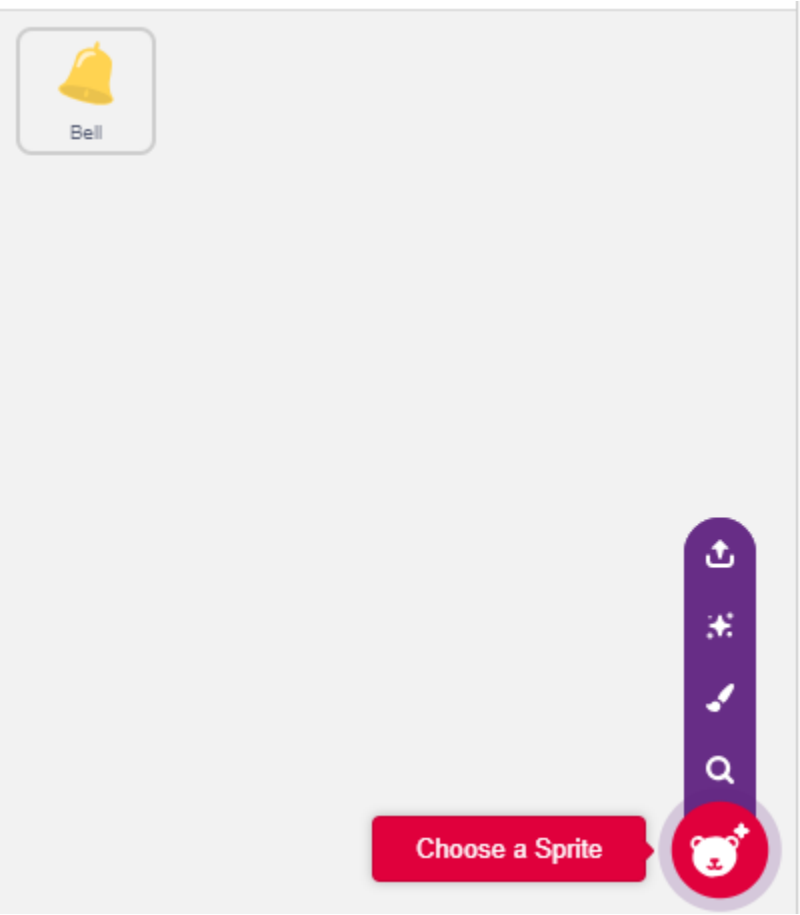


Wähle **Bedroom 1**.

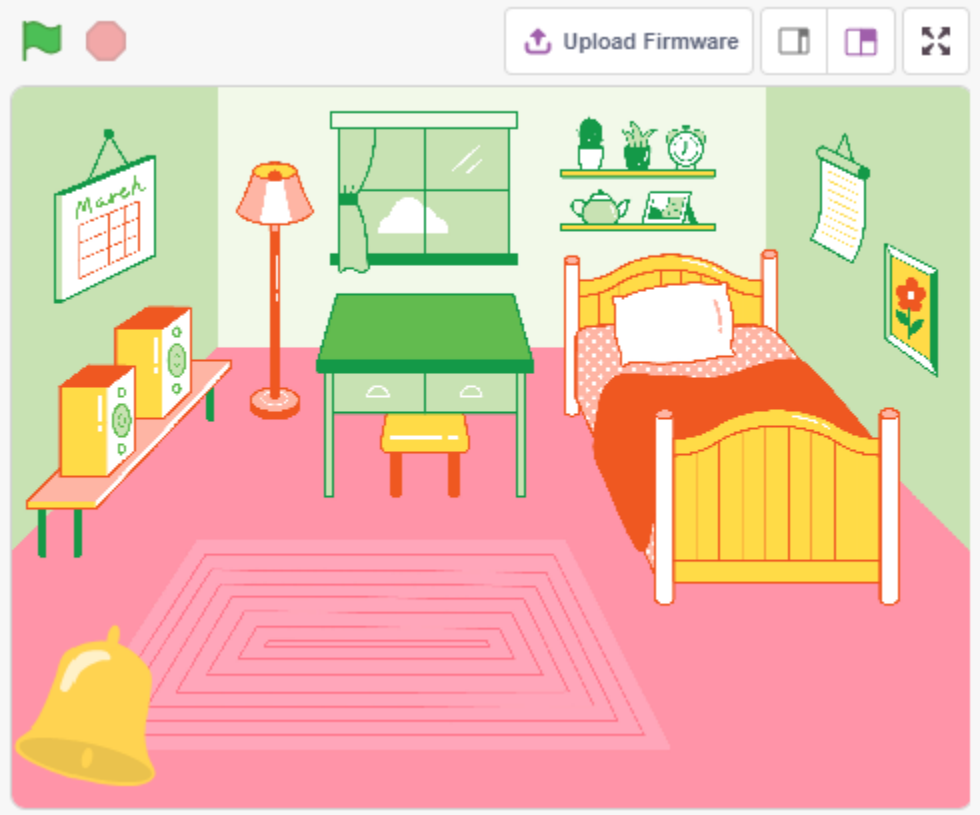


2. Wähle ein Sprite aus

Lösche das Standard-Sprite, klicke auf den **Choose a Sprite**-Button in der unteren rechten Ecke des Sprite-Bereichs, gib **bell** in das Suchfeld ein und klicke dann darauf, um es hinzuzufügen.



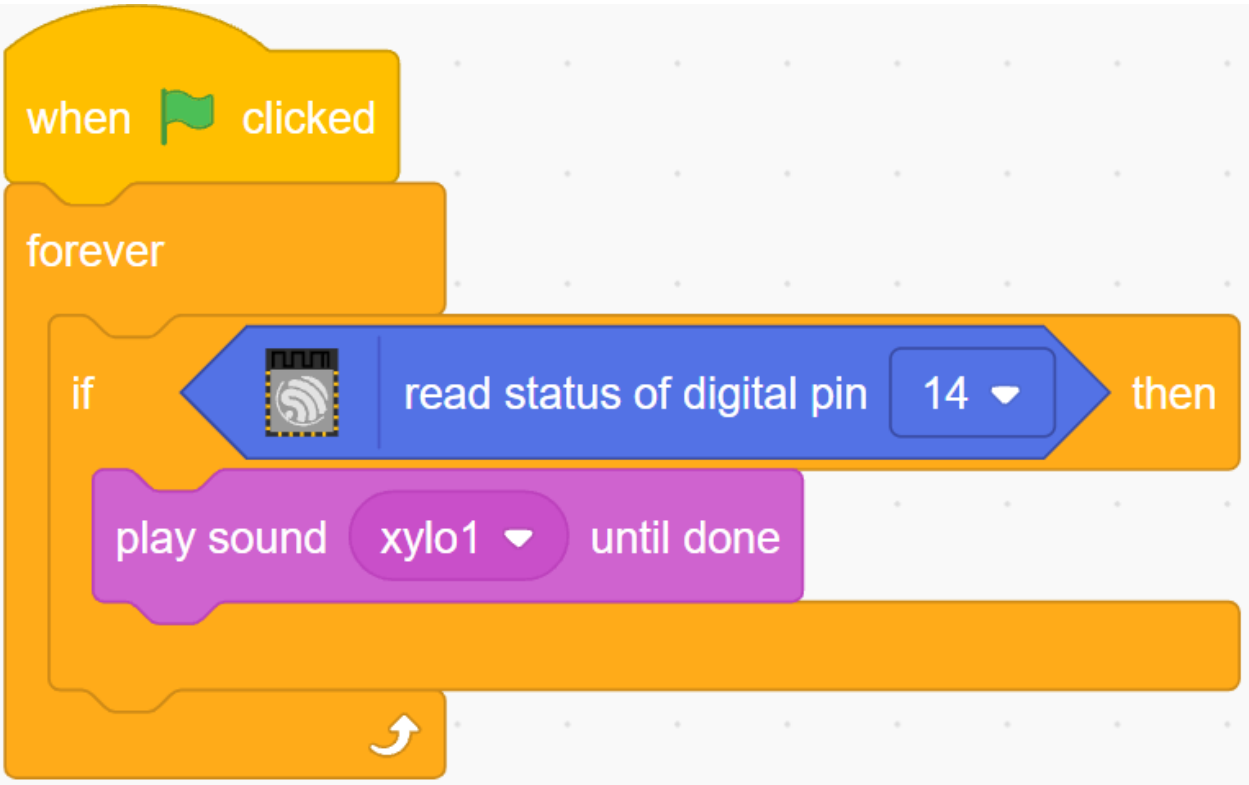
Wähle dann das **bell**-Sprite auf der Bühne aus und verschiebe es an die richtige Position.



3. Drücke den Knopf und die Glocke macht einen Ton

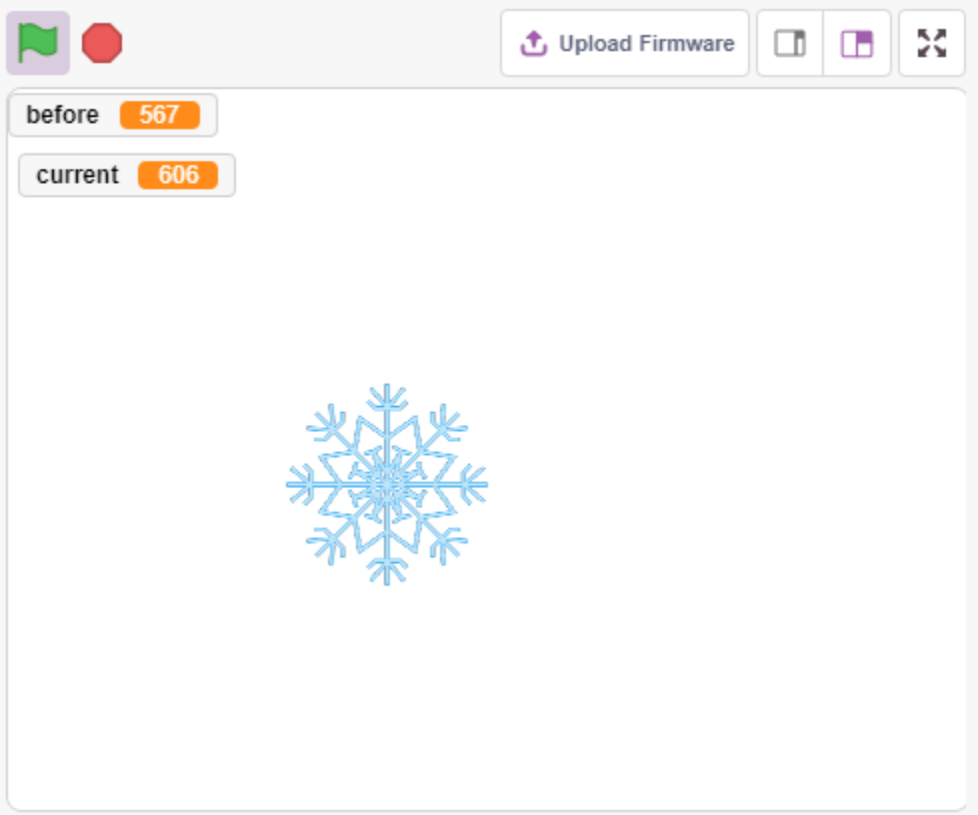
Verwende [if then] um eine bedingte Aussage zu machen, dass wenn der Wert von Pin14 gelesen gleich 1 ist (der Knopf ist gedrückt), wird der Ton **xylo1** gespielt.

- [read status of digital pin]: Dieser Block stammt aus der **ESP32**-Palette und wird verwendet, um den Wert eines digitalen Pins zu lesen, das Ergebnis ist 0 oder 1.
- [if then]: Dieser Block ist ein Steuerungsblock und stammt aus der **Control**-Palette. Wenn seine boolesche Bedingung wahr ist, werden die darin enthaltenen Blöcke ausgeführt und das beteiligte Skript wird fortgesetzt. Ist die Bedingung falsch, werden die Skripte im Block ignoriert. Die Bedingung wird nur einmal geprüft; wenn die Bedingung während des Laufens des Skripts im Block auf falsch wechselt, wird es weiterlaufen, bis es beendet ist.
- [play sound until done]: Dieser Block stammt aus der Ton-Palette und wird verwendet, um bestimmte Töne abzuspielen.



4.9 2.6 Niedrigtemperaturalarm

In diesem Projekt werden wir ein Niedrigtemperaturalarmsystem erstellen, bei dem das **Snowflake**-Sprite auf der Bühne erscheint, wenn die Temperatur unter den Schwellenwert fällt.



4.9.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Thermistor	

4.9.2 Was Sie Lernen Werden

- Funktionsprinzip des Thermistors
- Multivariable und subtraktive Operationen

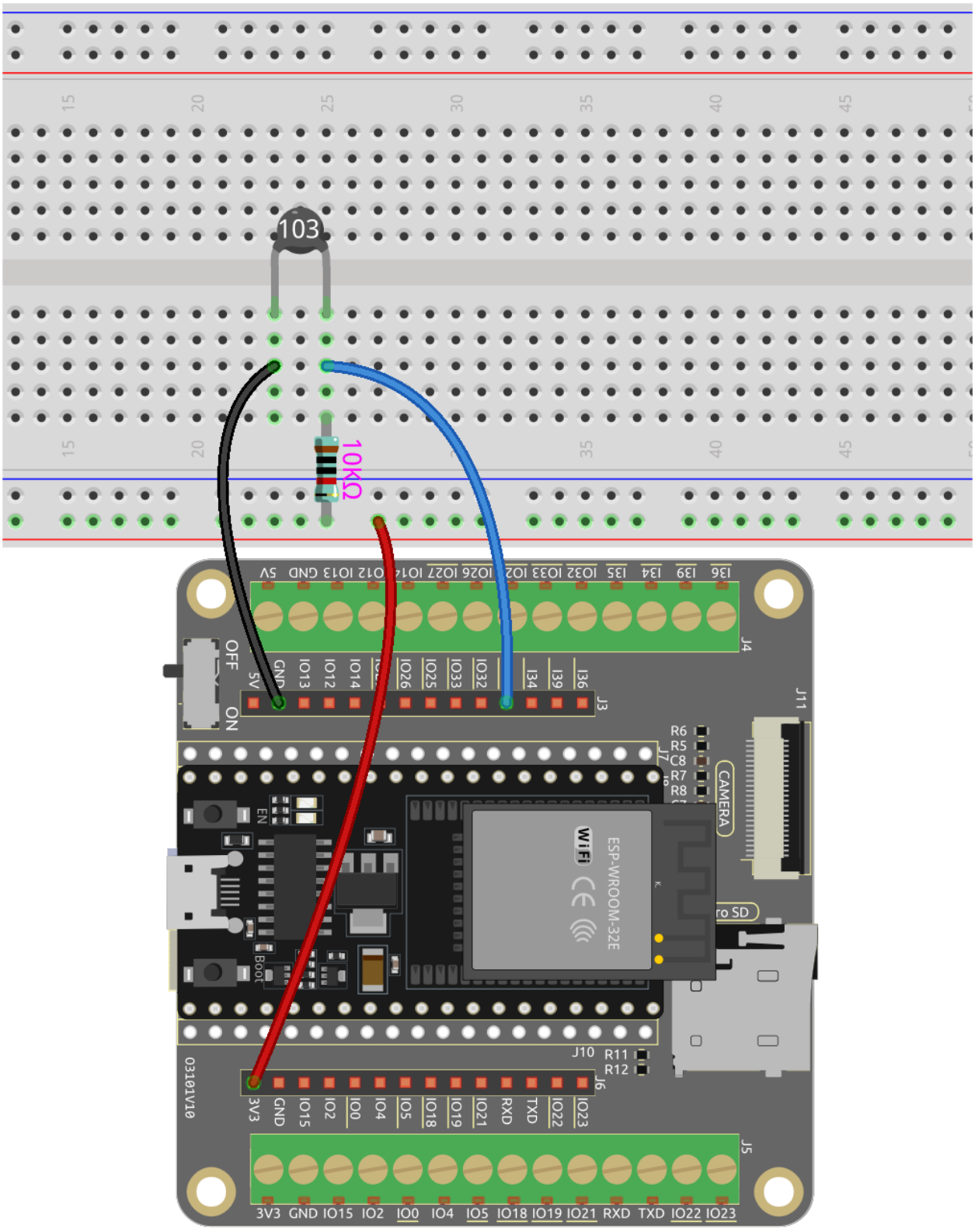
4.9.3 Schaltung Aufbauen

Ein Thermistor ist ein Typ von Widerstand, dessen Widerstand stark temperaturabhängig ist, stärker als bei Standardwiderständen, und es gibt zwei Arten von Widerständen, PTC (Widerstand erhöht sich mit steigender Temperatur) und PTC (Widerstand verringert sich mit steigender Temperatur).

Bauen Sie die Schaltung gemäß dem folgenden Diagramm auf.

Ein Ende des Thermistors ist mit GND verbunden, das andere Ende mit Pin35, und ein 10K-Widerstand ist in Serie zu 5V geschaltet.

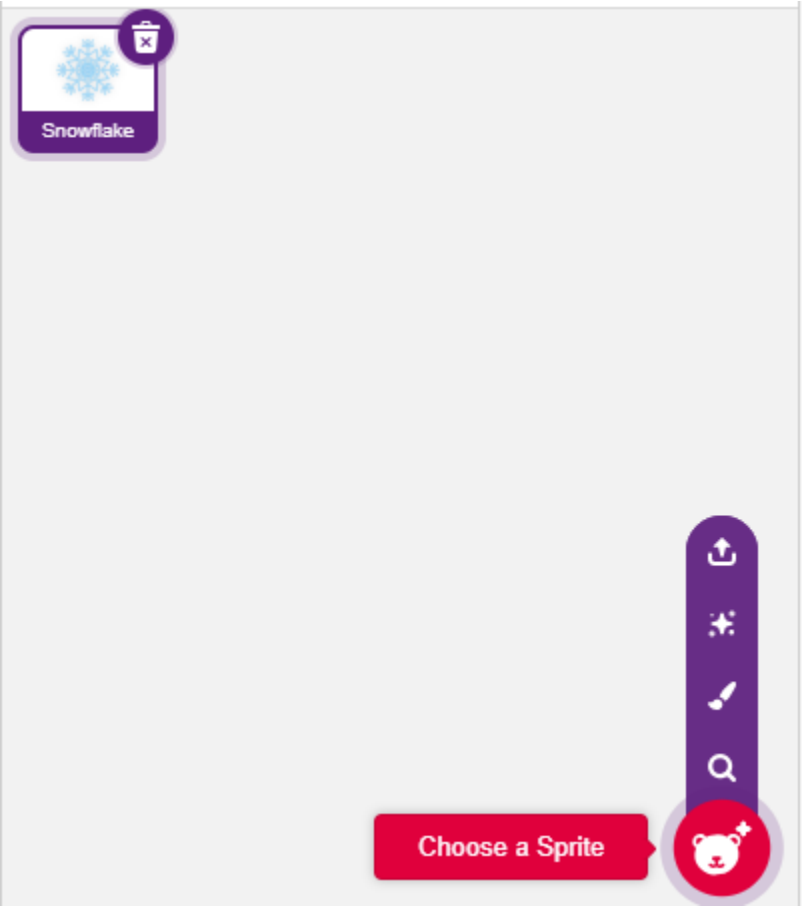
Hier wird der NTC-Thermistor verwendet, so dass, wenn die Temperatur steigt, der Widerstand des Thermistors abnimmt, die Spannungsteilung an Pin35 abnimmt und der von Pin35 erhaltene Wert sinkt und umgekehrt steigt.


103
10KΩ
OFF
ON
ESP-WROOM-32E
CAMERA
J3
J4
J5
J6
J10
J11
R11
R12
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

4.9.4 Programmierung

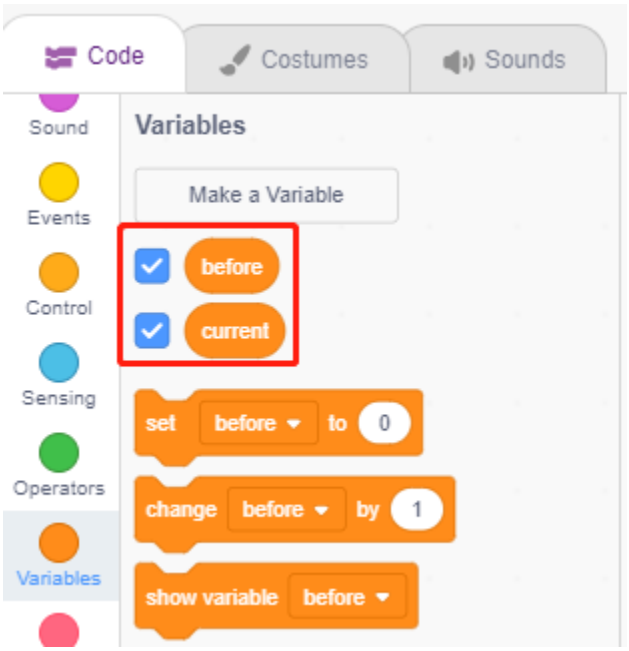
1. Wähle ein Sprite aus

Lösche das Standard-Sprite, klicke auf den **Choose a Sprite**-Button in der unteren rechten Ecke des Sprite-Bereichs, gib **Snowflake** in das Suchfeld ein und klicke dann darauf, um es hinzuzufügen.



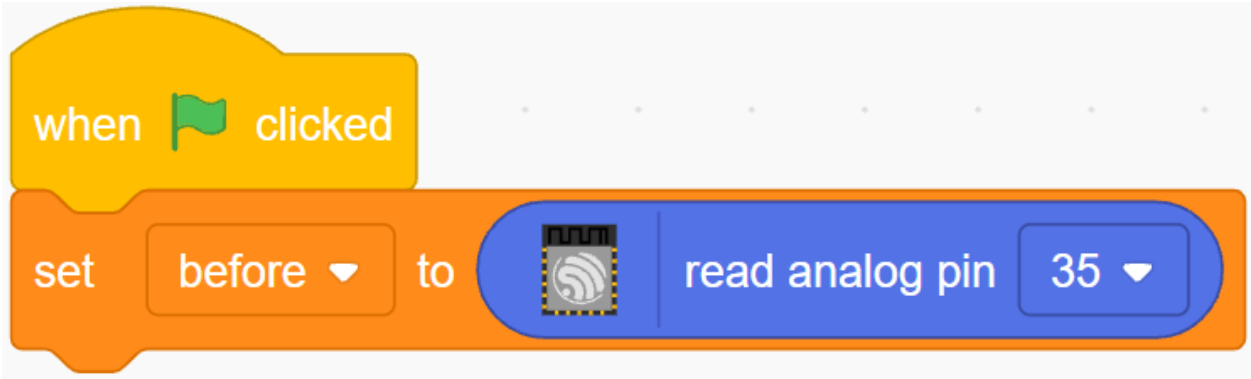
2. Erstelle 2 Variablen

Erstelle zwei Variablen, **before** und **current**, um den Wert von Pin35 in unterschiedlichen Fällen zu speichern.



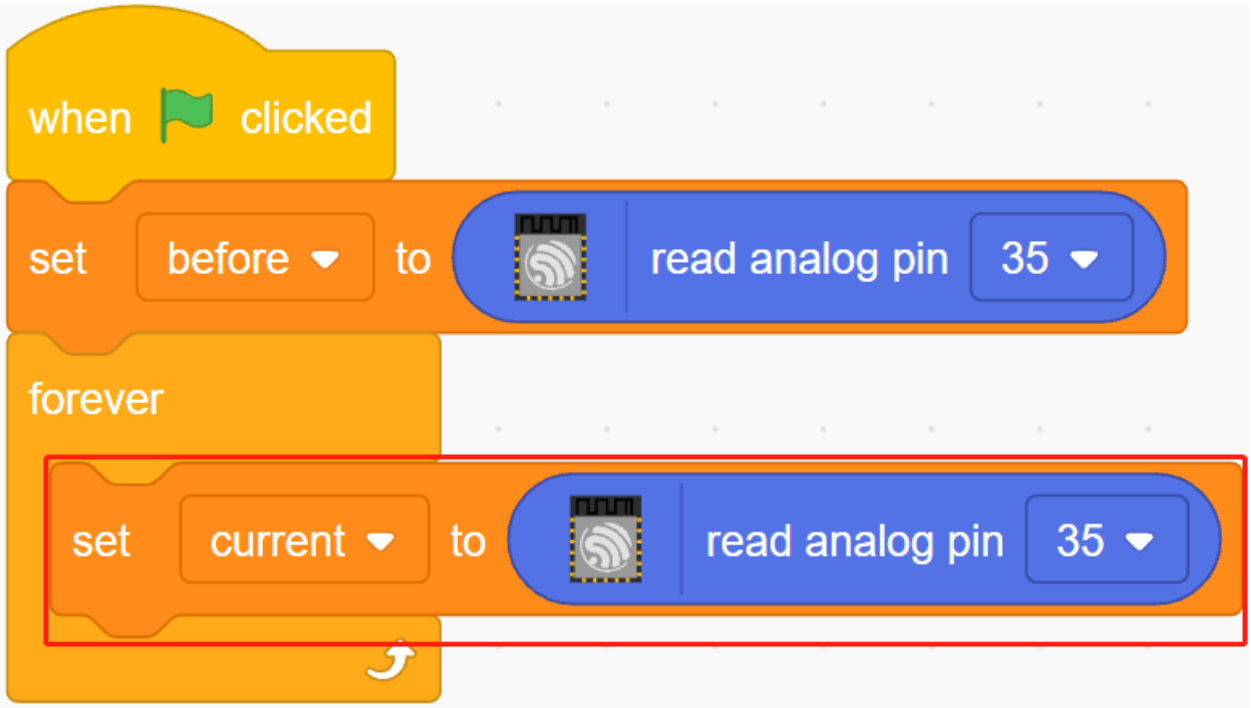
3. Lies den Wert von Pin35

Wenn die grüne Flagge angeklickt wird, wird der Wert von Pin35 gelesen und in der Variablen **before** gespeichert.



4. Lies den Wert von Pin35 erneut

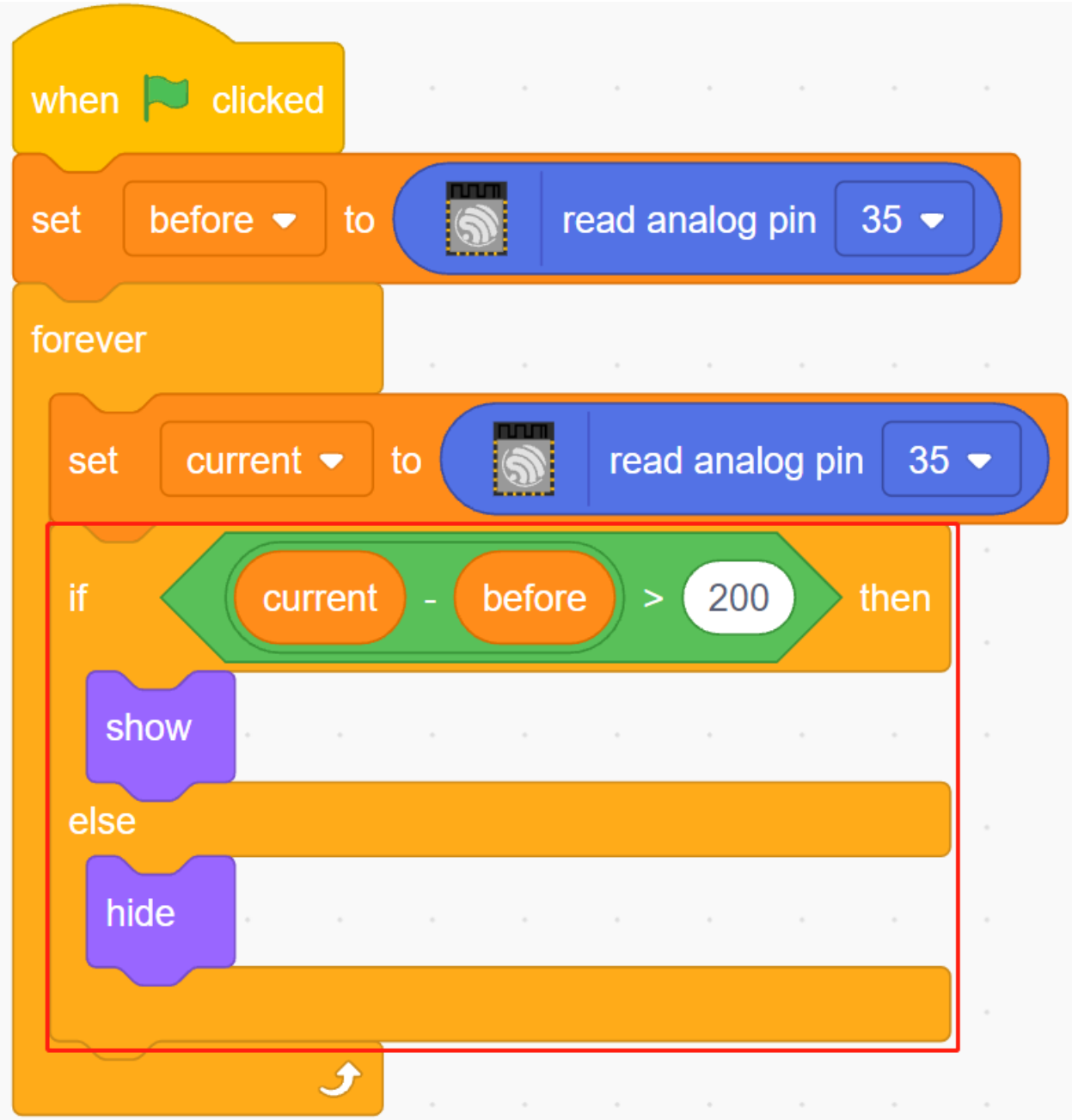
In [forever], lese den Wert von Pin35 erneut und speichere ihn in der Variablen **current**.



5. Bestimmung von Temperaturänderungen

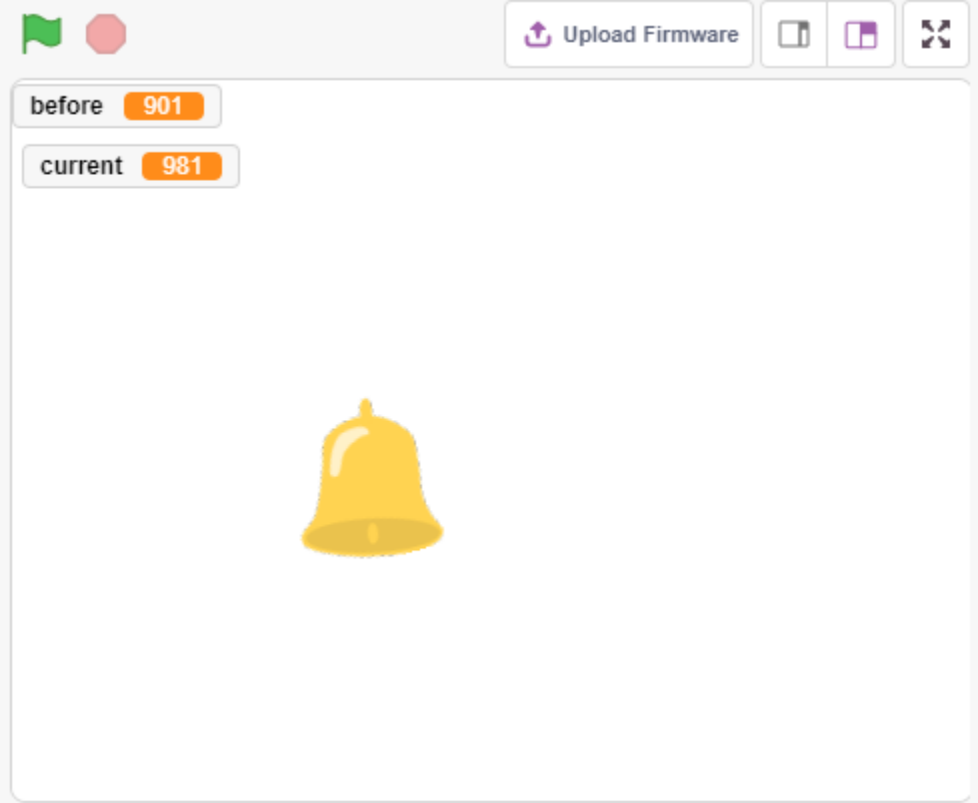
Verwende den [if else]-Block, um festzustellen, ob der aktuelle Wert von Pin35 um 200 größer als zuvor ist, was einen Temperaturrückgang darstellt. In diesem Fall lasse das **Snowflake**-Sprite erscheinen, andernfalls verstecke es.

- [-] & [>]: Subtraktions- und Vergleichsoperatoren aus der **Operators**-Palette.


when clicked
set before to read analog pin 35
forever
set current to read analog pin 35
if current - before > 200 then
show
else
hide

4.10 2.7 Lichtwecker

Im Alltag gibt es verschiedene Arten von Zeitweckern. Jetzt wollen wir einen lichtgesteuerten Wecker bauen. Wenn der Morgen kommt und die Helligkeit des Lichts zunimmt, wird dieser lichtgesteuerte Wecker Sie daran erinnern, dass es Zeit ist, aufzustehen.



4.10.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Fotowiderstand	

4.10.2 Was Sie Lernen Werden

- Funktionsprinzip des Fotowiderstands
- Beenden der Tonwiedergabe und Beenden der Ausführung von Skripten

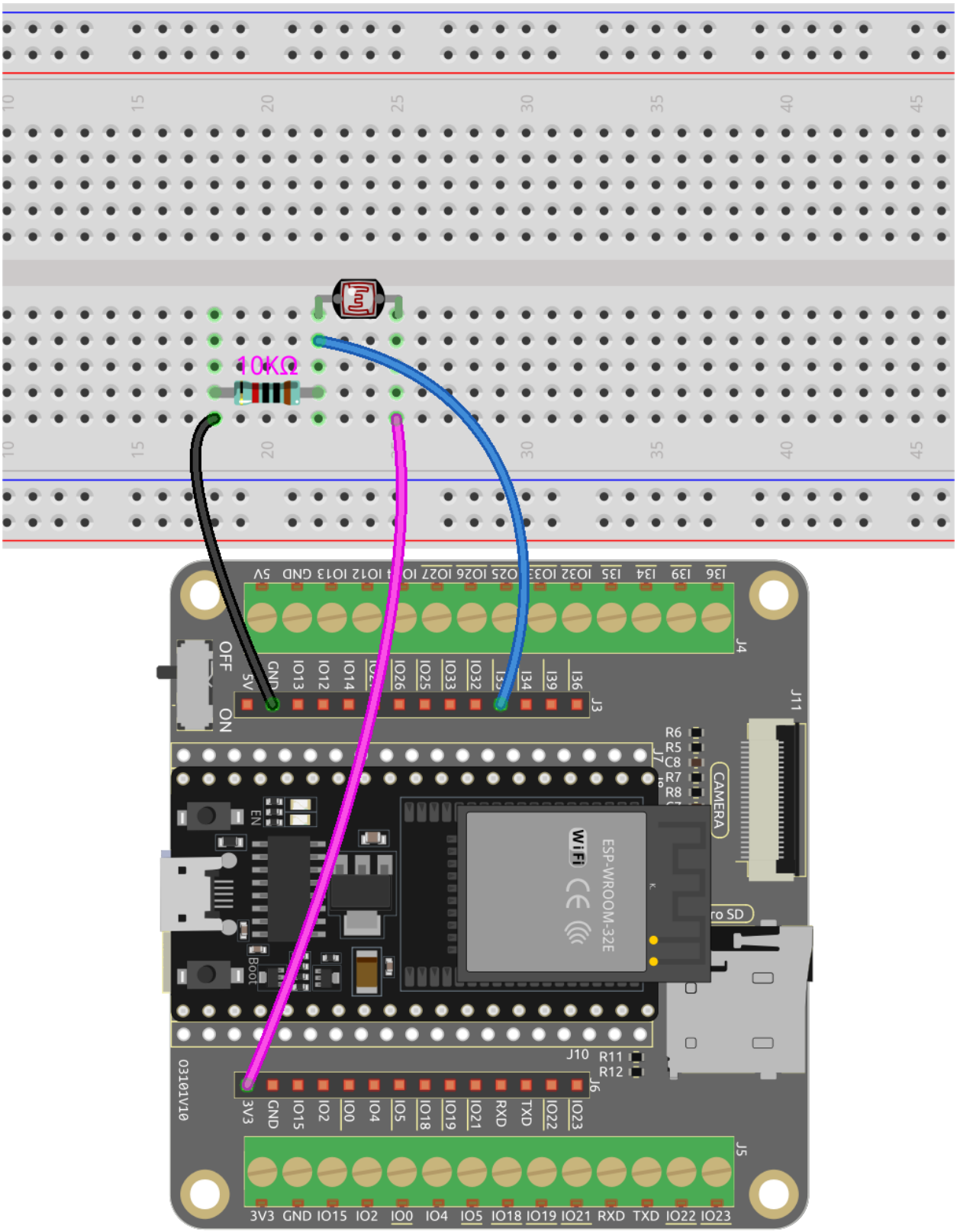
4.10.3 Schaltung Aufbauen

Ein Fotowiderstand oder eine Fotodiode ist ein lichtgesteuerter variabler Widerstand. Der Widerstand eines Fotowiderstands nimmt mit zunehmender einfallender Lichtintensität ab.

Bauen Sie die Schaltung gemäß dem folgenden Diagramm auf.

Verbinden Sie ein Ende des Fotowiderstands mit 5V, das andere Ende mit Pin35 und schalten Sie einen 10K-Widerstand in Serie mit GND an diesem Ende.

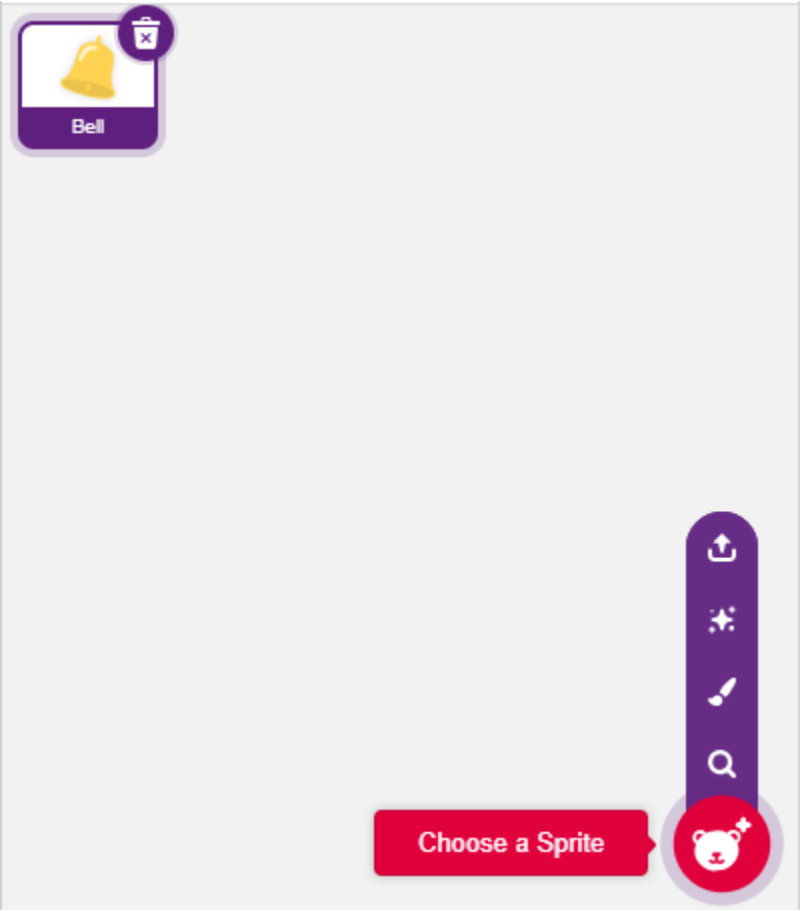
Wenn also die Lichtintensität zunimmt, verringert sich der Widerstand des Fotowiderstands, die Spannungsteilung des 10K-Widerstands nimmt zu, und der von Pin35 erhaltene Wert wird größer.


10KΩ
5V GND IO13 IO12 IO14 IO27 IO26 IO25 IO33 IO32 I35 I34 I39 I36
J4
OFF
ON
5V GND IO13 IO12 IO14 IO26 IO25 IO33 IO32 I35 I34 I39 I36
J3
J11
R6
R5
C8
R7
R8
CAMERA
EN
Boot
WiFi
ESP-WROOM-32E
J10
R11
R12
J6
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23
J5

4.10.4 Programmierung

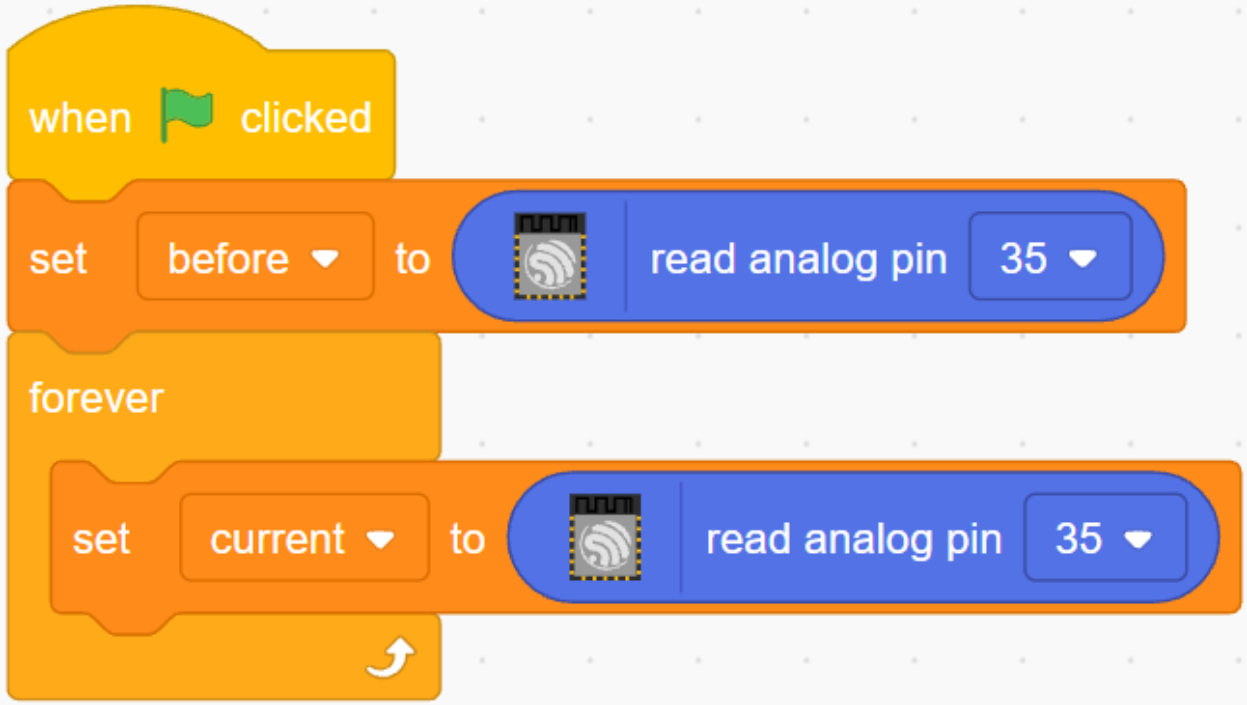
1. Wähle ein Sprite aus

Lösche das Standard-Sprite, klicke auf den **Choose a Sprite**-Button in der unteren rechten Ecke des Sprite-Bereichs, gib **bell** in das Suchfeld ein und klicke dann darauf, um es hinzuzufügen.



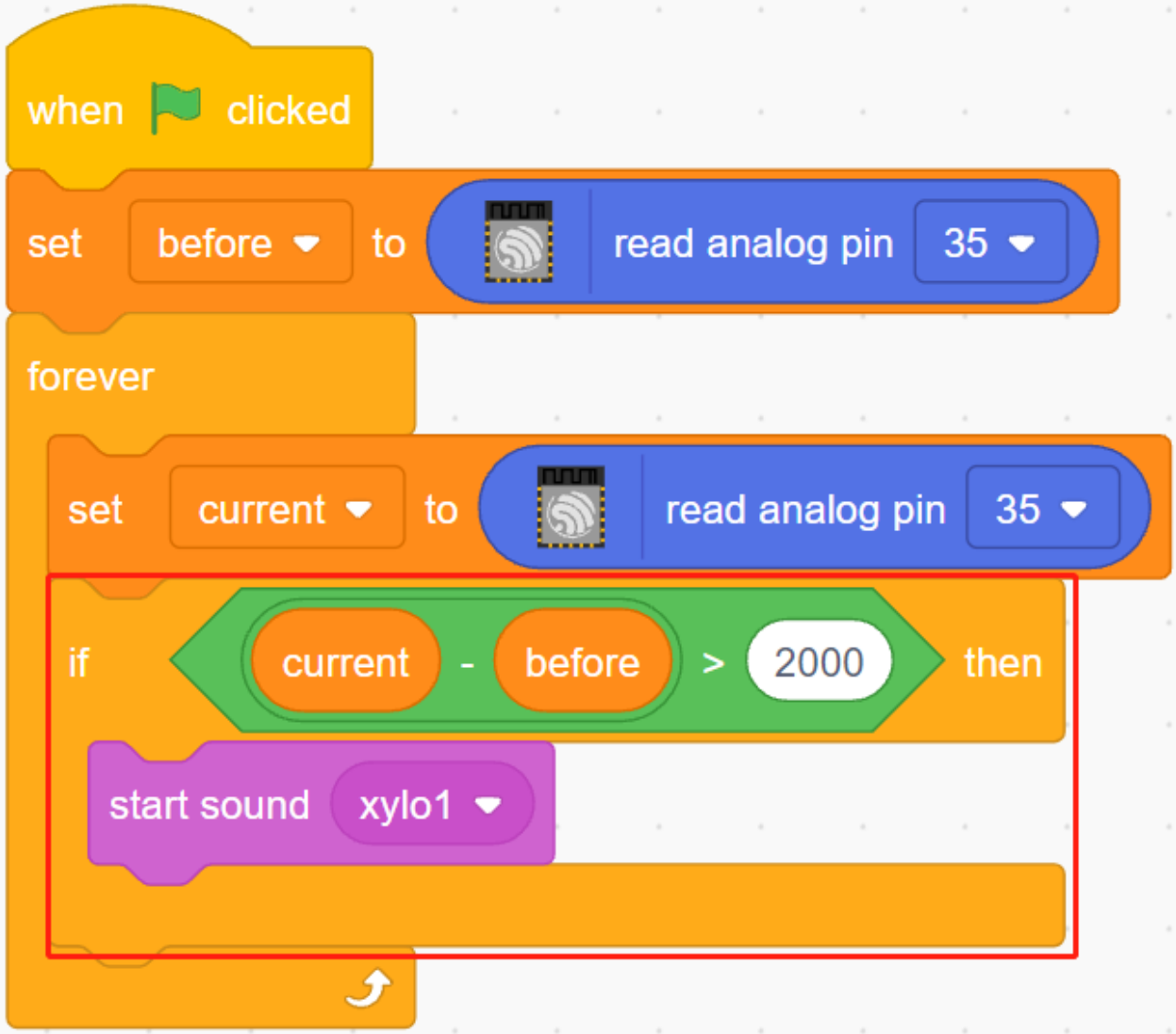
2. Lies den Wert von Pin35

Erstelle zwei Variablen **before** und **current**. Wenn die grüne Flagge angeklickt wird, lese den Wert von Pin35 und speichere ihn in der Variablen **before** als Referenzwert. In [forever], lese den Wert von Pin35 erneut, speichere ihn in der Variablen **current**.



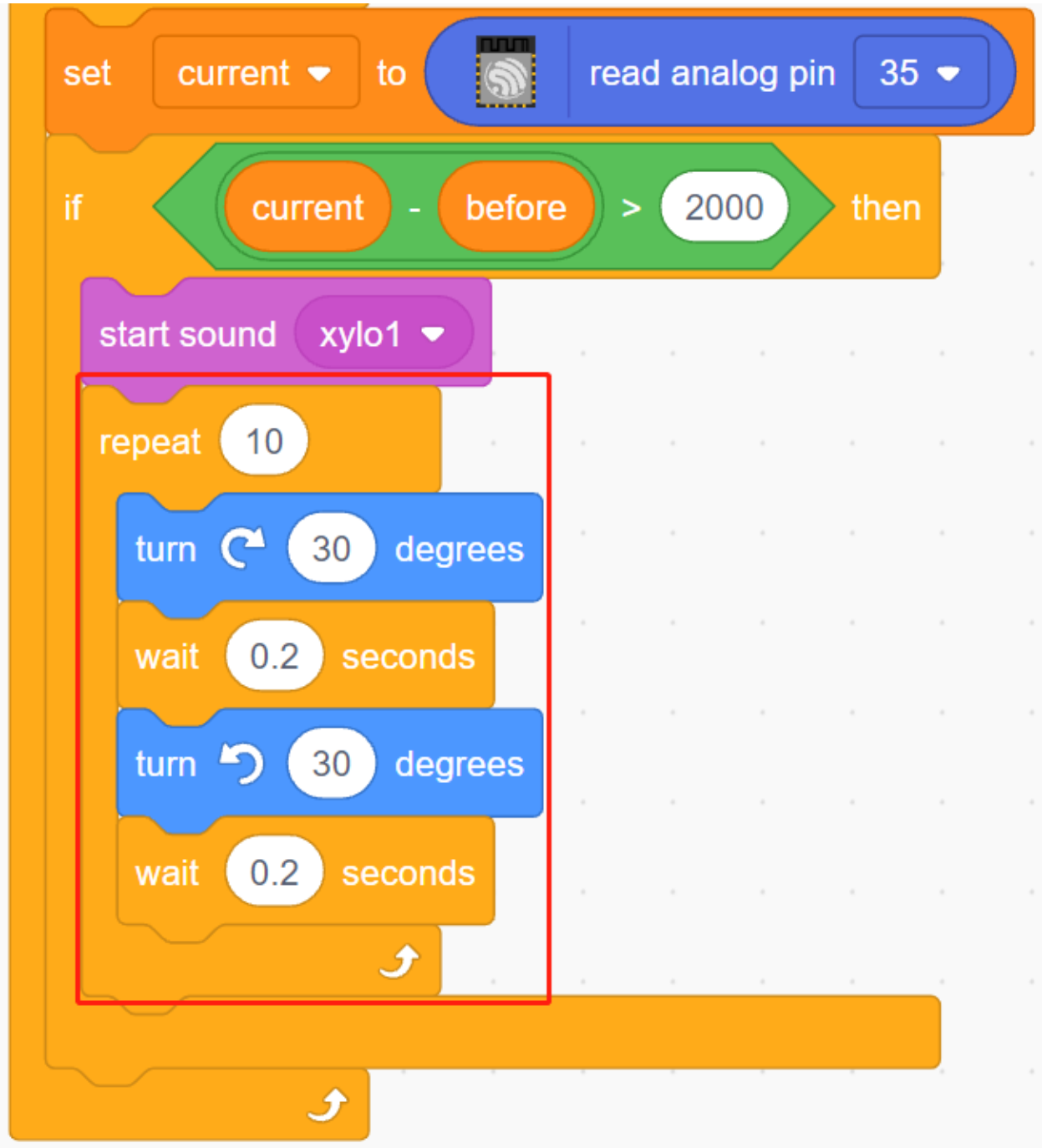
3. Einen Ton machen

Wenn der Wert des aktuellen Pin35 um mehr als 50 über dem vorherigen liegt, was bedeutet, dass die aktuelle Lichtintensität größer als der Schwellenwert ist, dann lasse das Sprite einen Ton machen.



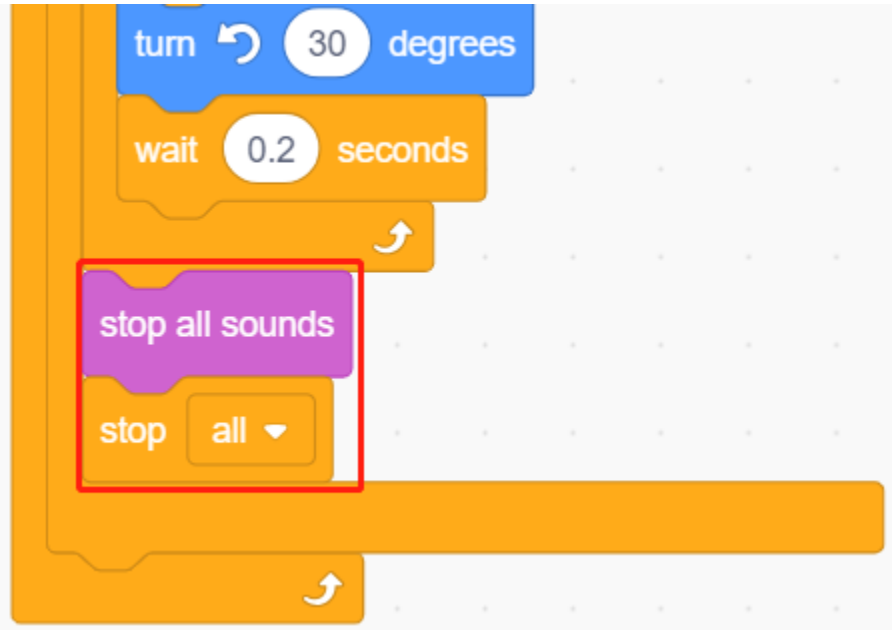
4. Das Sprite drehen

Verwende [turn block], um das **bell**-Sprite nach links und rechts zu drehen, um den Weckeffekt zu erzielen.



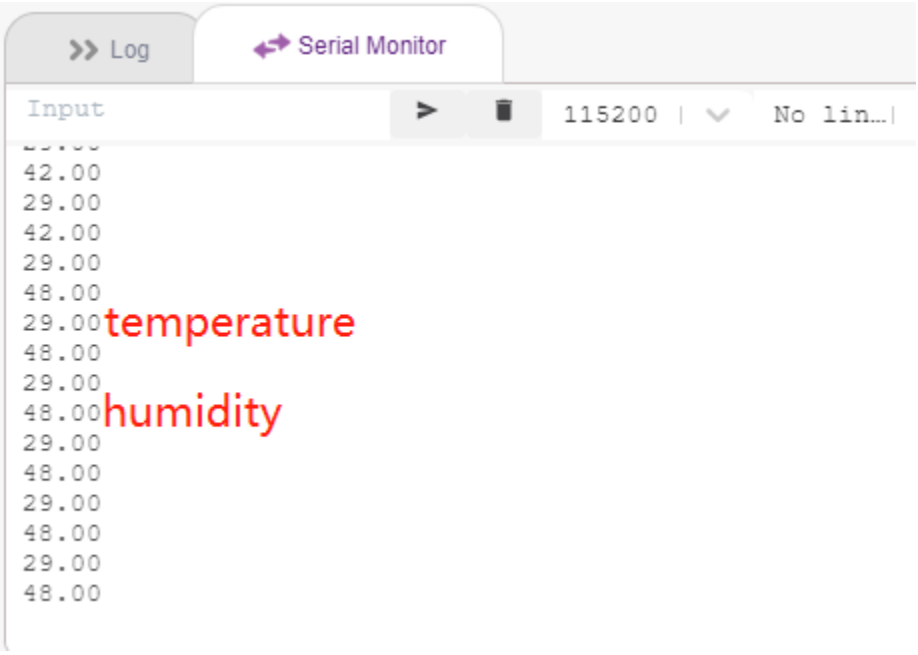
5. Alles stoppen

Stoppt den Alarm, wenn er eine Weile geklingelt hat.



4.11 2.8 Temperatur und Luftfeuchtigkeit lesen

In vorherigen Projekten haben wir den Bühnenmodus verwendet, aber einige Funktionen sind nur im Upload-Modus verfügbar, wie zum Beispiel die serielle Kommunikationsfunktion. In diesem Projekt werden wir die Temperatur und Luftfeuchtigkeit des DHT11 über den Serial Monitor im *Upload-Modus* ausgeben.



4.11.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
DHT11 Feuchtigkeits- und Temperatursensor	

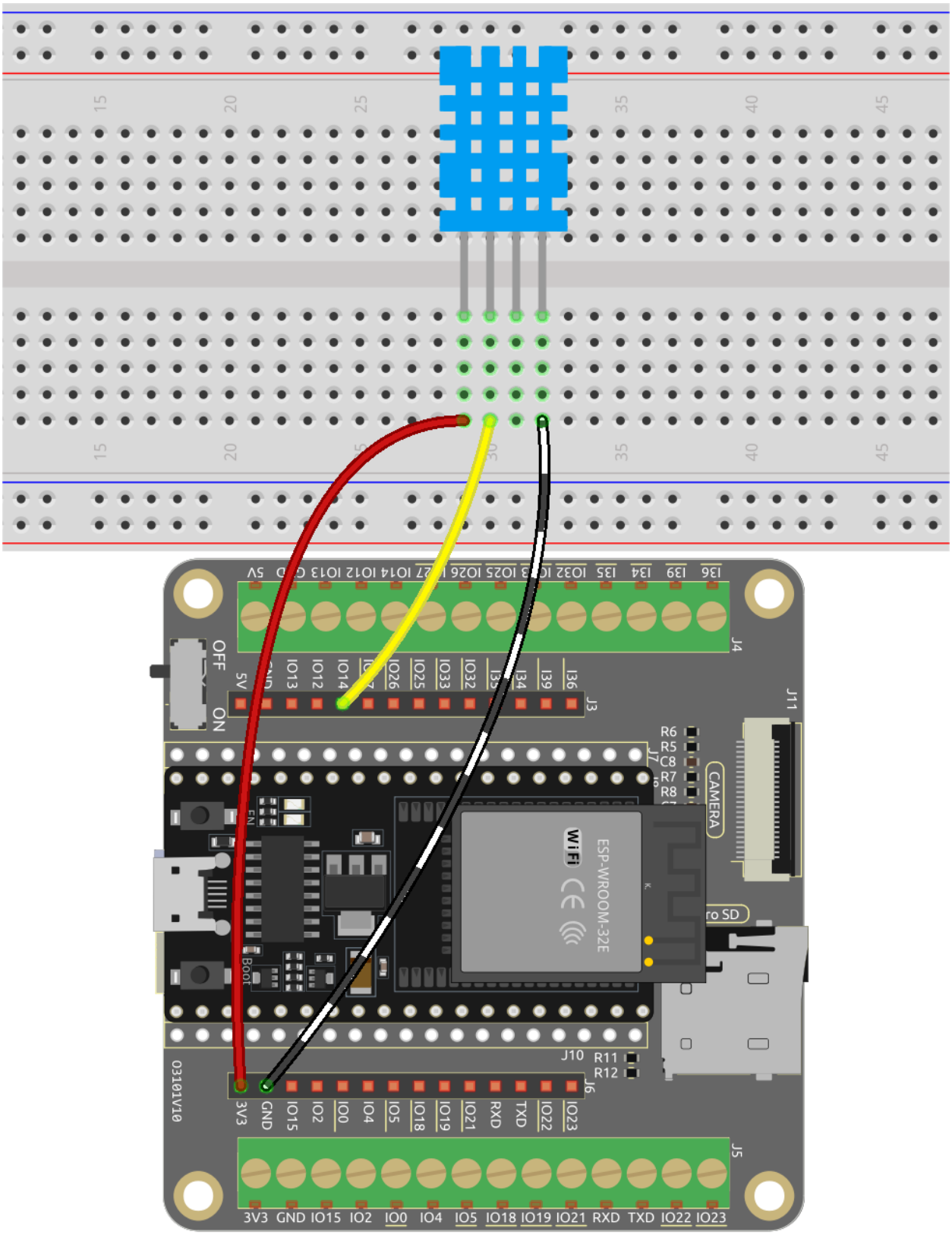
4.11.2 Was Sie Lernen Werden

- Temperatur und Luftfeuchtigkeit vom DHT11-Modul erhalten
- Serial Monitor für *Upload-Modus*
- Erweiterung hinzufügen

4.11.3 Schaltung Aufbauen

Der digitale Temperatur- und Feuchtigkeitssensor DHT11 ist ein zusammengesetzter Sensor, der einen kalibrierten digitalen Signalausgang von Temperatur und Feuchtigkeit enthält.

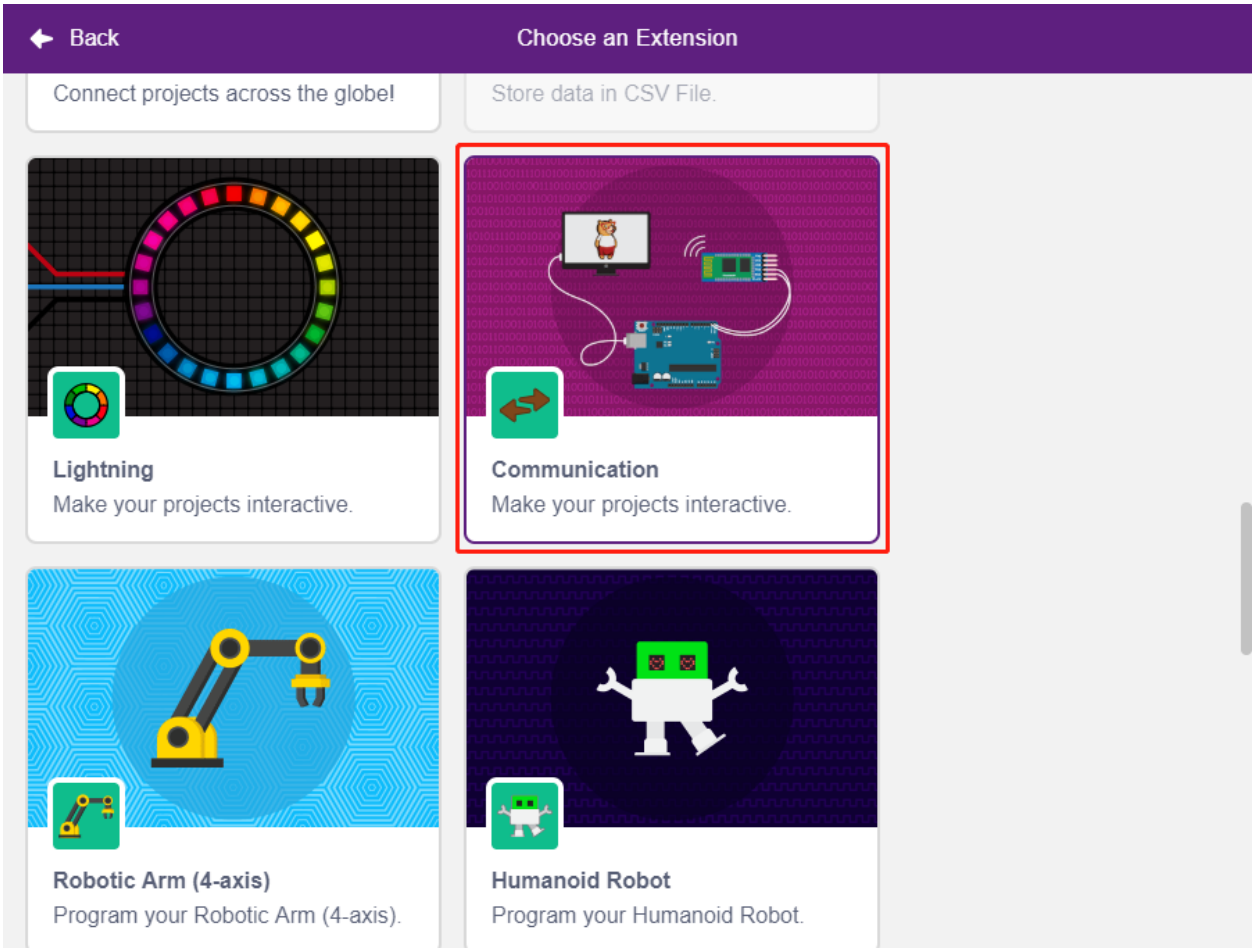
Bauen Sie die Schaltung gemäß dem folgenden Diagramm auf.


OFF
ON
J4
J3
J11
J10
J6
J5
R6
R5
C8
R7
R8
R11
R12
CAMERA
ESP-WROOM-32E
Boot
EN
03101V10
5V
GND
IO13
IO12
IO14
IO26
IO25
IO33
IO32
IO35
IO34
IO39
IO36
3V3
GND
IO15
IO2
IO0
IO4
IO5
IO18
IO19
IO21
RXD
TXD
IO22
IO23

4.11.4 Programmierung

1. Erweiterungen hinzufügen

Wechseln Sie in den **Upload**-Modus, klicken Sie auf den Button **Add Extension** in der unteren linken Ecke, wählen Sie dann **Communication** aus, um es hinzuzufügen, und es wird am Ende des Palettenbereichs erscheinen.



2. Initialisierung des ESP32 und des Serial Monitors

Im **Upload**-Modus starten Sie den ESP32 und setzen dann die Baudrate des seriellen Ports.

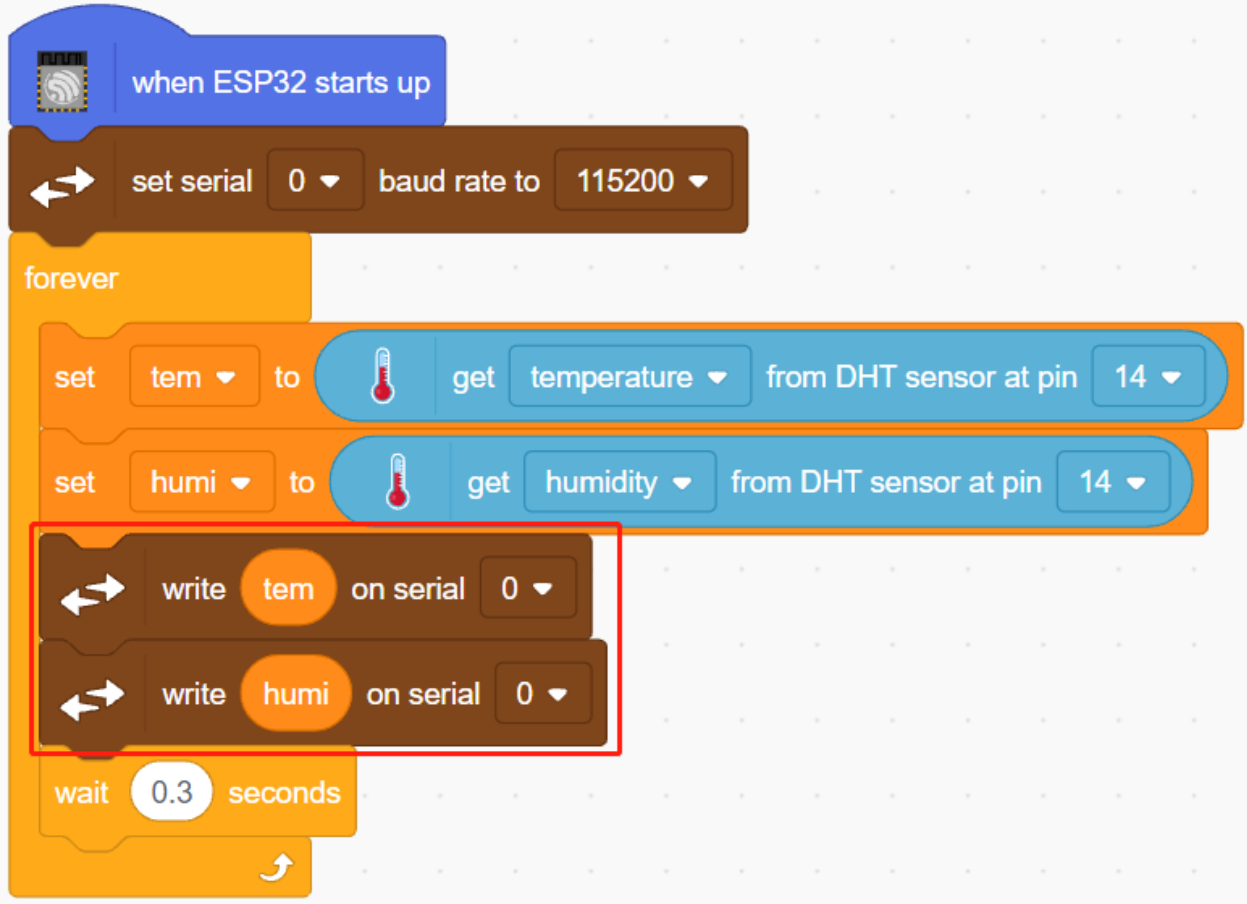
- [when ESP32 Starts up]: Im **Upload**-Modus starten Sie den ESP32.
- [set serial baud rate to]: Aus der **Communications**-Palette, verwendet um die Baudrate von seriellen Port 0 zu setzen, Standard ist 115200. Wenn Sie Mega2560 verwenden, können Sie wählen, die Baudrate in seriellen Port 0~2 zu setzen.

3. Temperatur und Luftfeuchtigkeit lesen

Erstellen Sie 2 Variablen **tem** und **humi** um die Temperatur und Luftfeuchtigkeit jeweils zu speichern, der Code erscheint auf der rechten Seite, während Sie den Block ziehen und ablegen.

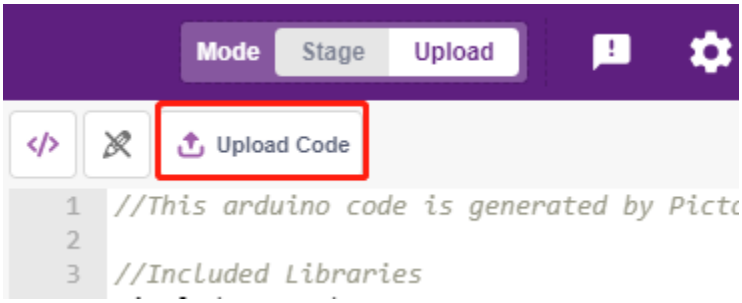
4. Auf dem Serial Monitor ausgeben

Schreiben Sie die gelesene Temperatur und Luftfeuchtigkeit auf den Serial Monitor. Um zu vermeiden, dass zu schnell übertragen wird und PictoBlox ins Stocken gerät, verwenden Sie den [wait seconds]-Block, um etwas Zeitintervall für die nächste Ausgabe hinzuzufügen.



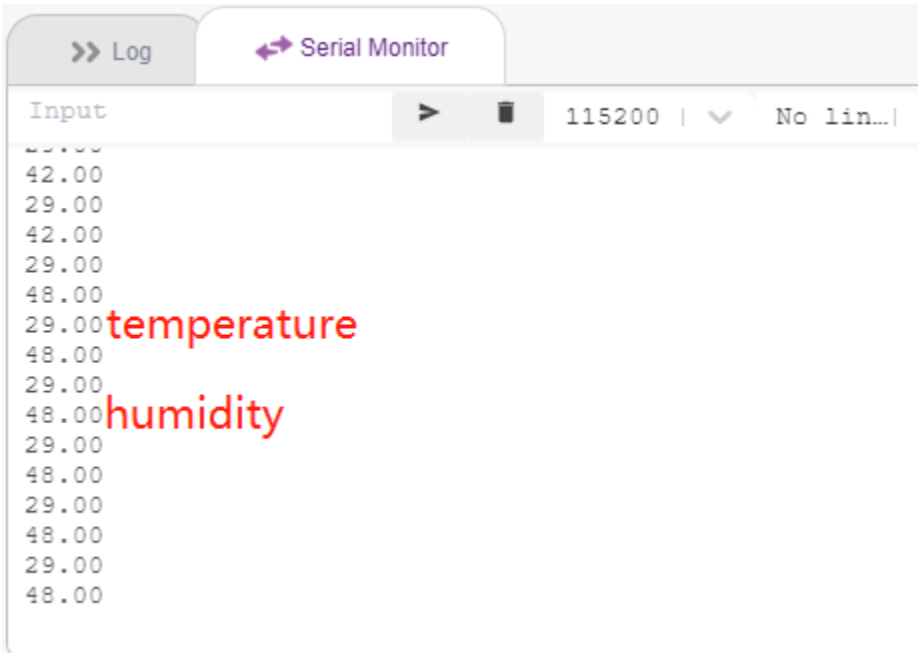
5. Code hochladen

Im Gegensatz zum **Stage**-Modus muss der Code im **Upload**-Modus auf das ESP32-Board hochgeladen werden, um den Effekt zu sehen, indem Sie den Button **Upload Code** verwenden. So können Sie auch das USB-Kabel abziehen und das Programm weiterlaufen lassen.



6. Serial Monitor öffnen

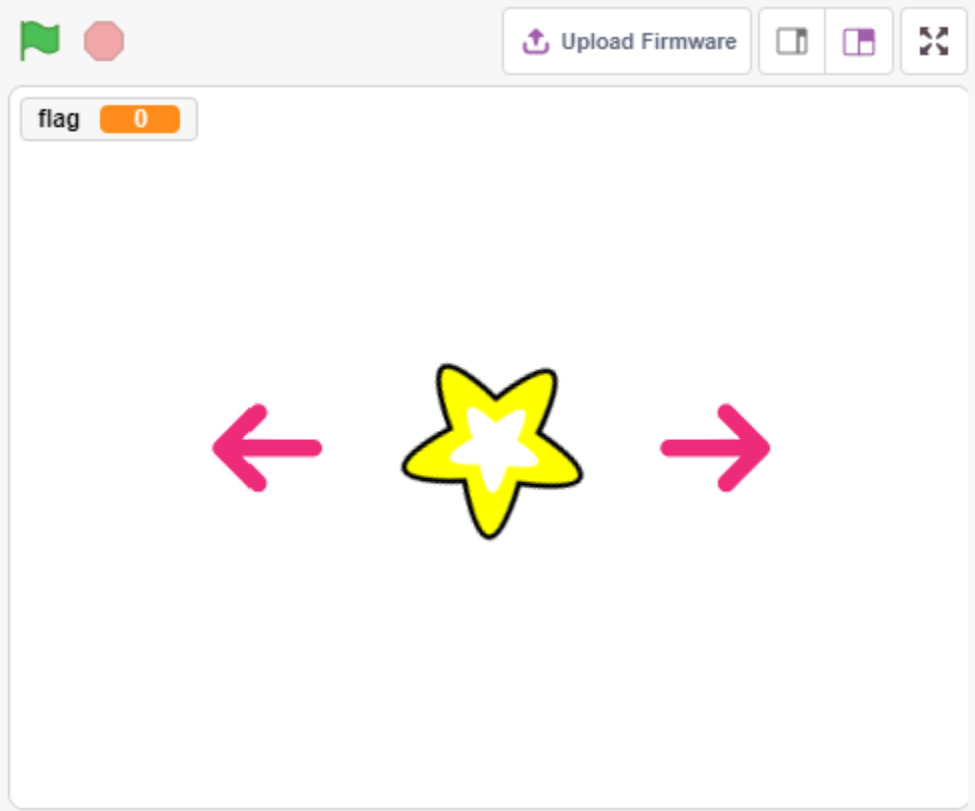
Öffnen Sie jetzt den **Serial Monitor**, um die Temperatur und Luftfeuchtigkeit zu sehen.



4.12 2.9 Rotierender Ventilator

In diesem Projekt werden wir ein rotierendes Stern-Sprite und einen Ventilator bauen.

Durch Klicken auf die links- und rechtsweisenden Pfeil-Sprites auf der Bühne wird die Drehrichtung des Motors und des Stern-Sprites im Uhrzeigersinn und gegen den Uhrzeigersinn gesteuert. Ein Klick auf das Stern-Sprite stoppt die Drehung.



4.12.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

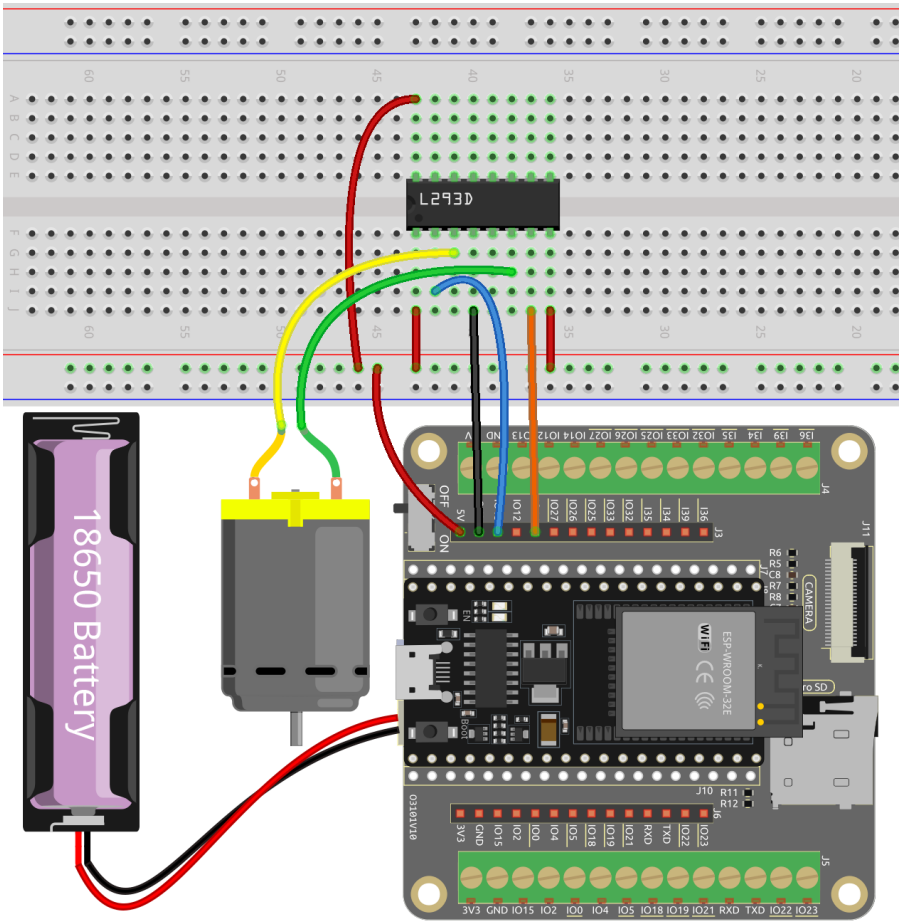
Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Gleichstrommotor	
L293D	-

4.12.2 Was Sie Lernen Werden

- Funktionsprinzip des Motors
- Broadcast-Funktion
- Block zum Stoppen anderer Skripte im Sprite

4.12.3 Schaltung Aufbauen

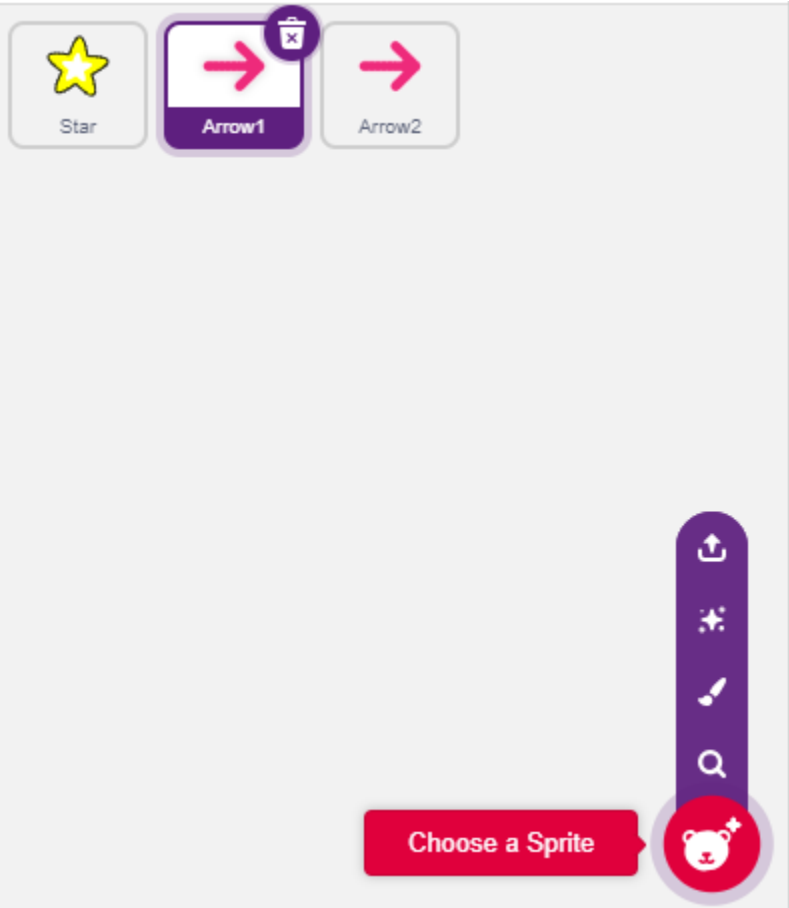


4.12.4 Programmierung

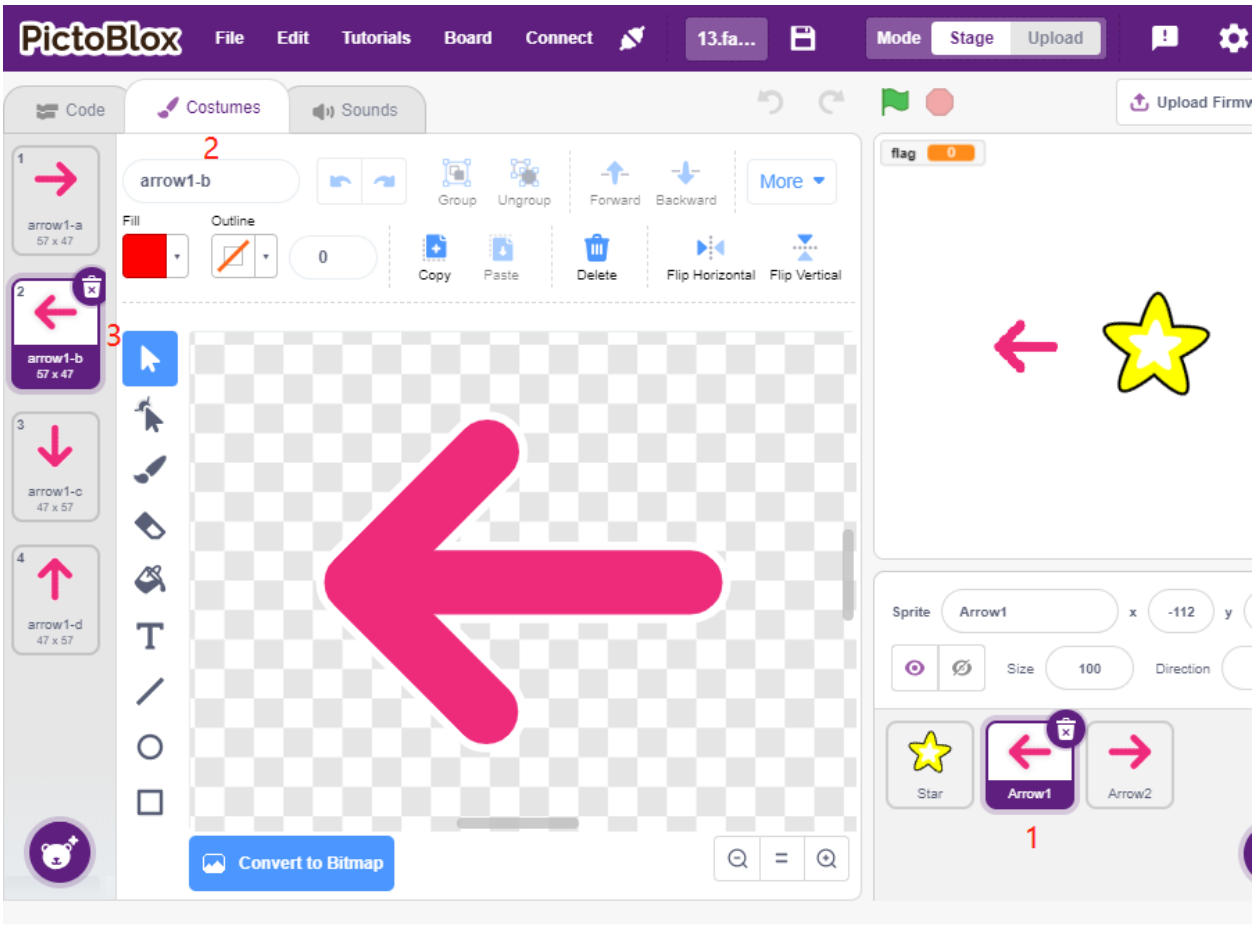
Das zu erreichende Ziel ist es, mit 2 Pfeil-Sprites die Drehung des Motors und des Stern-Sprites im Uhrzeigersinn und gegen den Uhrzeigersinn zu steuern. Ein Klick auf das Stern-Sprite stoppt die Drehung des Motors.

1. Sprites hinzufügen

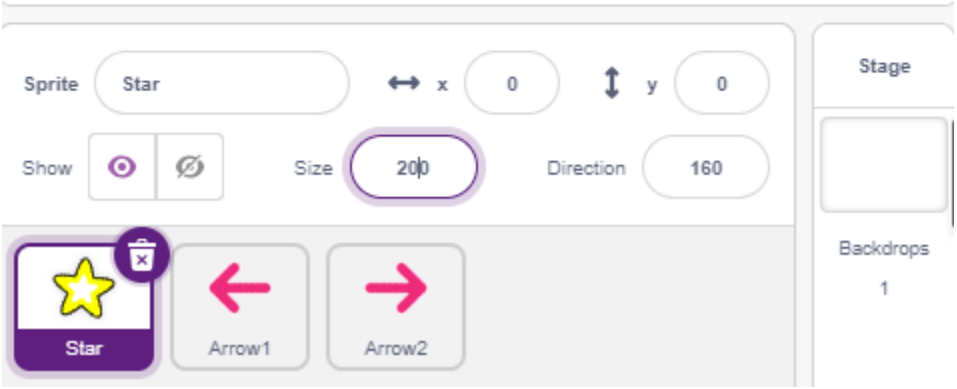
Lösche das Standard-Sprite, wähle dann das **Star**-Sprite und das **Arrow1**-Sprite aus und kopiere **Arrow1** einmal.



Unter der Option **Costumes** ändere das **Arrow1**-Sprite zu einem anderen Richtungs-Kostüm.



Passe die Größe und Position des Sprites entsprechend an.

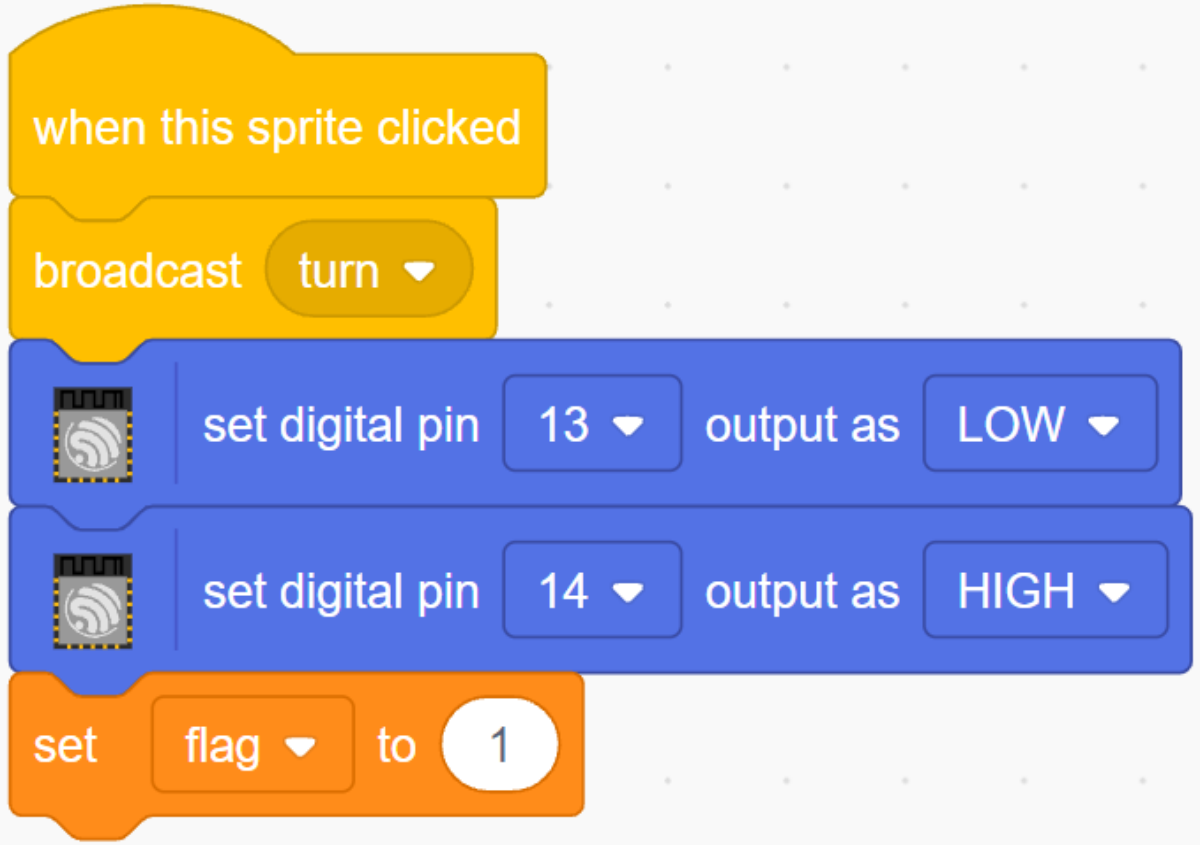


2. Linkes Pfeil-Sprite

Wenn dieses Sprite angeklickt wird, sendet es eine Nachricht - drehen, setzt dann digitalen Pin12 auf niedrig und Pin14 auf hoch und setzt die Variable **Flag** auf 1. Wenn du das linke Pfeil-Sprite anklickst, wirst du feststellen, dass sich der Motor gegen den Uhrzeigersinn dreht. Wenn sich dein Motor im Uhrzeigersinn dreht, dann tausche die Positionen von Pin12 und Pin14.

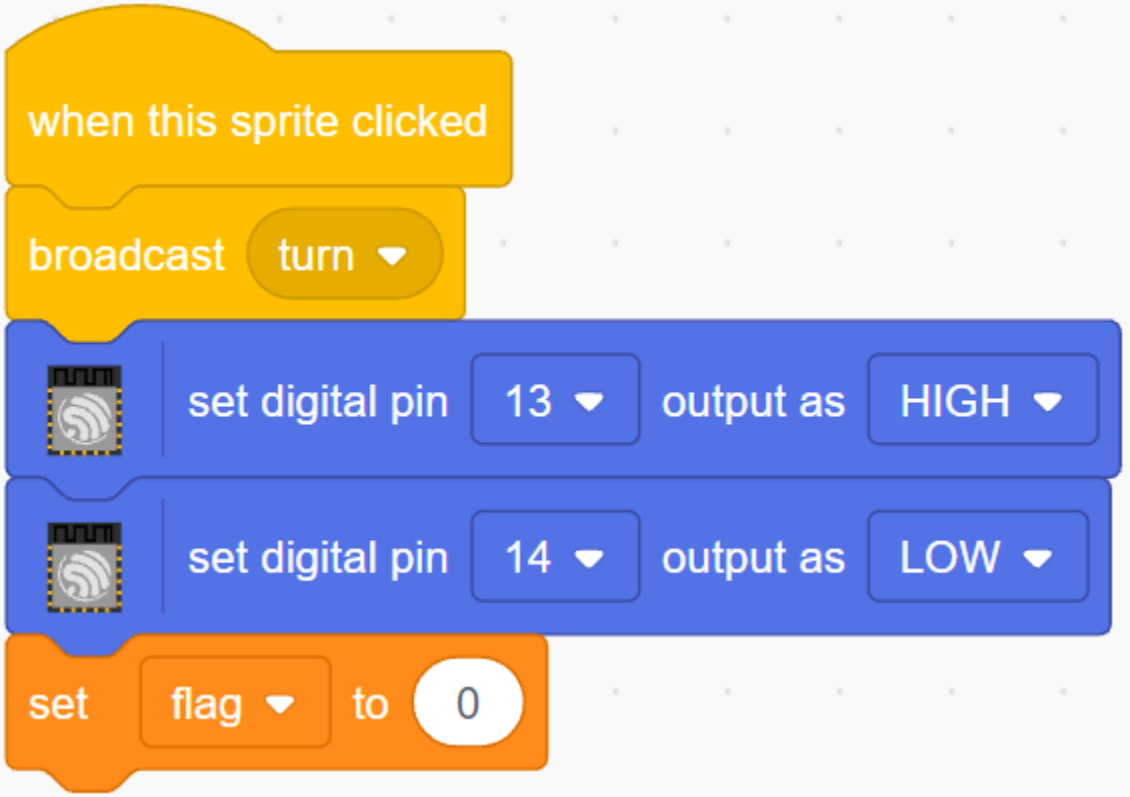
Hier gibt es 2 Punkte zu beachten.

- [broadcast]: aus der **Events**-Palette, verwendet um eine Nachricht an die anderen Sprites zu senden. Wenn die anderen Sprites diese Nachricht erhalten, führen sie ein bestimmtes Ereignis aus. Zum Beispiel hier **turn**, wenn das **star**-Sprite diese Nachricht erhält, führt es das Rotationsskript aus.
- Variable Flag: Die Drehrichtung des Stern-Sprites wird durch den Wert von Flag bestimmt. Wenn du also die **flag**-Variable erstellst, musst du sie für alle Sprites anwenden.



3. Rechtes Pfeil-Sprite

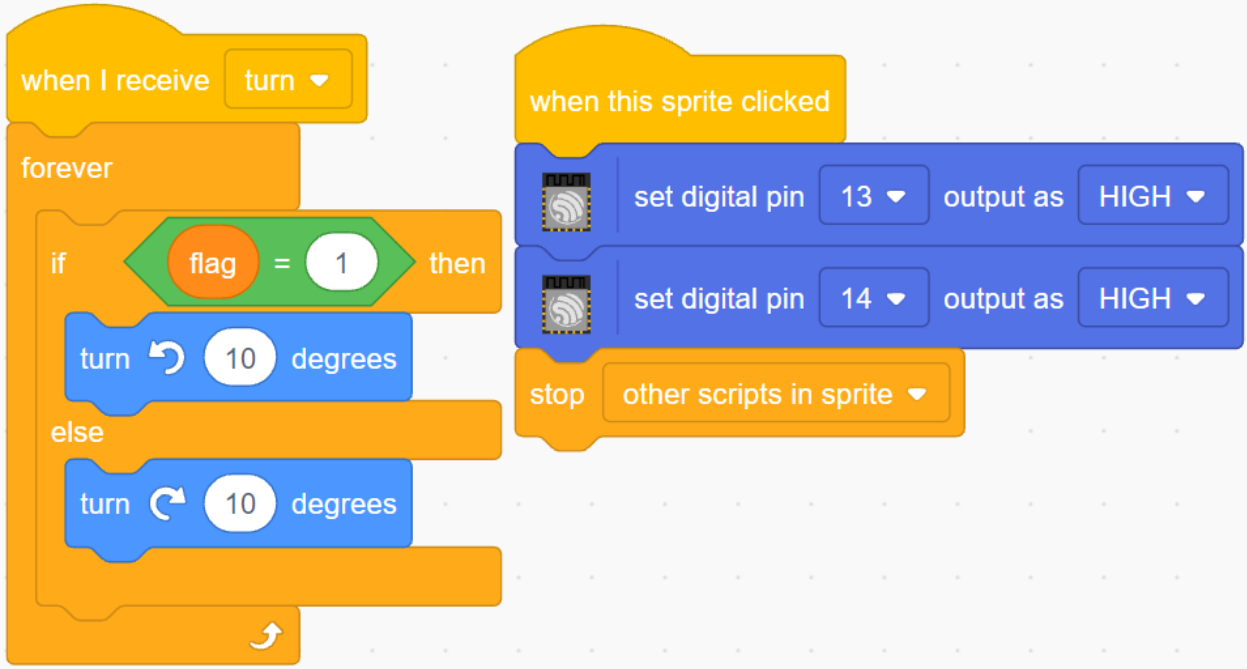
Wenn dieses Sprite angeklickt wird, sendet es eine Nachricht drehen, setzt dann digitalen Pin12 hoch und Pin14 niedrig, um den Motor im Uhrzeigersinn zu drehen und setzt die **flag**-Variable auf 0.



4. Stern-Sprite

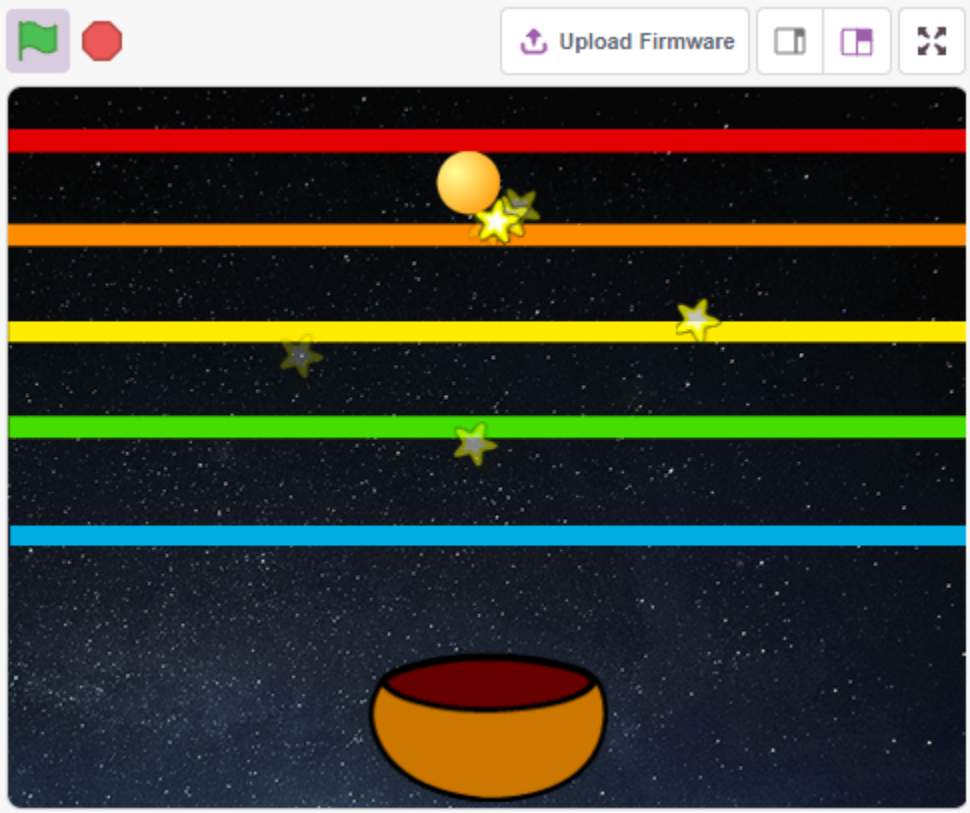
Hier sind 2 Ereignisse enthalten.

- Wenn das **star**-Sprite die gesendete Nachricht drehen erhält, bestimmt es den Wert von Flag; wenn Flag 1 ist, dreht es sich um 10 Grad nach links, andernfalls umgekehrt. Da es in [FOREVER] ist, wird es sich weiter drehen.
- Wenn dieses Sprite angeklickt wird, setze beide Pins des Motors auf hoch, um ihn zu stoppen und stoppe die anderen Skripte in diesem Sprite.



4.13 2.10 Lichtempfindlicher Ball

In diesem Projekt verwenden wir einen Fotowiderstand, um den Ball auf der Bühne nach oben fliegen zu lassen. Halten Sie Ihre Hand über den Fotowiderstand, um die Lichtintensität zu steuern, die er empfängt. Je näher Ihre Hand am Fotowiderstand ist, desto kleiner ist dessen Wert und desto höher fliegt der Ball auf der Bühne, andernfalls fällt er. Wenn der Ball die Schnur berührt, erzeugt er einen schönen Klang sowie funkelndes Sternenlicht.



4.13.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Fotowiderstand	

4.13.2 Was Sie Lernen Werden

- Füllen des Sprites mit Farben
- Berührung zwischen den Sprites

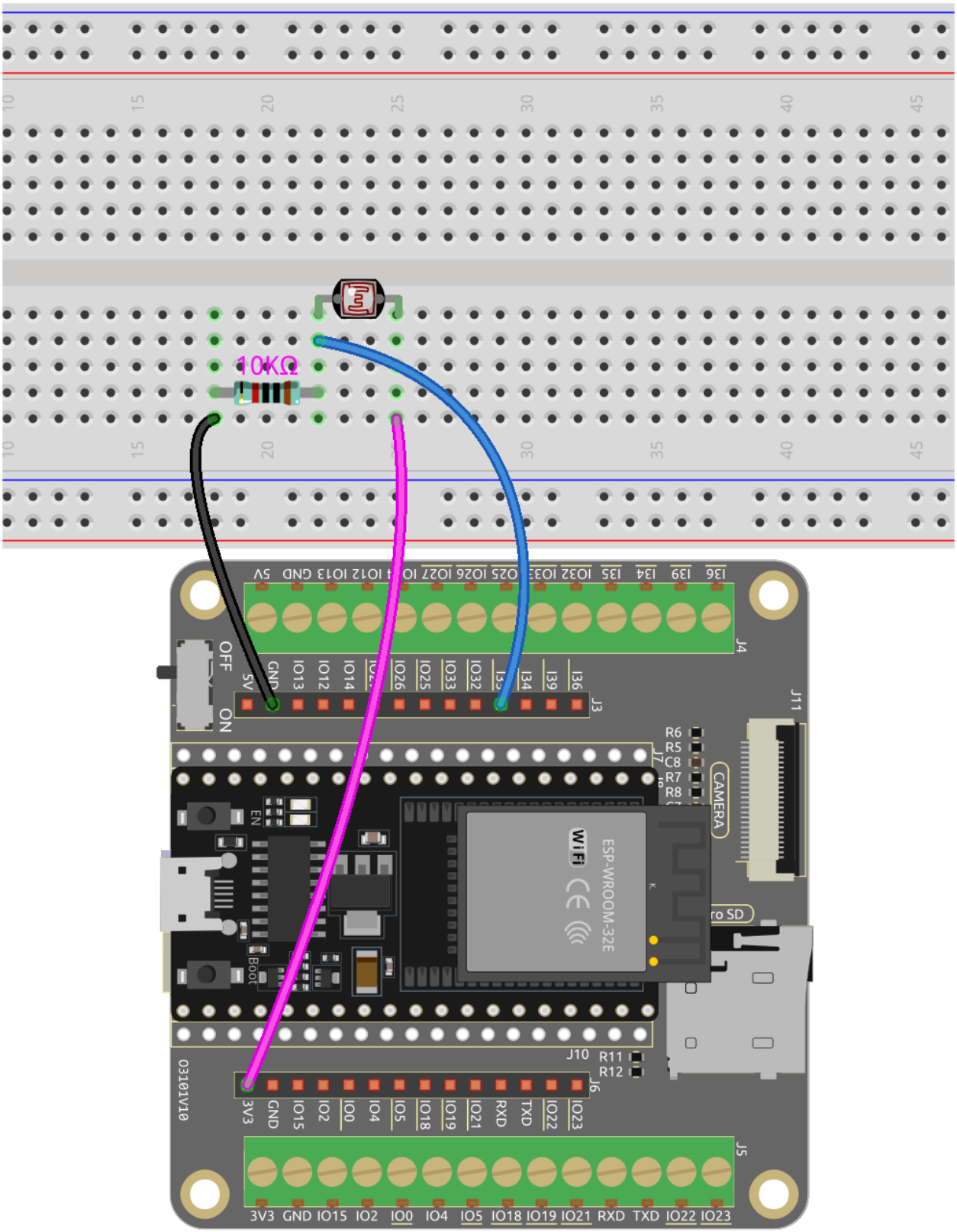
4.13.3 Schaltung Aufbauen

Ein Fotowiderstand oder eine Fotodiode ist ein lichtgesteuerter variabler Widerstand. Der Widerstand eines Fotowiderstands nimmt mit zunehmender einfallender Lichtintensität ab.

Bauen Sie die Schaltung gemäß dem folgenden Diagramm auf.

Verbinden Sie ein Ende des Fotowiderstands mit 5V, das andere Ende mit Pin35, und schalten Sie einen 10K-Widerstand in Serie mit GND an diesem Ende.

Wenn also die Lichtintensität zunimmt, verringert sich der Widerstand des Fotowiderstands, die Spannungsteilung des 10K-Widerstands nimmt zu, und der von Pin35 erhaltene Wert wird größer.

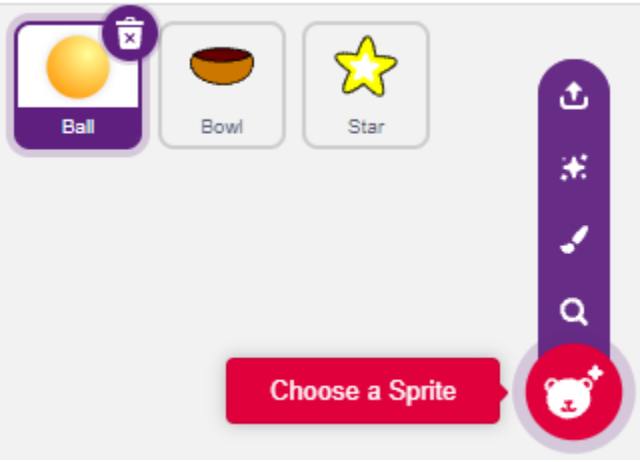

10KΩ
OFF
ON
ESP-WROOM-32E
CAMERA
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

4.13.4 Programmierung

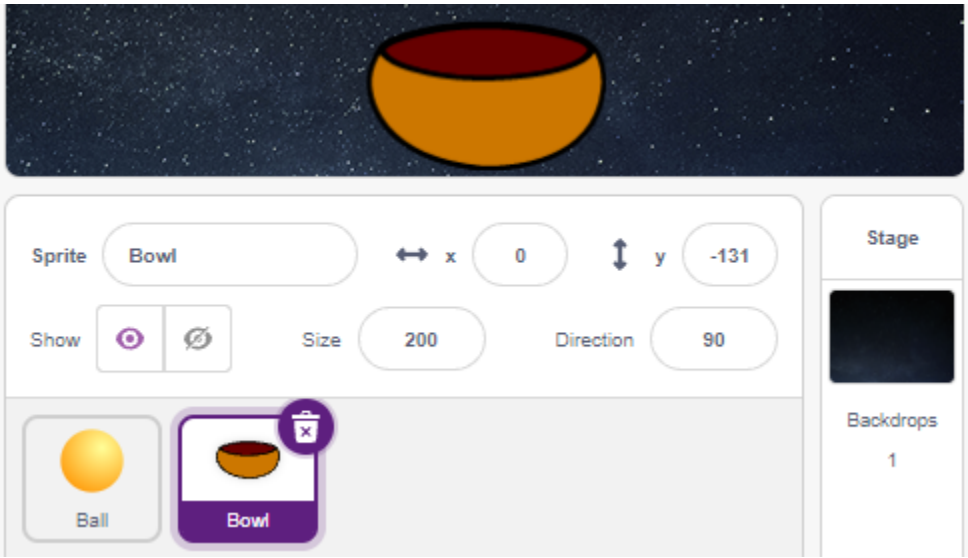
Das gewünschte Ergebnis ist, dass je näher Ihre Hand dem Fotowiderstand kommt, das Ball-Sprite auf der Bühne immer weiter nach oben geht, andernfalls fällt es auf das Schüssel-Sprite. Wenn es beim Aufsteigen oder Herunterfallen das Linien-Sprite berührt, erzeugt es einen musikalischen Klang und sendet Stern-Sprites in alle Richtungen.

1. Sprite und Hintergrund auswählen

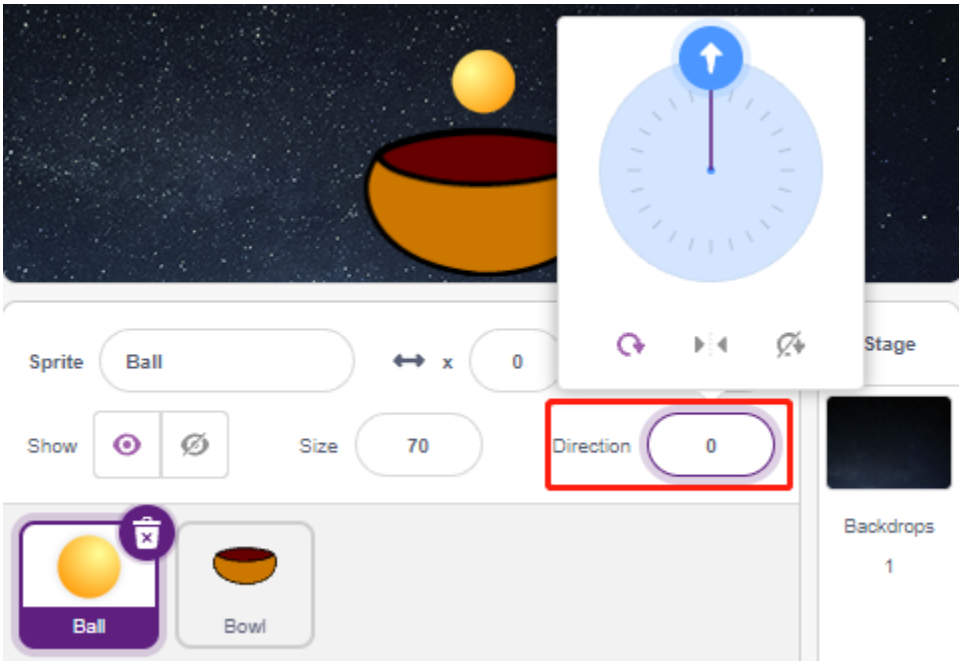
Lösche das Standard-Sprite, wähle die Sprites **Ball**, **Bowl** und **Star** aus.



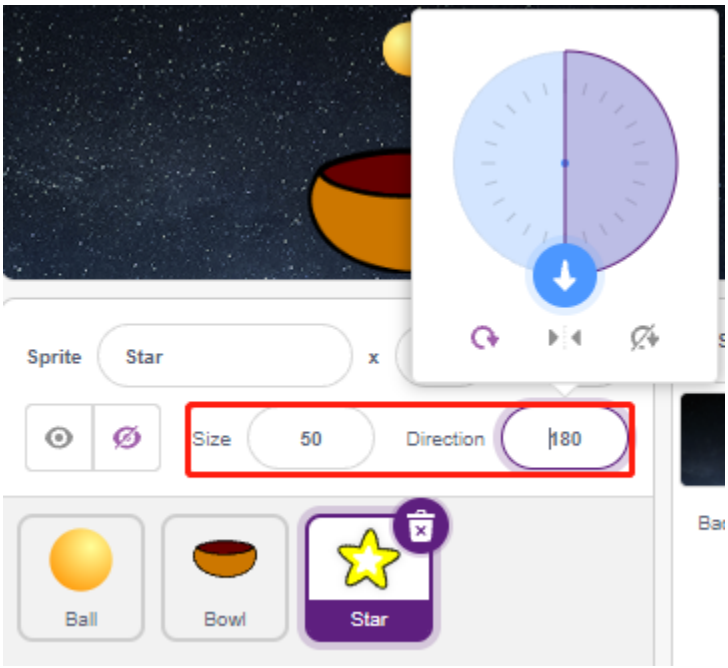
Bewege das **Bowl**-Sprite in die Mitte unten auf der Bühne und vergrößere seine Größe.



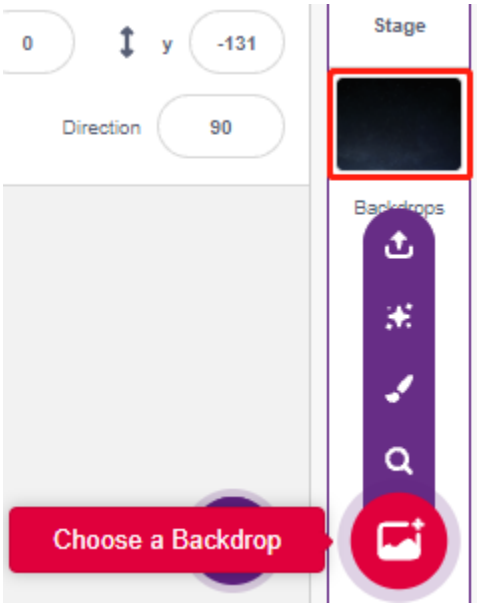
Da wir es nach oben bewegen müssen, setze die Richtung des **Ball**-Sprites auf 0.



Setze die Größe und Richtung des **Star**-Sprites auf 180, da es nach unten fallen soll, oder ändere es in einen anderen Winkel.

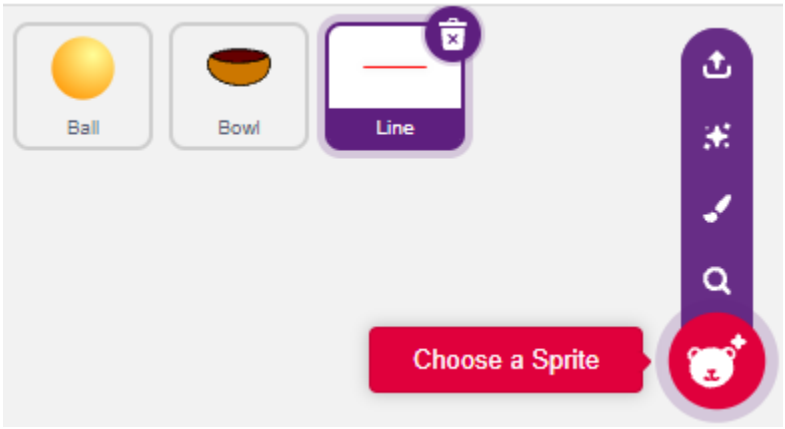


Füge nun den **Stars**-Hintergrund hinzu.

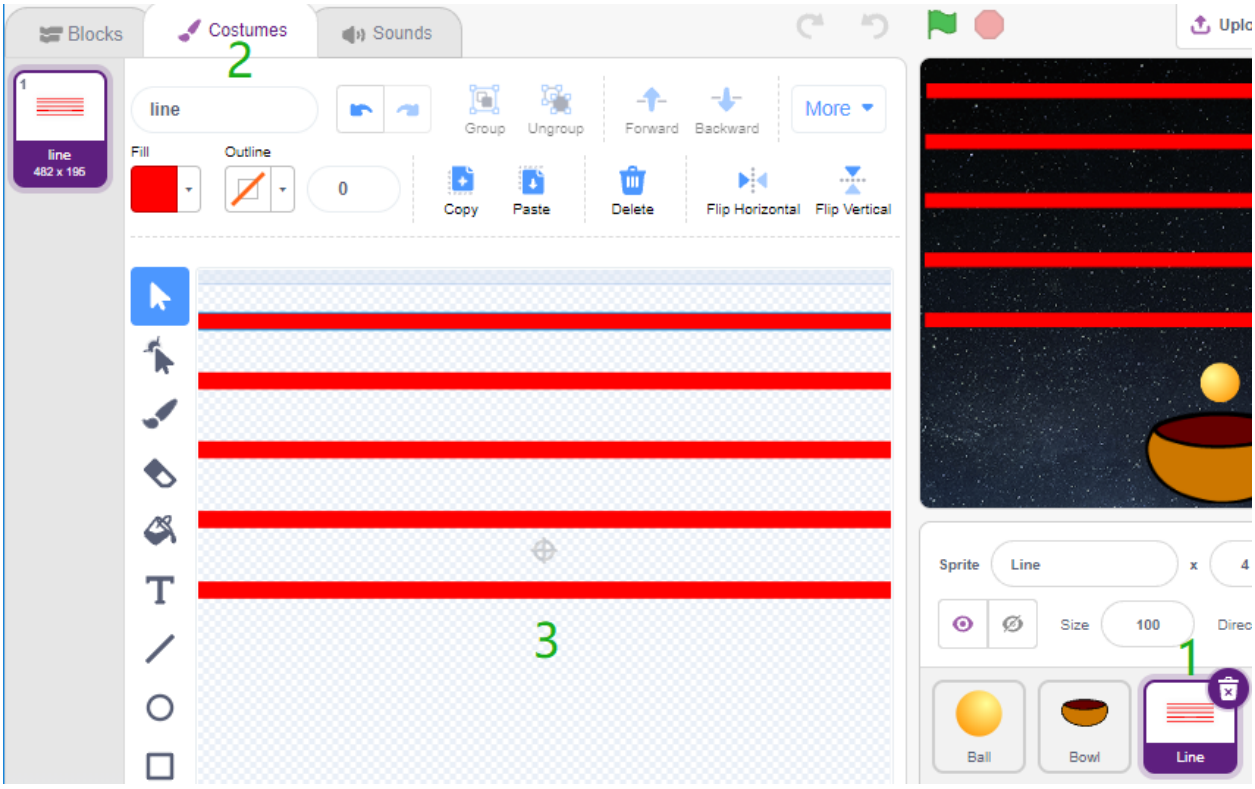


2. Ein Linien-Sprite zeichnen

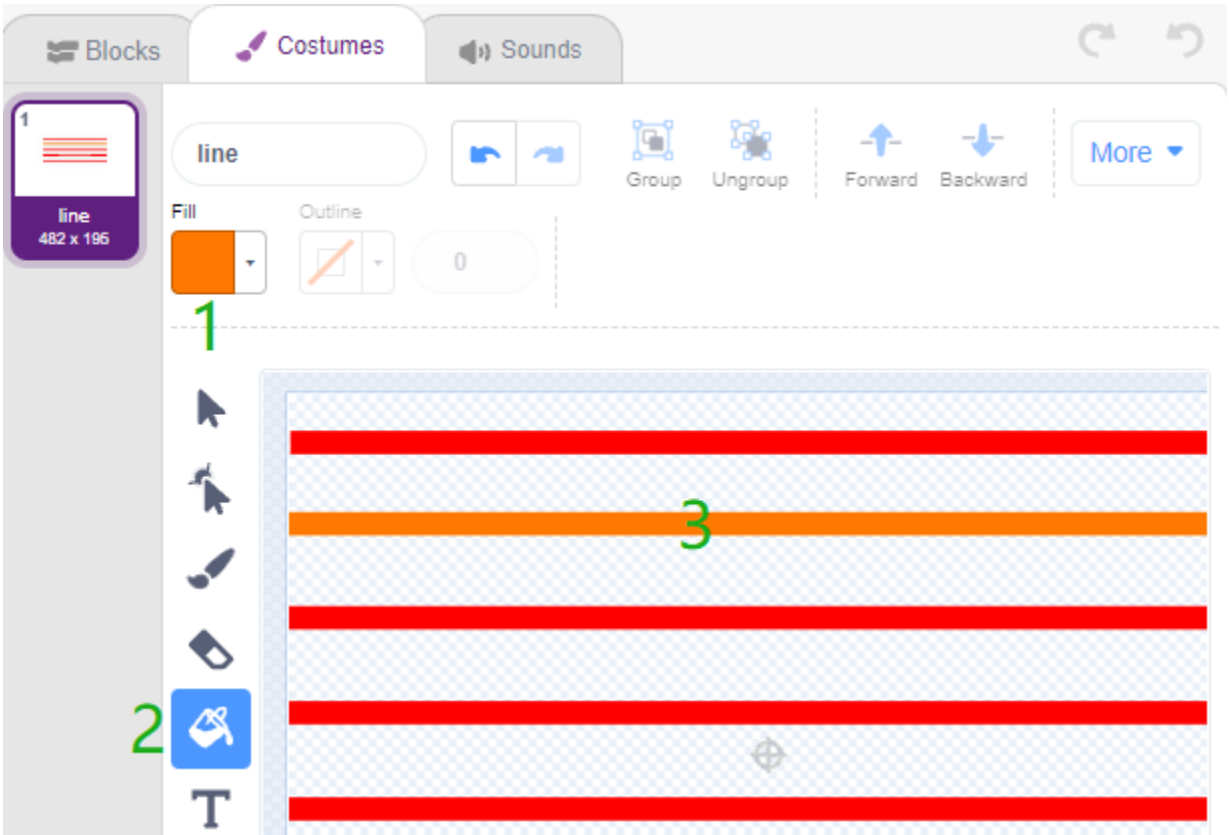
Füge ein Linien-Sprite hinzu.



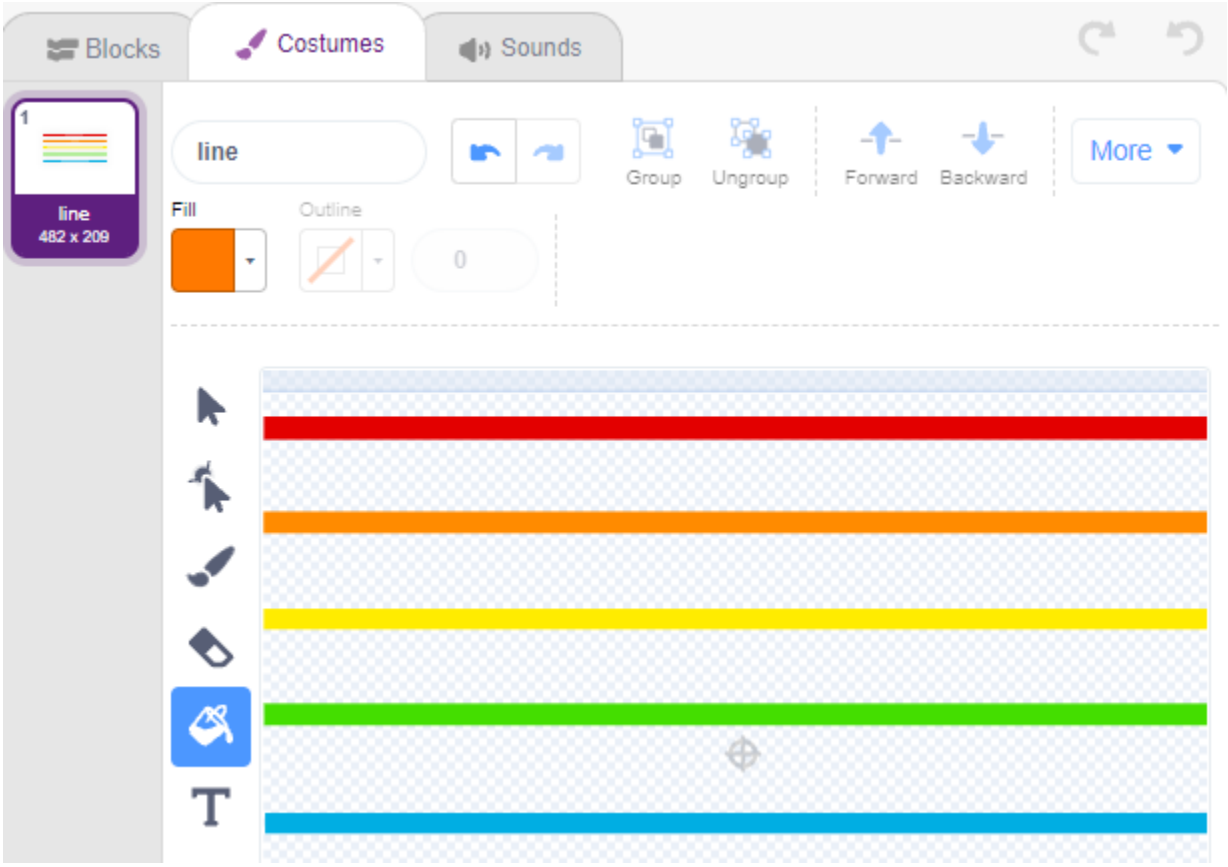
Gehe zur **Costumes**-Seite des **Line**-Sprites, reduziere die Breite der roten Linie auf der Leinwand leicht, kopiere sie dann 5 Mal und richte die Linien aus.



Fülle nun die Linien mit verschiedenen Farben. Wähle zuerst eine Farbe, die dir gefällt, klicke dann auf das **Fill**-Werkzeug und bewege die Maus über die Linie, um sie mit Farbe zu füllen.



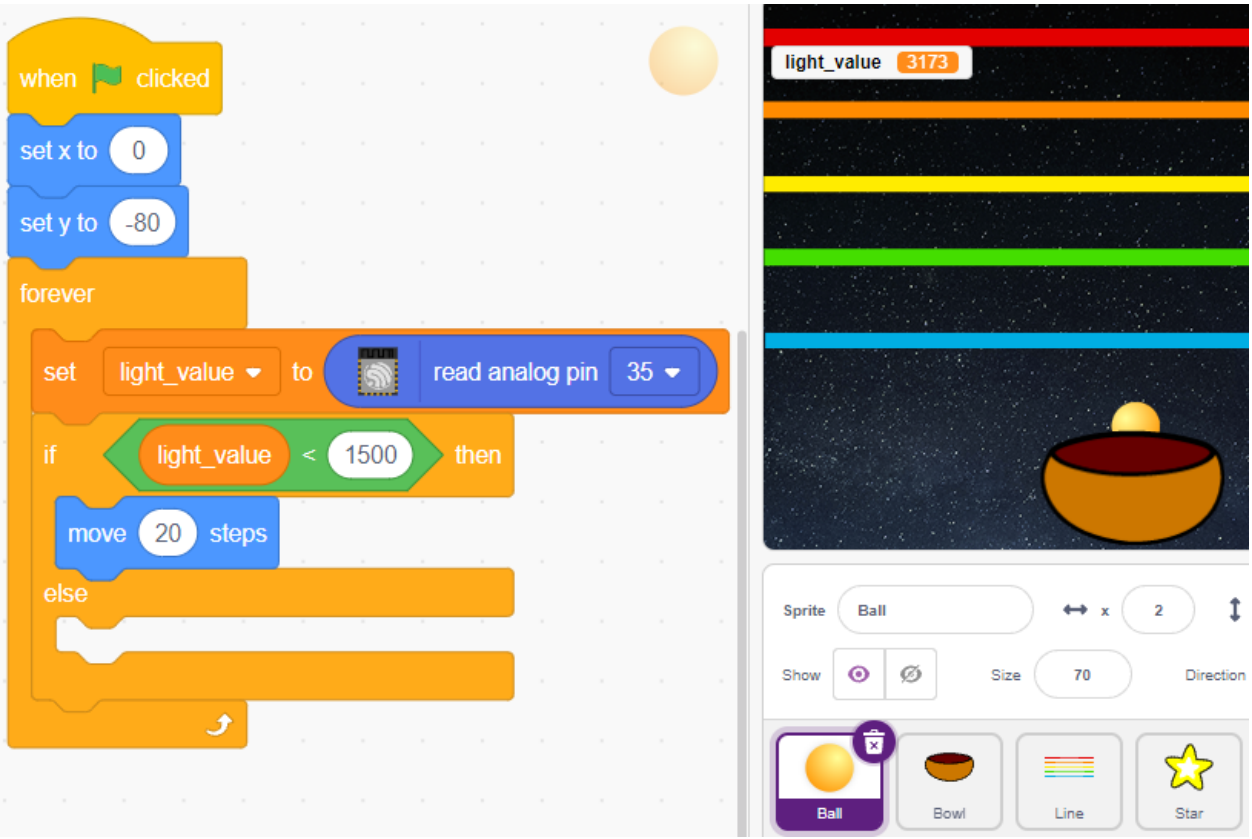
Verwende die gleiche Methode, um die Farbe der anderen Linien zu ändern.



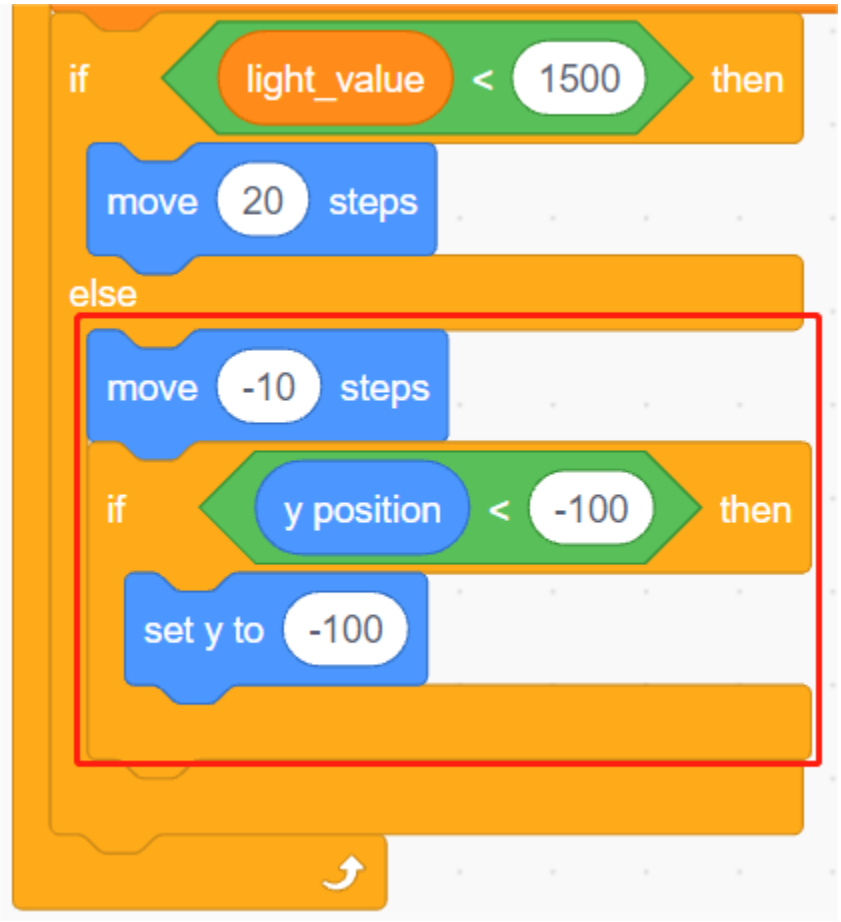
3. Skript für das Ball-Sprite

Setze die Anfangsposition des **Ball**-Sprites, dann, wenn der Lichtwert kleiner als 1500 ist (es kann ein anderer Wert sein, abhängig von deiner aktuellen Umgebung.), lass den Ball nach oben bewegen.

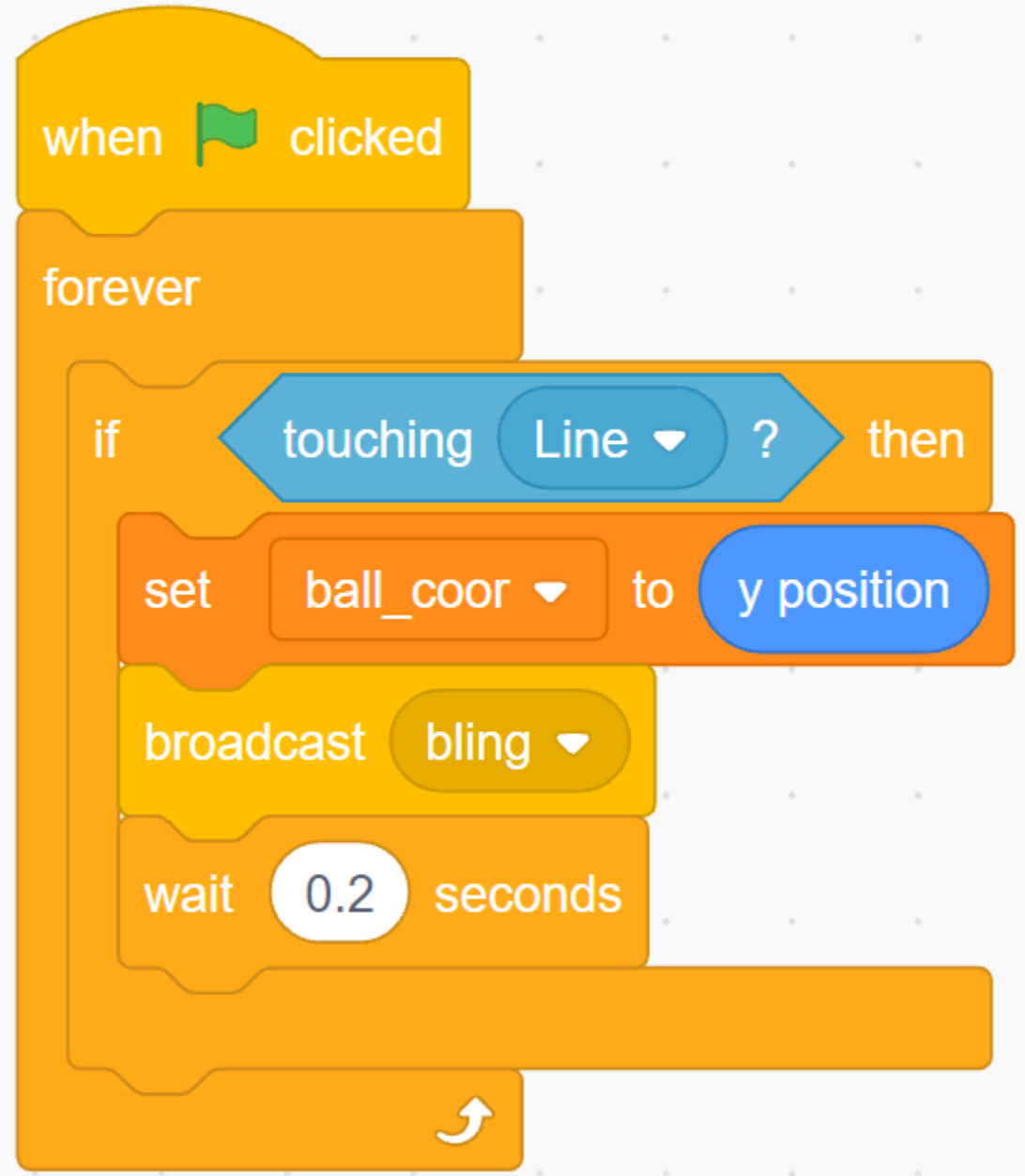
Du kannst die Variable light_value auf der Bühne anzeigen lassen, um die Änderung der Lichtintensität jederzeit zu beobachten.



Andernfalls fällt das **Ball**-Sprite und seine Y-Koordinate wird auf ein Minimum von -100 begrenzt. Dies kann geändert werden, damit es aussieht, als würde es auf das **Bowl**-Sprite fallen.

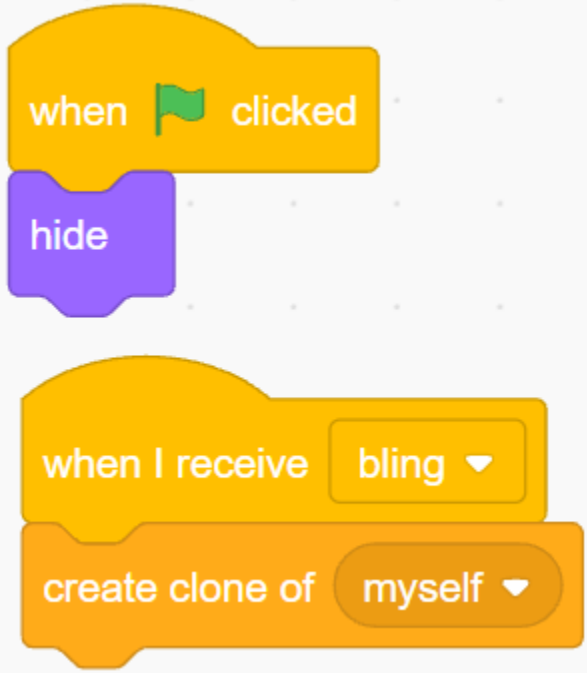


Wenn das **Line**-Sprite getroffen wird, wird die aktuelle Y-Koordinate in die Variable **ball_coor** gespeichert und eine **Bling**-Nachricht gesendet.

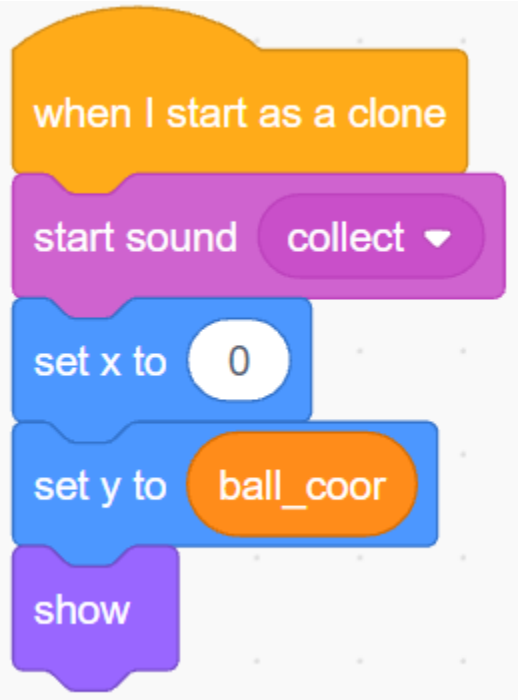


4. Skript für das Stern-Sprite

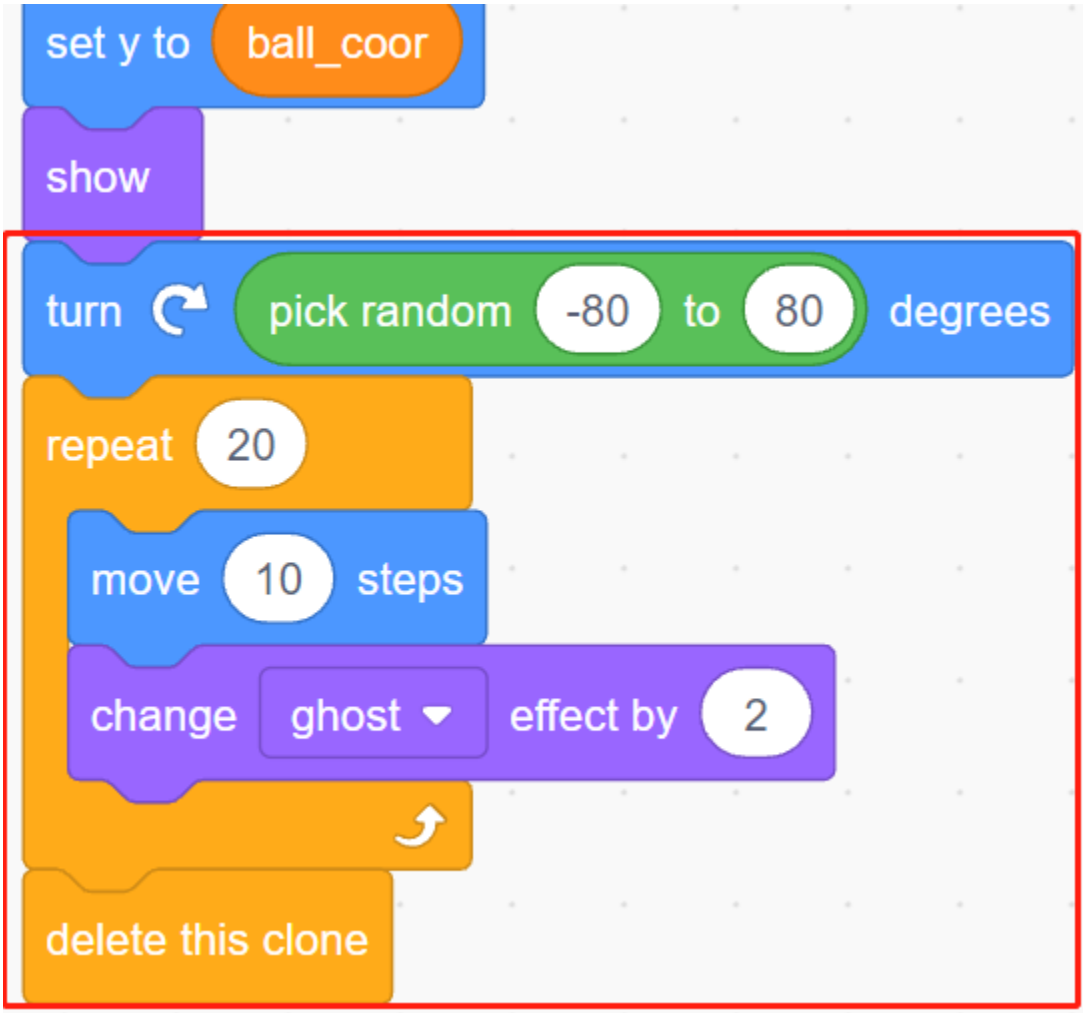
Wenn das Skript startet, verstecke zunächst das **Star**-Sprite. Wenn die **Bling**-Nachricht empfangen wird, klone das **Star**-Sprite.



Wenn das **Star**-Sprite als Klon erscheint, spiele den Toneffekt und setze seine Koordinaten synchron zum **Ball**-Sprite.



Erstelle den Effekt des erscheinenden **Star**-Sprites und passe ihn bei Bedarf an.

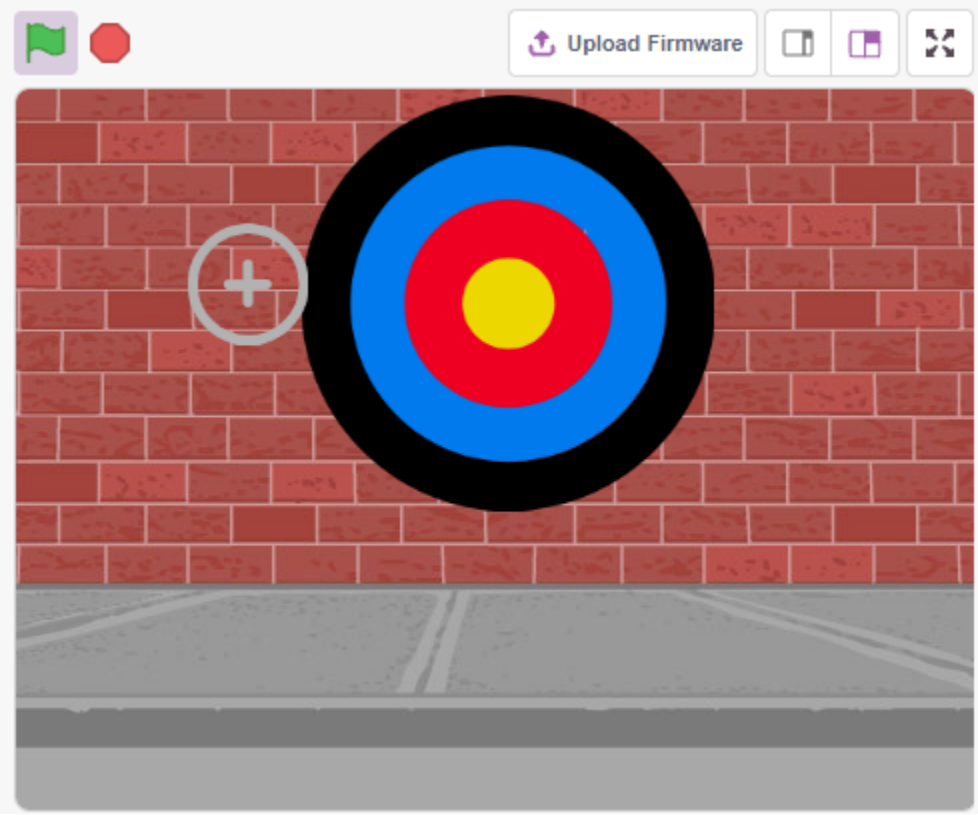


4.14 2.11 SPIEL - Schießen

Haben Sie schon einmal diese Schießspiele im Fernsehen gesehen? Je näher ein Teilnehmer mit einer Kugel am Ziel zum Bullseye schießt, desto höher ist seine Punktzahl.

Heute machen wir auch ein Schießspiel in Scratch. Im Spiel soll das Fadenkreuz so weit wie möglich zum Bullseye schießen, um eine höhere Punktzahl zu erreichen.

Klicken Sie auf die grüne Flagge, um zu starten. Verwenden Sie das Modul zur Hindernisvermeidung, um eine Kugel zu schießen.



4.14.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Modul zur Hindernisvermeidung	

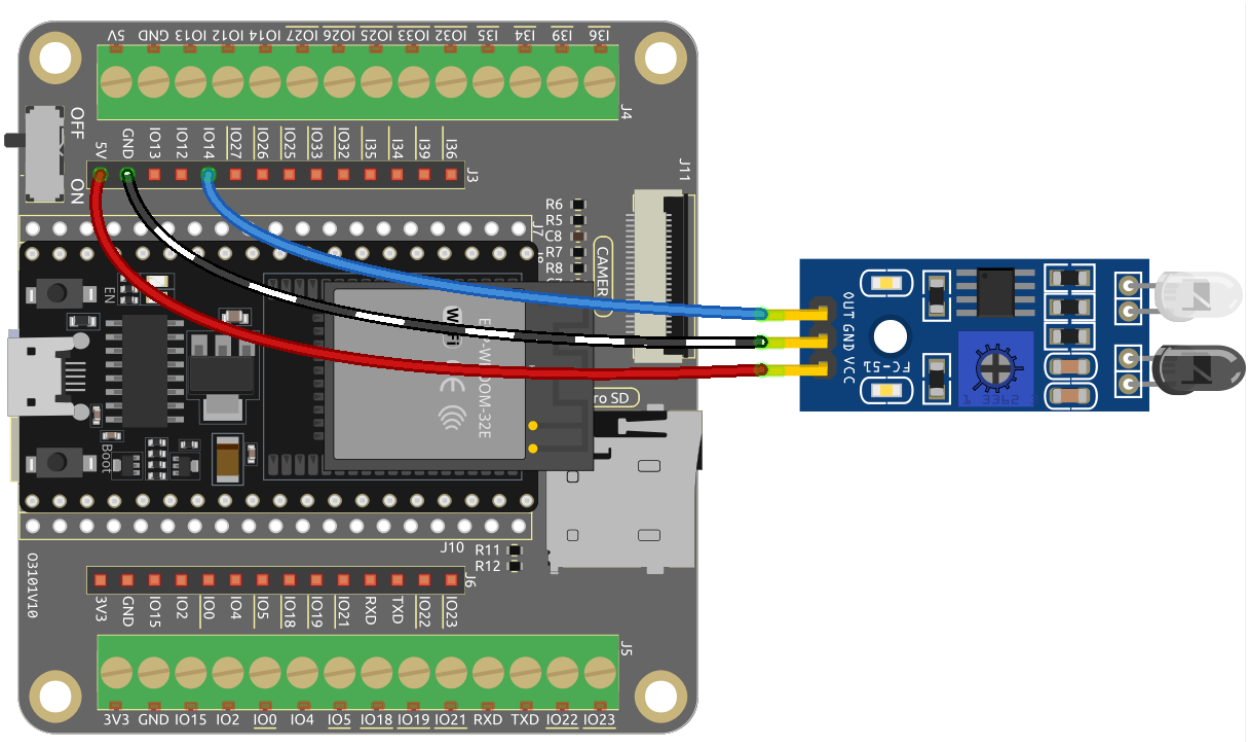
4.14.2 Was Sie Lernen Werden

- Funktionsweise des Hindernisvermeidungsmoduls und der Winkelbereich
- Unterschiedliche Sprites malen
- Farben berühren

4.14.3 Schaltung Aufbauen

Das Hindernisvermeidungsmodul ist ein infrarotbasierter Näherungssensor mit einstellbarer Distanz, dessen Ausgang normalerweise hoch ist und bei Erkennung eines Hindernisses niedrig wird.

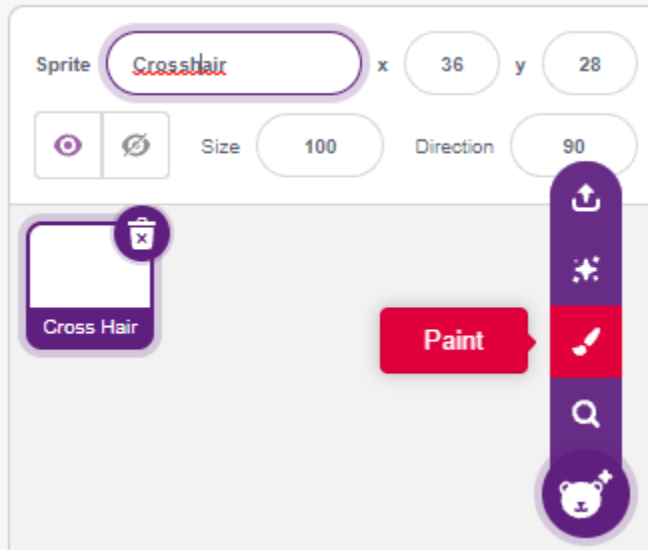
Bauen Sie die Schaltung gemäß dem untenstehenden Diagramm auf.



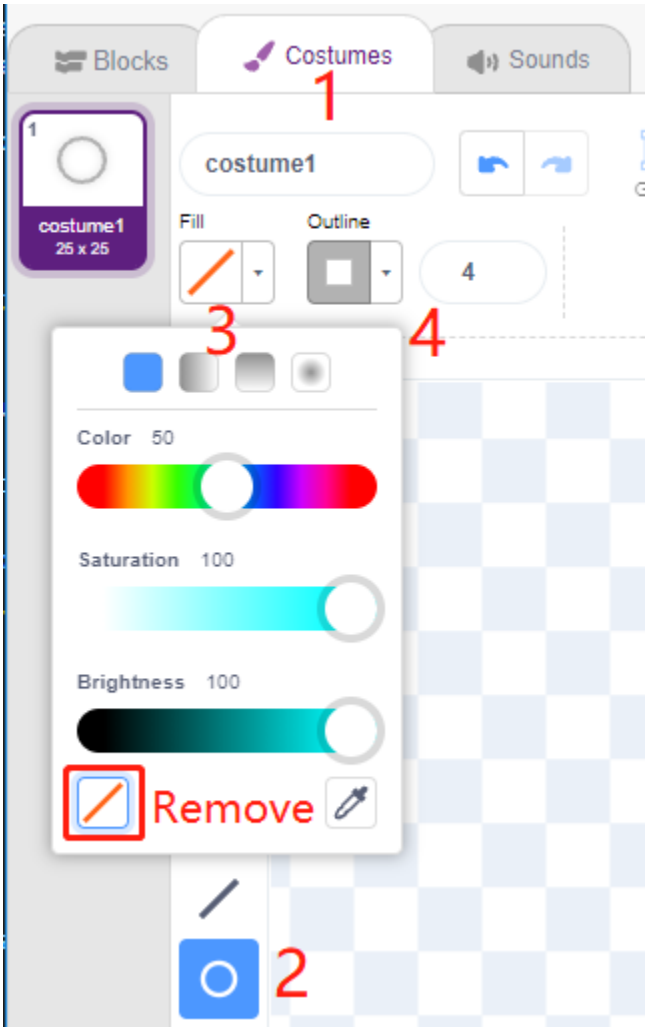
4.14.4 Programmierung

1. Das Fadenkreuz-Sprite malen

Lösche das Standard-Sprite, wähle den **Sprite**-Button und klicke auf **Paint**, es erscheint ein leeres Sprite **Sprite1** und benenne es **Crosshair**.

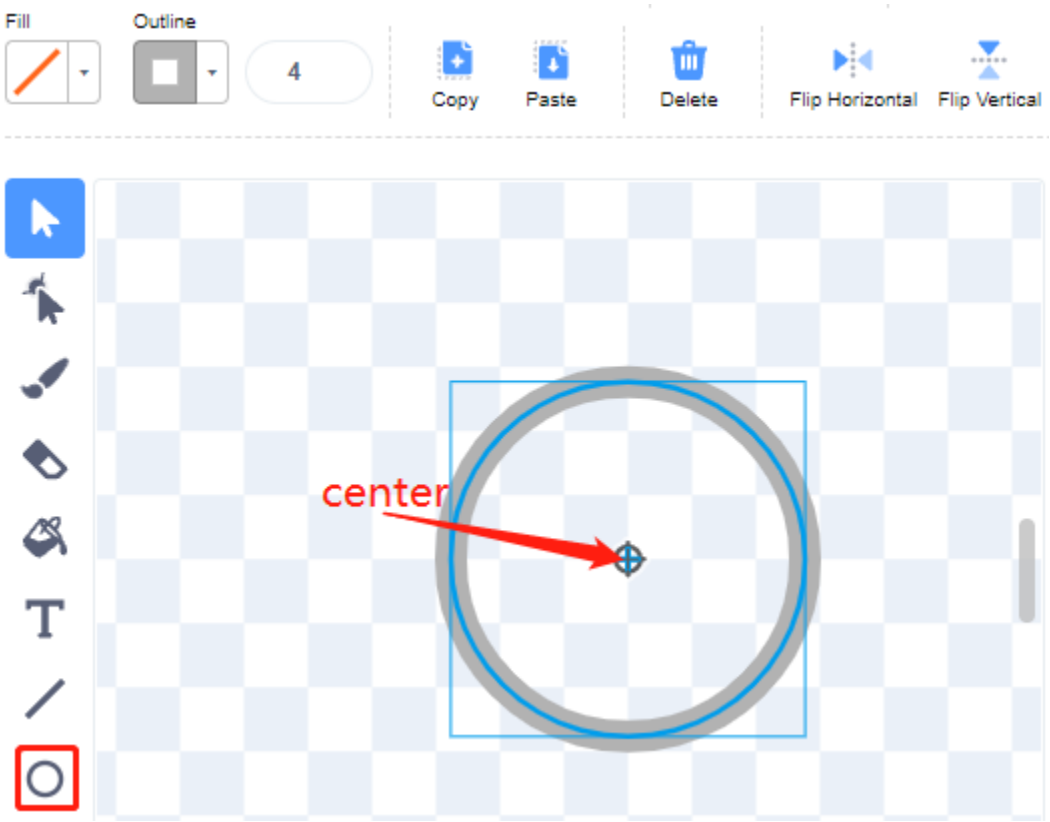


Gehe zur **Costumes**-Seite des **Crosshair**-Sprites. Klicke auf das **Circle**-Werkzeug, entferne die Füllfarbe und stelle die Farbe und Breite der Umrandung ein.

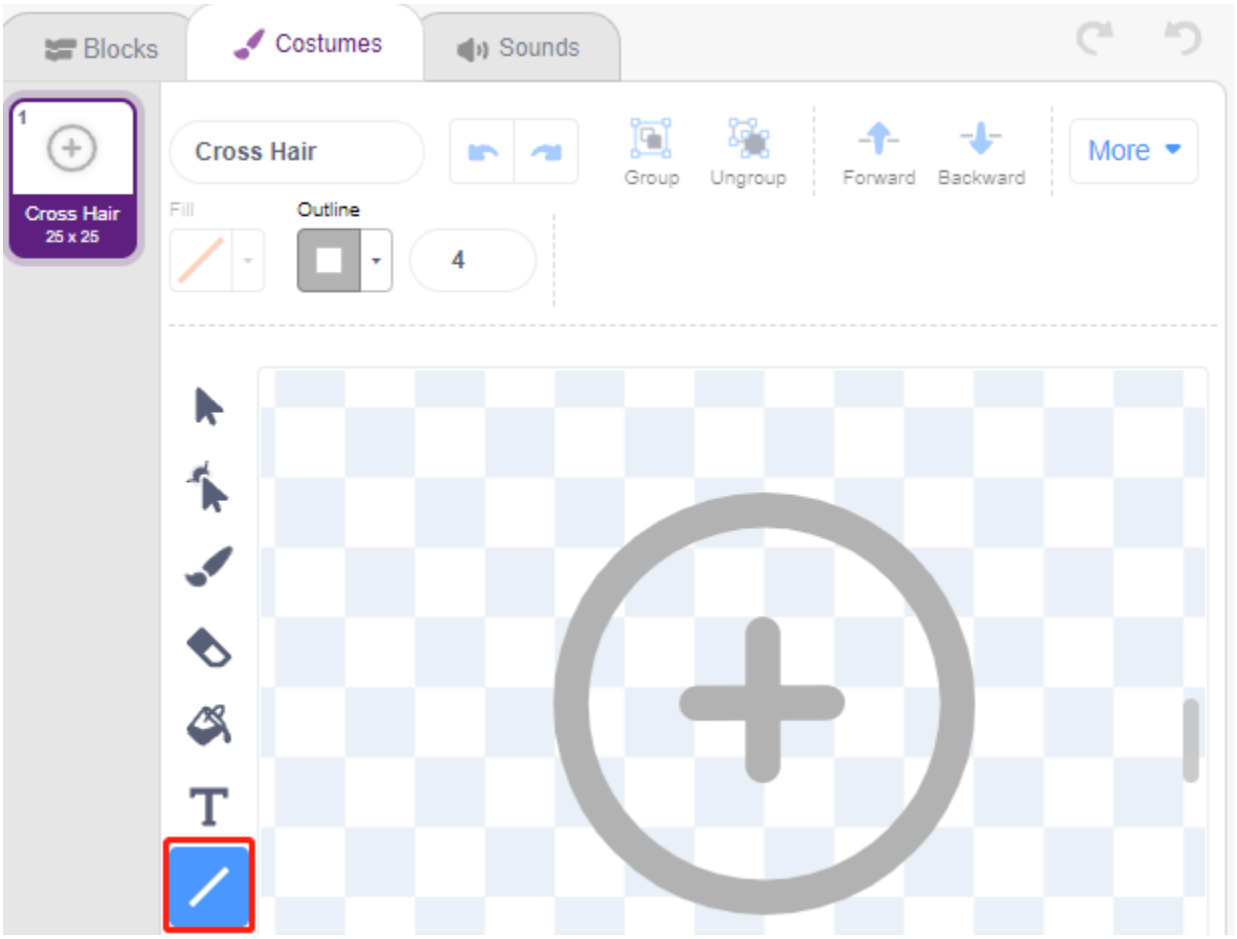


Zeichne jetzt einen Kreis mit dem **Circle**-Werkzeug. Nach dem Zeichnen kannst du das **Select**-Werkzeug anklicken

und den Kreis verschieben, sodass der ursprüngliche Punkt mit der Mitte der Leinwand ausgerichtet ist.

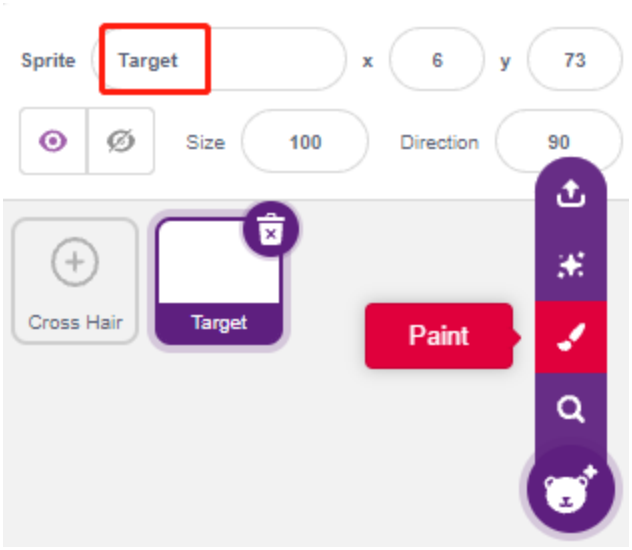


Verwende das **Line**-Werkzeug, um ein Kreuz in den Kreis zu zeichnen.

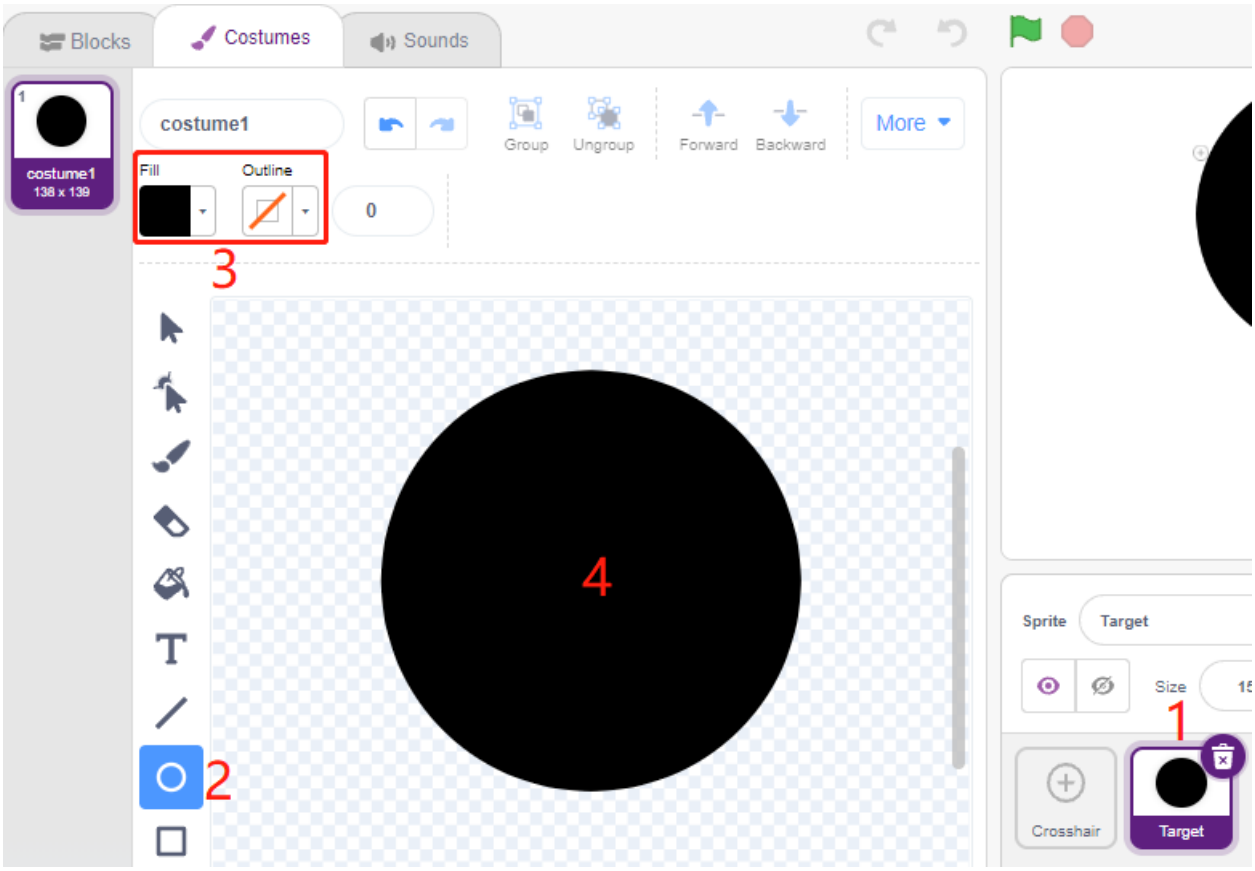


Male das Ziel-Sprite

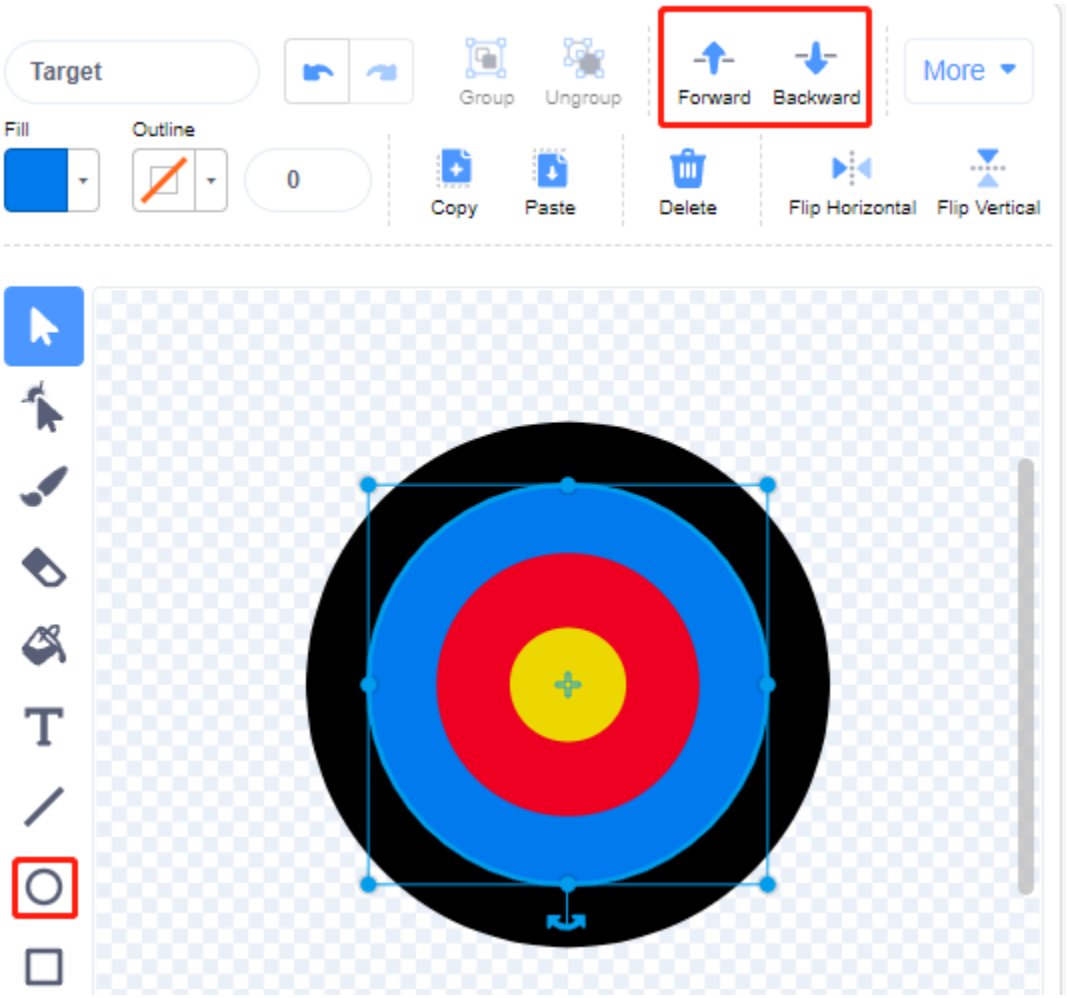
Erstelle ein neues Sprite namens **Target**-Sprite.



Gehe zur Kostüme-Seite des **Target**-Sprites, klicke auf das **Circle**-Werkzeug, wähle eine Füllfarbe und entferne die Umrandung und male einen großen Kreis.

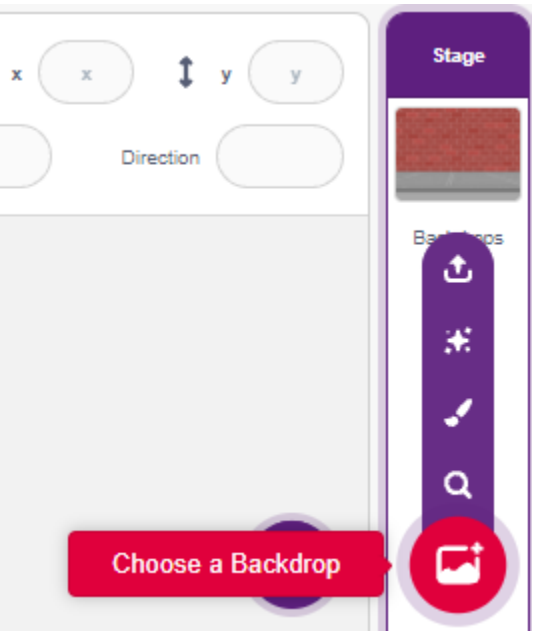


Verwende die gleiche Methode, um zusätzliche Kreise zu zeichnen, jeder mit einer anderen Farbe, und du kannst das **Forward**- oder **Backbard**-Werkzeug verwenden, um die Position der sich überlappenden Kreise zu ändern. Beachte, dass du auch das Werkzeug zum Verschieben der Kreise auswählen musst, sodass der Ursprung aller Kreise und die Mitte der Leinwand ausgerichtet sind.



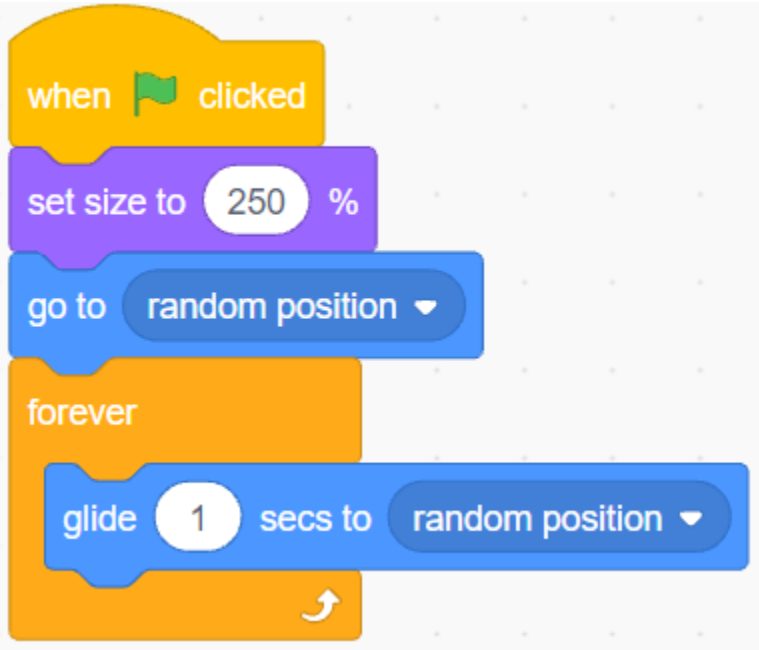
3. Einen Hintergrund hinzufügen

Füge einen passenden Hintergrund hinzu, der vorzugsweise nicht zu viele Farben hat und nicht mit den Farben im **Target**-Sprite übereinstimmt. Hier habe ich den **Wall1**-Hintergrund gewählt.

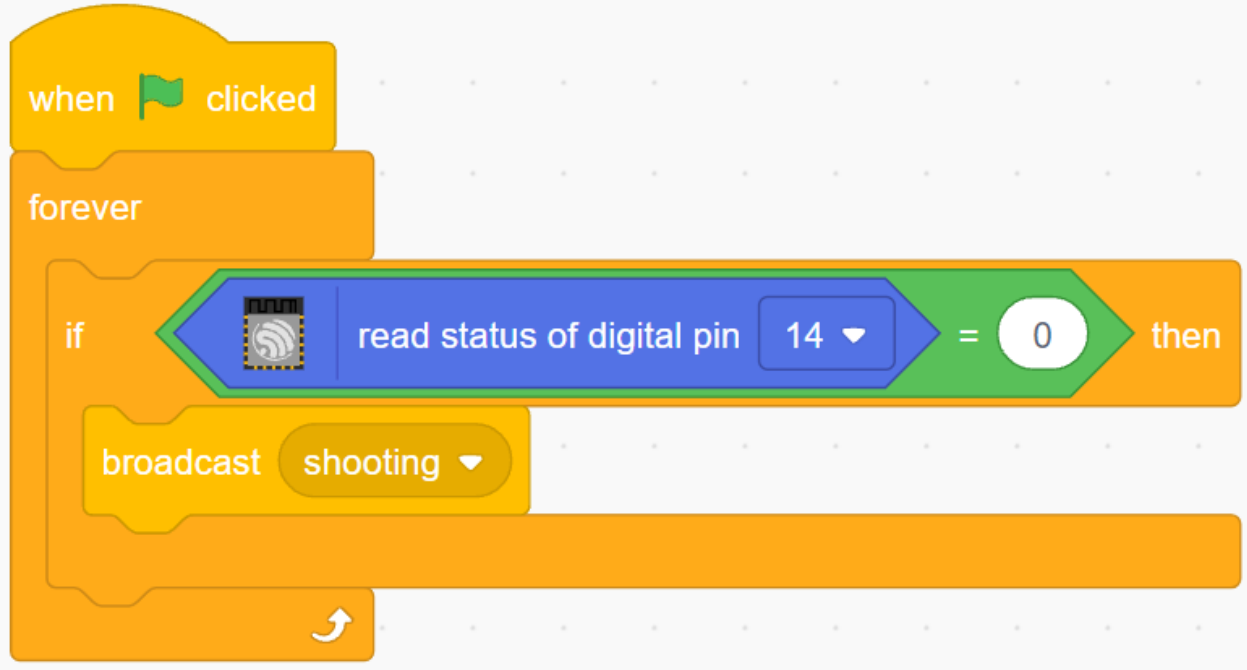


4. Skript für das Fadenkreuz-Sprite

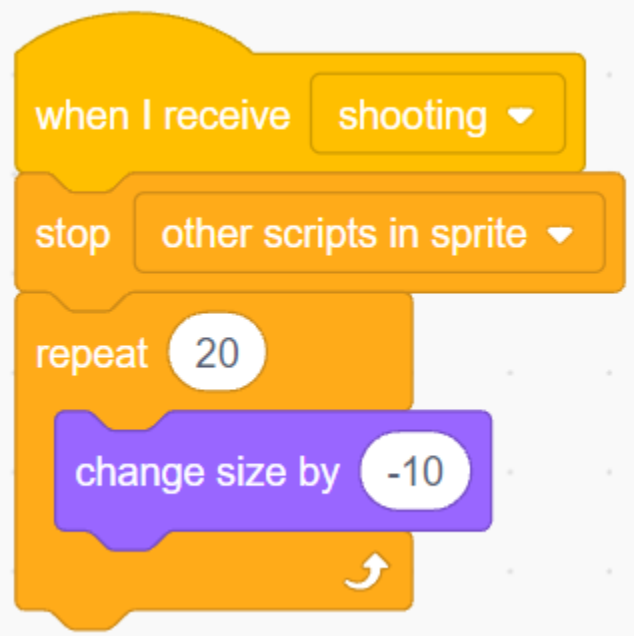
Stelle die zufällige Position und Größe des **Crosshair**-Sprites ein und lass es sich zufällig bewegen.



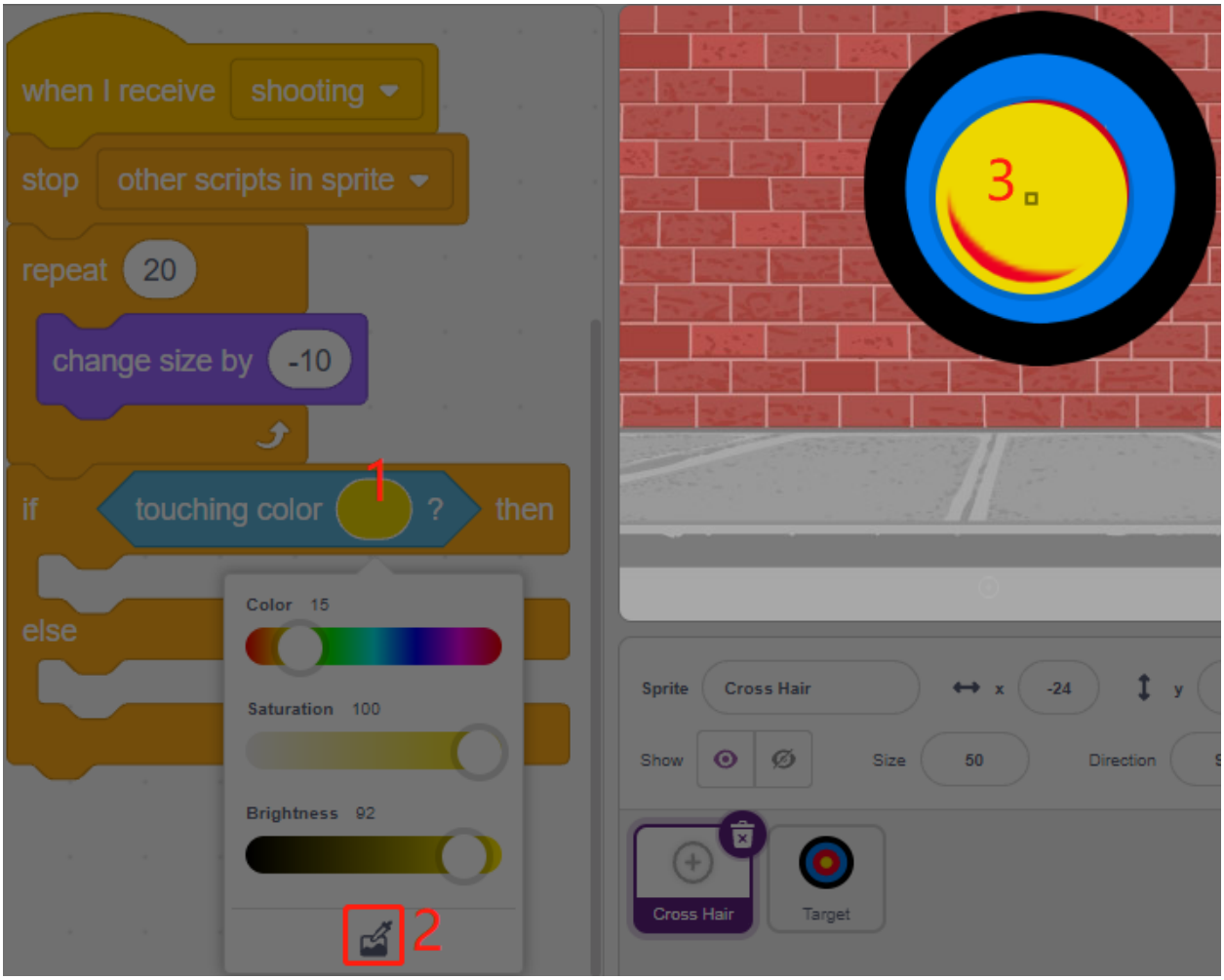
Wenn eine Hand vor das Hindernisvermeidungsmodul gehalten wird, gibt es ein niedriges Signal als Sendesignal aus.



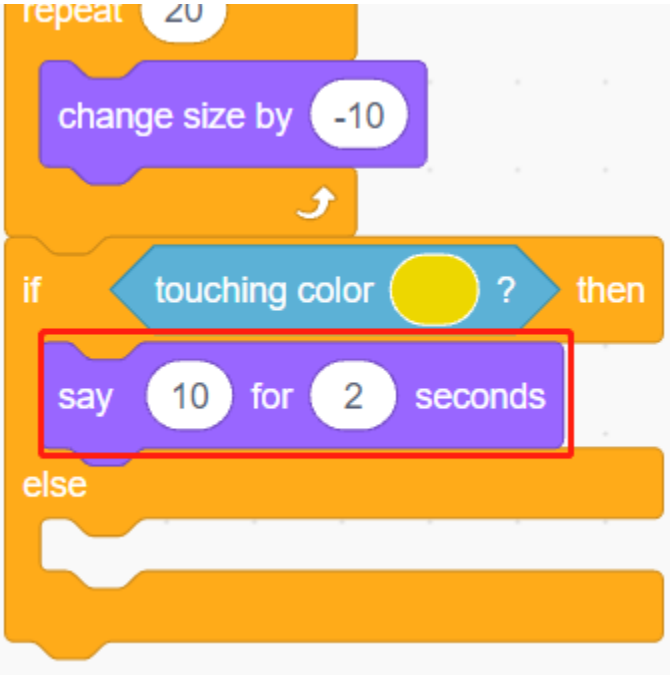
Wenn die Nachricht **shooting** empfangen wird, stoppt das Sprite seine Bewegung und schrumpft langsam, um den Effekt eines abgefeuerten Geschosses zu simulieren.



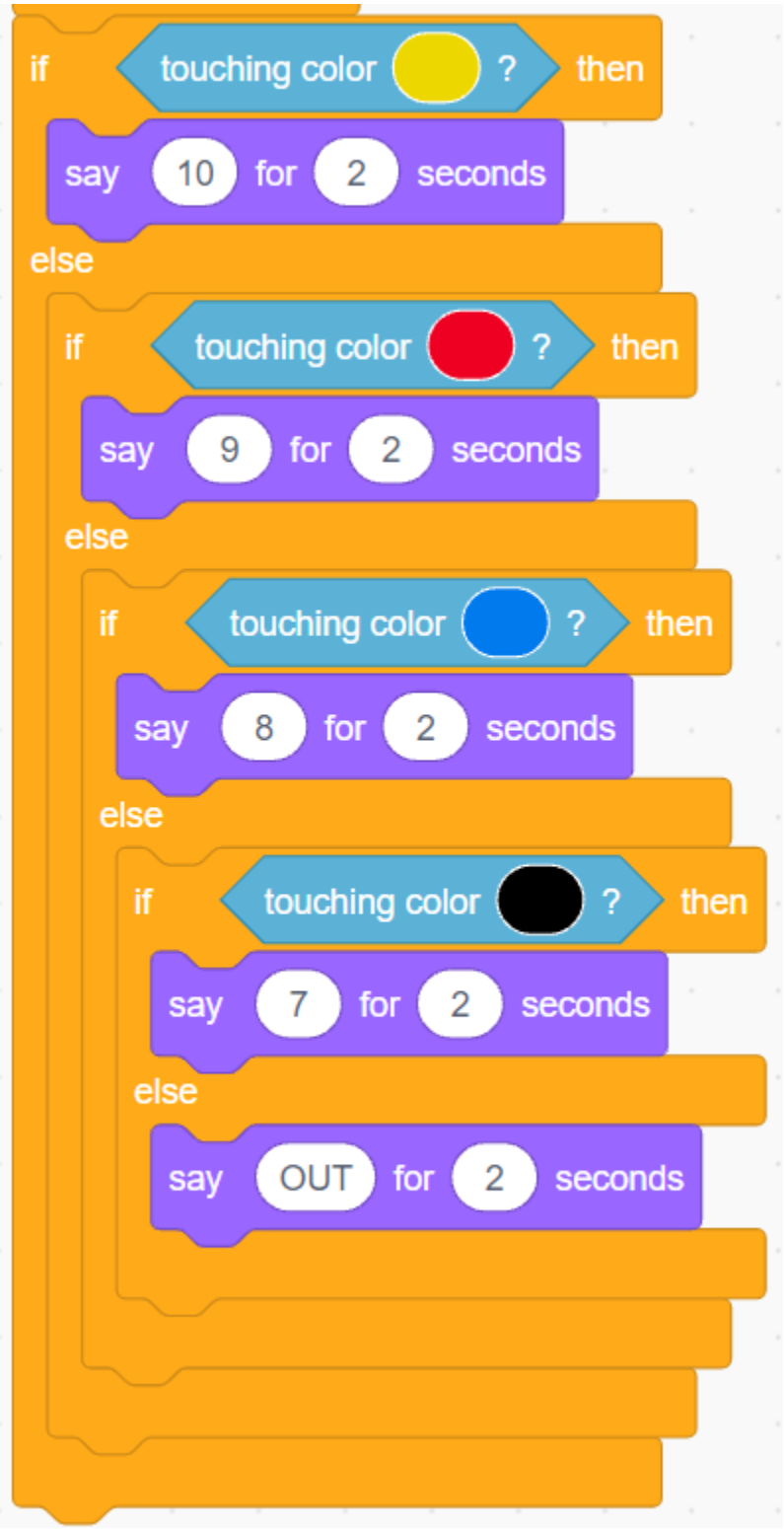
Verwende den Block [Touch color ()], um die Position des Schusses zu bestimmen.



Wenn der Schuss innerhalb des gelben Kreises liegt, werden 10 Punkte vergeben.



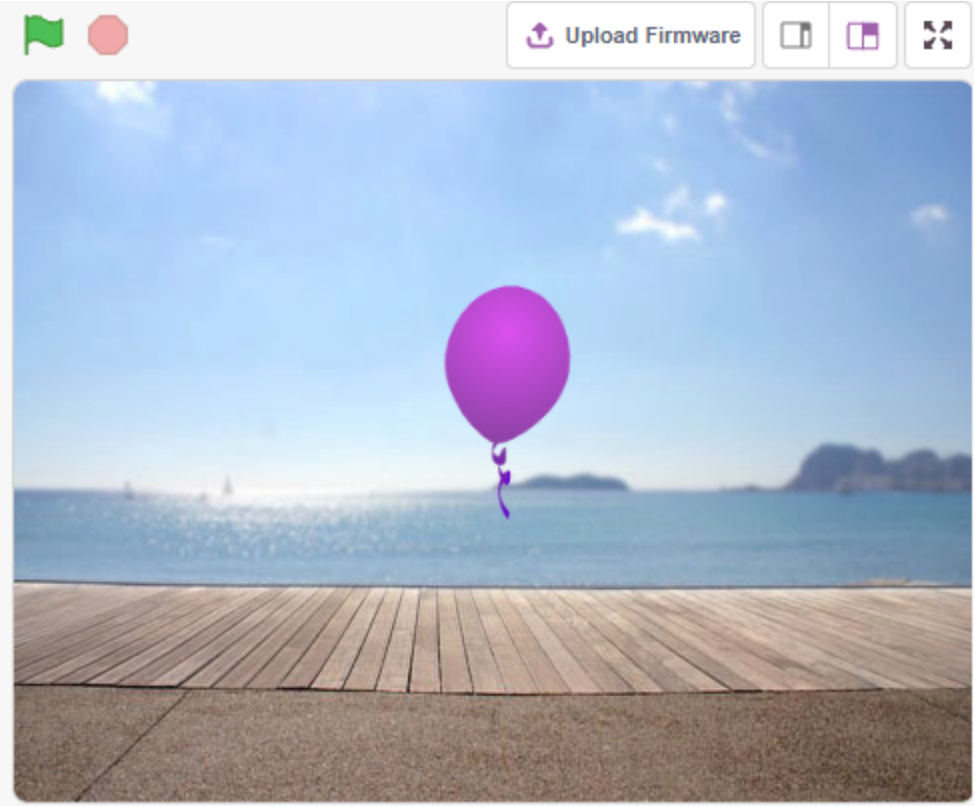
Verwende die gleiche Methode, um die Position des abgefeuerten Geschosses zu bestimmen. Wenn es nicht auf dem **Target**-Sprite landet, bedeutet das, dass es außerhalb des Kreises ist.


if touching color ? then
say 10 for 2 seconds
else
if touching color ? then
say 9 for 2 seconds
else
if touching color ? then
say 8 for 2 seconds
else
if touching color ? then
say 7 for 2 seconds
else
say OUT for 2 seconds

4.15 2.12 SPIEL - Ballon Aufblasen

Hier spielen wir ein Ballonaufblas-Spiel.

Nachdem die grüne Flagge angeklickt wurde, wird der Ballon immer größer. Wenn der Ballon zu groß wird, platzt er; wenn der Ballon zu klein ist, fällt er herunter; du musst beurteilen, wann du den Knopf drücken solltest, um ihn nach oben fliegen zu lassen.



4.15.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

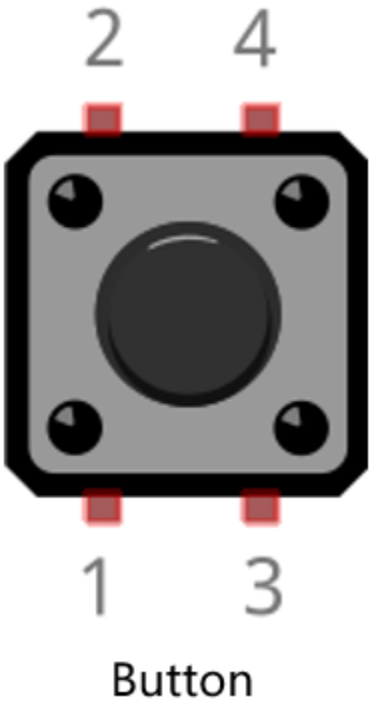
KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Taste	

4.15.2 Was Sie Lernen Werden

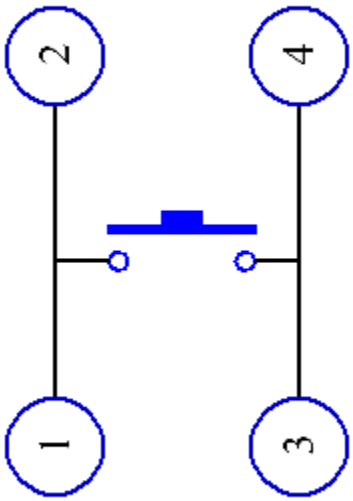
- Kostüm für das Sprite malen

4.15.3 Schaltung Aufbauen

Der Knopf ist ein 4-poliges Gerät, da Pin 1 mit Pin 2 verbunden ist und Pin 3 mit Pin 4, wenn der Knopf gedrückt wird, sind die 4 Pins verbunden, wodurch der Stromkreis geschlossen wird.



Button

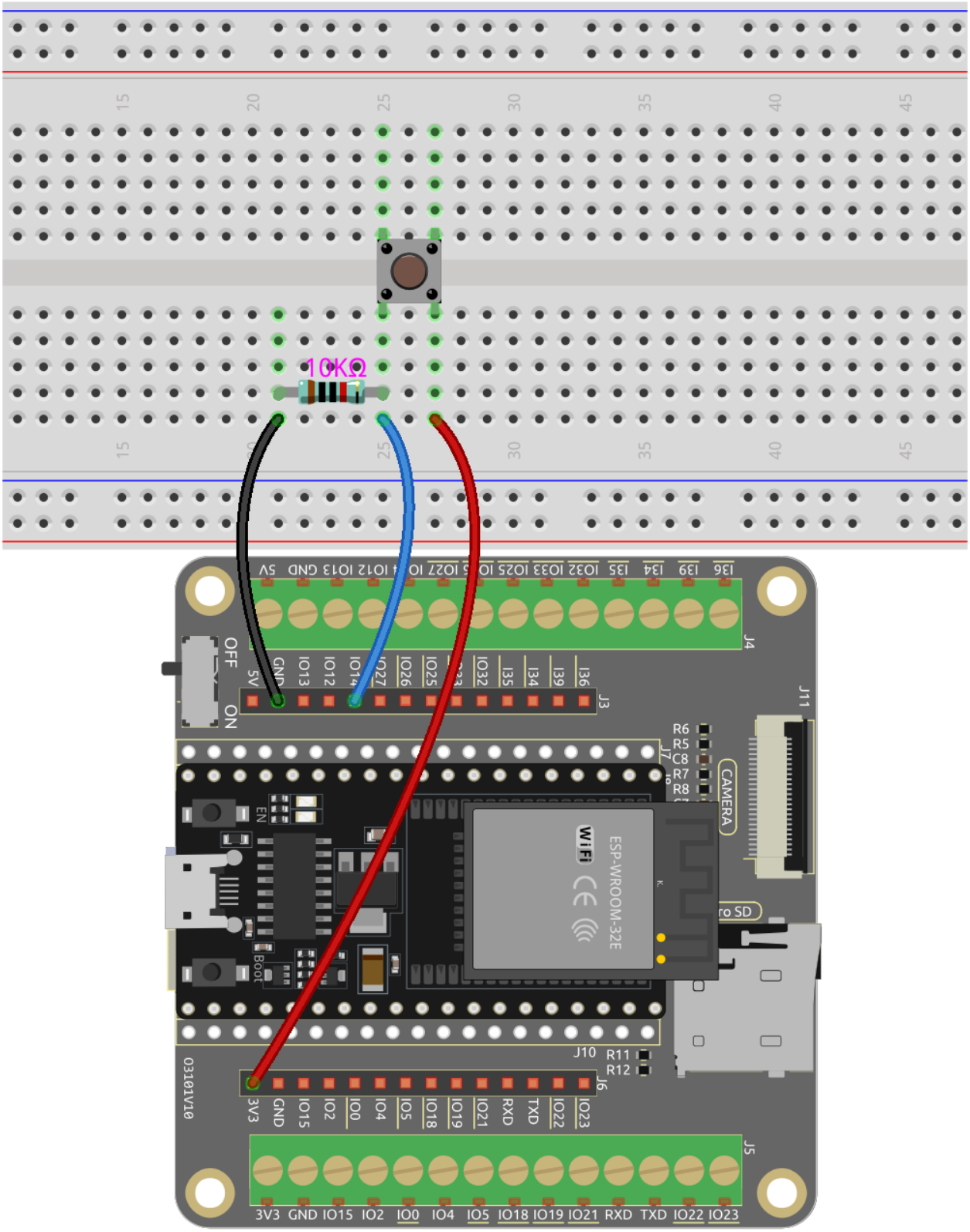


Internal Structure

Bauen Sie die Schaltung gemäß dem folgenden Diagramm auf.

- Verbinden Sie einen der Pins auf der linken Seite des Knopfes mit Pin14, der mit einem Pull-Down-Widerstand und einem 0,1uF (104) Kondensator verbunden ist (um Schwankungen zu eliminieren und ein stabiles Level auszugeben, wenn der Knopf betätigt wird).

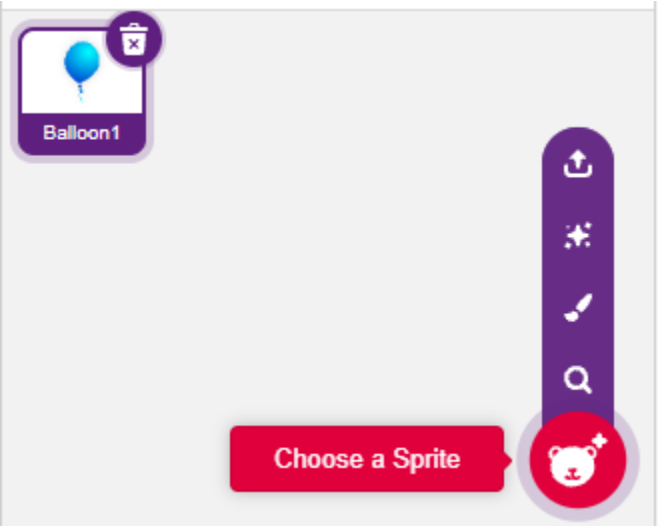
- Verbinden Sie das andere Ende des Widerstands und des Kondensators mit GND, und einen der Pins auf der rechten Seite des Knopfes mit 5V.



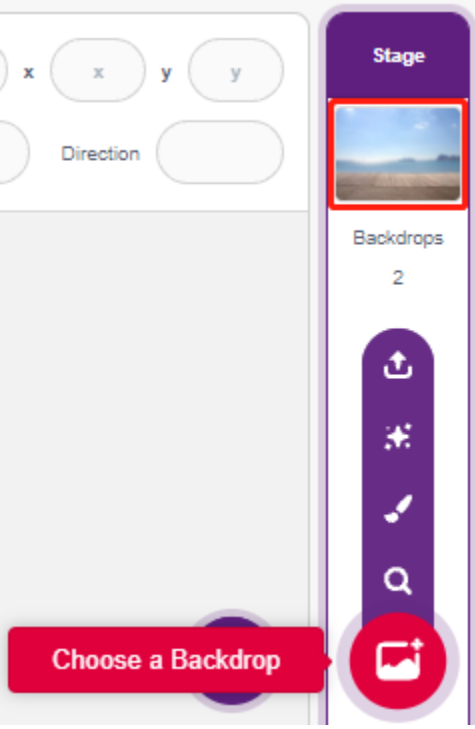
4.15.4 Programmierung

1. Ein Sprite und einen Hintergrund hinzufügen

Lösche das Standard-Sprite, klicke auf den **Choose a Sprite**-Button in der unteren rechten Ecke des Sprite-Bereichs und wähle das **Balloon1**-Sprite aus.



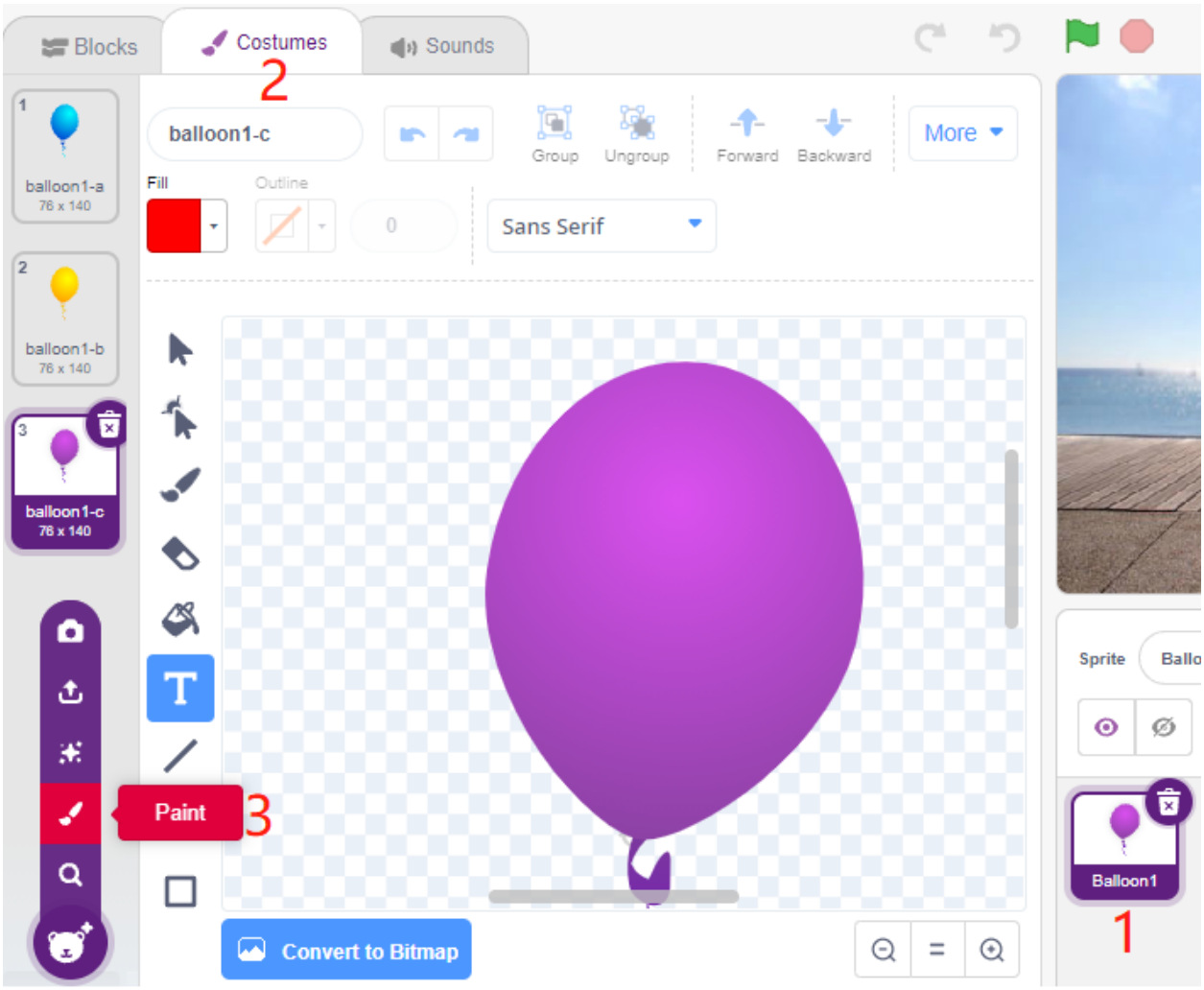
Füge einen **Boardwalk**-Hintergrund über den Button **Choose a backdrop** hinzu oder andere Hintergründe, die dir gefallen.



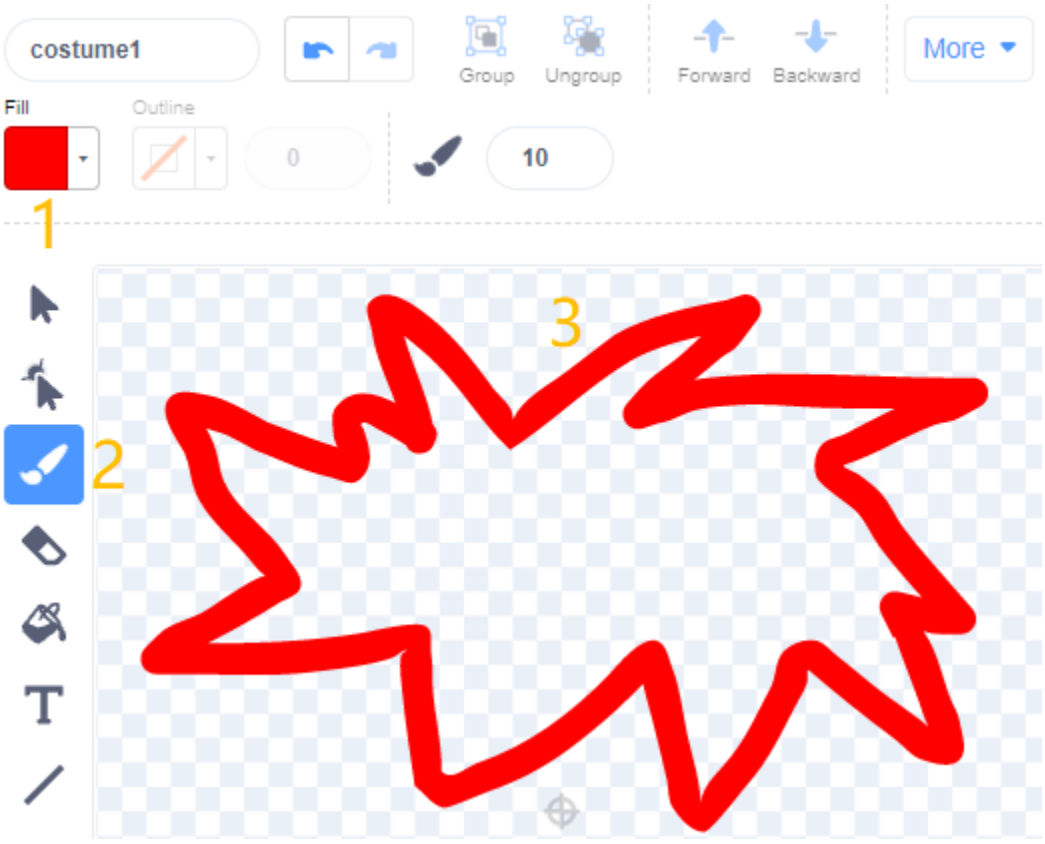
2. Ein Kostüm für das Ballon1-Sprite malen

Jetzt malen wir ein explodierendes Effektkostüm für das Ballon-Sprite.

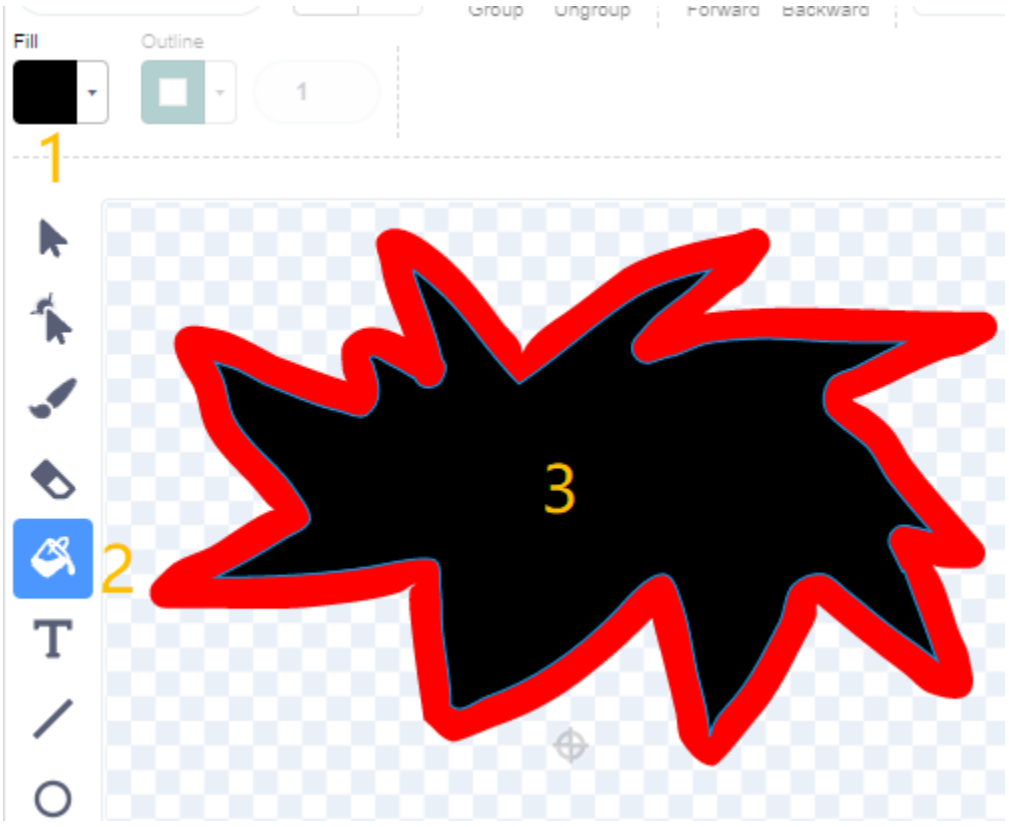
Gehe zur **Costumes**-Seite des **Balloon1**-Sprites, klicke auf den Button **Choose a Costume** in der unteren linken Ecke und wähle **Paint**, um ein leeres **Costume** zu erhalten.



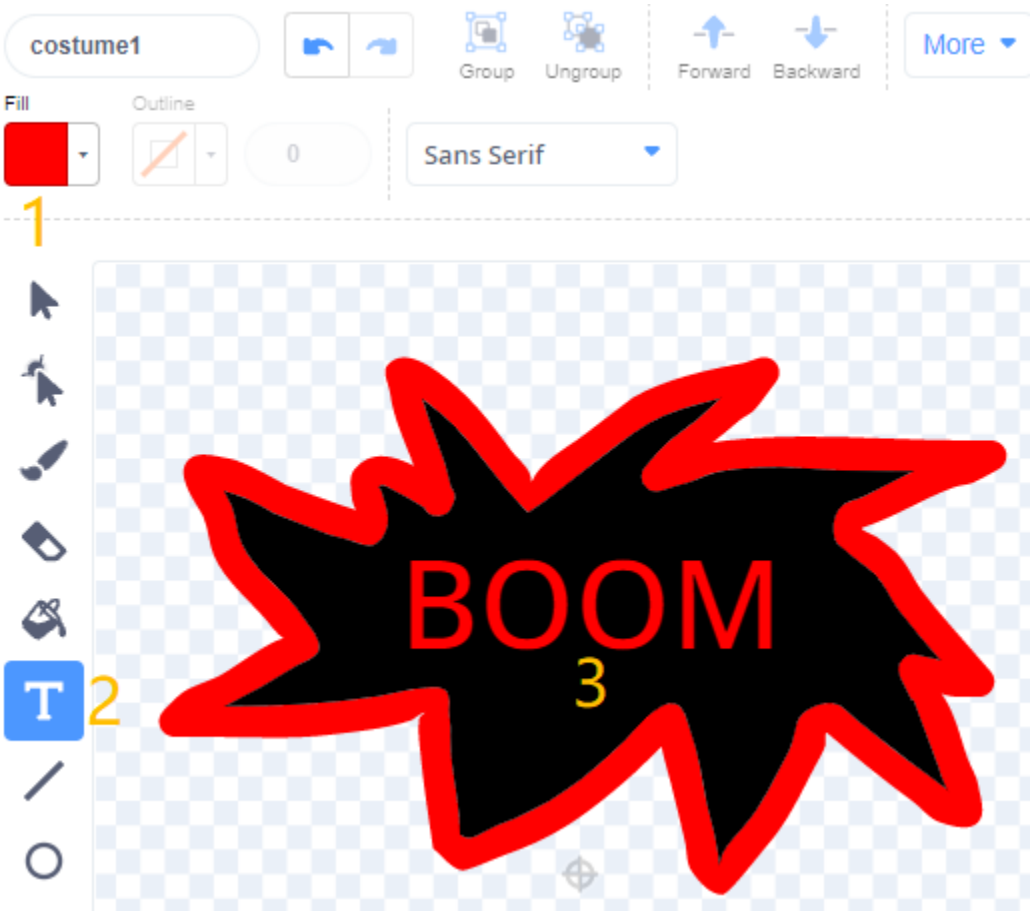
Wähle eine Farbe und benutze dann das **Brush**-Werkzeug, um ein Muster zu zeichnen.



Wähle erneut eine Farbe, klicke auf das Füllwerkzeug und bewege die Maus innerhalb des Musters, um es mit einer Farbe zu füllen.

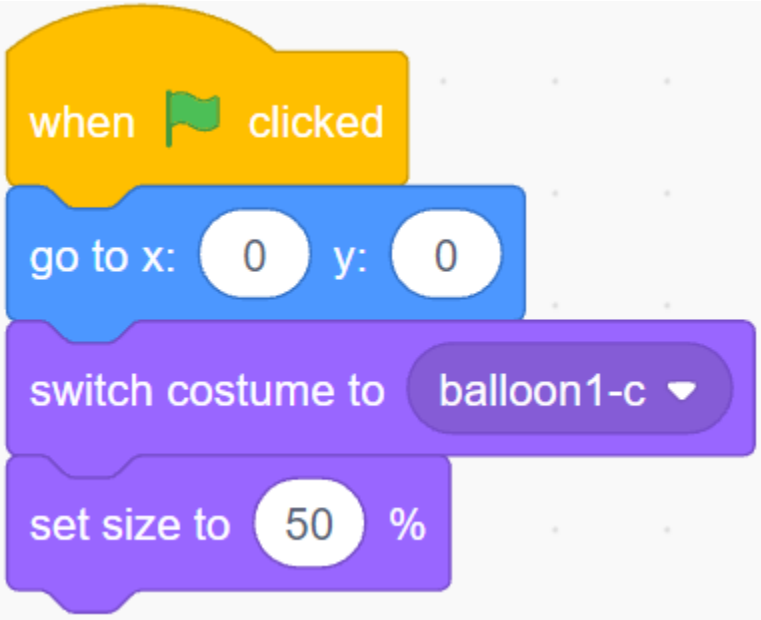


Schreibe abschließend den Text BOOM, damit ein Explosionseffektkostüm fertig ist.

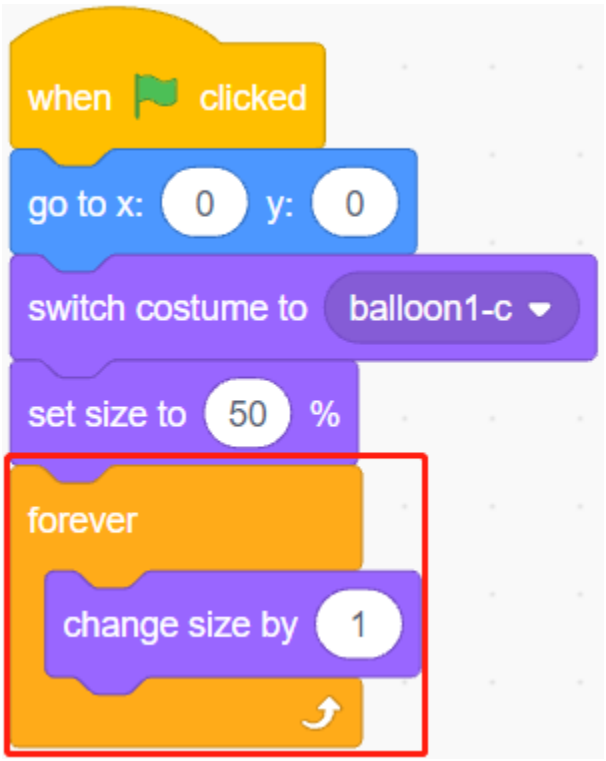


3. Skript für das Ballon-Sprite

Setze die Anfangsposition und Größe des **Balloon1**-Sprites.

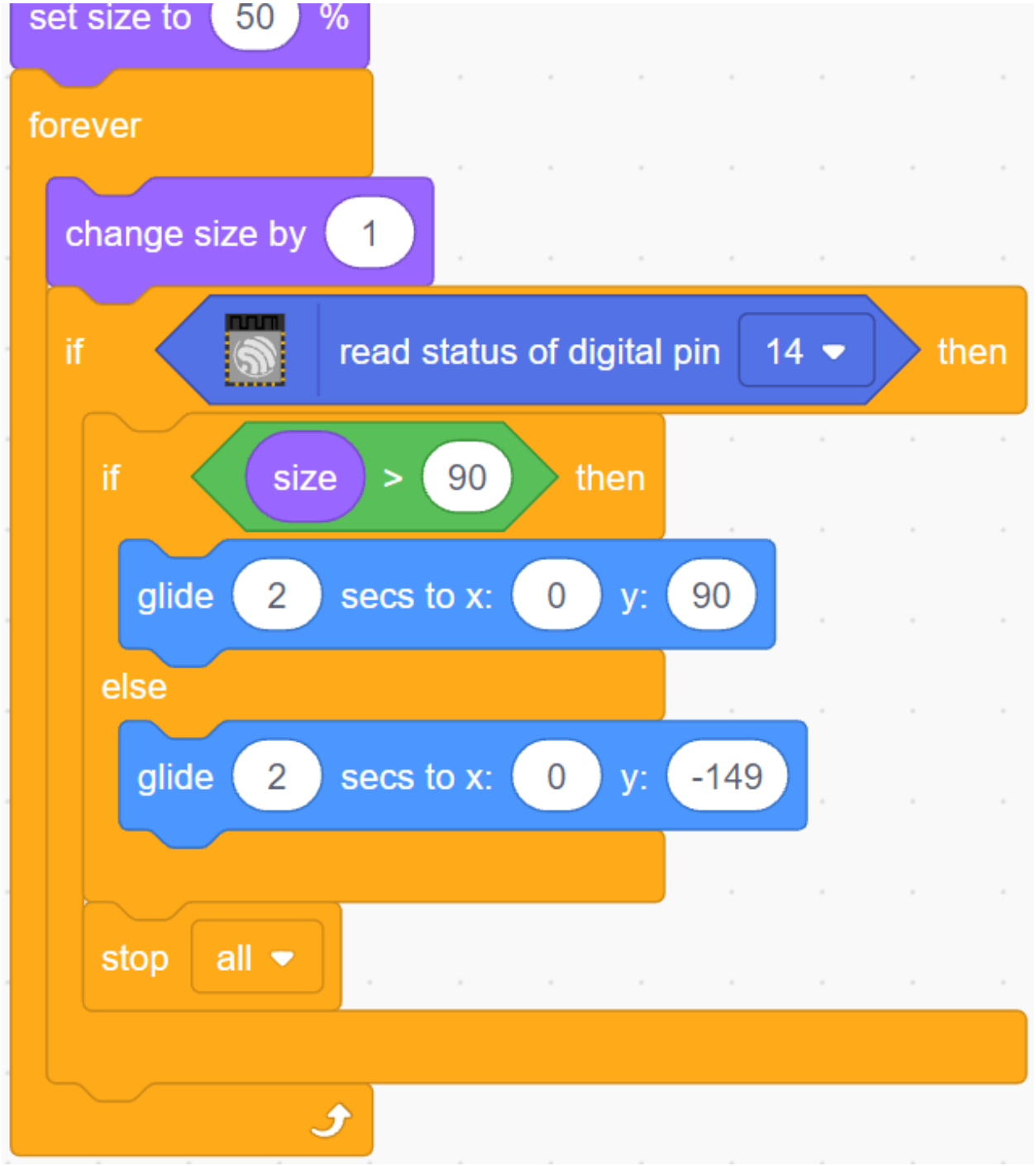


Lass dann das **Balloon**-Sprite langsam größer werden.

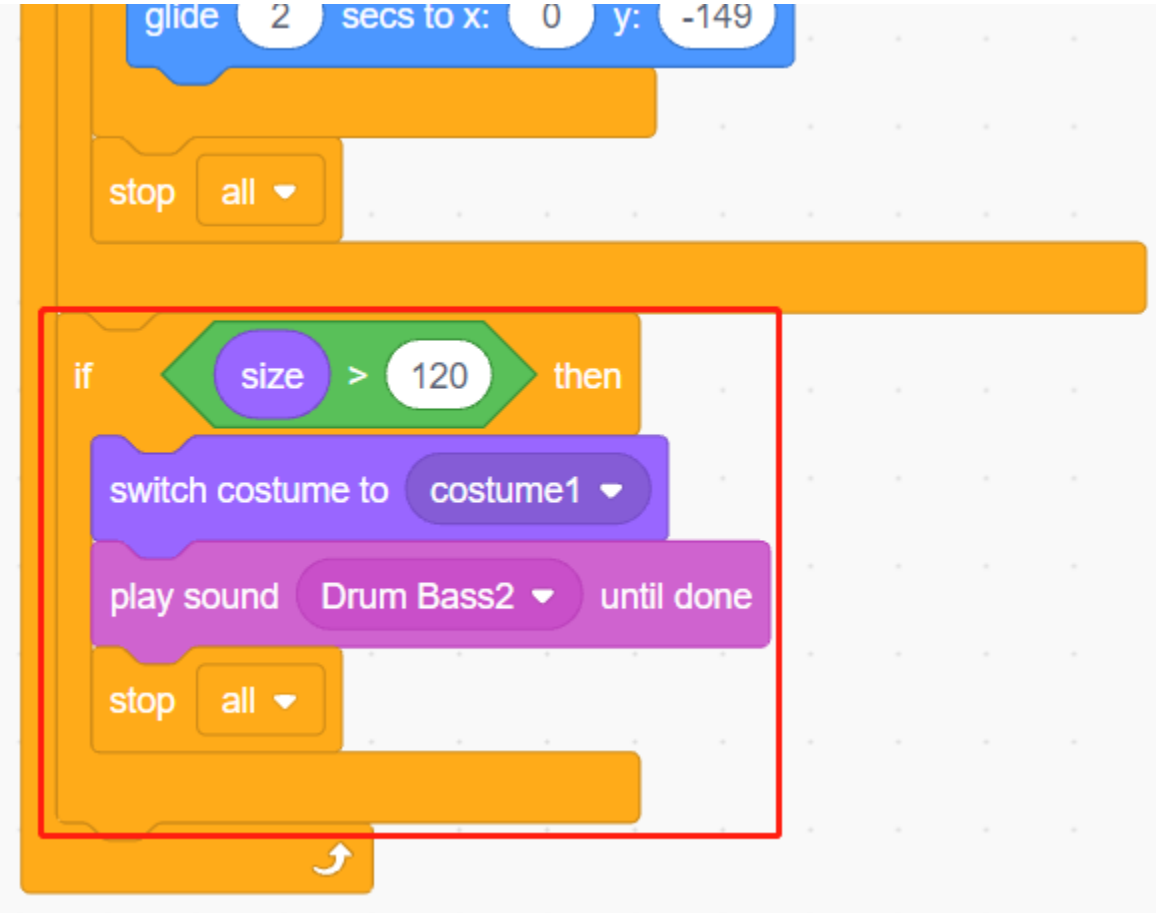


Wenn der Knopf gedrückt wird (Wert ist 1), hört das **Balloon1**-Sprite auf, größer zu werden.

- Wenn die Größe weniger als 90 beträgt, fällt es (y-Koordinate nimmt ab).
- Wenn die Größe größer als 90 und kleiner als 120 ist, fliegt es in den Himmel (y-Koordinate nimmt zu).



Wenn der Knopf nicht gedrückt wurde, wird der Ballon langsam größer und wenn die Größe größer als 120 ist, explodiert er (wechselt zum Explosionseffektkostüm).

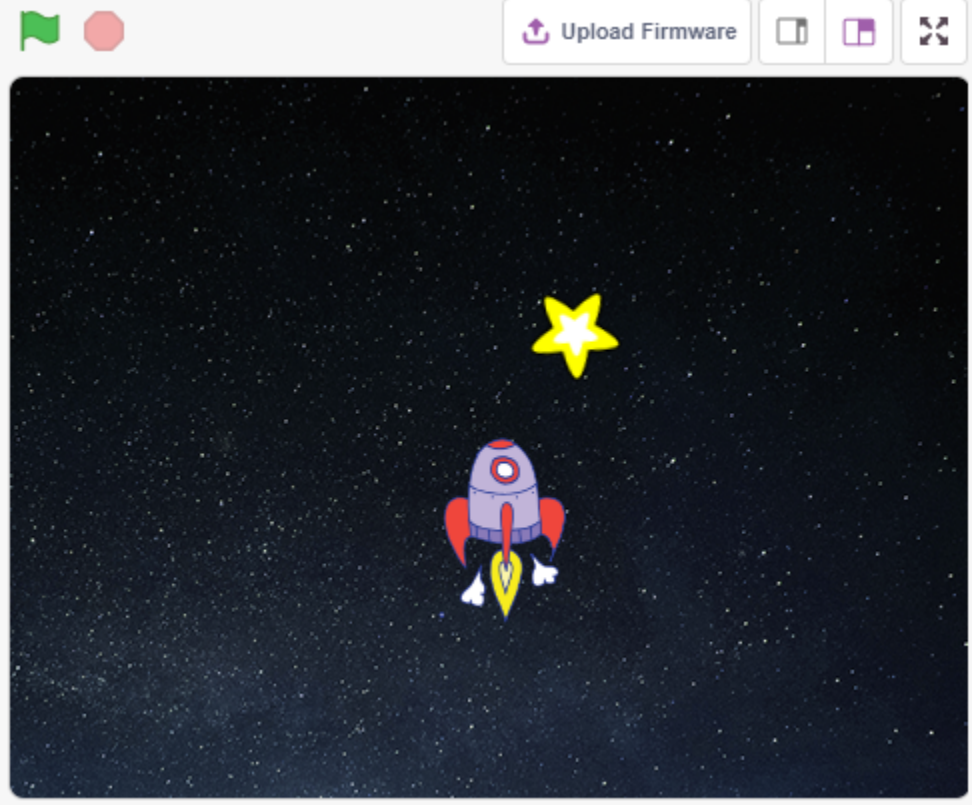


4.16 2.13 SPIEL - Sternenkreuzung

In den nächsten Projekten werden wir einige unterhaltsame Minispiele in PictoBlox spielen.

Hier verwenden wir das Joystick-Modul, um das Spiel Sternenkreuzung zu spielen.

Nachdem das Skript ausgeführt wurde, erscheinen Sterne zufällig auf der Bühne, und du musst den Joystick verwenden, um das Raumschiff zu steuern und den Sternen auszuweichen. Wenn du sie berührst, ist das Spiel vorbei.



4.16.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Joystick-Modul	

4.16.2 Was Sie Lernen Werden

- Funktionsweise des Joystick-Moduls
- Setzen der x- und y-Koordinaten des Sprites

4.16.3 Schaltung Aufbauen

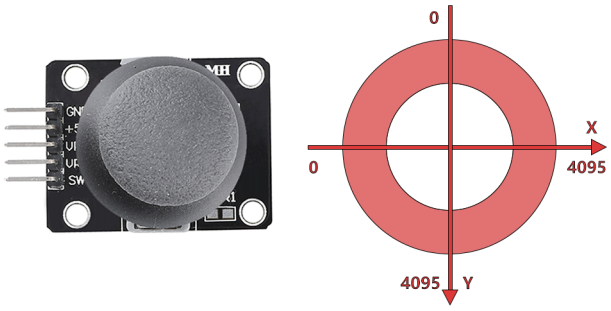
Ein Joystick ist ein Eingabegerät, das aus einem auf einer Basis schwenkbaren Stick besteht und dessen Winkel oder Richtung an das Gerät meldet, das er steuert. Joysticks werden häufig verwendet, um Videospiele und Roboter zu steuern.

Um einem Computer einen vollen Bewegungsbereich zu vermitteln, muss ein Joystick die Position des Sticks auf zwei Achsen messen - der X-Achse (links nach rechts) und der Y-Achse (oben nach unten).

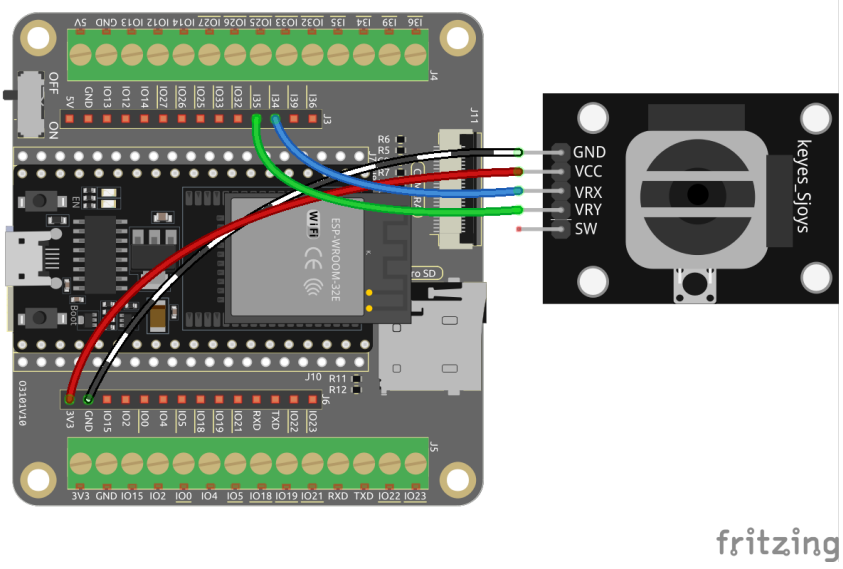
Die Bewegungskoordinaten des Joysticks werden in der folgenden Abbildung gezeigt.

Bemerkung:

- Die x-Koordinate verläuft von links nach rechts, der Bereich ist 0-4095.
- Die y-Koordinate verläuft von oben nach unten, Bereich ist 0-4095.



Bauen Sie die Schaltung gemäß dem folgenden Diagramm auf.

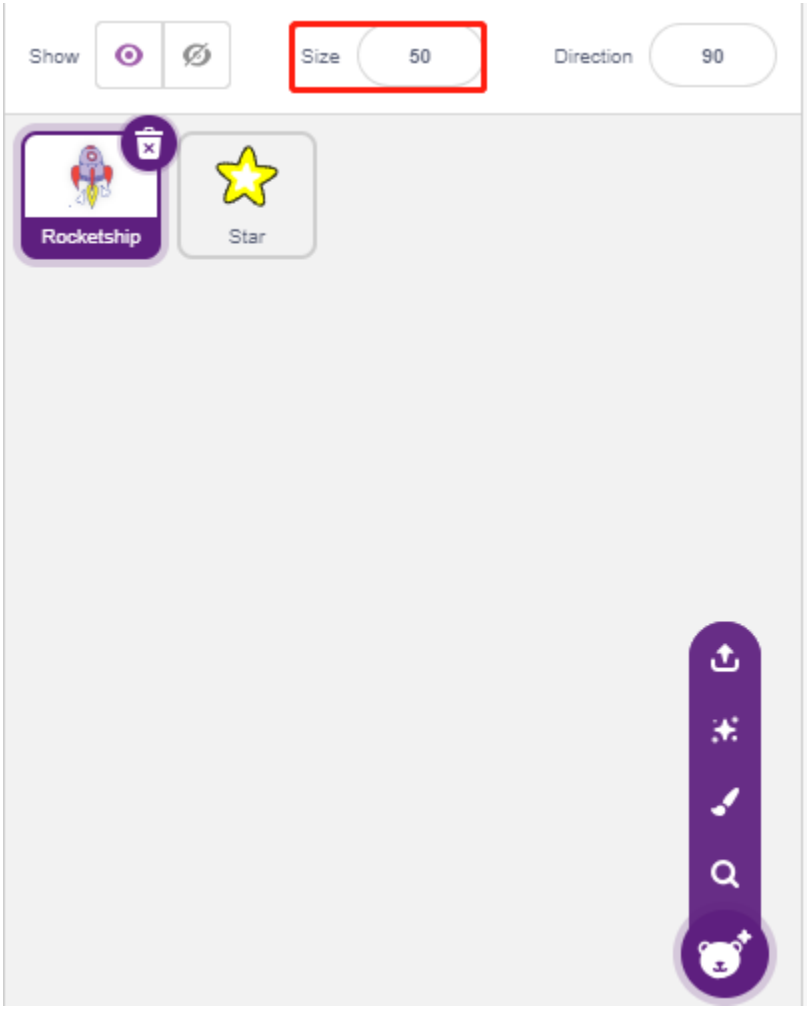


4.16.4 Programmierung

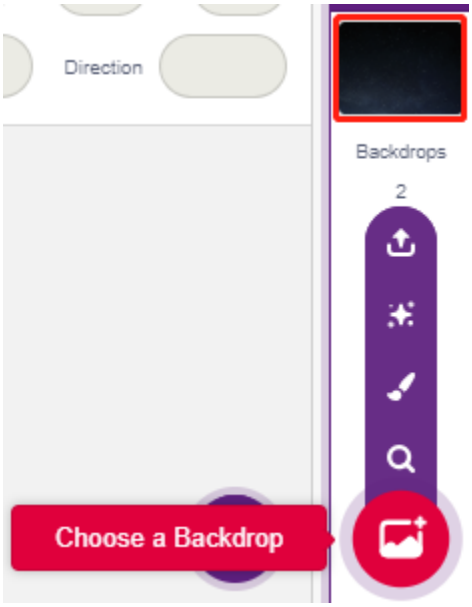
Das gesamte Skript soll den Effekt erzielen, dass beim Klicken auf die grüne Flagge das **Stars**-Sprite in einer Kurve auf der Bühne bewegt wird und du den Joystick verwenden musst, um das **Rocketship** zu bewegen, sodass es nicht vom **Star**-Sprite berührt wird.

1. Sprites und Hintergründe hinzufügen

Lösche das Standard-Sprite und füge über den Button **Choose a Sprite** das **Rocketship**-Sprite und das **Star**-Sprite hinzu. Beachte, dass die Größe des **Rocket**-Sprites auf 50% gesetzt wird.



Füge nun den **Stars**-Hintergrund über **Choose a Backdrop** hinzu.

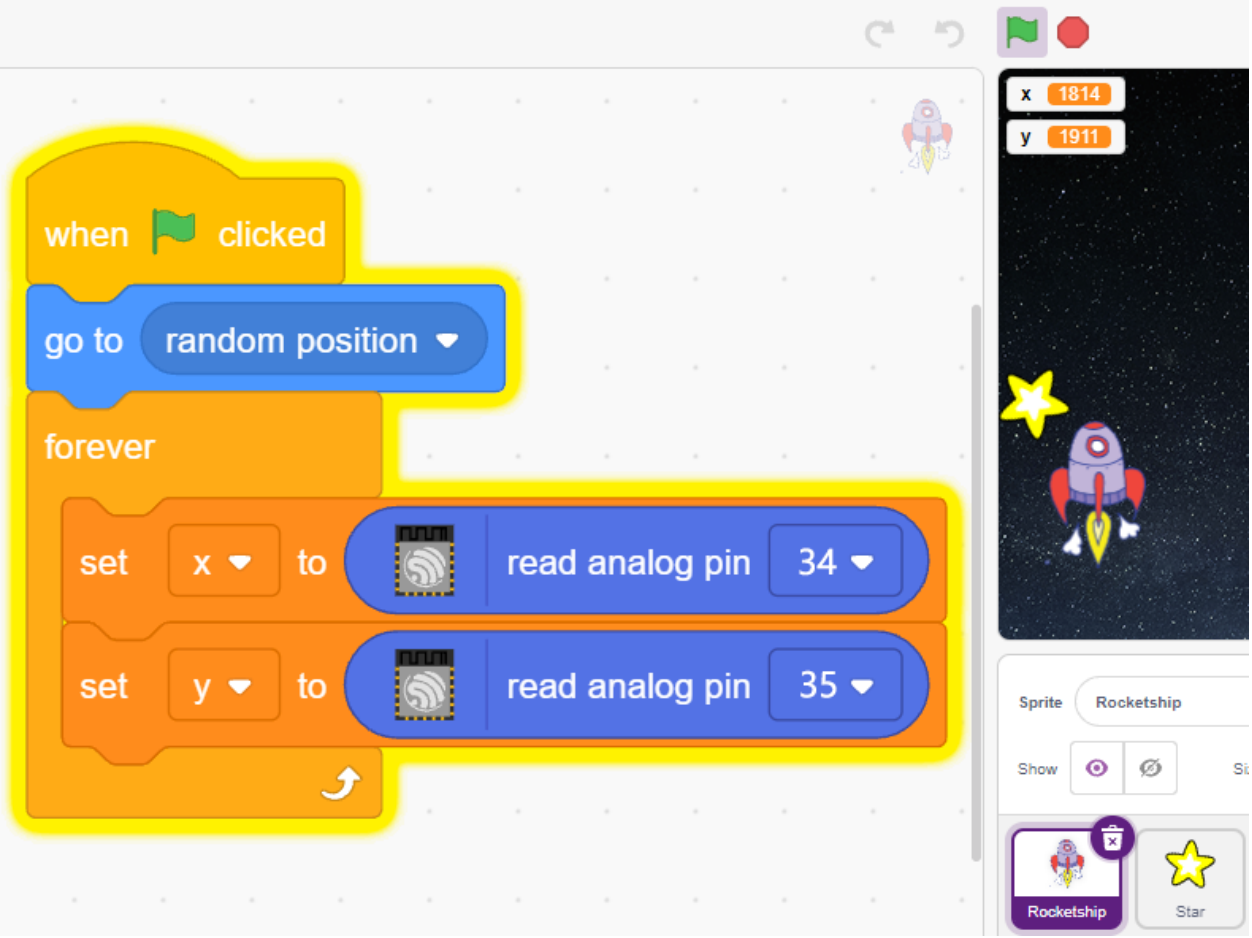


2. Skript für Raumschiff

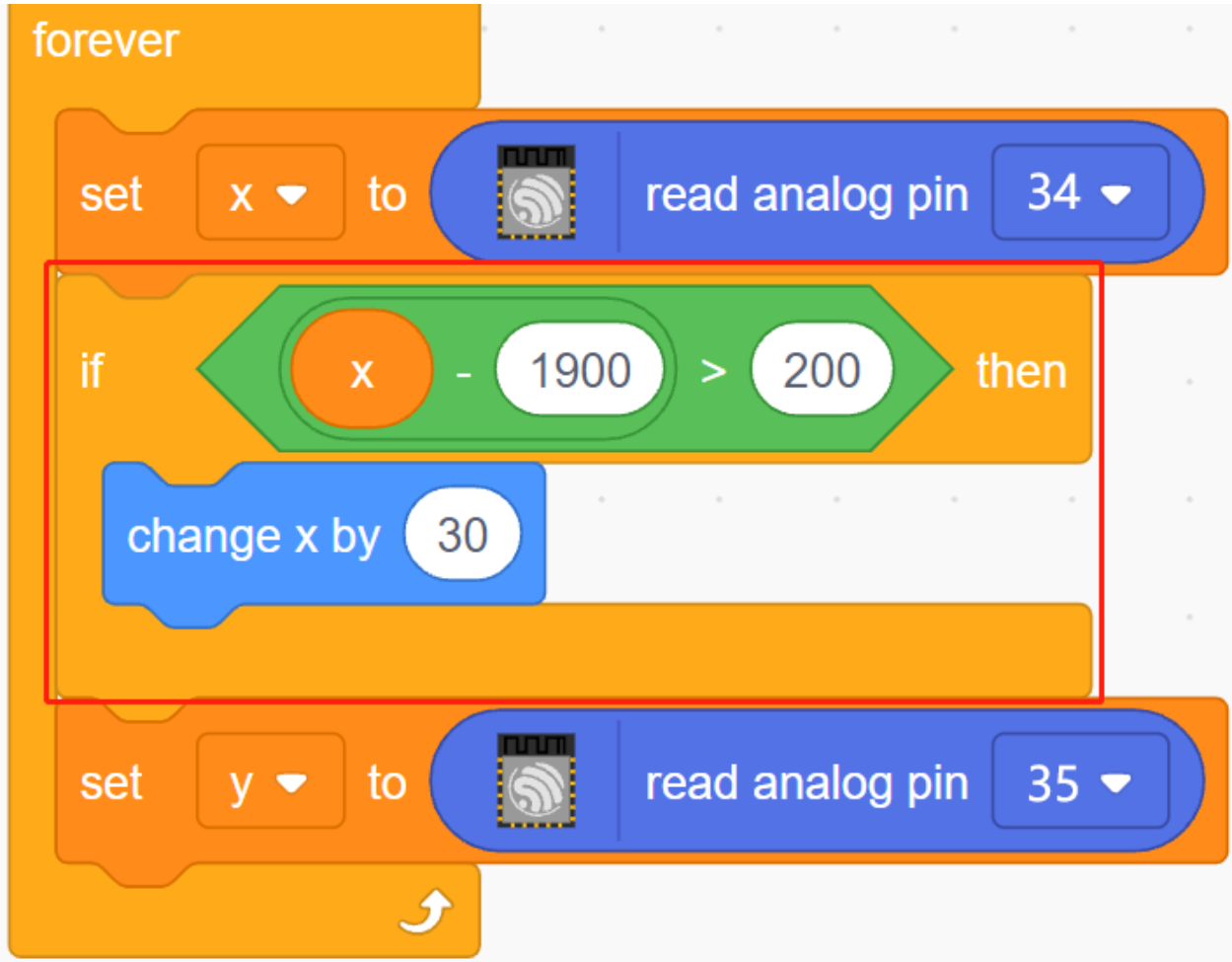
Das **Rocketship**-Sprite soll so programmiert werden, dass es an einer zufälligen Position erscheint und dann durch den Joystick gesteuert nach oben, unten, links und rechts bewegt wird.

Der Arbeitsablauf ist wie folgt.

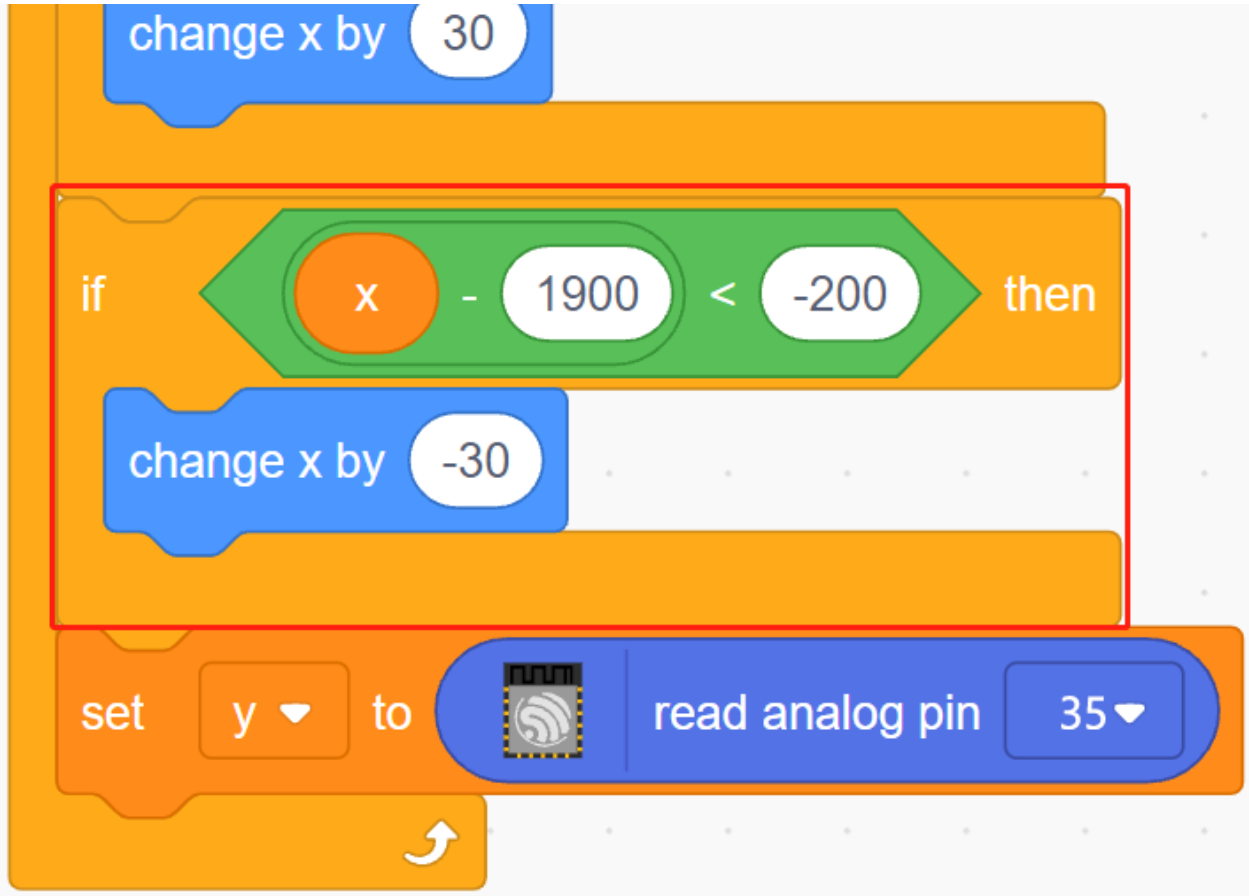
- Wenn die grüne Flagge angeklickt wird, gehe das Sprite zu einer zufälligen Position und erstelle 2 Variablen **x** und **y**, die die Werte von Pin33 (VRX des Joysticks) und Pin35 (VRY des Joysticks) speichern. Du kannst das Skript laufen lassen, den Joystick nach oben und unten, links und rechts bewegen, um den Wertebereich von x und y zu sehen.



- Der Wert von Pin33 liegt im Bereich 0-4095 (die Mitte ist etwa 1800). Verwende `x-1800>200`, um zu bestimmen, ob der Joystick nach rechts bewegt wird, und wenn ja, erhöhe die x-Koordinate des Sprites um +30 (um das Sprite nach rechts zu bewegen).



- Wenn der Joystick nach links bewegt wird, lasse die x-Koordinate des Sprites -30 sein (lass das Sprite nach links bewegen).



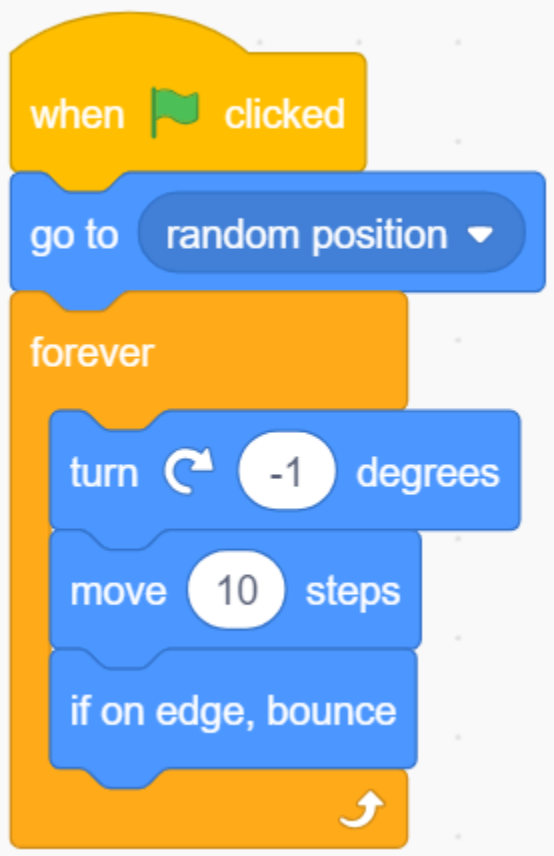
- Da die y-Koordinate des Joysticks von oben (0) nach unten (4095) verläuft und die y-Koordinate des Sprites von unten nach oben. Um also den Joystick nach oben und das Sprite nach oben zu bewegen, muss die y-Koordinate im Skript -30 sein.

- Wenn der Joystick nach unten bewegt wird, ist die y-Koordinate des Sprites +30.

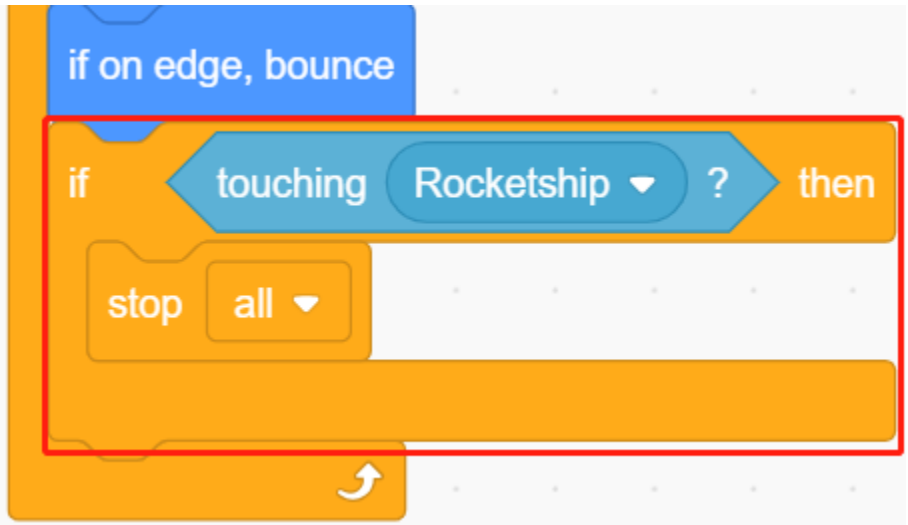
3. Skript für Stern

Das zu erzielende Ergebnis des **Star**-Sprites ist, dass es an einer zufälligen Stelle erscheint, und wenn es **Rocketship** trifft, stoppt das Skript und das Spiel endet.

- Wenn die grüne Flagge angeklickt wird und das Sprite zu einer zufälligen Position geht, ist der [turn degrees]-Block dafür gedacht, das **Star**-Sprite mit einer kleinen Winkeländerung vorwärts zu bewegen, sodass du sehen kannst, dass es in einer Kurve bewegt wird und wenn es am Rand ist, abprallt.



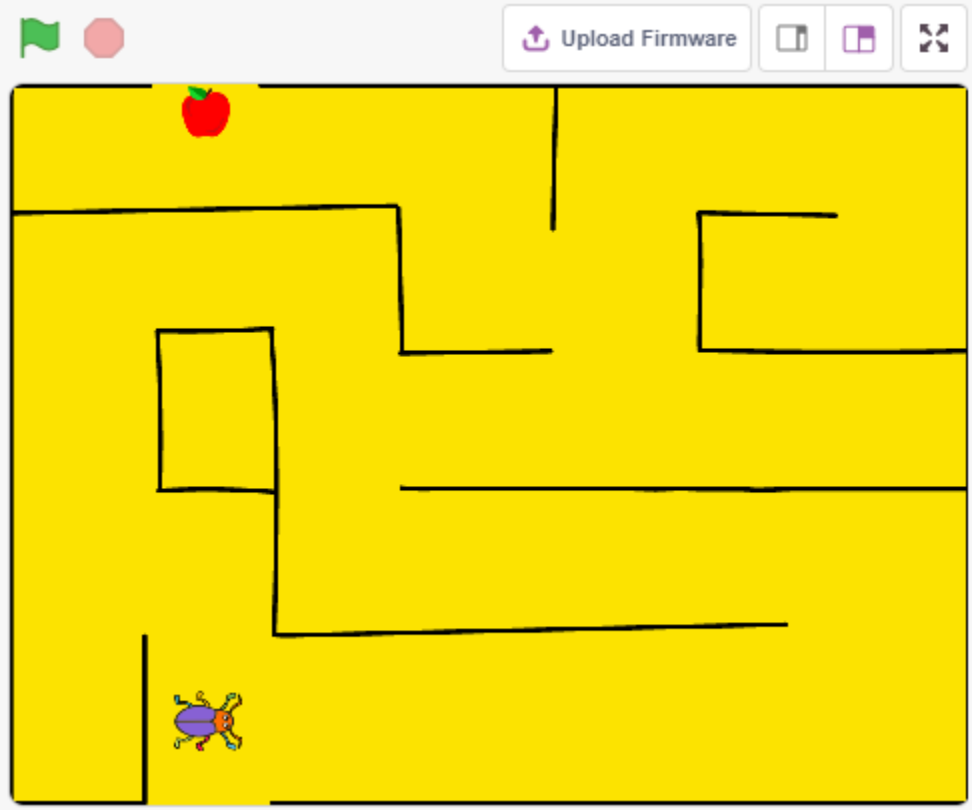
- Wenn das Sprite das **Rocketship**-Sprite während der Bewegung berührt, stoppe das Skript.



4.17 2.14 SPIEL - Apfel Essen

In diesem Projekt spielen wir ein Spiel, bei dem wir einen Knopf verwenden, um den Käfer zu steuern, damit er Äpfel isst.

Wenn die grüne Flagge angeklickt wird, drücke den Knopf und der Käfer wird sich drehen. Drücke den Knopf erneut und der Käfer hält an und bewegt sich in diesem Winkel vorwärts. Du musst den Winkel des Käfers so steuern, dass er sich vorwärts bewegt, ohne die schwarze Linie auf der Karte zu berühren, bis er den Apfel isst. Wenn er die schwarze Linie berührt, ist das Spiel vorbei.



4.17.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

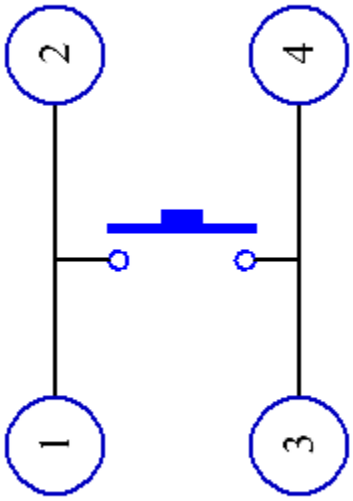
Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Taste	

4.17.2 Schaltung Aufbauen

Der Knopf ist ein 4-poliges Gerät, da Pin 1 mit Pin 2 verbunden ist und Pin 3 mit Pin 4, wenn der Knopf gedrückt wird, sind die 4 Pins verbunden, wodurch der Stromkreis geschlossen wird.

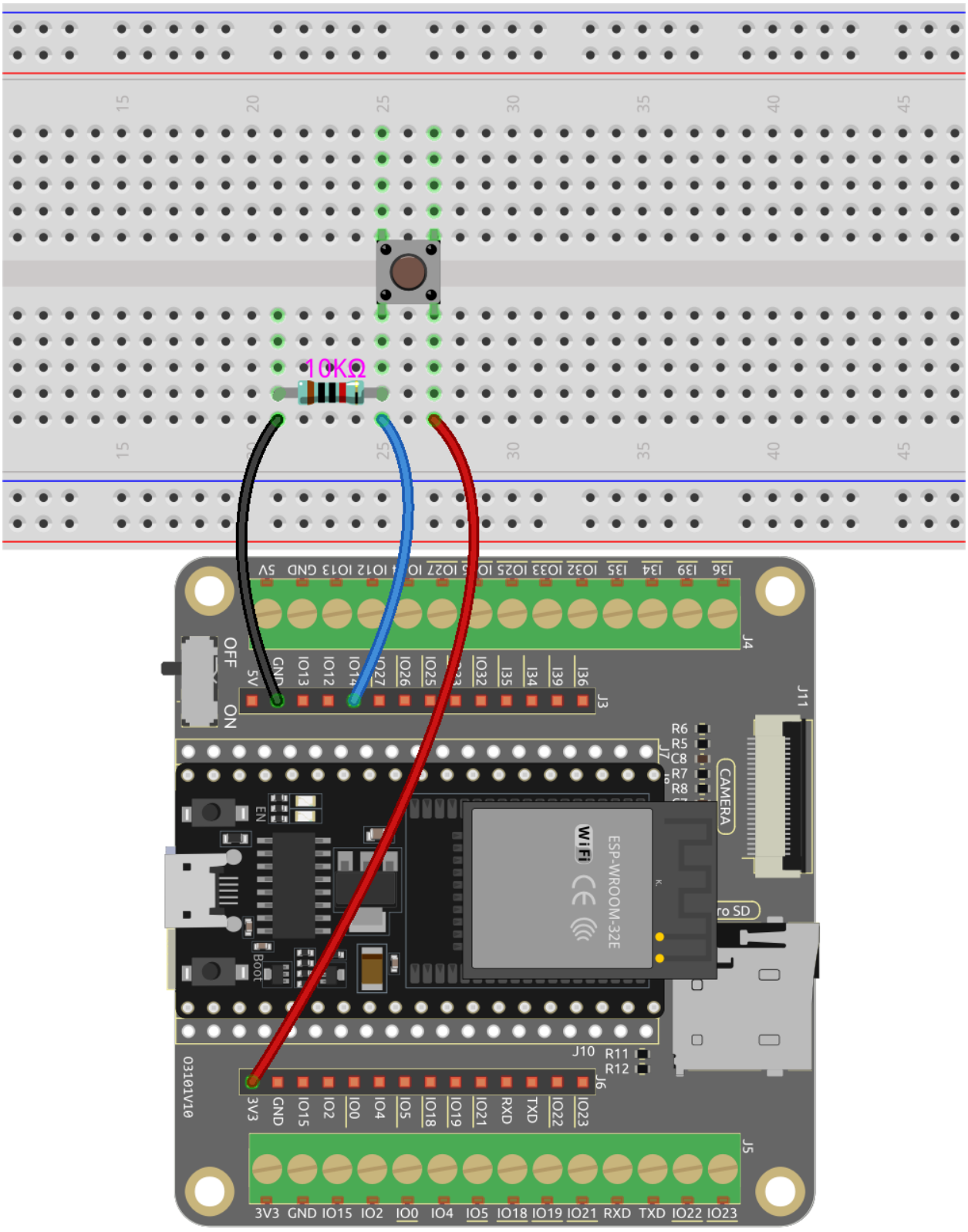
Button



Internal Structure

Bauen Sie die Schaltung gemäß dem folgenden Diagramm auf.

- Verbinden Sie einen der Pins auf der linken Seite des Knopfes mit Pin14, der mit einem Pull-Down-Widerstand und einem 0,1uF (104) Kondensator verbunden ist (um Schwankungen zu eliminieren und ein stabiles Level auszugeben, wenn der Knopf betätigt wird).
- Verbinden Sie das andere Ende des Widerstands und des Kondensators mit GND, und einen der Pins auf der rechten Seite des Knopfes mit 5V.


10KΩ
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23
ESP-WROOM-32E
CAMERA

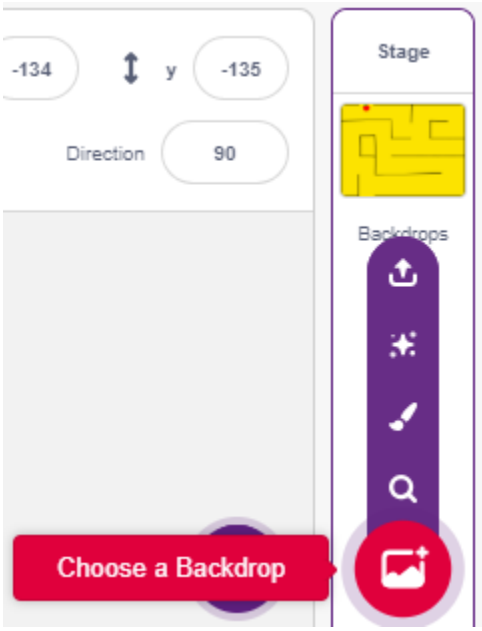
4.17.3 Programmierung

Das Ziel ist es, den Knopf zu verwenden, um die Richtung des **Beetle**-Sprites zu steuern, damit es vorwärts geht und den Apfel isst, ohne die schwarze Linie auf dem **Maze**-Hintergrund zu berühren. Wenn der Apfel gegessen wird, wird der Hintergrund gewechselt.

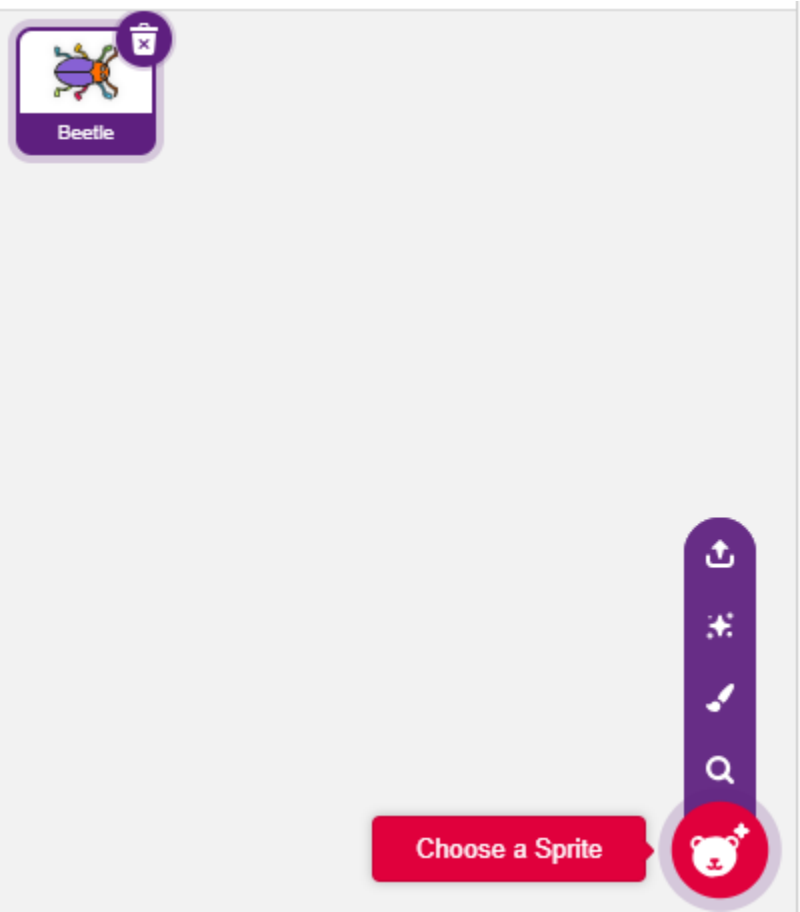
Füge nun die relevanten Hintergründe und Sprites hinzu.

1. Hintergründe und Sprites hinzufügen

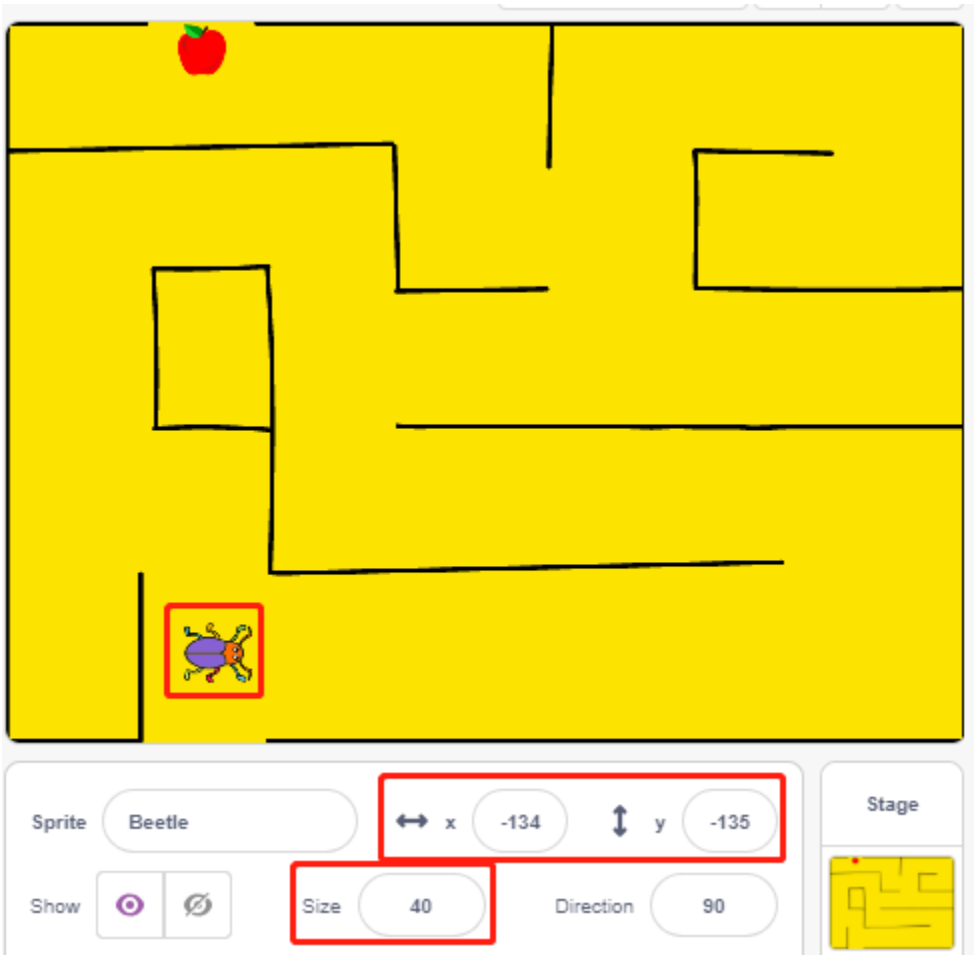
Füge einen **Maze**-Hintergrund über den Button **Choose a backdrop** hinzu.



Lösche das Standard-Sprite und wähle dann das **Beetle**-Sprite aus.



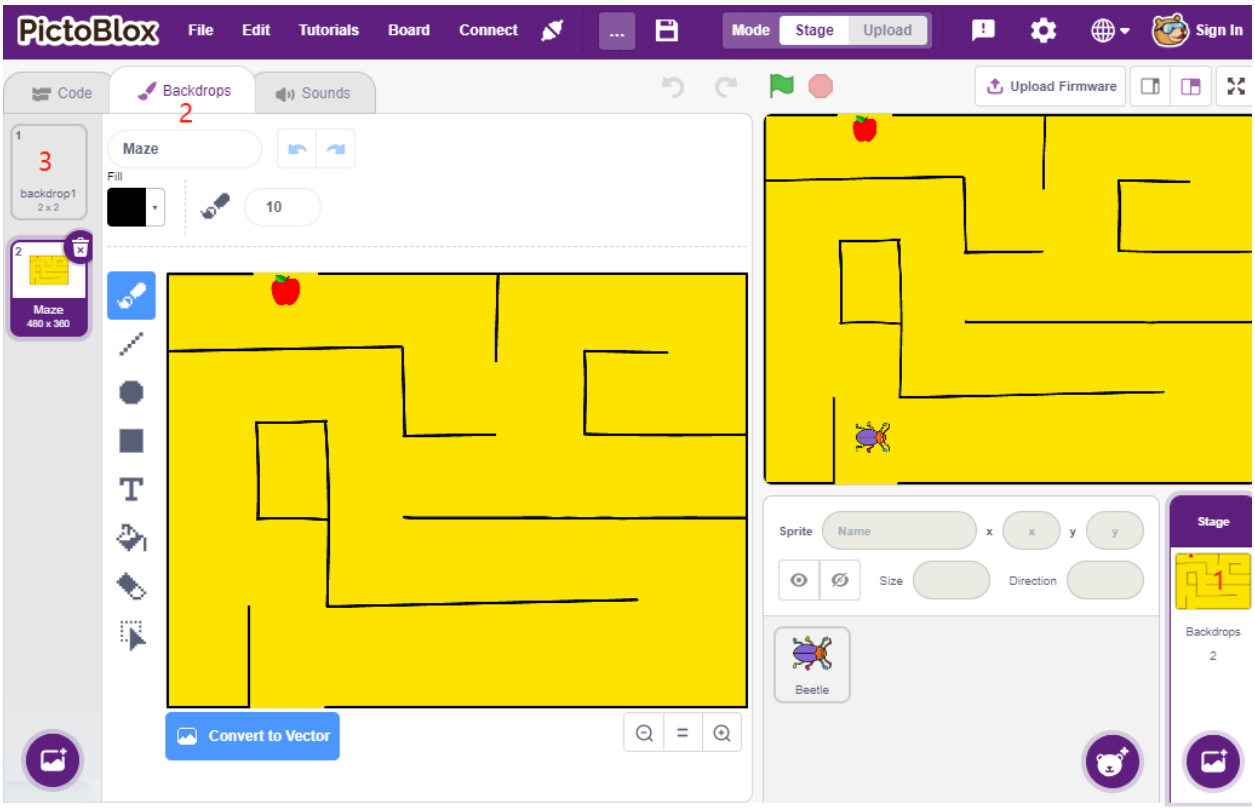
Platziere das **Beetle**-Sprite am Eingang des **Maze**-Hintergrunds, merke dir die x,y-Koordinatenwerte an diesem Punkt und ändere die Größe des Sprites auf 40%.



2. Einen Hintergrund zeichnen

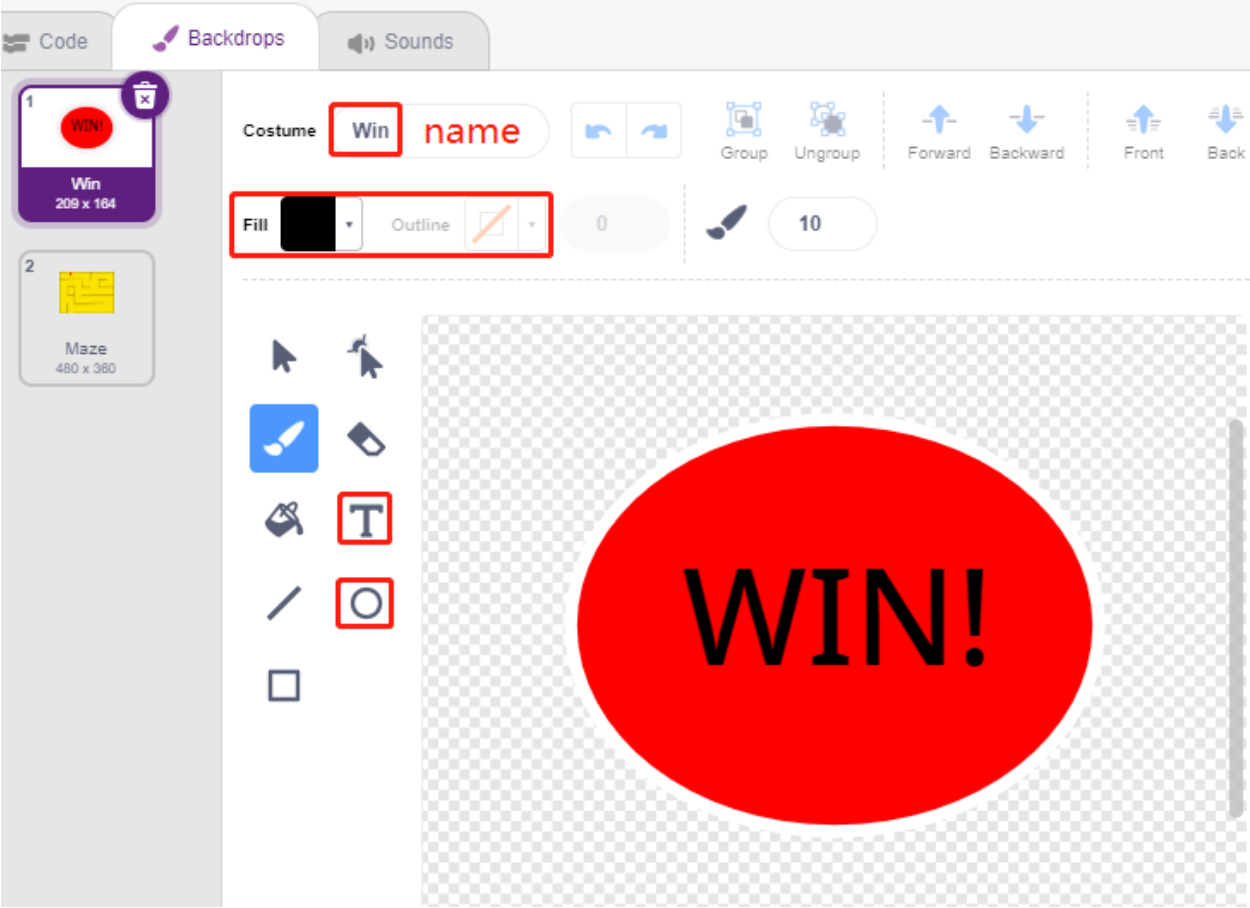
Jetzt ist es an der Zeit, einen einfachen Hintergrund mit dem WIN!-Zeichen darauf zu zeichnen.

Klicke zuerst auf die Miniaturansicht des Hintergrunds, um zur **Backdrops**-Seite zu gelangen und klicke auf den leeren Hintergrund1.



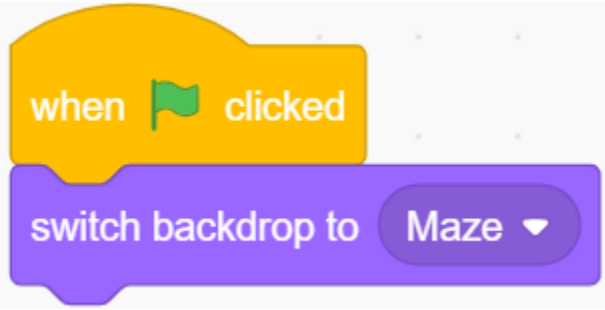
Beginne jetzt mit dem Zeichnen, du kannst das Bild unten als Referenz verwenden oder deinen eigenen Hintergrund zeichnen, solange der Ausdruck gewinnend ist.

- Verwende das **Circle**-Werkzeug, um eine Ellipse mit der Farbe Rot und ohne Umrandung zu zeichnen.
- Dann verwende das **Text**-Werkzeug, schreibe das Zeichen "WIN!", setze die Zeichenfarbe auf Schwarz und passe die Größe und Position des Zeichens an.
- Benenne den Hintergrund als **Win**.



3. Skript für den Hintergrund

Der Hintergrund muss jedes Mal, wenn das Spiel beginnt, auf **Maze** umgeschaltet werden.



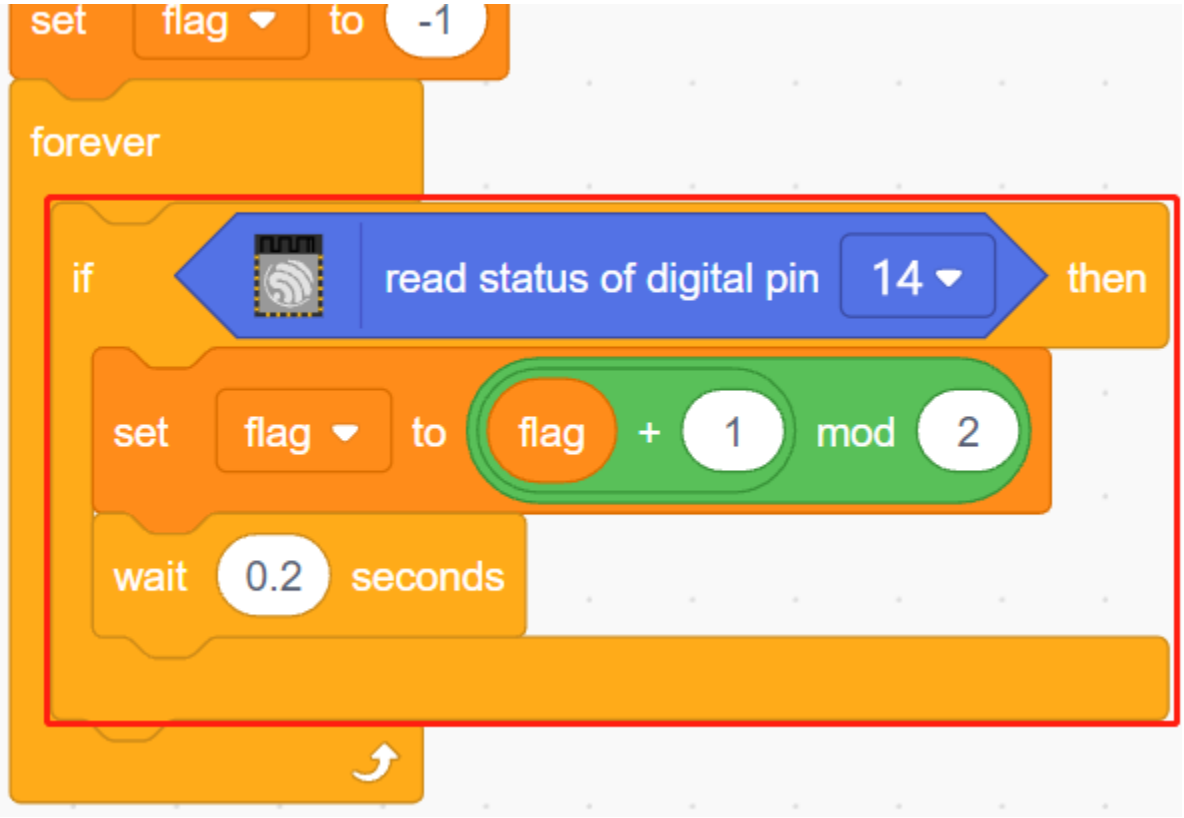
4. Skripte für das Sprite Käfer schreiben

Schreibe jetzt ein Skript für das Sprite **Beetle**, um vorwärtsbewegen und die Richtung unter Kontrolle eines Knopfes ändern zu können. Der Arbeitsablauf ist wie folgt.

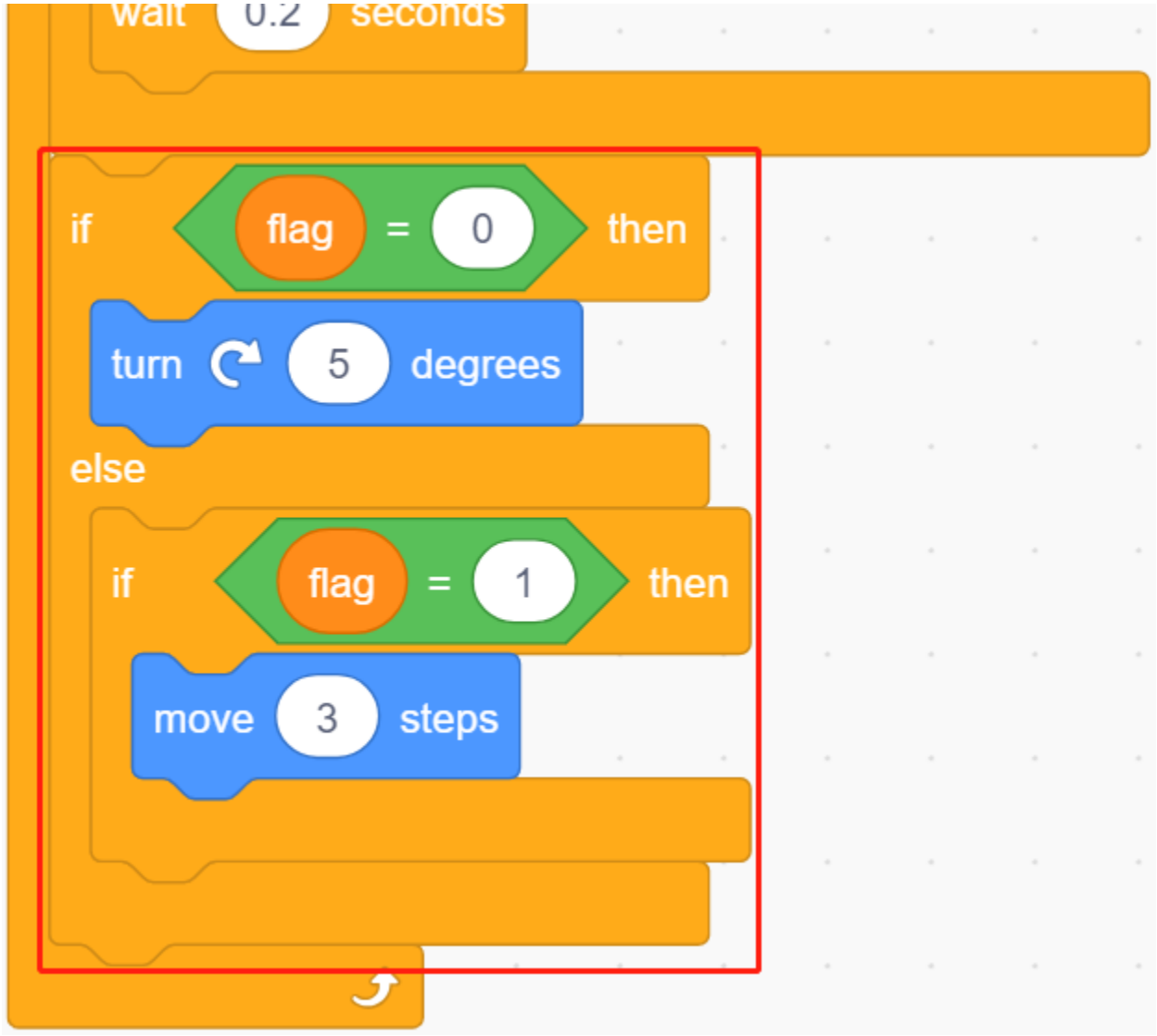
- Wenn die grüne Flagge angeklickt wird, setze den Winkel des **Beetle** auf 90 und die Position auf (-134, -134) oder ersetze sie durch den Koordinatenwert deiner eigenen Platzierung. Erstelle die Variable **flag** und setze den Anfangswert auf -1.

Als Nächstes werden im [forever]-Block vier [if]-Blöcke verwendet, um verschiedene mögliche Szenarien zu bestimmen.

- Wenn der Knopf 1 ist (gedrückt), verwende den [mod]-Block, um den Wert der Variable **flag** zwischen 0 und 1 umzuschalten (abwechselnd zwischen 0 für diesen Druck und 1 für den nächsten Druck).

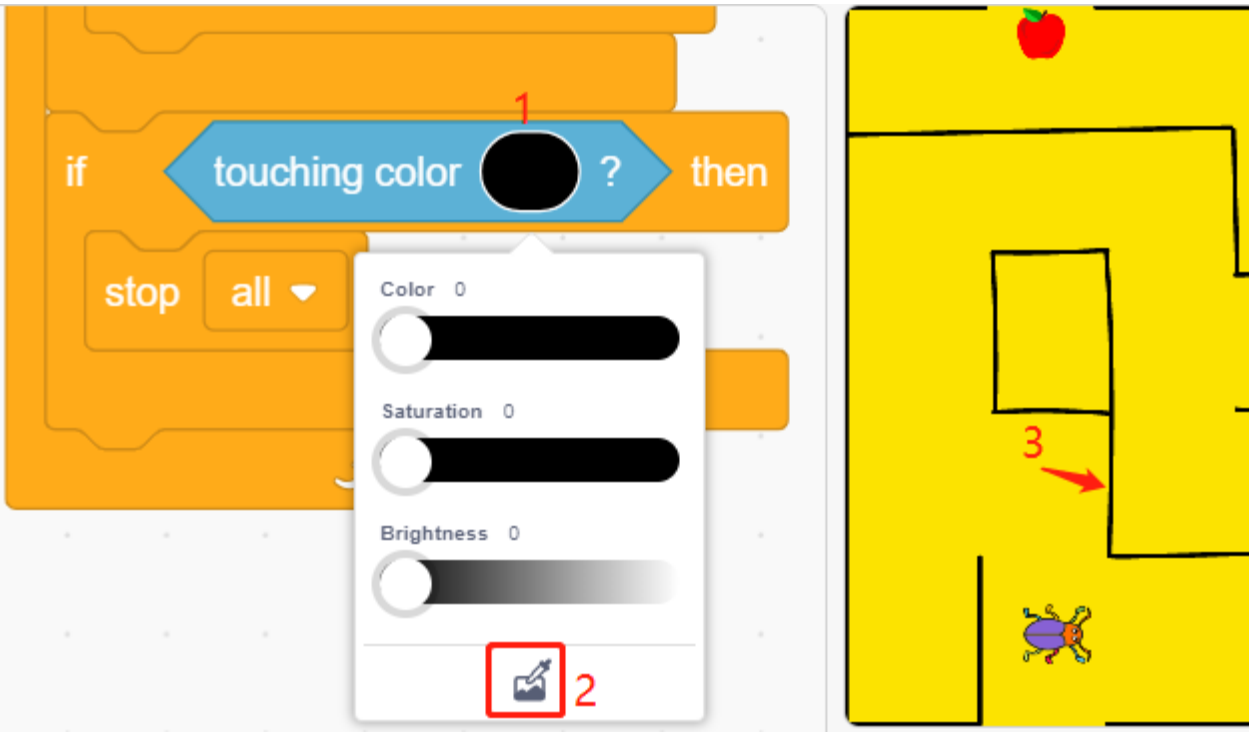


- Wenn Flag=0 (dieser Knopfdruck), lasse das **Beetle**-Sprite sich im Uhrzeigersinn drehen. Dann bestimme, ob Flag gleich 1 ist (Knopf erneut gedrückt), das **Beetle**-Sprite bewegt sich vorwärts. Andernfalls dreht es sich weiter im Uhrzeigersinn.

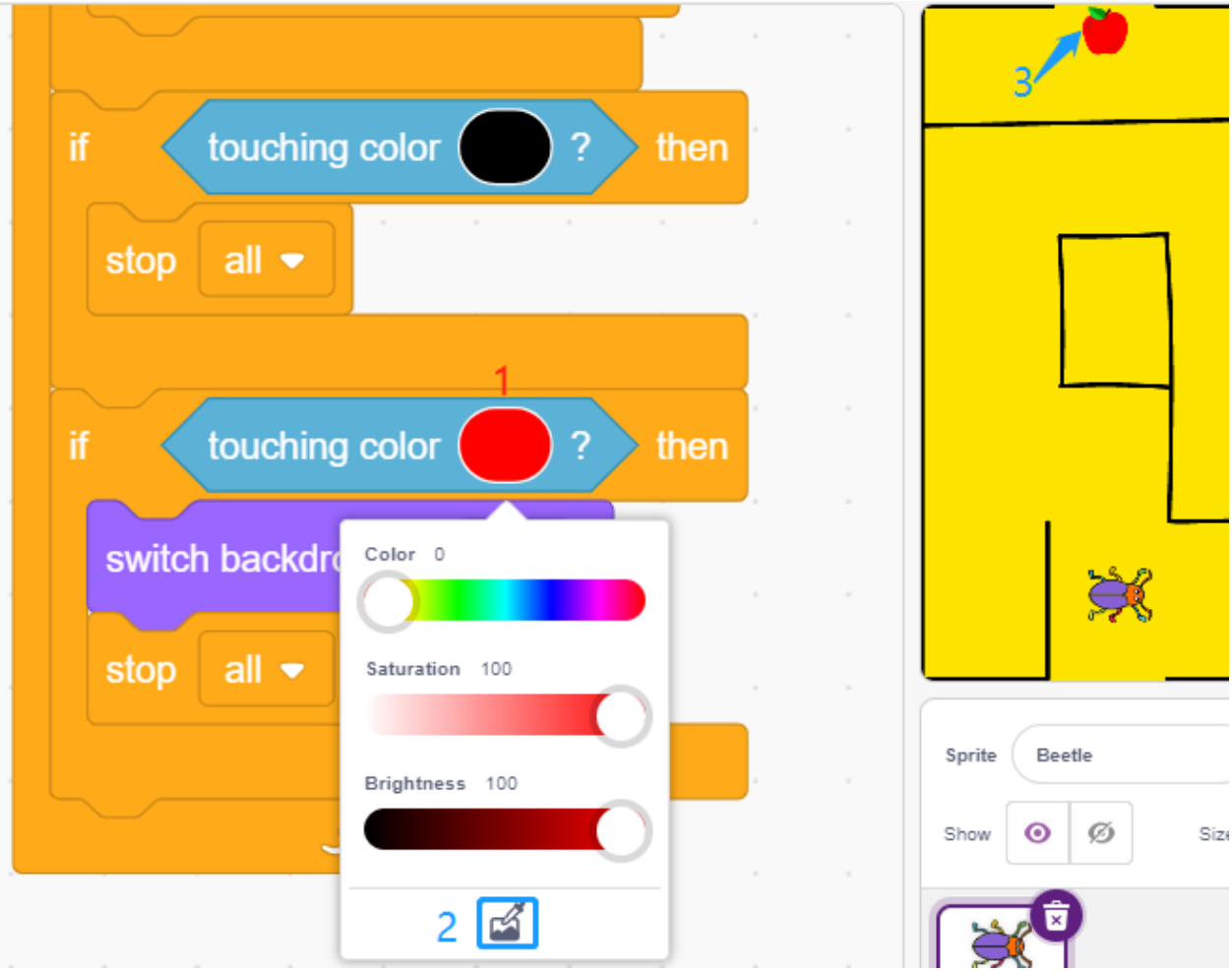


- Wenn das Käfer-Sprite Schwarz berührt (die schwarze Linie auf dem **Maze**-Hintergrund), endet das Spiel und das Skript stoppt.

Bemerkung: Du musst auf den Farbbereich im [Touch color]-Block klicken und dann das Pipettenwerkzeug verwenden, um die Farbe der schwarzen Linie auf der Bühne aufzunehmen. Wenn du willkürlich ein Schwarz wählst, funktioniert dieser [Touch color]-Block nicht.



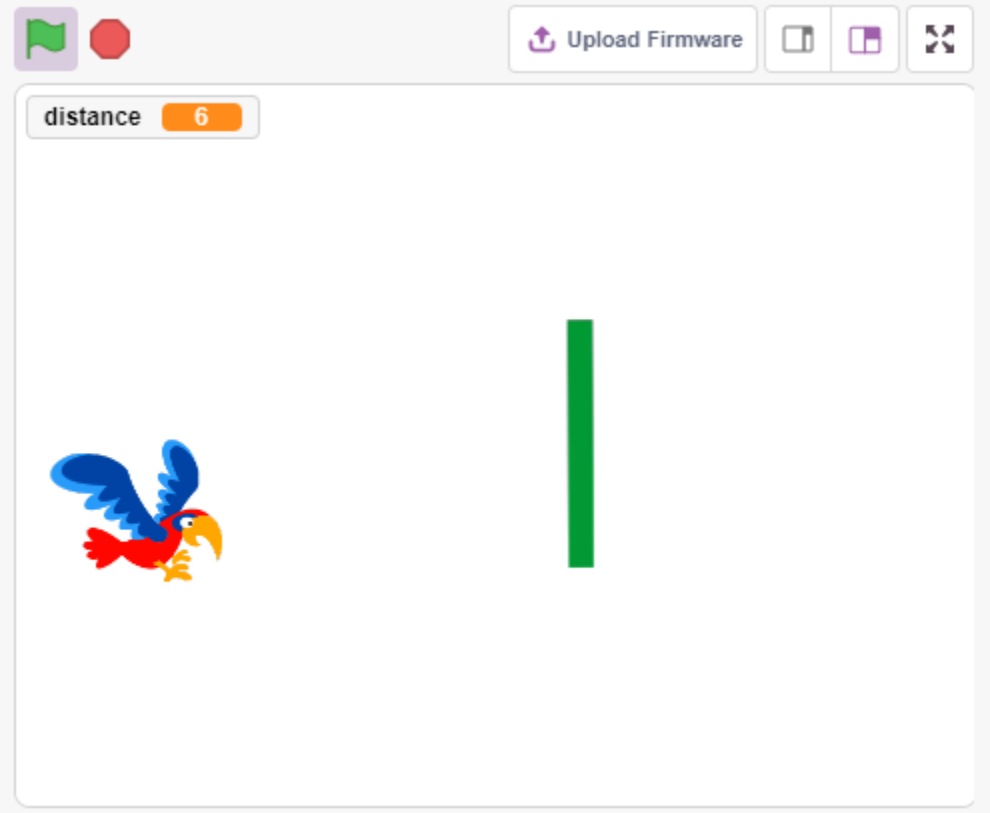
- Wenn Käfer Rot berührt (Verwende auch das Strohwerkzeug, um die rote Farbe des Apfels aufzunehmen), wird der Hintergrund auf **Win** umgeschaltet, was bedeutet, dass das Spiel erfolgreich ist und das Skript stoppt.



4.18 2.15 SPIEL - Flappy Papagei

Hier verwenden wir das Ultraschallmodul, um ein Flappy-Papagei-Spiel zu spielen.

Nachdem das Skript läuft, bewegt sich der grüne Bambus langsam von rechts nach links auf einer zufälligen Höhe. Platziere jetzt deine Hand über dem Ultraschallmodul, wenn der Abstand zwischen deiner Hand und dem Ultraschallmodul weniger als 10 beträgt, fliegt der Papagei nach oben, sonst fällt er nach unten. Du musst den Abstand zwischen deiner Hand und dem Ultraschallmodul kontrollieren, damit der Papagei den grünen Bambus (Schläger) meiden kann, wenn er ihn berührt, ist das Spiel vorbei.



4.18.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

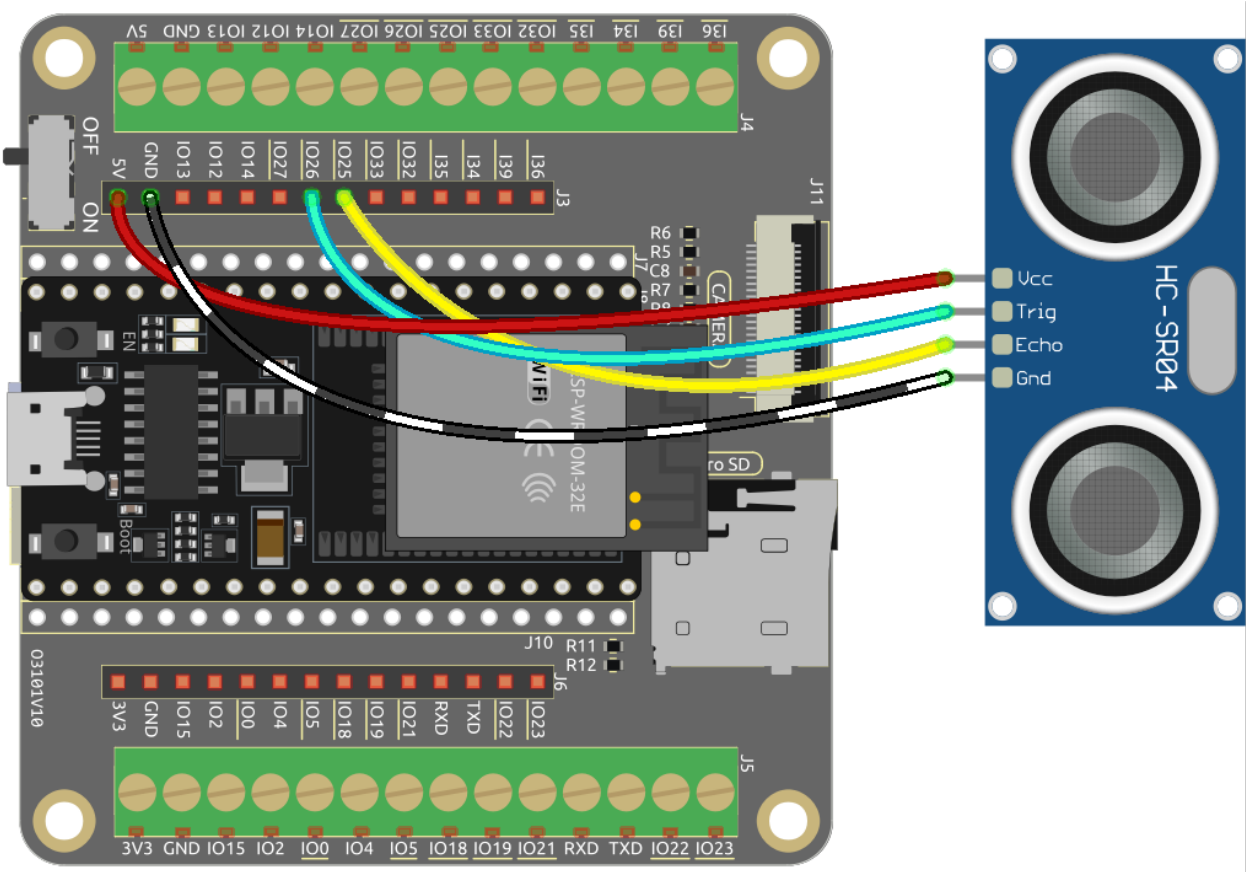
Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Ultraschall-Modul	

4.18.2 Schaltung Aufbauen

Ein Ultraschallsensormodul ist ein Instrument, das die Entfernung zu einem Objekt mit Ultraschallwellen misst. Es hat zwei Sonden. Eine sendet Ultraschallwellen und die andere empfängt die Wellen und verwandelt die Zeit des Sendens und Empfangens in eine Entfernung, um so die Entfernung zwischen dem Gerät und einem Hindernis zu messen.

Bauen Sie die Schaltung gemäß dem folgenden Diagramm auf.

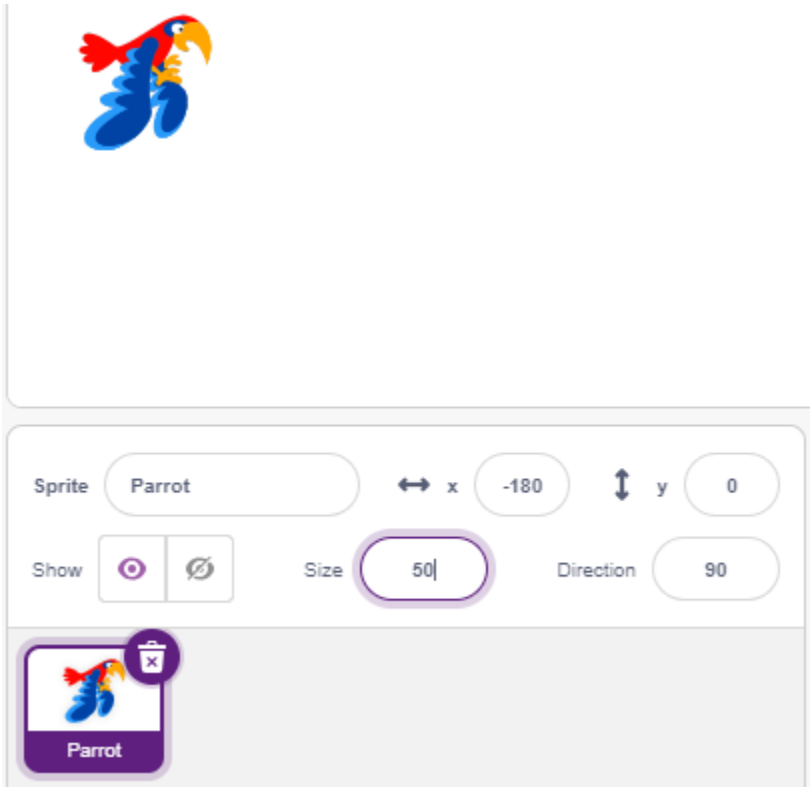


4.18.3 Programmierung

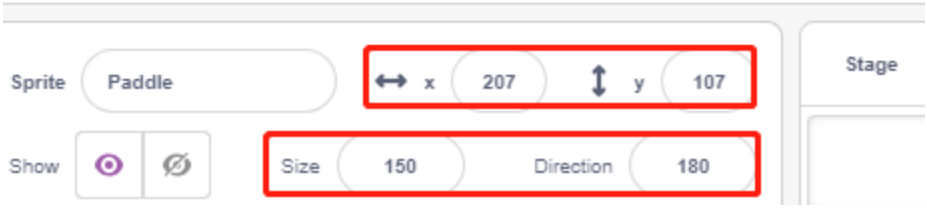
Das zu erreichende Ziel ist es, das Ultraschallmodul zu verwenden, um die Flughöhe des Sprites **Parrot** zu steuern, während es das Sprite **Paddle** vermeidet.

1. Ein Sprite hinzufügen

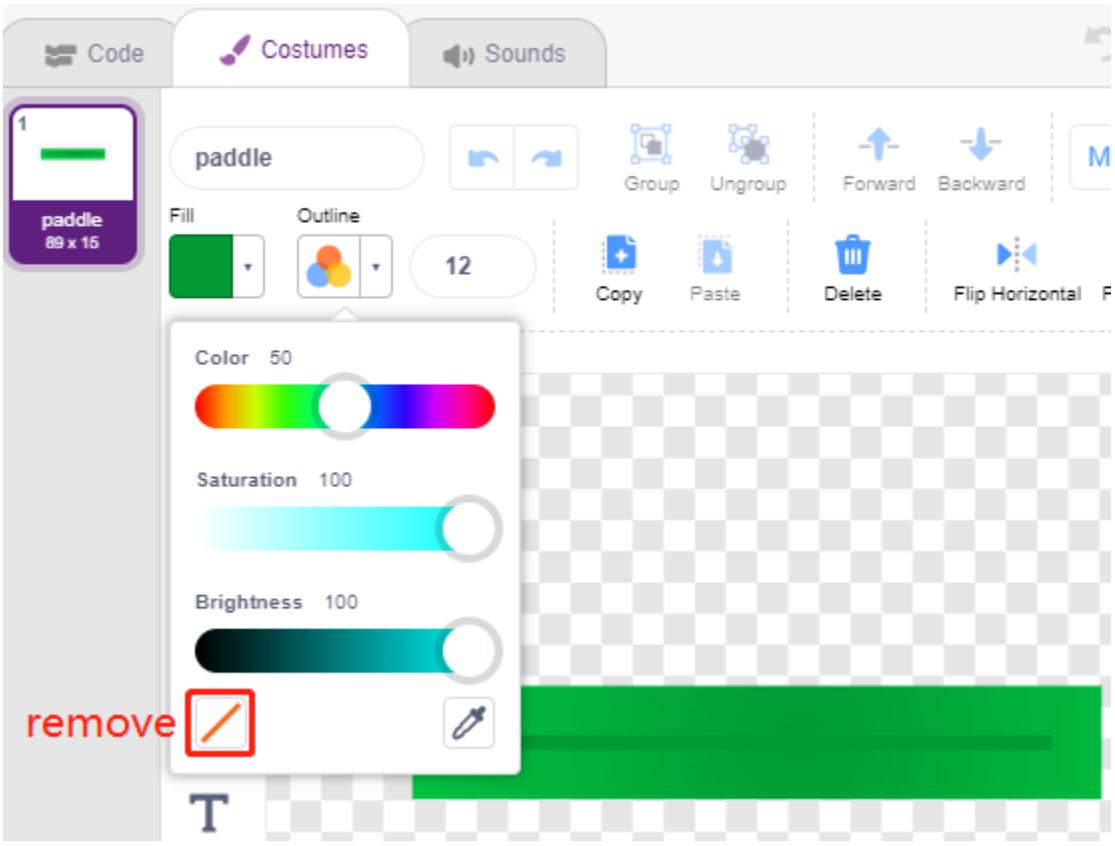
Lösche das Standard-Sprite und füge über den Button **Choose a Sprite** das **Parrot**-Sprite hinzu. Setze seine Größe auf 50% und verschiebe seine Position in die linke Mitte.



Füge nun das **Paddle**-Sprite hinzu, setze seine Größe auf 150%, stelle seinen Winkel auf 180 ein und verschiebe seine anfängliche Position in die obere rechte Ecke.



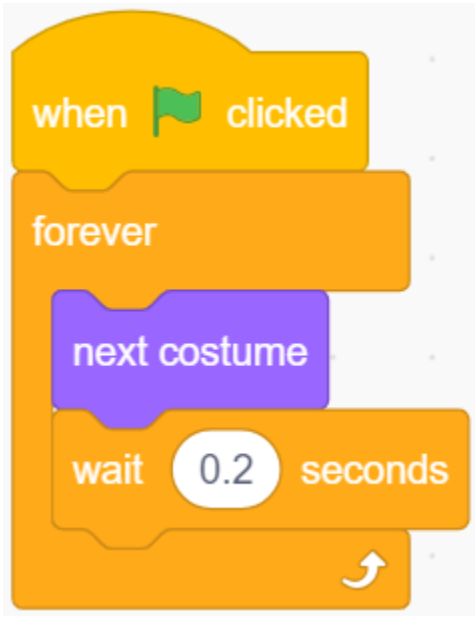
Gehe zur **Costumes**-Seite des **Paddle**-Sprites und entferne die Umrandung.



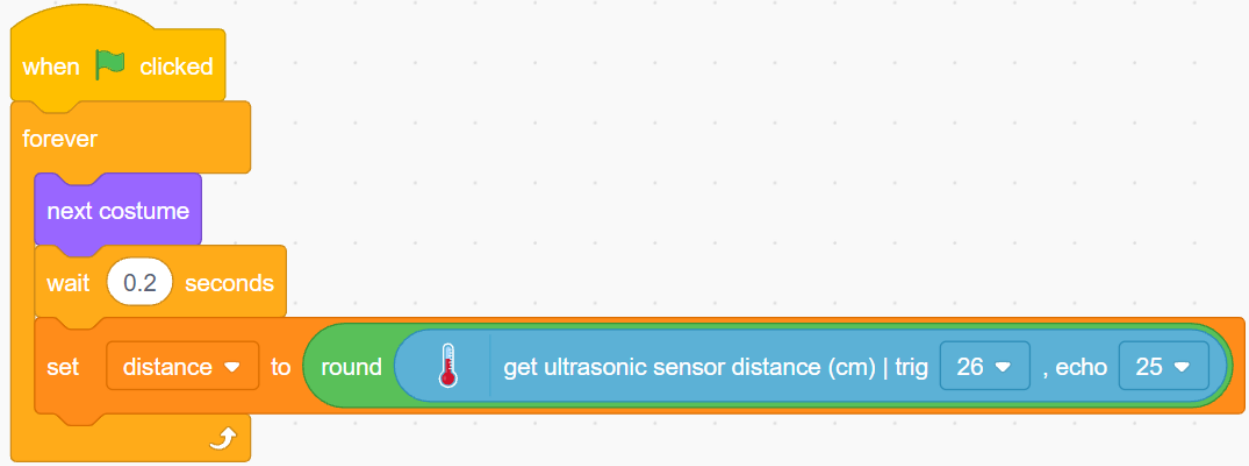
2. Skript für das Papagei-Sprite

Programmiere jetzt das **Parrot**-Sprite, das im Flug ist und dessen Flughöhe durch den Erfassungsabstand des Ultraschallmoduls bestimmt wird.

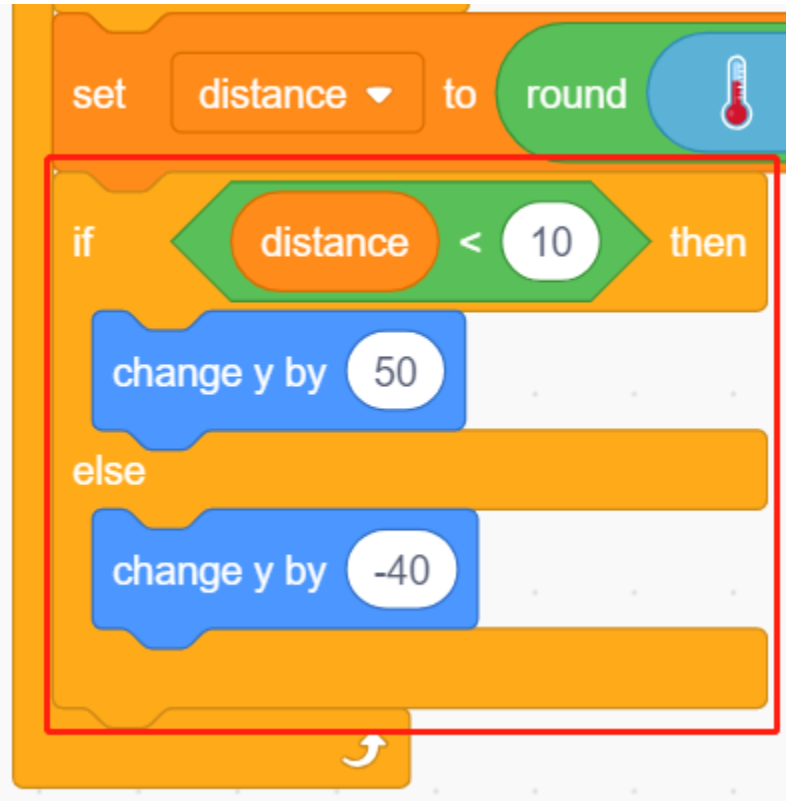
- Wenn die grüne Flagge angeklickt wird, wechsle das Kostüm alle 0,2 Sekunden, damit es immer im Flug ist.



- Lies den Wert des Ultraschallmoduls und speichere ihn nach dem Runden mit dem [round]-Block in der Variable **distance**.



- Wenn die Ultraschall-Erfassungsdistanz weniger als 10 cm beträgt, erhöhe die y-Koordinate um 50, das **Parrot**-Sprite fliegt nach oben. Andernfalls wird der y-Koordinatenwert um 40 verringert, **Parrot** fällt nach unten.

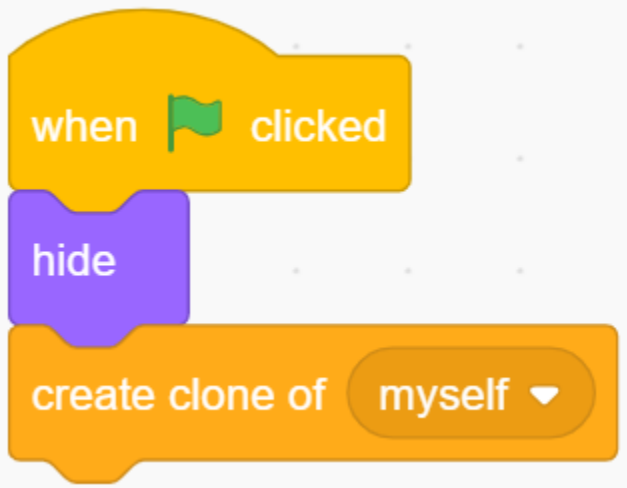


- Wenn das **Parrot**-Sprite das **Paddle**-Sprite berührt, endet das Spiel und das Skript stoppt.

3. Skript für das Schläger-Sprite

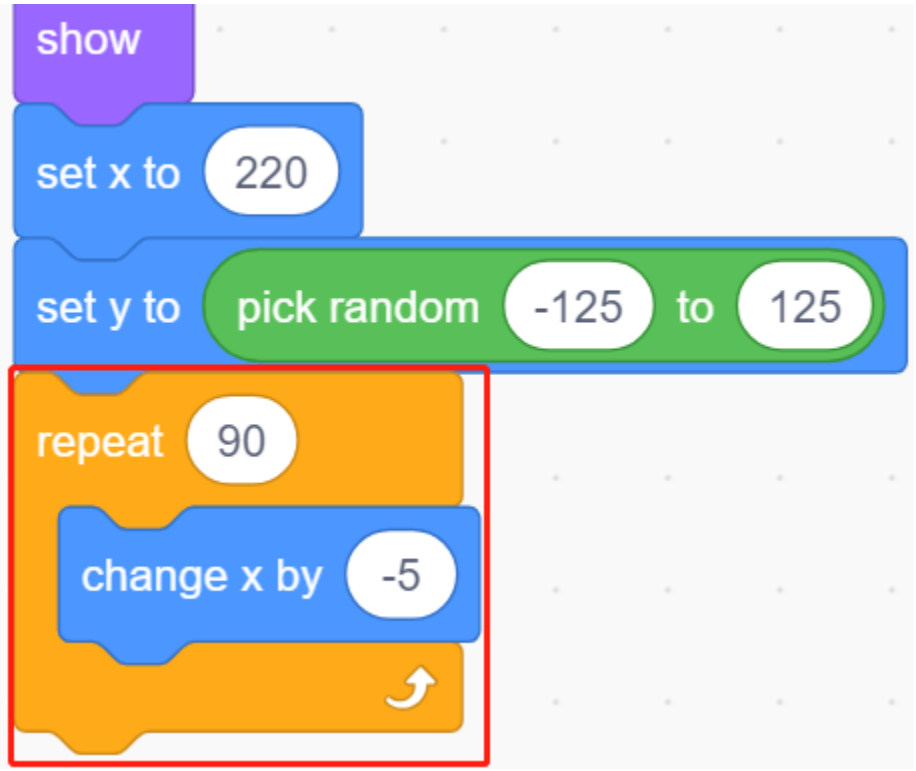
Schreibe jetzt das Skript für das **Paddle**-Sprite, das zufällig auf der Bühne erscheinen muss.

- Verstecke das Sprite **Paddle**, wenn die grüne Flagge angeklickt wird, und klone es gleichzeitig. Der [create clone of]-Block ist ein Steuerungsblock und ein Stapelblock. Er erstellt einen Klon des Sprites im Argument. Es kann auch das Sprite klonen, in dem es läuft, wodurch Klone von Klonen rekursiv erstellt werden.

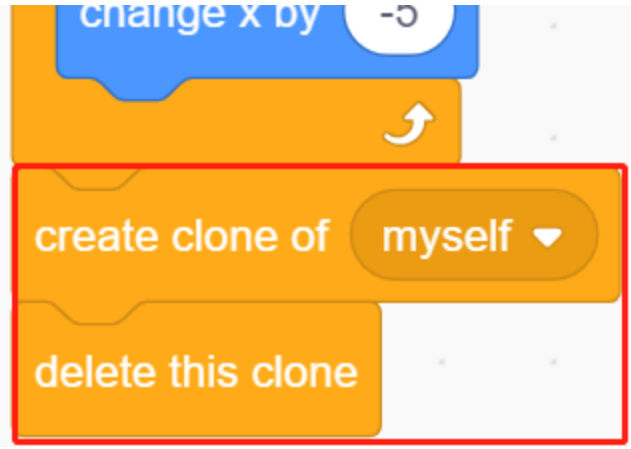


- Wenn **Paddle** als Klon präsentiert wird, ist seine Position 220 (rechts) für die x-Koordinate und seine y-Koordinate zufällig zwischen (-125 bis 125) (Höhe zufällig).

- Verwende den [repeat]-Block, um den x-Koordinatenwert langsam zu verringern, sodass du sehen kannst, wie der Klon des **Paddle**-Sprites langsam von rechts nach links bewegt wird, bis es verschwindet.



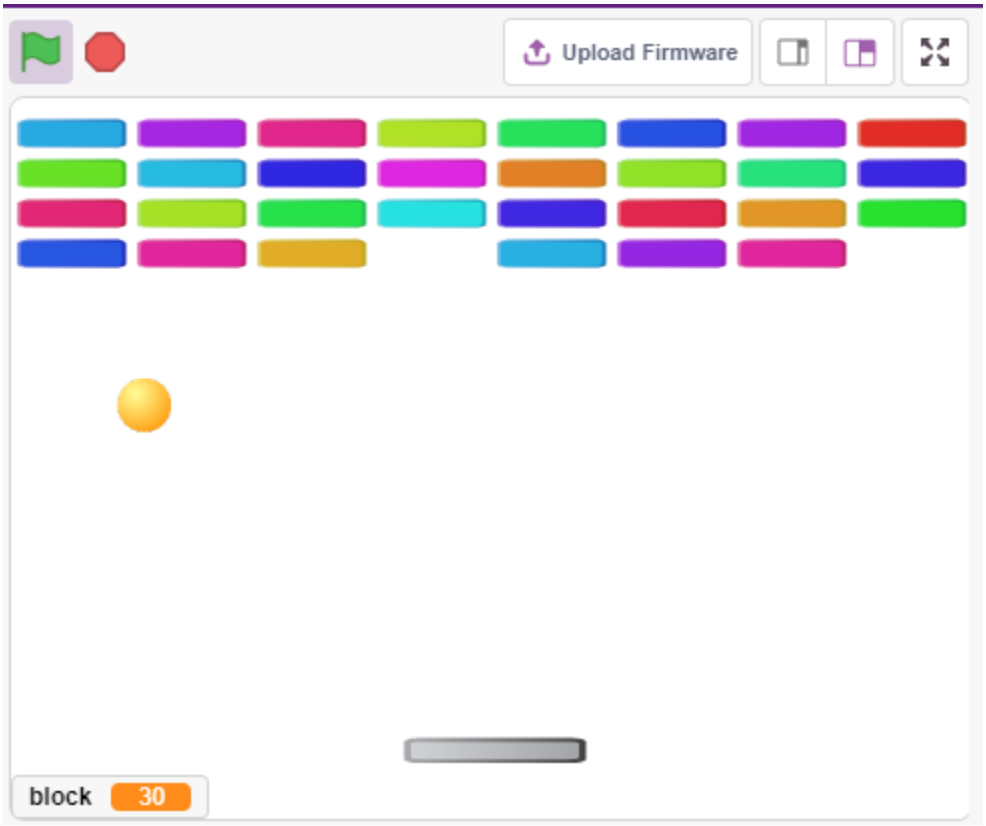
- Klone erneut ein neues **Paddle**-Sprite und lösche den vorherigen Klon.



4.19 2.16 SPIEL - Breakout-Klon

Hier verwenden wir den Potentiometer, um ein Breakout-Klon-Spiel zu spielen.

Nachdem du auf die grüne Flagge geklickt hast, musst du den Potentiometer verwenden, um das Schläger-Sprite auf der Bühne zu steuern, um den Ball zu fangen, damit er nach oben gehen und die Ziegel treffen kann. Wenn alle Ziegel verschwinden, ist das Spiel gewonnen, wenn du den Ball nicht fängst, ist das Spiel verloren.



4.19.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

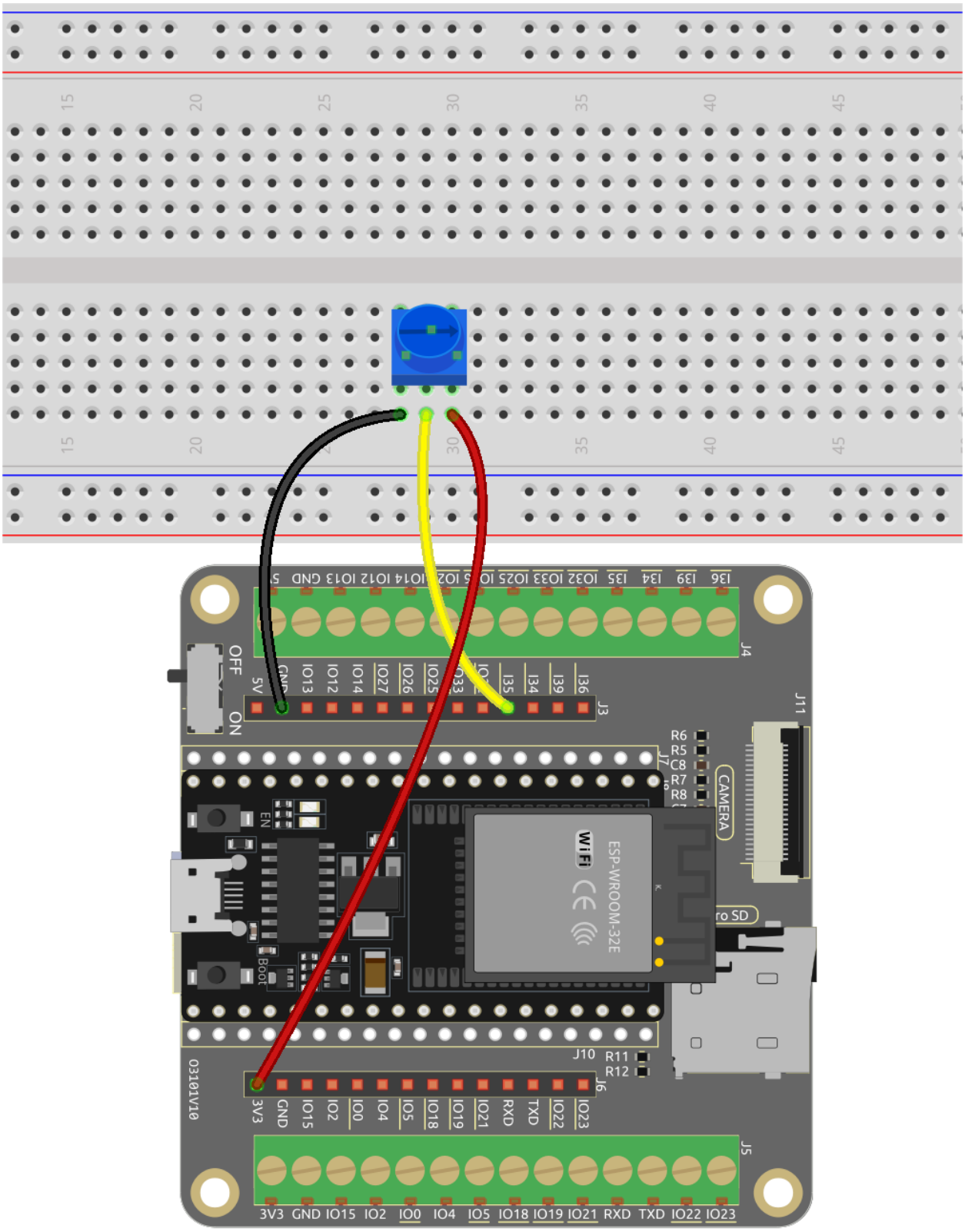
Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Potentiometer	

4.19.2 Schaltung Aufbauen

Der Potentiometer ist ein resistives Element mit 3 Anschlüssen, die beiden seitlichen Pins sind mit 5V und GND verbunden und der mittlere Pin mit Pin35. Nach der Umwandlung durch den ADC-Konverter des ESP32-Boards liegt der Wertebereich bei 0-4095.


OFF
ON
ESP-WROOM-32E
CAMERA
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

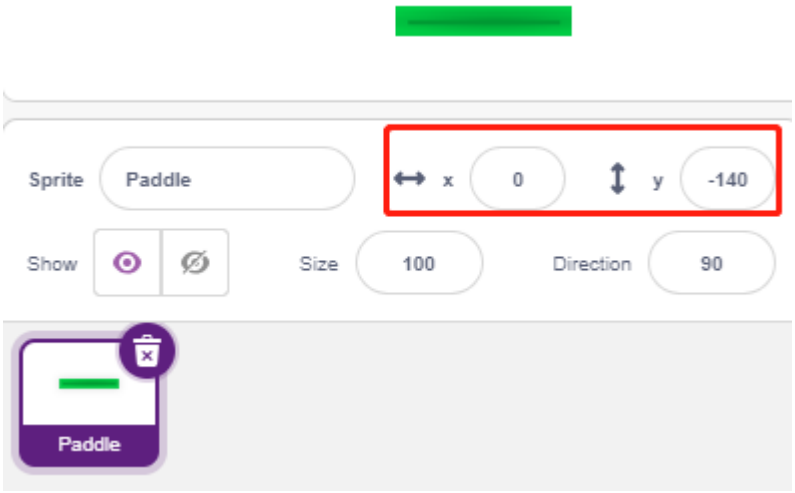
4.19.3 Programmierung

Es gibt 3 Sprites auf der Bühne.

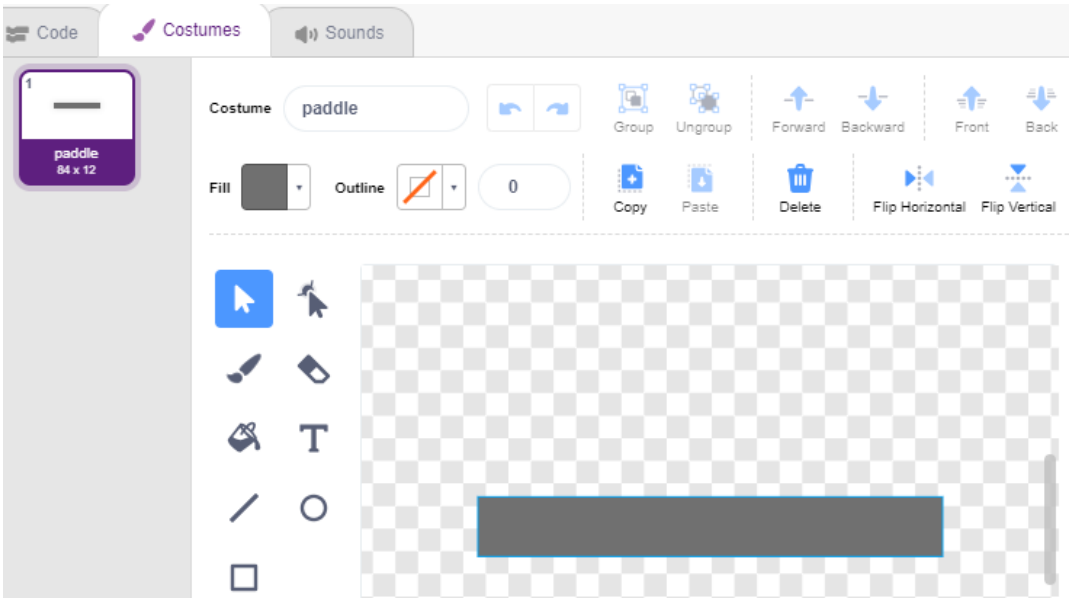
1. Schläger-Sprite

Das Ziel des **Paddle** ist es, die anfängliche Position in der Mitte unten auf der Bühne zu haben und es wird durch einen Potentiometer gesteuert, um es nach links oder rechts zu bewegen.

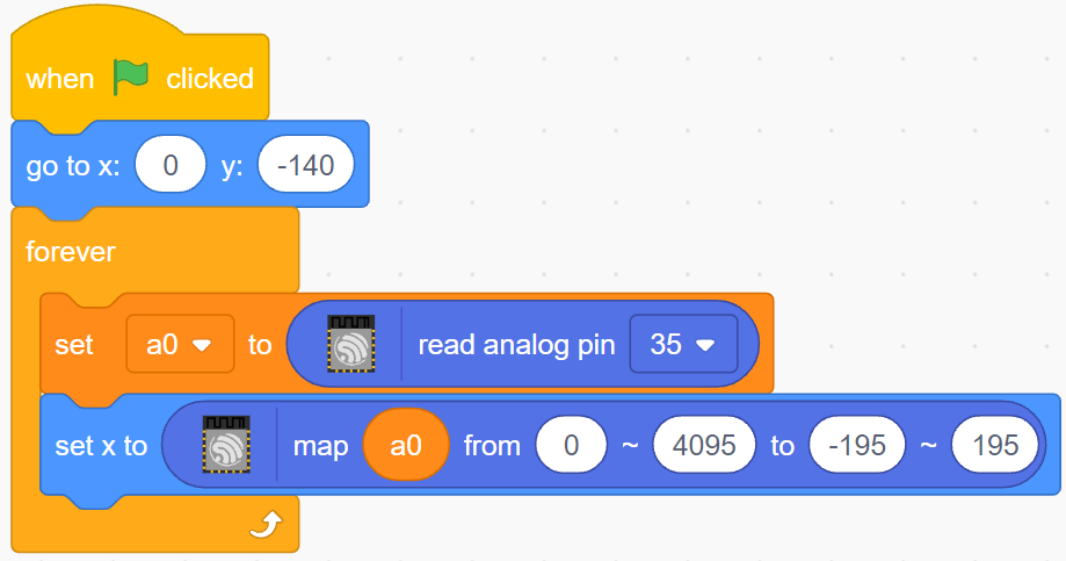
- Lösche das Standard-Sprite, verwende den Button **Choose a Sprite**, um das **Paddle**-Sprite hinzuzufügen, und setze seine x- und y-Koordinaten auf (0, -140).



- Gehe zur **Costumes**-Seite, entferne die Umrandung und ändere seine Farbe in Dunkelgrau.



- Programmiere jetzt das **Paddle**-Sprite, um seine anfängliche Position auf (0, -140) zu setzen, wenn die grüne Flagge angeklickt wird, und lies den Wert von Pin35 (Potentiometer) in die Variable **a0** ein. Da das **Paddle**-Sprite auf der Bühne von links nach rechts bei x-Koordinaten -195~195 bewegt, musst du den [map]-Block verwenden, um den Variablenbereich **a0** von 0~4095 auf -195~195 abzubilden.

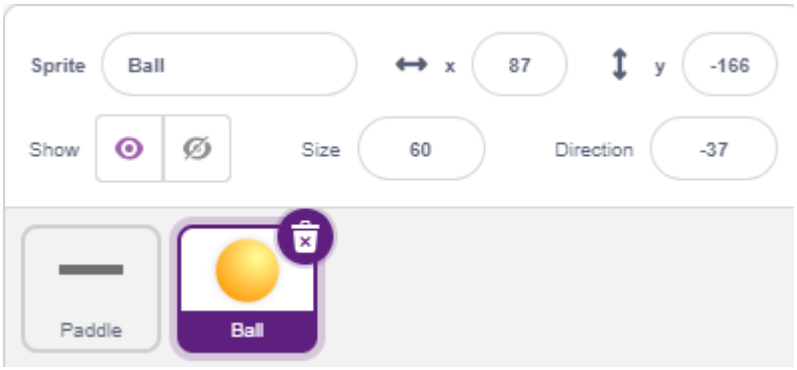


- Jetzt kannst du den Potentiometer drehen, um zu sehen, ob das **Paddle**-Sprite links und rechts auf der Bühne bewegt werden kann.

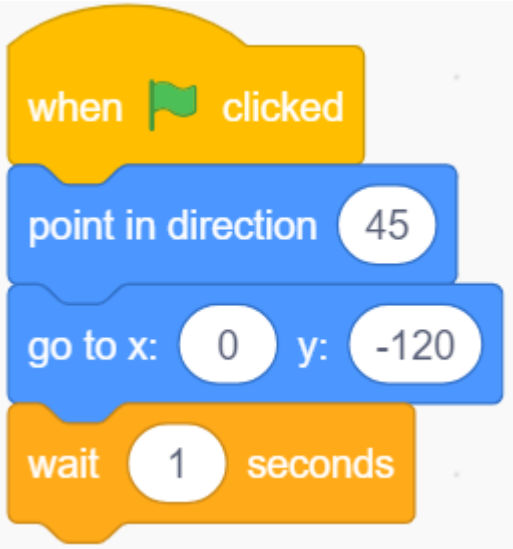
2. Ball-Sprite

Das Ball-Sprite bewegt sich auf der Bühne, prallt ab, wenn es den Rand berührt; es prallt nach unten ab, wenn es den Block über der Bühne berührt; es prallt nach oben ab, wenn es während des Fallens das Schläger-Sprite berührt; wenn nicht, stoppt das Skript und das Spiel endet.

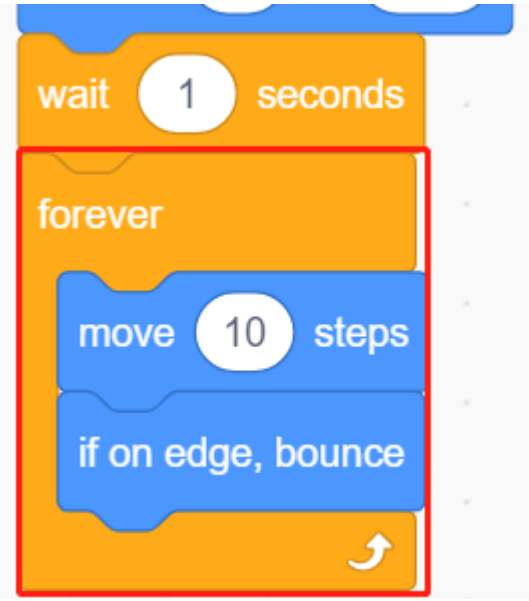
- Füge das **Ball**-Sprite hinzu.



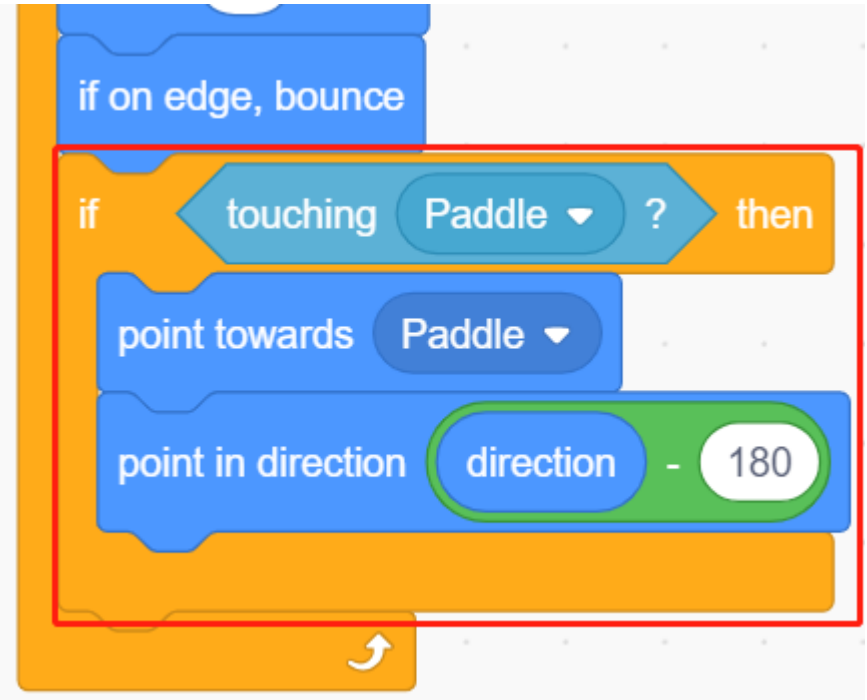
- Wenn die grüne Flagge angeklickt wird, setze den Winkel des **Ball**-Sprites auf 45° und die Anfangsposition auf (0, -120).

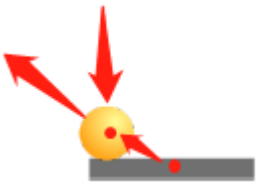


- Lass nun das **Ball**-Sprite sich auf der Bühne bewegen und abprallen, wenn es den Rand berührt, und klicke auf die grüne Flagge, um den Effekt zu sehen.



- Wenn das **Ball**-Sprite das **Paddle**-Sprite berührt, mache eine Reflexion. Der einfache Weg, dies zu tun, ist, den Winkel direkt umzukehren, aber dann wirst du feststellen, dass der Weg des Balls völlig festgelegt ist, was zu langweilig ist. Daher verwenden wir das Zentrum der beiden Sprites, um zu berechnen und den Ball in die entgegengesetzte Richtung des Zentrums des Schlägers abprallen zu lassen.





- Wenn das **Ball**-Sprite an den Rand der Bühne fällt, stoppt das Skript und das Spiel endet.

3. Block1-Sprite

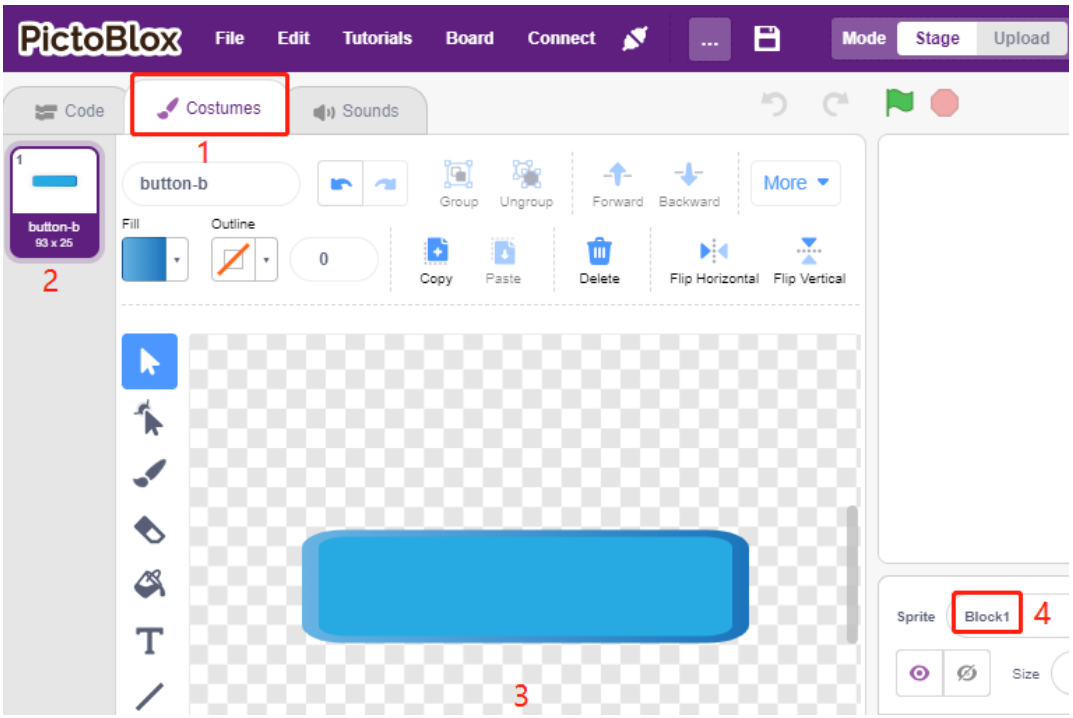
Das **Block1**-Sprite soll mit dem Effekt des Klonens 4x8 von sich selbst in einer zufälligen Farbe über der Bühne erscheinen und einen Klon löschen, wenn es vom **Ball**-Sprite berührt wird.

Das **Block1**-Sprite ist nicht in der **PictoBlox**-Bibliothek verfügbar, du musst es selbst zeichnen oder ein vorhandenes Sprite modifizieren. Hier werden wir es mit dem **Button3**-Sprite modifizieren.

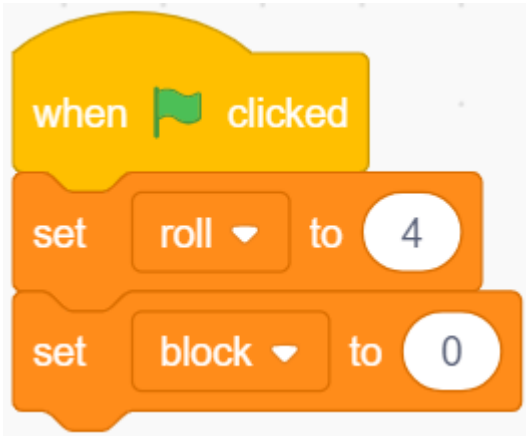
- Nachdem du das **Button3**-Sprite hinzugefügt hast, gehe zur **Costumes**-Seite. Lösche jetzt zuerst **button-a**, reduziere dann sowohl die Breite als auch die Höhe von **button-b** und ändere den Sprite-Namen in **Block1**, wie im folgenden Bild gezeigt.

Bemerkung:

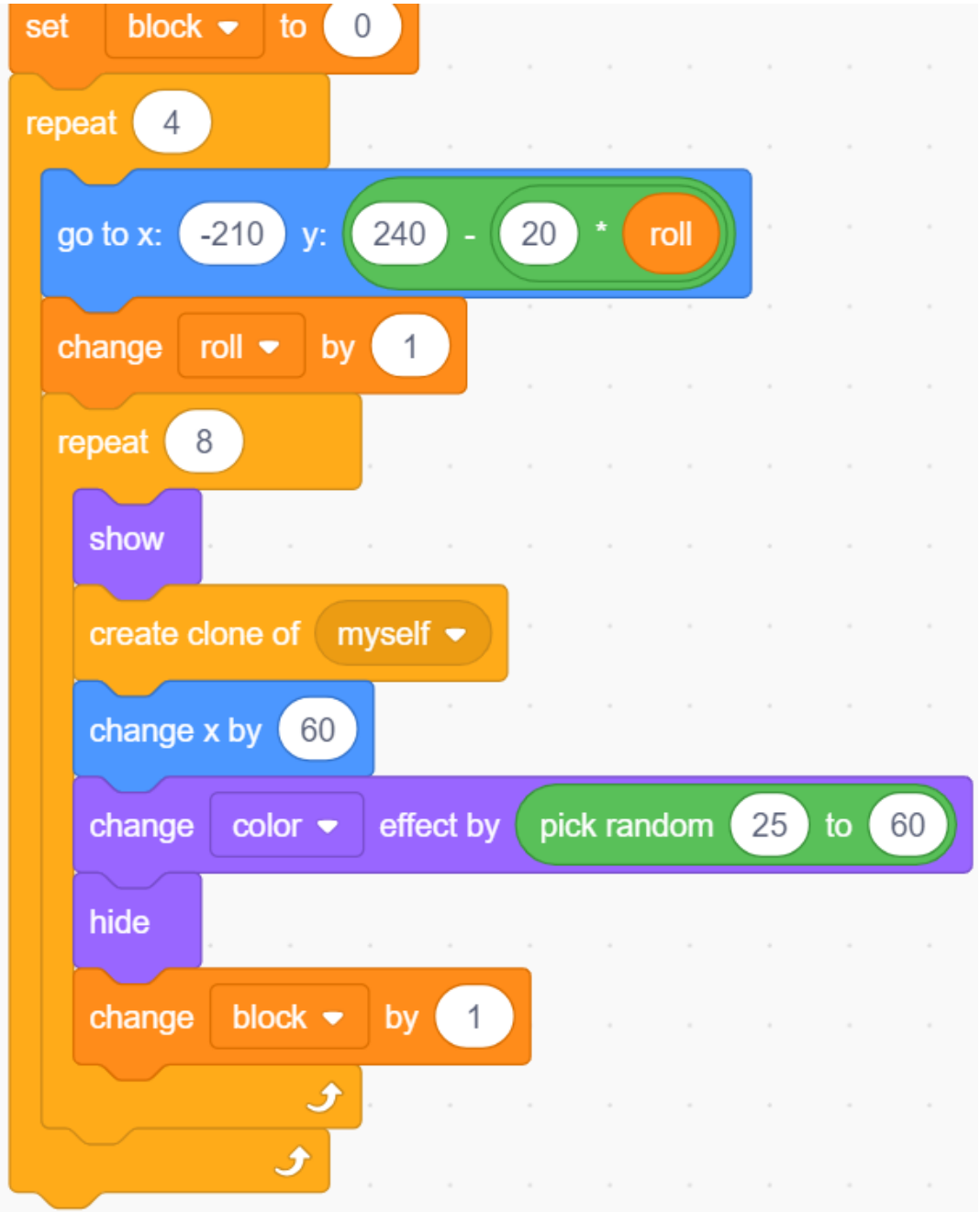
- Für die Breite von **Block1** kannst du es wahrscheinlich auf dem Bildschirm simulieren, um zu sehen, ob du 8 in einer Reihe unterbringen kannst, wenn nicht, dann reduziere die Breite entsprechend.
- Beim Verkleinern des **Block1**-Sprites musst du den Mittelpunkt in der Mitte des Sprites behalten.



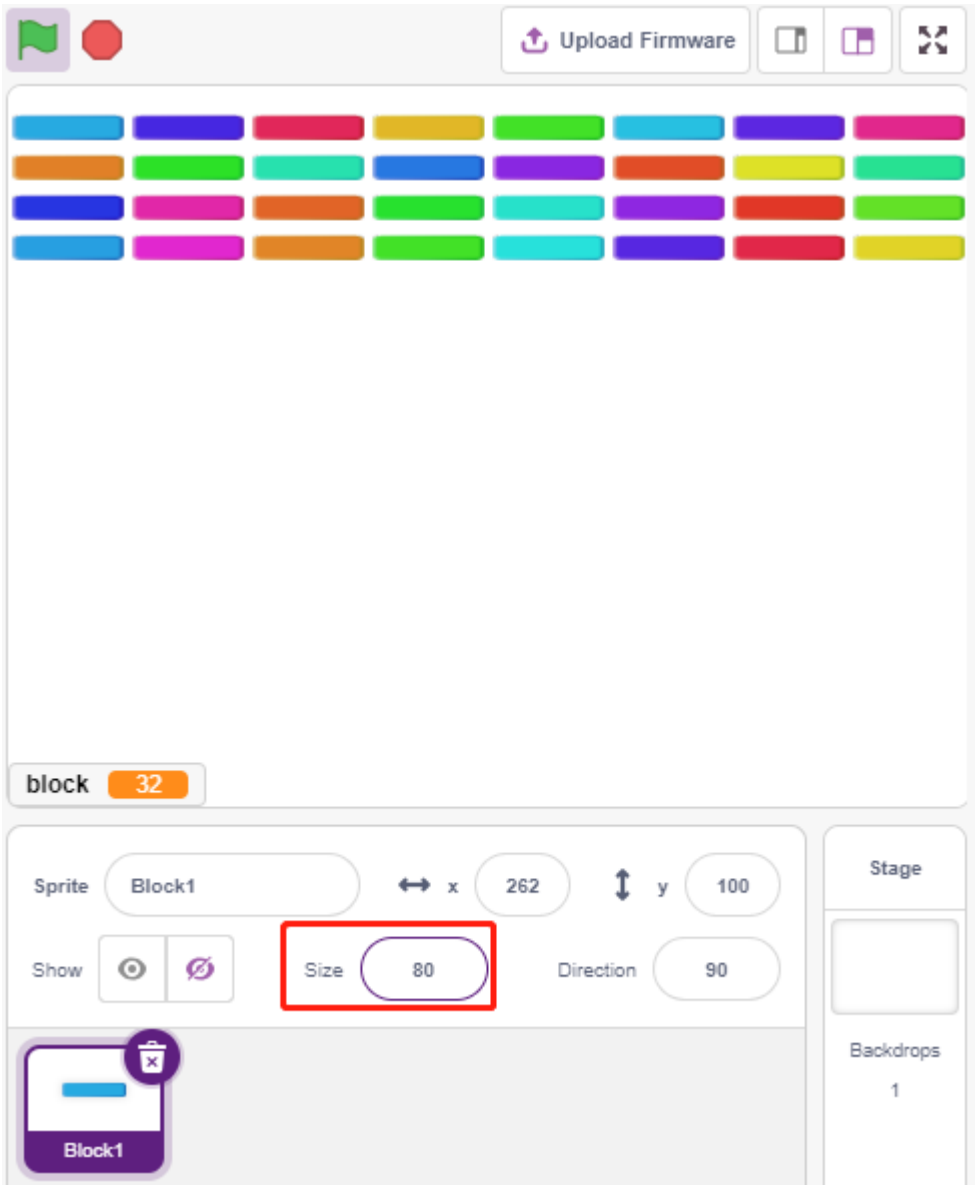
- Erstelle jetzt zuerst 2 Variablen, **block**, um die Anzahl der Blöcke und **roll** zu speichern, um die Anzahl der Reihen zu speichern.



- Wir müssen einen Klon des **Block1**-Sprites erstellen, sodass es sich von links nach rechts, von oben nach unten, eins nach dem anderen, insgesamt 4x8, mit zufälligen Farben anzeigt.

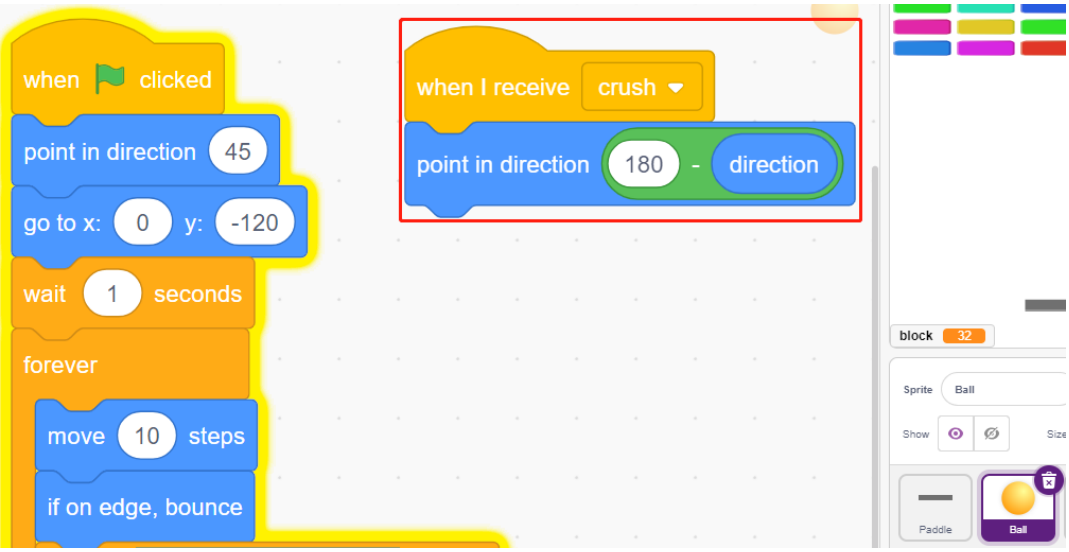


- Nachdem das Skript geschrieben ist, klicke auf die grüne Flagge und schaue dir die Anzeige auf der Bühne an, wenn es zu kompakt oder zu klein ist, kannst du die Größe ändern.



- Schreibe jetzt das Auslöseereignis. Wenn der geklonte **Block1**-Sprite das **Ball**-Sprite berührt, lösche den Klon und sende die Nachricht **crush**.

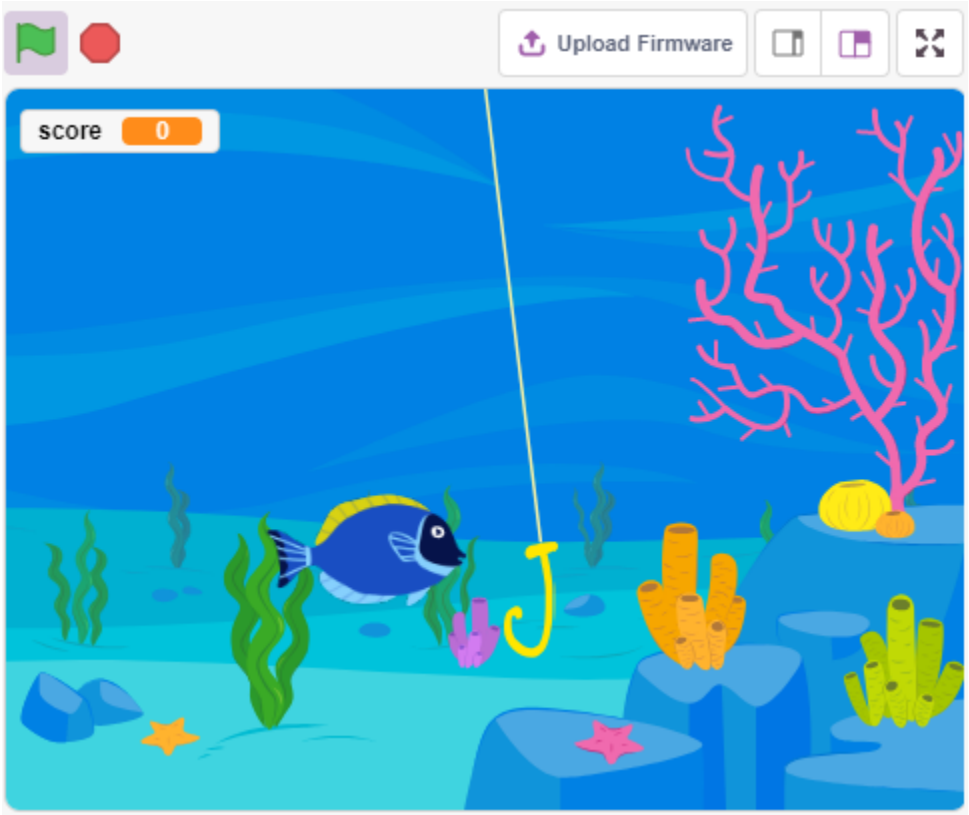
- Zurück zum **Ball**-Sprite, wenn die Sendung **crush** empfangen wird (das **Ball**-Sprite berührt den Klon des **Block1**-Sprites), wird der **Ball** aus der entgegengesetzten Richtung abgeprallt.



4.20 2.17 SPIEL - Angeln

Hier spielen wir ein Angelspiel mit einem Knopf.

Wenn das Skript läuft, schwimmen die Fische links und rechts auf der Bühne, und du musst den Knopf drücken, wenn der Fisch fast nahe am Haken ist (es wird empfohlen, ihn länger zu drücken), um den Fisch zu fangen, und die Anzahl der gefangenen Fische wird automatisch aufgezeichnet.



4.20.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

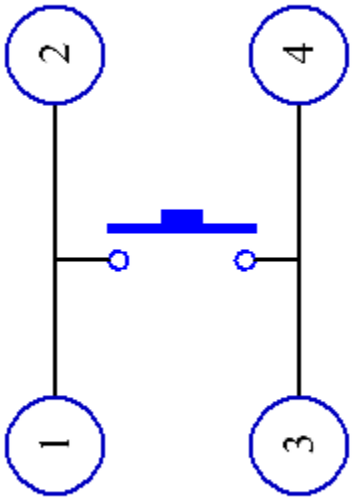
Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Widerstand	
Taste	

4.20.2 Schaltung Aufbauen

Der Knopf ist ein 4-poliges Gerät, da Pin 1 mit Pin 2 verbunden ist und Pin 3 mit Pin 4, wenn der Knopf gedrückt wird, sind die 4 Pins verbunden, wodurch der Stromkreis geschlossen wird.

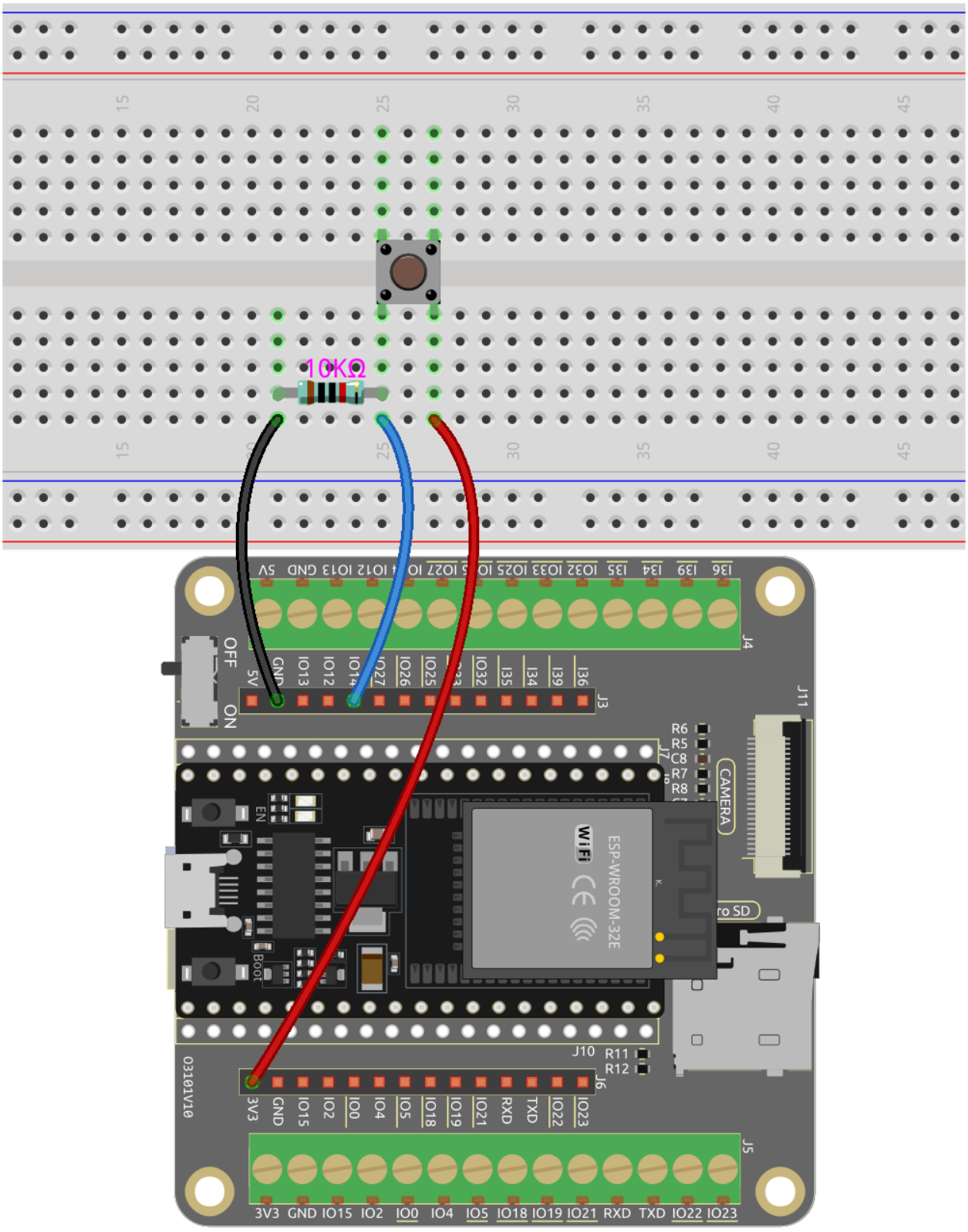
Button



Internal Structure

Bauen Sie die Schaltung gemäß dem folgenden Diagramm auf.

- Verbinden Sie einen der Pins auf der linken Seite des Knopfes mit Pin14, der mit einem Pull-Down-Widerstand und einem 0,1uF (104) Kondensator verbunden ist (um Schwankungen zu eliminieren und ein stabiles Level auszugeben, wenn der Knopf betätigt wird).
- Verbinden Sie das andere Ende des Widerstands und des Kondensators mit GND, und einen der Pins auf der rechten Seite des Knopfes mit 5V.

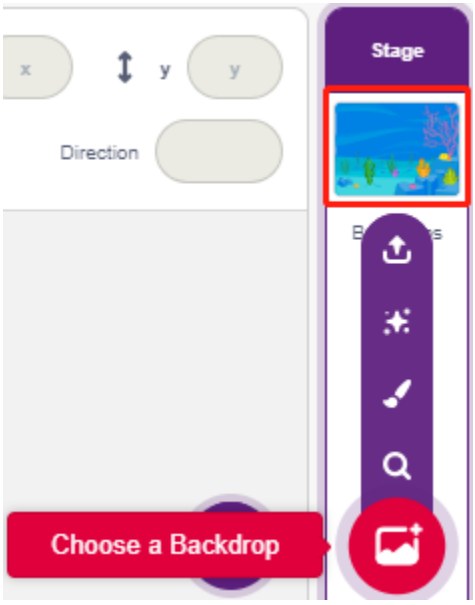

10KΩ
ESP-WROOM-32E
CAMERA
ON
OFF
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

4.20.3 Programmierung

Wir müssen zuerst einen **Underwater**-Hintergrund auswählen, dann ein **Fish**-Sprite hinzufügen und es auf der Bühne hin und her schwimmen lassen. Dann zeichne ein **Fishhook**-Sprite und steuere es mit einem Knopf, um mit dem Angeln zu beginnen. Wenn das **Fish**-Sprite im gehakten Zustand (wird rot) den **Fishhook**-Sprite berührt, wird es gehakt.

1. Hintergrund hinzufügen

Verwende den Button **Choose a Backdrop**, um einen **Underwater**-Hintergrund hinzuzufügen.

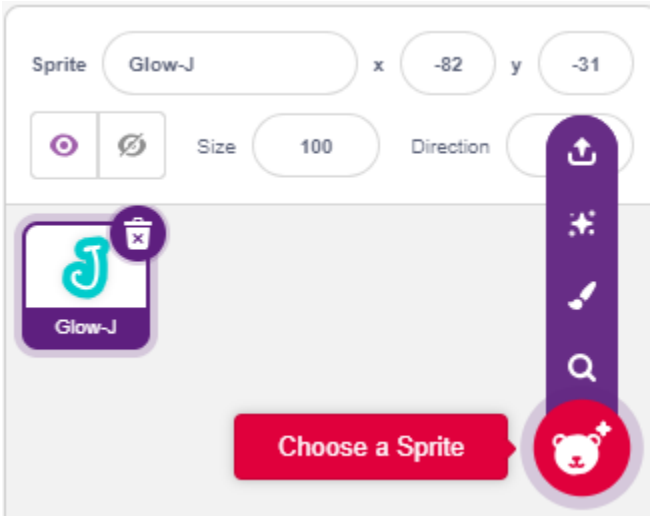


2. Angelhaken-Sprite

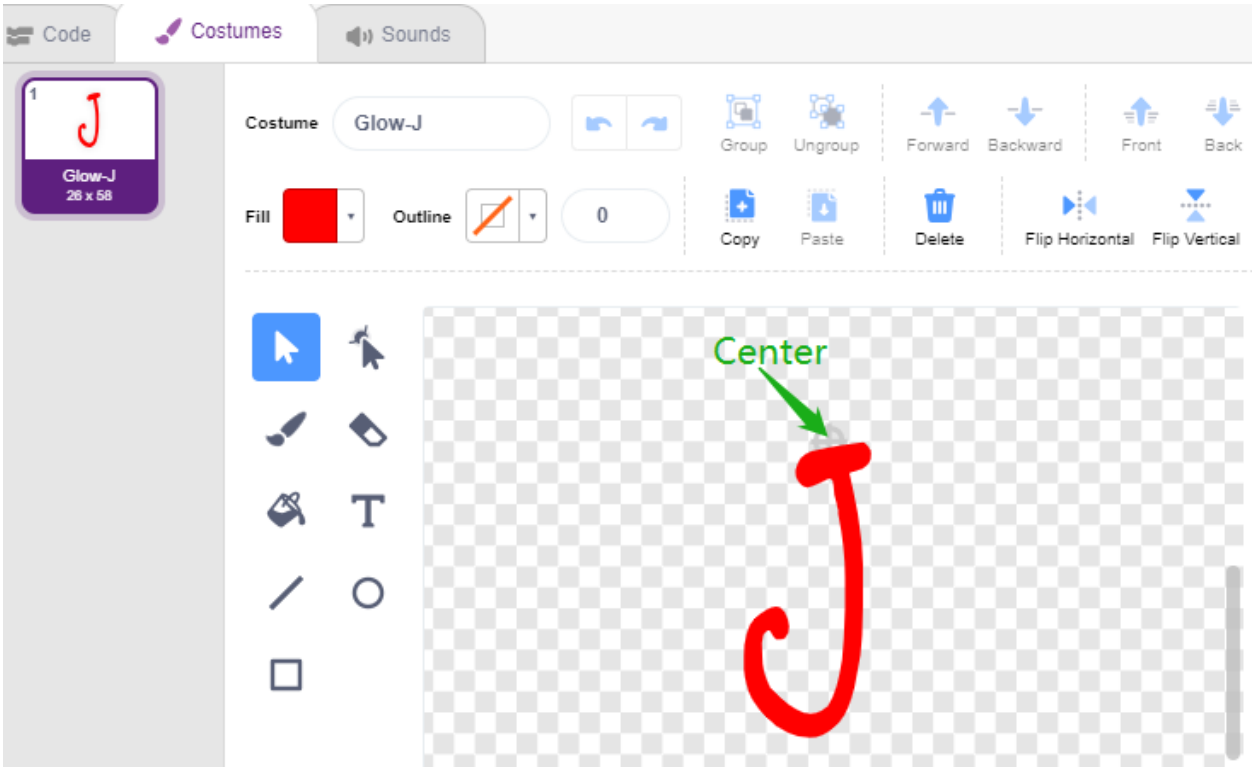
Das **Fishhook**-Sprite bleibt normalerweise unter Wasser im gelben Zustand; wenn der Knopf gedrückt wird, befindet es sich im Angelzustand (rot) und bewegt sich über der Bühne.

Da es kein **Fishhook**-Sprite in Pictoblox gibt, können wir das **Glow-J**-Sprite so modifizieren, dass es wie ein Angelhaken aussieht.

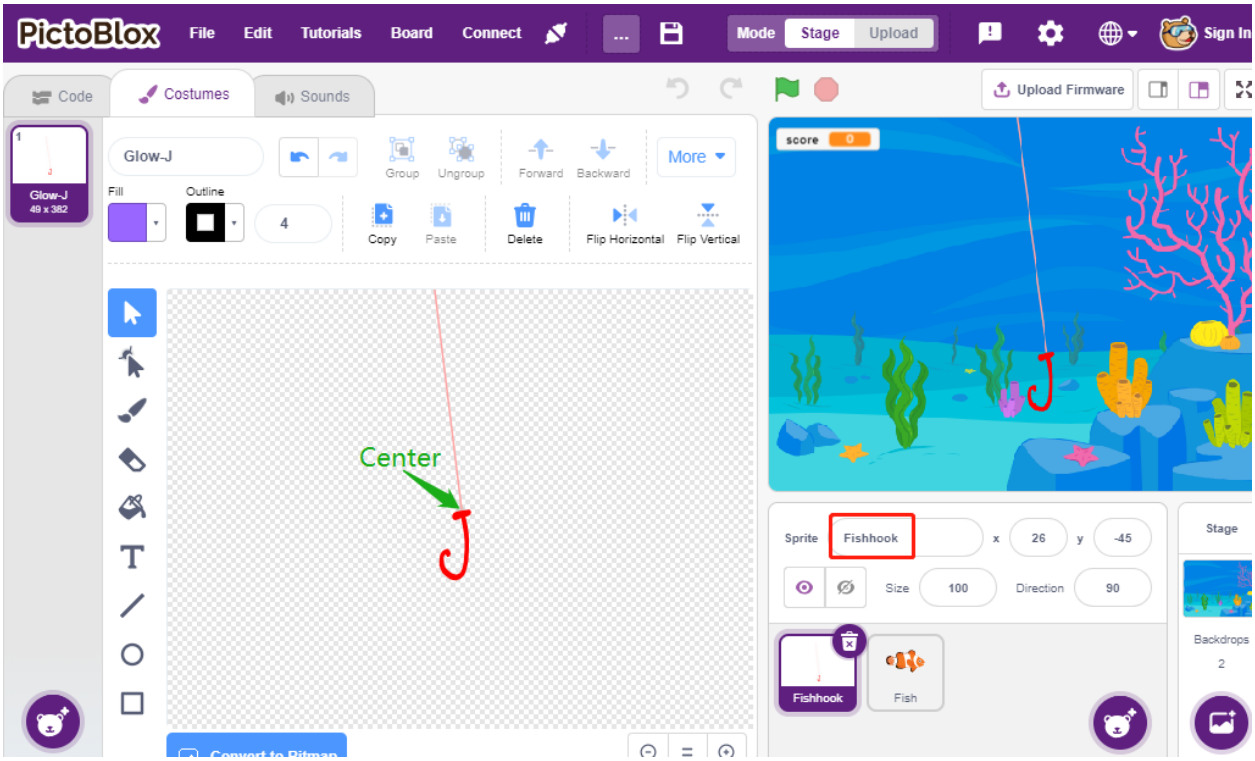
- Füge das **Glow-J**-Sprite über **Choose a Sprite** hinzu.



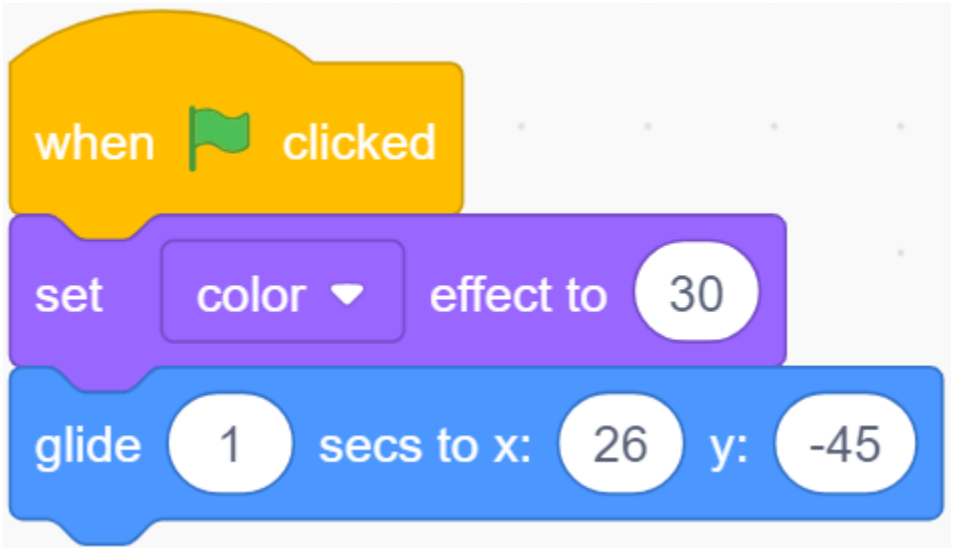
- Gehe nun zur **Costumes**-Seite des **Glow-J**-Sprites, wähle Cyans Füllung auf dem Bildschirm aus und entferne sie. Ändere dann die J-Farbe in Rot und verringere auch seine Breite. Der wichtigste Punkt ist, dass du den oberen Teil genau am Mittelpunkt haben musst.



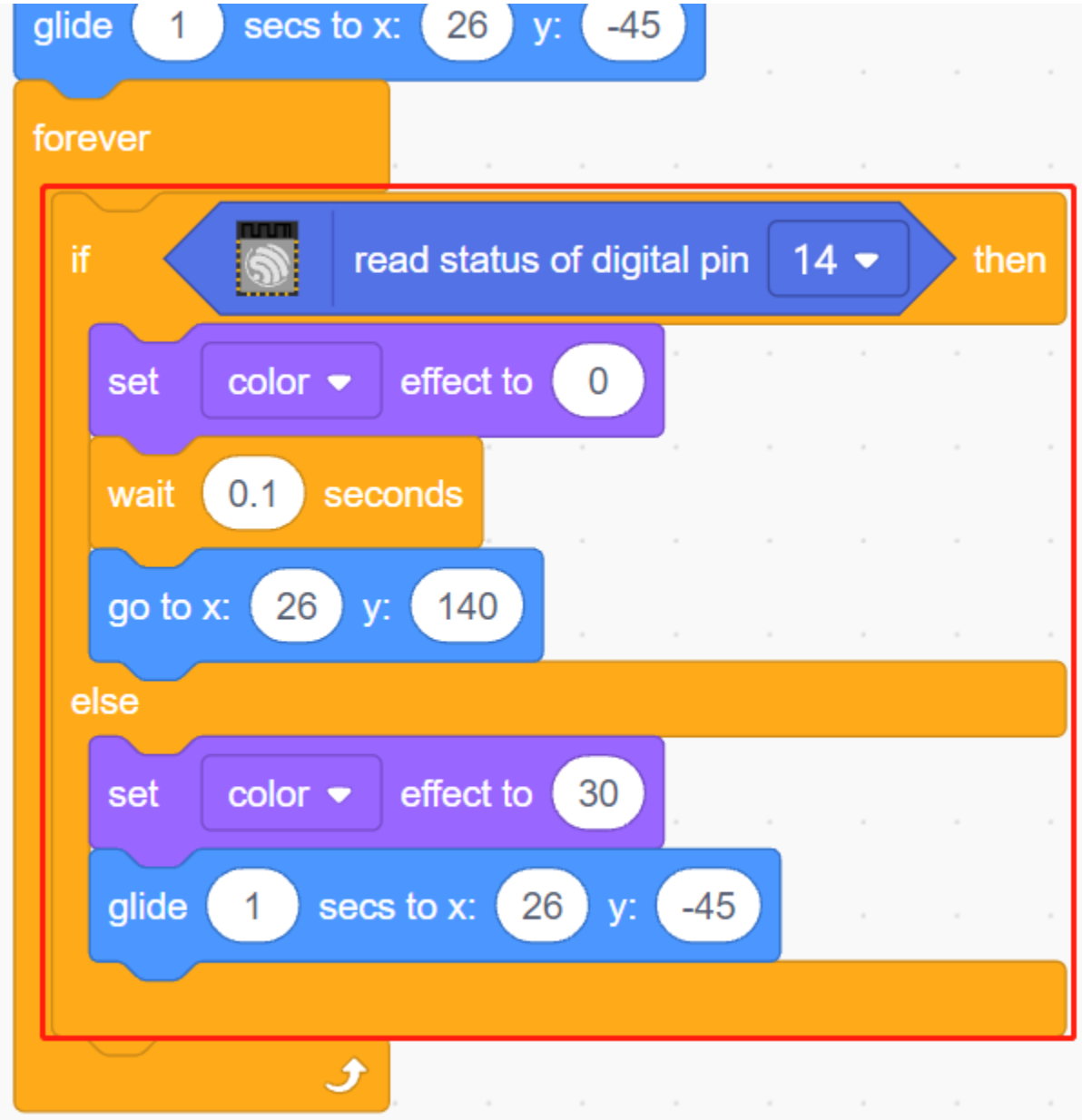
- Verwende das **Line tool**, um eine Linie so lang wie möglich vom Mittelpunkt nach oben zu zeichnen (Linie außerhalb der Bühne). Jetzt, wo das Sprite gezeichnet ist, setze den Sprite-Namen auf **Fishhook** und verschiebe es an die richtige Position.



- Wenn die grüne Flagge angeklickt wird, setze den Farbeffekt des Sprites auf 30 (Gelb) und setze seine Anfangsposition.



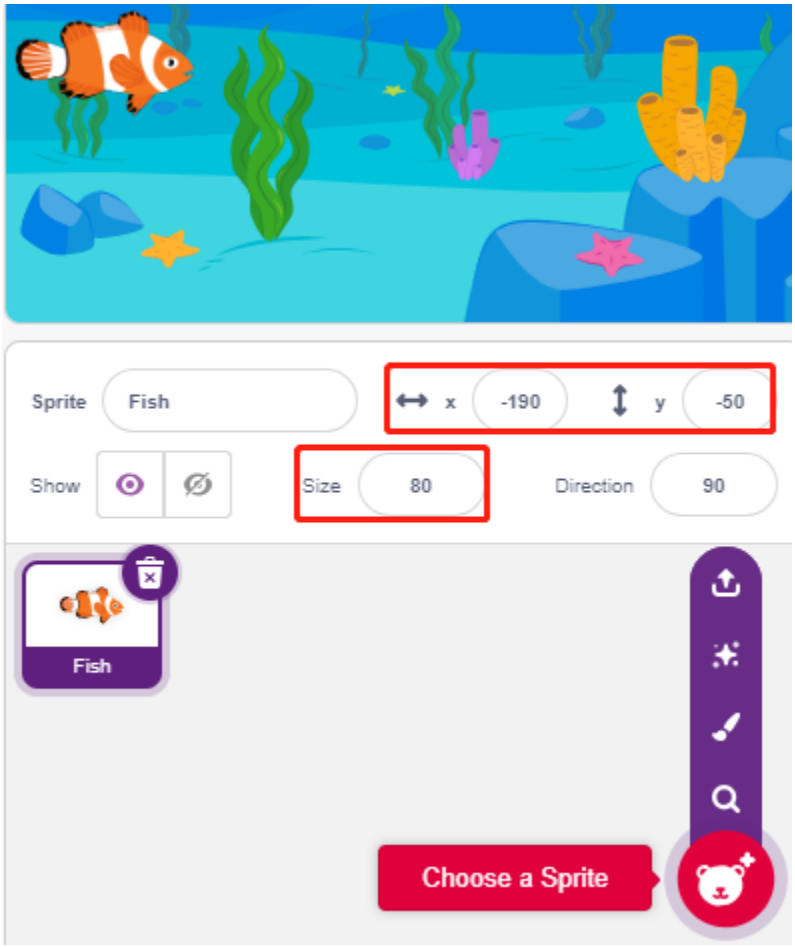
- Wenn der Knopf gedrückt wird, setze den Farbeffekt auf 0 (Rot, Angelzustand beginnen), warte 0,1 und bewege dann das **Fishhook**-Sprite an die Oberseite der Bühne. Lasse den Knopf los und lass den **Fishhook** an seine Anfangsposition zurückkehren.



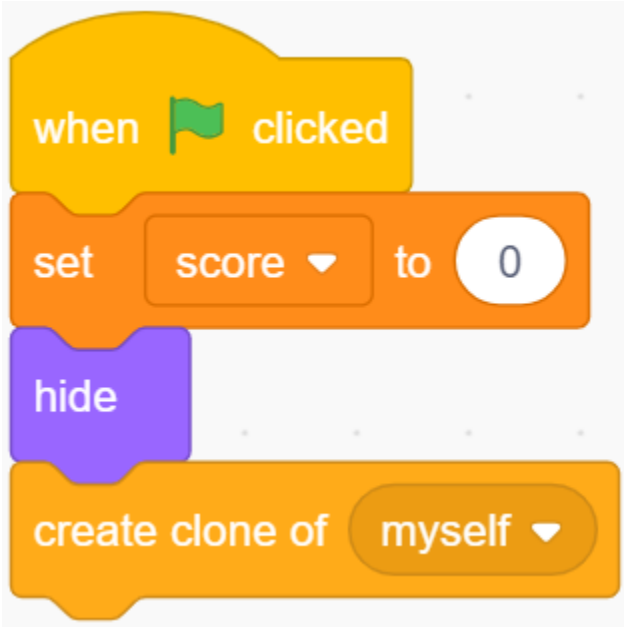
3. Fisch-Sprite

Das zu erreichende Ziel des **Fish**-Sprites ist es, sich links und rechts auf der Bühne zu bewegen, und wenn es auf ein **Fishhook**-Sprite im Angelzustand trifft, schrumpft es, bewegt sich an eine bestimmte Position und verschwindet dann, woraufhin ein neuer **fish**-Sprite geklont wird.

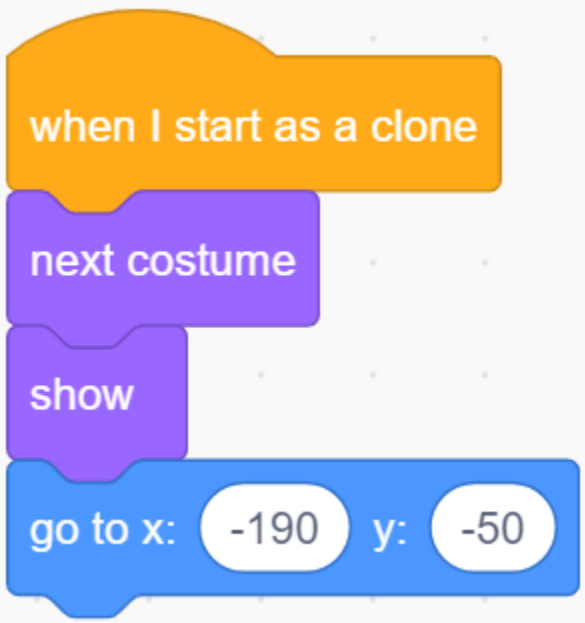
- Füge jetzt das **fish**-Sprite hinzu und passe seine Größe und Position an.



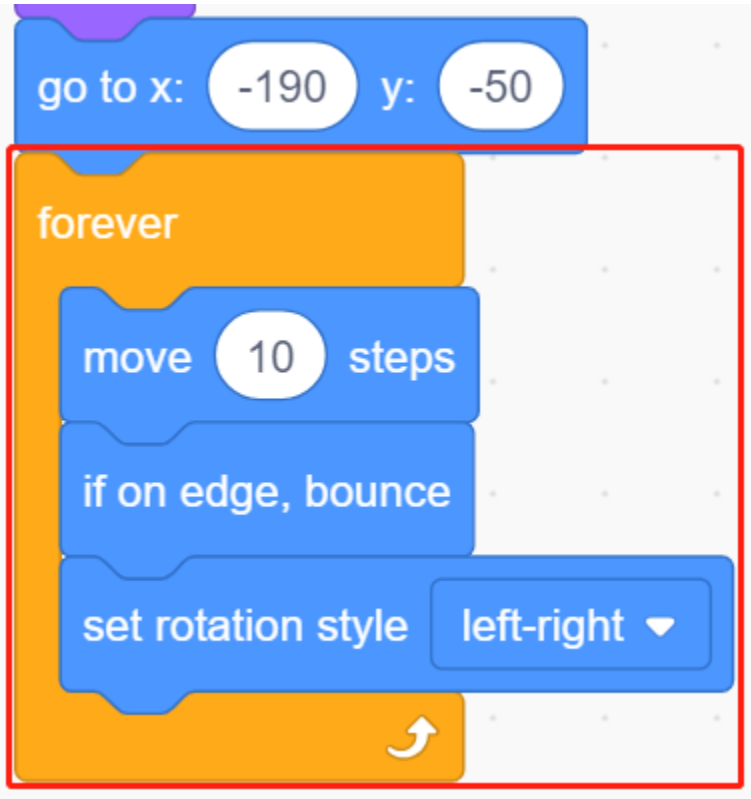
- Erstelle eine Variable **score**, um die Anzahl der gefangenen Fische zu speichern, verstecke dieses Sprite und klone es.



- Zeige den Klon des **fish**-Sprites, wechsle sein Kostüm und setze schließlich die Anfangsposition.

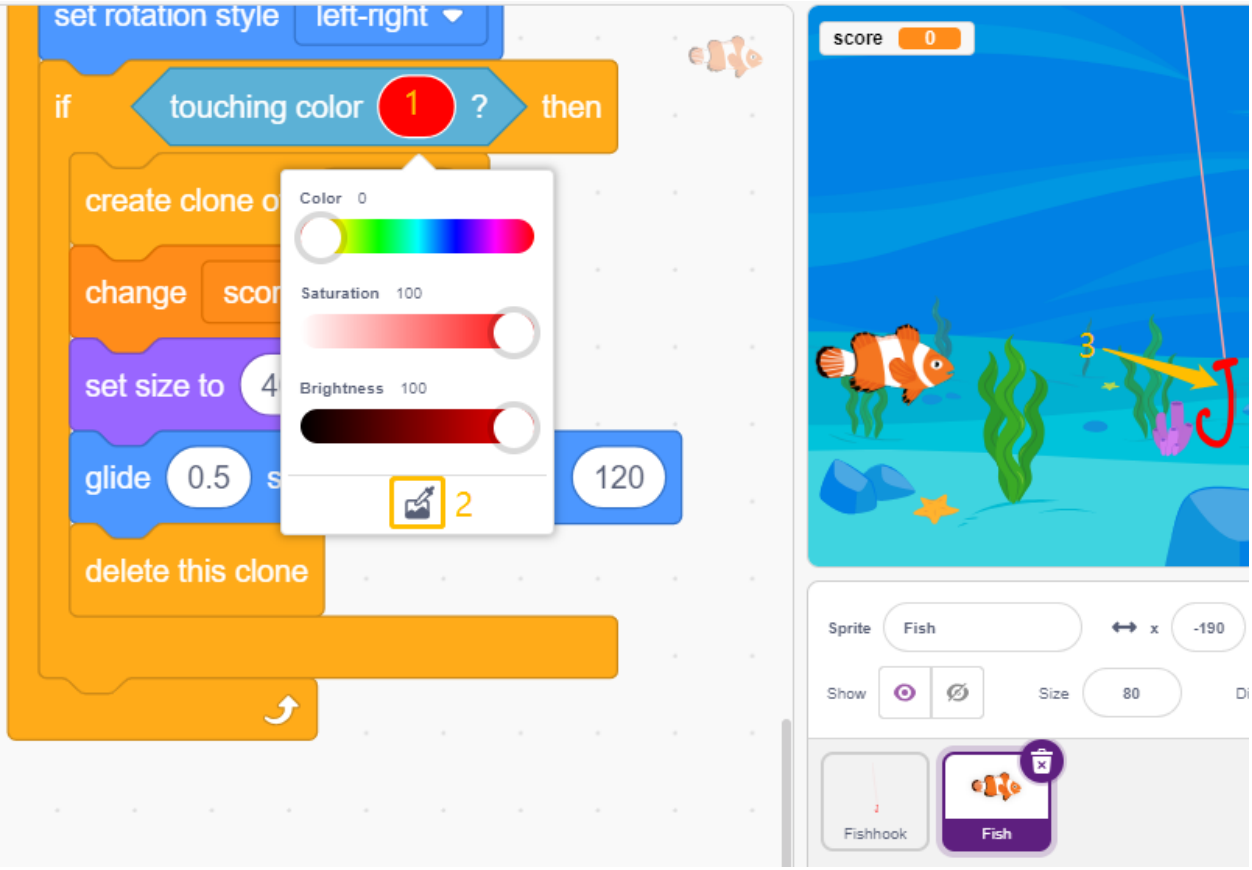


- Lasse den Klon des **fish**-Sprites sich links und rechts bewegen und pralle zurück, wenn es den Rand berührt.



- Das **fish**-Sprite (des Klons) reagiert nicht, wenn es am **Fishhook**-Sprite vorbeikommt; wenn es das **Fishhook**-Sprite im Angelzustand (wird rot) berührt, wird es gefangen, wobei die Punktzahl (Variable Punktzahl) +1 erhöht wird, und es zeigt auch eine Punktzahlanimation (schrumpft um 40%, bewegt sich schnell an die Position der Punkteanzeige und verschwindet). Gleichzeitig wird ein neuer Fisch erstellt (ein neuer Klon des Fisch-Sprites) und das Spiel geht weiter.

Bemerkung: Du musst auf den Farbbereich im [Touch color]-Block klicken und dann das Pipettenwerkzeug verwenden, um die rote Farbe des **Fishhook**-Sprites auf der Bühne aufzunehmen. Wenn du willkürlich eine Farbe wählst, funktioniert dieser [Touch color]-Block nicht.



4.21 2.18 SPIEL - Nicht auf das weiße Kachel tippen

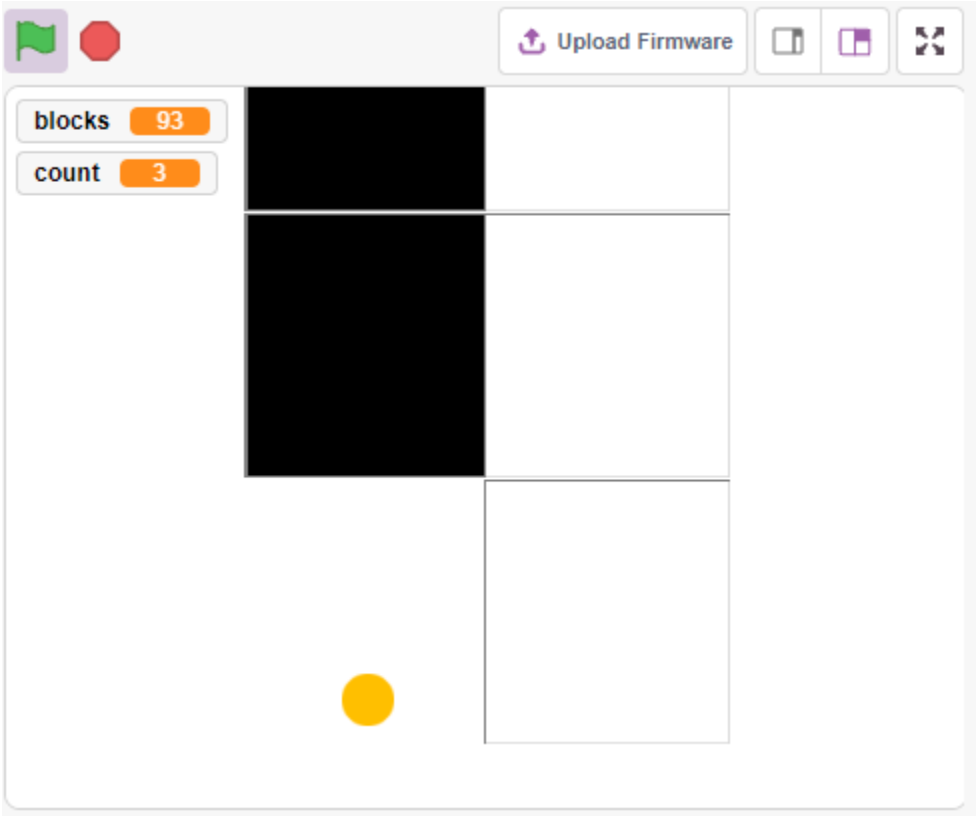
Ich bin sicher, viele von euch haben dieses Spiel auf euren Handys gespielt. In diesem Spiel geht es darum, auf zufällig erscheinende schwarze Kacheln zu tippen, um Punkte zu sammeln. Das Spiel wird immer schneller, tippt man auf weiße Kacheln oder verpasst schwarze Kacheln, ist das Spiel vorbei.

Jetzt verwenden wir PictoBlox, um es nachzubilden.

Stecke zwei IR-Hindernisvermeidungsmodule vertikal auf das Breadboard, wenn deine Hand über einem der IR-Module platziert ist, erscheint ein blinkender Punkt auf der Bühne, was einem Tipp entspricht.

Wenn der Tipp auf die schwarze Kachel geht, wird der Punktestand um 1 erhöht, berührt man die weiße Kachel, wird der Punktestand um 1 verringert.

Du musst entscheiden, ob du deine Hand über dem IR-Modul links oder über dem IR-Modul rechts platzierst, abhängig von der Position der schwarzen Kachel auf der Bühne.



4.21.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

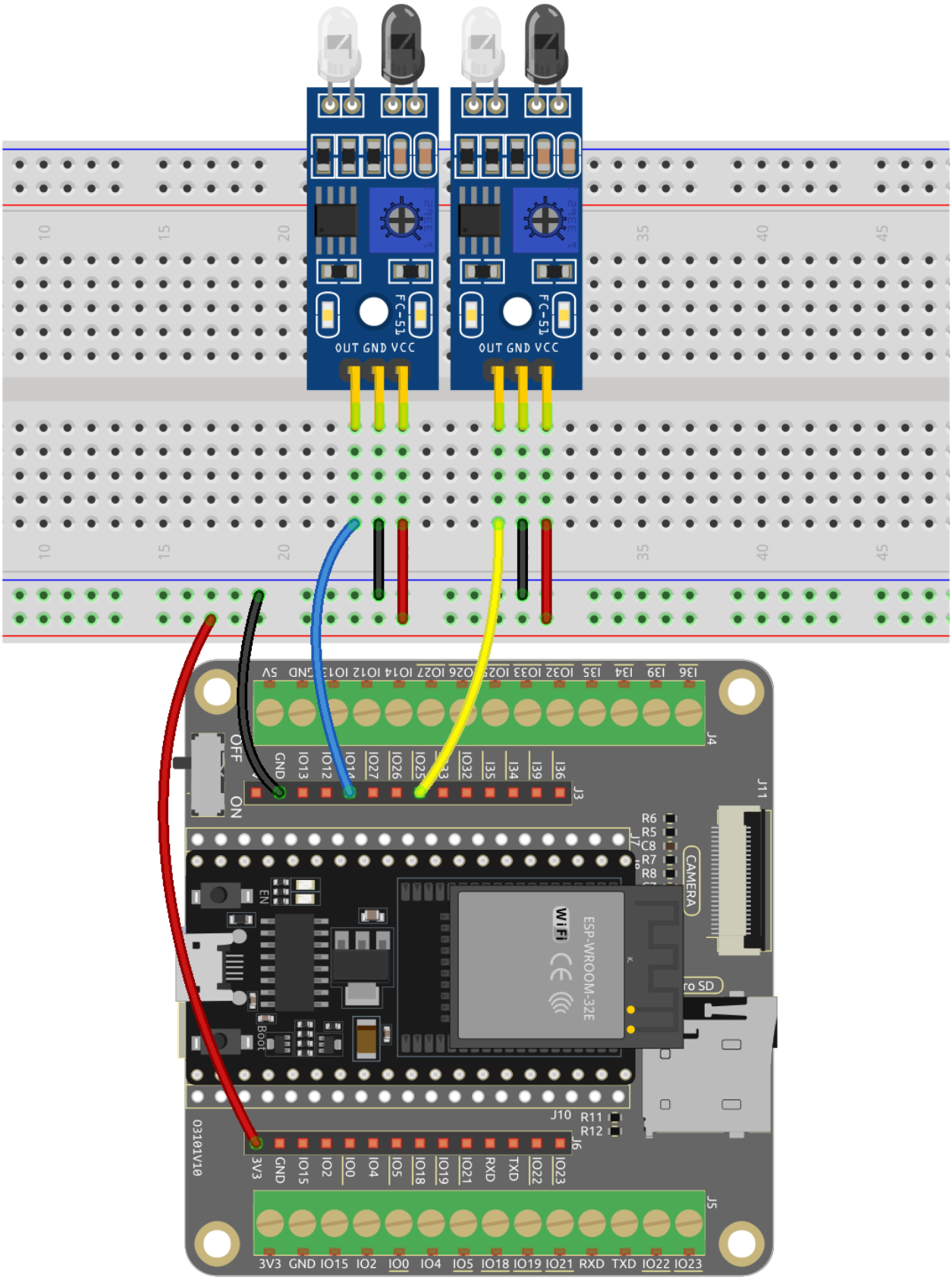
Sie können sie auch separat über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Steckbrett	
Überbrückungsdrähte	
Modul zur Hindernisvermeidung	

4.21.2 Schaltung Aufbauen

Das Hindernisvermeidungsmodul ist ein infraroter Näherungssensor mit einstellbarer Entfernung, dessen Ausgang normalerweise hoch ist und bei Erkennung eines Hindernisses niedrig wird.

Bauen Sie die Schaltung gemäß dem folgenden Diagramm auf.


FC-51
OUT GND VCC
FC-51
OUT GND VCC
ESP-WROOM-32E
CAMERA
OFF
ON
J3
J4
J5
J6
J10
J11
R11
R12
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23

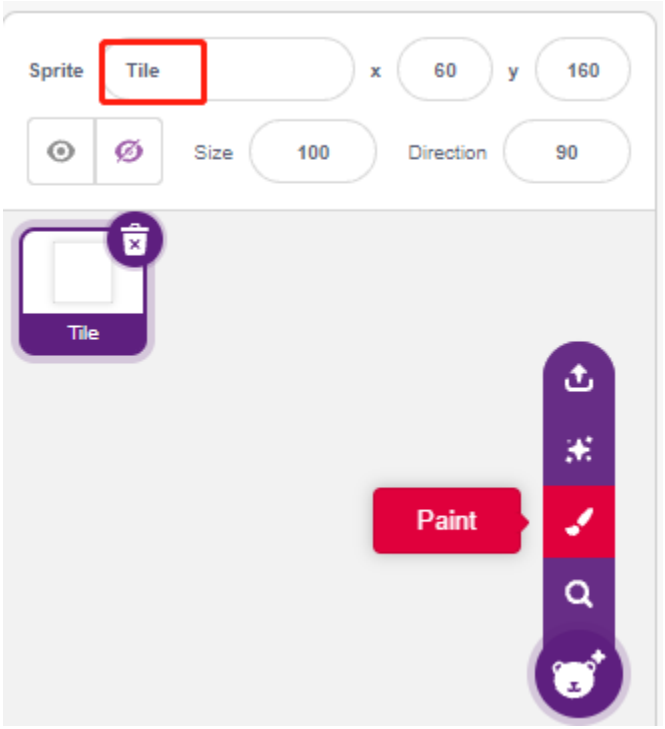
4.21.3 Programmierung

Hier benötigen wir 3 Sprites, **Tile**, **Left IR** und **Right IR**.

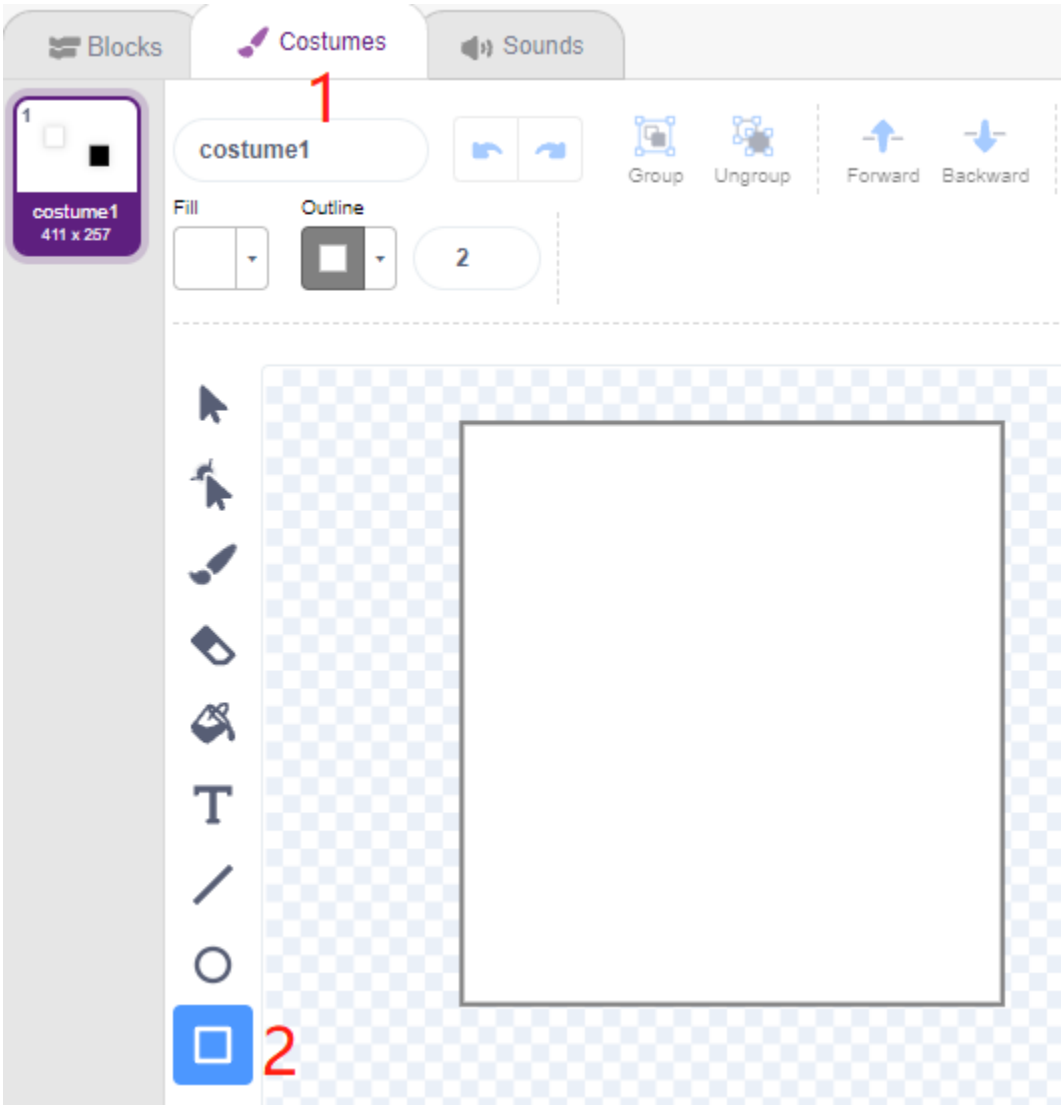
- **Tile**-Sprite: dient dazu, den Effekt abwechselnder schwarzer und weißer Kacheln nach unten zu erzielen. In der Handyspielversion dieses Spiels gibt es normalerweise 4 Spalten, hier machen wir nur zwei.
- **Left IR**-Sprite: dient dazu, den Klickeffekt zu erzielen. Wenn das linke IR-Modul deine Hand erkennt, sendet es eine Nachricht - **left** an das **Left IR**-Sprite, um es zu aktivieren. Wenn es die schwarze Kachel auf der Bühne berührt, wird der Punktestand um 1 erhöht, andernfalls wird er um 1 verringert.
- **Right IR**-Sprite: Die Funktion ist im Wesentlichen die gleiche wie bei **Left IR**, nur dass es die Nachricht **Right** empfängt.

1. Ein Kachel-Sprite malen.

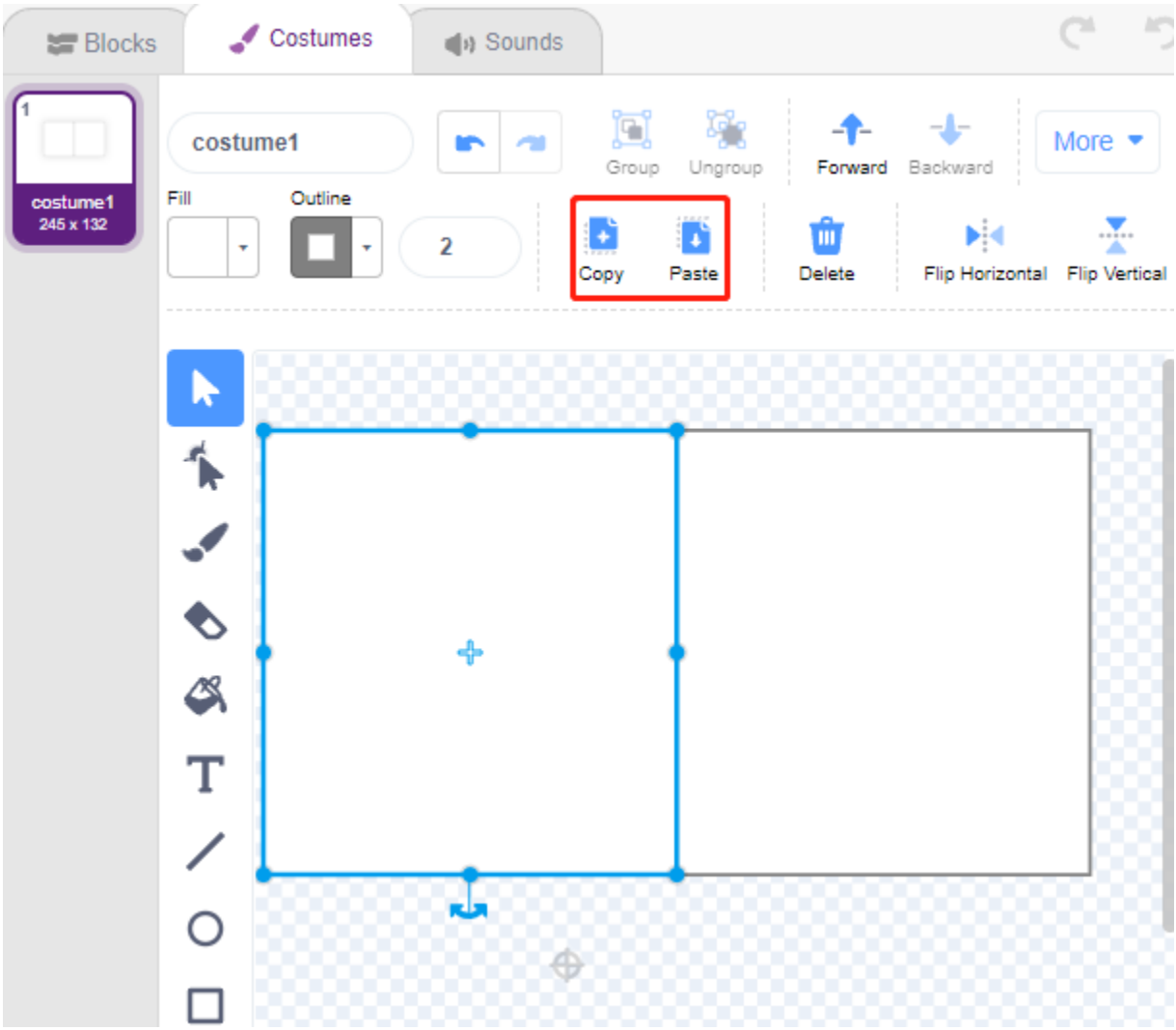
Lösche das Standard-Sprite, fahre mit der Maus über das Symbol **Add Sprite**, wähle **Paint** und ein leeres Sprite erscheint, nenne es **Tile**.



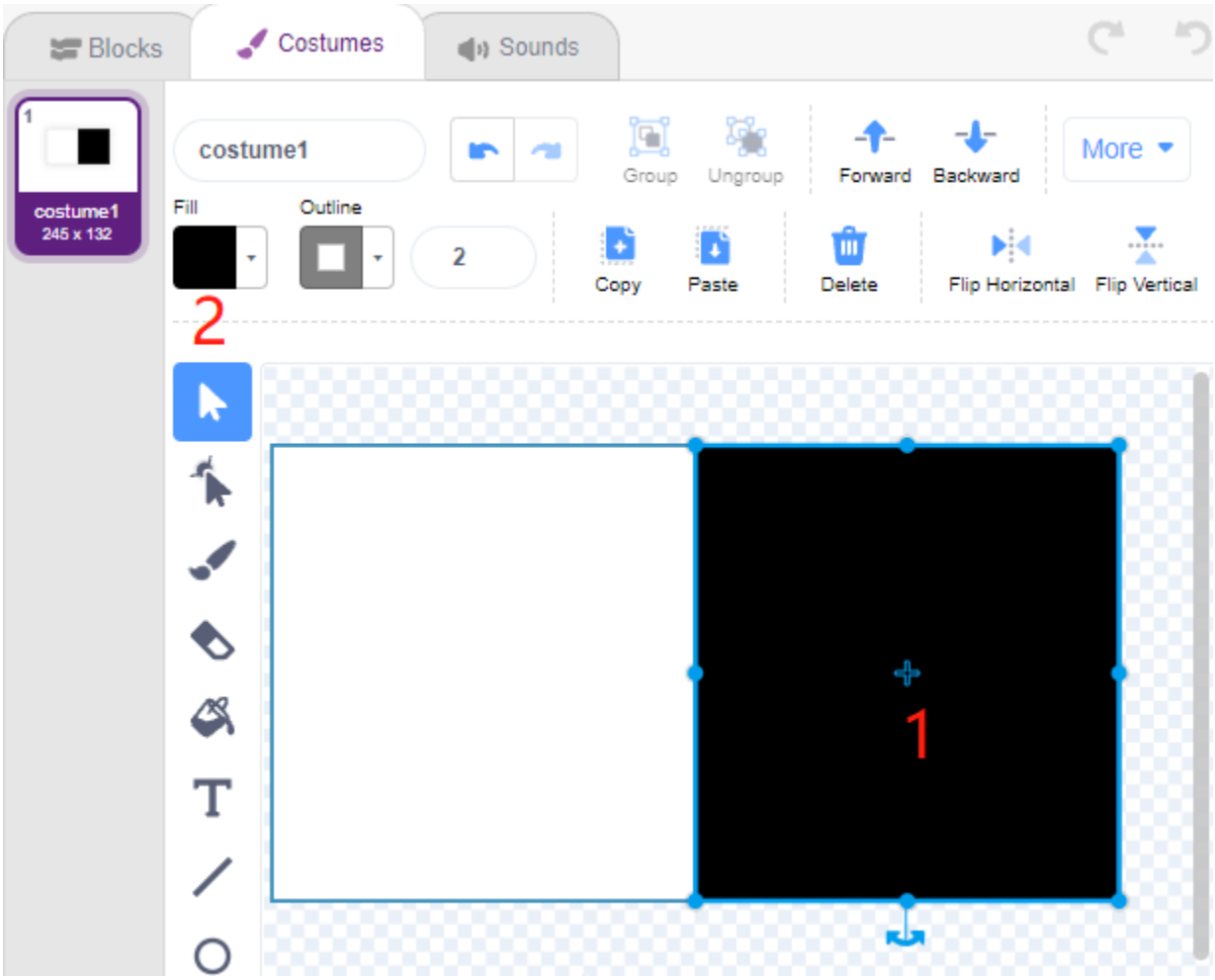
Gehe zur **Costumes**-Seite und verwende das **Rectangle**-Werkzeug, um ein Rechteck zu zeichnen.



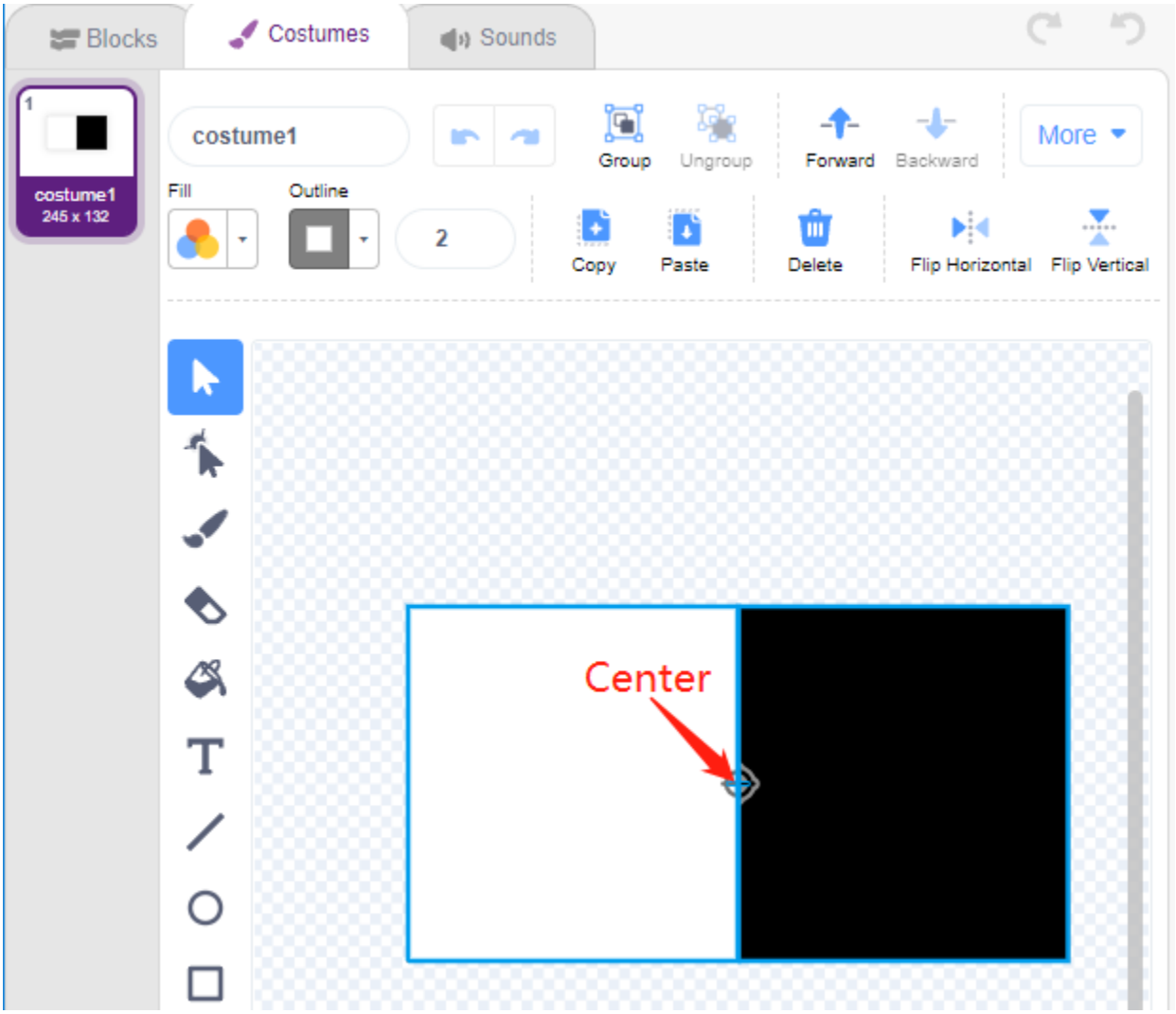
Wähle das Rechteck aus und klicke auf **Copy** -> **Paste**, um ein identisches Rechteck zu erstellen, und verschiebe dann die beiden Rechtecke in eine bündige Position.



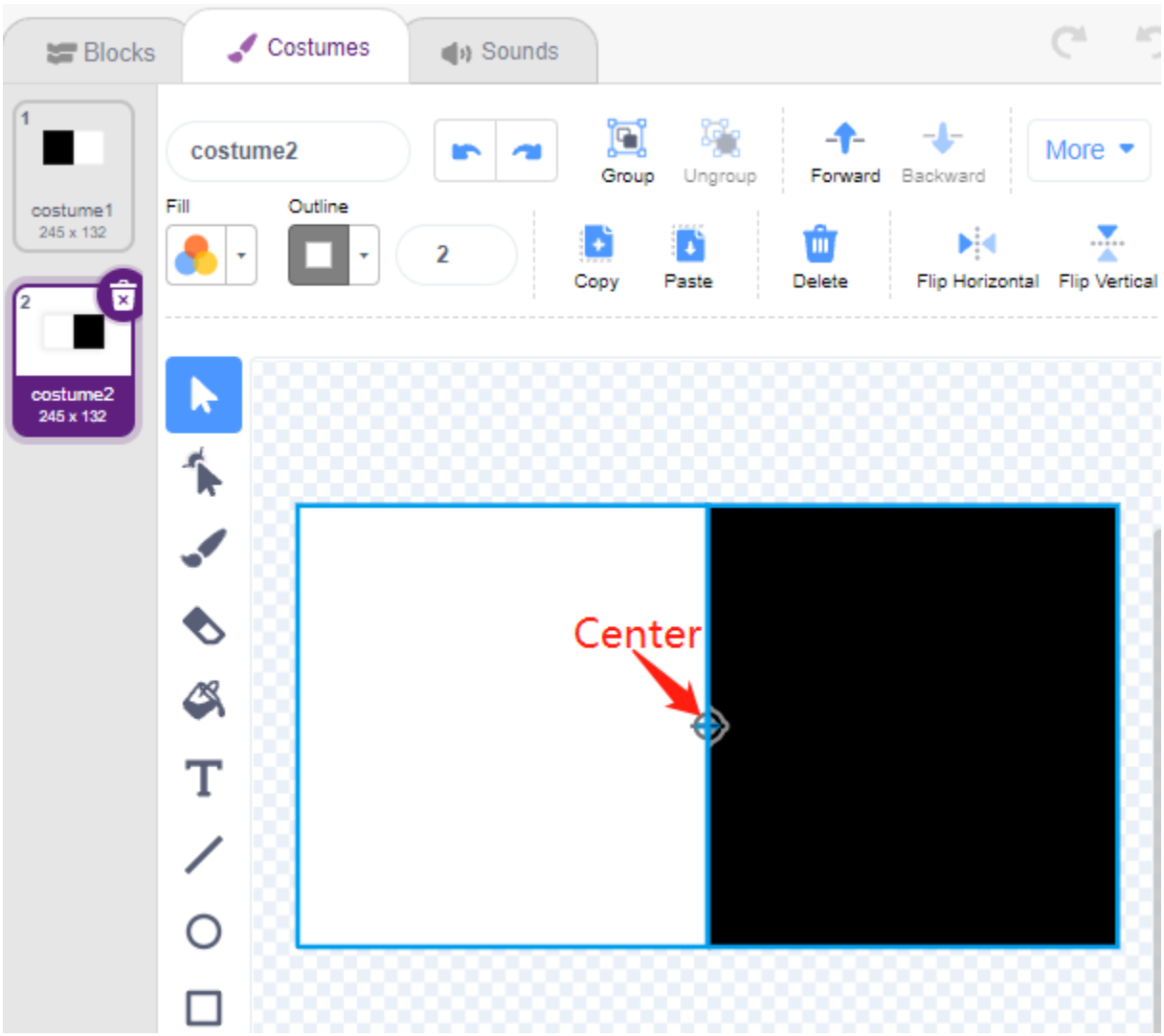
Wählen Sie eines der Rechtecke aus und färben Sie es schwarz.



Wählen Sie nun beide Rechtecke aus und verschieben Sie sie so, dass ihre Mittelpunkte mit dem Zentrum der Leinwand übereinstimmen.

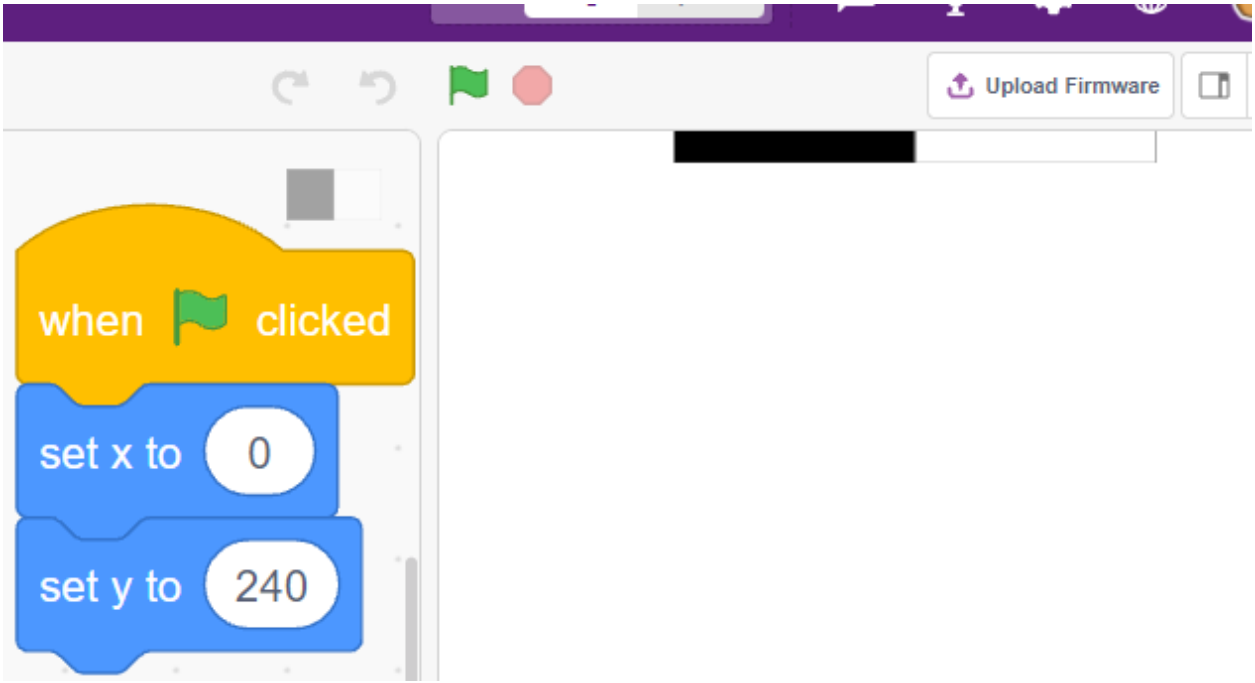


Duplizieren Sie das Kostüm1, indem Sie die Füllfarben der beiden Rechtecke abwechseln. Zum Beispiel ist die Füllfarbe von Kostüm1 links weiß und rechts schwarz, während die Füllfarbe von Kostüm2 links schwarz und rechts weiß ist.



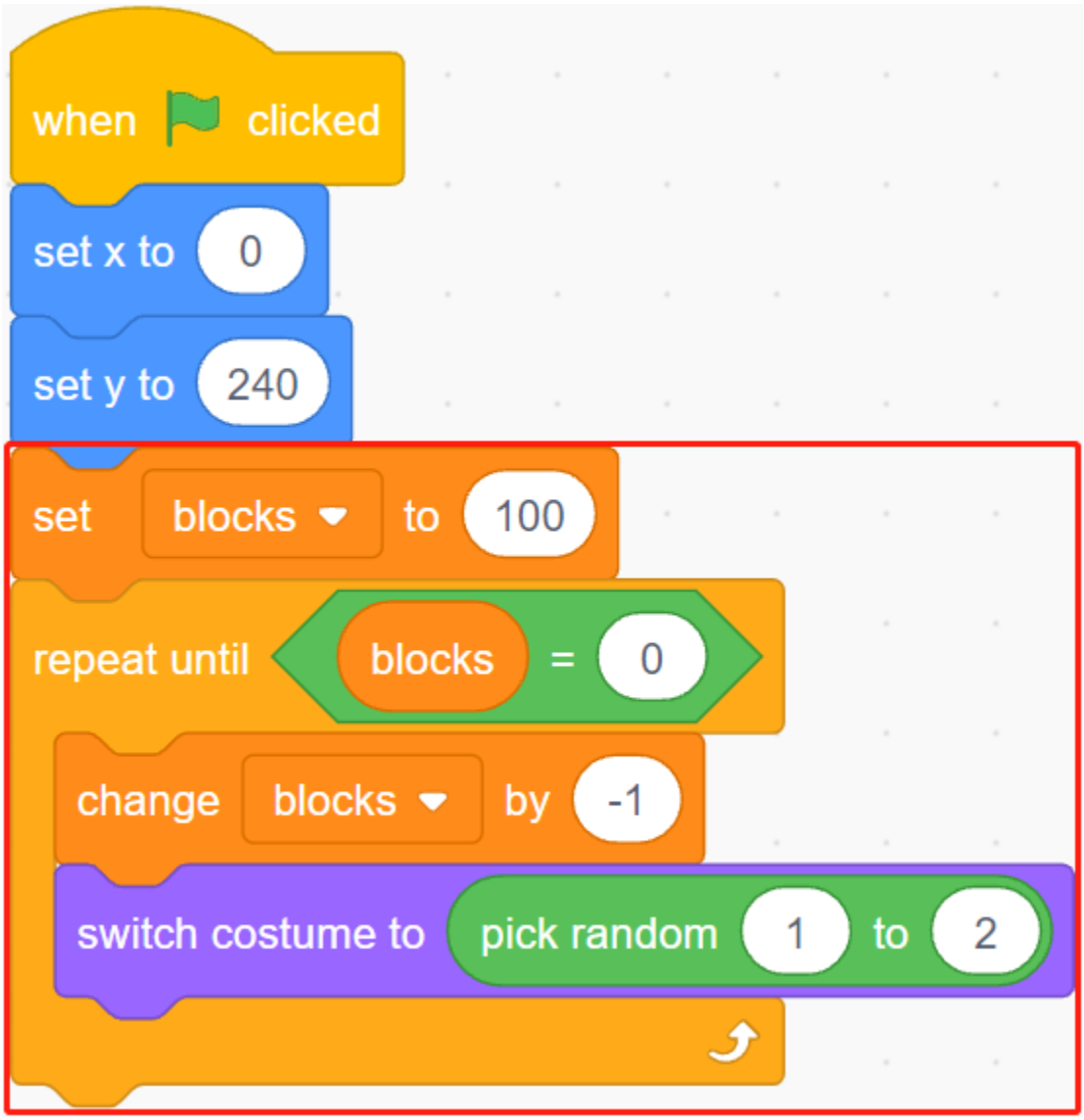
2. Programmieren des Fliesen-Sprites

Gehen Sie zurück zur Seite **Blocks** und setzen Sie die Anfangsposition des **Tile**-Sprites so, dass es sich oben auf der Bühne befindet.

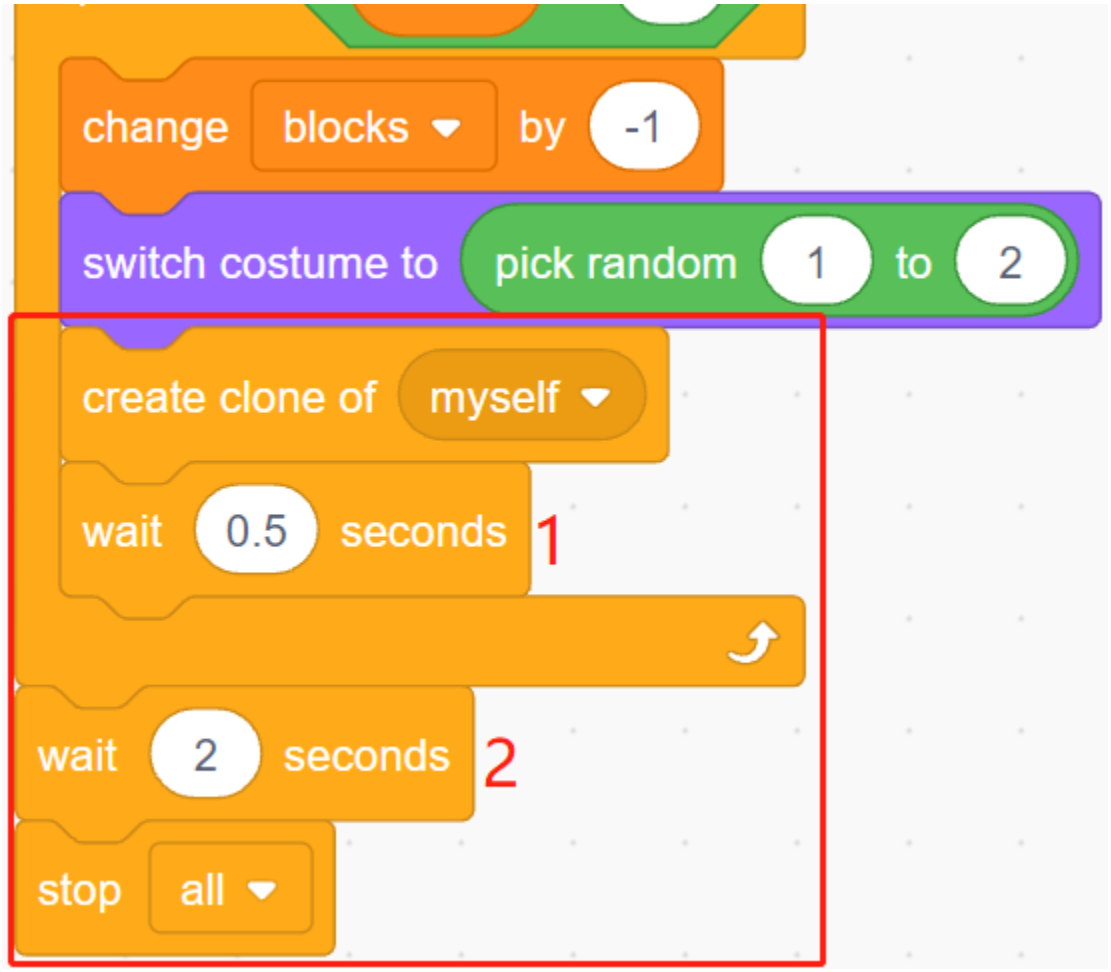


Erstellen Sie eine Variable - **blocks** und geben Sie ihr einen Anfangswert, um die Anzahl der Erscheinungen des **Tile**-Sprites zu bestimmen. Verwenden Sie den Block [repeat until], damit die Variable **blocks** allmählich abnimmt, bis **blocks** 0 ist. Währenddessen soll das Sprite **Tile** zufällig sein Kostüm wechseln.

Nach dem Klicken auf die grüne Fahne sehen Sie, wie das **Tile**-Sprite auf der Bühne schnell die Kostüme wechselt.

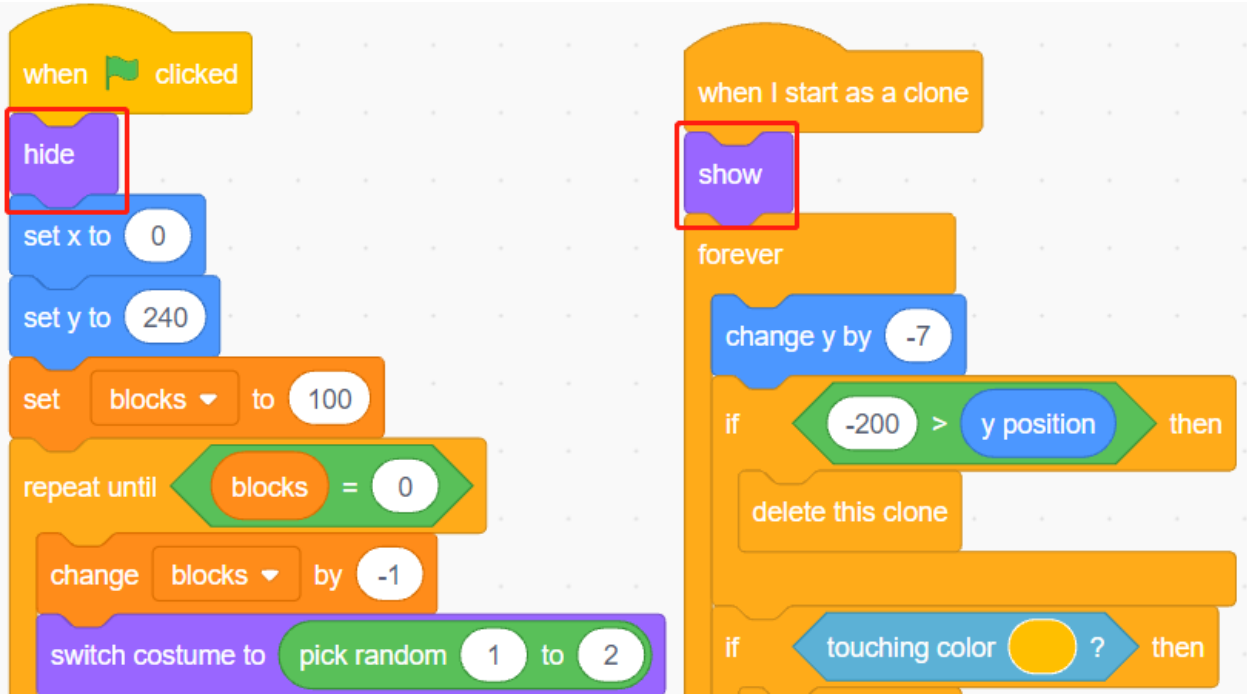


Erstellen Sie Klone des **Tile**-Sprites, während die Variable **blocks** abnimmt, und stoppen Sie das Skript, wenn Blöcke 0 ist. Zwei Blöcke [wait () seconds] werden hier verwendet, der erste begrenzt das Intervall zwischen den Klonen von **Tile's** und der zweite lässt die Variable Blöcke auf 0 sinken, ohne das Programm sofort zu stoppen, damit das letzte Fliesen-Sprite genug Zeit hat, sich zu bewegen.



Programmieren Sie nun den Klon des **Tile**-Sprites so, dass es langsam nach unten bewegt wird und löschen Sie es, wenn es den Boden der Bühne erreicht. Die Änderung der y-Koordinate beeinflusst die Fallgeschwindigkeit, je größer der Wert, desto schneller die Fallgeschwindigkeit.

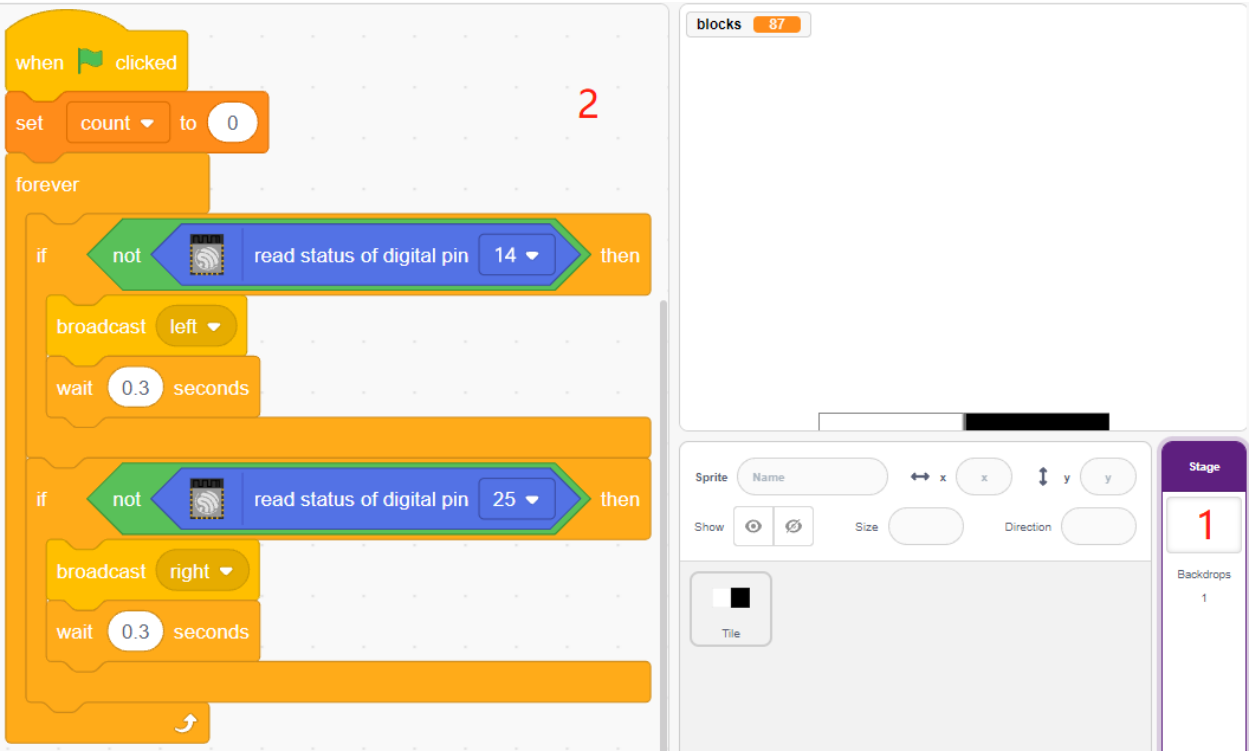
Verstecken Sie das Original und zeigen Sie den Klon.



3. Auslesen der Werte der 2 IR-Module

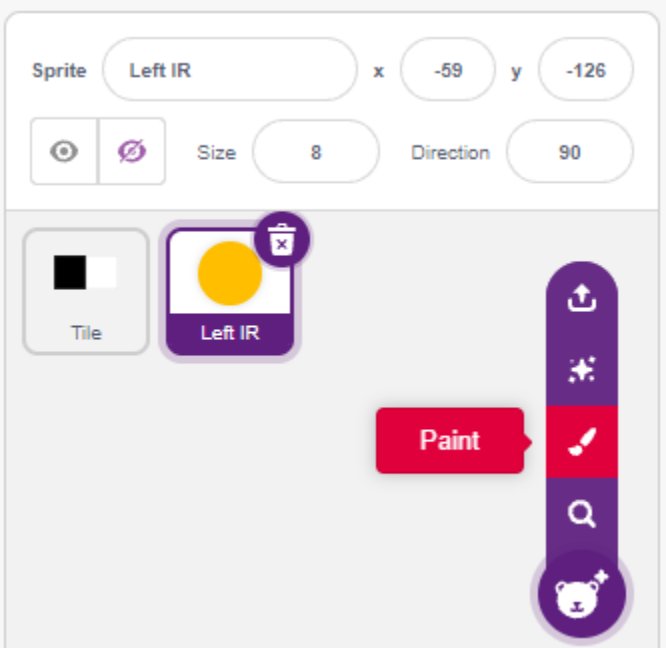
Lesen Sie im Hintergrund die Werte der 2 IR-Module aus und führen Sie die entsprechenden Aktionen durch.

- Wenn das linke IR-Hindernisvermeidungsmodul Ihre Hand erkennt, senden Sie eine Nachricht - **left**.
- Wenn das rechte IR-Vermeidungsmodul Ihre Hand erkennt, senden Sie eine Nachricht - **right**.

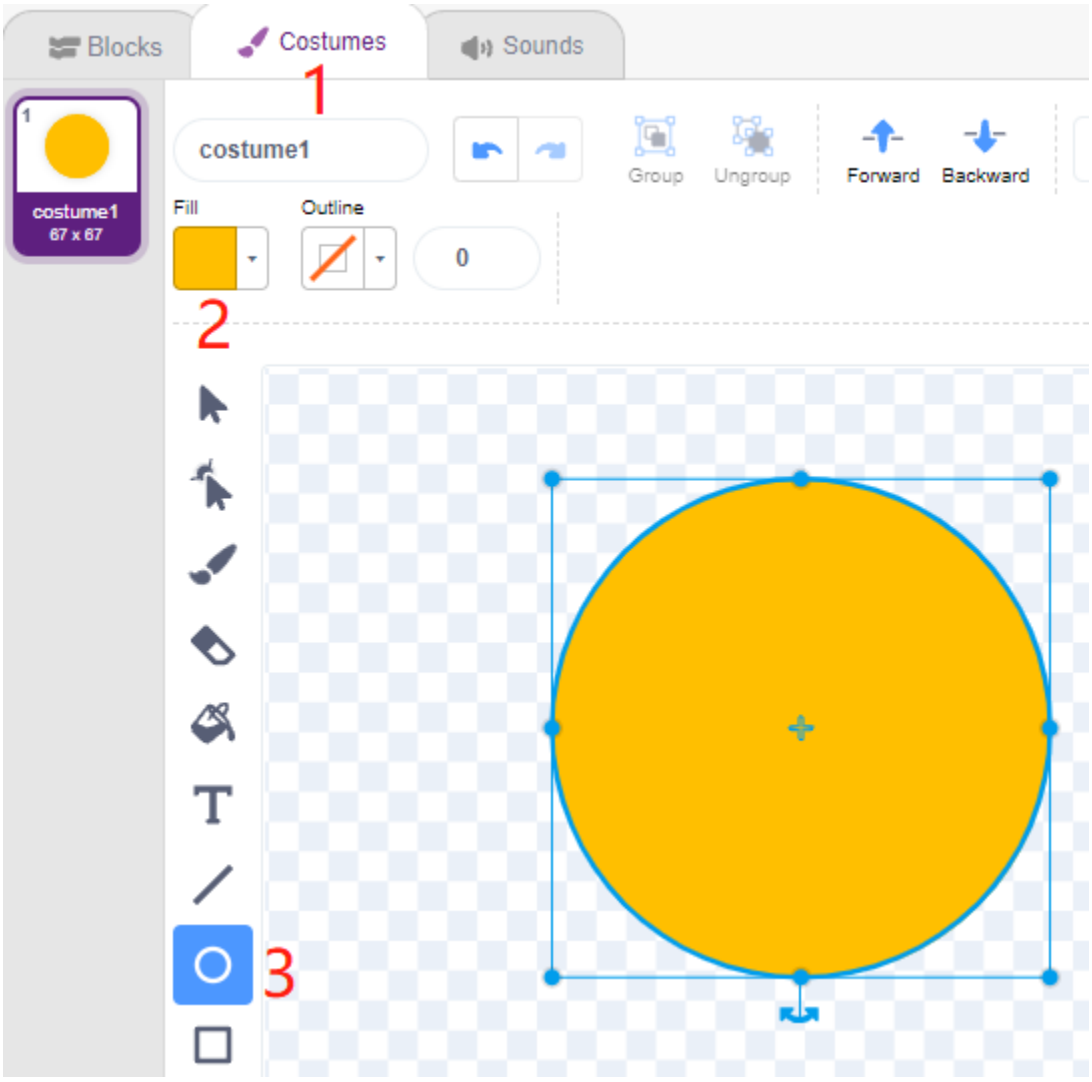


4. Links-IR-Sprite

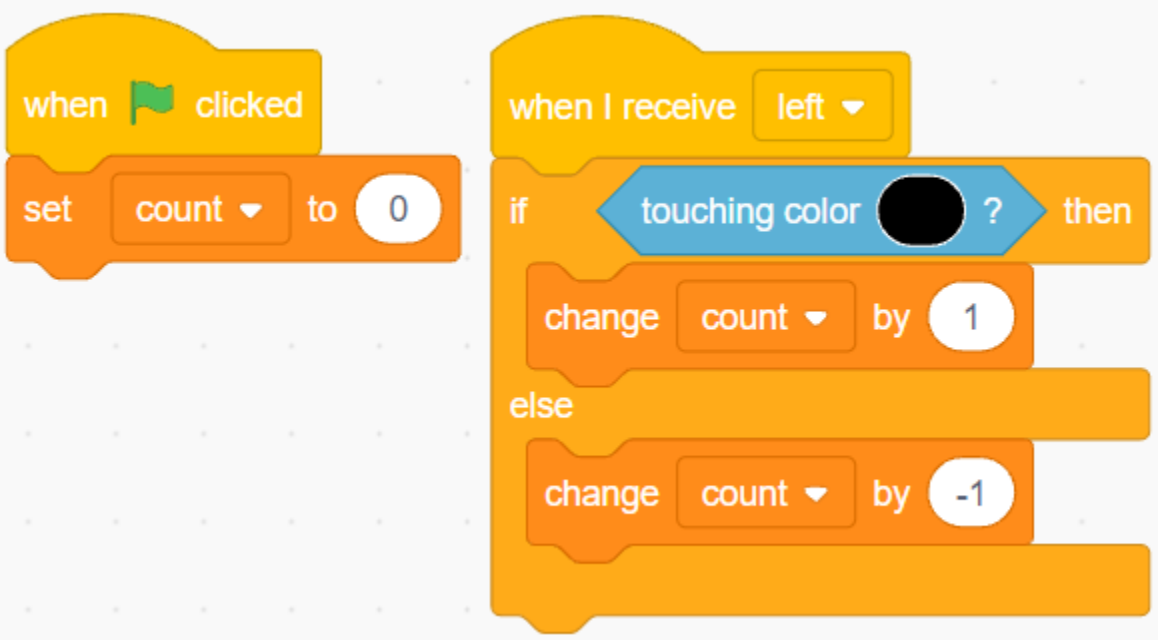
Fahren Sie mit der Maus über das Symbol **Add sprite** und wählen Sie **Paint**, um ein neues Sprite namens **Left IR** zu erstellen.



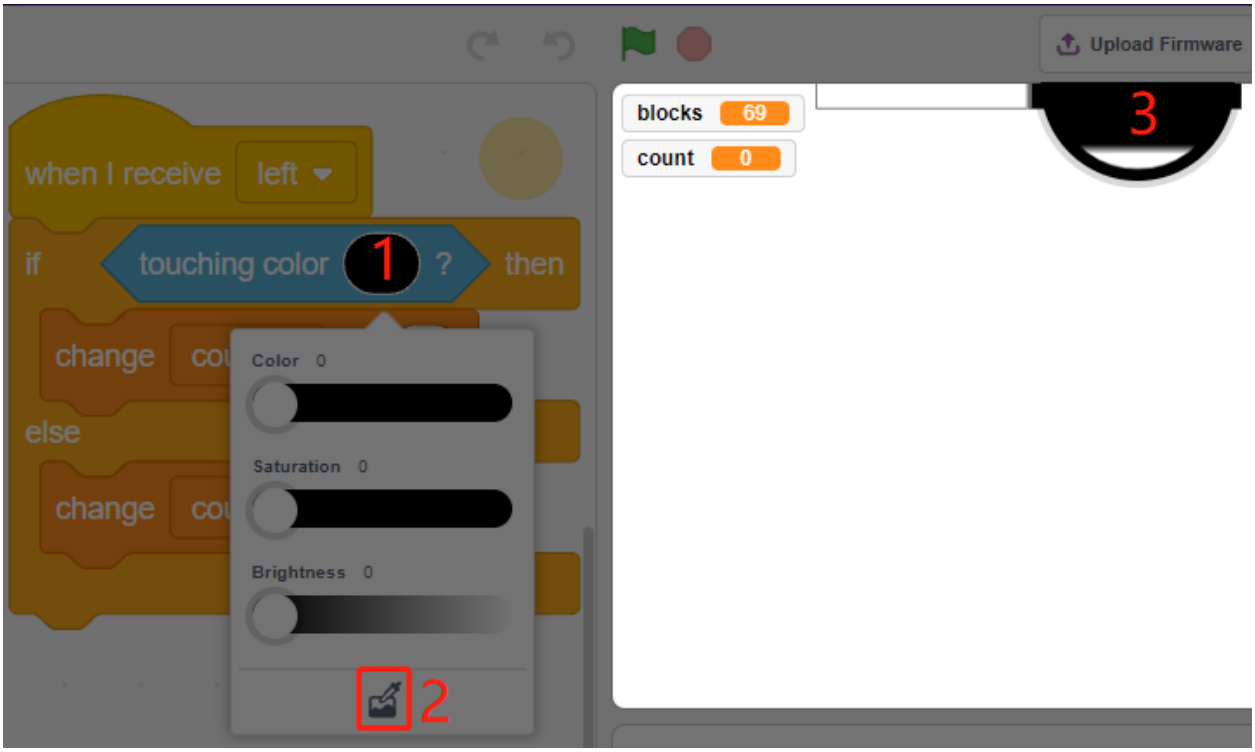
Gehen Sie zur Seite **Costumes** des **Left IR**-Sprites, wählen Sie eine Füllfarbe (irgendeine Farbe außer Schwarz und Weiß) und zeichnen Sie einen Kreis.



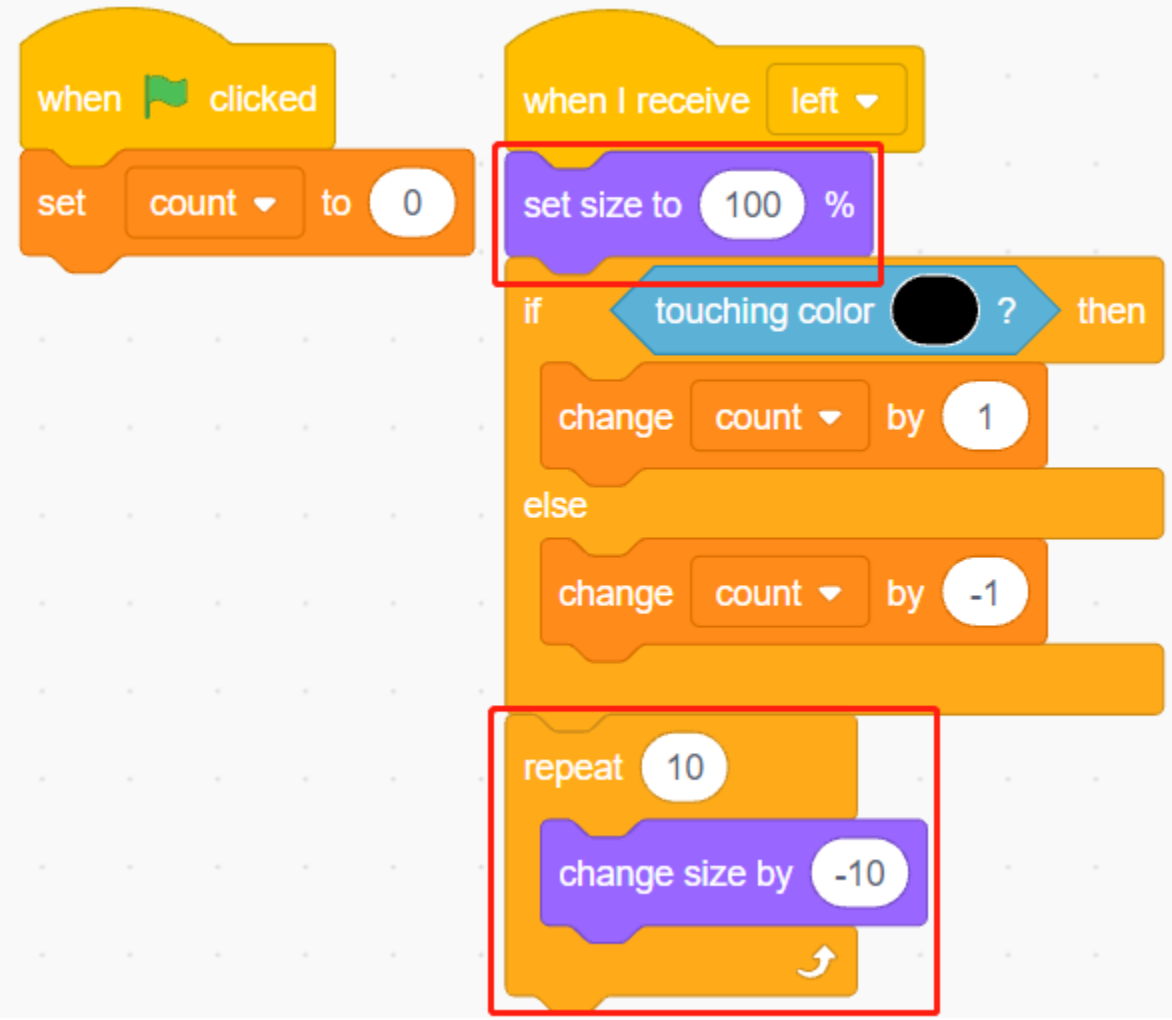
Beginnen Sie nun mit dem Programmieren des **Left IR**-Sprites. Wenn die Nachricht - **left** empfangen wird (das IR-Empfängermodul links erkennt ein Hindernis), dann prüfen Sie, ob der schwarze Block des **Tile**-Sprites berührt wird, und wenn ja, lassen Sie die Variable **count** um 1 erhöhen, andernfalls um 1 verringern.



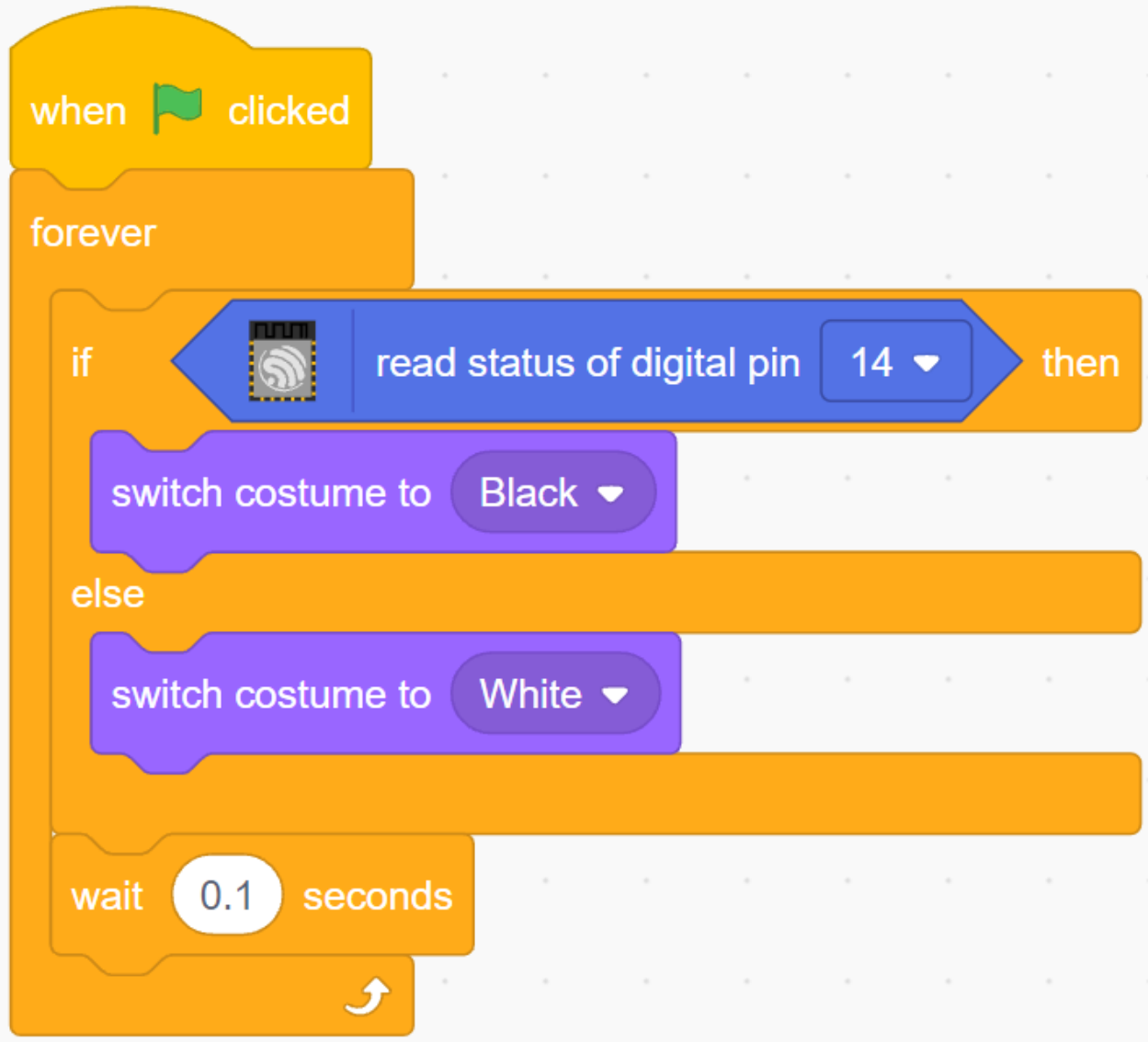
Bemerkung: Sie müssen das **Tile**-Sprite auf der Bühne erscheinen lassen und dann die Farbe des schwarzen Blocks im **Tile**-Sprite aufnehmen.



Nun realisieren Sie den Sensor-Effekt (Vergrößern und Verkleinern) für **Left IR**.

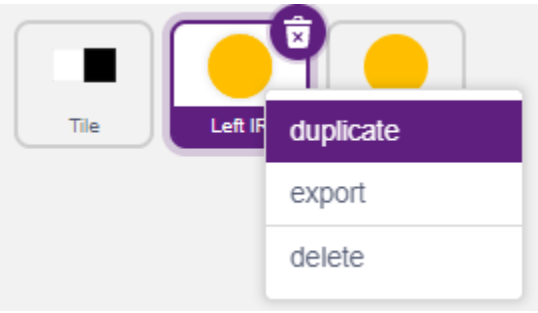


Lassen Sie das **Left IR**-Sprite verschwinden, wenn auf die grüne Fahne geklickt wird, erscheinen, wenn die Nachricht - **left** empfangen wird, und schließlich wieder verschwinden.

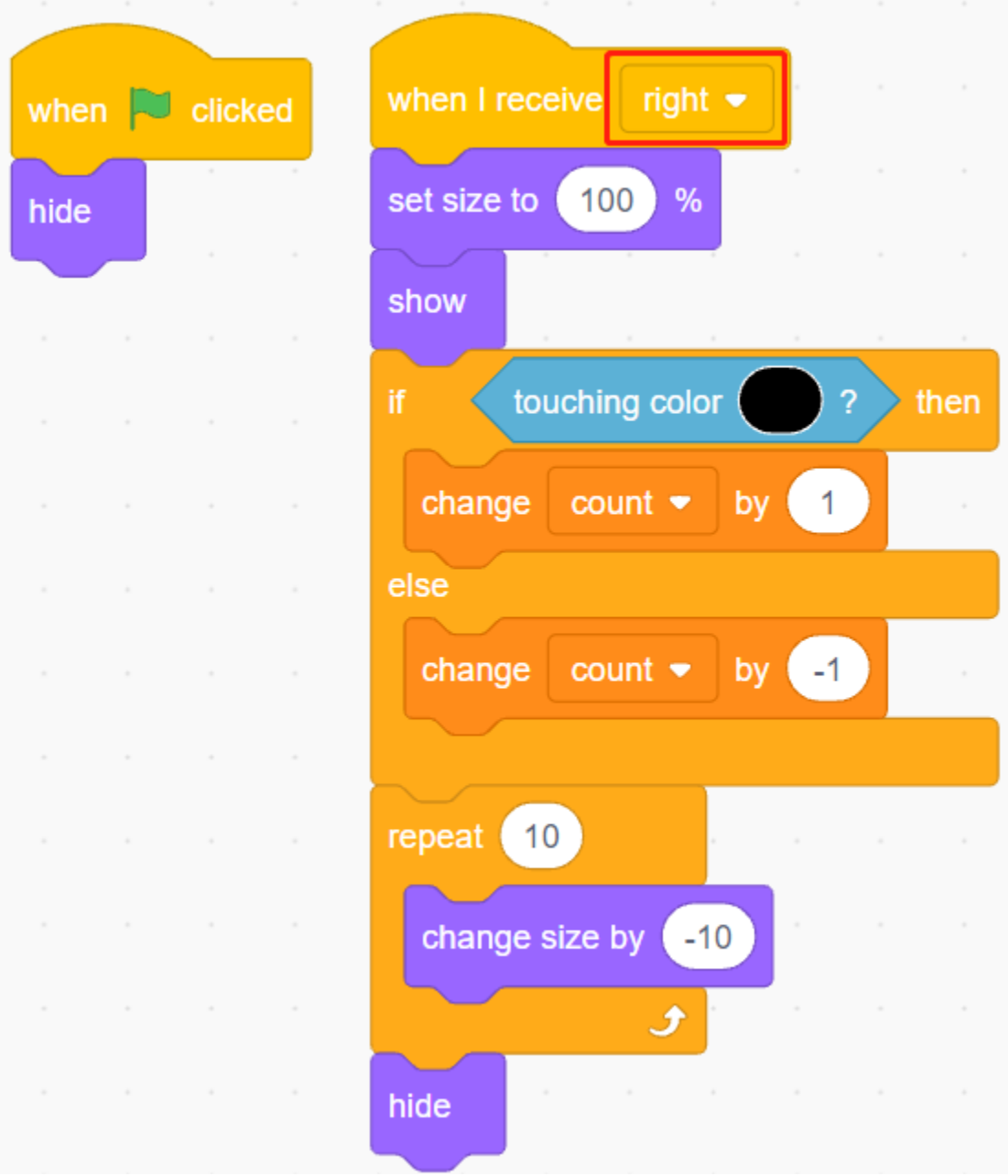


5. Rechts-IR-Sprite

Kopieren Sie das **Left IR**-Sprite und benennen Sie es in **Right IR** um.



Ändern Sie dann die empfangene Nachricht in - **right**.



Nun ist die gesamte Programmierung abgeschlossen und Sie können auf die grüne Fahne klicken, um das Skript auszuführen.

4.22 2.19 SPIEL - Schütze Dein Herz

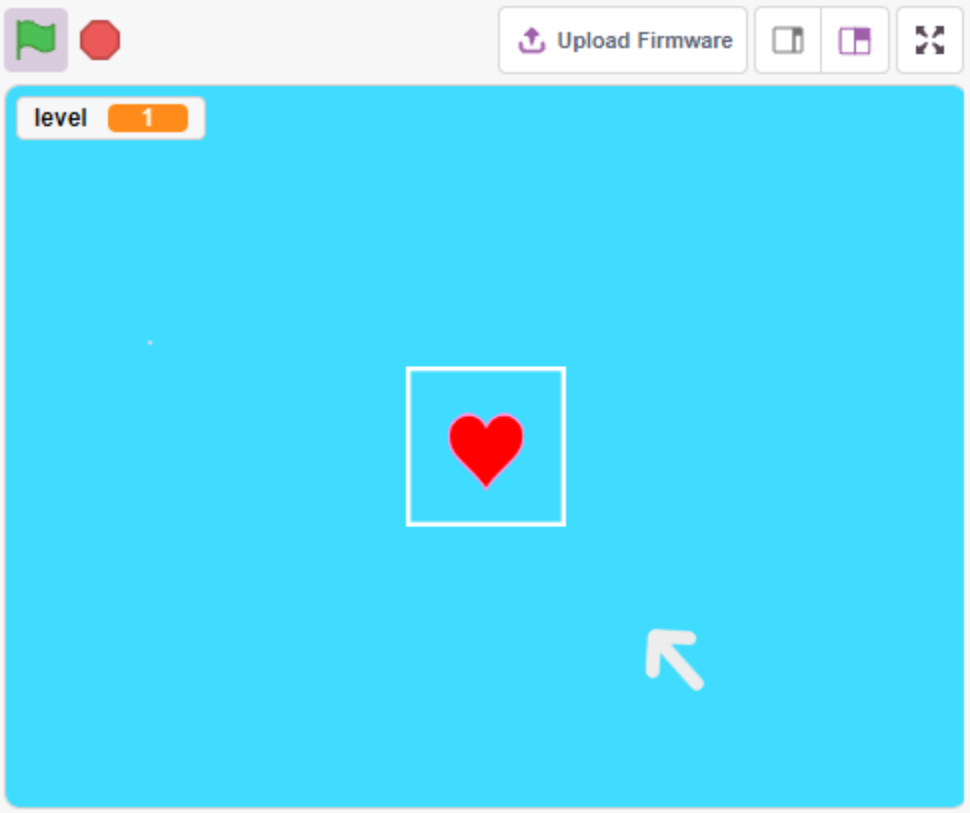
In diesem Projekt erstellen wir ein Spiel, das die Reaktionsgeschwindigkeit testet.

Auf der Bühne befindet sich ein Herz, geschützt in einem rechteckigen Kasten, und Pfeile fliegen von jeder Position der Bühne auf dieses Herz zu. Die Farbe der Pfeile wechselt zufällig zwischen Schwarz und Weiß, und die Pfeile fliegen immer schneller.

Wenn die Farbe des rechteckigen Kastens und die Pfeilfarbe gleich sind, wird der Pfeil außerhalb blockiert und das Level um 1 erhöht; sind die Farben nicht gleich, durchdringt der Pfeil das Herz und das Spiel ist beendet.

Hier wird die Farbe des Rechteckkastens durch das Linienverfolgungsmodul gesteuert. Wenn das Modul auf einer schwarzen Oberfläche (einer reflektierenden Oberfläche) platziert wird, ist die Farbe des Rechteckkastens schwarz, ansonsten weiß.

Sie müssen also entscheiden, ob Sie das Linienverfolgungsmodul auf eine weiße oder schwarze Oberfläche legen, je nach Pfeilfarbe.



4.22.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

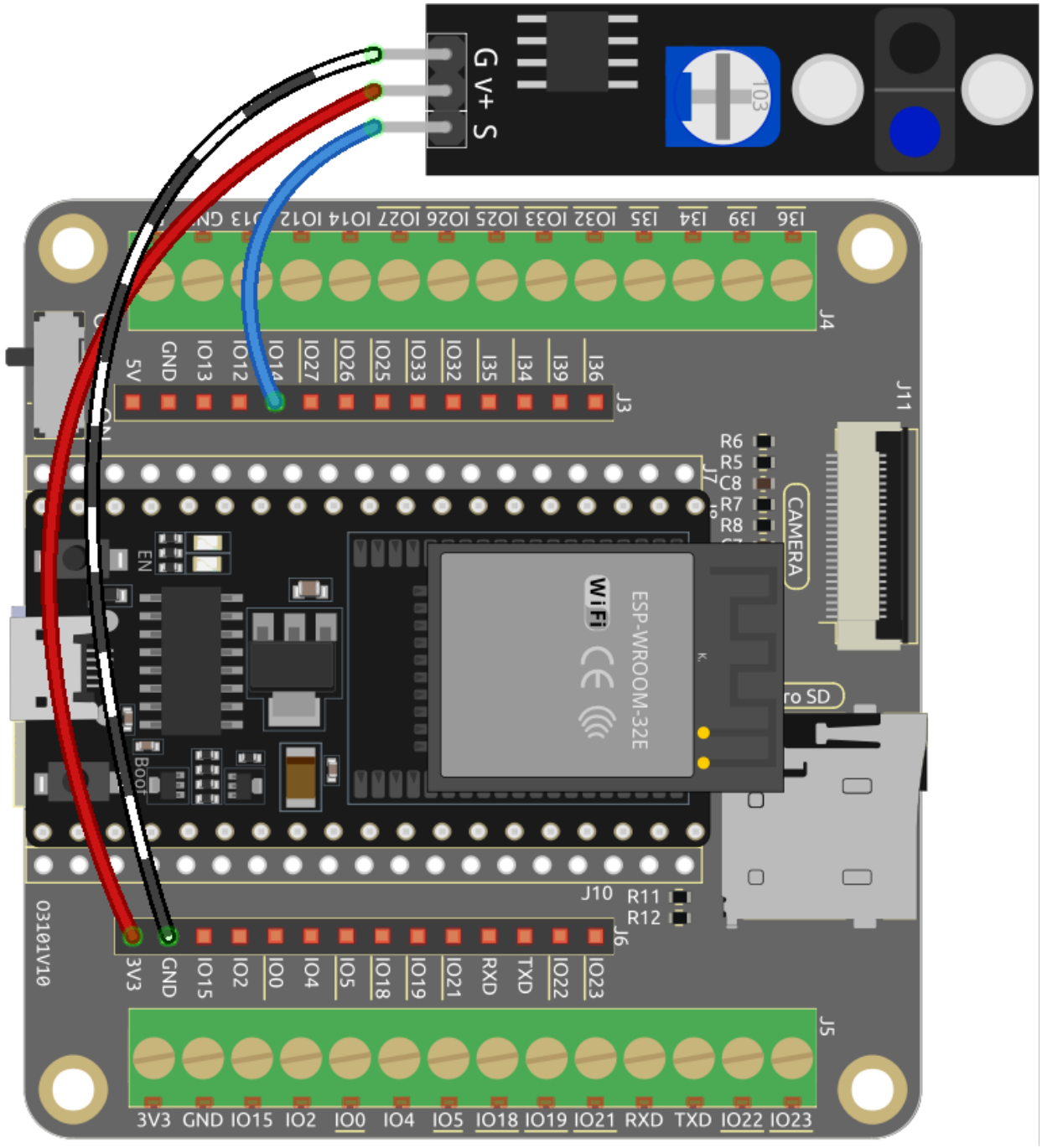
Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Linienverfolgungsmodul	

4.22.2 Schaltung Aufbauen

Dies ist ein digitales Linienverfolgungsmodul, das bei Erkennung einer schwarzen Linie 1 ausgibt; bei einer weißen Linie gibt es einen Wert von 0 aus. Zusätzlich können Sie seine Erfassungsentfernung über das Potentiometer auf dem Modul einstellen.

Bauen Sie den Schaltkreis nun gemäß der untenstehenden Abbildung auf.



Bemerkung: Bevor Sie mit dem Projekt beginnen, müssen Sie die Empfindlichkeit des Moduls einstellen.

Verkabeln Sie gemäß der obigen Abbildung und schalten Sie dann das R3-Board ein (entweder direkt in das USB-Kabel oder das 9V-Batterieknopfkabel), ohne den Code hochzuladen.

Kleben Sie nun ein schwarzes Isolierband auf den Schreibtisch, platzieren Sie das Linienverfolgungsmodul in einer Höhe von 2 cm über dem Schreibtisch.

Beobachten Sie mit dem Sensor nach unten gerichtet die Signalleuchte auf dem Modul, um sicherzustellen, dass sie auf dem weißen Tisch aufleuchtet und auf dem schwarzen Band erlischt.

Wenn nicht, müssen Sie das Potentiometer auf dem Modul so einstellen, dass es den obigen Effekt erzielen kann.

4.22.3 Programmierung

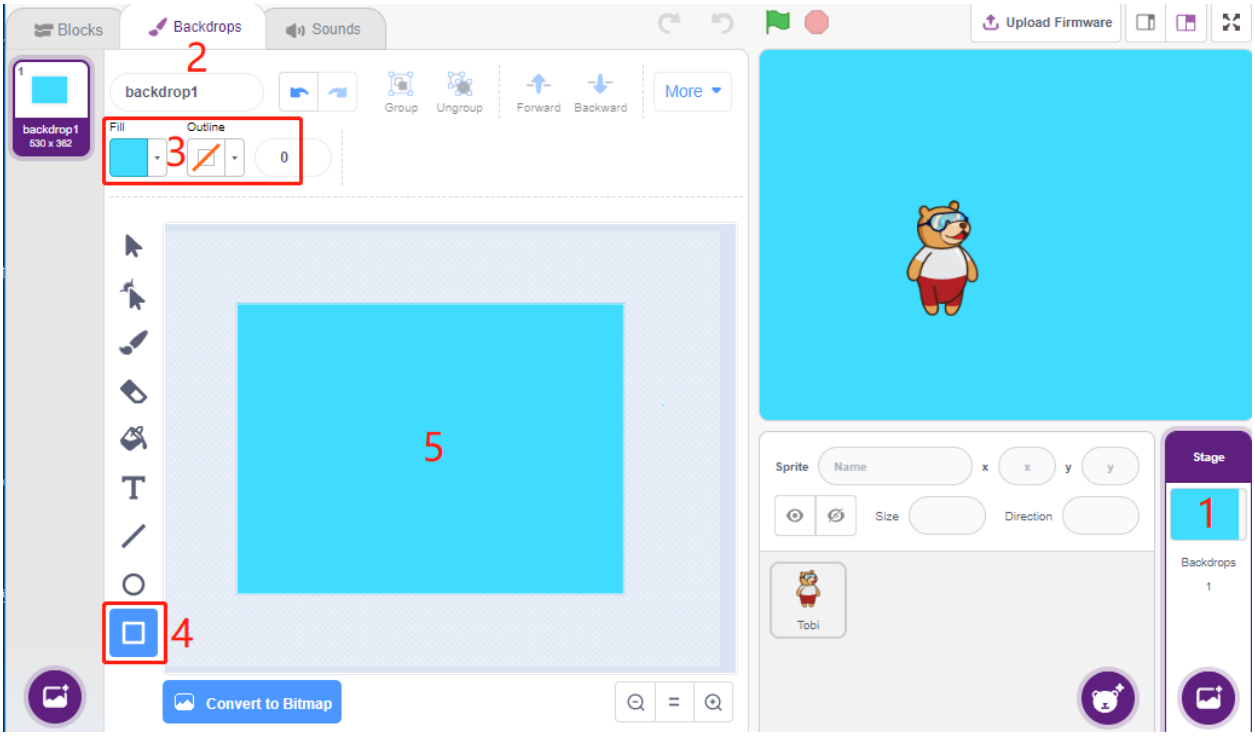
Hier müssen wir 3 Sprites erstellen: **Heart**, **Square Box** und **Arrow1**.

- **Heart**: bleibt in der Mitte der Bühne stehen, wenn es vom **Arrow1**-Sprite berührt wird, ist das Spiel vorbei.
- **Square Box**: Es gibt zwei Arten von Kostümen, schwarz und weiß, und sie wechselt die Kostüme entsprechend dem Wert des Linienverfolgungsmoduls.
- **Arrow**: fliegt von jeder Position in Schwarz/Weiß zur Mitte der Bühne; stimmt seine Farbe mit der Farbe des **Square Box**-Sprites überein, wird er blockiert und fliegt erneut von einer zufälligen Position zur Mitte der Bühne; stimmt seine Farbe nicht mit der des **Square Box**-Sprites überein, durchquert er das **Heart**-Sprite und das Spiel ist vorbei.

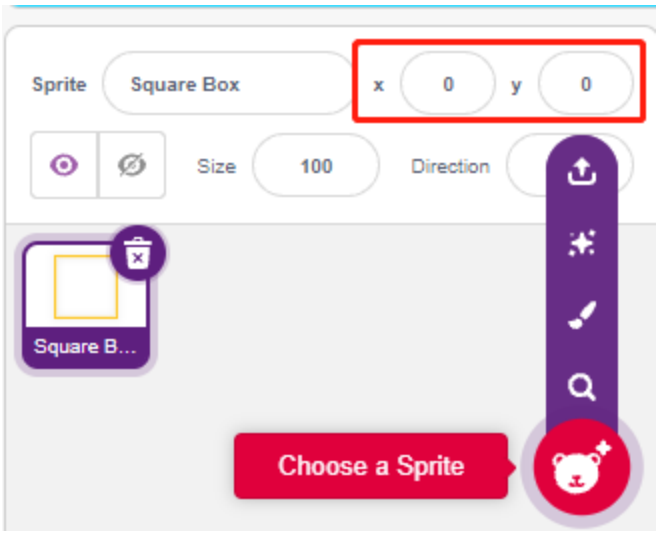
1. Quadratische Box-Sprite hinzufügen

Da das Arrow1- und Quadratische Box-Sprite beide weiße Kostüme haben, um sie auf der Bühne darzustellen, füllen Sie jetzt den Hintergrund mit einer Farbe, die jede Farbe außer Schwarz, Weiß und Rot sein kann.

- Klicken Sie auf **Backdrop1**, um zur Seite **Backdrops** zu gelangen.
- Wählen Sie die Farbe aus, mit der Sie füllen möchten.
- Verwenden Sie das **Rectangle**-Werkzeug, um ein Rechteck in der gleichen Größe wie die Zeichenfläche zu zeichnen.

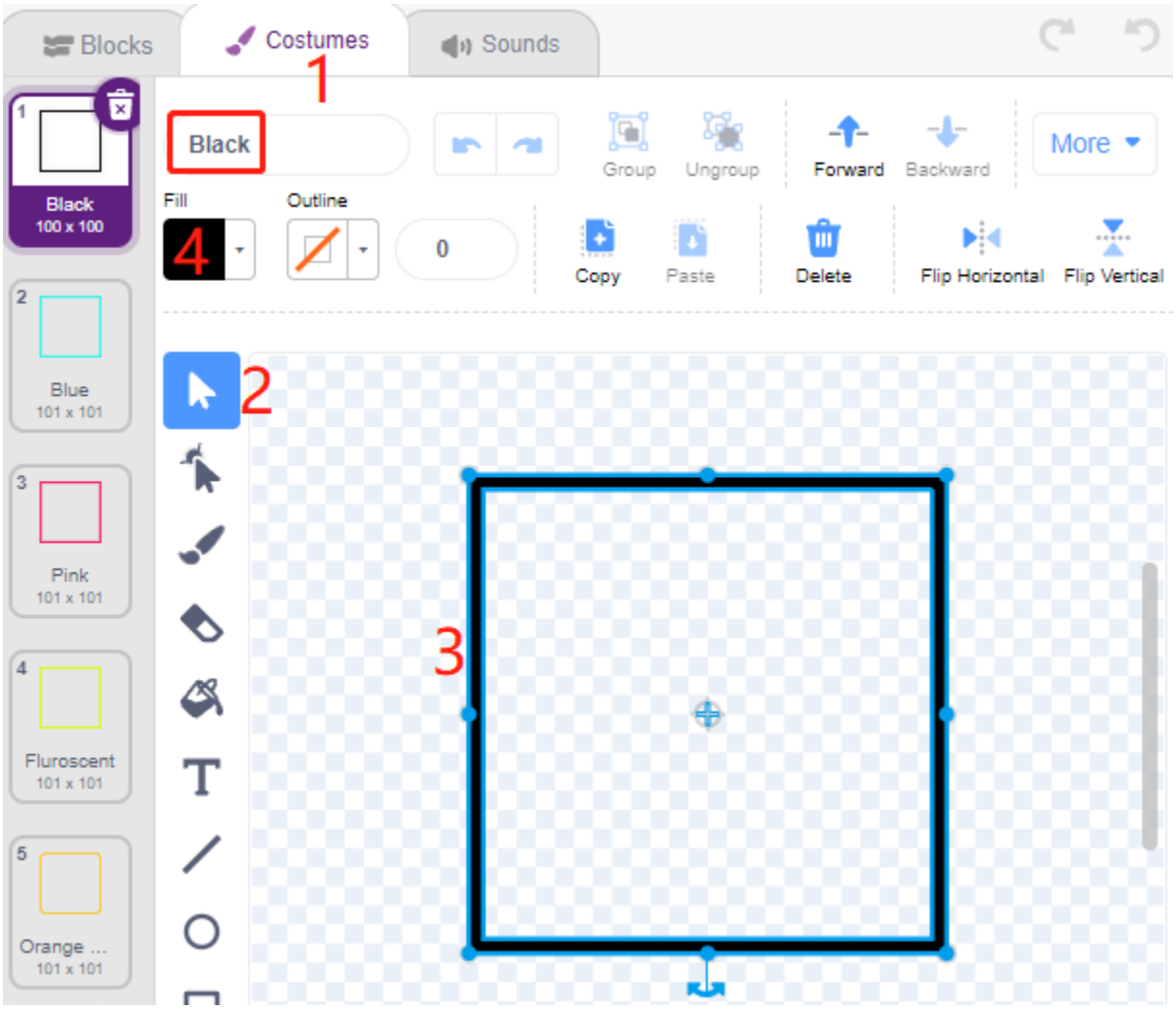


Löschen Sie das Standard-Sprite, verwenden Sie den Button **Choose a Sprite**, um das **Square Box**-Sprite hinzuzufügen, und setzen Sie seine x- und y-Koordinaten auf (0, 0).

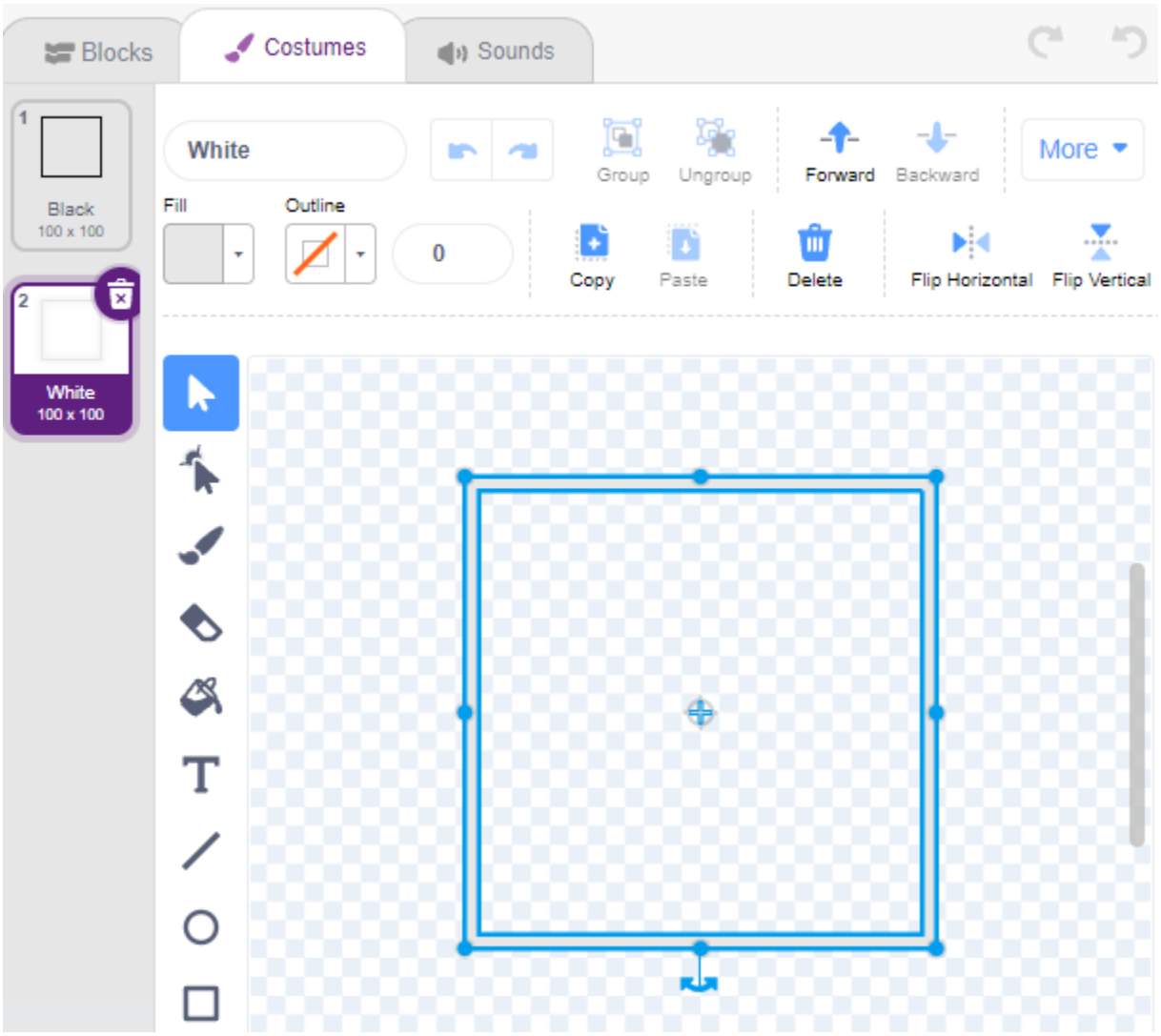


Gehen Sie zur Seite **Costumes** des **Square Box**-Sprites und stellen Sie die schwarz-weißen Kostüme ein.

- Klicken Sie das Auswahlwerkzeug
- Wählen Sie das Rechteck auf der Leinwand
- Wählen Sie die Füllfarbe Schwarz
- und benennen Sie das Kostüm **Black**

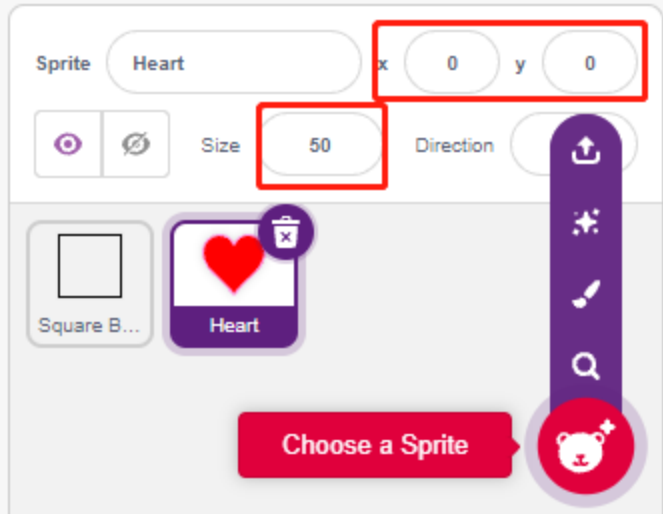


Wählen Sie das zweite Kostüm, stellen Sie die Füllfarbe auf Weiß, benennen Sie es Weiß und löschen Sie die restlichen Kostüme.

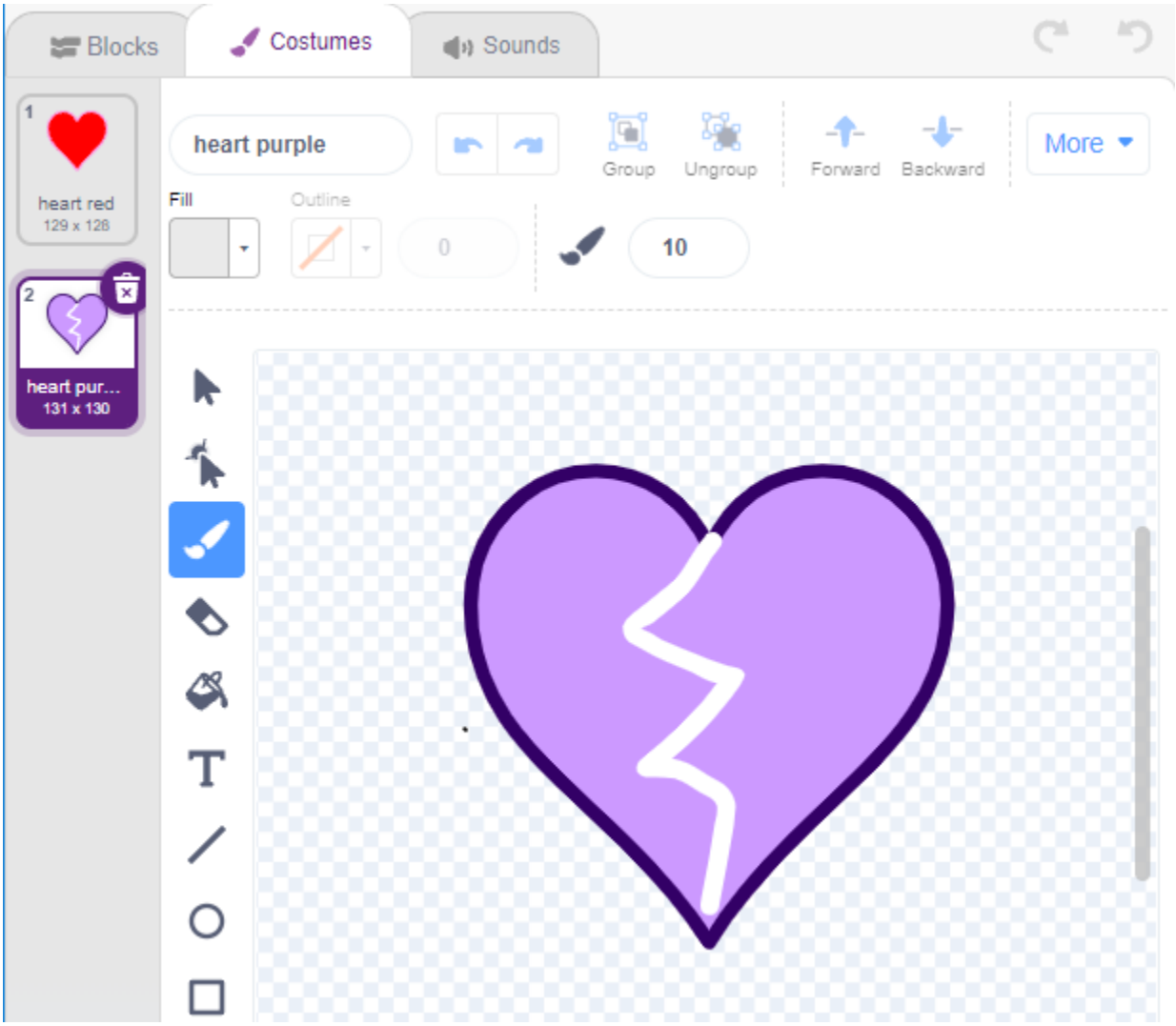


2. Herz-Sprite hinzufügen

Fügen Sie auch ein **Heart**-Sprite hinzu, setzen Sie seine Position auf (0, 0) und verkleinern Sie seine Größe, sodass es innerhalb der Quadratischen Box zu liegen scheint.

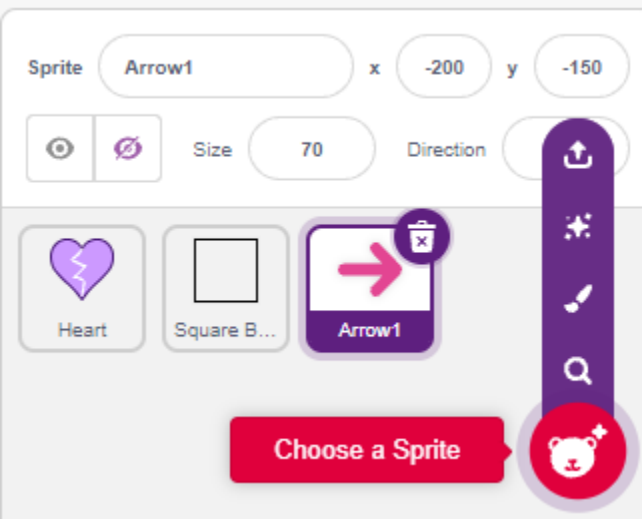


Auf der Seite **Costumes** passen Sie das lila Herz-Kostüm so an, dass es gebrochen erscheint.

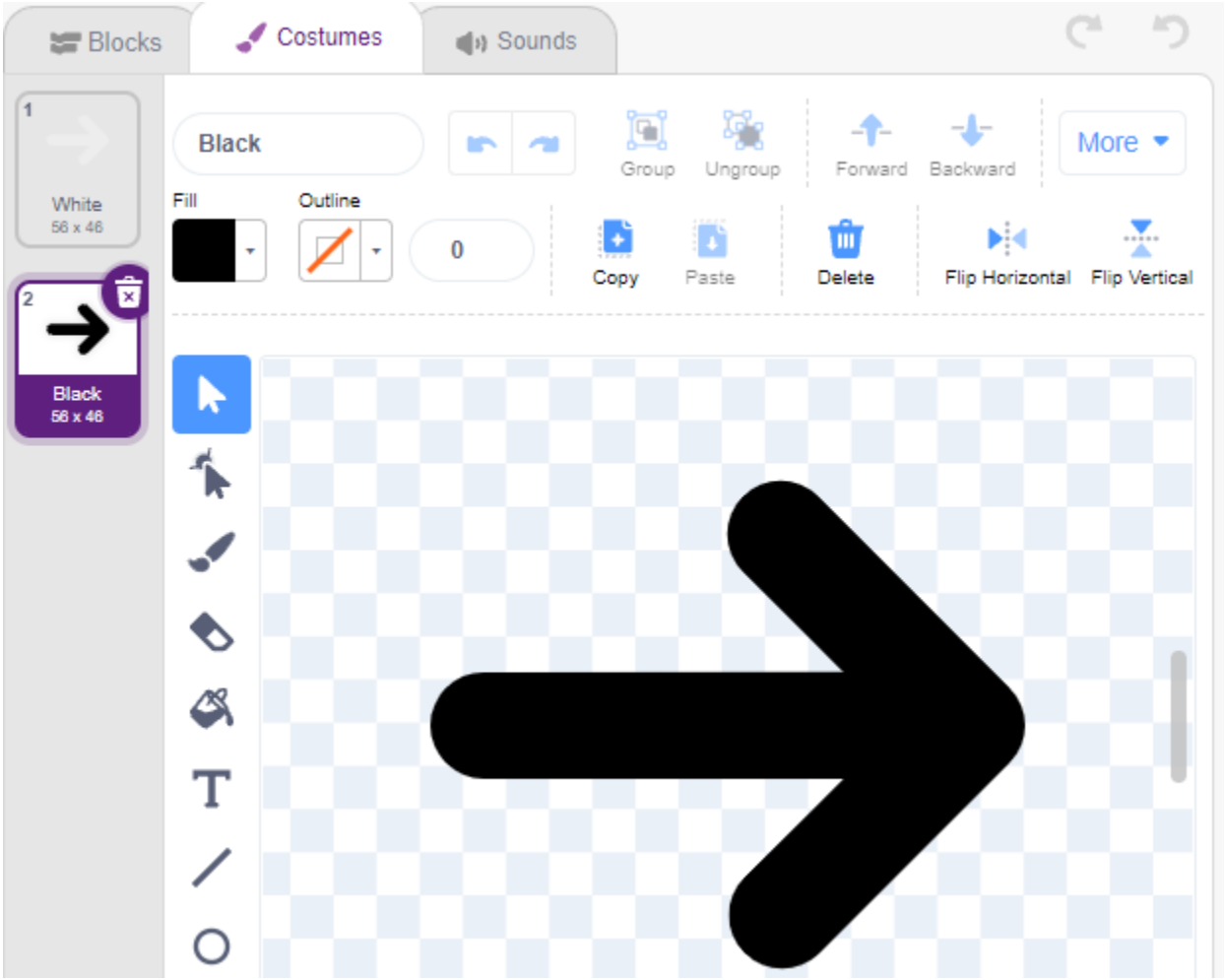


3. Pfeil1-Sprite hinzufügen

Fügen Sie ein **Arrow1**-Sprite hinzu.



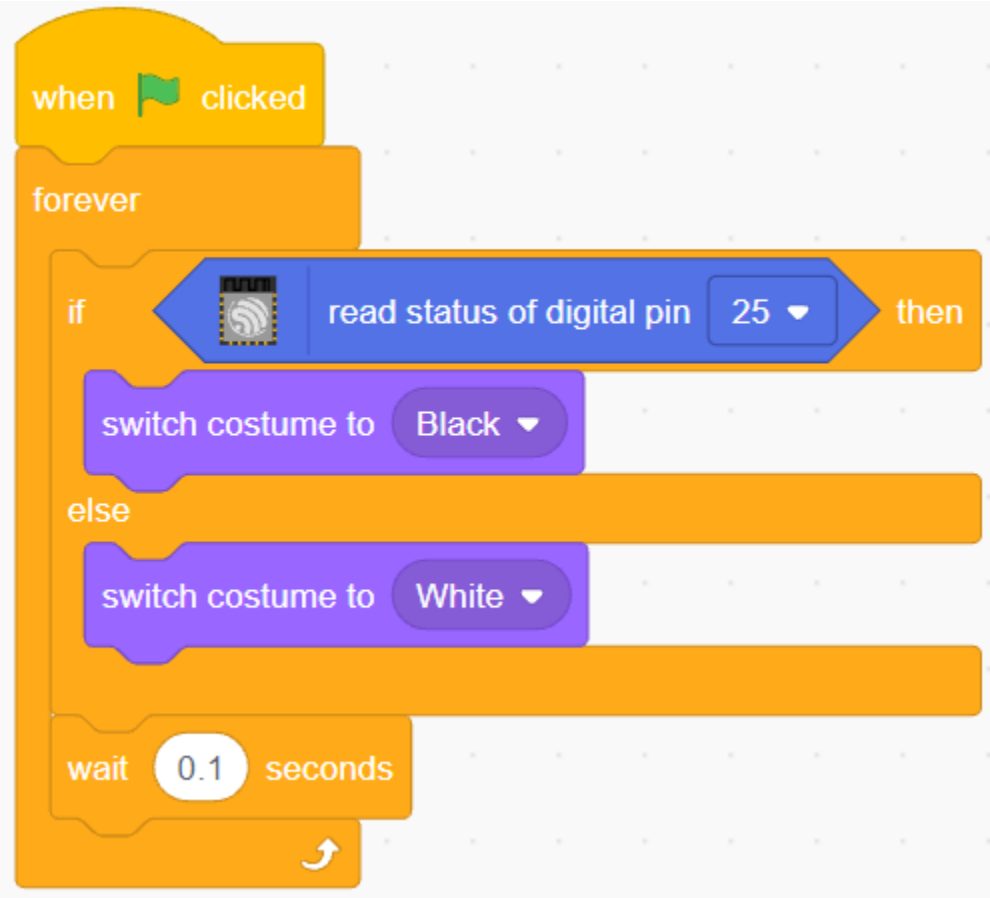
Auf der Seite **Costumes** behalten und kopieren Sie das nach rechts gerichtete Kostüm und stellen Sie seine Farbe auf Schwarz und Weiß ein.



4. Programmierung für das Quadratische Box-Sprite

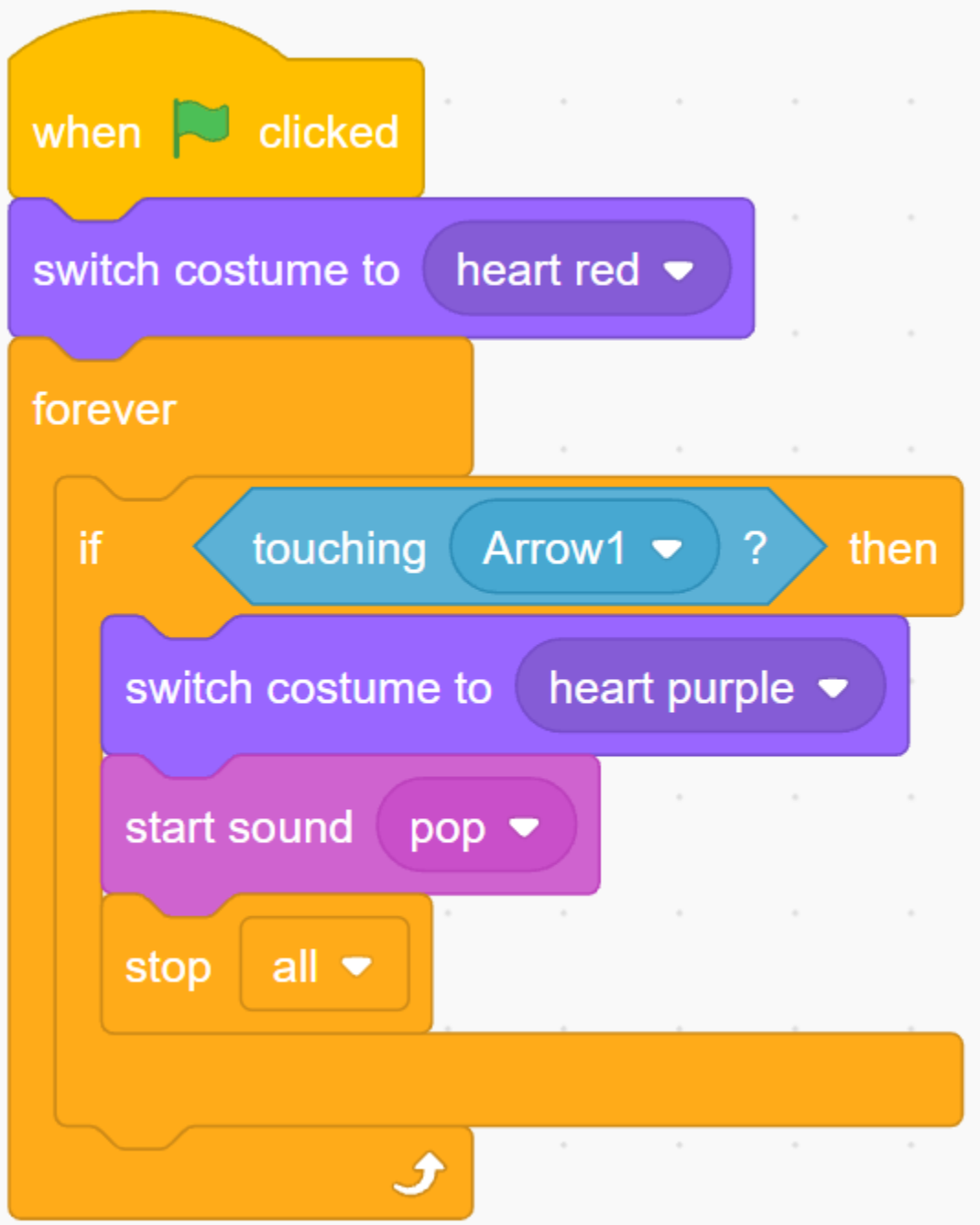
Gehen Sie zurück zur Seite **Blocks** und programmieren Sie das **Square Box**-Sprite.

- Wenn der Wert des digitalen Pins 2 (Linienfolgemodul) 1 ist (schwarze Linie erkannt), dann wechseln Sie das Kostüm zu **Black**.
- Andernfalls wechseln Sie das Kostüm zu **White**.



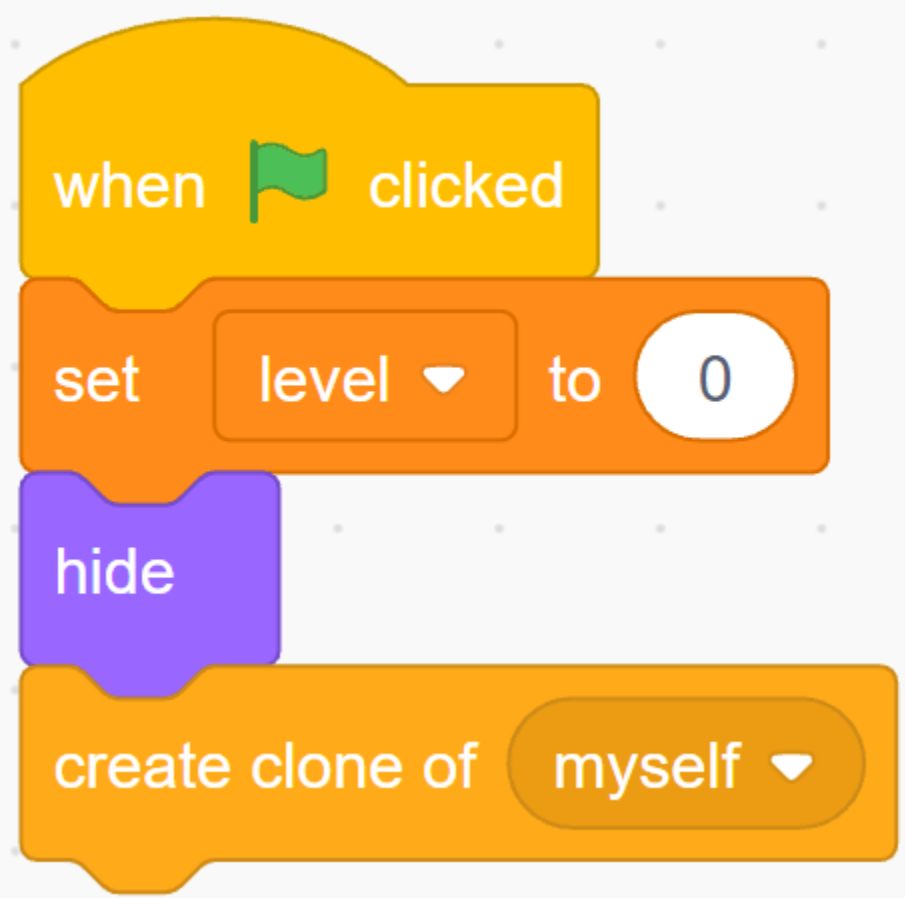
5. Programmierung für das Herz-Sprite

Das **Heart**-Sprite ist im **Square Box** geschützt und hat standardmäßig ein rotes Kostüm. Wenn das Pfeil1-Sprite es berührt, endet das Spiel.



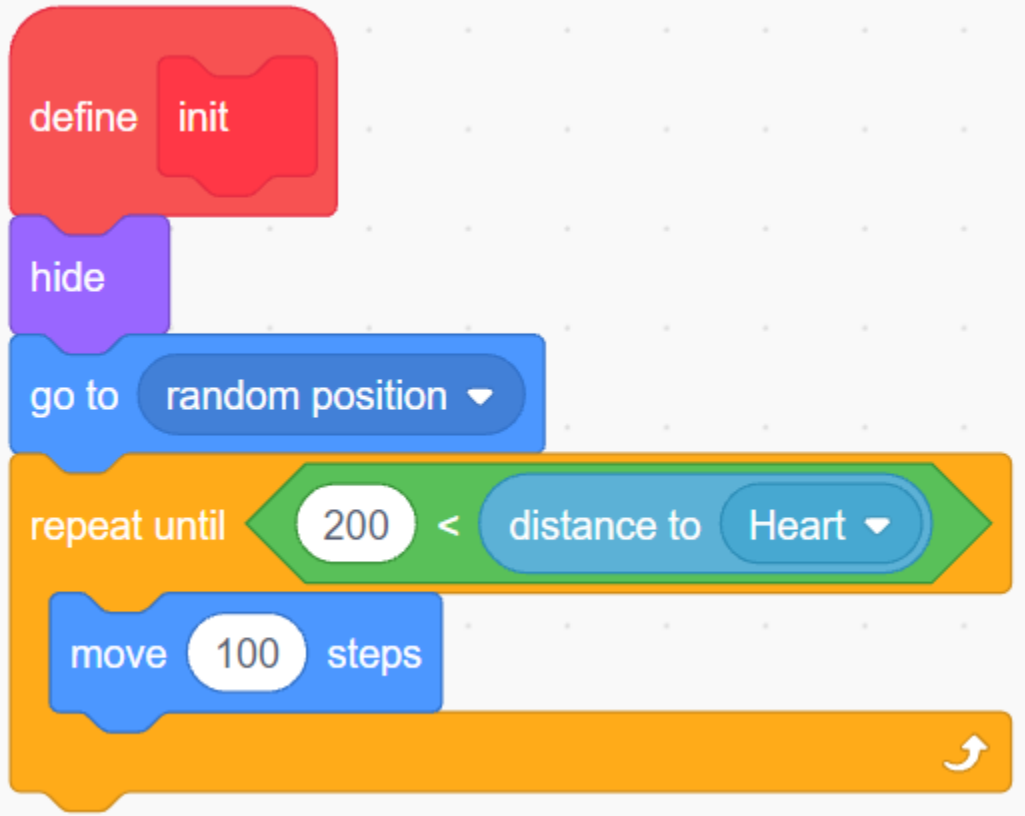
6. Programmierung für das Pfeil1-Sprite

Lassen Sie das **Arrow1**-Sprite verschwinden und erstellen Sie einen Klon, wenn auf die grüne Fahne geklickt wird.

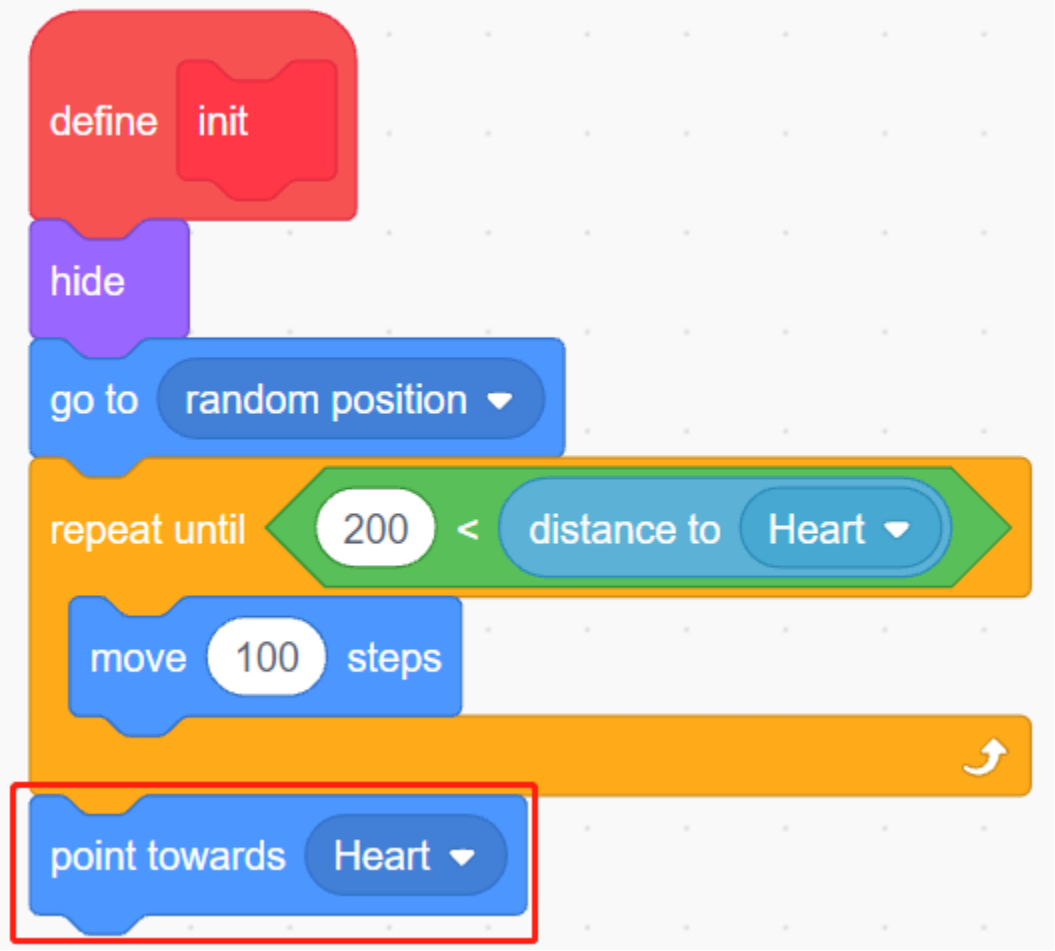


Erstellen Sie einen [init]-Block, um die Position, Ausrichtung und Farbe des **Arrow1**-Sprites zu initialisieren.

Es erscheint an einer zufälligen Position, und wenn der Abstand zwischen ihm und dem **Heart**-Sprite weniger als 200 beträgt, bewegt es sich nach außen, bis der Abstand größer als 200 ist.

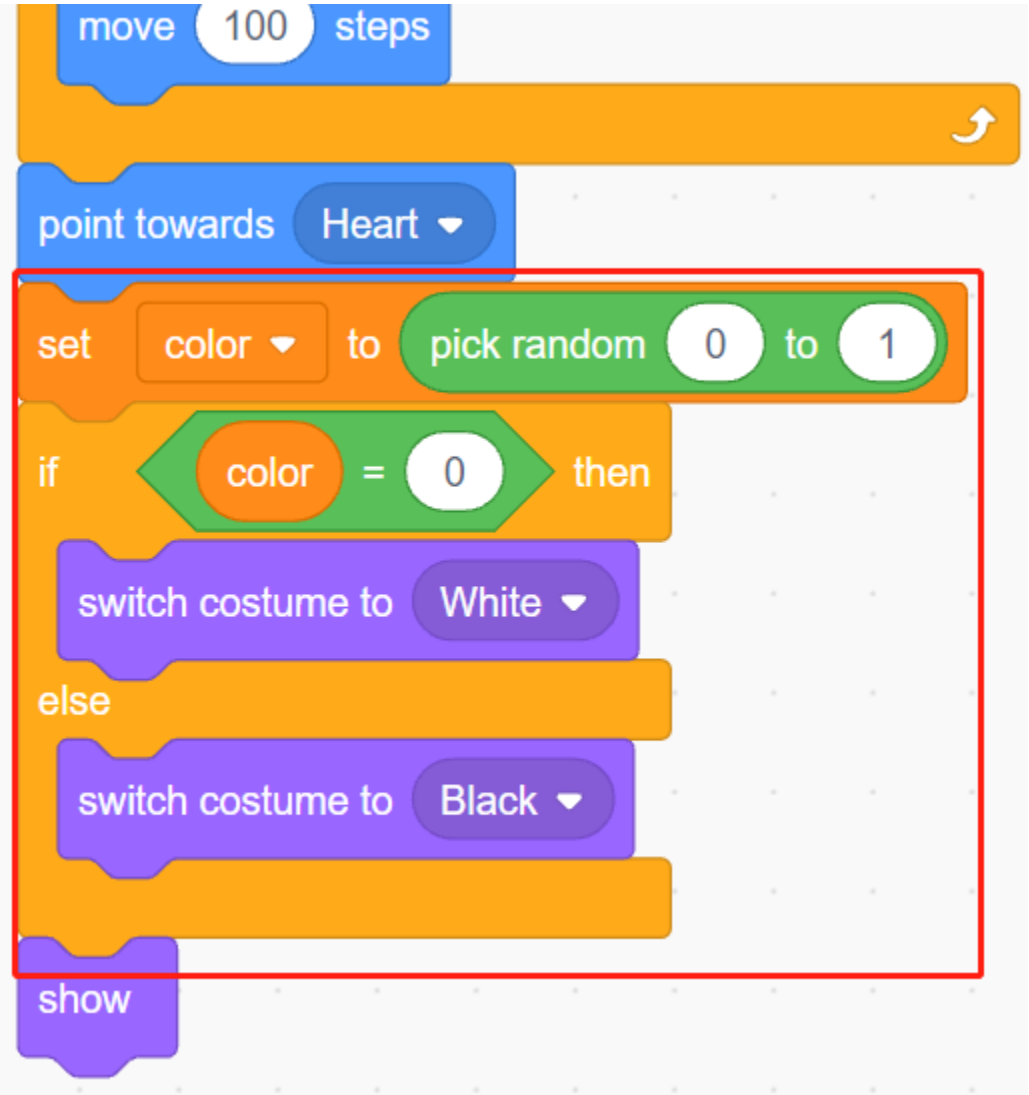


Richten Sie es in Richtung des **Heart**-Sprites aus.

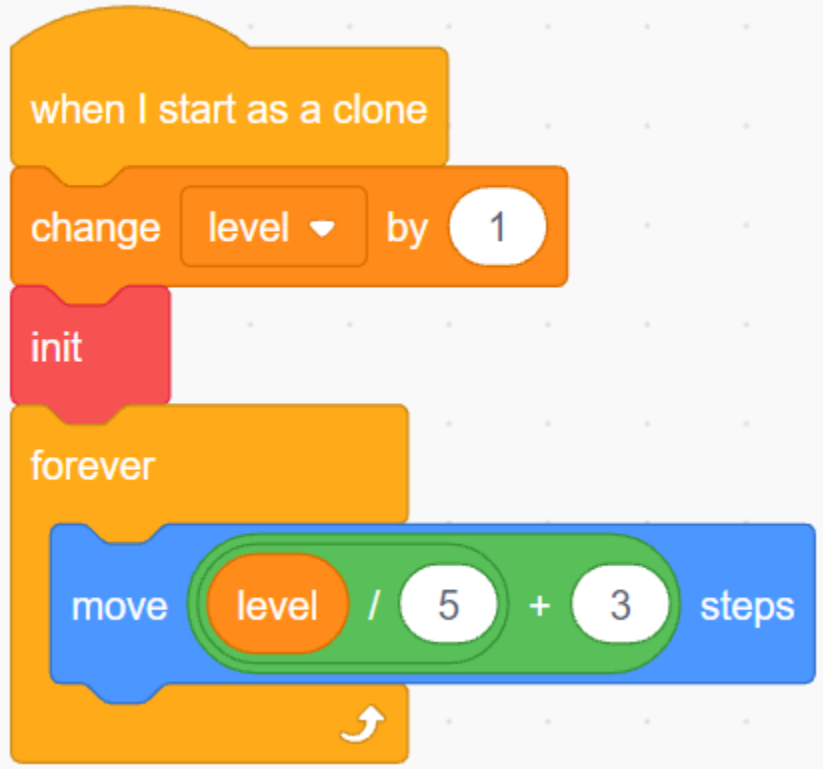


Lassen Sie seine Farbe zufällig zwischen Schwarz/Weiß wechseln.

- Ist die Variable Farbe 0, wechseln Sie das Kostüm zu **White**.
- Ist die Variable Farbe 1, wechseln Sie das Kostüm zu **Black**.

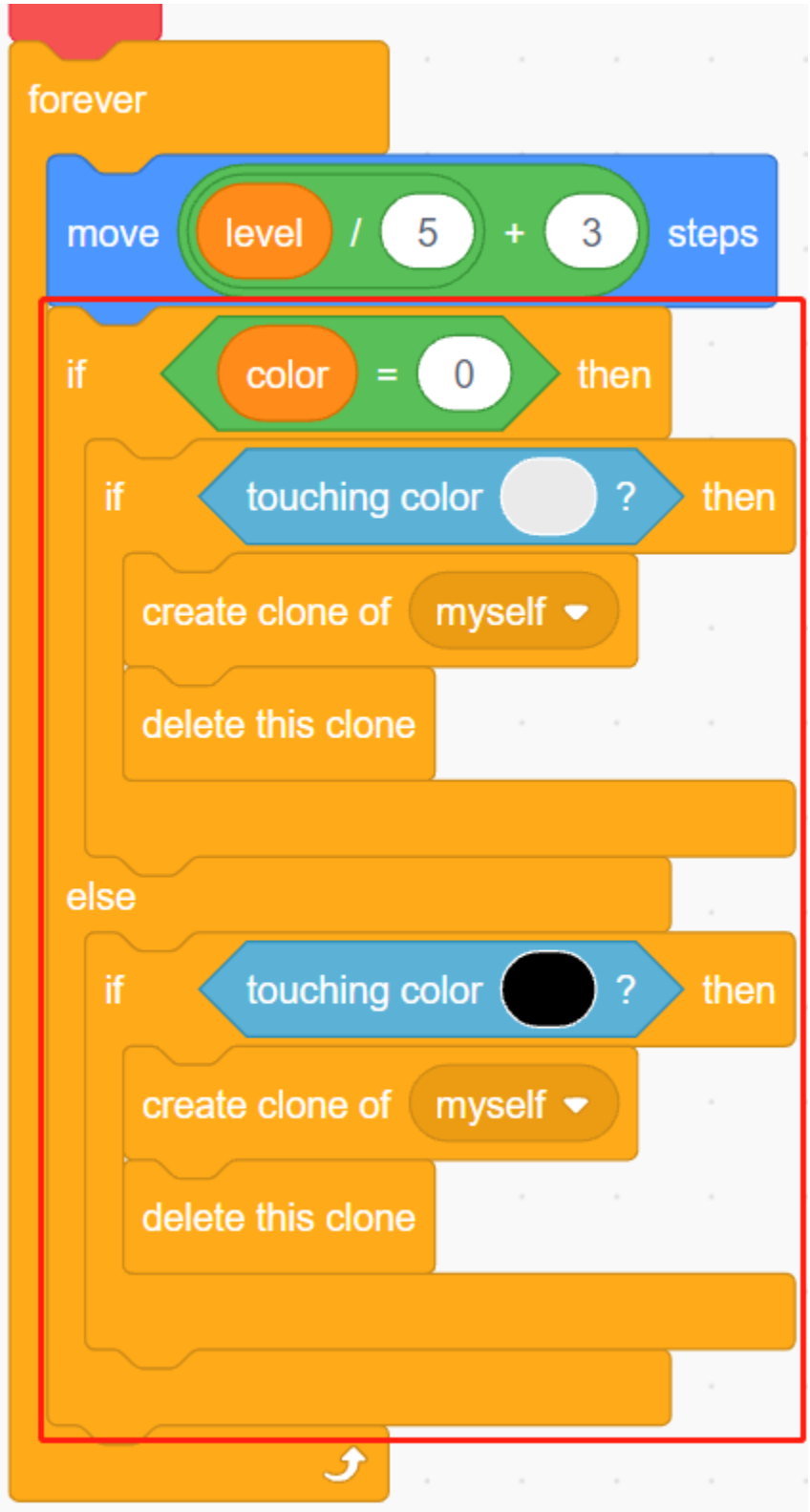


Lassen Sie es jetzt starten sich zu bewegen, es wird schneller, je höher der Wert der Variablen **level** steigt.

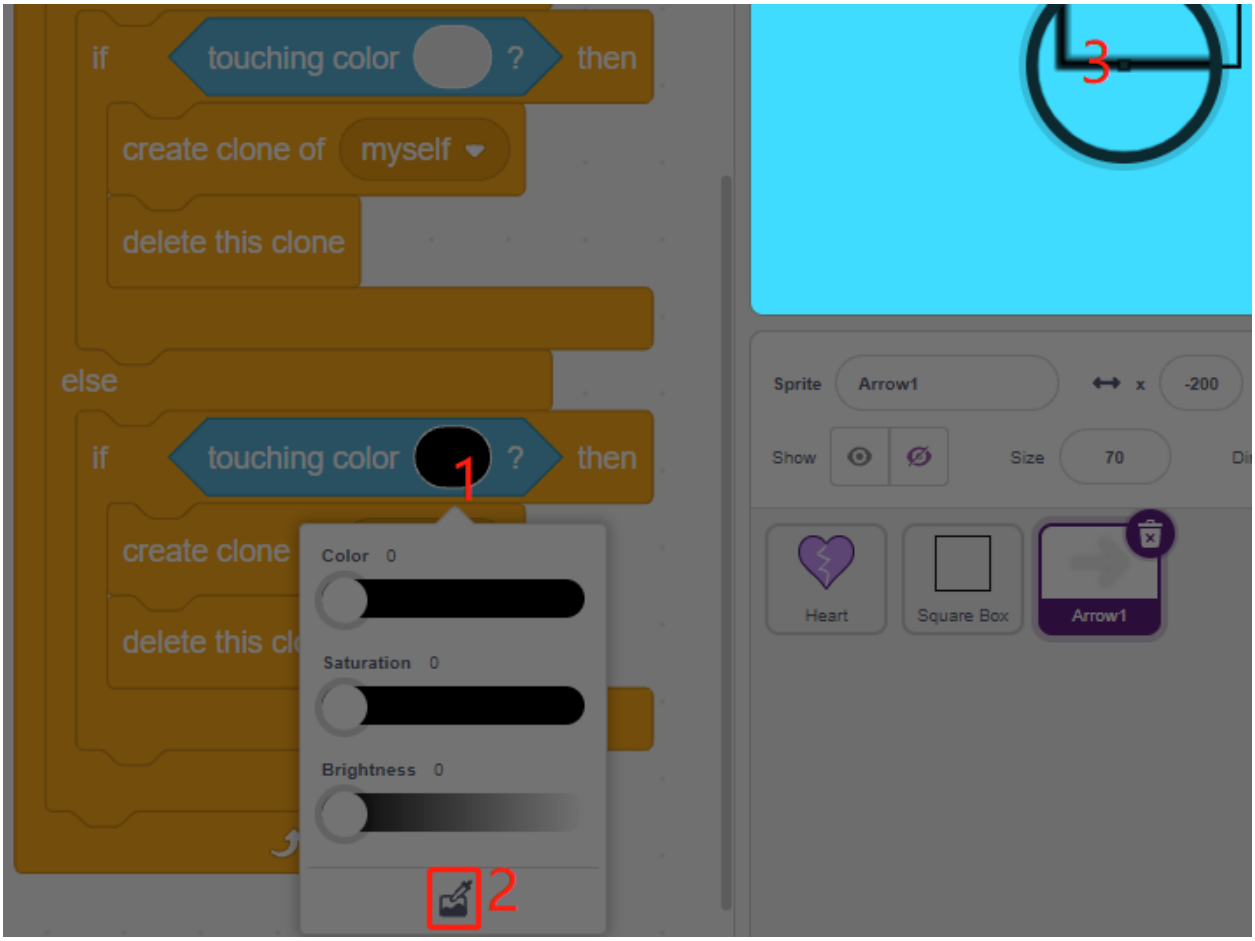


Stellen Sie nun seinen Kollisionseffekt mit dem **Square Box**-Sprite ein.

- Wenn das **Arrow1**-Sprite und das **Square Box**-Sprite dieselbe Farbe haben (die gemäß dem Wert des Linienverfolgungsmoduls geändert wird), entweder schwarz oder weiß, wird ein neuer Klon erstellt und das Spiel geht weiter.
- Stimmen ihre Farben nicht überein, bewegt sich das **Arrow1**-Sprite weiter und das Spiel endet, wenn es das **Heart**-Sprite berührt.



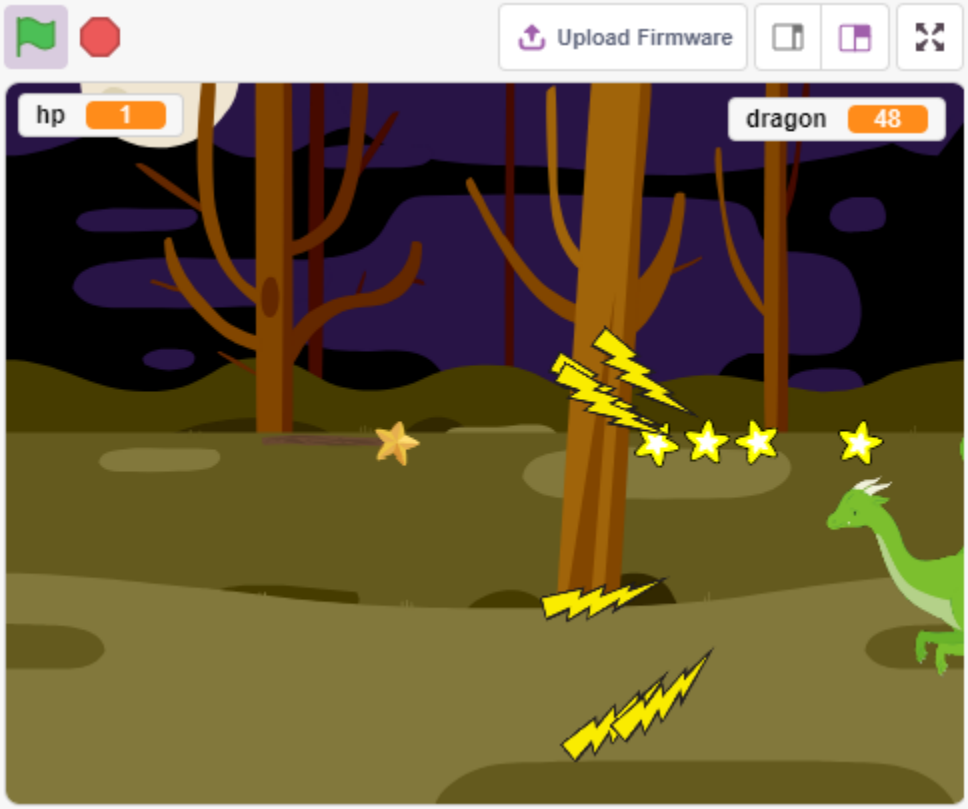
Bemerkung: Die beiden [touch color()]-Blöcke müssen die schwarz/weißen Kostüme der Quadratischen Box separat aufnehmen.



4.23 2.20 SPIEL - Drachen Töten

Hier verwenden wir den Joystick, um ein Drachentötungsspiel zu spielen.

Beim Klicken auf Grün schwebt der Drache auf der rechten Seite auf und ab und spuckt intermittierend Feuer. Sie müssen den Joystick verwenden, um die Bewegung des Zauberstabs zu steuern und Sternenangriffe auf den Drachen zu starten, während Sie den von ihm abgefeuerten Flammen ausweichen, um ihn schließlich zu besiegen.



4.23.1 Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
ESP32 Starter Kit	320+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

KOMPONENTENBESCHREIBUNG	KAUF-LINK
ESP32 WROOM 32E	
ESP32-Kameraerweiterung	-
Überbrückungsdrähte	
Joystick-Modul	

4.23.2 Schaltung Aufbauen

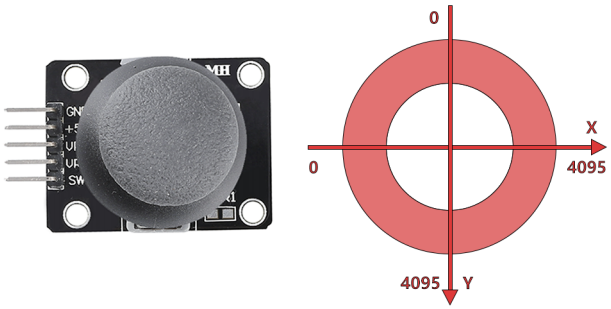
Ein Joystick ist ein Eingabegerät, bestehend aus einem Hebel, der auf einer Basis schwenkt und seinen Winkel oder seine Richtung an das Gerät meldet, das er steuert. Joysticks werden häufig verwendet, um Videospiele und Roboter zu steuern.

Um einen vollständigen Bewegungsbereich an den Computer zu übermitteln, muss ein Joystick die Position des Hebels auf zwei Achsen messen - die X-Achse (links nach rechts) und die Y-Achse (oben nach unten).

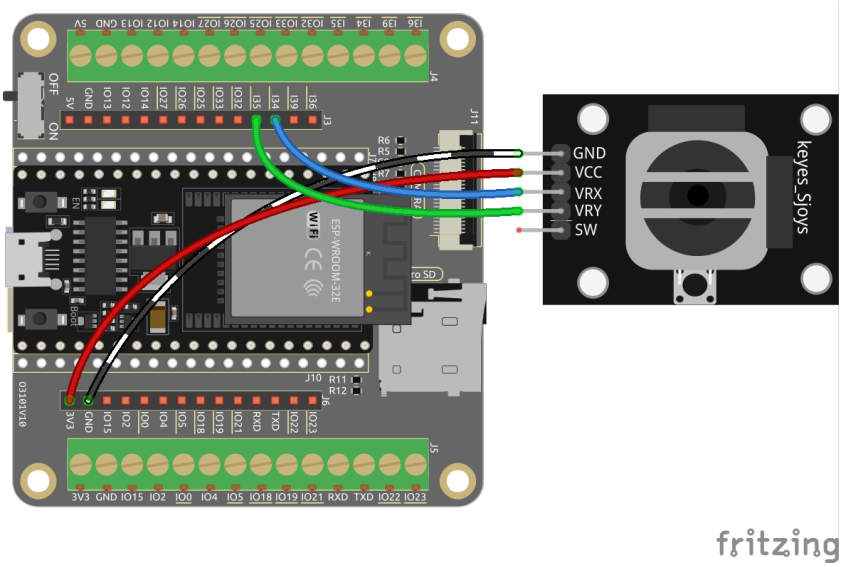
Die Bewegungskoordinaten des Joysticks werden in der folgenden Abbildung gezeigt.

Bemerkung:

- Die x-Koordinate verläuft von links nach rechts, der Bereich beträgt 0-1023.
- Die y-Koordinate verläuft von oben nach unten, der Bereich beträgt 0-1023.



Bauen Sie nun den Schaltkreis gemäß dem folgenden Diagramm auf.

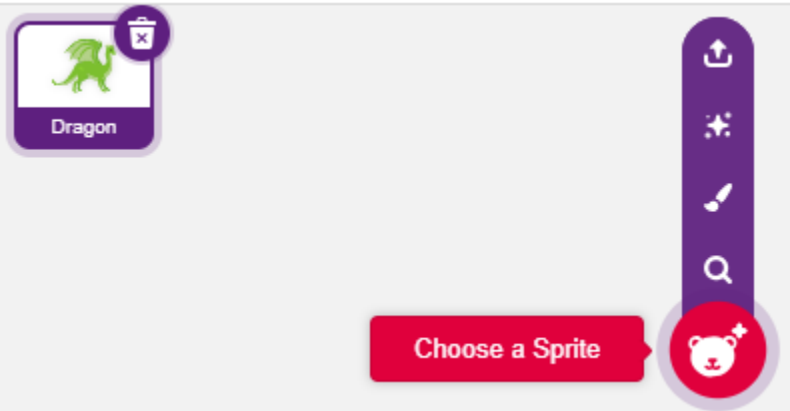


4.23.3 Programmierung

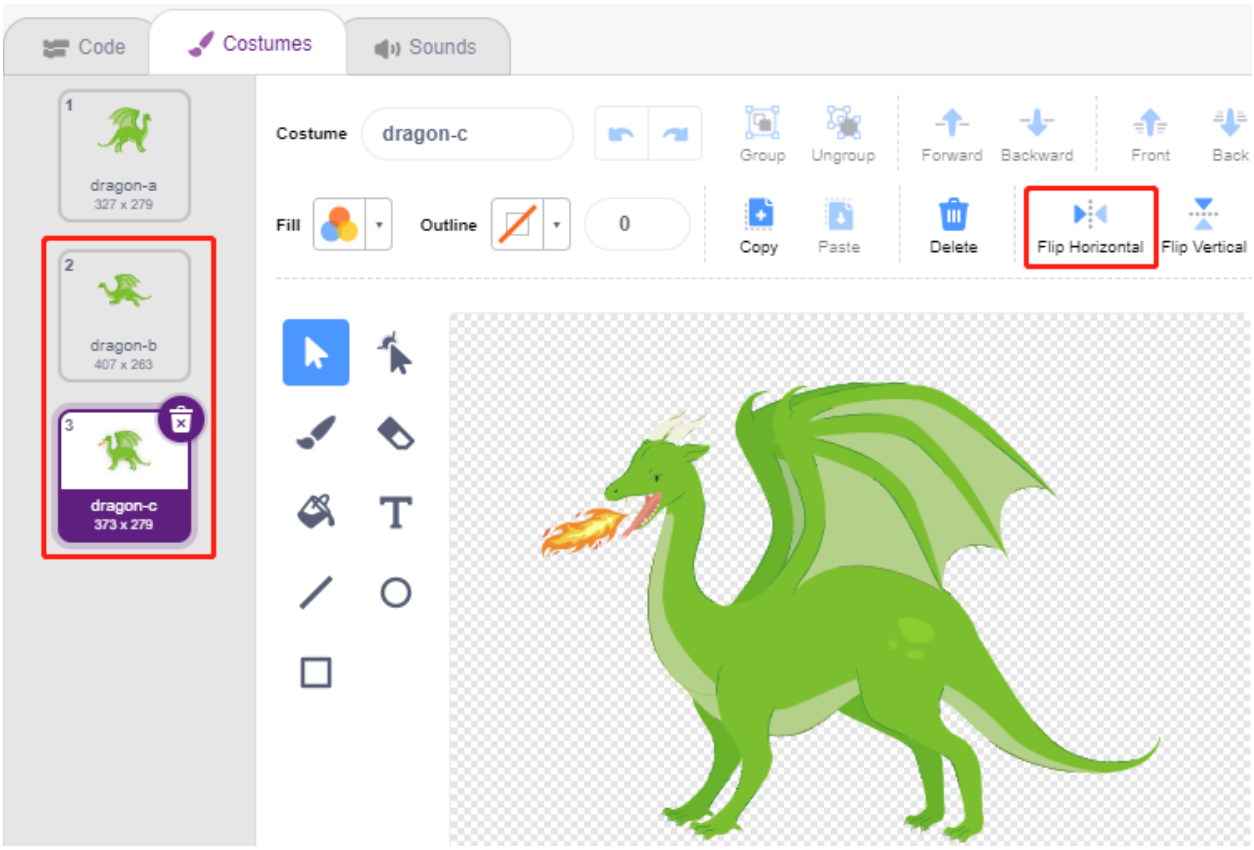
1. Drache

Woods-Hintergrund über den Button **Choose a Backdrop** hinzugefügt.

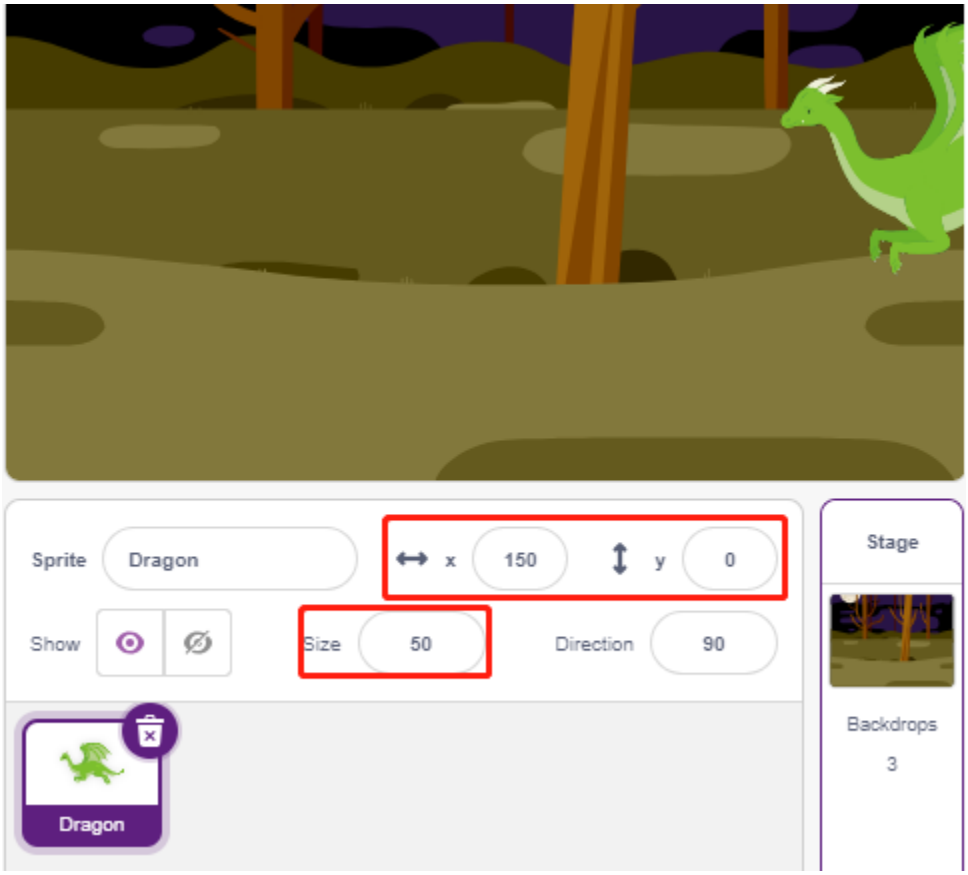
- Löschen Sie das Standard-Sprite und fügen Sie das **Dragon**-Sprite hinzu.



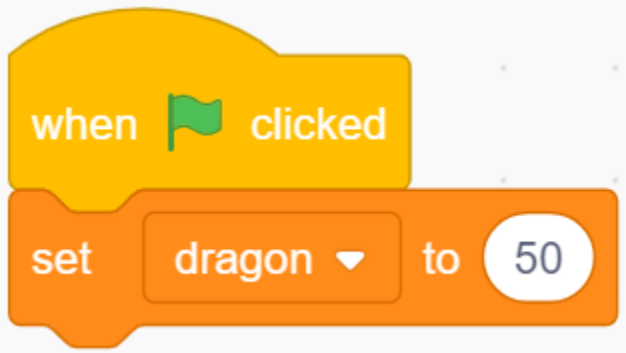
- Gehen Sie zur Seite **Costumes** und spiegeln Sie die Kostüme dragon-b und dragon-c horizontal.



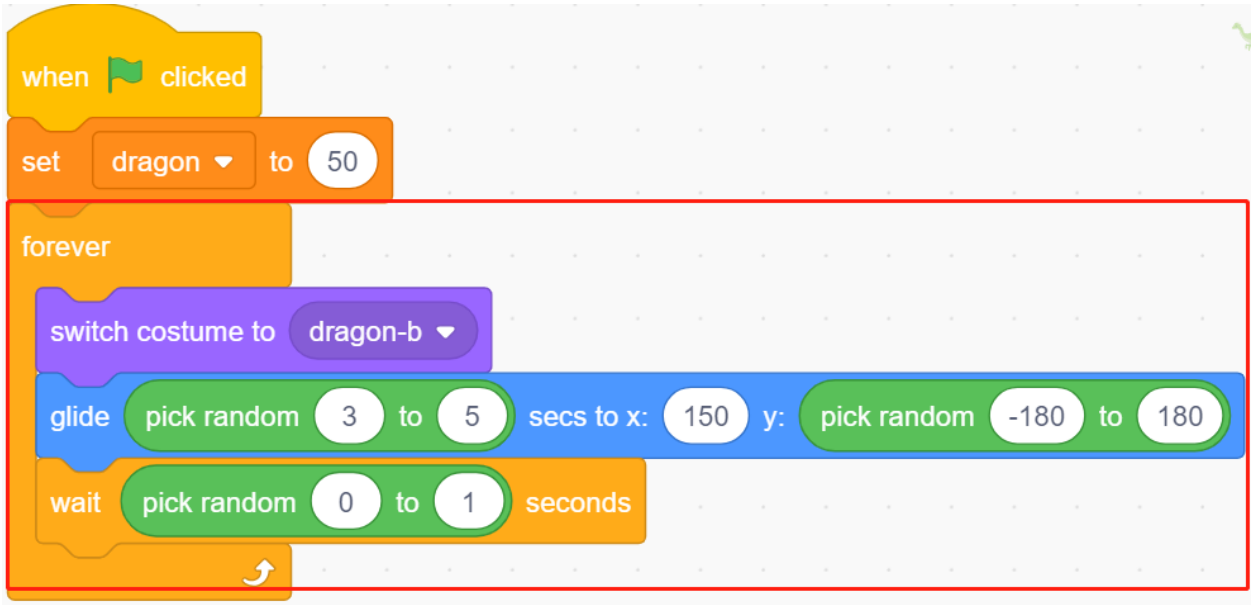
- Stellen Sie die Größe auf 50% ein.



- Erstellen Sie nun eine Variable - **dragon** - um die Lebenspunkte des Drachens aufzuzeichnen und setzen Sie den Anfangswert auf 50.

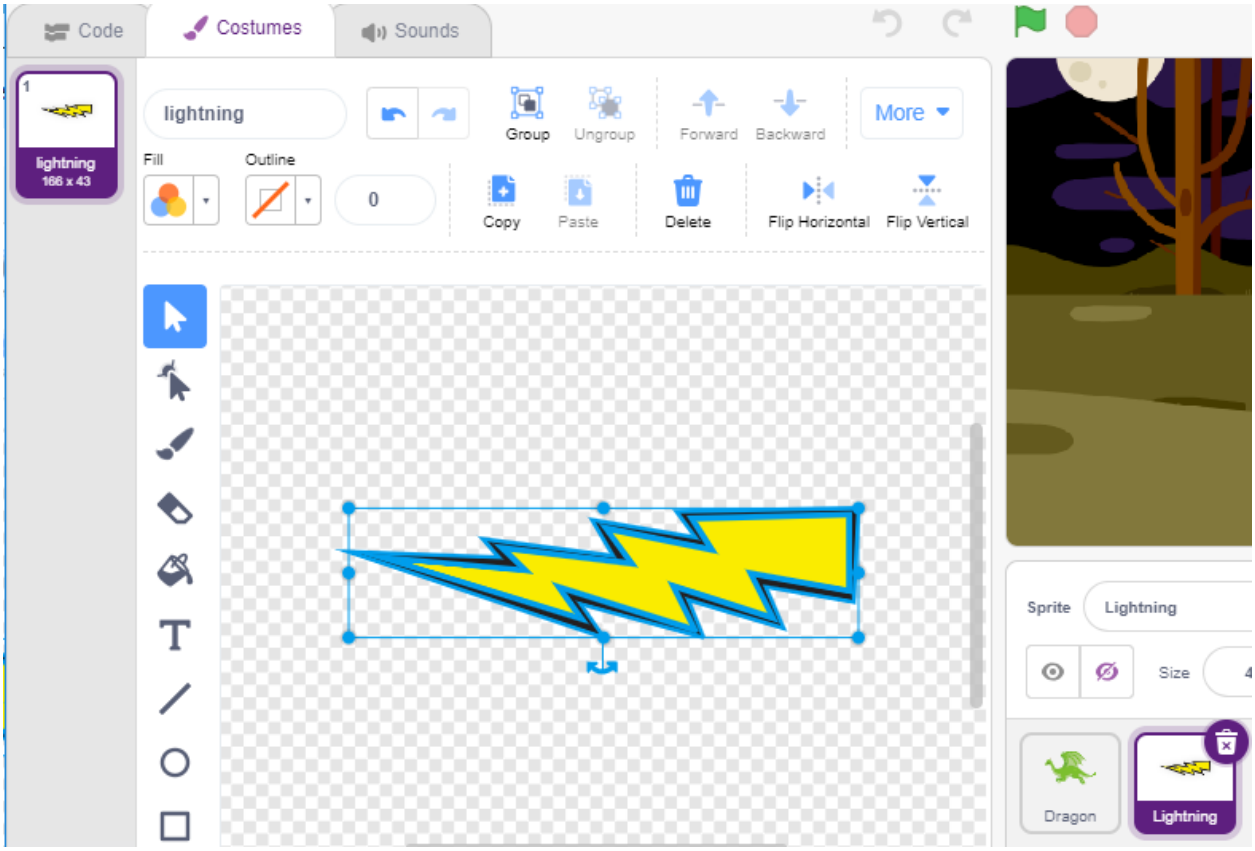


- Wechseln Sie anschließend das Kostüm des Sprites zu **dragon-b** und lassen Sie das **Dragon**-Sprite in einem Bereich auf und ab schweben.

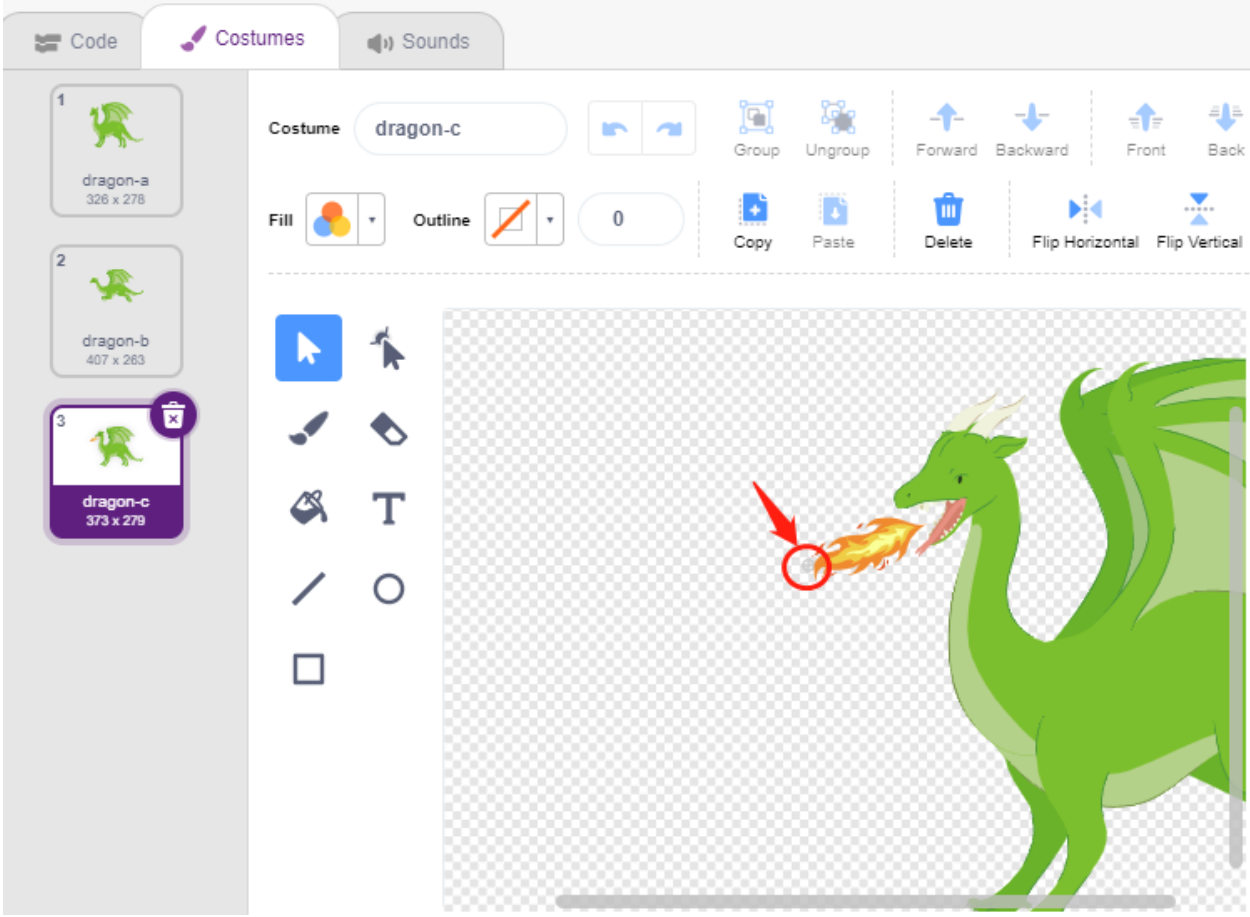


- Fügen Sie ein **Lightning**-Sprite als vom **Dragon**-Sprite geblasenes Feuer hinzu. Drehen Sie es auf der Seite Kostüme um 90° im Uhrzeigersinn, damit sich das **Lightning**-Sprite in die richtige Richtung bewegt.

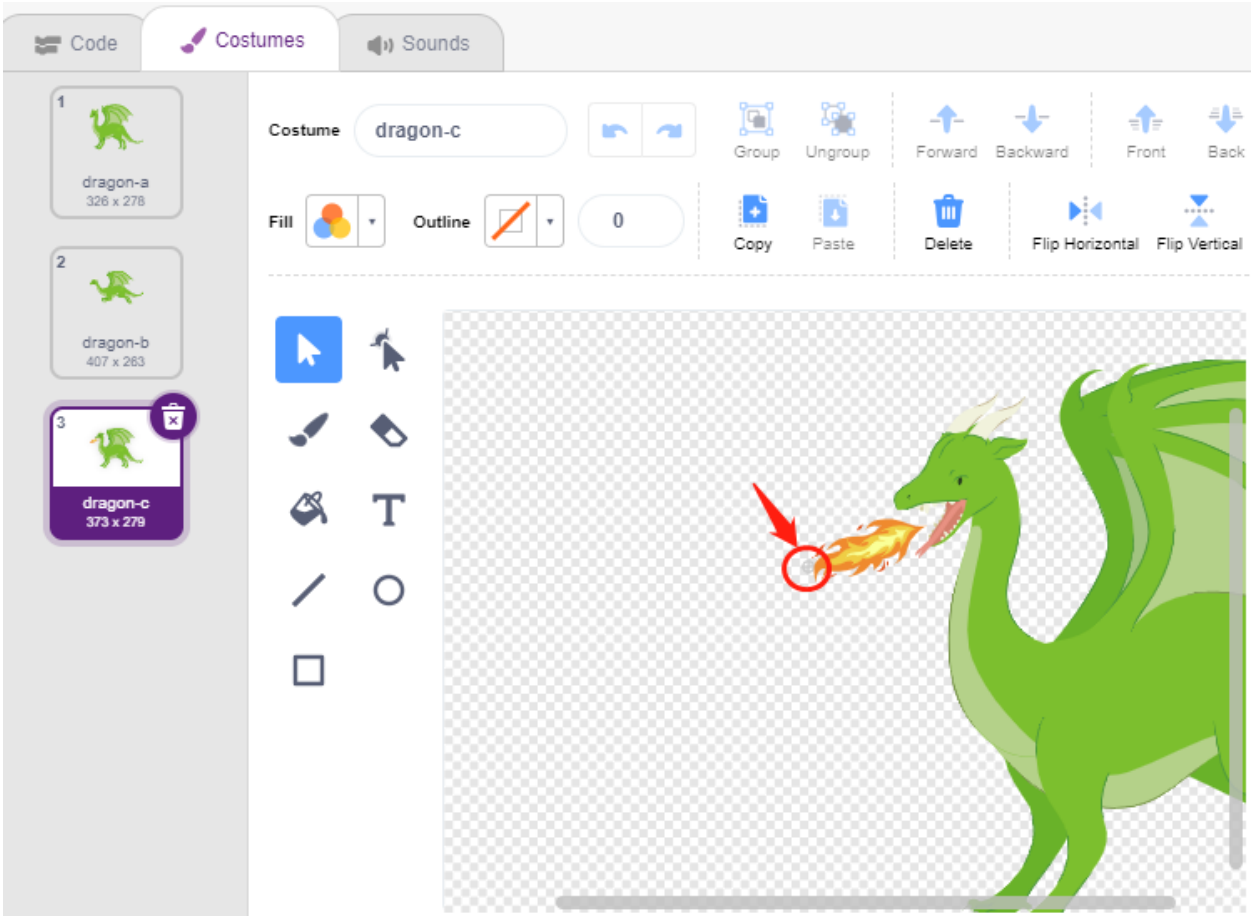
Bemerkung: Beim Anpassen des Kostüms des **Lightning**-Sprites kann es aus der Mitte verschoben werden, was unbedingt vermieden werden muss! Der Mittelpunkt muss genau in der Mitte des Sprites liegen!



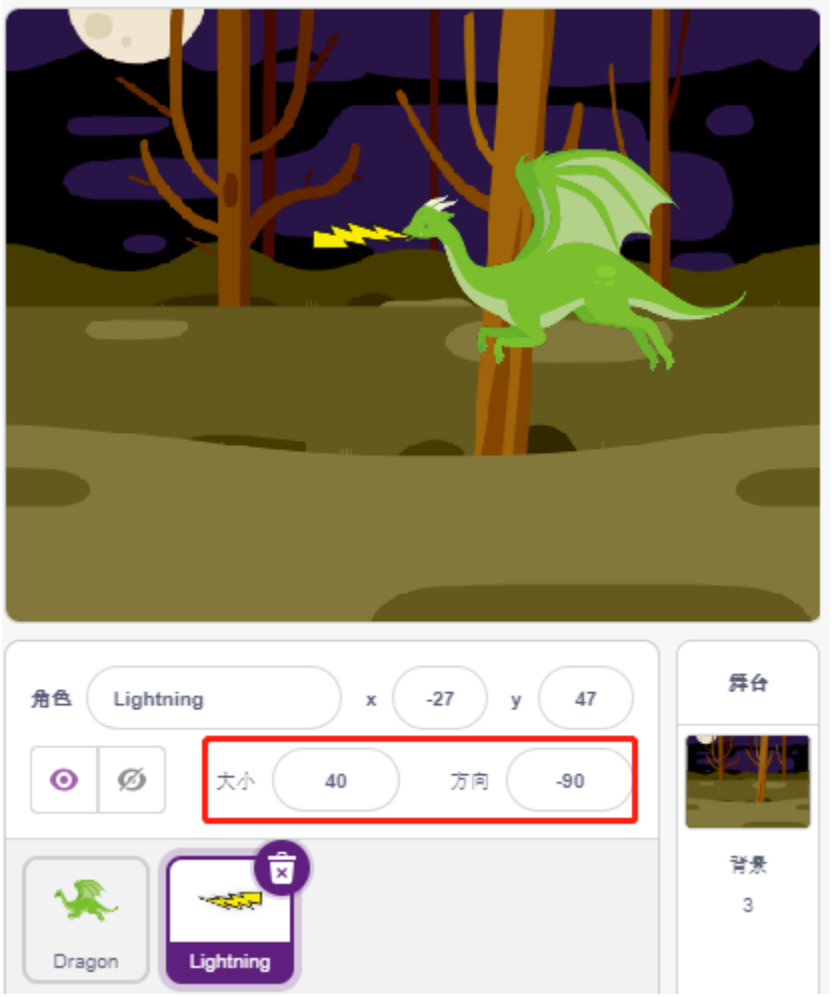
- Passen Sie dann das **dragon-c**-Kostüm des **Dragon**-Sprites so an, dass sein Mittelpunkt am Ende des Feuerschwanzes liegt. Dadurch werden die Positionen des **Dragon**-Sprites und des **Lightning**-Sprites korrekt und verhindern, dass der **Lightning** von den Füßen des Drachens abgefeuert wird.



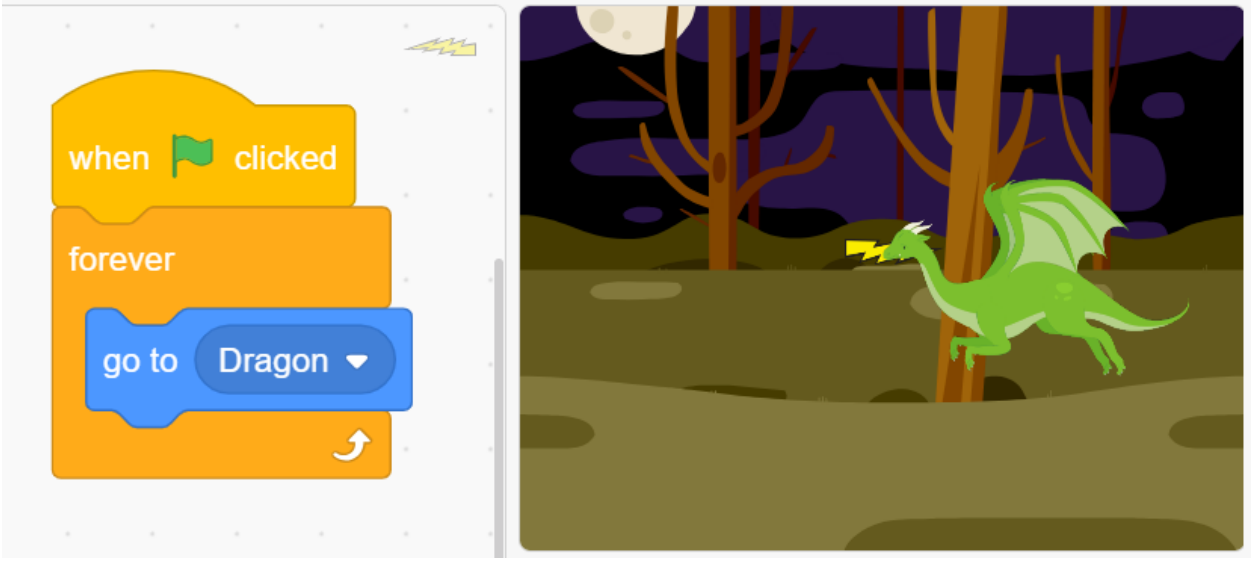
- Entsprechend muss bei **dragon-b** der Kopf des Drachens mit dem Mittelpunkt übereinstimmen.



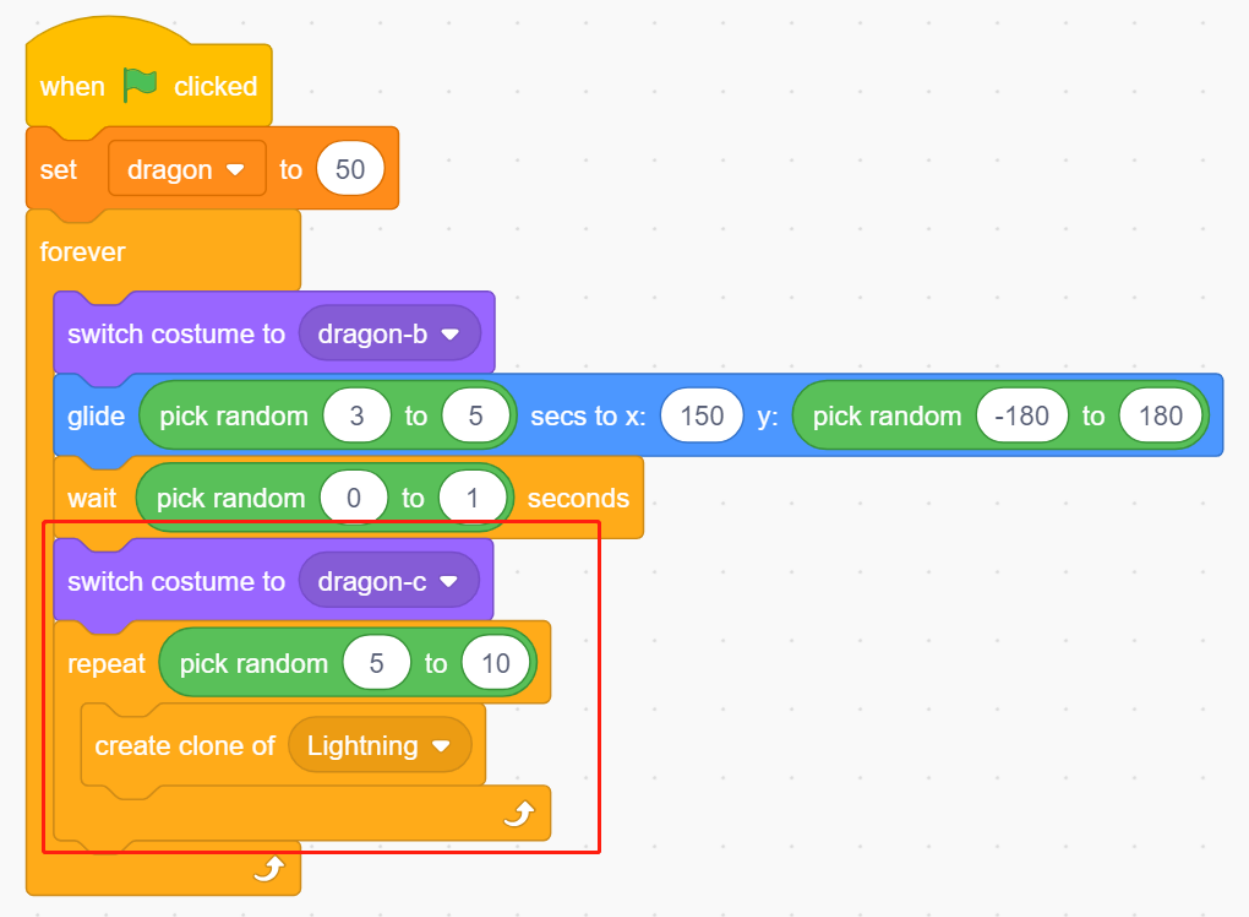
- Passen Sie die Größe und Ausrichtung des **Lightning**-Sprites an, um das Bild harmonischer wirken zu lassen.



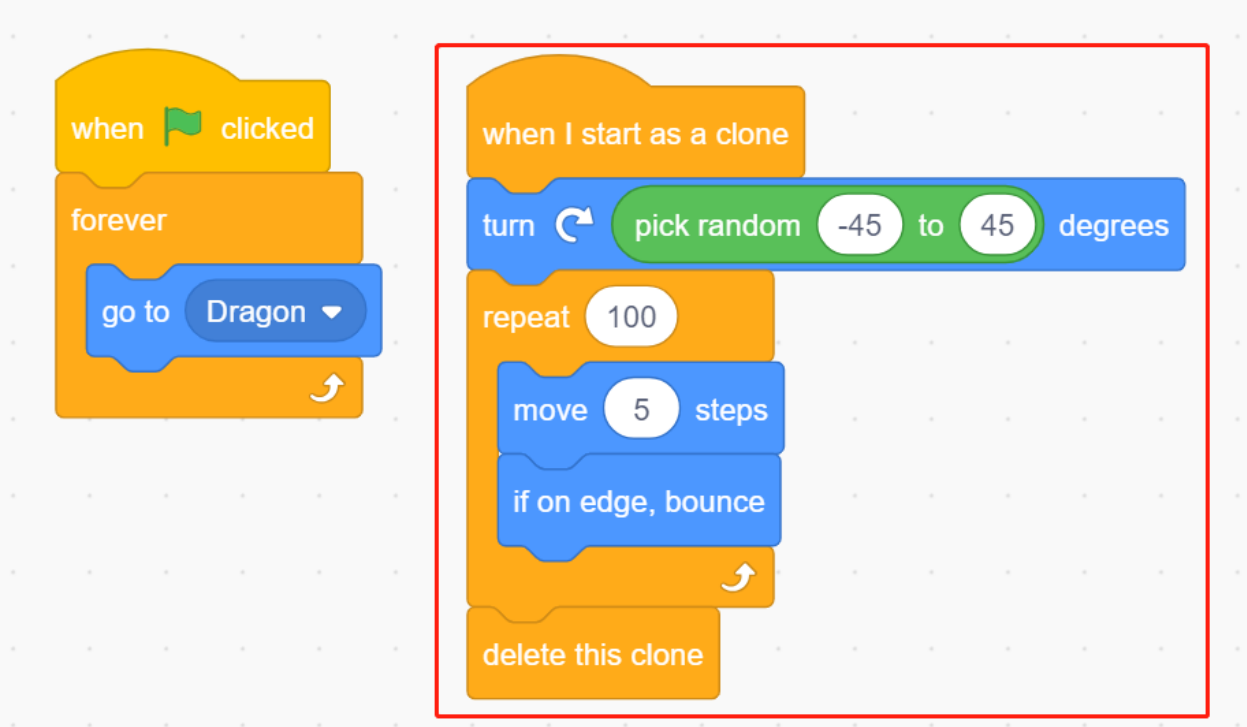
- Programmieren Sie nun das **Lightning**-Sprite. Das ist einfach, lassen Sie es immer dem **Dragon**-Sprite folgen. In diesem Moment klicken Sie auf die grüne Fahne und Sie werden sehen, wie der **Dragon** mit Blitz im Mund herumfliegt.



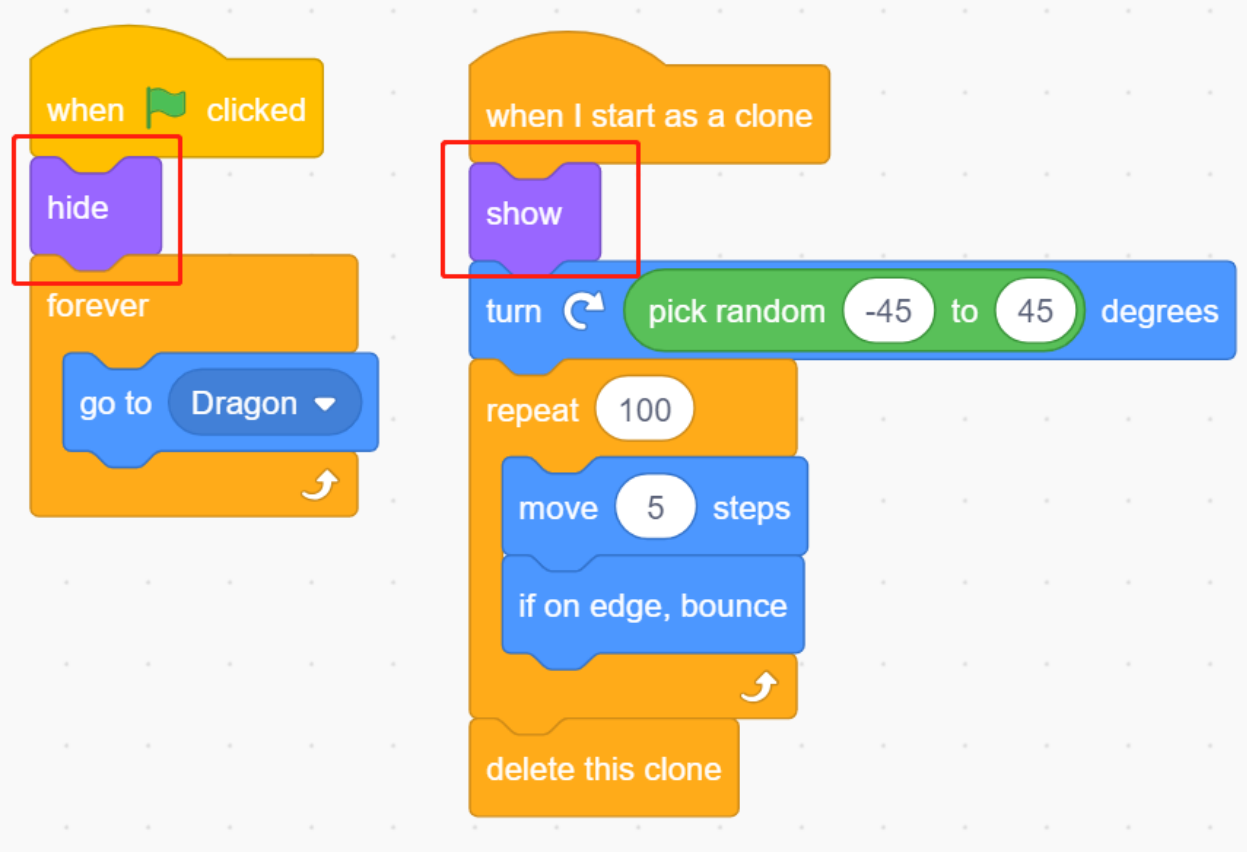
- Zurück zum **Dragon**-Sprite, lassen Sie es nun Feuer ausstoßen, wobei darauf zu achten ist, dass das Feuer im Mund nicht herausschießt, sondern ein Klon für das **Lightning**-Sprite erstellt wird.



- Klicken Sie auf das **Lightning**-Sprite und lassen Sie den **Lightning**-Klon in einem zufälligen Winkel abfeuern. Er prallt von der Wand ab und verschwindet nach einer bestimmten Zeit.



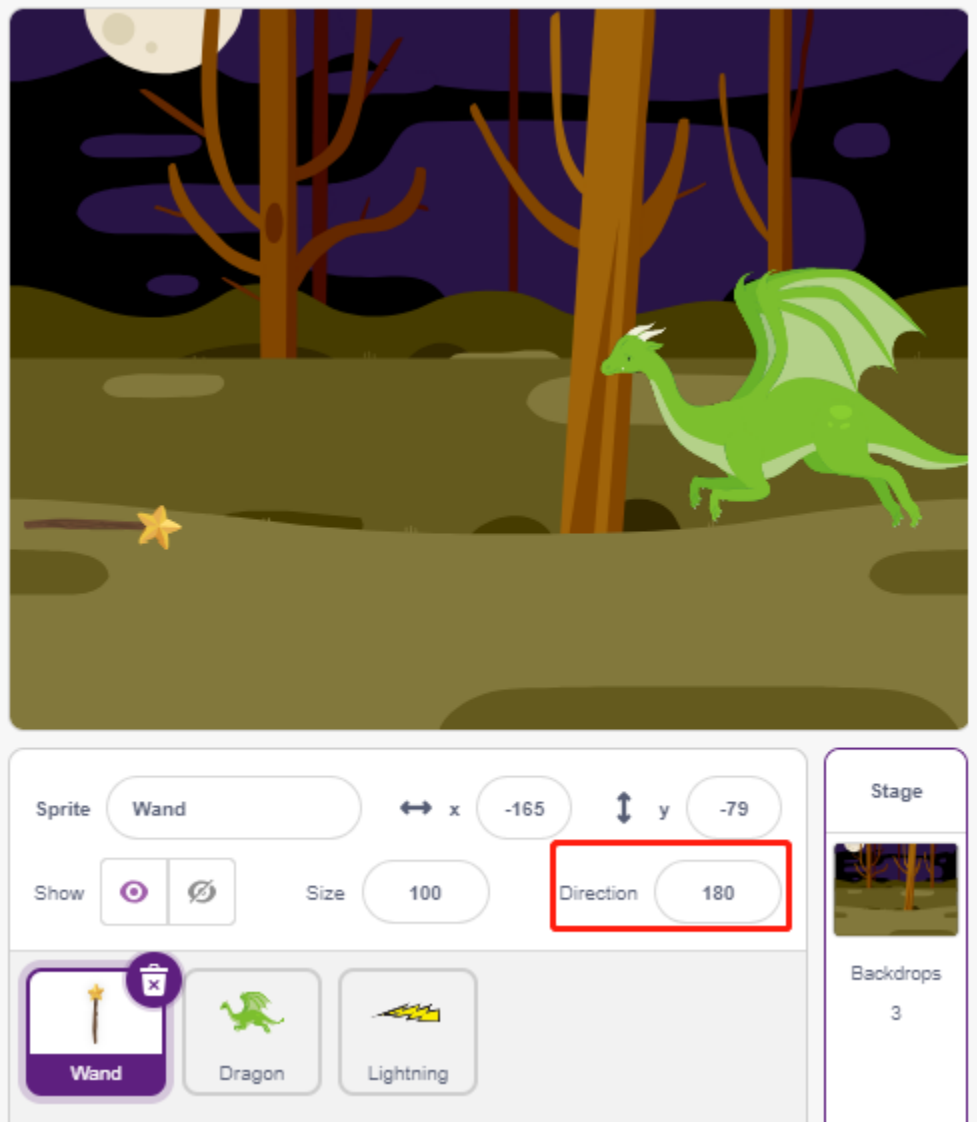
- Im **Lightning**-Sprite, verstecken Sie den Körper und zeigen Sie den Klon.



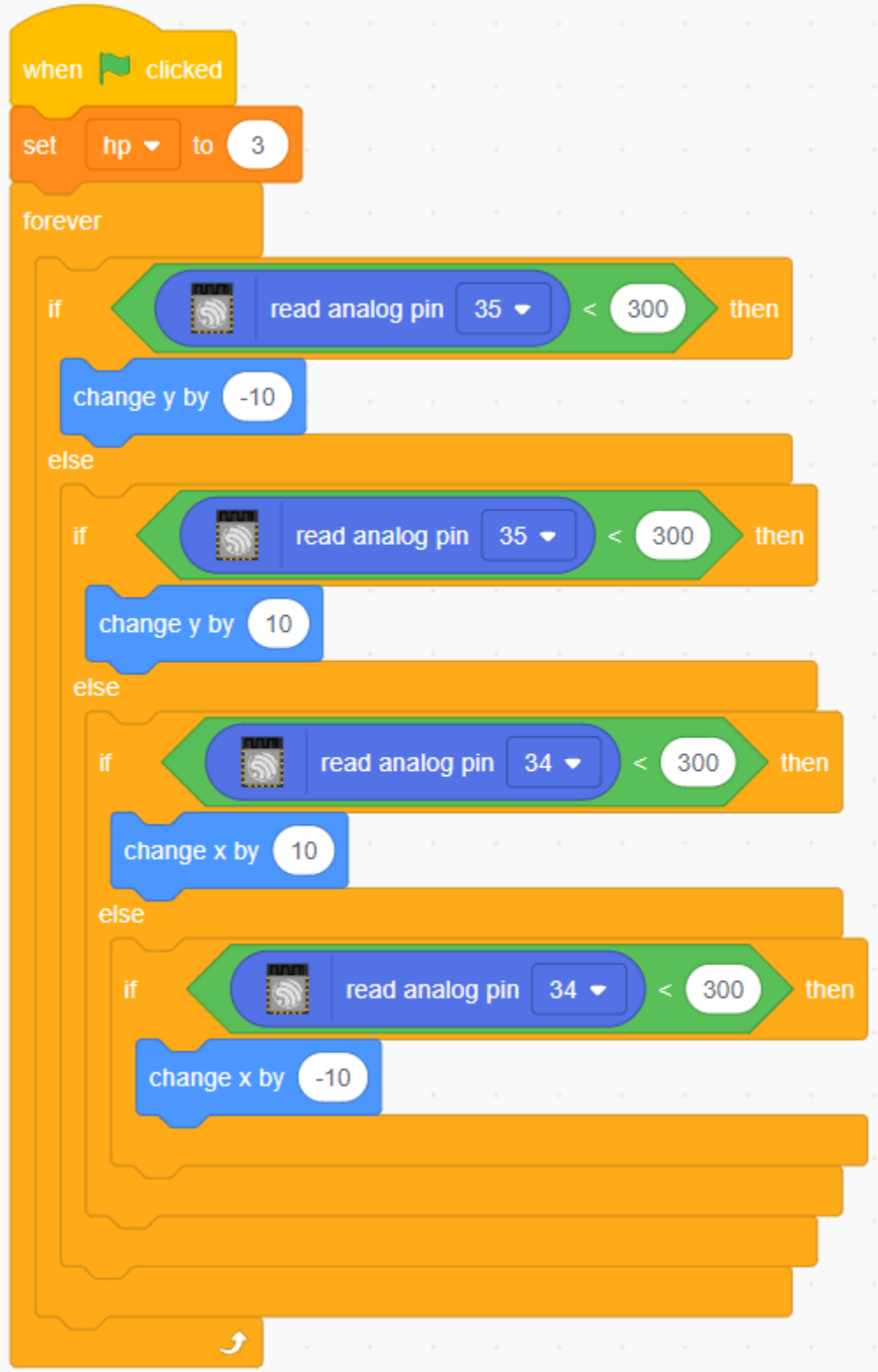
Jetzt kann der Drache auf und ab schweben und Feuer ausstoßen.

2. Zauberstab

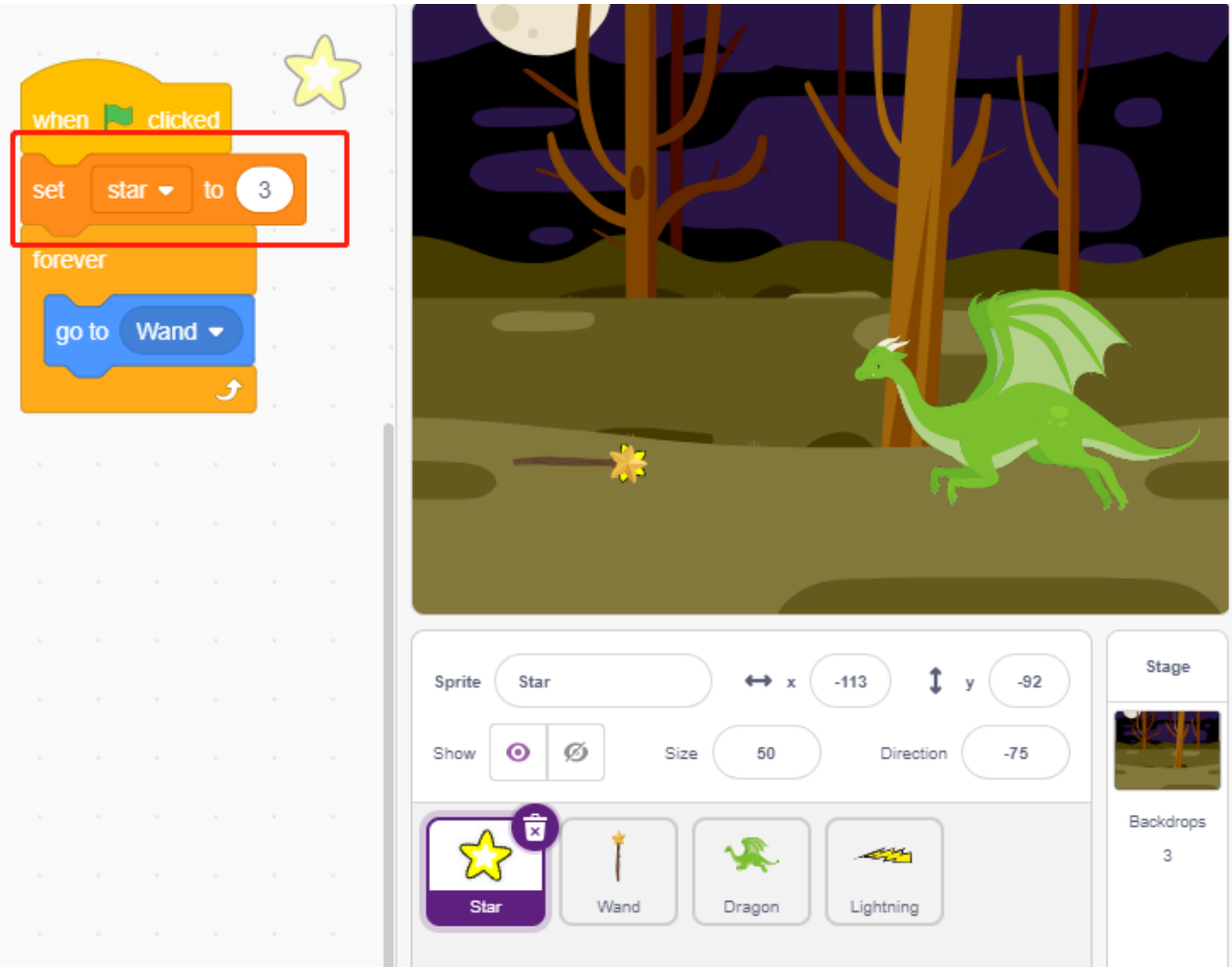
- Erstellen Sie ein **Wand**-Sprite und drehen Sie seine Richtung auf 180 Grad, um nach rechts zu zeigen.



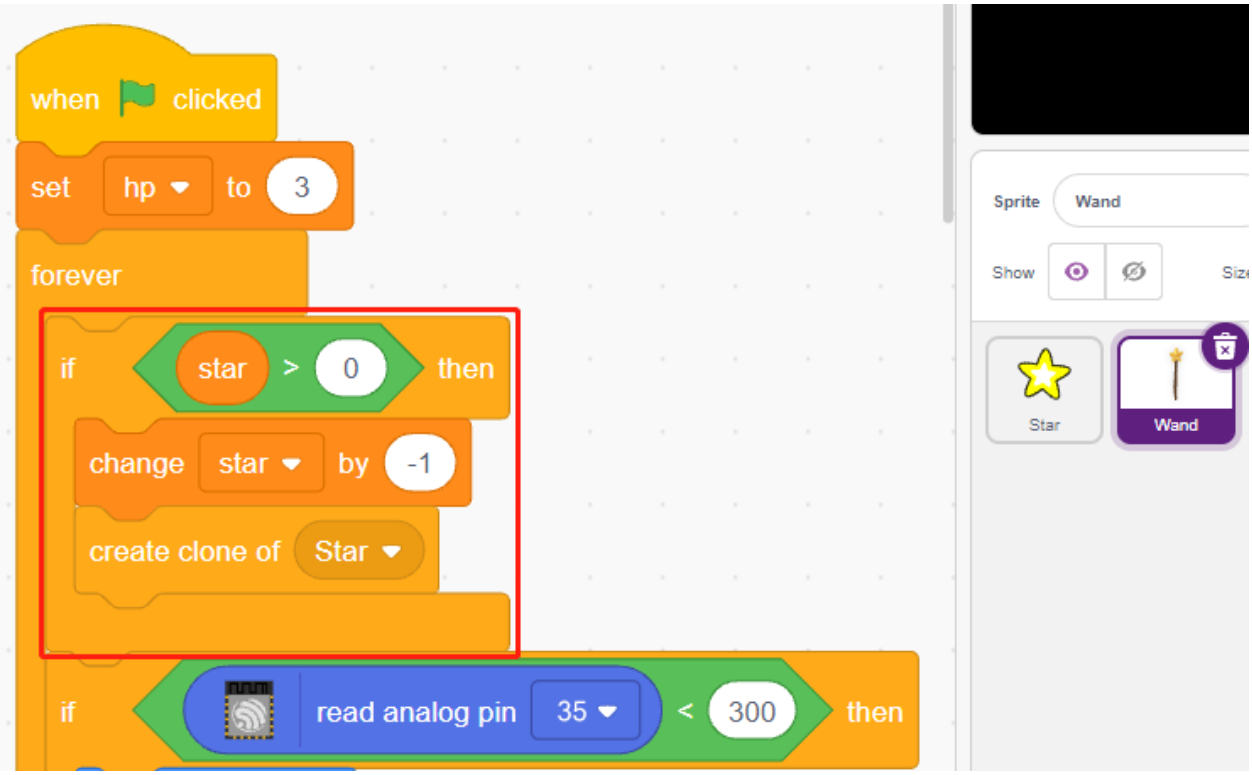
- Erstellen Sie nun eine Variable **hp**, um dessen Lebenswert aufzuzeichnen, anfänglich auf 3 gesetzt. Lesen Sie dann den Wert des Joysticks, der verwendet wird, um die Bewegung des Zauberstabs zu steuern.



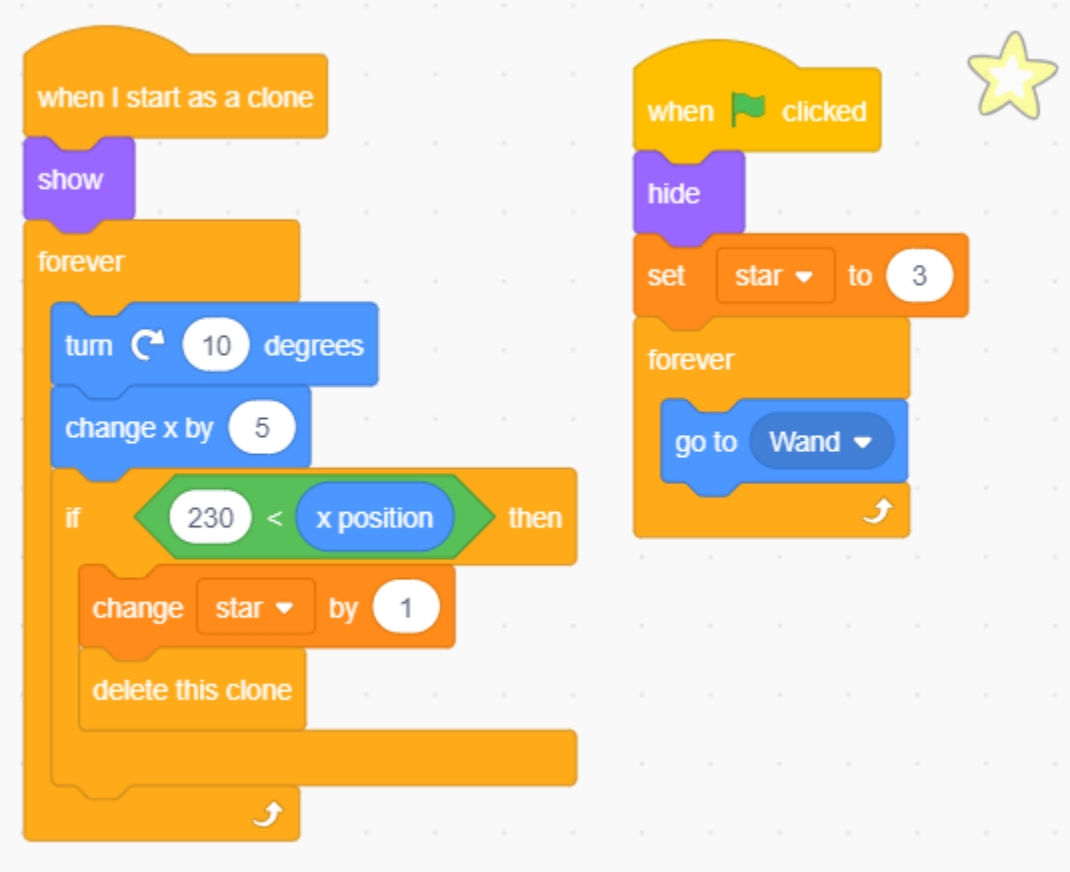
- Der Drache hat Blitze und der Zauberstab, der ihn zerschmettert, hat seine „magische Kugel“! Erstellen Sie ein **Star**-Sprite, passen Sie dessen Größe an und programmieren Sie es so, dass es immer dem **Wand**-Sprite folgt, und begrenzen Sie die Anzahl der Sterne auf drei.



- Lassen Sie das **Wand**-Sprite automatisch Sterne schießen. Das **Wand**-Sprite schießt Sterne auf die gleiche Weise, wie der Drache Feuer spuckt - durch das Erstellen von Klonen.



- Gehen Sie zurück zum **Star**-Sprite und programmieren Sie dessen Klon so, dass er sich dreht und nach rechts schießt, verschwindet, nachdem er die Bühne verlassen hat, und stellt die Anzahl der Sterne wieder her. Wie beim **Lightning**-Sprite, verstecken Sie den Körper und zeigen Sie den Klon.



Jetzt haben wir einen Zauberstab, der Sternenkugeln schießt.

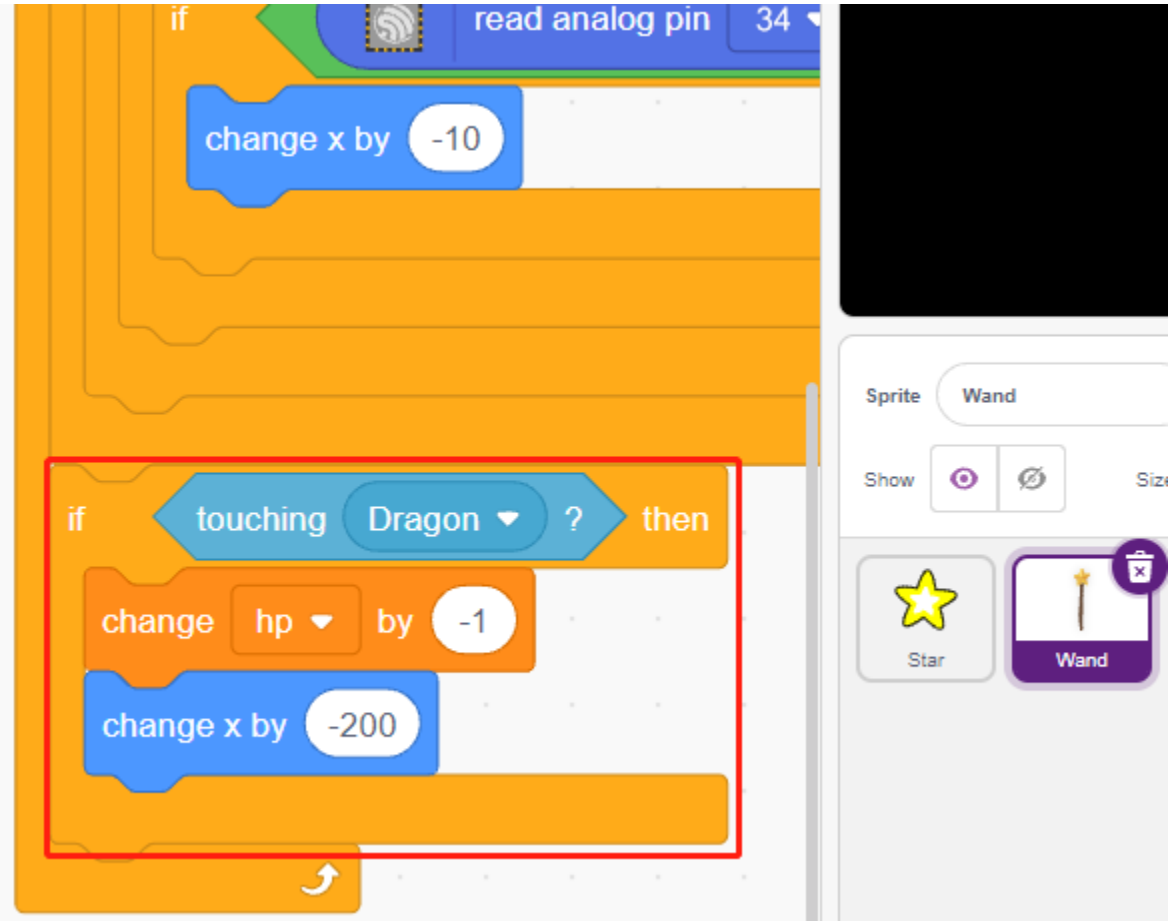
3. Kampf!

Der Zauberstab und der Drache sind derzeit noch im Konflikt miteinander, und wir werden sie gegeneinander kämpfen lassen. Der Drache ist stark, und der Zauberstab ist der mutige Mann, der gegen den Drachen kämpft. Die Interaktion zwischen ihnen besteht aus den folgenden Teilen.

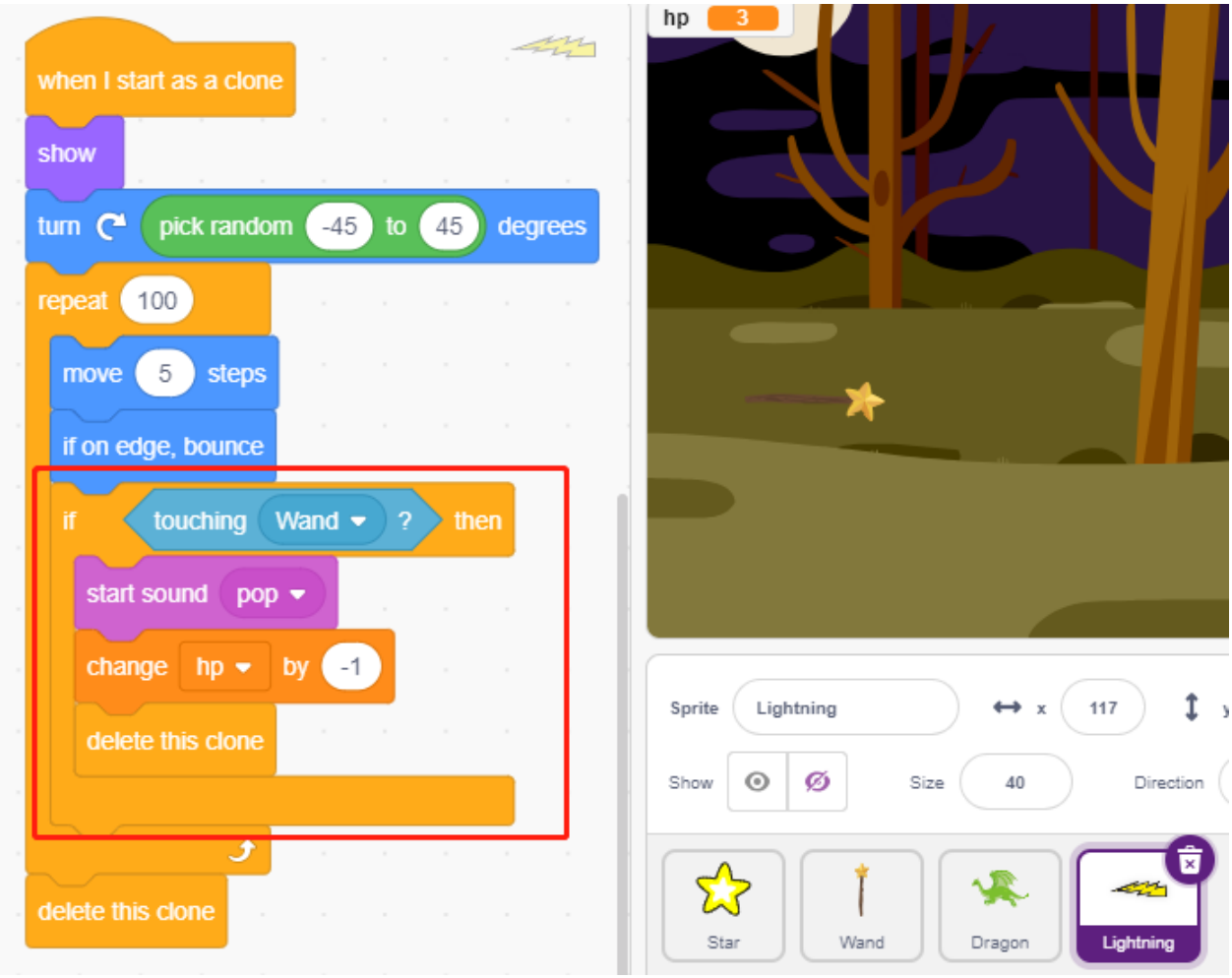
1. Wenn der Zauberstab den Drachen berührt, wird er zurückgestoßen und verliert Lebenspunkte.
2. Wenn Blitz den Zauberstab trifft, verliert der Zauberstab Lebenspunkte.
3. Wenn die Sternenkugel den Drachen trifft, verliert der Drache Lebenspunkte.

Sobald das geklärt ist, gehen wir weiter zur Änderung der Skripte für jedes Sprite.

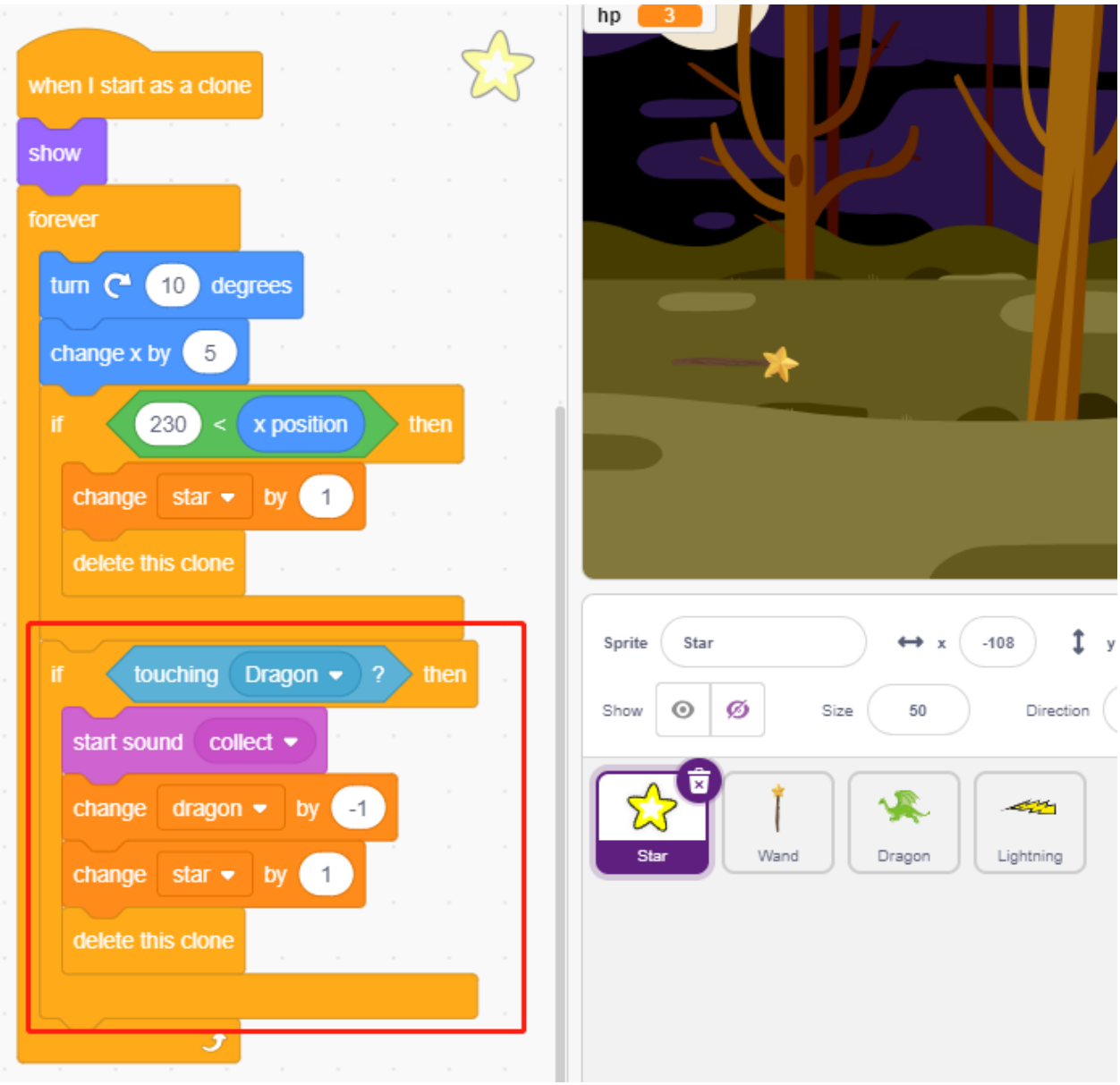
- Wenn der **Wand** den **Dragon** trifft, wird er zurückgestoßen und verliert Lebenspunkte.



- Wenn **Lightning** (ein Klon des **Lightning**-Sprites) das **Wand**-Sprite trifft, macht es ein Knallgeräusch und verschwindet, und der **Wand** verliert Lebenspunkte.



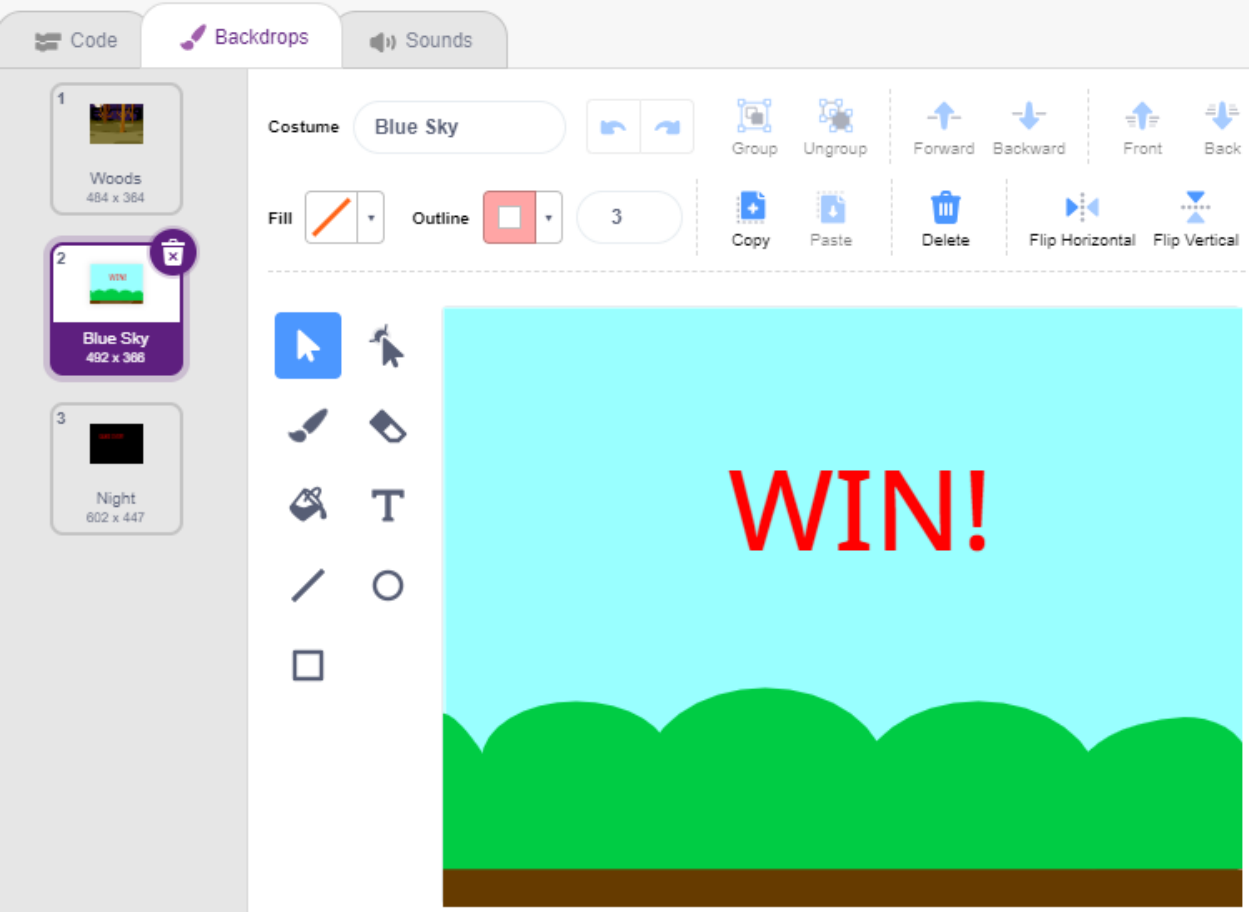
- Wenn ein **Star** (Klon des **Star**-Sprites) den **Dragon** trifft, gibt er ein Sammelgeräusch von sich und verschwindet, während er die **Star**-Zählung wiederherstellt, und der **Dragon** verliert Lebenspunkte.



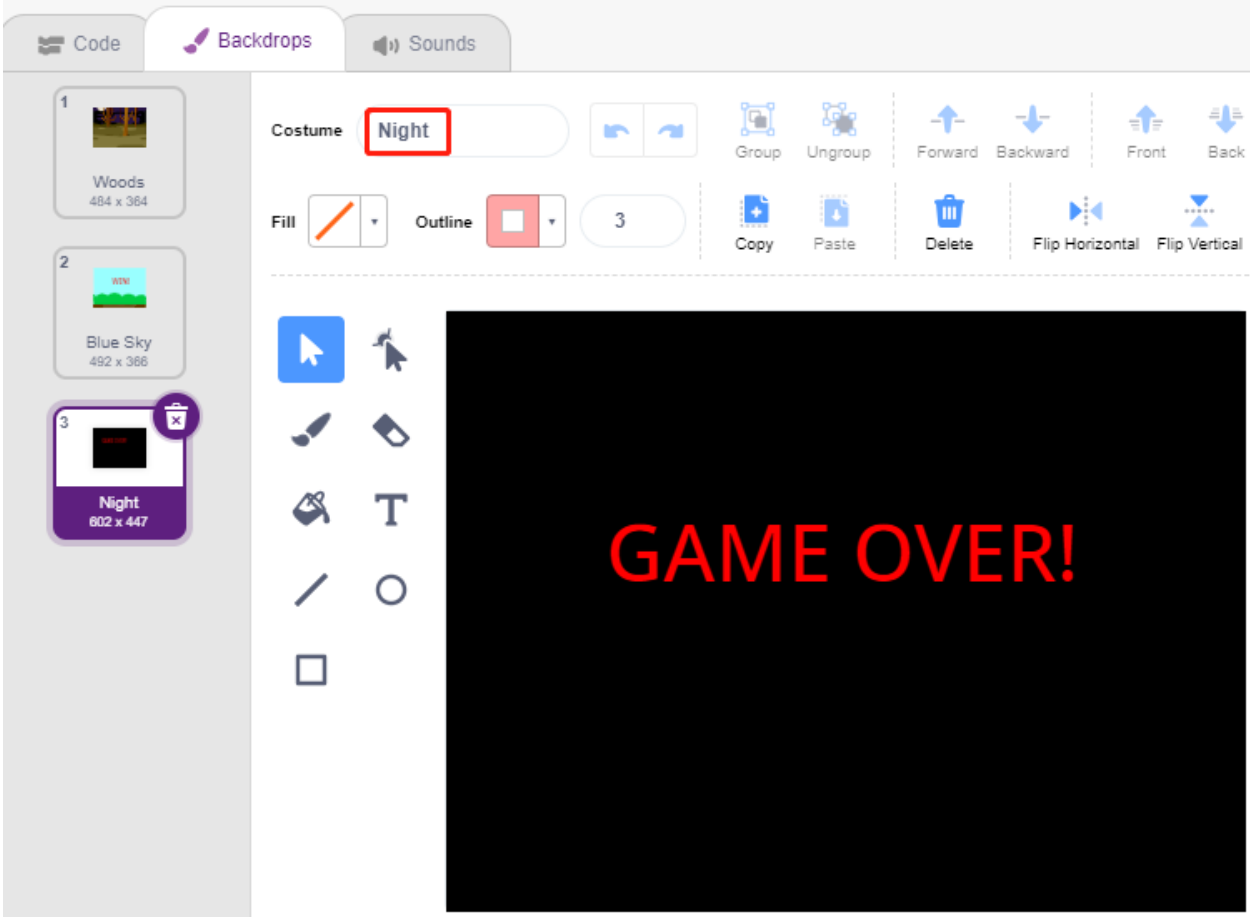
4. Bühne

Der Kampf zwischen dem **Wand** und dem **Dragon** wird letztendlich in Gewinner und Verlierer aufgeteilt, was wir mit der Bühne darstellen.

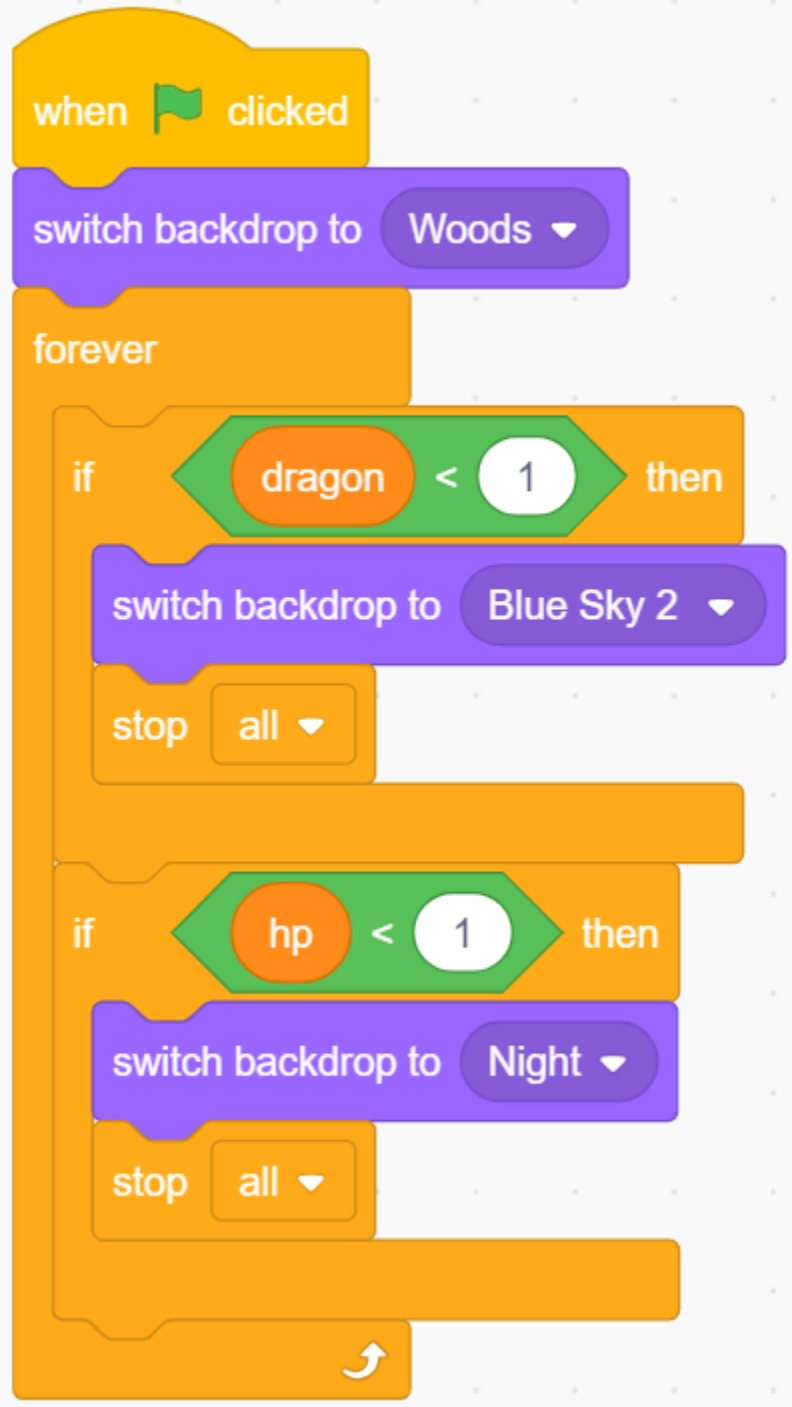
- Fügen Sie den **Blue Sky**-Hintergrund hinzu und schreiben Sie den Schriftzug „WIN!“ darauf, um darzustellen, dass der Drache besiegt wurde und die Dämmerung gekommen ist.



- Und ändern Sie den leeren Hintergrund wie folgt, um darzustellen, dass das Spiel fehlgeschlagen ist und alles in Dunkelheit versinken wird.



- Schreiben Sie nun ein Skript, um diese Hintergründe zu wechseln. Wenn die grüne Fahne angeklickt wird, wechseln Sie zum **Woods**-Hintergrund; wenn die Lebenspunkte des Drachens weniger als 1 betragen, ist das Spiel erfolgreich und wechseln Sie den Hintergrund zum **Blue Sky**; wenn der Lebenswert des **Wand** weniger als 1 beträgt, wechseln Sie zum **Night**-Hintergrund und das Spiel ist gescheitert.


when clicked
switch backdrop to Woods
forever
if dragon < 1 then
switch backdrop to Blue Sky 2
stop all
if hp < 1 then
switch backdrop to Night
stop all

KAPITEL 5

Lernen Sie die Komponenten in Ihrem Kit kennen

Nach dem Öffnen des Pakets überprüfen Sie bitte, ob die Anzahl der Komponenten mit der Produktbeschreibung übereinstimmt und ob alle Komponenten in einwandfreiem Zustand sind.

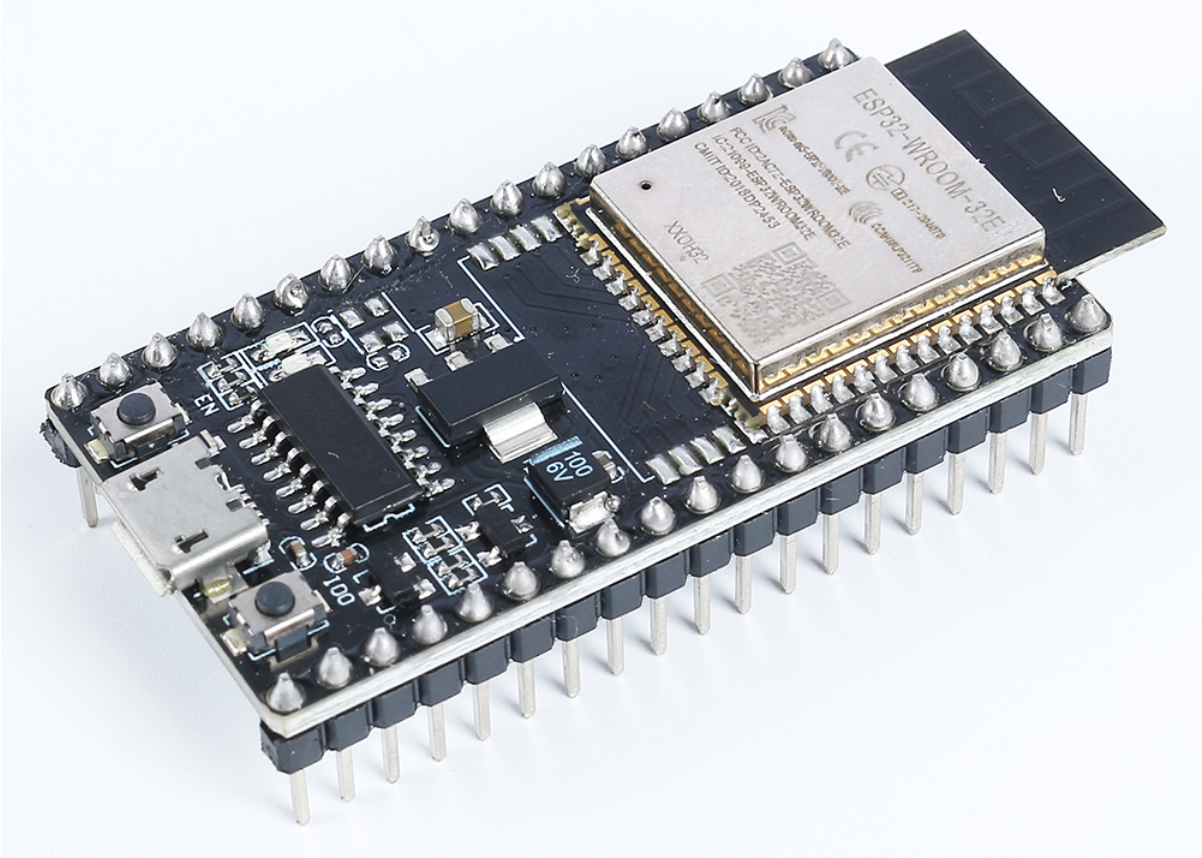
Packing List

Im Folgenden finden Sie eine Einführung in jede Komponente, einschließlich des Funktionsprinzips der Komponente und der entsprechenden Projekte.

Steuerplatine

5.1 ESP32 WROOM 32E

Der ESP32 WROOM-32E ist ein vielseitiges und leistungsstarkes Modul, basierend auf dem ESP32-Chipsatz von Espressif. Es bietet Dual-Core-Verarbeitung, integrierte Wi-Fi- und Bluetooth-Konnektivität und verfügt über eine breite Palette an Schnittstellen für Peripheriegeräte. Bekannt für seinen geringen Stromverbrauch, ist das Modul ideal für IoT-Anwendungen, da es intelligente Konnektivität und robuste Leistung in kompakten Formfaktoren ermöglicht.



Wesentliche Merkmale:

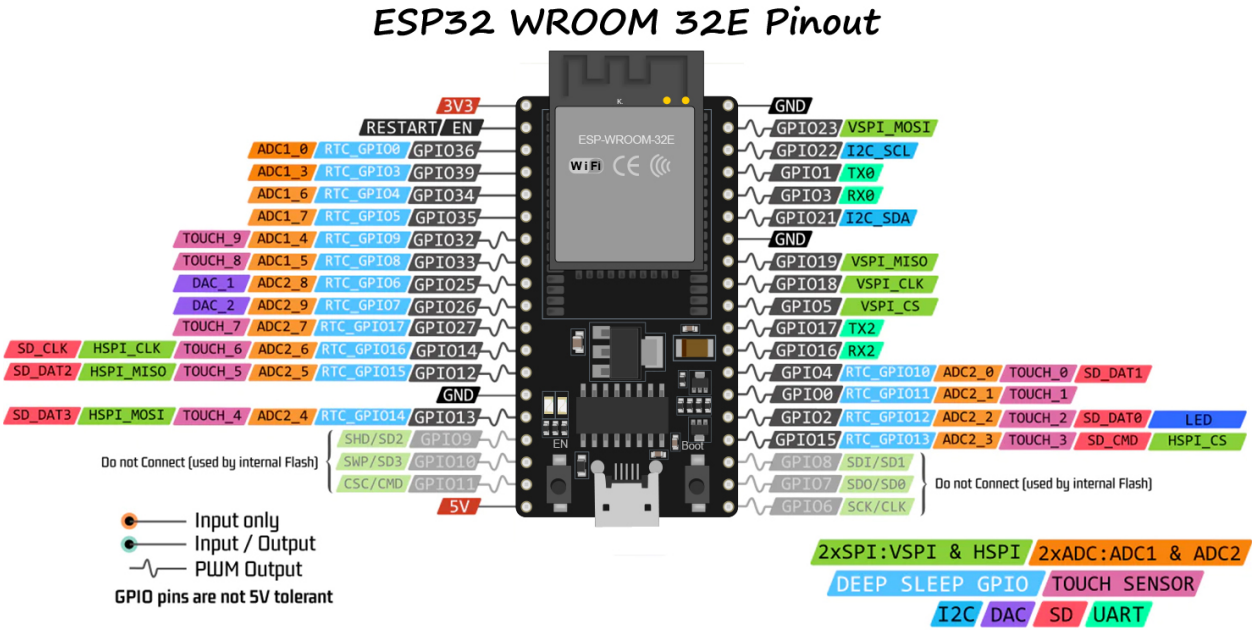
- **Verarbeitungsleistung**: Ausgestattet mit einem Dual-Core Xtensa® 32-Bit LX6-Mikroprozessor, bietet es Skalierbarkeit und Flexibilität.
- **Drahtlose Fähigkeiten**: Mit integriertem 2,4 GHz Wi-Fi und Dual-Mode-Bluetooth eignet es sich hervorragend für Anwendungen, die stabile drahtlose Kommunikation erfordern.
- **Speicher & Lagerung**: Es kommt mit reichlich SRAM und leistungsfähigem Flash-Speicher, um den Bedürfnissen von Benutzerprogrammen und Datenspeicherung gerecht zu werden.
- **GPIO**: Mit bis zu 34 GPIO-Pins unterstützt es eine Vielzahl externer Geräte und Sensoren.
- **Geringer Stromverbrauch**: Verschiedene Stromsparmodi sind verfügbar, was es ideal für batteriebetriebene oder energieeffiziente Szenarien macht.
- **Sicherheit**: Integrierte Verschlüsselungs- und Sicherheitsfunktionen gewährleisten den Schutz von Benutzerdaten und Privatsphäre.
- **Vielseitigkeit**: Vom einfachen Haushaltsgerät bis hin zu komplexer Industriemaschinerie bietet der WROOM-32E durchgehend konsistente, effiziente Leistung.

Zusammenfassend bietet der ESP32 WROOM-32E nicht nur robuste Verarbeitungskapazitäten und vielfältige Konnektivitätsoptionen, sondern zeichnet sich auch durch eine Reihe von Merkmalen aus, die ihn zu einer bevorzugten Wahl im IoT- und Smart-Device-Sektor machen.

-

5.1.1 Pinbelegungsdiagramm

Der ESP32 hat einige Einschränkungen bei der Pin-Nutzung aufgrund der gemeinsamen Nutzung bestimmter Pins für verschiedene Funktionen. Bei der Projektgestaltung ist es ratsam, die Pin-Nutzung sorgfältig zu planen und auf potenzielle Konflikte zu überprüfen, um eine ordnungsgemäße Funktion zu gewährleisten und Probleme zu vermeiden.



Hier sind einige der wichtigsten Einschränkungen und Überlegungen:

- **ADC1 und ADC2**: ADC2 kann nicht verwendet werden, wenn WiFi oder Bluetooth aktiv sind. ADC1 kann jedoch ohne Einschränkungen genutzt werden.
- **Bootstrap-Pins**: GPIO0, GPIO2, GPIO5, GPIO12 und GPIO15 werden während des Bootvorgangs für das Bootstrapping verwendet. Es sollte darauf geachtet werden, keine externen Komponenten anzuschließen, die den Bootvorgang auf diesen Pins stören könnten.
- **JTAG-Pins**: GPIO12, GPIO13, GPIO14 und GPIO15 können als JTAG-Pins für Debugging-Zwecke verwendet werden. Sind JTAG-Debugging-Funktionen nicht erforderlich, können diese Pins als reguläre GPIOs genutzt werden.
- **Touch-Pins**: Einige Pins unterstützen Touch-Funktionalitäten. Diese Pins sollten vorsichtig verwendet werden, wenn sie für die Touch-Erkennung genutzt werden sollen.
- **Strom-Pins**: Einige Pins sind für strombezogene Funktionen reserviert und sollten entsprechend verwendet werden. Beispielsweise sollte vermieden werden, übermäßigen Strom von Versorgungspins wie 3V3 und GND zu ziehen.
- **Nur-Eingangs-Pins**: Einige Pins sind ausschließlich Eingänge und sollten nicht als Ausgänge verwendet werden.

5.1.2 Strapping-Pins

Der ESP32 verfügt über fünf Strapping-Pins:

Strapping-Pins	Beschreibung
IO5	Standardmäßig auf Pull-up eingestellt, das Spannungsniveau von IO5 und IO15 beeinflusst das Timing des SDIO-Slave.
IO0	Standardmäßig auf Pull-up eingestellt, wenn auf Low gezogen, tritt der Downloadmodus ein.
IO2	Standardmäßig auf Pull-down eingestellt, IO0 und IO2 bringen den ESP32 in den Downloadmodus.
IO12(MTDI)	Standardmäßig auf Pull-down eingestellt, wenn auf High gezogen, startet der ESP32 nicht normal.
IO15(MTDO)	Standardmäßig auf Pull-up eingestellt, wenn auf Low gezogen, ist das Debug-Log nicht sichtbar. Zusätzlich beeinflusst das Spannungsniveau von IO5 und IO15 das Timing des SDIO-Slave.

Software kann die Werte dieser fünf Bits aus dem Register „GPIO_STRAPPING“ auslesen. Während des System-Resets des Chips (Power-on-Reset, RTC-Watchdog-Reset und Brownout-Reset) nehmen die Latches der Strapping-Pins die Spannungsebene als Strapping-Bits von „0“ oder „1“ auf und halten diese Bits, bis der Chip abgeschaltet oder heruntergefahren wird. Die Strapping-Bits konfigurieren den Boot-Modus des Geräts, die Betriebsspannung von VDD_SDIO und andere anfängliche Systemeinstellungen.

Jeder Strapping-Pin ist während des Chip-Resets mit seinem internen Pull-up/Pull-down verbunden. Folglich bestimmt der interne schwache Pull-up/Pull-down das Standard-Eingangsniveau der Strapping-Pins, wenn ein Strapping-Pin unverbunden ist oder der verbundene externe Stromkreis eine hohe Impedanz aufweist.

Um die Strapping-Bit-Werte zu ändern, können Benutzer externe Pull-down/Pull-up-Widerstände anwenden oder die GPIOs des Host-MCUs verwenden, um das Spannungsniveau dieser Pins beim Einschalten des ESP32 zu steuern.

Nach dem Reset-Release arbeiten die Strapping-Pins als Pins mit normaler Funktion. Die folgende Tabelle gibt detaillierte Informationen zur Boot-Modus-Konfiguration durch Strapping-Pins.

Voltage of Internal LDO (VDD_SDIO)			
Pin	Default	3.3 V	1.8 V
MTDI	Pull-down	0	1
Booting Mode			
Pin	Default	SPI Boot	Download Boot
GPIO0	Pull-up	1	0
GPIO2	Pull-down	Don't-care	0
Enabling/Disabling Debugging Log Print over U0TXD During Booting			
Pin	Default	U0TXD Active	U0TXD Silent
MTDO	Pull-up	1	0
Timing of SDIO Slave			

Pin	Default	FE Sampling FE Output	FE Sampling RE Output	RE Sampling FE Output	RE Sampling RE Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

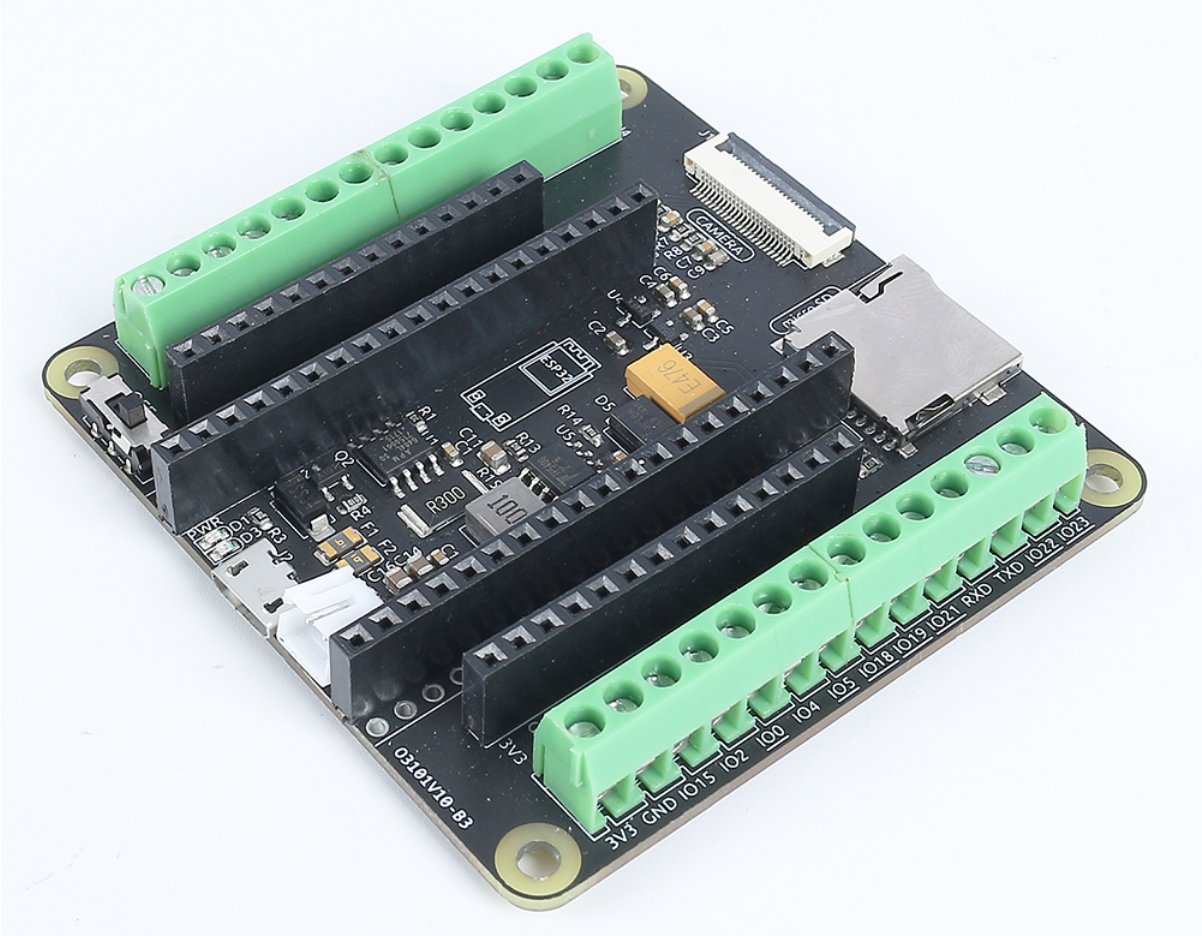
- FE: fallende Flanke, RE: steigende Flanke
- Die Firmware kann Registereinstellungen konfigurieren, um die Einstellungen von „Spannung des internen LDO (VDD_SDIO)“ und „Timing des SDIO-Slaves“ nach dem Booten zu ändern.
- Das Modul integriert einen 3,3 V SPI-Flash, daher kann der Pin MTDI beim Einschalten des Moduls nicht auf 1 gesetzt werden.

5.2 ESP32-Kameraerweiterung

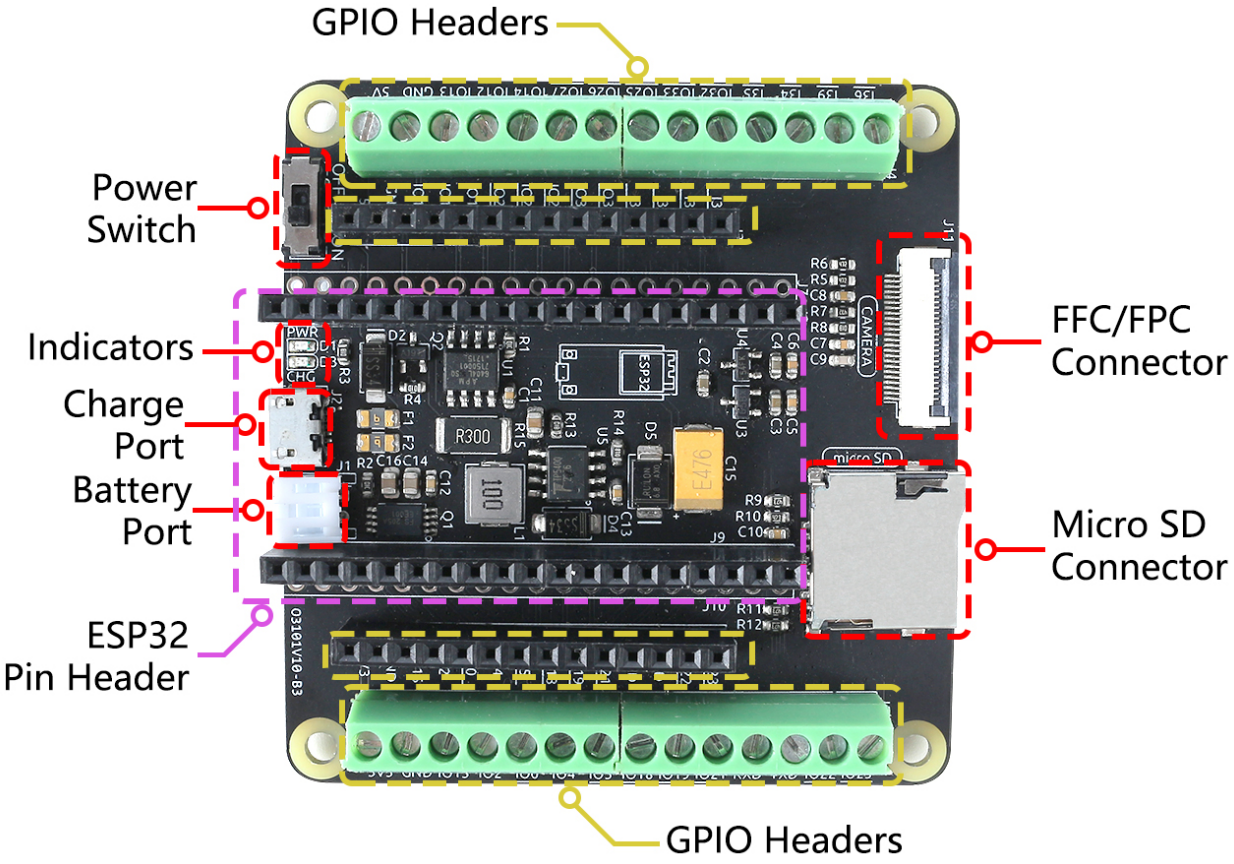
Wir haben ein Erweiterungsboard entworfen, das Ihnen ermöglicht, die Kamera- und SD-Karten-Funktionen des ESP32 WROOM 32E vollständig zu nutzen. Mit der Kombination aus der OV2640-Kamera, Micro-SD und ESP32 WROOM 32E erhalten Sie ein All-in-One-Erweiterungsboard.

Das Board bietet zwei Arten von GPIO-Headern – einen mit weiblichen Headern, perfekt für schnelle Prototyping-Projekte. Der andere Typ verfügt über Schraubklemmen, die stabile Drahtverbindungen gewährleisten und sich somit für IoT-Projekte eignen.

Zusätzlich können Sie Ihr Projekt mit einer einzelnen 3,7V 18650-Batterie betreiben. Wenn die Batterie leer ist, können Sie sie bequem aufladen, indem Sie einfach ein 5V Micro-USB-Kabel anschließen. Dies macht es zu einem großartigen Werkzeug für Outdoor-Projekte und Fernanwendungen.


CAMERA
ESP32
R300
PWR
3V3 GND IO15 IO2 IO0 IO4 IO5 IO18 IO19 IO21 RXD TXD IO22 IO23
0319IV10-B3

5.2.1 Schnittstellen-Einführung



- **Power Switch**
  - Steuert das Ein- und Ausschalten der Batteriestromversorgung.
- **Charging Port**
  - Beim Anschließen eines 5V Micro-USB-Kabels kann die Batterie aufgeladen werden.
- **Battery Port**
  - Verfügt über eine PH2.0-2P-Schnittstelle, kompatibel mit 3,7V 18650 Lithium-Batterien.
  - Versorgt sowohl das ESP32 WROOM 32E als auch die ESP32-Kameraerweiterung mit Strom.
- **ESP32 Pin Headers**
  - Bestimmt für das ESP32 WROOM 32E-Modul. Achten Sie auf die korrekte Orientierung; stellen Sie sicher, dass beide Micro-USB-Ports auf dieselbe Seite zeigen, um eine falsche Platzierung zu vermeiden.
- **GPIO Headers**
  - **Weibliche Header**: Zum Anschließen verschiedener Komponenten an das ESP32, perfekt für schnelle Prototyping-Projekte.
  - **Schraubklemme**: 3,5mm-Pitch 14pin-Schraubklemme, gewährleistet stabile Drahtverbindungen und eignet sich für IoT-Projekte.
- **Indicator Lights**

  - **PWR**: Leuchtet auf, wenn die Batterie eingeschaltet ist oder wenn ein Micro-USB direkt an das ESP32 angeschlossen ist.
  - **CHG**: Leuchtet auf, wenn ein Micro-USB an den Ladeanschluss der Platine angeschlossen wird, was den Beginn des Ladevorgangs anzeigt. Es erlischt, sobald die Batterie vollständig aufgeladen ist.

- **Micro SD Connector**
  - Federbeladener Steckplatz für einfaches Einsetzen und Entnehmen der Micro-SD-Karte.
- **24-pin 0.5mm FFC / FPC connector**
  - Entwickelt für die OV2640-Kamera, geeignet für verschiedene vision-bezogene Projekte.

5.2.2 ESP32 Kameraerweiterung Anschlussplan

Der Anschlussplan des ESP32 WROOM 32E ist unter *Pinbelegungsdiagramm* zu finden.

Wenn der ESP32 WROOM 32E jedoch auf das Erweiterungsboard gesteckt wird, können einige seiner Pins auch zur Steuerung der Micro SD-Karte oder einer Kamera verwendet werden.

Daher wurden diesen Pins Pull-up- oder Pull-down-Widerstände hinzugefügt. Wenn Sie diese Pins als Eingänge verwenden, ist es entscheidend, diese Widerstände zu berücksichtigen, da sie die Eingangspegel beeinflussen können.

Hier ist die Pinbelegung für die rechte Seite:

ESP32 WROOM 32E + Camera Extension Pinout												
	IO13	IO12	IO14	IO27	IO26	IO25	IO33	IO32	I35	I34	I39	I36
PWM Output												
Input/Output												
Input Only												
Analog	ADC2_4	ADC2_5	ADC2_6	ADC2_7	ADC2_9	ADC2_8	ADC1_5	ADC1_4	ADC1_7	ADC1_6	ADC1_3	ADC1_0
Touch Sensor	TOUCH_4	TOUCH_5	TOUCH_6	TOUCH_7			TOUCH_8	TOUCH_9				
DAC					DAC_2	DAC_1						
I2C												
UART												
SPI	H_MOSI	H_MISO	H_CLK									
LED												
Strapping												
SD	DAT3	DAT2	CLK									
Camera												
Pull-up 47K Resistor												
Pull-up 4.7K resistor												
Pull-down 1K resistor												

Und hier die Pinbelegung für die linke Seite:

ESP32 WROOM 32E + Camera Extension Pinout												
	IO15	IO2	IO0	IO4	IO5	IO18	IO19	IO21	RXD	TXD	IO22	IO23
PWM Output												
Input/Output												
Input Only												
Analog	ADC1_0	ADC2_3	ADC2_2	ADC2_1	ADC2_0							
Touch Sensor	TOUCH_3	TOUCH_2	TOUCH_1	TOUCH_0								
DAC												
I2C								SDA			SCL	
UART									RXD	TXD		
SPI	H_CS				V_CS	V_CLK	V_MISO					V_MOSI
LED		LED										
Strapping												
SD	CMD	DAT0		DAT1								
Camera												
Pull-up 47K Resistor												
Pull-up 4.7K resistor												
Pull-down 1K resistor												

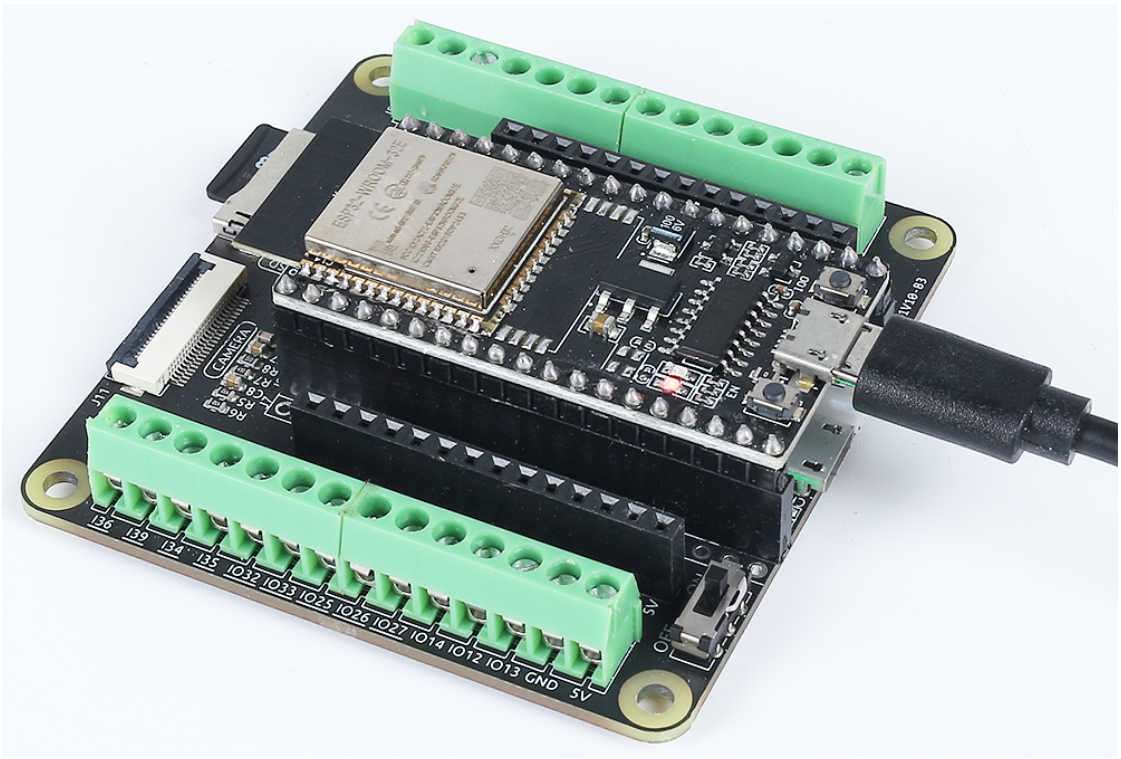
Bemerkung: Es gibt einige spezifische Einschränkungen:

- **IO33** ist mit einem 4,7K Pull-up-Widerstand und einem Filterkondensator verbunden, was verhindert, dass er den WS2812 RGB-Streifen ansteuert.

5.2.3 Anleitung zum Einsetzen der Schnittstellen

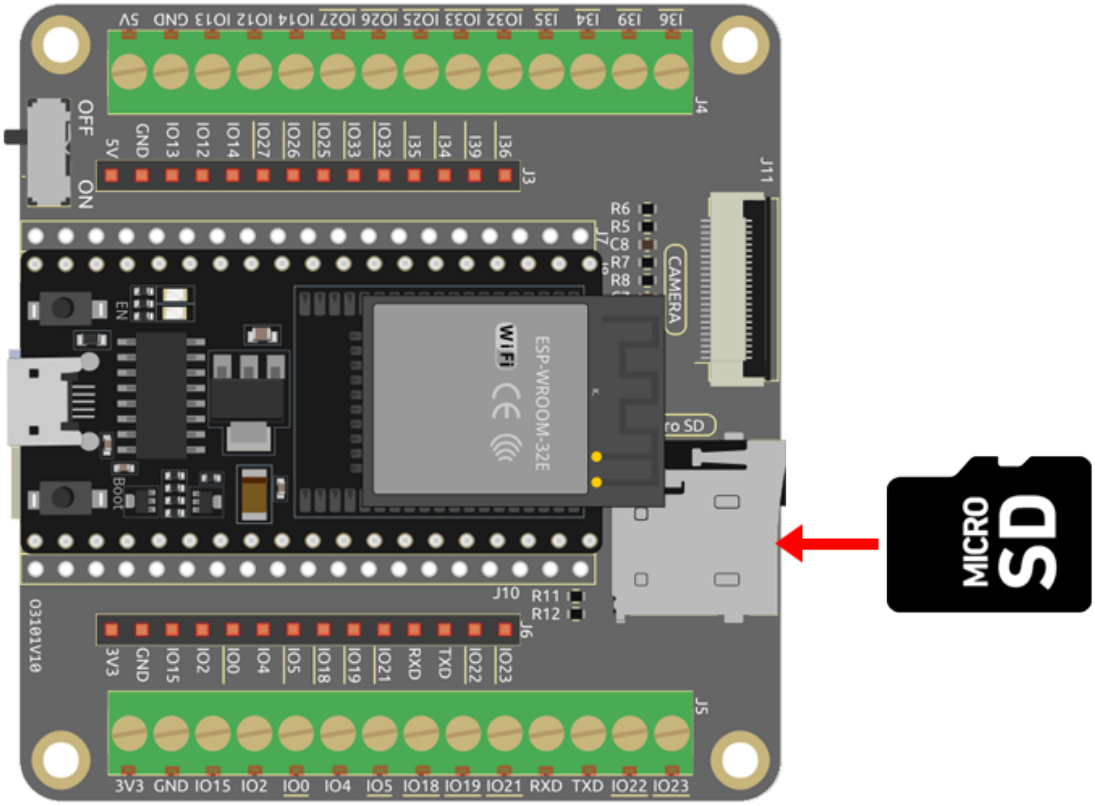
Code hochladen

Um Code auf den ESP32 WROOM 32E zu laden, verbinden Sie ihn über ein Micro USB-Kabel mit Ihrem Computer.



Einsetzen der Micro SD-Karte

Drücken Sie die Micro SD-Karte vorsichtig hinein, um sie zu sichern. Ein weiteres Drücken wird sie auswerfen.

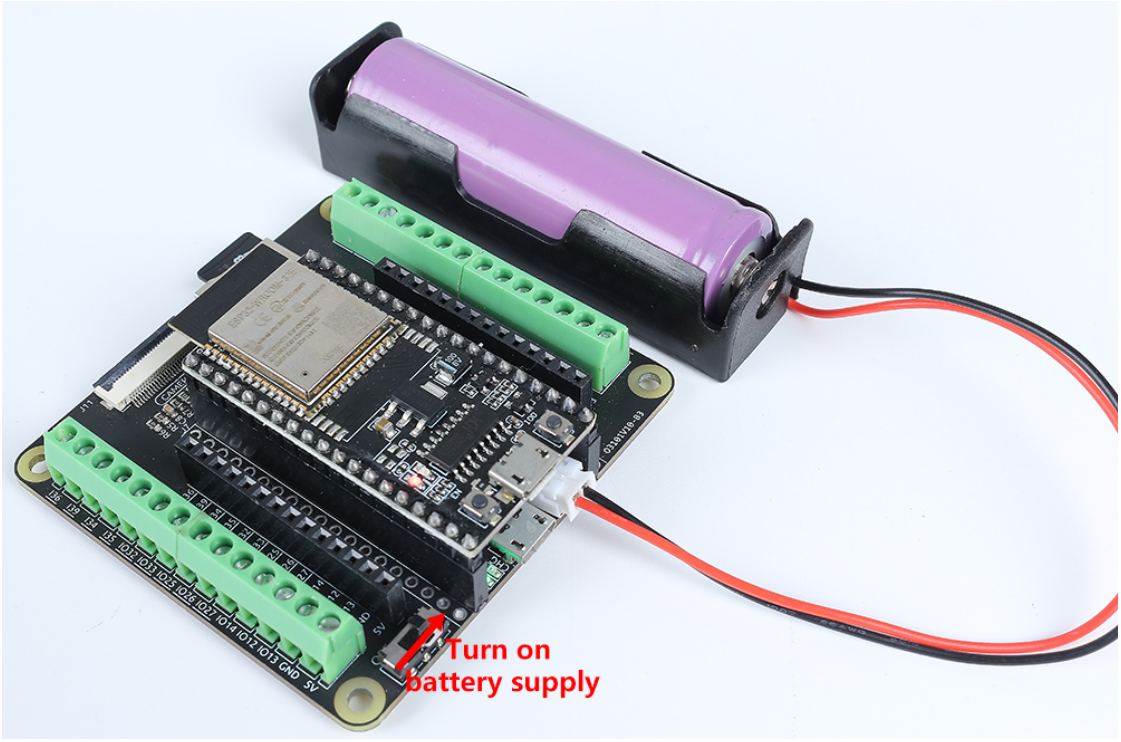


Anschließen der Kamera

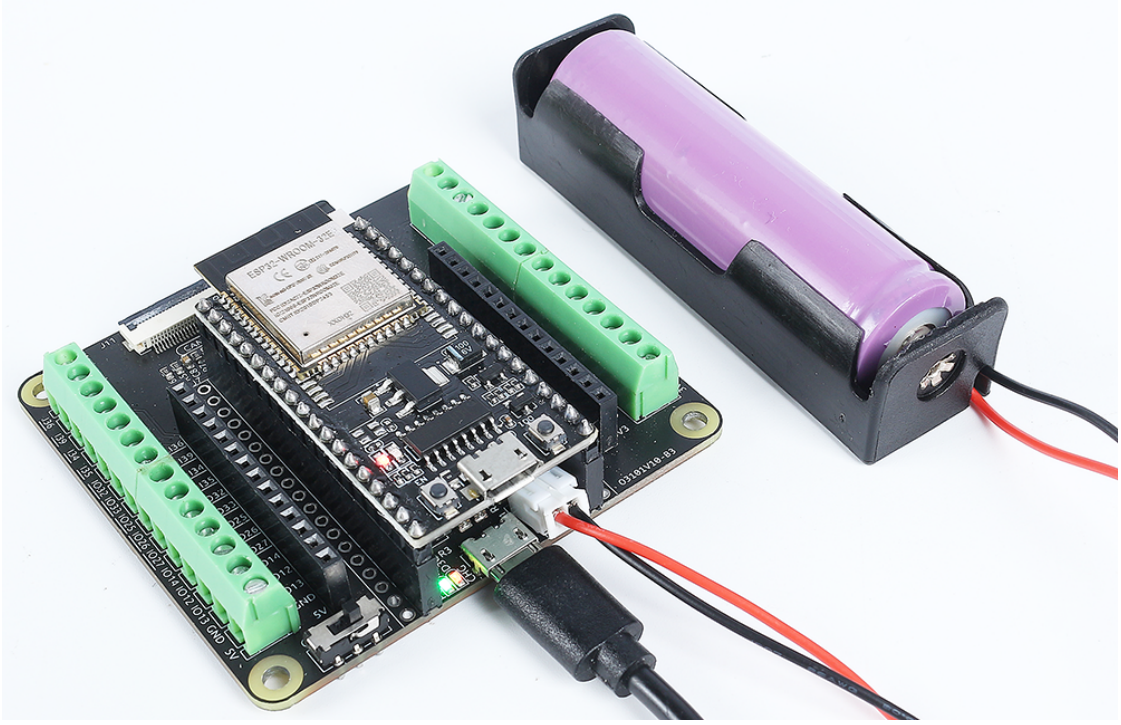
Beim Anschließen der Kamera achten Sie darauf, dass der schwarze Streifen des FPC-Kabels nach oben zeigt und vollständig in den Stecker eingeführt ist.

Batteriebetrieb und Aufladen

Stecken Sie das Batteriekabel vorsichtig in den Batterieanschluss, um zu vermeiden, dass Sie zu viel Kraft aufwenden und das Batterieterminal nach oben drücken. Wenn das Terminal nach oben gedrückt wird, ist es in Ordnung, solange die Stifte nicht gebrochen sind; Sie können es einfach wieder in Position drücken.



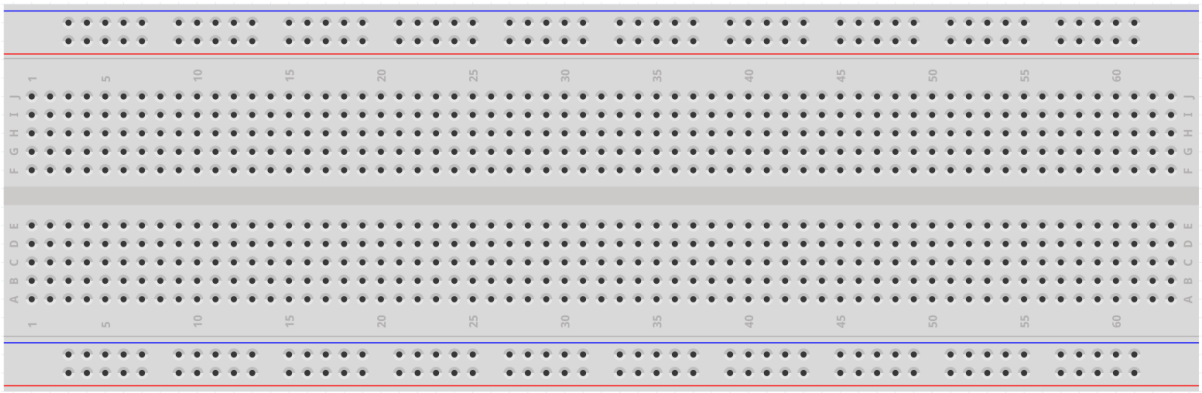
Wenn die Batterie entladen ist, schließen Sie ein 5V Micro USB-Kabel an, um sie aufzuladen.



Grundlegendes

5.3 Steckbrett

Was ist ein „lötfreies“ Steckbrett?



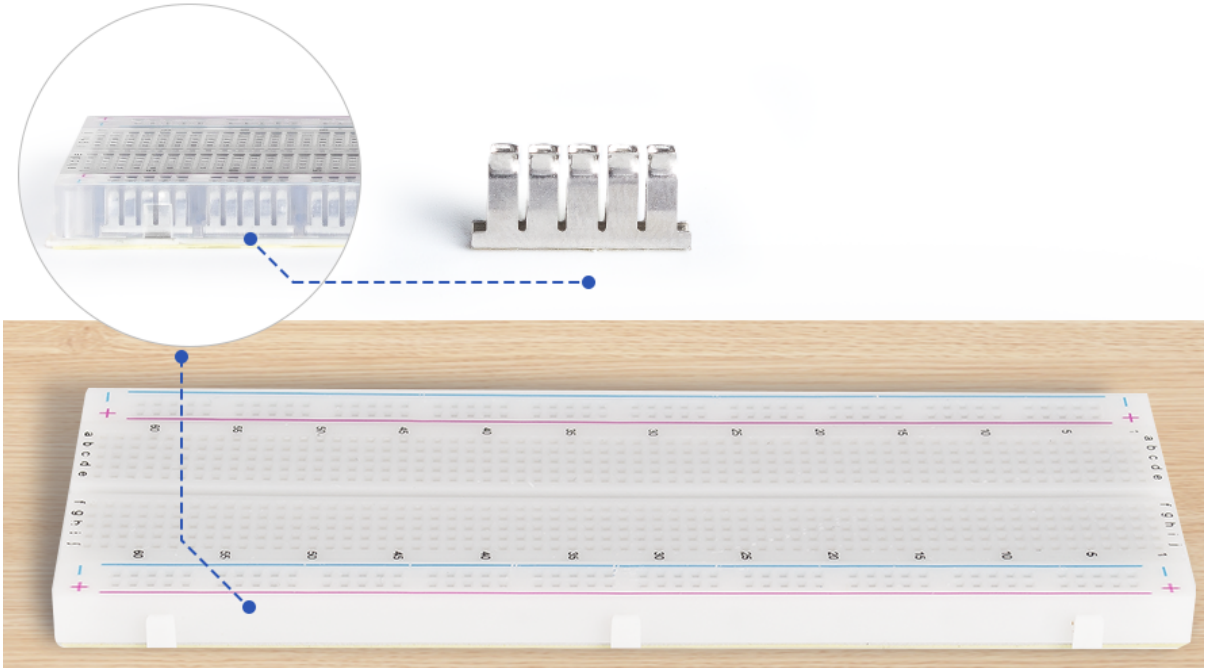
Ein Steckbrett ist eine rechteckige Kunststoffplatte mit vielen kleinen Löchern. Diese Löcher ermöglichen es Ihnen, elektronische Bauteile einzufügen, um Schaltungen zu bauen. Technisch gesehen sind diese Steckbretter als lötfreie Steckbretter bekannt, da sie keine Lötverbindungen zur Herstellung von Kontakten benötigen.

Merkmale

- Größe: 163 x 54 x 8 mm
- 830 Anschlusspunkte: 630 Anschlusspunkte für IC-Schaltkreise plus 2x100 Anschlussstreifen mit 4 Stromschienen.
- Drahtgröße: Geeignet für 20-29 AWG Drähte.

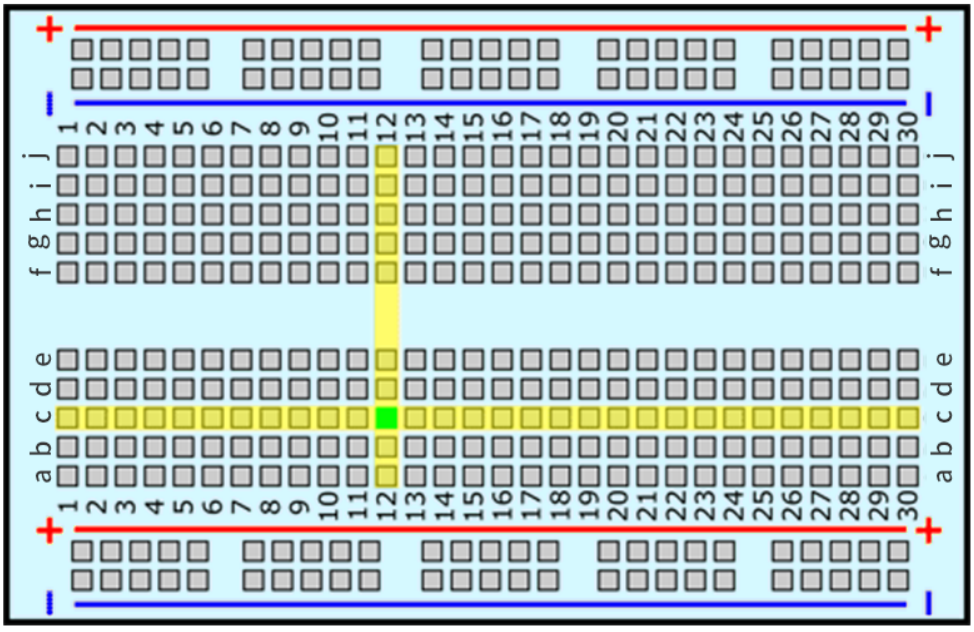
- Material: ABS-Kunststoffplatte, verzinntes Phosphorbronze-Kontaktblech.
- Spannung / Strom: 300V/3-5A.
- Mit Selbstklebeband auf der Rückseite

Was befindet sich im Steckbrett?



Das Innere des Steckbretts besteht aus Reihen kleiner Metallklammern. Wenn Sie die Anschlüsse eines Bauteils in die Löcher des Steckbretts stecken, wird einer der Klammern dies erfassen. Manche Steckbretter bestehen tatsächlich aus durchsichtigem Kunststoff, sodass Sie die Klammern im Inneren sehen können.

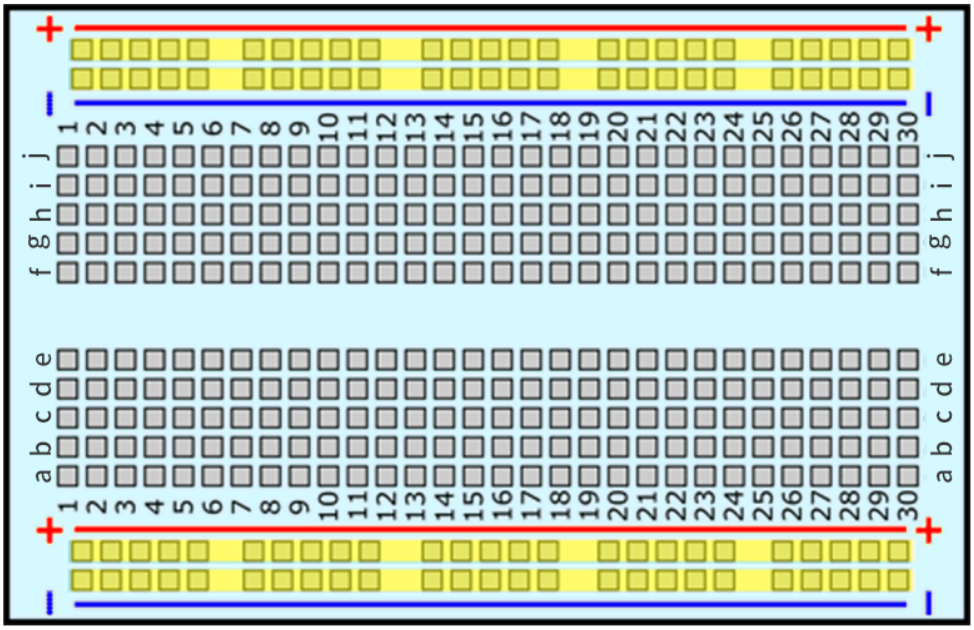
Was bedeuten die Buchstaben und Zahlen auf einem Steckbrett?



Die meisten Steckbretter haben einige Zahlen, Buchstaben und Plus- und Minuszeichen. Obwohl sich die Beschriftungen von Steckbrett zu Steckbrett unterscheiden, ist die Funktion im Grunde gleich. Diese Beschriftungen helfen Ihnen, die entsprechenden Löcher beim Bau Ihres Schaltkreises schneller zu finden.

Die Reihennummern und Spaltenbuchstaben helfen Ihnen, die Löcher auf dem Steckbrett präzise zu lokalisieren, zum Beispiel befindet sich das Loch „C12“ dort, wo Spalte C auf Reihe 12 trifft.

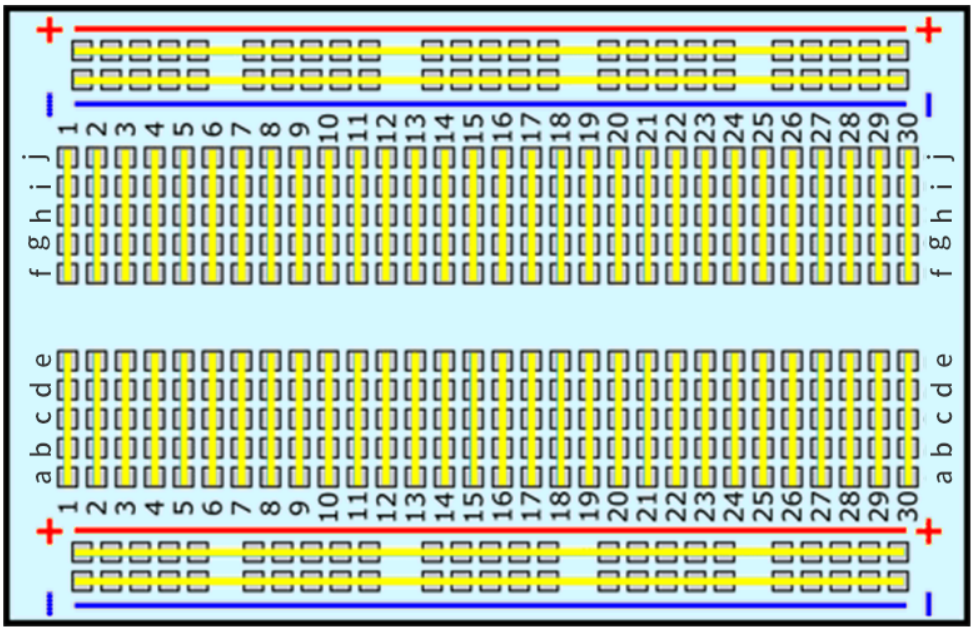
Was bedeuten die farbigen Linien und Plus- und Minuszeichen?



Die Seiten des Steckbretts sind in der Regel durch rote und blaue (oder andere Farben) sowie Plus- und Minuszeichen gekennzeichnet und werden üblicherweise verwendet, um eine Verbindung zur Stromversorgung herzustellen, bekannt als Stromschiene.

Beim Aufbau eines Schaltkreises ist es üblich, den negativen Pol mit der blauen (-) Spalte und den positiven Pol mit der roten (+) Spalte zu verbinden.

Wie sind die Löcher verbunden?

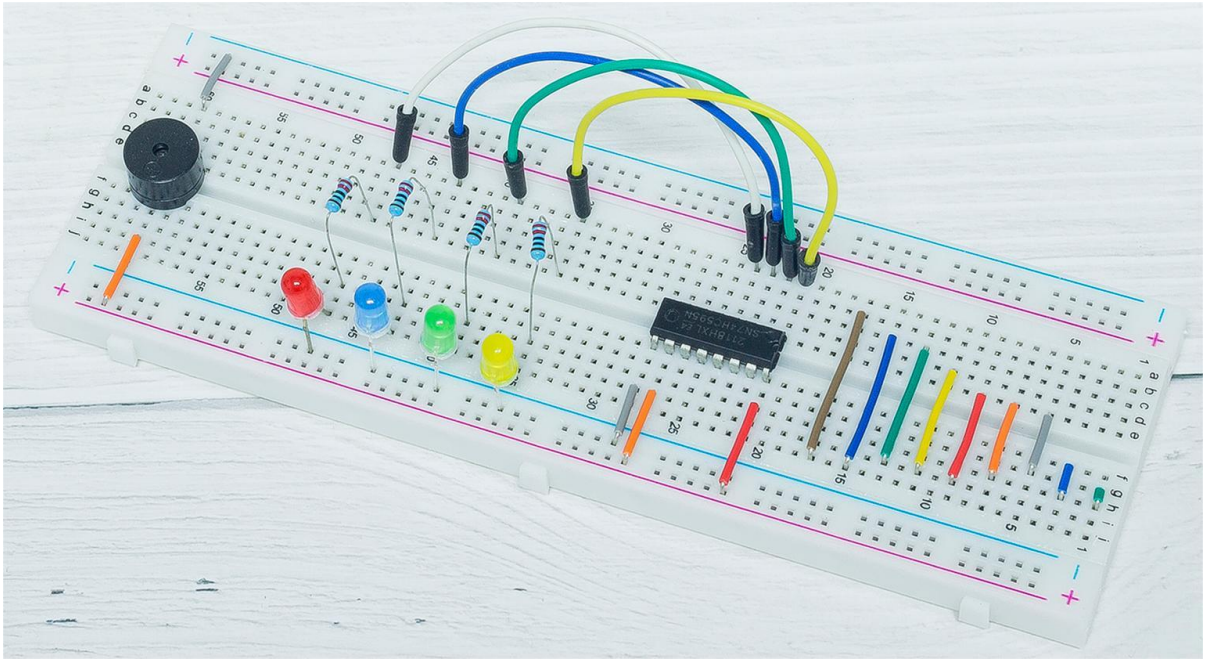


Wie in der Abbildung dargestellt, ist jede Gruppe von fünf Löchern im mittleren Bereich, Spalten A-E oder F-J, elektrisch miteinander verbunden. Das bedeutet beispielsweise, dass Loch A1 elektrisch mit den Löchern B1, C1, D1 und E1 verbunden ist.

Es ist nicht mit Loch A2 verbunden, da dieses Loch in einer anderen Reihe liegt, die eine separate Gruppe von Metallklammern hat. Es ist auch nicht mit den Löchern F1, G1, H1, I1 oder J1 verbunden, da diese sich in der anderen „Hälfte“ des Steckbretts befinden - die Klammern sind nicht über die mittlere Lücke hinweg verbunden.

Im Gegensatz zum mittleren Abschnitt, der in Fünfergruppen unterteilt ist, sind die Busse an den Seiten separat elektrisch verbunden. Zum Beispiel ist die Spalte, die mit Blau (-) markiert ist, als Ganzes elektrisch verbunden, und die Spalte, die mit Rot (+) markiert ist, ist ebenfalls elektrisch verbunden.

Welche elektronischen Teile sind mit Steckbrettern kompatibel?



Viele elektronische Komponenten haben lange Metallbeine, die als Anschlüsse bezeichnet werden. Fast alle Komponenten mit Anschlüssen funktionieren mit einem Steckbrett. Komponenten wie Widerstände, Kondensatoren, Schalter, Dioden usw. können in jede der Reihen eingefügt werden, aber ICs müssen über die mittlere Lücke hinweg angeordnet werden.

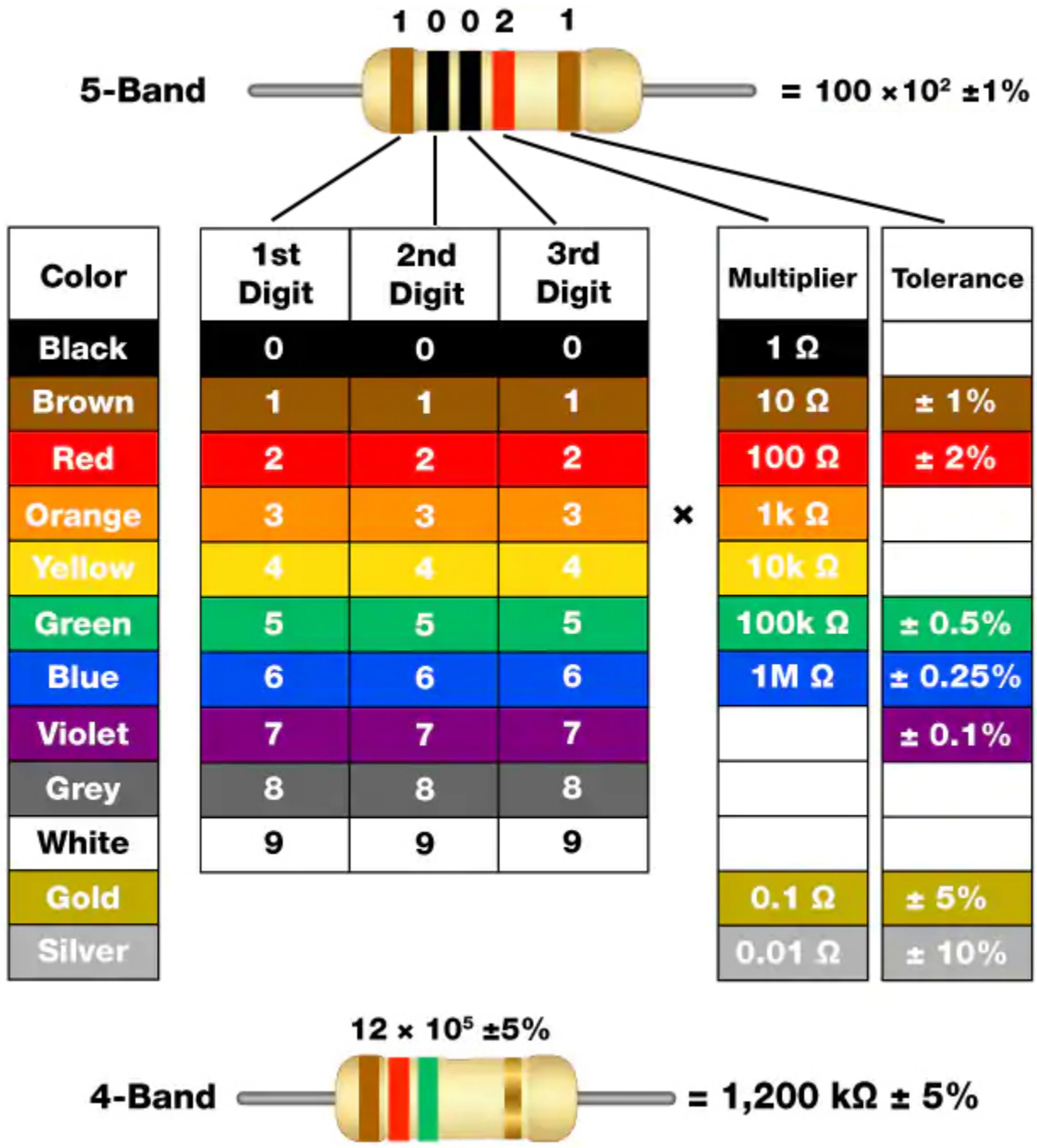
5.4 Widerstand

Ein Widerstand ist ein elektronisches Bauelement, das dazu dient, den Strom in einem Zweig zu begrenzen. Ein Festwiderstand ist eine Art von Widerstand, dessen Widerstandswert nicht verändert werden kann, im Gegensatz zu einem Potentiometer oder einem variablen Widerstand, bei denen der Widerstandswert einstellbar ist.

Es gibt zwei allgemein verwendete Schaltzeichen für Widerstände. Normalerweise ist der Widerstandswert darauf vermerkt. Wenn diese Symbole also in einem Schaltplan auftauchen, repräsentieren sie einen Widerstand.

 ist die Einheit des Widerstands, größere Einheiten sind K, M usw. Ihre Beziehung lässt sich wie folgt darstellen: 1 M = 1000 K, 1 K = 1000 . Normalerweise ist der Wert des Widerstands darauf vermerkt.

Bei der Verwendung eines Widerstands muss man zuerst seinen Widerstandswert kennen. Hier gibt es zwei Methoden: Sie können die Farbringe auf dem Widerstand beobachten oder ein Multimeter verwenden, um den Widerstand zu messen. Die erste Methode wird empfohlen, da sie bequemer und schneller ist.



Wie auf der Karte dargestellt, steht jede Farbe für eine Zahl.

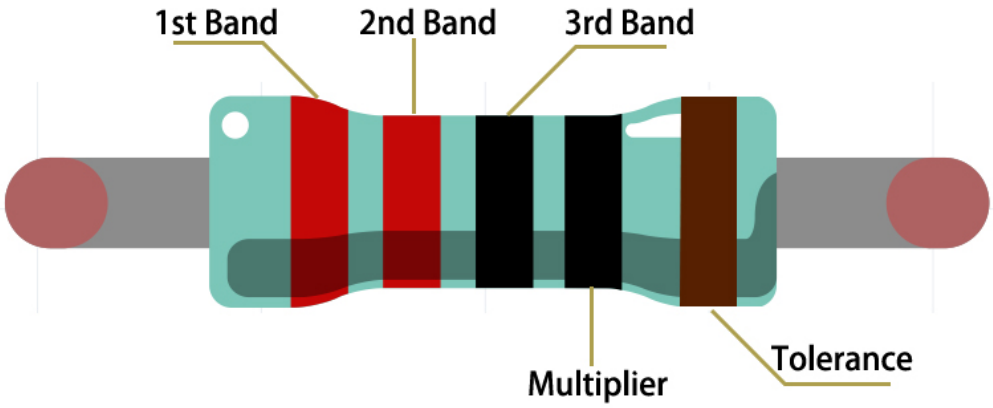
Schwarz	Braun	Rot	Orange	Gelb	Grün	Blau	Violett	Grau	Weiß	Gold	Silber
0	1	2	3	4	5	6	7	8	9	0,1	0,01

Häufig verwendet werden 4- und 5-Ring-Widerstände, auf denen sich 4 bzw. 5 farbige Ringe befinden.

Normalerweise kann es schwierig sein, bei einem neuen Widerstand zu entscheiden, an welchem Ende man mit dem Lesen der Farben beginnen soll. Ein Tipp ist, dass der Abstand zwischen dem vierten und fünften Ring vergleichsweise größer ist.

Deshalb können Sie den Abstand zwischen den beiden Farbringen an einem Ende des Widerstands beobachten; ist dieser größer als jeder andere Ringabstand, dann sollten Sie von der gegenüberliegenden Seite beginnen zu lesen.

Sehen wir uns an, wie der Widerstandswert eines 5-Ring-Widerstands wie unten dargestellt gelesen wird.



Für diesen Widerstand sollte der Wert also von links nach rechts gelesen werden. Der Wert sollte in diesem Format sein: 1. Band 2. Band 3. Band x 10^Multiplikator () und der zulässige Fehler ist ±Toleranz%. Der Widerstandswert dieses Widerstands beträgt daher 2(rot) 2(rot) 0(schwarz) x 10^0(schwarz) = 220 , und der zulässige Fehler beträgt ± 1% (braun).

Weitere Informationen über Widerstände finden Sie auf Wiki: Widerstand – Wikipedia.

5.5 Kondensator

Ein Kondensator bezeichnet die Menge der Ladungsspeicherung unter einer gegebenen Potenzialdifferenz, bezeichnet als C, mit der internationalen Einheit Farad (F). Allgemein gesprochen bewegen sich elektrische Ladungen unter Einwirkung einer Kraft in einem elektrischen Feld. Wenn ein Medium zwischen Leitern vorhanden ist, wird die Bewegung elektrischer Ladungen behindert und die elektrischen Ladungen sammeln sich an den Leitern, was zu einer Ansammlung von elektrischen Ladungen führt.

Die Menge der gespeicherten elektrischen Ladungen wird als Kapazität bezeichnet. Da Kondensatoren zu den am weitesten verbreiteten elektronischen Komponenten in elektronischen Geräten gehören, finden sie breite Anwendung in Gleichstrom-Isolation, Kopplung, Bypass, Filterung, Abstimmkreisen, Energieumwandlung und Steuerungsschaltungen. Kondensatoren werden unterteilt in Elektrolytkondensatoren, Festkörperkondensatoren und weitere.

Nach Materialeigenschaften können Kondensatoren unterteilt werden in: Aluminium-Elektrolytkondensatoren, Folienkondensatoren, Tantalkondensatoren, Keramikkondensatoren, Superkondensatoren usw.

In diesem Kit werden Keramikkondensatoren und Elektrolytkondensatoren verwendet.

- Keramikkondensator – Wikipedia
- Elektrolytkondensator – Wikipedia

Auf den Keramikkondensatoren befinden sich Bezeichnungen wie 103 oder 104, die den Kapazitätswert darstellen, 103=10x10^3pF, 104=10x10^4pF

Einheitenumrechnung

1F=10^3mF=10^6uF=10^9nF=10^12pF

5.6 Überbrückungsdrähte

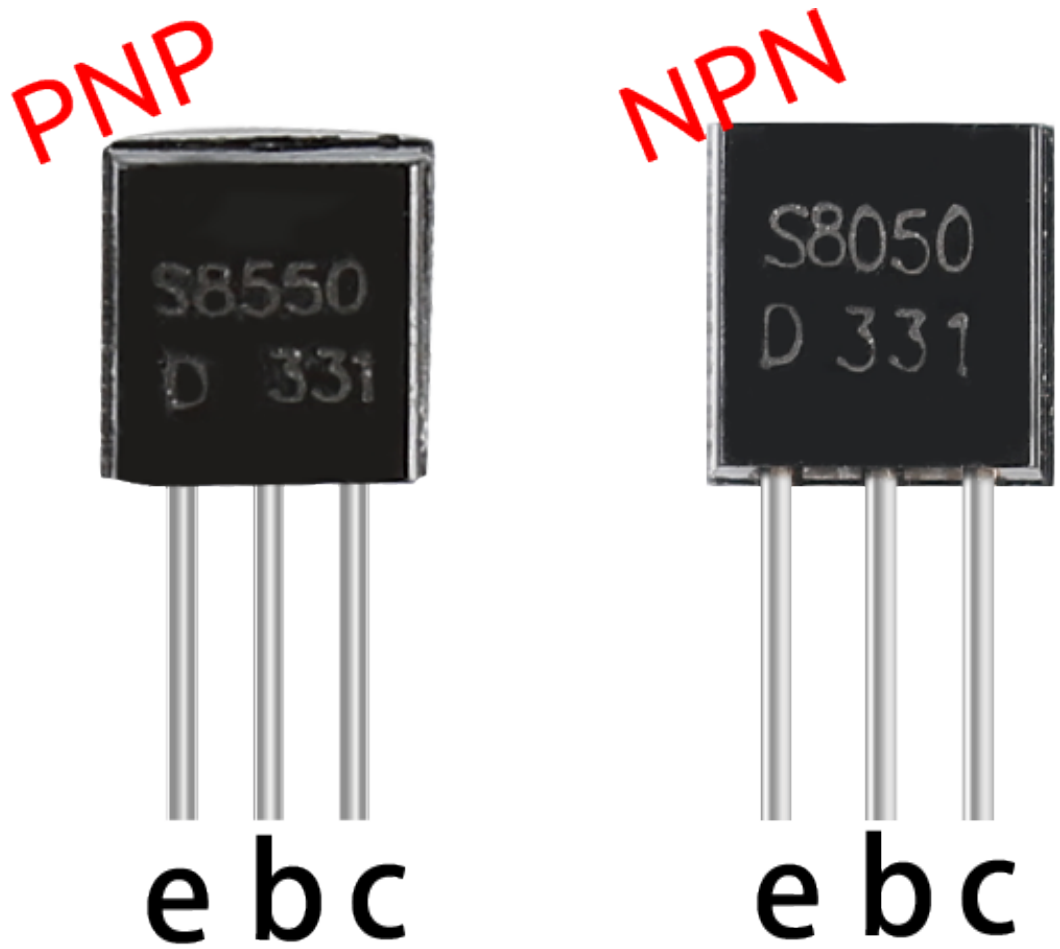
Drähte, die zwei Klemmen miteinander verbinden, werden als Schaltdrähte bezeichnet. Es gibt verschiedene Arten von Überbrückungsdrähten. Hier konzentrieren wir uns auf diejenigen, die in Breadboard. Sie werden unter anderem verwendet, um elektrische Signale zu übertragen von einer beliebigen Stelle auf dem Breadboard zu den Eingangs-/Ausgangspins eines Mikrocontrollers zu übertragen.

Überbrückungsdrähte werden durch Einsetzen ihrer „Endstecker“ in die Schlitze Steckplätze der Lochrasterplatine gesteckt, unter deren Oberfläche sich einige Sätze parallele Platten, die die Steckplätze je nach Fläche in Gruppen von Reihen oder Spalten verbinden je nach Bereich. Die „Endstecker“ werden ohne Löten in die in die Lochrasterplatine eingefügt, und zwar ohne zu löten, und zwar in den Schlitzen, die Prototypen angeschlossen werden müssen.

Es gibt drei Arten von Überbrückungsdraht: Buchse-zu-Buchse, Stecker-zu-Stecker, und Männlich-auf-Weiblich. Der Grund, warum wir sie Male-to-Female nennen, ist, dass sie die herausragende Spitze an einem Ende sowie ein versenktes weibliches Ende hat. Male-to-Male bedeutet, dass beide Seiten männlich sind und Female-to-Female bedeutet, dass beide Enden sind weiblich.

In einem Projekt kann mehr als ein Typ von ihnen verwendet werden. Die Farbe der Sprungdrähte ist unterschiedlich, aber das bedeutet nicht, dass ihre Funktion unterschiedlich ist. Funktion unterschiedlich ist; sie dient lediglich der besseren Identifizierung der Verbindung zwischen den einzelnen Schaltkreisen.

5.7 Transistor



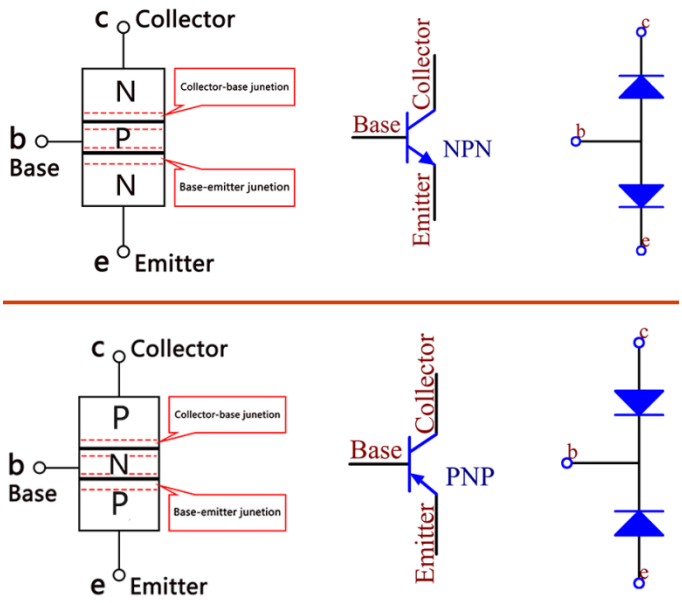
Ein Transistor ist ein Halbleitergerät, das den Stromfluss durch einen Strom steuert. Er funktioniert, indem er schwache Signale zu Signalen mit größerer Amplitude verstärkt und wird auch als kontaktloser Schalter eingesetzt.

Ein Transistor besteht aus einer dreischichtigen Struktur aus P-Typ- und N-Typ-Halbleitern. Diese bilden intern drei Regionen. Die mittlere, dünnere Region ist die Basisregion; die anderen beiden sind entweder N-Typ oder P-Typ - die kleinere Region mit intensiven Majoritätsträgern ist die Emitterregion, während die andere die Kollektorregion ist. Diese Zusammensetzung ermöglicht es dem Transistor, als Verstärker zu fungieren. Aus diesen drei Regionen entstehen jeweils drei Pole, die Basis (b), Emitter (e) und Kollektor (c). Sie bilden zwei P-N-Übergänge, nämlich den Emitterübergang und den Kollektorübergang. Die Richtung des Pfeils im Transistorschaltsymbol gibt die des Emitterübergangs an.

- P-N-Übergang - Wikipedia

Basierend auf dem Halbleitertyp können Transistoren in zwei Gruppen eingeteilt werden, in NPN- und PNP-Transistoren. Aus der Abkürzung können wir ableiten, dass der Erstere aus zwei N-Typ-Halbleitern und einem P-Typ besteht und der Letztere das Gegenteil ist. Siehe die Abbildung unten.

Bemerkung: Der s8550 ist ein PNP-Transistor und der s8050 ein NPN-Transistor. Sie sehen sehr ähnlich aus, daher müssen wir sorgfältig ihre Etiketten prüfen.



Wenn ein High-Level-Signal durch einen NPN-Transistor fließt, wird dieser aktiviert. Ein PNP-Transistor benötigt jedoch ein Low-Level-Signal zur Steuerung. Beide Transistortypen werden häufig für kontaktlose Schalter eingesetzt, wie in diesem Experiment.

- S8050 Transistor-Datenblatt
- S8550 Transistor-Datenblatt

Richten Sie die Beschriftungsseite zu sich und die Pins nach unten. Die Pins von links nach rechts sind Emitter (e), Basis (b) und Kollektor (c).

Bemerkung:

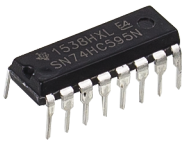
- Die Basis ist das Steuerelement für die größere Stromversorgung.
- Beim NPN-Transistor ist der Kollektor die größere Stromquelle und der Emitter der Ausgang für diese Versorgung, beim PNP-Transistor ist es genau umgekehrt.

Beispiele

- *5.6 Zwei Arten von Transistoren* (Arduino-Projekt)
- *3.1 Piepser* (Arduino-Projekt)
- *6.1 Fruchtpiano* (Arduino-Projekt)
- *5.6 Zwei Arten von Transistoren* (MicroPython-Projekt)
- *3.2 Eigener Klang* (MicroPython-Projekt)
- *6.3 Licht-Theremin* (MicroPython-Projekt)

Chip

5.8 74HC595



Sind Sie jemals in der Situation gewesen, eine Vielzahl von LEDs steuern zu wollen, oder benötigten Sie einfach mehr I/O-Pins, um Tasten, Sensoren und Servos gleichzeitig zu kontrollieren? Sicherlich können Sie einige Sensoren an Arduino-Pins anschließen, doch bald werden Ihnen die Pins am Arduino ausgehen.

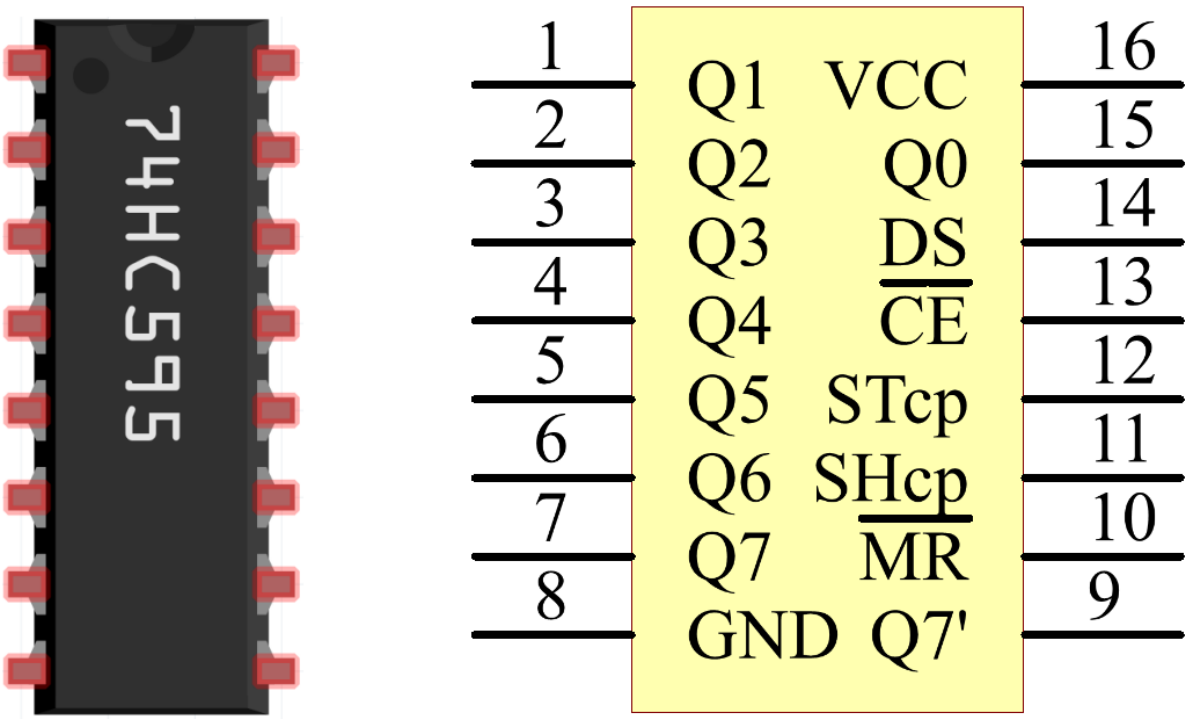
Die Lösung hierfür sind „Schieberegister“. Schieberegister ermöglichen es Ihnen, die Anzahl der I/O-Pins, die Sie vom Arduino (oder jedem Mikrocontroller) verwenden können, zu erweitern. Der 74HC595-Schieberegister ist einer der bekanntesten.

Der 74HC595 steuert im Wesentlichen acht unabhängige Ausgangspins und verwendet dabei nur drei Eingangspins. Wenn Sie mehr als acht zusätzliche I/O-Leitungen benötigen, können Sie problemlos beliebig viele Schieberegister kaskadieren und so eine große Anzahl an I/O-Leitungen schaffen. All dies wird durch das sogenannte Verschieben erreicht.

Merkmale

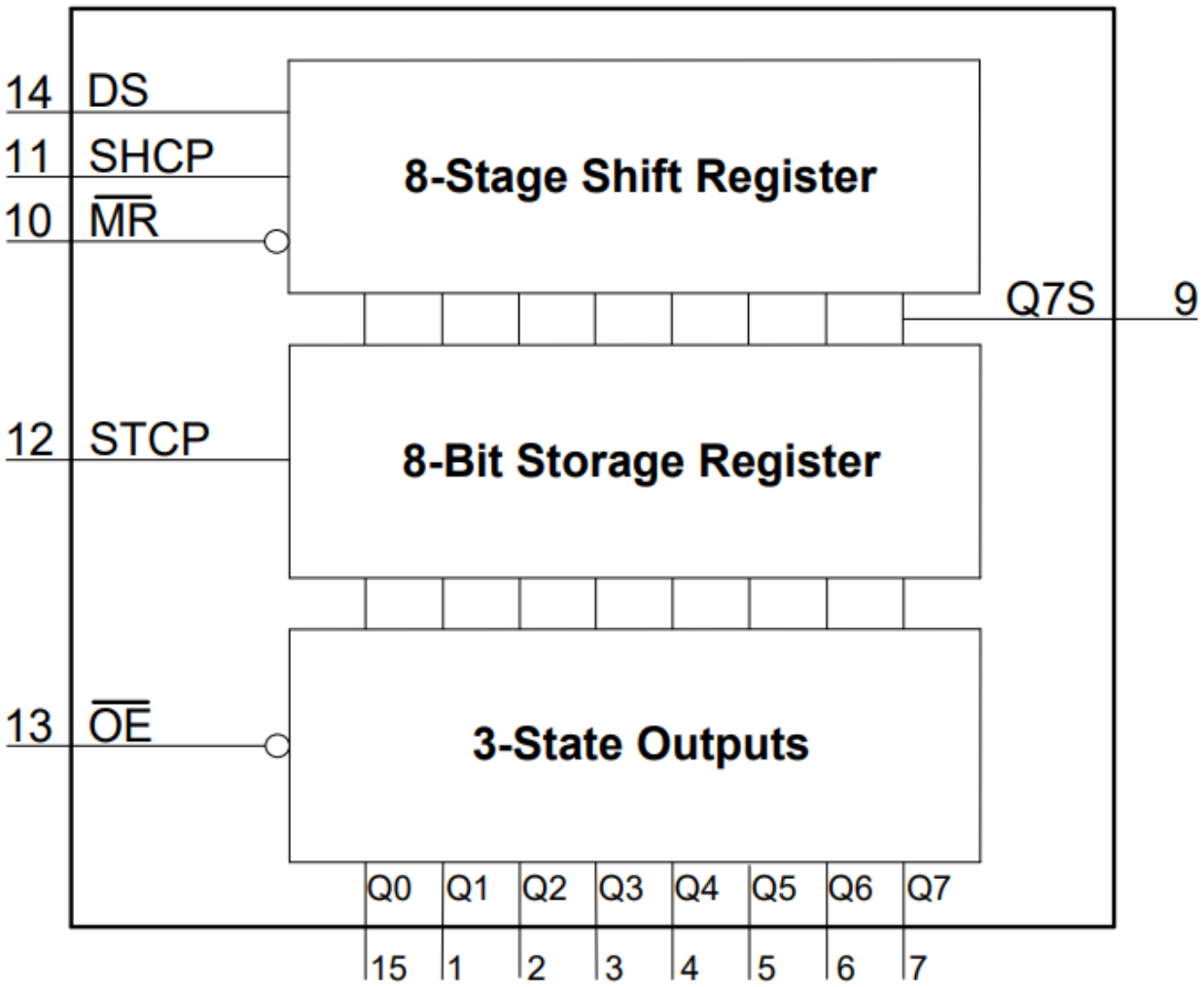
- 8-Bit serieller Eingang, paralleler Ausgang
- Breiter Betriebsspannungsbereich von 2 V bis 6 V
- Hochstrom-3-State-Ausgänge können bis zu 15 LSTTL-Lasten treiben
- Geringer Stromverbrauch, maximal 80 µA ICC
- Typische tPD = 14 ns
- ±6-mA-Ausgangstreiber bei 5 V
- Geringer Eingangsstrom von maximal 1 µA
- Schieberegister mit direkter Rückstellung

Pins des 74HC595 und ihre Funktionen:



- **Q0-Q7**: 8-Bit parallele Datenausgangspins, geeignet zur Steuerung von 8 LEDs oder 8 Pins eines 7-Segment-Displays direkt.
- **Q7'**: Serieller Ausgangspin, verbunden mit dem DS eines weiteren 74HC595, um mehrere 74HC595 in Serie zu schalten
- **MR**: Rücksetzpin, aktiv bei niedrigem Pegel;
- **SHcp**: Zeitsequenzeingang des Schieberegisters. Bei der steigenden Flanke bewegen sich die Daten im Schieberegister sukzessive um ein Bit, d. h., die Daten in Q1 gehen zu Q2 usw. Bei der fallenden Flanke bleiben die Daten im Schieberegister unverändert.
- **STcp**: Zeitsequenzeingang des Speicherregisters. Bei der steigenden Flanke bewegen sich die Daten aus dem Schieberegister in das Speicherregister.
- **OE**: Ausgangsaktivierungspin, aktiv bei niedrigem Pegel.
- **DS**: Serieller Dateneingangspin
- **VCC**: Positive Versorgungsspannung.
- **GND**: Erde.

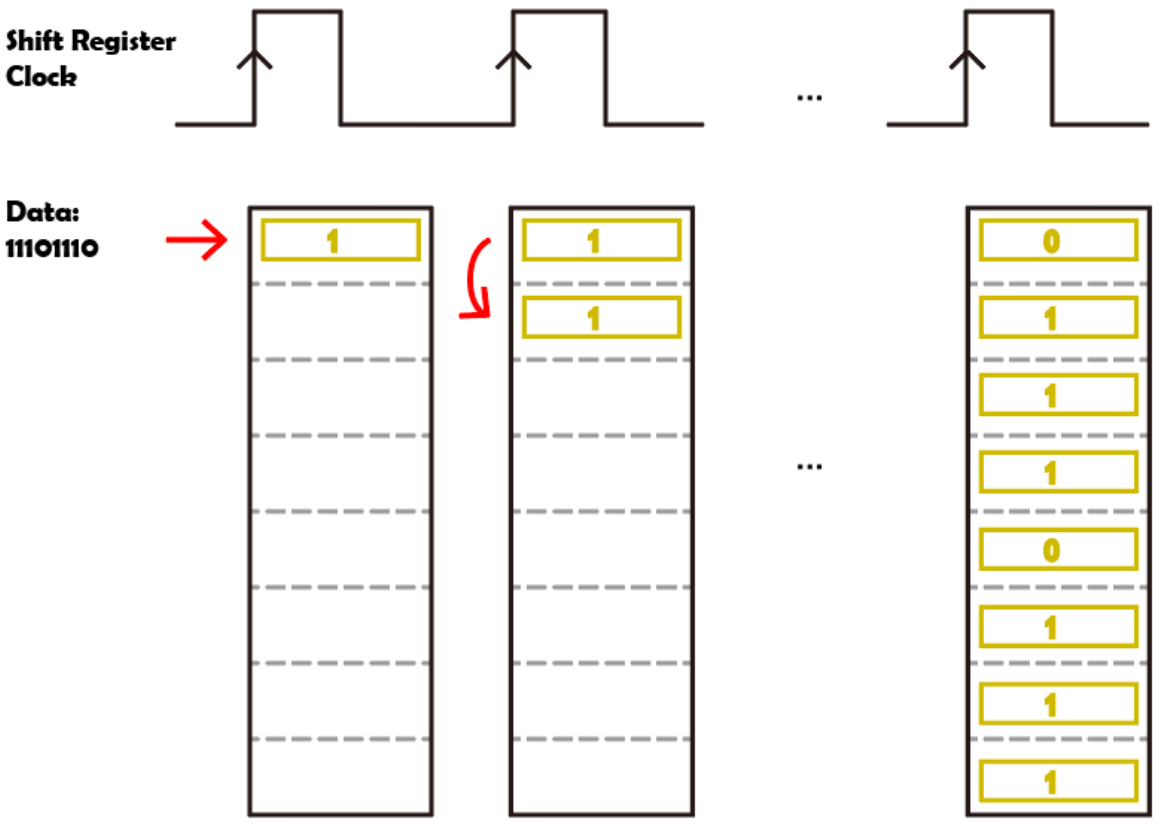
Funktionsdiagramm



Arbeitsprinzip

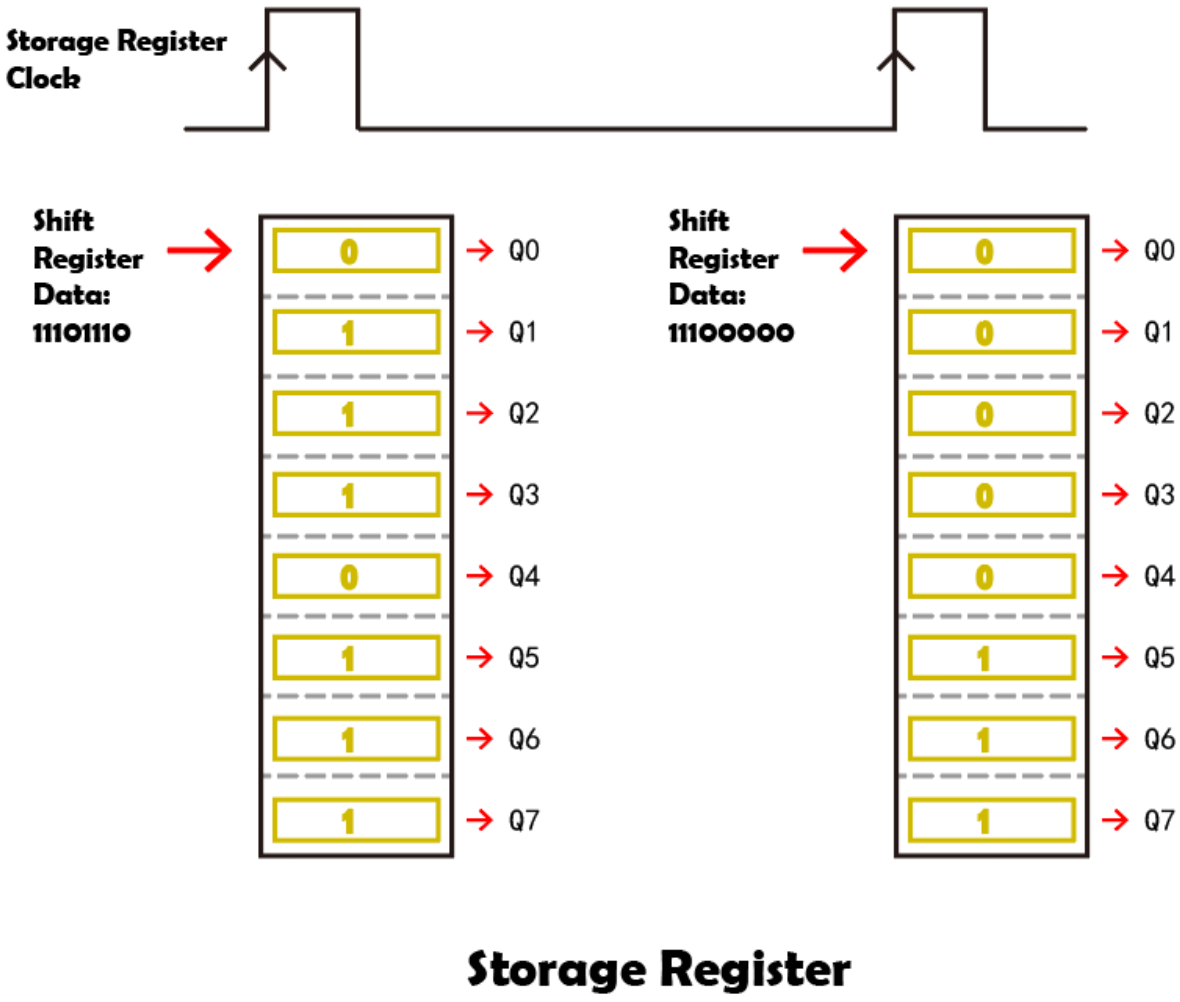
Wenn MR (Pin 10) auf hohem Pegel und OE (Pin 13) auf niedrigem Pegel ist, werden die Daten bei der steigenden Flanke von SHcp eingegeben und gelangen durch die steigende Flanke von STcp ins Speicherregister.

- Schieberegister
  - Angenommen, wir möchten die Binärdaten 1110 1110 in das Schieberegister des 74HC595 eingeben.
  - Die Daten werden ab Bit 0 des Schieberegisters eingegeben.
  - Bei jeder steigenden Flanke des Schieberegister-Takts werden die Bits im Schieberegister um einen Schritt verschoben. Zum Beispiel übernimmt Bit 7 den vorherigen Wert in Bit 6, Bit 6 bekommt den Wert von Bit 5 usw.



Shift Register

- Speicherregister
  - Wenn das Speicherregister im Zustand der steigenden Flanke ist, werden die Daten des Schieberegisters in das Speicherregister übertragen.
  - Das Speicherregister ist direkt mit den 8 Ausgangspins verbunden, Q0 bis Q7 können ein Byte Daten empfangen.
  - Das sogenannte Speicherregister bedeutet, dass die Daten in diesem Register bestehen und nicht mit einem Ausgang verschwinden.
  - Die Daten bleiben gültig und unverändert, solange der 74HC595 kontinuierlich mit Strom versorgt wird.
  - Wenn neue Daten ankommen, werden die Daten im Speicherregister überschrieben und aktualisiert.



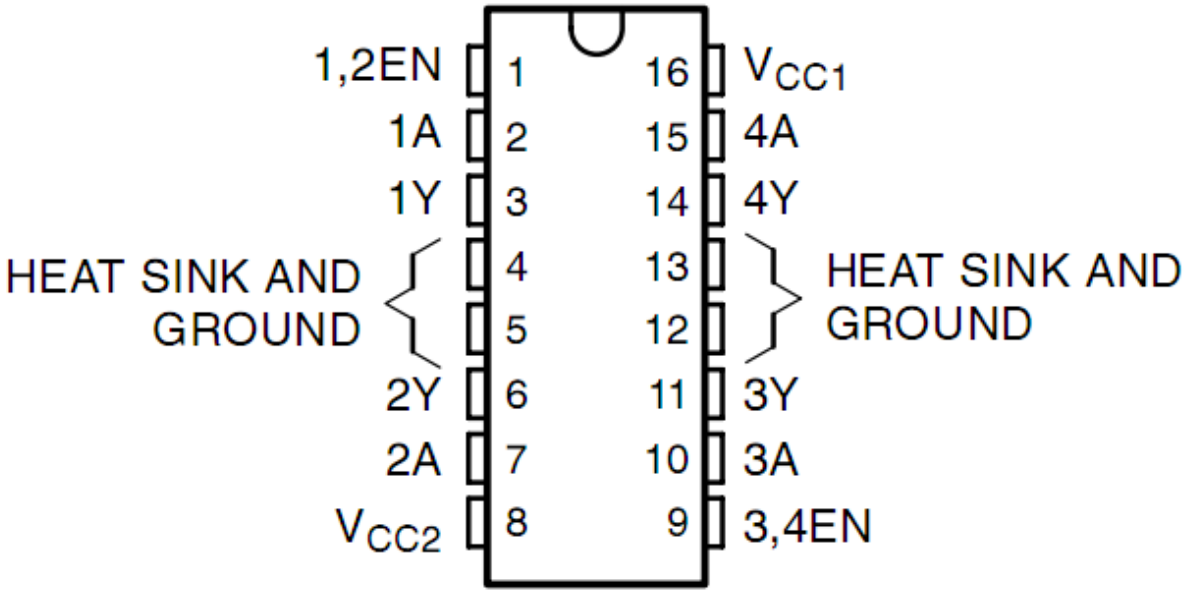
Beispiele

- *2.4 Microchip - 74HC595* (Arduino-Projekt)
- *2.5 7-Segment-Anzeige* (Arduino-Projekt)
- *6.4 Digitaler Würfel* (Arduino-Projekt)
- *2.4 Mikrochip - 74HC595* (MicroPython-Projekt)
- *2.5 Ziffernanzeige* (MicroPython-Projekt)
- *6.6 Digitaler Würfel* (MicroPython-Projekt)

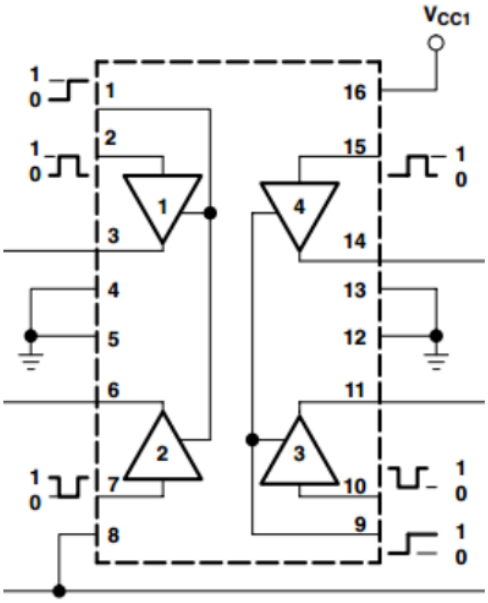
5.9 L293D

Der L293D ist ein integrierter 4-Kanal-Motortreiber-Chip mit hoher Spannung und hohem Strom. Er ist dafür konzipiert, an Standard-DTL, TTL-Logikpegel anzuschließen und induktive Lasten (wie Relaisspulen, Gleichstrom-, Schrittmotoren) sowie Leistungsschalttransistoren usw. zu steuern. Gleichstrommotoren sind Geräte, die Gleichstromenergie in mechanische Energie umwandeln. Aufgrund ihrer überlegenen Drehzahlregelungsleistung werden sie häufig in elektrischen Antrieben eingesetzt.

Siehe die folgende Abbildung der Pins. Der L293D hat zwei Pins (Vcc1 und Vcc2) für die Stromversorgung. Vcc2 dient zur Stromversorgung des Motors, während Vcc1 den Chip versorgt. Da hier ein kleiner Gleichstrommotor verwendet wird, verbinden Sie beide Pins mit +5V.



Folgend sehen Sie die interne Struktur des L293D. Pin EN ist ein Aktivierungspin und funktioniert nur auf hohem Niveau; A steht für Eingang und Y für Ausgang. Die Beziehung zwischen ihnen können Sie unten rechts sehen. Wenn Pin EN auf hohem Niveau ist, gibt Y bei hohem A ein hohes Signal aus und bei niedrigem A ein niedriges Signal. Wenn Pin EN auf niedrigem Niveau ist, funktioniert der L293D nicht.



INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)

- L293D Datenblatt

Beispiele

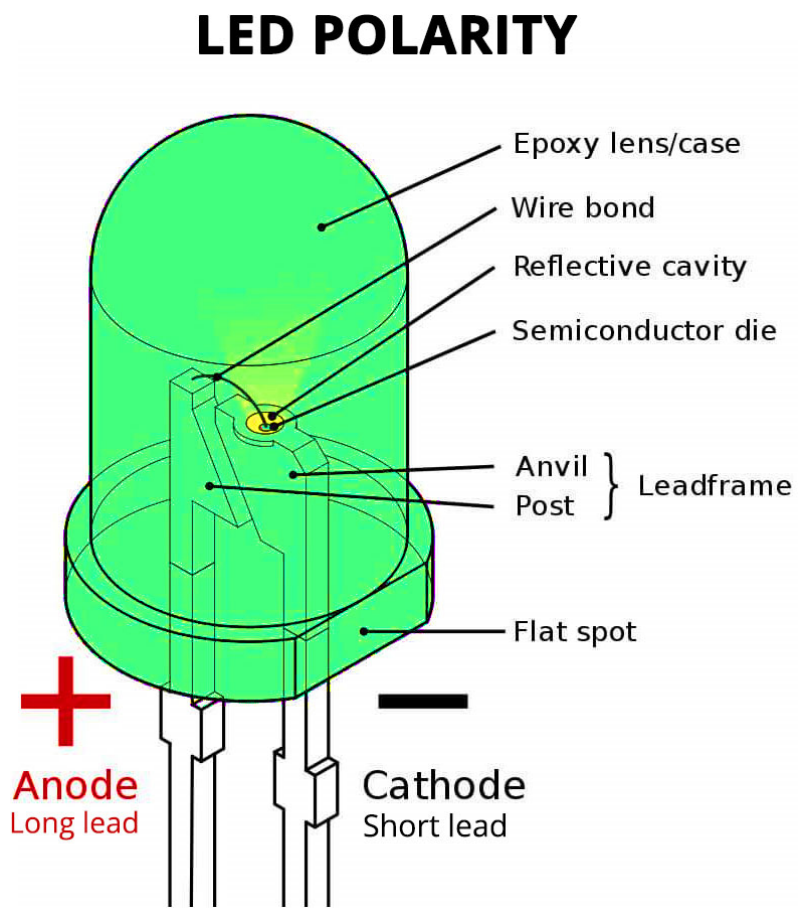
- *4.1 Motor* (Arduino-Projekt)
- *4.2 Pumpen* (Arduino-Projekt)
- *4.1 Kleiner Ventilator* (MicroPython-Projekt)
- *4.2 Pumpen* (MicroPython-Projekt)
- *2.9 Rotierender Ventilator* (Scratch-Projekt)

Anzeige

5.10 LED

Was ist eine LED?

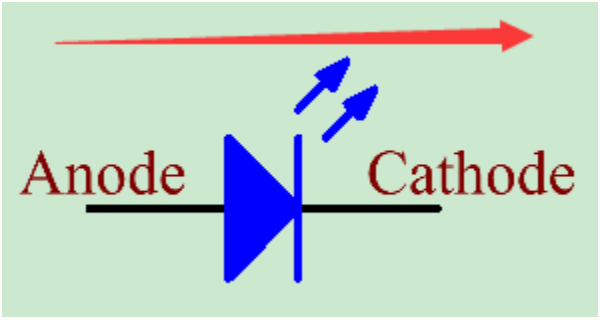

Cathode(-)
Anode(+)
Longer Lead



LEDs sind sehr verbreitete elektronische Bauteile, die häufig zur Dekoration Ihrer Räume während Festivals verwendet werden. Sie können sie auch als Anzeiger für verschiedene Zustände nutzen, zum Beispiel, um zu überprüfen, ob Ihre Haushaltsgeräte eingeschaltet sind oder nicht. LEDs gibt es in vielen verschiedenen Formen und Größen, am verbreitetsten sind Durchsteck-LEDs mit langen Anschlussdrähten, die sich einfach in ein Steckbrett einstecken lassen.

Die vollständige Bezeichnung für LED ist „lichtemittierende Diode“, was bedeutet, dass sie die Eigenschaften einer Diode hat, bei der der Strom in einer Richtung fließt, vom Anoden- (positiv) zum Kathodenanschluss (negativ).

Hier sind die elektrischen Symbole für LEDs.

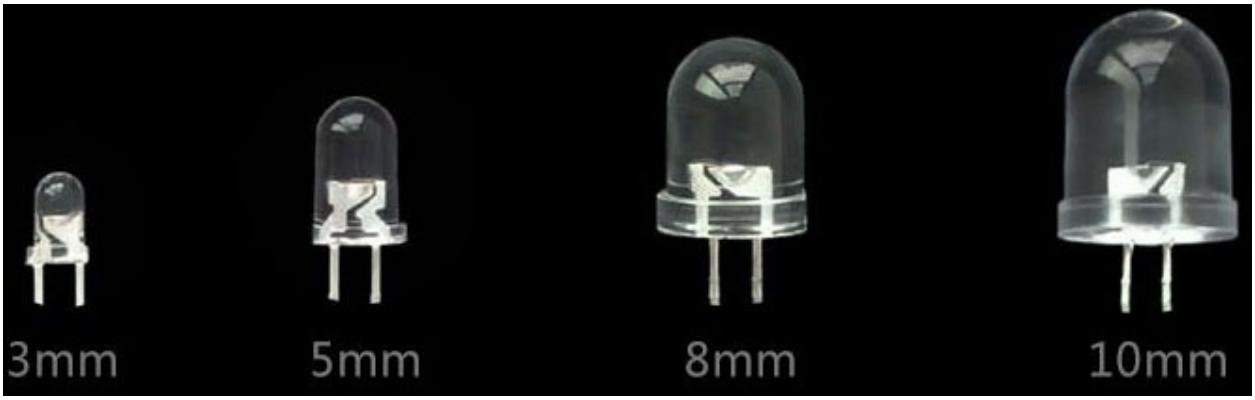


Verschiedene Größen und Farben

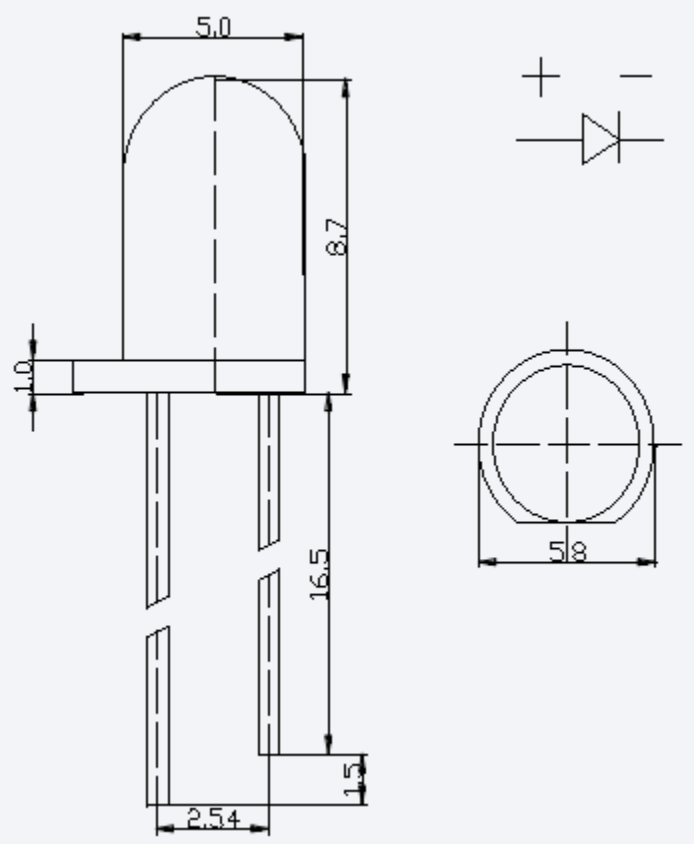
Die gängigsten LED-Farben sind Rot, Gelb, Blau, Grün und Weiß, wobei das emittierte Licht meistens die gleiche Farbe wie das äußere Erscheinungsbild hat.

Seltener verwenden wir LEDs, die transparent oder matt erscheinen, aber das emittierte Licht kann eine andere Farbe als Weiß haben.

LEDs gibt es in vier Größen: 3mm, 5mm, 8mm und 10mm, wobei 5mm die gängigste Größe ist.



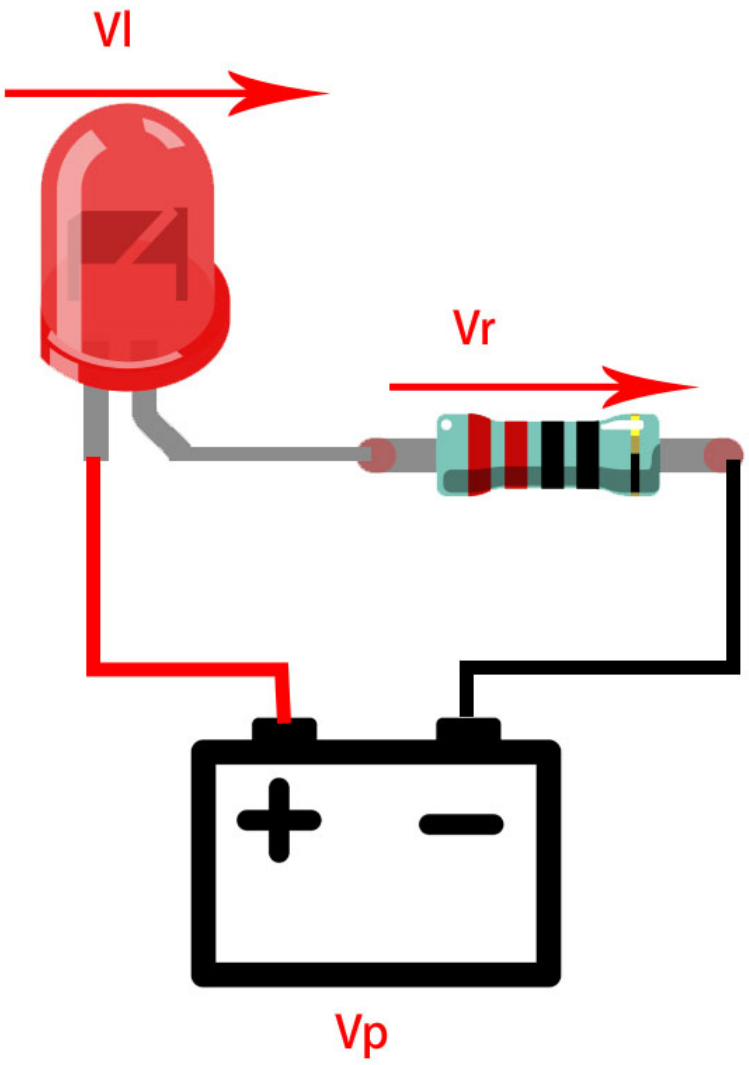
Nachfolgend finden Sie die Größe einer 5mm LED in mm.



Durchlassspannung

Die Durchlassspannung ist ein sehr wichtiger Parameter beim Einsatz von LEDs, da sie bestimmt, wie viel Leistung Sie verwenden und wie groß der vorzuschaltende Strombegrenzungswiderstand sein sollte.

Die Durchlassspannung ist die Spannung, die benötigt wird, damit die LED aufleuchtet. Für die meisten roten, gelben, orangen und hellgrünen LEDs liegt diese Spannung üblicherweise zwischen 1,9V und 2,1V.



Nach dem Ohmschen Gesetz nimmt der Strom in diesem Stromkreis ab, wenn der Widerstand erhöht wird, was dazu führt, dass die LED dunkler leuchtet.

I = (Vp-Vl)/R

Um die LEDs sicher und mit der richtigen Helligkeit zum Leuchten zu bringen, welchen Widerstand sollten wir im Stromkreis verwenden?

Für 99% der 5mm LEDs liegt der empfohlene Strom bei 20mA, wie Sie der Spalte „Bedingungen“ im Datenblatt entnehmen können.

Electrical / Optical Characteristics at TA=25°C

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Forward Voltage	VF	Lf=20mA	1.9		2.1	V
Reverd Current	IR	VR=5V			10	uA
Vtewing Angle	201/2	IF=20mA		20		deg
Lulninous Intensity	ϕv	IF=20mA	3000		4000	mcd
Chromaticity	Tc	IF=20mA	620		625	K
Chromaticity Coordinates	X.Y	IF=20mA	0. 29, 0. 29			

Nun wandeln Sie die obige Formel wie folgt um.

R = (Vp-Vl)/I

Wenn `Vp` 5V, `Vl` (Durchlassspannung) 2V und `I` 20mA beträgt, dann ist `R` 150.

Wir können die LED heller machen, indem wir den Widerstand des Widerstands verringern, aber es wird nicht empfohlen, unter 150 zu gehen (dieser Widerstandswert ist möglicherweise nicht sehr genau, da verschiedene Anbieter unterschiedliche LEDs liefern).

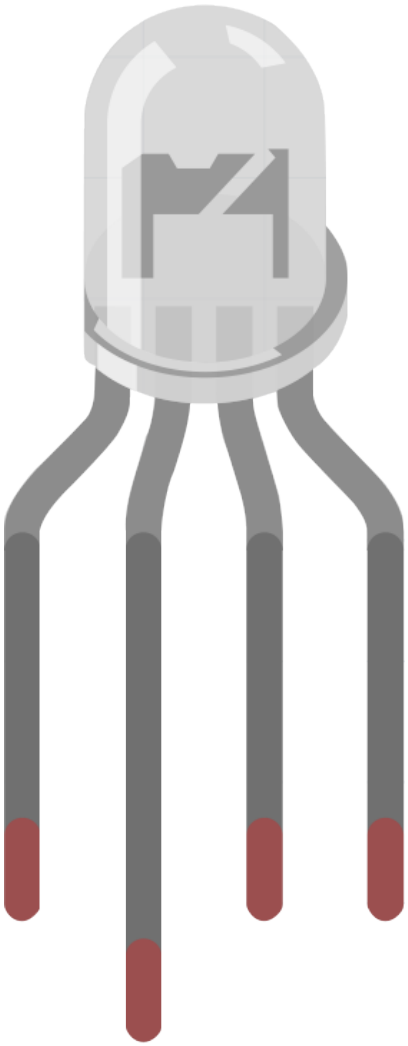
Nachfolgend finden Sie die Durchlassspannungen und Wellenlängen verschiedener farbiger LEDs, die Sie als Referenz verwenden können.

LED-Farbe	Durchlassspannung	Wellenlänge
Rot	1,8V ~ 2,1V	620 ~ 625
Gelb	1,9V ~ 2,2V	580 ~ 590
Grün	1,9V ~ 2,2V	520 ~ 530
Blau	3,0V ~ 3,2V	460 ~ 465
Weiß	3,0V ~ 3,2V	8000 ~ 9000

Beispiel

- *2.1 Hallo, LED!* (Arduino-Projekt)
- *2.2 Verblassen* (Arduino-Projekt)
- *2.1 Hallo, LED!* (MicroPython-Projekt)
- *2.2 Abblendende LED* (MicroPython-Projekt)
- *2.1 Tischlampe* (Scratch-Projekt)
- *2.2 Atmende LED* (Scratch-Projekt)

5.11 RGB LED



RGB-LEDs erzeugen Licht in verschiedenen Farben. Eine RGB-LED kombiniert drei LEDs in den Farben Rot, Grün und Blau in einer durchsichtigen oder halbdurchsichtigen Kunststoffhülle. Durch das Verändern der Eingangsspannung an den drei Pins und deren Überlagerung können verschiedenste Farben erzeugt werden, was laut Statistik bis zu 16.777.216 unterschiedliche Farbtöne ermöglicht.

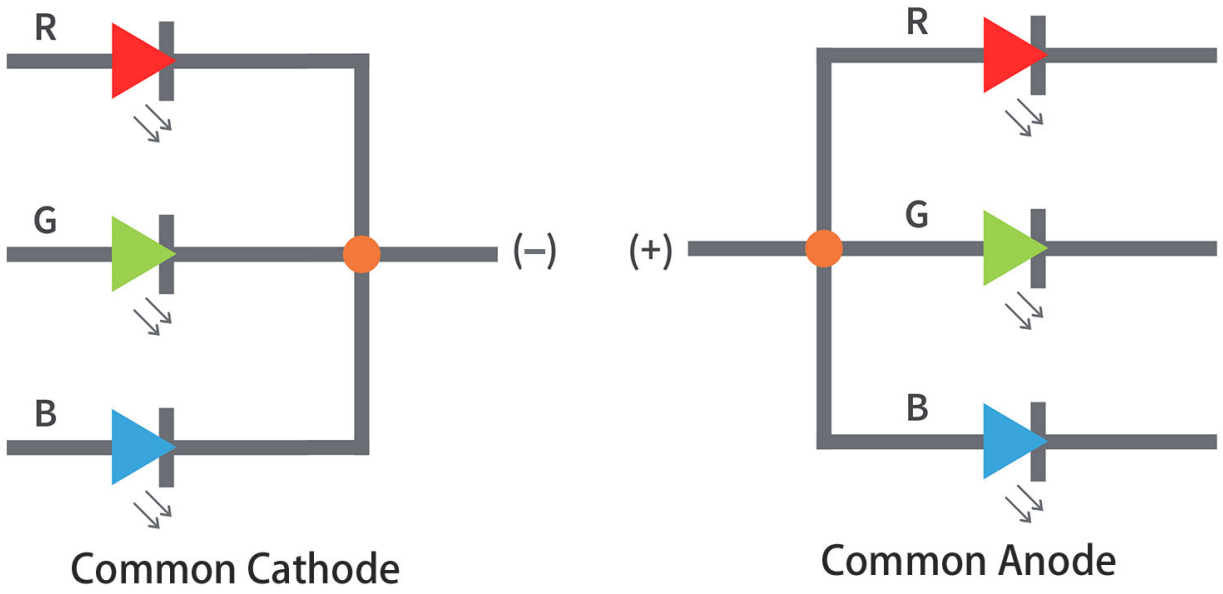
Merkmale

- Farbe: Dreifarbig (Rot/Grün/Blau)
- Gemeinsame Kathode
- 5mm klare, runde Linse
- Vorwärtsspannung: Rot: DC 2,0 - 2,2V; Blau&Grün: DC 3,0 - 3,2V (IF=20mA)
- 0,06 Watt DIP RGB LED
- Luminanz bis zu +20% heller
- Betrachtungswinkel: 30°

Gemeinsame Anode und gemeinsame Kathode

RGB-LEDs können in solche mit gemeinsamer Anode und gemeinsamer Kathode unterteilt werden.

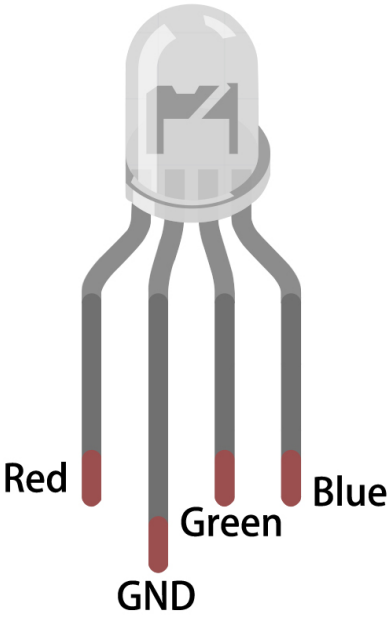
- Bei einer RGB-LED mit gemeinsamer Kathode teilen sich alle drei LEDs eine negative Verbindung (Kathode).
- Bei einer RGB-LED mit gemeinsamer Anode teilen sich alle drei LEDs eine positive Verbindung (Anode).



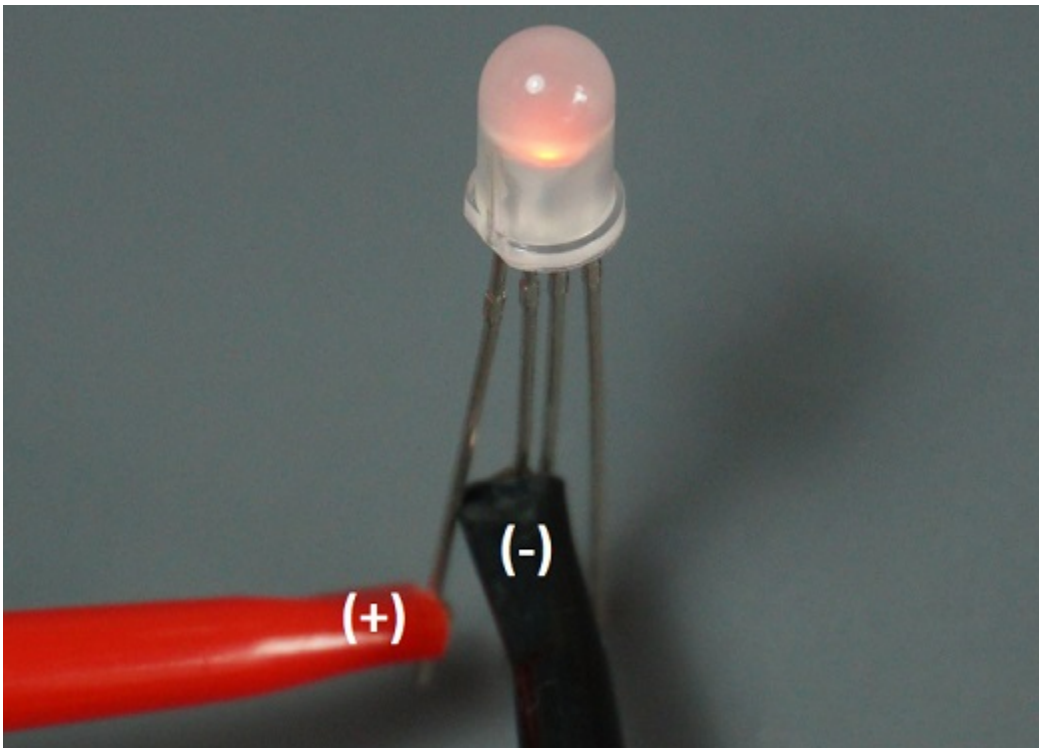
Bemerkung: Wir verwenden die mit gemeinsamer Kathode.

Pins einer RGB-LED

Eine RGB-LED hat 4 Pins: der längste ist GND; die anderen sind Rot, Grün und Blau. Positionieren Sie die RGB-LEDs wie gezeigt, sodass der längste Pin zweiter von links ist. Dann sollten die Pinnummern der RGB-LEDs Rot, GND, Grün und Blau sein.



Sie können auch das Multimeter im Diodentestmodus verwenden und wie unten gezeigt verbinden, um die Farbe jedes Pins zu messen.

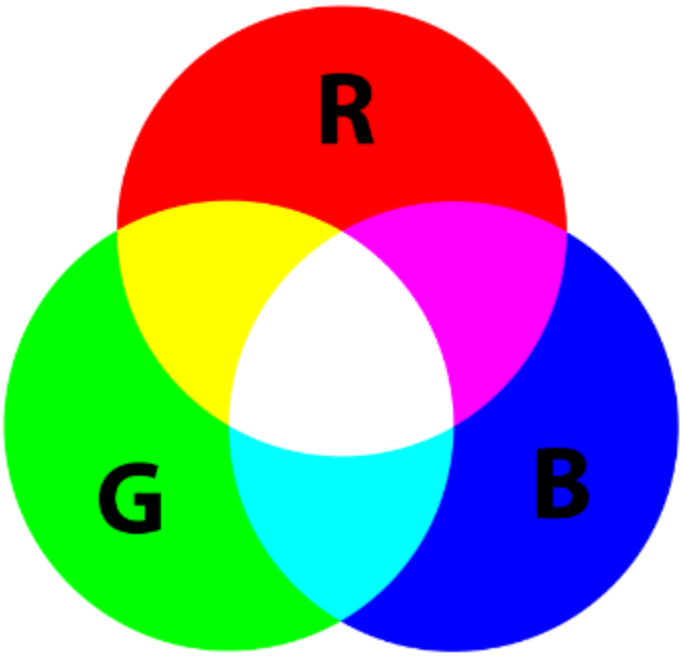


Farben mischen

Um zusätzliche Farben zu erzeugen, können Sie die drei Farben in unterschiedlichen Intensitäten kombinieren. Um die Intensität jeder LED anzupassen, können Sie ein PWM-Signal verwenden.

Da die LEDs so nah beieinander liegen, nehmen unsere Augen das Ergebnis der Farbkombination wahr, anstatt der drei einzelnen Farben.

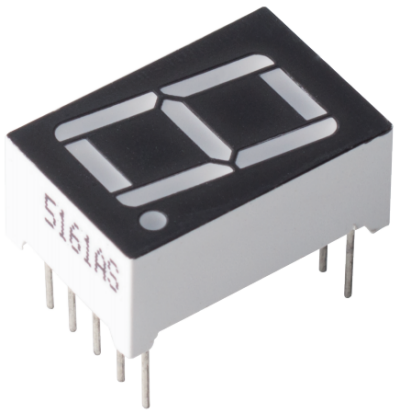
Schauen Sie sich die Tabelle unten an, um zu sehen, wie die Farben kombiniert werden. Sie gibt Ihnen eine Vorstellung davon, wie die Farbmischtabelle funktioniert und wie verschiedene Farben produziert werden.



Beispiele

- *2.3 Bunte Beleuchtung* (Arduino-Projekt)
- *6.5 Farbverlauf* (Arduino-Projekt)
- *2.3 Farbiges Licht* (MicroPython-Projekt)
- *2.3 Farbenfrohe Bälle* (Scratch-Projekt)

5.12 7-Segment-Anzeige



Eine 7-Segment-Anzeige ist eine 8-förmige Komponente, die 7 LEDs umfasst. Jede LED wird als Segment bezeichnet - bei Energiezufuhr bildet ein Segment einen Teil einer anzuzeigenden Ziffer.

- Jede der LEDs in der Anzeige erhält ein positionelles Segment, wobei einer ihrer Anschlussstifte aus dem rechteckigen Kunststoffgehäuse herausgeführt wird.
- Diese LED-Stifte sind von „a“ bis „g“ beschriftet und repräsentieren jede einzelne LED.

- Die anderen LED-Stifte sind miteinander verbunden und bilden einen gemeinsamen Stift.
- Durch das Vorschalten der geeigneten Stifte der LED-Segmente in einer bestimmten Reihenfolge leuchten einige Segmente auf, während andere dunkel bleiben, wodurch der entsprechende Charakter auf der Anzeige dargestellt wird.

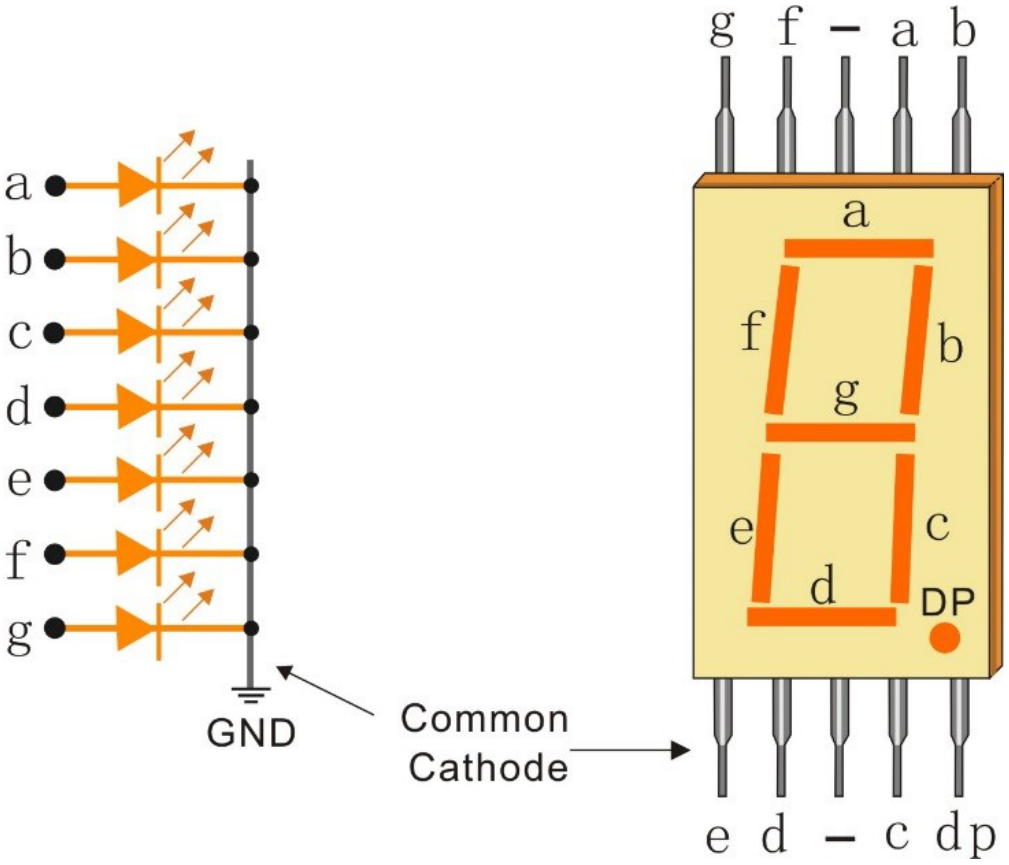
Eigenschaften

- Größe: 19 x 12,7 x 13,8mm(LxBxH, einschließlich der Stifte)
- Bildschirm: 0,56“
- Farbe: rot
- Gemeinsame Kathode
- Vorwärtsspannung: 1,8V
- 10 Stifte
- Abstand: standard 0,1“ (2,54mm)

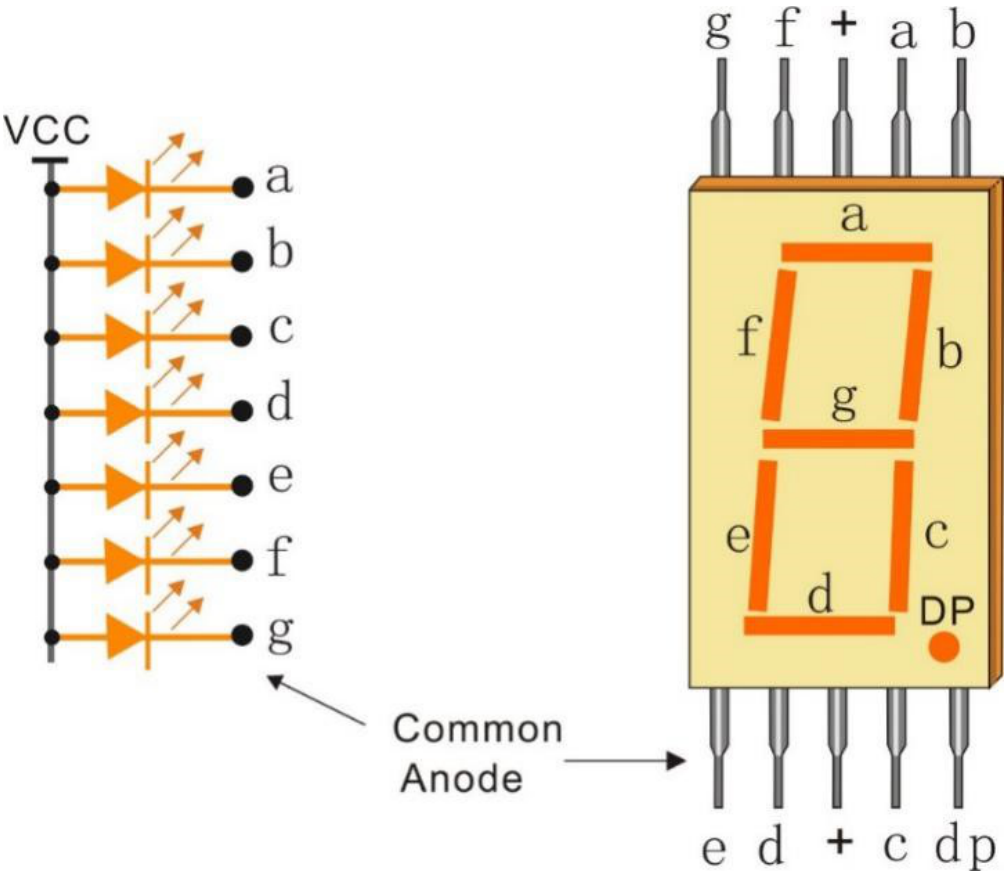
Gemeinsame Kathode (CC) oder Gemeinsame Anode (CA)

Es gibt zwei Arten von Stiftverbindungen: Gemeinsame Kathode (CC) und Gemeinsame Anode (CA). Wie der Name andeutet, hat eine CC-Anzeige alle Kathoden der 7 LEDs verbunden, während eine CA-Anzeige alle Anoden der 7 Segmente verbunden hat.

- Gemeinsame Kathoden 7-Segment-Anzeige

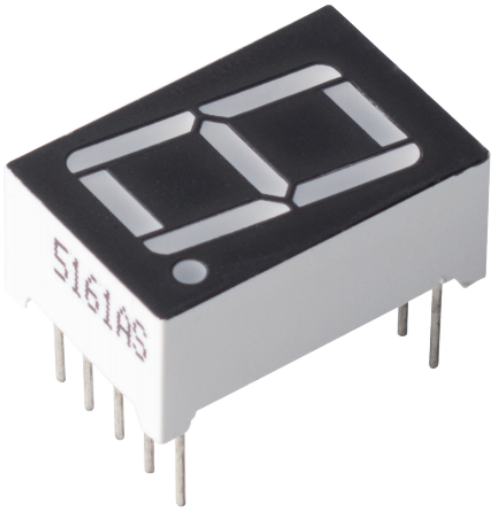


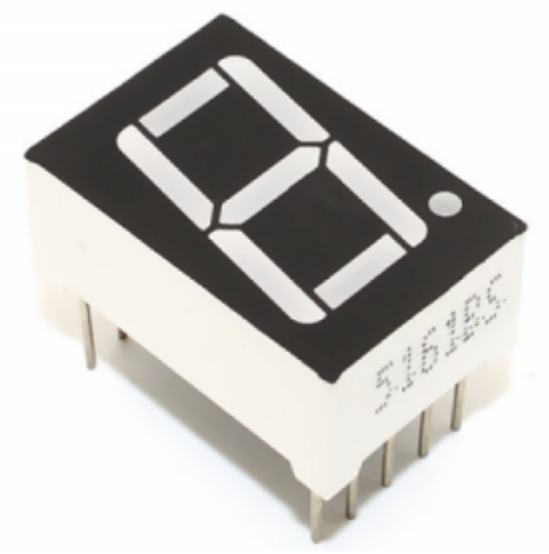
- Gemeinsame Anoden 7-Segment-Anzeige



Wie erkennt man CC oder CA?

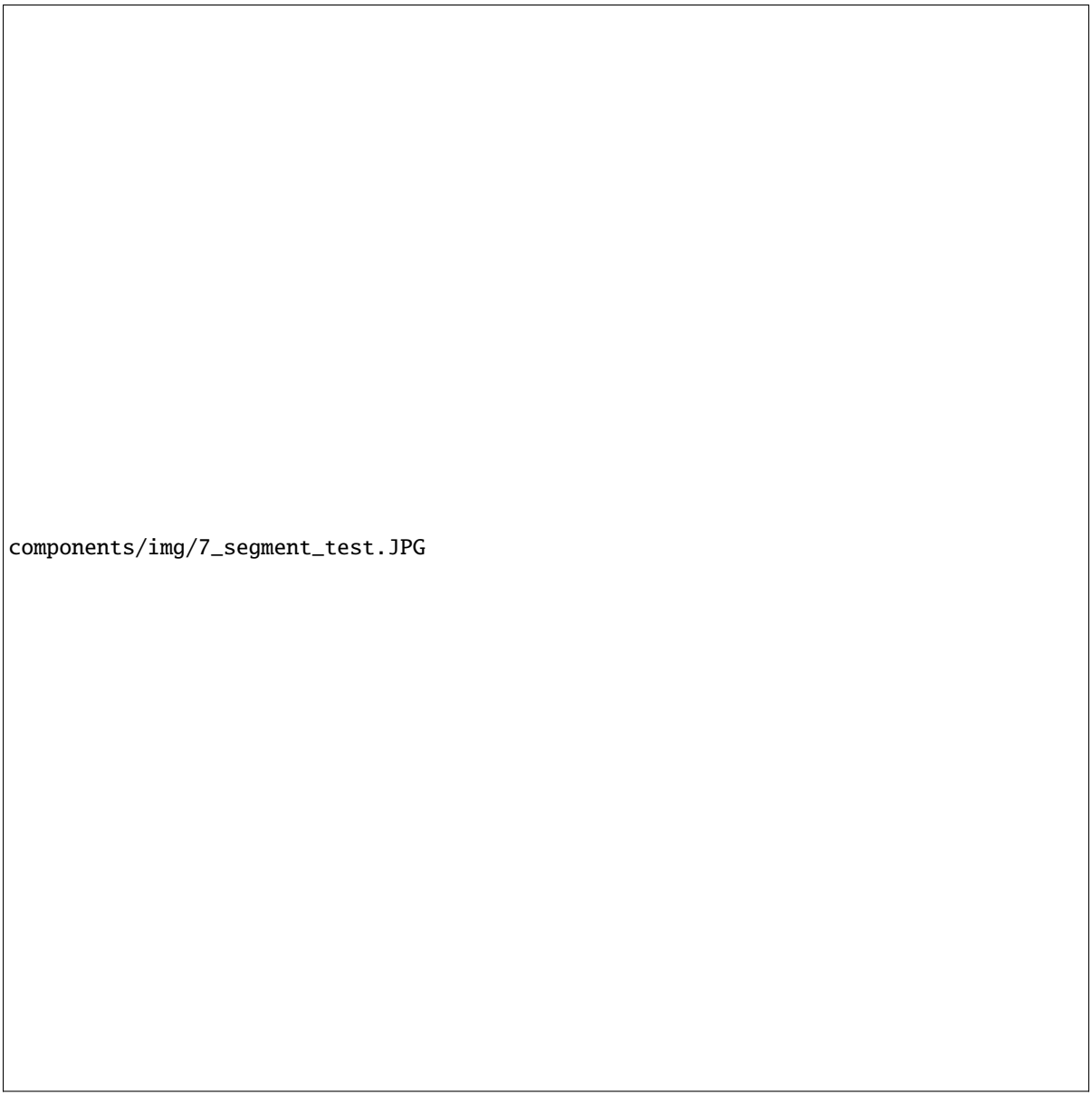
Normalerweise gibt es eine Beschriftung auf der Seite der 7-Segment-Anzeige, xxxAx oder xxxBx. Allgemein steht xxxAx für gemeinsame Kathode und xxxBx für gemeinsame Anode.





Sie können auch ein Multimeter verwenden, um die 7-Segment-Anzeige zu überprüfen, wenn keine Beschriftung vorhanden ist. Stellen Sie das Multimeter auf Diodentestmodus ein und verbinden Sie das schwarze Kabel mit dem mittleren Stift der 7-Segment-Anzeige und das rote Kabel mit einem beliebigen anderen Stift außer dem mittleren. Die 7-Segment-Anzeige ist eine gemeinsame Kathode, wenn ein Segment aufleuchtet.

Wechseln Sie die roten und schwarzen Messköpfe, wenn kein Segment leuchtet. Wenn ein Segment aufleuchtet, deutet dies auf eine gemeinsame Anode hin.

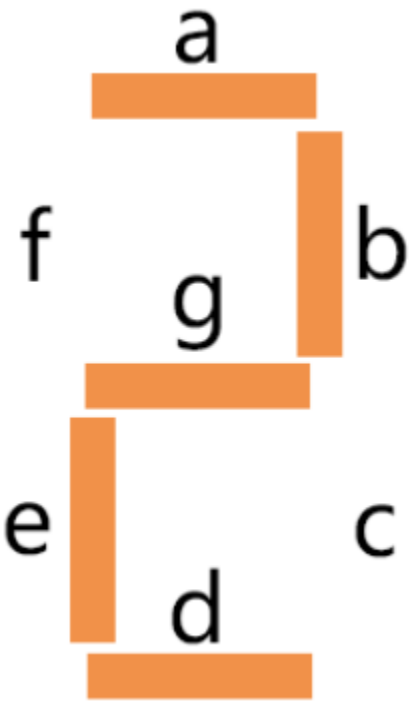


Anzeigecodes

Um Ihnen zu veranschaulichen, wie 7-Segment-Anzeigen (Gemeinsame Kathode) Zahlen darstellen, haben wir die folgende Tabelle erstellt. Die Zahlen sind die Nummern 0-F, die auf der 7-Segment-Anzeige dargestellt werden; (DP) GFEDCBA bezieht sich auf die entsprechende LED, die auf 0 oder 1 gesetzt ist.

Numbers	Common Cathode		Numbers	Common Cathode	
	(DP)GFEDCBA	Hex Code		(DP)GFEDCBA	Hex Code
0	00111111	0x3f	A	01110111	0x77
1	00000110	0x06	B	01111100	0x7c
2	01011011	0x5b	C	00111001	0x39
3	01001111	0x4f	D	01011110	0x5e
4	01100110	0x66	E	01111001	0x79
5	01101101	0x6d	F	01110001	0x71
6	01111101	0x7d			
7	00000111	0x07			
8	01111111	0x7f			
9	01101111	0x6f			

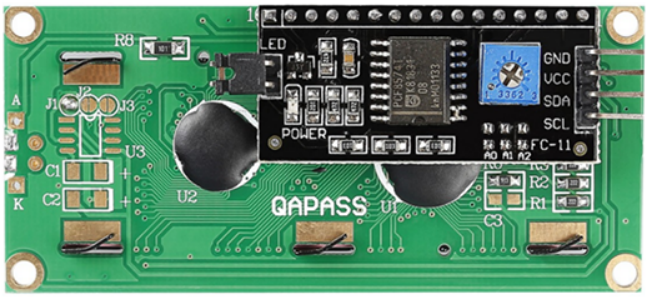
Zum Beispiel bedeutet 01011011, dass DP, F und C auf 0 gesetzt sind, während die anderen auf 1 gesetzt sind. Daher wird die Zahl 2 auf der 7-Segment-Anzeige angezeigt.



Beispiele

- *2.5 7-Segment-Anzeige* (Arduino-Projekt)
- *6.4 Digitaler Würfel* (Arduino-Projekt)
- *2.5 Ziffernanzeige* (MicroPython-Projekt)
- *6.6 Digitaler Würfel* (MicroPython-Projekt)

5.13 I2C LCD1602



- **GND**: Masse
- **VCC**: Spannungsversorgung, 5V.
- **SDA**: Serielle Datenleitung. Mit einem Pullup-Widerstand an VCC anschließen.
- **SCL**: Serielle Taktleitung. Mit einem Pullup-Widerstand an VCC anschließen.

Wie allgemein bekannt, bereichern LCDs und andere Anzeigen zwar die Mensch-Maschine-Interaktion, teilen aber eine gemeinsame Schwäche. Wenn sie an einen Controller angeschlossen werden, belegen sie mehrere IOs des Controllers, der nicht so viele externe Ports hat. Dies beschränkt auch andere Funktionen des Controllers.

Deshalb wurde das LCD1602 mit einem I2C-Modul entwickelt, um dieses Problem zu lösen. Das I2C-Modul hat einen eingebauten PCF8574 I2C-Chip, der die I2C-Serien-Daten in parallele Daten für das LCD-Display umwandelt.

- PCF8574 Datenblatt

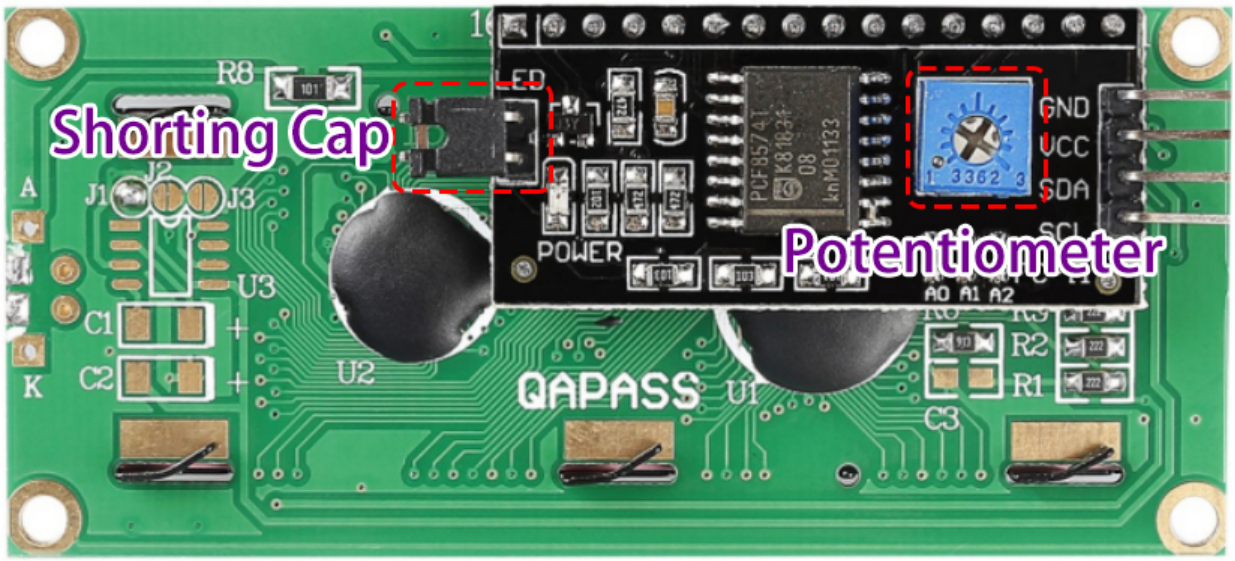
I2C-Adresse

Die Standardadresse ist grundsätzlich 0x27, in einigen Fällen kann sie 0x3F sein.

Am Beispiel der Standardadresse 0x27 kann die Geräteadresse geändert werden, indem die A0/A1/A2-Pads kurzgeschlossen werden; im Standardzustand ist A0/A1/A2 1, und wenn das Pad kurzgeschlossen wird, ist A0/A1/A2 0.

Hintergrundbeleuchtung/Kontrast

Die Hintergrundbeleuchtung kann durch einen Jumper aktiviert werden, entfernen Sie den Jumper, um die Hintergrundbeleuchtung zu deaktivieren. Das blaue Potentiometer auf der Rückseite dient zur Einstellung des Kontrasts (das Verhältnis der Helligkeit zwischen dem hellsten Weiß und dem dunkelsten Schwarz).

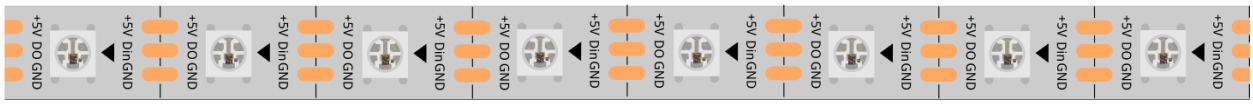


- **Kurzschlusskappe**: Die Hintergrundbeleuchtung kann mit dieser Kappe aktiviert werden, entfernen Sie diese Kappe, um die Hintergrundbeleuchtung zu deaktivieren.
- **Potentiometer**: Es wird verwendet, um den Kontrast (die Klarheit des angezeigten Textes) einzustellen, der im Uhrzeigersinn erhöht und im Gegenuhrzeigersinn verringert wird.

Beispiele

- *2.6 Zeichenanzeige* (Arduino-Projekt)
- *6.7 Zahlenraten* (Arduino-Projekt)
- *2.6 Zeichenanzeige* (MicroPython-Projekt)
- *6.7 Zahlenraten* (MicroPython-Projekt)

5.14 WS2812 RGB 8 LEDs Leiste



Der WS2812 RGB 8 LEDs Strip besteht aus 8 RGB-LEDs. Es wird nur ein Pin benötigt, um alle LEDs zu steuern. Jede RGB-LED hat einen WS2812-Chip, der unabhängig gesteuert werden kann. Er kann eine 256-stufige Helligkeitsanzeige und eine vollständige Echtfarbanzeige von 16.777.216 Farben realisieren. Gleichzeitig enthält das Pixel einen intelligenten digitalen Schnittstellen-Datenlatch-Signalformungs-Verstärker-Treiberschaltkreis, und eine Signalformungsschaltung ist eingebaut, um die Farbhöhe des Pixelpunktlichts effektiv zu gewährleisten Konsistent.

Es ist flexibel, kann nach Belieben angedockt, gebogen und geschnitten werden, und die Rückseite ist mit Klebeband ausgestattet, das auf der unebenen Oberfläche nach Belieben befestigt werden kann, und kann in einem engen Raum installiert werden.

Merkmale

- Arbeitsspannung: DC5V
- IC: Ein IC steuert eine RGB-LED
- Verbrauch: 0.3w jede LED

- Arbeitstemperatur: -15-50
- Farbe: Vollfarbe RGB
- RGB-Typ5050RGBBuilt-in IC WS2812B
- Dicke des Lichtstreifens: 2mm
- Jede LED kann einzeln gesteuert werden

WS2812B Einführung

- WS2812B Datenblatt

WS2812B ist eine intelligente Steuer-LED-Lichtquelle, bei der die Steuerschaltung und der RGB-Chip in einem einem Paket von 5050 Komponenten integriert sind. Intern sind ein intelligenter digitaler Port-Datenspeicher und ein Signalumformungs-Verstärker fikationsschaltung. Außerdem sind ein interner Präzisionsoszillator und ein programmierbarer 12V-Spannungskonstanthalter enthalten. e-nt Steuerteil, effektiv sicherzustellen, die Pixel Punkt Licht Farbhöhe konsistent.

Das Datenübertragungsprotokoll verwendet den einfachen NZR-Kommunikationsmodus. Nach dem Einschalt-Reset des Pixels empfängt der DIN Port die Daten vom Controller, das erste Pixel sammelt die ersten 24-Bit-Daten und sendet sie an den internen Daten-Latch, die anderen Daten, die durch den internen Signalumformungs-Verstärkungsschaltkreis umgestaltet werden, werden über den DO-Port an das nächste Kaskadenpixel gesendet. Pixel durch den DO-Port. Nach der Übertragung für jedes Pixel, das Signal zu reduzieren 24bit. Pixel verabschieden Auto resha Übertragungstechnik, so dass die Pixel-Kaskade Zahl ist nicht begrenzt die Signalübertragung, nur abhängig von der Geschwindigkeit der Signalübertragung ab.

LED mit niedriger Betriebsspannung, Umweltschutz und Energieeinsparung, hohe Helligkeit, großer Streuwinkel e ist groß, gute Konsistenz, geringe Leistung, lange Lebensdauer und andere Vorteile. Der in die LED integrierte Steuerchip oben immer mehr einfache Schaltung, kleines Volumen, bequeme Installation.

Beispiel

- *2.7 RGB-LED-Streifen* (Arduino-Projekt)
- *6.2 Fließendes Licht* (Arduino-Projekt)
- *2.7 RGB-LED-Streifen* (MicroPython-Projekt)
- *6.2 Fließendes Licht* (MicroPython-Projekt)
- *6.5 Farbverlauf* (MicroPython-Projekt)

Klang

5.15 Summer

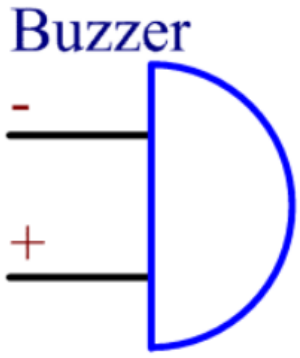
Als eine Art elektronischer Summer mit integrierter Struktur werden Summer, die mit Gleichstrom betrieben werden, häufig in Computern, Druckern, Kopierern, Alarmen, elektronischen Spielzeugen, automobilen Elektronikgeräten, Telefonen, Zeitgebern und anderen elektronischen Produkten oder Sprachgeräten eingesetzt.

Summer können in aktive und passive Summer unterteilt werden (siehe folgendes Bild). Wenn Sie den Summer so drehen, dass seine Pins nach oben zeigen, ist der Summer mit einer grünen Leiterplatte ein passiver Summer, während der mit schwarzem Klebeband umschlossene ein aktiver Summer ist.

Der Unterschied zwischen einem aktiven und einem passiven Summer:

Ein aktiver Summer verfügt über eine eingebaute Oszillationsquelle und erzeugt daher beim Anlegen von Strom einen Ton. Ein passiver Summer hat jedoch keine solche Quelle, daher gibt er keinen Ton von sich, wenn Gleichstromsignale verwendet werden; stattdessen müssen Sie ihn mit Rechteckwellen antreiben, deren Frequenz zwischen 2K und 5K liegt. Der aktive Summer ist oft teurer als der passive, aufgrund der mehrfach eingebauten Oszillationskreise.

Folgend ist das elektrische Symbol eines Summers dargestellt. Er hat zwei Pins mit positiven und negativen Polen. Ein + auf der Oberfläche zeigt den Anodenpol und der andere ist der Kathodenpol.



Sie können die Pins des Summers überprüfen, der längere ist die Anode und der kürzere die Kathode. Bitte verwechseln Sie diese nicht beim Anschließen, da sonst der Summer keinen Ton erzeugt.

Summer - Wikipedia

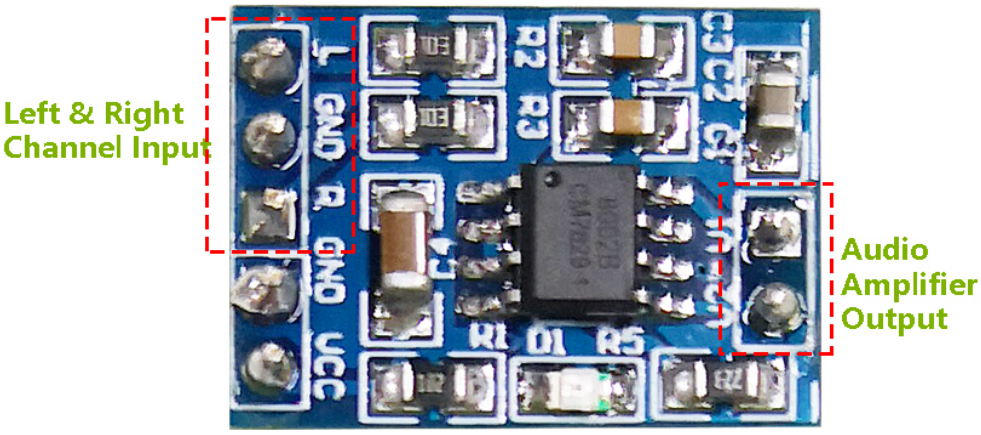
Beispiel

- *3.1 Piepser* (Arduino-Projekt)
- *3.2 Eigene Töne* (Arduino-Projekt)
- *6.3 Einparkhilfe* (Arduino-Projekt)

- *3.2 Eigener Klang* (MicroPython-Projekt)
- *3.1 Beep* (MicroPython-Projekt)
- *6.4 Einparkhilfe* (MicroPython-Projekt)

5.16 Audio-Modul und Lautsprecher

Audio-Verstärkermodul

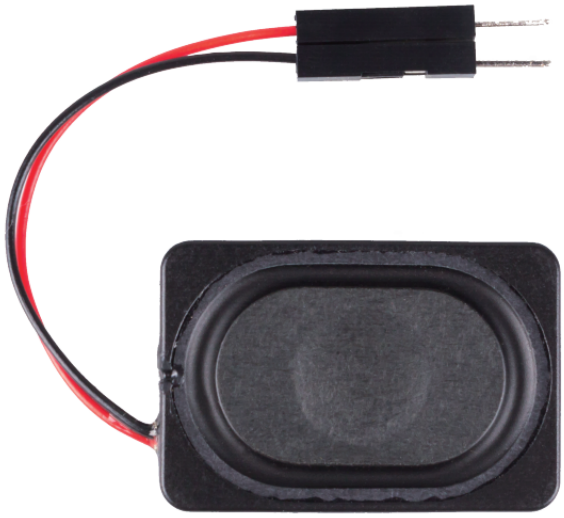


Das Audio-Verstärkermodul enthält einen HXJ8002 Audio-Leistungsverstärkerchip. Dieser Chip ist ein Leistungsverstärker mit geringer Stromversorgung, der 3W durchschnittliche Audiokraft für eine 3Ω BTL-Last bei niedriger harmonischer Verzerrung (unter 10% Schwellenverzerrung bei 1 kHz) aus einer 5V Gleichstromquelle liefern kann. Dieser Chip kann Audiosignale ohne jegliche Koppelkondensatoren oder Bootstrap-Kondensatoren verstärken.

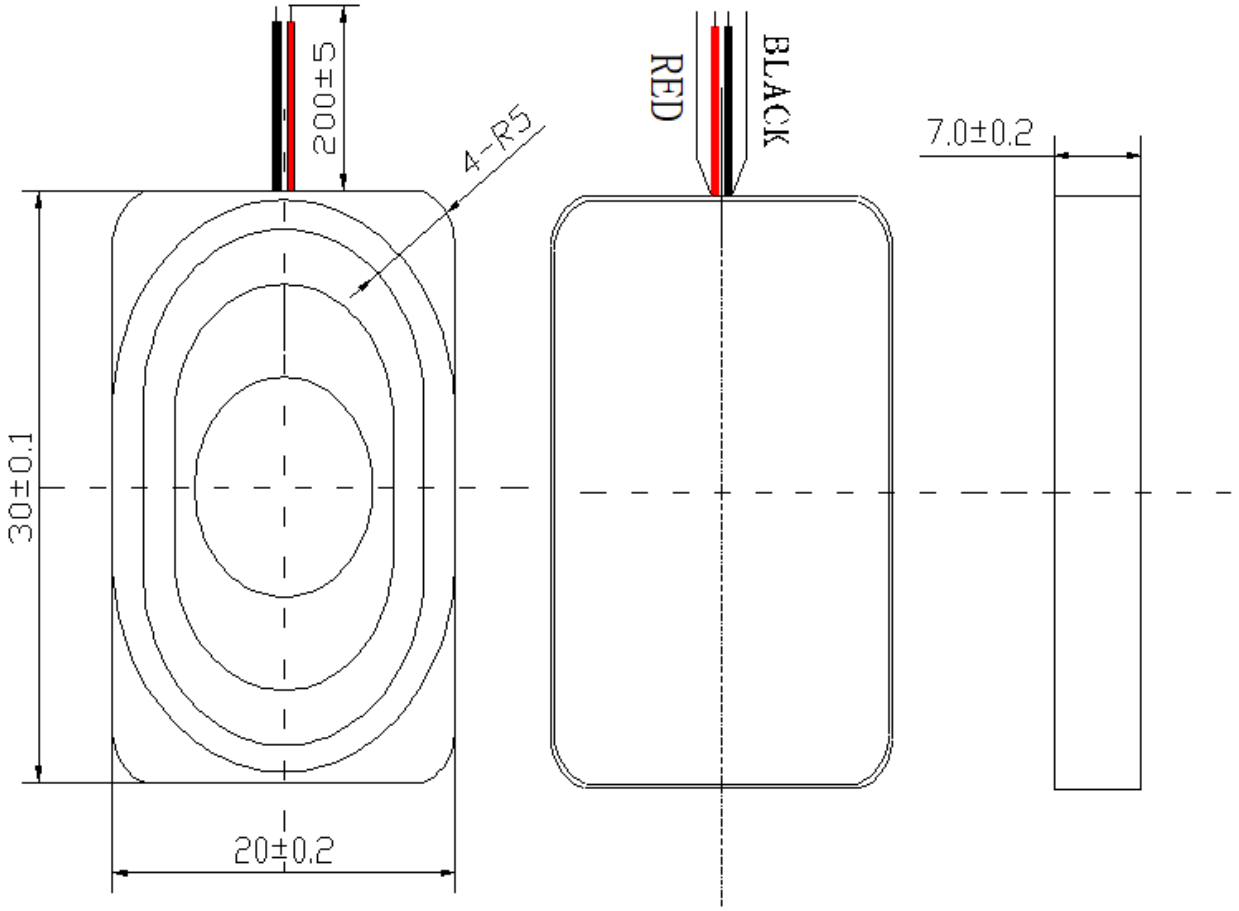
Das Modul kann mit einer Gleichstromquelle von 2,0V bis 5,5V und 10mA Betriebsstrom (0,6uA für typischen Standby-Strom) betrieben werden und erzeugt einen leistungsstarken, verstärkten Klang in einem 3, 4 oder 8 Impedanz-Lautsprecher. Dieses Modul verfügt über verbesserte Knack- und Knall-Schaltkreise, die das Übergangsrauschen beim Ein- und Ausschalten erheblich reduzieren. Die winzige Größe neben hoher Effizienz und geringer Stromversorgung macht es weit verbreitet in tragbaren und batteriebetriebenen Projekten und Mikrocontrollern einsetzbar.

- **IC**: HXJ8002
- **Eingangsspannung**: 2V ~ 5,5V
- **Standby-Modus Strom**: 0,6uA (typischer Wert)
- **Ausgangsleistung**: 3W (3Ω Last), 2,5W (4Ω Last), 1,5W (8Ω Last)
- **Ausgangsimpedanz des Lautsprechers**: 3Ω, 4Ω, 8Ω
- **Größe**: 19,8mm x 14,2mm

Lautsprecher



- **Größe**: 20x30x7mm
- **Impedanz**: 8ohm
- **Nennleistung**: 1,5W
- **Maximale Eingangsleistung**: 2,0W
- **Kabellänge**: 10cm



Die Größentabelle ist wie folgt

- `Datenblatt des 2030 Lautsprechers`

Beispiel

- *7.3 Bluetooth-Audio-Player* (Arduino-Projekt)
- *7.5 MP3-Player mit SD-Kartenunterstützung* (Arduino-Projekt)

Treiber

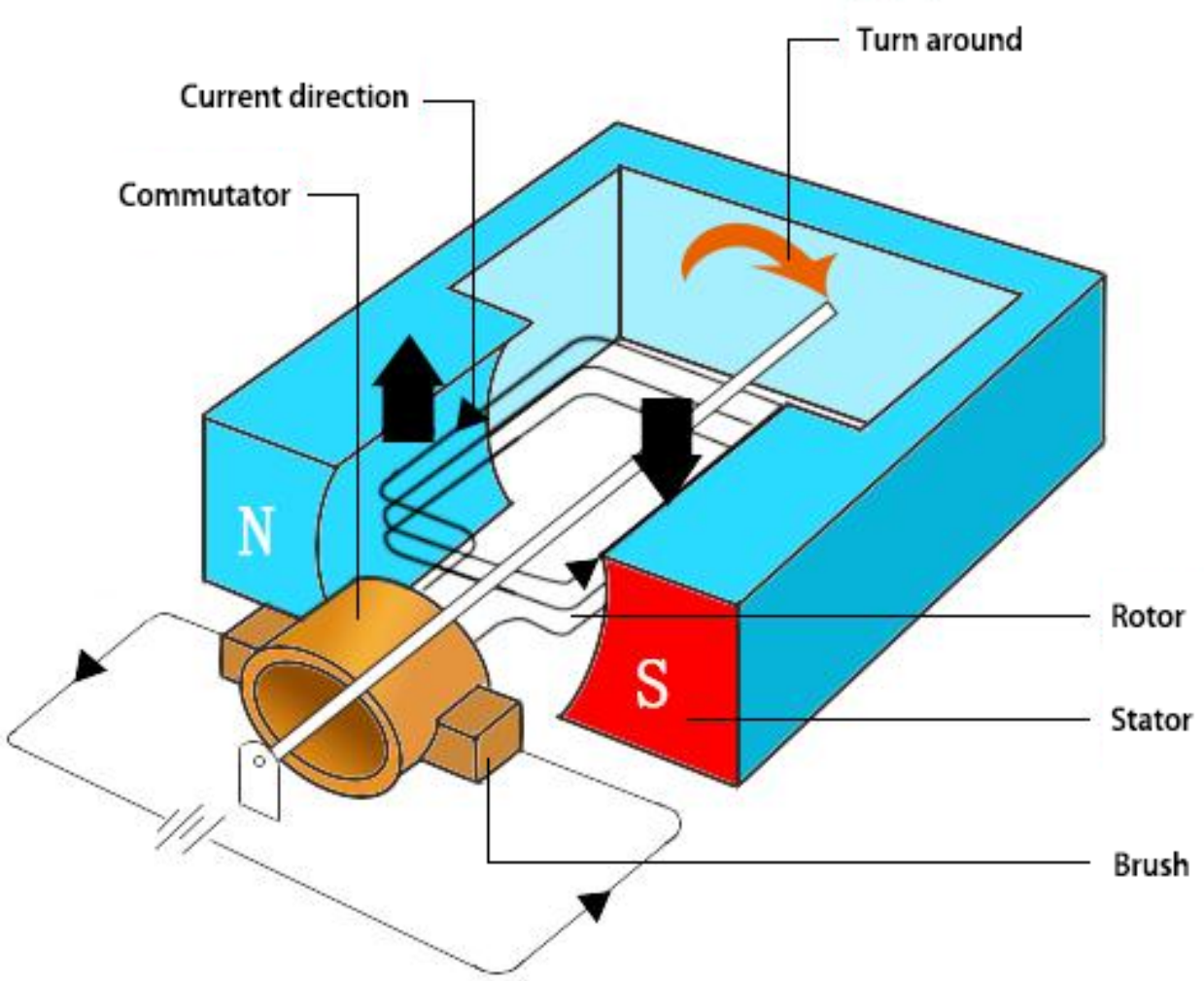
5.17 Gleichstrommotor

Hier handelt es sich um einen 3V-Gleichstrommotor. Wenn Sie an jeden der beiden Anschlüsse einen hohen und einen niedrigen Pegel anlegen, wird er sich drehen.

- **Länge**: 25mm
- **Durchmesser**: 21mm
- **Wellendurchmesser**: 2mm
- **Wellenlänge**: 8mm
- **Spannung**: 3-6V
- **Strom**: 0.35-0.4A
- **Geschwindigkeit bei 3V**: 19000 U/min (Umdrehungen pro Minute)
- **Gewicht**: Ungefähr 14g (für eine Einheit)

Ein Gleichstrommotor (DC-Motor) ist ein kontinuierlicher Aktuator, der elektrische Energie in mechanische Energie umwandelt. DC-Motoren ermöglichen es, dass rotierende Pumpen, Ventilatoren, Kompressoren, Impeller und andere Geräte durch kontinuierliche Drehbewegungen funktionieren.

Ein DC-Motor besteht aus zwei Teilen: dem feststehenden Teil des Motors, dem **Stator**, und dem internen Teil des Motors, dem **Rotor** (oder **Anker** eines Gleichstrommotors), der sich dreht, um Bewegung zu erzeugen. Der Schlüssel zur Bewegungserzeugung ist die Positionierung des Ankers innerhalb des Magnetfelds des Permanentmagneten (dessen Feld sich vom Nordpol zum Südpol erstreckt). Die Wechselwirkung des Magnetfelds und der bewegten geladenen Partikel (der stromführende Draht erzeugt das Magnetfeld) erzeugt das Drehmoment, das den Anker dreht.



Der Strom fließt vom positiven Pol der Batterie durch den Schaltkreis, über die Kupferbürsten zum Kommutator und dann zum Anker. Durch die beiden Lücken im Kommutator kehrt sich dieser Fluss jedoch bei jeder vollständigen Drehung zur Hälfte um.

Diese kontinuierliche Umkehrung wandelt die Gleichstromleistung der Batterie im Wesentlichen in Wechselstrom um, wodurch der Anker das Drehmoment in der richtigen Richtung zur richtigen Zeit erfährt, um die Drehung aufrechtzuerhalten.

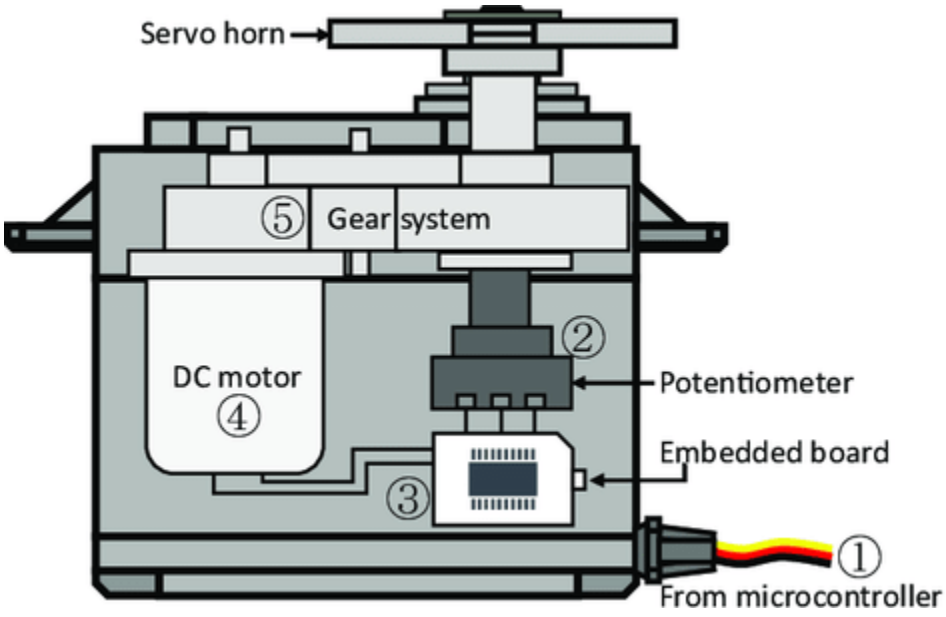
Beispiele

- *4.1 Motor* (Arduino-Projekt)
- *4.1 Kleiner Ventilator* (MicroPython-Projekt)
- *2.9 Rotierender Ventilator* (Scratch-Projekt)

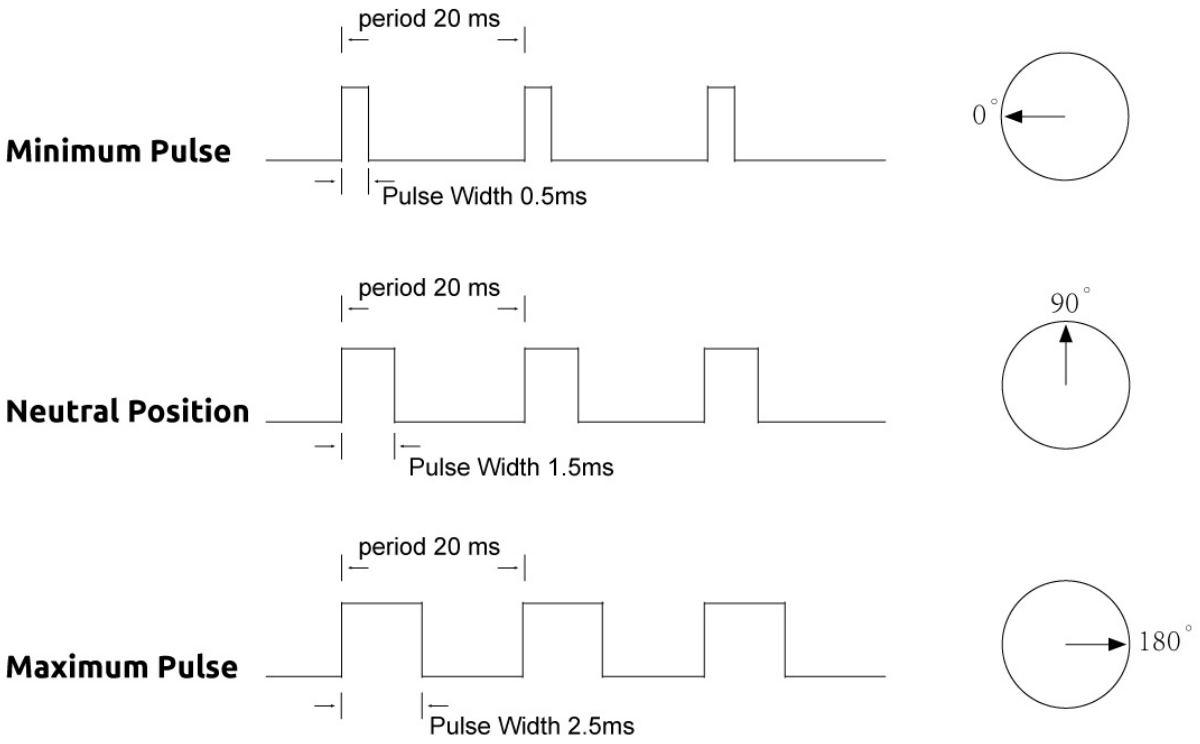
5.18 Servo

Ein Servomotor besteht im Allgemeinen aus folgenden Teilen: Gehäuse, Welle, Getriebesystem, Potentiometer, Gleichstrommotor und integrierte Platine.

Die Funktionsweise ist wie folgt: Der Mikrocontroller sendet PWM-Signale an den Servo. Die integrierte Platine im Servo empfängt die Signale über den Signalpin und steuert den internen Motor. Daraufhin treibt der Motor das Getriebesystem an und bewegt nach Verlangsamung die Welle. Die Welle und das Potentiometer des Servos sind miteinander verbunden. Wenn die Welle sich dreht, treibt sie das Potentiometer an, sodass das Potentiometer ein Spannungssignal an die Platine sendet. Die Platine bestimmt dann basierend auf der aktuellen Position die Drehrichtung und -geschwindigkeit, um genau in der definierten Position zu stoppen und dort zu verharren.



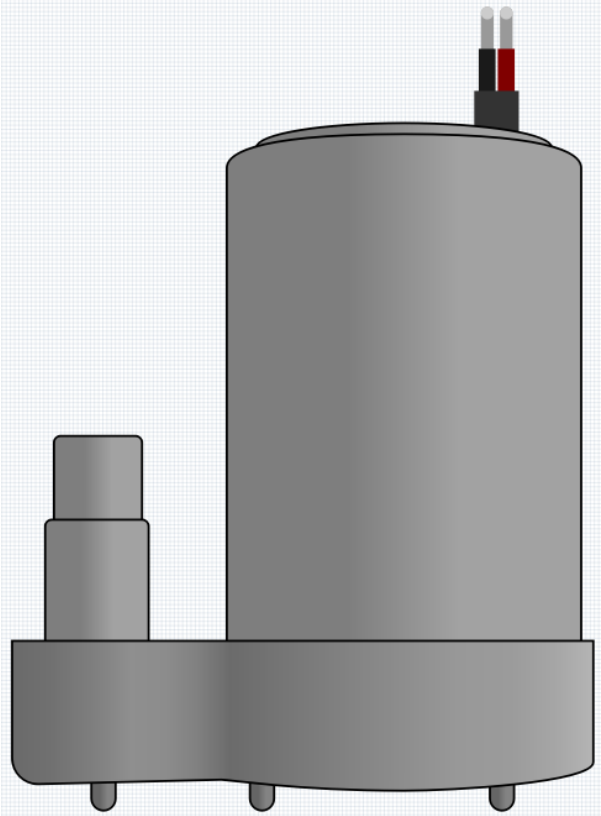
Der Winkel wird durch die Dauer eines Impulses bestimmt, der an das Steuerkabel angelegt wird. Dies wird als Pulsbreitenmodulation bezeichnet. Der Servo erwartet alle 20 ms einen Impuls. Die Länge des Impulses bestimmt, wie weit der Motor sich dreht. Ein 1,5-ms-Impuls beispielsweise lässt den Motor in die 90-Grad-Position (Neutrale Position) drehen. Wenn ein Impuls, der kürzer als 1,5 ms ist, an einen Servo gesendet wird, dreht sich der Servo in eine Position und hält seine Ausgangswelle einige Grad gegen den Uhrzeigersinn von der neutralen Position. Wenn der Impuls länger als 1,5 ms ist, tritt das Gegenteil ein. Die minimale und maximale Impulsbreite, die den Servo dazu bringt, sich in eine gültige Position zu drehen, sind abhängig vom jeweiligen Servo. Allgemein wird die minimale Impulsbreite etwa 0,5 ms und die maximale etwa 2,5 ms betragen.



Beispiele

- *4.3 Schwingender Servo* (Arduino-Projekt)
- *4.3 Schwingender Servomotor* (MicroPython-Projekt)

5.19 Zentrifugalpumpe



Die Zentrifugalpumpe wandelt Rotationsenergie in hydrodynamische Energie um, um Flüssigkeiten zu transportieren. Die Rotationsenergie stammt aus dem Elektromotor. Die Flüssigkeit tritt entlang oder in der Nähe der rotierenden Welle in das Pumpenlaufrad ein, wird vom Laufrad beschleunigt, fließt radial nach außen in das Diffusor- oder Spiralgehäuse und strömt von dort aus ab.

Häufige Einsatzgebiete von Zentrifugalpumpen sind Wasser-, Abwasser-, Landwirtschafts-, Erdöl- und Petrochemiepumpen.

- Zentrifugalpumpe - Wikipedia

Merkmale

- **Spannungsbereich**: DC 3 ~ 4,5V
- **Betriebsstrom**: 120 ~ 180mA
- **Leistung**: 0,36 ~ 0,91W
- **Maximale Förderhöhe**: 0,35 ~ 0,55M
- **Maximale Fördermenge**: 80 ~ 100 L/H
- **Dauerbetriebszeit**: 100 Stunden
- **Wasserdichtheitsgrad**: IP68
- **Antriebsmodus**: DC, magnetischer Antrieb
- **Material**: Technischer Kunststoff
- **Außendurchmesser des Auslasses**: 7,8 mm

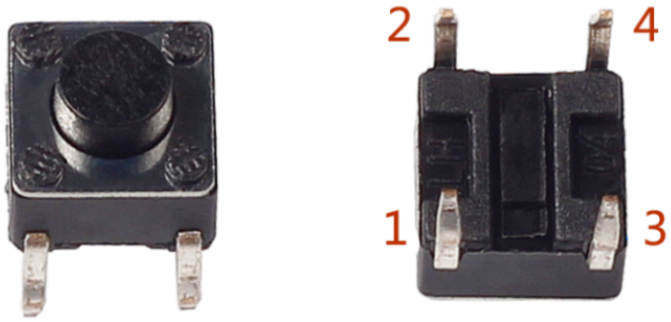
- **Innendurchmesser des Auslasses**: 6,5 mm
- Es handelt sich um eine Tauchpumpe, die auch so verwendet werden sollte. Es besteht Überhitzungsgefahr bei Betrieb außerhalb des Wassers, da sie sich sonst zu stark erwärmt.

Beispiele

- *4.2 Pumpen* (Arduino-Projekt)
- *6.6 Pflanzenüberwachung* (Arduino-Projekt)
- *4.2 Pumpen* (MicroPython-Projekt)
- *6.8 Pflanzenüberwachung* (MicroPython-Projekt)

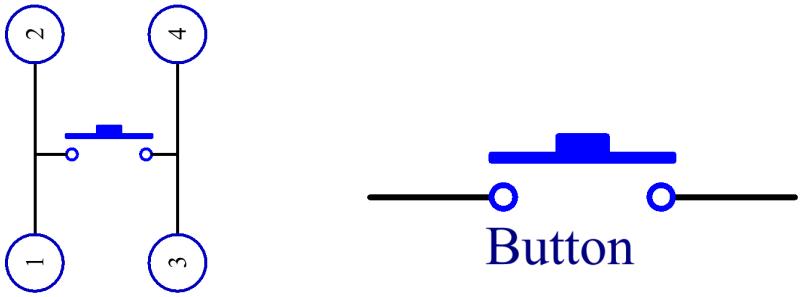
Controller

5.20 Taste

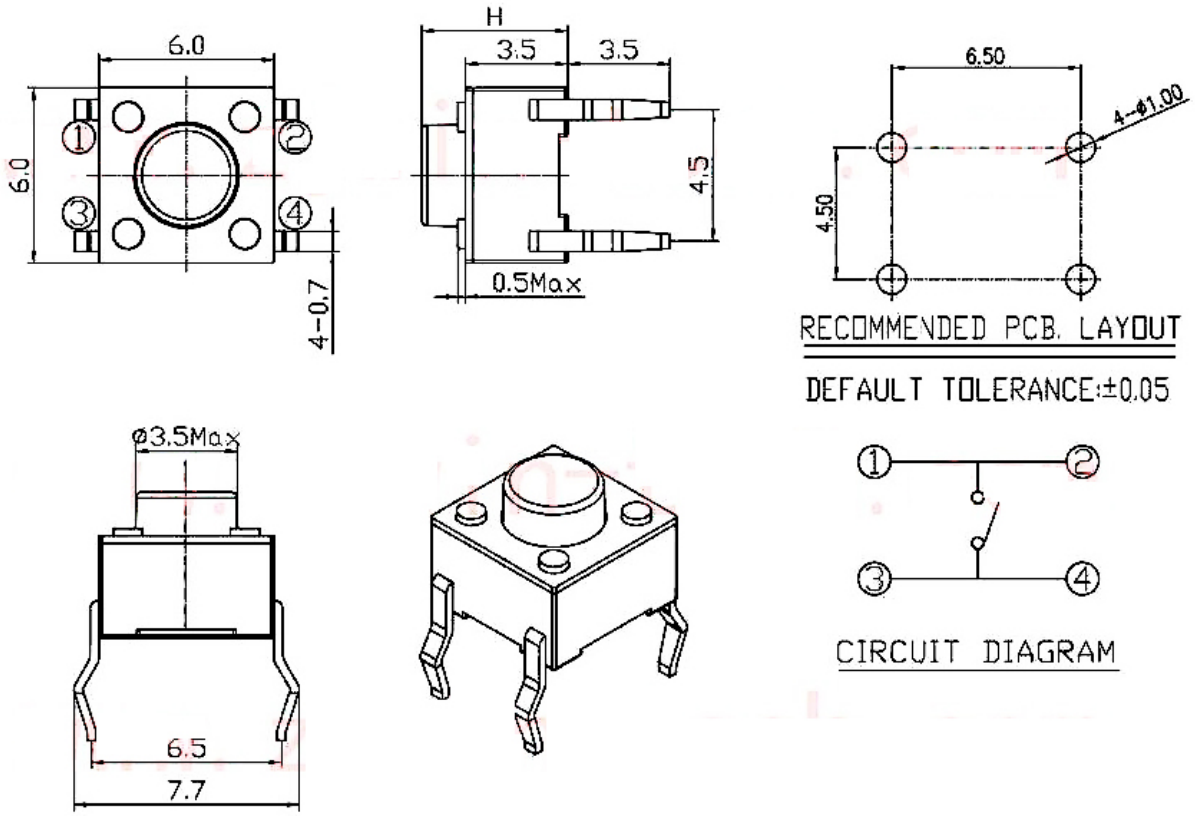


Tasten sind eine gängige Komponente zur Steuerung elektronischer Geräte. Sie werden üblicherweise als Schalter verwendet, um Stromkreise zu schließen oder zu unterbrechen. Obwohl es Tasten in verschiedenen Größen und Formen gibt, wird hier eine 6mm Mini-Taste verwendet, wie in den folgenden Bildern gezeigt. Pin 1 ist mit Pin 2 verbunden und Pin 3 mit Pin 4. Daher müssen Sie nur einen der Pins 1 und 2 mit Pin 3 oder Pin 4 verbinden.

Folgend ist die interne Struktur einer Taste dargestellt. Das Symbol rechts unten wird üblicherweise verwendet, um eine Taste in Schaltkreisen darzustellen.



Da Pin 1 mit Pin 2 und Pin 3 mit Pin 4 verbunden ist, werden beim Drücken der Taste die 4 Pins verbunden, wodurch der Stromkreis geschlossen wird.



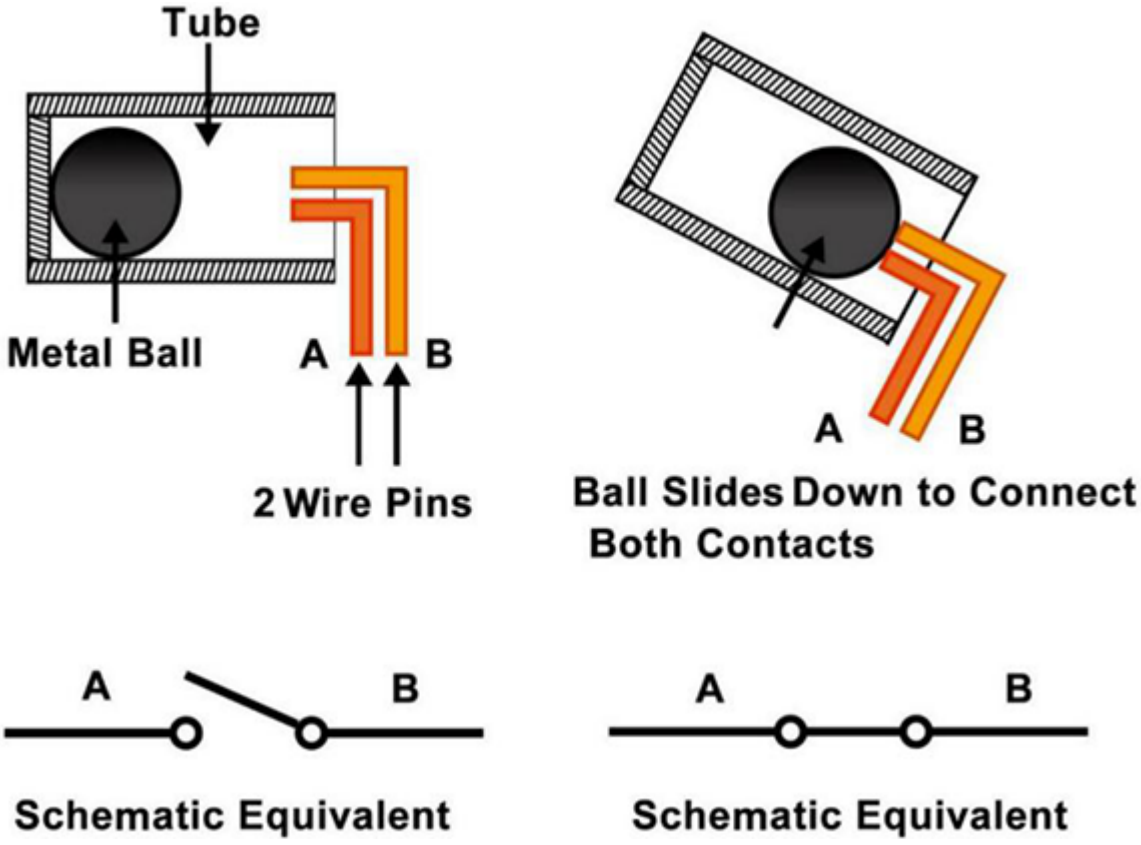
Beispiel

- *5.1 Lesen des Tasterwerts* (Arduino-Projekt)
- *5.1 Lesen des Tastenwertes* (MicroPython-Projekt)
- *2.5 Türklingel* (Scratch-Projekt)
- *2.14 SPIEL - Apfel Essen* (Scratch-Projekt)
- *2.17 SPIEL - Angeln* (Scratch-Projekt)

5.21 Neigungsschalter

Der hier verwendete Neigungsschalter ist ein Kugeltyp mit einer Metallkugel im Inneren. Er wird eingesetzt, um Neigungen in kleinem Winkel zu erkennen.

Das Prinzip ist sehr einfach. Wird der Schalter in einem bestimmten Winkel geneigt, rollt die Kugel nach unten und berührt die beiden Kontakte, die mit den äußeren Pins verbunden sind, und löst dadurch Schaltvorgänge aus. Andernfalls bleibt die Kugel von den Kontakten entfernt und unterbricht somit den Stromkreis.

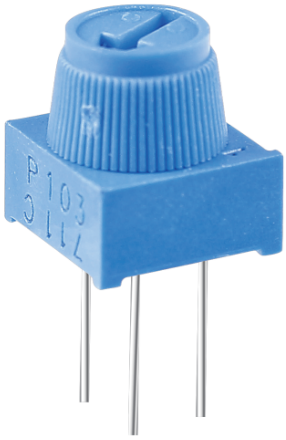


- SW520D Neigungsschalter-Datenblatt

Beispiele

- *5.2 Kippen!* (Arduino-Projekt)
- *5.2 Kippen Sie es!* (MicroPython-Projekt)

5.22 Potentiometer

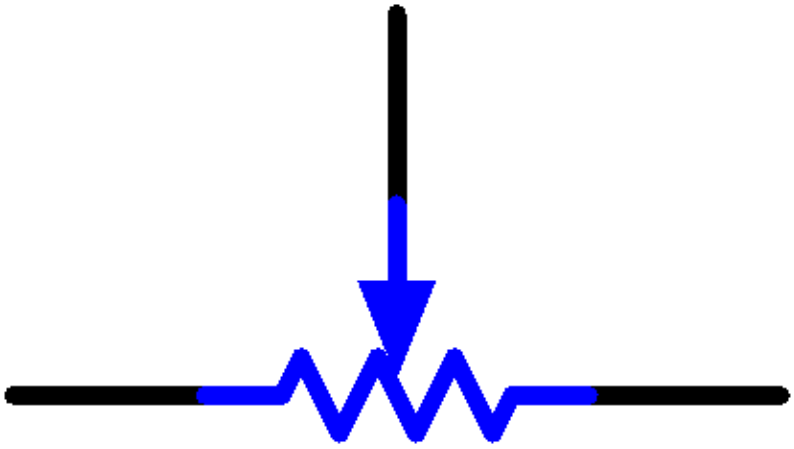


Ein Potentiometer ist ebenfalls ein Widerstandsbauteil mit drei Anschlüssen, dessen Widerstandswert entsprechend einer regelmäßigen Variation angepasst werden kann.

Potentiometer gibt es in verschiedenen Formen, Größen und Werten, aber sie haben alle Folgendes gemeinsam:

- Sie verfügen über drei Anschlüsse (oder Verbindungspunkte).
- Sie besitzen einen Drehknopf, eine Schraube oder einen Schieber, der bewegt werden kann, um den Widerstand zwischen dem mittleren Anschluss und einem der beiden äußeren Anschlüsse zu variieren.
- Der Widerstand zwischen dem mittleren Anschluss und einem der beiden äußeren Anschlüsse variiert von 0 bis zum maximalen Widerstand des Potentiometers, je nachdem wie weit der Drehknopf, die Schraube oder der Schieber bewegt wird.

Hier ist das Schaltungssymbol des Potentiometers.



Die Funktionen des Potentiometers in einer Schaltung sind wie folgt:

1. Als Spannungsteiler

   Das Potentiometer ist ein kontinuierlich einstellbarer Widerstand. Wenn Sie die Welle oder den Schiebegriff des Potentiometers einstellen, wird der bewegliche Kontakt auf dem Widerstand gleiten. An diesem Punkt kann eine Spannung ausgegeben werden, abhängig von der am Potentiometer angelegten Spannung und dem Winkel, den der bewegliche Arm gedreht hat, oder dem Weg, den er zurückgelegt hat.

2. Als Rheostat

   Wird das Potentiometer als Rheostat verwendet, verbinden Sie den mittleren Pin mit einem der beiden anderen Pins im Schaltkreis. Dadurch erhalten Sie einen sanft und kontinuierlich veränderbaren Widerstandswert innerhalb des Weges des beweglichen Kontakts.

3. Als Stromregler

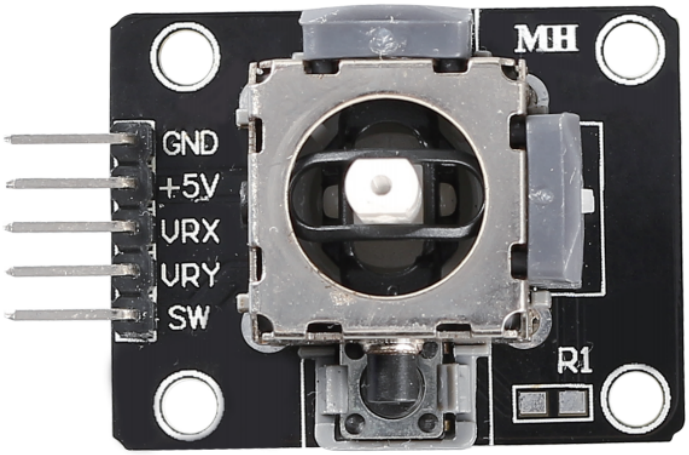
   Wenn das Potentiometer als Stromregler wirkt, muss der Schleifkontakt als einer der Ausgangsanschlüsse verbunden werden.

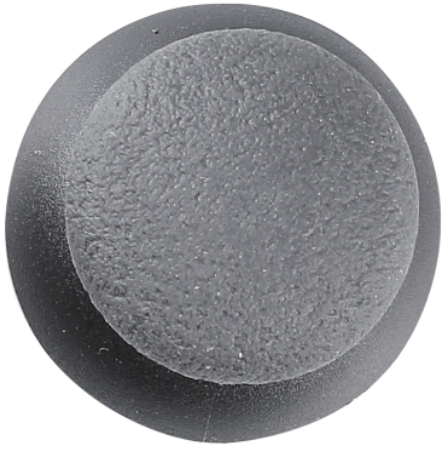
Wenn Sie mehr über Potentiometer erfahren möchten, siehe: Potentiometer - Wikipedia

Beispiele

- *5.8 Den Knopf Drehen* (Arduino-Projekt)
- *5.8 Drehen Sie den Knopf* (MicroPython-Projekt)
- *2.4 Bewegliche Maus* (Scratch-Projekt)
- *2.16 SPIEL - Breakout-Klon* (Scratch-Projekt)

5.23 Joystick-Modul





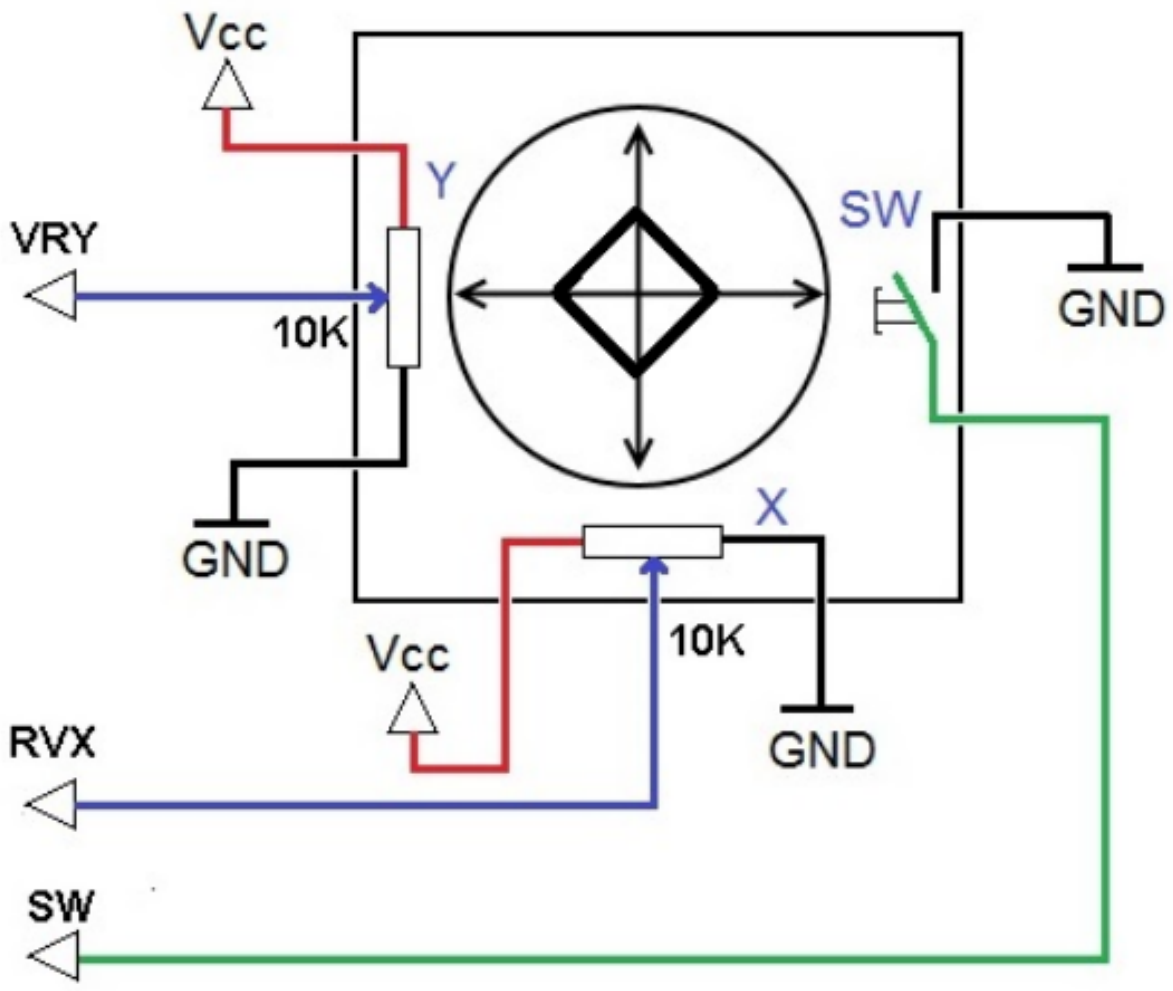
- **GND**: Masse.
- **+5V**: Stromversorgung, akzeptiert 3,3V bis 5V.
- **VRX**: Analoger Ausgang entsprechend der horizontalen (X-Achse) Position des Joysticks.
- **VRY**: Analoger Ausgang entsprechend der vertikalen (Y-Achse) Position des Joysticks.
- **SW**: Schalterausgang, wird aktiviert, wenn der Joystick nach unten gedrückt wird. Für eine ordnungsgemäße Funktion ist ein externer Pull-Up-Widerstand erforderlich. Mit dem Widerstand gibt der SW-Pin im Leerlauf ein hohes Signal aus und wird niedrig, wenn der Joystick gedrückt wird.

Die Grundidee eines Joysticks besteht darin, die Bewegung eines Sticks in elektronische Informationen umzusetzen, die ein Computer verarbeiten kann.

Um dem Computer eine vollständige Bewegungspalette zu übermitteln, muss ein Joystick die Position des Sticks auf zwei Achsen messen - der X-Achse (links nach rechts) und der Y-Achse (oben nach unten). Wie in der Grundgeometrie geben die X-Y-Koordinaten genau die Position des Sticks an.

Um die Position des Sticks zu bestimmen, überwacht das Joystick-Steuerungssystem einfach die Position jeder Achse. Das konventionelle analoge Joystick-Design macht dies mit zwei Potentiometern oder variablen Widerständen.

Der Joystick verfügt auch über einen digitalen Eingang, der aktiviert wird, wenn der Joystick nach unten gedrückt wird.



Beispiele

- *5.11 Den Joystick umschalten* (Arduino-Projekt)
- *5.11 Bedienung des Joysticks* (MicroPython-Projekt)
- *2.13 SPIEL - Sternenkreuzung* (Scratch-Projekt)
- *2.20 SPIEL - Drachen Töten* (Scratch-Projekt)

5.24 IR-Empfänger

IR-Empfänger

- OUT: Signalausgang
- GND: Erdung
- VCC: Stromversorgung, 3,3V~5V

SL838 Infrarot-Empfänger ist eine Komponente, die Infrarotsignale empfängt und in der Lage ist, eigenständig Infrarotstrahlen zu empfangen und Signale auszugeben, die mit dem TTL-Niveau kompatibel sind. Er ähnelt in der Größe einem normalen in Plastik verpackten Transistor und eignet sich für alle Arten von Infrarot-Fernbedienungen und Infrarot-Übertragungen.

Infrarot, oder IR, Kommunikation ist eine beliebte, kostengünstige und leicht zu verwendende drahtlose Kommunikationstechnologie. Infrarotlicht hat eine etwas längere Wellenlänge als sichtbares Licht, daher ist es für das menschliche Auge nicht wahrnehmbar - ideal für drahtlose Kommunikation. Ein gängiges Modulationsschema für die Infrarotkommunikation ist die 38kHz-Modulation.

- Kann für Fernsteuerung verwendet werden
- Breiter Betriebsspannungsbereich: 2,7~5V
- Interner Filter für PCM-Frequenz
- TTL- und CMOS-Kompatibilität
- Starke Störunterdrückungsfähigkeit
- RoHS-konform

Fernbedienung

Dies ist eine Mini-Dünne Infrarot-Fernbedienung mit 21 Funktionstasten und einer Übertragungsreichweite von bis zu 8 Metern, geeignet für die Bedienung einer Vielzahl von Geräten in einem Kinderzimmer.

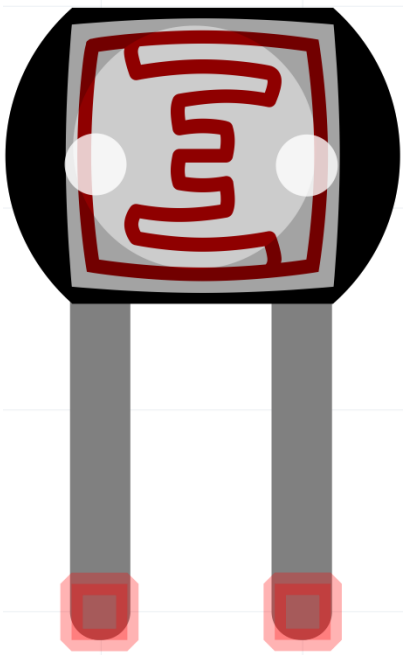
- Größe: 85x39x6mm
- Fernbedienungsreichweite: 8-10m
- Batterie: 3V Knopfzellen-Lithium-Mangan-Batterie
- Infrarot-Trägerfrequenz: 38KHz
- Oberflächenklebstoffmaterial: 0,125mm PET
- Effektive Lebensdauer: mehr als 20.000 Mal

Beispiele

- *5.14 IR-Empfänger* (Arduino-Projekt)
- *6.7 Zahlenraten* (Arduino-Projekt)
- *5.14 IR-Fernbedienung* (MicroPython-Projekt)
- *6.7 Zahlenraten* (MicroPython-Projekt)

Sensor

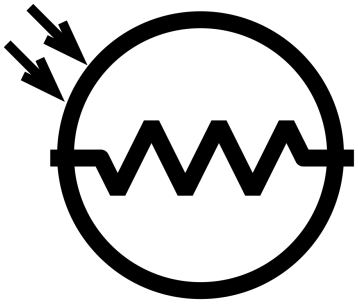
5.25 Fotowiderstand



Ein Fotowiderstand oder eine Fotodiode ist ein lichtgesteuerter variabler Widerstand. Der Widerstand eines Fotowiderstands verringert sich mit zunehmender Lichtintensität; anders ausgedrückt, zeigt er eine Foto-Leitfähigkeit.

Fotowiderstände können in lichtempfindlichen Detektorschaltungen sowie in licht- und dunkelaktivierten Schaltkreisen als Widerstandshalbleiter eingesetzt werden. Im Dunkeln kann ein Fotowiderstand einen Widerstand von mehreren Megaohm (M) haben, während er im Hellen einen Widerstand von nur einigen hundert Ohm erreichen kann.

Hier ist das elektronische Symbol für einen Fotowiderstand.

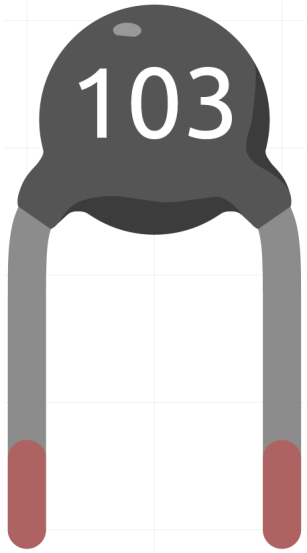


- Fotowiderstand - Wikipedia

Beispiele

- *5.7 Das Licht Fühlen* (Arduino-Projekt)
- *6.6 Pflanzenüberwachung* (Arduino-Projekt)
- *5.7 Das Licht erfühlen* (MicroPython-Projekt)
- *6.8 Pflanzenüberwachung* (MicroPython-Projekt)

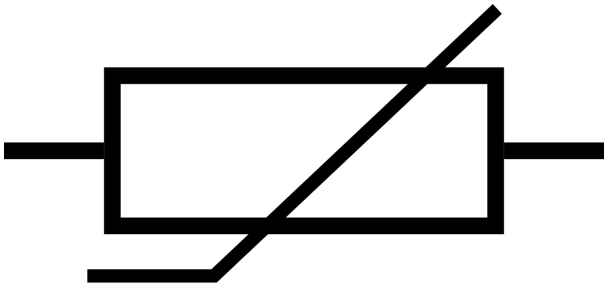
5.26 Thermistor



Ein Thermistor ist eine Art Widerstand, dessen Widerstand stark temperaturabhängig ist, stärker als bei Standardwiderständen. Der Begriff ist eine Kombination aus „thermal“ und „resistor“. Thermistoren werden häufig als Einschaltstrombegrenzer, Temperatursensoren (typischerweise vom Typ mit negativem Temperaturkoeffizienten, NTC), selbst zurücksetzende Überstromschutzvorrichtungen und selbstregulierende Heizelemente (typischerweise vom Typ mit positivem Temperaturkoeffizienten, PTC) verwendet.

- Thermistor – Wikipedia

Hier ist das elektronische Symbol eines Thermistors.



Es gibt zwei grundlegend entgegengesetzte Arten von Thermistoren:

- Bei NTC-Thermistoren nimmt der Widerstand mit steigender Temperatur ab, normalerweise aufgrund einer Zunahme der Leitungselektronen, die durch thermische Agitation aus dem Valenzband herausgestoßen werden. Ein NTC wird üblicherweise als Temperatursensor verwendet oder in Reihe mit einem Stromkreis als Einschaltstrombegrenzer.
- Bei PTC-Thermistoren steigt der Widerstand mit zunehmender Temperatur an, üblicherweise aufgrund verstärkter thermischer Gitterschwingungen, insbesondere von Verunreinigungen und Unvollkommenheiten. PTC-Thermistoren werden häufig in Reihe mit einem Stromkreis installiert und zum Schutz vor Überstrombedingungen als rückstellbare Sicherungen verwendet.

In diesem Kit verwenden wir einen NTC. Jeder Thermistor hat einen normalen Widerstand. Hier beträgt er 10k Ohm, gemessen bei 25 Grad Celsius.

Hier ist die Beziehung zwischen Widerstand und Temperatur:

RT = RN * expB(1/TK - 1/TN)

- **RT** ist der Widerstand des NTC-Thermistors bei der Temperatur TK.
- **RN** ist der Widerstand des NTC-Thermistors bei der Nenntemperatur TN. Hier beträgt der numerische Wert von RN 10k.
- **TK** ist eine Kelvin-Temperatur und die Einheit ist K. Hier beträgt der numerische Wert von TK 273,15 + Grad Celsius.
- **TN** ist eine Nenn-Kelvin-Temperatur; die Einheit ist ebenfalls K. Hier beträgt der numerische Wert von TN 273,15+25.
- **B(Beta)**, die Materialkonstante des NTC-Thermistors, wird auch als Wärmeempfindlichkeitsindex bezeichnet und hat den numerischen Wert 3950.
- **exp** ist die Abkürzung für exponentiell, und die Basiszahl e ist eine natürliche Zahl, die ungefähr 2,7 beträgt.

Konvertieren Sie diese Formel TK=1/(ln(RT/RN)/B+1/TN), um die Kelvin-Temperatur zu erhalten, von der 273,15 abgezogen die Grad Celsius ergibt.

Diese Beziehung ist eine empirische Formel. Sie ist nur genau, wenn die Temperatur und der Widerstand innerhalb des wirksamen Bereichs liegen.

Beispiele

- *5.10 Thermometer* (Arduino-Projekt)
- *8.4 IoT-Kommunikation mit MQTT* (Arduino-Projekt)
- *5.10 Temperaturmessung* (MicroPython-Projekt)
- *2.6 Niedrigtemperaturalarm* (Scratch-Projekt)

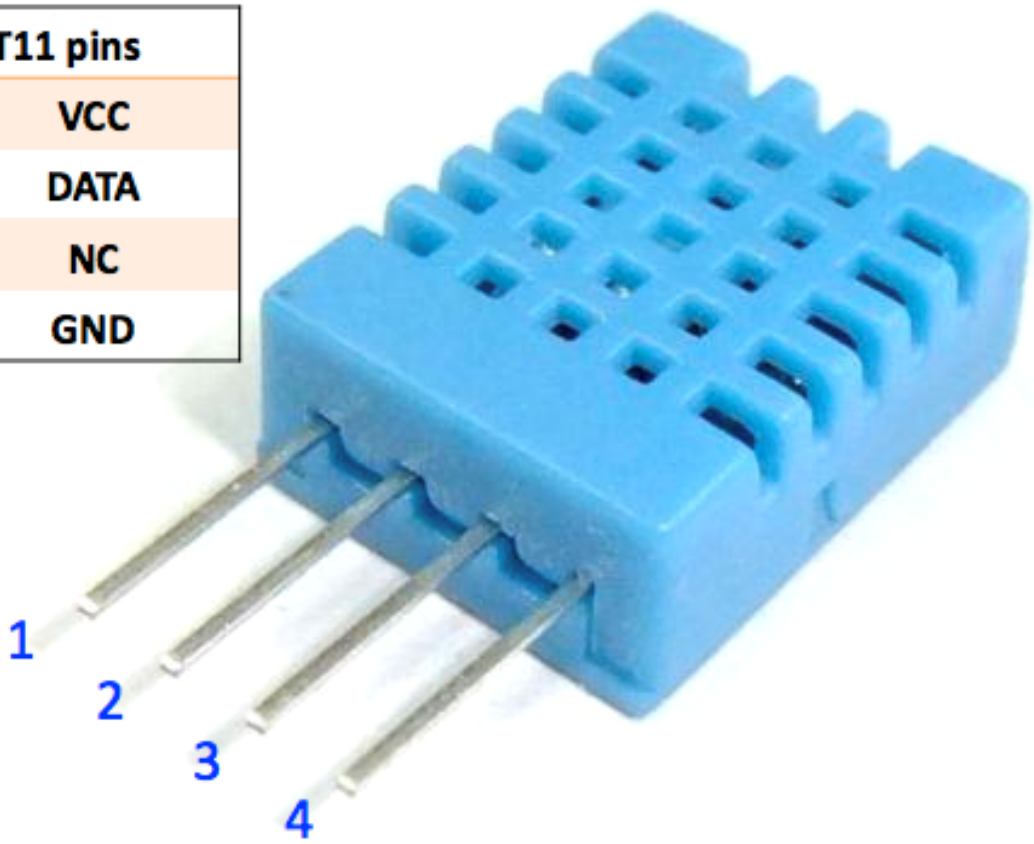
5.27 DHT11 Feuchtigkeits- und Temperatursensor

Der digitale Temperatur- und Feuchtigkeitssensor DHT11 ist ein zusammengesetzter Sensor, der einen kalibrierten digitalen Signaloutput für Temperatur und Feuchtigkeit enthält. Die Technologie spezialisierter digitaler Module sowie die Temperatur- und Feuchtigkeitssensorik werden eingesetzt, um hohe Zuverlässigkeit und exzellente Langzeitstabilität des Produkts zu gewährleisten.

Der Sensor umfasst einen resistiven Feuchtigkeitssensor und ein NTC-Temperaturmessgerät, verbunden mit einem leistungsfähigen 8-Bit-Mikrocontroller.

Nur drei Pins stehen zur Verfügung: VCC, GND und DATA. Der Kommunikationsprozess beginnt damit, dass die DATA-Leitung Startsignale an DHT11 sendet. DHT11 empfängt diese Signale und sendet ein Antwortsignal zurück. Anschließend empfängt der Host das Antwortsignal und beginnt, 40 Bit Feuchtigkeits- und Temperaturdaten zu empfangen (8 Bit Feuchtigkeit ganz, 8 Bit Feuchtigkeit Dezimal, 8 Bit Temperatur ganz, 8 Bit Temperatur Dezimal, 8 Bit Prüfsumme).

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



Eigenschaften

1. Feuchtigkeitsmessbereich: 20 - 90% RH
2. Temperaturmessbereich: 0 - 60℃
3. Ausgabe digitaler Signale für Temperatur und Feuchtigkeit
4. Betriebsspannung: DC 5V; PCB-Größe: 2,0 x 2,0 cm
5. Genauigkeit der Feuchtigkeitsmessung: ±5% RH
6. Genauigkeit der Temperaturmessung: ±2℃

- DHT11 Datenblatt

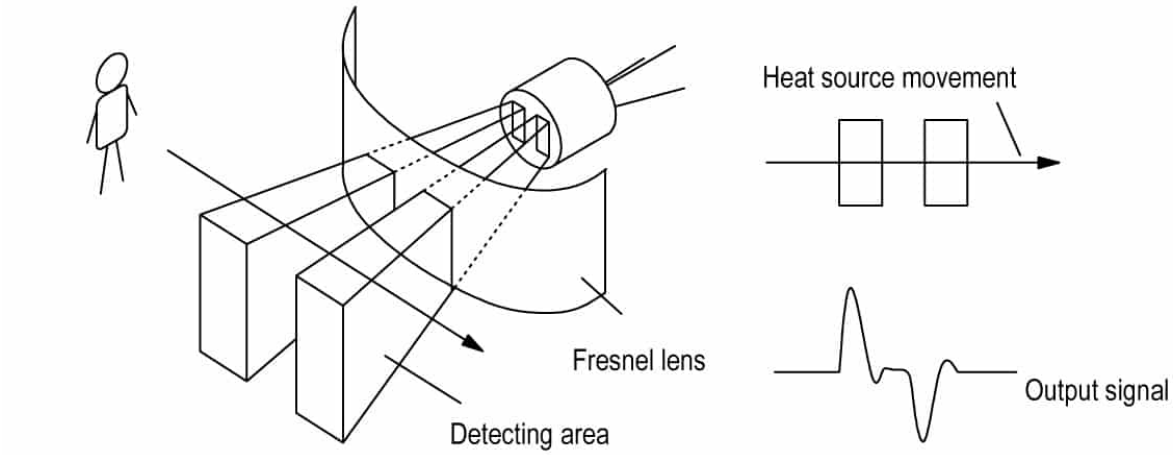
Beispiele

- *5.13 Temperatur - Feuchtigkeit* (Arduino-Projekt)
- *6.6 Pflanzenüberwachung* (Arduino-Projekt)
- *8.6 Temperatur- und Feuchtigkeitsüberwachung mit Adafruit IO* (Arduino-Projekt)
- *5.13 Temperatur - Feuchtigkeit* (MicroPython-Projekt)
- *6.8 Pflanzenüberwachung* (MicroPython-Projekt)

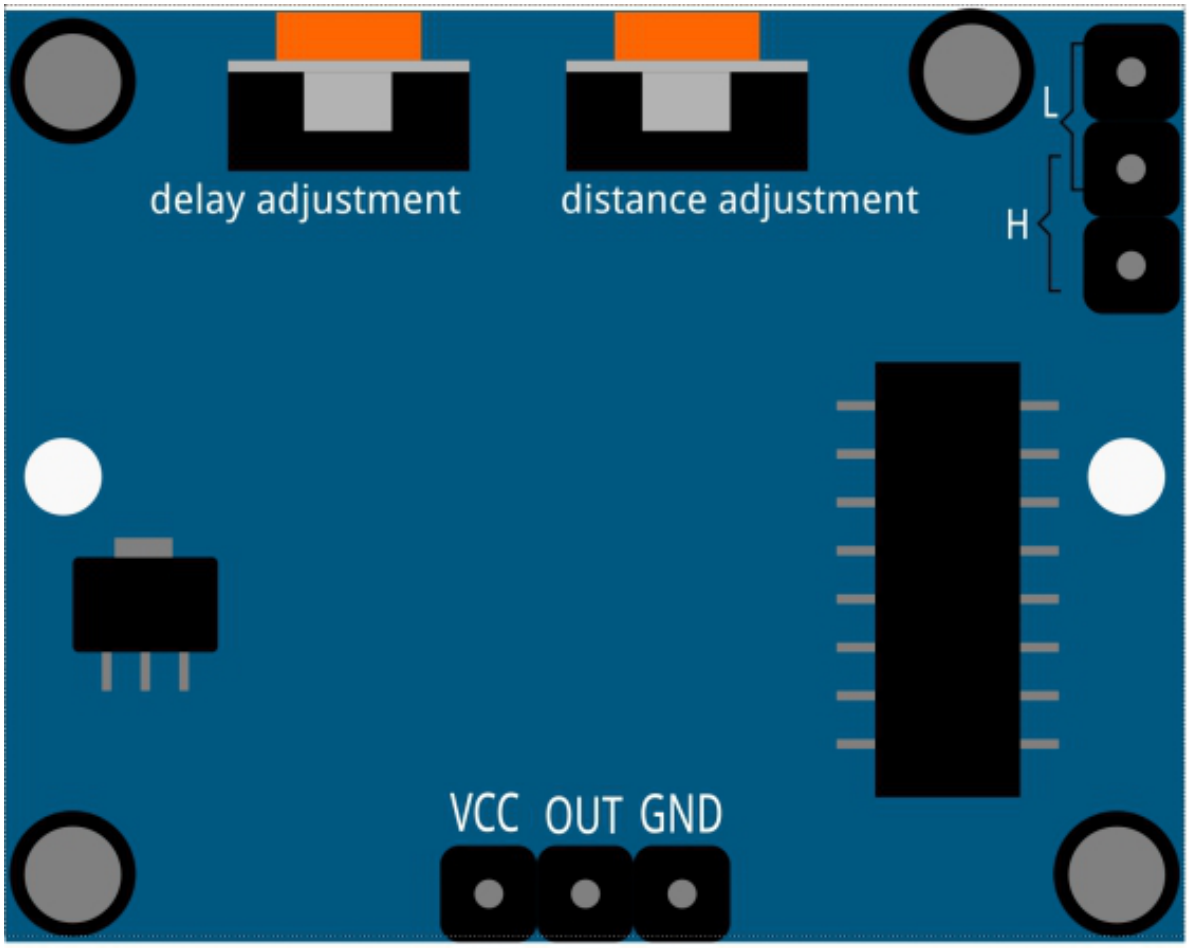
5.28 PIR-Bewegungssensormodul

Der PIR-Sensor erkennt infrarote Wärmestrahlung, die zur Detektion der Anwesenheit von Organismen verwendet werden kann, die infrarote Wärmestrahlung abgeben.

Der PIR-Sensor besteht aus zwei Schlitzen, die an einen Differenzverstärker angeschlossen sind. Wenn sich ein stationäres Objekt vor dem Sensor befindet, erhalten beide Schlitze dieselbe Menge an Strahlung und der Ausgang ist null. Bewegt sich jedoch ein Objekt vor dem Sensor, erhält einer der Schlitze mehr Strahlung als der andere, wodurch der Ausgang schwankt. Diese Änderung der Ausgangsspannung ist das Ergebnis der Bewegungserkennung.



Nachdem das Sensormodul verkabelt ist, erfolgt eine einminütige Initialisierung. Während der Initialisierung gibt das Modul 0-3 mal in Intervallen aus. Anschließend ist das Modul im Standby-Modus. Bitte halten Sie Störungen durch Lichtquellen und andere Quellen von der Oberfläche des Moduls fern, um Fehlfunktionen durch Störsignale zu vermeiden. Am besten verwenden Sie das Modul ohne zu viel Wind, da auch dieser den Sensor stören kann.



Distanzeinstellung

Drehen Sie den Knopf des Potentiometers zur Distanzeinstellung im Uhrzeigersinn, um den Bereich der Erfassungsdistanz zu erhöhen. Die maximale Erfassungsdistanz beträgt etwa 0-7 Meter. Drehen Sie ihn gegen den Uhrzeigersinn, verringert sich die Reichweite, und die minimale Erfassungsdistanz liegt bei etwa 0-3 Metern.

Verzögerungseinstellung

Drehen Sie den Knopf des Potentiometers zur Verzögerungseinstellung im Uhrzeigersinn, um die Sensing-Verzögerung zu erhöhen. Die maximale Verzögerung kann bis zu 300s betragen. Im Gegensatz dazu verkürzt sich die Verzögerung, wenn Sie es gegen den Uhrzeigersinn drehen, mit einem Minimum von 5s.

Zwei Auslösemodi

Wählen Sie unterschiedliche Modi mithilfe der Jumperkappe.

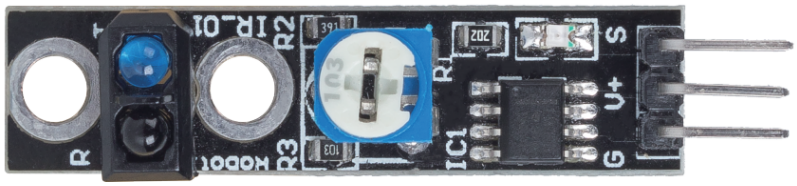
- **H**: Wiederholbarer Auslösemodus, nachdem der menschliche Körper erkannt wurde, gibt das Modul ein hohes Signal aus. Während des nachfolgenden Verzögerungszeitraums, wenn jemand den Erfassungsbereich betritt, bleibt das Signal auf hohem Niveau.
- **L**: Nicht wiederholbarer Auslösemodus, gibt ein hohes Signal aus, wenn es den menschlichen Körper erkennt. Nach der Verzögerung wechselt das Signal automatisch von hohem auf niedriges Niveau.

Beispiele

- *5.5 Menschliche Bewegung Erkennen* (Arduino-Projekt)
- *8.7 ESP-Kamera mit Telegram-Bot* (Arduino-Projekt)

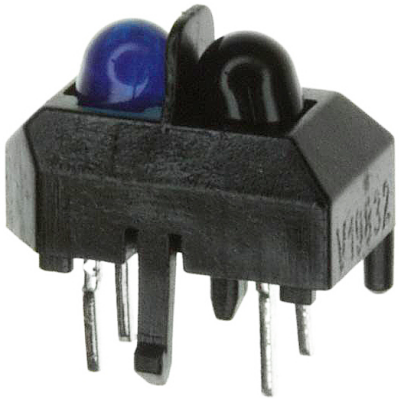
- *5.5 Menschliche Bewegungen erkennen* (MicroPython-Projekt)

5.29 Linienverfolgungsmodul



- S: Normalerweise auf niedrigem Pegel, hoher Pegel bei Erkennung einer schwarzen Linie.
- V+: Stromversorgung, 3.3V~5V
- G: Masse

Dies ist ein 1-Kanal-Linienverfolgungsmodul, das, wie der Name schon sagt, schwarze Linien auf weißem Hintergrund oder weiße Linien auf schwarzem Hintergrund verfolgt.



Das Modul verwendet einen TCRT500 Infrarotsensor, der aus einer Infrarot-LED (blau) und einem fotosensitiven Triplett (schwarz) besteht.

- Die blaue Infrarot-LED sendet beim Einschalten für das menschliche Auge unsichtbares Infrarotlicht aus.
- Der schwarze Phototransistor, der zum Empfangen von Infrarotlicht verwendet wird, hat einen internen Widerstand, dessen Widerstand sich mit dem empfangenen Infrarotlicht ändert; je mehr Infrarotlicht empfangen wird, desto niedriger sinkt sein Widerstand und umgekehrt.

Auf dem Modul befindet sich ein LM393-Komparator, der dazu dient, die Spannung des Phototransistors mit der eingestellten Spannung (mittels Potentiometer angepasst) zu vergleichen. Ist sie größer als die eingestellte Spannung, ist der Ausgang 1; andernfalls ist der Ausgang 0.

Daher, wenn der Infrarotsender auf eine schwarze Oberfläche leuchtet, wird, da Schwarz Licht absorbiert, der fotosensitive Transistor weniger Infrarotlicht empfangen, sein Widerstand wird steigen (Spannungszunahme), nach dem LM393-Komparator, der Ausgang hoher Pegel.

Ebenso, wenn es auf eine weiße Oberfläche leuchtet, wird das reflektierte Licht mehr und der Widerstand des fotosensitiven Transistors wird sinken (Spannungsabnahme); daher gibt der Komparator einen niedrigen Pegel aus und die Anzeige-LED leuchtet auf.

- TCRT5000

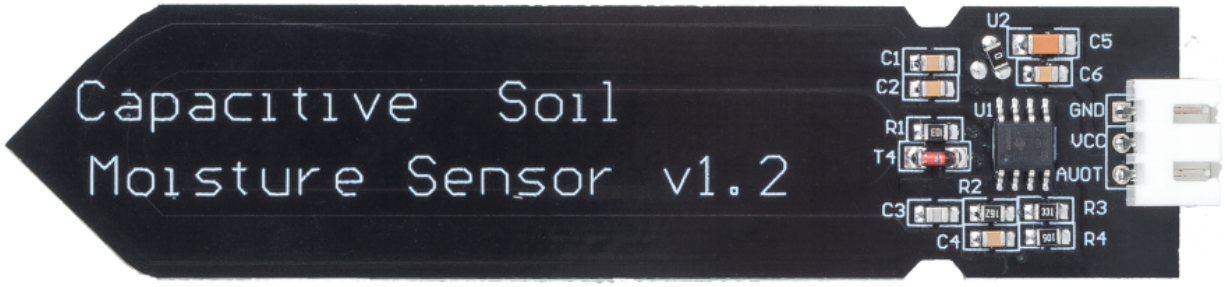
Merkmale

- Verwendung des Infrarotsensors TCRT5000
- Erfassungsabstand: 1-8 mm, Brennweite von 2,5 mm
- Komparatorausgangssignal klar, gute Wellenform, Treibfähigkeit größer als 15 mA
- Verwendung eines Potentiometers zur Empfindlichkeitseinstellung
- Betriebsspannung: 3,3 V-5 V
- Digitaler Ausgang: 0 (weiß) und 1 (schwarz)
- Verwendung eines breitspannungsfähigen LM393-Komparators.
- Größe: 42 mm x 10 mm

Beispiele

- *5.4 Linie Erkennen* (Arduino-Projekt)
- *5.4 Linie erkennen* (MicroPython-Projekt)
- *2.19 SPIEL - Schütze Dein Herz* (Scratch-Projekt)

5.30 Bodenfeuchtigkeitsmodul

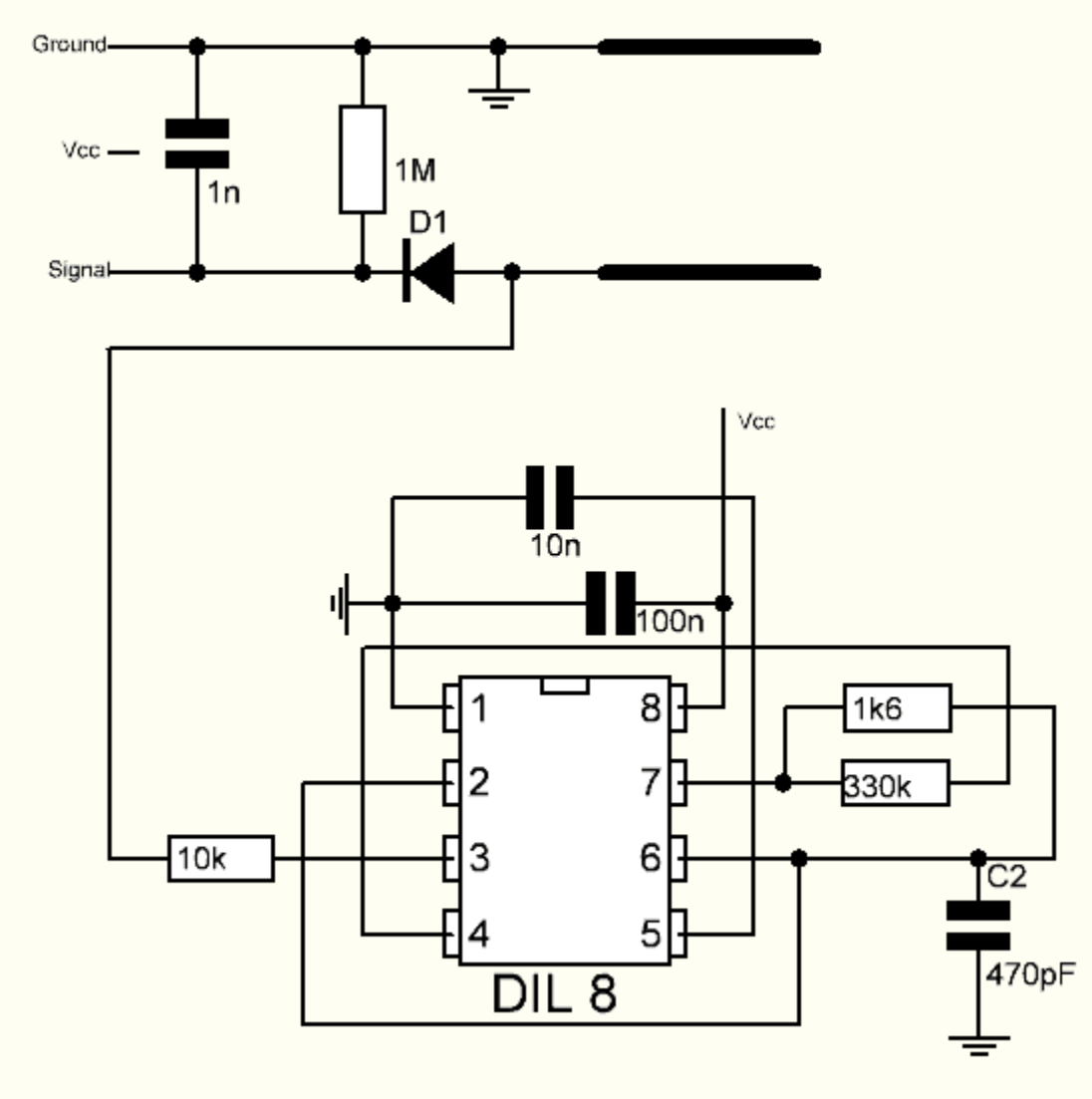


- GND: Masse
- VCC: Stromversorgung, 3.3V~5V
- AOUT: Gibt den Bodenfeuchtigkeitswert aus, je feuchter der Boden, desto kleiner der Wert.

Dieser kapazitive Bodenfeuchtigkeitssensor unterscheidet sich von den meisten auf dem Markt erhältlichen resistiven Sensoren. Er nutzt das Prinzip der kapazitiven Induktion zur Erkennung der Bodenfeuchtigkeit. Dadurch werden Probleme, wie die hohe Korrosionsanfälligkeit resistiver Sensoren, vermieden, was die Lebensdauer erheblich verlängert.

Er besteht aus korrosionsbeständigem Material und bietet eine hervorragende Lebensdauer. Einfach in den Boden rund um die Pflanzen stecken und Echtzeitdaten zur Bodenfeuchtigkeit überwachen. Das Modul enthält einen Spannungsregler an Bord, der es ihm ermöglicht, in einem Spannungsbereich von 3,3 ~ 5,5 V zu arbeiten. Es ist ideal für Mikrocontroller mit niedriger Spannung und Versorgungen von 3,3 V und 5 V.

Das Hardware-Schema des kapazitiven Bodenfeuchtigkeitssensors ist unten dargestellt.



Es gibt einen Festfrequenzoszillator, der mit einem 555-Timer-IC aufgebaut ist. Die erzeugte Rechteckwelle wird dann wie ein Kondensator an den Sensor geführt. Für das Rechteckwellensignal hat der Kondensator jedoch eine gewisse Reaktanz oder, um es einfach auszudrücken, einen Widerstand mit einem reinen ohmschen Widerstand (10k-Widerstand an Pin 3) bildet einen Spannungsteiler.

Je höher die Bodenfeuchtigkeit, desto höher die Kapazität des Sensors. Dementsprechend hat die Rechteckwelle weniger Reaktanz, was die Spannung auf der Signalleitung reduziert, und umso kleiner der Wert des analogen Eingangs durch den Mikrocontroller.

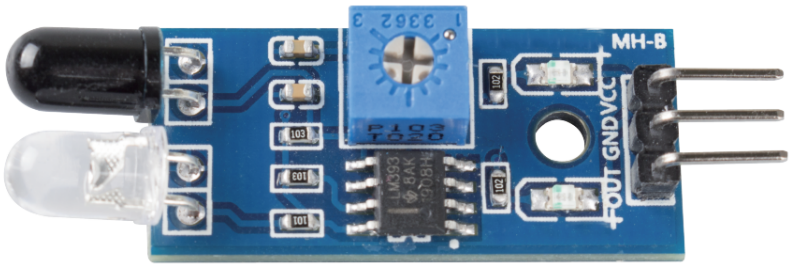
Spezifikation

- Betriebsspannung: 3,3 ~ 5,5 VDC
- Ausgangsspannung: 0 ~ 3,0VDC
- Betriebsstrom: 5mA
- Schnittstelle: PH2.0-3P
- Abmessungen: 98 x 23 mm (L x B)
- Gewicht: 15g

Beispiele

- *5.9 Bodenfeuchtigkeit messen* (Arduino-Projekt)
- *6.6 Pflanzenüberwachung* (Arduino-Projekt)
- *5.9 Messung der Bodenfeuchtigkeit* (MicroPython-Projekt)
- *6.8 Pflanzenüberwachung* (MicroPython-Projekt)

5.31 Modul zur Hindernisvermeidung

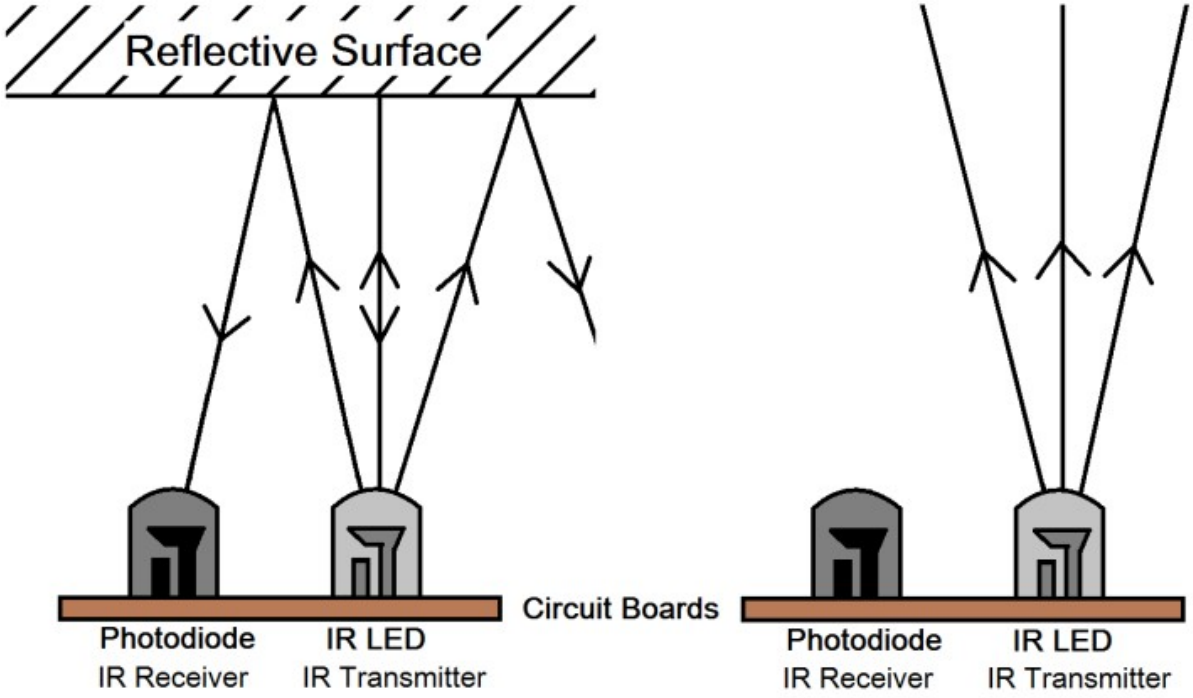


- **VCC**: Stromversorgung, 3,3 ~ 5V DC.
- **GND**: Masse
- **OUT**: Signalpin, normalerweise auf hohem Pegel, und auf niedrigem Pegel, wenn ein Hindernis erkannt wird.

Das IR-Hindernisvermeidungsmodul passt sich gut an die Umgebungslichtverhältnisse an und verfügt über ein Paar Infrarot-Sendungs- und Empfangsröhren.

Die Sendungsröhre strahlt Infrarotfrequenz aus, wenn in der Erkennungsrichtung ein Hindernis auftritt, wird die Infrarotstrahlung von der Empfangsröhre aufgenommen. Nach der Verarbeitung durch den Komparatorschaltkreis leuchtet die Anzeige auf und gibt ein Niedrigpegelsignal aus.

Die Erkennungsdistanz kann mittels eines Potentiometers eingestellt werden, der wirksame Distanzbereich liegt bei 2-30cm.



Beispiel

- *5.3 Hindernis Erkennen* (Arduino-Projekt)
- *5.3 Hinderniserkennung* (MicroPython-Projekt)
- *2.11 SPIEL - Schießen* (Scratch-Projekt)
- *2.18 SPIEL - Nicht auf das weiße Kachel tippen* (Scratch-Projekt)

5.32 Ultraschall-Modul

- **TRIG**: Trigger-Impuls-Eingang
- **ECHO**: Echo-Impuls-Ausgang
- **GND**: Masse

- **VCC**: 5V Versorgung

Dies ist der HC-SR04-Ultraschall-Distanzsensor, der berührungslose Messungen von 2 cm bis 400 cm mit einer Messgenauigkeit von bis zu 3 mm ermöglicht. Auf dem Modul befinden sich ein Ultraschall-Sender, ein Empfänger und eine Steuerschaltung.

Sie müssen nur 4 Pins anschließen: VCC (Strom), Trig (Auslöser), Echo (Empfang) und GND (Masse), um es für Ihre Messprojekte einfach zu verwenden.

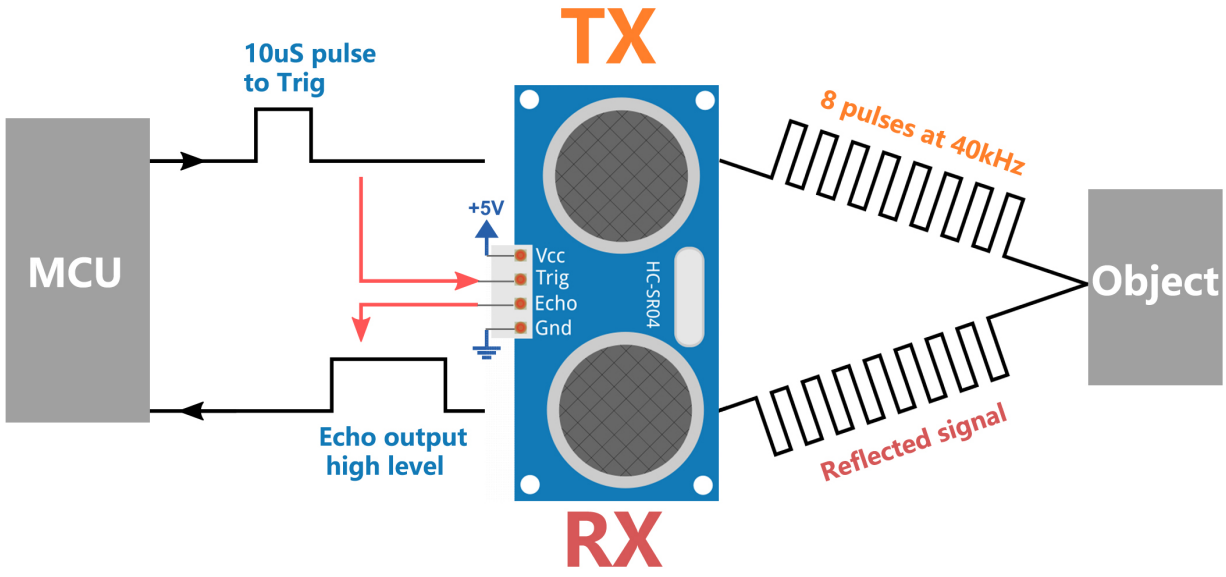
Merkmale

- Arbeitsspannung: DC5V
- Arbeitsstrom: 16mA
- Arbeitsfrequenz: 40Hz
- Maximale Reichweite: 500cm
- Min Reichweite: 2cm
- Trigger-Eingangssignal: 10uS TTL-Impuls
- Echo-Ausgangssignal: Eingang TTL Hebelsignal und die Reichweite im Verhältnis
- Anschluss: XH2.54-4P
- Abmessungen: 46x20,5x15 mm

Prinzip

Die Grundprinzipien sind wie folgt:

- Mit IO-Trigger für mindestens 10us High-Level-Signal.
- Das Modul sendet einen 8-Zyklen-Ultraschallburst mit 40 kHz und erkennt, ob ein Impulssignal empfangen wird.
- Das Echo gibt einen hohen Pegel aus, wenn ein Signal zurückkommt; die Dauer des hohen Pegels ist die Zeit von der Aussendung bis zur Rückkehr.
- Entfernung = (Hochpegelzeit x Schallgeschwindigkeit (340M/S)) / 2



Formel:

- us / 58 = Abstand in Zentimetern

- us / 148 = Zoll-Abstand
- Entfernung = Hochgeschwindigkeitszeit x Geschwindigkeit (340M/S) / 2

Bemerkung: Dieses Modul sollte nicht unter Strom angeschlossen werden, wenn nötig, lassen Sie die GND des Moduls zuerst angeschlossen werden. Andernfalls wird es die Arbeit des Moduls beeinträchtigen.

Die Fläche des zu messenden Objekts sollte mindestens 0,5 Quadratmeter betragen und möglichst flach sein. Andernfalls wird das Ergebnis beeinträchtigt.

Beispiel

- *5.12 Entfernung messen* (Arduino-Projekt)
- *6.3 Einparkhilfe* (Arduino-Projekt)
- *5.12 Distanzmessung* (MicroPython-Projekt)
- *6.4 Einparkhilfe* (MicroPython-Projekt)
- *2.15 SPIEL - Flappy Papagei* (Scratch-Projekt)

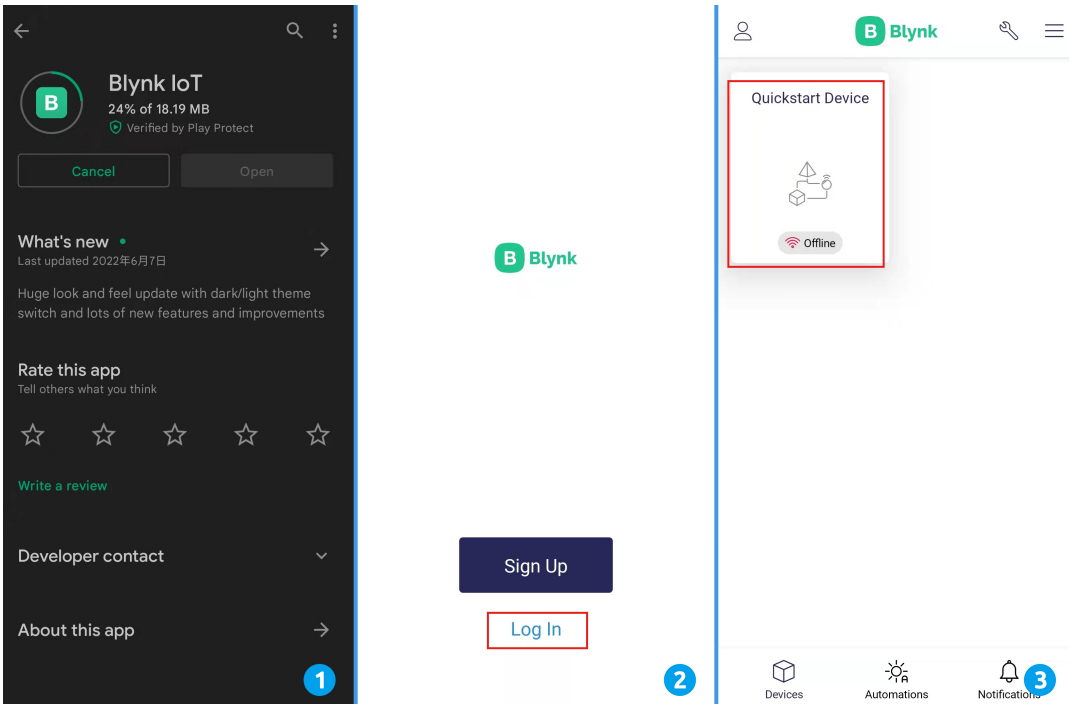
KAPITEL 6

FAQ

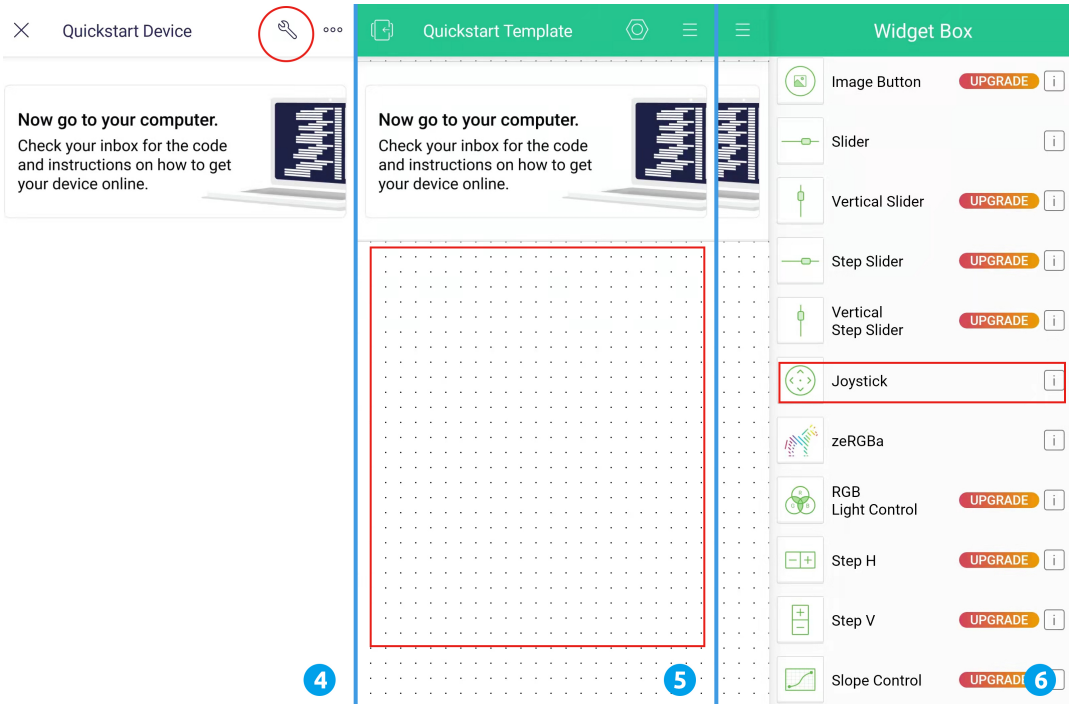
6.1 Wie verwendet man Blynk auf einem mobilen Gerät?

Bemerkung: Da Datenströme nur in Blynk im Web erstellt werden können, müssen Sie auf unterschiedliche Projekte im Web verweisen, um Datenströme zu erstellen, und dann der nachfolgenden Anleitung folgen, um Widgets in Blynk auf Ihrem mobilen Endgerät zu erstellen.

1. Öffnen Sie Google Play oder den APP Store auf Ihrem mobilen Gerät und suchen Sie nach „Blynk IoT“ (nicht Blynk (Legacy)), um es herunterzuladen.
2. Nachdem Sie die APP geöffnet haben, melden Sie sich an. Dieses Konto sollte dasselbe sein wie das Konto, das auf dem Webclient verwendet wird.
3. Gehen Sie dann zum **Dashboard** (wenn Sie noch keines haben, erstellen Sie eines), und Sie werden sehen, dass das **Dashboard** für Mobilgeräte und Web unabhängig voneinander sind.

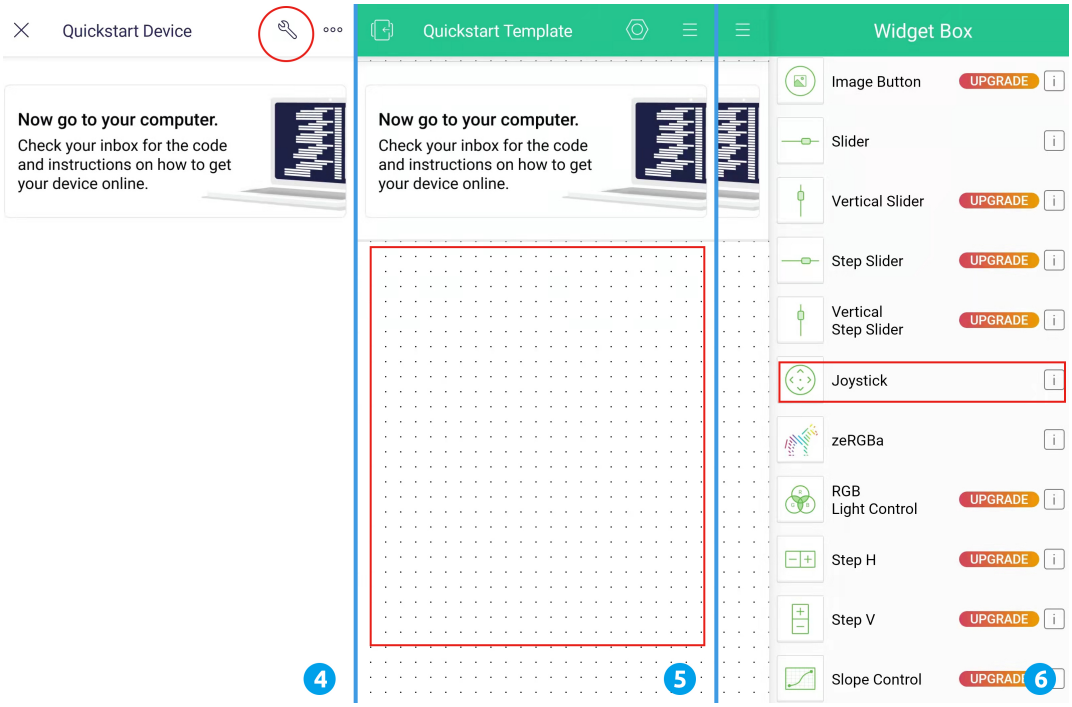


4. Klicken Sie auf das **Edit**-Symbol.
5. Klicken Sie auf den leeren Bereich.
6. Wählen Sie das gleiche Widget wie auf der Webseite aus, z.B. wählen Sie ein **Joystick**-Widget.



7. Jetzt sehen Sie ein **Joystick**-Widget im leeren Bereich, klicken Sie darauf.
8. Die Einstellungen für den **Joystick** werden angezeigt, wählen Sie die Datenströme **Xvalue** und **Yvalue**, die Sie gerade auf der Webseite eingestellt haben. Beachten Sie, dass jedes Widget einem anderen Datenstrom in jedem Projekt entspricht.

9. Gehen Sie zurück zur **Dashboard**-Seite, und Sie können den **Joystick** bedienen, wann immer Sie möchten.

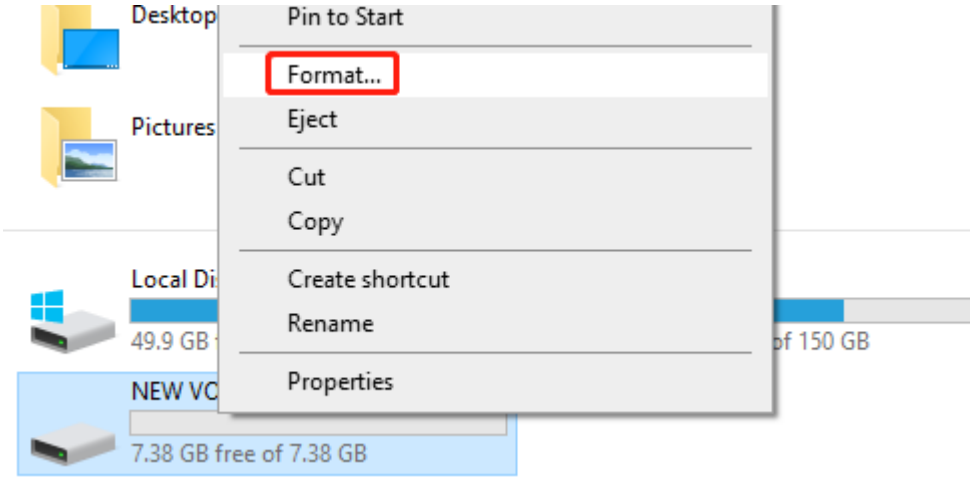


6.2 Wie formatiert man eine SD-Karte?

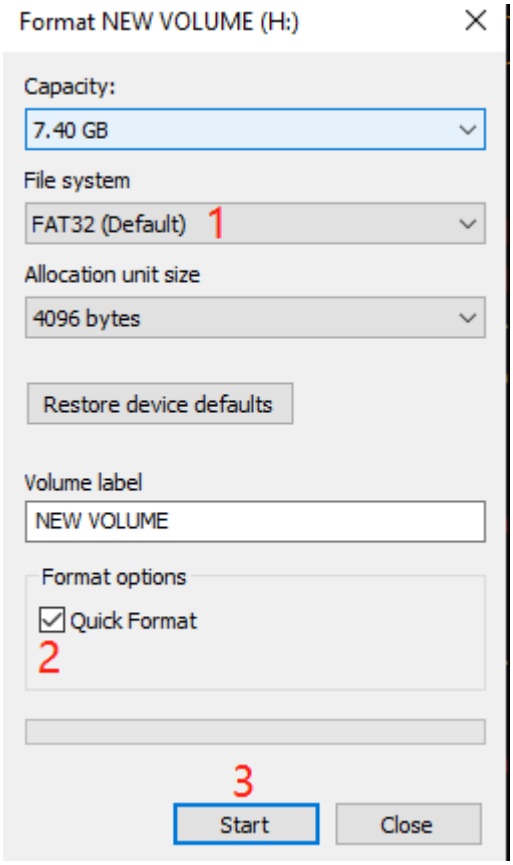
Die Schritte, um sicherzustellen, dass Ihre SD-Karte korrekt formatiert wird, können je nach Betriebssystem variieren. Hier sind einfache Anweisungen, wie man eine SD-Karte unter Windows, MacOS und Linux formatiert:

Windows

1. Legen Sie Ihre SD-Karte in den Computer ein, öffnen Sie dann „Mein Computer“ oder „Dieser PC“. Klicken Sie mit der rechten Maustaste auf Ihre SD-Karte und wählen Sie „Formatieren“.

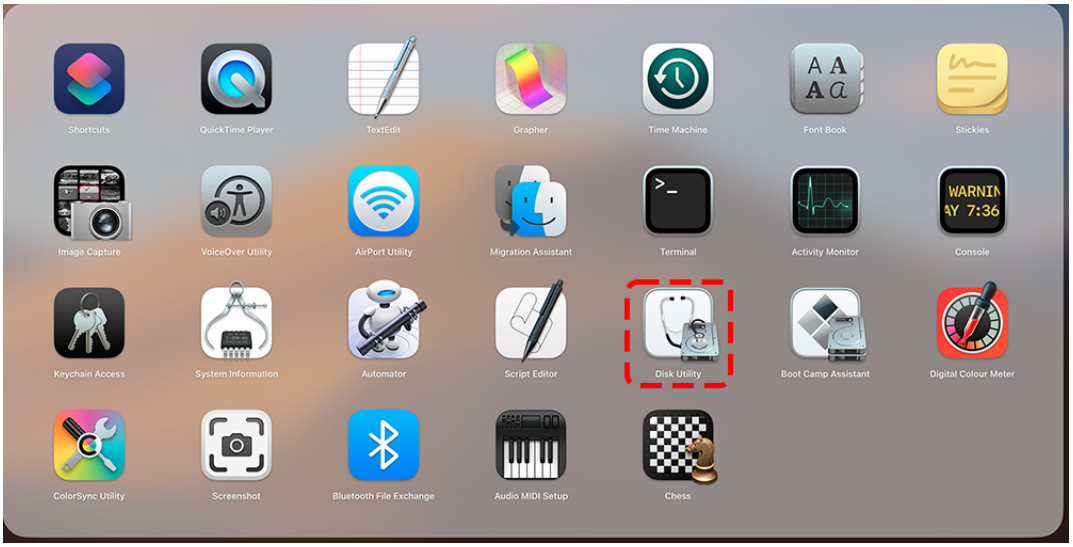


2. Wählen Sie im Dropdown-Menü des Dateisystems das gewünschte Dateisystem aus (normalerweise wählen Sie FAT32, oder für SD-Karten größer als 32 GB, müssen Sie möglicherweise exFAT wählen). Markieren Sie „Schnellformatierung“ und klicken Sie dann auf „Start“.

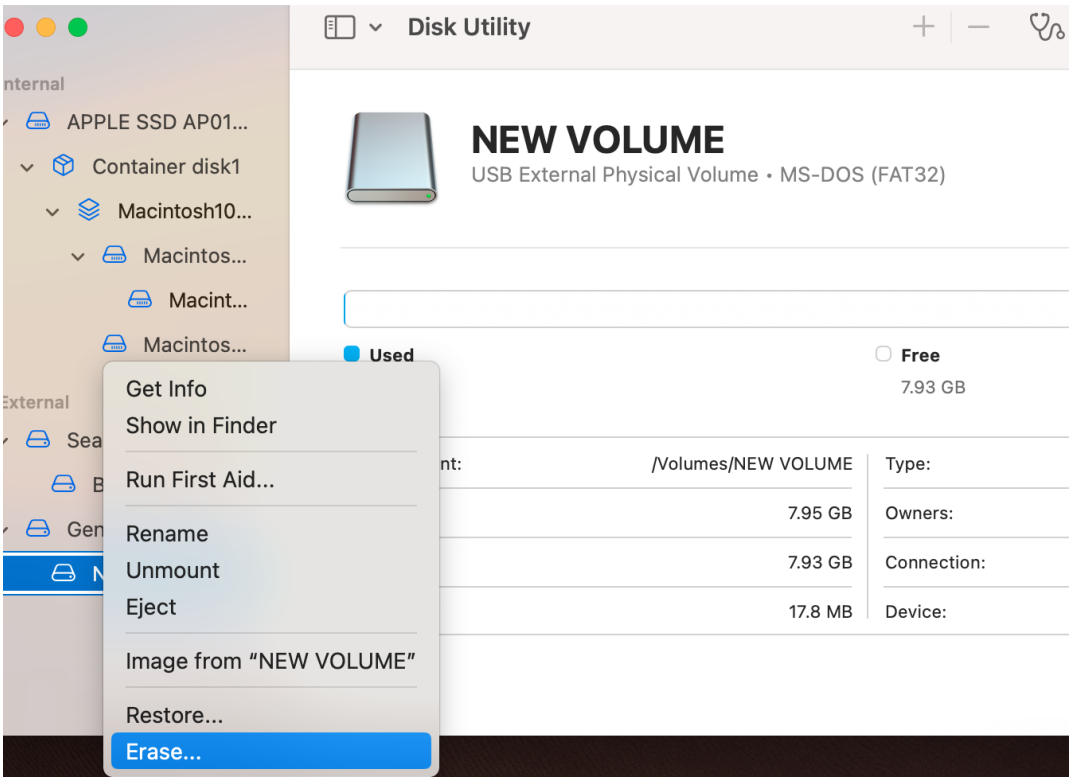


MacOS

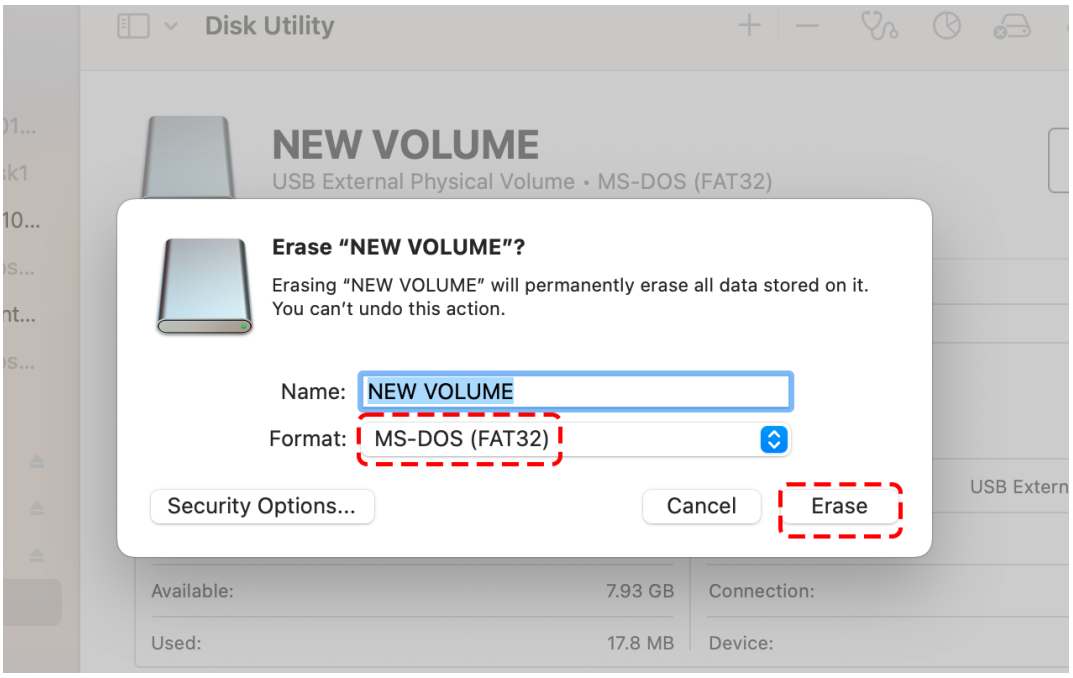
1. Legen Sie Ihre SD-Karte in den Computer ein. Öffnen Sie die Anwendung „Festplattendienstprogramm“ (zu finden im Ordner „Dienstprogramme“).



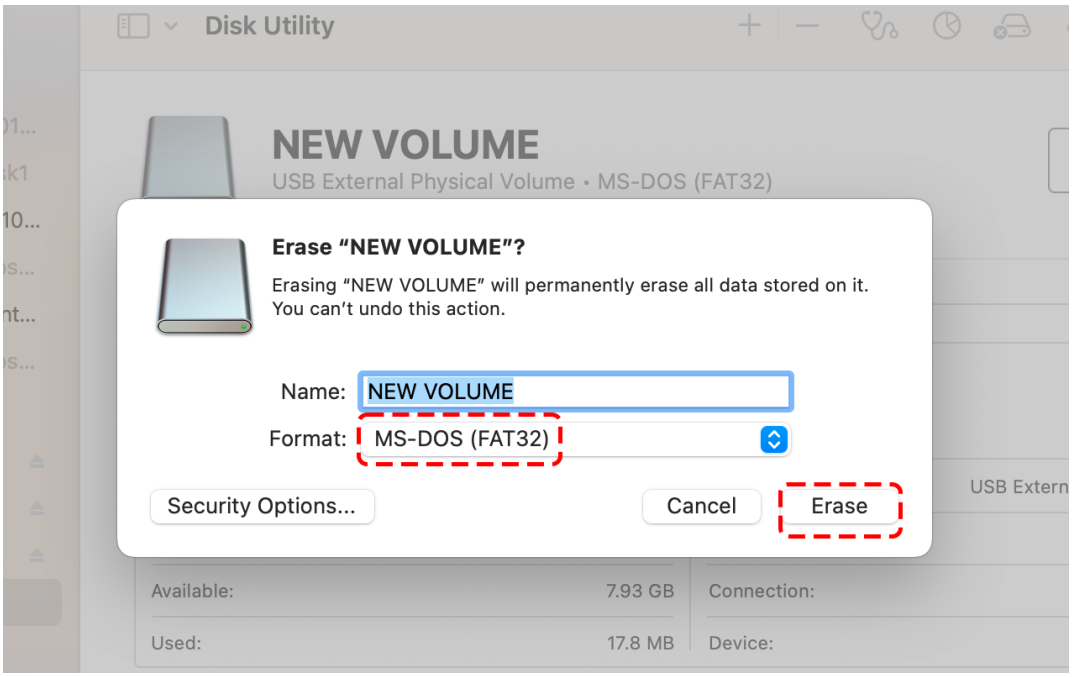
2. Wählen Sie Ihre SD-Karte aus der Liste links und klicken Sie dann auf „Löschen“.



3. Wählen Sie aus dem Dropdown-Menü Format Ihr gewünschtes Dateisystem aus (normalerweise wählen Sie MS-DOS (FAT) für FAT32, oder ExFAT für SD-Karten größer als 32 GB) und klicken Sie dann auf „Löschen“.



4. Warten Sie schließlich, bis die Formatierung abgeschlossen ist.



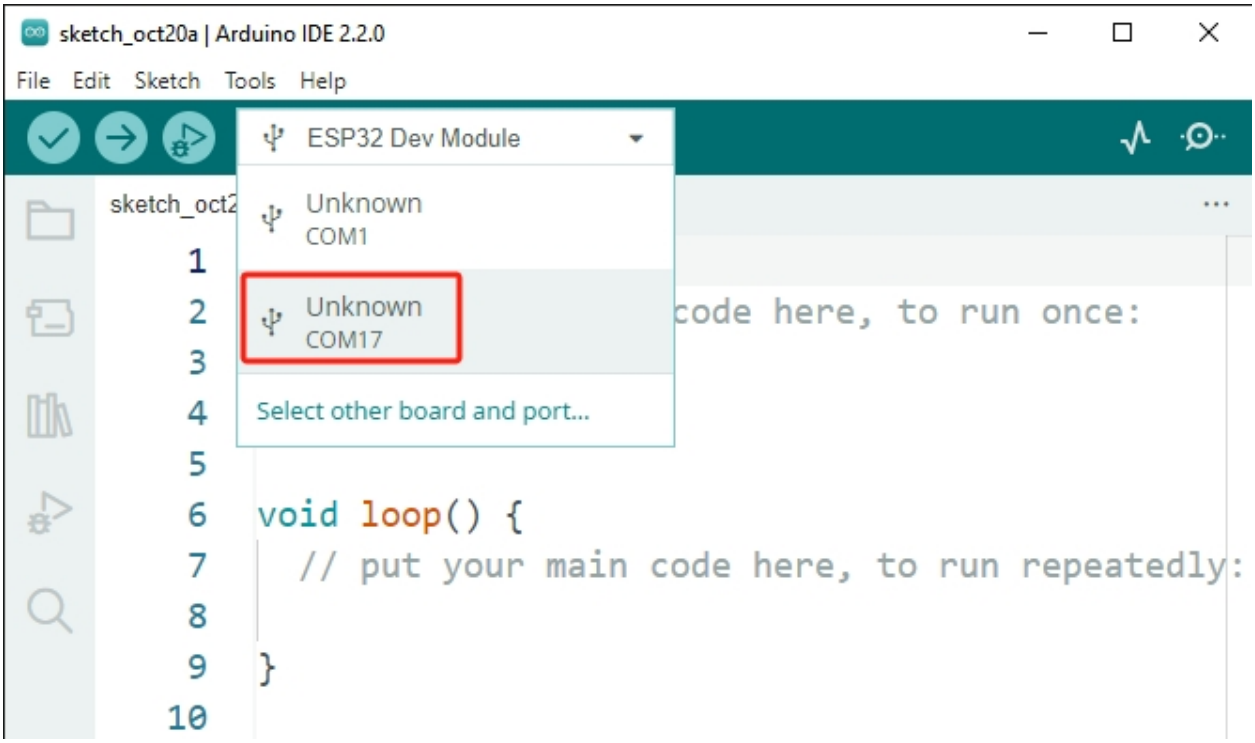
Linux

- Fügen Sie zuerst Ihre SD-Karte ein und öffnen Sie dann ein Terminal.
- Geben Sie `lsblk` ein und finden Sie den Namen Ihrer SD-Karte in der Geräteliste (z.B. könnte es `sdb` sein).
- Verwenden Sie den Befehl `umount`, um die SD-Karte auszuhängen, wie z.B. `sudo umount /dev/sdb*`.
- Verwenden Sie den Befehl `mkfs`, um die SD-Karte zu formatieren. Zum Beispiel wird `sudo mkfs.vfat /dev/sdb1` die SD-Karte mit einem FAT32-Dateisystem formatieren (für SD-Karten größer als 32 GB, müssen Sie möglicherweise `mkfs.exfat` verwenden).

Bevor Sie Ihre SD-Karte formatieren, stellen Sie sicher, dass Sie alle wichtigen Daten auf der SD-Karte sichern, da der Formatierungsvorgang alle Dateien auf der SD-Karte löschen wird.

6.3 „Unbekanntes COMxx“ wird immer angezeigt?

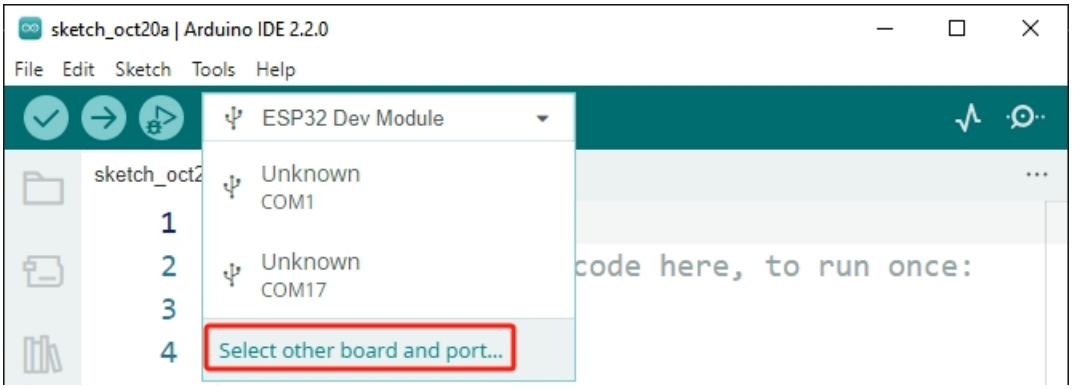
Wenn Sie den ESP32 an den Computer anschließen, zeigt die Arduino IDE oft `Unbekanntes COMxx` an. Warum passiert das?



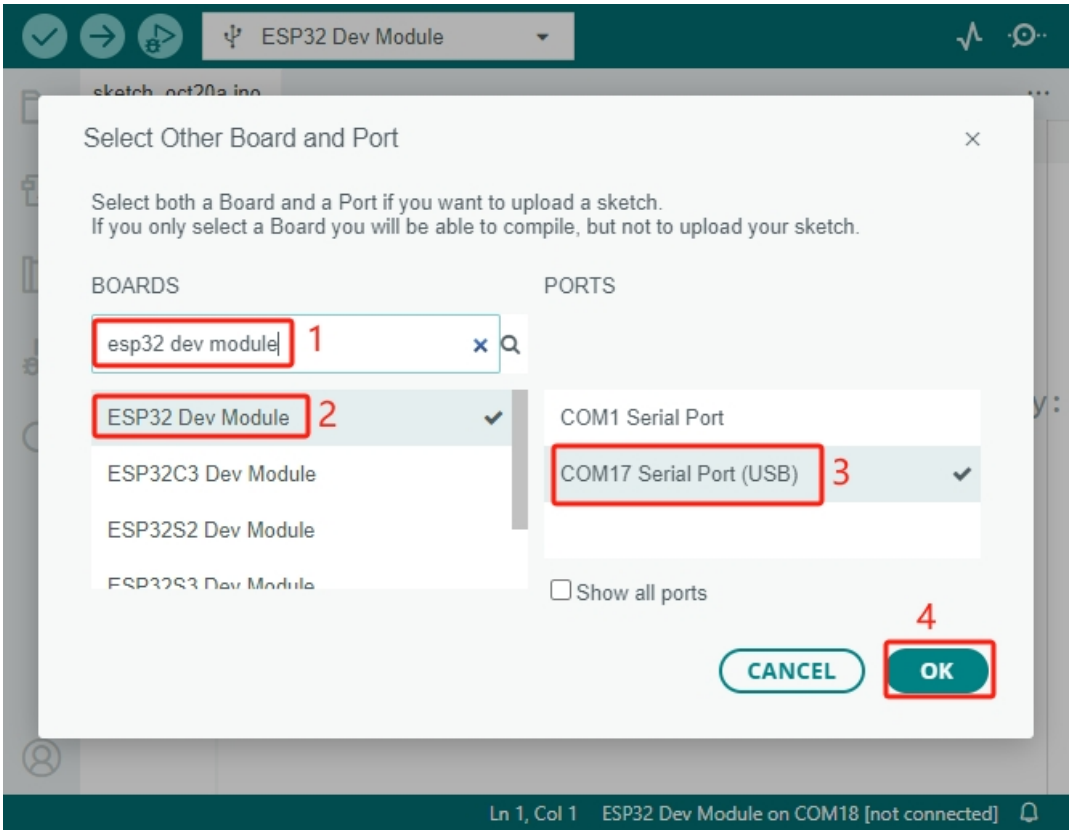
Dies liegt daran, dass der USB-Treiber für den ESP32 anders ist als bei den regulären Arduino-Boards. Die Arduino IDE kann dieses Board nicht automatisch erkennen.

In einem solchen Szenario müssen Sie das richtige Board manuell auswählen, indem Sie diese Schritte befolgen:

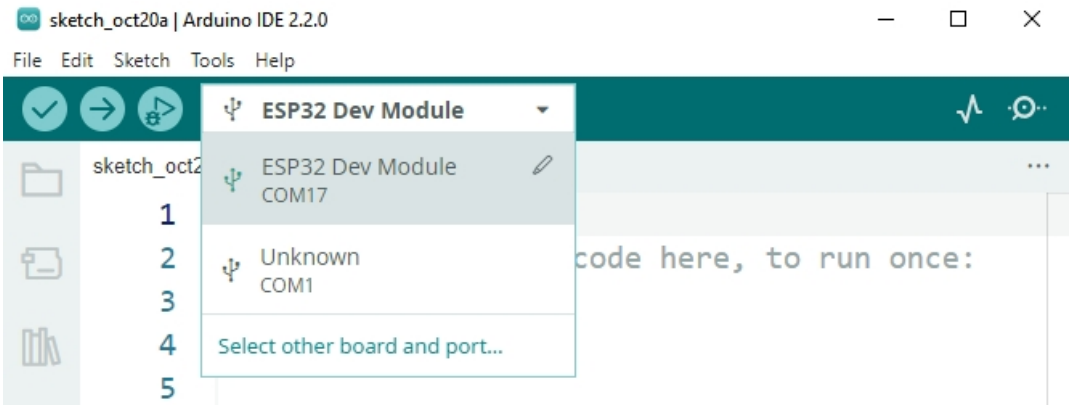
1. Klicken Sie auf **„Select the other board and port“**.



2. Geben Sie in der Suche **„esp32 dev module“** ein, wählen Sie dann das erscheinende Board aus. Wählen Sie anschließend den richtigen Port und klicken Sie auf **OK**.



3. Jetzt sollten Sie Ihr Board und Port in diesem Schnellansichtsfenster sehen können.



KAPITEL 7

Danke

Wir danken den Evaluatoren, die unsere Produkte bewertet haben, den Veteranen, die Vorschläge für das Tutorial gegeben haben, und den Benutzern, die uns stetig folgen und unterstützen. Ihre wertvollen Vorschläge sind unsere Motivation, bessere Produkte anzubieten!

Besonderer Dank

- Len Davisson
- Kalen Daniel
- Juan Delacosta

Könnten Sie sich bitte kurz Zeit nehmen, um diesen Fragebogen auszufüllen?

Bemerkung: Nach dem Absenden des Fragebogens, bitte nach oben scrollen, um die Ergebnisse anzusehen.

KAPITEL 8

Urheberrechtshinweis