SunFounder Ultimate Sensor Kit

www.sunfounder.com

06.02.2024

Inhaltsverzeichnis

Danke, dass Sie sich für unser Ultimate Sensor 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.

Enttäuscht von einem Online-Lernkit, das nur mit einer rudimentären Anleitung geliefert wurde? Schon immer den Wunsch gehabt, in die Welt der Elektronik einzutauchen, aber von der Komplexität abgeschreckt?

Fasziniert von genialen Elektronikprojekten, die von Enthusiasten vorgeführt werden, und sich gefragt, wie sie angefangen haben?

Wir präsentieren: Das Ultimate Sensor Kit mit Arduino Uno R4 Minima. Die Lösung für all Ihre kreativen Herausforderungen in einem kompakten Paket.

Angetrieben vom revolutionären Arduino R4 leitet dieses Kit eine neue Ära für Open-Source-Begeisterte ein. Mit einem leistungsstarken 32-Bit-Prozessor, reichlich Speicher und mehr ist es darauf ausgelegt, sich nahtlos in neue wie bestehende Projekte einzufügen.

Es enthält nicht nur über 30 hochmoderne Module wie den Ultraschallsensor, Flammensensor, Beschleunigungsmesser & Gyroskop und sogar den Pulsoximeter und Herzfrequenzsensor, sondern jede Komponente wird mit einem anfängerfreundlichen Code-Beispiel gepaart. So besitzen Sie nicht nur die Werkzeuge, sondern auch das nötige Grundwissen.

Tauchen Sie tief in die Welt des IoT ein mit maßgeschneiderten Projekten, die den Arduino mit Plattformen wie Blynk über das ESP8266 WiFi-Modul verbinden. Entfesseln Sie Ihre Innovationskraft durch die Gestaltung von Projekten wie dem Flammenwarnsystem, dem Einbruchswarnsystem und sogar einem Bluetooth-gesteuerten Umweltmonitor.

Aber das ist noch nicht alles; stürzen Sie sich in eine Vielzahl spannender Projekte! Bauen Sie einen intelligenten Mülleimer, entwickeln Sie einen automatischen Seifenspender oder vielleicht ein bewegungsgesteuertes Relais. Das Kit erweitert Ihren Horizont und ermöglicht es Ihnen, Ihre Vorstellungskraft in greifbare Kreationen umzusetzen.

Hier geht es nicht nur darum, Schritte zu befolgen; es geht um Verständnis, Experimentieren und Erfinden. Anstatt lediglich zu replizieren, werden Sie Projekte gestalten, die einzigartig Ihr Eigen sind.

Warum nur Zuschauer sein, wenn Sie auch Schöpfer sein können? Beginnen Sie Ihre Reise in die fesselnde Welt der Elektronik mit dem Ultimate Sensor Kit. Ihr Abenteuer beginnt hier!

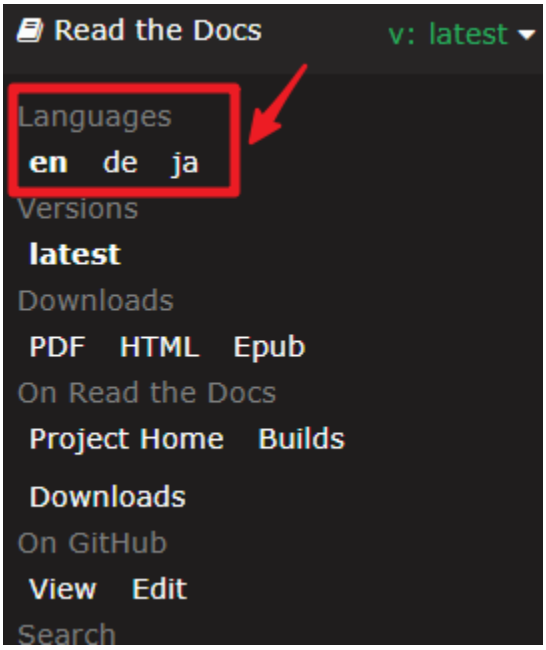
Bei Fragen oder weiteren interessanten Ideen können Sie gerne eine E-Mail an service@sunfounder.com senden.

KAPITEL 1

Über die Anzeigesprache

Bemerkung: Neben Englisch arbeiten wir an weiteren Sprachen für diesen Kurs. Wenn Sie Interesse an einer Mitarbeit haben, kontaktieren Sie bitte service@sunfounder.com, und wir werden Ihnen als Gegenleistung ein kostenloses Produkt zukommen lassen.

Aktuell werden im Online-Tutorial die Sprachen Englisch, Deutsch und Japanisch unterstützt. Um die Anzeigesprache zu ändern, klicken Sie bitte auf das Symbol **Read the Docs** in der unteren linken Ecke der Seite.



KAPITEL 2

Inhaltsverzeichnis

2.1 Einstieg in Arduino

Falls Sie keine Ahnung von Arduino haben, gibt es einige Begriffe, die ich Ihnen nahebringen möchte: Elektronik, Design, Programmierung und sogar Maker-Kultur. Manche könnten denken, dass diese Wörter weit entfernt von uns sind, aber in Wahrheit sind sie ganz nah. Denn Arduino kann uns in die Welt der Programmierung entführen und uns den Traum erfüllen, ein Maker zu werden. In dieser Sitzung lernen wir:

- Was ist Arduino?
- Was kann Arduino tun?
- Wie erstelle ich ein Arduino-Projekt?

2.1.1 Was ist Arduino?

Zuerst einmal eine kurze Einführung in Arduino.

Arduino ist eine praktische, flexible und benutzerfreundliche Open-Source-Plattform für den elektronischen Prototypenbau, bestehend aus verschiedenen Hardware-Arduino-Boards und der Software Arduino IDE. Es eignet sich nicht nur für Ingenieure zur schnellen Prototypenentwicklung, sondern auch für Künstler, Designer und Hobbyisten und ist fast schon ein Muss für moderne Maker.

Arduino ist ein weitreichendes System. Es umfasst Software, Hardware und eine sehr große Online-Community von Menschen, die sich zwar nie getroffen haben, aber dank eines gemeinsamen Hobbys zusammenarbeiten können. Jeder in der Arduino-Familie bringt seine Weisheit ein, schafft mit den Händen und teilt eine großartige Erfindung nach der anderen. Und auch Sie können ein Teil davon sein.

2.1.2 Was kann Arduino tun?

Vielleicht fragen Sie sich, was Arduino eigentlich leisten kann. Kurz gesagt, Arduino wird all Ihre Probleme lösen.

Technisch betrachtet ist Arduino ein programmierbarer Logikcontroller. Es handelt sich um ein Entwicklungsboard, mit dem sich viele aufregende und kreative elektronische Kreationen realisieren lassen: von ferngesteuerten Autos und Roboterarmen über bionische Roboter bis hin zu intelligenten Haushaltslösungen und mehr.

Arduino-Boards sind unkompliziert, einfach und leistungsstark und eignen sich für Schüler, Maker und sogar professionelle Programmierer.

Bis heute entwickeln Elektronikbegeisterte weltweit weiterhin kreative elektronische Projekte auf Basis von Arduino-Entwicklungsboards.

2.1.3 Wie erstelle ich ein Arduino-Projekt?

Folgen Sie diesen Schritten, um von Grund auf mit Arduino zu beginnen!

Download und Installation der Arduino IDE 2.0

Die Arduino IDE, bekannt als integrierte Entwicklungsumgebung von Arduino, bietet die gesamte Softwareunterstützung, die für die Durchführung eines Arduino-Projekts erforderlich ist. Sie ist eine speziell für Arduino konzipierte Programmiersoftware, die vom Arduino-Team bereitgestellt wird und es uns ermöglicht, Programme zu schreiben und auf das Arduino-Board hochzuladen.

Die Arduino IDE 2.0 ist ein Open-Source-Projekt. Sie stellt einen großen Fortschritt gegenüber ihrem robusten Vorgänger, der Arduino IDE 1.x, dar und kommt mit überarbeiteter Benutzeroberfläche, verbessertem Board- & Bibliotheksmanager, Debugger, Autovervollständigungsfunktion und vielem mehr.

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

Systemanforderungen

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

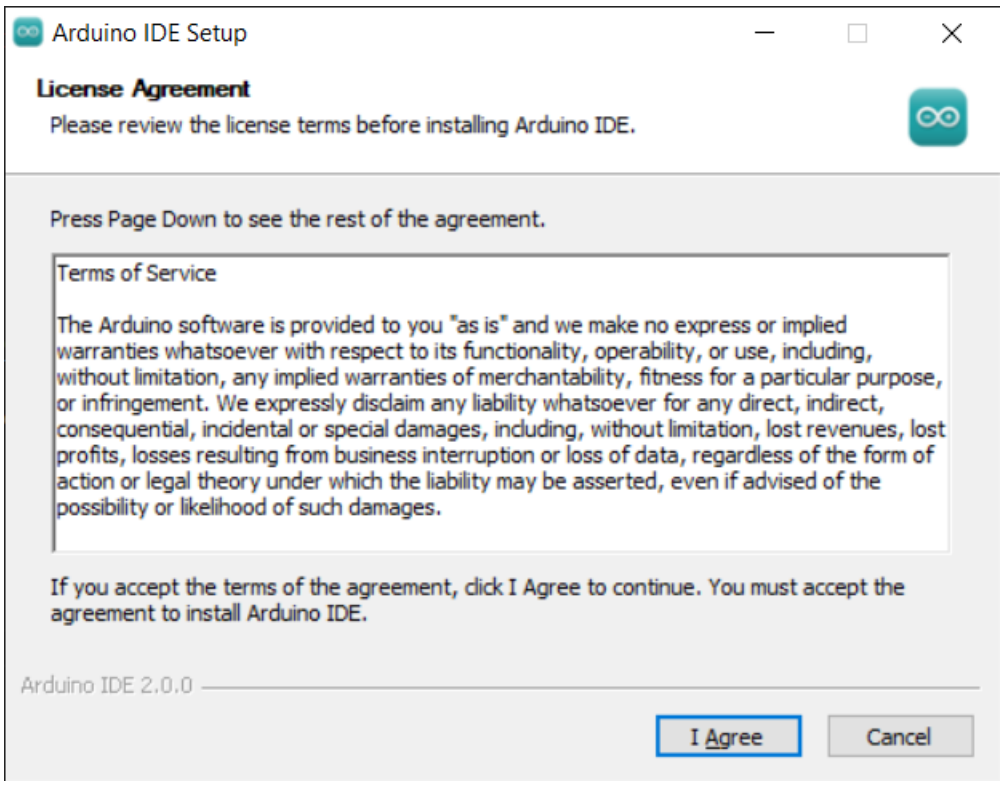
Arduino IDE 2.0 herunterladen

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

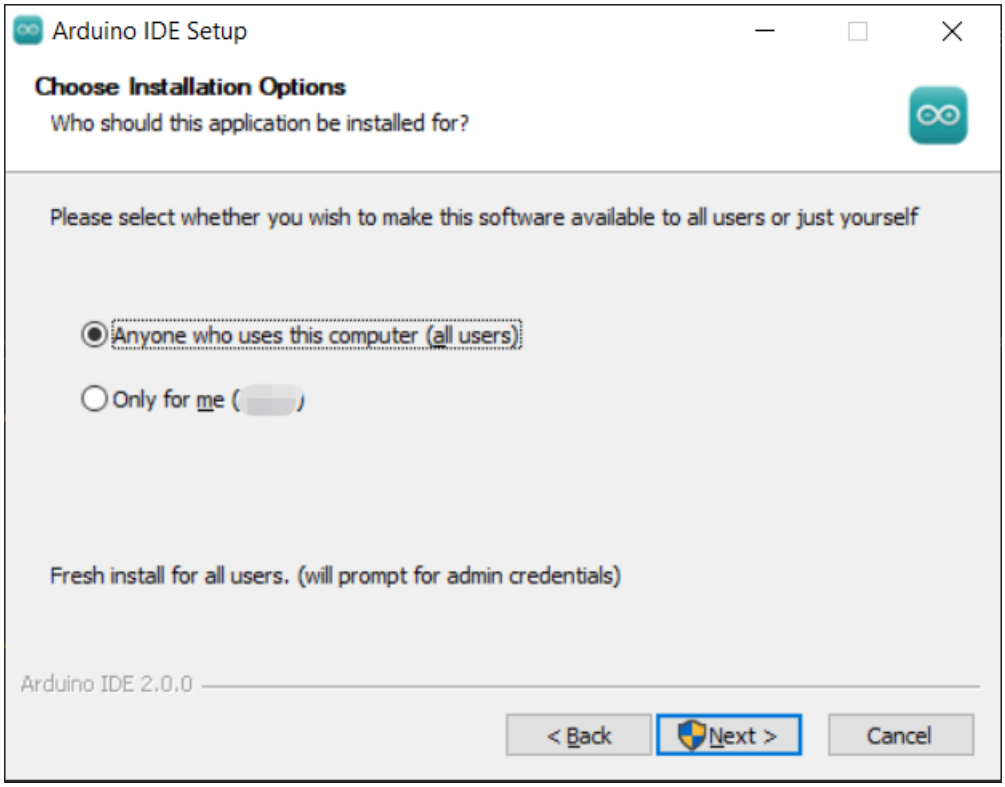
Installation

Windows

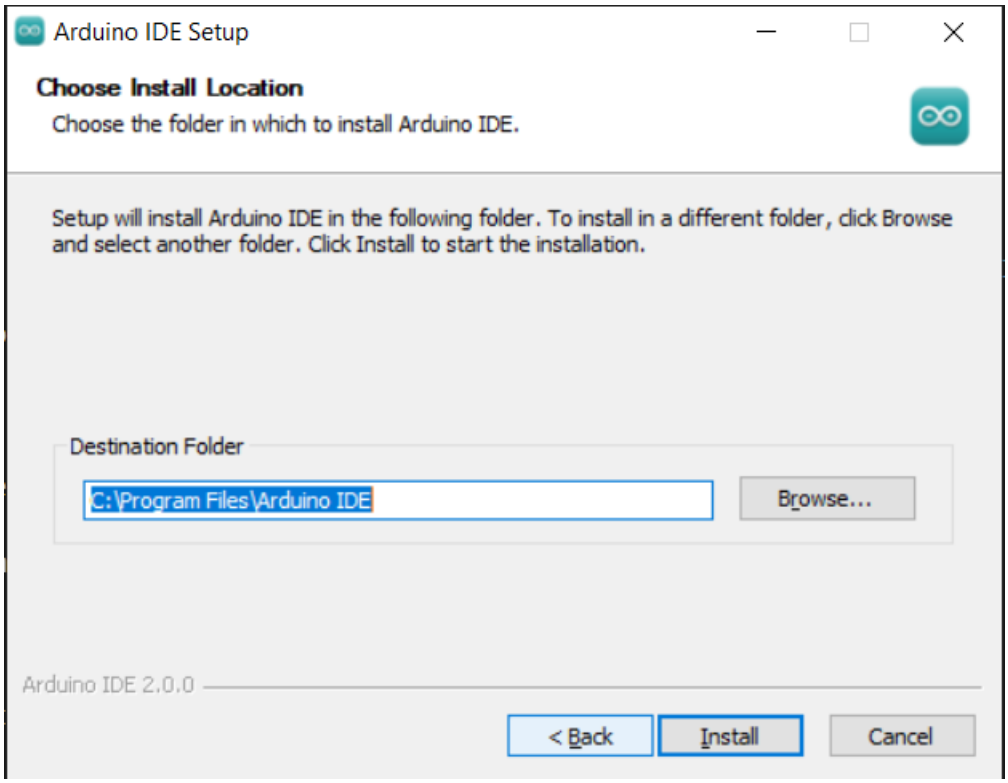
1. Doppelklicken Sie auf die Datei `arduino-ide_xxxx.exe`, um die heruntergeladene Datei auszuführen.
2. Lesen Sie die Lizenzvereinbarung und akzeptieren Sie diese.



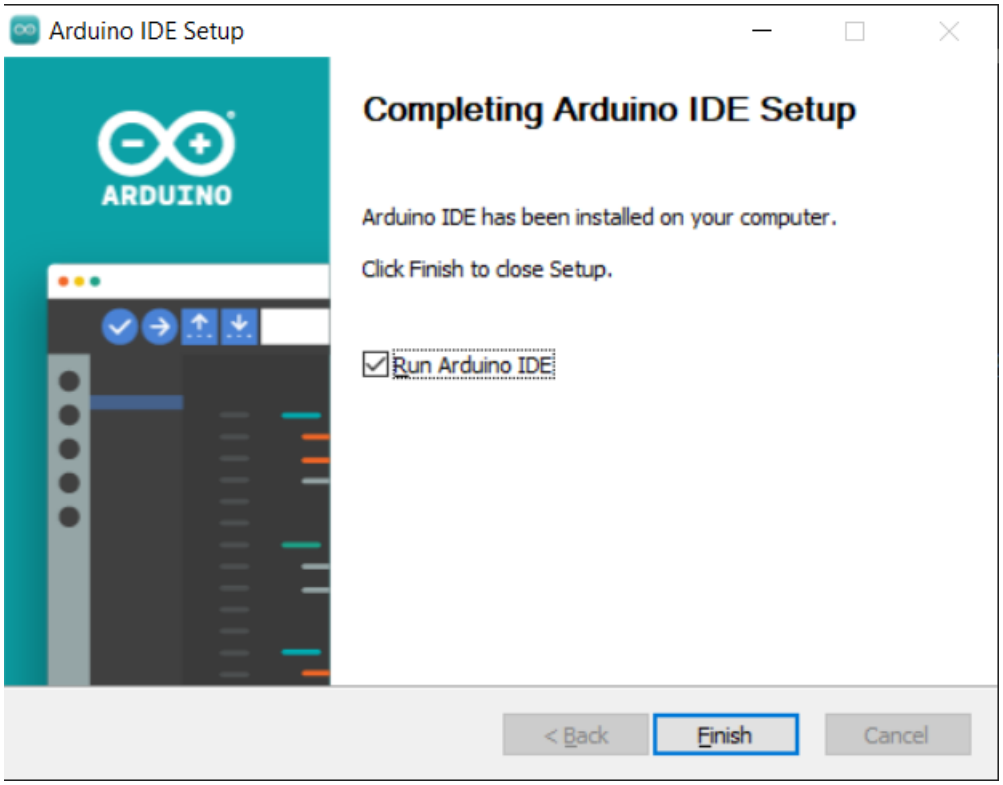
3. Wählen Sie die Installationsoptionen.



4. Wählen Sie den Installationsort. Es wird empfohlen, die Software auf einem Laufwerk zu installieren, das nicht das Systemlaufwerk ist.

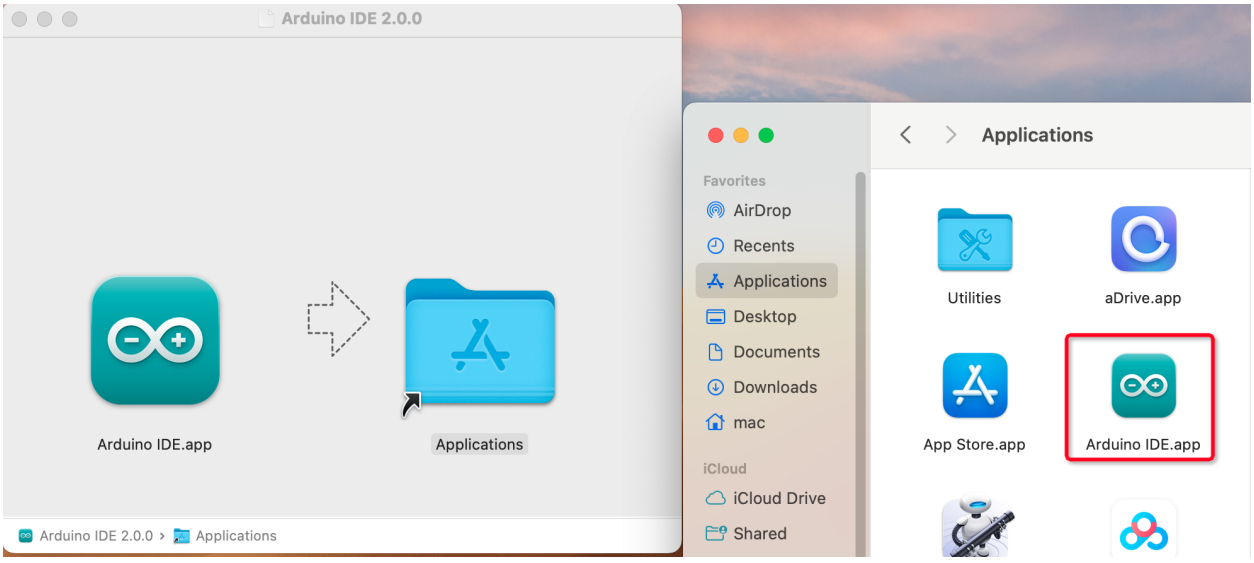


5. Dann abschließen.



macOS

Doppelklicken Sie auf die heruntergeladene Datei `arduino_ide_xxxx.dmg` und folgen Sie den Anweisungen, um die **Arduino IDE.app** in den **Anwendungen**-Ordner zu kopieren. Nach wenigen Sekunden sehen Sie, dass die Arduino IDE erfolgreich installiert wurde.

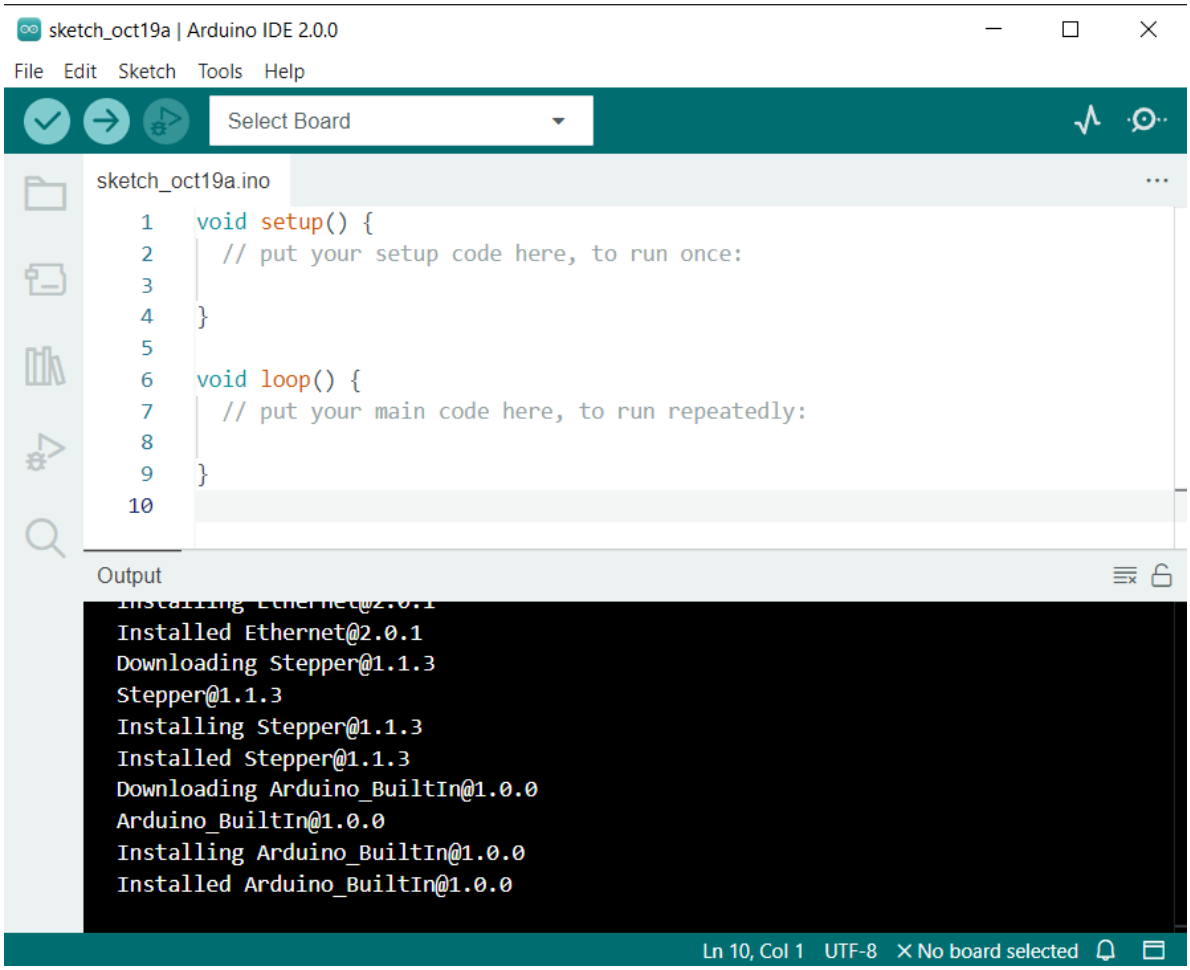


Linux

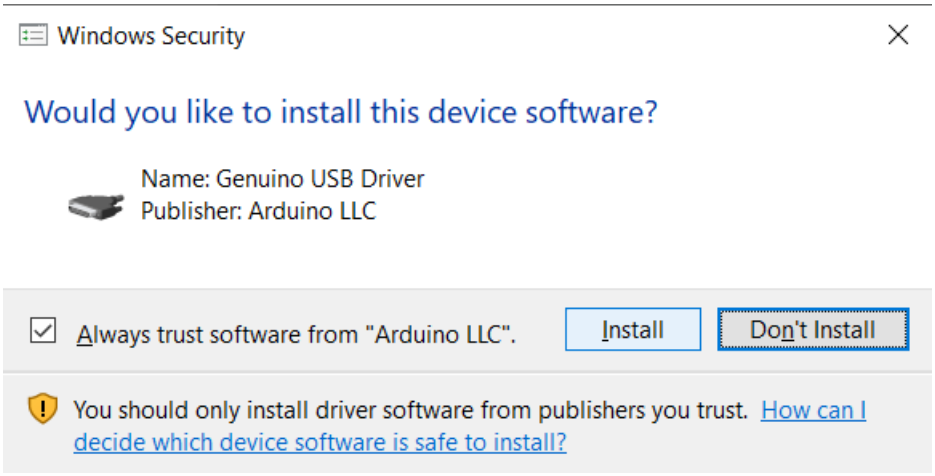
Für das Tutorial zur Installation der Arduino IDE 2.0 auf einem Linux-System folgen Sie bitte .

Die IDE öffnen

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.



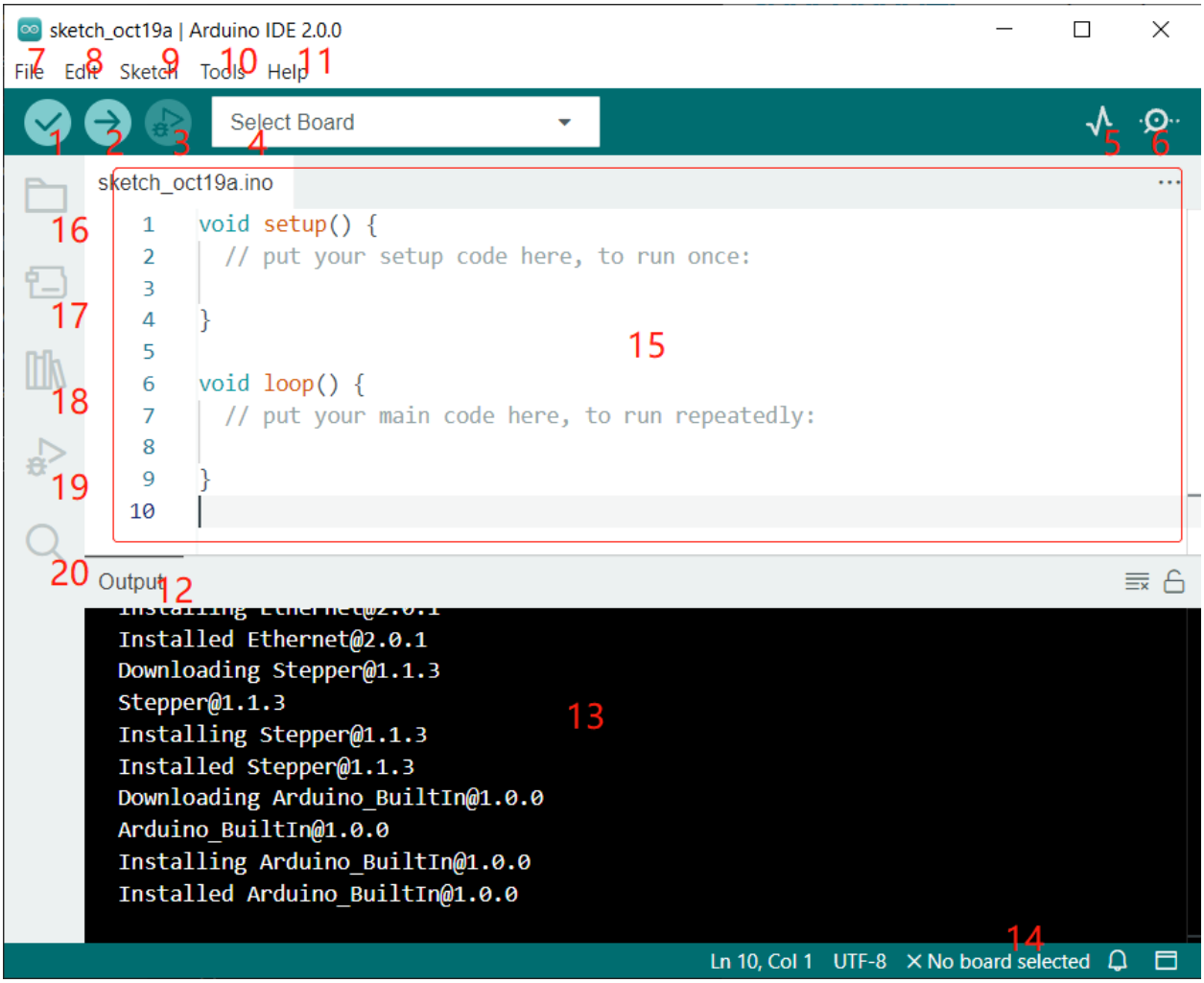
2. Darüber hinaus kann Ihr Firewall oder Sicherheitscenter mehrmals fragen, ob Sie einige Gerätetreiber installieren möchten. Bitte installieren Sie alle.



3. Jetzt ist Ihre Arduino IDE einsatzbereit!

Bemerkung: Falls einige Installationen aufgrund von Netzwerkproblemen oder aus anderen Gründen nicht funktioniert haben, können Sie die Arduino IDE erneut öffnen und der Rest der Installation wird abgeschlossen. Das Ausgabefenster öffnet sich nicht automatisch nach Abschluss aller Installationen, es sei denn, Sie klicken auf Überprüfen oder Hochladen.

Einführung in die Arduino IDE

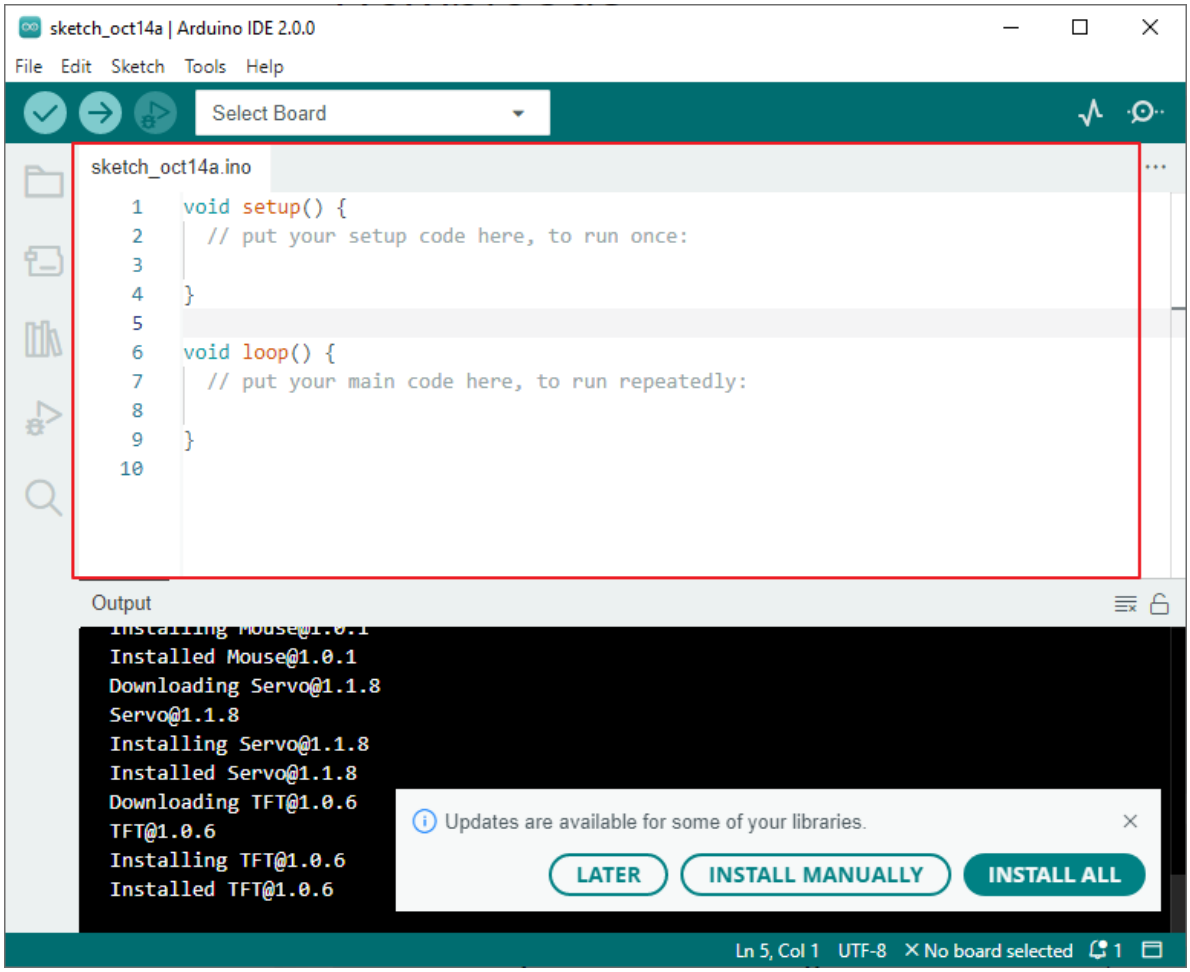


1. **Überprüfen (Verify)**: Kompiliert Ihren Code. Syntaxfehler werden durch Fehlermeldungen angezeigt.
2. **Hochladen (Upload)**: Lädt den Code auf Ihr Board. Beim Klicken auf den Button blinken die RX- und TX-LEDs auf dem Board schnell und hören erst auf, wenn der Upload abgeschlossen ist.
3. **Debuggen (Debug)**: Für zeilenweise Fehlerüberprüfung.
4. **Board wählen (Select Board)**: Schnelle Einrichtung von Board und Port.
5. **Serieller Plotter (Serial Plotter)**: Überwacht die Veränderungen der ausgelesenen Werte.
6. **Serieller Monitor (Serial Monitor)**: Ein Klick auf den Button öffnet ein Fenster, das die vom Steuerboard gesendeten Daten empfängt. Sehr nützlich zur Fehlerbehebung.
7. **Datei (File)**: Klicken Sie auf das Menü, und eine Dropdown-Liste erscheint mit Optionen zur Dateierstellung, -öffnung, -speicherung, -schließung und weiteren Konfigurationsmöglichkeiten.
8. **Bearbeiten (Edit)**: Klicken Sie auf das Menü. In der Dropdown-Liste gibt es Bearbeitungsoptionen wie **Ausschneiden**, **Kopieren**, **Einfügen**, **Suchen** usw. mit ihren entsprechenden Tastenkombinationen.
9. **Skizze (Sketch)**: Beinhaltet Funktionen wie **Überprüfen**, **Hochladen**, **Dateien hinzufügen** usw. Eine besonders wichtige Funktion ist **Bibliothek einbinden (Include Library)**, wo Sie Bibliotheken hinzufügen können.

10. **Werkzeuge (Tool)**: Enthält verschiedene Tools – am häufigsten verwendet sind Board (das verwendete Board) und Port (der Port, an dem Ihr Board angeschlossen ist). Bevor Sie den Code hochladen, müssen Sie diese auswählen oder überprüfen.
11. **Hilfe (Help)**: Wenn Sie Anfänger sind, finden Sie hier im Menü die benötigte Unterstützung, einschließlich Bedienung der IDE, Einführungsinformationen, Fehlerbehebung, Codeerklärungen usw.
12. **Ausgabebereich (Output Bar)**: Hier können Sie die Ausgaberegisterkarte wechseln.
13. **Ausgabefenster (Output Window)**: Zeigt Informationen an.
14. **Board und Port**: Hier können Sie eine Vorschau des ausgewählten Boards und Ports für den Code-Upload sehen. Bei Unstimmigkeiten können Sie diese erneut über **Werkzeuge (Tools)** -> **Board / Port** auswählen.
15. Der Bearbeitungsbereich der IDE. Hier können Sie Ihren Code schreiben.
16. **Skizzenbuch (Sketchbook)**: Zur Verwaltung von Skizzendateien.
17. **Board-Verwaltung (Board Manager)**: Zur Verwaltung der Board-Treiber.
18. **Bibliotheksverwalter (Library Manager)**: Zur Verwaltung Ihrer Bibliotheksdateien.
19. **Debuggen**: Unterstützt das Debugging des Codes.
20. **Suchen (Search)**: Durchsucht die Codes Ihrer Skizzen.

So erstellen, öffnen oder speichern Sie eine Skizze

1. Wenn Sie die Arduino IDE zum ersten Mal öffnen oder eine neue Skizze erstellen, erscheint eine Seite wie diese. Hier erstellt die Arduino IDE automatisch eine neue Datei für Sie, die als „Skizze“ bezeichnet wird.

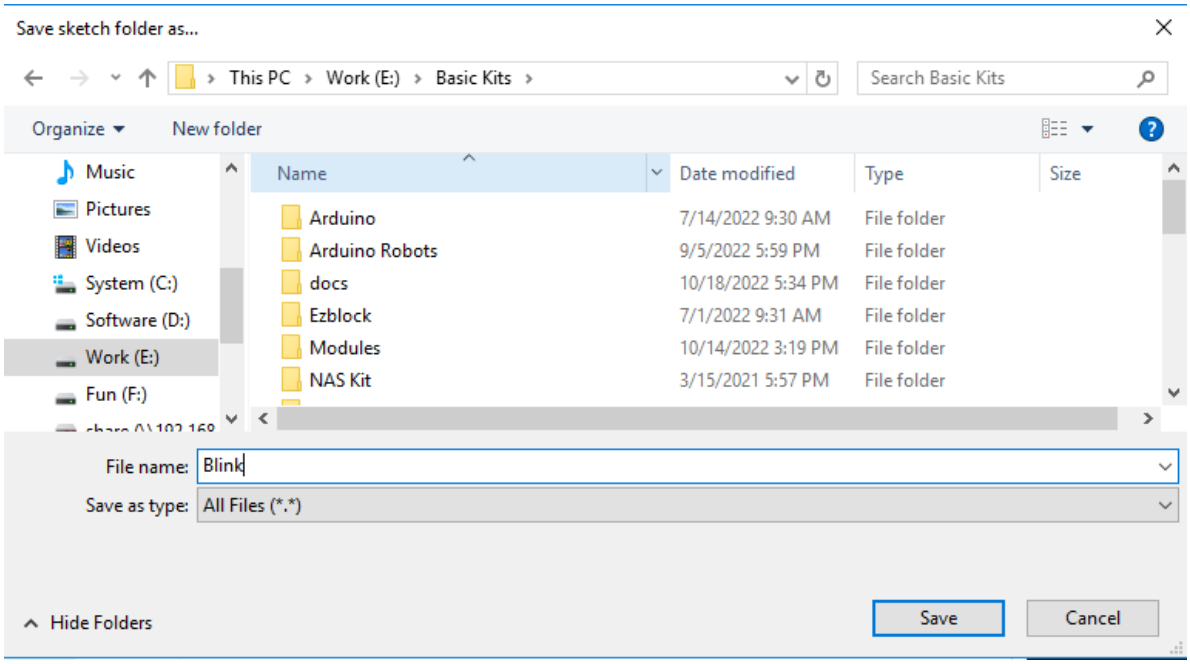


Diese Skizzendateien haben einen regulären temporären Namen, an dem Sie das Erstellungsdatum der Datei erkennen können. `sketch_oct14a.ino` steht für die erste Skizze vom 14. Oktober, und `.ino` ist das Dateiformat dieser Skizze.

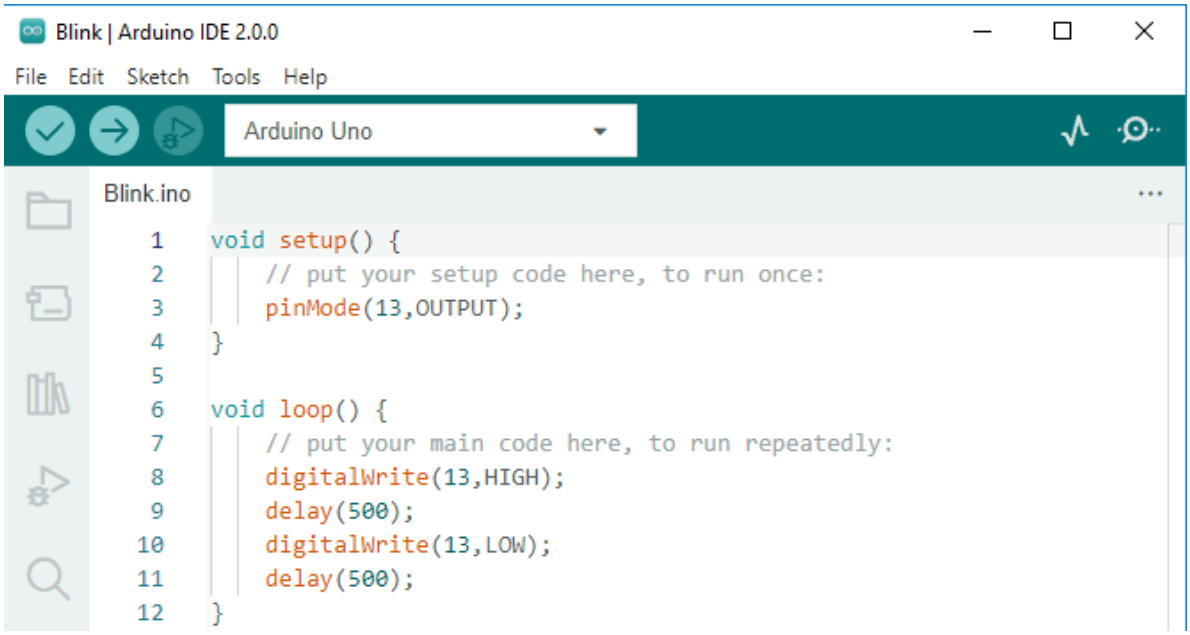
2. Lassen Sie uns nun eine neue Skizze erstellen. Kopieren Sie den folgenden Code in die Arduino IDE, um den ursprünglichen Code zu ersetzen.


```
void setup() {
    // put your setup code here, to run once:
    pinMode(13,OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    digitalWrite(13,HIGH);
    delay(500);
    digitalWrite(13,LOW);
    delay(500);
}
```

3. Drücken Sie `Strg+S` oder klicken Sie auf **Datei** -> **Save**. Die Skizze wird standardmäßig unter `C:\Users\{Ihr_Benutzername}\Documents\Arduino` gespeichert. Sie können den Namen ändern oder einen neuen Speicherort wählen.



4. Nach erfolgreichem Speichern wird der Name in der Arduino IDE aktualisiert.



Bitte fahren Sie mit dem nächsten Abschnitt fort, um zu erfahren, wie Sie diese erstellte Skizze auf Ihr Arduino-Board hochladen können.

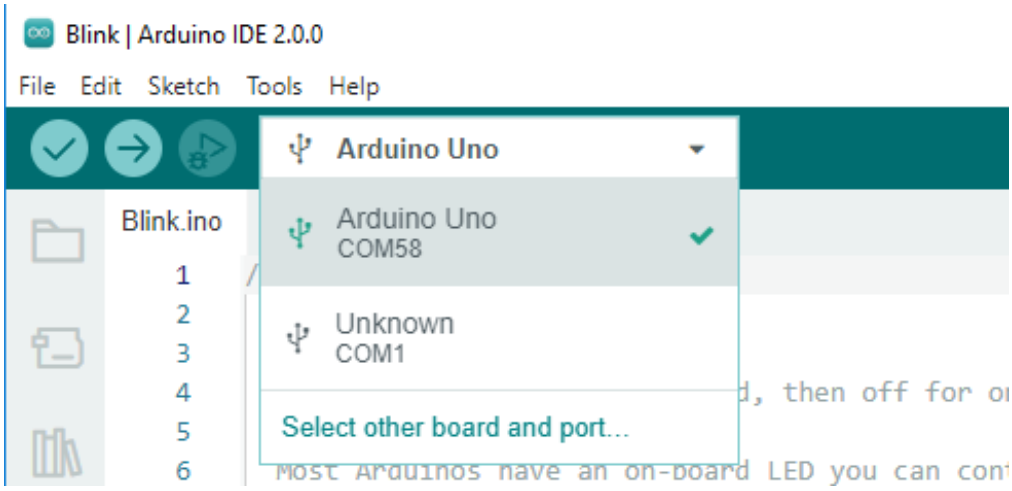
So laden Sie eine Skizze auf das Board hoch

In diesem Abschnitt erfahren Sie, wie Sie die zuvor erstellte Skizze auf das Arduino-Board hochladen und auf was Sie achten sollten.

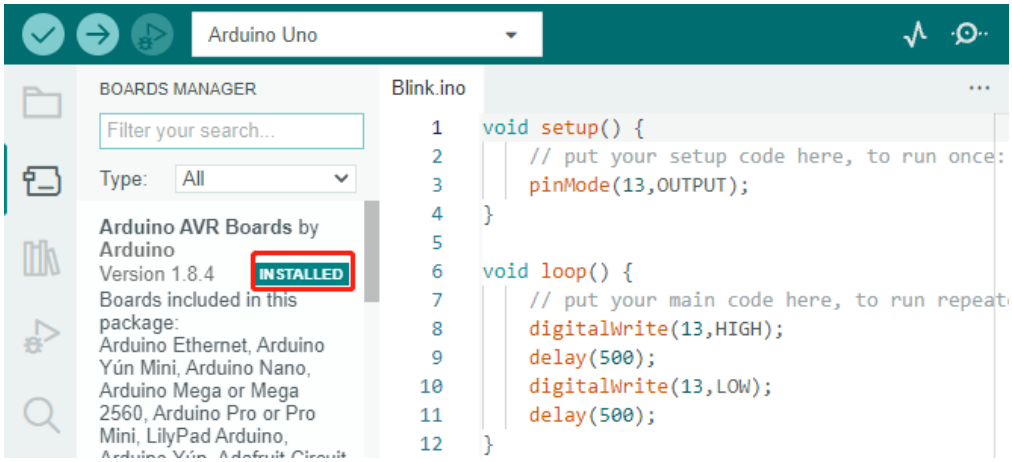
1. Wählen Sie Board und Port aus

Arduino-Entwicklungsboards werden in der Regel mit einem USB-Kabel geliefert. Verbinden Sie das Board damit mit Ihrem Computer.

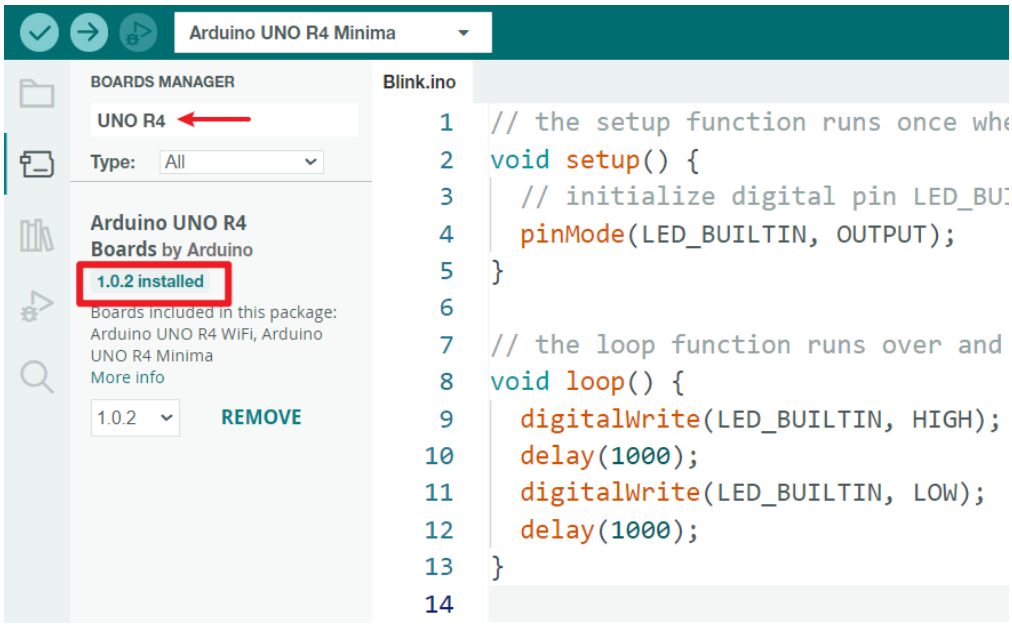
Wählen Sie im Arduino IDE das korrekte **Board** und den korrekten **Port** aus. Normalerweise erkennen Computer Arduino-Boards automatisch und weisen ihnen einen Port zu, den Sie hier auswählen können.



Sollte Ihr Board bereits angeschlossen, aber nicht erkannt sein, überprüfen Sie, ob das **installiert**-Logo im Abschnitt **Arduino AVR Boards** des **Board-Verwaltung** erscheint. Falls nicht, scrollen Sie ein wenig nach unten und klicken auf **Installieren**.



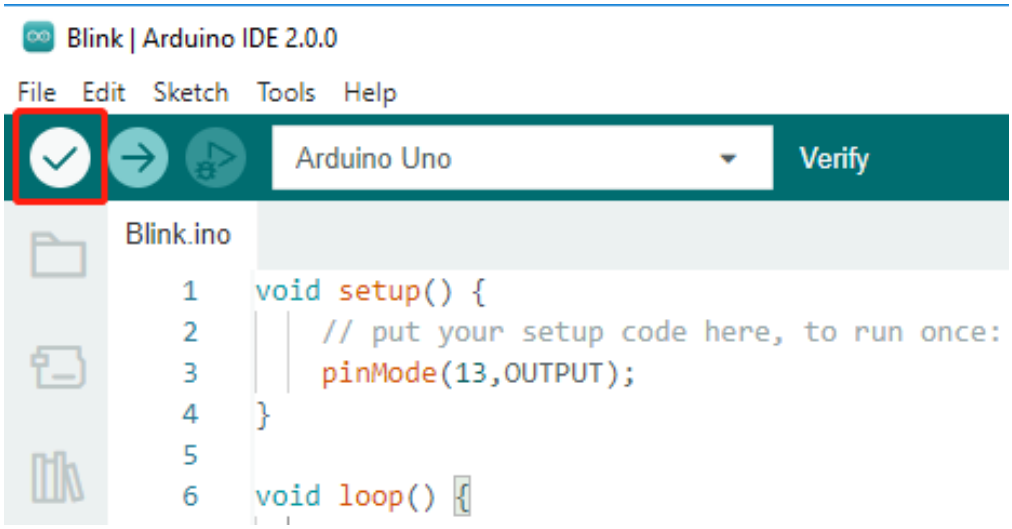
Speziell für UNO R4, suchen Sie im **Board-Verwaltung** nach **„UNO R4“** und prüfen, ob die zugehörige Bibliothek installiert ist.



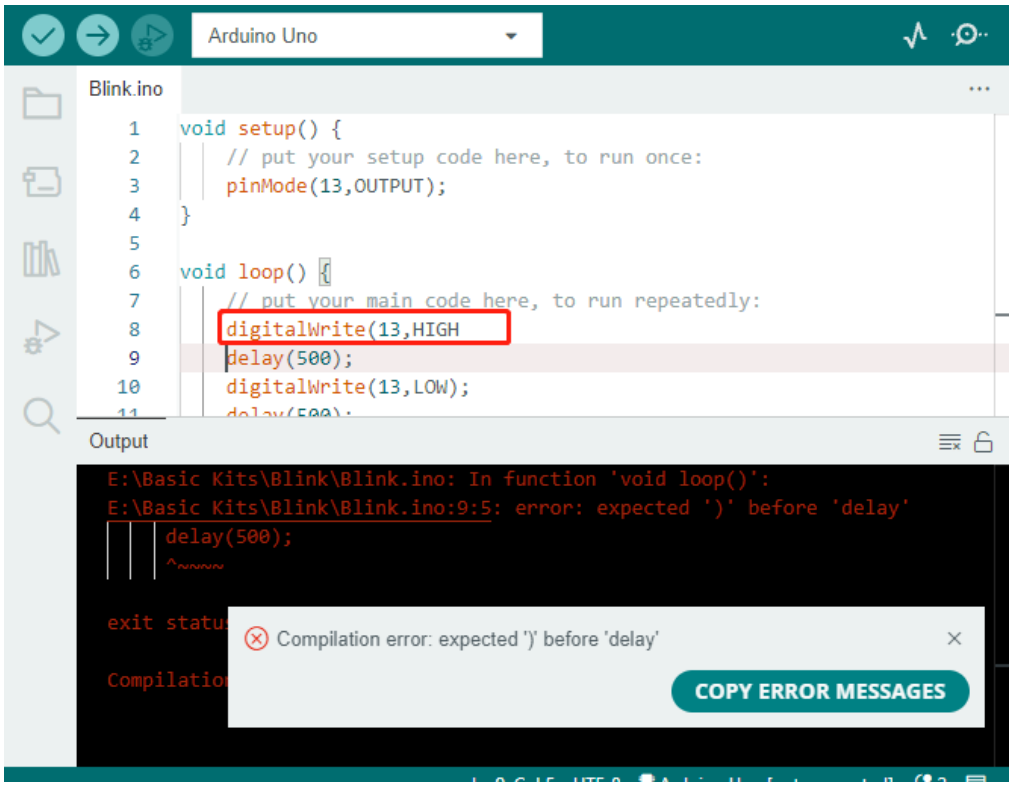
Ein Neustart der Arduino IDE und das erneute Anschließen des Arduino-Boards beheben die meisten Probleme. Alternativ können Sie auch **Werkzeuge** -> **Board** oder **Port** anklicken, um sie auszuwählen.

2. Überprüfen Sie die Skizze

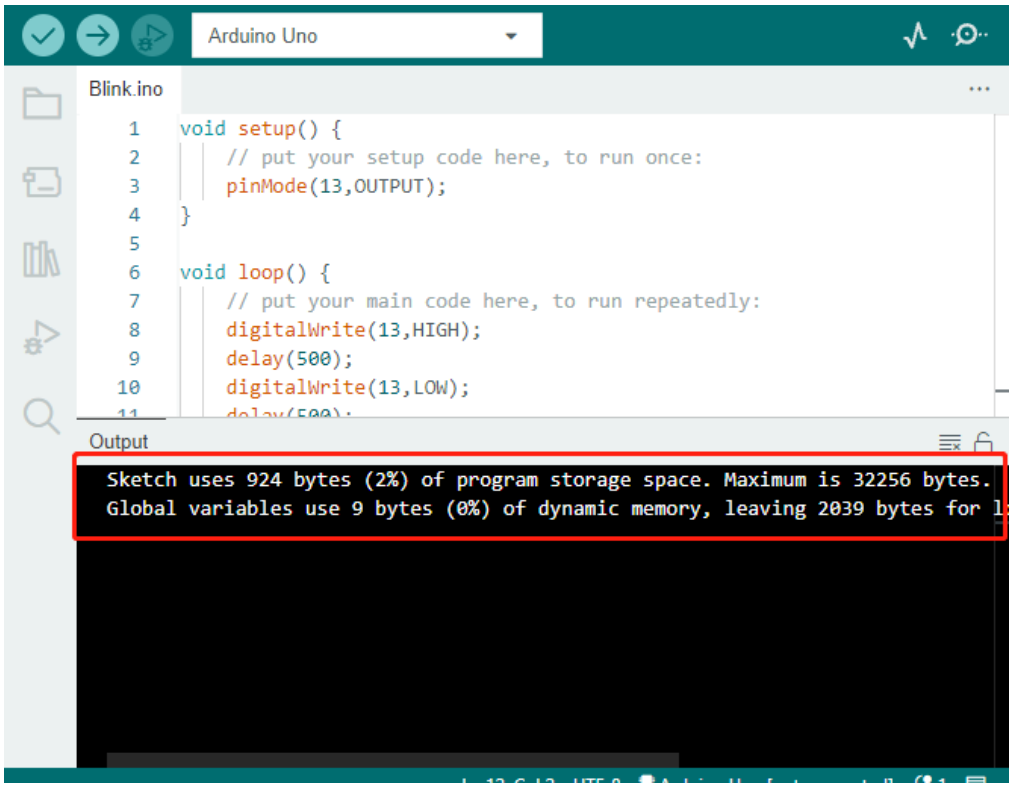
Nachdem Sie auf die Schaltfläche Überprüfen geklickt haben, wird die Skizze kompiliert, um mögliche Fehler zu identifizieren.



Sie können damit Fehler finden, falls Sie Zeichen gelöscht oder Buchstaben versehentlich eingetippt haben. Anhand der Nachrichtenzeile sehen Sie, wo und welche Art von Fehlern aufgetreten sind.



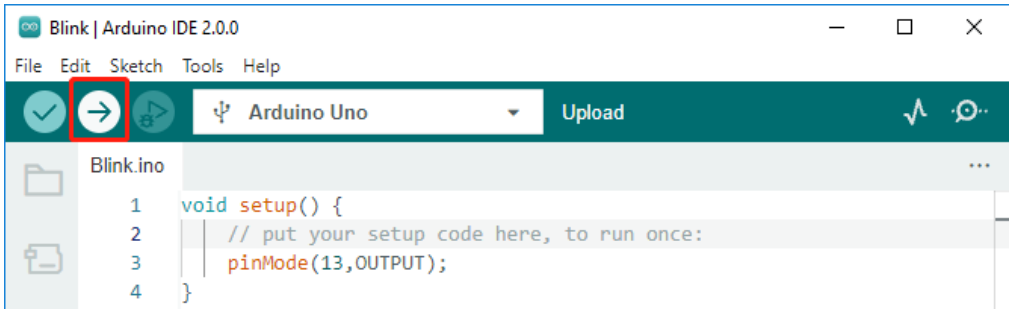
Wenn keine Fehler vorliegen, wird eine Meldung wie die folgende angezeigt.



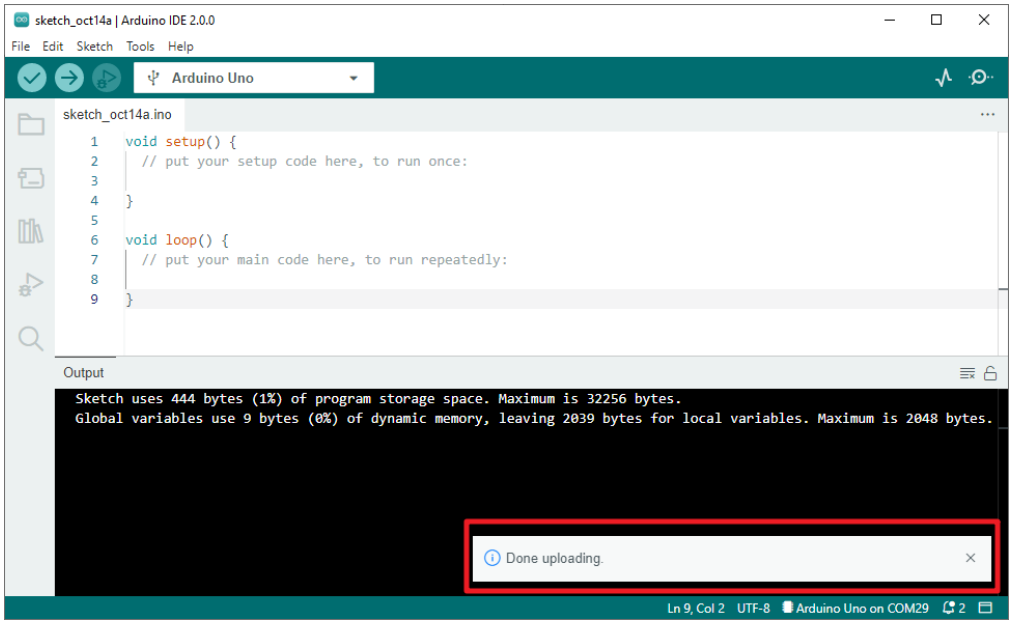
3. Skizze hochladen

Nach Abschluss der oben genannten Schritte klicken Sie auf die Schaltfläche **Hochladen**, um die Skizze auf das Board

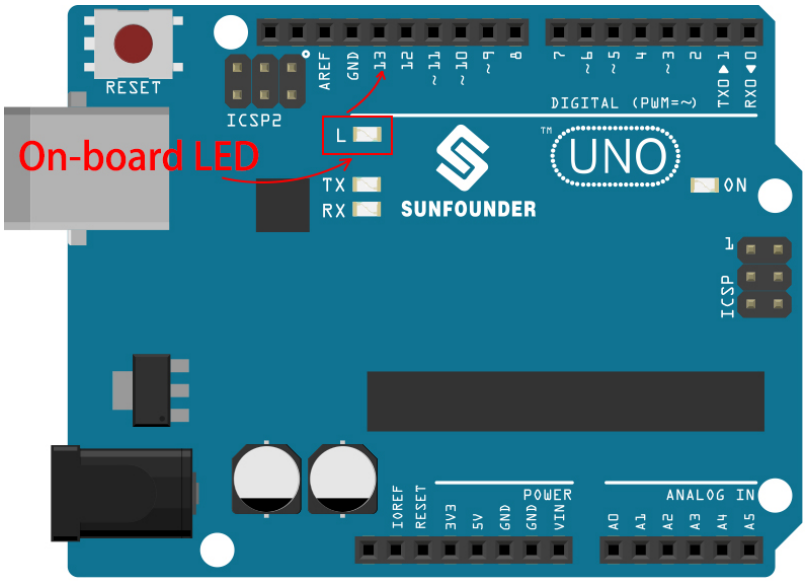
zu übertragen.



Bei Erfolg erscheint die folgende Aufforderung.



Gleichzeitig blinkt die LED auf dem Board.



Das Arduino-Board führt die Skizze automatisch aus, sobald die Stromversorgung nach dem Hochladen der Skizze hergestellt ist. Das laufende Programm kann durch Hochladen einer neuen Skizze überschrieben werden.

Struktur eines Arduino-Programms

Schauen wir uns die neue Skizze an. Obwohl sie bereits einige Codezeilen enthält, handelt es sich tatsächlich um eine „leere“ Skizze. Das Hochladen dieser Skizze auf das Entwicklungsboard wird keine Auswirkung haben.

```
void setup() {
// Hier kommt der Einrichtungscode hin, der einmal ausgeführt wird:

}

void loop() {
// Hier kommt der Hauptcode hin, der wiederholt ausgeführt wird:

}
```

Wenn wir `setup()` und `loop()` entfernen und die Skizze zu einer echten „leeren“ Datei machen, werden Sie feststellen, dass sie die Überprüfung nicht besteht. Sie sind das Äquivalent zum menschlichen Skelett und unverzichtbar.

Während der Skizzenerstellung wird `setup()` zuerst ausgeführt und der darin enthaltene Code (innerhalb von `{}`) wird nach dem Einschalten oder Zurücksetzen des Boards genau einmal ausgeführt. `loop()` dient dazu, die Hauptfunktion zu schreiben, und der darin enthaltene Code wird in einer Schleife ausgeführt, nachdem `setup()` abgearbeitet ist.

Um `setup()` und `loop()` besser zu verstehen, verwenden wir vier Skizzen. Ihr Ziel ist es, die integrierte LED des Arduino blinken zu lassen. Führen Sie bitte jedes Experiment nacheinander aus und dokumentieren Sie die spezifischen Effekte.

- Skizze 1: Die integrierte LED blinkt kontinuierlich.

```
void setup() {
    // Einrichtungscode, der einmal ausgeführt wird:
    pinMode(13, OUTPUT);
}

void loop() {
    // Hauptcode, der wiederholt ausgeführt wird:
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(500);
}
```

- Skizze 2: Die integrierte LED blinkt nur einmal.

```
void setup() {
    // Einrichtungscode, der einmal ausgeführt wird:
    pinMode(13, OUTPUT);
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(500);
}
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
void loop() {
    // Hauptcode, der wiederholt ausgeführt wird:
}
```

- Skizze 3: Die integrierte LED blinkt einmal langsam und dann schnell.

```
void setup() {
    // Einrichtungscode, der einmal ausgeführt wird:
    pinMode(13, OUTPUT);
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
}

void loop() {
    // Hauptcode, der wiederholt ausgeführt wird:
    digitalWrite(13, HIGH);
    delay(200);
    digitalWrite(13, LOW);
    delay(200);
}
```

- Skizze 4: Fehlermeldung.

```
void setup() {
    // Einrichtungscode, der einmal ausgeführt wird:
    pinMode(13, OUTPUT);
}

digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);

void loop() {
    // Hauptcode, der wiederholt ausgeführt wird:
}
```

Mit Hilfe dieser Skizzen können wir mehrere Eigenschaften von `setup-loop` zusammenfassen.

- `loop()` wird wiederholt ausgeführt, nachdem das Board eingeschaltet ist.
- `setup()` wird nur einmal ausgeführt, nachdem das Board eingeschaltet ist.
- Nach dem Einschalten des Boards wird zuerst `setup()` und dann `loop()` ausgeführt.
- Der Code muss innerhalb des `{}`-Bereichs von `setup()` oder `loop()` geschrieben werden, außerhalb dieses Rahmens führt es zu einem Fehler.

Bemerkung: Anweisungen wie `digitalWrite(13, HIGH)` dienen zur Steuerung der integrierten LED. Ihre genaue Verwendung wird in späteren Kapiteln detailliert erläutert.

Skizzen-Schreibregeln

Wenn du einen Freund bittest, das Licht für dich einzuschalten, kannst du einfach sagen: „Schalte das Licht ein“ oder „Mach Licht, Kumpel.“ Dabei spielt der Tonfall keine Rolle.

Wenn du jedoch möchtest, dass ein Arduino-Board etwas für dich tut, musst du die Programmschreibregeln von Arduino befolgen.

Dieses Kapitel enthält die grundlegenden Regeln der Arduino-Sprache und hilft dir zu verstehen, wie du natürliche Sprache in Code übersetzen kannst.

Natürlich erfordert dieser Prozess eine gewisse Eingewöhnungszeit und ist gerade für Anfänger am fehleranfälligsten. Wenn du also oft Fehler machst, ist das kein Problem, versuche es einfach noch ein paar Mal.

Semikolon ;

Genau wie im Briefverkehr, wo am Ende jedes Satzes ein Punkt steht, verlangt die Arduino-Sprache, dass du ; verwendest, um das Board über das Ende eines Befehls zu informieren.

Nehmen wir das bekannte Beispiel der „an Bord befindlichen LED-Blinkanzeige“. Ein fehlerfreies Skript könnte so aussehen:

Beispiel:

```
void setup() {
    // Füge hier deinen Setup-Code ein, der einmal ausgeführt wird:
    pinMode(13, OUTPUT);
}

void loop() {
    // Füge hier deinen Hauptcode ein, der wiederholt ausgeführt wird:
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(500);
}
```

Als nächstes betrachten wir die folgenden beiden Skizzen und überlegen, ob sie vom Arduino korrekt erkannt werden, bevor wir sie ausführen.

Skizze A:

```
void setup() {
    // Füge hier deinen Setup-Code ein, der einmal ausgeführt wird:
    pinMode(13, OUTPUT);
}

void loop() {
    // Füge hier deinen Hauptcode ein, der wiederholt ausgeführt wird:
    digitalWrite(13, HIGH)
    delay(500)
    digitalWrite(13, LOW)
    delay(500)
}
```

Skizze B:

```
void setup() {
    // Füge hier deinen Setup-Code ein, der einmal ausgeführt wird:
    pinMode(13, OUTPUT);
}

void loop() {
    // Füge hier deinen Hauptcode ein, der wiederholt ausgeführt wird:
    digitalWrite(13,
HIGH);  delay
    (500
    );
    digitalWrite(13,

    LOW);
            delay(500)
    ;
}
```

Das Ergebnis ist, dass **Skizze A** einen Fehler meldet und **Skizze B** ausgeführt wird.

- Die Fehler in **Skizze A** sind fehlende `;` und obwohl die Skizze normal aussieht, kann sie von Arduino nicht gelesen werden.
- **Skizze B** mag unleserlich erscheinen, aber Einrückungen, Zeilenumbrüche und Leerzeichen in den Anweisungen sind für den Arduino-Compiler irrelevant. Daher sieht die Skizze für ihn genauso aus wie im Beispiel.

Bitte schreibe deinen Code jedoch nicht wie in **Skizze B**, da es in der Regel Menschen sind, die den Code schreiben und lesen. Erschwere dir also nicht unnötig die Arbeit.

Geschweifte Klammern {}

Geschweifte Klammern {} sind ein Hauptbestandteil der Arduino-Programmiersprache und müssen immer paarweise vorkommen. Eine gängige Programmierrichtlinie besteht darin, direkt nach dem Tippen der öffnenden geschweiften Klammer auch die schließende geschweifte Klammer einzufügen und den Cursor dann zwischen die Klammern zu setzen, um die Anweisung einzufügen.

Kommentare //

Kommentare sind die Teile einer Skizze, die der Compiler ignoriert. Sie dienen in der Regel dazu, anderen zu erklären, wie das Programm funktioniert.

Schreibt man zwei aufeinanderfolgende Schrägstriche in einer Codezeile, wird alles bis zum Ende der Zeile vom Compiler ignoriert.

Wenn wir eine neue Skizze erstellen, enthält sie bereits zwei Kommentare. Entfernen wir diese, hat dies keinen Einfluss auf die Funktionsweise der Skizze.

```
void setup() {
    // Setze hier deinen Initialisierungscode ein, der einmal ausgeführt wird:

}

void loop() {
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    // Setze hier deinen Hauptcode ein, der ständig wiederholt wird:

}
```

Kommentare sind in der Programmierung sehr nützlich, und einige gängige Anwendungsbeispiele sind nachfolgend aufgelistet.

- Anwendungsfall A: Erkläre dir oder anderen, was dieser Codeabschnitt macht.

```
void setup() {
    pinMode(13, OUTPUT); // Setze Pin 13 in den Ausgabemodus, er steuert die onboard LED
}

void loop() {
    digitalWrite(13, HIGH); // Aktiviere die onboard LED durch Setzen von Pin 13 auf HIGH
    delay(500); // Halte den Status für 500 ms
    digitalWrite(13, LOW); // Schalte die onboard LED aus
    delay(500); // Halte den Status für 500 ms
}
```

- Anwendungsfall B: Temporäres Deaktivieren von Anweisungen (ohne sie zu löschen), um sie bei Bedarf wieder zu aktivieren, ohne sie neu schreiben zu müssen. Dies ist beim Debuggen von Code und bei der Fehlersuche sehr hilfreich.

```
void setup() {
    pinMode(13, OUTPUT);
    // digitalWrite(13, HIGH);
    // delay(1000);
    // digitalWrite(13, LOW);
    // delay(1000);
}

void loop() {
    digitalWrite(13, HIGH);
    delay(200);
    digitalWrite(13, LOW);
    delay(200);
}
```

Bemerkung: Verwenden Sie die Tastenkombination `Strg+/`, um Ihren Code schnell zu kommentieren oder zu dekommentieren.

Kommentar /**/

Gleichwertig zu // für Kommentare. Diese Art von Kommentar kann über mehrere Zeilen gehen. Sobald der Compiler /* liest, ignoriert er alles, was darauf folgt, bis er */ trifft.

Example 1:

```
/* Blink */

void setup() {
    pinMode(13,OUTPUT);
}

void loop() {
    /*
    The following code will blink the onboard LED
    You can modify the number in delay() to change the blinking frequency
    */
    digitalWrite(13,HIGH);
    delay(500);
    digitalWrite(13,LOW);
    delay(500);
}
```

#define

Dies ist ein nützliches Werkzeug in C++.

```
#define identifier Token-String
```

Der Compiler ersetzt beim Lesen automatisch den `identifier` durch den `Token-String`, was meist zur Definition von Konstanten verwendet wird.

Hier ein Beispiel für eine Skizze, die *#define* verwendet, um die Lesbarkeit des Codes zu erhöhen.

```
#define ONBOARD_LED 13
#define VERZOEGERUNGSZEIT 500

void setup() {
    pinMode(ONBOARD_LED, OUTPUT);
}

void loop() {
    digitalWrite(ONBOARD_LED, HIGH);
    delay(VERZOEGERUNGSZEIT);
    digitalWrite(ONBOARD_LED, LOW);
    delay(VERZOEGERUNGSZEIT);
}
```

Aus Sicht des Compilers sieht der Code eigentlich so aus:

```
void setup() {
    pinMode(13, OUTPUT);
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
}

void loop() {
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(500);
}
```

Wie man sehen kann, wird der `Bezeichner` ersetzt und existiert nicht im Programm. Daher gibt es einige Vorsichtsmaßnahmen bei der Verwendung.

1. Ein `Token-String` kann nur manuell geändert und nicht durch Rechenoperationen im Programm in andere Werte umgewandelt werden.
2. Vermeiden Sie die Verwendung von Symbolen wie `;`. Zum Beispiel:

```
#define ONBOARD_LED 13;

void setup() {
    pinMode(ONBOARD_LED, OUTPUT);
}

void loop() {
    digitalWrite(ONBOARD_LED, HIGH);
}
```

Der Compiler wird es wie folgt interpretieren, was zu einem Fehler führen wird:

```
void setup() {
    pinMode(13;, OUTPUT);
}

void loop() {
    digitalWrite(13;, HIGH);
}
```

Bemerkung: Eine Namenskonvention für `#define` ist es, den `identifier` zu groß zu schreiben, um Verwechslungen mit Variablen zu vermeiden.

Variable

Eine Variable ist eines der mächtigsten und wichtigsten Werkzeuge in einem Programm. Sie ermöglicht es uns, Daten in unseren Programmen zu speichern und abzurufen.

Das folgende Skizzen-File verwendet Variablen. Es speichert die Pin-Nummern der integrierten LED in der Variable `ledPin` und die Zahl „500“ in der Variable `delayTime`.

```
int ledPin = 13;
int delayTime = 500;
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

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

void loop() {
    digitalWrite(ledPin,HIGH);
    delay(delayTime);
    digitalWrite(ledPin,LOW);
    delay(delayTime);
}
```

Ist das eine Wiederholung von dem, was `#define` macht? Die Antwort ist NEIN.

- Die Rolle von `#define` ist es lediglich, Text einfach und direkt zu ersetzen; es wird vom Compiler nicht als Bestandteil des Programms angesehen.
- Eine `Variable` hingegen existiert im Programm und wird zum Speichern und Abrufen von Werten verwendet. Sie kann ihren Wert im Programm auch ändern, was `#define` nicht kann.

Im folgenden Skizzen-File addiert sich die Variable selbst, sodass die integrierte LED nach jedem Blinken länger blinkt.

```
int ledPin = 13;
int delayTime = 500;

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

void loop() {
    digitalWrite(ledPin,HIGH);
    delay(delayTime);
    digitalWrite(ledPin,LOW);
    delay(delayTime);
    delayTime = delayTime + 200; // Jede Ausführung erhöht den Wert um 200
}
```

Variable deklarieren

Eine Variable zu deklarieren bedeutet, eine Variable zu erstellen.

Für die Deklaration einer Variable benötigen Sie zwei Dinge: den Datentyp und den Variablennamen. Der Datentyp muss vom Variablennamen durch ein Leerzeichen getrennt und die Variablendeklaration durch ein `;` beendet werden.

Nehmen wir diese Variable als Beispiel.

```
int delayTime;
```

Datentyp

Hier ist `int` ein Datentyp namens Integer, der zur Speicherung von Ganzzahlen von -32768 bis 32766 verwendet werden kann. Er kann nicht zur Speicherung von Dezimalzahlen verwendet werden.

Neben Ganzzahlen können Variablen auch andere Arten von Daten speichern. Die Arduino-Sprache (die, wie man sich erinnern sollte, C++ ist) bietet eingebaute Unterstützung für einige davon (hier sind nur die am häufigsten verwendeten und nützlichsten aufgelistet):

- `float`: Speichert eine Dezimalzahl, z.B. 3,1415926.
- `byte`: Kann Zahlen von 0 bis 255 halten.
- `boolean`: Nimmt nur zwei mögliche Werte an, `True` oder `False`, obwohl es im Speicher ein Byte belegt.
- `char`: Speichert eine Zahl von -127 bis 127. Da es als `char` markiert ist, versucht der Compiler, es einem Zeichen aus der zuzuordnen.
- `string`: Kann eine Zeichenfolge speichern, z.B. `Halloween`.

Variablenname

Sie können der Variable jeden beliebigen Namen geben, z.B. `i`, `apple`, `Bruce`, `R2D2`, `Sectumsempra`, aber es gibt einige Grundregeln.

1. Beschreiben Sie den Verwendungszweck. Hier habe ich die Variable delayTime genannt, damit man leicht versteht, wofür sie dient. Wenn ich sie `barryAllen` nenne, verwirrt das die Person, die den Code betrachtet.
2. Verwenden Sie eine gängige Benennung. Wie ich es getan habe, können Sie CamelCase verwenden, wobei das T in `delayTime` groß geschrieben ist, sodass man leicht erkennen kann, dass die Variable aus zwei Wörtern besteht. Alternativ können Sie UnderScoreCase verwenden, um die Variable als `delay_time` zu schreiben. Das beeinflusst nicht die Ausführung des Programms, erleichtert aber dem Programmierer das Lesen des Codes, wenn Sie die von Ihnen bevorzugte Benennung verwenden.
3. Verwenden Sie keine Schlüsselwörter. Ähnlich wie bei der Eingabe von „int“ wird die Arduino-IDE es farblich hervorheben, um Sie daran zu erinnern, dass es ein Wort mit einer speziellen Bedeutung ist und nicht als Variablenname verwendet werden kann. Ändern Sie den Namen der Variable, wenn sie farblich hervorgehoben ist.
4. Spezielle Symbole sind nicht erlaubt. Zum Beispiel Leerzeichen, #, $, /, +, % usw. Die Kombination aus englischen Buchstaben (Groß- und Kleinschreibung), Unterstrichen und Zahlen (aber Zahlen können nicht als erstes Zeichen eines Variablennamens verwendet werden) ist ausreichend vielfältig.

Einen Wert einer Variable zuweisen

Nachdem wir die Variable deklariert haben, ist es an der Zeit, die Daten zu speichern. Wir verwenden den Zuweisungsoperator (d.h. =), um den Wert in die Variable zu setzen.

Wir können der Variable Werte zuweisen, sobald wir sie deklariert haben.

```
int delayTime = 500;
```

Es ist auch möglich, ihr später einen neuen Wert zuzuweisen.

```
int delayTime; // kein Wert
delayTime = 500; // Wert ist 500
delayTime = delayTime + 200; // Wert ist 700
```

2.2 Arduino Videokurs

Begleiten Sie uns auf einer Entdeckungsreise durch die Welt von Arduino mit dem umfassenden Arduino Videokurs, unter Verwendung des Ultimate Sensor Kits von SunFounder. Diese Serie beginnt mit einer Einführung in das Arduino-Ökosystem und die Fähigkeiten des UNO R4 Minima Boards, was den Grundstein für einen intensiven Einblick in praktische Anwendungen und Programmierungstechniken legt. Sie erlernen die Grundlagen der LED-Steuerung, verstehen serielle Kommunikation und üben den Umgang mit verschiedenen Komponenten wie RGB-LEDs, Tasten und Potentiometern. Wenn Sie dem Kurs Schritt für Schritt folgen, werden Sie Arduino meistern, indem Sie nicht nur Code kopieren und einfügen, sondern Ihren eigenen Code schreiben und Ihre Arduino-Projekte nach Ihren Wünschen umsetzen.

Der wird kontinuierlich aktualisiert, bleiben Sie dran für mehr!

Katalog

2.2.1 Video 1 - Arduino Uno R4 Minima

Eine schrittweise Einführung in Arduino Uno R4 Minima, die dessen Eigenschaften, Einrichtung und ein grundlegendes Programm zum Blinken einer LED umfasst.

- **Einführung**: Überblick über den Arduino Uno R4 Minima Schulungskurs und seinen Lehrplan.
- **Arduino Vergleich**: Detaillierte Darstellung der Verbesserungen und Funktionen des Uno R4 Minima im Vergleich zum Uno R3.
- **Wesentliche Merkmale**: Erforschung der fortgeschrittenen Funktionen und Fähigkeiten des Uno R4 Minima.
- **Pin-Funktionen**: Umfassender Leitfaden zu den verschiedenen Pinbelegungen und ihren Funktionalitäten.
- **IDE-Einrichtung**: Anleitung zur Installation und Konfiguration der Arduino IDE für Uno R4 Minima.
- **Erstes Programm**: Demonstration eines grundlegenden LED-Blinkprogramms als Einführungsprojekt.

Video

Zugehörige Online-Tutorials

- *Arduino UNO R4 Minima-Platine*

2.2.2 Video 2 - Einführung in die Arduino IDE

Erforschen Sie die Grundlagen der Arduino-Programmierung und die neuen Funktionen der Arduino IDE Version 2, ideal für Anfänger, die in die Welt von Arduino eintauchen.

- **Einführung in Arduino**: Ziel ist es, die Lücke in der Anleitung für Anfänger zu schließen, die den Arduino Uno R4 verwenden.
- **Erkundung der Arduino IDE**: Ein detaillierter Blick auf die Funktionen und Werkzeuge der Arduino IDE Version 2, um sie für neue Benutzer zugänglich zu machen.
- **Programmiergrundlagen**: Verständnis der Struktur von Arduino-Code, einschließlich der Funktionen `setup` und `loop`.
- **Steuerung der integrierten LEDs**: Praktische Beispiele zur Steuerung der eingebauten LEDs von Arduino, um grundlegende Programmierungstechniken zu demonstrieren.
- **Syntax und Funktionen**: Betonung der Bedeutung der Syntax in der Programmierung und Einführung essentieller Funktionen wie `pinMode` und `digitalWrite`.

Video

Zugehörige Online-Tutorials

- *Einstieg in Arduino*

2.2.3 Video 3: Grundlagen der Schaltkreise und des Breadboardings

Ein detailliertes Tutorial zum Aufbau grundlegender Schaltkreise und zur Verwendung eines Steckbretts, mit Fokus auf praktische Anwendungen mit dem Arduino Uno R4 Minima.

- **Grundlagen der Schaltkreise**: Einführung in die wesentlichen Komponenten eines elektrischen Schaltkreises.
- **Ohmsches Gesetz**: Erklärung der Beziehung zwischen Strom, Spannung und Widerstand.
- **Technik des Breadboardings**: Anleitung zur Verwendung eines Steckbretts für den Schaltkreisaufbau.
- **Reihenschaltungen**: Verständnis der Spannungsteilung und der Prinzipien von Reihenverbindungen.
- **Parallelschaltungen**: Erforschung der Eigenschaften und Vorteile von Parallelschaltungen.
- **LED-Schaltkreise**: Praktische Tipps zur Auswahl von Widerstandswerten und zum Aufbau von LED-Schaltkreisen mit Arduino.

Video

Zugehörige Online-Tutorials

- *RGB-Modul*

2.2.4 Video 4: Variablen, Datentypen und Konstanten

Dieses Tutorial behandelt die Grundlagen der Verwendung von Variablen, Datentypen, Konstanten und Präprozessoranweisungen in der Arduino-Programmierung, demonstriert anhand einer Ampelsimulation.

- **Variablen in Arduino**: Demonstration der Verwendung von Variablen für dynamische Programmierung in Arduino.
- **Datentypen**: Erläuterung verschiedener Datentypen wie `int`, `float`, `double`, `char`, `bool` und deren Speichernutzung.
- **Effiziente Speichernutzung**: Hervorhebung der Bedeutung der Auswahl geeigneter Datentypen für effizientes Speichermanagement in Mikrocontrollern.
- `const` **und** `#define`: Diskussion über die Verwendung von Konstanten und Präprozessoranweisungen zur Erstellung fester Werte im Code.
- **Ampelsimulation**: Anleitung zum Aufbau und zur Programmierung einer Ampelsimulation zur Anwendung dieser Konzepte.

Video

Zugehörige Online-Tutorials

- *Ampel-Modul*

2.2.5 Video 5: Digitaleingänge und bedingte Logik

Entdecken Sie, wie man digitale Eingänge integriert, schwebende Pins mit Pull-Up-/Down-Widerständen stabilisiert und bedingte Aussagen in Arduino Uno R4 Minima-Projekten anwendet, exemplarisch dargestellt an einer Ampelsimulation.

- **Digitaleingänge**: Erlernen des Anschlusses von Druckknopfschaltern als digitale Eingänge.
- **Pull-Up-/Down-Widerstände**: Verständnis für die Verwendung von Widerständen zur Stabilisierung schwebender Pins in Digitaleingängen.

- **Serienmonitor-Werkzeug**: Verwendung des Serienmonitors der Arduino IDE für textuelle Ausgaben vom Arduino-Board.
- **If-Else-Anweisungen**: Implementierung von If-Else-Logik für Entscheidungsfindungen in Arduino-Sketchen.
- **Ampelsimulation**: Demonstration der Konzepte mit einer Ampelschaltung unter Verwendung von LEDs und Drucktasten.
- `digitalRead()` **Funktion**: Einsatz dieser Funktion zur Lesung digitaler Eingänge und entsprechenden Steuerung der Ausgänge.

Video

Zugehörige Online-Tutorials

- *Tastenmodul*
- *Ampel-Modul*

2.2.6 Video 6: Analog-Digital-Umwandlung

Dieses Tutorial vertieft sich in die Analog-Digital-Umwandlung, erläutert Binärzahlen, Bits und Bytes, veranschaulicht durch die Verwendung eines Potentiometers mit Arduino Uno R4 Minima.

- **Analog-Digital-Umwandlung**: Demonstration der Umwandlungsprinzipien mit einem Potentiometer.
- **Digitale Darstellung**: Illustration, wie analoge Signale digitalisiert werden.
- **Programmierung für die Umwandlung**: Erstellung von Arduino-Sketchen zur Umwandlung von Dezimal- in Binärzahlen.
- **Bits und Bytes**: Erkundung der Grundlagen von Binärzahlen und deren Bedeutung in der digitalen Datenverarbeitung.
- `analogRead()` **Funktion**: Einsatz dieser Funktion zum Lesen und Umwandeln analoger Signale.
- **ADC-Auflösung**: Verständnis, wie die Auflösung die Präzision von analogen Eingangsmessungen beeinflusst.

Video

Zugehörige Online-Tutorials

- *Potentiometer-Modul*
- *Potentiometer-Skalenwert*

2.3 Code herunterladen

Laden Sie den entsprechenden Code über den unten stehenden Link herunter.

- `SunFounder Ultimate Sensor Kit Code`
- Oder schauen Sie sich den Code auf SunFounder Ultimate Sensor Kit - GitHub an.

2.4 Grundlagen der Komponenten

Nach dem Öffnen der Verpackung überprüfen Sie bitte, ob die Anzahl der Komponenten der Produktbeschreibung entspricht und ob alle Teile einwandfrei sind.

- `Liste der Komponenten des SunFounder Ultimate Sensor Kits`

Im Folgenden wird eine Einführung zu jeder Komponente gegeben, einschließlich ihres Funktionsprinzips und des zugehörigen Projekts. **Zu jeder Komponente gibt es ein einfaches Code-Beispiel, das Ihnen einen schnellen Einstieg ermöglicht.**

Steuerplatine

2.4.1 Arduino UNO R4 Minima-Platine

Das **Arduino UNO R4 Minima** ist eine Entwicklungsplatine im klassischen UNO-Formfaktor und basiert auf dem RA4M1()-Mikrocontroller von Renesas. Die Platine ist schneller und bietet mehr Speicher als ihre Vorgängermodelle. Sie verfügt über eine Reihe integrierter Funktionen wie einen DAC, RTC und HID. Das UNO R4 Minima ist eine **ausschließlich 5-V-Platine**. Sie hat 14 digitale I/O-Pins, 6 analoge Eingänge mit einer Auflösung von bis zu 14 Bit, eine Taktfrequenz von 48 MHz sowie 32 kB SRAM, 256 kB Flash-Speicher und 8 kB EEPROM.

Technische Parameter

Platine	Name	Arduino® UNO R4 Minima
Mikrocontroller	Renesas RA4M1 (Arm® Cortex®-M4)	
USB	USB-C®	Programmieranschluss
Pins	Digitale I/O-Pins	14
Pins	Analoge Eingangspins	6
	DAC	1
	PWM-Pins	6
Kommunikation	UART	Ja, 1x
	I2C	Ja, 1x
	SPI	Ja, 1x
	CAN	Ja, 1 CAN-Bus
Stromversorgung	Betriebsspannung der Schaltung	5 V
	Eingangsspannung (VIN)	6-24 V
	Gleichstrom pro I/O-Pin	8 mA
Taktfrequenz	Hauptkern	48 MHz
Speicher	RA4M1	256 kB Flash, 32 kB RAM
Abmessungen	Breite	68,85 mm
	Länge	53,34 mm

Weiterführende Informationen

- Arduino IDE
- *Download und Installation der Arduino IDE 2.0*
- Arduino Programmiersprachen-Referenz
-
-
-

Sensorik

2.4.2 Ultraschall-Sensormodul (HC-SR04)

Einleitung

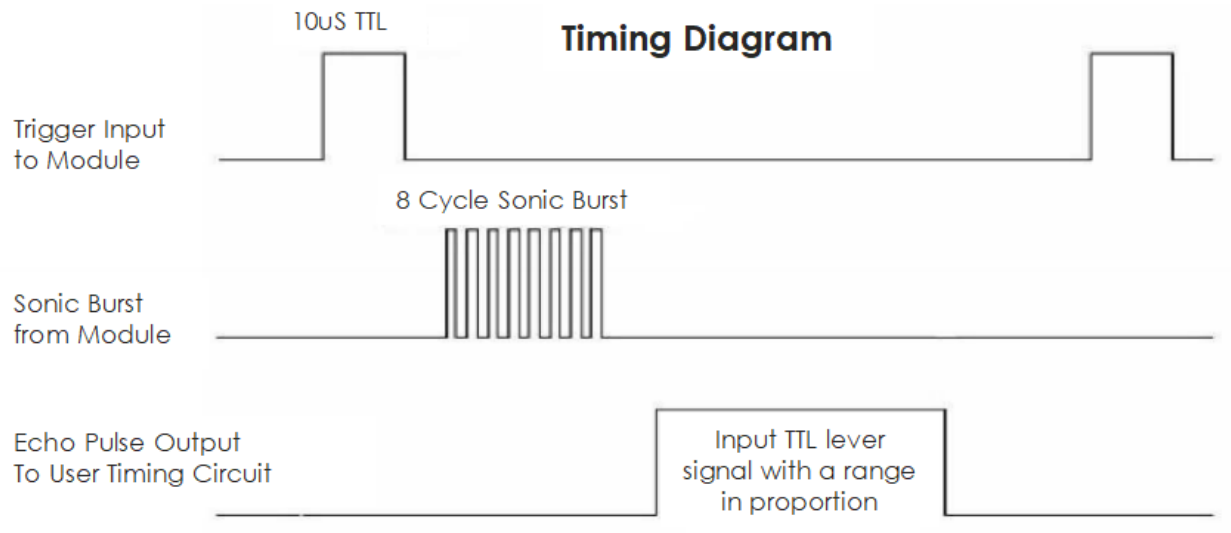
Das Ultraschall-Sensormodul (HC-SR04) ist ein Sensor, der mithilfe von Ultraschallwellen Entfernungen zwischen 2 cm und 4 Metern messen kann. Es wird häufig in Robotik- und Automatisierungsprojekten eingesetzt, um Objekte zu erkennen und Entfernungen zu messen. Das Modul besteht aus einem Ultraschall-Sender und -Empfänger, die gemeinsam Ultraschallwellen senden und empfangen.

Funktionsprinzip

Das Modul umfasst Ultraschallsender, Empfänger und Steuerschaltung. Die grundlegenden Prinzipien sind wie folgt:

1. Ein IO-Flipflop verarbeitet ein High-Level-Signal von mindestens 10us.
2. Das Modul sendet automatisch acht 40-kHz-Signale aus und prüft, ob ein Pulssignal zurückkehrt.
3. Wenn das Signal zurückkehrt und das High-Level passiert, beträgt die Ausgabe-IO-Dauer die Zeit vom Senden der Ultraschallwelle bis zu ihrer Rückkehr. Hier gilt: Testentfernung = (High-Level-Zeit x Schallgeschwindigkeit (340 m/s) / 2.

Das Timing-Diagramm ist unten dargestellt.



Es genügt, einen kurzen 10us-Impuls für den Trigger-Eingang zu liefern, um die Entfernungsmessung zu starten. Das Modul sendet dann eine 8-Zyklus-Serie von Ultraschallwellen mit 40 kHz aus und erhebt sein Echo. Die Entfernung kann durch das Zeitintervall zwischen dem Senden des Trigger-Signals und dem Empfang des Echo-Signals berechnet werden.

Bemerkung: Es wird empfohlen, einen Messzyklus von mehr als 60 ms zu verwenden, um Signalüberlappungen zwischen Trigger- und Echo-Signal zu vermeiden.

Formel:

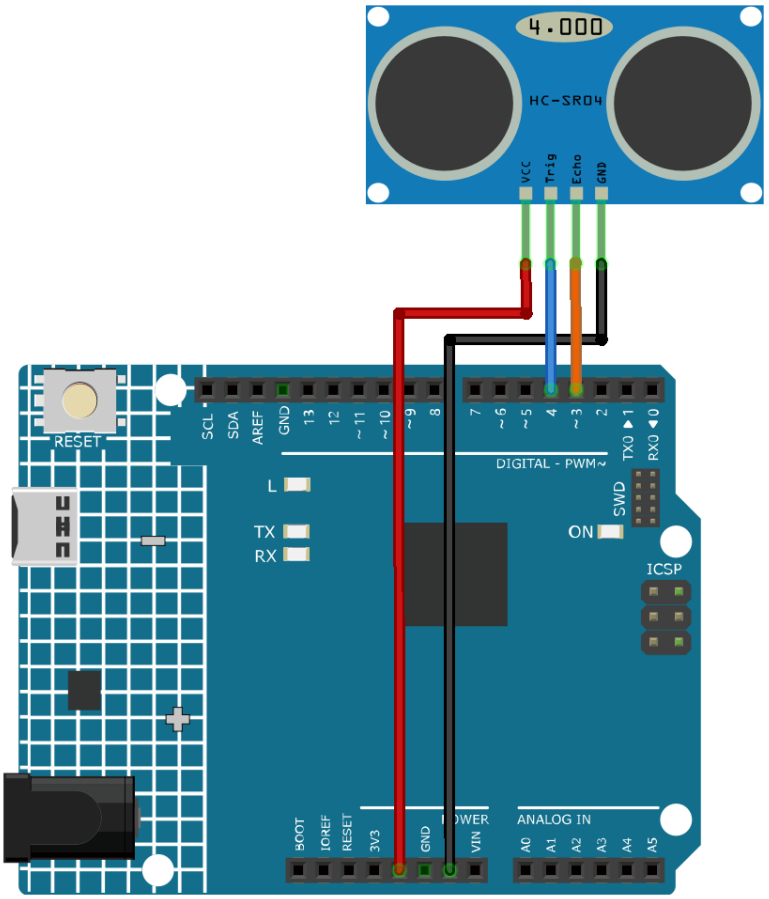
- us / 58 = Zentimeter
- us / 148 = Zoll
- Entfernung = High-Level-Zeit * Schallgeschwindigkeit (340m/s) / 2;

Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3-Board * 1
- Ultraschall-Sensormodul * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Programmcode Erklärung

1. Pin-Deklaration:

   Beginnen Sie mit der Definition der Pins für den Ultraschallsensor. `echoPin` und `trigPin` werden als Ganzzahlen deklariert und ihre Werte entsprechen den physischen Anschlüssen auf dem Arduino-Board.

```
const int echoPin = 3;
const int trigPin = 4;
```

2. `setup()` Funktion:

   Die `setup()` Funktion initialisiert die serielle Kommunikation, setzt die Pin-Modi und gibt eine Meldung aus, um anzuzeigen, dass der Ultraschallsensor bereit ist.

```
void setup() {
  Serial.begin(9600);
  pinMode(echoPin, INPUT);
  pinMode(trigPin, OUTPUT);
  Serial.println("Ultrasonic sensor:");
}
```

3. `loop()` Funktion:

Die `loop()` Funktion liest die Entfernung vom Sensor und zeigt sie auf dem seriellen Monitor an. Anschließend erfolgt eine Verzögerung von 400 Millisekunden, bevor der Vorgang wiederholt wird.

```
void loop() {
  float distance = readDistance();
  Serial.print(distance);
  Serial.println(" cm");
  delay(400);
}
```

4. `readDistance()` Funktion:

   Die `readDistance()` Funktion aktiviert den Ultraschallsensor und berechnet die Entfernung anhand der Zeit, die das Signal für den Hin- und Rückweg benötigt.

```
float readDistance() {
  digitalWrite(trigPin, LOW);   // Set trig pin to low to ensure a clean pulse
  delayMicroseconds(2);         // Delay for 2 microseconds
  digitalWrite(trigPin, HIGH);  // Send a 10 microsecond pulse by setting trig pin
↪to high
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);  // Set trig pin back to low
  float distance = pulseIn(echoPin, HIGH) / 58.00;  // Formula: (340m/s * 1us) / 2
  return distance;
}
```

Zusätzliche Ideen

- Anzeige der Entfernung auf einem LCD-Bildschirm statt auf dem seriellen Monitor
- Hinzufügen von LEDs, die leuchten, wenn ein Objekt innerhalb einer bestimmten Entfernung ist

Weitere Projekte

- *Intelligenter Mülleimer*

2.4.3 Gas-/Rauch-Sensormodul (MQ2)

Einleitung

Der MQ-2-Sensor ist ein vielseitiger Gassensor, der eine breite Palette von Gasen wie Alkohol, Kohlenmonoxid, Wasserstoff, Isobuten, Flüssiggas, Methan, Propan und Rauch erkennen kann. Aufgrund seiner kostengünstigen und benutzerfreundlichen Eigenschaften ist er besonders bei Einsteigern beliebt.

Funktionsprinzip

Der MQ-2-Sensor basiert auf dem Prinzip der Widerstandsänderung in Gegenwart verschiedener Gase. Wenn das Zielgas mit dem erhitzten MOS-Material (Metalloxid-Halbleiter) in Kontakt kommt, finden Oxidations- oder Reduktionsreaktionen statt, die den Widerstand des MOS-Materials verändern. Es ist bemerkenswert, dass der MQ-2-Sensor mehrere Gase erkennen kann, jedoch nicht in der Lage ist, zwischen ihnen zu unterscheiden. Dies ist eine übliche Eigenschaft der meisten Gassensoren.

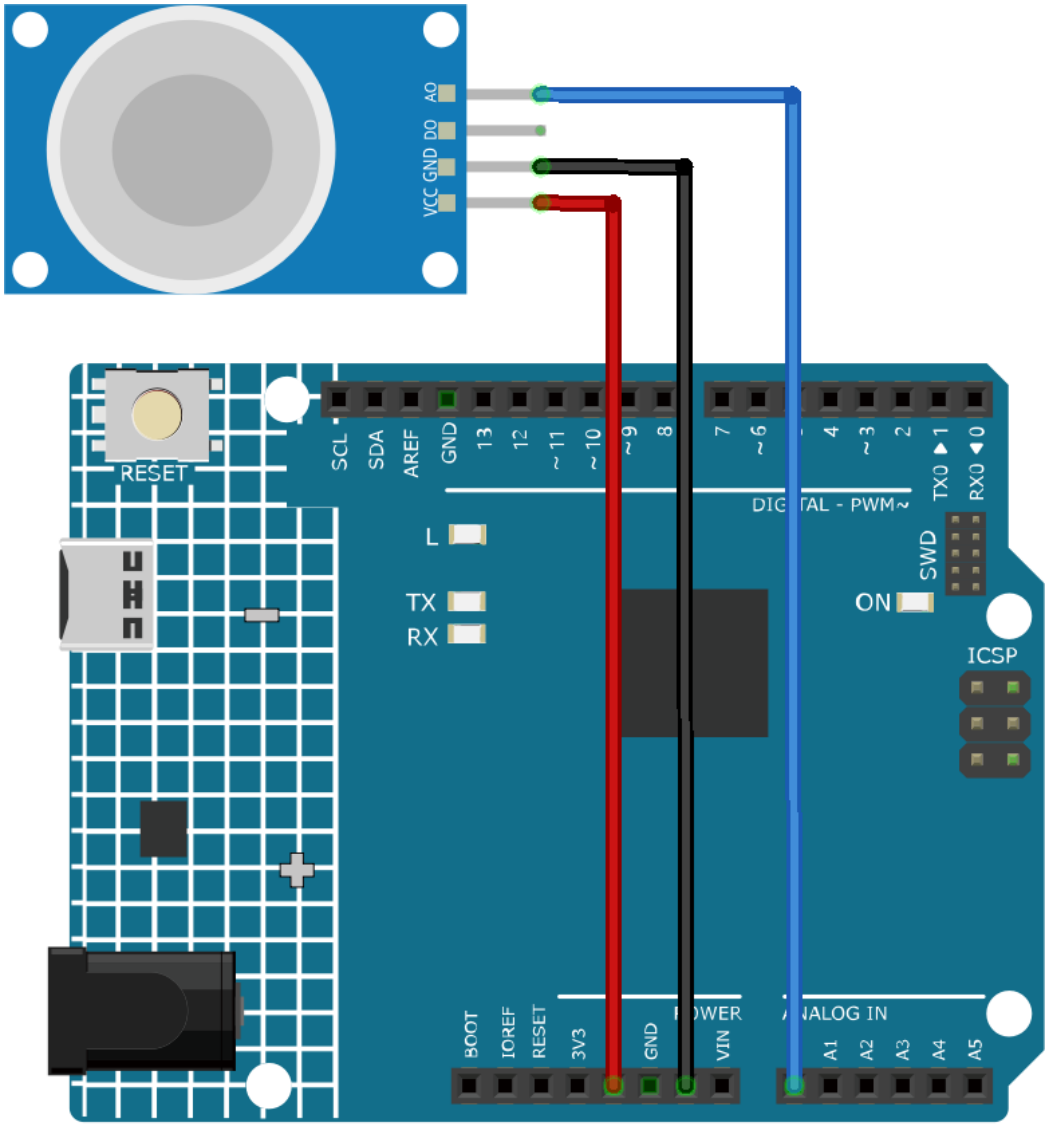
Der Sensor verfügt über ein integriertes Potentiometer, das es ermöglicht, die digitale Ausgangsschwelle (D0) einzustellen. Überschreitet die Gaskonzentration in der Luft einen bestimmten Schwellenwert, ändert sich der Widerstand des Sensors. Diese Widerstandsänderung wird dann in ein elektrisches Signal umgewandelt, das von einem Arduino-Board ausgelesen werden kann.

Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Gas-Sensormodul (MQ2) * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erläuterung

1. Die erste Codezeile ist eine konstante Ganzzahldeklaration für den Sensorpin des Gassensors. Wir verwenden den analogen Pin A0, um die Ausgabe des Gassensors auszulesen.

```
const int sensorPin = A0;
```

2. In der `setup()`-Funktion initialisieren wir unsere serielle Kommunikation mit einer Baudrate von 9600. Dies ist notwendig, um die Messwerte des Gassensors im seriellen Monitor anzuzeigen.

```
void setup() {
  Serial.begin(9600);  // Start serial communication at 9600 baud rate
}
```

3. In der `loop()`-Funktion lesen wir kontinuierlich den Analogwert des Gassensors aus und zeigen ihn im seriellen Monitor an. Wir verwenden die Funktion `analogRead()` zum Auslesen des Analogwerts. Anschließend warten wir 50 Millisekunden bis zur nächsten Messung. Diese Verzögerung gibt dem seriellen Monitor etwas Zeit zur Datenverarbeitung.

```
void loop() {
  Serial.print("Analog output: ");
  Serial.println(analogRead(sensorPin));  // Read the analog value of the gas␣
↪sensor and print it to the serial monitor
  delay(50);                              // Wait for 50 milliseconds
}
```

Bemerkung: Der MQ2 ist ein heizgetriebener Sensor, der normalerweise vor der Verwendung eine Vorheizphase benötigt. Während dieser Vorheizphase sind die Sensormesswerte in der Regel hoch und nehmen allmählich ab, bis sie sich stabilisieren.

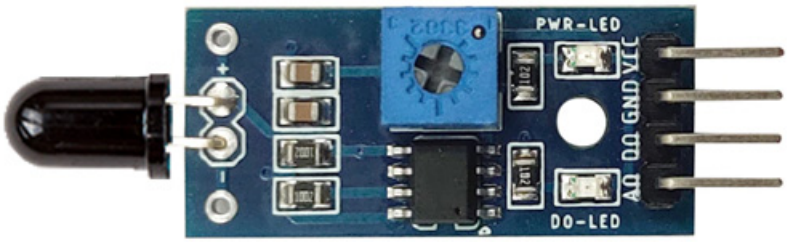
Weitere Ideen

- Hinzufügen eines visuellen oder akustischen Warnsystems (mittels LEDs oder eines Buzzers), das auslöst, wenn die Gaskonzentration bestimmte Schwellenwerte überschreitet.

Weitere Projekte

- *Gasleck-Alarm*

2.4.4 Flammen-Sensormodul



Einleitung

Das Flammen-Sensormodul ist in der Lage, die Anwesenheit von Feuer oder Flammen zu detektieren. Es erkennt Brände hauptsächlich durch die Erfassung der von Flammen oder Feuer emittierten Infrarotstrahlung. Es wird weitgehend in Brandmeldesystemen in Privathaushalten und in der Industrie verwendet.

Funktionsprinzip

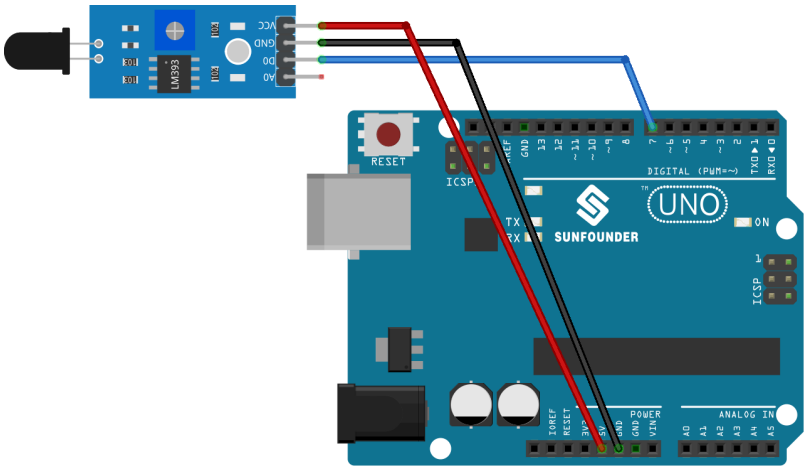
Das Flammen-Sensormodul arbeitet auf Basis der Infrarotdetektion. Der Sensor besitzt einen IR-Empfänger, der die von den Flammen emittierte Infrarotstrahlung erkennt. Wenn Feuer brennt, emittiert es eine geringe Menge an infrarotem Licht, das von der Fotodiode (IR-Empfänger) auf dem Sensor empfangen wird. Mittels eines Operationsverstärkers wird dann die Spannungsänderung am IR-Empfänger überprüft, sodass bei erkanntem Feuer der Ausgangspin (DO) 0V (LOW) liefert und bei Nichterkennung 5V (HIGH).

Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Flammen-Sensormodul * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erläuterung

1. Die erste Codezeile definiert eine Konstante für den Pin des Flammen-Sensors. Hier verwenden wir den digitalen Pin 7, um die Ausgabe vom Flammen-Sensor zu lesen.

```
const int sensorPin = 7;
```

2. Die `setup()` Funktion initialisiert den Pin des Flammen-Sensors als Eingang und den eingebauten LED-Pin als Ausgang. Zudem wird die serielle Kommunikation mit einer Baudrate von 9600 für die Ausgabe von Nachrichten im seriellen Monitor gestartet.

```
void setup() {
  pinMode(sensorPin, INPUT);      // Set the flame sensor pin as input
  pinMode(LED_BUILTIN, OUTPUT);   // Set the built-in LED pin as output
  Serial.begin(9600);             // Initialize the serial monitor at a baud rate of␣
↪9600
}
```

3. In der `loop()` Funktion wird kontinuierlich der Status des Flammen-Sensors überprüft. Erkennt der Sensor eine Flamme, wird die integrierte LED eingeschaltet und eine Nachricht im seriellen Monitor ausgegeben. Wird keine Flamme erkannt, wird die LED ausgeschaltet und eine andere Nachricht ausgegeben. Der Vorgang wiederholt sich alle 100 Millisekunden.

```
void loop() {
  // Check if the sensor is detecting a fire
  if (digitalRead(sensorPin) == 0) {
    digitalWrite(LED_BUILTIN, HIGH);  // Turn on the built-in LED
    Serial.println("** Fire detected!!! **");
  } else {
    digitalWrite(LED_BUILTIN, LOW);  // Turn off the built-in LED
    Serial.println("No Fire detected");
  }
  delay(100);
}
```

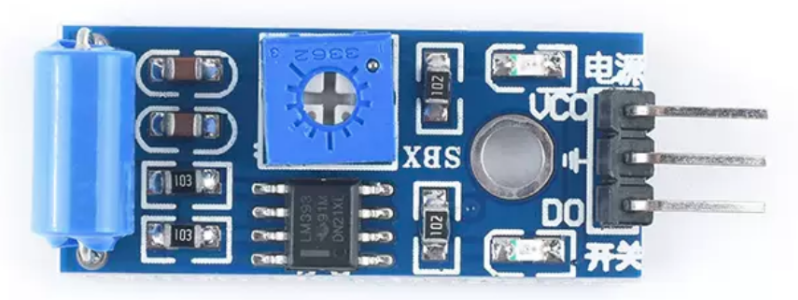
Weitere Ideen

- Den Code so modifizieren, dass ein Summer oder Alarm ausgelöst wird, wenn eine Flamme erkannt wird.
- Zusätzlich zum Flammen-Sensor einen Rauchsensor integrieren, um die Branderkennung zu verbessern.
- Den **analogen Ausgang** anstelle des einfachen digitalen HIGH/LOW verwenden. Dazu den **AO** Pin verwenden.

Weitere Projekte

- *Flammenwarnsystem mit Blynk*

2.4.5 Vibrationssensor-Modul (SW-420)



Einführung

Das SW-420 Vibrationssensor-Modul ist in der Lage, Vibrationen oder Erschütterungen auf einer Oberfläche zu erfassen. Es kann für verschiedene Zwecke eingesetzt werden, wie z.B. zur Erkennung von Türklopfen, Maschinenfehlern, Autounfällen oder Alarmanlagen. Das Modul arbeitet mit einer Spannung von 3,3 V bis 5 V und verfügt über drei Peripheriegeräte: zwei LEDs, eine für den Betriebsstatus und die andere für den Sensorausgang, sowie ein Potentiometer zur Feineinstellung des Ansprechpunktes für Vibrationen.

Funktionsprinzip

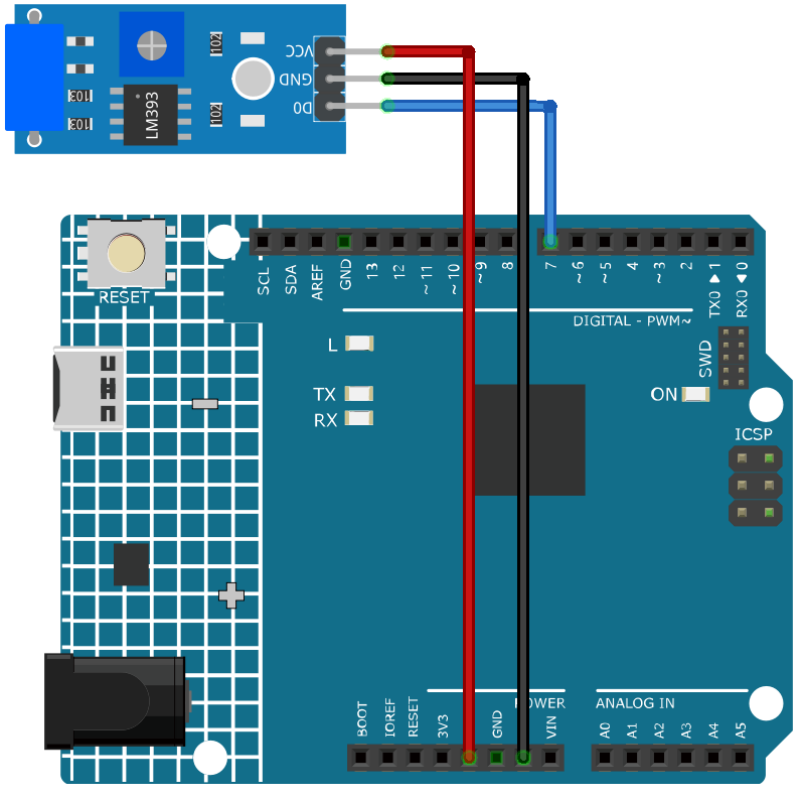
Das SW-420 Vibrationssensor-Modul setzt sich aus einem SW-420 Vibrationsschalter und einem LM393 Spannungsvergleicher zusammen. Der SW-420 Vibrationsschalter enthält eine Feder und einen Stab innerhalb eines Rohres. Bei Vibrationen kommt die Feder mit dem Stab in Kontakt und schließt den Stromkreis. Der im Modul integrierte Vibrationssensor erfasst diese Schwingungen und wandelt sie in elektrische Signale um. Der LM393 Vergleicherchip vergleicht diese Signale dann mit einer Referenzspannung, die am Potentiometer eingestellt wird. Überschreitet die Amplitude des Signals diese Referenzspannung, gibt der Vergleicher ein hohes Signal (1) aus, andernfalls ein niedriges (0).

Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Vibrationssensor-Modul (SW-420) * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erklärung

1. Die erste Zeile des Codes deklariert einen konstanten Integer für den Vibrationssensor-Anschluss. Der digitale Pin 7 wird verwendet, um die Ausgabe des Vibrationssensors zu lesen.

```
const int sensorPin = 7;
```

2. In der `setup()`-Funktion initialisieren wir die serielle Kommunikation mit einer Baudrate von 9600, um die Messwerte des Vibrationssensors im seriellen Monitor anzuzeigen. Außerdem wird der Vibrationssensor-Anschluss als Eingang definiert.

```
void setup() {
  Serial.begin(9600);         // Start serial communication at 9600 baud rate
  pinMode(sensorPin, INPUT);  // Set the sensorPin as an input pin
}
```

3. In der `loop()`-Funktion prüfen wir kontinuierlich, ob vom Sensor Vibrationen erkannt werden. Bei Erkennung einer Vibration wird „Detected vibration…“ im seriellen Monitor ausgegeben. Wird keine Vibration erkannt, wird „…“ ausgegeben. Die Schleife wiederholt sich alle 100 Millisekunden.

```
void loop() {
  if (digitalRead(sensorPin)) {               // Check if there is any vibration␣
↪detected by the sensor
    Serial.println("Detected vibration...");  // Print "Detected vibration..." if␣
↪vibration detected
  }
  else {
    Serial.println("...");  // Print "..." otherwise
  }
  // Add a delay to avoid flooding the serial monitor
  delay(100);
}
```

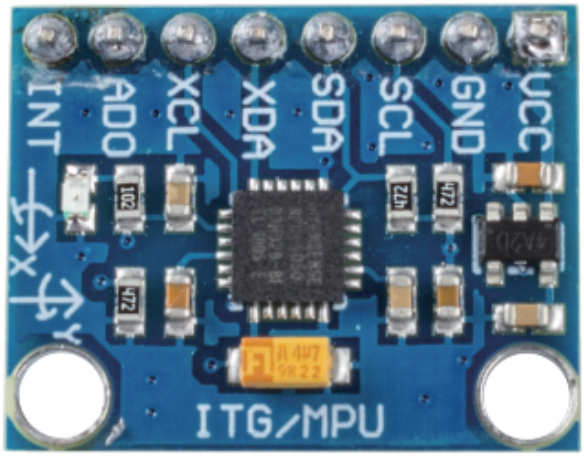
Weitere Ideen

- Anschluss einer LED, die bei erkannter Vibration leuchtet
- Auslösen eines Alarms oder Buzzers bei Vibrationserkennung

Weitere Projekte

- *Einbruchmeldeanlage mit Blynk*

2.4.6 Beschleunigungssensor & Gyroskop-Modul (MPU6050)



Einleitung

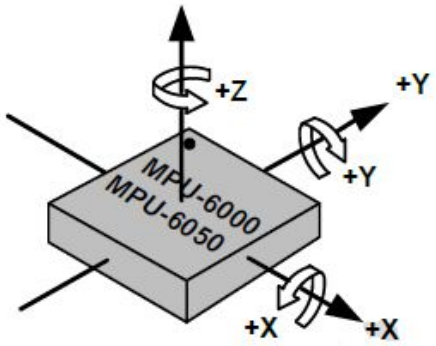
Der MPU-6050 ist ein 6-achsiger Bewegungssensor, der einen 3-achsigen Gyroskop und einen 3-achsigen Beschleunigungsmesser kombiniert. Er erfasst Veränderungen in Bewegung, Beschleunigung und Rotation. Häufig findet er Einsatz in Robotik, Game-Controllern und anderen elektronischen Geräten, die eine Bewegungserkennung benötigen. Aufgrund seiner hohen Genauigkeit und geringen Kosten ist er besonders bei der DIY-Community beliebt.

Funktionsprinzip

Ein MPU-6050 Sensor-Modul besteht aus einem 3-achsigen Beschleunigungsmesser und einem 3-achsigen Gyroskop.

Die drei Koordinatensysteme sind wie folgt definiert:

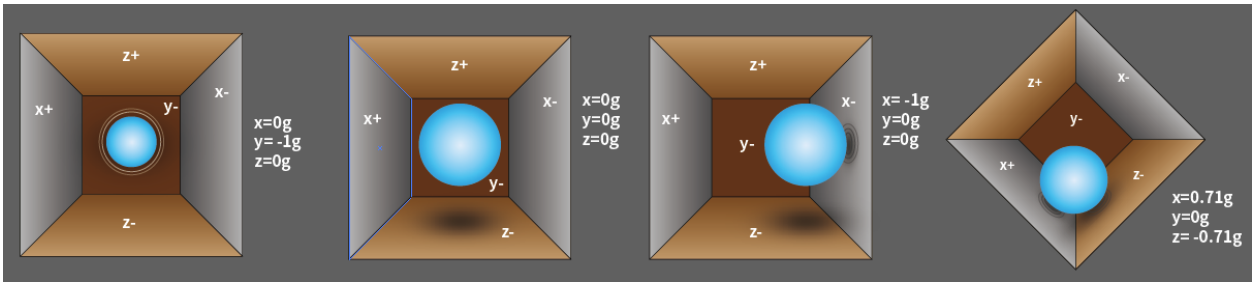
Legen Sie den MPU6050 flach auf den Tisch, sodass das Etikett nach oben zeigt und ein Punkt auf dieser Oberfläche in der oberen linken Ecke ist. Dann ist die aufrechte Richtung nach oben die Z-Achse des Chips. Die Richtung von links nach rechts wird als X-Achse betrachtet. Entsprechend ist die Richtung von hinten nach vorne als Y-Achse definiert.



3-achsiger Beschleunigungsmesser

Der Beschleunigungsmesser funktioniert nach dem Prinzip des piezoelektrischen Effekts, der es bestimmten Materialien ermöglicht, bei mechanischer Belastung eine elektrische Ladung zu erzeugen.

Stellen Sie sich eine quaderförmige Box mit einer kleinen Kugel darin vor, wie im obigen Bild. Die Wände dieser Box bestehen aus piezoelektrischen Kristallen. Wenn Sie die Box neigen, bewegt sich die Kugel aufgrund der Schwerkraft in Richtung der Neigung. Die Wand, gegen die die Kugel stößt, erzeugt kleine piezoelektrische Ströme. Insgesamt gibt es drei gegenüberliegende Wandpaare in einem Quader, die jeweils einer Achse im 3D-Raum entsprechen: den X-, Y- und Z-Achsen. Abhängig von den Strömen, die von den piezoelektrischen Wänden erzeugt werden, können wir die Richtung und das Ausmaß der Neigung bestimmen.



Der MPU6050 kann zur Erfassung der Beschleunigung auf jeder Koordinatenachse verwendet werden (im stationären Zustand ist die Beschleunigung der Z-Achse 1 Gravitationseinheit, X und Y sind 0). Bei Neigung oder in einem zustandslosen/überlasteten Zustand ändert sich der entsprechende Wert.

Es gibt vier wählbare Messbereiche: +/-2g, +/-4g, +/-8g und +/-16g (standardmäßig 2g), die jeweils einer Präzision entsprechen. Die Werte reichen von -32768 bis 32767.

Die Beschleunigungsablesung wird durch Abbildung des Messbereichs auf den Lesebereich in einen Beschleunigungswert umgewandelt.

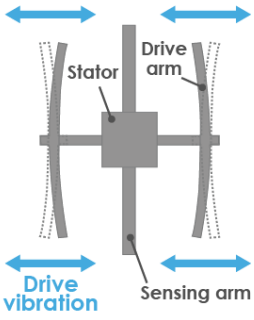
Beschleunigung = (Rohdaten der Beschleunigungsachsen / 65536 * voller Beschleunigungsbereich) g

Nehmen Sie die X-Achse als Beispiel, wenn die Rohdaten der Beschleunigungs-X-Achse 16384 betragen und der Bereich +/- 2g ausgewählt ist:

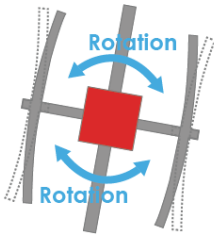
Beschleunigung entlang der X-Achse = (16384 / 65536 * 4) g = 1g

3-achsiger Gyroskop

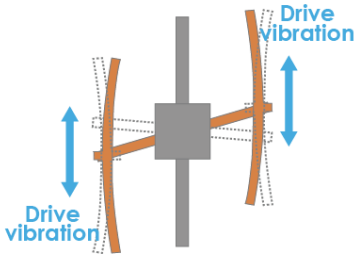
Gyroskope funktionieren nach dem Prinzip der Coriolis-Beschleunigung. Stellen Sie sich eine gabelähnliche Struktur vor, die sich ständig hin und her bewegt und durch piezoelektrische Kristalle an Ort und Stelle gehalten wird. Wenn Sie versuchen, diese Anordnung zu neigen, erfahren die Kristalle eine Kraft in Richtung der Neigung. Dies entsteht aufgrund der Trägheit der beweglichen Gabel. Die Kristalle erzeugen daraufhin einen Strom gemäß dem piezoelektrischen Effekt, der dann verstärkt wird.



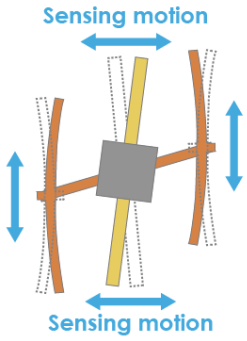
1.Normally, a drive arm vibrates in a certain direction.



2.Direction of rotation



3.When the gyro is rotated,the Coriolis force acts on the drive arms, producing vertical vibration.



4.The stationary part bends due to vertical drive arm vibration, producing a sensing motion in the sensing arms.

Auch der Gyroskop hat vier Arten von Messbereichen: +/- 250, +/- 500, +/- 1000, +/- 2000. Die Berechnungsmethode und die Beschleunigung sind im Grunde konsistent.

Die Formel zur Umwandlung der Ablesung in die Winkelgeschwindigkeit lautet wie folgt:

Winkelgeschwindigkeit = (Rohdaten der Gyroskopachsen / 65536 * voller Gyroskopbereich) °/s

Nehmen Sie als Beispiel die X-Achse, wenn die Rohdaten der Beschleunigungs-X-Achse 16384 betragen und der Bereich + / - 250 ° / s beträgt:

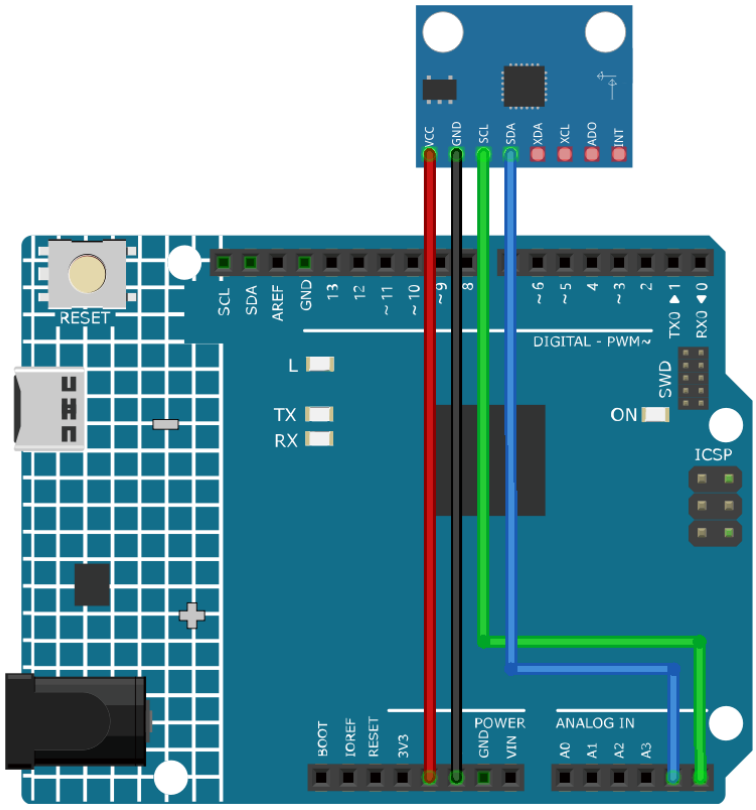
Winkelgeschwindigkeit entlang der X-Achse = (16384 / 65536 * 500)°/s = 125°/s

Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Beschleunigungssensor & Gyroskop Modul (MPU6050) * 1
- Jumperkabel

Schaltkreismontage



Programmcode

Bemerkung: Um die Bibliothek zu installieren, nutzen Sie den Arduino Bibliotheksmanager und suchen nach **„Adafruit MPU6050“**, um sie zu installieren.

Code-Erklärung

1. Der Code beginnt mit dem Einbinden der erforderlichen Bibliotheken und der Erstellung eines Objekts für den MPU6050-Sensor. Dieser Code verwendet die Adafruit_MPU6050-Bibliothek, die Adafruit_Sensor-Bibliothek und die Wire-Bibliothek. Die `Adafruit_MPU6050`-Bibliothek dient der Kommunikation mit dem MPU6050-Sensor und der Abfrage von Beschleunigungs-, Rotations- und Temperaturdaten. Die `Adafruit_Sensor`-Bibliothek bietet eine allgemeine Schnittstelle für verschiedene Sensortypen. Die `Wire`-Bibliothek wird für die I2C-Kommunikation verwendet, die für die Kommunikation mit dem MPU6050-Sensor erforderlich ist.

   Bemerkung: Um die Bibliothek zu installieren, nutzen Sie den Arduino Bibliotheksmanager und suchen nach **„Adafruit MPU6050“**, um sie zu installieren.

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
Adafruit_MPU6050 mpu;
```

2. Die `setup()`-Funktion initialisiert die serielle Kommunikation und prüft, ob der Sensor erkannt wird. Wenn der Sensor nicht gefunden wird, tritt das Arduino in eine Endlosschleife mit der Meldung „MPU6050-Chip nicht gefunden“ ein. Wird er gefunden, werden der Beschleunigungsbereich, der Gyrobereich und die Filterbandbreite eingestellt, und eine Verzögerung wird für die Stabilität hinzugefügt.

```
void setup(void) {
  // Initialize the serial communication
  Serial.begin(9600);

  // Check if the MPU6050 sensor is detected
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

  // set accelerometer range to +-8G
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);

  // set gyro range to +- 500 deg/s
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);

  // set filter bandwidth to 21 Hz
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

  // Add a delay for stability
  delay(100);
}
```

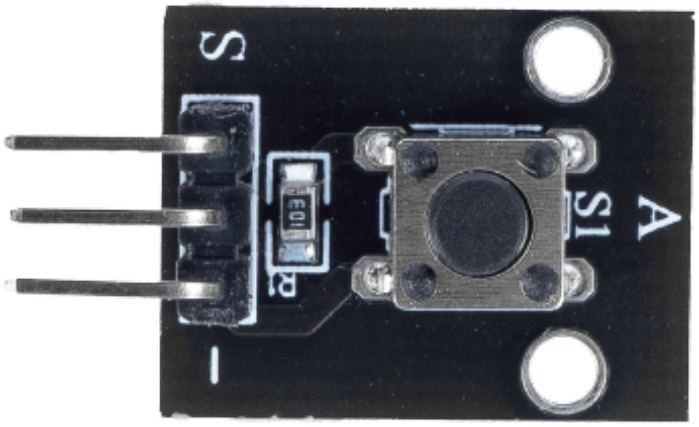
3. In der `loop()`-Funktion erstellt das Programm Events, um die Sensormesswerte zu speichern, und ruft dann diese Messwerte ab. Die Werte für Beschleunigung, Rotation und Temperatur werden dann auf dem seriellen Monitor ausgegeben.

```
void loop() {
  // Get new sensor events with the readings
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  // Print out the acceleration, rotation, and temperature readings
  // ...

  // Add a delay to avoid flooding the serial monitor
  delay(1000);
}
```

Weitere Ideen

- Visualisiere Sensordaten in grafischer Form auf einem LCD oder OLED

2.4.7 Tastenmodul



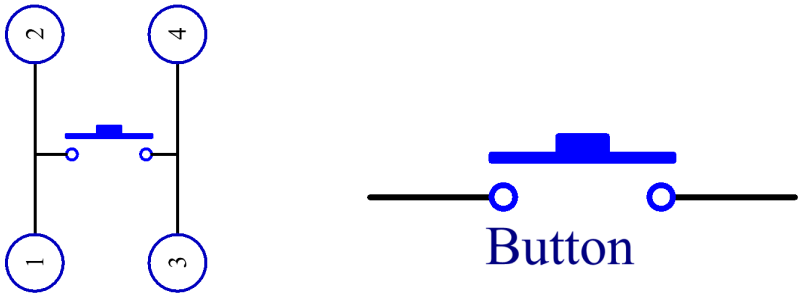
Einführung

Das Tastenmodul ist ein elektronisches Gerät zur Erfassung des Zustands einer Taste. Üblicherweise werden sie als Schalter zum Verbinden oder Unterbrechen von Schaltkreisen eingesetzt. Tasten finden in vielen Anwendungsbereichen Verwendung, etwa bei Türklingeln, Schreibtischlampen, Fernbedienungen, Aufzügen, Brandmeldern usw.

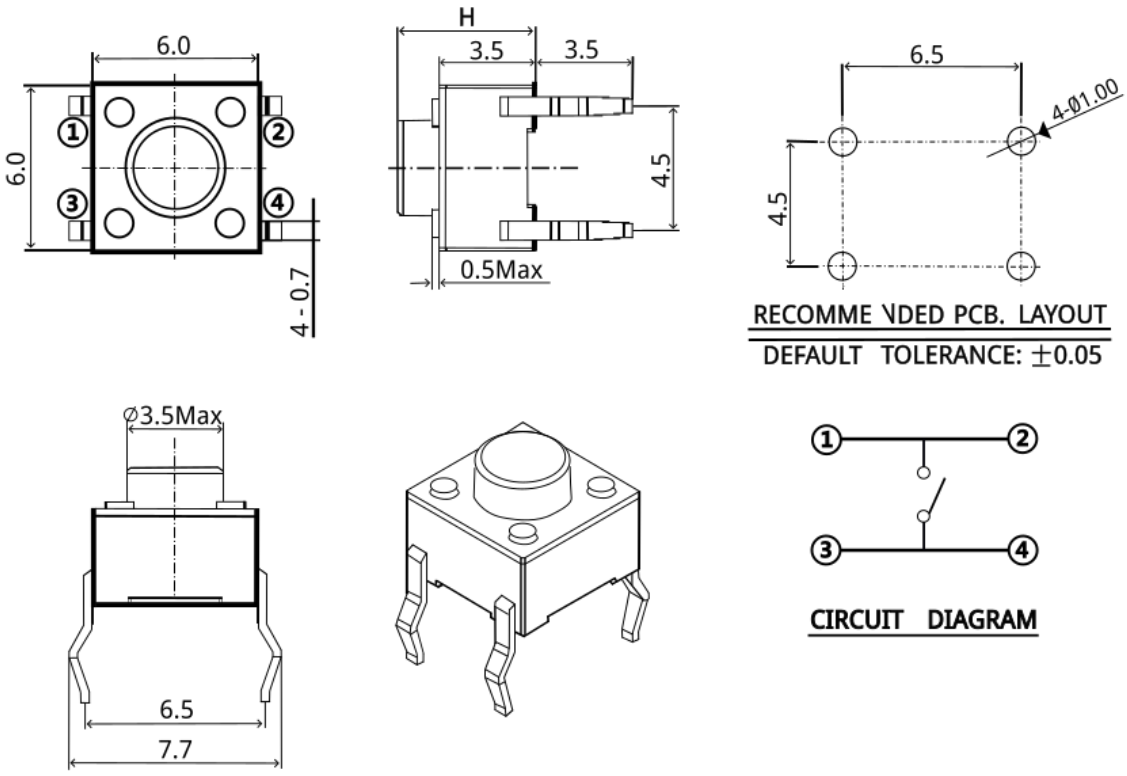
Funktionsprinzip

Das Tastenmodul funktioniert nach dem Prinzip eines Schalters. Ein Schalter ist ein elektrisches Bauteil, das dazu dient, einen Stromkreis zu öffnen oder zu schließen.

Die nachfolgende Abbildung zeigt den internen Aufbau einer Taste. Das Symbol rechts unten wird meist verwendet, um eine Taste in Schaltungen darzustellen.



Da der Pin 1 mit dem Pin 2 und der Pin 3 mit dem Pin 4 verbunden ist, schließt das Drücken der Taste alle 4 Pins und somit den Stromkreis.

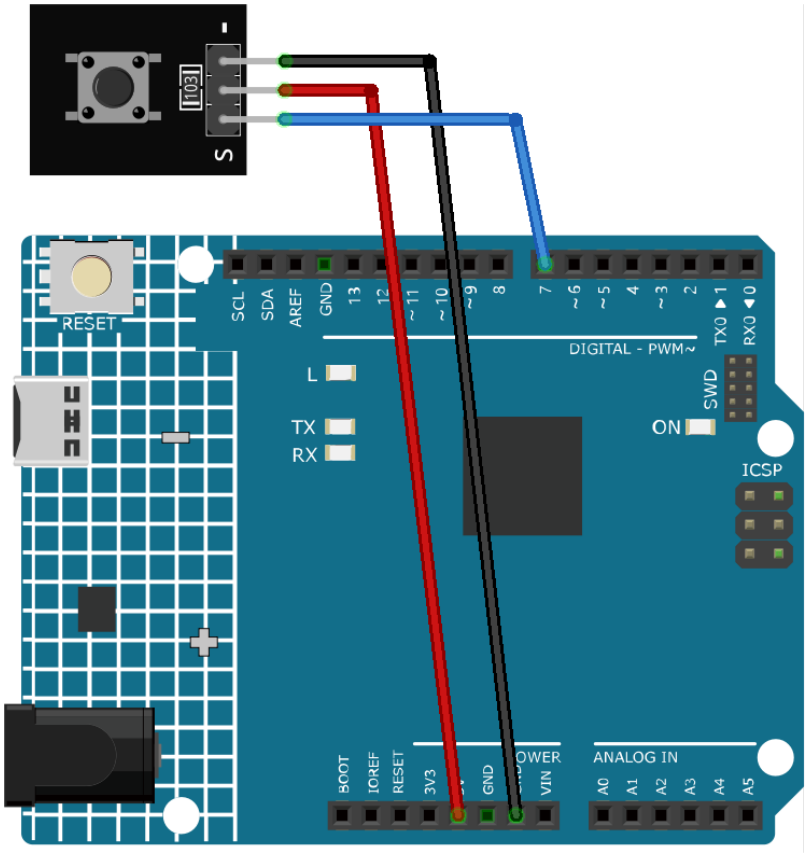


Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3 Platine * 1
- Tastenmodul * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erklärung

1. **Einrichtung**: Im ersten Abschnitt des Codes deklarieren wir zunächst `sensorPin` als eine Konstante vom Typ Integer und weisen ihr die Pin-Nummer zu, an die wir unsere Taste am Arduino-Board anschließen werden. Die Funktion `setup()` setzt den Modus von `sensorPin` auf `INPUT`, was bedeutet, dass wir Daten von der Taste über diesen Pin empfangen werden. Die Funktion `Serial.begin()` initiiert die serielle Kommunikation mit einer Baudrate von 9600.

```
const int sensorPin = 7;

void setup() {
  pinMode(sensorPin, INPUT);
  Serial.begin(9600);
}
```

2. **Die Schleife**: Die Funktion `loop()` enthält die Hauptlogik des Programms. Sie liest kontinuierlich den Zustand der Taste aus und gibt ihn alle 50 Millisekunden im seriellen Monitor aus. Die Funktion `digitalRead()` liest den Zustand der Taste, und die Funktion `Serial.println()` gibt diesen Wert im seriellen Monitor aus. Die Funktion `delay()` pausiert dann die Ausführung für 50 Millisekunden, bevor die nächste Messung erfolgt. Die Taste gibt ein niedriges Signal aus, wenn sie gedrückt wird, und ein hohes, wenn sie losgelassen wird.

```
void loop() {
  Serial.println(digitalRead(sensorPin));
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  delay(50);
}
```

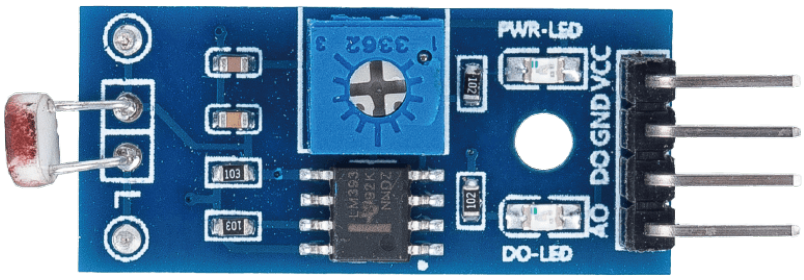
Weitere Ideen

- Verwenden Sie die Taste in Kombination mit if-Anweisungen, um unterschiedliche Szenarien in einem Programm zu steuern.
- Lassen Sie die Taste eine LED ein- und ausschalten, anstatt nur Ausgaben im seriellen Monitor zu erzeugen.

Weitere Projekte

- *Türklingel*

2.4.8 Fotowiderstands-Modul



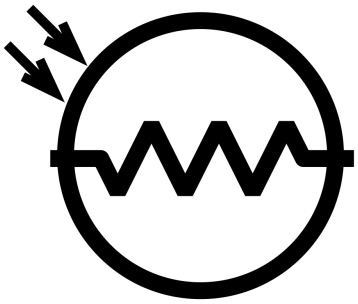
Einleitung

Ein Fotowiderstands-Modul ist ein Gerät zur Messung der Lichtintensität in der Umgebung. Es kann für verschiedene Zwecke eingesetzt werden, wie etwa zur Anpassung der Helligkeit eines Geräts, zur Erkennung von Tag und Nacht oder zur Aktivierung eines Lichtschalters.

Ein wesentlicher Bestandteil des Fotowiderstands-Moduls ist der Fotowiderstand selbst. Ein Fotowiderstand ist ein lichtgesteuerter variabler Widerstand. Mit steigender Lichteinstrahlung verringert sich der Widerstand des Fotowiderstands; er zeigt also eine Foto-Leitfähigkeit.

Der Fotowiderstand kann in lichtempfindlichen Detektorschaltungen sowie in licht- und dunkelaktivierten Schaltkreisen als Halbleiterwiderstand eingesetzt werden. Im Dunkeln kann der Widerstand eines Fotowiderstands mehrere Megaohm (M) betragen, während er im Hellen auf nur wenige hundert Ohm sinken kann.

Hier ist das elektronische Symbol für den Fotowiderstand:



Funktionsprinzip

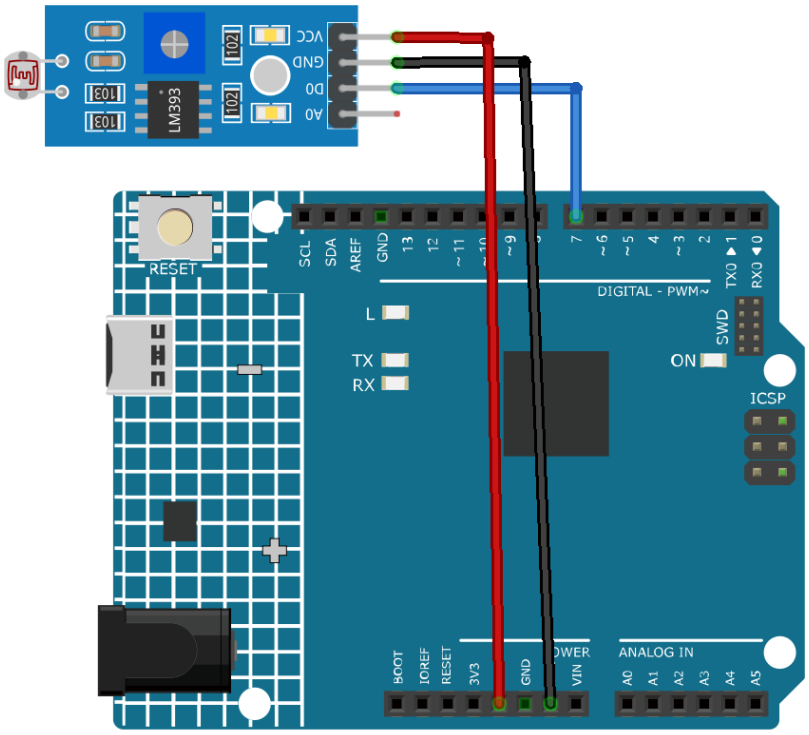
Das Fotowiderstands-Modul arbeitet nach dem Prinzip der Widerstandsveränderung in Abhängigkeit von unterschiedlichen Lichtintensitäten. Der Sensor verfügt über ein integriertes Potentiometer, das den digitalen Ausgang (D0) des Sensors anpasst. Überschreitet die Lichtintensität einen bestimmten Schwellenwert, ändert sich der Widerstand des Sensors. Diese Widerstandsänderung wird dann in ein elektrisches Signal umgewandelt, das vom Arduino-Board ausgelesen werden kann.

Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Fotowiderstands-Modul * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erklärung

1. Diese Codezeile definiert die Pin-Nummer, an die der Fotowiderstandssensor auf dem Arduino-Board angeschlossen ist.

```
const int sensorPin = 7;
```

2. Die Funktion `setup()` wird nur einmal ausgeführt, wenn das Arduino-Board eingeschaltet oder zurückgesetzt wird. In diesem Projekt wird der sensorPin als INPUT festgelegt, da wir von ihm Werte ablesen. Der Befehl Serial.begin(9600) initiiert die serielle Kommunikation mit einer Baudrate von 9600.

```
void setup() {
  pinMode(sensorPin, INPUT);
  Serial.begin(9600);
}
```

3. Die Funktion `loop()` ist die Hauptfunktion, in der das Programm fortlaufend abläuft. In dieser Funktion liest die Funktion digitalRead den digitalen Wert vom Fotowiderstandssensor und gibt ihn mit Serial.println auf dem seriellen Monitor aus. Der Befehl `delay(50)` sorgt für eine Wartezeit von 50 Millisekunden vor der nächsten Messung.

```
void loop() {
  Serial.println(digitalRead(sensorPin));
  delay(50);
}
```

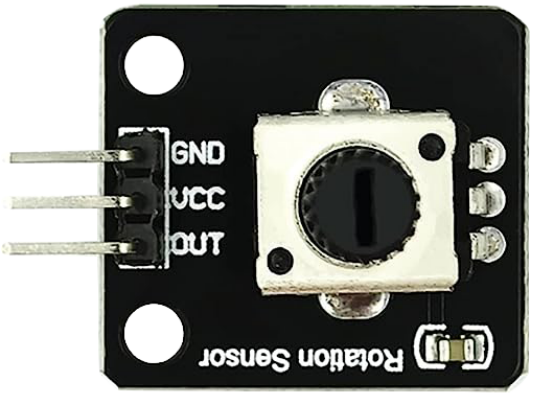
Weitere Ideen

- Verwenden Sie den Sensor, um eine LED oder ein Relais ein- und auszuschalten.
- Plotten Sie den **analogen Ausgang** anstelle von einfachem digitalen HIGH/LOW. Verwenden Sie dafür den **AO**-Pin.

Weitere Projekte

- *Lichtsteuerungsschalter*

2.4.9 Potentiometer-Modul



Einleitung

Das Potentiometer-Modul ist ein elektronisches Bauelement, dessen Widerstandswert sich je nach Stellung des Drehknopfs ändert. Es findet vielseitige Anwendungsbereiche, wie zum Beispiel die Lautstärkeregelung eines Lautsprechers, die Helligkeitssteuerung einer LED oder die Geschwindigkeitskontrolle eines Motors.

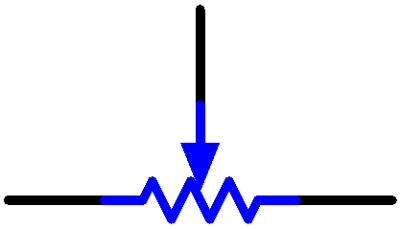
Funktionsprinzip

Ein Potentiometer ist ebenfalls ein Widerstandsbauelement mit drei Anschlüssen, dessen Widerstandswert gemäß einer bestimmten Regelung angepasst werden kann.

Potentiometer gibt es in verschiedenen Formen, Größen und Werten, aber sie haben alle folgende Eigenschaften gemeinsam:

- Sie verfügen über drei Anschlüsse.
- Sie haben einen Drehknopf, eine Schraube oder einen Schieber, um den Widerstand zwischen dem mittleren und einem der äußeren Anschlüsse zu variieren.
- Der Widerstand zwischen dem mittleren und einem der äußeren Anschlüsse variiert von 0 bis zum maximalen Widerstand des Potentiometers, wenn der Drehknopf, die Schraube oder der Schieber bewegt wird.

Hier ist das Schaltungssymbol eines Potentiometers.



Die Funktionen des Potentiometers im Stromkreis sind wie folgt:

1. **Als Spannungsteiler**
   Das Potentiometer ist ein kontinuierlich einstellbarer Widerstand. Wenn man die Welle oder den Schiebegriff des Potentiometers verstellt, bewegt sich der gleitende Kontakt auf dem Widerstandselement. Dabei

kann eine Spannung in Abhängigkeit von der am Potentiometer anliegenden Spannung und dem Drehwinkel oder dem Verfahrweg des beweglichen Armes abgegriffen werden.

2. **Als Rheostat**
   Wenn das Potentiometer als Rheostat eingesetzt wird, werden der mittlere Pin und einer der anderen beiden Pins im Stromkreis verbunden. So erhält man einen stufenlos und kontinuierlich veränderbaren Widerstandswert innerhalb des Verfahrwegs des gleitenden Kontakts.

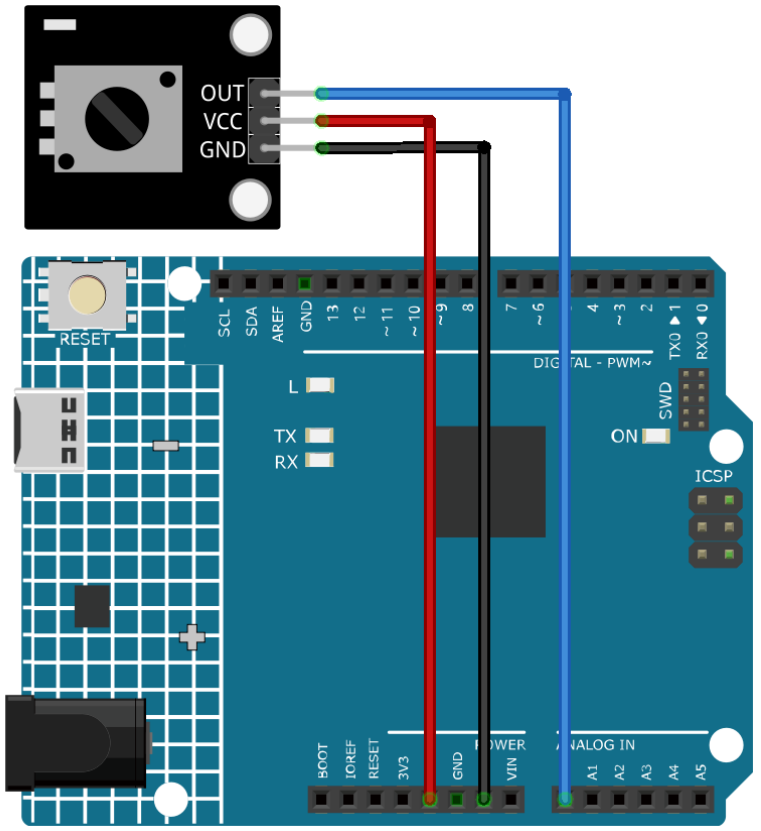
3. **Als Stromregler**
   Wenn das Potentiometer als Stromregler fungiert, muss der gleitende Kontakt als einer der Ausgangsanschlüsse verbunden werden.

Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Potentiometer-Modul * 1
- Jumper-Kabel

Schaltungsaufbau



Programmcode

Code-Erläuterung

1. Diese Codezeile definiert die Pin-Nummer, an die das Potentiometer am Arduino-Board angeschlossen ist.

```
const int sensorPin = A0;
```

2. Die `setup()`-Funktion ist eine spezielle Funktion im Arduino-Umfeld, die nur einmal ausgeführt wird, wenn das Arduino-Board eingeschaltet oder zurückgesetzt wird. In diesem Projekt wird die serielle Kommunikation mit `Serial.begin(9600)` bei einer Baudrate von 9600 eingeleitet.

```
void setup() {
  Serial.begin(9600);
}
```

3. Die `loop()`-Funktion ist die Hauptfunktion, in der das Programm wiederholt ausgeführt wird. In dieser Funktion liest die `analogRead()`-Funktion den Analogwert vom Potentiometer und gibt ihn mit `Serial.println()` im seriellen Monitor aus. Der Befehl `delay(50)` lässt das Programm für 50 Millisekunden pausieren, bevor der nächste Wert gelesen wird.

```
void loop() {
  Serial.println(analogRead(sensorPin));
  delay(50);
}
```

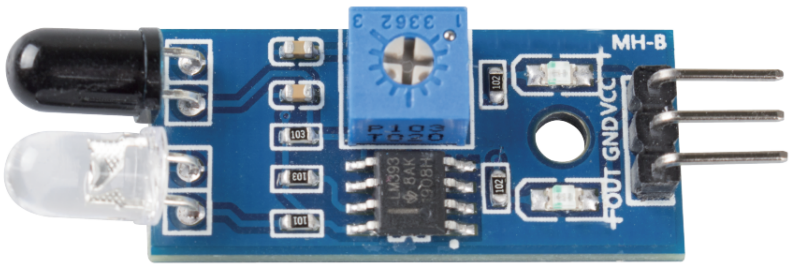
Weitere Ideen

- Steuerung der LED-Helligkeit: Der Analogwert des Potentiometers könnte verwendet werden, um die Helligkeit einer an einem PWM-fähigen Pin des Arduino angeschlossenen LED zu steuern.
- Steuerung der Position eines Servomotors: Durch die Zuordnung des Analogwerts zum Bereich der Position des Servomotors (in der Regel von 0 bis 180 Grad) könnte das Potentiometer als Controller für den Servomotor dienen.

Weitere Projekte

- *Potentiometer-Skalenwert*

2.4.10 IR-Hindernisvermeidungssensor-Modul

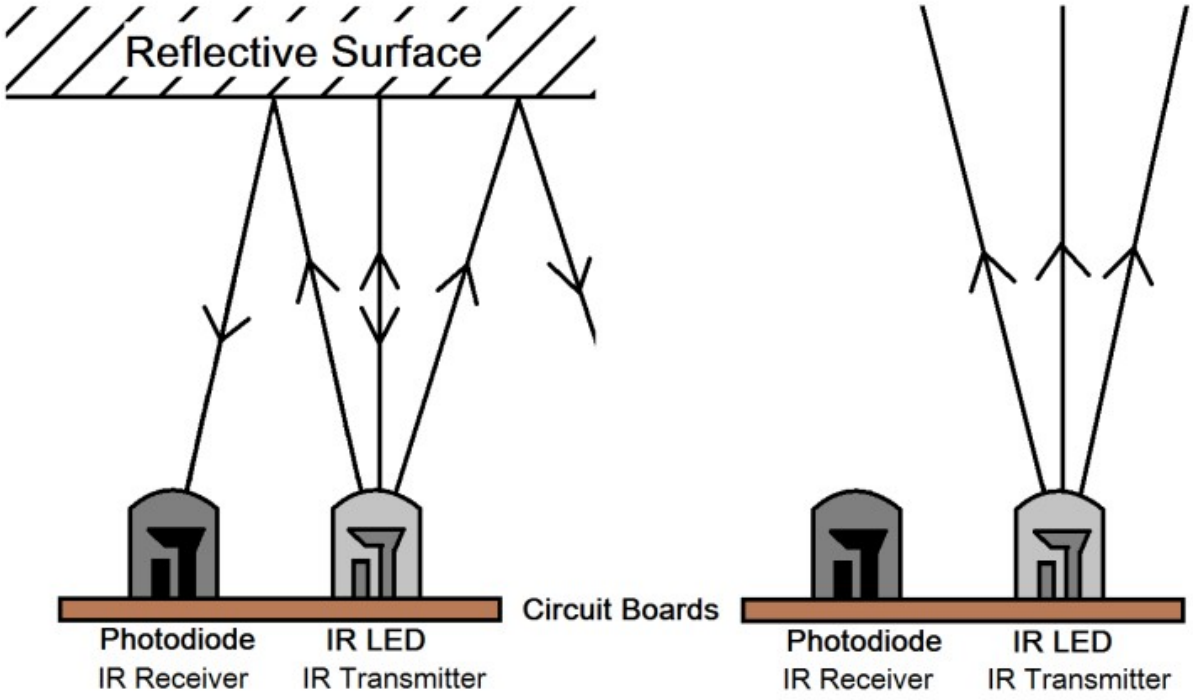


Einleitung

Ein IR-Hindernisvermeidungssensor funktioniert nach dem Prinzip der Infrarotreflexion, um Hindernisse zu detektieren. Wenn kein Objekt im Weg ist, empfängt der Infrarotempfänger keine Signale. Befindet sich jedoch ein Hindernis im Weg, das das Infrarotlicht reflektiert, dann nimmt der Infrarotempfänger Signale auf.

Funktionsweise

Ein Hindernisvermeidungssensor setzt sich in der Regel aus einem Infrarotsender, einem Infrarotempfänger und einem Potentiometer zusammen. Je nach Reflektionseigenschaften des Objekts schwächt der ausgesendete Infrarotstrahl mit zunehmender Entfernung ab und verschwindet letztlich. Trifft der Strahl auf ein Hindernis, wird er zurück zum Infrarotempfänger reflektiert. Dieser erkennt das Signal und signalisiert ein Hindernis im Pfad. Die Erfassungsreichweite kann über das integrierte Potentiometer justiert werden.

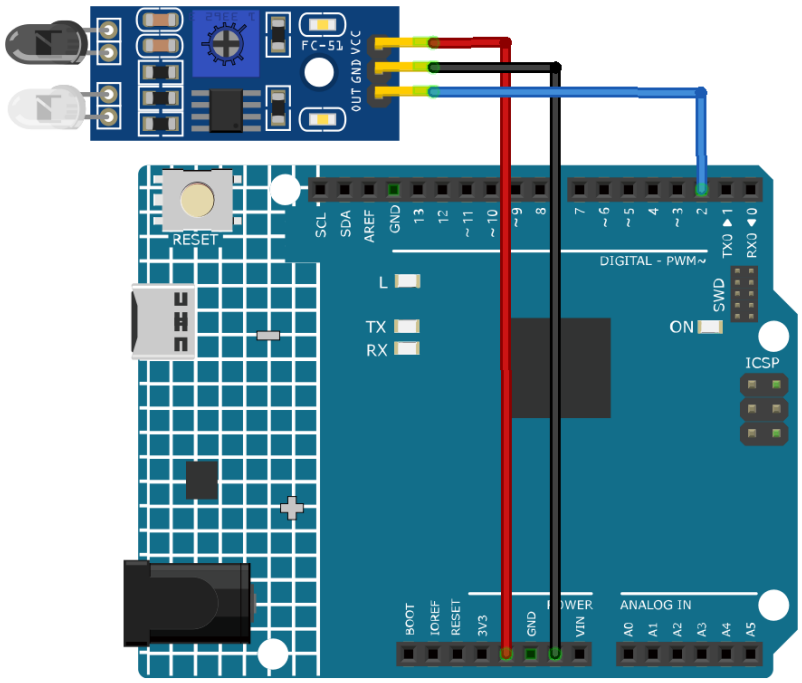


Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- IR-Hindernisvermeidungssensor-Modul * 1
- Verbindungskabel

Schaltungsaufbau



Programmcode

Code-Erläuterung

1. Sensoranschlusspin festlegen:

```
const int sensorPin = 2;
```

Schließen Sie den Ausgangspin des Sensors an den Arduino-Pin 2 an.

2. Serielle Kommunikation initialisieren und Sensorpin als Eingang definieren:

```
void setup() {
  pinMode(sensorPin, INPUT);
  Serial.begin(9600);
}
```

Initialisieren Sie die serielle Kommunikation mit einer Baudrate von 9600 für die Ausgabe auf dem seriellen Monitor. Setzen Sie den Sensorpin als Eingang, um das Signal einzulesen.

3. Sensorwert auslesen und im seriellen Monitor anzeigen:

```
void loop() {
  Serial.println(digitalRead(sensorPin));
  delay(50);
}
```

Lesen Sie kontinuierlich den digitalen Wert vom Sensorpin mit `digitalRead()` und geben Sie diesen Wert mit `Serial.println()` im seriellen Monitor aus. Fügen Sie eine 50-ms-Pause zwischen den Ausgaben ein, um die Ansicht zu verbessern.

Bemerkung: Sollte der Sensor nicht wie erwartet arbeiten, justieren Sie den Infrarotsender und -empfänger so, dass sie parallel zueinander stehen. Zusätzlich können Sie die Erfassungsreichweite mittels des eingebauten Potentiometers anpassen.

Weitere Ideen

- Integrieren Sie einen Summer, der bei Hinderniserkennung einen Ton ausgibt

Weitere Projekte

- *Automatischer Seifenspender*

2.4.11 Kapazitives Bodenfeuchtigkeitsmodul

Einführung

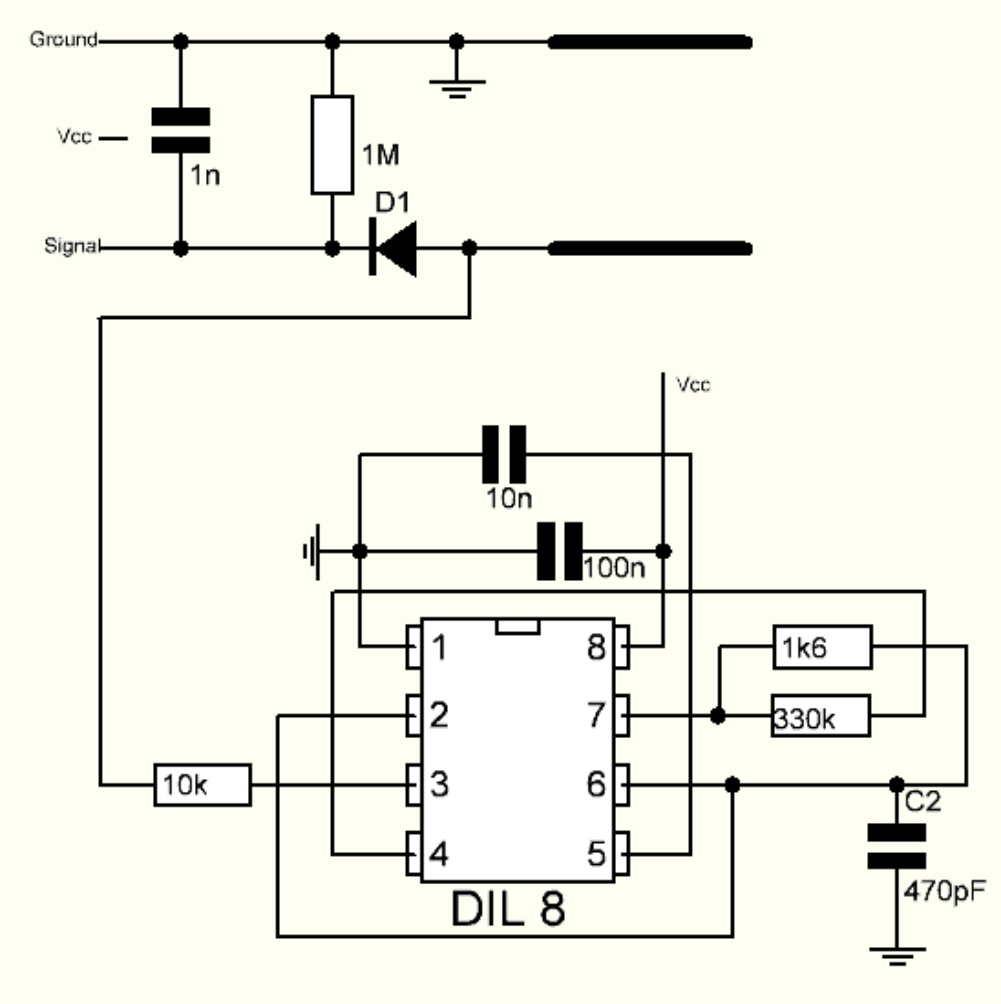
Das Bodenfeuchtigkeitsmodul ist ein Sensor zur Messung des Feuchtigkeitsgehalts von Erde. Es findet insbesondere in der Landwirtschaft Anwendung, um den Feuchtigkeitszustand des Bodens zu überwachen und Landwirten eine bessere Bewässerungsplanung zu ermöglichen.

Funktionsprinzip

Im Gegensatz zu den meisten resistiven Sensoren, die derzeit auf dem Markt erhältlich sind, basiert dieses kapazitive Bodenfeuchtigkeitsmodul auf dem Prinzip der kapazitiven Induktion. Dadurch wird das Problem der Korrosionsanfälligkeit, das bei resistiven Sensoren auftritt, effektiv umgangen, was die Lebensdauer des Sensors erheblich verlängert.

Der Sensor besteht aus korrosionsbeständigen Materialien und bietet eine ausgezeichnete Langlebigkeit. Einfach in die Erde neben den Pflanzen einstecken und die Bodenfeuchtigkeit in Echtzeit überwachen. Das Modul enthält einen integrierten Spannungsregler und arbeitet in einem Spannungsbereich von 3,3 bis 5,5 V, wodurch es für Mikrocontroller mit 3,3 V und 5 V Versorgungsspannung ideal ist.

Die Hardware-Schaltung des kapazitiven Bodenfeuchtigkeitssensors ist unten dargestellt.



Der Sensor verfügt über einen festfrequenten Oszillator, der mit einem 555-Timer-IC realisiert ist. Das generierte Rechtecksignal wird dem Sensor zugeführt, der als Kondensator wirkt. Das Signal durchläuft eine Reaktanz, die in Kombination mit einem rein ohmschen Widerstand (10k Widerstand am Pin 3) einen Spannungsteiler bildet.

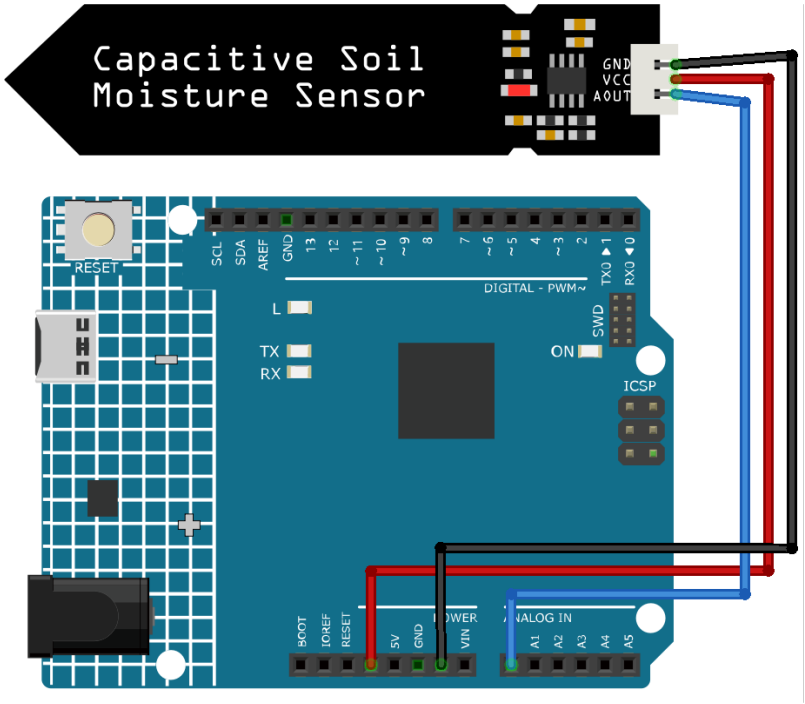
Je höher die Bodenfeuchtigkeit, desto größer ist die Kapazität des Sensors. Dadurch verringert sich die Reaktanz des Rechtecksignals, was die Spannung auf der Signalleitung reduziert und den Analogeingangswert am Mikrocontroller entsprechend verkleinert.

Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Bodenfeuchtigkeitsmodul * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Codeerklärung

1. Definition des Sensorpins:

```
const int sensorPin = A0;
```

In diesem Codeabschnitt wird eine Konstante mit dem Namen *sensorPin* definiert und dem Wert A0 zugewiesen, der dem analogen Eingangspin auf dem Arduino-Board entspricht, an den der Bodenfeuchtigkeitssensor angeschlossen ist.

2. Initialisierung der seriellen Kommunikation:

```
void setup() {
  Serial.begin(9600);
}
```

Die Funktion `setup()` wird einmal aufgerufen, wenn der Arduino eingeschaltet oder zurückgesetzt wird. Hier initialisieren wir die serielle Kommunikation mit einer Baudrate von 9600.

3. Daten lesen und auf dem seriellen Monitor ausgeben:

```
void loop() {
  Serial.println(analogRead(sensorPin));
  delay(500);
}
```

In der *loop()*-Funktion wird die Hauptlogik des Programms ausgeführt. Diese Schleife läuft ununterbrochen, sobald das Programm gestartet ist. Wir verwenden die Funktion `analogRead()`, um die Daten vom Feuchtigkeitssensor zu lesen und sie auf dem seriellen Monitor auszugeben. Anschließend wird das Programm für 500 Millisekunden pausiert, bevor der nächste Wert erfasst wird.

Bemerkung: Je kleiner der Wert, desto höher ist der Feuchtigkeitsgehalt im Boden.

Weitere Ideen

- Integration eines Buzzers oder einer LED, die aktiviert wird, wenn der Feuchtigkeitswert unter einen bestimmten Schwellenwert fällt. So erhalten Sie einen physischen Hinweis, wann es Zeit ist, Ihre Pflanzen zu gießen.
- Automatisierung des Bewässerungsprozesses durch Anschluss einer Wasserpumpe. Fällt die Bodenfeuchtigkeit unter einen bestimmten Wert, kann der Arduino die Pumpe aktivieren, um die Pflanzen zu bewässern.

Weitere Projekte

- *Pflanzenüberwachung mit Blynk*
- *Automatisches Bewässerungssystem mit Blynk*

2.4.12 Regentropfen-Erkennungsmodul

Einleitung

Ein Regentropfensensor oder auch Regenfall-Erkennungssensor wird genutzt, um festzustellen, ob es regnet und wie stark der Niederschlag ist. Er findet breite Anwendung in automatischen Scheibenwischersystemen, intelligenten Beleuchtungssystemen und Schiebedachsystemen von Automobilen.

Funktionsprinzip

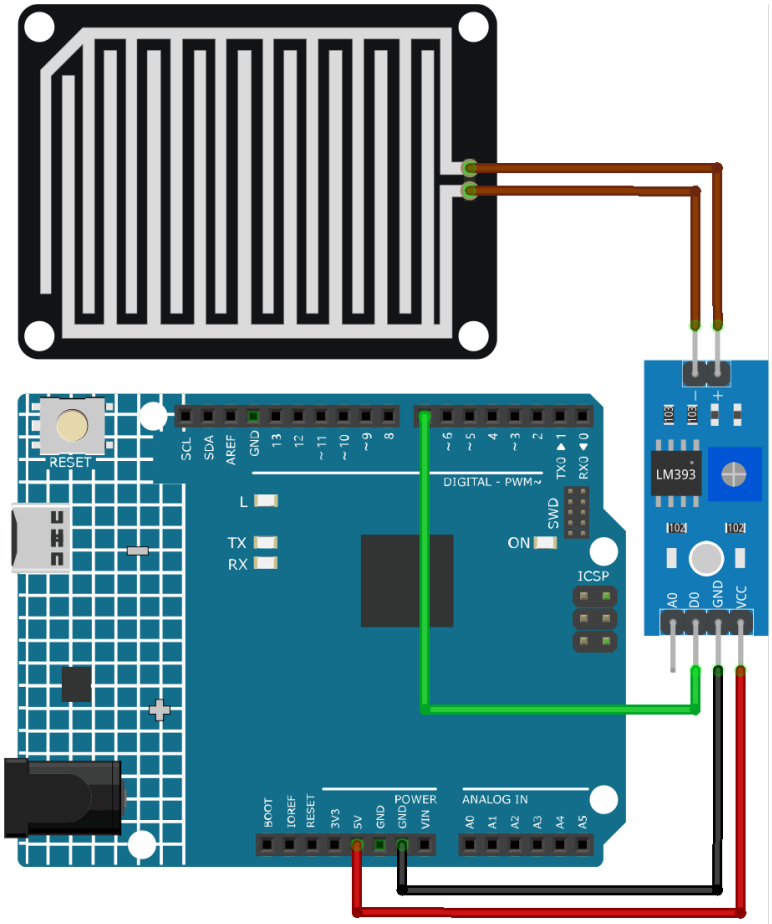
Der Regentropfensensor besteht im Grunde aus einer Platte, die mit Nickel in Linienform beschichtet ist. Er arbeitet auf dem Prinzip des elektrischen Widerstands. Wenn die Platte trocken ist, ist der Widerstand hoch und dementsprechend ist auch die Spannung nach der Formel V=IR hoch. Bei Kontakt mit Regentropfen sinkt der Widerstand, da Wasser ein elektrischer Leiter ist und die Nickel-Linien parallel schließt, was den Widerstand und den Spannungsabfall reduziert. Je stärker der Regenfall, desto geringer ist der Widerstand.

Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3 Platine * 1
- Regentropfen-Erkennungsmodul * 1
- Verbindungskabel

Schaltungsaufbau



Code

Code-Erklärung

1. Sensorpin definieren Hier wird eine Konstante des Typs Integer namens `sensorPin` definiert und dem Wert 7 zugewiesen. Dies entspricht dem digitalen Pin auf dem Arduino-Board, an den der Regentropfen-Erkennungssensor angeschlossen ist.

```
const int sensorPin = 7;
```

2. Pin-Modus einstellen und serielle Kommunikation initiieren In der `setup()`-Funktion werden zwei wesentliche Schritte durchgeführt. Erstens wird mit `pinMode()` der `sensorPin` als Eingang konfiguriert, damit digitale Werte vom Regentropfensensor gelesen werden können. Zweitens wird die serielle Kommunikation mit einer Baudrate von 9600 initialisiert.

```
void setup() {
  pinMode(sensorPin, INPUT);
  Serial.begin(9600);
}
```

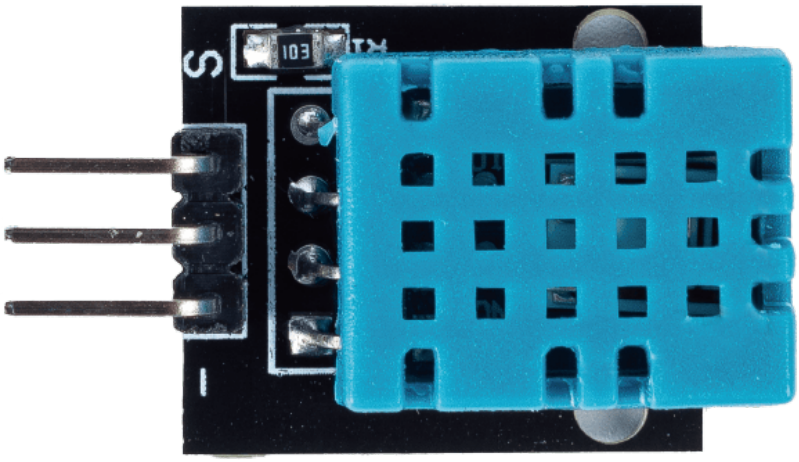
3. Den digitalen Wert lesen und an das serielle Monitor senden. Die `loop()`-Funktion liest den digitalen Wert vom Regentropfensensor mit `digitalRead()` aus. Dieser Wert (entweder HIGH oder LOW) wird an das serielle Monitor ausgegeben. Das Programm wartet dann 50 Millisekunden, bevor die nächste Messung erfolgt.

```
void loop() {
  Serial.println(digitalRead(sensorPin));
  delay(50);
}
```

Weitere Ideen

- Fügen Sie eine LED-Anzeige hinzu, die aufleuchtet, wenn Regen erkannt wird.
- Verbinden Sie einen Summer mit dem Arduino, um einen Alarmton auszulösen, wenn Regen erkannt wird. Dies könnte als Frühwarnsystem für Veranstaltungen wie Picknicks oder Outdoor-Aktivitäten dienen.

2.4.13 Temperatur- und Feuchtigkeitssensor-Modul (DHT11)



Einführung

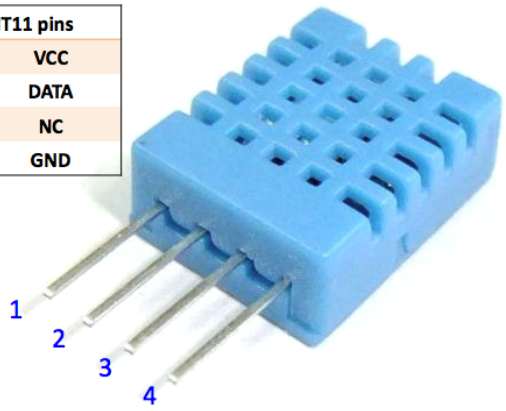
Der digitale Temperatur- und Feuchtigkeitssensor DHT11 ist ein kombinierter Sensor, der kalibrierte digitale Signalausgaben für Temperatur und Feuchtigkeit liefert. Die Verbindung aus spezieller digitaler Modultechnologie und Temperatur- sowie Feuchtigkeitssensorik sorgt für hohe Zuverlässigkeit und langfristige Stabilität des Produkts.

Der Sensor setzt sich aus einer resistiven Feuchtigkeitsmesskomponente und einer NTC-Temperaturmesseinheit zusammen, die mit einem leistungsfähigen 8-Bit-Mikrocontroller verbunden sind.

Funktionsprinzip

Der Sensor hat lediglich drei benutzbare Anschlüsse: VCC, GND und DATA. Der Kommunikationsprozess beginnt mit Startsignalen, die die DATA-Leitung an den DHT11 sendet. Nach dem Empfang der Signale sendet der DHT11 ein Antwortsignal zurück. Anschließend empfängt der Host das Antwortsignal und beginnt mit dem Empfang der 40-Bit-Temperatur- und Feuchtigkeitsdaten (8-Bit Feuchtigkeit Ganzzahl + 8-Bit Feuchtigkeit Dezimal + 8-Bit Temperatur Ganzzahl + 8-Bit Temperatur Dezimal + 8-Bit Prüfsumme).

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND

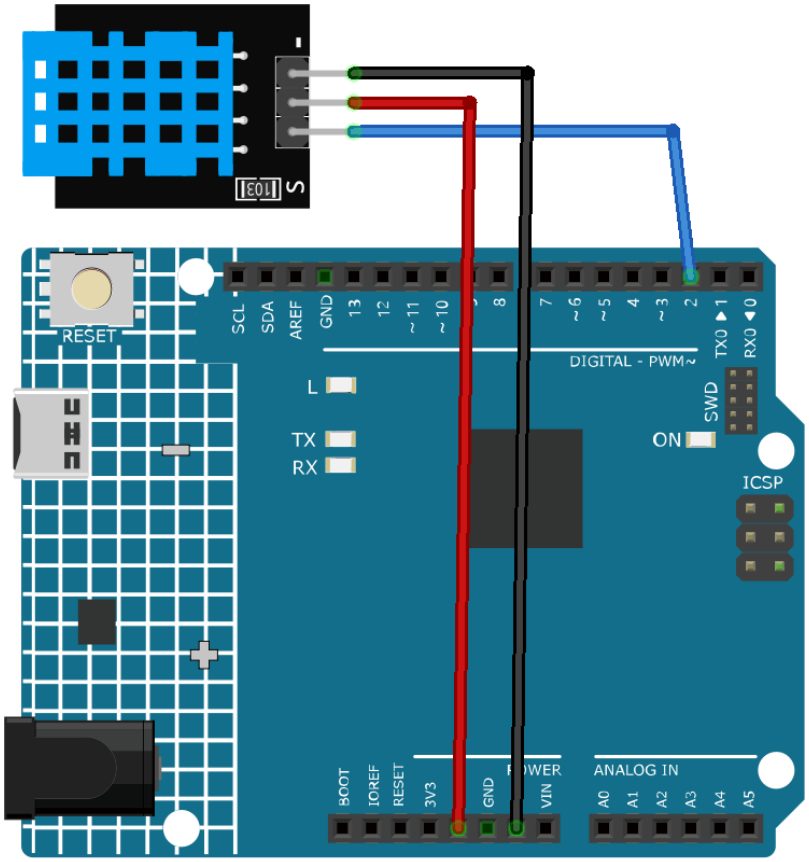


Anwendungsbeispiele

Benötigte Hardware-Komponenten

- Arduino Uno R4 oder R3 Platine * 1
- Temperatur- und Feuchtigkeitssensor-Modul (DHT11) * 1
- Jumperkabel

Schaltungsaufbau



Code

Bemerkung: Zur Installation der Bibliothek nutzen Sie den Arduino Library Manager und suchen Sie nach **„DHT sensor library“** und installieren Sie diese.

Code-Erklärung

1. Einbindung der erforderlichen Bibliotheken und Definition der Konstanten. Dieser Codeabschnitt enthält die DHT-Sensorbibliothek und definiert die verwendete Pinnummer und den Sensortyp für dieses Projekt.

   Bemerkung: Zur Installation der Bibliothek nutzen Sie den Arduino Library Manager und suchen Sie nach **„DHT sensor library“** und installieren Sie diese.

```
#include <DHT.h>
#define DHTPIN 2       // Define the pin used to connect the sensor
#define DHTTYPE DHT11  // Define the sensor type
```

2. Erstellung eines DHT-Objekts. Hier erstellen wir ein DHT-Objekt mit der definierten Pinnummer und dem definierten Sensortyp.

```
DHT dht(DHTPIN, DHTTYPE);  // Create a DHT object
```

3. Initialisierungsfunktion. Diese Funktion wird einmalig beim Start des Arduino ausgeführt. Hier initialisieren wir die serielle Kommunikation und den DHT-Sensor.

```
void setup() {
  Serial.begin(9600);
  Serial.println(F("DHT11 test!"));
  dht.begin();  // Initialize the DHT sensor
}
```

4. Hauptloop. Die `loop()`-Funktion läuft kontinuierlich nach der Setup-Funktion. Hier lesen wir die Feuchtigkeits- und Temperaturwerte aus, berechnen den Hitzeindex und geben diese Werte an den seriellen Monitor weiter. Sollte der Sensorauslesevorgang fehlschlagen (NaN zurückgeben), wird eine Fehlermeldung ausgegeben.

   Bemerkung: Der ist ein Maß für das gefühlte Außentemperatur, das durch Kombination von Lufttemperatur und Luftfeuchtigkeit ermittelt wird.

```
void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  float hif = dht.computeHeatIndex(f, h);
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.print(f);
  Serial.print(F("°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
  Serial.print(hif);
  Serial.println(F("°F"));
}
```

Weitere Ideen

- Anzeige der Messwerte auf einem LCD- oder OLED-Display

Weitere Projekte

- *Pflanzenüberwachung mit Blynk*
- *Bluetooth-Umweltmonitor*

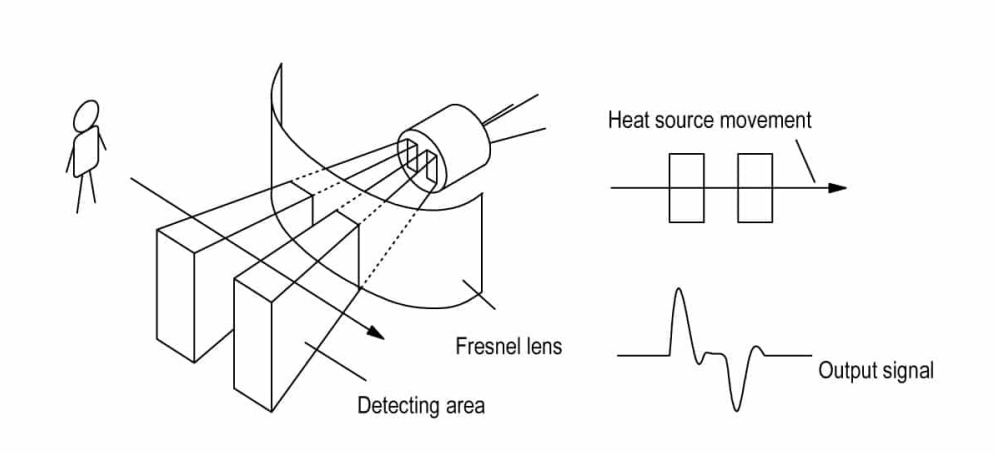
2.4.14 PIR-Bewegungsmelder-Modul (HC-SR501)

Einleitung

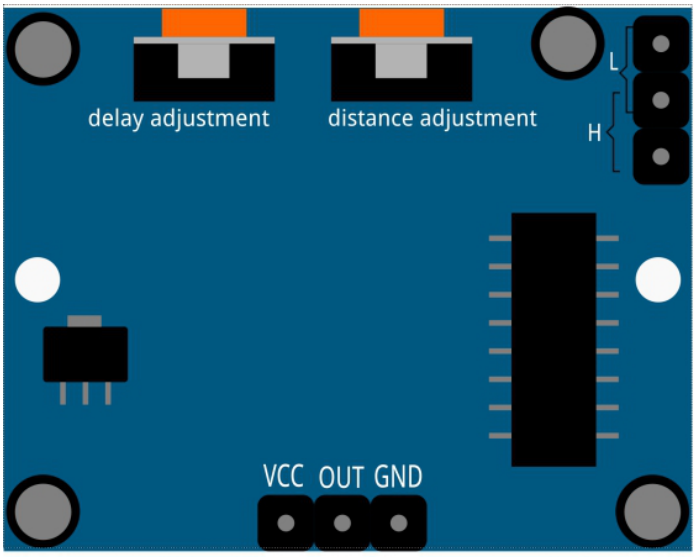
Der Passive Infrarot (PIR) Bewegungsmelder ist ein Sensor zur Erfassung von Bewegungen. Er findet häufig Einsatz in Sicherheitssystemen und automatischen Beleuchtungsanlagen. Der Sensor besitzt zwei Schlitze, die Infrarotstrahlung erkennen. Passiert ein Objekt, etwa eine Person, den Erfassungsbereich des Sensors, so registriert dieser eine Veränderung der Infrarotstrahlung und löst ein Ausgangssignal aus.

Funktionsprinzip

Der PIR-Sensor ist in zwei Schlitze unterteilt, die mit einem Differenzverstärker verbunden sind. Wenn ein stationäres Objekt vor dem Sensor ist, empfangen beide Schlitze dieselbe Menge an Strahlung, und der Ausgang bleibt neutral. Bewegt sich jedoch ein Objekt vor dem Sensor, nimmt einer der Schlitze mehr Strahlung auf als der andere, was zu Schwankungen im Ausgangssignal führt. Diese Veränderung der Ausgangsspannung ist ein Indikator für erkannte Bewegung.



Nach der Verkabelung des Sensormoduls erfolgt eine einminütige Initialisierung. Während dieser Zeit kann das Modul 0 bis 3-mal in Abständen ein Signal ausgeben. Danach geht das Modul in den Standby-Modus. Um Fehlbedienungen durch störende Signale zu vermeiden, sollte der Sensor von Lichtquellen und anderen Störeinflüssen ferngehalten werden. Am besten wird der Sensor in einer windgeschützten Umgebung eingesetzt, da auch Wind die Sensorik beeinträchtigen kann.



Reichweiteneinstellung

Durch Drehen des Potentiometers für die Reichweiteneinstellung im Uhrzeigersinn wird die Erfassungsreichweite erhöht, mit einem Maximalbereich von etwa 0-7 Metern. Dreht man es gegen den Uhrzeigersinn, verringert sich die Reichweite auf etwa 0-3 Meter.

Verzögerungseinstellung

Dreht man das Potentiometer für die Verzögerungseinstellung im Uhrzeigersinn, erhöht sich die Verzögerungszeit. Die maximale Verzögerungszeit kann bis zu 300 Sekunden betragen. In entgegengesetzter Richtung verringert sich die Verzögerung auf ein Minimum von 5 Sekunden.

Zwei Trigger-Modi

Durch Verwendung einer Jumperkappe können verschiedene Modi ausgewählt werden.

- H: Wiederholbarer Trigger-Modus. Nach der Erfassung einer Bewegung gibt das Modul ein hohes Signal aus. Während der anschließenden Verzögerungszeit bleibt das Ausgangssignal bei erneuter Bewegungserkennung hoch.
- L: Nicht wiederholbarer Trigger-Modus. Nach der Erfassung einer Bewegung gibt das Modul ein hohes Signal aus, das nach Ablauf der Verzögerungszeit automatisch auf ein niedriges Niveau zurückfällt.

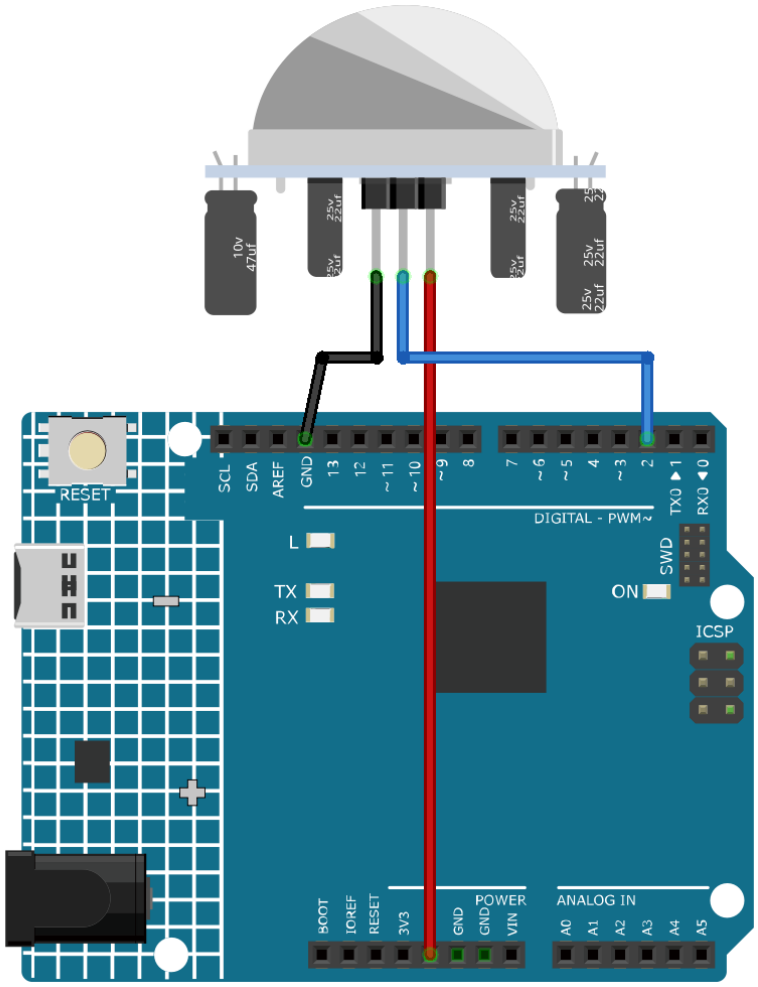
Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Platine * 1
- PIR-Bewegungsmelder-Modul * 1
- Jumperkabel

Schaltungsaufbau

Bemerkung: Die Pin-Markierungen sind durch die Fresnel-Linse verdeckt. Zum Ansehen kann diese geöffnet werden.



Programmcode

Code-Erklärung

1. Einrichten des PIR-Sensor-Pins. Der Pin für den PIR-Sensor wird als Pin 2 definiert.

```
const int pirPin = 2;
int state = 0;
```

2. Initialisierung des PIR-Sensors. In der `setup()` Funktion wird der Pin des PIR-Sensors als Eingang definiert. Dadurch kann der Arduino den Status des PIR-Sensors lesen.

```
void setup() {
  pinMode(pirPin, INPUT);
  Serial.begin(9600);
}
```

3. Auslesen des PIR-Sensors und Anzeige der Ergebnisse. In der `loop()` Funktion wird der Status des PIR-Sensors kontinuierlich ausgelesen.

```
void loop() {
  state = digitalRead(pirPin);
  if (state == HIGH) {
    Serial.println("Somebody here!");
  } else {
    Serial.println("Monitoring...");
    delay(100);
  }
}
```

Wenn der Status HIGH ist, also eine Bewegung erkannt wird, erscheint die Meldung „Somebody here!“ im seriellen Monitor. Andernfalls wird „Monitoring…“ angezeigt.

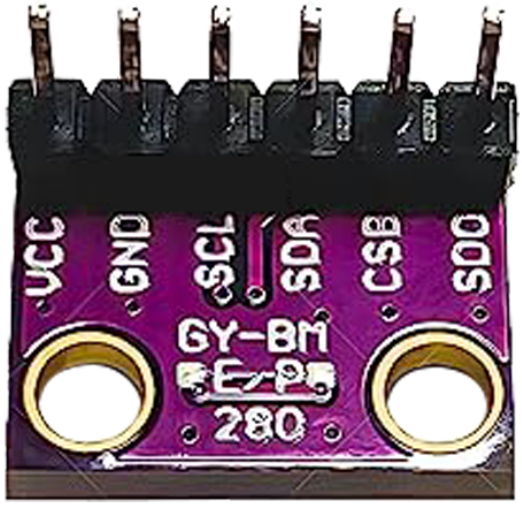
Weitere Ideen

- Einbindung einer LED, die bei erkannter Bewegung aufleuchtet oder blinkt. Dies bietet eine visuelle Rückmeldung zusätzlich zur Nachricht im seriellen Monitor.
- Integration eines Summers, der bei Bewegungserkennung einen Alarmton ausgibt.

Weitere Projekte

- *Bewegungsgesteuertes Relais*
- *Einbruchmeldeanlage mit Blynk*

2.4.15 Temperatur-, Feuchtigkeits- & Drucksensor (BMP280)



Einleitung

Das GY-BMP280-3.3 Präzisionsmodul für die atmosphärische Druckmessung ist ein Gerät, das in der Lage ist, Luftdruck und Temperatur mit hoher Genauigkeit zu erfassen. Es eignet sich hervorragend zur Überwachung von Wetterbedingungen oder für Projekte, die Daten zur Höhe oder zum barometrischen Druck nutzen.

Funktionsprinzip

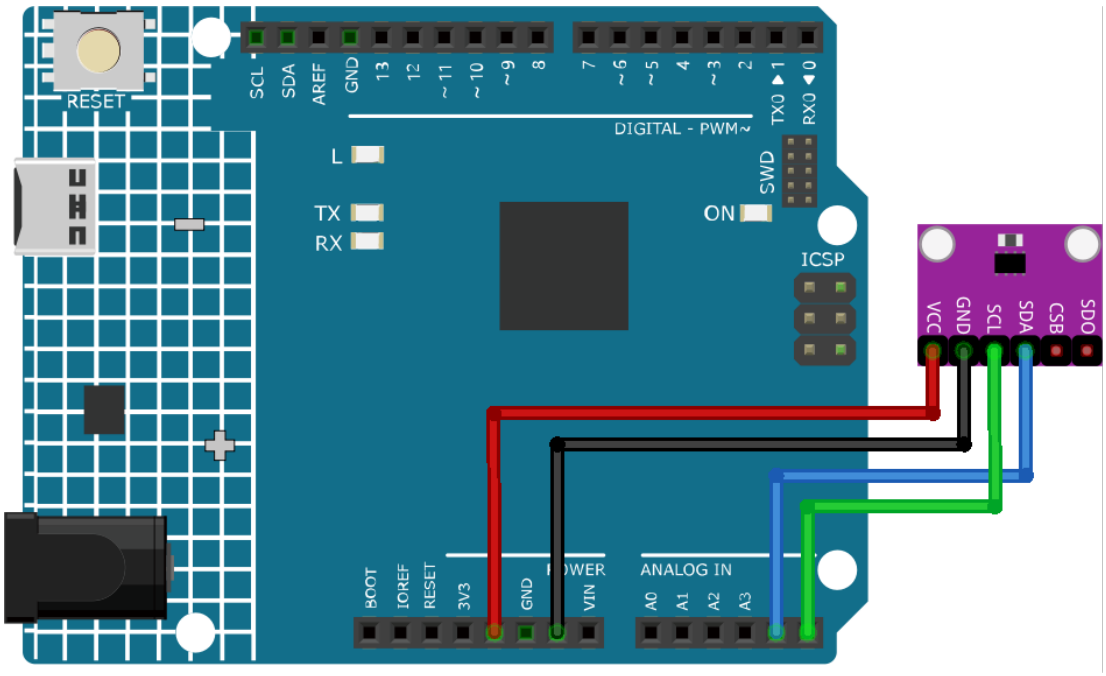
Das GY-BMP280-3.3 Modul arbeitet mit einem BMP280-Sensor von Bosch, der sowohl den Druck als auch die Temperatur messen kann. Innerhalb einer abgedichteten Metallkammer befinden sich ein piezoresistiver Drucksensor und ein Thermistor. Der piezoresistive Sensor ändert seinen Widerstand je nach dem auf die Kammer ausgeübten Druck, während der Thermistor seinen Widerstand in Abhängigkeit von der Innentemperatur der Kammer ändert. Das Modul beinhaltet einen integrierten Schaltkreis, der diese Widerstandswerte in digitale Signale umwandelt und diese über die I2C- oder SPI-Schnittstelle an den Arduino weiterleitet.

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Platine * 1
- Temperatur-, Feuchtigkeits- & Drucksensor (GY-BMP280-3.3) * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Bemerkung: Um die Bibliothek zu installieren, verwenden Sie den Arduino-Bibliotheksmanager und suchen Sie nach **„Adafruit BMP280“**, um es zu installieren.

Code-Erläuterung

1. Einbindung der Bibliotheken und Initialisierung. Die erforderlichen Bibliotheken werden eingebunden und der BMP280-Sensor wird für die Kommunikation über die I2C-Schnittstelle initialisiert.

   Bemerkung: Zur Installation der Bibliothek nutzen Sie den Arduino-Bibliotheksmanager und suchen nach **„Adafruit BMP280“** und installieren Sie es.

   - Adafruit BMP280 Bibliothek: Diese Bibliothek bietet eine benutzerfreundliche Schnittstelle für den BMP280-Sensor und ermöglicht es dem Benutzer, Temperatur, Druck und Höhe auszulesen.
   - Wire.h: Wird für die I2C-Kommunikation verwendet.

```
#include <Wire.h>
#include <Adafruit_BMP280.h>
#define BMP280_ADDRESS 0x76
Adafruit_BMP280 bmp;  // use I2C interface
```

2. Die Funktion `setup()` initialisiert die serielle Kommunikation, prüft den BMP280-Sensor und konfiguriert ihn mit den Standard-Einstellungen.

```
void setup() {
  Serial.begin(9600);
  while (!Serial) delay(100);
  Serial.println(F("BMP280 test"));
  unsigned status;
  status = bmp.begin(BMP280_ADDRESS);
  // ... (rest of the setup code)
```

3. Die Funktion `loop()` liest kontinuierlich Daten für Temperatur, Druck und Höhe vom BMP280-Sensor und gibt diese auf dem Serial Monitor aus.

```
void loop() {
  // ... (read and print temperature, pressure, and altitude data)
  delay(2000);  // 2-second delay between readings.
}
```

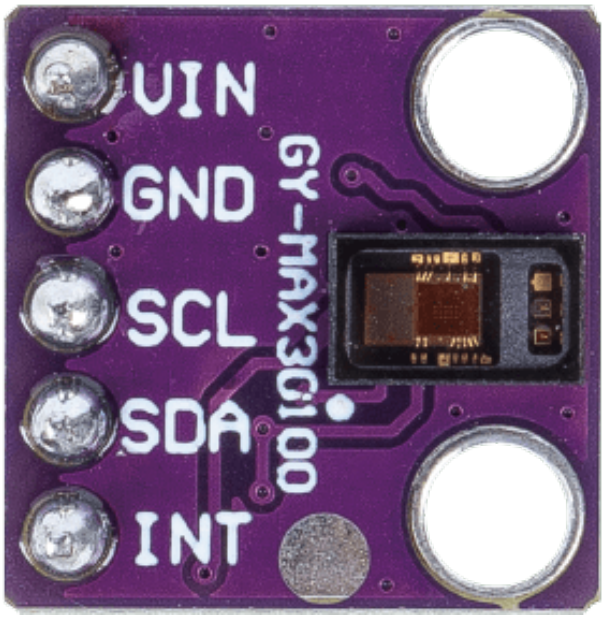
Weitere Ideen

- Integrieren Sie ein LCD-Displaymodul, um die Messwerte zusätzlich oder alternativ auf dem Serial Monitor anzuzeigen.
- Legen Sie Schwellenwerte für Temperatur und Druck fest. Verwenden Sie einen Summer oder eine LED, um bei Überschreitung dieser Grenzwerte zu alarmieren.

Weitere Projekte

- iot_Weather_monitor

2.4.16 Pulsoximeter und Herzfrequenzsensor (MAX30102)



Einleitung

Der MAX30102 ist ein Sensor, der ein Pulsoximeter und ein Herzfrequenzmessgerät kombiniert. Es handelt sich um einen optischen Sensor, der die Absorption von pulsierendem Blut durch einen Fotodetektor misst, nachdem er zwei Lichtwellenlängen von zwei LEDs ausgestrahlt hat - eine rote und eine infrarote. Diese spezielle LED-Farbkombination ist so konzipiert, dass Daten mit der Fingerspitze gelesen werden können.

Prinzip

Der MAX30102 funktioniert, indem er beide Lichter auf den Finger oder Ohrläppchen (oder im Grunde überall, wo die Haut nicht zu dick ist, sodass beide Lichter das Gewebe leicht durchdringen können) strahlt und die Menge des reflektierten Lichts mit einem Fotodetektor misst. Diese Methode der Pulserfassung durch Licht wird als Photoplethysmogramm bezeichnet.

Die Funktionsweise des MAX30102 kann in zwei Teile unterteilt werden: Herzfrequenzmessung und Pulsoximetrie (Messung des Sauerstoffgehalts des Blutes).

Herzfrequenzmessung

Das sauerstoffreiche Hämoglobin (HbO2) im arteriellen Blut hat die Eigenschaft, IR-Licht zu absorbieren. Je röter das Blut (je höher das Hämoglobin), desto mehr IR-Licht wird absorbiert. Da das Blut mit jedem Herzschlag durch den Finger gepumpt wird, ändert sich die Menge des reflektierten Lichts, wodurch sich eine veränderliche Wellenform am Ausgang des Fotodetektors ergibt. Wenn Sie weiterhin Licht ausstrahlen und Fotodetektor-Messungen durchführen, erhalten Sie schnell eine Herzfrequenz (HR) Messung.

Pulsoximetrie

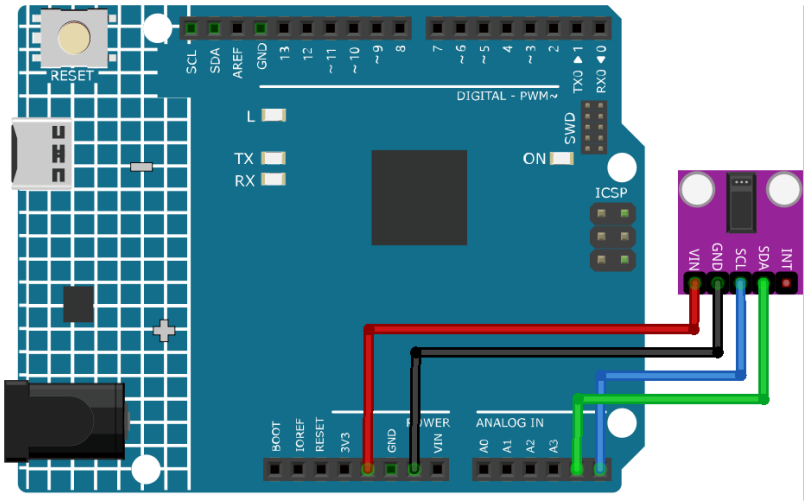
Die Pulsoximetrie basiert auf dem Prinzip, dass die Menge des absorbierten ROTEN und IR-Lichts je nach Menge des Sauerstoffs in Ihrem Blut variiert.

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Pulsoximeter und Herzfrequenzsensor(MAX30102) * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Warnung: Dieser Sketch erkennt die Herzfrequenz optisch. Diese Methode kann zu falschen Messwerten führen. Bitte **NICHT** für echte medizinische Diagnosen verwenden.

Bemerkung: Um die Bibliothek zu installieren, verwenden Sie den Arduino Library Manager und suchen nach **„SparkFun MAX3010x“** und installieren Sie diese.

Code-Erklärung

1. **Einbinden von Bibliotheken & Initialisierung globaler Variablen**:

   Die erforderlichen Bibliotheken werden importiert, das Sensorobjekt wird instanziert und globale Variablen für die Datenverwaltung werden festgelegt.

   Bemerkung: Um die Bibliothek zu installieren, verwenden Sie den Arduino Library Manager und suchen nach **„SparkFun MAX3010x“** und installieren Sie diese.

```
#include <Wire.h>
#include "MAX30105.h"
#include "heartRate.h"
MAX30105 particleSensor;
// ... (andere globale Variablen)
```

2. **Setup-Funktion & Sensorinitialisierung**:

   Die serielle Kommunikation wird mit einer Baudrate von 9600 initialisiert. Die Verbindung des Sensors wird überprüft und bei Erfolg wird eine Initialisierungssequenz ausgeführt. Bei Nichterkennung des Sensors wird eine Fehlermeldung angezeigt.

```
void setup() {
  Serial.begin(9600);
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) {
    Serial.println("MAX30102 not found.");
    while (1) ;  // Infinite loop if sensor not detected.
  }
  // ... (further setup)
```

3. **IR-Wert lesen & Herzschlag überprüfen**:

   Der IR-Wert, der auf den Blutfluss hinweist, wird vom Sensor abgerufen. Die Funktion `checkForBeat()` prüft anhand dieses Wertes, ob ein Herzschlag erkannt wird.

```
long irValue = particleSensor.getIR();
if (checkForBeat(irValue) == true) {
    // ... (heartbeat detected actions)
}
```

4. **Berechnung der Herzschläge pro Minute (BPM)**:

   Bei Erkennung eines Herzschlags wird der BPM-Wert anhand der Zeitdifferenz seit dem letzten erkannten Herzschlag berechnet. Der Code stellt sicher, dass der BPM-Wert in einem realistischen Bereich liegt, bevor der Durchschnitt aktualisiert wird.

```
long delta = millis() - lastBeat;
beatsPerMinute = 60 / (delta / 1000.0);
if (beatsPerMinute < 255 && beatsPerMinute > 20) {
    // ... (store and average BPM)
}
```

5. **Werte im Serial Monitor ausgeben**:

   Der IR-Wert, der aktuelle BPM-Wert und der durchschnittliche BPM-Wert werden im Serial Monitor angezeigt. Zusätzlich prüft der Code, ob der IR-Wert zu niedrig ist, was auf das Fehlen eines Fingers hindeuten könnte.

```
//Print the IR value, current BPM value, and average BPM value to the serial monitor
Serial.print("IR=");
Serial.print(irValue);
Serial.print(", BPM=");
Serial.print(beatsPerMinute);
Serial.print(", Avg BPM=");
Serial.print(beatAvg);

if (irValue < 50000)
  Serial.print(" No finger?");
```

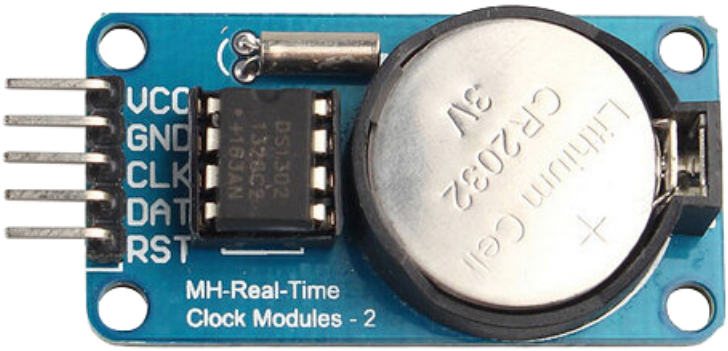
Weitere Ideen

- LEDs hinzufügen, die bei jedem erkannten Schlag aufleuchten
- Ein kleines OLED- oder LCD-Display verwenden, um Echtzeit-BPM-Werte und andere relevante Daten anzuzeigen.

Weitere Projekte

- fun_heartrate_monitor

2.4.17 Echtzeituhr-Modul (DS1302)



Einleitung

Das DS1302 Echtzeituhr-Modul ist ein Gerät zur präzisen Zeit- und Datumsverwaltung. Es eignet sich hervorragend für Projekte, die genaue Zeitangaben und Planungen erfordern, oder um eine digitale Uhr mit Arduino zu realisieren.

Funktionsweise

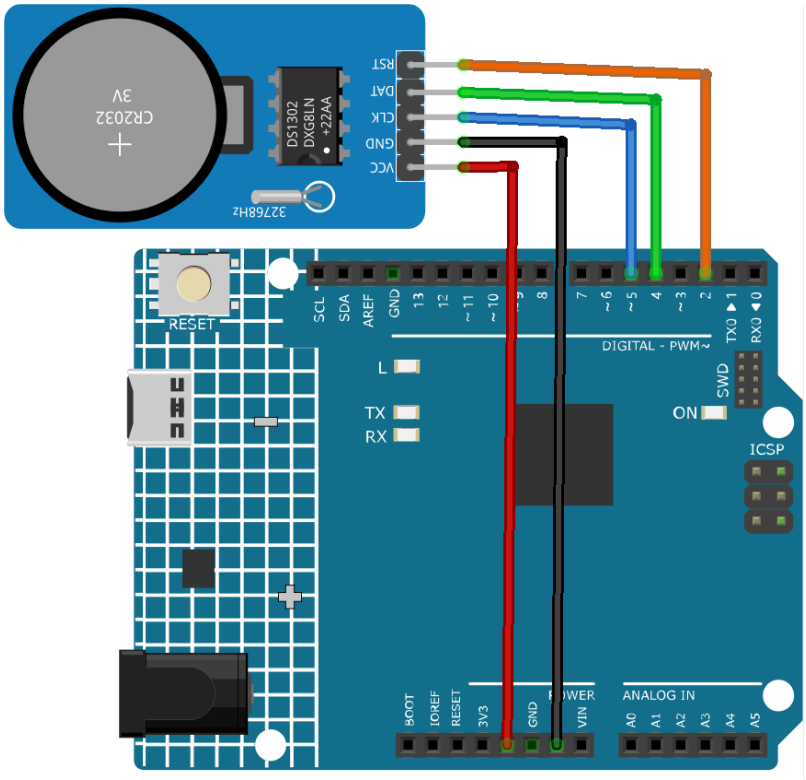
DS1302 ist ein von DALLAS in den USA eingeführter Uhrenbaustein mit Trickle-Ladung. Er verfügt über eine integrierte Echtzeituhr und einen 31-Byte-Statischen RAM. Der Baustein kommuniziert mit dem Mikrocontroller (MCU) über einfache serielle Schnittstellen. Er stellt Informationen zu Sekunde, Minute, Stunde, Tag, Woche, Monat und Jahr bereit. Zudem passt der DS1302 die Anzahl der Tage pro Monat sowie Schaltjahre automatisch an. Ob das 24-Stunden- oder das 12-Stunden-System verwendet wird, lässt sich über eine AM/PM-Auswahl festlegen. Die Kommunikation mit dem MCU erfolgt synchron über nur drei Anschlusskabel: Reset (RST), I/O-Daten (SDA) und serielle Uhr (SCL).

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Platine * 1
- Echtzeituhr-Modul (DS1302) * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Bemerkung: Um die Bibliothek zu installieren, nutzen Sie den Arduino-Bibliotheksmanager und suchen Sie nach **„Rtc by Makuna“**, um sie zu installieren.

Code-Erklärung

1. Initialisierung und Einbindung der Bibliotheken

   Bemerkung: Um die Bibliothek zu installieren, nutzen Sie den Arduino-Bibliotheksmanager und suchen Sie nach **„Rtc by Makuna“**, um sie zu installieren.

   Hier werden die erforderlichen Bibliotheken für das DS1302 RTC-Modul eingebunden.

```
#include <ThreeWire.h>
#include <RtcDS1302.h>
```

2. Pin-Definitionen und Erstellung der RTC-Instanz

   Die Pins für die Kommunikation werden definiert und eine Instanz des RTC wird erstellt.

```
const int IO = 4;     // DAT
const int SCLK = 5;   // CLK
const int CE = 2;     // RST

ThreeWire myWire(4, 5, 2);  // IO, SCLK, CE
RtcDS1302<ThreeWire> Rtc(myWire);
```

3. `setup()` Funktion

   Diese Funktion initialisiert die serielle Kommunikation und nimmt die Grundkonfiguration des RTC-Moduls vor. Es werden diverse Prüfungen durchgeführt, um sicherzustellen, dass die RTC korrekt arbeitet.

```
void setup() {
  Serial.begin(9600);

  Serial.print("compiled: ");
  Serial.print(__DATE__);
  Serial.println(__TIME__);

  Rtc.Begin();

  RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
  printDateTime(compiled);
  Serial.println();

  if (!Rtc.IsDateTimeValid()) {
    // Common Causes:
    //    1) first time you ran and the device wasn't running yet
    //    2) the battery on the device is low or even missing

    Serial.println("RTC lost confidence in the DateTime!");
    Rtc.SetDateTime(compiled);
  }

  if (Rtc.GetIsWriteProtected()) {
    Serial.println("RTC was write protected, enabling writing now");
    Rtc.SetIsWriteProtected(false);
  }

  if (!Rtc.GetIsRunning()) {
    Serial.println("RTC was not actively running, starting now");
    Rtc.SetIsRunning(true);
  }

  RtcDateTime now = Rtc.GetDateTime();
  if (now < compiled) {
    Serial.println("RTC is older than compile time!  (Updating DateTime)");
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    Rtc.SetDateTime(compiled);
  } else if (now > compiled) {
    Serial.println("RTC is newer than compile time. (this is expected)");
  } else if (now == compiled) {
    Serial.println("RTC is the same as compile time! (not expected but all is fine)
↪");
  }
}
```

4. `loop()` Funktion

   Diese Funktion liest regelmäßig das aktuelle Datum und die aktuelle Uhrzeit vom RTC aus und gibt sie im seriellen Monitor aus. Sie prüft auch, ob die RTC weiterhin eine gültige Zeit und ein gültiges Datum beibehält.

```
void loop() {
  RtcDateTime now = Rtc.GetDateTime();

  printDateTime(now);
  Serial.println();

  if (!now.IsValid()) {
    // Common Causes:
    //    1) the battery on the device is low or even missing and the power line␣
↪was disconnected
    Serial.println("RTC lost confidence in the DateTime!");
  }

  delay(5000);  // five seconds
}
```

5. Datum- und Zeitdruckfunktion

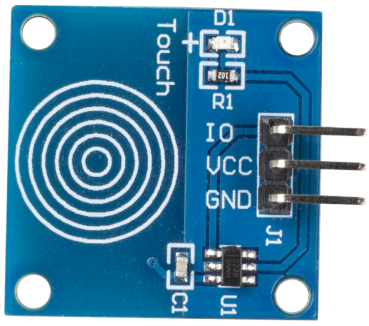
   Eine Hilfsfunktion, die ein `RtcDateTime`-Objekt nimmt und das formatierte Datum und die Uhrzeit im seriellen Monitor ausgibt.

```
void printDateTime(const RtcDateTime& dt) {
  char datestring[20];

  snprintf_P(datestring,
             countof(datestring),
             PSTR("%02u/%02u/%04u %02u:%02u:%02u"),
             dt.Month(),
             dt.Day(),
             dt.Year(),
             dt.Hour(),
             dt.Minute(),
             dt.Second());
  Serial.print(datestring);
}
```

Weitere Ideen

- Anzeige der Uhrzeit auf einem LCD oder im seriellen Monitor
- Planung von Ereignissen/Weckern zu bestimmten Zeiten

2.4.18 Berührungs-Sensormodul



Einführung

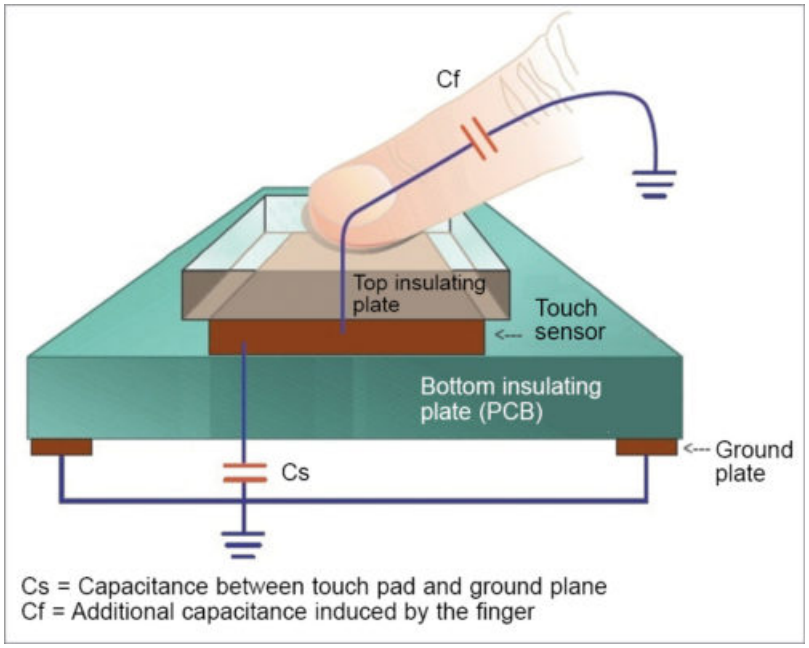
Der Berührungsschalter (auch Touch-Button oder Touch-Schalter genannt) findet breite Anwendung in verschiedenen Geräten, wie beispielsweise berührungsempfindlichen Lampen. Er übernimmt dieselben Funktionen wie ein herkömmlicher Knopf, verleiht dem Produkt jedoch ein sauberes, modernes Erscheinungsbild.

Funktionsprinzip

Das Modul basiert auf einem Berührungssensor-IC (TTP223B) und fungiert als kapazitiver Touch-Schalter. Im Ruhezustand gibt das Modul ein niedriges Signal bei geringem Stromverbrauch aus. Bei Berührung wechselt das Ausgangssignal auf ein hohes Niveau und kehrt nach Loslassen des Fingers wieder auf ein niedriges Niveau zurück.

Funktionsweise des kapazitiven Touch-Schalters:

Der kapazitive Touch-Schalter besteht aus verschiedenen Schichten – einer oberen Isolierschicht, gefolgt von der Berührungsplatte, einer weiteren Isolierschicht und schließlich einer Erdungsplatte.



In der Praxis kann ein kapazitiver Sensor auf einer doppelseitigen Leiterplatte realisiert werden, wobei eine Seite als Sensorfläche und die gegenüberliegende Seite als Erdungsplatte dient. Bei angelegter Spannung laden sich beide Platten auf. Im Gleichgewicht entspricht die Spannung der Platten der des Stromnetzes.

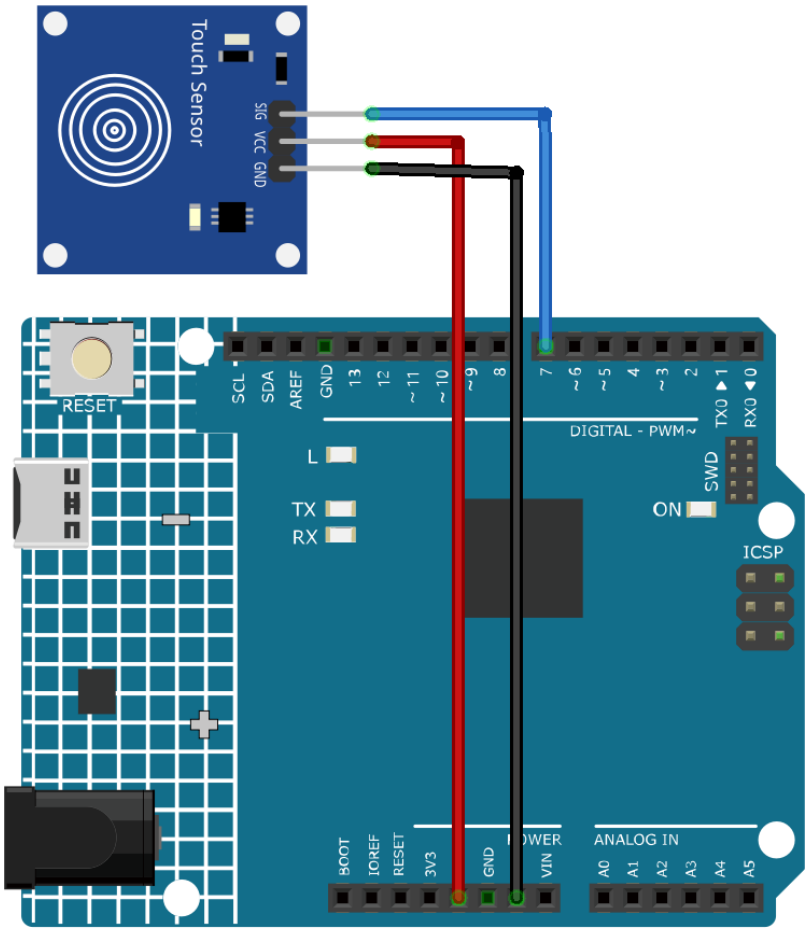
Der Detektorschaltkreis beinhaltet einen Oszillator, dessen Frequenz von der Kapazität der Sensorfläche abhängt. Nähert sich ein Finger der Sensorfläche, verändert die zusätzliche Kapazität die Frequenz des internen Oszillators. Der Detektorschaltkreis überwacht in festgelegten Intervallen die Oszillatorfrequenz und löst bei Überschreiten eines bestimmten Schwellenwerts ein Tastendruckereignis aus.

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Berührungs-Sensormodul * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erklärung

1. Definition der benötigten Variablen. Zunächst wird die Pinnummer definiert, an die der Berührungssensor angeschlossen ist.

```
const int sensorPin = 7;
```

2. Initialisierung in der `setup()`-Funktion. Hier legen wir fest, dass der Sensor-Pin für den Eingang und die integrierte LED für den Ausgang genutzt werden. Zudem wird die serielle Kommunikation gestartet, um Nachrichten an den seriellen Monitor zu senden.

```
void setup() {
  pinMode(sensorPin, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
}
```

3. Kontinuierlich überprüft das Arduino, ob der Berührungssensor aktiviert ist. Bei Berührung leuchtet die LED auf und eine Meldung „Touch detected!“ wird ausgegeben. Ist keine Berührung vorhanden, wird die LED ausgeschaltet und die Meldung „No touch detected…“ erscheint. Eine Verzögerung verhindert, dass der Sensor zu schnell abgefragt wird.

```
void loop() {
  if (digitalRead(sensorPin) == 1) {
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.println("Touch detected!");
  } else {
    digitalWrite(LED_BUILTIN, LOW);
    Serial.println("No touch detected...");
  }
  delay(100);
}
```

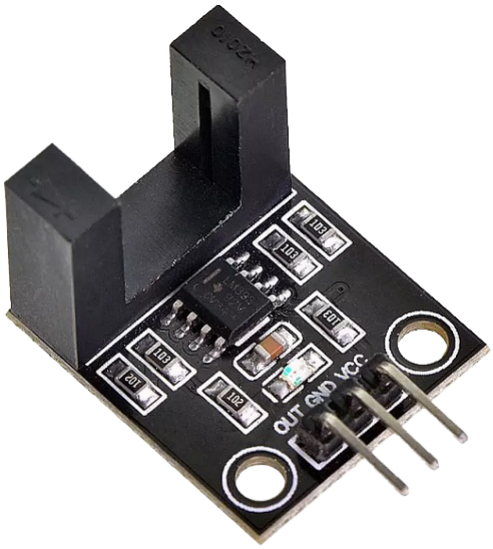
Weitere Ideen

- Verwendung unterschiedlicher LED-Farben zur Anzeige einer Berührung
- Der Berührungssensor könnte auch zur Steuerung komplexerer Elemente wie einem Motor oder einem Relais verwendet werden.

Weitere Projekte

- *Berührungsaktivierte Ampel*

2.4.19 Infrarot-Geschwindigkeitssensor-Modul (LM393)

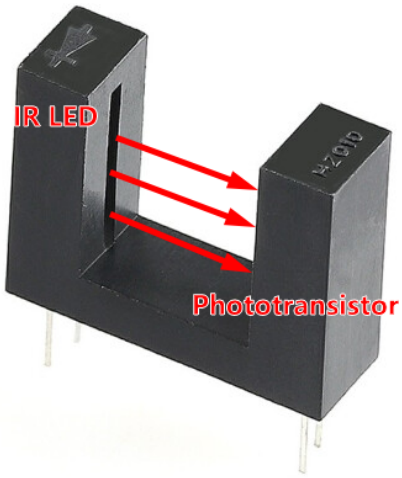


Einführung

Das LM393-Modul ist ein Infrarot-Zähler, der sowohl einen Infrarot-Sender als auch einen Empfänger enthält. Sobald ein Hindernis zwischen diesen Sensoren platziert wird, sendet das Modul ein Signal an den Mikrocontroller. Dieses Modul eignet sich in Kombination mit einem Mikrocontroller für Anwendungen wie Motordrehzahlerkennung, Impulszählung, Positionsbeschränkung und Ähnliches.

Funktionsprinzip

Das LM393-Modul enthält eine H2010-Fotodiode, die aus einem Fototransistor und einer Infrarot-Lichtquelle besteht und in einem 10 cm breiten schwarzen Kunststoffgehäuse verpackt ist.



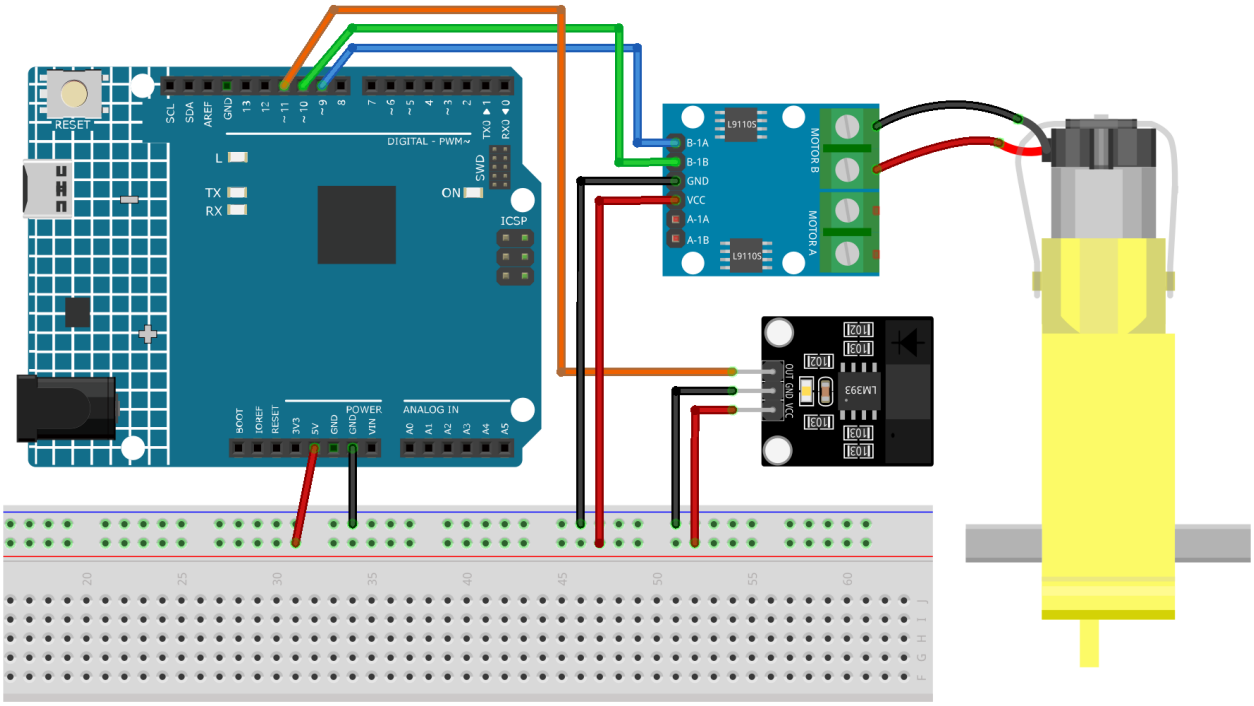
Im Betrieb emittiert die Infrarot-Lichtquelle kontinuierlich Infrarot-Licht (unsichtbares Licht), und die lichtempfindliche Triode wird leitfähig, wenn sie dieses Licht empfängt.

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Infrarot-Geschwindigkeitssensor-Modul * 1
- Verbindungskabel

Schaltungsaufbau



Programmcode

Code-Erklärung

1. Einrichtung der Pins und Initialisierung der Variablen. Hier definieren wir die Pins für den Motor und den Geschwindigkeitssensor. Darüber hinaus initialisieren wir die Variablen, die zur Messung und Berechnung der Motordrehzahl verwendet werden.

```
// Define the sensor and motor pins
const int sensorPin = 11;
const int motorB_1A = 9;
const int motorB_2A = 10;

// Define variables for measuring speed
unsigned long start_time = 0;
unsigned long end_time = 0;
int steps = 0;
float steps_old = 0;
float temp = 0;
float rps = 0;
```

2. Initialisierung in der `setup()`-Funktion. In diesem Abschnitt wird die serielle Kommunikation eingerichtet, die Modi der Pins konfiguriert und die Anfangsgeschwindigkeit des Motors festgelegt.

```
void setup() {
  Serial.begin(9600);
  pinMode(sensorPin, INPUT);
  pinMode(motorB_1A, OUTPUT);
  pinMode(motorB_2A, OUTPUT);
  analogWrite(motorB_1A, 160);
  analogWrite(motorB_2A, 0);
}
```

3. Messung der Motorgeschwindigkeit in der `loop()`-Funktion. In diesem Abschnitt werden die Schritte des Motors für eine Dauer von einer Sekunde gemessen. Anhand dieser Schritte wird die Umdrehungszahl pro Sekunde (rps) berechnet und an den seriellen Monitor gesendet.

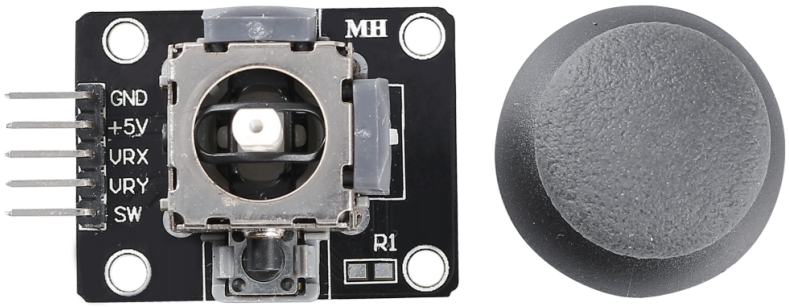
   `millis()` gibt die Anzahl der Millisekunden zurück, die seit dem Start des aktuellen Programms auf dem Arduino-Board vergangen sind.

```
void loop() {
  start_time = millis();
  end_time = start_time + 1000;
  while (millis() < end_time) {
    if (digitalRead(sensorPin)) {
      steps = steps + 1;
      while (digitalRead(sensorPin))
        ;
    }
  }
  temp = steps - steps_old;
  steps_old = steps;
  rps = (temp / 20);
  Serial.print("rps:");
  Serial.println(rps);
}
```

Zusätzliche Ideen

- Anzeige der rps auf einem LCD-Bildschirm für eine benutzerfreundlichere Bedienoberfläche.

2.4.20 Joystick-Modul



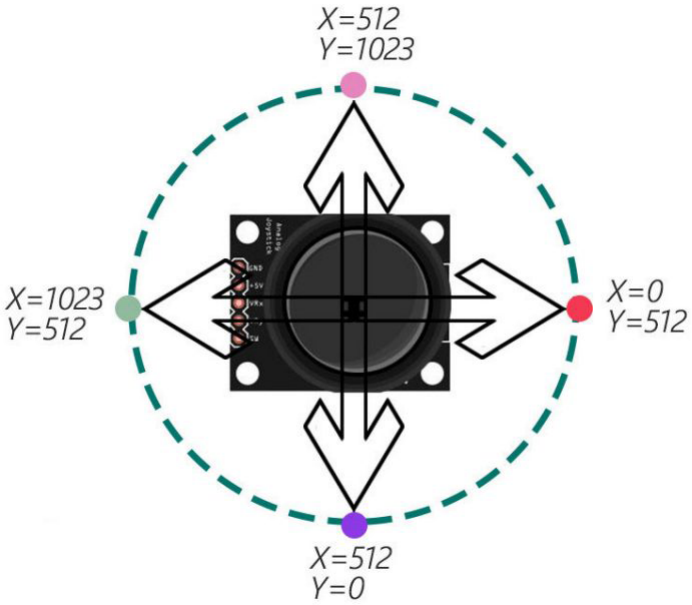
Einführung

Ein Joystick-Modul ist ein Gerät, das die Bewegungen eines Knüppels in zwei Richtungen messen kann: horizontal (X-Achse) und vertikal (Y-Achse). Es eignet sich zur Steuerung verschiedenster Anwendungen wie Spiele, Roboter, Kameras und mehr.

Funktionsprinzip

Ein Joystick funktioniert auf Basis der Widerstandsänderung zweier Potentiometer (üblicherweise 10 Kiloohm). Durch Veränderung des Widerstands in X- und Y-Richtung empfängt das Arduino Board variierende Spannungen, die als X- und Y-Koordinaten interpretiert werden. Der Prozessor benötigt eine ADC-Einheit, um die analogen Werte des Joysticks in digitale Werte umzuwandeln und die erforderliche Verarbeitung durchzuführen.

Arduino-Boards verfügen über sechs 10-Bit-ADC-Kanäle, was bedeutet, dass die Referenzspannung (5 Volt) des Arduino in 1024 Segmente unterteilt ist. Bewegt sich der Joystick entlang der X-Achse, steigt der ADC-Wert von 0 auf 1023, wobei der Mittelwert bei 512 liegt. Das untenstehende Bild zeigt den ungefähren ADC-Wert in Abhängigkeit von der Position des Joysticks.

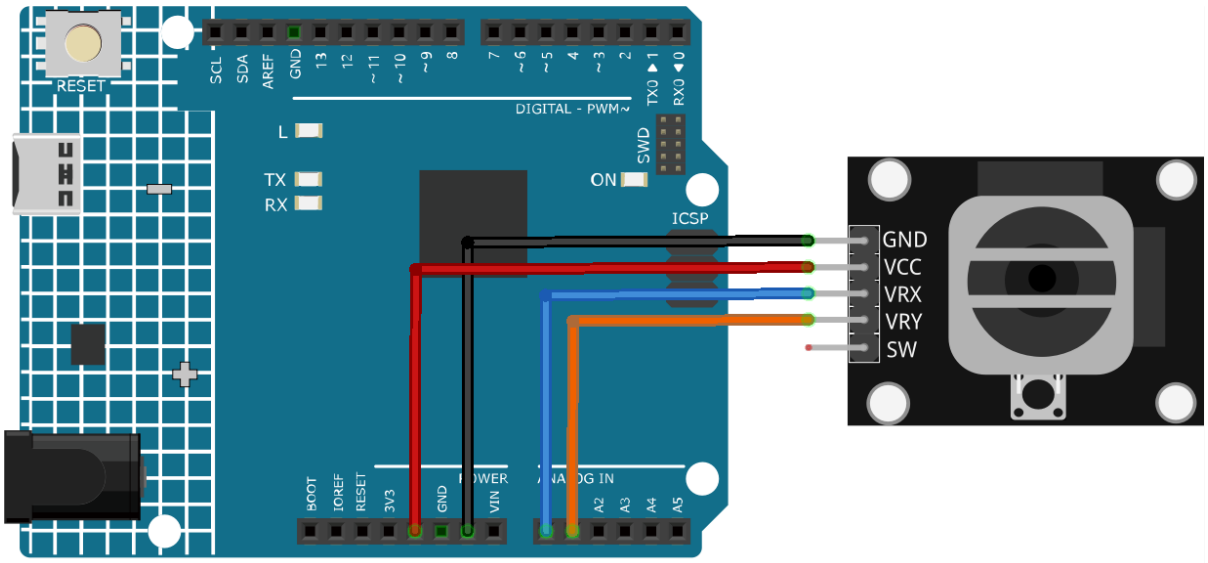


Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Joystick-Modul * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erklärung

1. Festlegung der Anschlüsse für den Joystick. Hier definieren wir, an welche analogen Pins die X- und Y-Achsen des Joysticks angeschlossen sind.

```
const int xPin = A0;
const int yPin = A1;
```

2. Initialisierung in der `setup()`-Funktion. In diesem Abschnitt wird die serielle Kommunikation eingerichtet, die es uns ermöglicht, Nachrichten von und zum Arduino über den seriellen Monitor zu senden und zu empfangen.

```
void setup() {
  Serial.begin(9600);
}
```

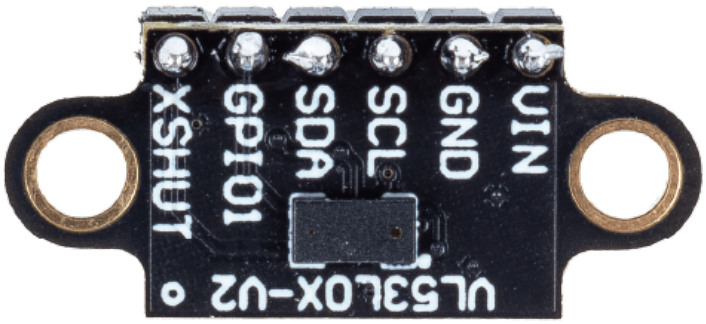
3. Auslesen der Joystick-Werte in der `loop()`-Funktion. Kontinuierlich liest das Arduino die X- und Y-Werte vom Joystick aus und gibt sie auf dem seriellen Monitor aus. Zwischen den einzelnen Ausgaben gibt es eine kurze Verzögerung, um die Lesungen besser lesbar zu machen und den seriellen Monitor nicht zu überlasten.

```
void loop() {
  Serial.print("X: ");
  Serial.print(analogRead(xPin));
  Serial.print(" | Y: ");
  Serial.println(analogRead(yPin));
  delay(50);
}
```

Weitere Ideen

- Die Joystick-Werte zur Steuerung eines Servomotors verwenden, der auf die Bewegungen des Joysticks reagiert.

2.4.21 Time-of-Flight Mikro-LIDAR-Entfernungssensor (VL53L0X)



Einführung

Der VL53L0X ist ein Time-of-Flight (ToF) Distanzmesssensor, der mithilfe von Lasertechnologie präzise Entfernungen bis zu 2 Metern messen kann. Das Modul ist ein Multisensor-System, das einen integrierten Laseremitter, Detektor und Mikrocontroller enthält. Es ist vollständig ausgestattet mit erforderlichen Komponenten wie Pull-up-Widerständen und Kondensatoren und bietet einen Messbereich von 50 - 1200 mm.

Funktionsprinzip

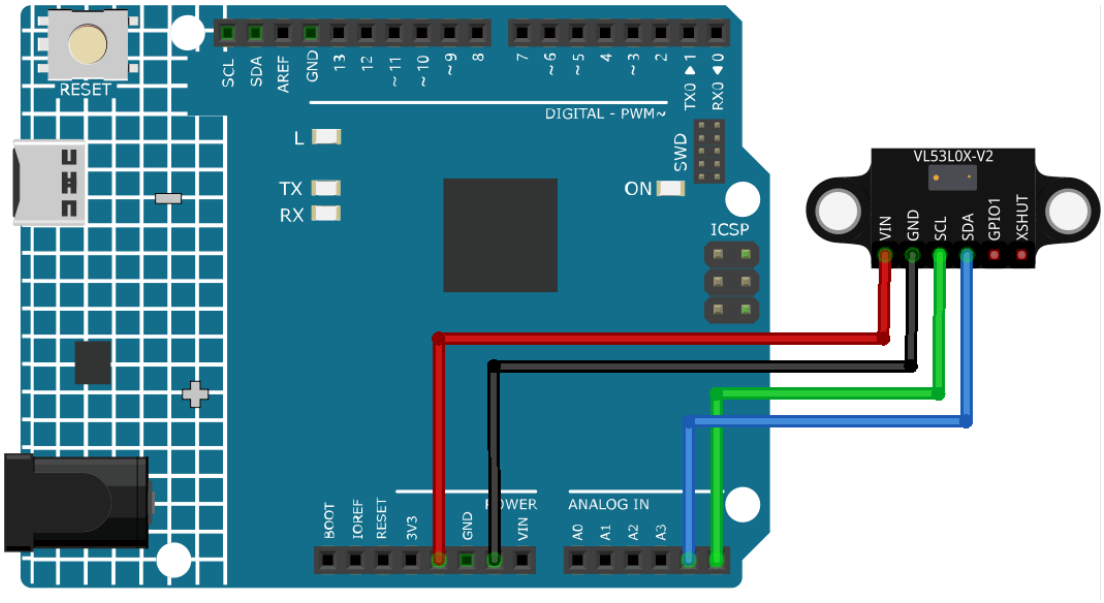
Das VL53L0X-Modul arbeitet auf dem Prinzip der Flugzeit (ToF). Es sendet einen Laserimpuls aus und misst die Zeit, die der Impuls benötigt, um zurückzukommen. Diese Zeit ist proportional zur Entfernung zwischen Sensor und Objekt. Das Modul nutzt ein SPAD-Array (Single-Photon-Avalanche-Diode) für die Detektion des reflektierten Lichts vom Objekt. Das SPAD-Array kann selbst einzelne Photonen detektieren. Ein integrierter Mikrocontroller verarbeitet die Daten des SPAD-Arrays und berechnet die Entfernung zum Objekt.

Anwendungsbeispiele

Hardwarekomponenten

- Arduino Uno R4 oder R3 Board * 1
- Time-of-Flight Mikro-LIDAR-Entfernungssensor * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Bemerkung: Für die Installation der Bibliothek verwenden Sie den Arduino-Bibliotheksmanager und suchen Sie nach **„Adafruit_VL53L0X“** und installieren Sie diese.

Code-Erläuterung

1. Einbindung der notwendigen Bibliothek und Initialisierung des Sensorobjekts. Wir beginnen mit der Einbindung der Bibliothek für den VL53L0X-Sensor und erstellen eine Instanz der Klasse Adafruit_VL53L0X.

   Bemerkung: Verwenden Sie den Arduino-Bibliotheksmanager und suchen Sie nach **„Adafruit_VL53L0X“** und installieren Sie diese.

```
#include <Adafruit_VL53L0X.h>
Adafruit_VL53L0X lox = Adafruit_VL53L0X();
```

2. Initialisierung in der Funktion `setup()`. Hier richten wir die serielle Kommunikation ein und initialisieren den Entfernungssensor. Kann der Sensor nicht initialisiert werden, hält das Programm an.

```
void setup() {
  Serial.begin(115200);
  while (!Serial) {
    delay(1);
  }
  Serial.println("Adafruit VL53L0X test");
  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while (1)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
      ;
  }
  Serial.println(F("VL53L0X API Simple Ranging example\n\n"));
}
```

3. Erfassung und Anzeige der Messwerte in der Funktion `loop()`. Kontinuierlich nimmt der Arduino eine Entfernungsmessung vor und zeigt diese, falls gültig, im seriellen Monitor an.

```
void loop() {
  VL53L0X_RangingMeasurementData_t measure;
  Serial.print("Reading a measurement... ");
  lox.rangingTest(&measure, false);
  if (measure.RangeStatus != 4) {
    Serial.print("Distance (mm): ");
    Serial.println(measure.RangeMilliMeter);
  } else {
    Serial.println(" out of range ");
  }
  delay(100);
}
```

Weitere Ideen

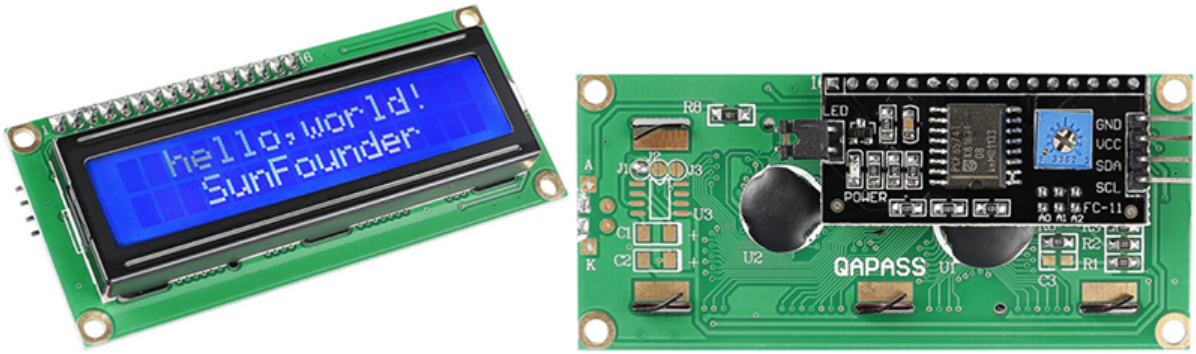
- Integration des Sensors in eine Anzeige (wie ein OLED), um die Entfernungsdaten anzuzeigen.
- Verwenden der Entfernungsdaten, um andere Komponenten wie LEDs oder Summer auszulösen, wenn ein Objekt in einen bestimmten Bereich kommt.

Weitere Projekte

- fun_tof_distance_monitor

Anzeigeelemente

2.4.22 I2C LCD 1602



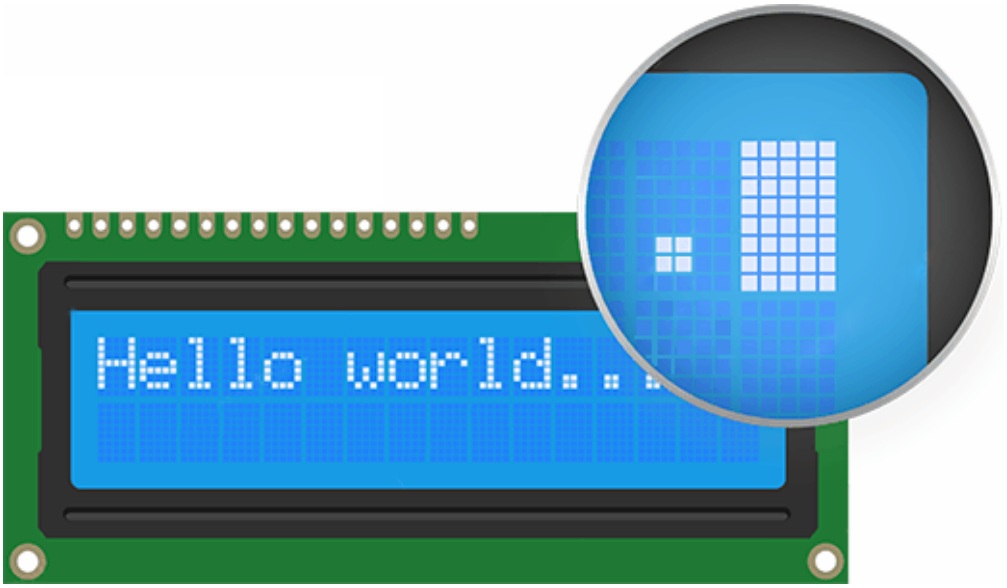
Einführung

Ein I2C LCD1602 ist ein Gerät, das Text und Zeichen auf einem 16x2 Flüssigkristall-Display (LCD) über das I2C-Protokoll darstellen kann. Es eignet sich hervorragend zur Anzeige von Informationen in Ihren Arduino-Projekten, wie etwa Sensorwerten, Meldungen oder Menüs. Das I2C-Modul enthält einen integrierten PCF8574 I2C-Chip, der serielle I2C-Daten in parallele Daten für das LCD umwandelt.

-

Funktionsprinzip

Das I2C LCD1602 setzt sich aus einem standardmäßigen LCD1602 und einem auf der Rückseite befestigten I2C-Modul zusammen. Dieses Modul ermöglicht die Erweiterung der I/O-Ports des Arduino über das I2C-Protokoll. Dabei kommen lediglich zwei Drähte zum Einsatz: SDA (Serial Data) und SCL (Serial Clock). Der I2C-Chip wandelt die Signale vom Arduino in Befehle für das LCD um. Jede der 16x2 Zellen des LCD kann Zeichen oder Symbole anzeigen. Diese Zellen bestehen aus je 5x8 Punkten, die durch Spannung ein- oder ausgeschaltet werden können. Verschiedene Kombinationen der Punkte erlauben die Darstellung unterschiedlicher Zeichen und Symbole.



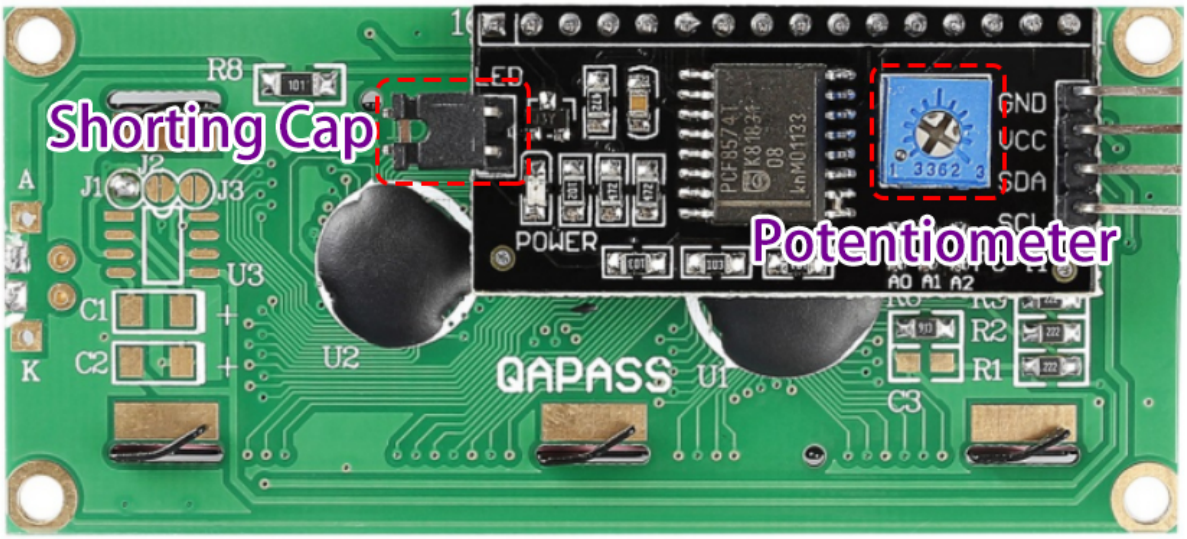
I2C-Adresse

Die Standardadresse ist normalerweise 0x27, in einigen Fällen jedoch auch 0x3F.

Wenn wir die Standardadresse von 0x27 als Beispiel nehmen, kann die Geräteadresse durch Kurzschließen der A0/A1/A2-Pads geändert werden. Im Ausgangszustand ist A0/A1/A2 auf 1 gesetzt und wenn das Pad kurzgeschlossen wird, ist A0/A1/A2 auf 0.

Hintergrundbeleuchtung/Kontrast

Die Hintergrundbeleuchtung kann mittels Jumper aktiviert werden; zum Deaktivieren wird dieser entfernt. Das blaue Potentiometer auf der Rückseite dient zur Kontrasteinstellung.



- **Shorting Cap**: Aktiviert die Hintergrundbeleuchtung; zum Deaktivieren entfernen.
- **Potentiometer**: Dient zur Kontrastanpassung (Klarheit der Textanzeige), im Uhrzeigersinn erhöht, gegen den Uhrzeigersinn verringert.

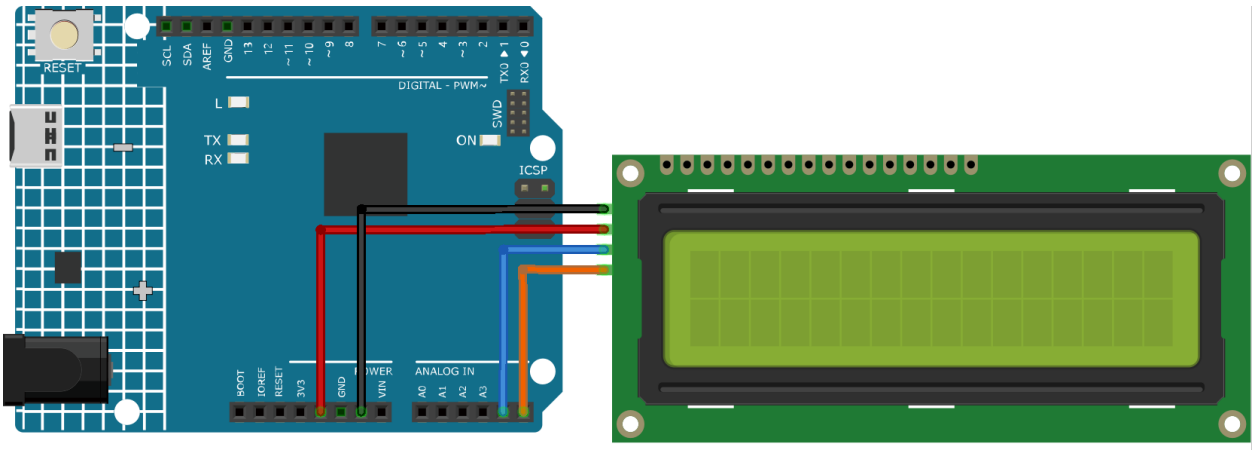
Bemerkung: Nach dem Verdrahten des LCD sollte man das Arduino einschalten und den Kontrast mittels Drehen des Potentiometers so einstellen, dass die erste Reihe von Rechtecken erscheint.

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Platine * 1
- I2C LCD1602 * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Bemerkung: Um die Bibliothek zu installieren, nutzen Sie den Arduino-Bibliotheksmanager und suchen nach **„LiquidCrystal I2C“**.

Code-Erklärung

1. Einbinden der Bibliothek und Initialisierung des LCD:

   Die LiquidCrystal I2C-Bibliothek wird eingebunden, um Funktionen und Methoden für die LCD-Ansteuerung bereitzustellen. Danach wird ein LCD-Objekt der Klasse LiquidCrystal_I2C erstellt, wobei die I2C-Adresse sowie die Anzahl der Zeilen und Spalten angegeben werden.

   ```
   #include <LiquidCrystal_I2C.h>
   LiquidCrystal_I2C lcd(0x27, 16, 2);
   ```

2. Setup-Funktion:

   Die `setup()`-Funktion wird einmalig bei Arduino-Start ausgeführt. Hier wird das LCD initialisiert, der Bildschirm geleert und die Hintergrundbeleuchtung aktiviert. Anschließend werden zwei Nachrichten auf dem LCD angezeigt.

```
void setup() {
  lcd.init();       // initialize the LCD
  lcd.clear();      // clear the LCD display
  lcd.backlight();  // Make sure backlight is on

  // Print a message on both lines of the LCD.
  lcd.setCursor(2, 0);  //Set cursor to character 2 on line 0
  lcd.print("Hello world!");

  lcd.setCursor(2, 1);  //Move cursor to character 2 on line 1
  lcd.print("LCD Tutorial");
}
```

Weitere Ideen

- Integration eines Temperatursensors zur Anzeige der aktuellen Raumtemperatur auf dem LCD.

Weitere Projekte

- *Potentiometer-Skalenwert*
- *Bluetooth LCD*

2.4.23 OLED-Display-Modul

Einführung

Ein OLED-Display-Modul (Organic Light-Emitting Diode) ermöglicht die Darstellung von Text, Grafiken und Bildern auf einem dünnen, flexiblen Bildschirm. Es nutzt organische Materialien, die Licht aussenden, wenn elektrischer Strom angelegt wird.

Ein wesentlicher Vorteil von OLED-Displays besteht darin, dass sie ihr eigenes Licht erzeugen und keine zusätzliche Hintergrundbeleuchtung benötigen. Dies führt oft zu besserem Kontrast, höherer Helligkeit und besseren Betrachtungswinkeln im Vergleich zu LCD-Displays.

Ein weiteres Highlight von OLED-Displays sind die tiefen Schwarzwerte. Da jedes Pixel sein eigenes Licht emittiert, kann zur Darstellung der Farbe Schwarz das entsprechende Pixel einfach ausgeschaltet werden.

OLED-Displays zeichnen sich zudem durch einen geringeren Stromverbrauch aus, da nur leuchtende Pixel Strom verbrauchen. Daher sind sie besonders beliebt in batteriebetriebenen Geräten wie Smartwatches, Gesundheitstrackern und anderen Wearables.

Funktionsprinzip

Ein OLED-Display-Modul besteht aus einem OLED-Panel und einem auf der Rückseite montierten OLED-Treiber-Chip. Das Panel besteht aus zahlreichen winzigen Pixeln, die unterschiedliche Farben erzeugen können. Jedes Pixel setzt sich aus mehreren Schichten organischer Materialien zwischen zwei Elektroden (Anode und Kathode) zusammen. Fließt Strom durch die Elektroden, senden die organischen Materialien Licht unterschiedlicher Wellenlängen aus, je nach ihrer Zusammensetzung.

Der OLED-Treiber-Chip steuert die Pixel des OLED-Panels über das serielle Kommunikationsprotokoll I2C (Inter-Integrated Circuit).

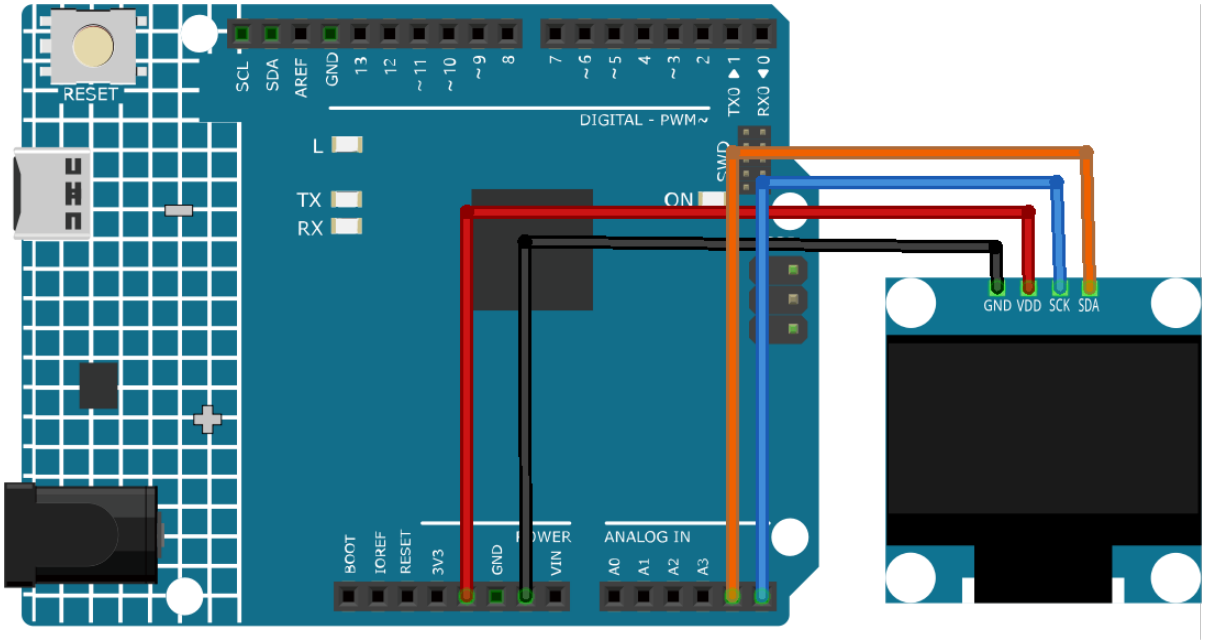
Mit speziellen Bibliotheken wie der Adafruit SSD1306 lässt sich das OLED-Display-Modul über den Arduino ansteuern, um beispielsweise Helligkeitsstufen einzustellen oder Text und Grafiken darzustellen.

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- OLED-Display-Modul * 1
- Jumperkabel

Schaltungsaufbau



Code

Bemerkung: Zur Installation der Bibliothek verwenden Sie den Arduino-Bibliotheksmanager und suchen nach **„Adafruit SSD1306“** sowie **„Adafruit GFX“** und installieren diese.

Code-Erläuterung

1. **Einbindung der Bibliotheken und initiale Definitionen**: Die notwendigen Bibliotheken für die Kommunikation mit dem OLED-Display werden eingebunden. Anschließend erfolgen die Definitionen für die Abmessungen und die I2C-Adresse des OLEDs.
    - **Adafruit SSD1306**: Diese Bibliothek ist darauf ausgelegt, die Kommunikation mit dem SSD1306 OLED-Display zu erleichtern. Sie stellt Methoden zur Initialisierung des Displays, zur Steuerung seiner Einstellungen und zur Anzeige von Inhalten bereit.
    - **Adafruit GFX Library**: Dies ist eine Kerngrafikbibliothek, die es ermöglicht, Text anzuzeigen, Farben zu erzeugen, Formen zu zeichnen usw., auf verschiedenen Bildschirmen, einschließlich OLEDs.

Hinweis: Um die Bibliothek zu installieren, verwenden Sie den Arduino-Bibliotheksmanager und suchen nach **„Adafruit SSD1306“** und **„Adafruit GFX“**, um sie zu installieren.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128  // OLED display width, in pixels
#define SCREEN_HEIGHT 64  // OLED display height, in pixels
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C
```

2. **Bitmap-Daten**: Bitmap-Daten für die Anzeige eines benutzerdefinierten Symbols auf dem OLED-Bildschirm. Diese Daten repräsentieren ein Bild in einem Format, das das OLED interpretieren kann.

   Sie können dieses Online-Tool namens verwenden, um Ihr Bild in ein Array umzuwandeln.

   Das Schlüsselwort `PROGMEM` gibt an, dass das Array im Programmspeicher des Arduino-Mikrocontrollers gespeichert ist. Die Speicherung von Daten im Programmspeicher (PROGMEM) anstelle von RAM kann bei großen Datenmengen nützlich sein, da sie sonst zu viel Platz im RAM beanspruchen würden.

```
static const unsigned char PROGMEM sunfounderIcon[] = {...};
```

3. **Setup-Funktion (Initialisierung und Anzeige)**: Die `setup()`-Funktion initialisiert das OLED und zeigt eine Reihe von Mustern, Texten und Animationen an.

```
void setup() {
   ...  // Serial initialization and OLED object initialization
   ...  // Displaying various text, numbers, and animations
}
```

Weitere Ideen

- Verwenden Sie Tasten, um die angezeigten Nachrichten zu ändern oder zwischen verschiedenen Mustern und Animationen umzuschalten.
- Zeigen Sie Sensormessungen (wie Temperatur oder Luftfeuchtigkeit) in Echtzeit auf dem OLED an.

Weitere Projekte

- fun_heartrate_monitor
- fun_tof_distance_monitor
- *Bluetooth OLED*

2.4.24 Ampel-Modul

Einführung

Das Ampel-Modul ist ein kleines Gerät, das rote, gelbe und grüne Lichter anzeigen kann, genau wie eine echte Ampel. Es eignet sich zur Erstellung eines Modells eines Ampelsystems oder zum Erlernen der Steuerung von LEDs mit Arduino. Es zeichnet sich durch seine kompakte Bauform, einfache Verkabelung, zielgerichtete Anwendung und individuelle Installationsmöglichkeiten aus. Über den PWM-Pin lässt sich die Helligkeit der LED steuern.

Funktionsprinzip

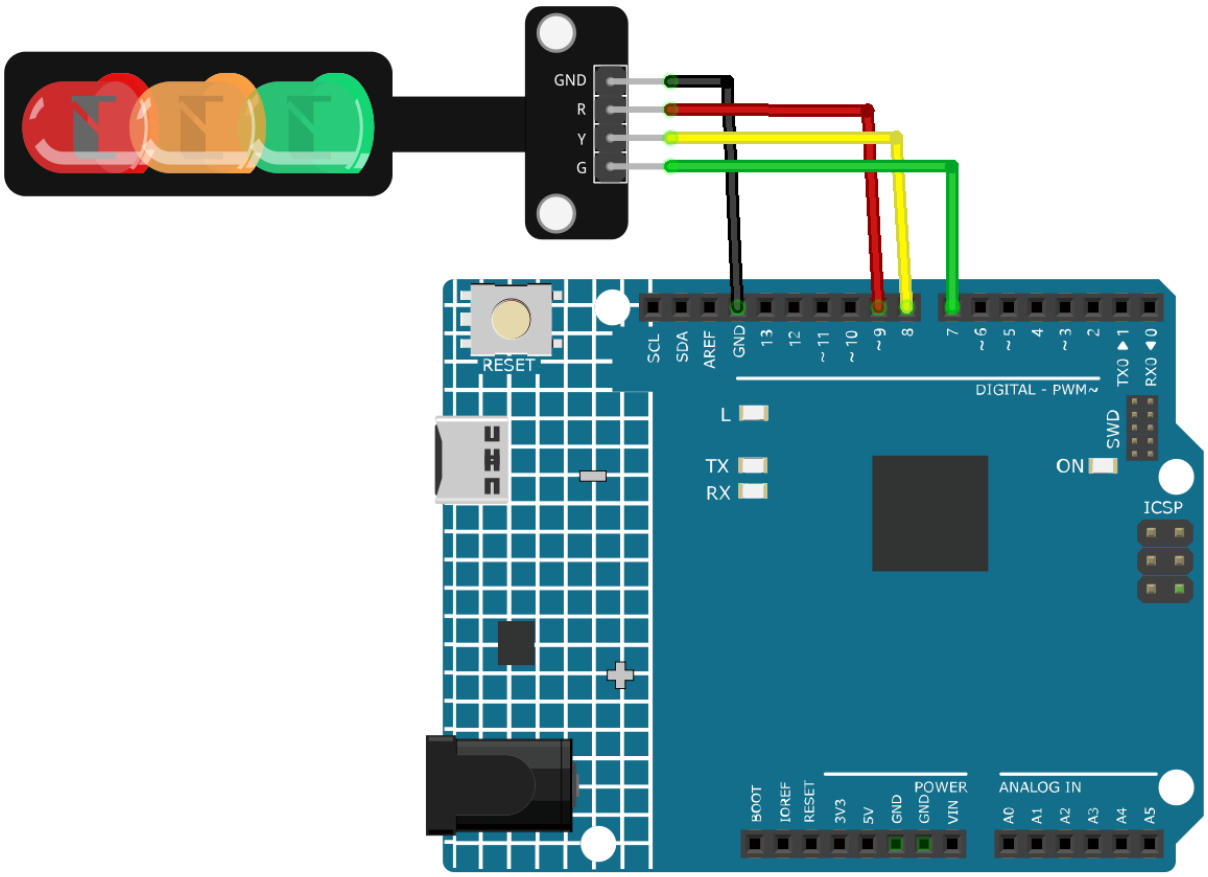
Das Ampel-Modul kann auf zwei Hauptarten gesteuert werden. Die einfachere Methode verwendet digitale Eingänge vom Arduino, bei denen ein HIGH- oder LOW-Signal die entsprechende LED direkt ein- oder ausschaltet. Alternativ kann PWM (Pulsweitenmodulation) verwendet werden, insbesondere wenn die Helligkeit der LED variiert werden soll. PWM ist eine Technik, bei der der Tastgrad eines digitalen Signals verändert wird, um die Helligkeit der LED zu modulieren. Der Tastgrad gibt den prozentualen Anteil der Zeit an, während der ein Signal in einem bestimmten Zeitraum aktiv ist. Ein Tastgrad von 50% bedeutet beispielsweise, dass das Signal für die Hälfte der Zeit aktiv und für den Rest inaktiv ist.

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Ampel-Modul * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erklärung

1. Bevor wir mit den Operationen beginnen, definieren wir Konstanten für die Pins, an denen die LEDs angeschlossen sind. Dies macht unseren Code leichter lesbar und modifizierbar.

```
const int rledPin = 9;  //rot
const int yledPin = 8;  //gelb
const int gledPin = 7;  //grün
```

2. Hier legen wir die Pin-Modi für unsere LED-Pins fest. Alle sind auf `OUTPUT` gesetzt, da wir vorhaben, Spannung an sie zu senden.

```
void setup() {
  pinMode(rledPin, OUTPUT);
  pinMode(yledPin, OUTPUT);
  pinMode(gledPin, OUTPUT);
}
```

3. Hier wird die Logik für unseren Ampelzyklus implementiert. Die Abfolge der Operationen lautet:
   - Die grüne LED für 5 Sekunden einschalten.
   - Die gelbe LED dreimal blinken lassen (jedes Blinken dauert 0,5 Sekunden).

- Die rote LED für 5 Sekunden einschalten.

```
void loop() {
  digitalWrite(gledPin, HIGH);
  delay(5000);
  digitalWrite(gledPin, LOW);

  digitalWrite(yledPin, HIGH);
  delay(500);
  digitalWrite(yledPin, LOW);
  delay(500);
  digitalWrite(yledPin, HIGH);
  delay(500);
  digitalWrite(yledPin, LOW);
  delay(500);
  digitalWrite(yledPin, HIGH);
  delay(500);
  digitalWrite(yledPin, LOW);
  delay(500);

  digitalWrite(rledPin, HIGH);
  delay(5000);
  digitalWrite(rledPin, LOW);
}
```

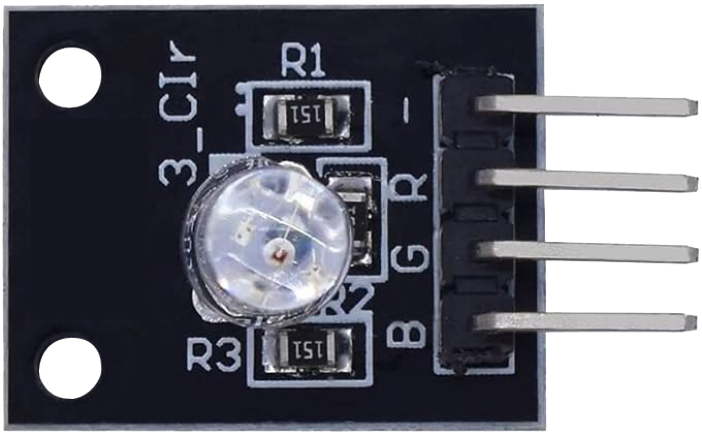
Weitere Ideen

- Integrieren Sie einen Summer, um akustische Signale beim Wechsel von Grün auf Rot zu geben, was für sehbehinderte Personen hilfreich ist.

Weitere Projekte

- *Berührungsaktivierte Ampel*
- *Ferngesteuerter Relais-Controller mit Blynk*
- iot_Bluetooth_voice_control_relay
- *Bluetooth-Ampel*

2.4.25 RGB-Modul



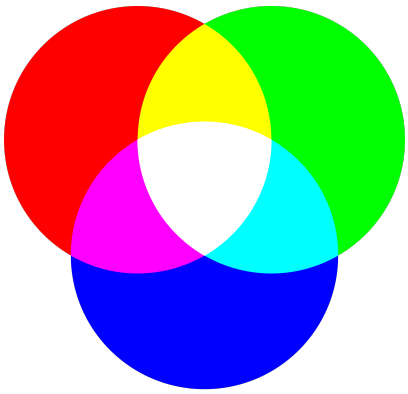
Einführung

Das RGB-Vollfarb-LED-Modul erzeugt durch die Mischung von rotem, grünem und blauem Licht ein Spektrum an Farben. Jede Farbe wird mittels PWM (Pulsweitenmodulation) eingestellt. Das Modul kann für bunte Lichteffekte verwendet oder zum Erlernen der PWM-Steuerung mit Arduino genutzt werden.

Funktionsprinzip

Das RGB-Modul funktioniert mit einer Vollfarb-LED, die über R-, G- und B-Anschlüsse mit einstellbarer PWM-Spannungseingabe verfügt. Die Farben der LED können miteinander kombiniert werden. Zum Beispiel erzeugt die Mischung von blauem und grünem Licht Zyan, die Mischung von rotem und grünem Licht Gelb. Dies wird als „additive Farbmischung“ bezeichnet.

- Additive Farbe - Wikipedia



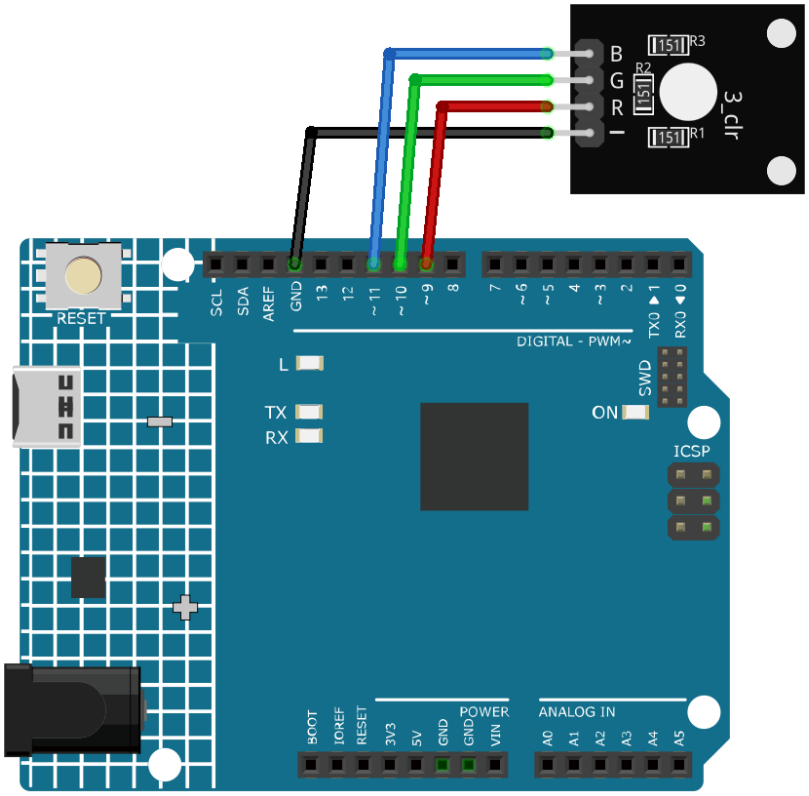
Durch diese Methode können wir die drei Grundfarben in verschiedenen Verhältnissen mischen, um sichtbares Licht jeder gewünschten Farbe zu erzeugen. Beispielsweise kann Orange durch mehr Rot und weniger Grün erzeugt werden. Durch Anpassung der Stärke der Grundfarben (Rot, Blau, Grün) mittels PWM kann eine vollständige Farbmischung erreicht werden.

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- RGB-Modul * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erklärung

1. Im ersten Codeabschnitt werden die Pins deklariert und initialisiert, an die die Farbkanäle der RGB-LED angeschlossen sind.

```
const int rledPin = 9;  // pin connected to the red color channel
const int gledPin = 10;   // pin connected to the green color channel
const int bledPin = 11;  // pin connected to the blue color channel
```

2. Die Funktion `setup()` initialisiert diese Pins als OUTPUT, da wir Signale von diesen Pins zum RGB-LED-Modul senden.

```
void setup() {
  pinMode(rledPin, OUTPUT);
  pinMode(gledPin, OUTPUT);
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  pinMode(bledPin, OUTPUT);
}
```

3. In der Funktion `loop()` wird die Funktion `setColor()` mit verschiedenen Parametern aufgerufen, um verschiedene Farben anzuzeigen. Die Funktion `delay()` wird nach dem Einstellen jeder Farbe verwendet, um für 1000 Millisekunden (oder 1 Sekunde) zu pausieren, bevor zur nächsten Farbe übergegangen wird.

```
void loop() {
  setColor(255, 0, 0);  // Set RGB LED color to red
  delay(1000);
  setColor(0, 255, 0);  // Set RGB LED color to green
  delay(1000);
  // The rest of the color sequence...
}
```

4. Die Funktion `setColor()` verwendet die Funktion `analogWrite()`, um die Helligkeit der einzelnen Farbkanäle im RGB-LED-Modul zu steuern. Mit der Funktion `analogWrite()` und PWM können verschiedene Spannungsausgänge simuliert werden. Durch Steuerung des PWM-Tastverhältnisses kann die Helligkeit jedes Farbkanals kontrolliert und so die Mischung verschiedener Farben ermöglicht werden.

```
void setColor(int R, int G, int B) {
  analogWrite(rledPin, R);  // Use PWM to control the brightness of the red color␣
↪channel
  analogWrite(gledPin, G);  // Use PWM to control the brightness of the green color␣
↪channel
  analogWrite(bledPin, B);  // Use PWM to control the brightness of the blue color␣
↪channel
}
```

Weitere Ideen

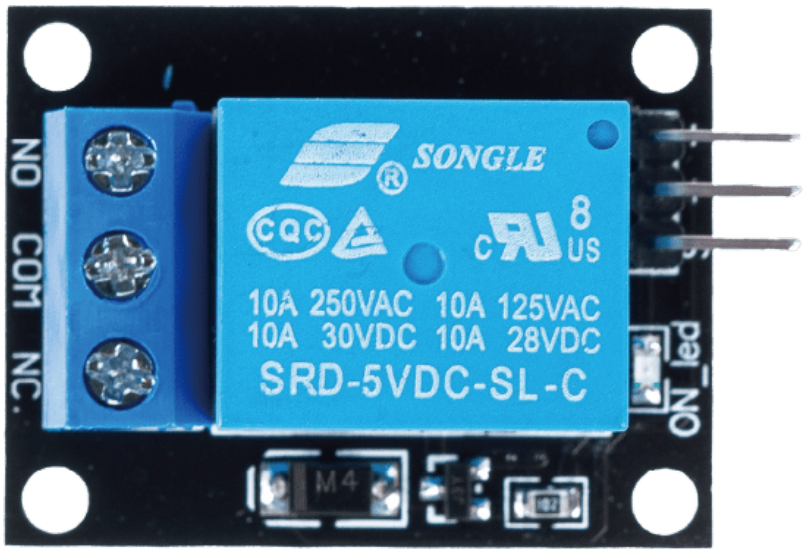
- Versuchen Sie, andere Farben anzuzeigen.
- Integrieren Sie die RGB-LED mit Sensoren und zeigen Sie verschiedene Farben basierend auf dem Sensorwert an.

Weitere Projekte

- *Gasleck-Alarm*
- *Lichtsteuerungsschalter*
- *Bewegungsgesteuertes Relais*
- *Bluetooth RGB-Controller*
- *Bluetooth-Fernrelais*

Aktuatoren

2.4.26 5V-Relaismodul



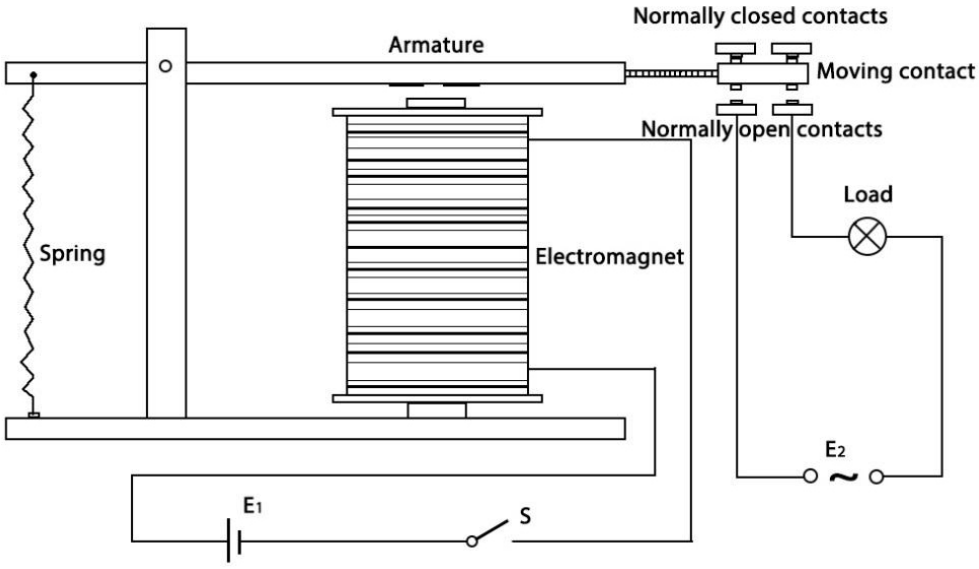
Einleitung

5V-Relaismodule sind Geräte, die Hochspannungs- oder Hochstromgeräte mit einem 5V-Signal vom Arduino ein- und ausschalten können. Sie eignen sich zur Steuerung von Geräten wie Lampen, Lüftern, Motoren oder Magnetventilen. Ein 5V-Relais verfügt über drei Hochspannungsanschlüsse (NC, C und NO), die an das zu steuernde Gerät angeschlossen werden. Auf der anderen Seite befinden sich drei Niederspannungsanschlüsse (Ground, Vcc und Signal), die mit dem Arduino verbunden werden.

Funktionsprinzip

Ein Relais ist ein Bauelement, das zur Herstellung einer Verbindung zwischen zwei oder mehr Punkten oder Geräten in Reaktion auf ein eingegebenes Signal verwendet wird. Mit anderen Worten, Relais bieten eine Isolation zwischen dem Controller und den angeschlossenen Geräten, die entweder mit Wechsel- oder Gleichstrom arbeiten können. Da sie Signale von einem Mikrocontroller erhalten, der mit Gleichstrom arbeitet, ist ein Relais erforderlich, um die Lücke zu überbrücken. Relais sind besonders nützlich, wenn man mit einem kleinen elektrischen Signal eine große Menge an Strom oder Spannung steuern möchte.

Ein Relais besteht aus fünf Hauptkomponenten:



Elektromagnet - Er besteht aus einem Eisenkern, um den eine Drahtspule gewickelt ist. Wird Strom durch ihn geleitet, wird er magnetisch.

Anker - Der bewegliche magnetische Streifen wird als Anker bezeichnet. Wenn Strom durch die Spule fließt, wird ein Magnetfeld erzeugt, das dazu dient, die Kontakte entweder zu schließen oder zu öffnen. Der Anker kann sowohl mit Gleich- als auch mit Wechselstrom bewegt werden.

Feder - Wenn kein Strom durch die Spule des Elektromagneten fließt, zieht die Feder den Anker zurück, sodass der Stromkreis unterbrochen wird.

Elektrische Kontakte - Es gibt zwei Kontaktstellen:

- Normalerweise offen - geschlossen, wenn das Relais aktiviert ist, und offen, wenn es inaktiv ist.
- Normalerweise geschlossen - offen, wenn das Relais aktiviert ist, und geschlossen, wenn es inaktiv ist.

Gehäuse - Dies ist typischerweise aus Kunststoff gefertigt und bietet strukturelle Unterstützung und Schutz für das Relais.

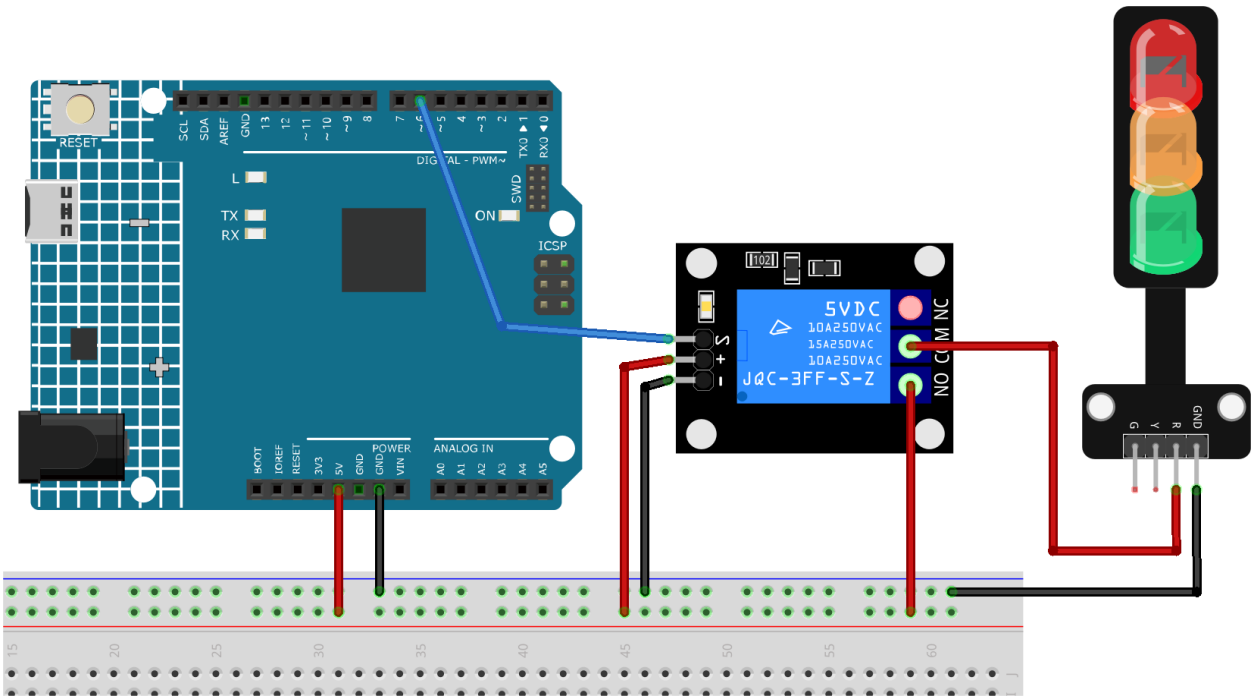
Die Funktionsweise eines Relais ist einfach. Sobald Strom zugeführt wird, beginnt der Strom durch die Steuerspule zu fließen, und der Elektromagnet wird aktiviert. Dann wird der Anker zum Elektromagneten gezogen, und die Kontakte schließen sich. Dadurch wird der Lastkreis eingeschaltet. Um den Kreislauf zu unterbrechen, wird der Anker durch die Kraft der Feder zurückgezogen. Auf diese Weise kann das Ein- und Ausschalten des Relais den Zustand einer Last steuern.

Anwendungsbeispiele

Hardwarekomponenten

- Arduino Uno R4 oder R3 Board * 1
- 5V-Relaismodul * 1
- Jumperkabel

Schaltungsaufbau



Warnung: Das folgende Beispiel zeigt die Verwendung eines Relais zur Steuerung eines LED-Moduls. **Obwohl das Relais in realen Anwendungen auch für andere Geräte verwendet werden kann, ist bei der Arbeit mit HOHER Wechselspannung äußerste Vorsicht geboten. Unsachgemäßer oder fehlerhafter Gebrauch kann zu schweren Verletzungen oder sogar zum Tod führen. Daher ist es für Personen gedacht, die sich mit HOHER Wechselspannung auskennen. Sicherheit hat immer Vorrang.**

Programmcode

Code-Erklärung

1. Einrichten des Relais-Pins:
   - Das Relaismodul ist an Pin 6 des Arduino angeschlossen. Dieser Pin wird im Code als `relayPin` bezeichnet.

```
const int relayPin = 6;
```

2. Konfigurieren des Relais-Pins als Ausgang:
   - In der `setup()` Funktion wird der Relais-Pin als OUTPUT festgelegt, um Signale (entweder HIGH oder LOW) an diesen Pin zu senden.

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

3. Ein- und Ausschalten des Relais:
   - In der `loop()` Funktion wird das Relais zunächst auf den OFF-Zustand gesetzt (`digitalWrite(relayPin, LOW)`) und bleibt 3 Sekunden in diesem Zustand (`delay(3000)`).

- Anschließend wird das Relais auf den ON-Zustand gesetzt (`digitalWrite(relayPin, HIGH)`) und bleibt ebenfalls 3 Sekunden in diesem Zustand.
- Dieser Zyklus wiederholt sich unendlich.

```
void loop() {
  digitalWrite(relayPin, LOW);
  delay(3000);

  digitalWrite(relayPin, HIGH);
  delay(3000);
}
```

Weitere Ideen

- Einführung eines physischen Schalters zur manuellen Steuerung des Relaiszustands.
- Integration von Sensoren (wie Temperatur- oder Lichtsensor) zur Auslösung des Relais abhängig von den Umgebungsbedingungen.

Weitere Projekte

- *Lichtsteuerungsschalter*
- *Bewegungsgesteuertes Relais*
- *Ferngesteuerter Relais-Controller mit Blynk*
- *Bluetooth-Fernrelais*
- iot_Bluetooth_voice_control_relay

2.4.27 Passives Summermodul

Einleitung

Ein passiver Summer ist ein Bauelement, das Töne erzeugt, sobald ihm ein elektrisches Signal zugeführt wird. Es wird als passiv bezeichnet, da es keinen internen Oszillator besitzt, um eigenständig Töne zu generieren. Stattdessen benötigt es ein externes Signal von einem Mikrocontroller wie dem Arduino. Das passive Summermodul ist eine kompakte elektronische Komponente, die neben dem passiven Summer auch die nötige Schaltungstechnik enthält, um die Anwendung mit dem Arduino zu vereinfachen.

Funktionsprinzip

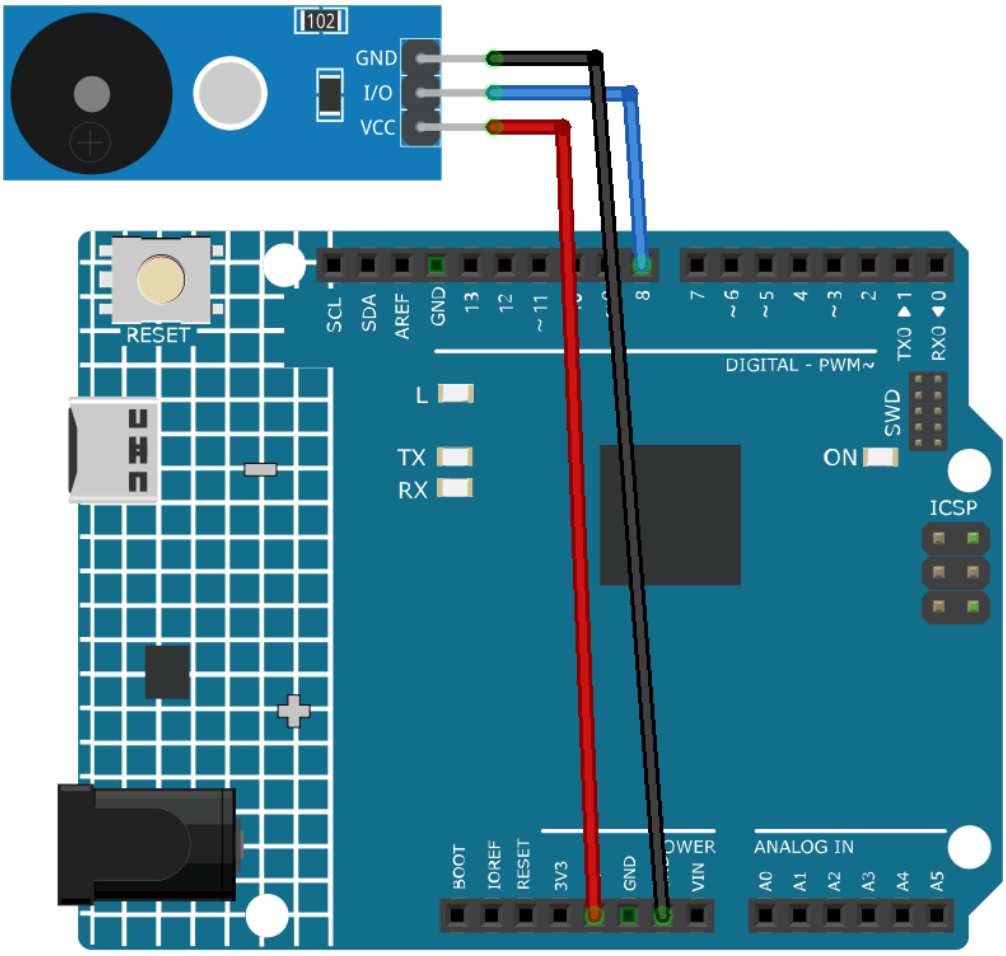
Das Arbeitsprinzip des passiven Summermoduls basiert auf dem piezoelektrischen Effekt. Sobald ein elektrisches Signal an den Summer angelegt wird, bringt es einen im Inneren befindlichen piezoelektrischen Kristall zum Schwingen. Diese Schwingungen erzeugen Schallwellen. Die Frequenz des erzeugten Tons hängt von der Frequenz des zugeführten elektrischen Signals ab. Durch Variation der Signal-Frequenz kann die Tonhöhe verändert werden.

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Passives Summermodul * 1
- Jumperkabel

Schaltungsaufbau



Code

Code-Erläuterung

1. Einbindung der Tonhöhen-Bibliothek: Diese Bibliothek stellt die Frequenzwerte für verschiedene Musiknoten bereit.

```
#include "pitches.h"
```

2. Definition von Konstanten und Arrays:
   - `buzzerPin` ist der digitale Pin am Arduino, an dem der Summer angeschlossen ist.
   - `melody[]` ist ein Array, das die Reihenfolge der zu spielenden Noten speichert.
   - `noteDurations[]` ist ein Array, das die Dauer jeder Note in der Melodie speichert.

```
const int buzzerPin = 8;
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};
int noteDurations[] = {
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  4, 8, 8, 4, 4, 4, 4, 4
};
```

3. Melodiewiedergabe:

   - Die `for`-Schleife geht jede Note der Melodie durch.
   - Die Funktion `tone()` spielt eine Note über den Summer für eine bestimmte Dauer ab.
   - Eine Verzögerung zwischen den Noten sorgt für eine klare Trennung.
   - Die Funktion `noTone()` beendet die Tonausgabe.

```
void setup() {
  for (int thisNote = 0; thisNote < 8; thisNote++) {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(buzzerPin, melody[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(buzzerPin);
  }
}
```

4. Leere Loop-Funktion: Da die Melodie nur einmal im Setup gespielt wird, ist die Loop-Funktion leer.

Weitere Ideen

- Melodie ändern: Experimentieren Sie mit anderen Noten und Dauern in den Arrays `melody[]` und `noteDurations[]`, um eigene Melodien zu erzeugen. Es gibt ein GitHub-Repository (), das Arduino-Codes für verschiedene Lieder bereitstellt.
- Schalter integrieren: Fügen Sie einen Taster in die Schaltung ein und passen Sie den Code so an, dass die Melodie nur dann gespielt wird, wenn der Taster gedrückt wird.

Weitere Projekte

- *Türklingel*
- *Gasleck-Alarm*
- *Bluetooth-Klavier*

2.4.28 Servomotor (SG90)

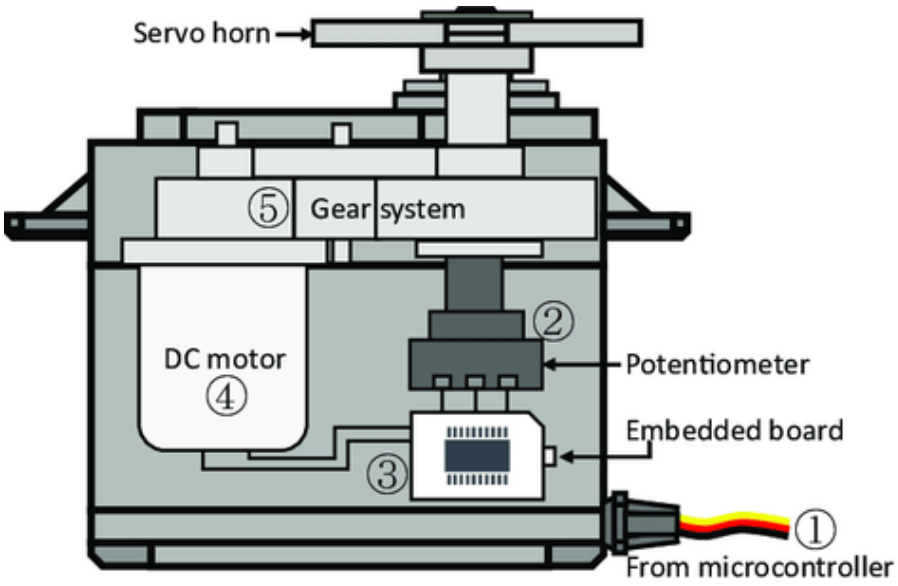
Einleitung

Servomotoren sind Geräte, die sich auf einen bestimmten Winkel oder eine bestimmte Position drehen können. Sie kommen beispielsweise in Roboterarmen, Lenkrädern oder Kameragimbals zum Einsatz. Ein Servomotor hat drei Kabel: Stromversorgung, Masse und Signal. Das rote Kabel dient der Stromversorgung und sollte an den 5V-Pin des Arduino-Boards angeschlossen werden. Das schwarze oder braune Kabel ist die Masse und sollte mit einem Ground-Pin des Boards verbunden werden. Das gelbe oder orangefarbene Kabel ist das Signalkabel und sollte an einen PWM-Pin des Boards angeschlossen werden.

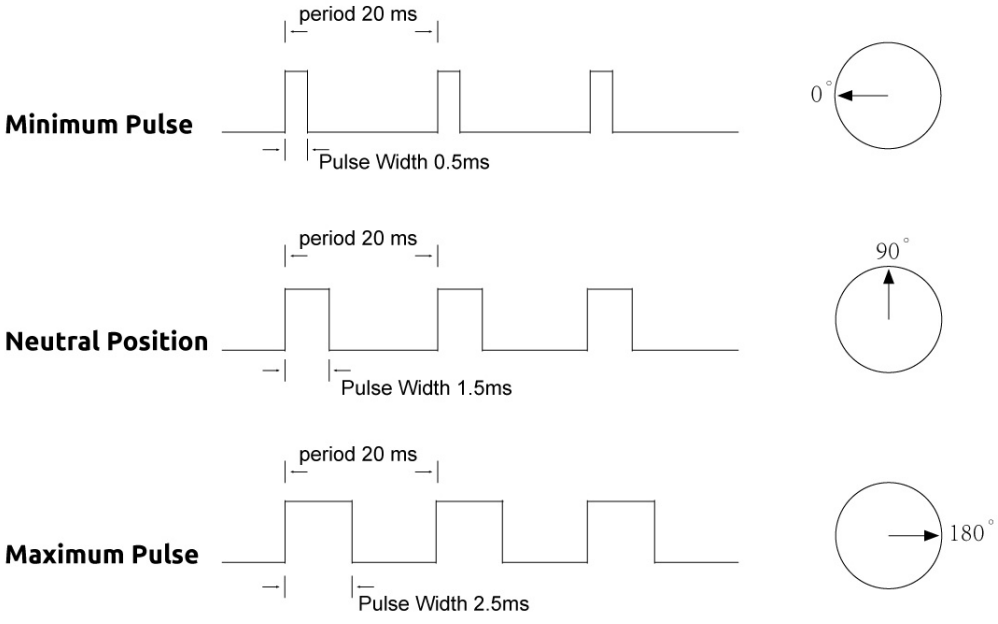
Funktionsprinzip

Ein Servo besteht in der Regel aus den folgenden Teilen: Gehäuse, Welle, Getriebesystem, Potenziometer, Gleichstrommotor und einer eingebetteten Platine.

So funktioniert es: Der Mikrocontroller sendet PWM-Signale an den Servo. Die eingebettete Platine im Servo empfängt diese Signale und steuert den internen Motor. Daraufhin treibt der Motor das Getriebesystem an, das die Welle dreht. Die Welle und das Potenziometer des Servos sind miteinander verbunden. Bei einer Drehung der Welle verändert sich die Spannung am Potenziometer, die wiederum an die Platine gesendet wird. Diese steuert dann die Drehrichtung und -geschwindigkeit, sodass der Servo präzise in der definierten Position stoppen und dort verharren kann.



Der Winkel wird durch die Dauer eines Pulses bestimmt, der auf das Steuerkabel aufgebracht wird. Dies wird als Pulsweitenmodulation bezeichnet. Der Servo erwartet alle 20 ms einen Puls. Die Länge des Pulses bestimmt, wie weit sich der Motor dreht. Zum Beispiel wird ein 1,5 ms langer Puls den Motor in die 90-Grad-Position (Neutralstellung) drehen. Wenn ein Puls mit einer Dauer von weniger als 1,5 ms an einen Servo gesendet wird, dreht dieser sich um einige Grade gegen den Uhrzeigersinn von der Neutralstellung weg und hält dort. Ist der Puls länger als 1,5 ms, tritt das Gegenteil ein. Die minimale und maximale Pulslänge, die den Servo zu einer gültigen Position bewegen, sind eigenschaften jedes einzelnen Servos. In der Regel beträgt die minimale Pulslänge etwa 0,5 ms und die maximale Pulslänge etwa 2,5 ms.

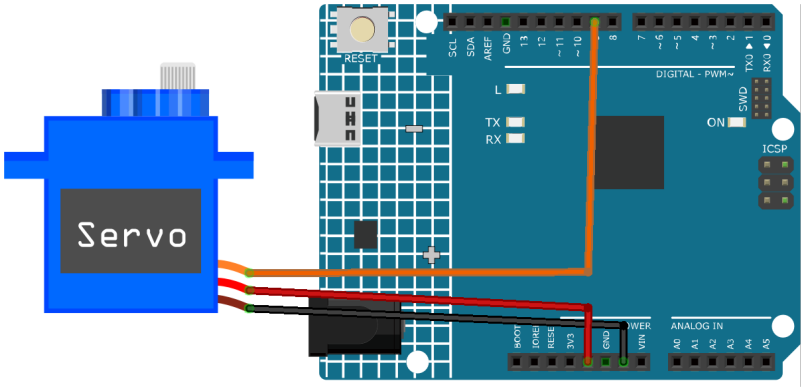


Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Servomotor * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erläuterung

1. Hier wird die `Servo`-Bibliothek eingebunden, die eine einfache Steuerung des Servomotors ermöglicht. Der Pin, an dem der Servo angeschlossen ist, und der Anfangswinkel werden ebenfalls definiert.

```
#include <Servo.h>
const int servoPin = 9;  // Define the servo pin
int angle = 0;           // Initialize the angle variable to 0 degrees
Servo servo;             // Create a servo object
```

2. Die Funktion `setup()` wird einmal ausgeführt, wenn der Arduino startet. Mit der Funktion `attach()` wird der Servo am definierten Pin angeschlossen.

```
void setup() {
  servo.attach(servoPin);
}
```

3. Die Haupt-Schleife enthält zwei `for`-Schleifen. Die erste Schleife erhöht den Winkel von 0 bis 180 Grad, und die zweite verringert ihn von 180 bis 0 Grad. Der Befehl `servo.write(angle)` setzt den Servo auf den angegebenen Winkel. Die `delay(15)`-Anweisung bewirkt, dass der Servo 15 Millisekunden wartet, bevor er zum nächsten Winkel übergeht.

```
void loop() {
  // scan from 0 to 180 degrees
  for (angle = 0; angle < 180; angle++) {
    servo.write(angle);
    delay(15);
  }
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  // now scan back from 180 to 0 degrees
  for (angle = 180; angle > 0; angle--) {
    servo.write(angle);
    delay(15);
  }
}
```

Weitere Ideen

- Steuerung des Servos über ein Potenziometer. Die Drehung des Potenziometers könnte direkt den Winkel des Servos steuern.

Weitere Projekte

- *Intelligenter Mülleimer*
- *Bluetooth-Schlosssteuerung*

2.4.29 Kreiselpumpe

Einleitung

Eine Kreiselpumpe ist ein Gerät, das Flüssigkeiten mittels eines rotierenden Laufrads von einem Ort zum anderen befördern kann. Sie kann zum Pumpen von Wasser, Öl, Chemikalien usw. verwendet werden. Eine Kreiselpumpe besteht aus zwei Hauptkomponenten: einem Motor und einer Pumpe. Der Motor versorgt die Pumpe mit Energie, und die Pumpe wandelt die Rotationsenergie in Druck und Durchfluss um.

Funktionsprinzip

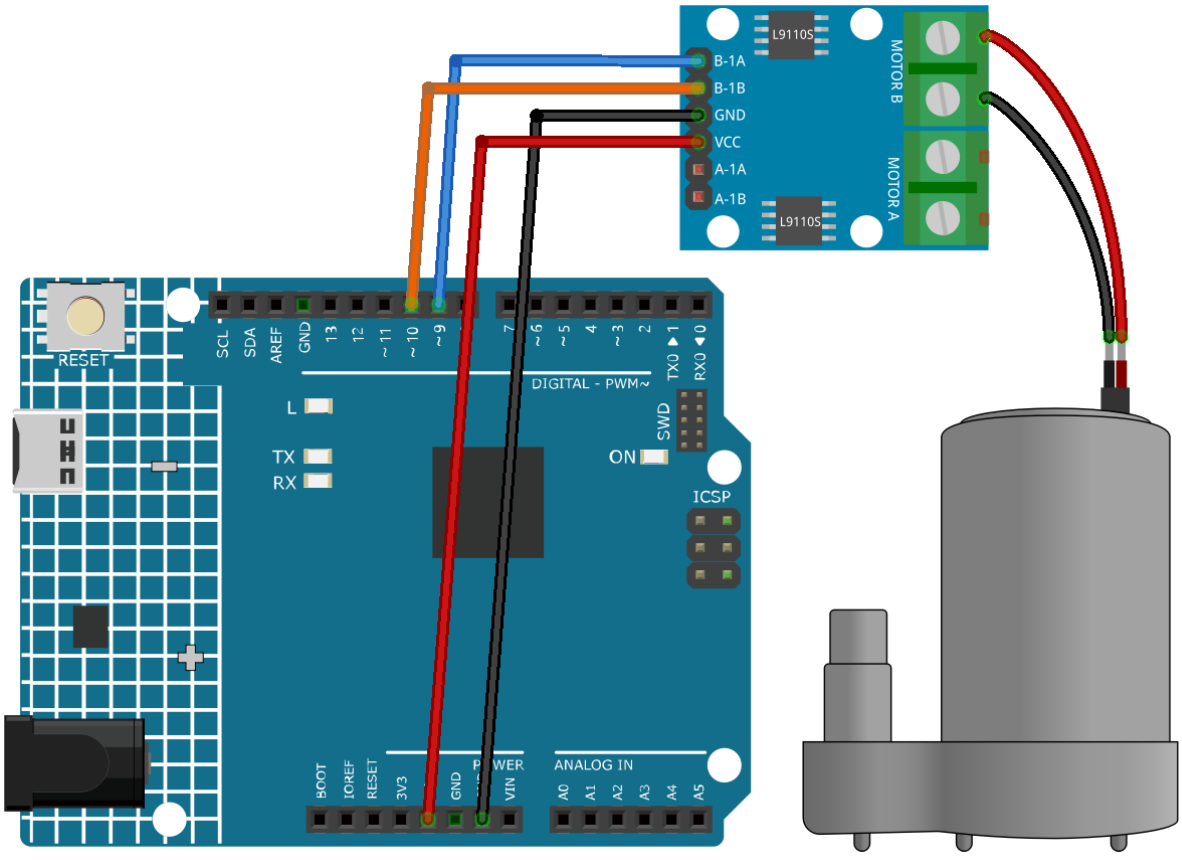
Die Kreiselpumpe funktioniert mit einem sich drehenden Laufrad, das die Geschwindigkeit der Flüssigkeit erhöht und sie durch ein Einlassrohr in die Pumpe zieht. Sobald die Flüssigkeit den äußeren Rand des Laufrads verlässt, wird sie durch die Zentrifugalkraft durch ein Auslassrohr gedrückt, was zu einem erhöhten Druck führt. Je schneller das Laufrad rotiert, desto höher sind der Druck und der Durchfluss der Flüssigkeit.

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- Kreiselpumpe * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erklärung

1. Zwei Pins werden für die Motorsteuerung definiert, konkret `motorB_1A` und `motorB_2A`. Diese Pins verbinden sich mit der L9110 Motorsteuerplatine, um die Richtung und Geschwindigkeit des Motors zu steuern.

```
const int motorB_1A = 9;
const int motorB_2A = 10;
```

2. Konfiguration der Pins und Steuerung des Motors:

   - Die `setup()`-Funktion initialisiert die Pins als `OUTPUT`, sodass sie Signale an die Motorsteuerplatine senden können.
   - Die Funktion `analogWrite()` wird verwendet, um die Motorgeschwindigkeit einzustellen. Hier bewirkt das Setzen eines Pins auf `HIGH` und des anderen auf `LOW`, dass sich die Pumpe in eine Richtung dreht. Nach einer Verzögerung von 5 Sekunden werden beide Pins auf 0 gesetzt, um den Motor auszuschalten.

```
void setup() {
   pinMode(motorB_1A, OUTPUT);  // Pin 1 der Pumpe als Ausgang definieren
   pinMode(motorB_2A, OUTPUT);  // Pin 2 der Pumpe als Ausgang definieren
   analogWrite(motorB_1A, HIGH);
   analogWrite(motorB_2A, LOW);
   delay(5000);  // 5 Sekunden warten
   analogWrite(motorB_1A, 0);  // Pumpe ausschalten
   analogWrite(motorB_2A, 0);
}
```

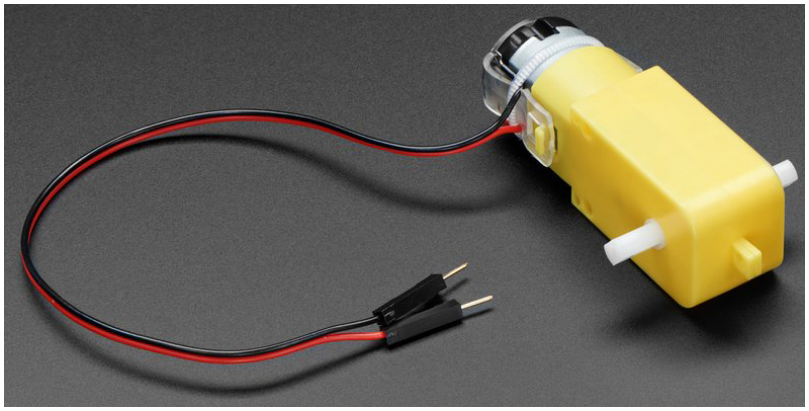
Weitere Ideen

- Ändern der Pumpenrichtung durch Vertauschen der `HIGH` und `LOW` Werte zwischen den Pins.
- Implementieren Sie ein System, bei dem der Pumpenzustand (an/aus) durch einen Tastendruck umgeschaltet wird.
- Verwenden Sie ein Potenziometer, um die Geschwindigkeit der Pumpe mittels PWM zu steuern.
- Integrieren Sie Sensoren, um den Pumpenbetrieb automatisch auf der Grundlage bestimmter Bedingungen zu steuern, z. B. das Ein- und Ausschalten der Pumpe je nach Wasserstand in einem Tank.

Weitere Projekte

- *Automatischer Seifenspender*
- *Automatisches Bewässerungssystem mit Blynk*

2.4.30 TT Motor



Einführung

Ein TT-Motor ist eine Art von Gleichstrommotor, an den ein Getriebe angekoppelt ist. Dieses Getriebe reduziert die Geschwindigkeit des Motors und erhöht sein Drehmoment. TT-Motoren werden häufig in Anwendungen wie Antriebsrädern, Propellern, Ventilatoren und ähnlichem eingesetzt. Ein TT-Motor verfügt über zwei Kabel: ein positives und ein negatives. Das positive Kabel ist in der Regel rot und das negative Kabel in der Regel schwarz.

Im Produkt wird ein TT-Gleichstromgetriebemotor mit einem Übersetzungsverhältnis von 1:48 verwendet. Er ist mit 2 x 200 mm Kabeln mit 0,1“-Steckverbindern ausgestattet, die in ein Steckbrett passen. Ideal zum Einstecken in ein Steckbrett oder eine Anschlussleiste.

Sie können diese Motoren mit 3 ~ 6VDC betreiben, allerdings werden sie bei höheren Spannungen natürlich schneller laufen.

Funktionsprinzip

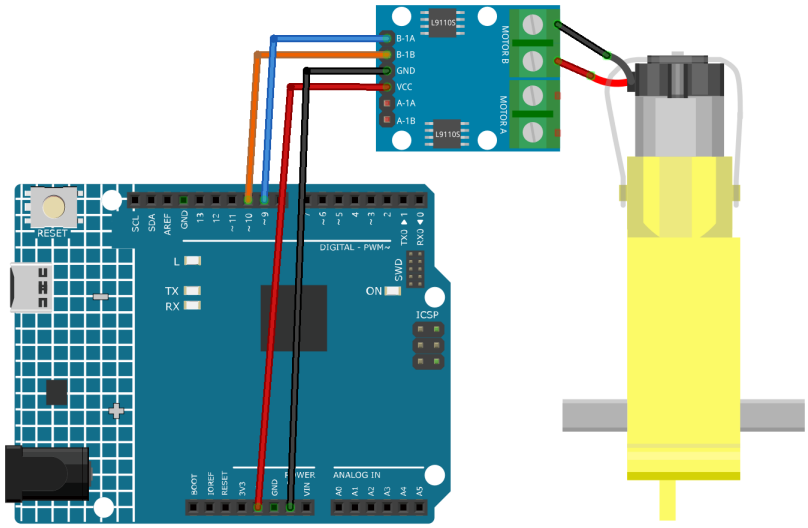
Ein TT-Motor funktioniert, indem er elektrische Energie in mechanische Energie umwandelt. Wenn eine Spannung an die Kabel des Motors angelegt wird, entsteht ein Magnetfeld, das den Motor zum Drehen bringt. Die Geschwindigkeit und die Drehrichtung des Motors hängen von der Spannung und der Polarität der Stromquelle ab. Je höher die Spannung, desto schneller dreht der Motor. Umkehren der Polarität bewirkt, dass der Motor in die entgegengesetzte Richtung dreht.

Anwendungsbeispiele

Hardware-Komponenten

- Arduino Uno R4 oder R3 Board * 1
- TT Motor * 1
- Jumperkabel

Schaltungsaufbau



Programmcode

Code-Erklärung

1. Der erste Teil des Codes definiert die Steuerpins des Motors. Diese sind mit der L9110-Motorsteuerplatine verbunden.

```
// Motorpins definieren
const int motorB_1A = 9;
const int motorB_2A = 10;
```

2. Die `setup()`-Funktion initialisiert die Motorsteuerpins als Ausgang mit der Funktion `pinMode()`. Anschließend wird mit `analogWrite()` die Geschwindigkeit des Motors eingestellt. Der an `analogWrite()` übergebene Wert kann zwischen 0 (aus) und 255 (volle Geschwindigkeit) variieren. Danach wird die Funktion `delay()` verwendet, um den Code für 5000 Millisekunden (oder 5 Sekunden) anzuhalten, wonach die Motorgeschwindigkeit auf 0 (aus) gesetzt wird.

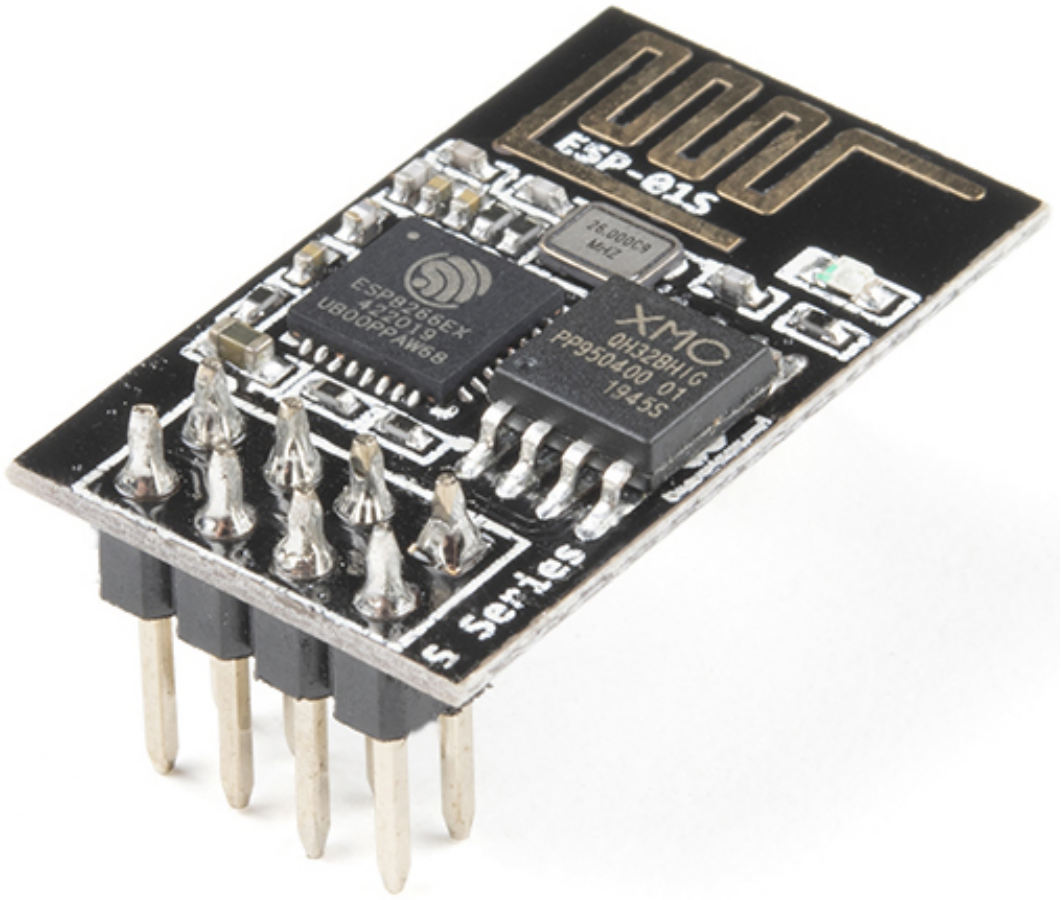
```
void setup() {
  pinMode(motorB_1A, OUTPUT);  // set motor pin 1 as output
  pinMode(motorB_2A, OUTPUT);  // set motor pin 2 as output

  analogWrite(motorB_1A, 255);  // set motor speed (0-255)
  analogWrite(motorB_2A, 0);

  delay(5000);

  analogWrite(motorB_1A, 0);
  analogWrite(motorB_2A, 0);
}
```

Weitere Ideen

- Geschwindigkeitssteuerung des Motors mit einem Potentiometer: Anstatt die Motorgeschwindigkeit fest vorzugeben, könnten Sie ein Potentiometer verwenden, um die Geschwindigkeit des Motors dynamisch zu steuern.

Drahtlose Kommunikation & IoT

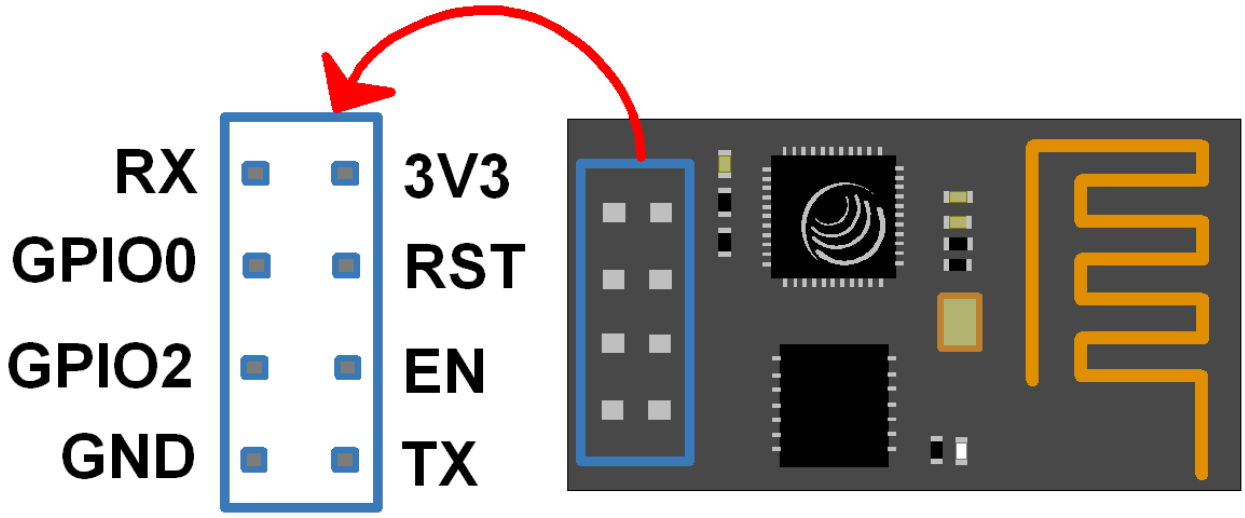
2.4.31 ESP8266-Modul



Der ESP8266 ist ein kostengünstiger Wi-Fi-Mikrochip, ausgestattet mit integrierter TCP/IP-Netzwerksoftware sowie Mikrocontroller-Fähigkeiten, produziert von Espressif Systems in Shanghai, China.

Auf den Westen wurde erstmals im August 2014 durch das ESP-01-Modul aufmerksam gemacht, welches von einem Dritthersteller, Ai-Thinker, produziert wird. Dieses kleine Modul ermöglicht es Mikrocontrollern, sich mit einem Wi-Fi-Netzwerk zu verbinden und einfache TCP/IP-Verbindungen mit Hayes-ähnlichen Befehlen herzustellen. Zu Beginn gab es kaum Dokumentation in englischer Sprache zu dem Chip und den akzeptierten Befehlen. Die sehr geringen Kosten und die minimale Anzahl externer Bauteile, die auf eine kostengünstige Massenproduktion hindeuten, weckten das Interesse vieler Entwickler.

Pinbelegung des ESP8266 und deren Funktionen:

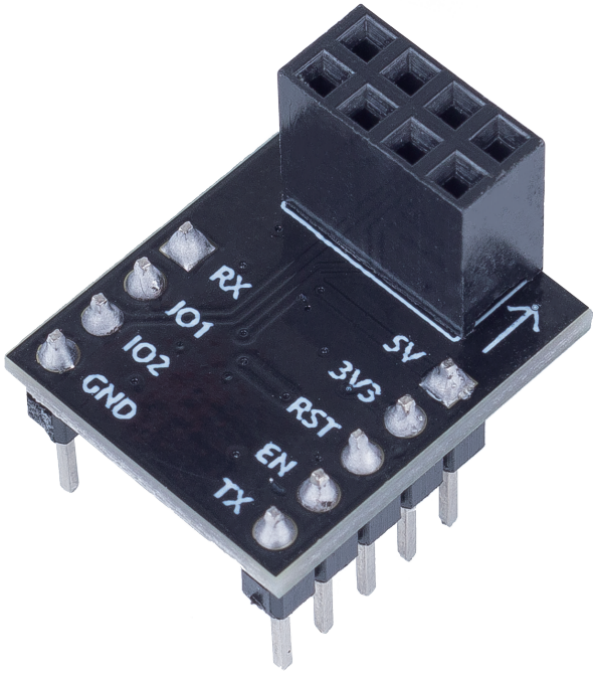


Tab. 1: ESP8266-01 Pins

Pin	Name	Beschreibung
1	TXD	UART_TXD, Senden; Allgemeiner Eingabe/Ausgabe-Pin: GPIO1; Pull-down beim Start nicht zulässig.
2	GND	GND
3	CU_PD	Funktioniert auf hohem Pegel; Abschaltung bei niedrigem Pegel.
4	GPIO2	Muss beim Einschalten auf hohem Pegel sein, Hardware-Pull-down ist nicht zulässig; Standardmäßig Pull-up.
5	RST	Externes Reset-Signal, Reset bei niedrigem Pegel; Funktioniert bei hohem Pegel (standardmäßig hoher Pegel).
6	GPIO0	WiFi-Statusanzeige; Betriebsmodus-Auswahl: Pull-up: Flash-Boot, Betriebsmodus; Pull-down: UART-Download, Download-Modus.
7	VCC	Stromversorgung (3,3V)
8	RXD	UART_RXD, Empfangen; Allgemeiner Eingabe/Ausgabe-Pin: GPIO3;

- ESP8266 - Espressif
-

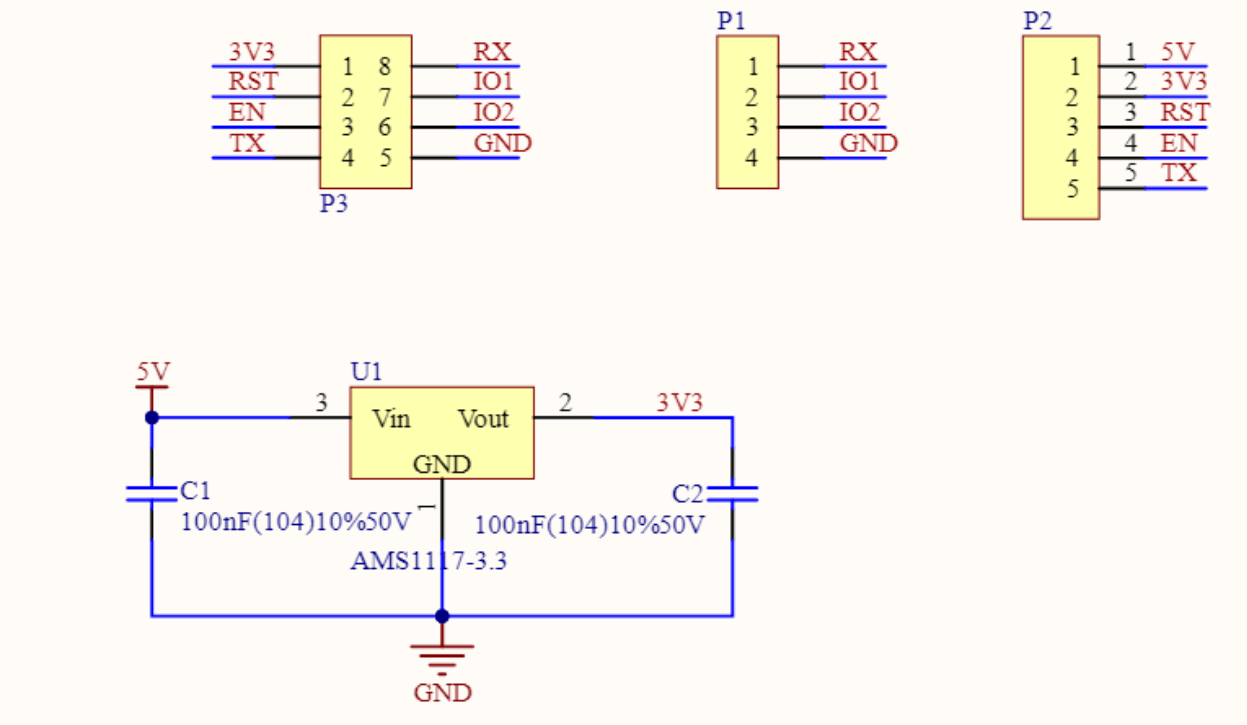
ESP8266-Adapter



Der ESP8266-Adapter ist eine Erweiterungsplatine, die es ermöglicht, das ESP8266-Modul auf einem Steckbrett zu verwenden.

Die Platine ist perfekt auf die Pins des ESP8266 abgestimmt und fügt zudem einen 5V-Pin hinzu, um die Spannung vom Arduino-Board zu erhalten. Der integrierte AMS1117-Chip wird verwendet, um das ESP8266-Modul nach Spannungsreduktion auf 3,3V zu betreiben.

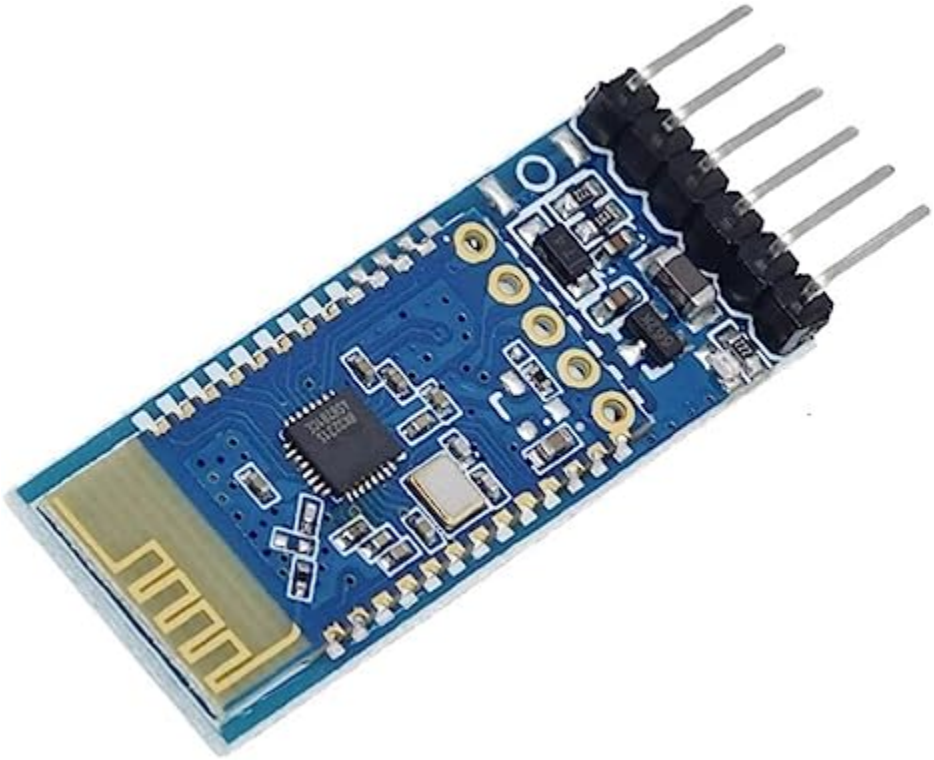
Das Schaltbild ist wie folgt:



Weitere Projekte

- *IoT-Projekte*

2.4.32 JDY-31 Bluetooth-Modul

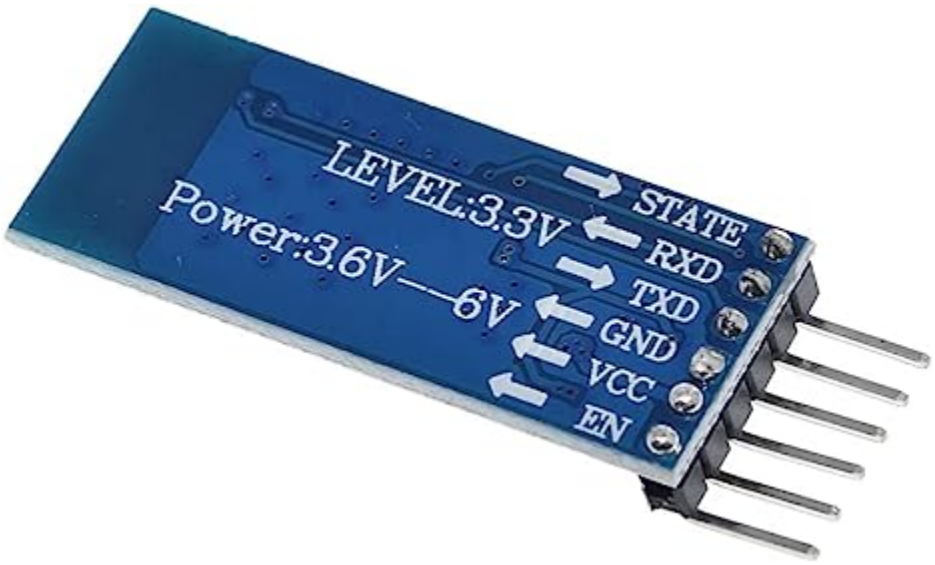


Warnung: Dieses Modul **unterstützt keine Verbindungen mit Apple-Geräten**, daher sind für Tutorials mit diesem Modul ein Android-Smartphone oder -Tablet erforderlich.

Das JDY-31 Bluetooth-Modul dient als pin-kompatibler Ersatz für das HC-06 Bluetooth-Modul. Es ist unkomplizierter und einfacher in der Anwendung als das HC-06 und oft zu einem etwas geringeren Preis erhältlich.

Das JDY-31 Bluetooth-Modul basiert auf dem Bluetooth 3.0 SPP-Design und unterstützt die Datenübertragung unter Windows, Linux und Android. Die Betriebsfrequenz des JDY-31 Bluetooth-Moduls beträgt 2,4 GHz mit der Modulationsart GFSK. Die maximale Sendeleistung beträgt 8 dB, und die maximale Übertragungsentfernung liegt bei 30 Metern. Über AT-Befehle können Benutzer den Gerätenamen, die Baudrate und weitere Einstellungen ändern.

Pins des JDY-31 und ihre Funktionen:



Tab. 2: JDY-31 Pins

Pin	Bezeichnung	Beschreibung
1	STATE	Statuspin für die Verbindung (niedriges Niveau bei fehlender Verbindung, hohes Niveau nach Verbindung)
2	RXD	Empfängerpin, dieser Pin muss mit dem TX-Pin des nächsten Geräts verbunden sein.
3	TXD	Senderpin, dieser Pin muss mit dem RX-Pin des nächsten Geräts verbunden sein.
4	GND	Masse (GND)
5	VCC	Stromversorgung (1,8-3,6V, empfohlen 3,3V)
6	EN	Aktiviert oder deaktiviert das Modul. Wird dieser Pin auf hohem Niveau gehalten, ist das Modul aktiv und beginnt mit der Datenübertragung.

Patch-Anwendung: Für allgemeine Anwendungen müssen nur die Pins VCC, GND, RXD und TXD verbunden werden. Um die Verbindung aktiv zu trennen, senden Sie im Verbindungsstatus den Befehl AT+DISC.

AT-Befehlssatz

Befehl	Funktion	Standard
AT+VERSION	Versionsnummer	JDY-31-V1.2
AT+RESET	Soft-Reset	
AT+DISC	Trennen (gültig bei bestehender Verbindung)	
AT+LADDR	MAC-Adresse des Moduls abfragen	
AT+PIN	Verbindungspasswort setzen oder abfragen	1234
AT+BAUD	Baudrate setzen oder abfragen	9600
AT+NAME	Sendenamen setzen oder abfragen	JDY-31-SPP
AT+DEFAULT	Werkseinstellungen wiederherstellen	
AT+ENLOG	Statusausgabe der seriellen Schnittstelle	1

Weitere Projekte

- *Bluetooth-Einstieg*
- *Bluetooth LCD*
- *Bluetooth-Ampel*
- *Bluetooth-Schlosssteuerung*
- *Bluetooth RGB-Controller*
- *Bluetooth-Umweltmonitor*
- *Bluetooth-Klavier*
- *Bluetooth OLED*
- *Bluetooth-Fernrelais*
- iot_Bluetooth_voice_control_relay

2.5 IoT-Projekte

Dieses Kit enthält das ESP8266-WLAN-Modul und das JDY-31-Bluetooth-Modul. Mit dem ESP8266 können Sie Ihr Arduino an das Internet anschließen, um IoT-Experimente durchzuführen. Das Bluetooth-Modul ermöglicht hingegen drahtlose Kommunikation auf kurzer Distanz.

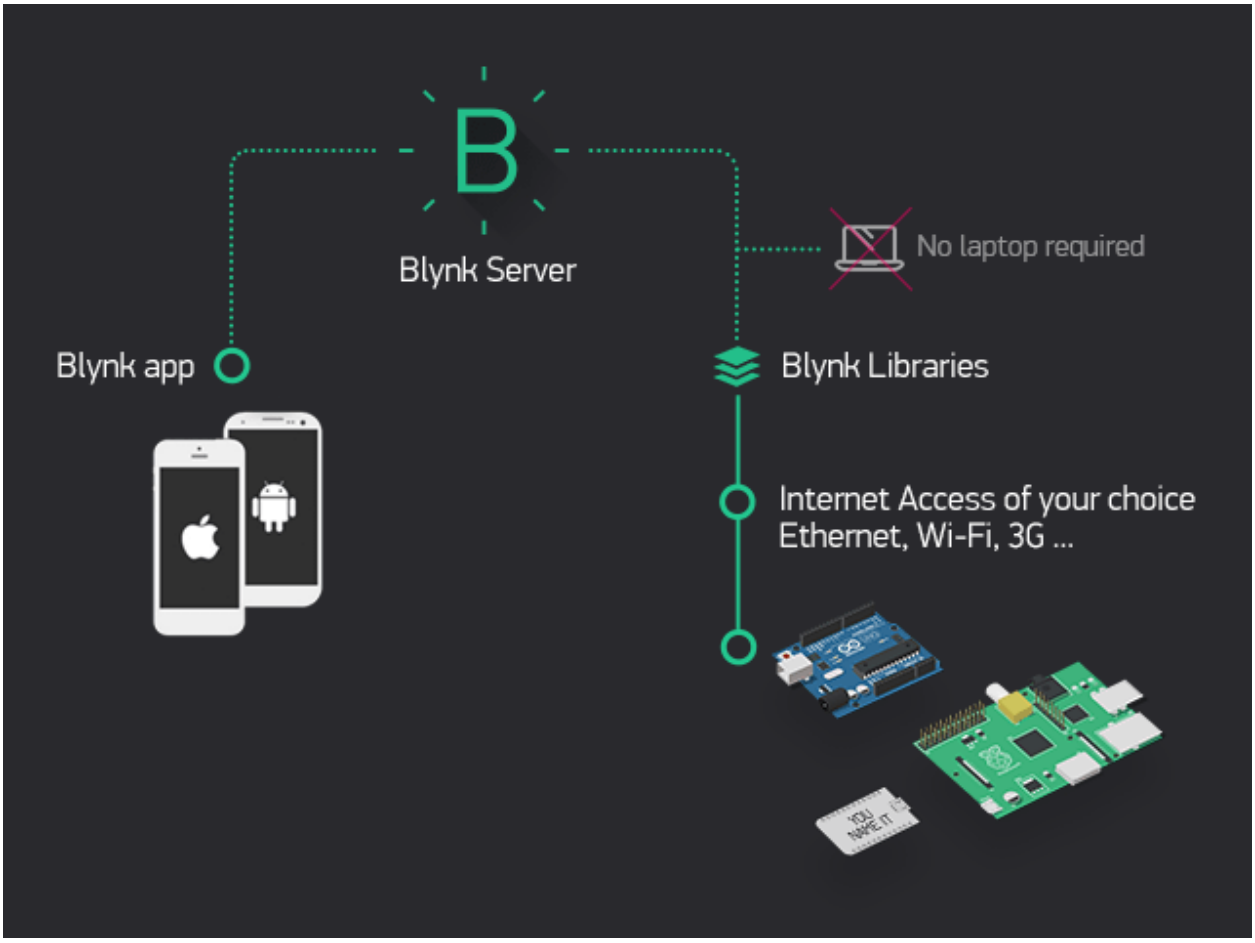
Mithilfe des ESP8266-WLAN-Moduls können Sie eine Vielzahl von IoT-Projekten erkunden, die die Verbindung Ihres Arduino mit dem Internet ermöglichen. Wir bieten Ihnen schrittweise Anleitungen zur Arbeit mit Plattformen wie und an. Damit können Sie Alarmierungssysteme, Fernbedienungen und Umweltüberwachungssysteme einrichten, um nur einige Anwendungen zu nennen.

Das JDY-31-Bluetooth-Modul eröffnet Ihnen Möglichkeiten für lokale drahtlose Kommunikation. Wir leiten Sie durch die Kopplung Ihres Arduino mit Smartphones oder anderen Bluetooth-fähigen Geräten für unterschiedliche Steuerungs- und Überwachungsaufgaben. Zudem können Sie Projekte mit Apps realisieren, die mit entwickelt wurden.

WLAN

2.5.1 Einführung in Blynk

Blynk ist eine umfassende Softwarelösung, die alle notwendigen Komponenten zum Prototyping, zur Implementierung und zur Fernverwaltung von vernetzten elektronischen Geräten bereitstellt – egal, ob es sich um persönliche IoT-Projekte oder um Millionen kommerziell vernetzte Produkte handelt. Mit Blynk kann jeder seine Hardware mit der Cloud verbinden und Code-freie Anwendungen für iOS, Android und Web entwickeln. Diese Anwendungen ermöglichen die Analyse von Echtzeit- und Historiendaten der Geräte, die Fernsteuerung aus der ganzen Welt, den Empfang wichtiger Benachrichtigungen und vieles mehr.



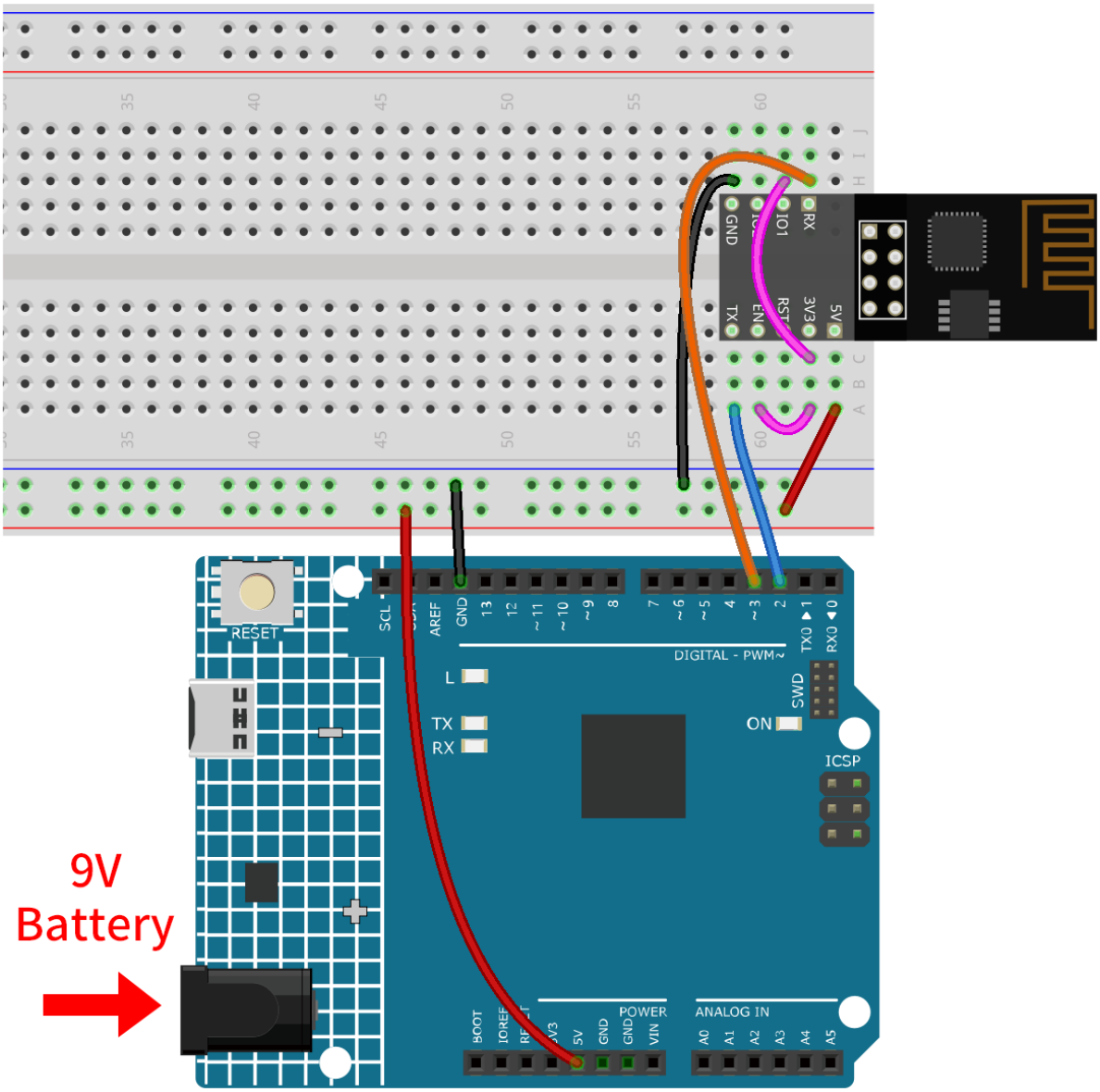
Um die Kommunikation des R4-Boards mit Blynk zu ermöglichen, ist bei der ersten Verwendung von Blynk eine Konfiguration erforderlich.

Folgen Sie den nachstehenden Schritten in der vorgegebenen Reihenfolge, und lassen Sie keine Kapitel aus.

1.1 Konfiguration des ESP8266

Das im Kit enthaltene ESP8266-Modul ist bereits mit der AT-Firmware vorinstalliert. Dennoch müssen Sie seine Konfiguration gemäß den folgenden Schritten anpassen.

1. Bauen Sie den Schaltkreis.



2. Öffnen Sie die Datei `00-Set_software_serial.ino` unter dem Pfad `ultimate-sensor-kit\iot_project\wifi\00-Set_software_serial`. Oder kopieren Sie diesen Code in die Arduino-IDE und laden Sie den Code hoch.

   Der Code etabliert eine softwaregesteuerte serielle Kommunikation mit Hilfe der SoftwareSerial-Bibliothek von Arduino. Dadurch kann das Arduino-Board mit dem ESP8266-Modul über die digitalen Pins 2 und 3 (als Rx und Tx) kommunizieren. Es prüft die Datenübertragung zwischen den beiden und leitet empfangene Nachrichten mit einer Baudrate von 115200 weiter. **Mit diesem Code können Sie den seriellen Monitor des Arduino verwenden, um AT-Firmware-Befehle an das ESP8266-Modul zu senden und dessen Antworten zu empfangen.**

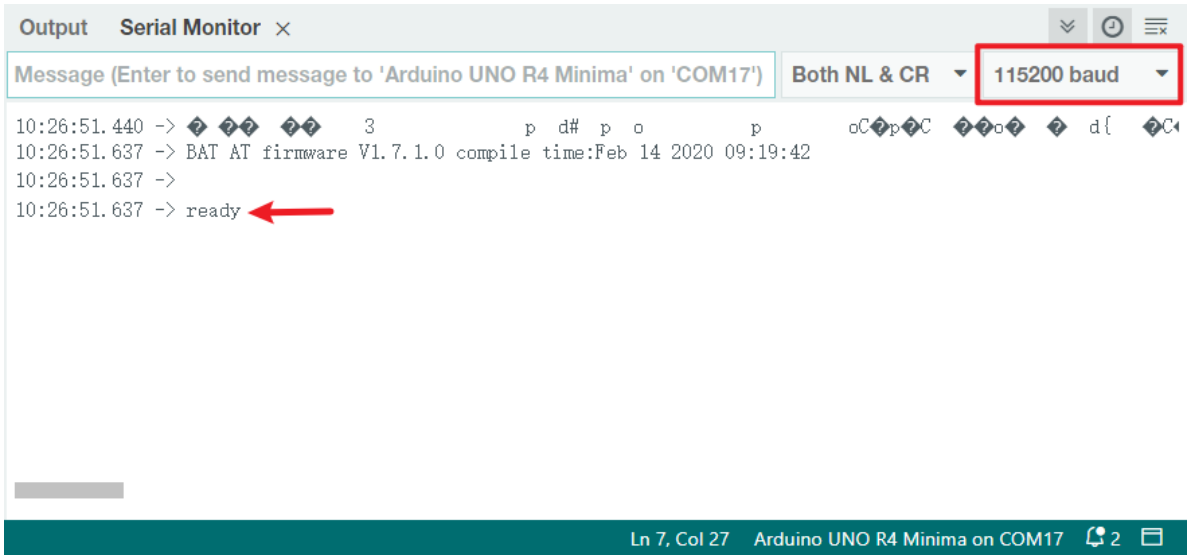
```
#include <SoftwareSerial.h>
SoftwareSerial espSerial(2, 3); //Rx,Tx

void setup() {
    // Put your setup code here, to run once:
    Serial.begin(115200);
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    espSerial.begin(115200);
}

void loop() {
    if (espSerial.available()) {
        Serial.write(espSerial.read());
    }
    if (Serial.available()) {
        espSerial.write(Serial.read());
    }
}
```

3. Klicken Sie auf das Lupensymbol (Serial Monitor) in der oberen rechten Ecke und stellen Sie die Baudrate auf **115200** ein. (Sie könnten einige Informationen wie ich gedruckt haben oder auch nicht, das spielt keine Rolle, gehen Sie einfach zum nächsten Schritt über.)

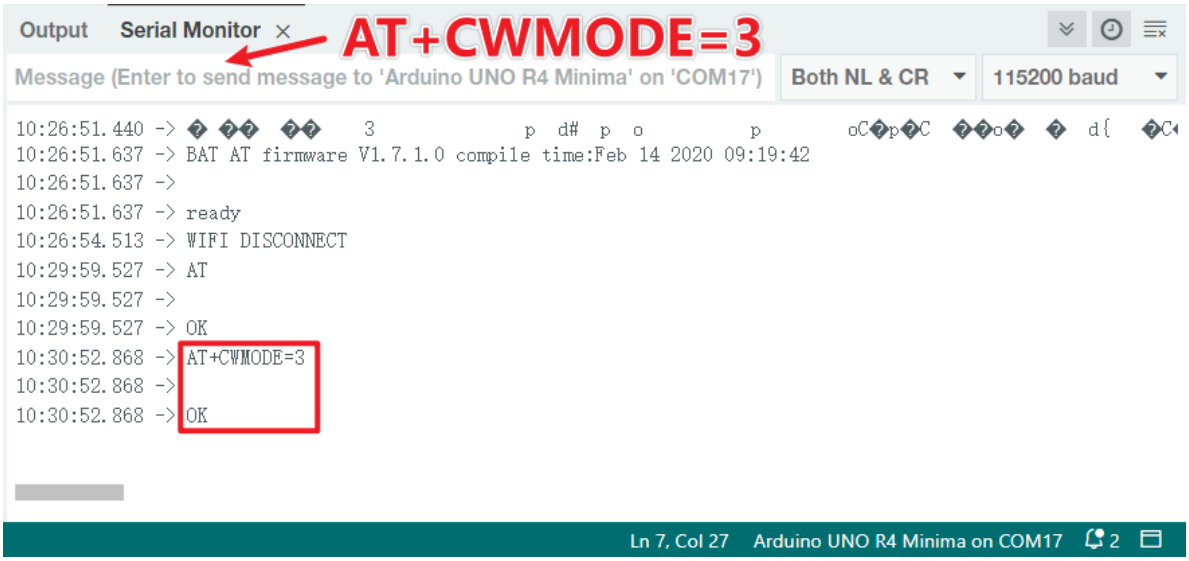


Warnung:

- Sollte `ready` nicht erscheinen, können Sie versuchen, das ESP8266-Modul zurückzusetzen (RST an GND anschließen) und den Serial Monitor erneut zu öffnen.
- Falls das Ergebnis `OK` lautet, müssen Sie möglicherweise die Firmware erneut brennen. Bitte beziehen Sie sich für Details auf *Wie man die Firmware für das ESP8266-Modul erneut aufspielt*. Sollten Sie immer noch Probleme haben, machen Sie bitte einen Screenshot des seriellen Monitors und senden Sie ihn an sevice@sunfounder.com. Wir werden Ihnen so schnell wie möglich weiterhelfen.

4. Klicken Sie auf **NEWLINE DROPDOWN BOX**, wählen Sie `both NL & CR` im Dropdown-Menü aus, geben Sie `AT` ein; wenn OK zurückgegeben wird, bedeutet dies, dass die Verbindung zwischen ESP8266 und R4-Board erfolgreich hergestellt wurde.

5. Geben Sie `AT+CWMODE=3` ein, um den Verwaltungsmodus auf **Station und AP**-Koexistenz zu ändern.

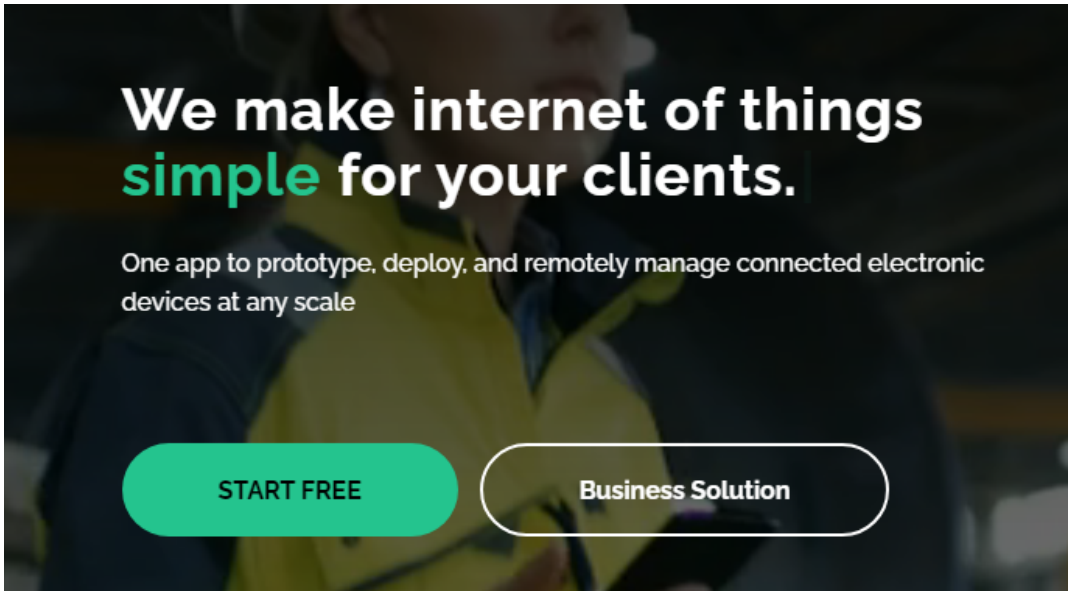


Referenzen

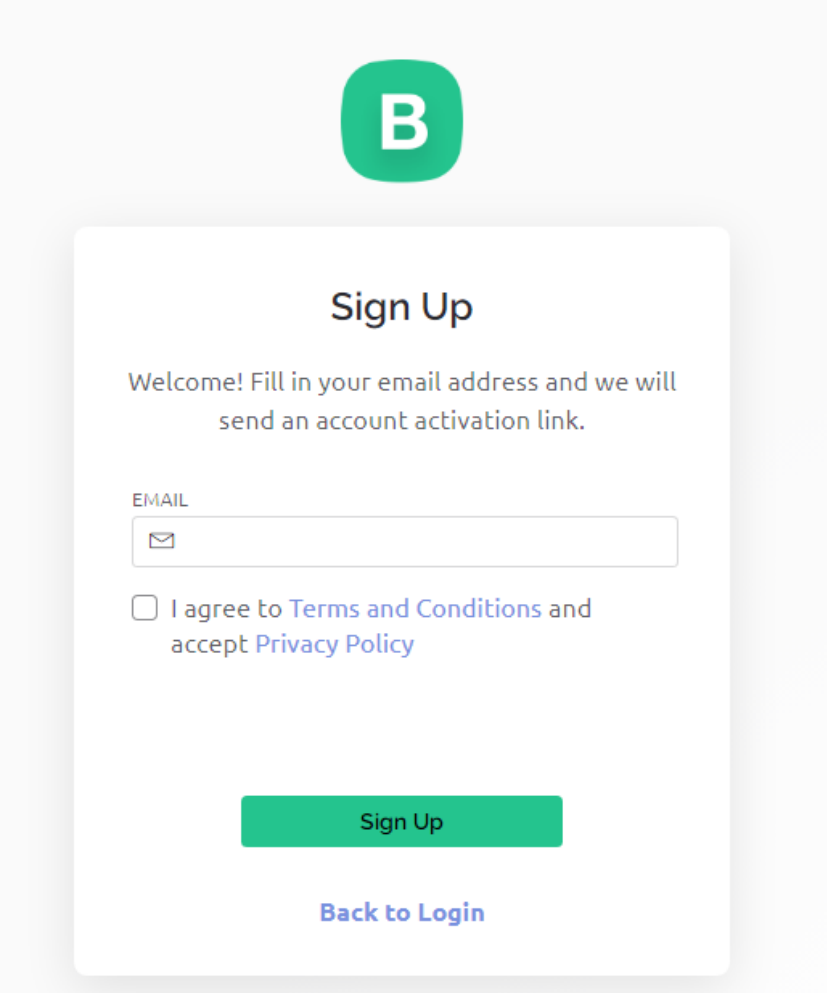
-

1.2 Konfiguration von Blynk

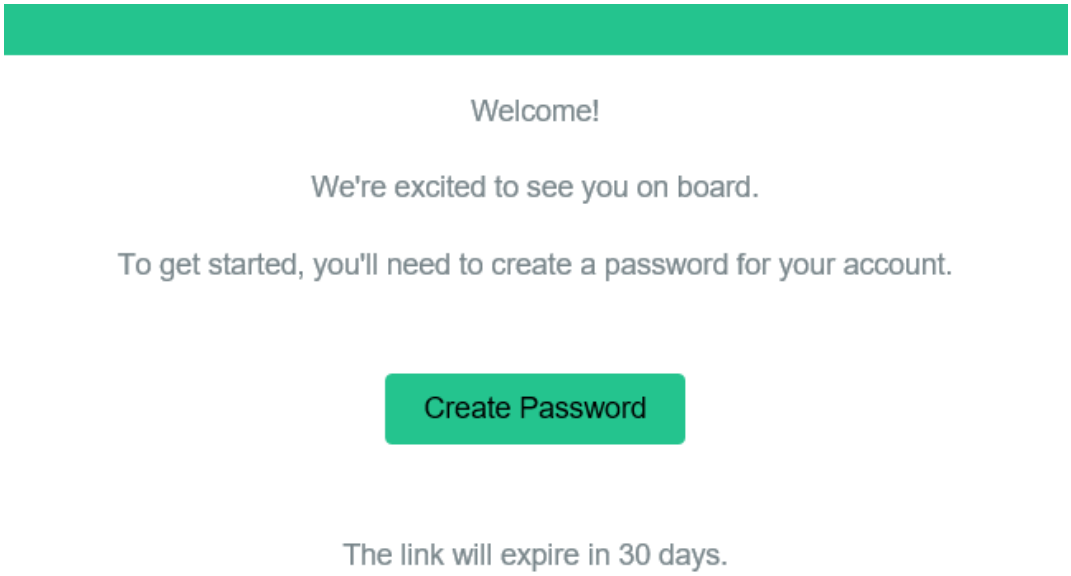
1. Besuchen Sie die BLYNK Webseite und klicken Sie auf **START FREE**.



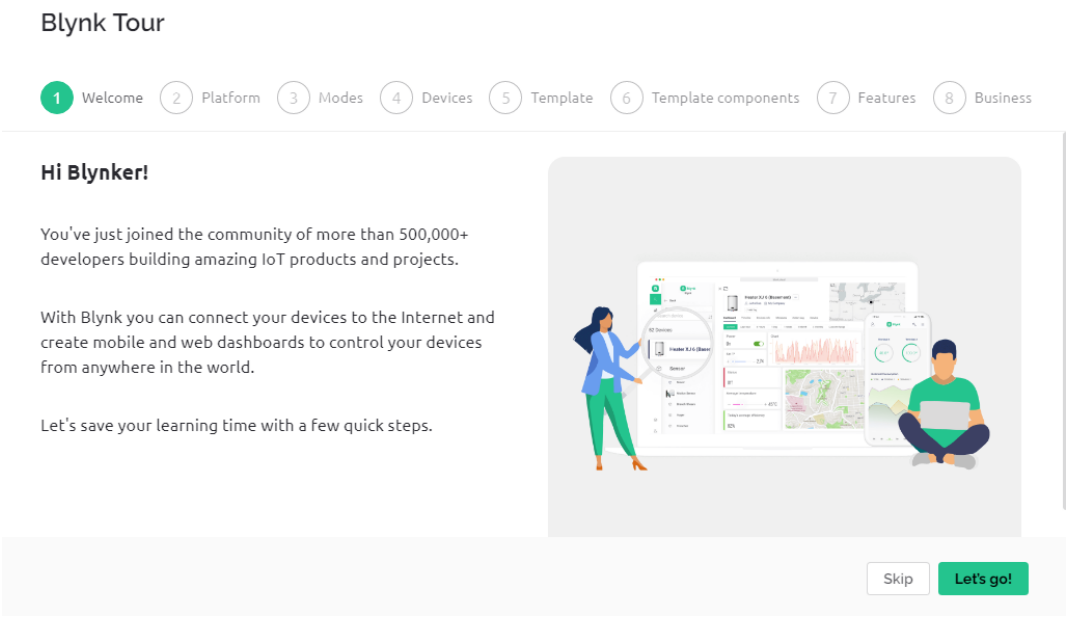
2. Geben Sie Ihre E-Mail-Adresse ein, um ein Konto zu registrieren.



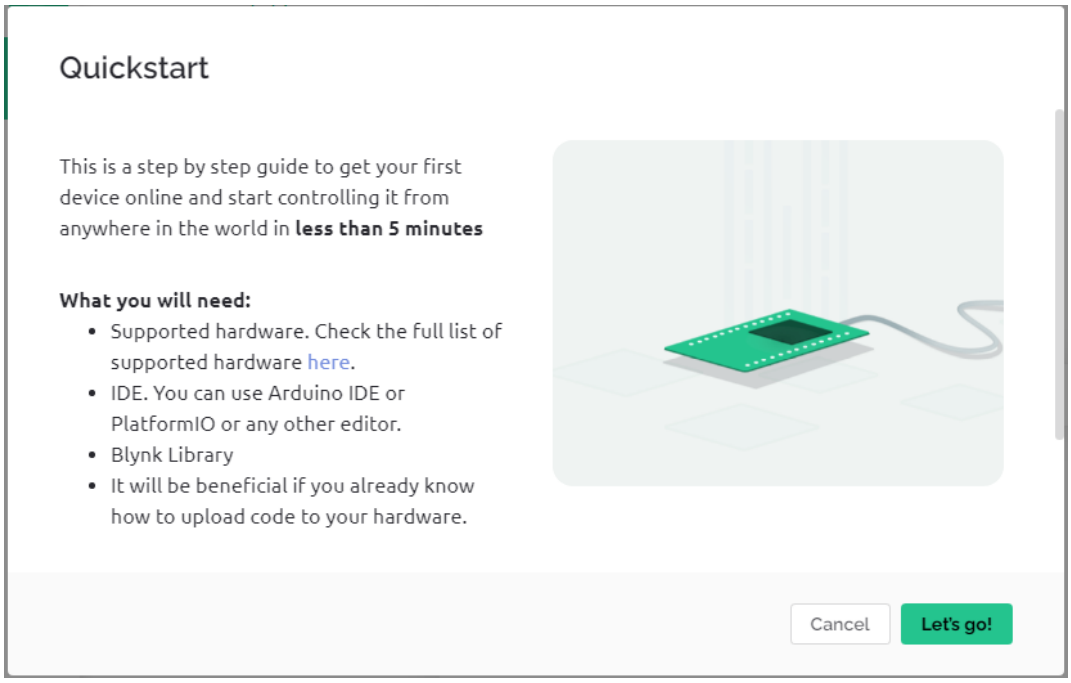
3. Öffnen Sie Ihre E-Mail, um die Registrierung abzuschließen.



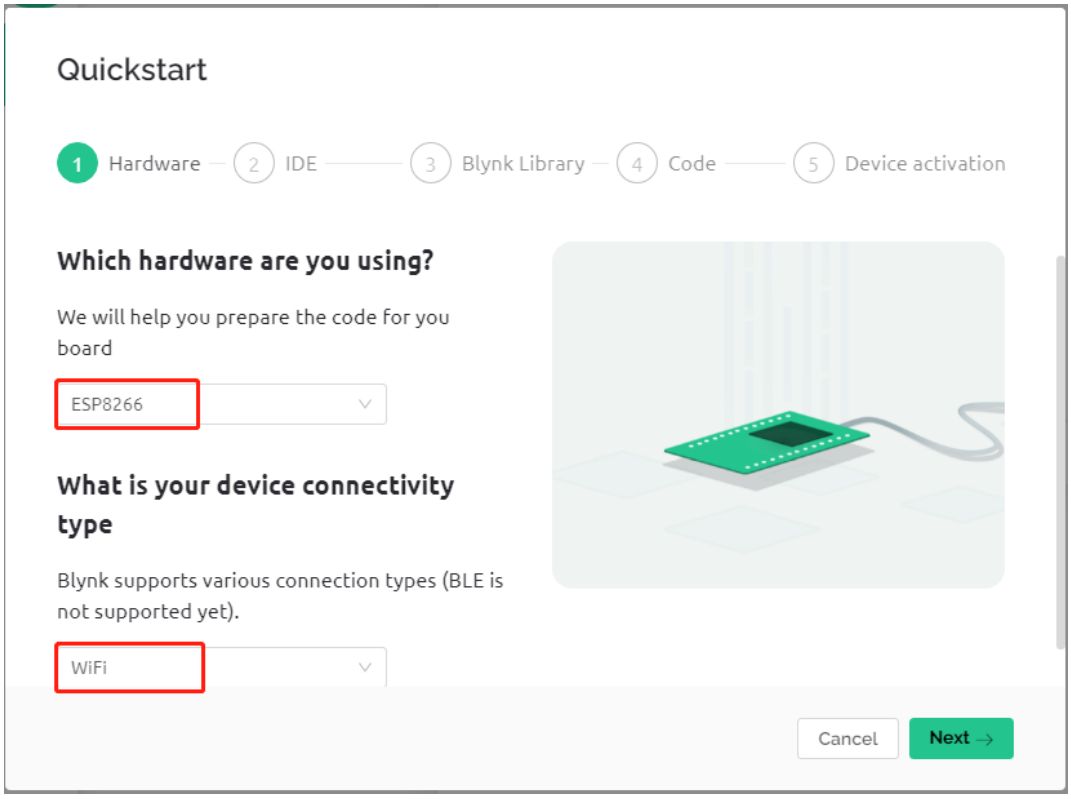
4. Anschließend erscheint die **Blynk Tour**, die Sie lesen können, um grundlegende Informationen über Blynk zu erhalten.



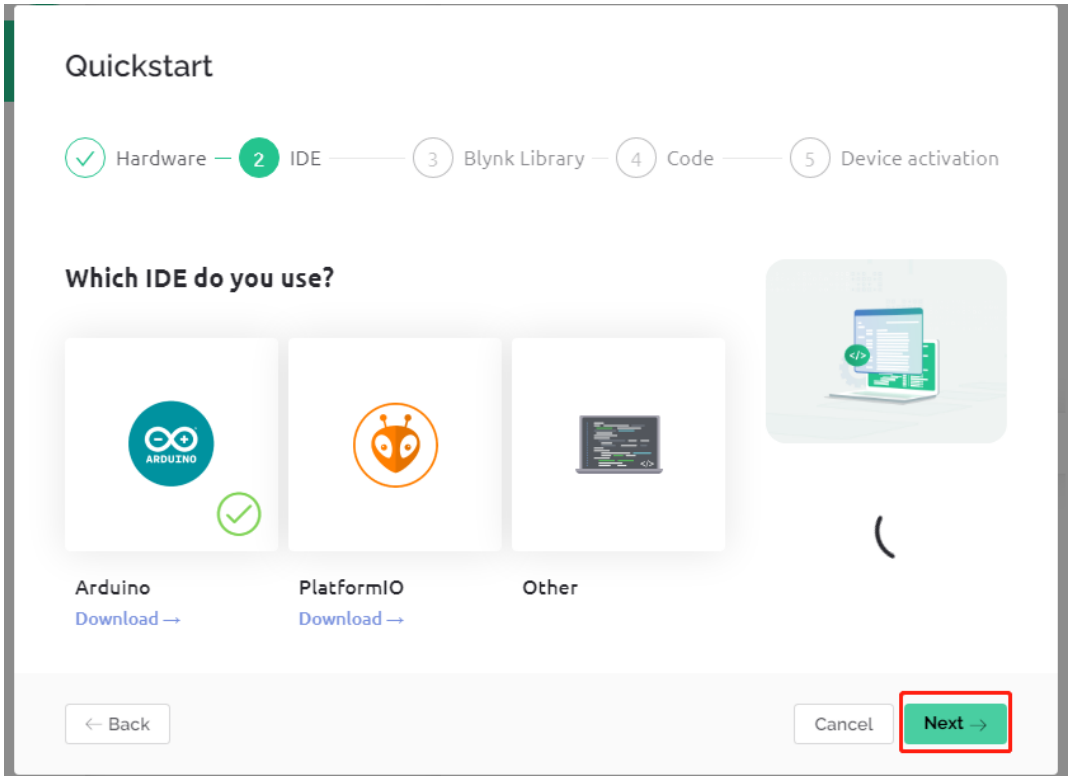
5. Als nächstes müssen wir mit diesem **Quick Start** eine Vorlage und ein Gerät erstellen. Klicken Sie auf **Let's go**.



6. Wählen Sie die Hardware und den Verbindungstyp aus.

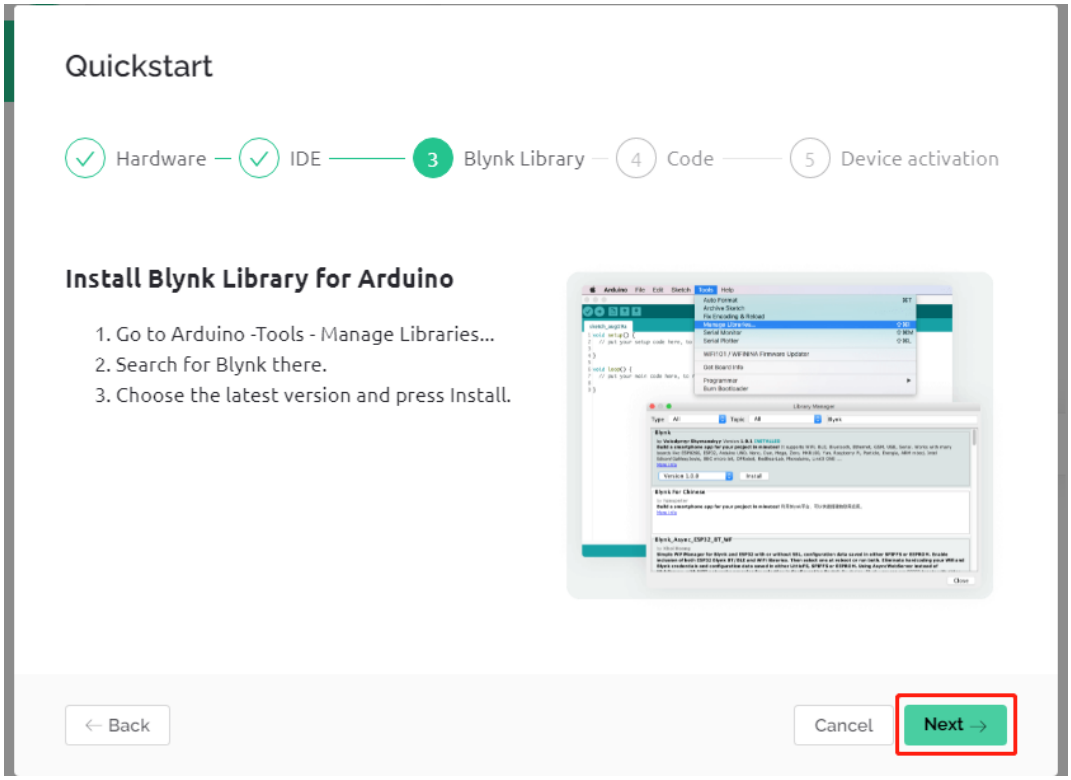


7. Hier wird Ihnen mitgeteilt, welche IDE Sie benötigen. Wir empfehlen die **Arduino IDE**.

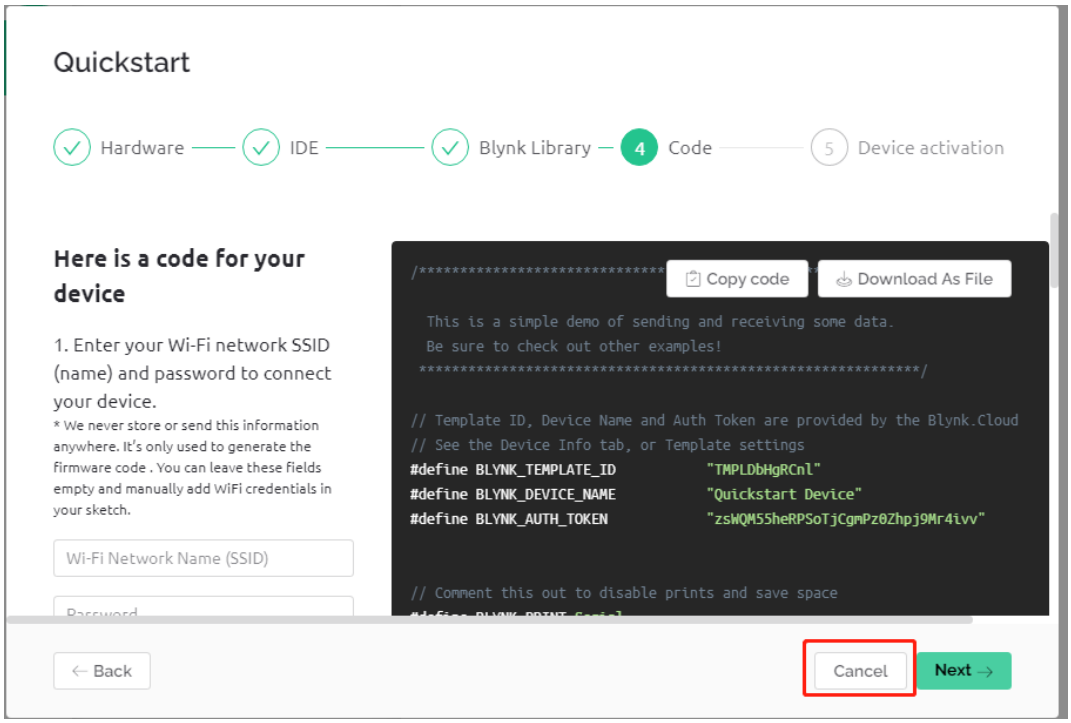


8. Hier ist die Bibliothek, die Sie hinzufügen müssen. Die hier empfohlene Bibliothek ist jedoch etwas problematisch. Wir müssen andere Bibliotheken manuell hinzufügen (darauf kommen wir später zurück). Klicken Sie hier

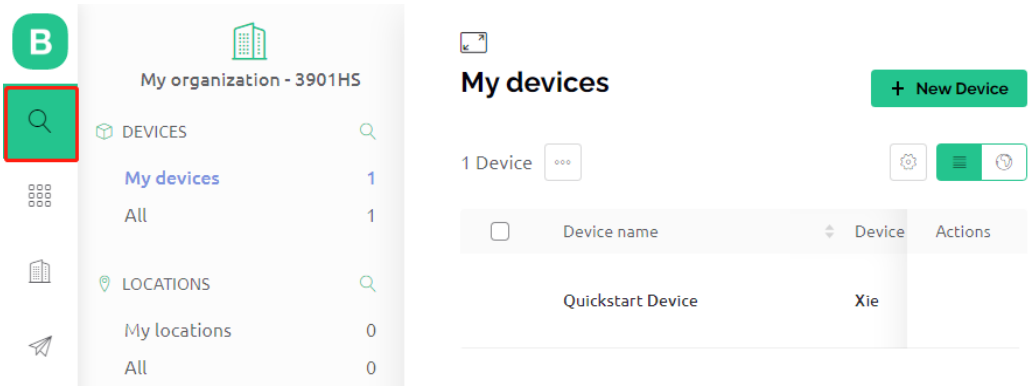
auf **Next**, und eine neue Vorlage und ein neues Gerät werden erstellt.



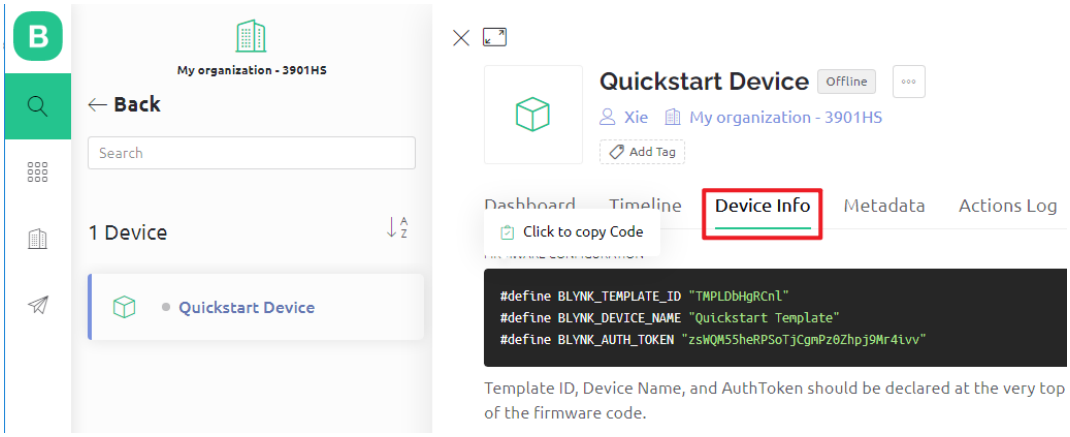
9. Die nächsten Schritte bestehen darin, den entsprechenden Code hochzuladen und Ihr Board mit Blynk zu verbinden. Da es jedoch ein Problem mit der zuvor bereitgestellten Bibliothek gibt, müssen Sie weitere Bibliotheken hinzufügen. Klicken Sie also hier auf **Cancel**, um **Quick Start** zu beenden.



10. Klicken Sie auf die **Search**-Schaltfläche, und Sie sehen das neu erstellte Gerät.



11. Gehen Sie zu diesem **Quickstart Device** und klicken Sie auf **Device Info**. Auf der **Device info**-Seite sehen Sie `TEMPLATE_ID`, `DEVICE_NAME` und `AUTH_TOKEN`, die Sie später kopieren müssen.

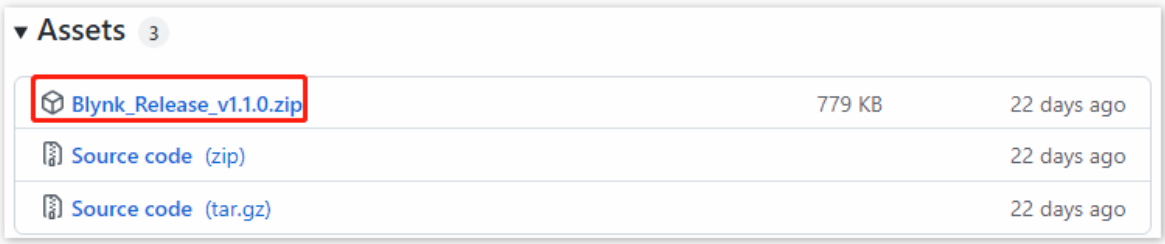


1.3 Hinzufügen der erforderlichen Bibliotheken

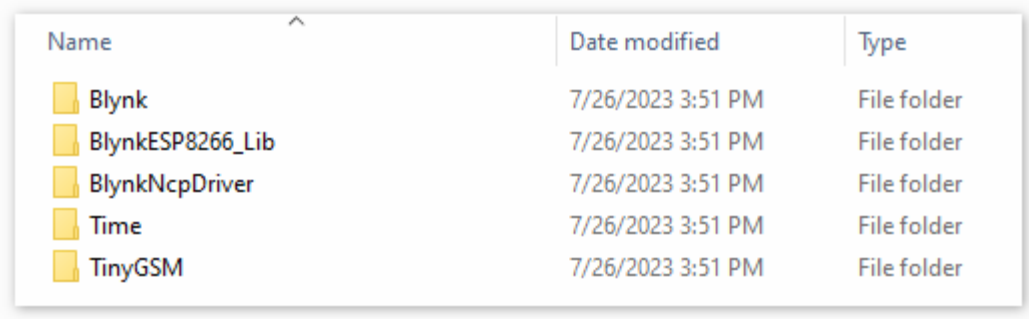
Für die Verwendung von Blynk im Arduino IDE müssen die richtigen Bibliotheken hinzugefügt werden.

1. Klicken Sie auf , scrollen Sie nach unten zu **„Assets“** und laden Sie die erste `.zip`-Datei herunter.

Bemerkung: Bitte beachten Sie, dass die in der untenstehenden Abbildung angezeigte Versionsnummer veraltet sein könnte. Wir empfehlen dringend, die neueste verfügbare Version herunterzuladen und zu installieren.

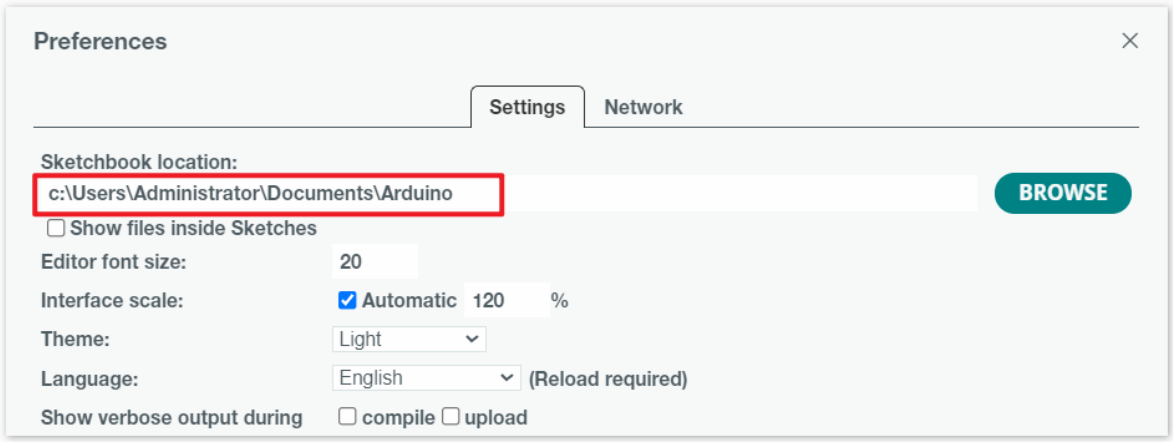


2. Entpacken Sie diese Datei und gehen Sie in den Ordner `libraries`, um die folgenden Ordner zu sehen.

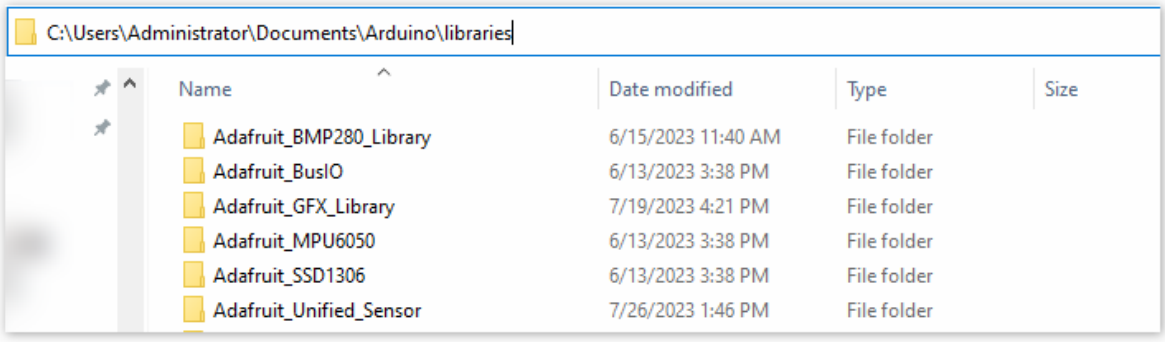


3. Kopieren Sie alle und fügen Sie sie in den Ordner `libraries` Ihres Sketchbooks ein.

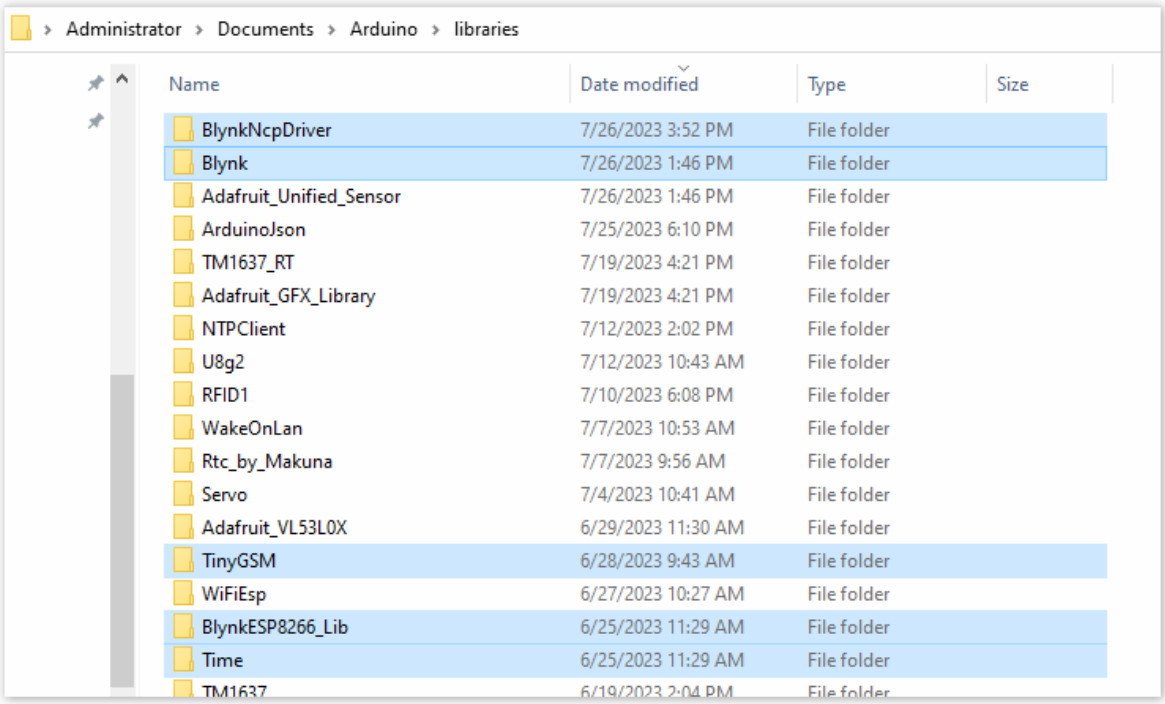
   Schritt 1: Sie können den Speicherort Ihres Bibliotheksordners unter `Datei` > `Einstellungen` > `Sketchbook-Speicherort` finden oder ändern.



   Schritt 2: Gehen Sie zum Speicherort Ihres Sketchbooks (aus der Arduino IDE auffindbar). Finden Sie den Ordner `libraries` und klicken Sie darauf, um ihn zu öffnen.



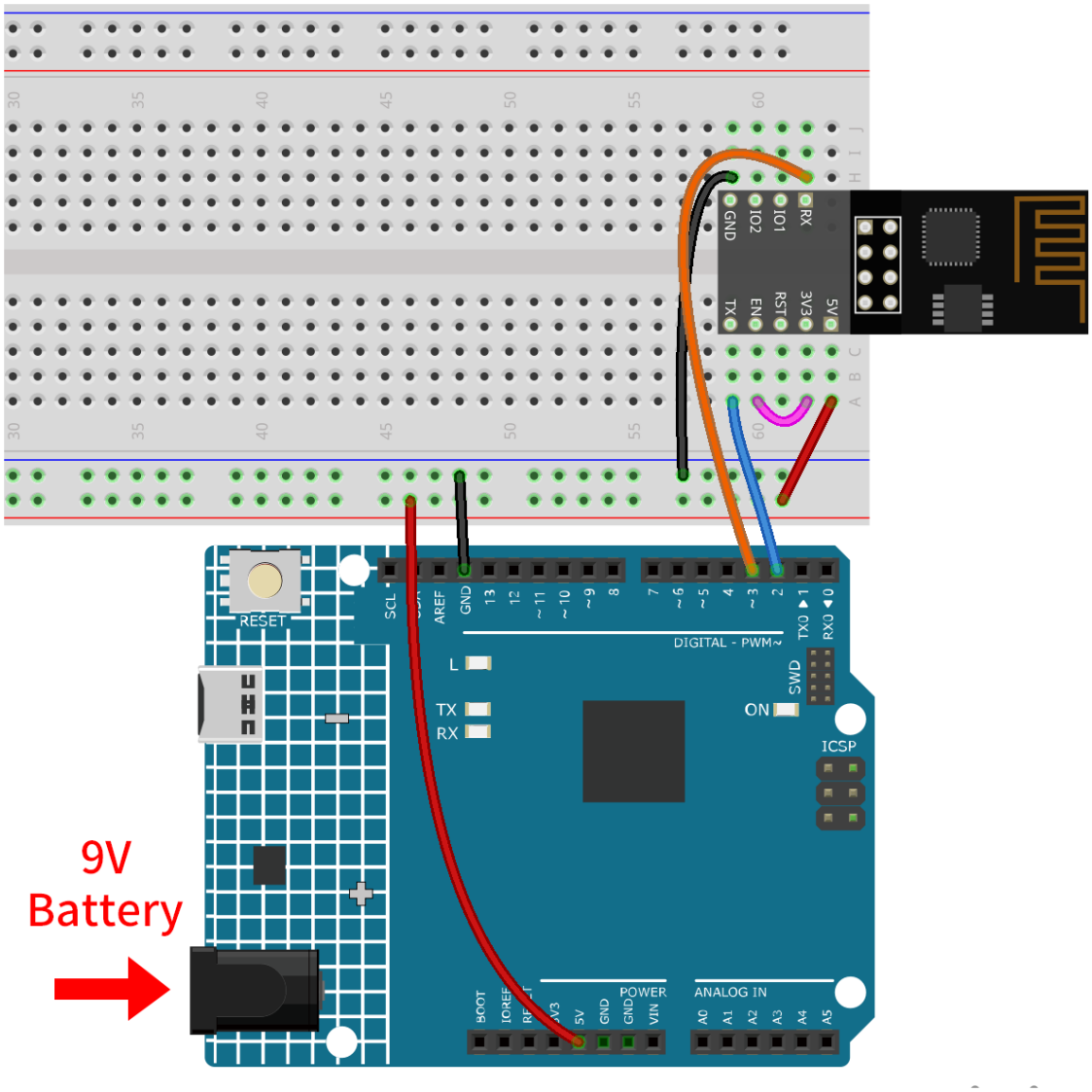
   Schritt 3: Fügen Sie alle entpackten Ordner von `Blynk_Release_vx.x.x\libraries` in den Bibliotheksordner ein.



1.4 Verbindung des R4-Boards mit Blynk herstellen

1. Verbinden Sie das ESP8266-Modul und das R4-Board erneut. Hier wird die Software-Schnittstelle verwendet, sodass TX und RX jeweils mit den Pins 2 und 3 des R4-Boards verbunden sind.

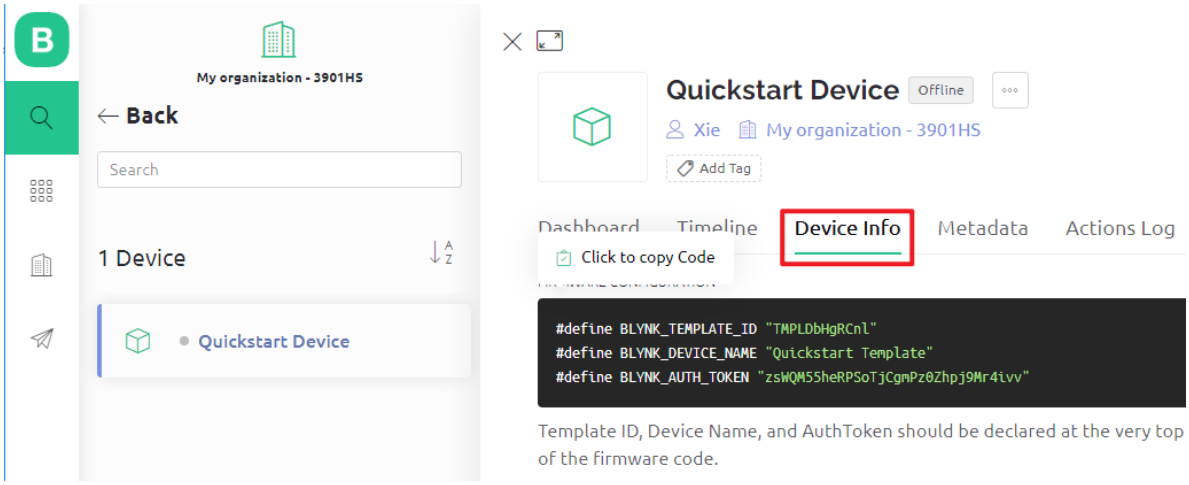
Bemerkung: Das ESP8266-Modul benötigt einen hohen Strom, um eine stabile Betriebsumgebung zu gewährleisten. Stellen Sie daher sicher, dass die 9V-Batterie angeschlossen ist.



1. Öffnen Sie die Datei `00-Blynk_quick_start.ino` unter dem Pfad `ultimate-sensor-kit\iot_project\wifi\00-Blynk_quick_start`. Oder kopieren Sie diesen Code in die **Arduino IDE**.

2. Ersetzen Sie die folgenden drei Codezeilen, die Sie von der **Geräteinfo**-Seite Ihres Kontos kopieren können. Diese drei Zeilen ermöglichen es dem R4-Board, Ihr Blynk-Konto zu finden.

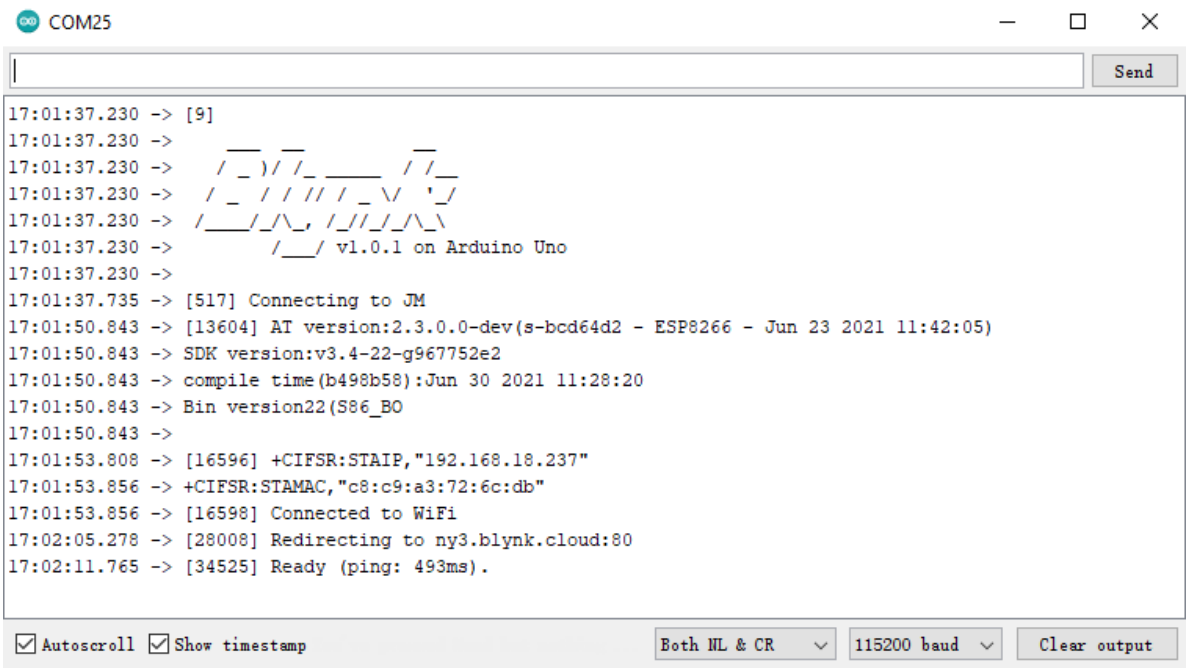
```
#define BLYNK_TEMPLATE_ID "TMPLxxxxxx"
#define BLYNK_DEVICE_NAME "Device"
#define BLYNK_AUTH_TOKEN "YourAuthToken"
```



3. Geben Sie die `ssid` und das `Passwort` des verwendeten WLANs ein.

```
char ssid[] = "ssid";
char pass[] = "password";
```

4. Laden Sie den Code auf das R4-Board, öffnen Sie dann den seriellen Monitor und stellen Sie die Baudrate auf 115200 ein. Wenn das R4-Board erfolgreich mit Blynk kommuniziert, wird im seriellen Monitor das Zeichen `ready` angezeigt.

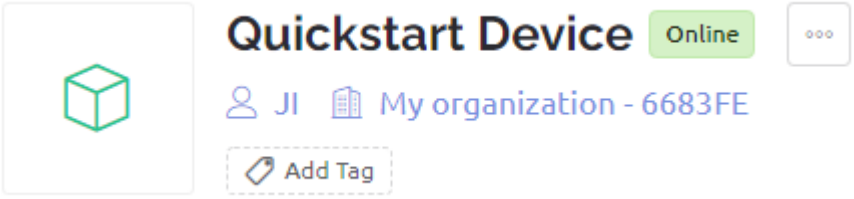


Bemerkung: Sollte die Meldung `ESP is not responding` erscheinen, befolgen Sie bitte diese Schritte:

- Stellen Sie sicher, dass die 9V-Batterie angeschlossen ist.
- Setzen Sie das ESP8266-Modul zurück, indem Sie den Pin RST für 1 Sekunde auf GND legen und dann wieder entfernen.
- Drücken Sie die Reset-Taste auf dem R4-Board.

Manchmal müssen Sie die obigen Schritte 3-5 Mal wiederholen. Bitte haben Sie Geduld.

5. Der Status von Blynk ändert sich von **offline** auf **online**.

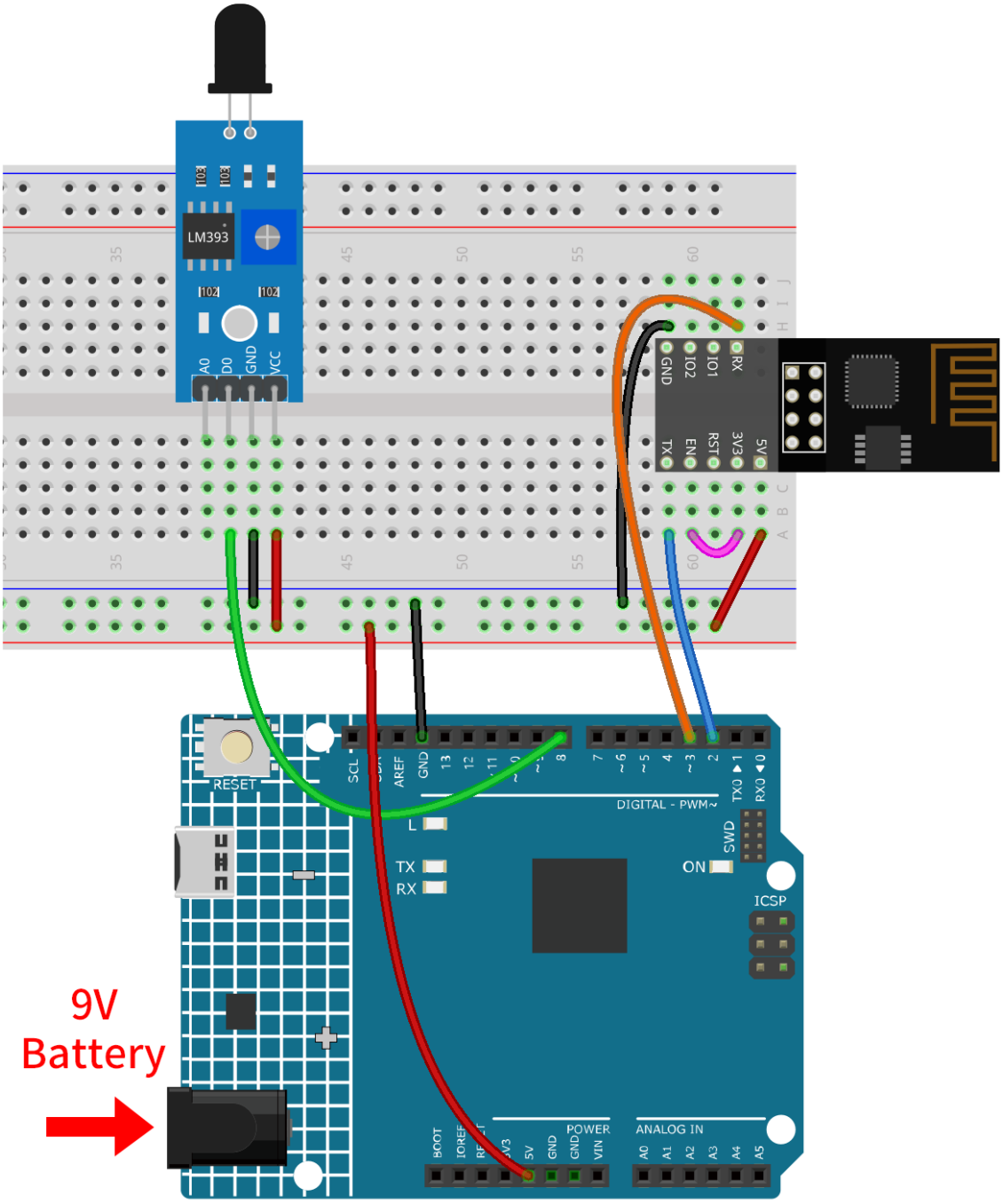


2.5.2 Flammenwarnsystem mit Blynk

In diesem Kapitel zeigen wir Ihnen, wie Sie mit Blynk ein Flammenalarmsystem für Ihr Zuhause einrichten können. Mit einem Flammensensor können Sie potenzielle Brände in Ihren vier Wänden frühzeitig erkennen. Über Blynk können Sie die erfassten Daten im Internet überwachen. Im Falle eines Feuers erhalten Sie umgehend eine E-Mail-Benachrichtigung.

1. Schaltung aufbauen

Bemerkung: Das ESP8266-Modul benötigt eine hohe Stromstärke für einen stabilen Betrieb. Achten Sie daher darauf, dass die 9V-Batterie angeschlossen ist.

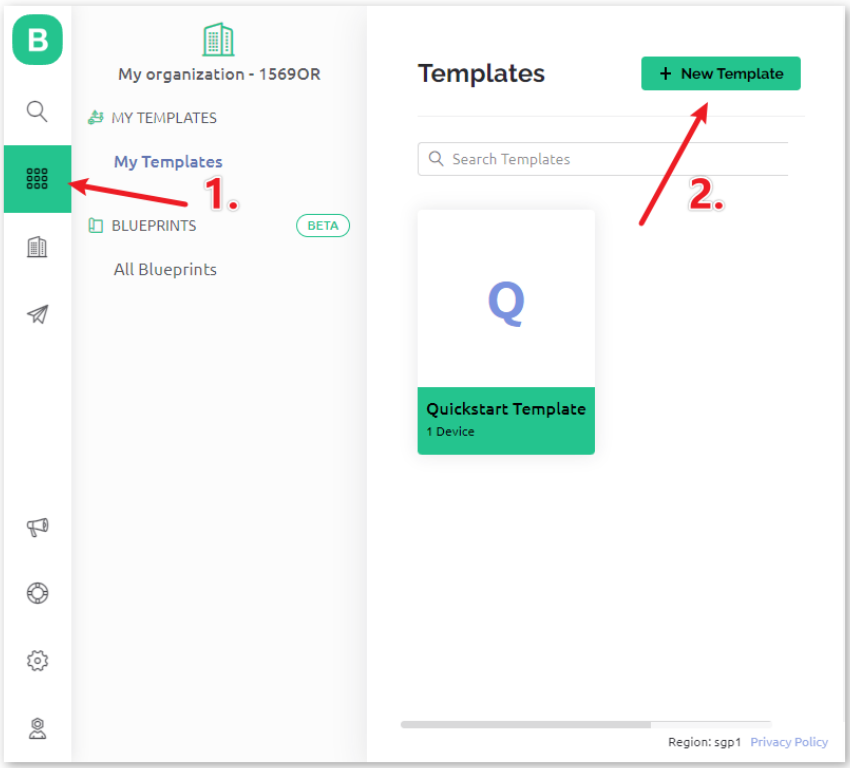


- *Arduino UNO R4 Minima-Platine*
- *Flammen-Sensormodul*
- *ESP8266-Modul*

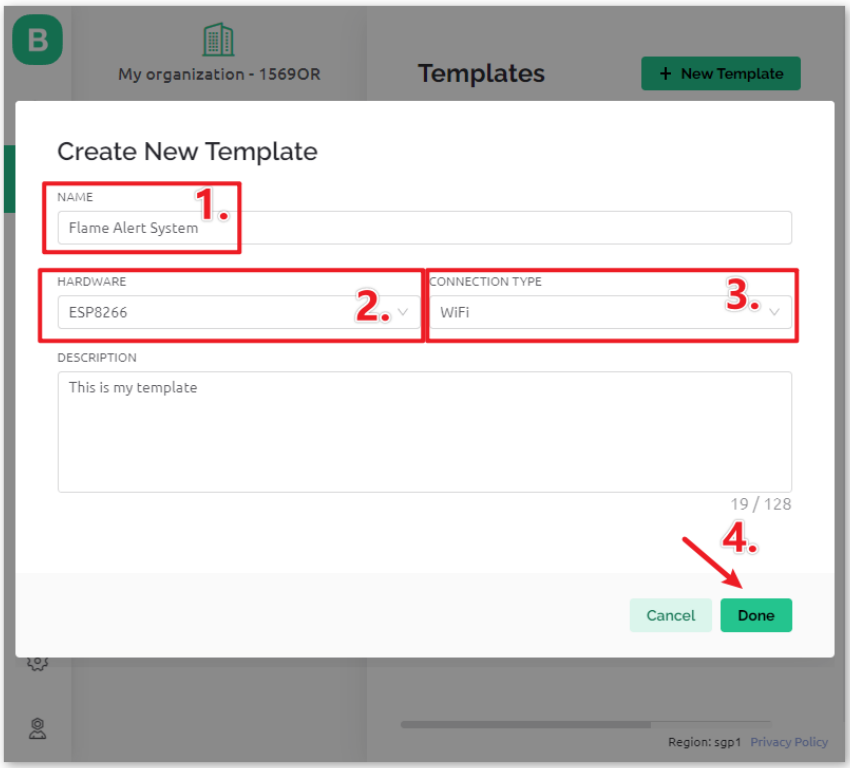
2. Blynk konfigurieren

2.1 Vorlage erstellen

Zuerst erstellen wir in Blynk eine Vorlage für das **„Flame Alert System“**.

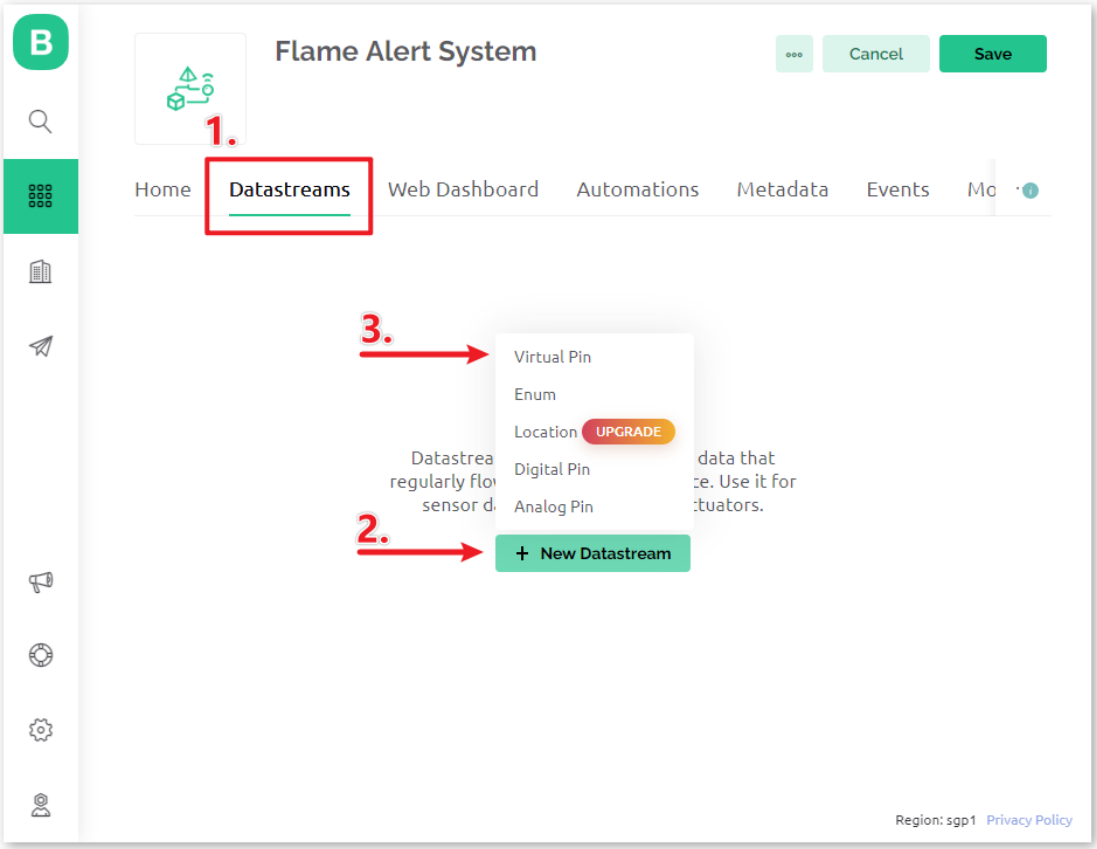


Achten Sie darauf, dass bei **HARDWARE ESP8266** und bei **CONNECT TYPE WiFi** eingestellt ist.

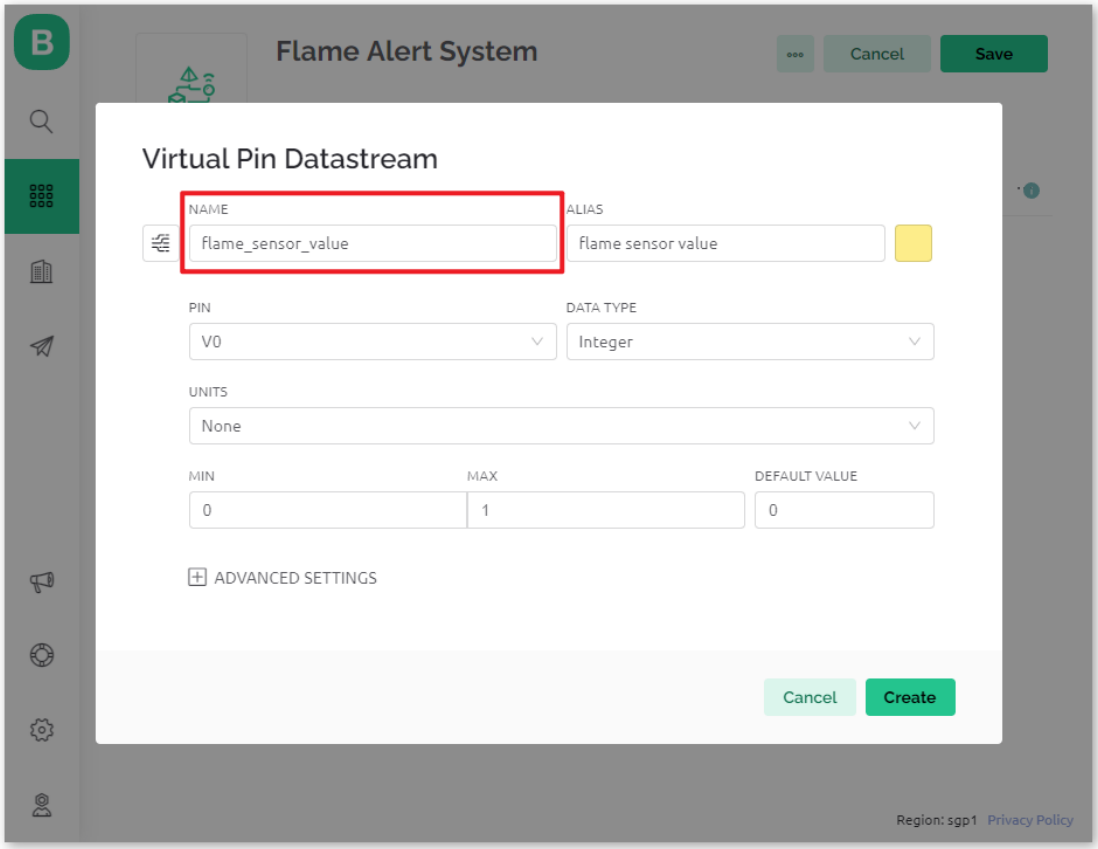


2.2 Datenstrom

Erstellen Sie im Bereich **Datastream** einen **Datastream** des Typs **Virtual Pin**, um den Wert des Flammensensors zu erfassen.

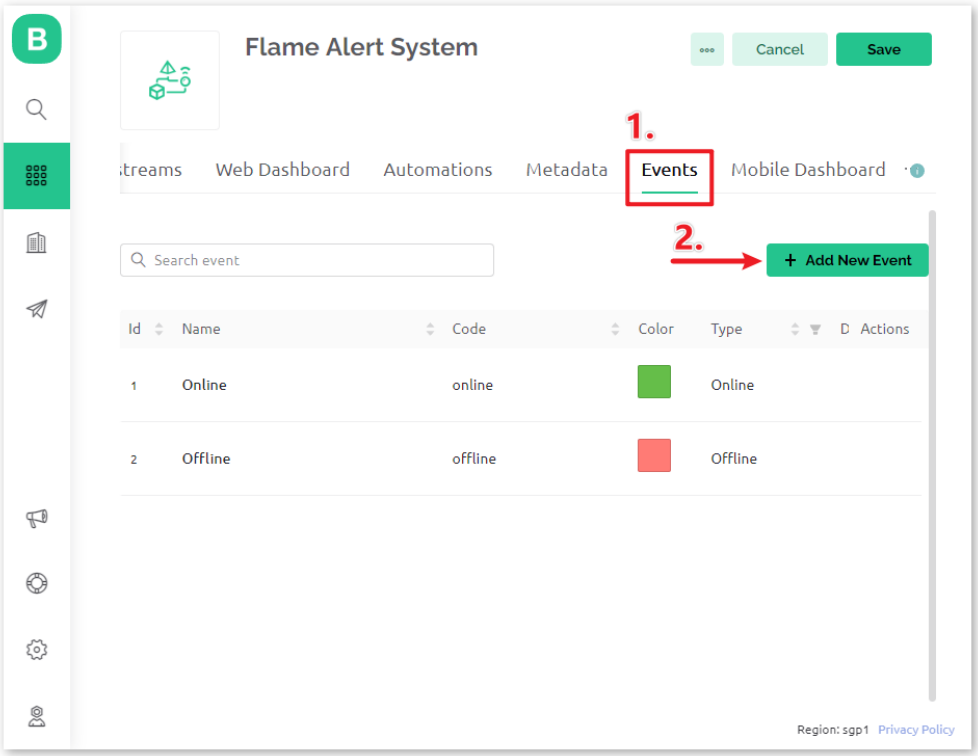


Geben Sie dem **Virtual Pin** den Namen `flame_sensor_value`. Setzen Sie den **DATA TYPE** auf **Integer** und MIN und MAX auf **0** und **1**.



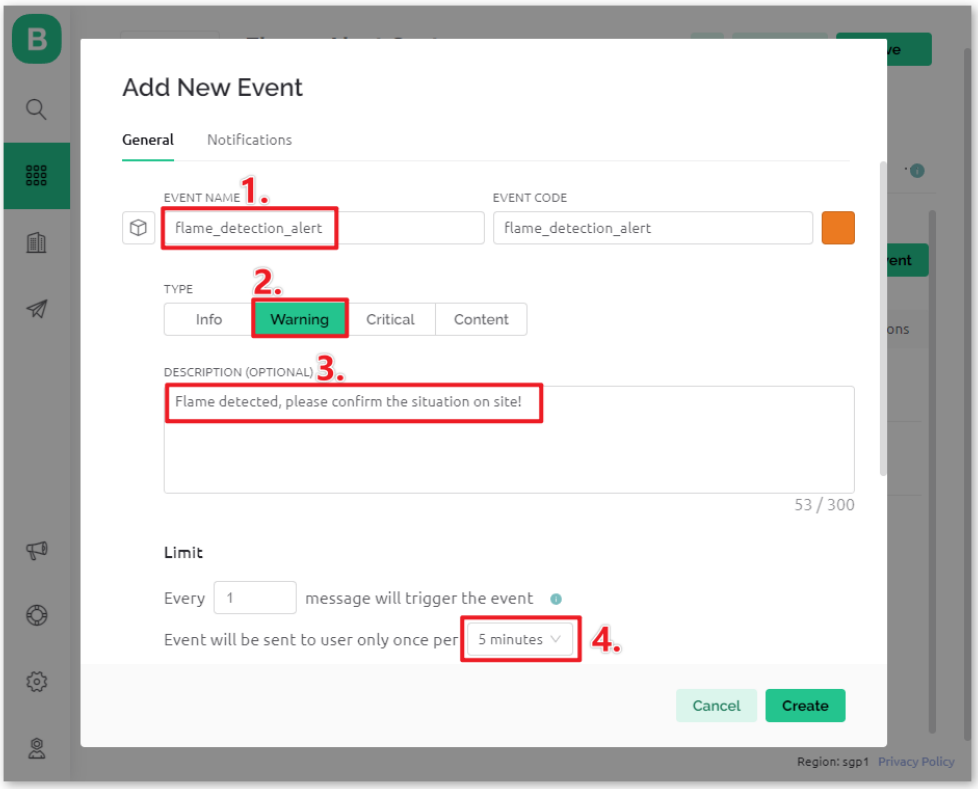
2.3 Ereignis

Nun legen wir ein **event** an, das bei Flammenerkennung protokolliert und eine E-Mail-Benachrichtigung verschickt.

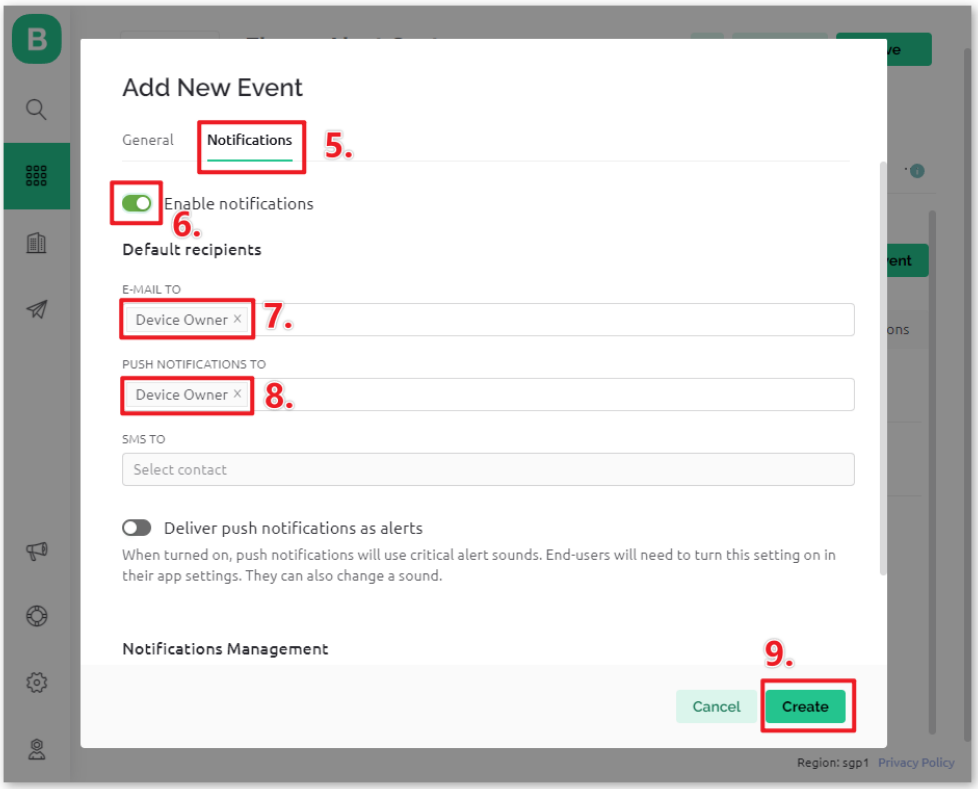


Bemerkung: Es wird empfohlen, die Einstellungen beizubehalten, um den Code ohne weitere Anpassungen verwenden zu können.

Legen Sie den **EVENT NAME** auf `flame_detection_alert` fest. Sie können den Inhalt der versendeten E-Mail anpassen, indem Sie eine **DESCRIPTION** für die Auslösung des Ereignisses festlegen. Darunter können Sie auch die Häufigkeitsbeschränkungen für die Ereignisauslösung einstellen.



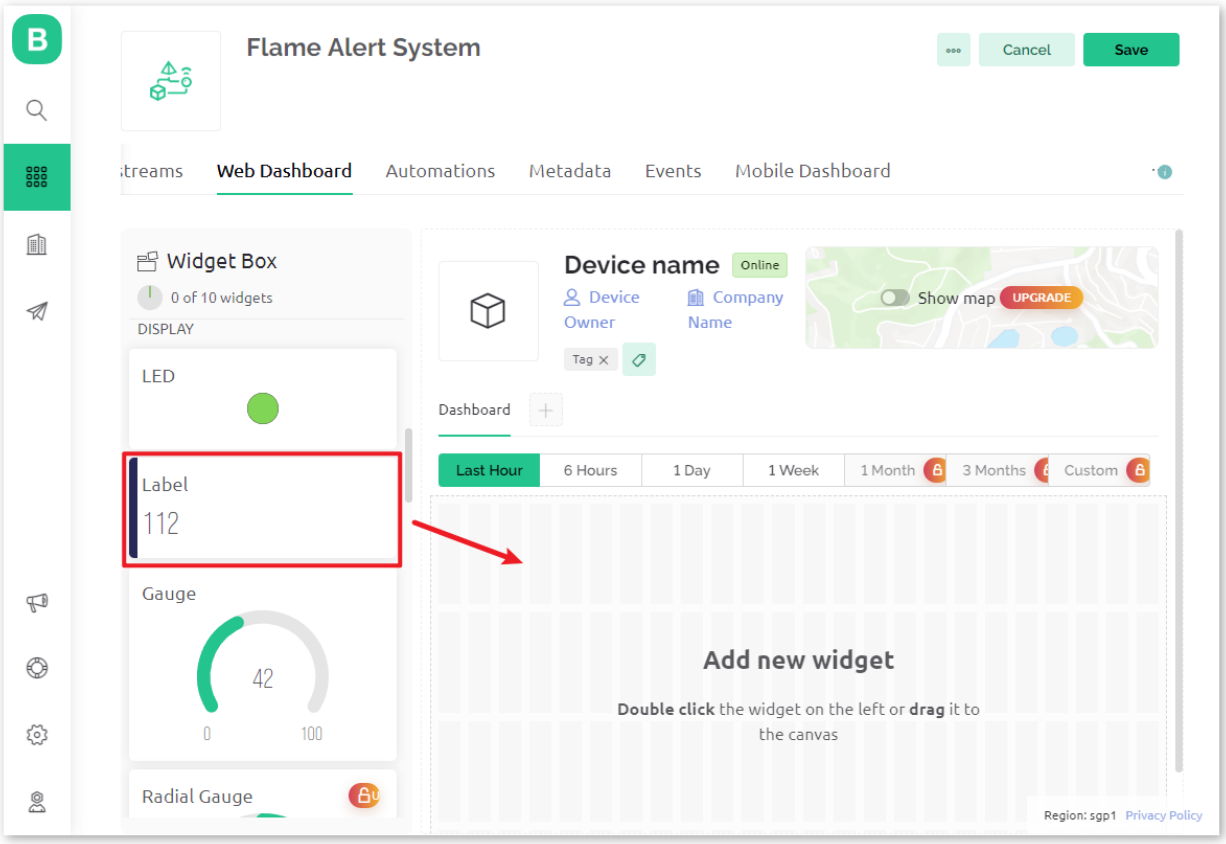
Navigieren Sie zur **Notifications**-Seite und konfigurieren Sie die E-Mail-Einstellungen.



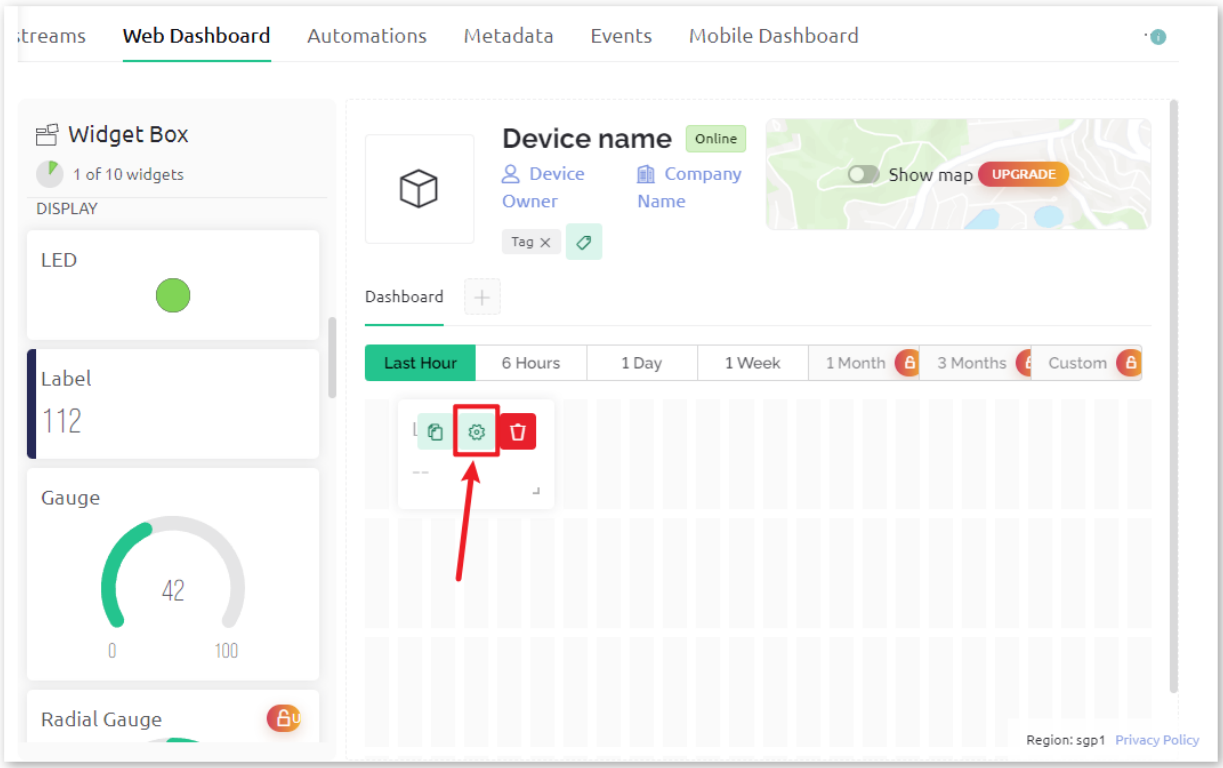
2.4 Web-Dashboard

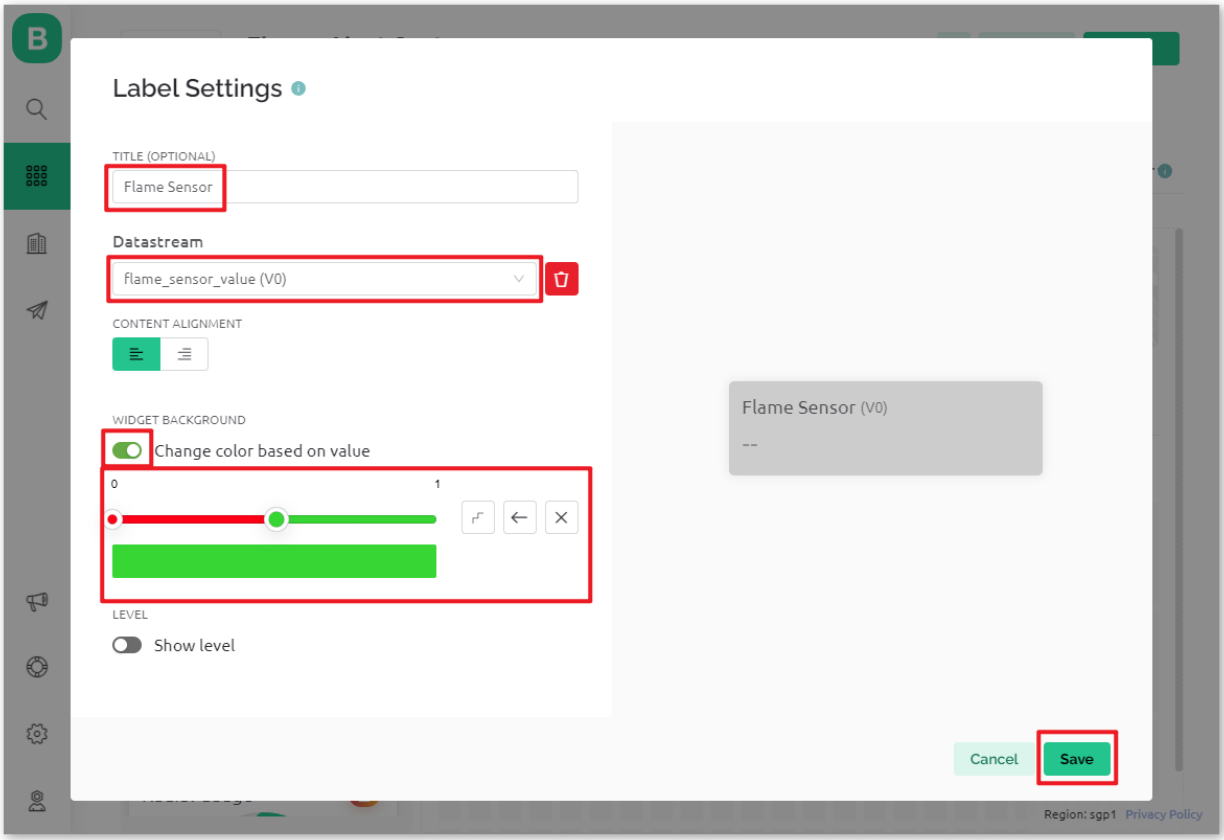
Wir richten auch das **Web-Dashboard** ein, um die vom Uno-Board übermittelten Sensordaten darzustellen.

Fügen Sie auf der **Web-Dashboard**-Seite ein **Label-Widget** hinzu.



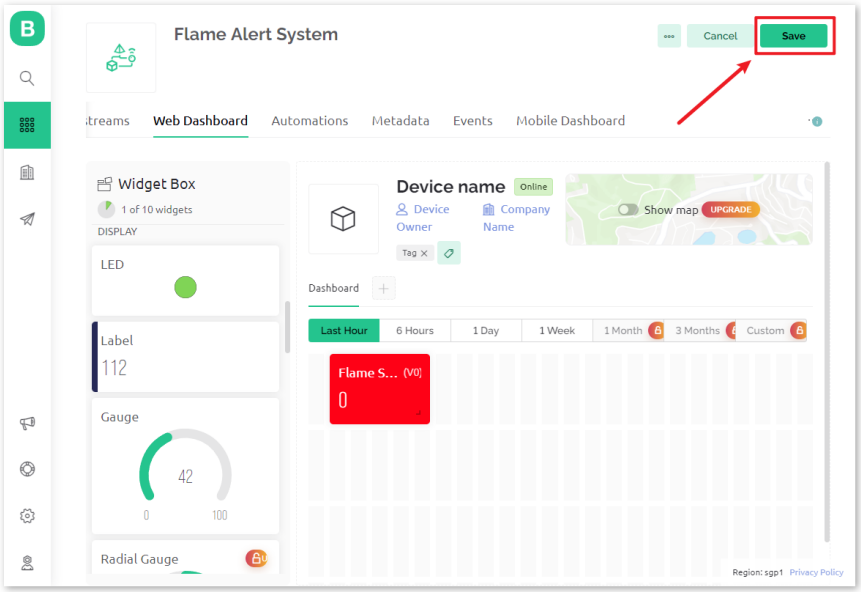
In den Einstellungen des **Label-Widgets** wählen Sie als **Datenstrom flame_sensor_value(V0)** aus. Legen Sie dann die Farbe des **WIDGET BACKGROUND** so fest, dass sie sich mit dem Datenwert ändert. Bei einem Wert von 1 wird der Hintergrund grün, bei einem Wert von 0 rot dargestellt.


Streams
Web Dashboard
Automations
Metadata
Events
Mobile Dashboard
Widget Box
1 of 10 widgets
DISPLAY
LED
Label
112
Gauge
42
0
100
Radial Gauge
Device name
Online
Device Owner
Company Name
Tag
Show map
UPGRADE
Dashboard
Last Hour
6 Hours
1 Day
1 Week
1 Month
3 Months
Custom
Region: sgp1
Privacy Policy

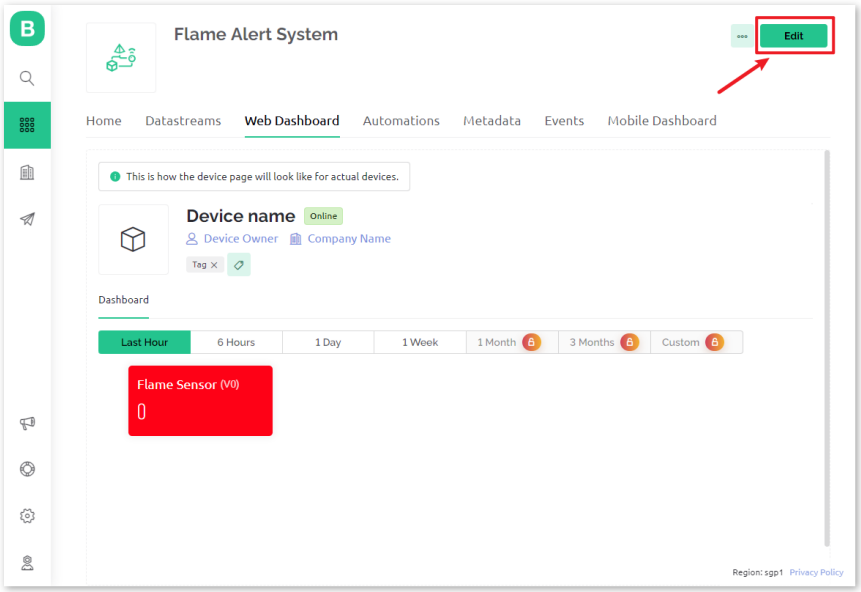

Label Settings
TITLE (OPTIONAL)
Flame Sensor
Datastream
flame_sensor_value (V0)
CONTENT ALIGNMENT
WIDGET BACKGROUND
Change color based on value
0
1
LEVEL
Show level
Flame Sensor (V0)
--
Cancel
Save
Region: sgp1
Privacy Policy

2.5 Vorlage speichern

Vergessen Sie nicht, die Vorlage zum Schluss zu speichern.



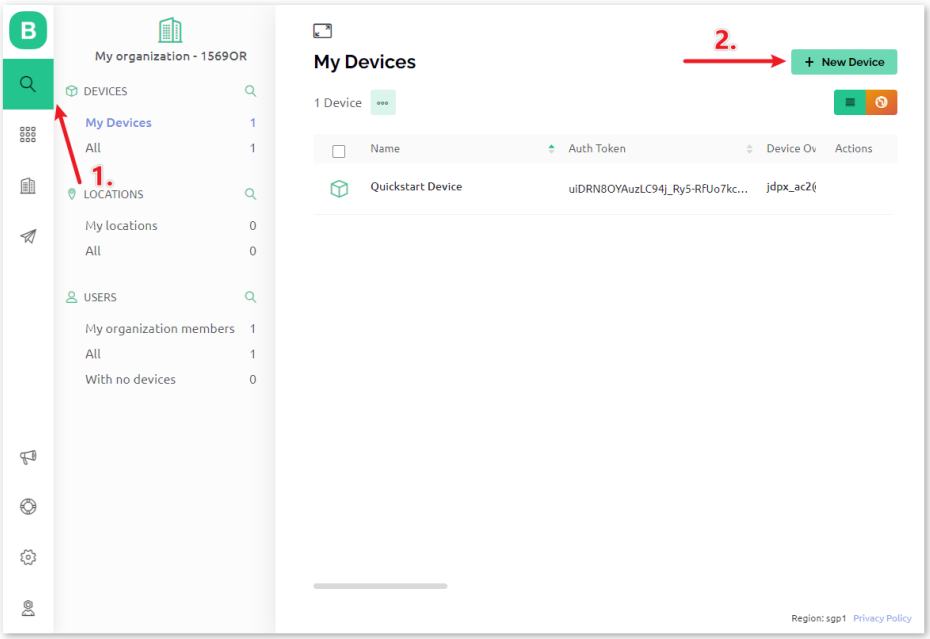
Falls Sie die Vorlage nachträglich bearbeiten müssen, können Sie oben rechts auf das Bearbeitungssymbol klicken.

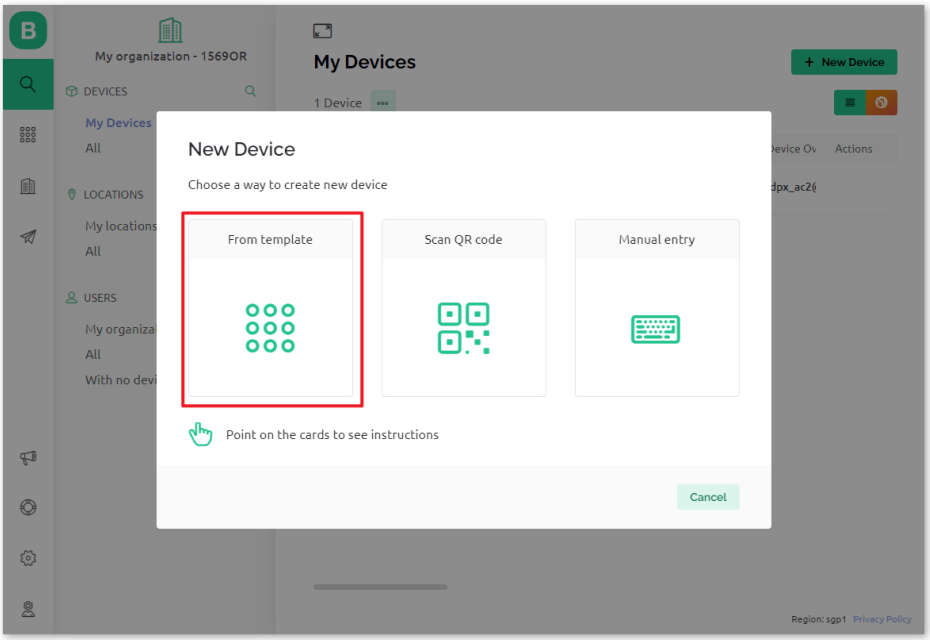


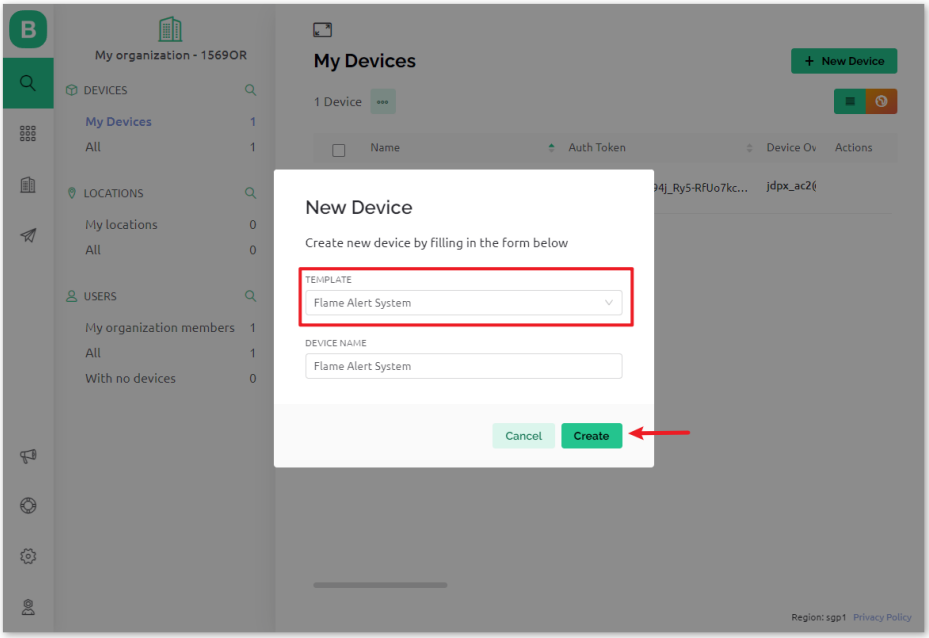
3. Code ausführen

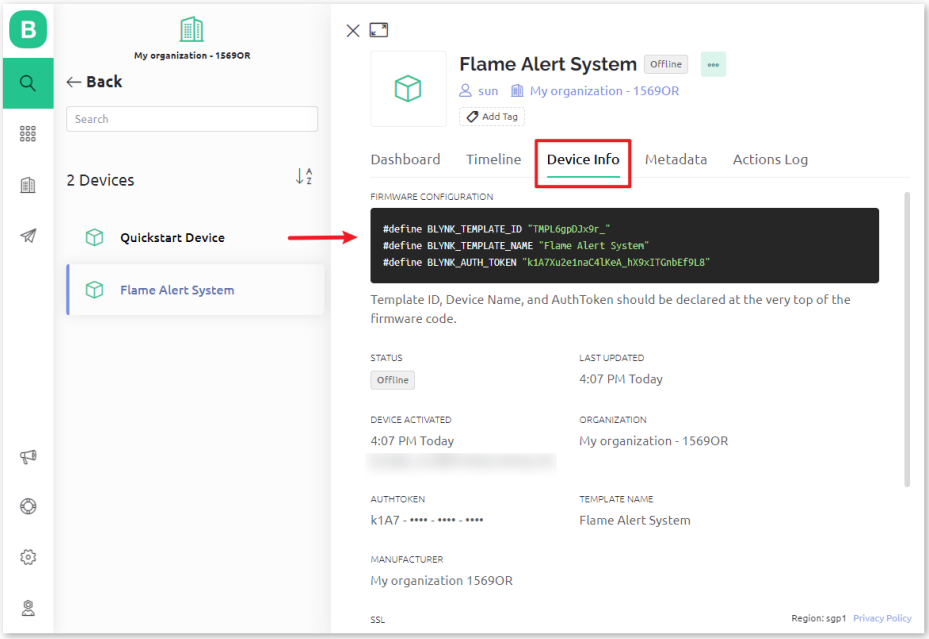
1. Öffnen Sie die Datei `01-Flame_alert_system.ino` im Verzeichnis `ultimate-sensor-kit\iot_project\wifi\01-Flame_alert_system`, oder kopieren Sie diesen Code in die **Arduino IDE**.
2. Legen Sie ein Blynk-Gerät mit der Vorlage für die Flammenerkennung an. Anschließend ersetzen Sie `BLYNK_TEMPLATE_ID`, `BLYNK_TEMPLATE_NAME` und `BLYNK_AUTH_TOKEN` durch Ihre eigenen Angaben.

```
#define BLYNK_TEMPLATE_ID "TMPxxxxxxx"
#define BLYNK_TEMPLATE_NAME "Flame Alert System"
#define BLYNK_AUTH_TOKEN "xxxxxxxxxxxxx"
```





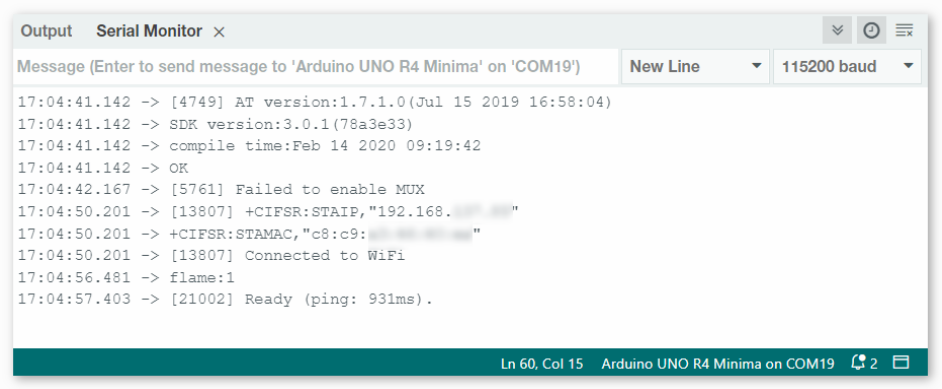




3. Geben Sie auch die `SSID` und das `Passwort` Ihres WLANs ein.

```
char ssid[] = "your_ssid";
char pass[] = "your_password";
```

4. Wählen Sie das korrekte Board und den Port aus und klicken Sie auf die Schaltfläche **Hochladen**.
5. Öffnen Sie den seriellen Monitor (Baudrate auf 115200 einstellen) und warten Sie auf eine erfolgreiche Verbindungsmeldung.

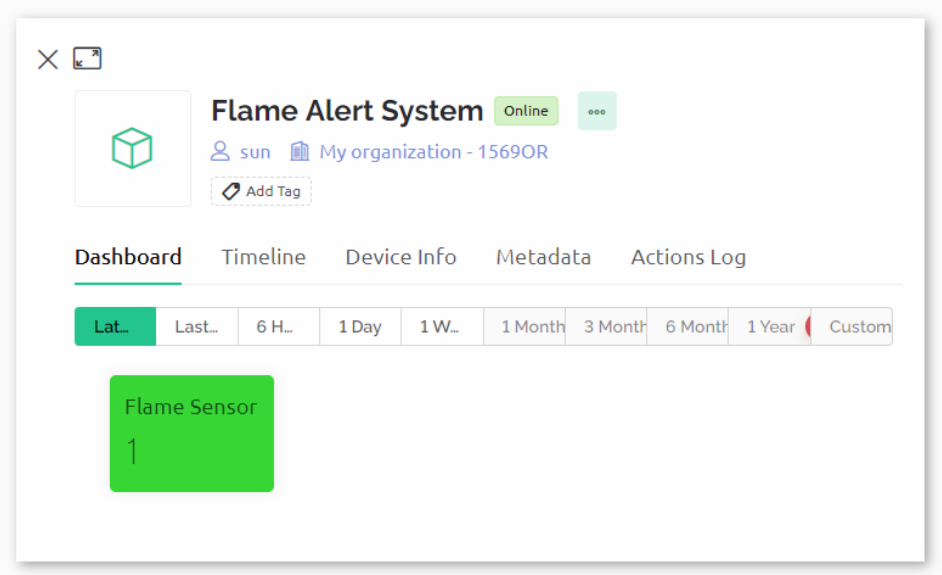


Bemerkung: Falls die Meldung `ESP is not responding` erscheint, befolgen Sie bitte diese Schritte:

- Stellen Sie sicher, dass die 9V-Batterie angeschlossen ist.
- Setzen Sie das ESP8266-Modul zurück, indem Sie den Pin RST für 1 Sekunde mit GND verbinden und dann wieder trennen.
- Drücken Sie die Reset-Taste auf dem R4-Board.

Manchmal müssen Sie diese Schritte 3-5 Mal wiederholen. Bitte haben Sie Geduld.

6. Blynk zeigt jetzt die vom Flammensensor gelesenen Daten an. Im Label-Widget sehen Sie den vom Flammensensor gelesenen Wert. Bei einem angezeigten Wert von 1 wird der Hintergrund des Labels grün, bei einem Wert von 0 rot angezeigt, und Blynk sendet Ihnen eine Warn-E-Mail.



7. Wenn Sie Blynk auf mobilen Geräten verwenden möchten, beachten Sie bitte *Wie verwendet man Blynk auf dem Mobilgerät?*.

4. Code-Erklärung

1. **Bibliotheksinitialisierung**

   Bevor wir beginnen, ist es wichtig, die erforderlichen Bibliotheken und Einstellungen für die Kommunikation zwischen Arduino, dem ESP8266-WLAN-Modul und der Blynk-App einzurichten. Dieser Code setzt die benötigten Bibliotheken auf und konfiguriert eine Software-Serielle Verbindung zwischen Arduino und ESP8266-Modul mit der passenden Baudrate für die Datenübertragung.

```
//Set debug prints on Serial Monitor
#define BLYNK_PRINT Serial

#include <ESP8266_Lib.h>                // Library for ESP8266
#include <BlynkSimpleShieldEsp8266.h>  // Library for Blynk

// Software Serial on Uno
#include <SoftwareSerial.h>
SoftwareSerial EspSerial(2, 3);  // RX, TX
#define ESP8266_BAUD 115200      // Set the ESP8266 baud rate
ESP8266 wifi(&EspSerial);
```

2. **Blynk- und WLAN-Konfiguration**

   Damit das Projekt mit der Blynk-App kommunizieren kann, muss es sich mit einem WLAN-Netzwerk verbinden. Die Anmeldeinformationen werden hier angegeben.

```
// Template ID, Device Name and Auth Token are provided by the Blynk Cloud
// See the Device Info tab, or Template settings
#define BLYNK_TEMPLATE_ID "TMPxxxxxx"
#define BLYNK_TEMPLATE_NAME "Flame Alert System"
#define BLYNK_AUTH_TOKEN "xxxxxxxxxxxxxxx"

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "your_ssid";
char pass[] = "your_password";
```

3. **Sensor-Pin & Timer-Deklaration**

   Definieren Sie die Pin-Nummer für die Flamme. Die Blynk-Bibliothek bietet einen integrierten Timer, und wir erstellen ein Timer-Objekt. Bitte beachten Sie .

```
const int sensorPin = 8;
BlynkTimer timer;
```

4. **setup()-Funktion**

   Hier werden anfängliche Konfigurationen wie das Setzen des Pin-Modus für sensorPin, die Initiierung der seriellen Kommunikation, die Einstellung des BlynkTimers und die Verbindung zur Blynk-App durchgeführt.

   - Wir verwenden `timer.setInterval(1000L, myTimerEvent)` um das Timer-Intervall in setup() festzulegen. Hier setzen wir es so, dass die Funktion `myTimerEvent()` alle **1000 ms** ausgeführt wird. Den ersten Parameter von `timer.setInterval(1000L, myTimerEvent)` können Sie ändern, um das Intervall zwischen den Ausführungen von `myTimerEvent` zu ändern.

```
void setup() {
  pinMode(sensorPin, INPUT);
  Serial.begin(115200);
  EspSerial.begin(ESP8266_BAUD);
  delay(1000);
  timer.setInterval(1000L, myTimerEvent);
  Blynk.config(wifi, BLYNK_AUTH_TOKEN);
  Blynk.connectWiFi(ssid, pass);
}
```

5. **loop()-Funktion**

   Die Hauptloop führt die Blynk- und Timer-Dienste kontinuierlich aus.

```
void loop() {
  Blynk.run();
  timer.run();
}
```

6. **myTimerEvent() & sendData()-Funktion**

```
void myTimerEvent() {
  // Please don't send more that 10 values per second.
  sendData();  // Call function to send sensor data to Blynk app
}
```

   Die Funktion `sendData()` liest den Wert vom Flammensensor und sendet ihn an Blynk. Wenn eine Flamme erkannt wird (Wert 0), sendet sie ein `Flammenerkennungsalarm`-Ereignis an die Blynk-App.

   - Verwenden Sie `Blynk.virtualWrite(vPin, Wert)` um Daten an den virtuellen Pin V0 in Blynk zu senden. Mehr dazu unter .
   - Nutzen Sie `Blynk.logEvent("Ereigniscode")` um ein Ereignis in Blynk zu protokollieren. Mehr dazu unter .

```
void sendData() {
  int data = digitalRead(sensorPin);
  Blynk.virtualWrite(V0, data);  // send data to virtual pin V0 on Blynk
  Serial.print("flame:");
  Serial.println(data);  // Print flame status on Serial Monitor
  if (data == 0) {
    Blynk.logEvent("flame_alert");  // log flame alert event if sensor detects flame
  }
}
```

Referenzen

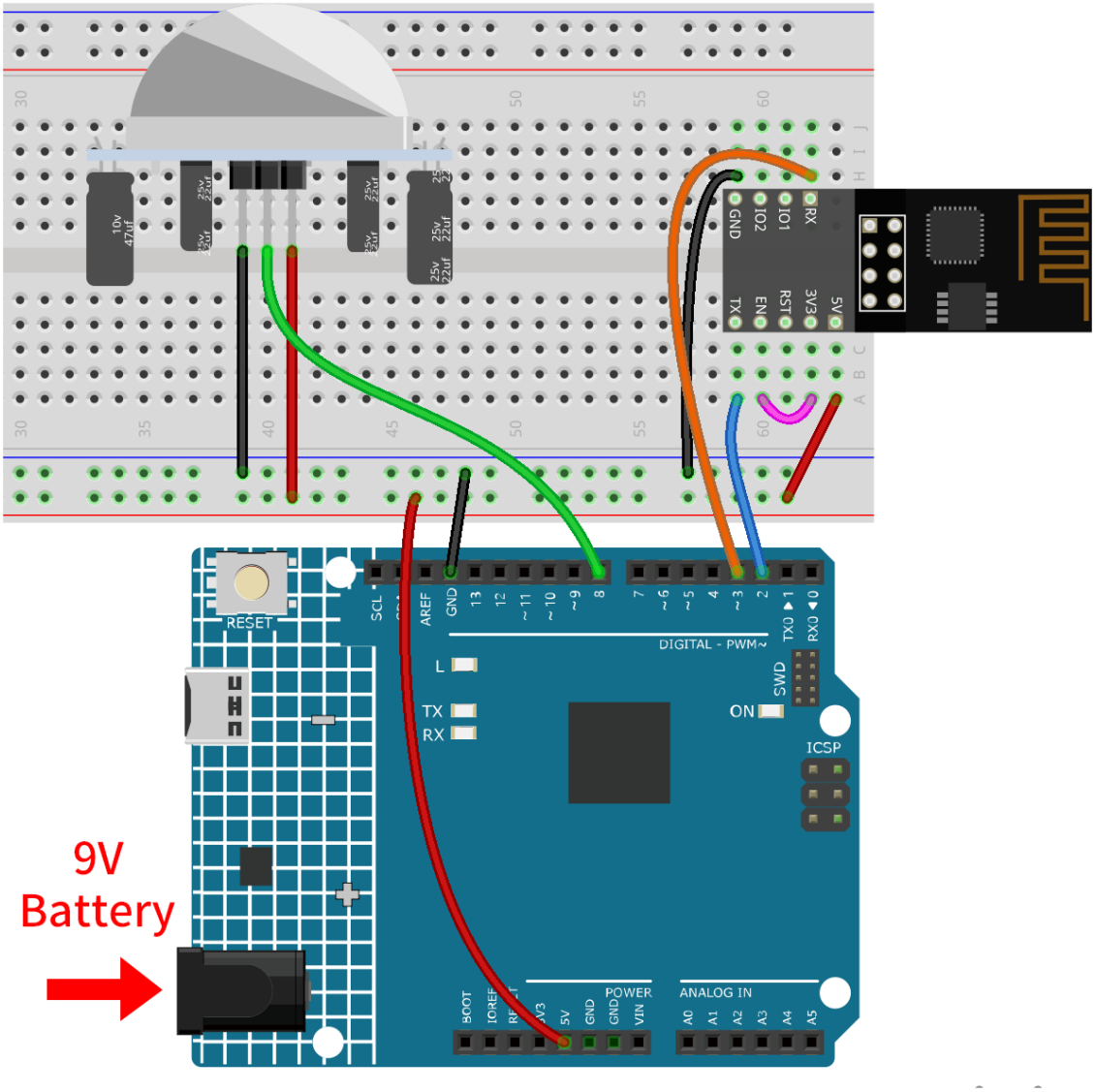
-
-
-
-
-

2.5.3 Einbruchmeldeanlage mit Blynk

Dieses Projekt demonstriert eine einfache häusliche Einbruchmeldeanlage mit einem passiven Infrarot (PIR) Sensor (HC-SR501). Wenn das System über die Blynk-App auf den ‚Abwesenheits'-Modus gesetzt ist, überwacht der PIR-Sensor Bewegungen. Jede erkannte Bewegung löst eine Benachrichtigung in der Blynk-App aus und warnt den Benutzer vor einem möglichen Einbruch.

1. Schaltung aufbauen

Bemerkung: Das ESP8266-Modul benötigt einen hohen Strom, um eine stabile Betriebsumgebung zu gewährleisten. Stellen Sie daher sicher, dass die 9V-Batterie angeschlossen ist.



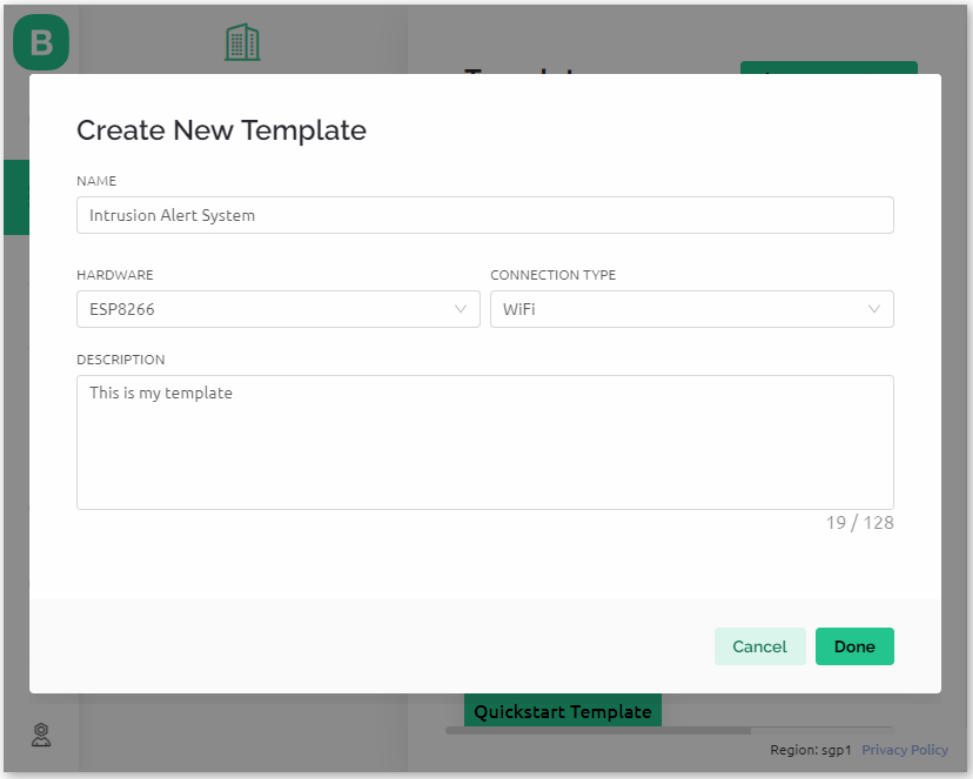
- *Arduino UNO R4 Minima-Platine*
- *ESP8266-Modul*
- cpn_pir_motion

2. Blynk konfigurieren

Bemerkung: Wenn Sie mit Blynk noch nicht vertraut sind, wird dringend empfohlen, diese beiden Tutorials zuerst zu lesen. *Einführung in Blynk* ist ein Einsteigerleitfaden für Blynk, der auch die Konfiguration von ESP8266 und die Registrierung bei Blynk umfasst. Und *Flammenwarnsystem mit Blynk* ist ein einfaches Beispiel, aber die Schritterklärung wird detaillierter sein.

2.1 Vorlage erstellen

Zunächst müssen wir eine Vorlage in Blynk erstellen. Folgen Sie den untenstehenden Schritten, um eine **„Intrusion Alert System“**-Vorlage zu erstellen.

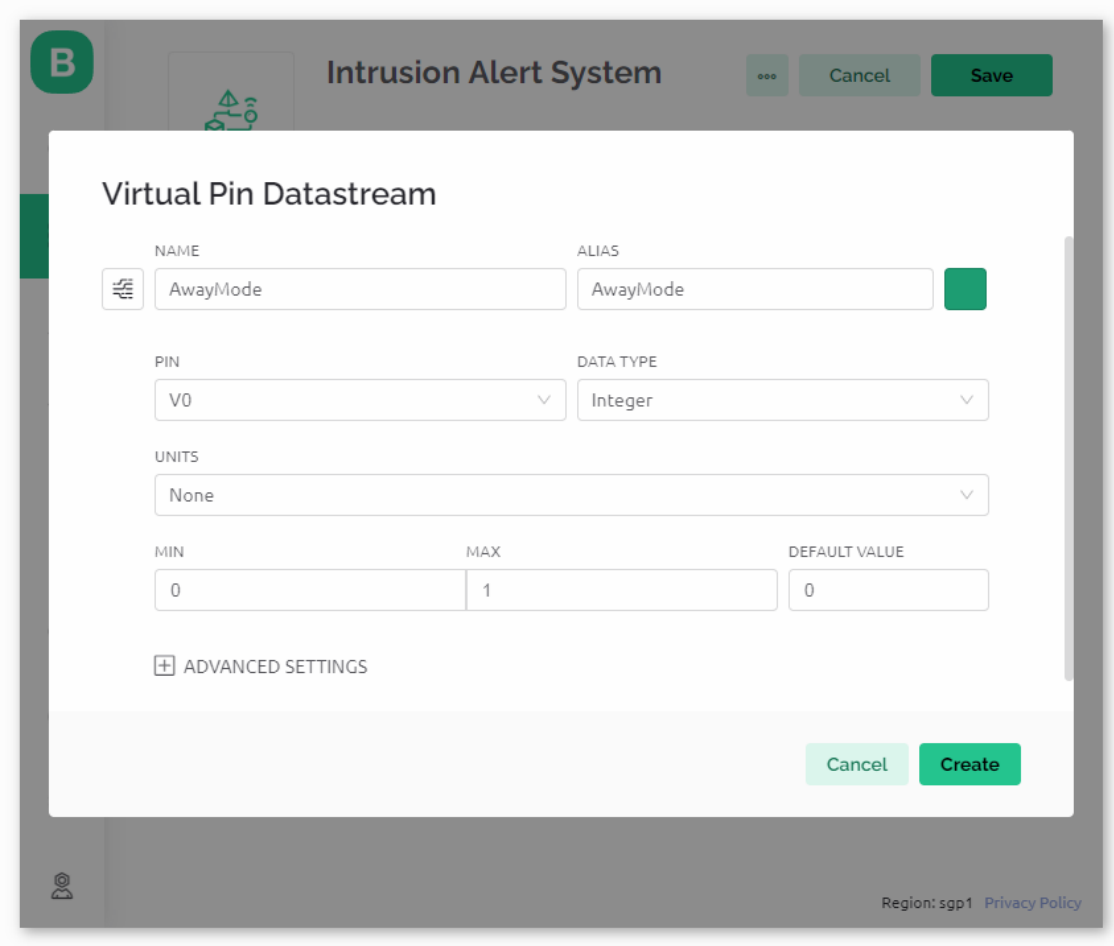


2.2 Datenströme

Erstellen Sie **Datastreams** vom Typ **Virtual Pin** auf der **Datenstrom**-Seite, um Daten von ESP8266 und Uno R4-Board zu empfangen.

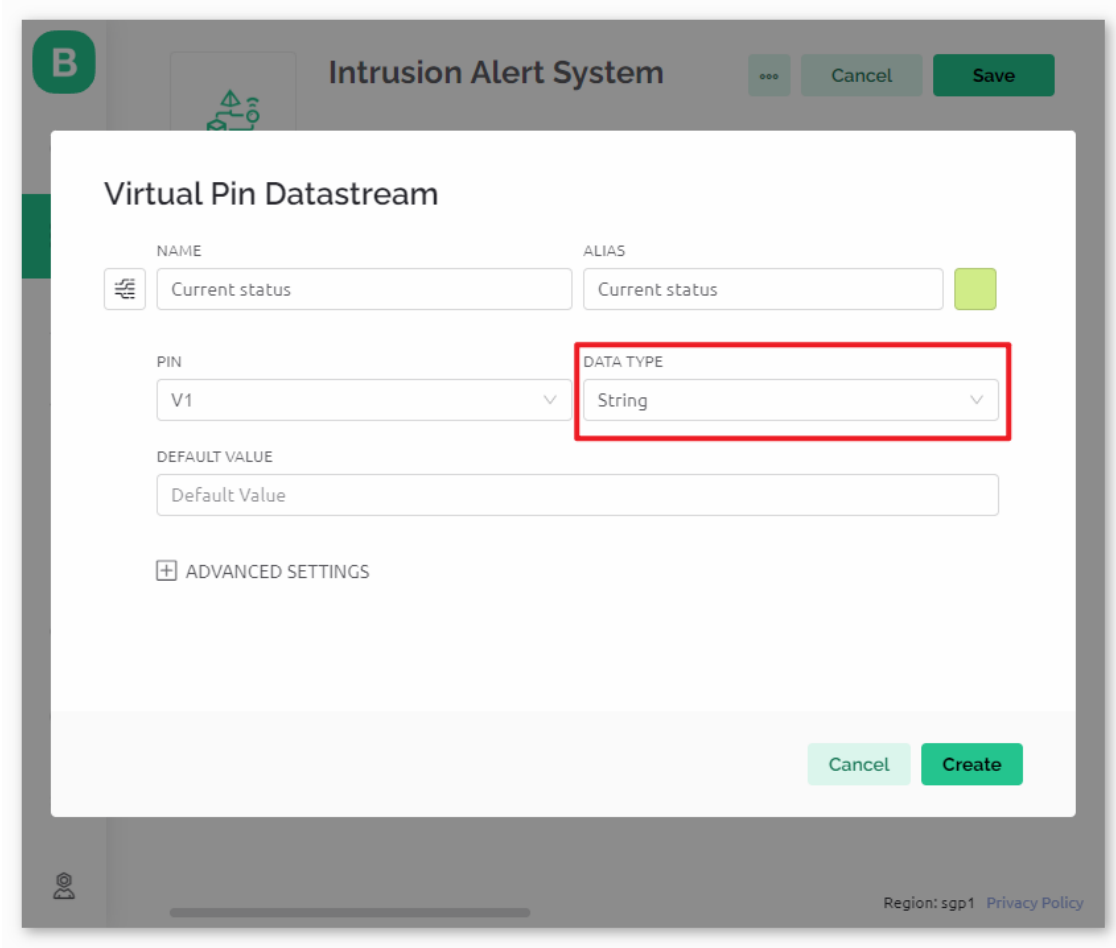
- Erstellen Sie den virtuellen Pin V0 gemäß dem folgenden Schema:

  Setzen Sie den Namen des **Virtual Pin V0** auf **AwayMode**. Legen Sie den **DATA TYPE** auf **Integer** und MIN und MAX auf **0** und **1** fest.

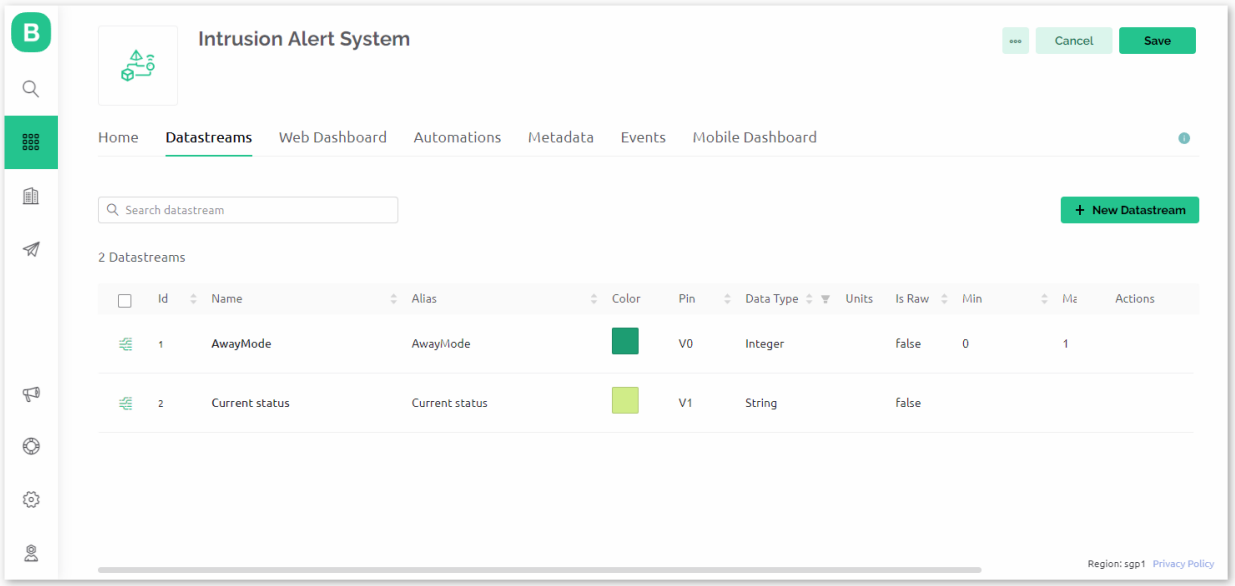


- Erstellen Sie den virtuellen Pin V1 gemäß dem folgenden Schema:

  Setzen Sie den Namen des **Virtual Pin V1** auf **Current status**. Legen Sie den **DATA TYPE** auf **String** fest.



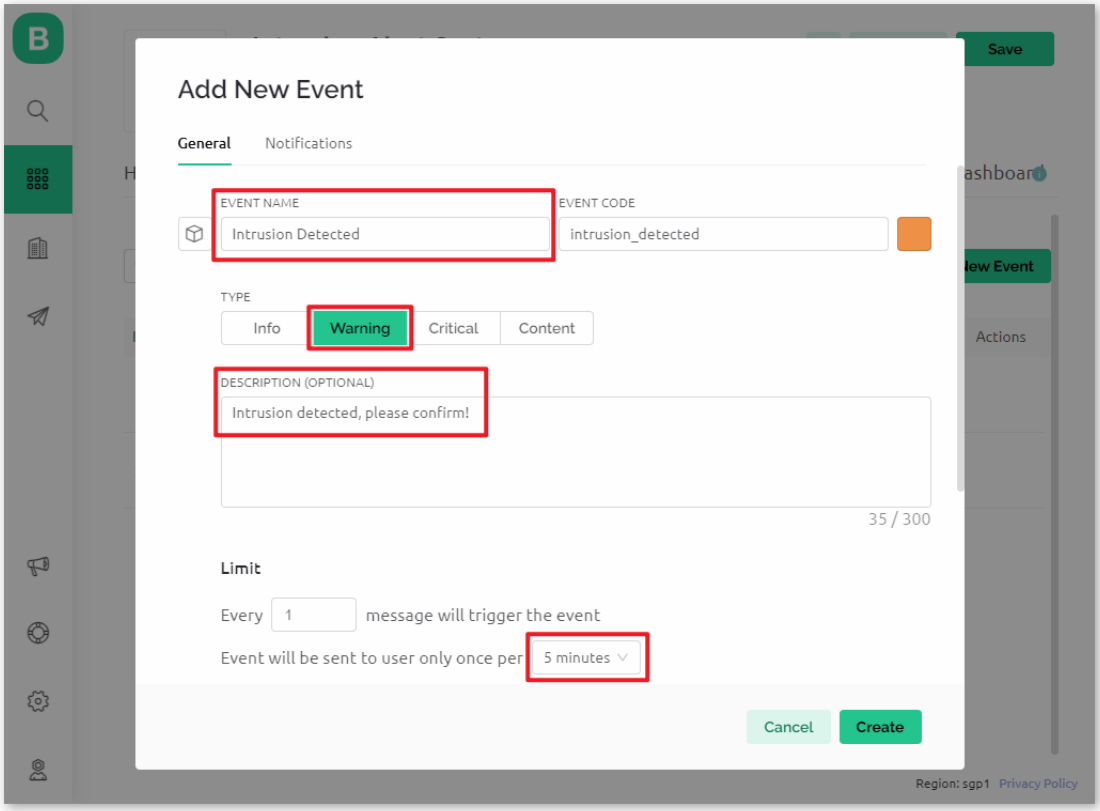
Stellen Sie sicher, dass Sie zwei virtuelle Pins gemäß den oben beschriebenen Schritten eingerichtet haben.



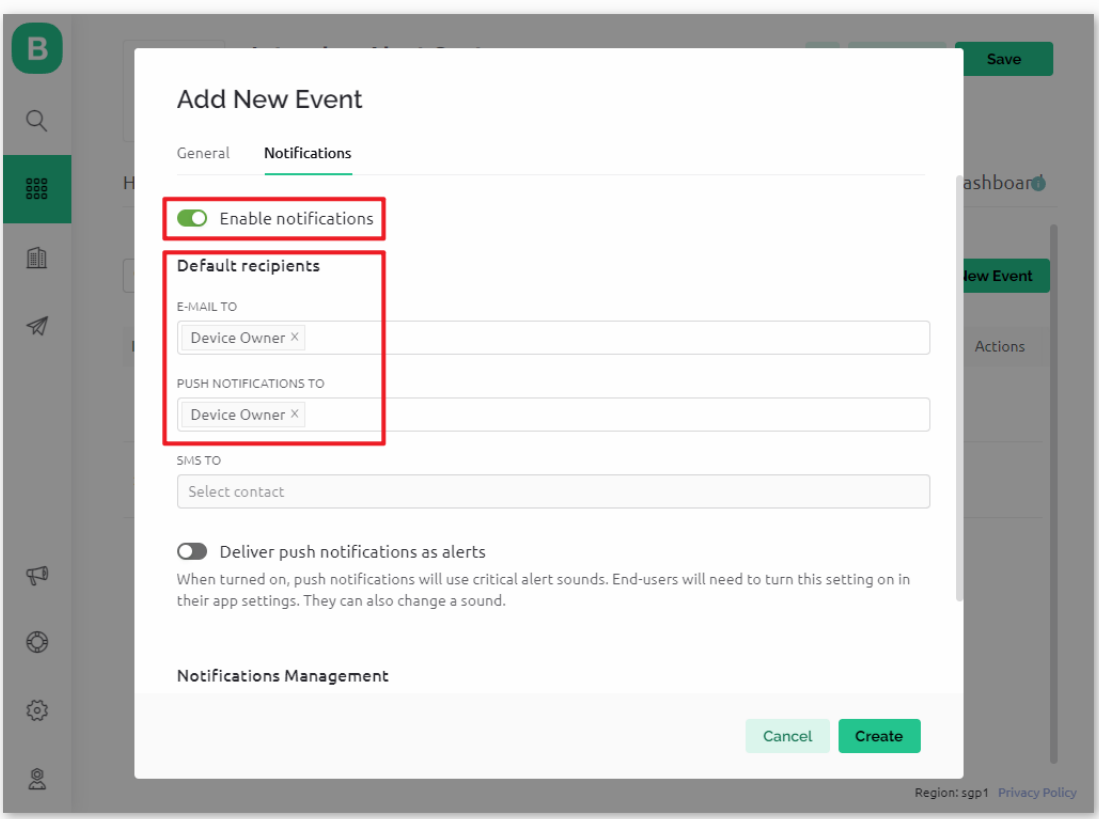
2.3 Ereignis

Als nächstes erstellen wir ein **event**, das die Erkennung von Einbrüchen protokolliert und eine E-Mail-Benachrichtigung sendet.

Bemerkung: Es wird empfohlen, die Einstellungen mit meinen Einstellungen konsistent zu halten. Andernfalls müssen Sie möglicherweise den Code ändern, um das Projekt auszuführen. Stellen Sie sicher, dass der **EVENT CODE** als `intrusion_detected` festgelegt ist.



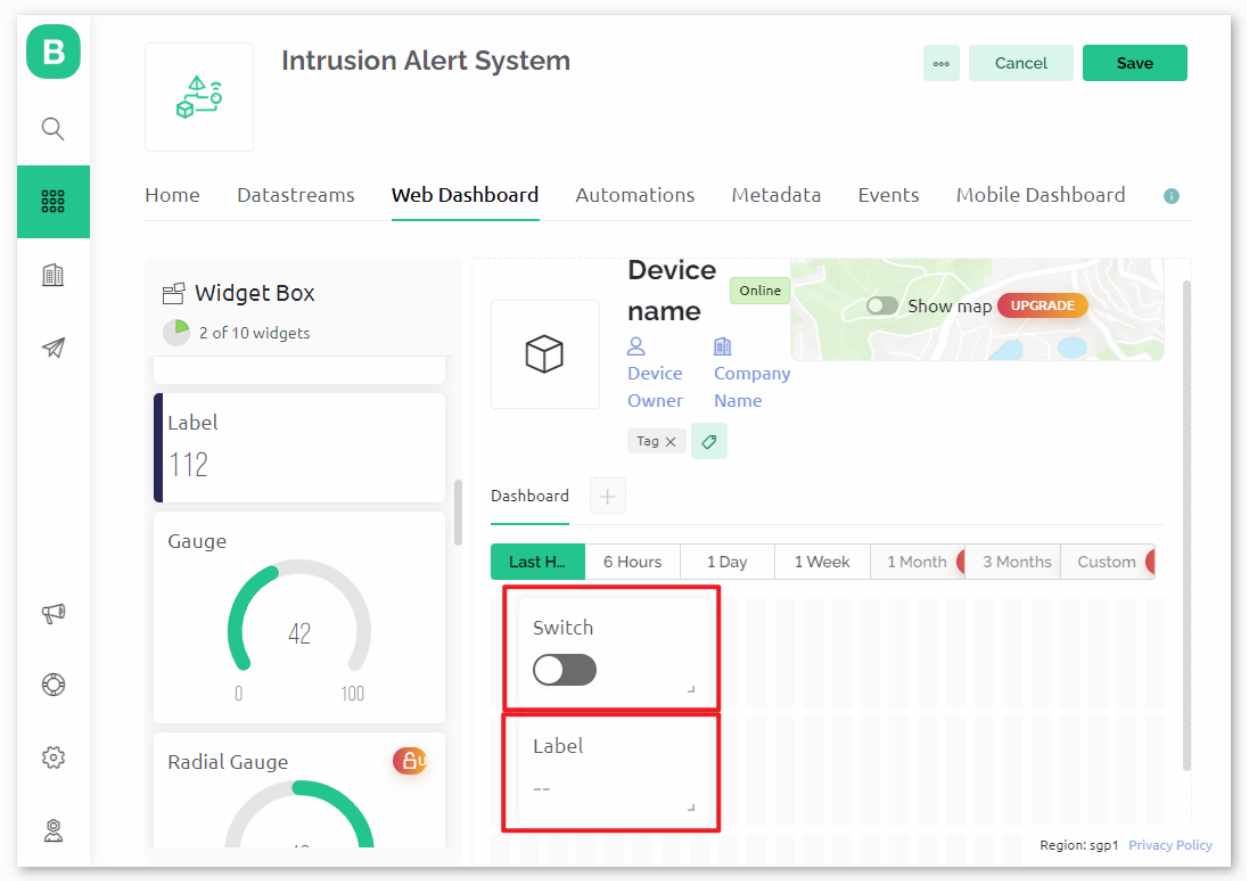
Gehen Sie zur **Notifications**-Seite und konfigurieren Sie die E-Mail-Einstellungen.



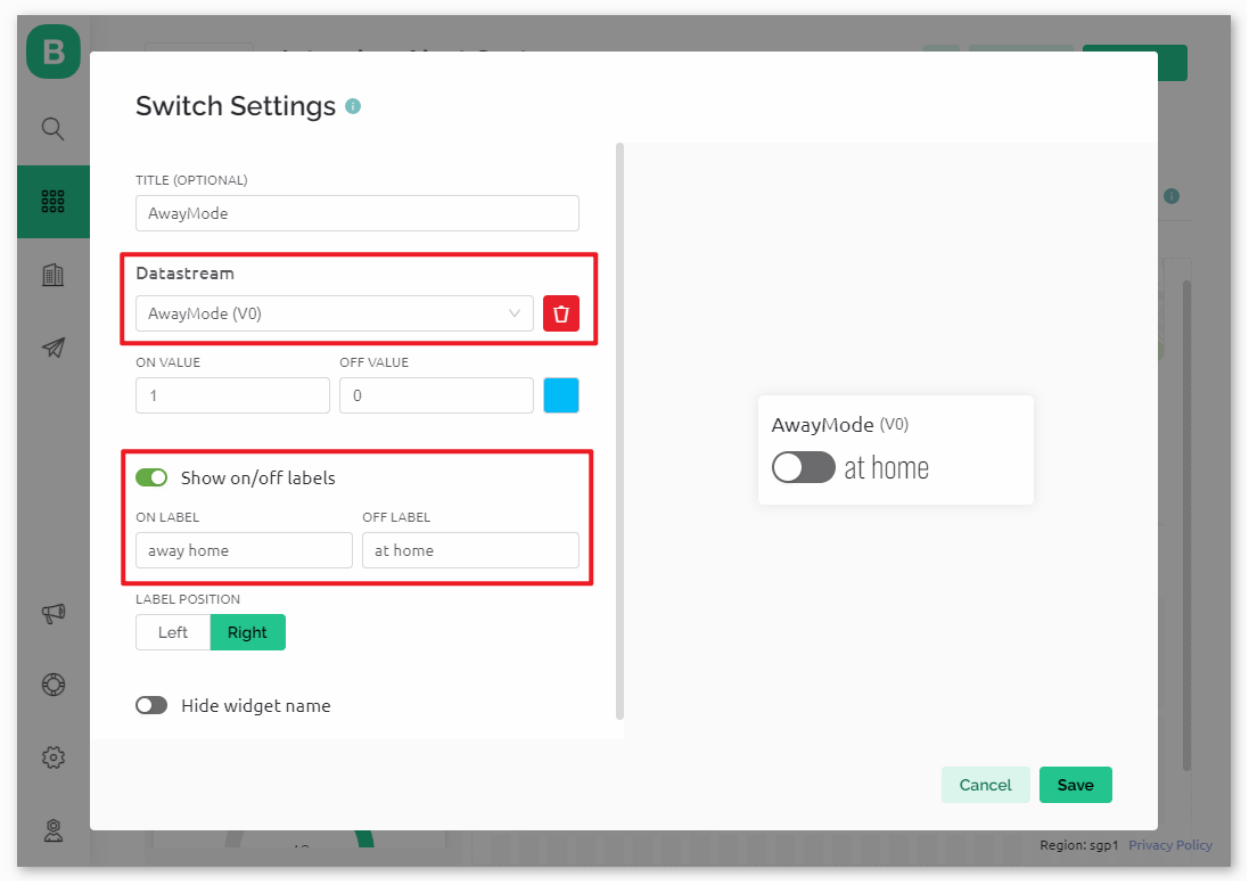
2.4 Web-Dashboard

Wir müssen auch das **Web-Dashboard** konfigurieren, um mit der Einbruchmeldeanlage interagieren zu können.

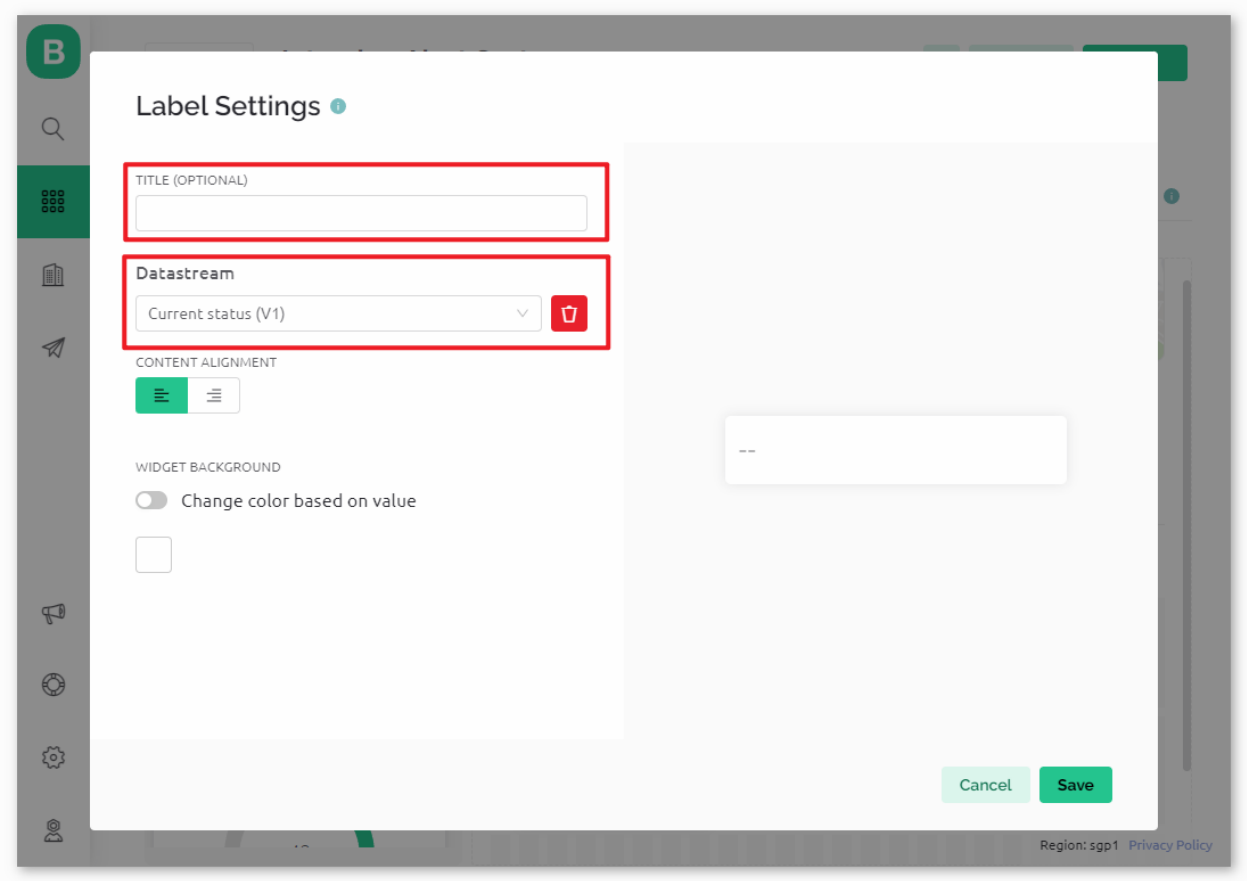
Ziehen Sie ein **Switch widget** und ein **Label widget** auf die **Web-Dashboard**-Seite.



Auf der Einstellungsseite des **Switch widget** wählen Sie **Datenstrom** als **AwayMode(V0)**. Setzen Sie **ONLABEL** und **OFFLABEL**, um „abwesend zu Hause“ anzuzeigen, wenn der Schalter eingeschaltet ist, und „zu Hause“, wenn der Schalter ausgeschaltet ist.

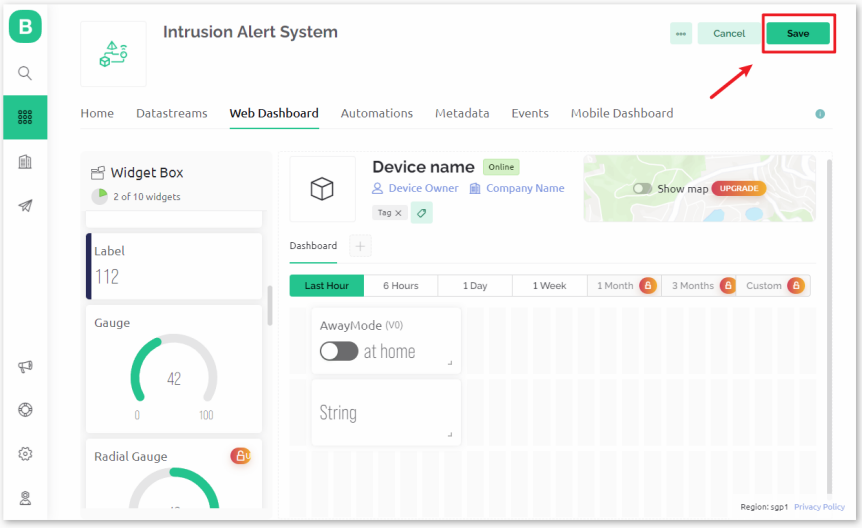


Auf der Einstellungsseite des **Label widget** wählen Sie **Datenstrom** als **Current status(V1)**.



2.5 Vorlage speichern

Zum Schluss vergessen Sie nicht, die Vorlage zu speichern.



3. Code ausführen

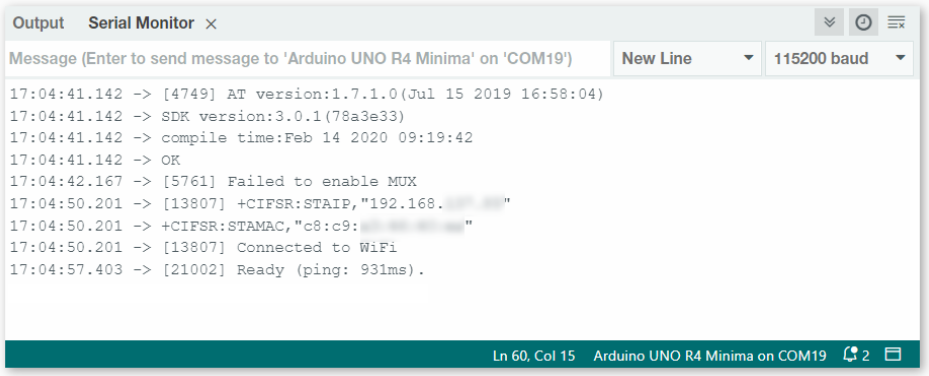
1. Öffnen Sie die Datei `02-Intrusion_alert_system.ino` unter dem Pfad `ultimate-sensor-kit\iot_project\wifi\02-Intrusion_alert_system` oder kopieren Sie diesen Code in die **Arduino IDE**.
2. Erstellen Sie ein Blynk-Gerät mit der Vorlage „Einbruchmeldesystem“. Anschließend ersetzen Sie die Werte für `BLYNK_TEMPLATE_ID`, `BLYNK_TEMPLATE_NAME` und `BLYNK_AUTH_TOKEN` durch Ihre eigenen.

```
#define BLYNK_TEMPLATE_ID "TMPxxxxxxx"
#define BLYNK_TEMPLATE_NAME "Intrusion Alert System"
#define BLYNK_AUTH_TOKEN "xxxxxxxxxxxxx"
```

3. Sie müssen auch die `ssid` und das `Passwort` des von Ihnen verwendeten WLANs eingeben.

```
char ssid[] = "your_ssid";
char pass[] = "your_password";
```

4. Nach der Auswahl des richtigen Boards und Ports klicken Sie auf die Schaltfläche **Hochladen**.
5. Öffnen Sie den seriellen Monitor (Baudrate auf 115200 einstellen) und warten Sie auf eine Meldung, die eine erfolgreiche Verbindung anzeigt.



Bemerkung: Sollte die Meldung `ESP is not responding` erscheinen, wenn Sie eine Verbindung herstellen, befolgen Sie bitte diese Schritte:

- Stellen Sie sicher, dass die 9V-Batterie angeschlossen ist.
- Setzen Sie das ESP8266-Modul zurück, indem Sie den Pin RST für 1 Sekunde mit GND verbinden und dann trennen.
- Drücken Sie die Reset-Taste auf dem R4-Board.

Manchmal müssen Sie die oben genannten Schritte 3-5 Mal wiederholen, bitte haben Sie Geduld.

---

4. Code-Erklärung

1. **Konfiguration & Bibliotheken**

   Hier werden Konstanten und Anmeldedaten für Blynk festgelegt. Die erforderlichen Bibliotheken für das ESP8266-WiFi-Modul und Blynk werden eingebunden.

```
#define BLYNK_TEMPLATE_ID "TMPxxxx"
#define BLYNK_TEMPLATE_NAME "Intrusion Alert System"
#define BLYNK_AUTH_TOKEN "xxxxxx-"
#define BLYNK_PRINT Serial

#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
```

2. **WiFi-Einrichtung**

   Konfigurieren der WLAN-Zugangsdaten und Einrichten der seriellen Software-Kommunikation mit dem ESP01-Modul.

```
char ssid[] = "your_ssid";
char pass[] = "your_password";

SoftwareSerial EspSerial(2, 3);
#define ESP8266_BAUD 115200
ESP8266 wifi(&EspSerial);
```

3. **PIR-Sensor-Konfiguration**

   Definieren des Pins, an dem der PIR-Sensor angeschlossen ist, und Initialisieren der Zustandsvariablen.

```
const int sensorPin = 8;
int state = 0;
int awayHomeMode = 0;
BlynkTimer timer;
```

4. **setup() Funktion**

   Diese initialisiert den PIR-Sensor als Eingang, richtet die serielle Kommunikation ein, stellt eine Verbindung zum WLAN her und konfiguriert Blynk.

   - Mit `timer.setInterval(1000L, myTimerEvent)` wird das Timerintervall in setup() festgelegt. Hier legen wir fest, dass die Funktion `myTimerEvent()` alle **1000 ms** ausgeführt wird. Sie können den ersten Parameter von `timer.setInterval(1000L, myTimerEvent)` ändern, um das Intervall zwischen den `myTimerEvent`-Ausführungen zu ändern.

```
void setup() {
   pinMode(sensorPin, INPUT);
   Serial.begin(115200);
   EspSerial.begin(ESP8266_BAUD);
   delay(10);
   Blynk.config(wifi, BLYNK_AUTH_TOKEN);
   Blynk.connectWiFi(ssid, pass);
   timer.setInterval(1000L, myTimerEvent);
}
```

5. **loop() Funktion**

Die Schleifenfunktion führt wiederholt Blynk und die Blynk-Timer-Funktionen aus.

```
void loop() {
   Blynk.run();
   timer.run();
}
```

6. **Blynk-App-Interaktion**

   Diese Funktionen werden aufgerufen, wenn das Gerät eine Verbindung zu Blynk herstellt und wenn sich der Zustand des virtuellen Pins V0 in der Blynk-App ändert.

   - Jedes Mal, wenn das Gerät eine Verbindung zum Blynk-Server herstellt oder aufgrund schlechter Netzwerkbedingungen erneut eine Verbindung herstellt, wird die Funktion `BLYNK_CONNECTED()` aufgerufen. Der Befehl `Blynk.syncVirtual()` fordert einen einzelnen virtuellen Pin-Wert an. Der angegebene virtuelle Pin wird `BLYNK_WRITE()` aufrufen. Weitere Details finden Sie unter .
   - Wann immer sich der Wert eines virtuellen Pins auf dem BLYNK-Server ändert, wird `BLYNK_WRITE()` ausgelöst. Weitere Details unter .

```
// 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 bei erkannter Bewegung eine Benachrichtigung an Blynk.

   - Wir verwenden `Blynk.virtualWrite(V1, "Jemand ist in Ihrem Haus! Bitte überprüfen! ")`, um den Text eines Labels zu ändern.
   - Mit `Blynk.logEvent("intrusion_detected");` wird ein Ereignis in Blynk protokolliert.

```
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
      if (state == HIGH) {
        Serial.println("Somebody here!");
        Blynk.virtualWrite(V1, "Somebody in your house! Please check!");
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        Blynk.logEvent("intrusion_detected");
      }
   }
}
```

Referenzen

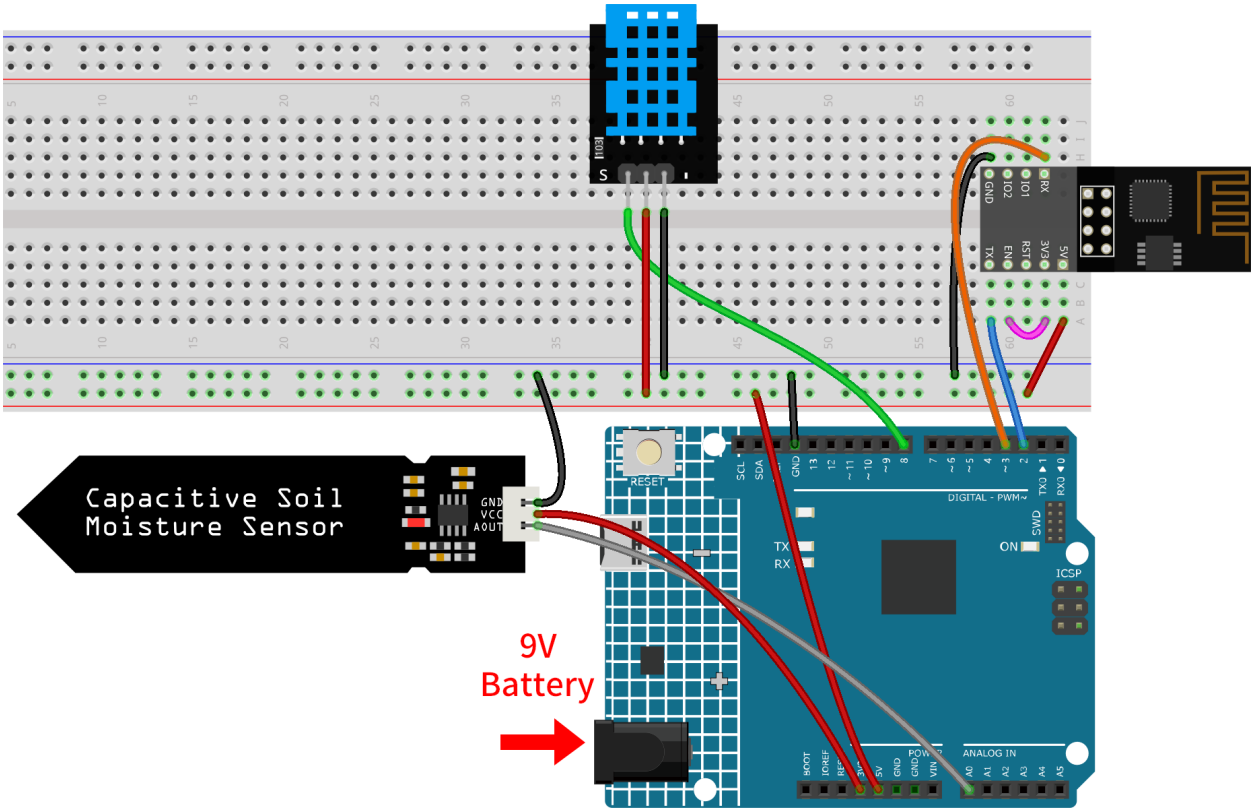
-
-
-
-
-
-
-

2.5.4 Pflanzenüberwachung mit Blynk

Dieses Projekt realisiert ein Demonstrations-System zur Pflanzenüberwachung, das aktuelle Temperatur, Luftfeuchtigkeit, Lichtintensität und Bodenfeuchtigkeit erfasst. Die Daten werden anschließend in Blynk angezeigt, ergänzt durch Vorschläge, die sich an den Feuchtigkeitswerten des Bodens orientieren.

1. Schaltung aufbauen

Bemerkung: Das ESP8266-Modul benötigt einen hohen Stromfluss, um stabil zu funktionieren. Stellen Sie daher sicher, dass die 9V-Batterie angeschlossen ist.



- *Arduino UNO R4 Minima-Platine*
- *ESP8266-Modul*
- *Temperatur- und Feuchtigkeitssensor-Modul (DHT11)*
- *Kapazitives Bodenfeuchtigkeitsmodul*

2. Blynk konfigurieren

Bemerkung: Wenn Sie mit Blynk noch nicht vertraut sind, empfehlen wir dringend, zuerst diese beiden Anleitungen zu lesen. *Einführung in Blynk* ist ein Einsteigerleitfaden für Blynk, der auch die Konfiguration von ESP8266 und die Registrierung bei Blynk enthält. *Flammenwarnsystem mit Blynk* ist ein einfaches Beispiel, aber die Beschreibung der Schritte ist detaillierter.

2.1 Vorlage erstellen

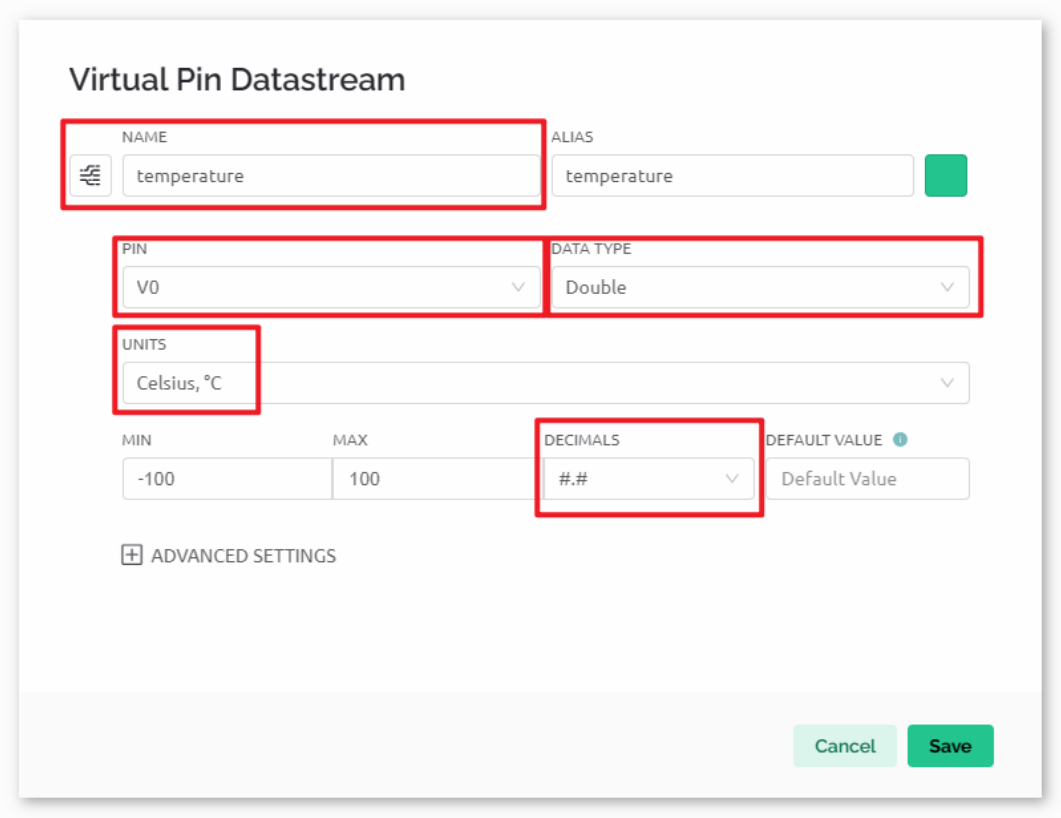
Zunächst müssen wir eine Vorlage in Blynk erstellen. Erstellen Sie eine Vorlage namens **„Pflanzenüberwachung“**.

2.2 Datenstrom

Erstellen Sie **Datastreams** vom Typ **Virtual Pin** auf der **Datastreams**-Seite, um Daten von ESP8266 und Uno R4 zu empfangen.

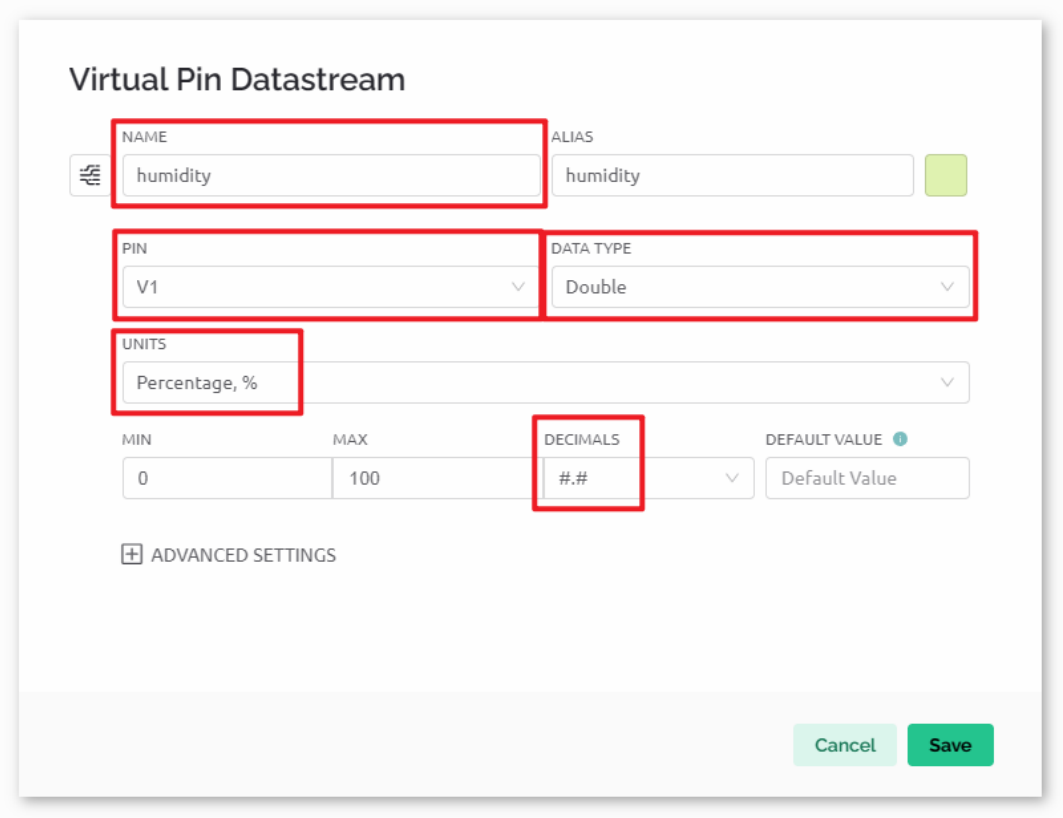
- Erstellen Sie den **Virtual Pin V0** gemäß folgendem Schema:

  Benennen Sie den **Virtual Pin V0** in **temperature** um. Setzen Sie den **DATA TYPE** auf **Double** und MIN und MAX auf **-100** und **100**. Setzen Sie die **UNITS** auf **Celsius,℃**.



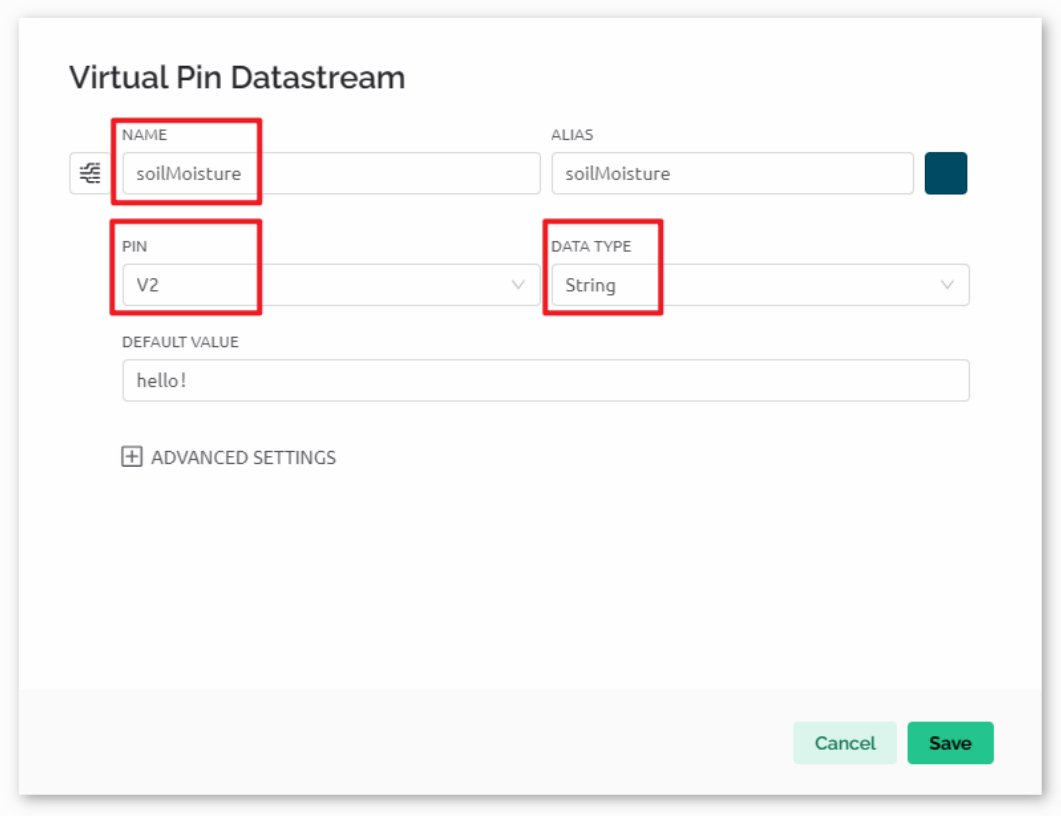
- Erstellen Sie den **Virtual Pin V1** gemäß folgendem Schema:

  Benennen Sie den **Virtual Pin V1** in **humidity** um. Setzen Sie den **DATA TYPE** auf **Double** und MIN und MAX auf **0** und **100**. Setzen Sie die **UNITS** auf **Percentage,%**.



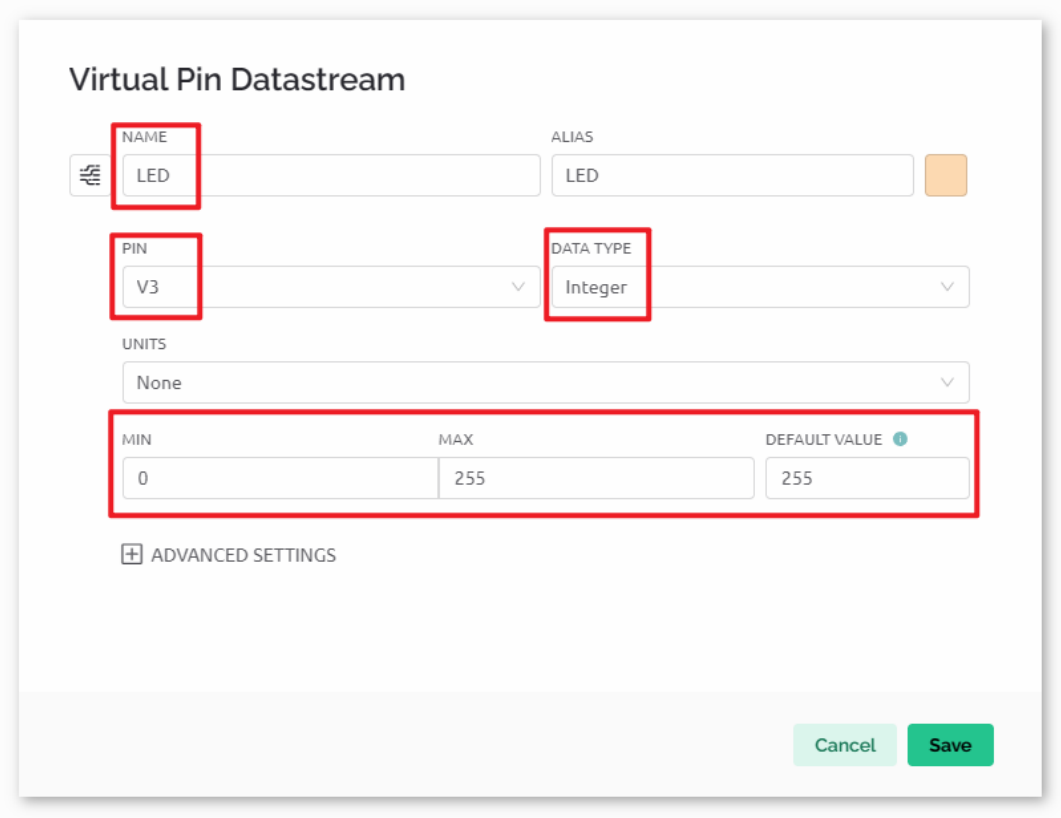
- Erstellen Sie den **Virtual Pin V2** gemäß folgendem Schema:

  Benennen Sie den **Virtual Pin V2** in **soilMoisture** um. Setzen Sie den **DATENTYP** auf **String**.



- Erstellen Sie den **Virtual Pin V3** gemäß folgendem Schema:

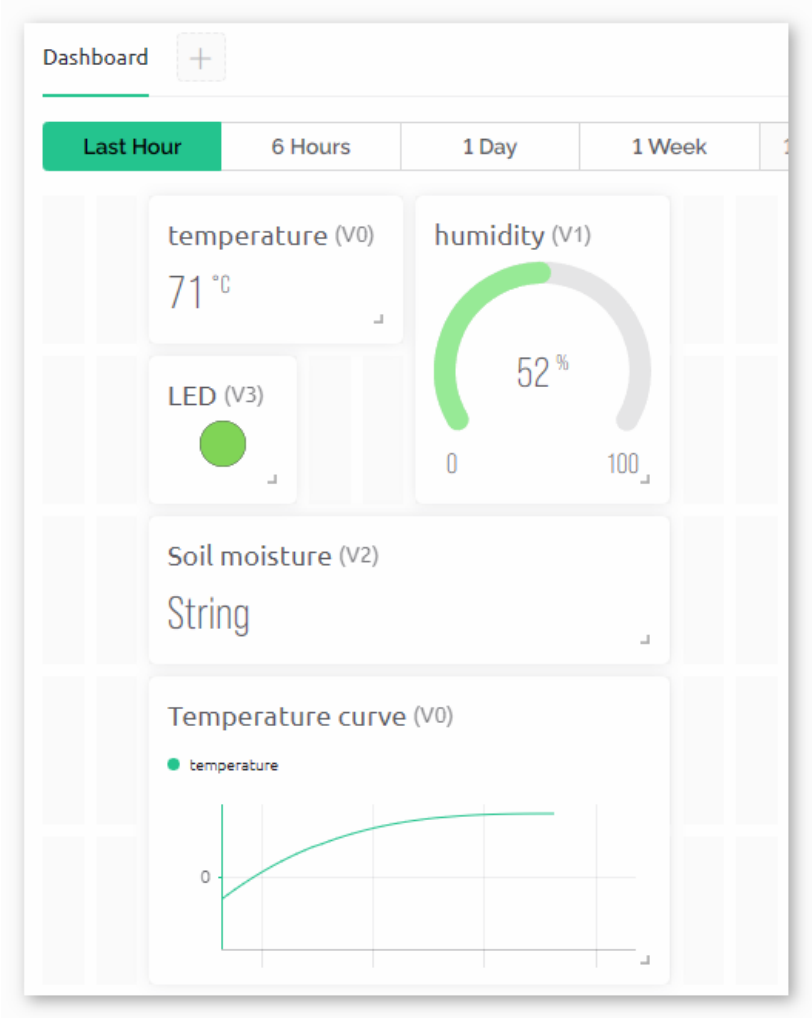
  Benennen Sie den **Virtual Pin V3** in **LED** um. Setzen Sie den **DATA TYPE** auf **Integer** und MIN und MAX auf **0** und **255**.



2.3 Web-Dashboard

Wir müssen auch das **Web-Dashboard** konfigurieren, um mit der Pflanzenüberwachung interagieren zu können.

Konfigurieren Sie das Web-Dashboard gemäß dem folgenden Schema. Wir verwenden Widgets wie Beschriftungen, Anzeigen, LEDs und Diagramme. Achten Sie darauf, jedes Widget an seinen entsprechenden virtuellen Pin zu binden.



2.4 Vorlage speichern

Vergessen Sie nicht, die Vorlage am Ende zu speichern.

3. Code ausführen

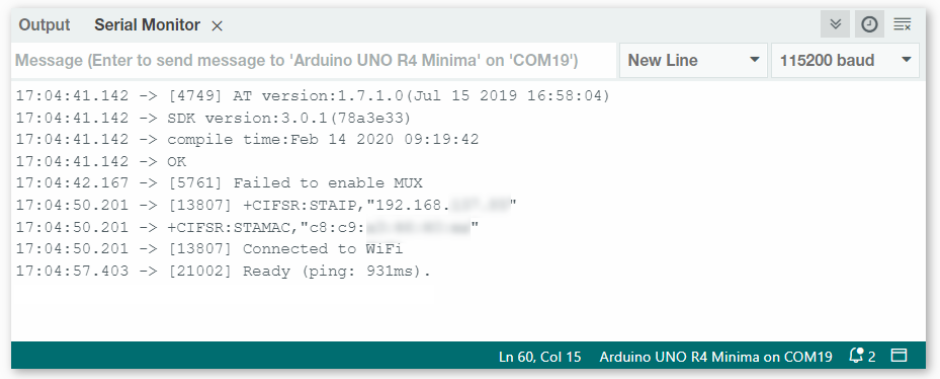
1. Öffnen Sie die Datei `03-Plant_monitor.ino` im Pfad `ultimate-sensor-kit\iot_project\wifi\03-Plant_monitor` oder kopieren Sie den Code in die **Arduino IDE**.
2. Erstellen Sie ein Blynk-Gerät mit der Vorlage „Pflanzenüberwachung“. Ersetzen Sie dann die Werte für `BLYNK_TEMPLATE_ID`, `BLYNK_TEMPLATE_NAME` und `BLYNK_AUTH_TOKEN` durch Ihre eigenen.

```
#define BLYNK_TEMPLATE_ID "TMPxxxxxxx"
#define BLYNK_TEMPLATE_NAME "Plant Monitor"
#define BLYNK_AUTH_TOKEN "xxxxxxxxxxxxx"
```

3. Geben Sie ebenfalls die `ssid` und das `Passwort` des genutzten WLANs ein.

```
char ssid[] = "Ihr_SSID";
char pass[] = "Ihr_Passwort";
```

4. Nach der Auswahl des korrekten Boards und Ports klicken Sie auf den **Hochladen**-Button.
5. Öffnen Sie den Seriellen Monitor (Baudrate auf 115200 einstellen) und warten Sie auf eine Erfolgsmeldung zur Verbindung.



Bemerkung: Falls die Meldung `ESP is not responding` erscheint, gehen Sie bitte wie folgt vor:

- Stellen Sie sicher, dass die 9V-Batterie angeschlossen ist.
- Setzen Sie das ESP8266-Modul zurück, indem Sie den RST-Pin für eine Sekunde auf GND legen und dann wieder entfernen.
- Drücken Sie den Reset-Knopf auf dem R4-Board.

Manchmal müssen Sie die obigen Schritte 3-5 Mal wiederholen. Bitte haben Sie Geduld.

---

4. Code-Erklärung

1. **Initialisierung von Bibliotheken und Definition von Konstanten**:

   Dieser Codeabschnitt beinhaltet die benötigten Bibliotheken und definiert bestimmte Konstanten, wie die Blynk-Vorlageninformationen und WLAN-Zugangsdaten.

```
#define BLYNK_TEMPLATE_ID "TMPLxxxxxxx"
#define BLYNK_TEMPLATE_NAME "Pflanzenüberwachung"
#define BLYNK_AUTH_TOKEN "xxxxxxxxxx"
#define BLYNK_PRINT Serial
#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
char ssid[] = "Ihr_SSID";
char pass[] = "Ihr_Passwort";
#include <SoftwareSerial.h>
SoftwareSerial EspSerial(2, 3);
#define ESP8266_BAUD 115200
ESP8266 wifi(&EspSerial);
```

2. **Einrichten des DHT-Sensors**:

   Der DHT-Sensor wird initialisiert und relevante Variablen zur Speicherung von Temperatur und Luftfeuchtigkeit festgelegt.

```
#include <DHT.h>
#define DHTPIN 8
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
float temperature;
float humidity;
```

3. **Einrichten des Bodenfeuchtesensors**:

   Konfiguration des Bodenfeuchtesensors. Grenzwerte für nasse und trockene Bedingungen werden festgelegt.

   Sie müssen Ihre eigenen Werte für `wetSoil` und `drySoil` gemäß Ihrer konkreten Situation ermitteln. Messen Sie den Wert des Bodenfeuchtemoduls als `drySoil`, wenn der Boden trocken ist, und den Wert innerhalb eines geeigneten Bereichs, den Sie als am feuchtesten betrachten (jenseits dieses Bereichs wäre zu nass), als `wetSoil`.

```
#define wetSoil 320
#define drySoil 400
const int moistureSensorPin = A0;
int moisture;
String soilStatus;
```

4. **Timer-Einrichtung**:

   Ein Timer wird konfiguriert, der die Häufigkeit der Datenerfassung und -aktualisierung steuert.

```
BlynkTimer timer;
```

5. **Initialisierung in der Setup-Funktion**:

   In diesem Abschnitt wird die serielle Kommunikation eingerichtet, das ESP8266 für das WLAN konfiguriert und der DHT-Sensor gestartet.

   - Mit `timer.setInterval(5000L, myTimerEvent)` wird das Timer-Intervall in der Setup()-Funktion festgelegt. Hier haben wir es so eingestellt, dass die Funktion `myTimerEvent()` alle **5000 ms** ausgeführt wird. Sie können den ersten Parameter von `timer.setInterval(1000L, myTimerEvent)` ändern, um das Intervall zwischen den `myTimerEvent`-Ausführungen zu ändern.

```
void setup() {
   Serial.begin(115200);
   EspSerial.begin(ESP8266_BAUD);
   delay(10);
   Blynk.config(wifi, BLYNK_AUTH_TOKEN);
   Blynk.connectWiFi(ssid, pass);
   timer.setInterval(5000L, myTimerEvent);
   dht.begin();
}
```

6. **loop()-Funktion**:

   Die Hauptschleife führt den Blynk-Prozess und den Timer aus.

```
void loop() {
   Blynk.run();
   timer.run();
}
```

7. **sendData()-Funktion**:

Diese Funktion liest Werte vom DHT- und Bodenfeuchtesensor, bestimmt den Zustand des Bodens und sendet die Daten an die Blynk-App.

- Verwenden Sie `Blynk.virtualWrite(vPin, value)` um Daten an virtuelle Pins in Blynk zu senden. Siehe dazu .
- Verwenden Sie `Blynk.setProperty(V3, "color", color)` um die Farbe der LED in Blynk einzustellen. Weitere Details finden Sie unter .

```
void sendData() {
   // (code for reading and determining values)
   Blynk.virtualWrite(V0, temperature);
   Blynk.virtualWrite(V1, humidity);
   Blynk.virtualWrite(V2, soilStatus);
   Blynk.virtualWrite(V3, 255);            // set blynk LED brightness
   Blynk.setProperty(V3, "color", color);  // set blynk LED color
}
```

8. **Daten auf Serial Monitor ausgeben**:

   Diese Funktion ist nützlich für Debugging-Zwecke und zur lokalen Überprüfung der Messwerte im seriellen Monitor der Arduino IDE.

```
void printData() {
   // (Code zur Ausgabe der Werte auf den Serial Monitor)
}
```

Referenzen

-
-
-

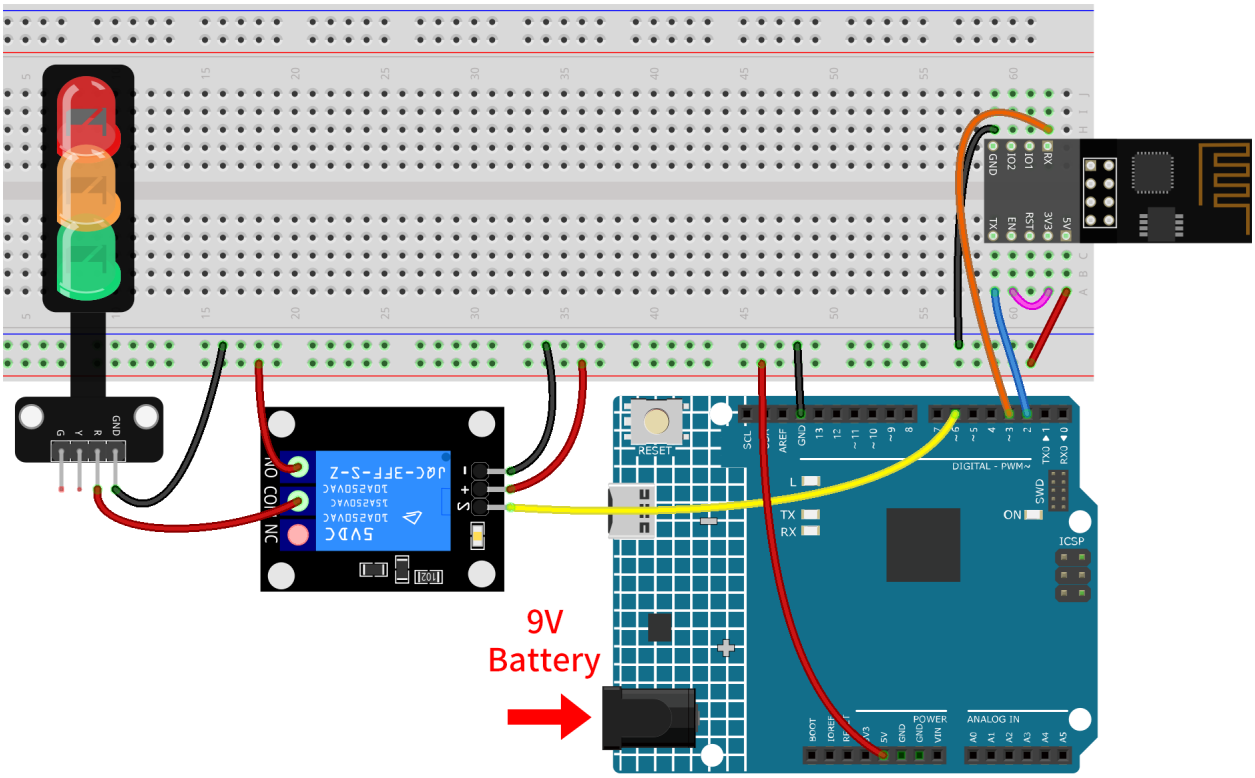
2.5.5 Ferngesteuerter Relais-Controller mit Blynk

Dieses Projekt zielt darauf ab, einen ferngesteuerten Relais-Controller zu entwickeln, der über einen virtuellen Schalter in der Blynk-App bedient werden kann. Wenn der Schalter eingeschaltet wird, wird der mit dem Relais verbundene digitale Pin auf HIGH gesetzt. Wird der Schalter ausgeschaltet, wird der Pin auf LOW gesetzt. Dies ermöglicht eine einfache Fernsteuerung des Relais und fungiert effektiv als Fernschalter.

1. Schaltkreis aufbauen

Warnung: Das folgende Beispiel zeigt, wie ein Relais zur Steuerung eines LED-Moduls verwendet wird. **Obwohl Sie das Relais in realen Anwendungen auch an andere Geräte anschließen können, ist bei der Arbeit mit HOCHSPANNUNG-Wechselstrom äußerste Vorsicht geboten. Unsachgemäße oder fehlerhafte Verwendung kann zu schweren Verletzungen oder sogar zum Tod führen. Dies ist daher nur für Personen gedacht, die mit HOCHSPANNUNG vertraut sind und darüber Bescheid wissen. Sicherheit hat immer Vorrang.**

Bemerkung: Das ESP8266-Modul benötigt einen hohen Strom, um eine stabile Betriebsumgebung zu gewährleisten. Stellen Sie daher sicher, dass die 9V-Batterie angeschlossen ist.



- *Arduino UNO R4 Minima-Platine*
- *ESP8266-Modul*
- *5V-Relaismodul*
- *Ampel-Modul*

2. Blynk konfigurieren

Bemerkung: Wenn Sie noch nicht mit Blynk vertraut sind, wird dringend empfohlen, zunächst diese beiden Tutorials zu lesen. *Einführung in Blynk* ist ein Einsteigerleitfaden für Blynk, der auch die Konfiguration des ESP8266 und die Registrierung bei Blynk umfasst. Und *Flammenwarnsystem mit Blynk* ist ein einfaches Beispiel, jedoch werden die Schritte darin ausführlicher beschrieben.

2.1 Vorlage erstellen

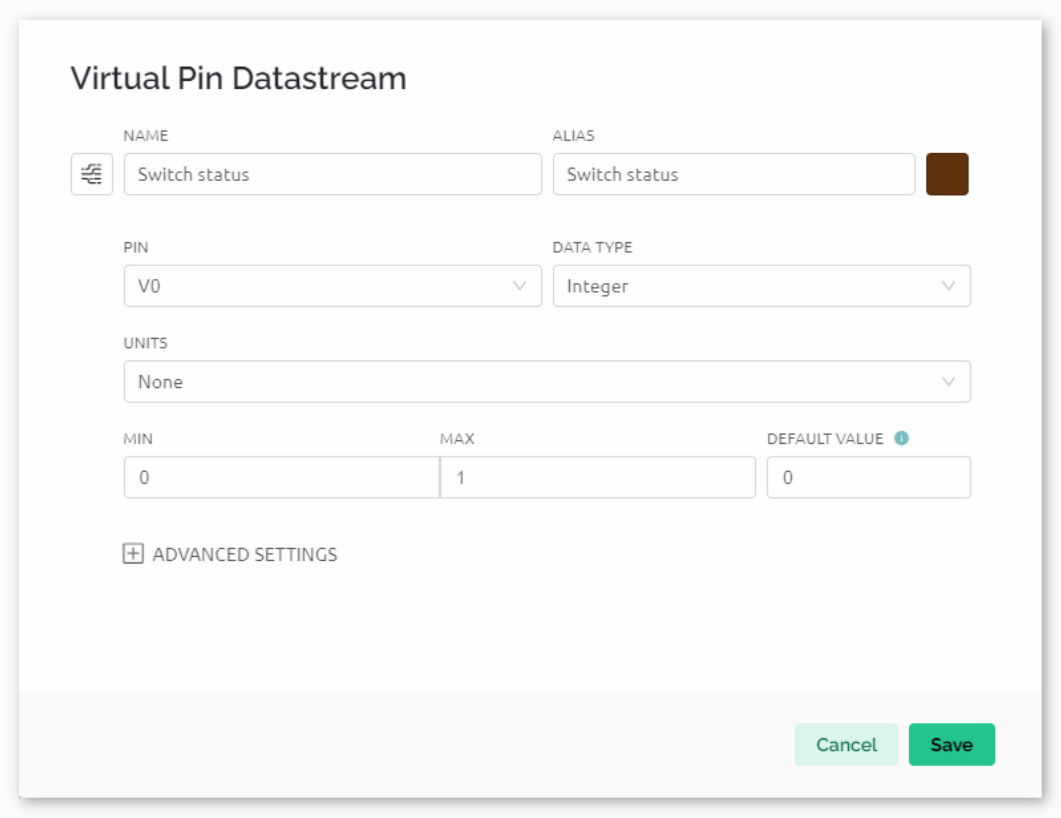
Zuerst müssen wir eine Vorlage in Blynk erstellen. Erstellen Sie eine **„Ferngesteuertes Relais“**-Vorlage.

2.2 Datenströme

Erstellen Sie **Datastreams** vom Typ **Virtual Pin** auf der **Datastreams**-Seite, um Daten vom esp8266 und uno r4 Board zu empfangen.

- Erstellen Sie den virtuellen Pin V0 gemäß dem folgenden Diagramm:

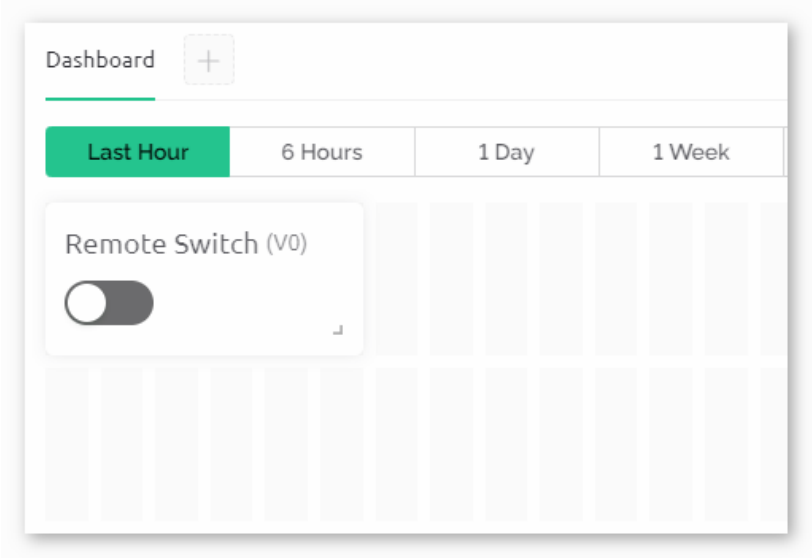
  Benennen Sie den **Virtual Pin V0** in **Switch status** um. Setzen Sie den **DATA TYPE** auf **Integer** und MIN und MAX auf **0** und **1**. Setzen Sie die **UNITS** auf **None**.



2.3 Web-Dashboard

Das **Web-Dashboard** muss ebenfalls konfiguriert werden, um mit dem ferngesteuerten Relais zu interagieren.

Konfigurieren Sie das Web-Dashboard entsprechend dem folgenden Diagramm. Stellen Sie sicher, dass jedes Widget an seinen entsprechenden virtuellen Pin gebunden ist.



3. Den Code ausführen

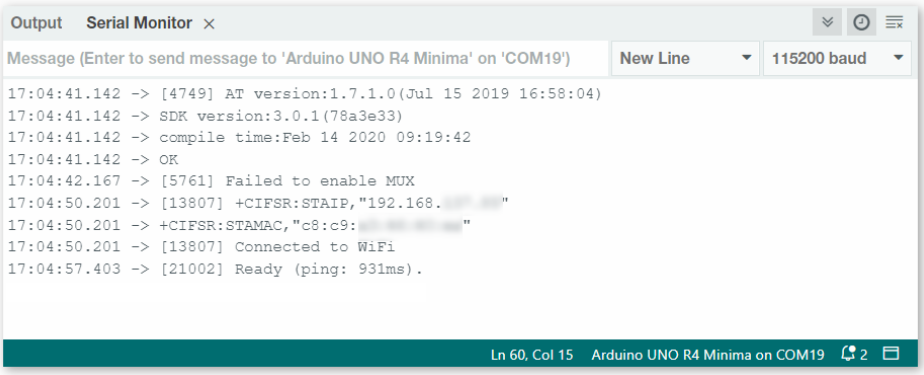
1. Öffnen Sie die Datei `06-Remote_relay_controller.ino` unter dem Pfad `ultimate-sensor-kit\iot_project\wifi\06-Remote_relay_controller`, oder kopieren Sie diesen Code in die **Arduino IDE**.
2. Legen Sie ein Blynk-Gerät mit der Vorlage „Ferngesteuertes Relais“ an. Ersetzen Sie dann die Werte für `BLYNK_TEMPLATE_ID`, `BLYNK_TEMPLATE_NAME` und `BLYNK_AUTH_TOKEN` durch Ihre eigenen.

```
#define BLYNK_TEMPLATE_ID "TMPxxxxxxx"
#define BLYNK_TEMPLATE_NAME "Remote relay"
#define BLYNK_AUTH_TOKEN "xxxxxxxxxxxxx"
```

3. Geben Sie auch die `ssid` und das `Passwort` des verwendeten WLANs ein.

```
char ssid[] = "Ihre_ssid";
char pass[] = "Ihr_Passwort";
```

4. Nach Auswahl des korrekten Boards und Ports klicken Sie auf die Schaltfläche **Hochladen**.
5. Öffnen Sie den seriellen Monitor (Baudrate auf 115200 einstellen) und warten Sie auf eine Meldung, die eine erfolgreiche Verbindung anzeigt.



Bemerkung: Sollte die Meldung `ESP is not responding` erscheinen, befolgen Sie diese Schritte.

- Stellen Sie sicher, dass die 9V-Batterie angeschlossen ist.
- Setzen Sie das ESP8266-Modul zurück, indem Sie den Pin RST für 1 Sekunde mit GND verbinden, dann ziehen Sie den Stecker.
- Drücken Sie die Reset-Taste auf dem R4-Board.

Manchmal müssen Sie die oben genannten Schritte 3-5 Mal wiederholen. Bitte haben Sie Geduld.

4. Code-Erklärung

1. Einrichten der Blynk-Zugangsdaten:

   Dieser Abschnitt enthält spezifische Einstellungen für die Blynk-App, wie zum Beispiel die Template-ID, den Gerätenamen und den Authentifizierungstoken.

```
#define BLYNK_TEMPLATE_ID "TMPLxxxxxxxxx"
#define BLYNK_TEMPLATE_NAME "Remote relay"
#define BLYNK_AUTH_TOKEN "xxxxxxxxxxx"
```

2. Einbindung der erforderlichen Bibliotheken:

   Hier binden wir die für das Projekt erforderlichen Bibliotheken ein. Diese ermöglichen die Kommunikation unseres Arduino über WiFi und die Interaktion mit der Blynk-App.

```
#define BLYNK_PRINT Serial
#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
#include <SoftwareSerial.h>
```

3. Konfiguration von WiFi und seriellen Einstellungen:

   Die SSID und das Passwort für das WiFi werden festgelegt. Darüber hinaus werden die Pins für die Software-Serienkommunikation mit dem ESP01 definiert. `ESP8266_BAUD` legt die Baudrate für das ESP8266-Modul fest.

```
char ssid[] = "Ihre_ssid";
char pass[] = "Ihr_Passwort";
SoftwareSerial EspSerial(2, 3);  // RX, TX
#define ESP8266_BAUD 115200
ESP8266 wifi(&EspSerial);
```

4. Definition des Relais-Pins:

   Wir definieren den digitalen Pin des Arduino, der zur Steuerung des Relais dient. Zudem initialisieren wir eine Variable `switchStatus`, um den Zustand unseres virtuellen Schalters in der Blynk-App zu speichern.

```
const int RelayPin = 6;
int switchStatus = 0;
```

5. Die Funktion setup():

   In dieser Funktion initialisieren wir den Relais-Pin als Ausgang, starten die serielle Kommunikation für Debugging-Zwecke und stellen die Verbindung zu Blynk mit den angegebenen WiFi-Zugangsdaten her.

```
void setup() {
  pinMode(RelayPin, OUTPUT);
  Serial.begin(115200);
  EspSerial.begin(ESP8266_BAUD);
  delay(10);
  Blynk.config(wifi, BLYNK_AUTH_TOKEN);
  Blynk.connectWiFi(ssid, pass);
}
```

6. Die Funktion loop():

   Sie führt kontinuierlich zwei essenzielle Funktionen aus, um die Verbindung zu Blynk aufrechtzuerhalten und Ereignisse (wie Änderungen an virtuellen Pins) zu verarbeiten.

```
void loop() {
  Blynk.run();
  timer.run();
}
```

7. Steuerung des virtuellen Pins in Blynk:

   Hier lesen wir den Status des virtuellen Pins V0 aus der Blynk-App aus und steuern das Relais entsprechend. Ist der Schalter in der App aktiviert (d.h., V0 ist 1), wird der Relais-Pin auf HIGH gesetzt; ist er deaktiviert, wird er auf LOW gesetzt.

   - Jedes Mal, wenn der Wert eines virtuellen Pins auf dem BLYNK-Server ändert, wird `BLYNK_WRITE()` ausgelöst. Weitere Details unter .

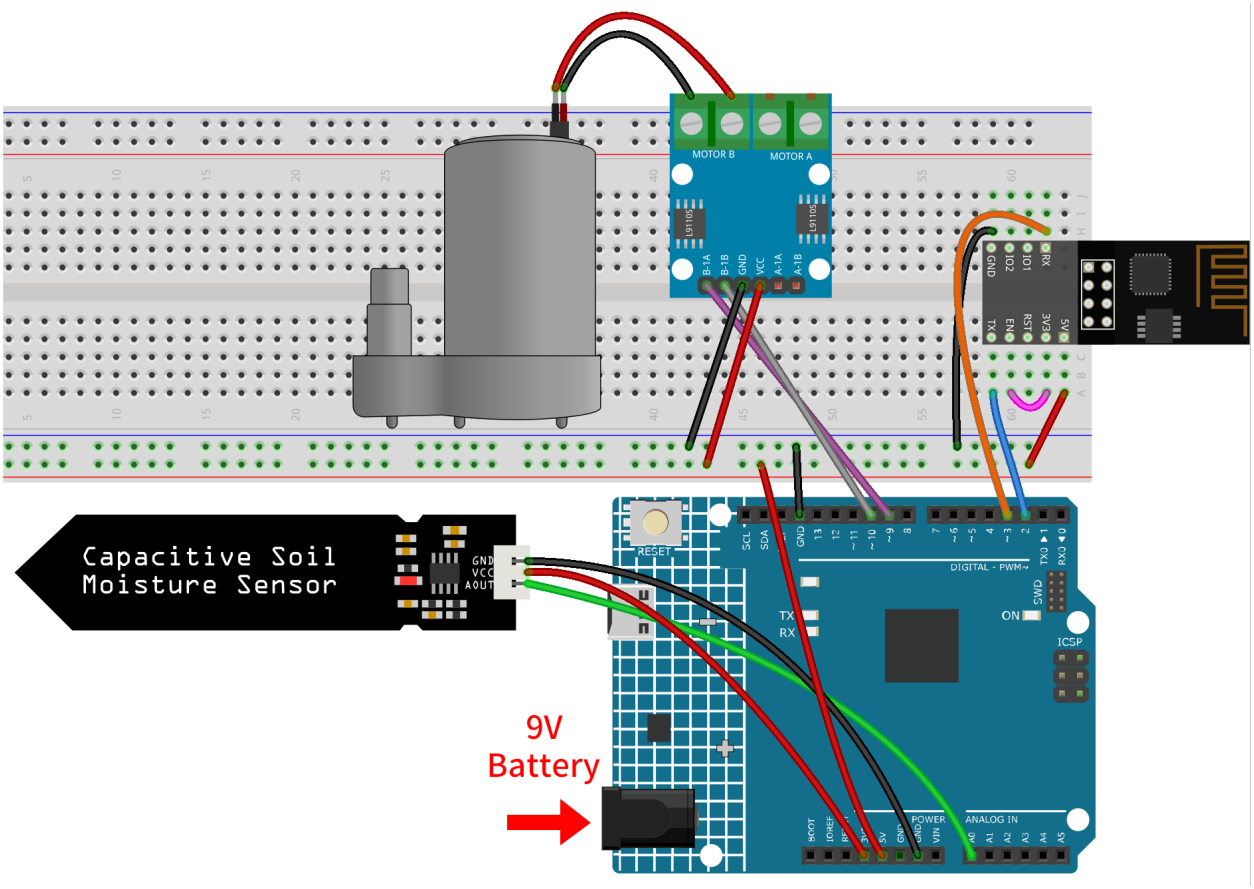
```
// This function is called every time the Virtual Pin 0 state changes
BLYNK_WRITE(V0) {
  switchStatus = param.asInt();  // Set incoming value from pin V0 to a variable

  if (switchStatus == 1) {
    Serial.println("The switch on Blynk has been turned on.");
    digitalWrite(RelayPin, HIGH);
  } else {
    Serial.println("The switch on Blynk has been turned off.");
    digitalWrite(RelayPin, LOW);
  }
}
```

2.5.6 Automatisches Bewässerungssystem mit Blynk

Dieses Projekt demonstriert ein automatisiertes Bewässerungssystem, das Bodenfeuchtesensoren und Wasserpumpen nutzt. Die Blynk-App dient der Benutzerinteraktion und empfängt Daten zur Bodenfeuchte, während sie den Bewässerungsschwellenwert und den Status des Automatikmodus an das System sendet. Wenn die Bodenfeuchte unter den eingestellten Schwellenwert im Automatikmodus fällt, aktiviert das System die Wasserpumpe.

1. Schaltung aufbauen

Bemerkung: Das ESP8266-Modul benötigt einen hohen Stromfluss für einen stabilen Betrieb. Stellen Sie daher sicher, dass die 9V-Batterie angeschlossen ist.



- *Arduino UNO R4 Minima-Platine*
- *ESP8266-Modul*
- *Kapazitives Bodenfeuchtigkeitsmodul*
- *Kreiselpumpe*

2. Blynk konfigurieren

Bemerkung: Wenn Sie mit Blynk noch nicht vertraut sind, empfehlen wir dringend, zunächst diese beiden Tutorials zu lesen. *Einführung in Blynk* ist ein Einsteigerleitfaden zu Blynk, der auch die Konfiguration des ESP8266 und die Registrierung bei Blynk beinhaltet. *Flammenwarnsystem mit Blynk* ist ein einfaches Beispiel, aber die Schritterklärungen sind detaillierter.

2.1 Vorlage erstellen

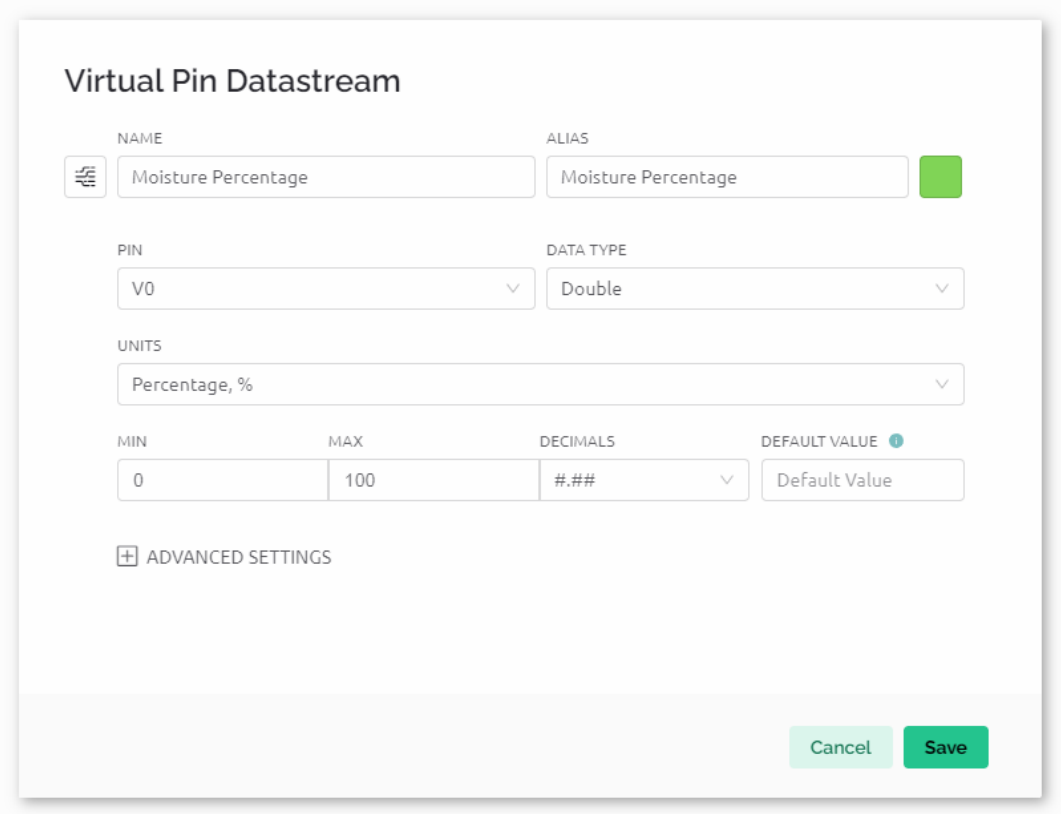
Zuerst müssen wir eine Vorlage in Blynk erstellen. Legen Sie eine **„Auto watering system“** Vorlage an.

2.2 Datenströme

Erstellen Sie **Datastreams** vom Typ **Virtual Pin** auf der **Datastreams**-Seite, um Daten vom ESP8266 und dem Uno R4 Board zu empfangen.

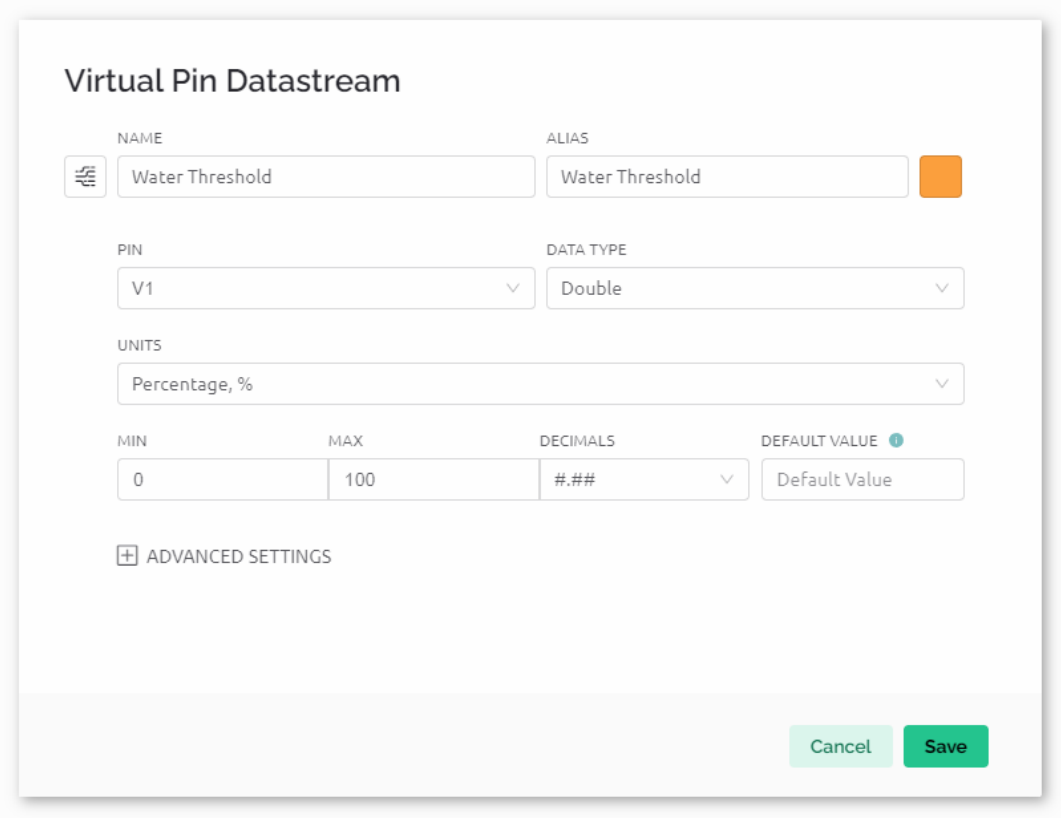
- Virtuellen Pin V0 nach folgendem Schema erstellen:

  Benennen Sie den **Virtual Pin V0** in **Moisture Percentage** um. Setzen Sie den **DATA TYPE** auf **Double** und MIN und MAX auf **0** und **100**. Die **UNITS** sollen **Percentage,%** sein.



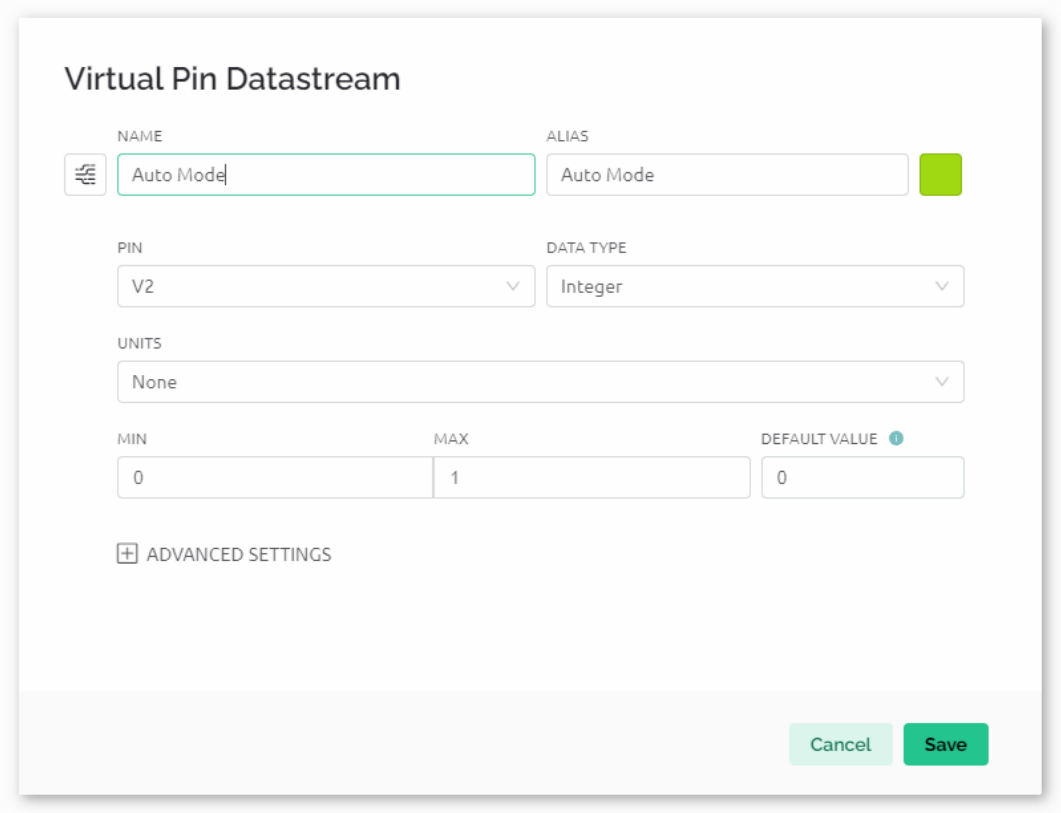
- Virtuellen Pin V1 nach folgendem Schema erstellen:

  Benennen Sie den **Virtual Pin V1** in **Water Threshold** um. Setzen Sie den **DATA TYPE** auf **Double** und MIN und MAX auf **0** und **100**. Die **UNITS** sollen **Percentage,%** sein.



- Virtuellen Pin V2 nach folgendem Schema erstellen:

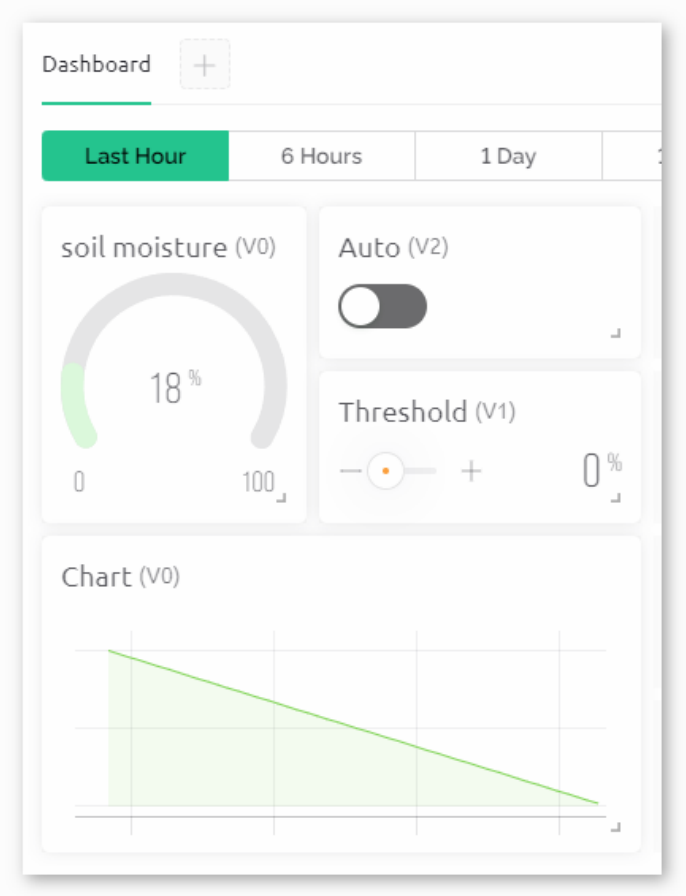
  Benennen Sie den **Virtual Pin V2** in **Auto Mode** um. Setzen Sie den **DATA TYPE** auf **Integer** und MIN und MAX auf **0** und **1**.



2.3 Web-Dashboard

Wir müssen auch das **Web-Dashboard** konfigurieren, um mit dem automatischen Bewässerungssystem interagieren zu können.

Richten Sie das Web-Dashboard entsprechend dem folgenden Schema ein. Wir verwenden Widgets wie Beschriftungen, Anzeigen, Schalter, Schieberegler und Diagramme. Achten Sie darauf, jedes Widget seinem entsprechenden virtuellen Pin zuzuordnen.



3. Den Code ausführen

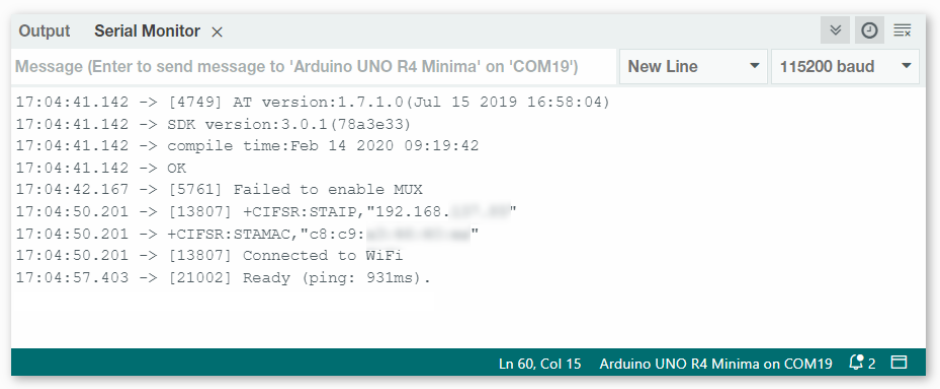
1. Öffnen Sie die Datei `07-Auto_watering_system.ino` im Pfad `ultimate-sensor-kit\iot_project\wifi\07-Auto_watering_system`, oder kopieren Sie diesen Code in die **Arduino IDE**.
2. Legen Sie ein Blynk-Gerät mit der Vorlage „Auto watering system“ an. Anschließend ersetzen Sie die Werte für `BLYNK_TEMPLATE_ID`, `BLYNK_TEMPLATE_NAME` und `BLYNK_AUTH_TOKEN` durch Ihre eigenen.

```
#define BLYNK_TEMPLATE_ID "TMPxxxxxxx"
#define BLYNK_TEMPLATE_NAME "Auto watering system"
#define BLYNK_AUTH_TOKEN "xxxxxxxxxxxxx"
```

3. Sie müssen auch die `ssid` und das `Passwort` des von Ihnen verwendeten WLANs eingeben.

```
char ssid[] = "Ihr_SSID";
char pass[] = "Ihr_Passwort";
```

4. Nach der Auswahl des richtigen Boards und Ports klicken Sie auf den **Hochladen**-Button.
5. Öffnen Sie den seriellen Monitor (Baudrate auf 115200 einstellen) und warten Sie auf eine Meldung, die eine erfolgreiche Verbindung anzeigt.



Bemerkung: Falls die Meldung `ESP is not responding` erscheint, befolgen Sie bitte diese Schritte.

- Stellen Sie sicher, dass die 9V-Batterie angeschlossen ist.
- Setzen Sie das ESP8266-Modul zurück, indem Sie den Pin RST für 1 Sekunde mit GND verbinden und dann wieder trennen.
- Drücken Sie den Reset-Knopf auf dem R4-Board.

Manchmal müssen Sie die oben genannten Schritte 3-5 Mal wiederholen. Bitte haben Sie Geduld.

4. Code-Erläuterung

1. **Einrichtung der Blynk-Cloud und Import der Bibliotheken**

   In diesen Zeilen werden die eindeutigen IDs und Tokens festgelegt, die zur Identifizierung und Authentifizierung Ihres Arduino-Geräts mit der Blynk-Cloud notwendig sind. Zusätzlich werden wichtige Bibliotheken für die Verwendung des ESP8266-WiFi-Moduls, Blynk-Funktionalitäten und Software-Serienschnittstelle eingebunden.

```
#define BLYNK_TEMPLATE_ID "TMPLxxxxxx"
#define BLYNK_TEMPLATE_NAME "Auto watering system"
#define BLYNK_AUTH_TOKEN "xxxxxxxxxxxxxxx"
#define BLYNK_PRINT Serial
#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
#include <SoftwareSerial.h>
```

2. **WiFi- und Hardware-Konfiguration**

   Hier werden die WLAN-Zugangsdaten (`ssid` und `pass`) definiert. Die Pins `2` (RX) und `3` (TX) sind für die Software-Serienschnittstelle zwischen dem Arduino und dem ESP8266 konfiguriert. Die Baudrate für diese Kommunikation ist auf `115200` festgelegt.

```
char ssid[] = "Ihr_WLAN-Name";
char pass[] = "Ihr_Passwort";
SoftwareSerial EspSerial(2, 3);
#define ESP8266_BAUD 115200
ESP8266 wifi(&EspSerial);
```

3. **Definition von Pins und globalen Variablen**

Die Pins für die Steuerung der Wasserpumpe und des Bodenfeuchtesensors werden hier zusammen mit globalen Variablen für Sensorwerte, Schwellenwerte und Betriebsmodi festgelegt.

Sie sollten Ihre eigenen Werte für `moistureInAir` und `moistureInWater` anhand Ihrer tatsächlichen Bedingungen messen. Um zu verhindern, dass der berechnete `moisturePercentage` den Bereich von 0-100% überschreitet, korrigieren Sie den Wert für `moistureInAir` nach oben und den Wert für `moistureInWater` nach unten, um für Sensorabweichungen zu kompensieren.

```
// Define pin configurations for the water pump
const int pump1A = 9;
const int pump1B = 10;
bool pumpStatus = 0;  // 0 indicates OFF, 1 indicates ON

// Define the soil moisture sensor
const float moistureInAir = 535;    // Measure by placing the sensor in air
const float moistureInWater = 280;  // Measure by immersing the sensor in water
const int sensorPin = A0;
int sensorValue = 0;  // Stores the raw sensor value

int autoMode = 0;
int waterThreshold = 0;        // The soil moisture percentage threshold to␣
↪activate watering
float moisturePercentage = 0;  // The calculated soil moisture percentage
```

4. **Anfangskonfigurationen in der setup()-Funktion**

   Hier werden zwei Timer konfiguriert:

   - Mit `timer.setInterval(10000L, updateDataTimer)` wird das Zeitintervall für die Funktion `updateDataTimer()` auf alle **10000ms** eingestellt. Sie können den ersten Parameter anpassen, um das Intervall zwischen den Ausführungen von `updateDataTimer()` zu ändern.
   - Mit `timer.setInterval(35000L, autoWaterTimer)` wird das Zeitintervall für die Funktion `autoWaterTimer()` auf alle **35000ms** eingestellt. Auch hier können Sie den ersten Parameter anpassen.

```
void setup() {
  pinMode(pump1A, OUTPUT);    // set pump1A as output
  pinMode(pump1B, OUTPUT);    // set pump1B as output
  digitalWrite(pump1B, LOW);  // Keep pump1B low

  Serial.begin(115200);           // Start serial communication at 115200 baud rate␣
↪for debugging
  EspSerial.begin(ESP8266_BAUD);  // Set ESP8266 baud rate
  delay(10);

  // Configure Blynk and connect to WiFi
  Blynk.config(wifi, BLYNK_AUTH_TOKEN);
  Blynk.connectWiFi(ssid, pass);

  // Configure timer events
  timer.setInterval(10000L, updateDataTimer);  // Update sensor data every 10␣
↪seconds
  timer.setInterval(35000L, autoWaterTimer);   // Check watering conditions every␣
↪35 seconds
}
```

5. **loop() Funktion**

   Diese kontinuierlich laufende Schleife ermöglicht es der Blynk-Bibliothek, nach Aktualisierungen zu suchen und die definierten Zeitereignisse zu verarbeiten.

```
void loop() {
   Blynk.run();
   timer.run();
}
```

6. **Interaktion mit der Blynk-App**

   Diese Funktionen werden durch bestimmte Interaktionen mit der Blynk-App ausgelöst:

   - `BLYNK_CONNECTED()`: Wird aufgerufen, wenn das Gerät eine Verbindung zu Blynk herstellt. Synchronisiert die Anfangszustände der virtuellen Pins.
   - `BLYNK_WRITE(V1)`: Wird ausgelöst, wenn sich der virtuelle Pin 1 ändert (Wasserschwellwert).
   - `BLYNK_WRITE(V2)`: Wird ausgelöst, wenn sich der virtuelle Pin 2 ändert (Automatik-Modus Status).

```
// This function is called every time the device is connected to the Blynk.Cloud
BLYNK_CONNECTED() {
  Blynk.syncVirtual(V1);  // Sync water threshold
  Blynk.syncVirtual(V2);  // Sync auto mode status
}

// This function is called every time the Virtual Pin 1 state changes
BLYNK_WRITE(V1) {
  waterThreshold = param.asInt();  // Update watering threshold
  Serial.print("Received threshold.   waterThreshold:");
  Serial.println(waterThreshold);
}

// This function is called every time the Virtual Pin 2 state changes
BLYNK_WRITE(V2) {
  autoMode = param.asInt();  // Update auto mode status

  if (autoMode == 1) {
    Serial.println("The switch on Blynk has been turned on.");
  } else {
    Serial.println("The switch on Blynk has been turned off.");
  }
}
```

7. **Timer-Callbacks und automatische Bewässerungslogik**

   Diese Funktionen steuern die Aufgaben, die von den Timern ausgeführt werden:

   - `updateDataTimer()`: Ruft `sendData()` auf, um aktuelle Feuchtigkeitsdaten an Blynk zu senden.
   - `autoWaterTimer()`: Ruft `autoWater()` auf, um zu prüfen, ob Bewässerung erforderlich ist.
   - `sendData()`: Berechnet den Bodenfeuchtigkeitsprozentsatz, protokolliert ihn und sendet ihn an die Blynk-App.
   - `autoWater()`: Überprüft, ob der Boden auf Basis des festgelegten Schwellwerts und des Automatik-Modus bewässert werden muss.

```
void updateDataTimer() {
  sendData();
}

void autoWaterTimer() {
  autoWater();
}

// Function to send sensor data to Blynk app
void sendData() {
  // Calculate soil moisture percentage
  sensorValue = analogRead(sensorPin);
  moisturePercentage = 1 - (sensorValue - moistureInWater) / (moistureInAir -␣
↪moistureInWater);

  Serial.println("-----------------------------");
  Serial.println("Update soil moisture data ...");
  Serial.print("sensorValue:");
  Serial.print(sensorValue);
  Serial.print("  moisturePercentage:");
  Serial.println(moisturePercentage * 100);

  // Send moisture percentage to Blynk app
  Blynk.virtualWrite(V0, moisturePercentage * 100);
}

// Function to control automatic watering based on soil moisture and user settings
void autoWater() {
  if (autoMode == 1 && moisturePercentage * 100 < waterThreshold) {

    if (!pumpStatus) {
      turnOnPump();
      Serial.println("-----------------------------");
      Serial.println("Watering...");

      // Turn off pump after 2 seconds
      timer.setTimeout(2000L, turnOffPump);
    }
  }
}
```

8. **Pumpensteuerungsfunktionen**

   These functions directly control the operation of the water pump:

   - `turnOnPump()`: Activates the pump.
   - `turnOffPump()`: Deactivates the pump.

```
// Function to turn on the water pump
void turnOnPump() {
  digitalWrite(pump1A, HIGH);
  pumpStatus = 1;
}
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
// Function to turn off the water pump
void turnOffPump() {
  digitalWrite(pump1A, LOW);
  pumpStatus = 0;
}
```

Referenzen

-
-
-
-

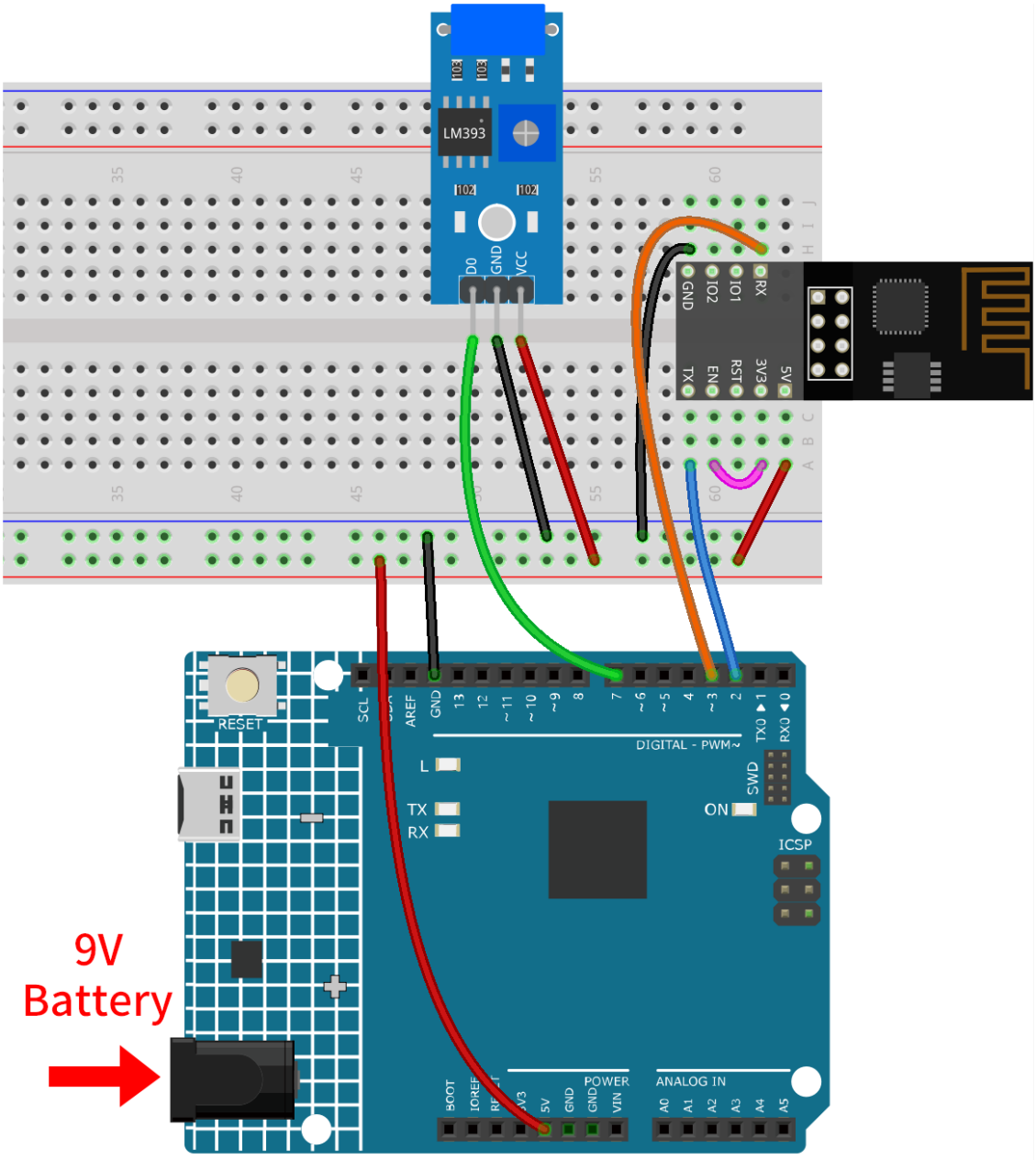
2.5.7 Vibrationserkennungssystem mit IFTTT

Dieses Projekt implementiert ein Vibrationserkennungssystem mit Hilfe eines Arduino-Boards (Uno R4 oder R3), eines ESP8266-Moduls und eines Vibrationssensors (SW-420). Bei erkannter Vibration sendet das System eine HTTP-Anfrage an einen IFTTT-Server und kann dadurch diverse Aktionen auslösen, wie zum Beispiel das Versenden einer Benachrichtigung oder einer E-Mail.

Um übermäßige Warnmeldungen in kurzer Zeit zu vermeiden, ist das System so programmiert, dass diese HTTP-Anfragen nur im Mindestabstand von 2 Minuten (120.000 Millisekunden) gesendet werden. Dieses Intervall kann nach Bedarf angepasst werden.

1. Den Schaltkreis aufbauen

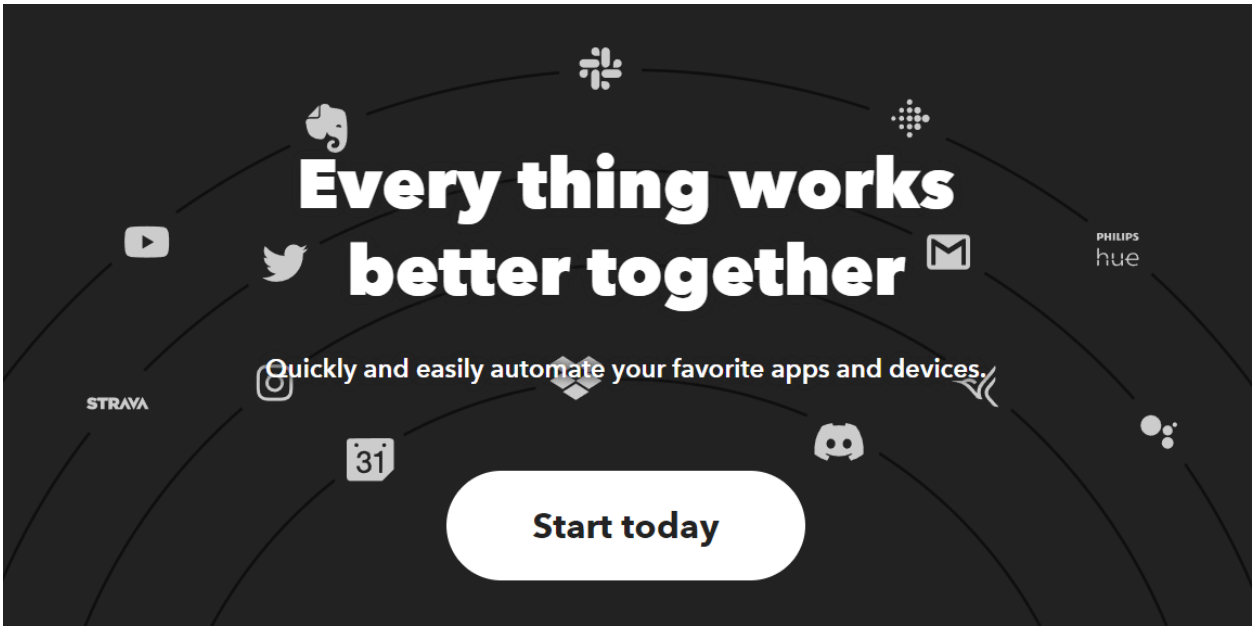
Bemerkung: Das ESP8266-Modul benötigt einen hohen Stromfluss, um stabil zu funktionieren. Stellen Sie daher sicher, dass die 9-V-Batterie angeschlossen ist.



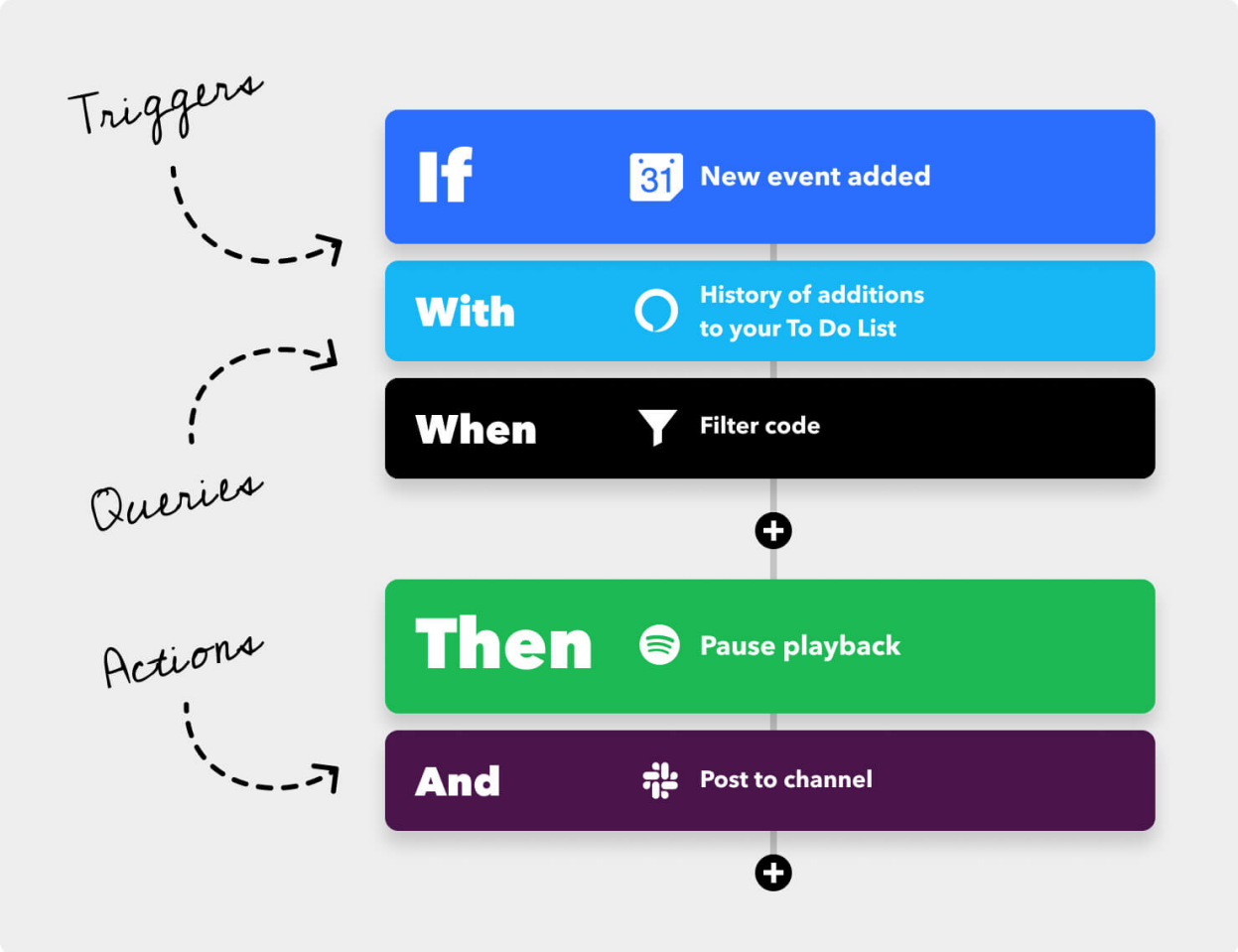
- *Arduino UNO R4 Minima-Platine*
- *ESP8266-Modul*
- *Vibrationssensor-Modul (SW-420)*

2. IFTTT konfigurieren

ist ein privates Handelsunternehmen, das 2011 gegründet wurde und digitale Automatisierungsplattformen als Dienstleistung anbietet. Die Plattform ermöglicht es den Nutzern, mit einer visuellen Oberfläche plattformübergreifende „Wenn-Dann“-Bedingungen zu erstellen. Stand 2020 zählt die Plattform 18 Millionen Nutzer.

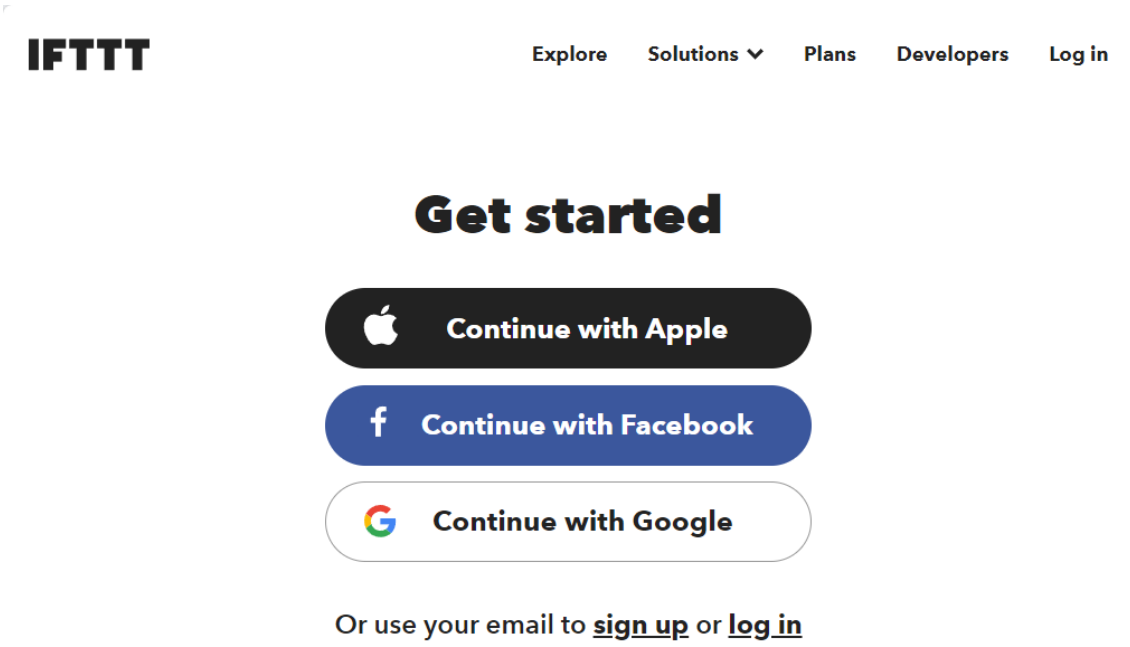


IFTTT steht für „If This Then That“. Einfach gesagt: Wenn bestimmte Bedingungen erfüllt sind, wird eine entsprechende Aktion ausgelöst. Der „If This“-Teil wird Trigger genannt und der „Then That“-Teil wird als Aktion bezeichnet. IFTTT verbindet Smart-Home-Geräte, soziale Medien, Liefer-Apps und vieles mehr, um automatisierte Aufgaben durchzuführen.

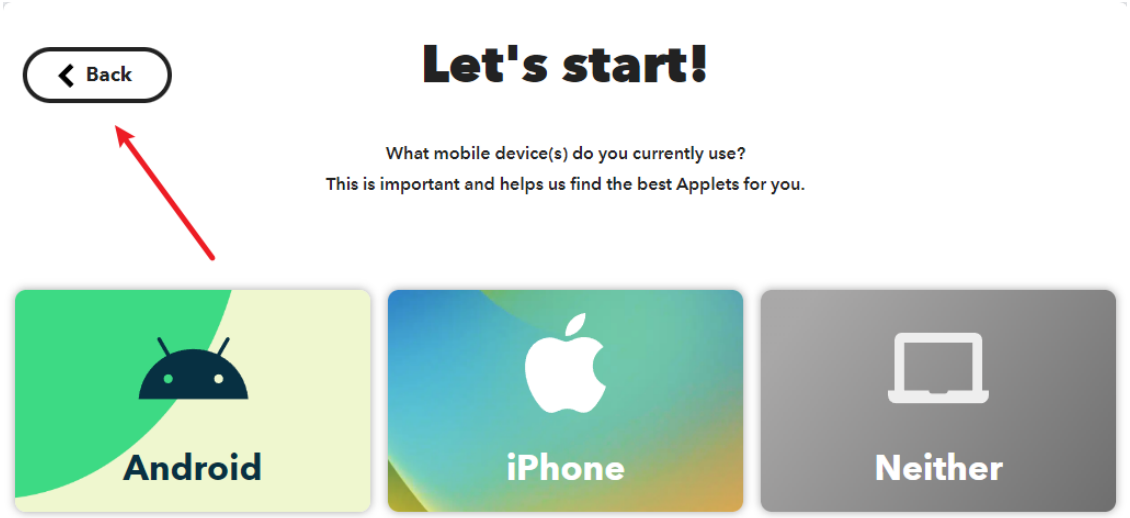


2.1 Bei IFTTT anmelden

Geben Sie „https://ifttt.com“ in Ihren Browser ein und klicken Sie auf den zentral platzierten Button „Get started“. Füllen Sie das Formular mit Ihren Informationen aus, um ein Konto zu erstellen.

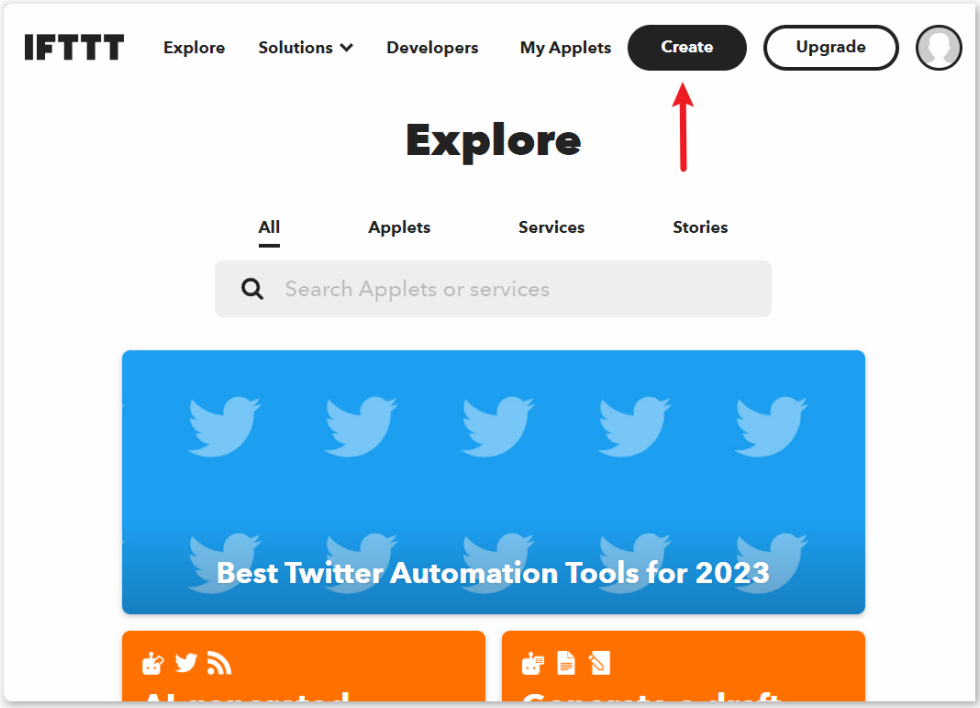


Klicken Sie auf „Back“, um den Schnellstart zu verlassen, kehren Sie zur IFTTT-Startseite zurück, aktualisieren Sie die Seite und melden Sie sich erneut an.



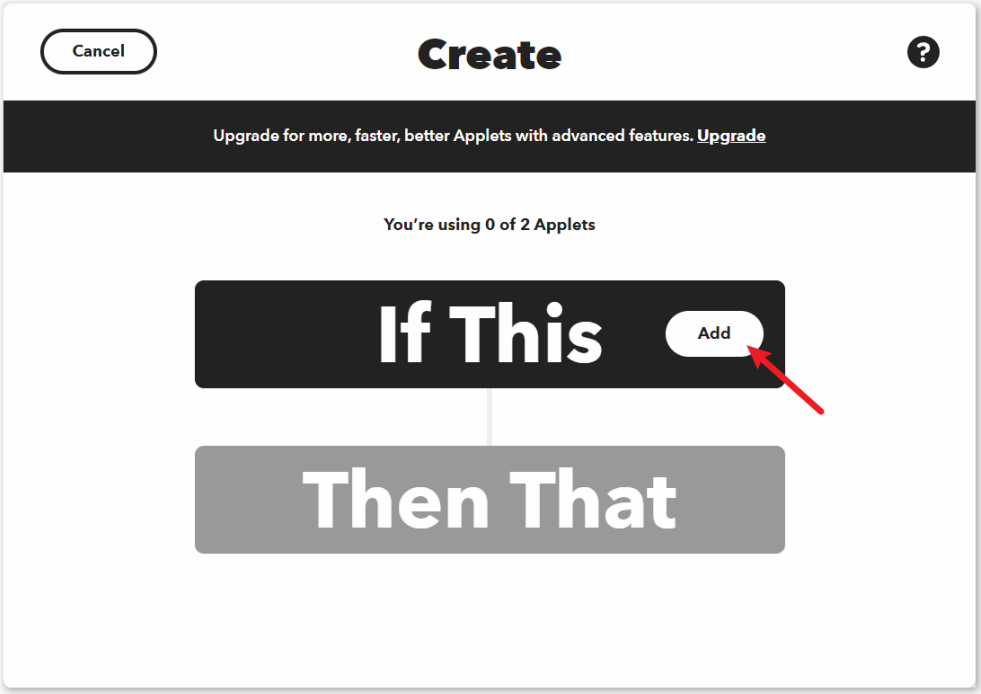
2.2 Das Applet erstellen

Klicken Sie auf „Create“, um mit der Erstellung des Applets zu beginnen.

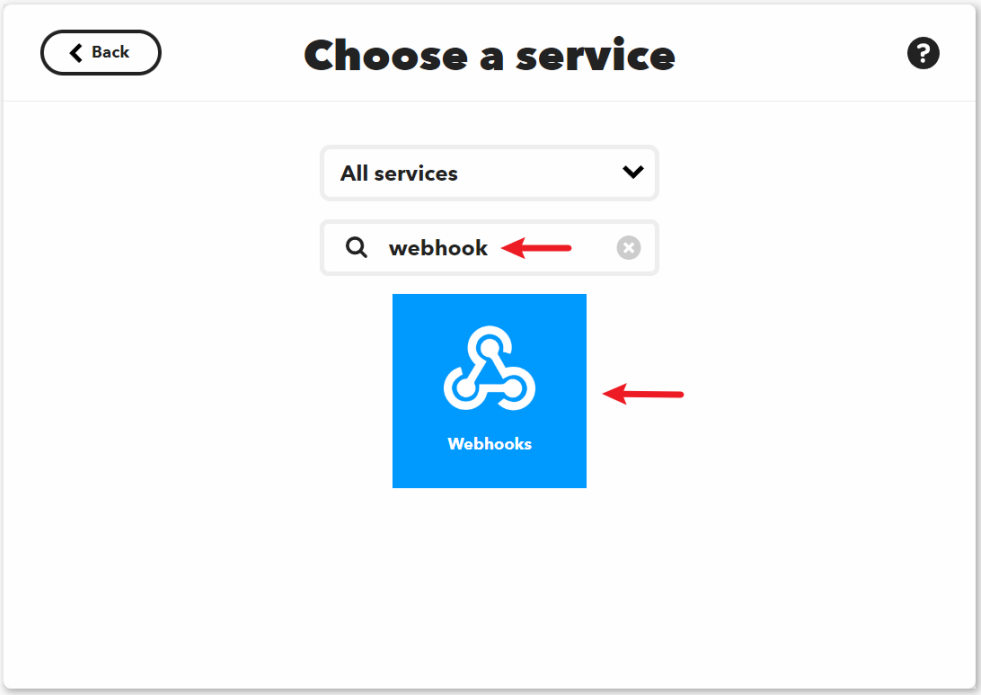


Wenn dieser Auslöser

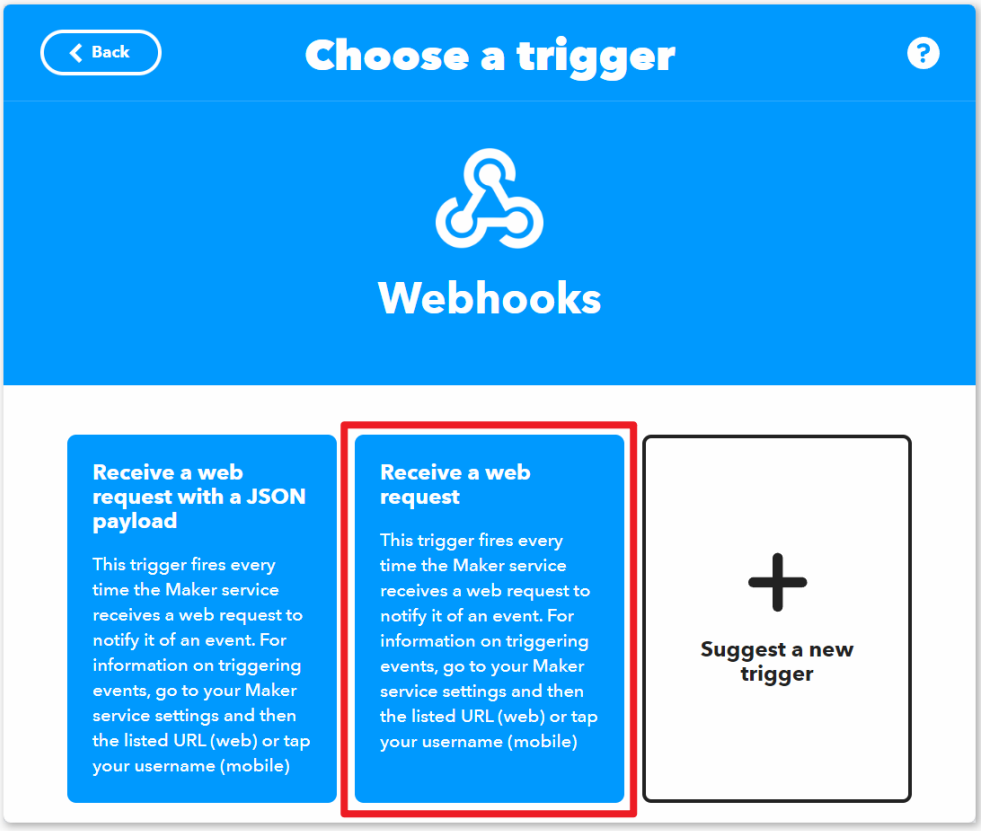
Klicken Sie neben „If This“ auf „Add“, um einen Auslöser hinzuzufügen.



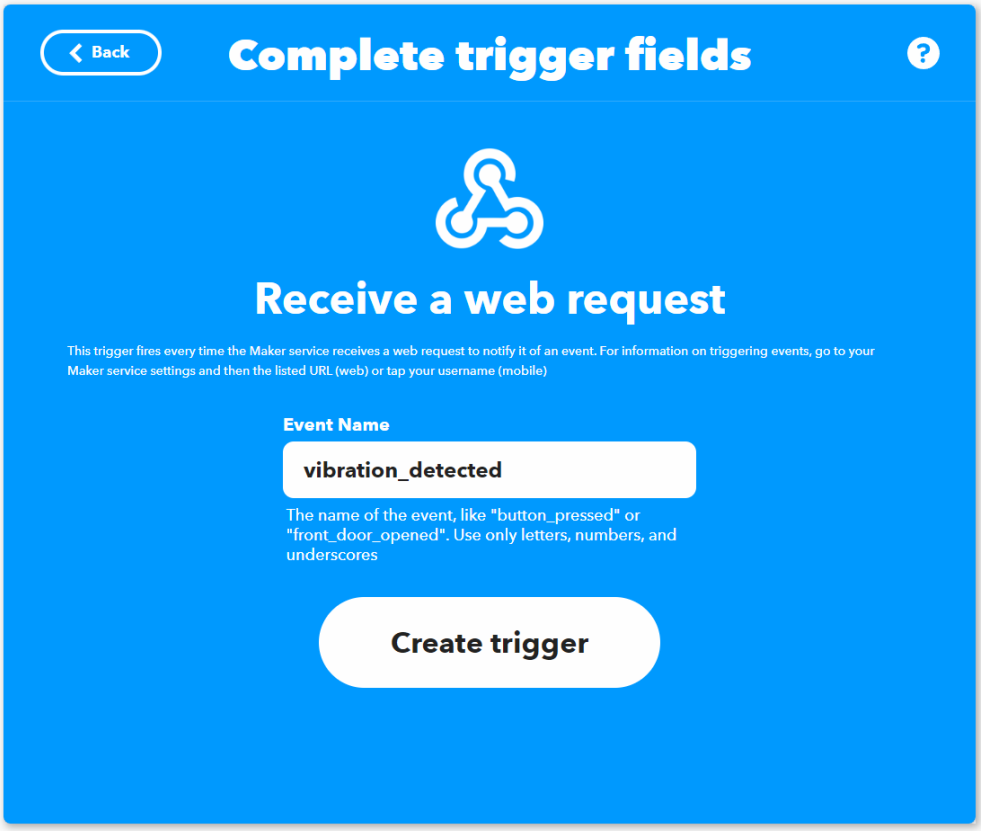
Suchen Sie nach „Webhook“ und klicken Sie darauf.



Klicken Sie auf der folgenden Seite auf „Receive a web request“.

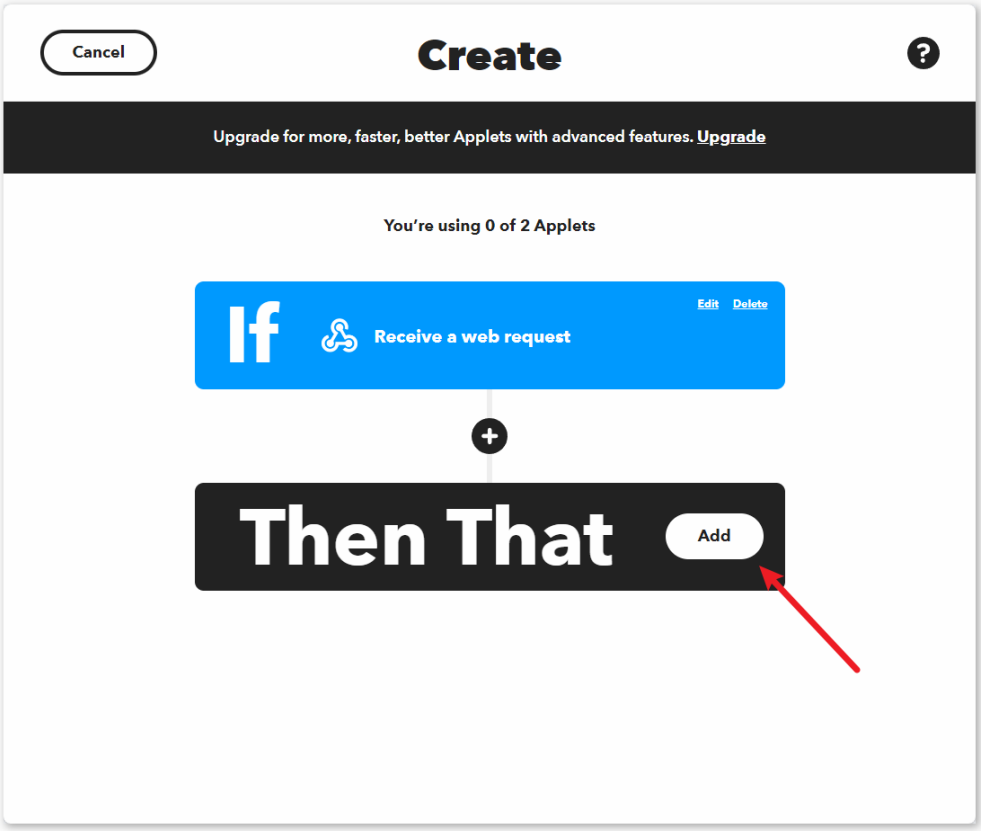


Setzen Sie den „Event Name“ auf „vibration_detected“.

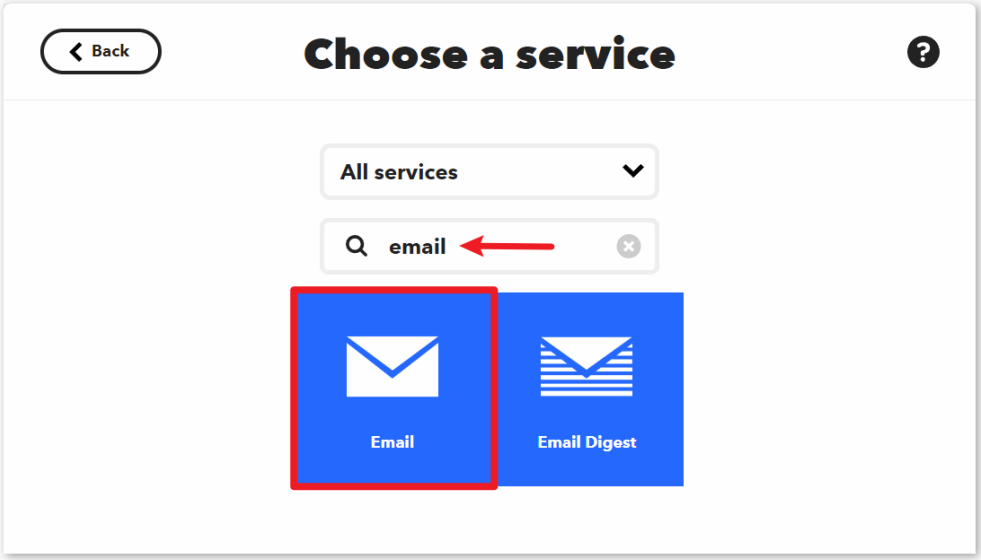


Then That action

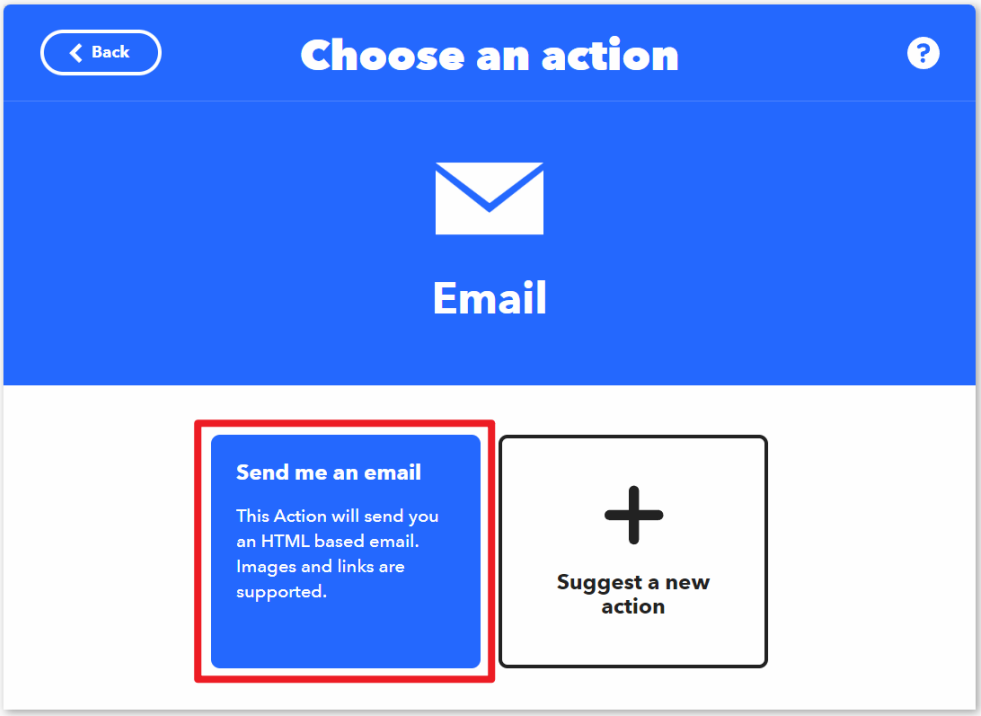
Klicken Sie neben „Then That“ auf „Add“, um eine Aktion hinzuzufügen.



Suchen Sie nach „email“ und klicken Sie darauf.

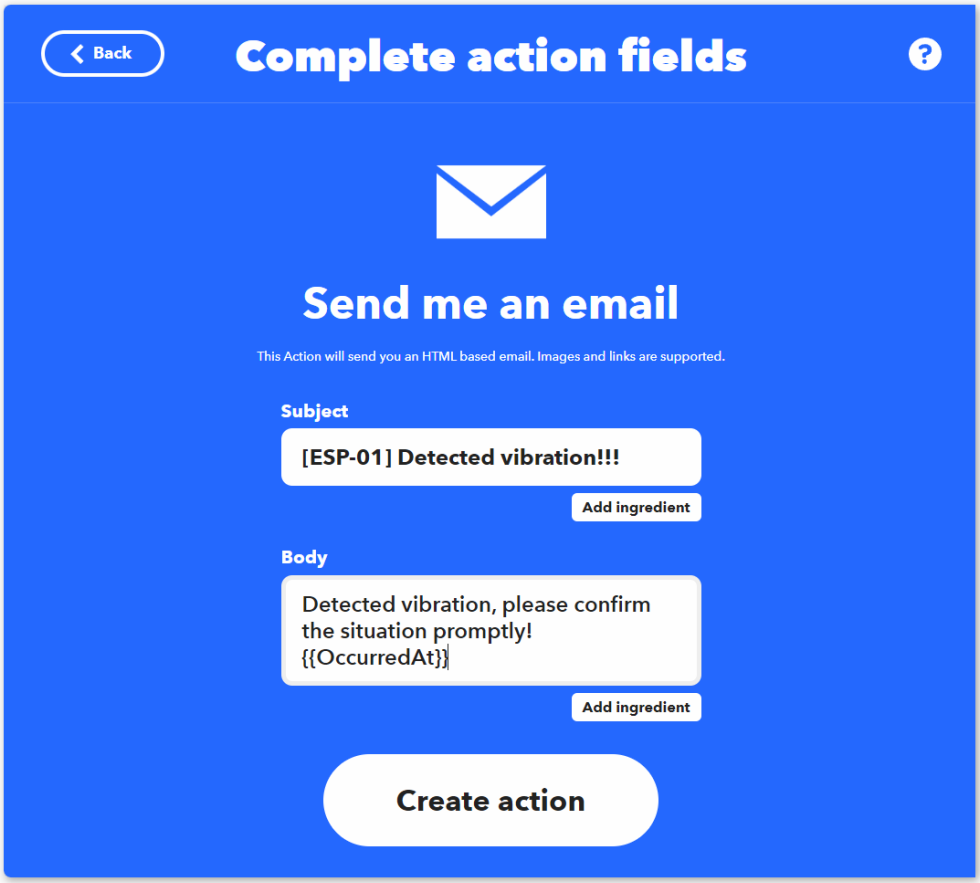


Klicken Sie auf der folgenden Seite auf „Send me a email“.

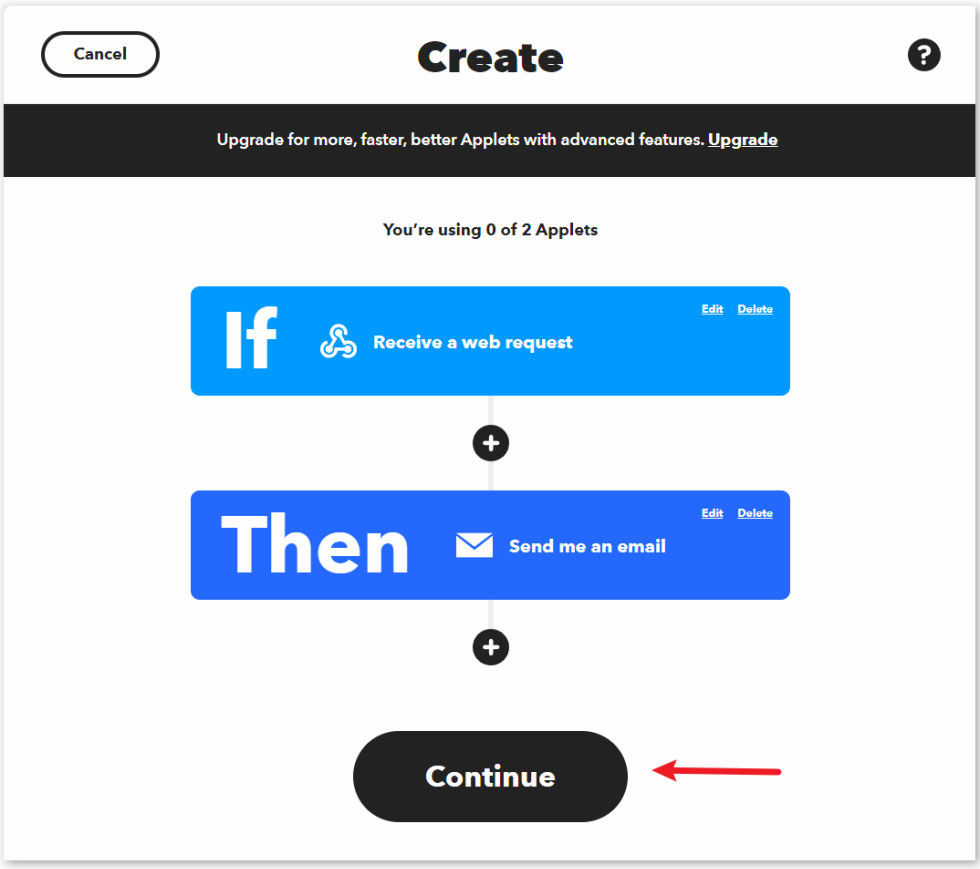


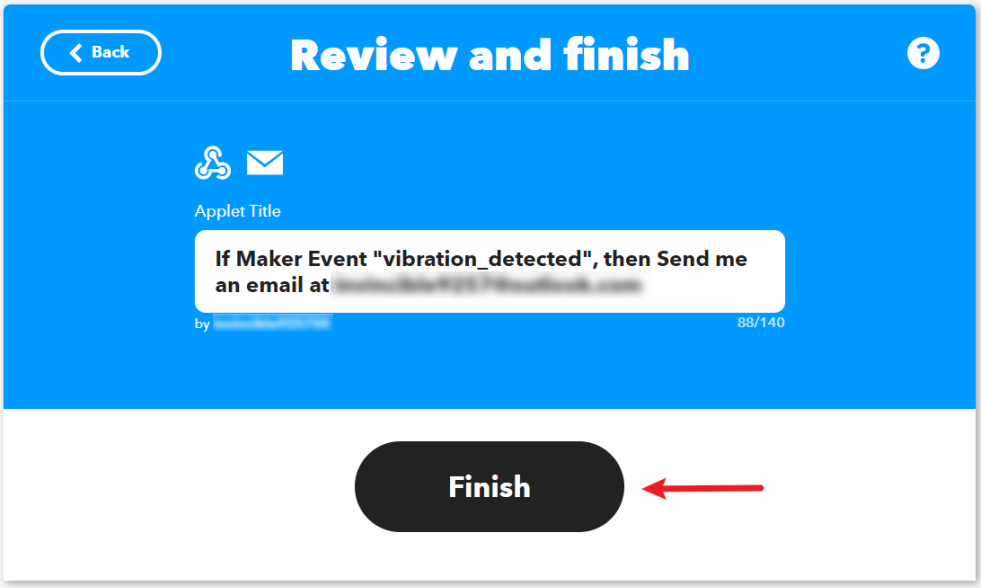
Legen Sie den Betreff und Inhalt der E-Mail fest, die bei erkannter Vibration versendet wird.

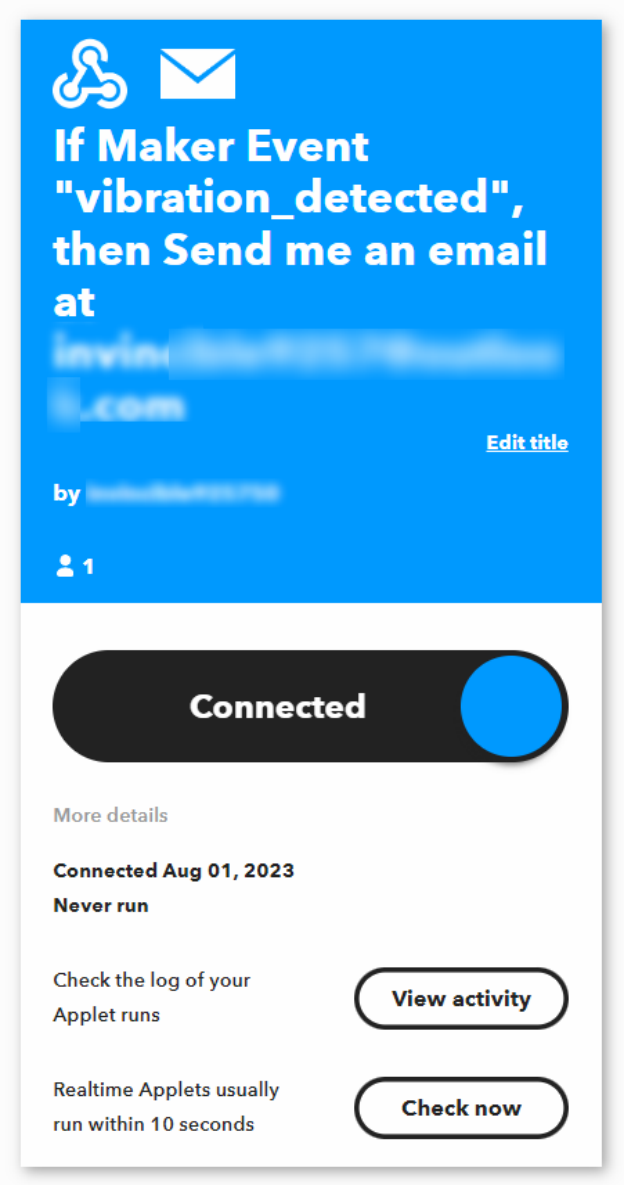
Als Referenz: Der Betreff ist auf „[ESP-01] Detected vibration!!!“ gesetzt und der Inhalt lautet „Detected vibration, please confirm the situation promptly! {{OccurredAt}}“. Beim Versand der E-Mail wird `{{OccurredAt}}` automatisch durch den Zeitpunkt des Ereignisses ersetzt.



Befolgen Sie die folgenden Schritte, um die Erstellung des Applets abzuschließen.


Cancel
Create
Upgrade for more, faster, better Applets with advanced features. Upgrade
You're using 0 of 2 Applets
If
Receive a web request
Edit
Delete
Then
Send me an email
Edit
Delete
Continue


Back
Review and finish
Applet Title
If Maker Event "vibration_detected", then Send me an email at
by
88/140
Finish



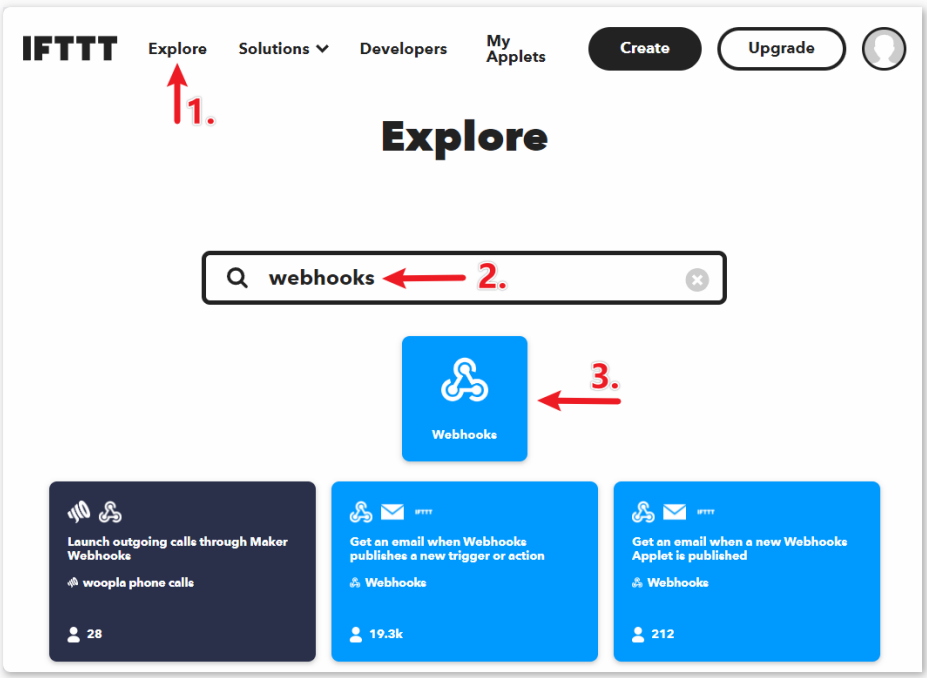
3. Code ausführen

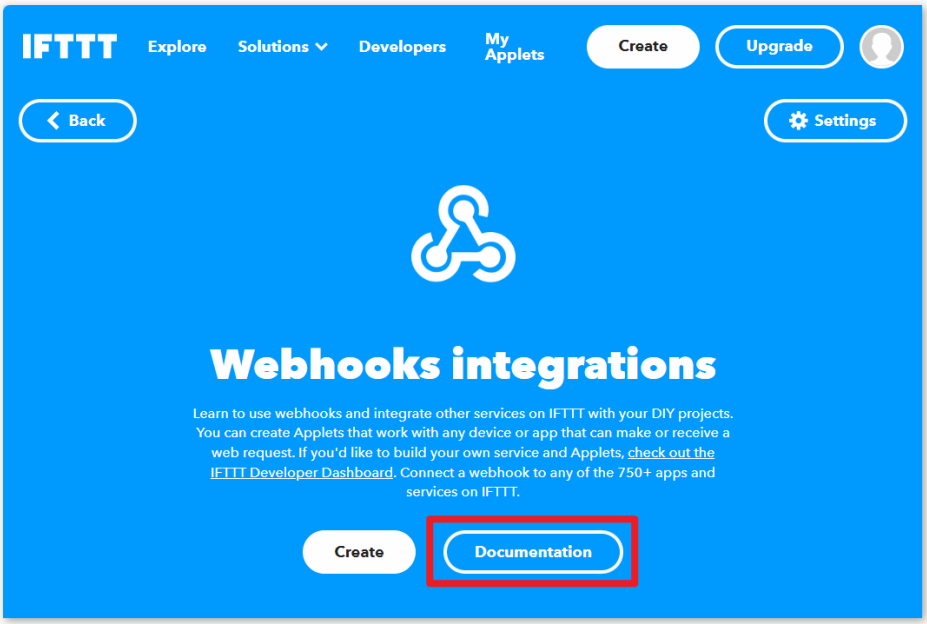
1. Öffnen Sie die Datei `04-Vibration_alert_system.ino` im Verzeichnis `ultimate-sensor-kit\iot_project\wifi\04-Vibration_alert_system`, oder kopieren Sie den folgenden Code in die **Arduino IDE**.
2. Tragen Sie die `mySSID` und das `myPWD` des verwendeten WLANs ein.

```
String mySSID = "Ihr_SSID";     // WLAN-SSID
String myPWD = "Ihr_Passwort";  // WLAN-Passwort
```

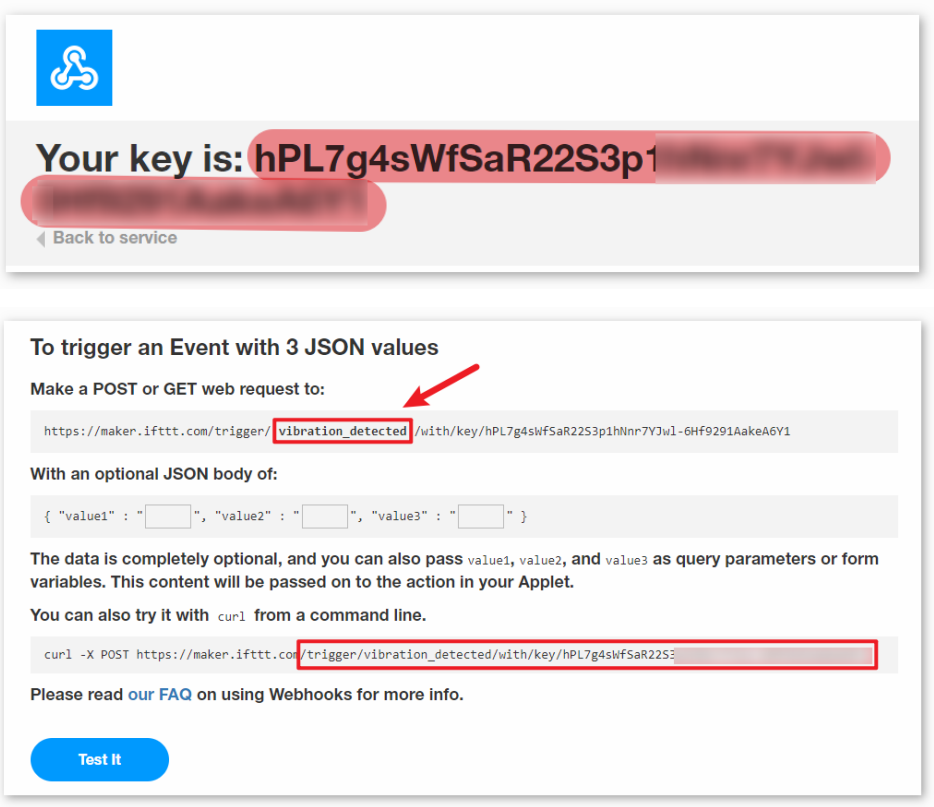
3. Sie müssen auch die `URL` anpassen. Hierbei sollten Sie sowohl den von Ihnen festgelegten Ereignisnamen als auch Ihren API-Schlüssel eintragen.

```
String URL = "/trigger/vibration_detected/with/key/xxxxxxxxxxxxxxxxx";
```

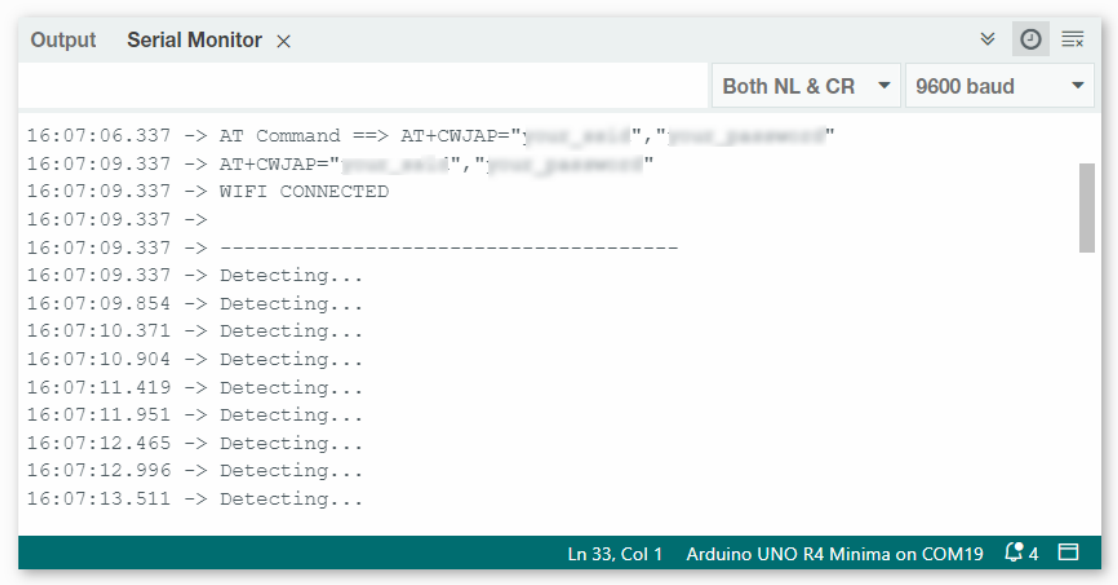




An dieser Stelle finden Sie **Ihren persönlichen API-Schlüssel, den Sie unbedingt geheim halten sollten**. Geben Sie den Ereignisnamen als `vibration_erkannt` ein. Die vollständige URL wird am unteren Ende der Webseite angezeigt. Kopieren Sie diese URL.



4. Wählen Sie das korrekte Board und den passenden Port aus und klicken Sie auf die Schaltfläche **Hochladen**.
5. Öffnen Sie den seriellen Monitor (Baudrate auf **9600** einstellen) und warten Sie auf eine entsprechende Meldung, die den erfolgreichen Verbindungsaufbau anzeigt.



4. Code-Erklärung

Das ESP8266-Modul, das im Kit enthalten ist, ist bereits mit der AT-Firmware vorinstalliert. Daher kann das ESP8266-Modul über AT-Befehle gesteuert werden. In diesem Projekt verwenden wir die SoftwareSerial-Bibliothek, um die Kommunikation zwischen dem Arduino Uno Board und dem ESP8266-Modul zu ermöglichen. Das Arduino Uno Board sendet AT-Befehle an das ESP8266-Modul, um eine Netzwerkverbindung herzustellen und Anfragen zu senden. Weitere Informationen finden Sie unter .

Das Uno-Board liest Sensordaten und sendet AT-Befehle an das ESP8266-Modul, welches sich dann mit dem Netzwerk verbindet und Anfragen an die IFTTT-Server sendet.

1. Einbindung der SoftwareSerial-Bibliothek für die serielle Kommunikation zwischen Arduino und ESP8266

```
#include <SoftwareSerial.h>
SoftwareSerial espSerial(2, 3);
```

2. Konfiguration der WLAN-Zugangsdaten und IFTTT-Serverdetails

```
String mySSID = "Ihre_SSID";
String myPWD = "Ihr_Passwort";
String myHOST = "maker.ifttt.com";
String myPORT = "80";
String URL = "/trigger/xxx/with/key/xxxx";
```

3. Definition der Variablen für den Vibrationssensor und die Steuerung der Alarmfrequenz

```
unsigned long lastAlertTime = 0;
const unsigned long postingInterval = 120000L;
const int sensorPin = 7;
```

4. In der `setup()`-Methode Initialisierung der seriellen Kommunikation, des ESP8266-Moduls und Verbindung zum WLAN herstellen

```
void setup() {
  Serial.begin(9600);
  espSerial.begin(115200);

  // Initialize the ESP8266 module
  sendATCommand("AT+RST", 1000, DEBUG);   //Reset the ESP8266 module
  sendATCommand("AT+CWMODE=1", 1000, DEBUG);  //Set the ESP mode as station mode
  sendATCommand("AT+CWJAP=\"" + mySSID + "\",\"" + myPWD + "\"", 3000, DEBUG);  //
↪Connect to WiFi network

  while (!espSerial.find("OK")) {
    //Wait for connection
  }
}
```

5. In der `loop()`-Methode Vibrationen erkennen und Alarm senden, falls das Zeitintervall überschritten wurde

```
void loop() {

  if (digitalRead(sensorPin)) {
    if (lastAlertTime == 0 || millis() - lastAlertTime > postingInterval) {
      Serial.println("Detected vibration!!!");
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
      sendAlert();  //Send an HTTP request to IFTTT server
    } else {
      Serial.print("Detected vibration!!! ");
      Serial.println("Since an email has been sent recently, no warning email will␣
↪be sent this time to avoid bombarding your inbox.");
    }
  } else {
    if (DEBUG) {
      Serial.println("Detecting...");
    }
  }
  delay(500);
}
```

6. Die Funktion sendAlert() erstellt eine HTTP-Anfrage und sendet sie über das ESP8266-Modul

```
void sendAlert() {

  String sendData = "GET " + URL + " HTTP/1.1" + "\r\n";
  sendData += "Host: maker.ifttt.com\r\n";

  sendATCommand("AT+CIPMUX=0",1000,DEBUG);
  sendATCommand("AT+CIPSTART=...",3000,DEBUG);
  sendATCommand("AT+CIPSEND=" + String(sendData.length()),1000,DEBUG);
  espSerial.println(sendData);

}
```

7. Behandlung der AT-Befehle mit der Methode sendATCommand()

   Diese Funktion sendet AT-Befehle an das ESP8266-Modul und sammelt die Antworten.

```
void sendATCommand(String command, const int timeout, boolean debug) {
  // Print and send command
  Serial.print("AT Command ==> ");
  Serial.print(command);
  Serial.println();
  espSerial.println(command);  // Send the AT command

  // Get the response from the ESP8266 module
  String response = "";
  long int time = millis();
  while ((time + timeout) > millis()) {  // Wait for the response until the timeout
    while (espSerial.available()) {
      char c = espSerial.read();
      response += c;
    }
  }

  // Print response if debug mode is on
  if (debug) {
    Serial.println(response);
    Serial.println("----------------------------------------");
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
}
```

Referenzen

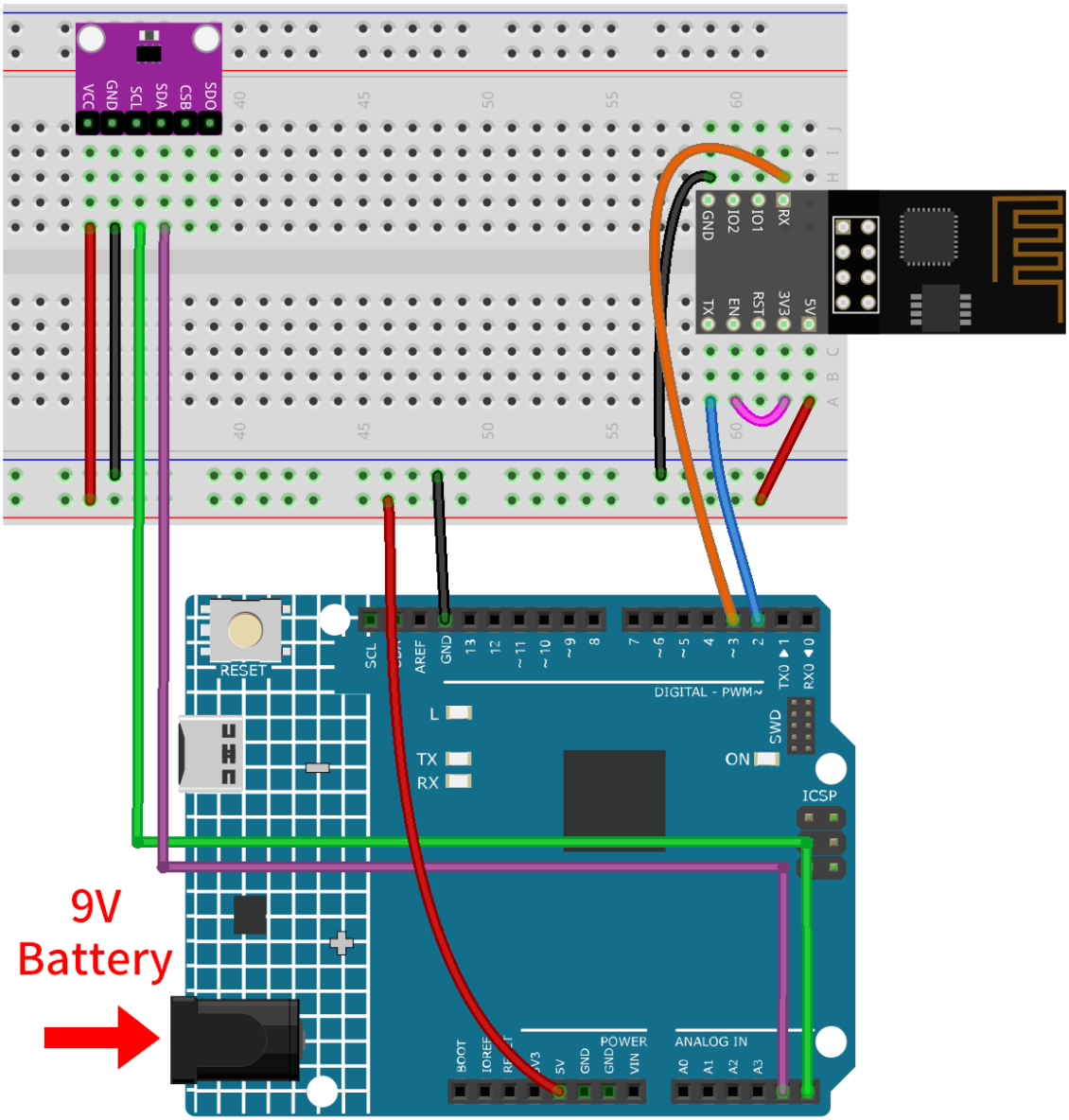
-
-
-

2.5.8 Wetterüberwachung mit ThingSpeak

Dieses Projekt erfasst Temperatur- und Druckdaten mithilfe eines Atmosphärendrucksensors. Die gesammelten Daten werden in regelmäßigen Abständen über ein ESP8266-Modul und ein WLAN-Netzwerk an die ThingSpeak-Cloud-Plattform übertragen.

1. Schaltung aufbauen

Bemerkung: Das ESP8266-Modul benötigt einen hohen Stromfluss, um eine stabile Betriebsumgebung zu gewährleisten. Stellen Sie daher sicher, dass die 9-V-Batterie angeschlossen ist.



- *Arduino UNO R4 Minima-Platine*
- *ESP8266-Modul*
- *Temperatur-, Feuchtigkeits- & Drucksensor (BMP280)*

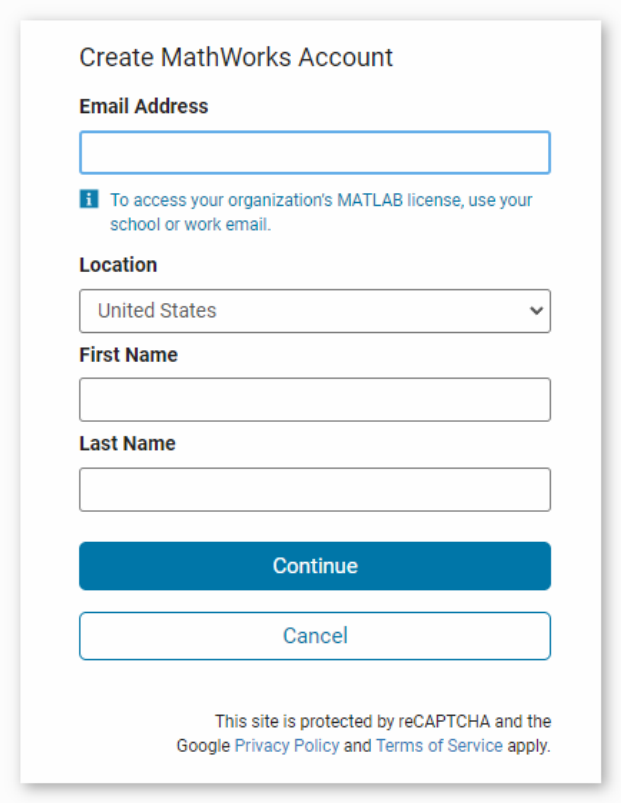
2. ThingSpeak konfigurieren

™ ist eine IoT-Analyseplattform, die die Aggregation, Visualisierung und Analyse von Live-Datenströmen in der Cloud ermöglicht. ThingSpeak bietet sofortige Visualisierungen von Daten, die von Ihren Geräten an ThingSpeak gesendet werden. Mit der Möglichkeit, MATLAB®-Code in ThingSpeak auszuführen, können Sie eine Echtzeitanalyse und -verarbeitung der eingehenden Daten durchführen. ThingSpeak wird häufig für Prototypen und Machbarkeitsnachweise von IoT-Systemen mit Analyseanforderungen eingesetzt.

2.1 ThingSpeak-Konto erstellen

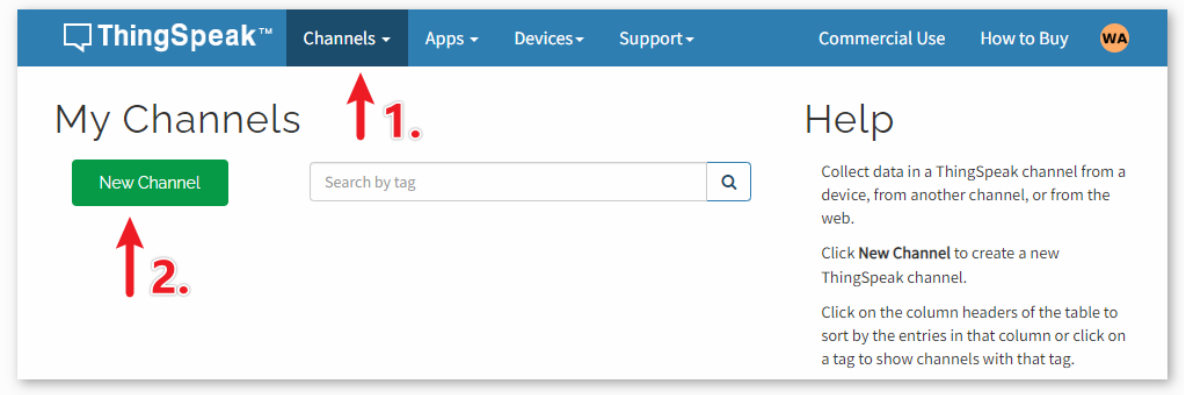
Als Erstes müssen Sie ein Konto bei ThingSpeak erstellen. Da es eine Zusammenarbeit mit MATLAB gibt, können Sie sich mit Ihren MathWorks-Anmeldedaten bei anmelden.

Wenn Sie noch keine haben, müssen Sie ein Konto bei MathWorks erstellen und sich bei der ThingSpeak-Anwendung anmelden.



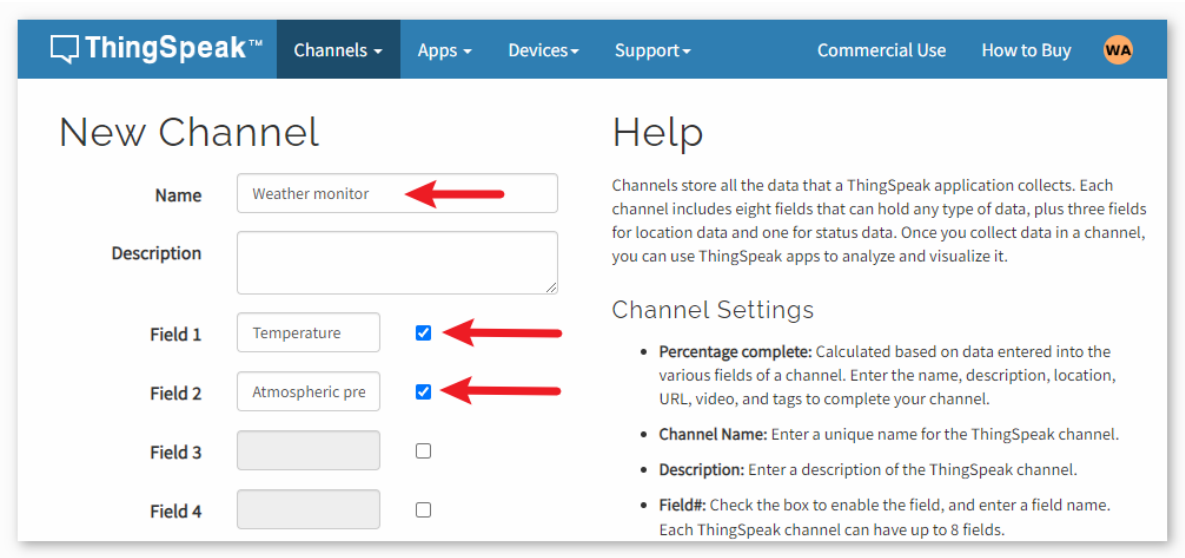
2.2 Einen Kanal erstellen

Nach der Anmeldung erstellen Sie einen neuen Kanal zur Datenspeicherung, indem Sie zu „Channels“ > „My Channels“ gehen und auf „New Channel“ klicken.



Für dieses Projekt müssen wir einen Kanal namens „**Weather Monitor**“ mit zwei Feldern erstellen: **Field 1** für

„**Temperature**“ und **Field 2** für „**Atmospheric Pressure**“.



3. Den Code ausführen

1. Öffnen Sie die Datei `05-Weather_monitor.ino` im Verzeichnispfad `ultimate-sensor-kit\iot_project\wifi\05-Weather_monitor` oder kopieren Sie diesen Code in die **Arduino IDE**.

Bemerkung: Um die Bibliothek zu installieren, nutzen Sie den Arduino-Bibliotheksmanager und suchen Sie nach **„Adafruit BMP280“**, um sie zu installieren.

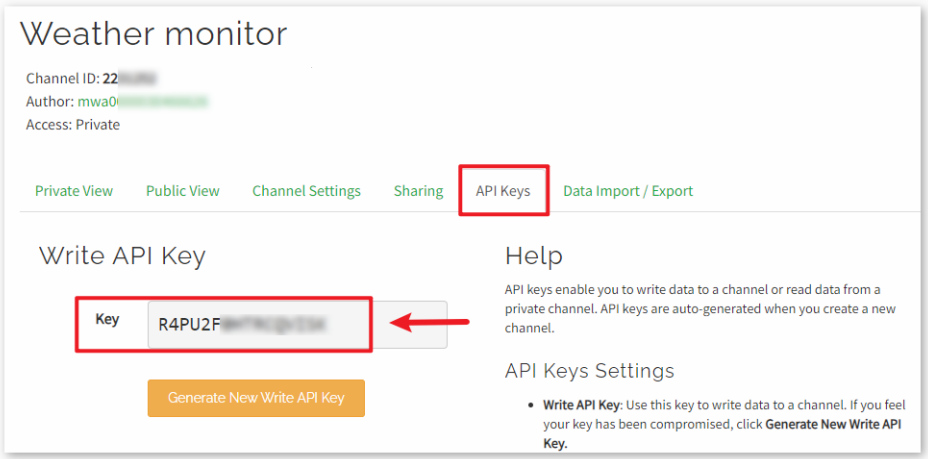
---

2. Sie müssen die `mySSID` und `myPWD` des von Ihnen verwendeten WLANs eingeben.

```
String mySSID = "Ihre_SSID";     // WiFi SSID
String myPWD = "Ihr_Passwort";  // WiFi-Passwort
```

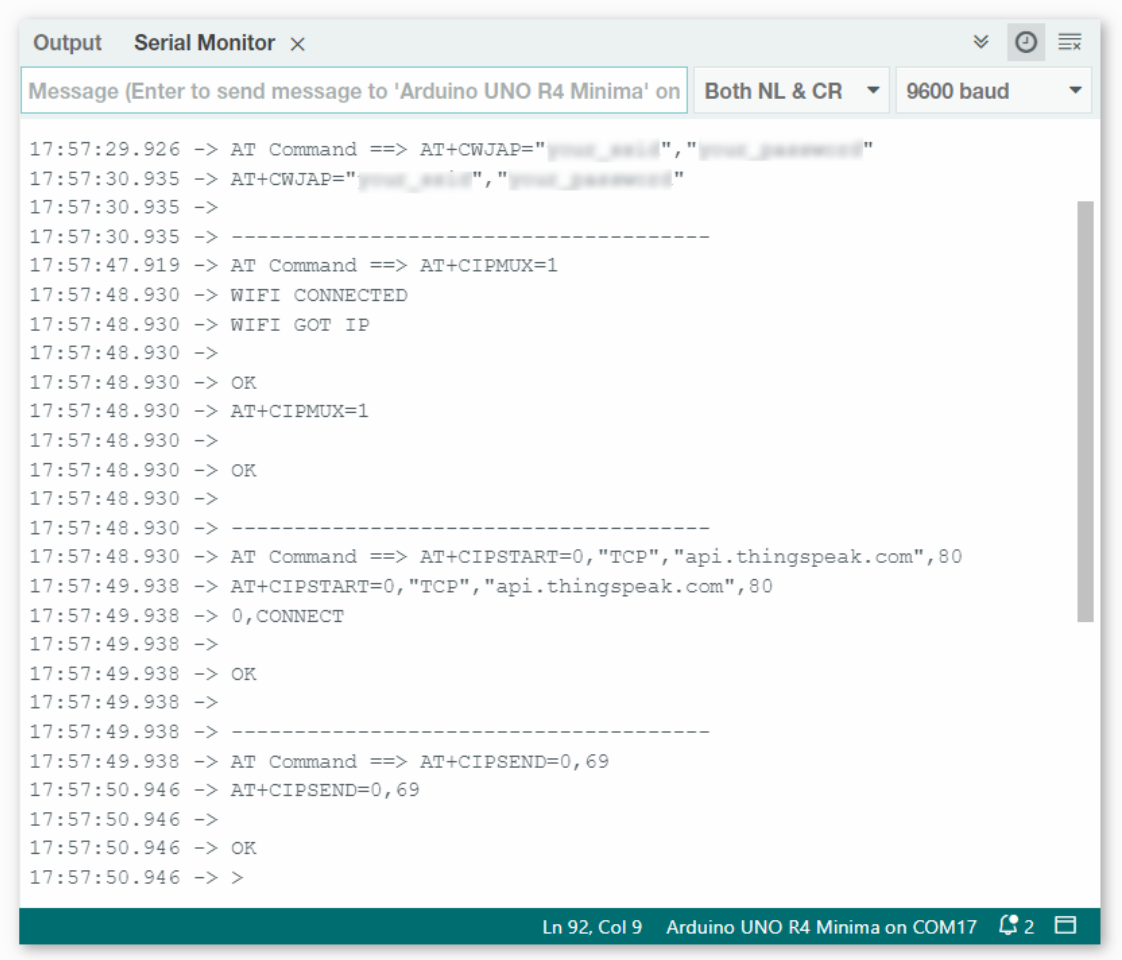
3. Außerdem müssen Sie die `myAPI` mit Ihrem ThingSpeak Channel-API-Schlüssel anpassen.

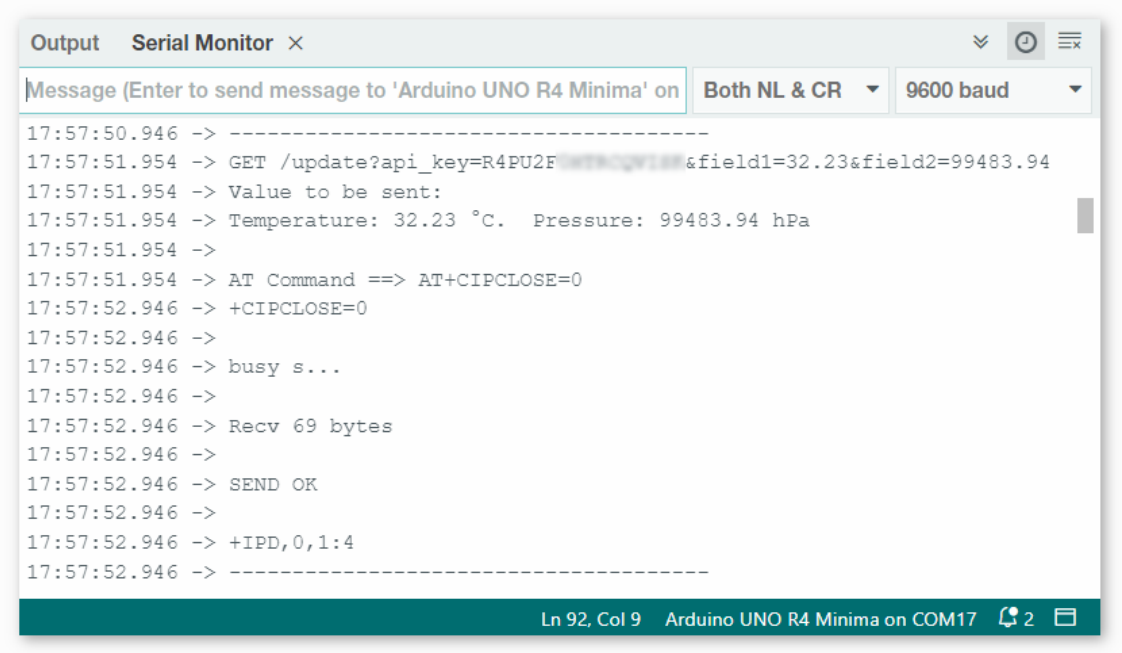
```
String myAPI = "xxxxxxxxxxxx";  // API-Schlüssel
```



Hier finden Sie **Ihren einzigartigen API-Schlüssel, den Sie geheim halten müssen**.

4. Nach der Auswahl des korrekten Boards und Ports klicken Sie auf die Schaltfläche **Hochladen**.
5. Öffnen Sie das serielle Monitor (Baudrate auf **9600** einstellen) und warten Sie auf eine Meldung wie eine erfolgreiche Verbindung.





4. Code-Erklärung

Das im Kit enthaltene ESP8266-Modul ist bereits ab Werk mit einer AT-Firmware vorprogrammiert. Dadurch lässt sich das ESP8266-Modul über AT-Befehle steuern. In diesem Projekt verwenden wir die Software-Seriell-Kommunikation, um die Kommunikation zwischen dem Arduino Uno Board und dem ESP8266-Modul zu ermöglichen. Das Arduino Uno Board sendet AT-Befehle an das ESP8266-Modul, um eine Netzwerkverbindung herzustellen und Anfragen zu senden. Weitere Informationen finden Sie unter .

Das Uno-Board liest Sensordaten und sendet AT-Befehle an das ESP8266-Modul. Das ESP8266-Modul verbindet sich mit einem Netzwerk und sendet Anfragen an die ThingSpeak-Server.

1. **Einrichtung & Globale Variablen**:

   Dieser Abschnitt stellt die Kommunikation mit dem ESP8266-Modul her und deklariert notwendige globale Variablen.

```
#include <SoftwareSerial.h>
SoftwareSerial espSerial(2, 3);
#define DEBUG true
String mySSID = "Ihre_SSID";
String myPWD = "Ihr_Passwort";
String myAPI = "xxxxxxxxxxx";
String myHOST = "api.thingspeak.com";
String myPORT = "80";
unsigned long lastConnectionTime = 0;
const unsigned long postingInterval = 20000L;
```

2. **BMP280 Sensor-Konfiguration**:

   Dieses Code-Segment konfiguriert den BMP280-Sensor für die Datenerfassung.

```
#include <Wire.h>
#include <Adafruit_BMP280.h>
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
#define BMP280_ADDRESS 0x76
Adafruit_BMP280 bmp;
unsigned bmpStatus;
float pressure;
float temperature;
```

3. **Initialisierung (Setup-Funktion)**:

   Die Funktion `setup()` initialisiert die serielle Kommunikation, verbindet das ESP8266-Modul mit dem WLAN und initialisiert den BMP280-Sensor.

```
void setup() {
  Serial.begin(9600);
  espSerial.begin(115200);

  // Initialize the ESP8266 module
  sendATCommand("AT+RST", 1000, DEBUG);                                    //
↪Reset the ESP8266 module
  sendATCommand("AT+CWMODE=1", 1000, DEBUG);                               //
↪Set the ESP mode as station mode
  sendATCommand("AT+CWJAP=\"" + mySSID + "\",\"" + myPWD + "\"", 1000, DEBUG);  //
↪Connect to WiFi network

  // Initialize the bmp280 sensor
  bmpStatus = bmp.begin(BMP280_ADDRESS);
  if (!bmpStatus) {
    Serial.println(F("Could not find a valid BMP280 sensor, check wiring or "
                     "try a different address!"));
    while (1) delay(10);  // Stop code execution if the sensor is not found.
  }

  /* Default settings from datasheet. */
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,     /* Operating Mode. */
                  Adafruit_BMP280::SAMPLING_X2,     /* Temp. oversampling */
                  Adafruit_BMP280::SAMPLING_X16,    /* Pressure oversampling */
                  Adafruit_BMP280::FILTER_X16,      /* Filtering. */
                  Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */
}
```

4. **loop()-Funktion**:

   Die Haupt-Schleife prüft, ob seit der letzten Datenübertragung 20 Sekunden vergangen sind. Ist dies der Fall, werden die Daten gesendet. Sie können den Wert der Variable `postingInterval` anpassen, um das Intervall der Datenübertragung zu ändern.

```
void loop() {
  // Datenübertragung gemäß dem eingestellten Zeitintervall.
  if (millis() - lastConnectionTime > postingInterval) {
    sendData();
  }
}
```

5. **Datenübertragung**:

   Diese Funktion liest die Temperatur und den Druck, konstruiert die GET-Anfrage und sendet die Daten an

ThingSpeak.

Wir haben eine GET-Anfrage in Form von `GET /update?api_key=xxxxxx&field1=xx&field2=xxxxxx` erstellt und drei Parameter an den ThingSpeak-Server gesendet.

- `api_key`: API-Schlüssel für Authentifizierung und Zugriffskontrolle
- `field1`: ein Parameter namens „field1“ zur Aufzeichnung der Temperatur
- `field2`: ein Parameter namens „field2“ zur Aufzeichnung des Luftdrucks

```
void sendData() {
  // Read the temperature and pressure from the BMP280 sensor
  pressure = bmp.readPressure();
  temperature = bmp.readTemperature();

  // If the data is invalid, print an error message and stop sending it
  if (isnan(pressure) || isnan(temperature)) {
    Serial.println("Failed to read from BMP sensor!");
    return;
  }

  // Construct the GET request for ThingSpeak
  String sendData = "GET /update?api_key=" + myAPI;
  sendData += "&field1=" + String(temperature);
  sendData += "&field2=" + String(pressure);

  // Send the GET request to ThingSpeak via the ESP8266
  sendATCommand("AT+CIPMUX=1", 1000, DEBUG);  //Allow multiple connections
  sendATCommand("AT+CIPSTART=0,\"TCP\",\"" + myHOST + "\"," + myPORT, 1000, DEBUG);␣
↪ // Start a TCP connection to ThingSpeak
  sendATCommand("AT+CIPSEND=0," + String(sendData.length() + 4), 1000, DEBUG);     ␣
↪ // Send the GET request
  espSerial.find(">");    // Wait for the ">" character from the ESP8266
  espSerial.println(sendData);    // Send the GET request
  Serial.println(sendData);

  // Print the values
  Serial.println("Value to be sent: ");
  printBMP();  // Call the printBMP function to print the temperature and pressure

  sendATCommand("AT+CIPCLOSE=0", 1000, DEBUG);  // Close the TCP connection
  lastConnectionTime = millis();          // Update the last connection time
}
```

6. **Hilfsfunktionen**:

   Diese Funktionen unterstützen beim Senden von AT-Befehlen an das ESP8266-Modul und beim Anzeigen der Messwerte des BMP280-Sensors.

```
void sendATCommand(String command, const int timeout, boolean debug) {
    ... // (refer to the provided code for the full sendATCommand function)
}

void printBMP() {
    ... // (refer to the provided code for the full printBMP function)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
}
```

Referenz

-

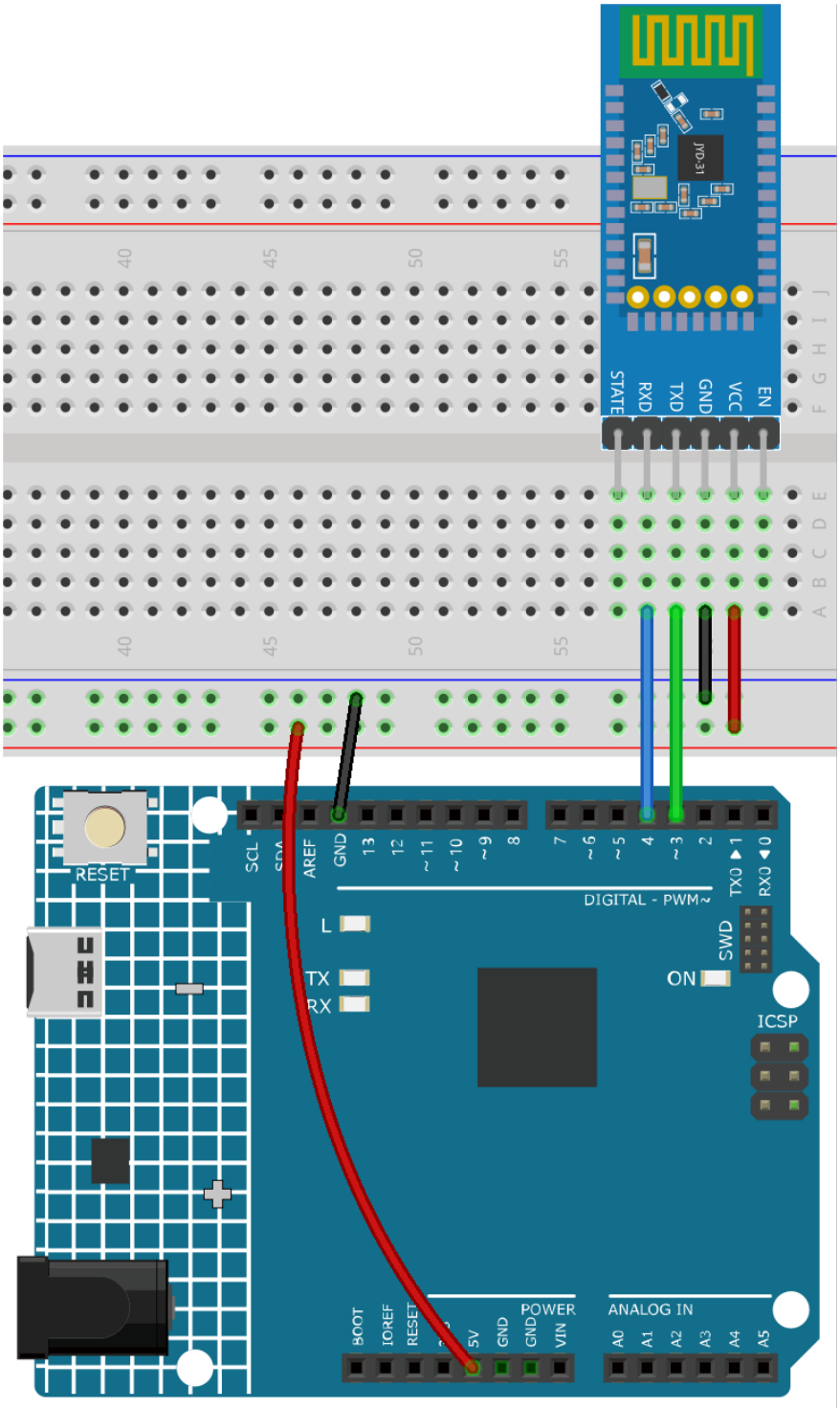
Bluetooth

2.5.9 Bluetooth-Einstieg

In diesem Projekt zeigen wir, wie eine Kommunikation mit einem Bluetooth-Modul über Arduino erfolgen kann.

Zunächst müssen wir die Schaltung aufbauen und die serielle Kommunikation über Software nutzen. Verbinden Sie den TX-Pin des Bluetooth-Moduls mit Pin 3 des Uno-Boards und den RX-Pin des Bluetooth-Moduls mit Pin 4 des Uno-Boards.

1. Schaltungsaufbau



- *Arduino UNO R4 Minima-Platine*
- *JDY-31 Bluetooth-Modul*

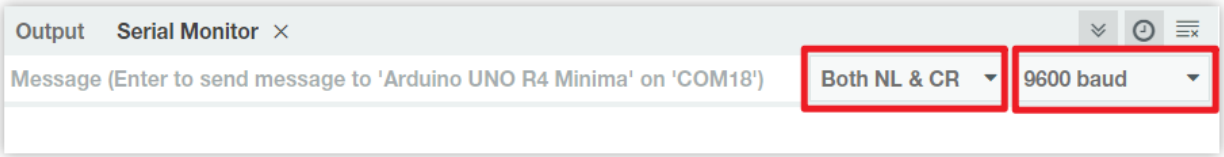
2. Code hochladen

Öffnen Sie die Datei `00-Bluetooth_start.ino` im Pfad `ultimate-sensor-kit\iot_project\bluetooth\00-Bluetooth_start` oder kopieren Sie diesen Code in die **Arduino IDE**.

Der Code stellt eine serielle Kommunikation über die SoftwareSerial-Bibliothek von Arduino her. Dadurch kann das Arduino mit dem JDY-31 Bluetooth-Modul über die digitalen Pins 3 und 4 (als Rx und Tx) kommunizieren. Es überprüft den Datentransfer zwischen beiden und leitet empfangene Nachrichten mit einer Baudrate von 9600 weiter. **Mit diesem Code können Sie AT-Befehle an das JDY-31 Bluetooth-Modul senden und dessen Antworten empfangen**.

3. Bluetooth-Modul konfigurieren

Klicken Sie auf das Lupensymbol (Serieller Monitor) in der oberen rechten Ecke und stellen Sie die Baudrate auf `9600` ein. Wählen Sie dann `both NL & CR` aus dem Drop-down-Menü für die `New Line`.



Im Folgenden finden Sie einige Beispiele für die Verwendung von AT-Befehlen zur Konfiguration von Bluetooth-Modulen: Geben Sie `AT+NAME` ein, um den Namen des Bluetooth-Geräts zu erfahren. Wenn Sie den Bluetooth-Namen ändern möchten, fügen Sie nach `AT+NAME` einen neuen Namen hinzu.

- **Bluetooth-Gerätenamen abfragen:** `AT+NAME`
- **Bluetooth-Gerätenamen festlegen:** `AT+NAME` (gefolgt vom neuen Namen). `+OK` bedeutet, dass die Einstellung erfolgreich war. Sie können `AT+NAME` erneut senden, um dies zu überprüfen.

Bemerkung: Um ein konsistentes Lernerlebnis zu gewährleisten, wird empfohlen, die Standard-Baudrate des Bluetooth-Moduls nicht zu ändern und **sie auf ihrem Standardwert von 4 (d. h. 9600 Baud) zu belassen**. In relevanten Kursen kommunizieren wir mit einer Baudrate von 9600 über Bluetooth.

- **Bluetooth-Baudrate einstellen:** `AT+BAUD` (gefolgt von der Zahl, die die Baudrate angibt).
  - 4 == 9600
  - 5 == 19200
  - 6 == 38400
  - 7 == 57600
  - 8 == 115200
  - 9 == 128000

Weitere AT-Befehle finden Sie in der folgenden Tabelle.

Befehl	Funktion	Standard
AT+VERSION	Versionsnummer	JDY-31-V1.2
AT+RESET	Soft-Reset	
AT+DISC	Trennen (gültig bei Verbindung)	
AT+LADDR	MAC-Adresse des Moduls abfragen	
AT+PIN	Verbindungspasswort setzen oder abfragen	1234
AT+BAUD	Baudrate setzen oder abfragen	9600
AT+NAME	Sendenamen setzen oder abfragen	JDY-31-SPP
AT+DEFAULT	Werkseinstellungen wiederherstellen	
AT+ENLOG	Serieller Port Statusausgabe	1

4. Kommunikation über Bluetooth-Debugging-Tools auf Mobiltelefonen

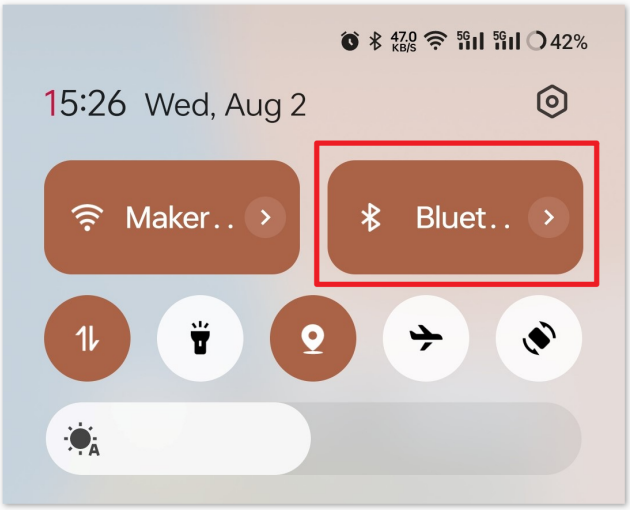
Mit einer App namens „Serial Bluetooth Terminal“ können wir Nachrichten vom Bluetooth-Modul an Arduino senden, um den Vorgang der Bluetooth-Interaktion zu simulieren. Das Bluetooth-Modul sendet empfangene Nachrichten über die serielle Schnittstelle an Arduino. Ebenso kann Arduino Nachrichten über die serielle Schnittstelle an das Bluetooth-Modul senden.

a. **Serial Bluetooth Terminal installieren**

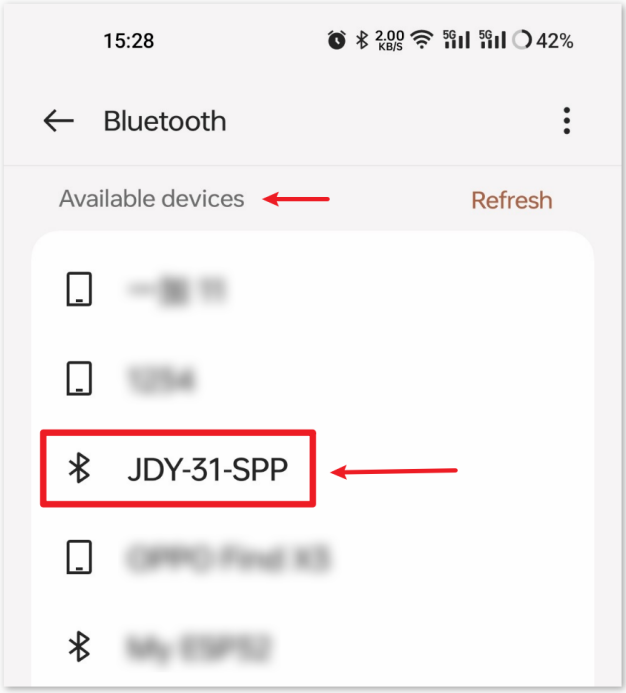
   Laden Sie die App im Google Play Store herunter und installieren Sie sie.

b. **Bluetooth verbinden**

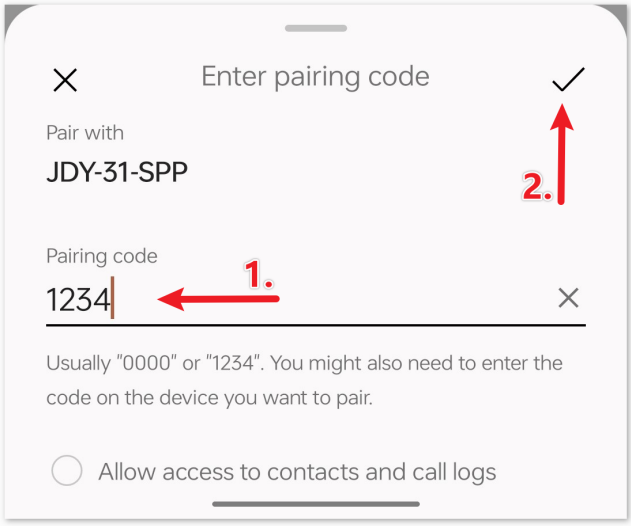
   Aktivieren Sie zunächst **Bluetooth** auf Ihrem Smartphone.



   Navigieren Sie zu den **Bluetooth-Einstellungen** auf Ihrem Smartphone und suchen Sie nach Namen wie **JDY-31-SPP**.

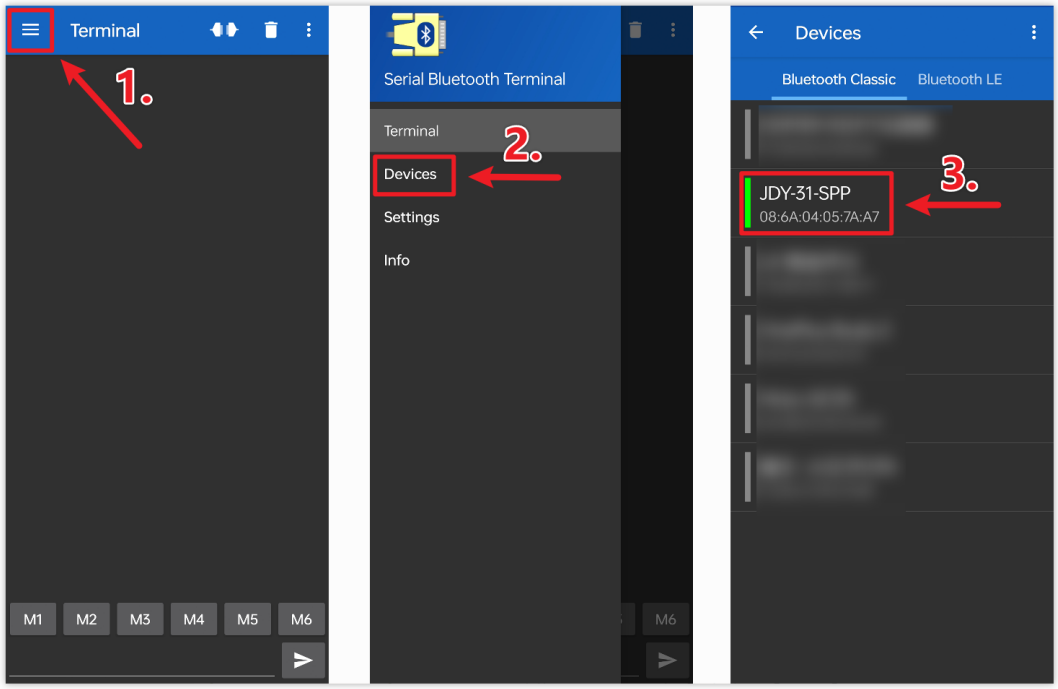


Nach dem Anklicken stimmen Sie der **Pairing-Anfrage** im Popup-Fenster zu. Wenn nach einem Pairing-Code gefragt wird, geben Sie „1234“ ein.



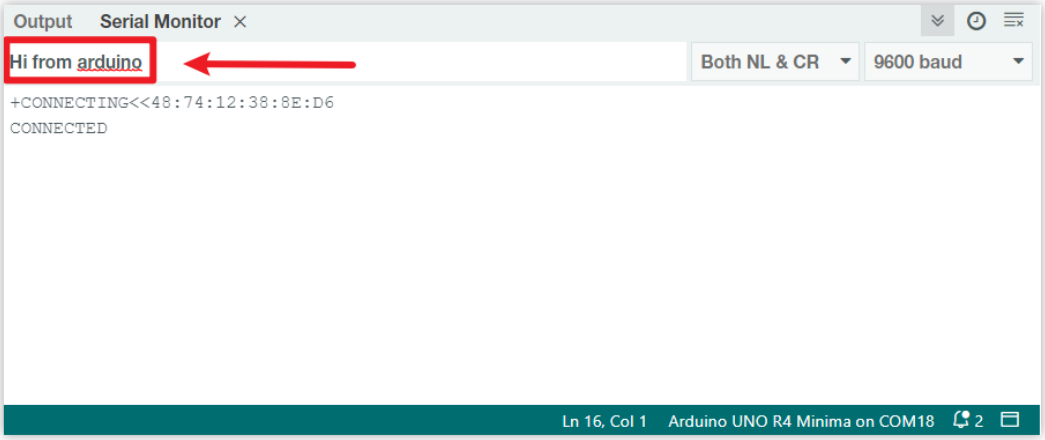
c. **Mit dem Bluetooth-Modul kommunizieren**

Öffnen Sie das Serial Bluetooth Terminal und verbinden Sie sich mit „JDY-31-SPP“.

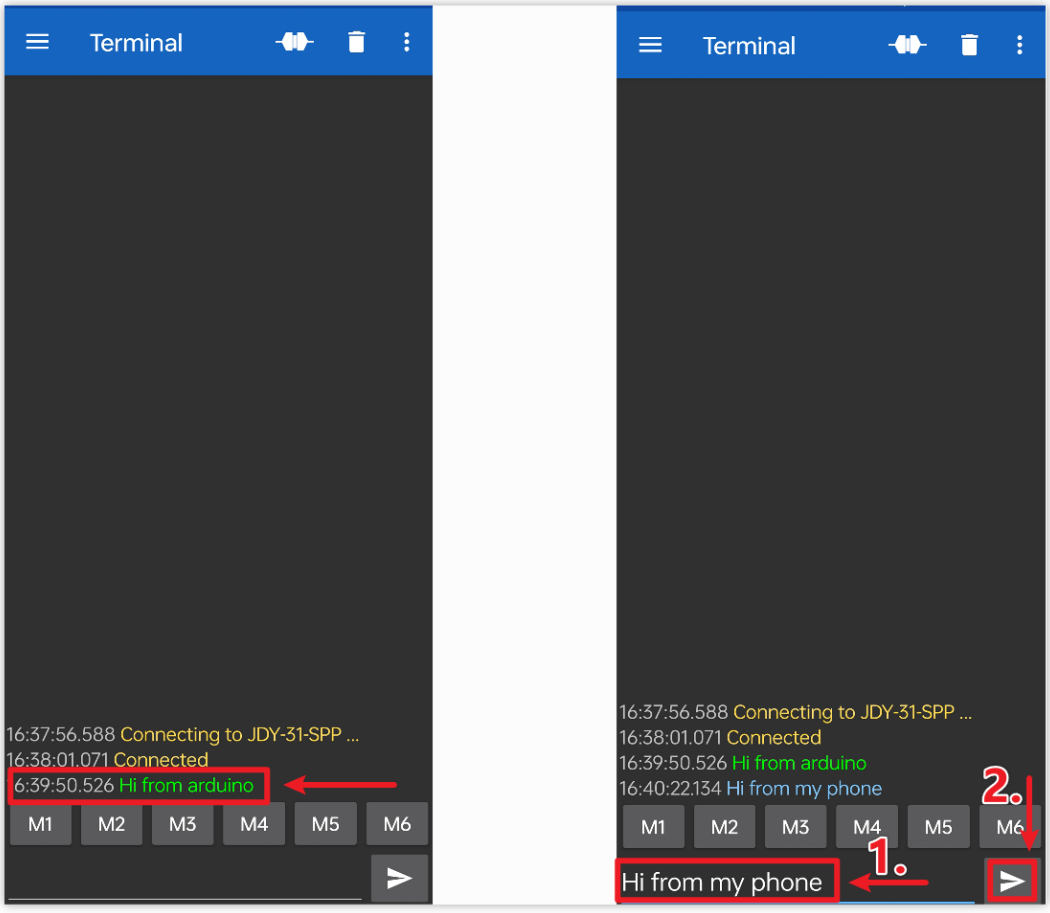


Nach erfolgreicher Verbindung wird im seriellen Monitor eine Erfolgsmeldung angezeigt.

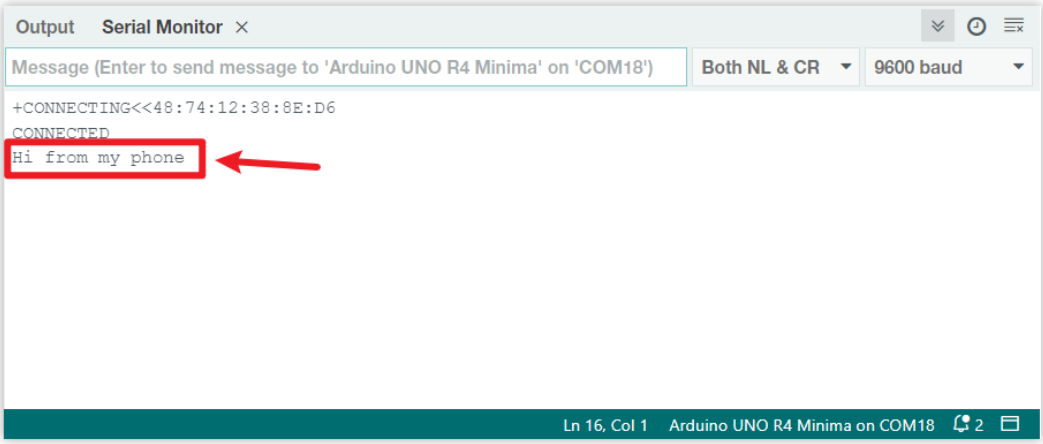
Geben Sie die Nachricht im seriellen Monitor ein und senden Sie sie an das Bluetooth-Modul.



Nach dem Senden sehen Sie diese Nachricht in der Serial Bluetooth Terminal-App. Ebenso können Daten über die **Serial Bluetooth Terminal**-App per Bluetooth an Arduino gesendet werden.



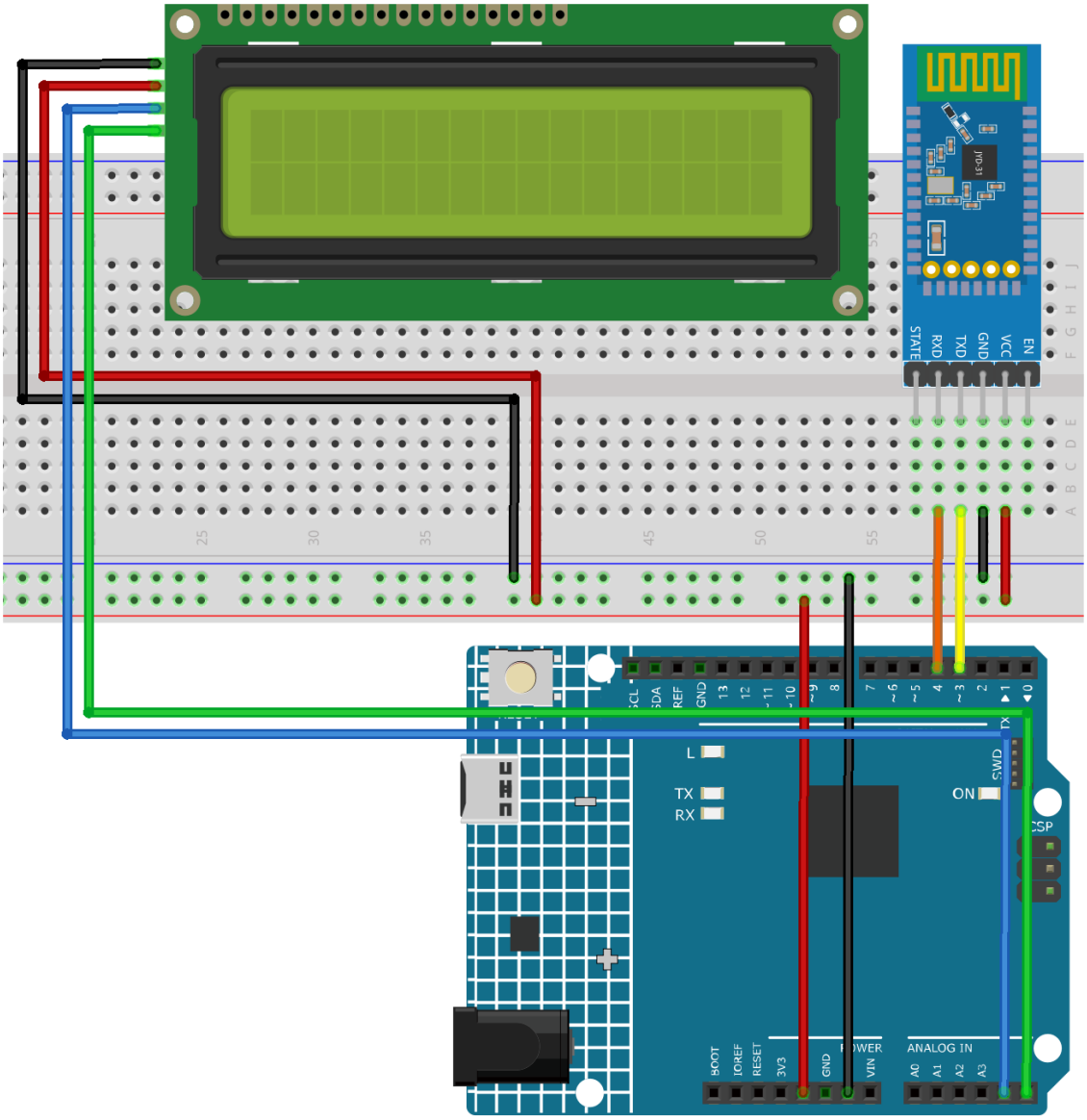
Diese Nachricht vom Bluetooth-Gerät ist im seriellen Monitor sichtbar.



2.5.10 Bluetooth LCD

Das Projekt empfängt Nachrichten über ein mit dem UNO-Board verbundenes Bluetooth-Modul und zeigt die empfangenen Nachrichten auf einem LCD-Bildschirm an.

1. Schaltung aufbauen



- *Arduino UNO R4 Minima-Platine*
- *JDY-31 Bluetooth-Modul*
- *I2C LCD 1602*

2. Code hochladen

1. Öffnen Sie die Datei `01-Bluetooth_lcd.ino` im Verzeichnis `ultimate-sensor-kit\iot_project\bluetooth\01-Bluetooth_lcd`, oder kopieren Sie diesen Code in die **Arduino IDE**.

   Bemerkung: Um die Bibliothek zu installieren, nutzen Sie den Arduino-Bibliotheksmanager und suchen nach **„LiquidCrystal I2C“**, um es zu installieren.

   ---

2. Wählen Sie das korrekte Board und den passenden Port aus und klicken Sie dann auf den **Upload**-Button.
3. Öffnen Sie den Seriellen Monitor (Baudrate auf **9600** einstellen), um Debug-Nachrichten anzuzeigen.

3. App und Bluetooth-Modul verbinden

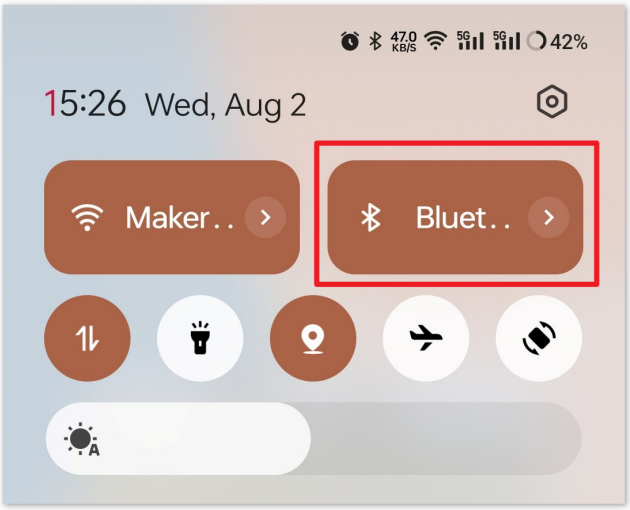
Eine App namens „Serial Bluetooth Terminal“ ermöglicht es, Nachrichten vom Bluetooth-Modul zum Arduino zu senden.

a. **Serial Bluetooth Terminal installieren**

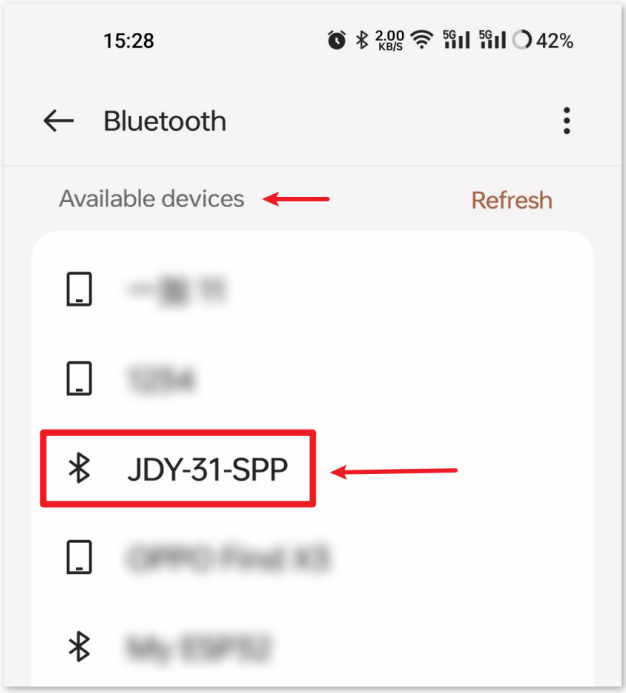
   Laden und installieren Sie aus dem Google Play Store.

b. **Bluetooth verbinden**

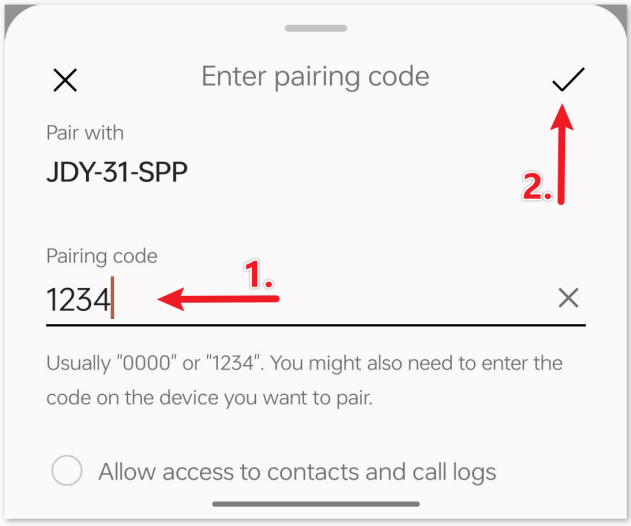
   Aktivieren Sie zuerst **Bluetooth** auf Ihrem Smartphone.



   Navigieren Sie zu den **Bluetooth-Einstellungen** auf Ihrem Smartphone und suchen Sie nach Namen wie **JDY-31-SPP**.

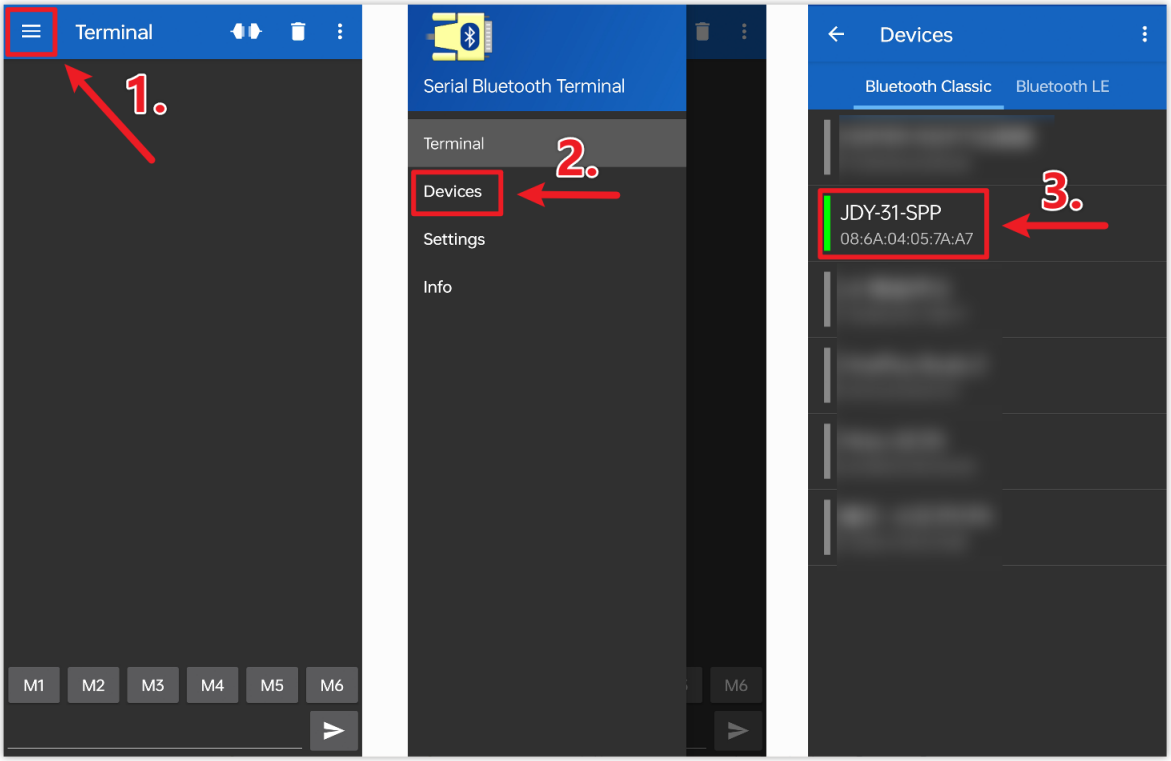


Nach dem Anklicken bestätigen Sie die **Kopplungsanfrage** im Pop-up-Fenster. Falls ein Kopplungscode erforderlich ist, geben Sie „1234“ ein.



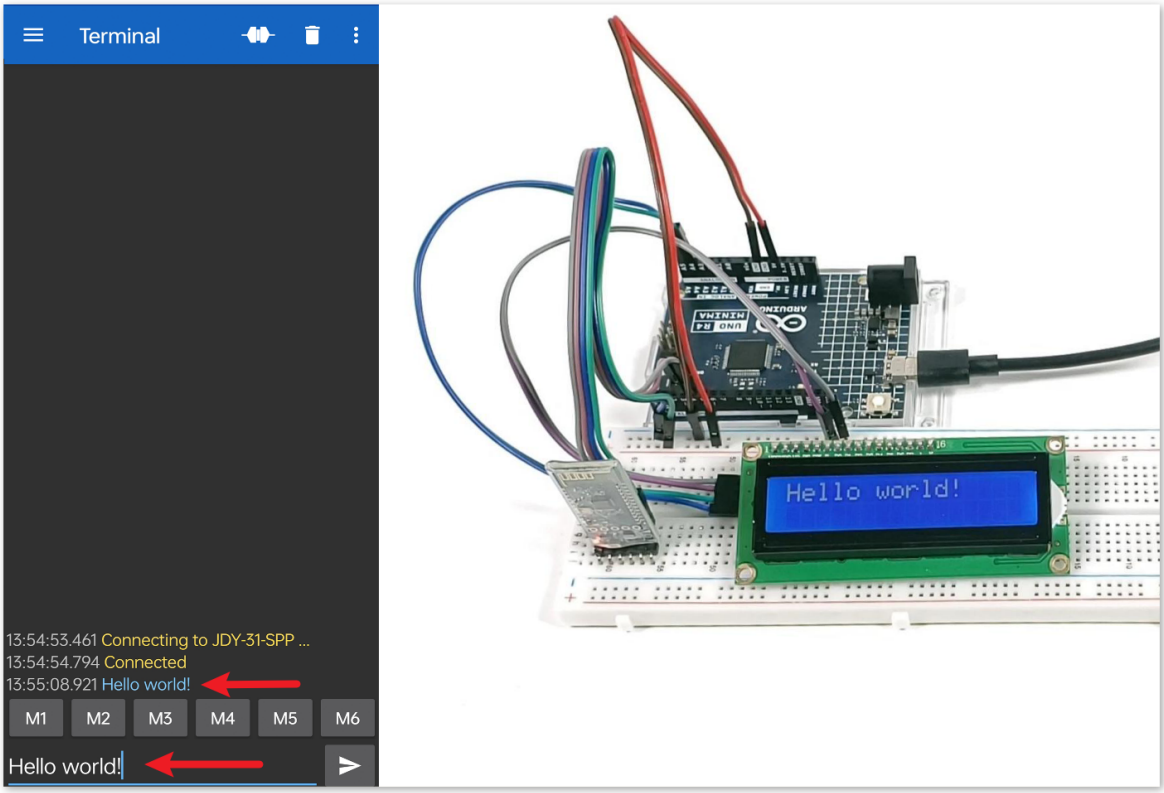
c. **Kommunikation mit dem Bluetooth-Modul**

Öffnen Sie das Serial Bluetooth Terminal und verbinden Sie es mit „JDY-31-SPP“.



d. **Befehle senden**

Verwenden Sie die Serial Bluetooth Terminal App, um Nachrichten über Bluetooth an den Arduino zu senden. Die über Bluetooth gesendete Nachricht wird auf dem LCD angezeigt.



4. Code-Erläuterung

Bemerkung: Zur Bibliotheksinstallation nutzen Sie den Arduino-Bibliotheksmanager und suchen nach **„LiquidCrystal I2C“**, um die Bibliothek zu installieren.

1. Einrichtung des LCDs

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

Dieses Code-Segment bindet die LiquidCrystal_I2C-Bibliothek ein und initialisiert das LCD-Modul mit der I2C-Adresse `0x27`. Zudem wird festgelegt, dass das LCD 16 Spalten und 2 Zeilen hat.

2. Einrichtung der Bluetooth-Kommunikation

```
#include <SoftwareSerial.h>
const int bluetoothTx = 3;
const int bluetoothRx = 4;
SoftwareSerial bleSerial(bluetoothTx, bluetoothRx);
```

Hier wird die SoftwareSerial-Bibliothek eingebunden, um die Kommunikation zwischen dem JDY-31 Bluetooth-Modul und dem Arduino über die Pins 3 (TX) und 4 (RX) zu ermöglichen.

3. Initialisierung

```
void setup() {
   lcd.init();
   lcd.clear();
   lcd.backlight();

   Serial.begin(9600);
   bleSerial.begin(9600);
}
```

Die Funktion `setup()` initialisiert das LCD und entfernt etwaigen vorhandenen Inhalt. Auch wird die Hintergrundbeleuchtung des LCDs aktiviert. Die Kommunikation mit dem seriellen Monitor sowie dem Bluetooth-Modul wird bei einer Baudrate von `9600` gestartet.

4. Hauptprogrammschleife

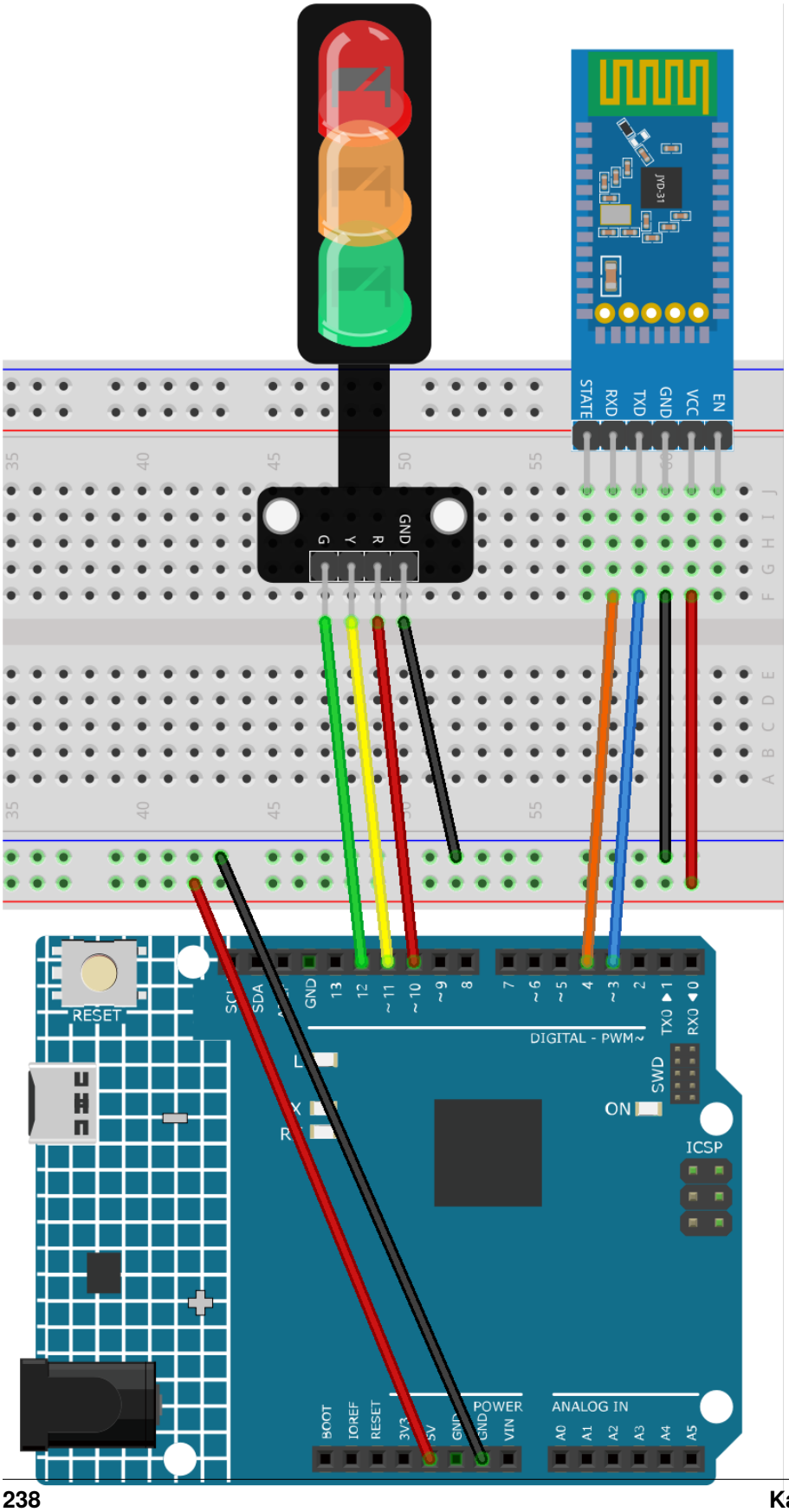
```
void loop() {
   String data;

   if (bleSerial.available()) {
      data += bleSerial.readString();
      data = data.substring(0, data.length() - 2);
      Serial.print(data);

      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(data);
   }

   if (Serial.available()) {
      bleSerial.write(Serial.read());
   }
}
```

Dies ist die Hauptprogrammschleife des Arduino-Programms. Sie überprüft kontinuierlich den Eingang von Daten sowohl vom Bluetooth-Modul als auch vom seriellen Monitor. Bei Empfang von Daten über das Bluetooth-Gerät werden diese verarbeitet, auf dem seriellen Monitor angezeigt und auf dem LCD dargestellt. Wenn Daten im seriellen Monitor eingegeben werden, werden diese Daten an das Bluetooth-Modul gesendet.

2.5.11 Bluetooth-Ampel

Dieses Projekt dient der Steuerung einer Verkehrsampel (rote, gelbe, grüne LEDs) über Bluetooth-Kommunikation. Der Nutzer kann einen Buchstaben (‚R', ‚Y' oder ‚G') von einem Bluetooth-Gerät senden. Bei Empfang eines dieser Zeichen leuchtet die entsprechende LED auf, während die beiden anderen LEDs ausgeschaltet bleiben.

1. Schaltung aufbauen



- *Arduino UNO R4 Minima-Platine*
- *JDY-31 Bluetooth-Modul*
- *Ampel-Modul*

2. Code hochladen

1. Öffnen Sie die Datei `02-Bluetooth_traffic_light.ino` im Pfad `ultimate-sensor-kit\iot_project\bluetooth\02-Bluetooth_traffic_light` oder kopieren Sie diesen Code in die **Arduino IDE**.
2. Wählen Sie das korrekte Board und den entsprechenden Port aus, und klicken Sie auf den **Hochladen**-Button.
3. Öffnen Sie den Seriellen Monitor (Baudrate auf **9600** einstellen), um Debug-Meldungen anzuzeigen.

3. App und Bluetooth-Modul verbinden

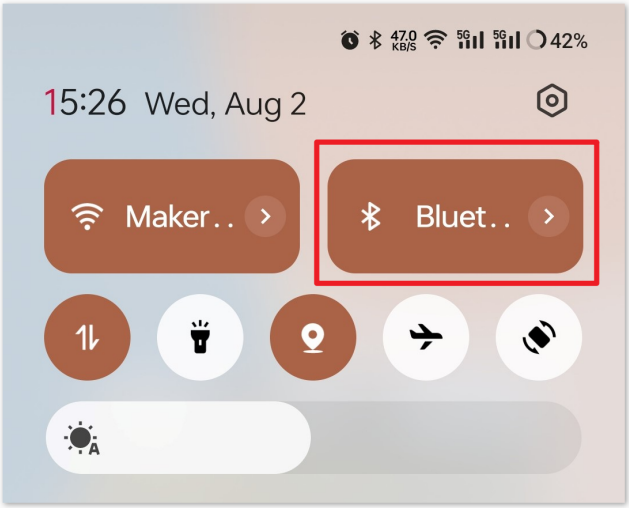
Zur Kommunikation können wir die App „Serial Bluetooth Terminal“ verwenden, um Nachrichten vom Bluetooth-Modul an den Arduino zu senden.

a. **Serial Bluetooth Terminal installieren**

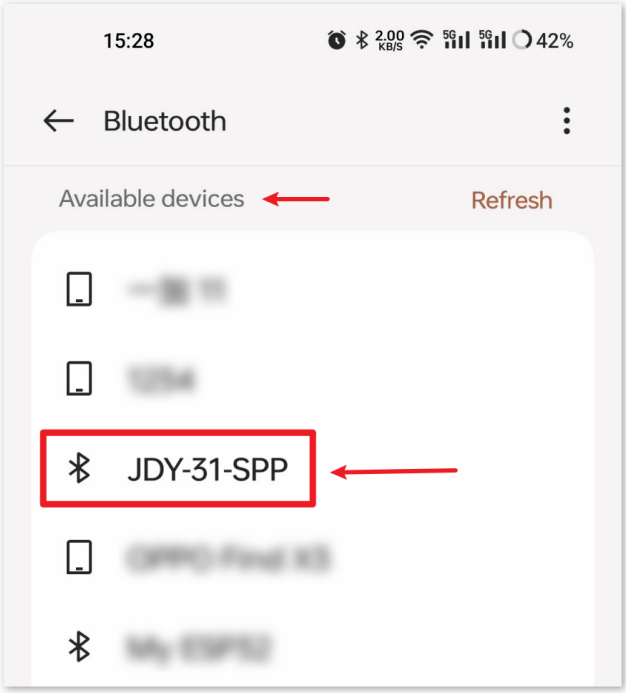
   Laden und installieren Sie über den Google Play Store.

b. **Bluetooth verbinden**

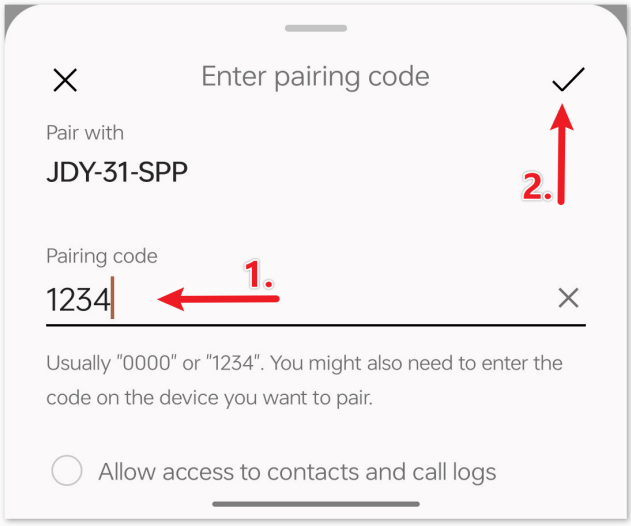
   Aktivieren Sie zuerst **Bluetooth** auf Ihrem Smartphone.



   Navigieren Sie zu den **Bluetooth-Einstellungen** auf Ihrem Smartphone und suchen Sie nach Namen wie **JDY-31-SPP**.

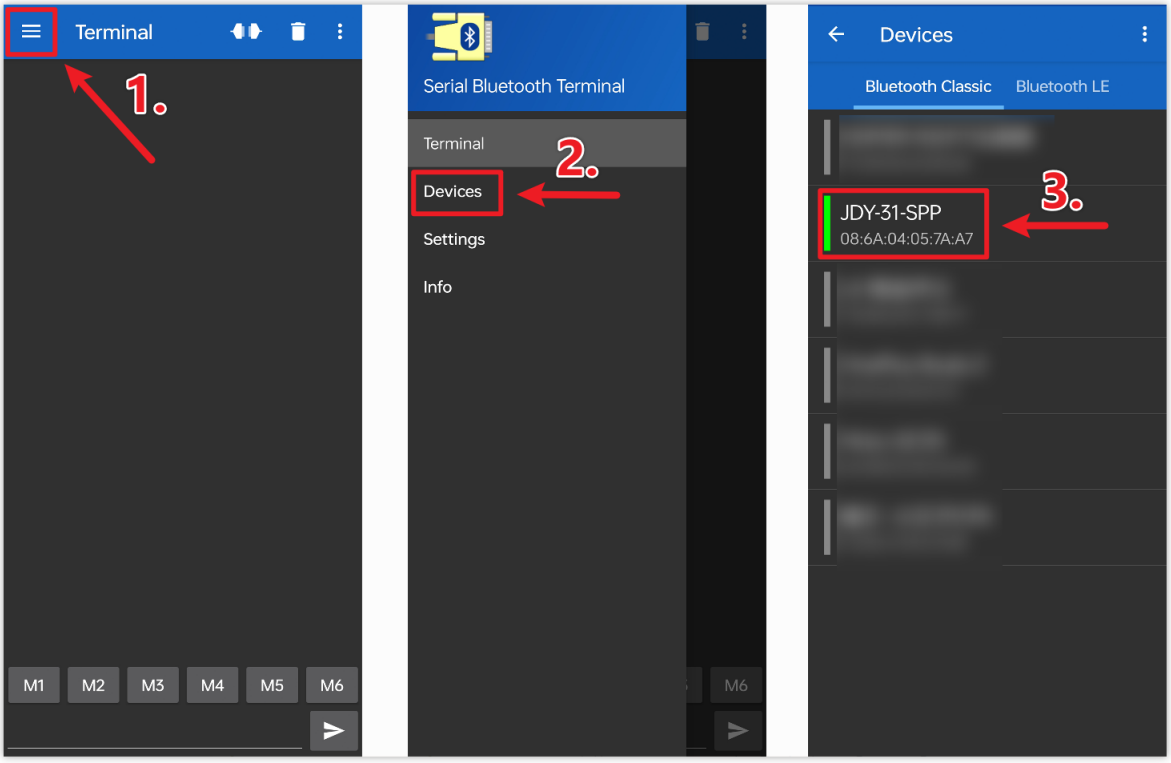


Nach dem Anklicken stimmen Sie der **Kopplungsanfrage** im Pop-up-Fenster zu. Falls nach einem Kopplungscode gefragt wird, geben Sie „1234“ ein.



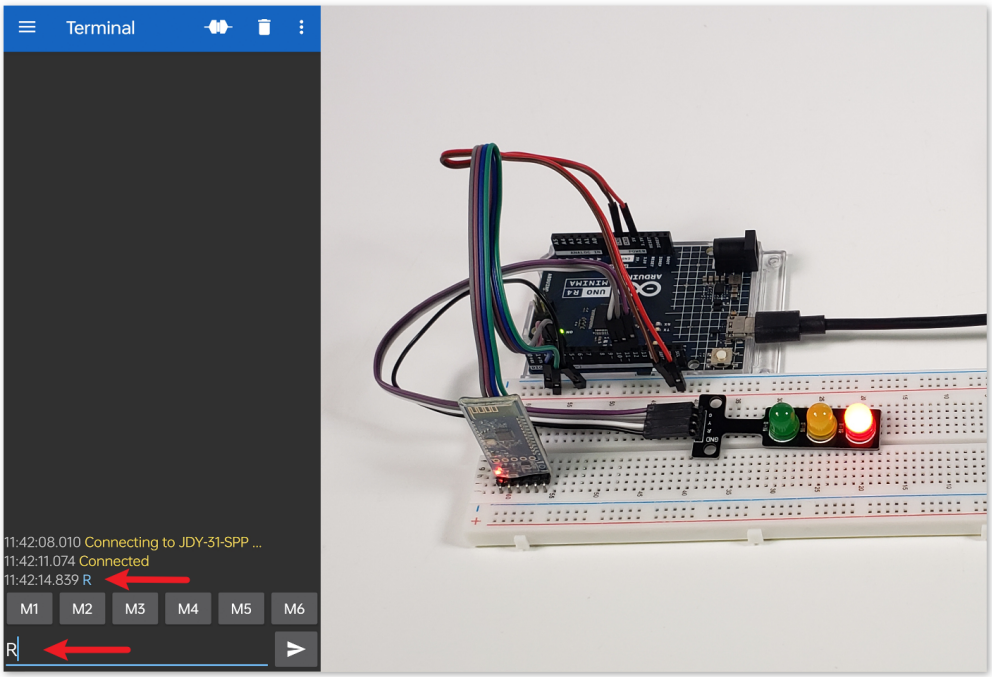
c. **Mit dem Bluetooth-Modul kommunizieren**

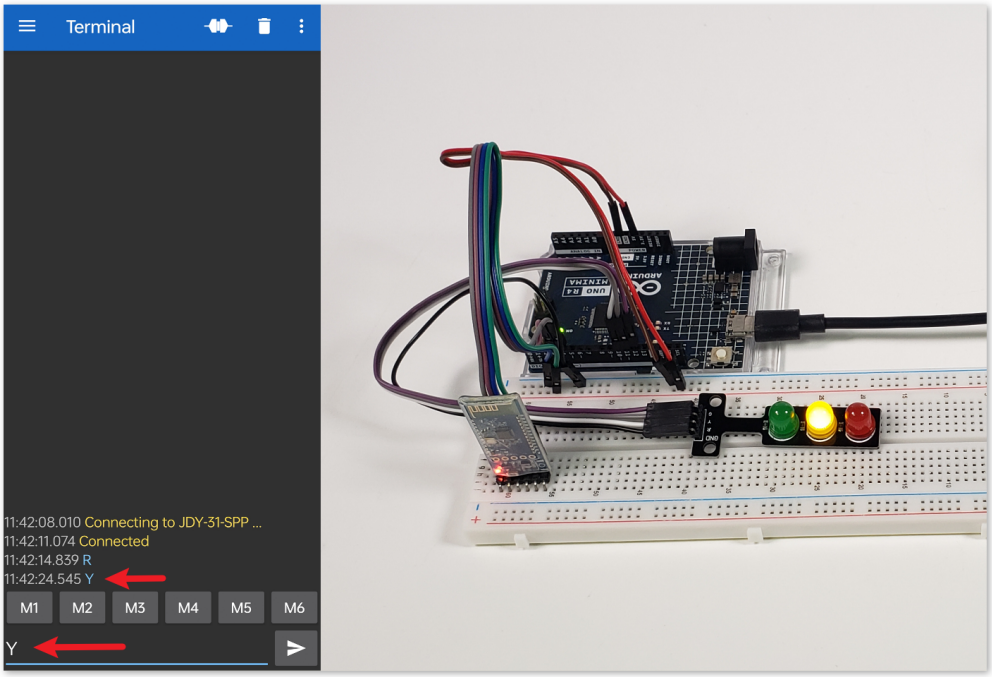
Öffnen Sie das Serial Bluetooth Terminal und verbinden Sie sich mit „JDY-31-SPP“.

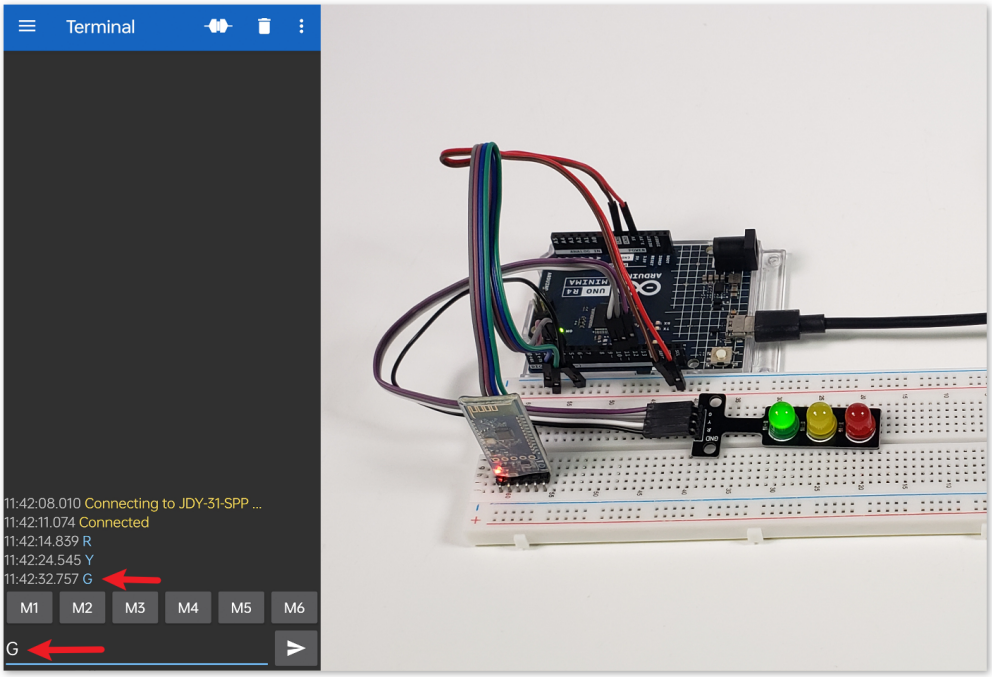


d. **Befehle senden**

Verwenden Sie die Serial Bluetooth Terminal App, um Befehle über Bluetooth an den Arduino zu senden. Senden Sie R, um das rote Licht einzuschalten, Y für gelb und G für grün.




Terminal
11:42:08.010 Connecting to JDY-31-SPP ...
11:42:11.074 Connected
11:42:14.839 R
11:42:24.545 Y
M1
M2
M3
M4
M5
M6
Y


Terminal
11:42:08.010 Connecting to JDY-31-SPP ...
11:42:11.074 Connected
11:42:14.839 R
11:42:24.545 Y
11:42:32.757 G
M1
M2
M3
M4
M5
M6
G

4. Code-Erklärung

1. Initialisierung und Bluetooth-Konfiguration

```
// Set up Bluetooth module communication
#include <SoftwareSerial.h>
const int bluetoothTx = 3;
const int bluetoothRx = 4;
SoftwareSerial bleSerial(bluetoothTx, bluetoothRx);
```

Zunächst binden wir die SoftwareSerial-Bibliothek ein, um die Bluetooth-Kommunikation zu erleichtern. Danach definieren wir die TX- und RX-Pins des Bluetooth-Moduls und ordnen sie den Pins 3 und 4 am Arduino zu. Abschließend initialisieren wir das `bleSerial`-Objekt für die Bluetooth-Kommunikation.

2. Definition der LED-Pins

```
// Pin numbers for each LED
const int rledPin = 10;  //red
const int yledPin = 11;  //yellow
const int gledPin = 12;  //green
```

Hier legen wir fest, an welchen Arduino-Pins unsere LEDs angeschlossen sind. Die rote LED ist am Pin 10, die gelbe am 11 und die grüne am 12 angeschlossen.

3. Funktion setup()

```
void setup() {
   pinMode(rledPin, OUTPUT);
   pinMode(yledPin, OUTPUT);
   pinMode(gledPin, OUTPUT);

   Serial.begin(9600);
   bleSerial.begin(9600);
}
```

In der Funktion `setup()` setzen wir die LED-Pins als `OUTPUT`. Des Weiteren starten wir die serielle Kommunikation sowohl für das Bluetooth-Modul als auch für die Standard-Serielle-Schnittstelle (verbunden mit dem Computer) mit einer Baudrate von 9600.

4. Hauptloop() für die Bluetooth-Kommunikation

```
void loop() {
   while (bleSerial.available() > 0) {
      character = bleSerial.read();
      Serial.println(character);

      if (character == 'R') {
         toggleLights(rledPin);
      } else if (character == 'Y') {
         toggleLights(yledPin);
      } else if (character == 'G') {
         toggleLights(gledPin);
      }
   }
}
```

In unserer Haupt-Schleife `loop()` prüfen wir fortlaufend, ob Daten vom Bluetooth-Modul verfügbar sind. Falls Daten empfangen werden, lesen wir das Zeichen aus und zeigen es im seriellen Monitor an. Je nach empfangenem Zeichen (R, Y oder G) schalten wir die jeweilige LED mit der Funktion `toggleLights()` um.

5. Funktion zum Umschalten der Lichter

```
void toggleLights(int targetLight) {
   digitalWrite(rledPin, LOW);
   digitalWrite(yledPin, LOW);
   digitalWrite(gledPin, LOW);

   digitalWrite(targetLight, HIGH);
}
```

Diese Funktion, `toggleLights()`, schaltet zuerst alle LEDs aus. Nachdem sichergestellt ist, dass alle LEDs ausgeschaltet sind, wird die spezifizierte Ziel-LED eingeschaltet. Dadurch wird gewährleistet, dass jeweils nur eine LED aktiv ist.

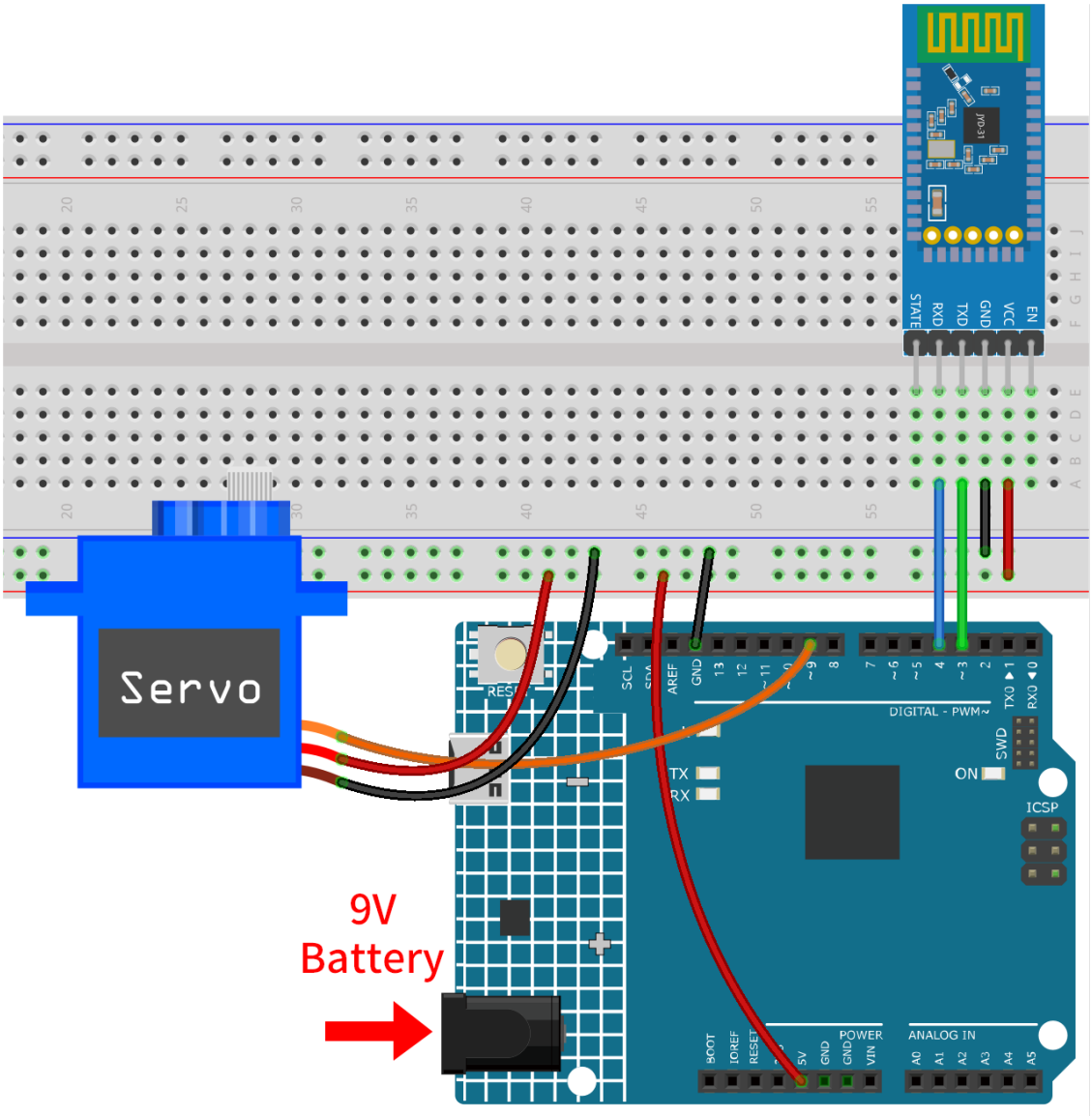
2.5.12 Bluetooth-Schlosssteuerung

Dieses Projekt nutzt eine mit dem MIT App Inventor erstellte Android-App, um über Bluetooth einen Servomotor aus der Ferne zu steuern und so einen Verriegelungsmechanismus zu simulieren. Benutzer können über die App spezifische Nachrichten senden, um den Servo entweder in die „verriegelte“ oder „entriegelte“ Position zu bewegen.

Das System verwendet ein JDY-31 Bluetooth-Modul, um diese Nachrichten zu empfangen und gibt entsprechende Anweisungen an ein Arduino Uno Board, um den Winkel des Servomotors anzupassen. Bei Empfang der Nachricht ‚1‘ bewegt sich der Servo in die „verriegelte“ Position und bei Empfang der Nachricht ‚0‘ in die „entriegelte“ Position.

Diese Android-Anwendung wird mithilfe einer ergänzenden webbasierten Plattform namens erstellt. Das Projekt bietet eine hervorragende Gelegenheit, die Schnittstelle zwischen einem Arduino und einem Smartphone kennen zu lernen.

1. Schaltungsaufbau



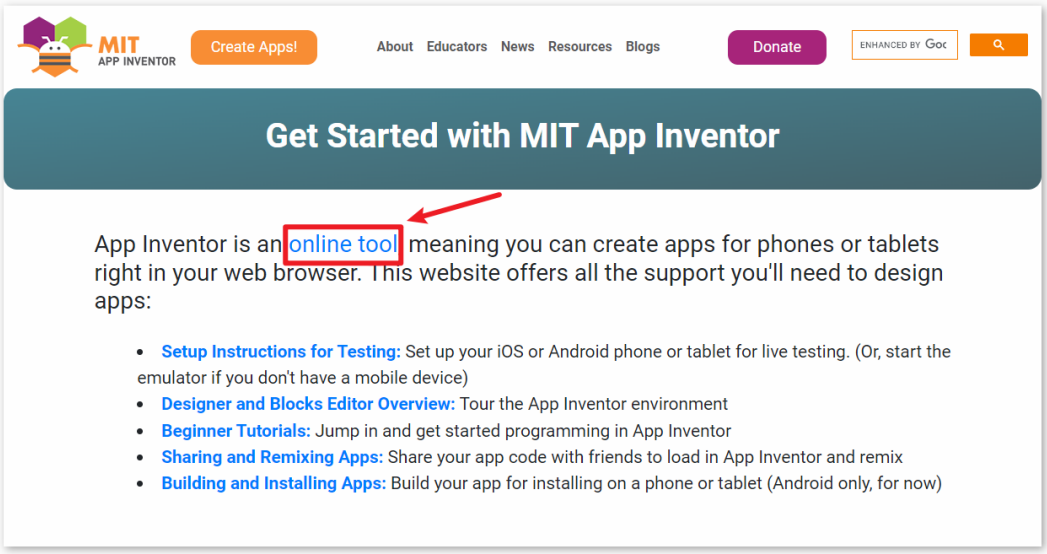
- *Arduino UNO R4 Minima-Platine*
- *JDY-31 Bluetooth-Modul*
- *Servomotor (SG90)*

2. Erstellung der Android-App

Die Android-Anwendung wird mit einer kostenlosen Webanwendung namens entwickelt. Der MIT App Inventor eignet sich hervorragend als Einstiegspunkt für die Android-Entwicklung dank seiner intuitiven Drag-and-Drop-Funktionen, mit denen einfache Anwendungen erstellt werden können.

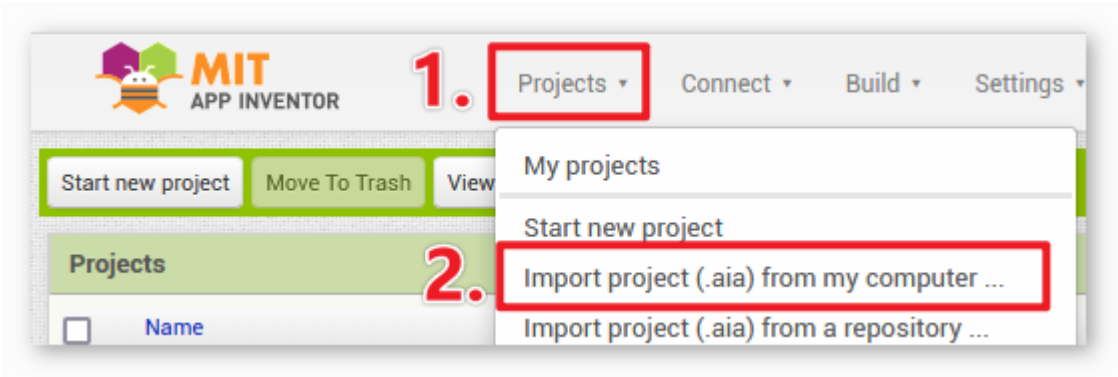
Lassen Sie uns beginnen.

1. Besuchen Sie und klicken Sie auf „Online-Tool“ zum Einloggen. Zum Registrieren bei MIT App Inventor benötigen Sie ein Google-Konto.



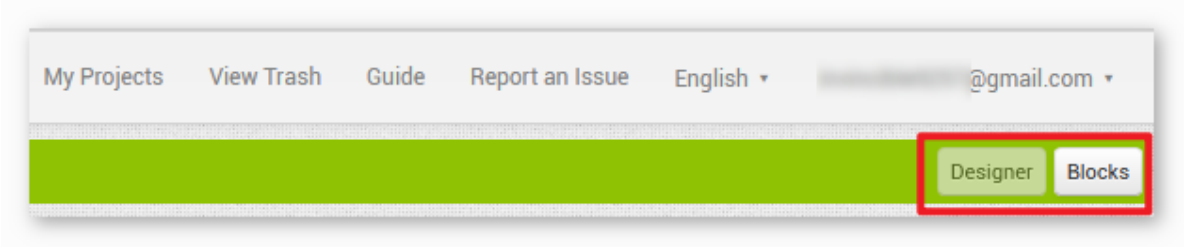
2. Nach dem Einloggen navigieren Sie zu **Projects** -> **Import project (.aia) from my computer**. Laden Sie anschließend die Datei `Bluetooth_controlled_lock.aia` aus dem Pfad `ultimate-sensor-kit\iot_project\bluetooth\03-Bluetooth_lock_controller` hoch.

   Sie können es auch direkt hier herunterladen: `Bluetooth_controlled_lock.aia`

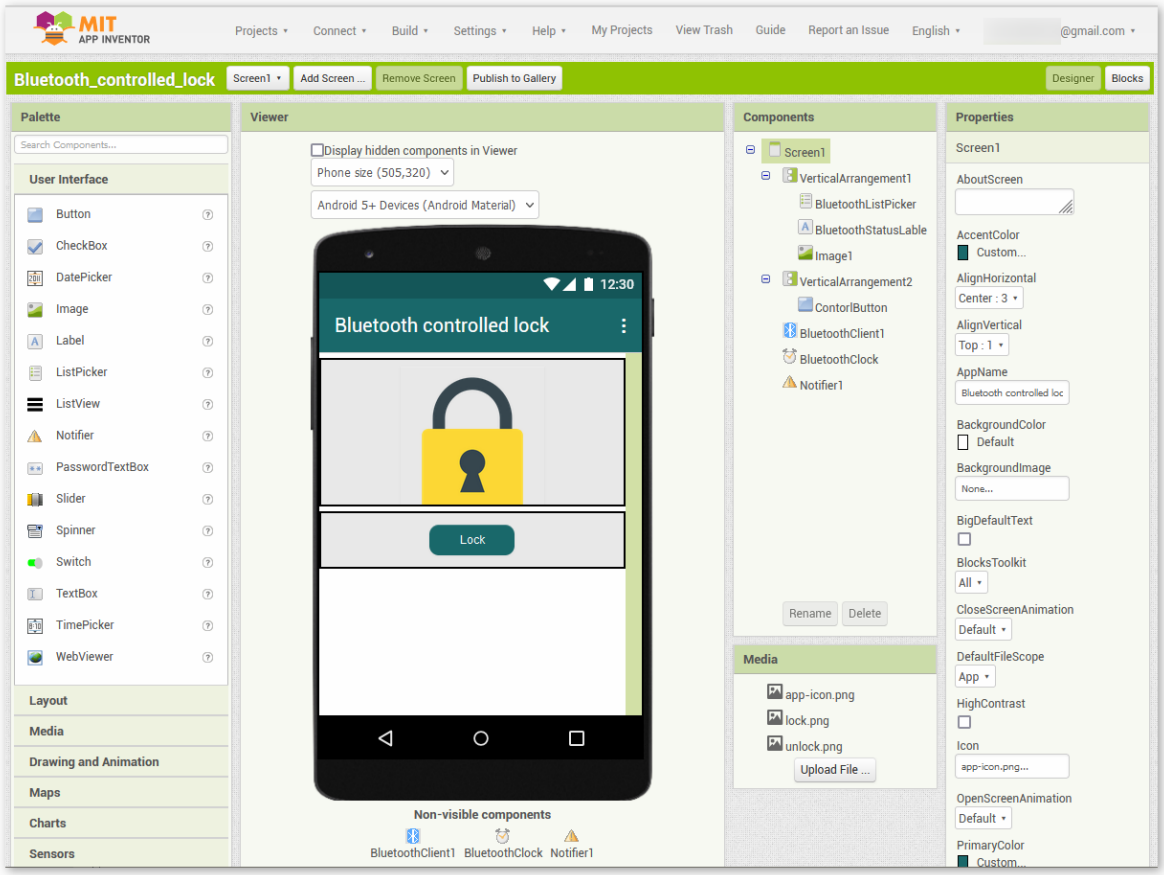


3. Nach dem Hochladen der `.aia`-Datei wird die Anwendung in der MIT App Inventor Software angezeigt. Dies ist eine vorkonfigurierte Vorlage. Sie können diese Vorlage nach dem Vertrautmachen mit dem MIT App Inventor über die folgenden Schritte anpassen.

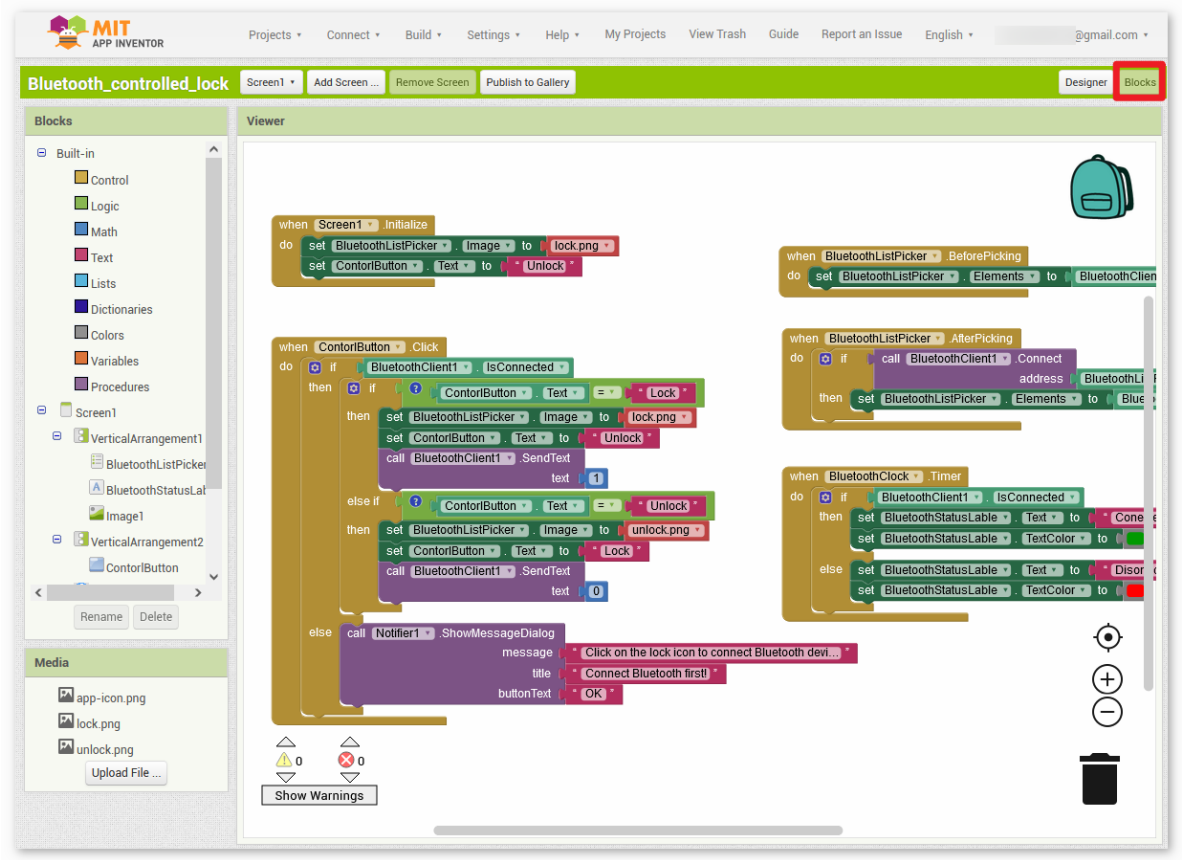
4. Im MIT App Inventor gibt es zwei Hauptbereiche: den **Designer** und die **Blocks**. Sie können in der oberen rechten Ecke der Seite zwischen diesen beiden Bereichen wechseln.



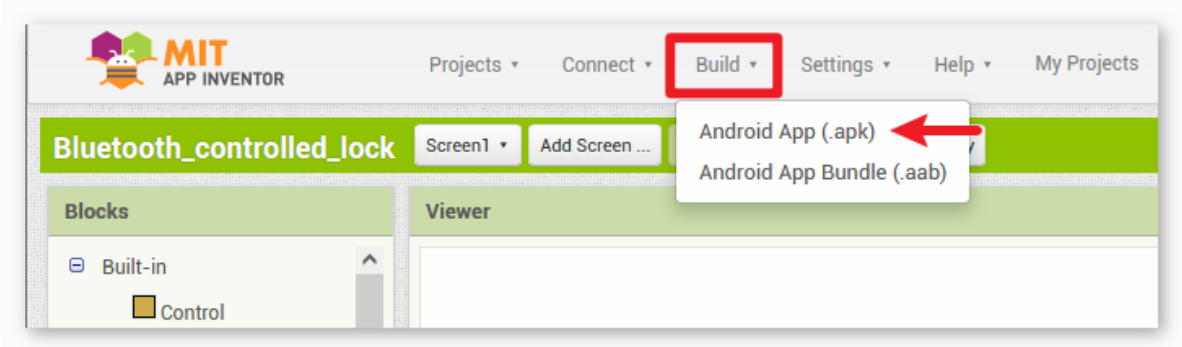
5. Mit dem **Designer** können Sie Schaltflächen, Texte, Bildschirme hinzufügen und das gesamte Aussehen Ihrer Anwendung ändern.



6. Weiter gibt es den Bereich **Blocks**. Dieser Bereich ermöglicht es Ihnen, benutzerdefinierte Funktionen für Ihre App zu erstellen, sodass Sie jedes Element auf der GUI der App programmieren können, um gewünschte Funktionen zu erreichen.



7. Um die Anwendung auf einem Smartphone zu installieren, gehen Sie zum Tab **Build**.



- Sie können eine `.apk`-Datei generieren. Nach Auswahl dieser Option 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 Anwendungsinstallation abzuschließen.

  Sie können auch unsere vorab kompilierte APK hier herunterladen: `Bluetooth_controlled_lock.apk`

- Falls Sie die App im Google Play Store oder einem anderen App-Marktplatz veröffentlichen möchten, können Sie eine `.aab`-Datei generieren.

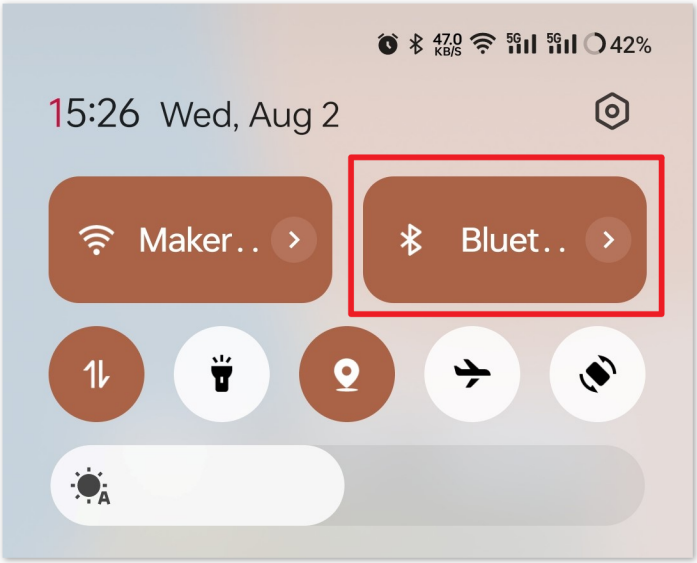
3. Code hochladen

1. Öffnen Sie die Datei `03-Bluetooth_lock_controller.ino` unter dem Pfad `ultimate-sensor-kit\iot_project\bluetooth\03-Bluetooth_lock_controller` oder kopieren Sie diesen Code in die **Arduino IDE**.
2. Wählen Sie das richtige Board und den richtigen Port aus und klicken Sie auf den **Hochladen**-Button.
3. Öffnen Sie den Seriellen Monitor (Baudrate auf **9600** einstellen), um Debug-Nachrichten anzuzeigen.

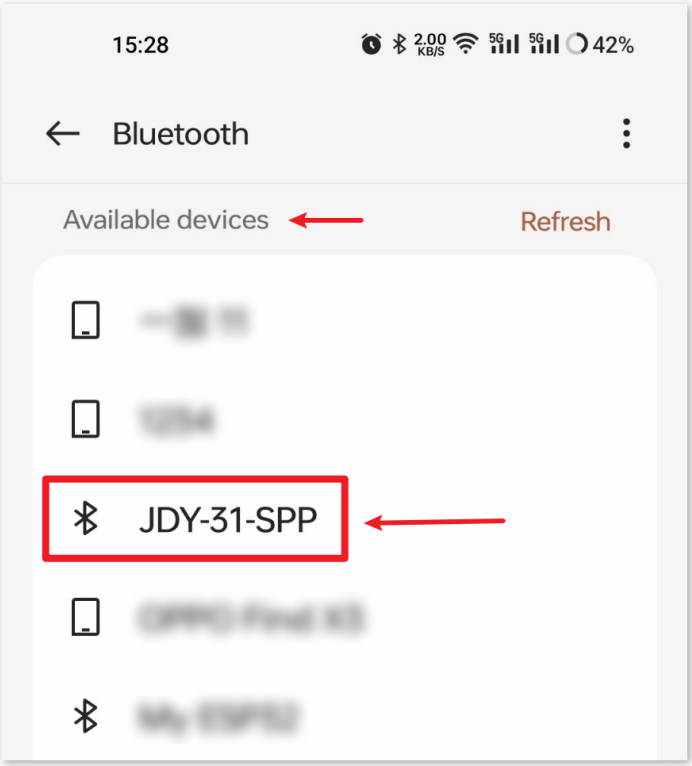
4. Verbindung von App und Bluetooth-Modul

Vergewissern Sie sich, dass die zuvor erstellte Anwendung auf Ihrem Smartphone installiert ist.

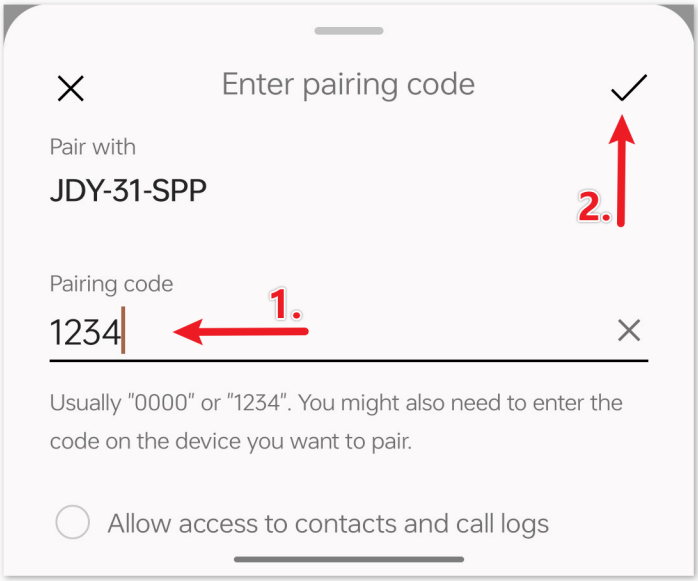
1. Aktivieren Sie zuerst **Bluetooth** auf Ihrem Smartphone.



2. Navigieren Sie zu den **Bluetooth-Einstellungen** auf Ihrem Smartphone und suchen Sie nach Bezeichnungen wie **JDY-31-SPP**.

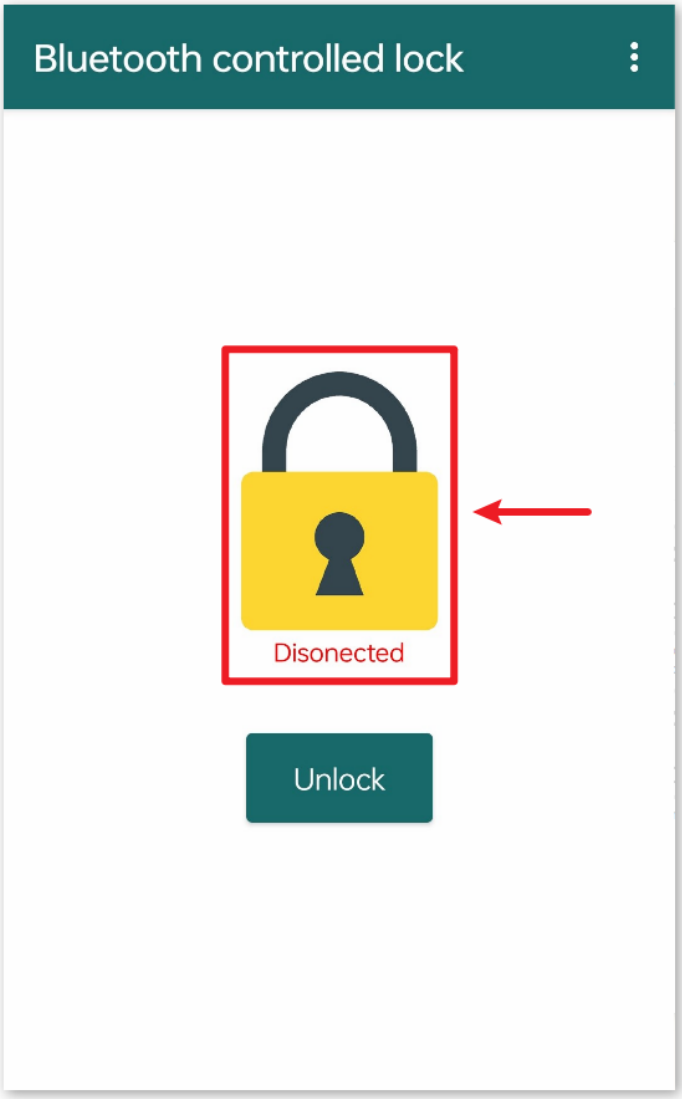


3. Nach dem Anklicken stimmen Sie der **Kopplungsanfrage** im Popup-Fenster zu. Falls ein Kopplungscode erforderlich ist, geben Sie bitte „1234“ ein.

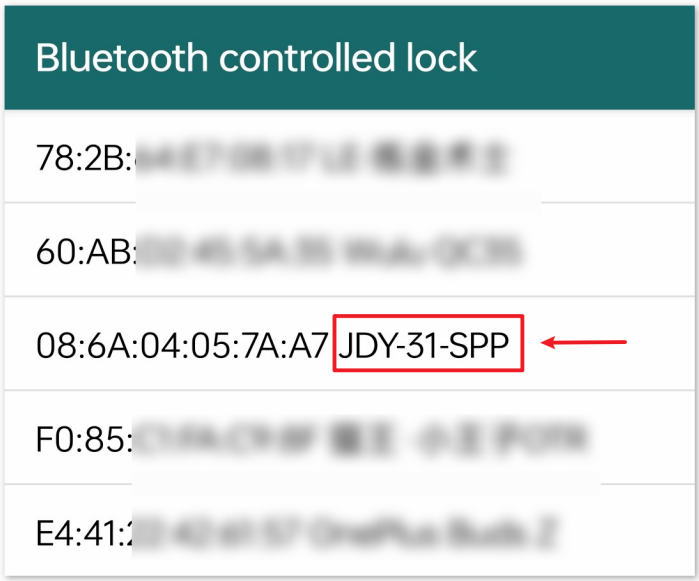


4. Öffnen Sie nun die neu installierte **Control_RGB_LED**-App.

5. In der App klicken Sie auf das **Schloss-Symbol**, um eine Verbindung zwischen der App und dem Bluetooth-Modul herzustellen.

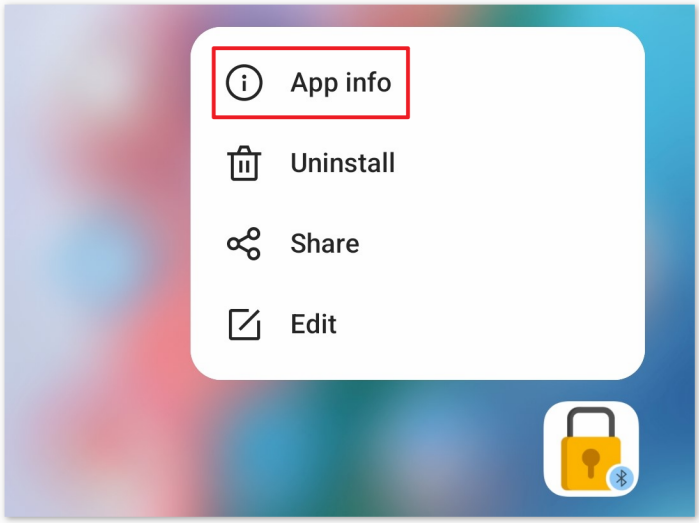


6. Diese Seite zeigt eine Liste aller gekoppelten Bluetooth-Geräte an. Wählen Sie die Option `xx.xx.xx.xx.xx.xx JDY-31-SPP` aus der Liste. Der Name jedes Geräts wird neben seiner MAC-Adresse aufgeführt.

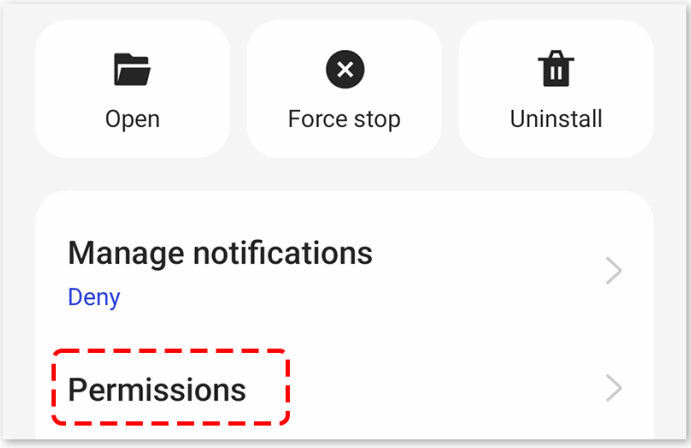


7. Falls Sie auf der oben dargestellten Seite keine Geräte sehen, könnte dies daran liegen, dass dieser App die Berechtigung zur Suche nach nahegelegenen Geräten fehlt. In einem solchen Fall müssen Sie die Einstellungen manuell anpassen.

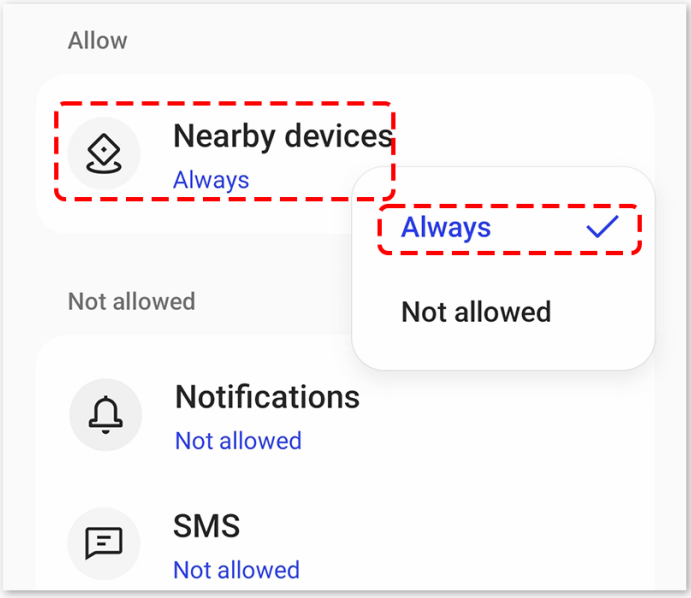
    - Um zur **App-Infoseite** zu gelangen, halten Sie das App-Symbol gedrückt und wählen es aus. Oder nutzen Sie eine andere Methode, um auf diese Seite zu gelangen, falls Ihnen eine bekannt ist.



    - Navigieren Sie zur Seite **Berechtigungen**.

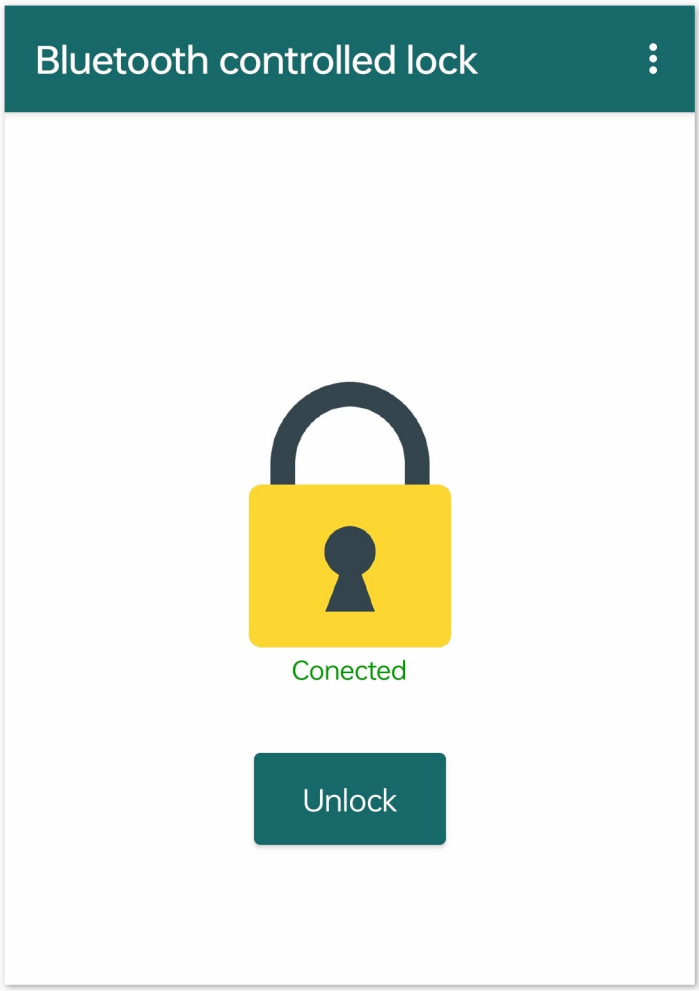


- Um der App das Scannen von nahegelegenen Geräten zu ermöglichen, gehen Sie zu **Nahegelegene Geräte** und wählen **Immer**.



- Starten Sie nun die App neu und wiederholen Sie die Schritte 5 und 6, um eine erfolgreiche Verbindung zu Bluetooth herzustellen.

8. Nach einer erfolgreichen Verbindung werden Sie zur Hauptseite weitergeleitet, auf der „connected“ angezeigt wird. Nun können Sie entweder auf „Unlock“ oder „Lock“ klicken, um den Verriegelungsmechanismus zu steuern.



5. Code-Erklärung

1. Kommunikationspins definieren und SoftwareSerial-Bibliothek initialisieren

```
const int bluetoothTx = 3;
const int bluetoothRx = 4;
SoftwareSerial bleSerial(bluetoothTx, bluetoothRx);
```

Im obigen Code werden die Sendepins (Tx) und Empfangspins (Rx), die vom JDY-31 Bluetooth-Modul zur Kommunikation genutzt werden, definiert. Anschließend wird die SoftwareSerial-Bibliothek initialisiert, die es dem Bluetooth-Modul ermöglicht, mit dem Arduino-Board zu kommunizieren.

2. Servo-relevante Konstanten definieren und Servo-Objekt erstellen

```
const int servoPin = 9;
const int lockAngle = 180;
const int unlockAngle = 90;
Servo myservo;
```

Hier werden der dem Servo zugeordnete Pin sowie die Winkel für die „verriegelte“ und „entriegelte“ Position definiert. Ein Servo-Objekt `myservo` wird ebenfalls für die Steuerung des Servomotors erstellt.

3. Servo und serielle Kommunikation initialisieren

```
void setup() {
  myservo.attach(servoPin);
  Serial.begin(9600);
  bleSerial.begin(9600);
}
```

4. Servo-Steuerung basierend auf Eingaben des Bluetooth-Moduls

```
void loop() {
  if (bleSerial.available() > 0) {
    char message = bleSerial.read();
    if (message == '1') {
      myservo.write(lockAngle);
      Serial.println("Locked");
    }
    else if (message == '0') {
      myservo.write(unlockAngle);
      Serial.println("Unlocked");
    }
  }
}
```

Die `loop()`-Funktion wird fortlaufend ausgeführt. Sie liest eingehende Nachrichten vom Bluetooth-Modul. Wenn die Nachricht ‚1' lautet, wird der Servo in die „verriegelte" Position bewegt. Bei der Nachricht ‚0' wird der Servo in die „entriegelte" Position bewegt. Der aktuelle Status („Verriegelt" oder „Entriegelt") wird im seriellen Monitor ausgegeben.

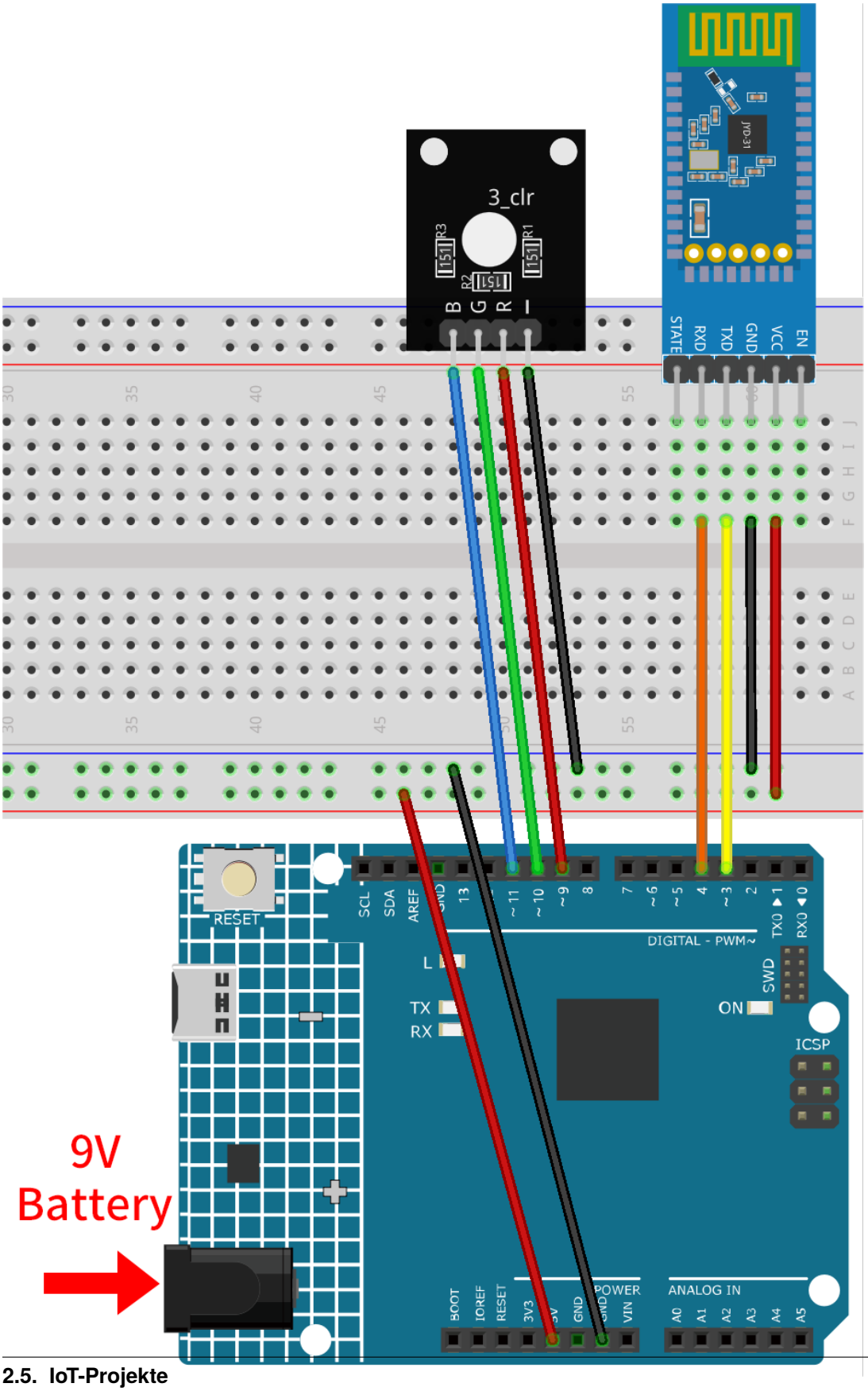
2.5.13 Bluetooth RGB-Controller

Dieses Projekt ermöglicht die Steuerung einer RGB-LED über eine Android-App mittels Bluetooth-Technologie und einem Smartphone.

Für die Android-Anwendung wird eine kostenfreie, webbasierte Plattform namens verwendet. Das Projekt bietet eine hervorragende Gelegenheit, die Schnittstelle zwischen einem Arduino und einem Smartphone kennenzulernen.

Dieses Projekt steuert eine mit einem Arduino Uno verbundene RGB-LED über ein JDY-31 Bluetooth-Modul. Die Android-App sendet verschiedenfarbige Werte an das Arduino Uno-Board via Bluetooth, abhängig von den Benutzereingaben in der grafischen Oberfläche. Das Programm auf dem Uno-Board empfängt RGB-Farbdaten als Zeichen über eine serielle Schnittstelle via Bluetooth und passt die Farbe der LED entsprechend an.

1. Schaltungsaufbau



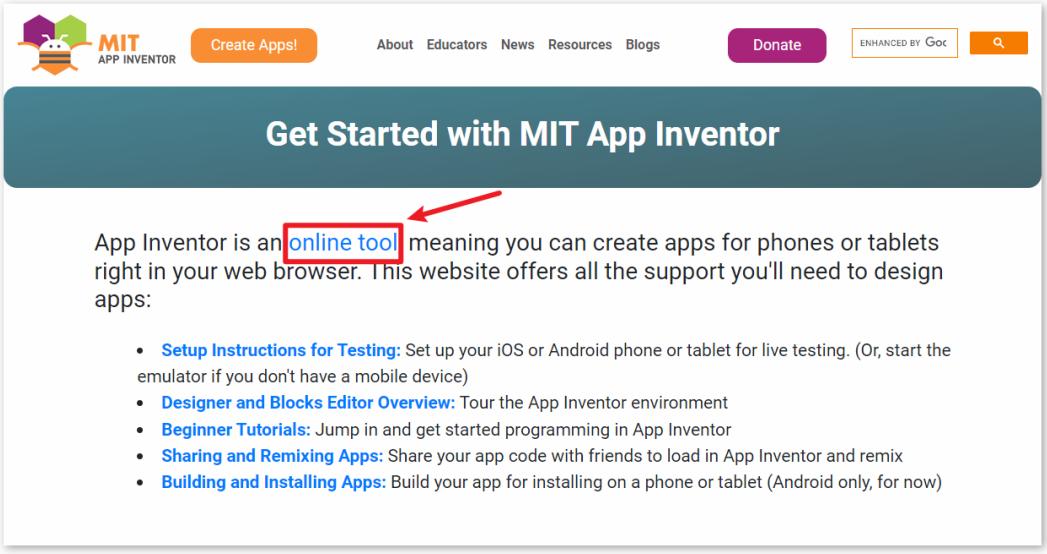
- *Arduino UNO R4 Minima-Platine*
- *JDY-31 Bluetooth-Modul*
- *RGB-Modul*

2. Android-App erstellen

Die Android-Anwendung wird mit einer kostenfreien Web-Anwendung namens entwickelt. MIT App Inventor dient als ausgezeichneter Einstiegspunkt in die Android-Entwicklung, dank seiner intuitiven Drag-and-Drop-Funktionen für die Erstellung einfacher Anwendungen.

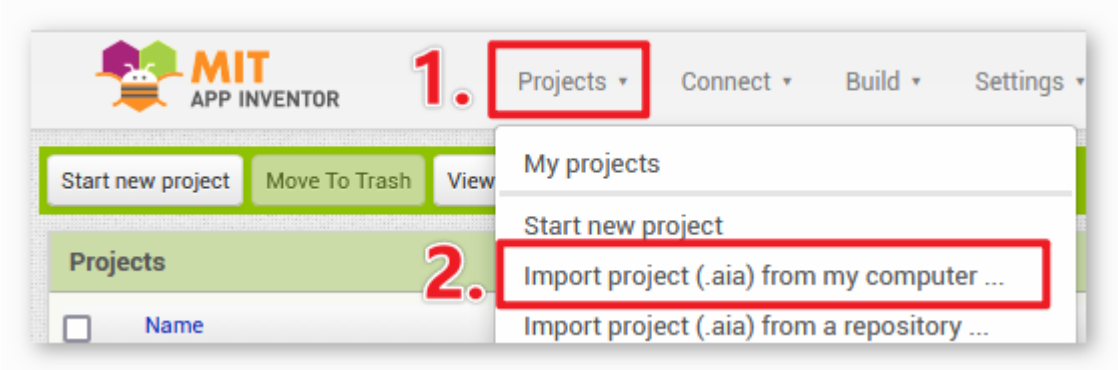
Beginnen wir nun.

1. Besuchen Sie und klicken Sie auf „Online-Tool“ zum Einloggen. Sie benötigen ein Google-Konto, um sich bei MIT App Inventor anzumelden.

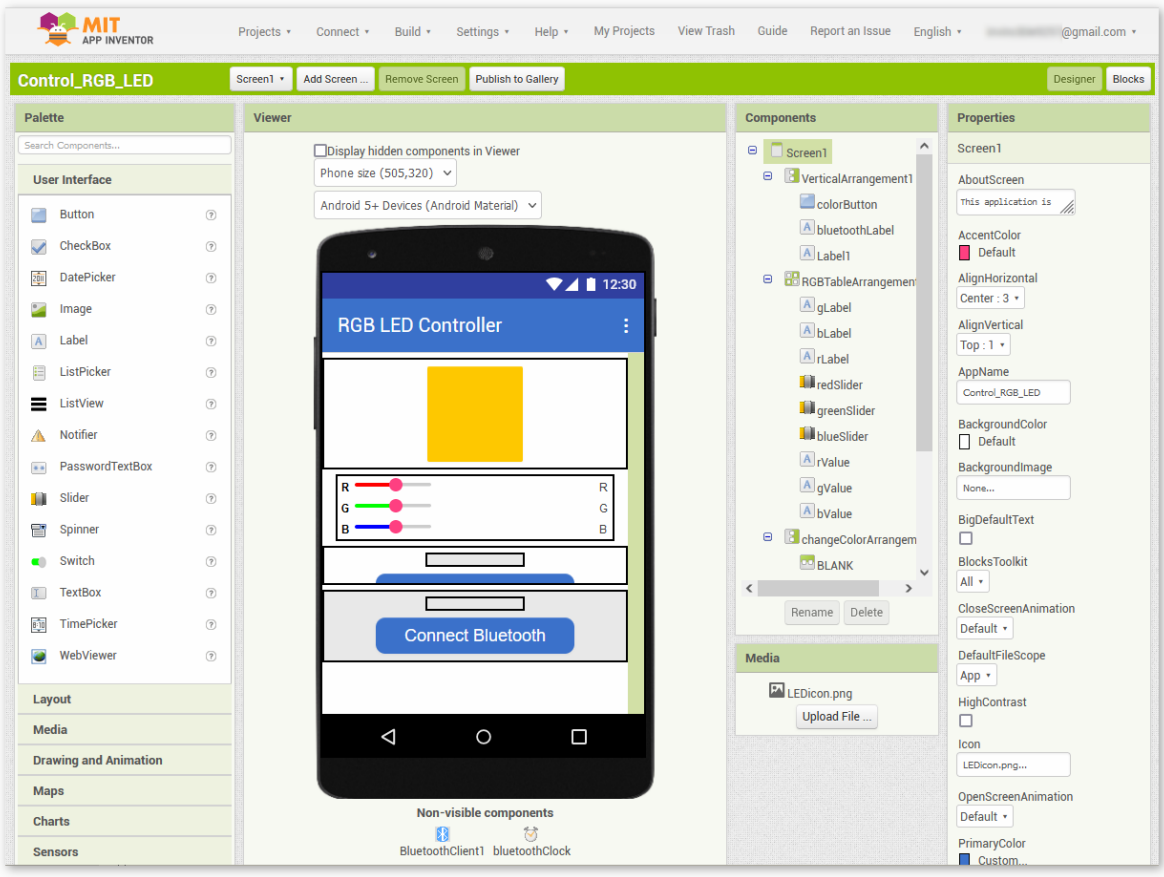


2. Nach dem Einloggen navigieren Sie zu **Projects** -> **Import project (.aia) from my computer**. Anschließend laden Sie die Datei `Control_RGB_LED.aia` hoch, die im Pfad `ultimate-sensor-kit\iot_project\bluetooth\04-Bluetooth_RGB_controller` zu finden ist.

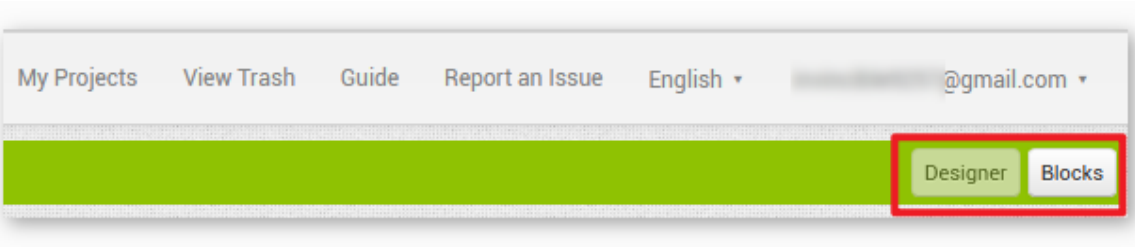
   Sie können sie auch direkt hier herunterladen: `Control_RGB_LED.aia`



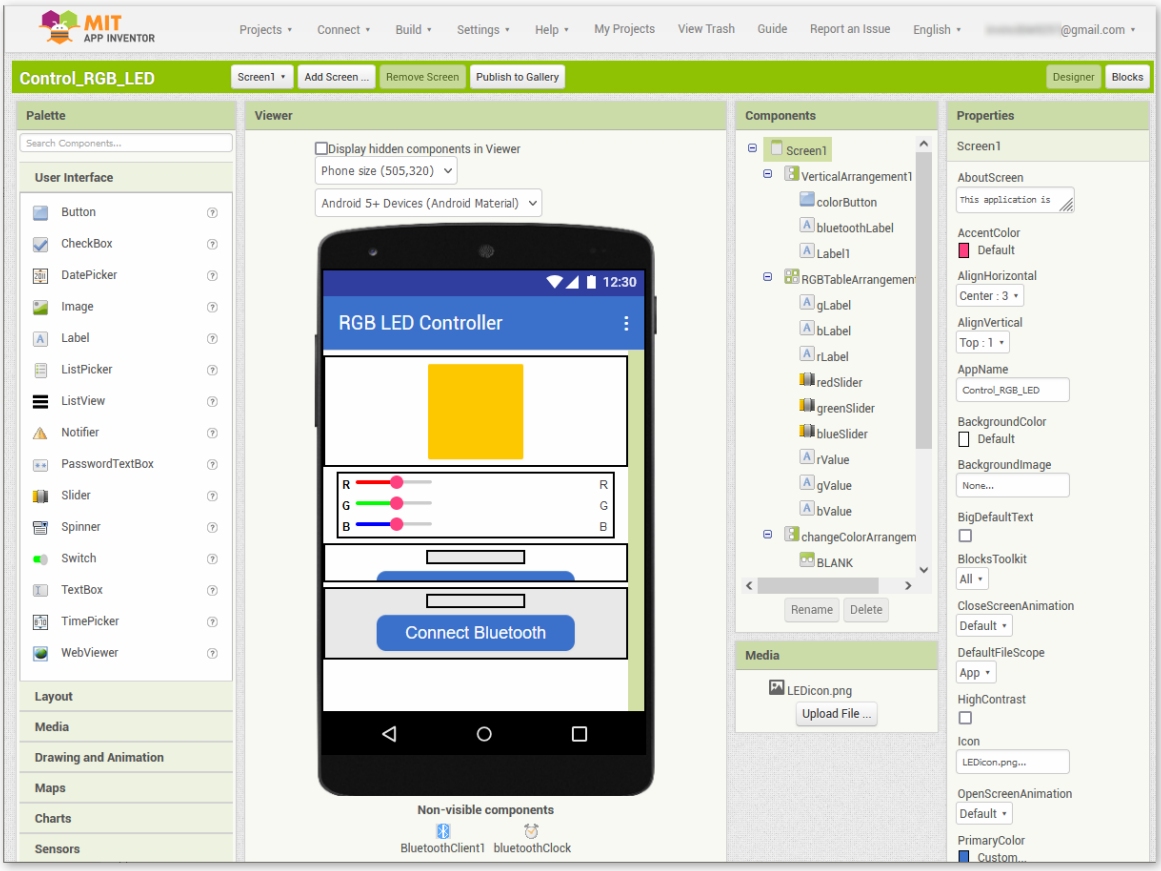
3. Nach dem Hochladen der `.aia`-Datei erscheint die Anwendung in der MIT App Inventor-Software. Das ist eine vorkonfigurierte Vorlage, die Sie nach dem Vertrautwerden mit MIT App Inventor anpassen können.



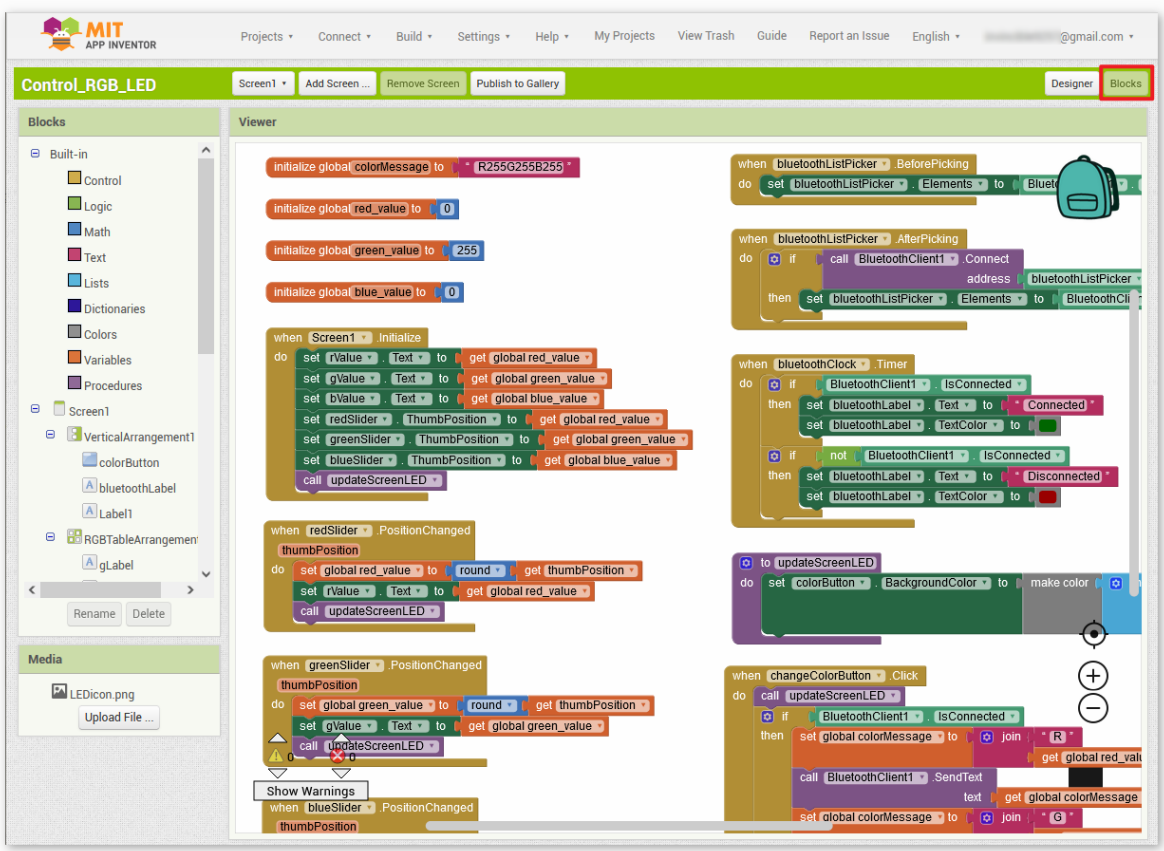
4. Im MIT App Inventor gibt es zwei Hauptbereiche: den **Designer** und die **Blocks**. Sie können zwischen diesen beiden Bereichen in der oberen rechten Ecke der Seite wechseln.



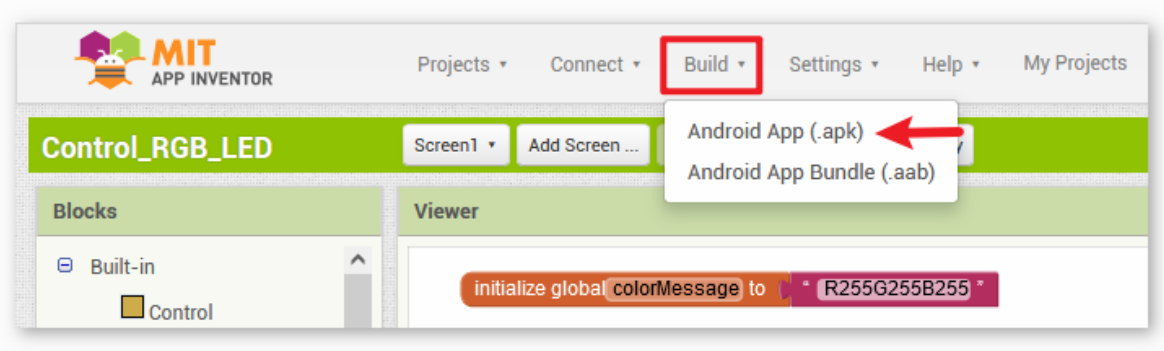
5. Der **Designer** ermöglicht es Ihnen, Buttons, Textfelder, Bildschirme hinzuzufügen und das gesamte Design Ihrer Anwendung zu modifizieren.



6. Anschließend gibt es den Bereich **Blocks**. Hier können Sie spezifische Funktionen für Ihre App programmieren, indem Sie jedes Element in der GUI der App programmieren, um gewünschte Funktionen zu erzielen.



7. Um die Anwendung auf einem Smartphone zu installieren, navigieren Sie zur **Build**-Registerkarte.



- Sie können eine `.apk`-Datei generieren. Nach der Auswahl erscheint eine Seite, auf der Sie zwischen dem Herunterladen einer `.apk`-Datei oder dem Scannen eines QR-Codes für die Installation wählen können. Folgen Sie der Installationsanleitung, um die Anwendungsinstallation abzuschließen.

  Sie können auch unsere vorab kompilierte APK hier herunterladen: `Control_RGB_LED.apk`

- Falls Sie die App im Google Play Store oder einem anderen App-Marktplatz veröffentlichen möchten, können Sie eine `.aab`-Datei generieren.

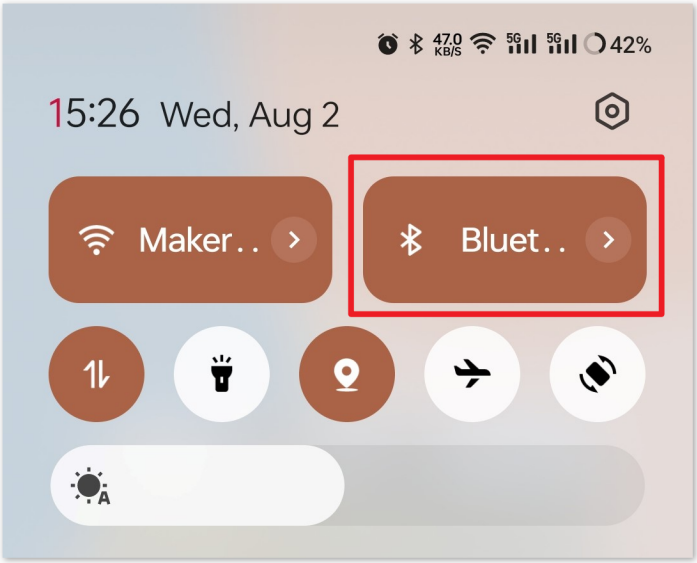
3. Code hochladen

1. Öffnen Sie die Datei `04-Bluetooth_RGB_controller.ino` im Pfad `ultimate-sensor-kit\iot_project\bluetooth\04-Bluetooth_RGB_controller`, oder kopieren Sie diesen Code in die **Arduino IDE**.
2. Nach der Auswahl des korrekten Boards und Ports klicken Sie auf den **Hochladen**-Button.
3. Öffnen Sie den Seriellen Monitor (Baudrate auf **9600** einstellen), um Debug-Nachrichten anzuzeigen.

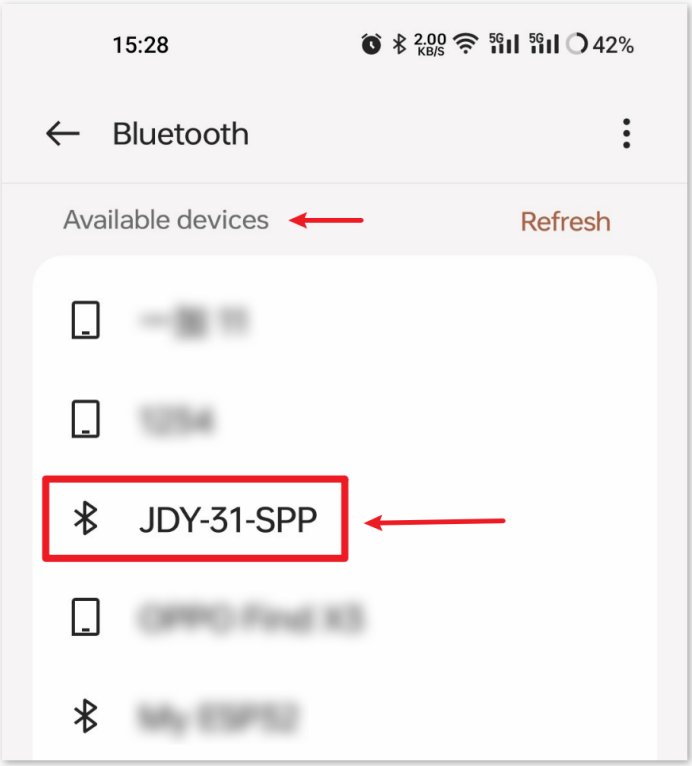
4. Verbindung zwischen App und Bluetooth-Modul herstellen

Stellen Sie sicher, dass die zuvor erstellte Anwendung auf Ihrem Smartphone installiert ist.

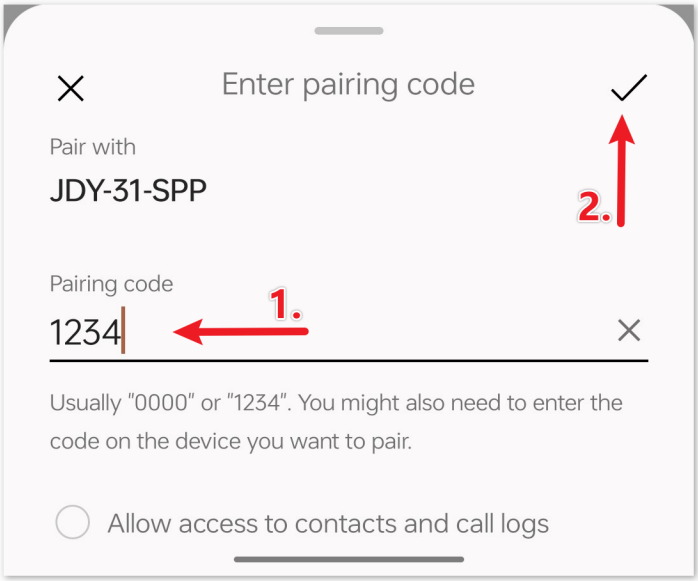
1. Aktivieren Sie zunächst **Bluetooth** auf Ihrem Smartphone.



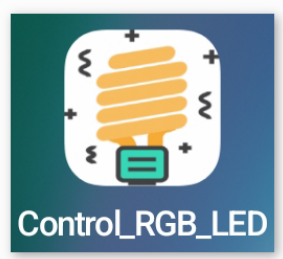
2. Navigieren Sie zu den **Bluetooth-Einstellungen** auf Ihrem Smartphone und suchen Sie nach Bezeichnungen wie **JDY-31-SPP**.



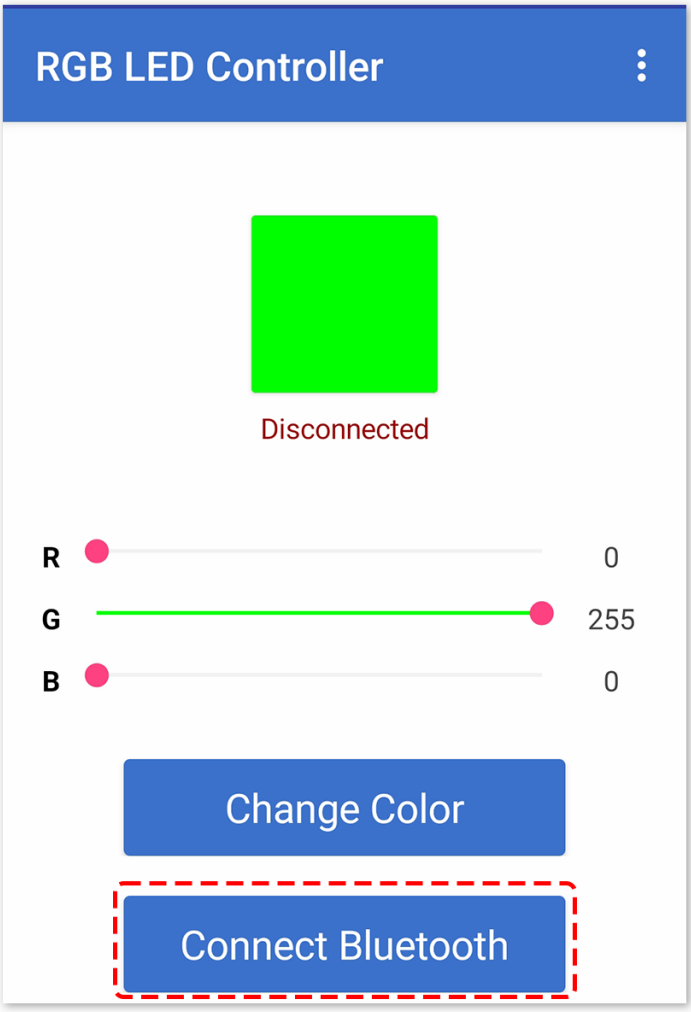
3. Nach dem Anklicken stimmen Sie der **Pairing-Anfrage** im Popup-Fenster zu. Falls nach einem Pairing-Code gefragt wird, geben Sie „1234“ ein.



4. Öffnen Sie nun die neu installierte **Control_RGB_LED** App.



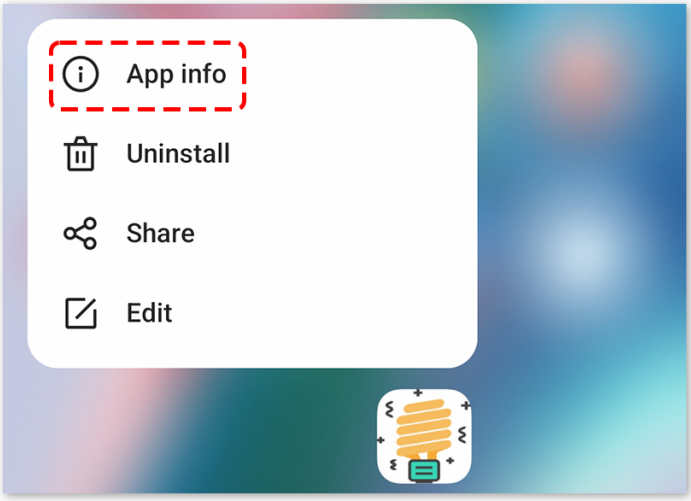
5. In der App klicken Sie auf **Connect Bluetooth**, um eine Verbindung zwischen der App und dem Bluetooth-Modul herzustellen.



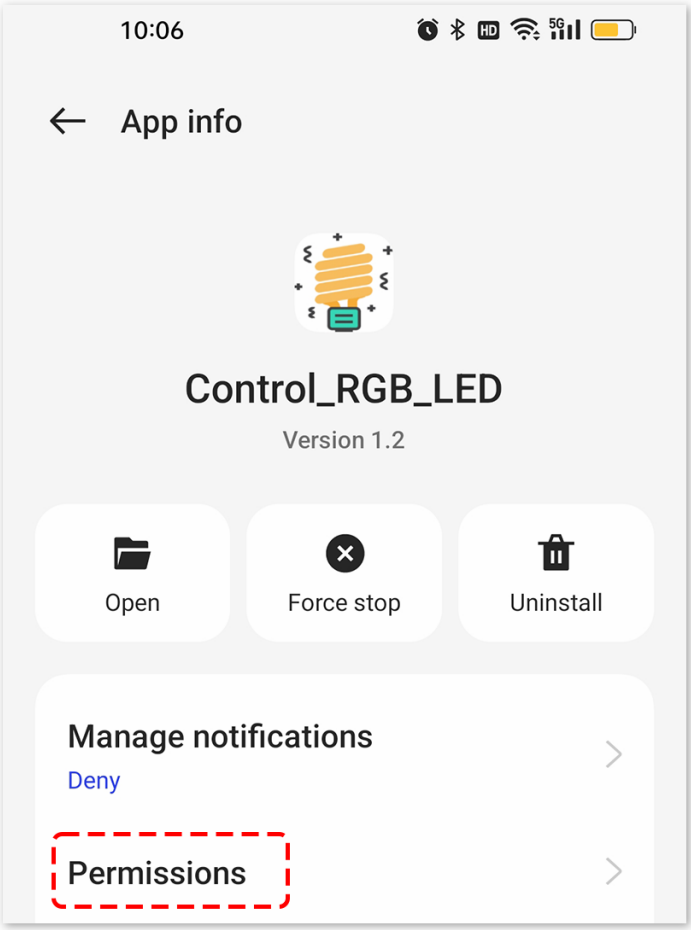
6. Diese Seite zeigt eine Liste aller gekoppelten Bluetooth-Geräte. Wählen Sie die Option `xx.xx.xx.xx.xx.xx JDY-31-SPP` aus der Liste aus. Der Name jedes Geräts wird neben seiner MAC-Adresse angezeigt.

7. Wenn auf der oben gezeigten Seite keine Geräte angezeigt werden, könnte dies daran liegen, dass dieser App die Berechtigung fehlt, nach nahegelegenen Geräten zu suchen. In diesem Fall müssen Sie die Einstellungen manuell anpassen.

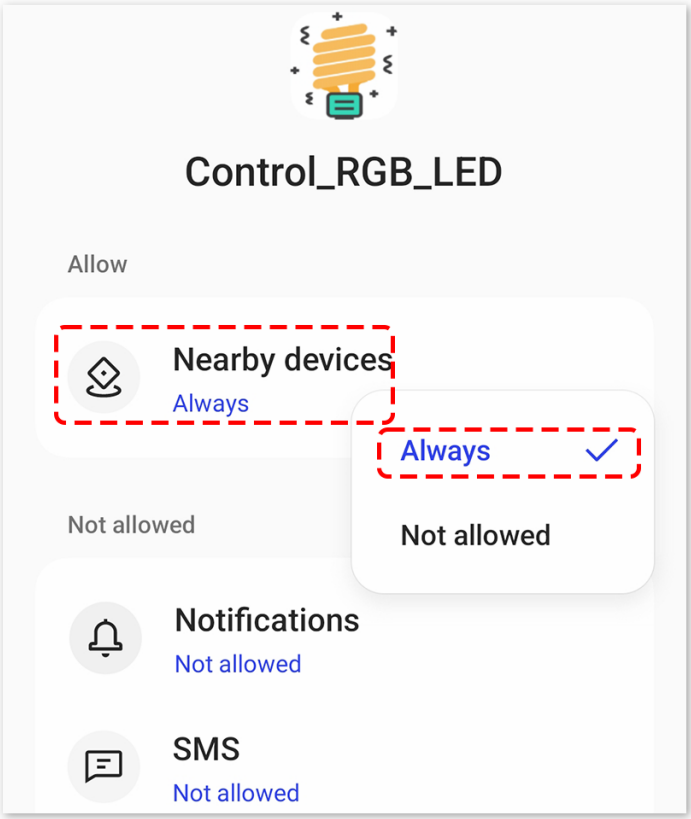
    - Um zur Seite **App-Info** zu gelangen, halten Sie das App-Symbol gedrückt und wählen Sie es aus. Alternativ können Sie jede andere Methode verwenden, um zu dieser Seite zu gelangen.



    - Navigieren Sie zur Seite **Berechtigungen**.

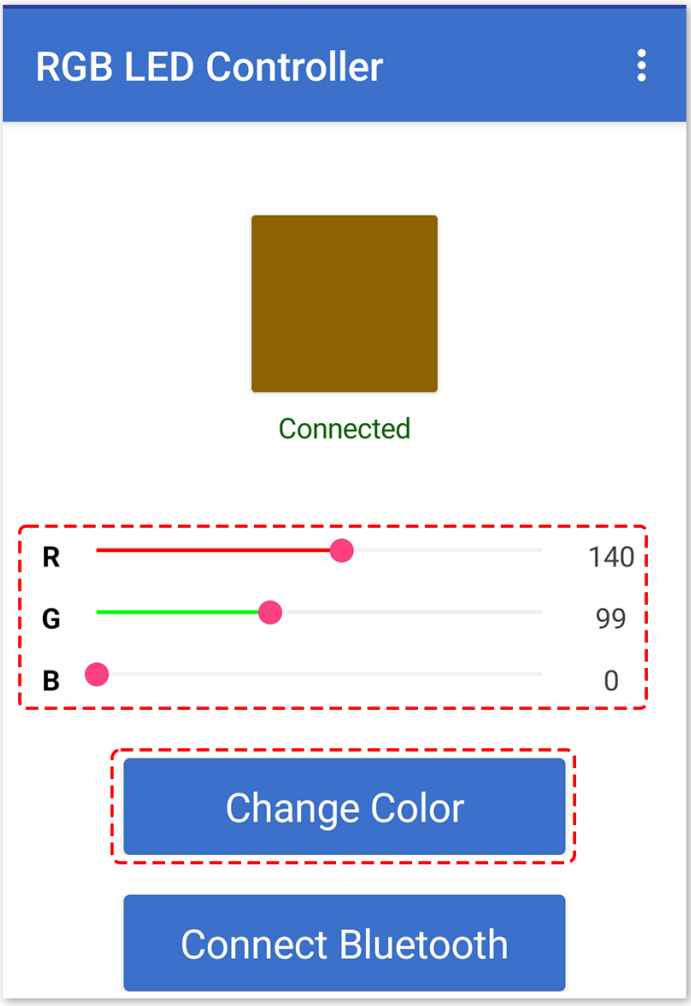


- Um der App das Scannen von nahegelegenen Geräten zu ermöglichen, gehen Sie zu **Nahegelegene Geräte** und wählen Sie **Immer**.



- 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 zur Hauptseite weitergeleitet, auf der „connected“ angezeigt wird. Von dort aus können Sie problemlos die RGB-Werte ändern und die Displayfarbe durch Klicken auf die Schaltfläche **Change Color** anpassen.



5. Code-Erklärung

1. Einrichten des Bluetooth-Moduls und Initialisieren der Variablen:

   Der Code beginnt mit dem Einbinden der `SoftwareSerial`-Bibliothek und der Initialisierung der notwendigen Variablen.

```
#include <SoftwareSerial.h>
SoftwareSerial bleSerial(3, 4);  //Rx,Tx

#define max_char 12
char message[max_char];
char r_char;
byte currentIndex = 0;

const int redPin = 9;
const int greenPin = 10;
const int bluePin = 11;

int redValue = 0;
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
int greenValue = 255;
int blueValue = 0;

String redTempValue;
String greenTempValue;
String blueTempValue;

int flag = 0;
char currentColor;
```

2. Funktion setup():

   Hier werden die Pins für die RGB-LED als Ausgangspins festgelegt, und die serielle Kommunikation wird mit einer Baudrate von 9600 sowohl für die Hauptserielle Schnittstelle des Arduino als auch für das Bluetooth-Modul initialisiert.

```
void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  Serial.begin(9600);
  bleSerial.begin(9600);
}
```

3. Lesen und Verarbeiten der Daten:

   In der Hauptprogrammschleife prüft der Code kontinuierlich auf eingehende Daten vom Bluetooth-Modul. Bei Empfang von Daten werden die Zeichen verarbeitet, um die RGB-Werte zu identifizieren und die Farbe der RGB-LED entsprechend einzustellen.

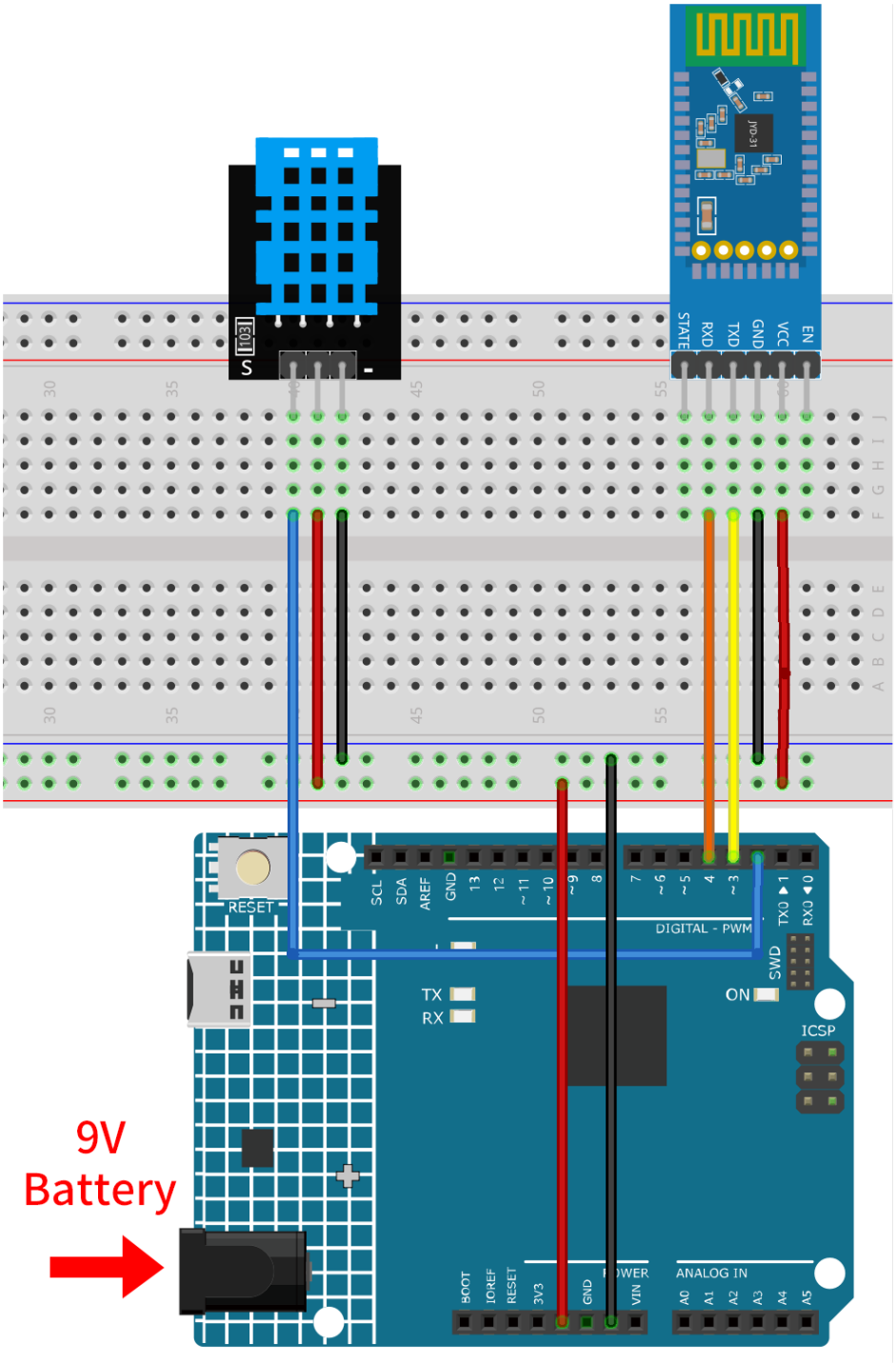
```
void loop() {
  while (bleSerial.available() > 0) {
    ... [Datenlesen und -verarbeiten]
  }

  if (flag == 0) {
    Serial.println(message);
    analogWrite(redPin, redTempValue.toInt());
    analogWrite(greenPin, greenTempValue.toInt());
    analogWrite(bluePin, blueTempValue.toInt());

    flag = 1;

    for (int i = 0; i < 12; i++) {
      message[i] = '\0';
    }
    currentIndex = 0;
  }
}
```

2.5.14 Bluetooth-Umweltmonitor

Dieses Projekt nutzt eine Android-App, die mit dem MIT App Inventor erstellt wurde, um Umweltdaten von einem Arduino-Board zu empfangen und anzuzeigen. Das Arduino-Board sammelt Daten von einem DHT11-Sensor, um Temperatur und Luftfeuchtigkeit zu messen. Diese Daten werden dann über das JDY-31-Modul via Bluetooth übertragen. Die App zeigt die Daten auf dem Bildschirm an, sobald sie empfangen werden.

Die Android-Anwendung wird mit einer kostenlosen Webplattform namens erstellt. Das Projekt bietet eine hervorragende Gelegenheit, Erfahrungen in der Schnittstellenanbindung eines Arduino-Boards an ein Smartphone zu sammeln.

1. Den Schaltkreis aufbauen



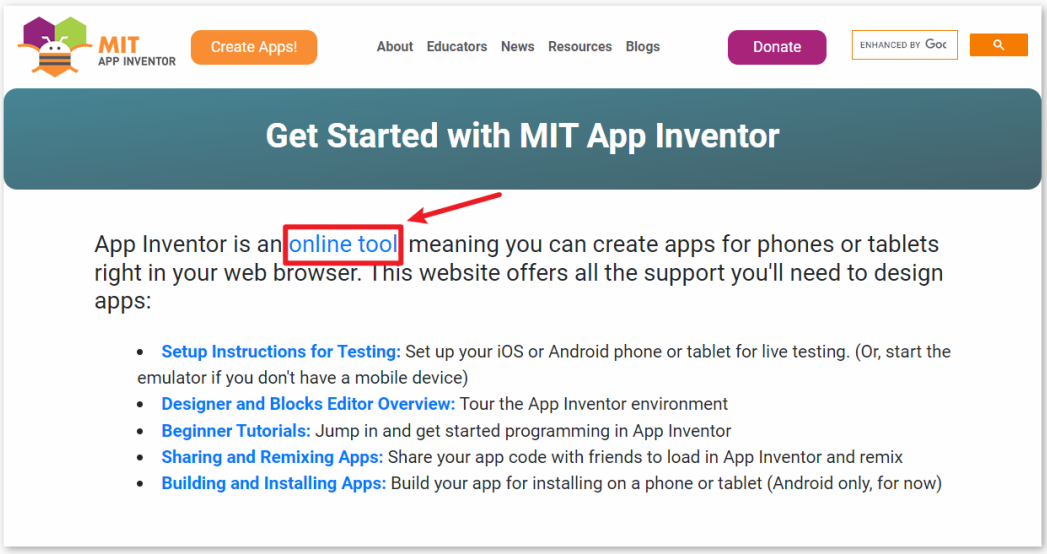
- *Arduino UNO R4 Minima-Platine*
- *JDY-31 Bluetooth-Modul*
- *Temperatur- und Feuchtigkeitssensor-Modul (DHT11)*

2. Die Android-App erstellen

Die Android-Anwendung wird mit einer kostenlosen Webanwendung namens entwickelt. Der MIT App Inventor ist ein ausgezeichneter Einstiegspunkt für die Android-Entwicklung, da er intuitive Drag-and-Drop-Funktionen für die Erstellung einfacher Anwendungen bietet.

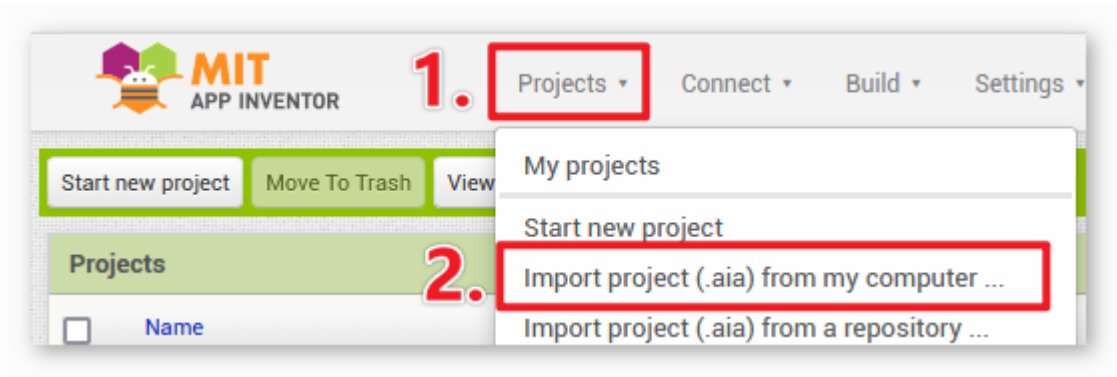
Jetzt geht's los.

1. Besuchen Sie und klicken Sie auf „Online-Tool", um sich anzumelden. Sie benötigen ein Google-Konto, um sich beim MIT App Inventor zu registrieren.



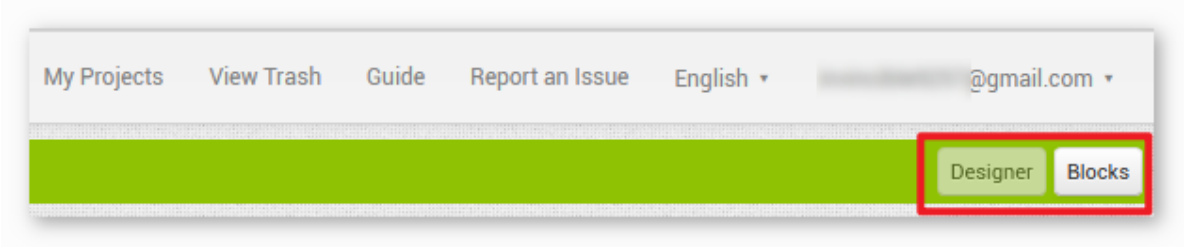
2. Nach der Anmeldung navigieren Sie zu **Projects** -> **Import project (.aia) from my computer**. Laden Sie anschließend die Datei `Bluetooth_controlled_lock.aia` hoch, die im Verzeichnis `ultimate-sensor-kit\iot_project\bluetooth\05-Bluetooth_environmental_monitor` zu finden ist.

   Sie können die Datei auch direkt hier herunterladen: `Bluetooth_Environmental_Monitor.aia`

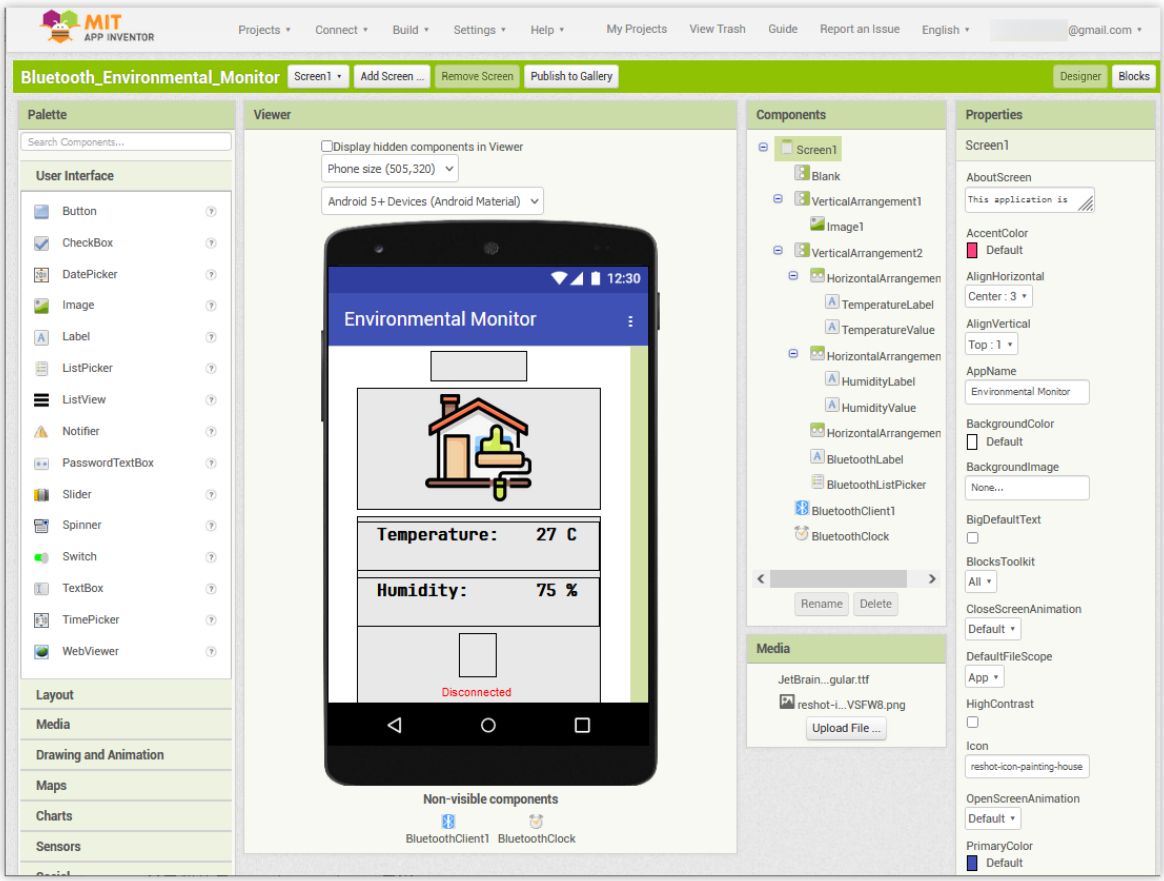


3. Nach dem Hochladen der `.aia`-Datei wird die Anwendung im MIT App Inventor angezeigt. Es handelt sich hierbei um eine vorkonfigurierte Vorlage, die Sie nach einer Einarbeitungsphase im MIT App Inventor anpassen können.

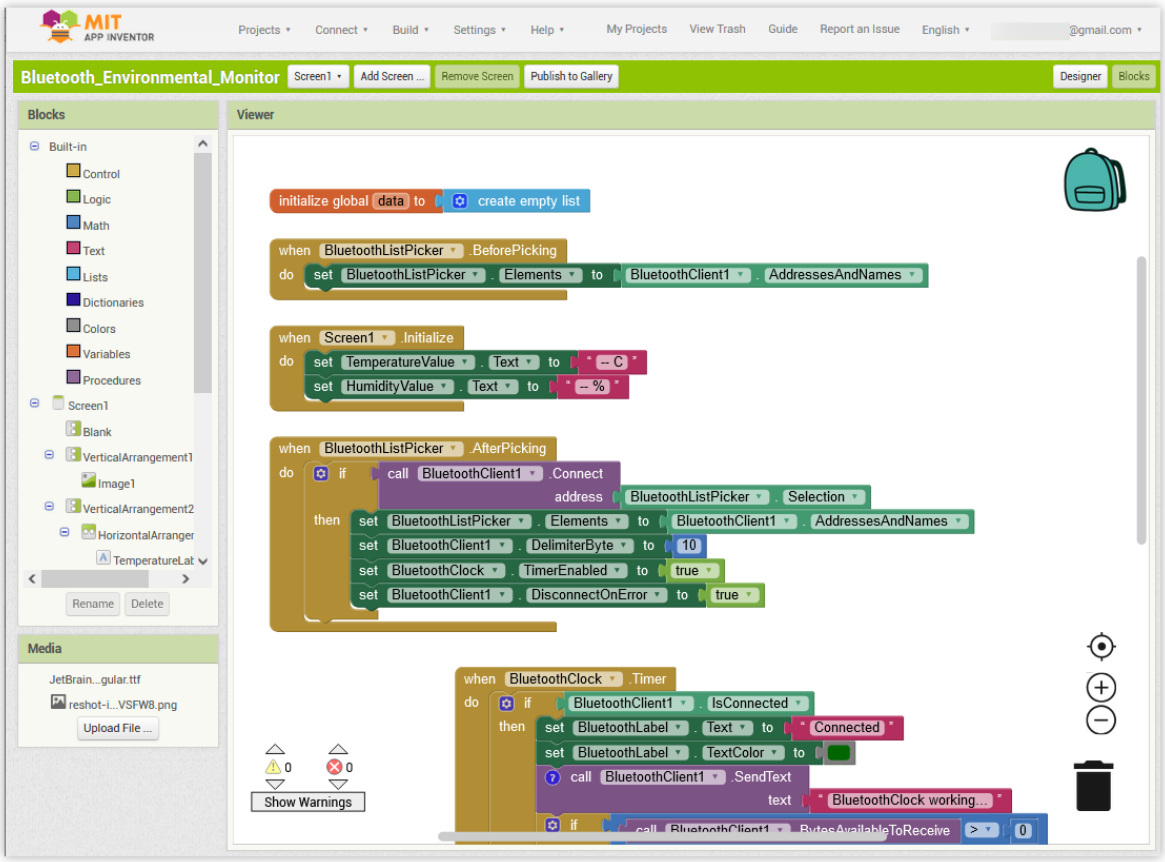
4. Im MIT App Inventor gibt es zwei Hauptbereiche: den **Designer** und die **Blocks**. Sie können zwischen diesen beiden Bereichen in der oberen rechten Ecke der Seite wechseln.



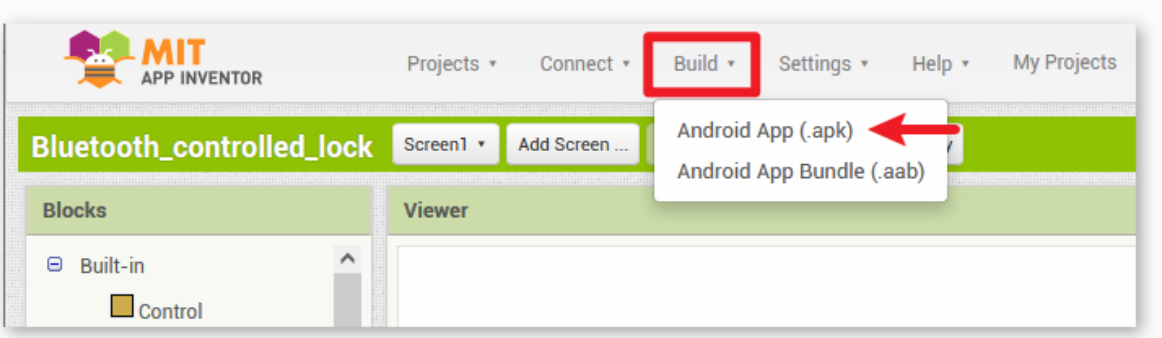
5. Der **Designer** ermöglicht das Hinzufügen von Buttons, Texten, Bildschirmen und das Anpassen des Gesamtdesigns Ihrer Anwendung.



6. Als Nächstes gibt es den Bereich **Blocks**. Hier können Sie benutzerdefinierte Funktionen für Ihre App erstellen und so jedes Element der Benutzeroberfläche der App programmieren, um gewünschte Funktionen zu erreichen.



7. Um die Anwendung auf einem Smartphone zu installieren, navigieren Sie zum **Build**-Tab.



- Sie können eine `.apk`-Datei generieren. Wählen Sie diese Option, erscheint eine Seite, die Ihnen die Wahl lässt, entweder eine `.apk`-Datei herunterzuladen oder einen QR-Code für die Installation zu scannen. Folgen Sie der Installationsanleitung, um die Installation der Anwendung abzuschließen.

  Sie können auch unsere bereits kompilierte APK hier herunterladen: `Bluetooth_Environmental_Monitor.apk`

- Wenn Sie die App im Google Play Store oder einem anderen App-Marktplatz veröffentlichen möchten, können Sie eine `.aab`-Datei generieren.

3. Hochladen des Codes

1. Öffnen Sie die Datei `05-Bluetooth_environmental_monitor.ino` im Pfad `ultimate-sensor-kit\iot_project\bluetooth\05-Bluetooth_environmental_monitor` oder kopieren Sie den Code in die **Arduino IDE**.

   Bemerkung: Um die Bibliothek zu installieren, verwenden Sie den Arduino-Bibliotheksmanager und suchen Sie nach **„DHT sensor library“** und installieren Sie diese.

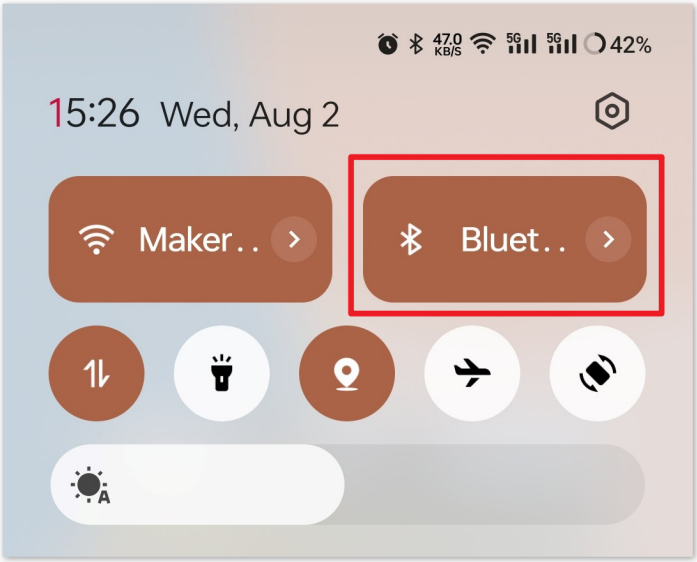
   ---

2. Wählen Sie das korrekte Board und den Port aus und klicken Sie auf die Schaltfläche **Hochladen**.
3. Öffnen Sie den seriellen Monitor (Baudrate auf **9600** einstellen), um Debug-Nachrichten zu sehen.

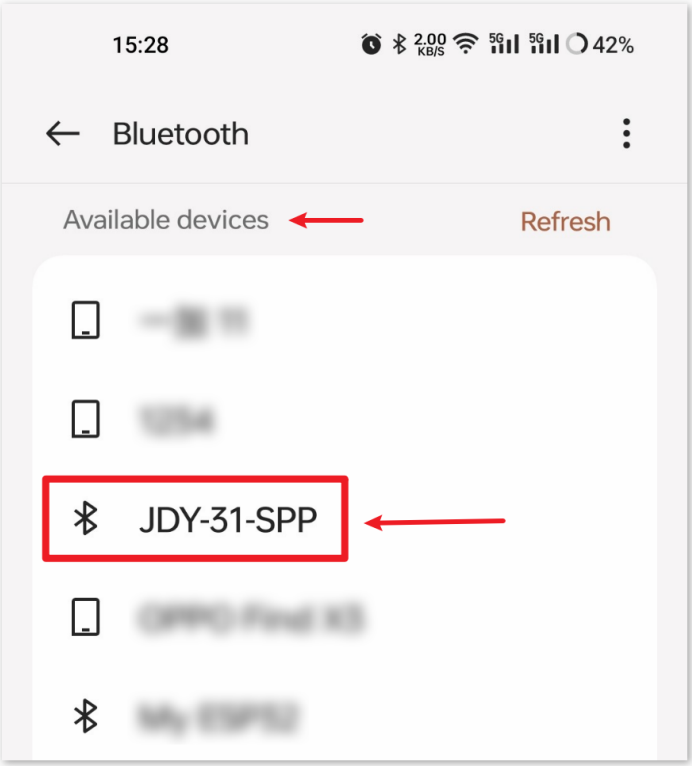
4. Verbindung der App und des Bluetooth-Moduls

Vergewissern Sie sich, dass die zuvor erstellte Anwendung auf Ihrem Smartphone installiert ist.

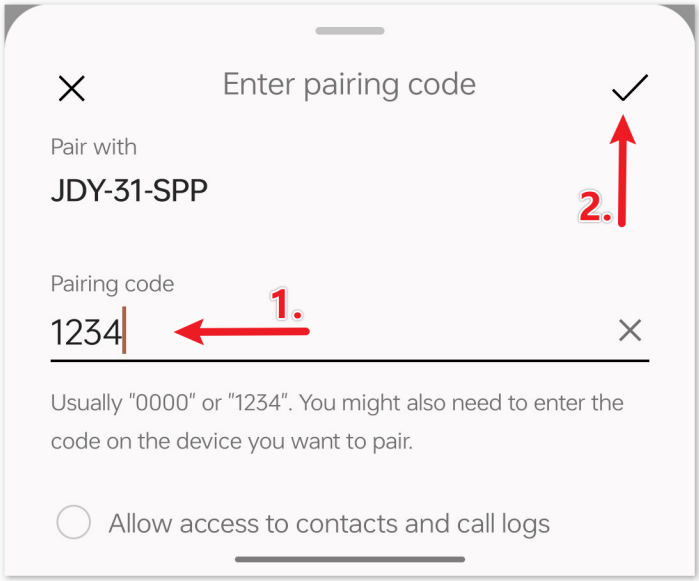
1. Aktivieren Sie zunächst **Bluetooth** auf Ihrem Smartphone.



2. Navigieren Sie zu den **Bluetooth-Einstellungen** auf Ihrem Smartphone und suchen Sie nach Namen wie **JDY-31-SPP**.

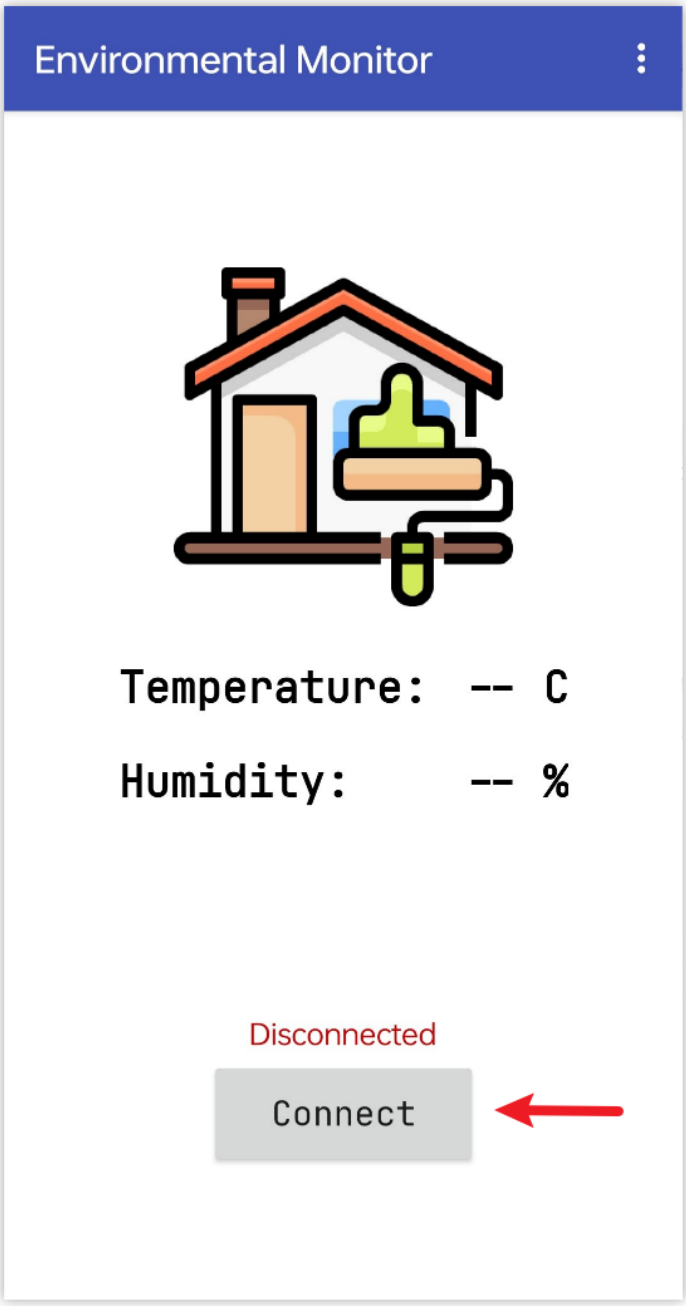


3. Nach dem Anklicken stimmen Sie der **Kopplungsanfrage** im aufspringenden Fenster zu. Falls ein Kopplungscode erforderlich ist, geben Sie „1234“ ein.

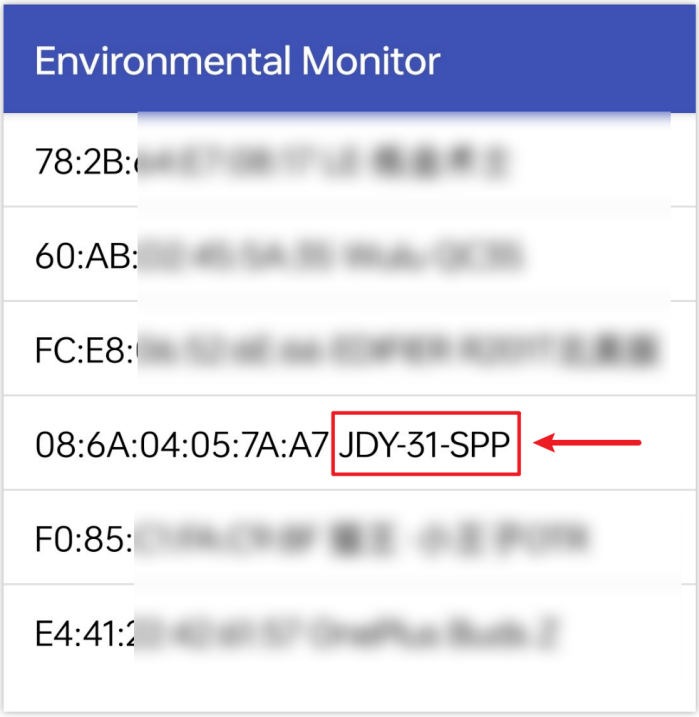


4. Öffnen Sie nun die neu installierte App **Environmental Monitor**.

5. In der App klicken Sie auf die Schaltfläche **Connect**, um eine Verbindung zwischen der App und dem Bluetooth-Modul herzustellen.

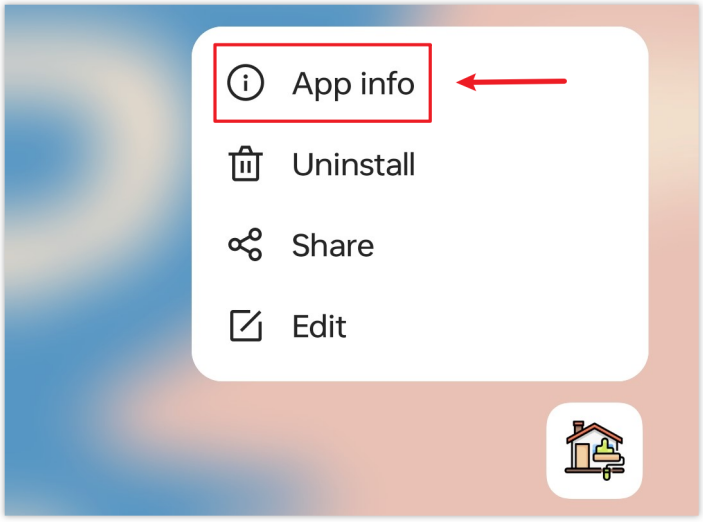


6. Diese Seite zeigt eine Liste aller gekoppelten Bluetooth-Geräte an. Wählen Sie die Option `xx.xx.xx.xx.xx.xx JDY-31-SPP` aus der Liste. Der Name jedes Geräts steht neben seiner MAC-Adresse.

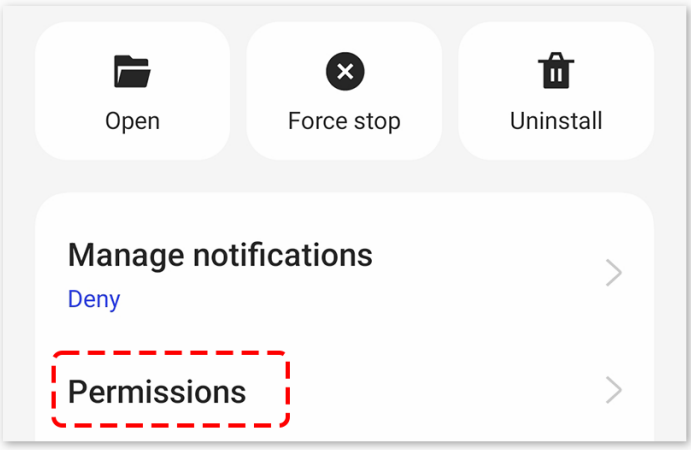


7. Wenn Sie auf der oben gezeigten Seite keine Geräte sehen, könnte das daran liegen, dass der App nicht erlaubt ist, nach nahegelegenen Geräten zu suchen. In diesem Fall müssen Sie die Einstellungen manuell anpassen.

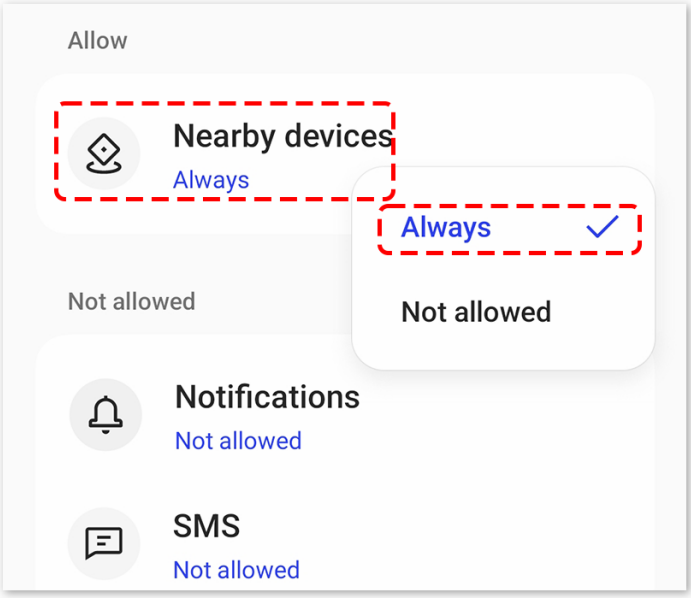
   - Um zur **App-Info-Seite** zu gelangen, halten Sie das App-Symbol gedrückt und wählen Sie es aus. Alternativ können Sie auch eine andere Methode verwenden, um auf diese Seite zu gelangen.



   - Navigieren Sie zur Seite **Berechtigungen**.

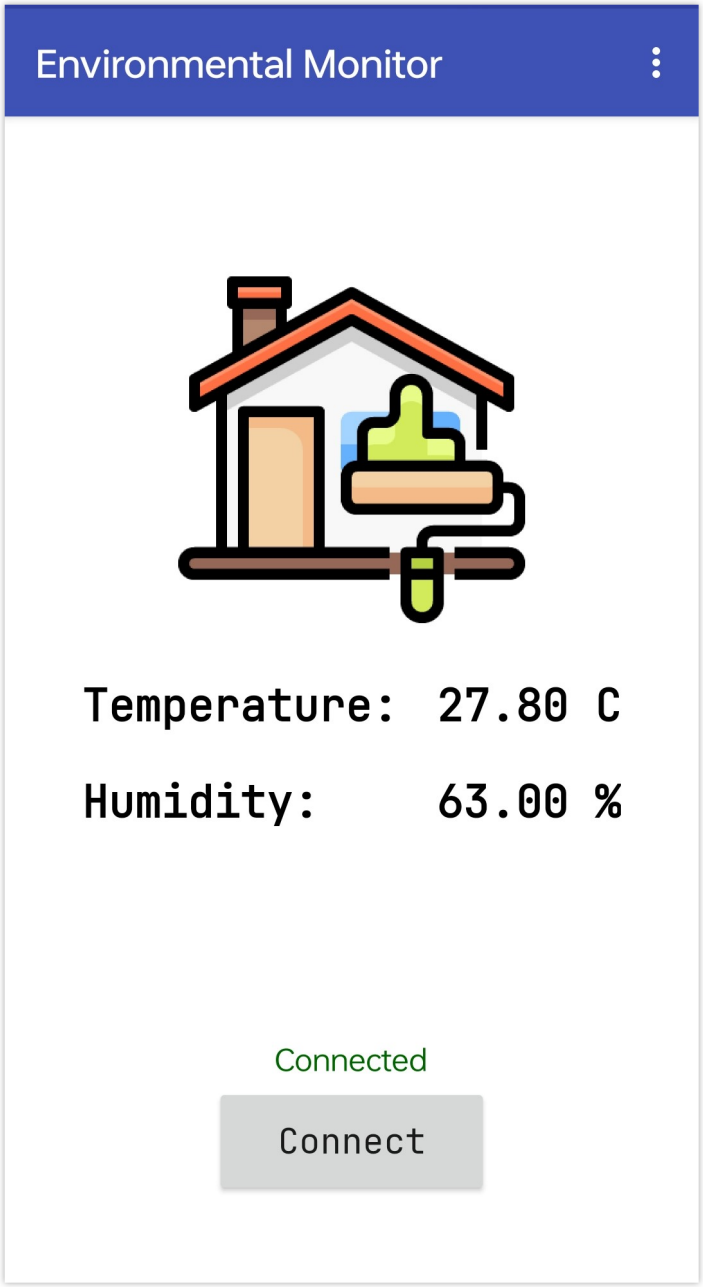


- Um der App die Suche nach nahegelegenen Geräten zu ermöglichen, gehen Sie zu **Nahegelegene Geräte** und wählen Sie **Immer**.



- Starten Sie nun die App neu und wiederholen Sie die Schritte 5 und 6, um erfolgreich eine Bluetooth-Verbindung herzustellen.

8. Nach einer erfolgreichen Verbindung werden Sie zur Hauptseite weitergeleitet, auf der Temperatur und Luftfeuchtigkeit angezeigt werden.



5. Code-Erläuterung

1. Einrichtung der Bluetooth-Kommunikation und des DHT11-Sensors.

```
#include <SoftwareSerial.h>
const int bluetoothTx = 3;
const int bluetoothRx = 4;
SoftwareSerial bleSerial(bluetoothTx, bluetoothRx);

#include <DHT.h>
#define DHTPIN 2
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
```

Der Code bindet die erforderlichen Bibliotheken ein und definiert die Pins für das Bluetooth-Modul sowie den DHT11-Sensor. Zudem werden Objekte für die Bluetooth-Kommunikation und den DHT11 deklariert.

2. Initialisierung in der Setup-Funktion.

```
void setup() {
  Serial.begin(9600);
  bleSerial.begin(9600);
  dht.begin();
}
```

Dieser Abschnitt initialisiert die serielle Kommunikation für Debugging-Zwecke sowie das Bluetooth-Modul und den DHT-Sensor.

3. Datenerfassung und Übermittlung via Bluetooth.

```
void loop() {

  delay(2000);
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  // For debug
  // Print the humidity and temperature to the serial monitor
  Serial.print(F("Humidity: "));
  Serial.print(humidity);
  Serial.print(F("%  Temperature: "));
  Serial.print(temperature);
  Serial.println(F("°C "));

  sensorData = String(temperature) + "," + String(humidity);  // Concatenate␣
↪temperature and humidity values
  Serial.print("Data to send: ");
  Serial.println(sensorData);

  bleSerial.println(sensorData);  // Send temperature and humidity values to the␣
↪Bluetooth module
}
```

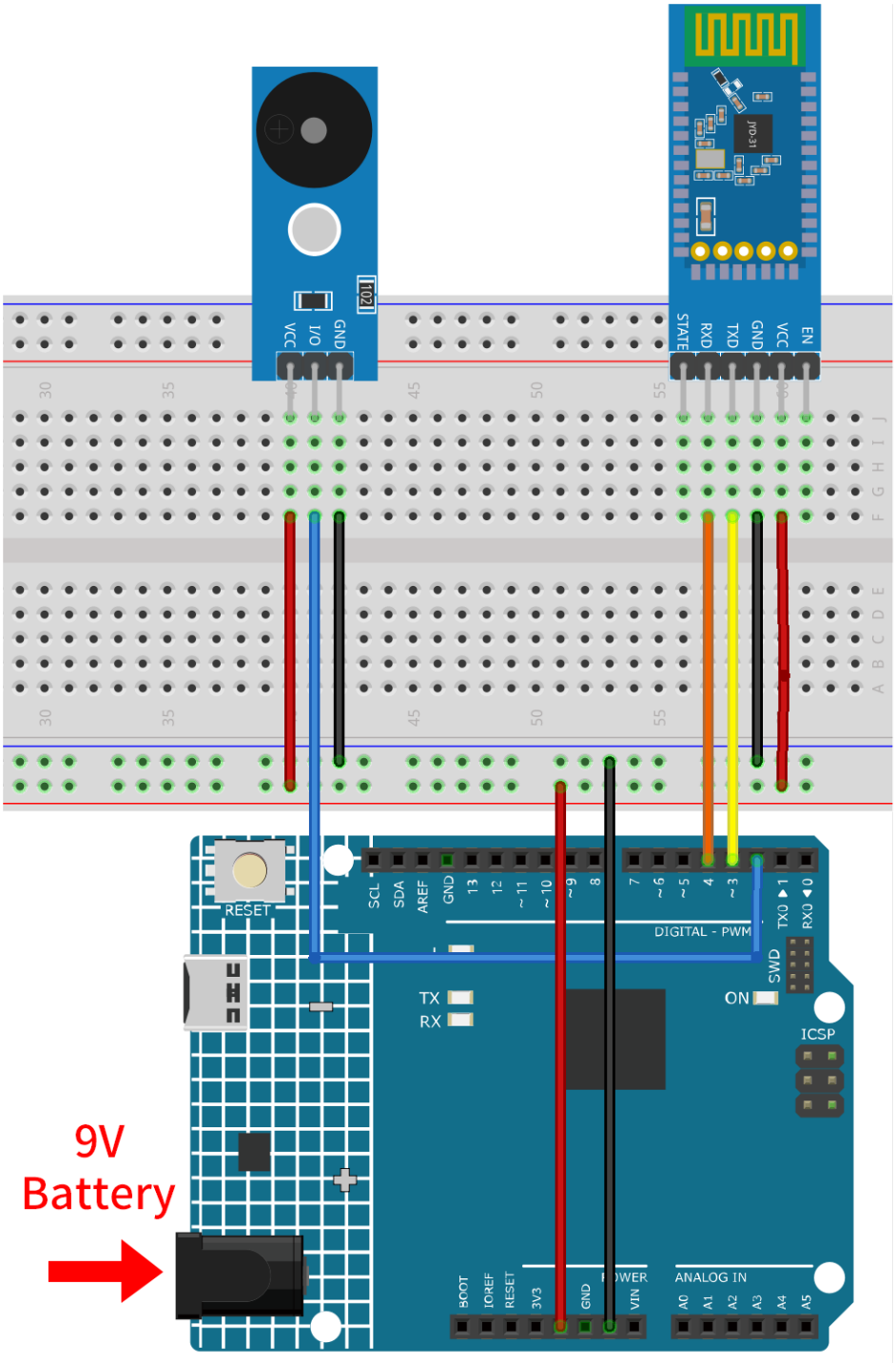
Dieser Abschnitt liest alle 2 Sekunden Temperatur und Luftfeuchtigkeit vom DHT11-Sensor. Bei fehlerhaften Messungen wird eine Fehlermeldung ausgegeben. Ansonsten werden die Messdaten im seriellen Monitor angezeigt und in einem kommagetrennten Format via Bluetooth-Modul gesendet. Die App wertet die im Format „Temperatur,Luftfeuchtigkeit“ empfangenen Daten aus und stellt sie in der Benutzeroberfläche dar.

2.5.15 Bluetooth-Klavier

Dieses Projekt nutzt eine mit dem MIT App Inventor erstellte Android-App, um eine einfache „Klavier“-Funktion mittels eines JDY-31 Bluetooth-Moduls und eines passiven Summer-Moduls zu ermöglichen. Das Bluetooth-Klavier-Projekt erlaubt den Nutzern, verschiedene Musiknoten auf einem passiven Summer-Modul mithilfe eines JDY-31 Bluetooth-Moduls zu spielen. Durch das Senden spezifischer Notenanweisungen via Bluetooth an den Arduino können Nutzer die entsprechenden Töne auf dem Summer erzeugen.

Die Android-Anwendung wird mithilfe einer kostenlosen, webbasierten Plattform namens erstellt. Das Projekt bietet eine hervorragende Möglichkeit, Erfahrung im Umgang mit der Schnittstelle zwischen einem Arduino und einem Smartphone zu sammeln.

1. Schaltung aufbauen



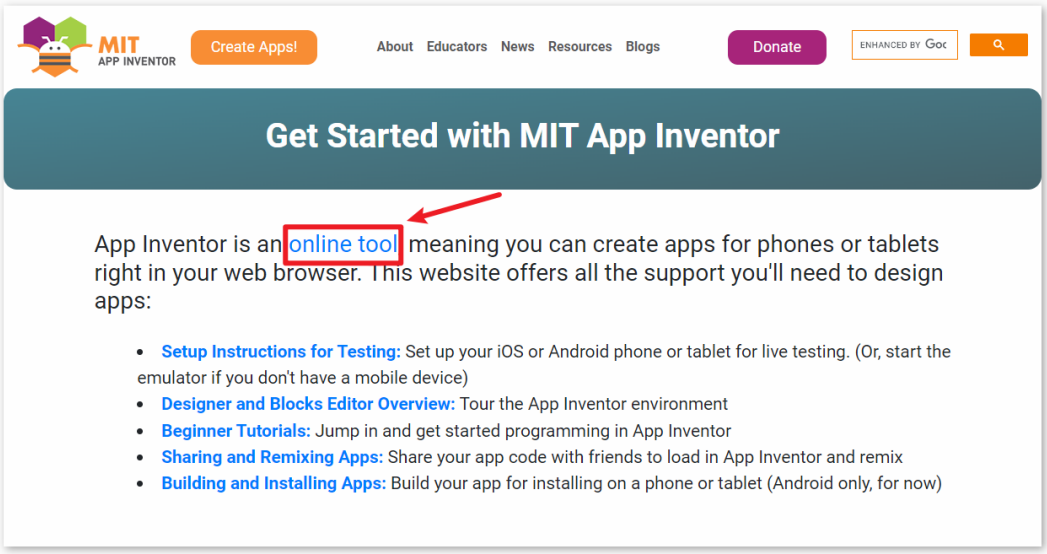
- *Arduino UNO R4 Minima-Platine*
- *JDY-31 Bluetooth-Modul*
- *Passives Summermodul*

2. Android-App erstellen

Die Android-Anwendung wird mit einer kostenlosen Webanwendung namens entwickelt. Der MIT App Inventor dient als ausgezeichneter Einstieg in die Android-Entwicklung, dank seiner intuitiven Drag-and-Drop-Funktionen zur Erstellung einfacher Anwendungen.

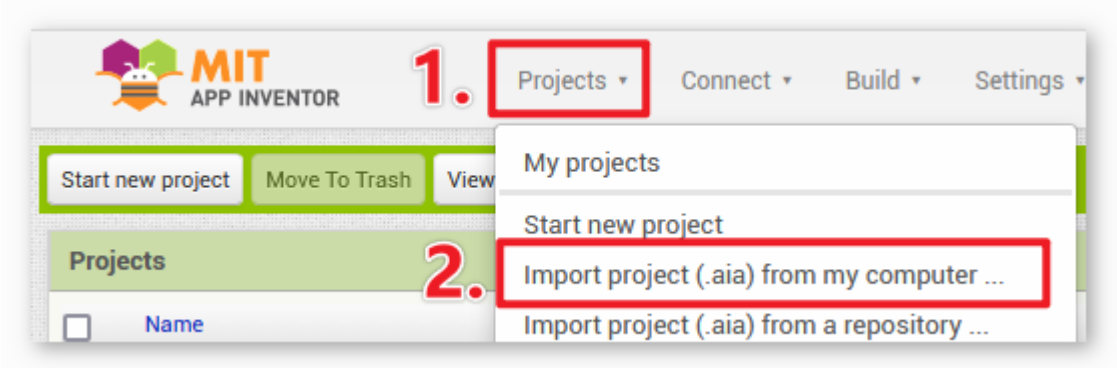
Jetzt geht's los.

1. Rufen Sie auf und klicken Sie auf „Online-Tool“, um sich anzumelden. Sie benötigen ein Google-Konto, um sich beim MIT App Inventor zu registrieren.

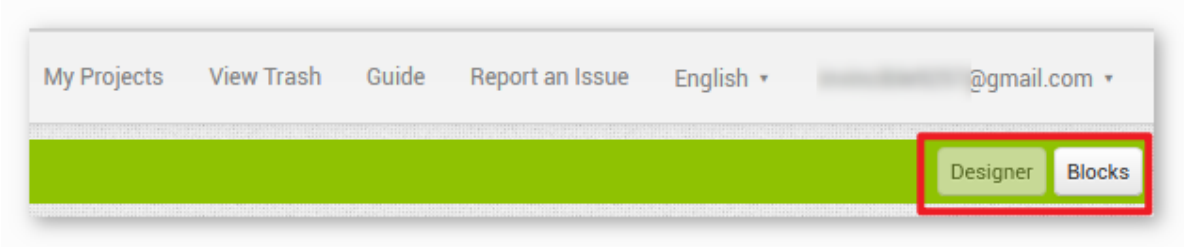


2. Nach dem Login navigieren Sie zu **Projects** -> **Import project (.aia) from my computer**. Laden Sie anschließend die Datei `piano.aia` hoch, die sich im Pfad `ultimate-sensor-kit\iot_project\bluetooth\06-Bluetooth_piano` befindet.

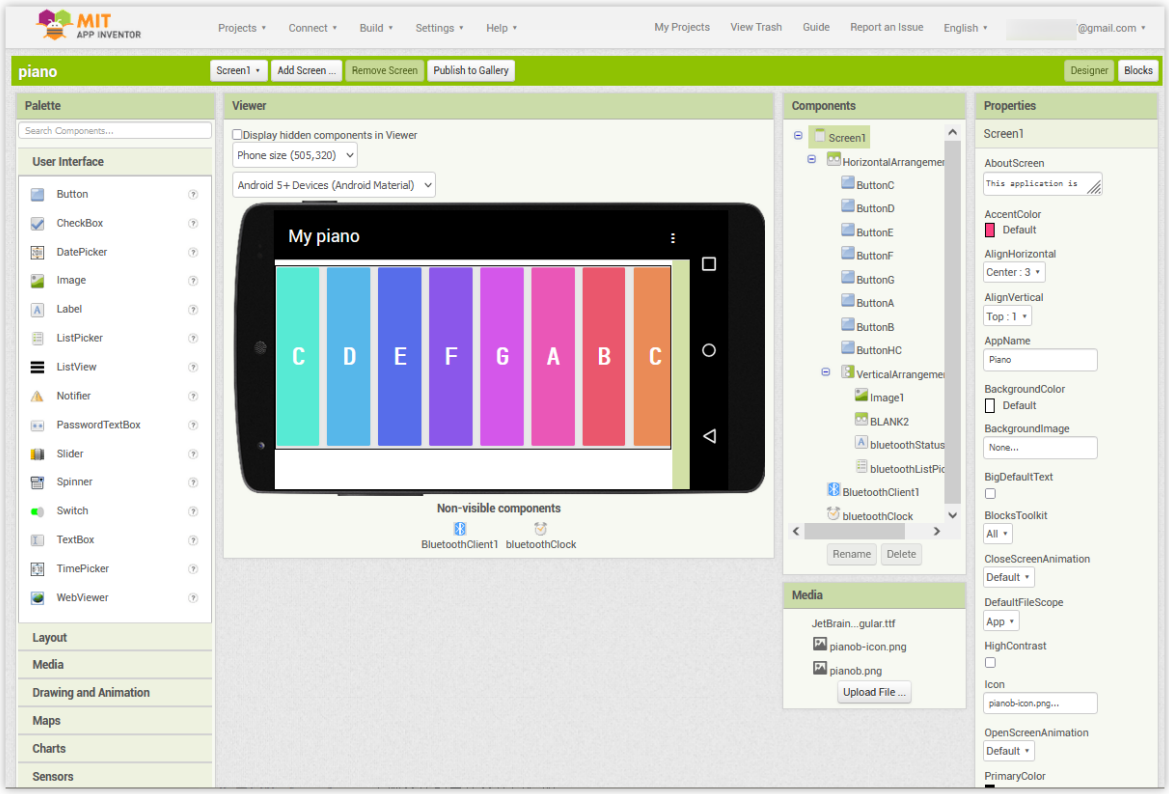
   Direkter Download hier möglich: `piano.aia`



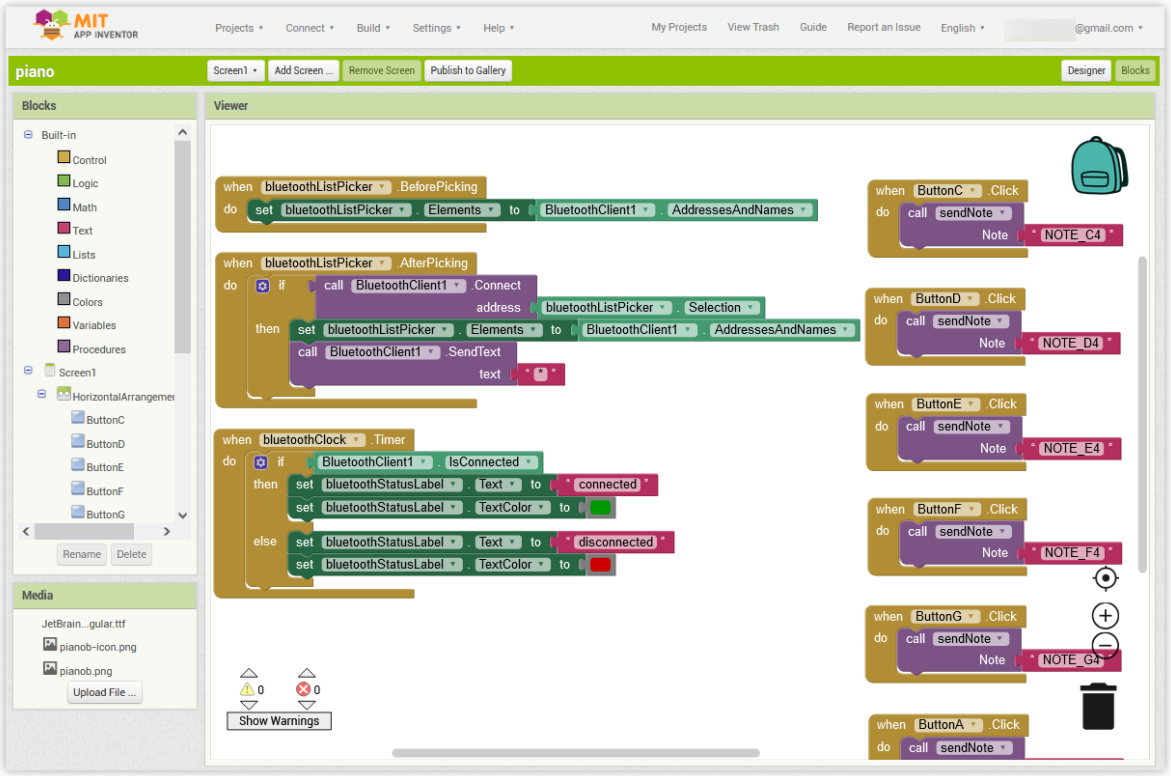
3. Nach dem Hochladen der `.aia`-Datei erscheint die Anwendung in der Software des MIT App Inventors. Dies ist eine vorkonfigurierte Vorlage, die Sie nach dem Kennenlernen des MIT App Inventors modifizieren können.
4. Im MIT App Inventor gibt es zwei Hauptbereiche: den **Designer** und die **Blocks**. Sie können in der oberen rechten Ecke der Seite zwischen diesen beiden Bereichen wechseln.



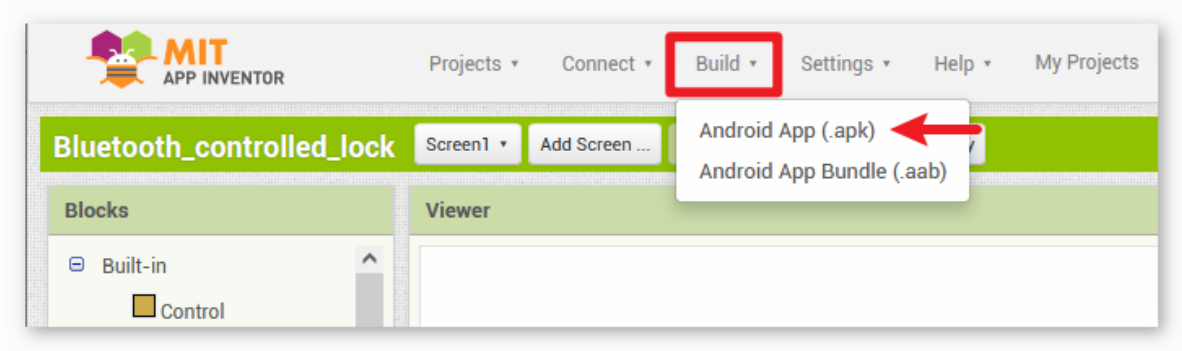
5. Der **Designer** ermöglicht das Hinzufügen von Buttons, Texten, Bildschirmen und die Anpassung des Gesamtdesigns Ihrer Anwendung.



6. Als Nächstes gibt es den Bereich **Blocks**. In diesem Abschnitt können Sie individuelle Funktionen für Ihre App programmieren und so jedes Element in der GUI der App nach Ihren Wünschen gestalten.



7. Um die Anwendung auf einem Smartphone zu installieren, navigieren Sie zum **Build**-Tab.



- Sie können eine `.apk`-Datei generieren. Nach der Auswahl dieser Option erscheint eine Seite, auf der Sie wählen können, ob Sie eine `.apk`-Datei herunterladen oder einen QR-Code zum Installieren scannen möchten. Folgen Sie dem Installationsleitfaden, um die Installation der Anwendung abzuschließen.

  Unsere vorab kompilierte APK können Sie auch hier herunterladen: `piano.apk`

- Falls Sie diese App im Google Play Store oder einem anderen App-Marktplatz veröffentlichen möchten, können Sie eine `.aab`-Datei generieren.

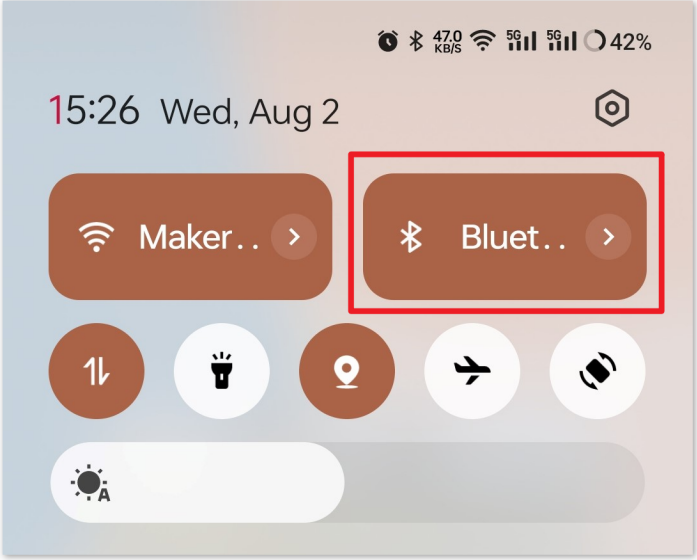
3. Den Code hochladen

1. Öffnen Sie die Datei `06-Bluetooth_piano.ino` unter dem Pfad `ultimate-sensor-kit\iot_project\bluetooth\06-Bluetooth_piano` oder kopieren Sie diesen Code in die **Arduino IDE**.
2. Nach der Auswahl des richtigen Boards und Ports, klicken Sie auf die **Hochladen**-Schaltfläche.
3. Öffnen Sie den seriellen Monitor (Baudrate auf **9600** einstellen), um Debug-Nachrichten anzuzeigen.

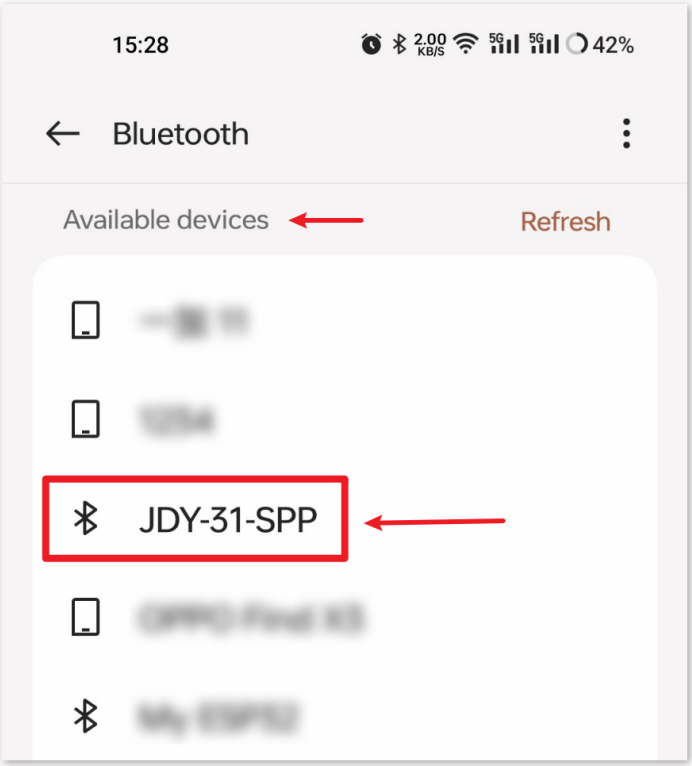
4. App und Bluetooth-Modul verbinden

Stellen Sie sicher, dass die zuvor erstellte Anwendung auf Ihrem Smartphone installiert ist.

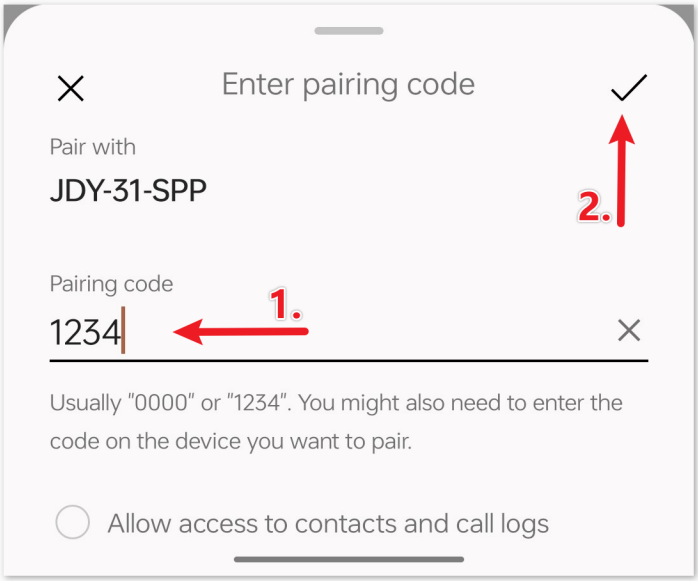
1. Aktivieren Sie zunächst **Bluetooth** auf Ihrem Smartphone.



2. Navigieren Sie zu den **Bluetooth-Einstellungen** auf Ihrem Smartphone und suchen Sie nach Namen wie **JDY-31-SPP**.

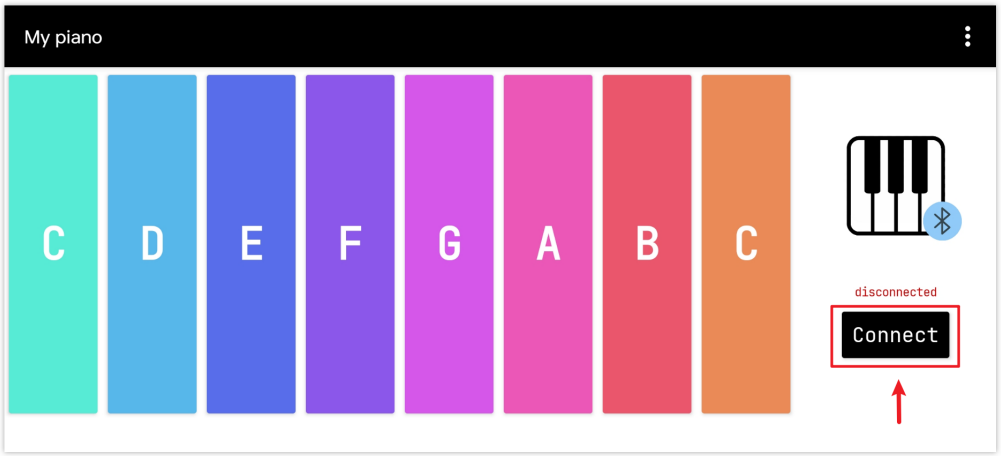


3. Klicken Sie darauf und bestätigen Sie die **Kopplungsanfrage** im aufpoppenden Fenster. Falls ein Kopplungscode erforderlich ist, geben Sie „1234“ ein.

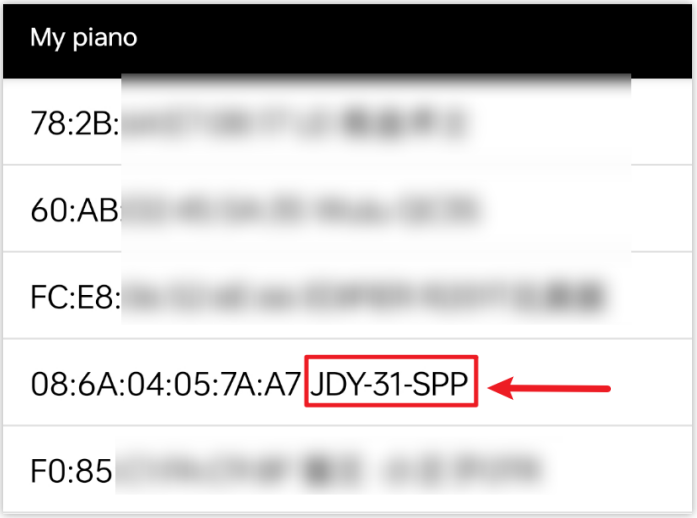


4. Öffnen Sie nun die neu installierte **Piano**-App.

5. In der App klicken Sie auf die **Verbinden**-Schaltfläche, um eine Verbindung zwischen der App und dem Bluetooth-Modul herzustellen.

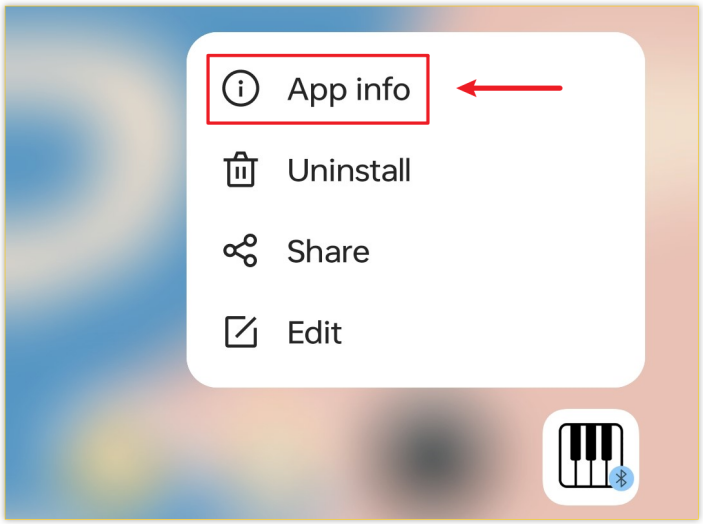


6. Diese Seite zeigt eine Liste aller gekoppelten Bluetooth-Geräte an. Wählen Sie die Option `xx.xx.xx.xx.xx.xx JDY-31-SPP` aus der Liste aus. Der Name jedes Geräts ist neben seiner MAC-Adresse aufgeführt.

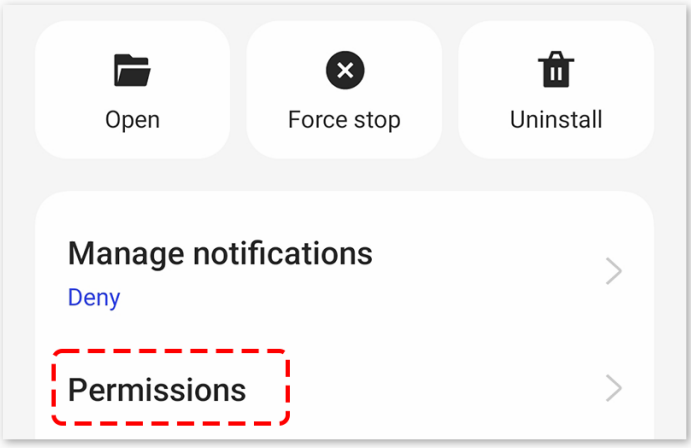


7. Falls Sie auf der oben gezeigten Seite keine Geräte sehen, könnte dies daran liegen, dass der App die Berechtigung zur Suche nach nahegelegenen Geräten fehlt. In diesem Fall müssen Sie die Einstellungen manuell anpassen.

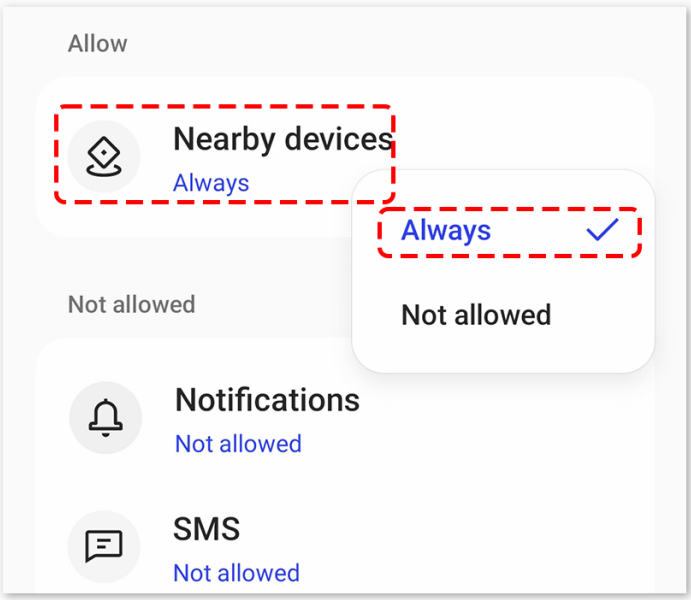
    - Um zur **App-Info**-Seite zu gelangen, halten Sie das App-Symbol gedrückt und wählen Sie es aus. Alternativ können Sie auch eine andere Methode verwenden, um diese Seite zu erreichen.



- Navigieren Sie zur **Berechtigungen**-Seite.

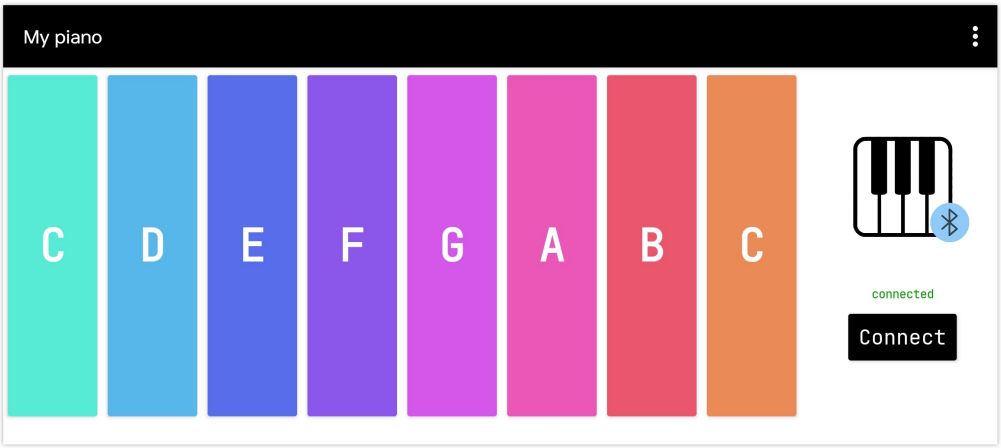


- Um der App das Scannen nach nahegelegenen Geräten zu ermöglichen, gehen Sie zu **Nahegelegene Geräte** und wählen **Immer** aus.



- Starten Sie nun die App neu und wiederholen Sie die Schritte 5 und 6, um erfolgreich eine Bluetooth-Verbindung herzustellen.

8. Nach einer erfolgreichen Verbindung können Sie in der App auf die Tasten klicken, um verschiedene Noten zu spielen, und sogar einige einfache Lieder ausführen.



5. Code-Erklärung

1. **Bibliotheken und Pins einrichten**

```
#include "pitches.h"
#include <SoftwareSerial.h>
const int bluetoothTx = 3;
const int bluetoothRx = 4;
SoftwareSerial bleSerial(bluetoothTx, bluetoothRx);
const int buzzerPin = 2;
```

- `pitches.h`: Diese Datei enthält die Frequenzwerte für musikalische Noten.

2. **Variablendeklaration zur Speicherung von Bluetooth-Daten**

```
char character;
String noteType;
```

- `character`: Speichert einzelne Zeichen, die über Bluetooth empfangen werden.
- `noteType`: Fasst die Zeichen zusammen, um die vollständige Notenanweisung zu bilden.

3. **Setup-Funktion - Initialisierung der seriellen Kommunikation**

```
void setup() {
  Serial.begin(9600);
  bleSerial.begin(9600);
}
```

- Initialisiert die serielle Kommunikation mit einer Baudrate von 9600.
- Die Standard-`Serial` dient zur Fehlersuche, während `bleSerial` speziell für die Bluetooth-Kommunikation vorgesehen ist.

4. **Hauptschleife - Lesen von Bluetooth-Daten und Abspielen entsprechender Noten**

```
void loop() {
  while (bleSerial.available() > 0) {
    character = bleSerial.read();
    noteType = noteType + character;
    if (character == '*') {
      noteType = noteType.substring(0, noteType.length() - 1);
      Serial.println(noteType);
      if (noteType == "NOTE_C4") {
        tone(buzzerPin, NOTE_C4);
      } // ... weitere Noten werden ähnlich überprüft ...
      noteType = "";
      delay(200);
      noTone(buzzerPin);
    }
  }
}
```

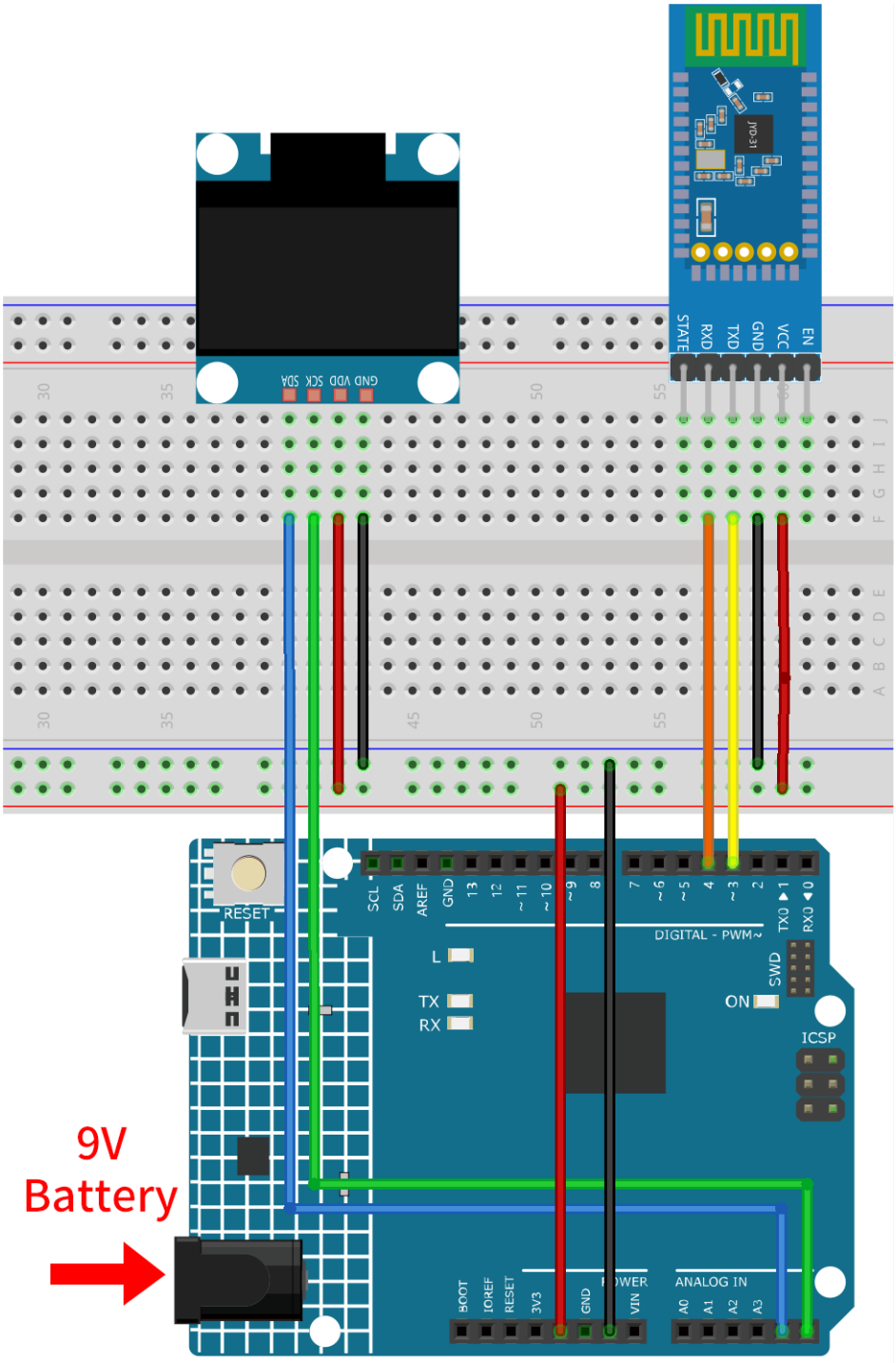
- Liest Zeichen von Bluetooth und bildet den `noteType`.
- Wenn ein Sternchen (‚*‘) erkannt wird, signalisiert dies das Ende der Notenanweisung. Die Note wird dann abgespielt, kurz verzögert und dann gestoppt.

2.5.16 Bluetooth OLED

Dieses Projekt nutzt eine mit dem MIT App Inventor erstellte Android-App, um Nachrichten via Bluetooth an ein Arduino-Gerät zu senden. Der Arduino zeigt die empfangenen Nachrichten auf einem OLED-Bildschirm an. Die Android-App punktet mit einer benutzerfreundlichen Oberfläche und ermöglicht es, Nachrichten einzugeben und per Knopfdruck zu senden.

Die Android-Anwendung wird mit einer kostenlosen, webbasierten Plattform namens erstellt. Das Projekt bietet eine hervorragende Gelegenheit, die Kommunikation zwischen einem Arduino und einem Smartphone näher kennenzulernen.

1. Schaltung aufbauen



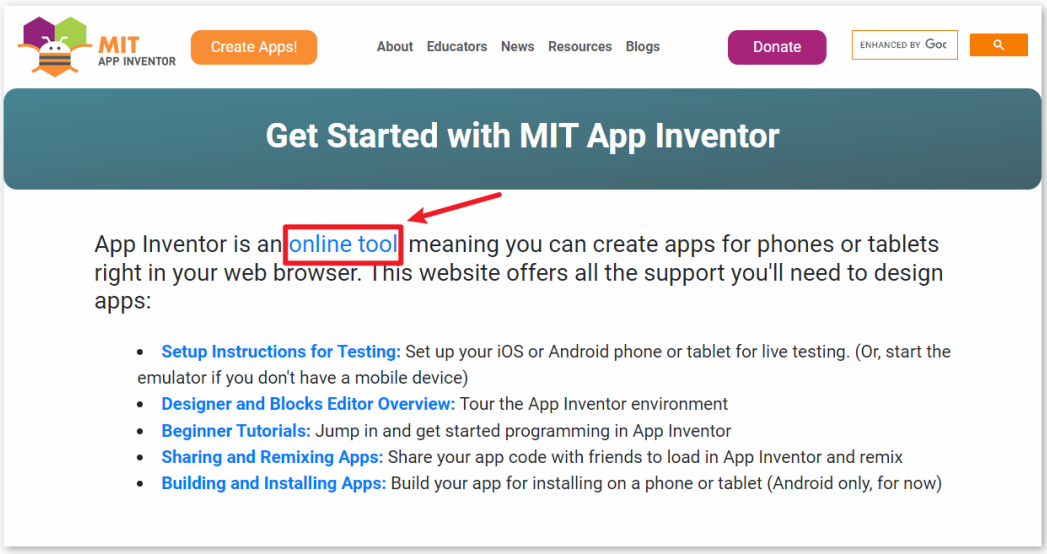
- *Arduino UNO R4 Minima-Platine*
- *JDY-31 Bluetooth-Modul*
- *OLED-Display-Modul*

2. Android-App erstellen

Für die Entwicklung der Android-Anwendung wird eine kostenlose Webanwendung namens verwendet. MIT App Inventor bietet dank seiner intuitiven Drag-and-Drop-Funktionen einen ausgezeichneten Einstieg in die Android-Entwicklung.

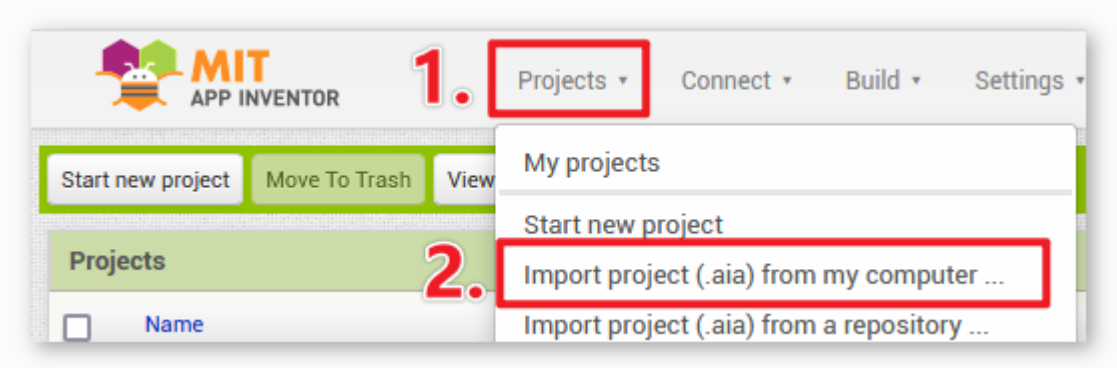
Jetzt kann es losgehen.

1. Besuchen Sie und klicken Sie auf „Online-Tool“ zum Anmelden. Sie benötigen ein Google-Konto zur Registrierung bei MIT App Inventor.

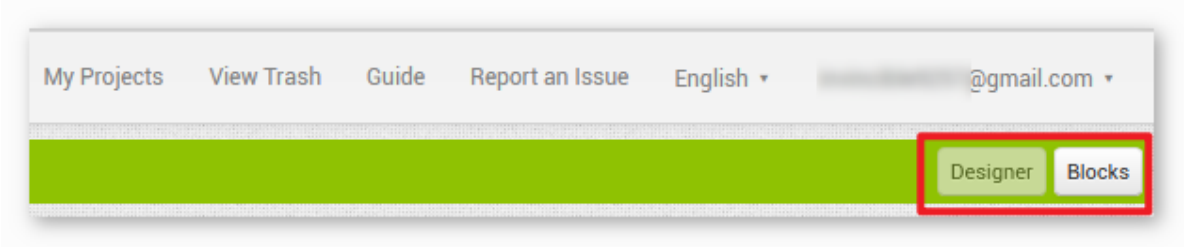


2. Nach dem Login navigieren Sie zu **Projects** -> **Import project (.aia) from my computer**. Laden Sie anschließend die `oled.aia`-Datei hoch, die sich im Pfad `ultimate-sensor-kit\iot_project\bluetooth\07-Bluetooth_oled` befindet.

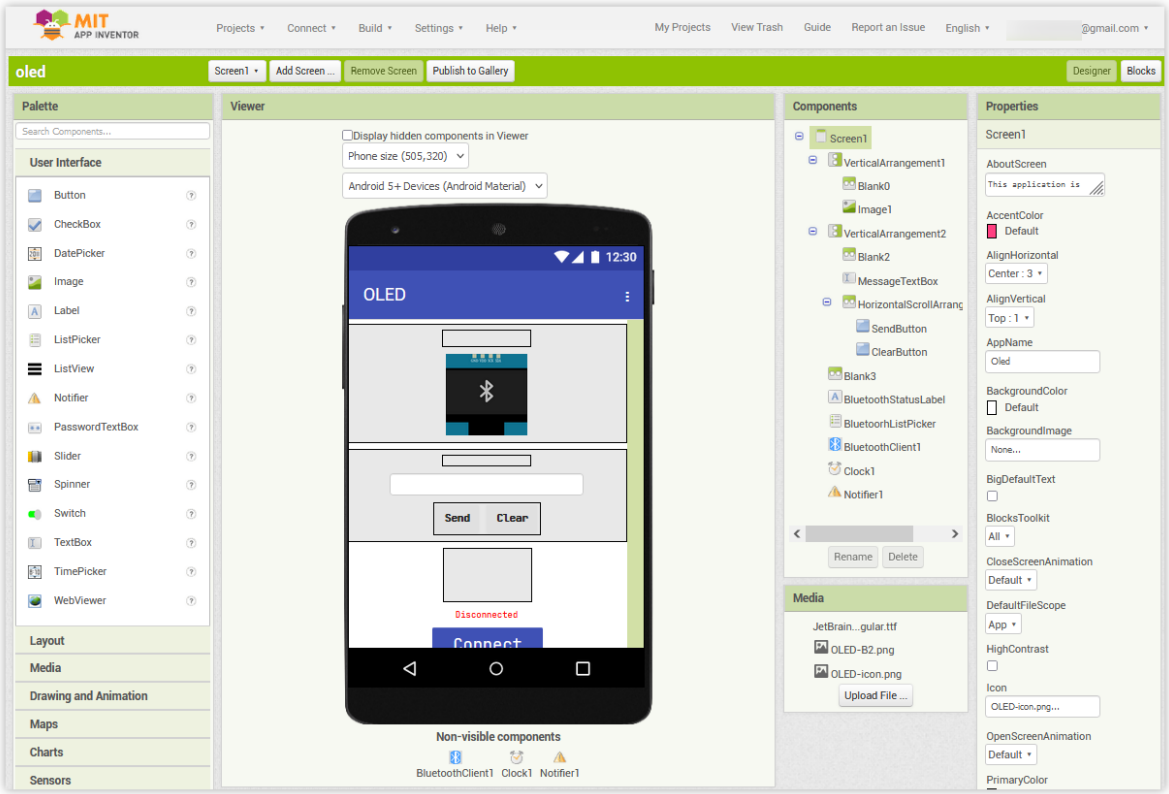
   Direkter Download hier möglich: `oled.aia`



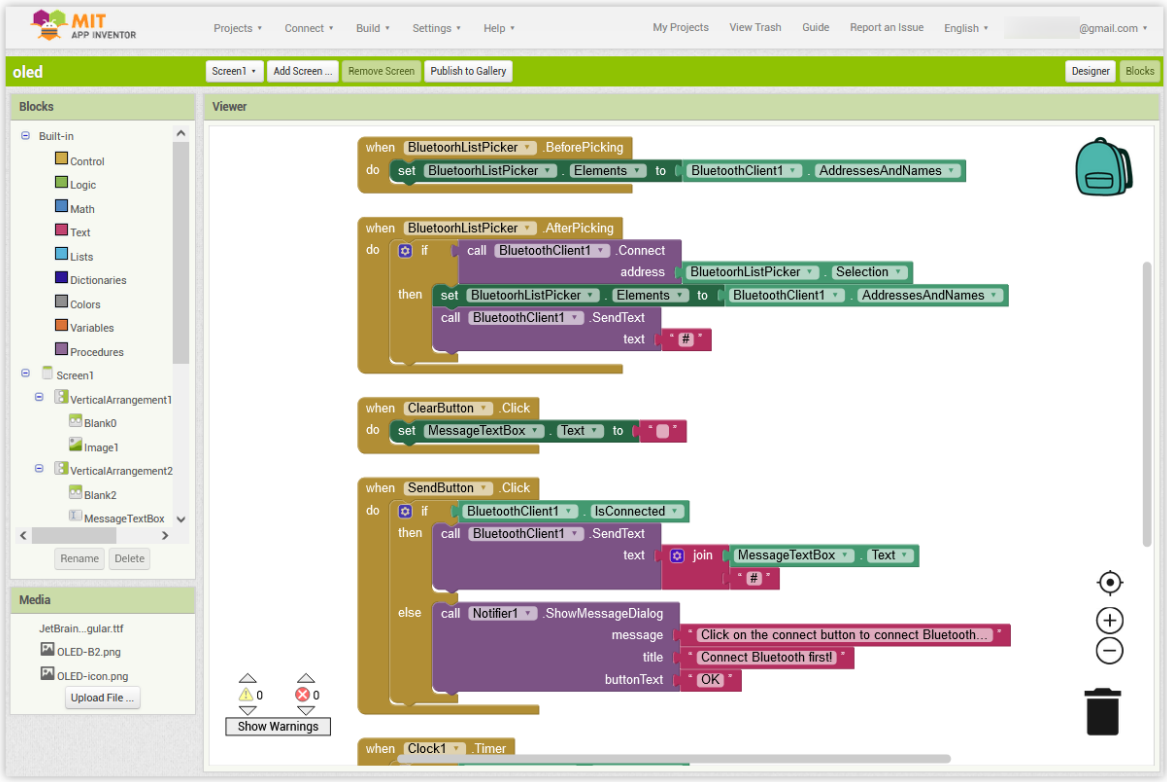
3. Nach dem Hochladen der `.aia`-Datei erscheint die Anwendung in der MIT App Inventor-Software. Dies ist eine vorkonfigurierte Vorlage. Sie können diese Vorlage modifizieren, nachdem Sie sich mit den folgenden Schritten vertraut gemacht haben.
4. In MIT App Inventor gibt es zwei Hauptbereiche: den **Designer** und die **Blocks**. Sie können oben rechts auf der Seite zwischen diesen beiden Bereichen wechseln.



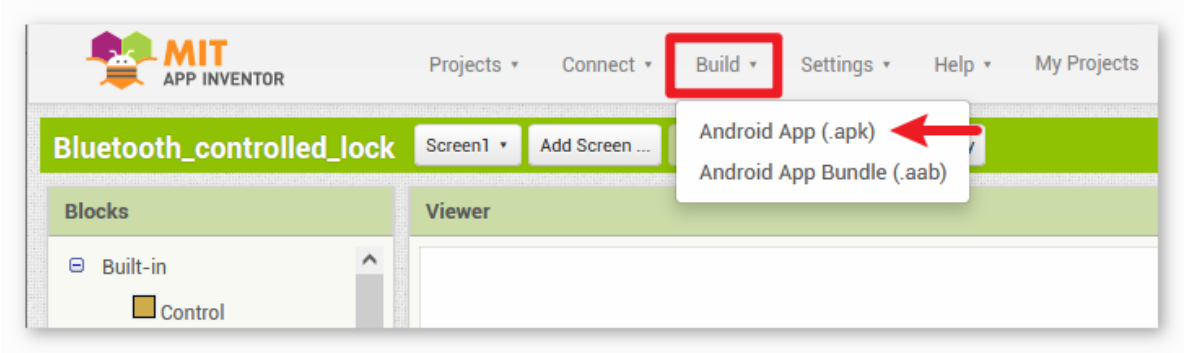
5. Der **Designer** ermöglicht es Ihnen, Schaltflächen, Text, Bildschirme hinzuzufügen und das allgemeine Erscheinungsbild Ihrer Anwendung anzupassen.



6. Dann gibt es den Bereich **Blocks**. Hier können Sie individuelle Funktionen für Ihre App programmieren und so jede Komponente in der App-Oberfläche nach Ihren Wünschen gestalten.



7. Um die Anwendung auf einem Smartphone zu installieren, navigieren Sie zur Registerkarte **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 für die Installation wählen können. Befolgen Sie die Installationsanleitung, um die Installation der Anwendung abzuschließen.

  Sie können auch unsere vorkompilierte APK hier herunterladen: `piano.apk`

- Wenn Sie die App im Google Play Store oder einem anderen App-Marktplatz veröffentlichen möchten, können Sie eine `.aab`-Datei generieren.

3. Code hochladen

1. Öffnen Sie die Datei `07-Bluetooth_oled.ino` im Pfad `ultimate-sensor-kit\iot_project\bluetooth\07-Bluetooth_oled`, oder kopieren Sie diesen Code in die **Arduino IDE**.

   Bemerkung: Um die Bibliothek zu installieren, verwenden Sie den Arduino-Bibliotheksmanager und suchen nach **„Adafruit SSD1306“** und **„Adafruit GFX“** und installieren diese.

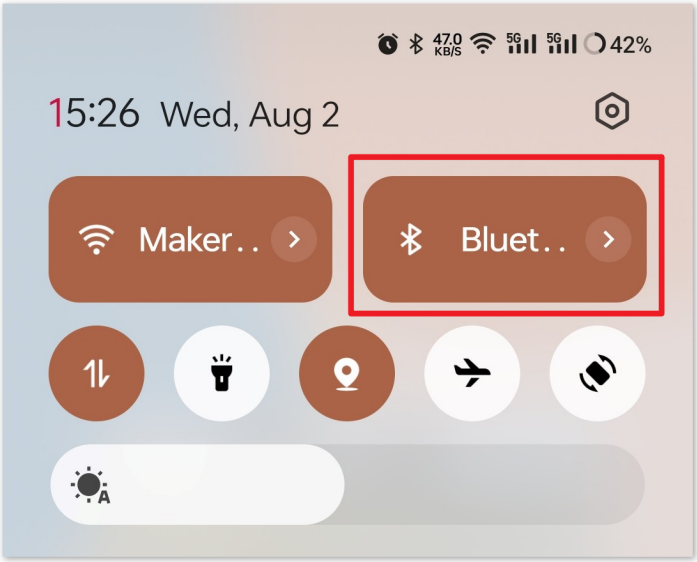
   ---

2. Nach der Auswahl des richtigen Boards und Ports klicken Sie auf die Schaltfläche **Hochladen**.
3. Öffnen Sie den Seriellen Monitor (Baudrate auf **9600** einstellen), um Debug-Nachrichten anzuzeigen.

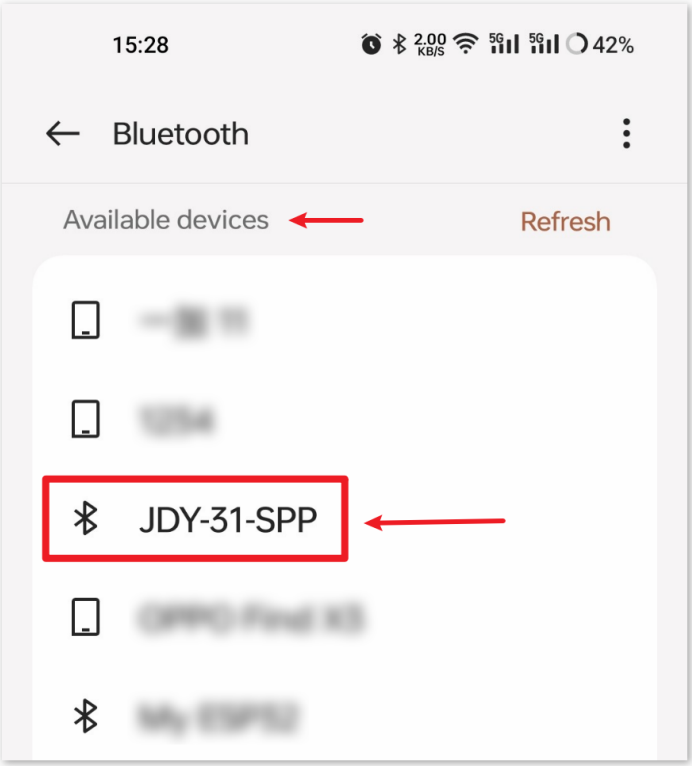
4. Verbindung der App mit dem Bluetooth-Modul

Vergewissern Sie sich, dass die zuvor erstellte App auf Ihrem Smartphone installiert ist.

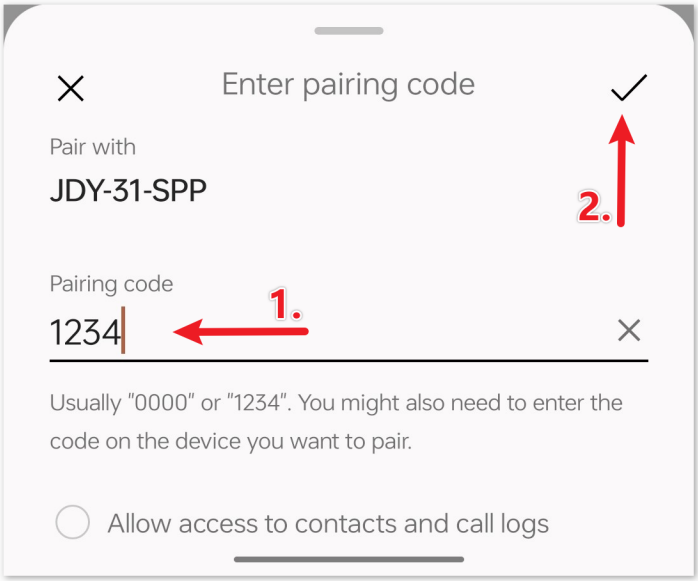
1. Aktivieren Sie zu Beginn die **Bluetooth**-Funktion Ihres Smartphones.



2. Navigieren Sie zu den **Bluetooth-Einstellungen** auf Ihrem Smartphone und suchen Sie nach Namen wie **JDY-31-SPP**.

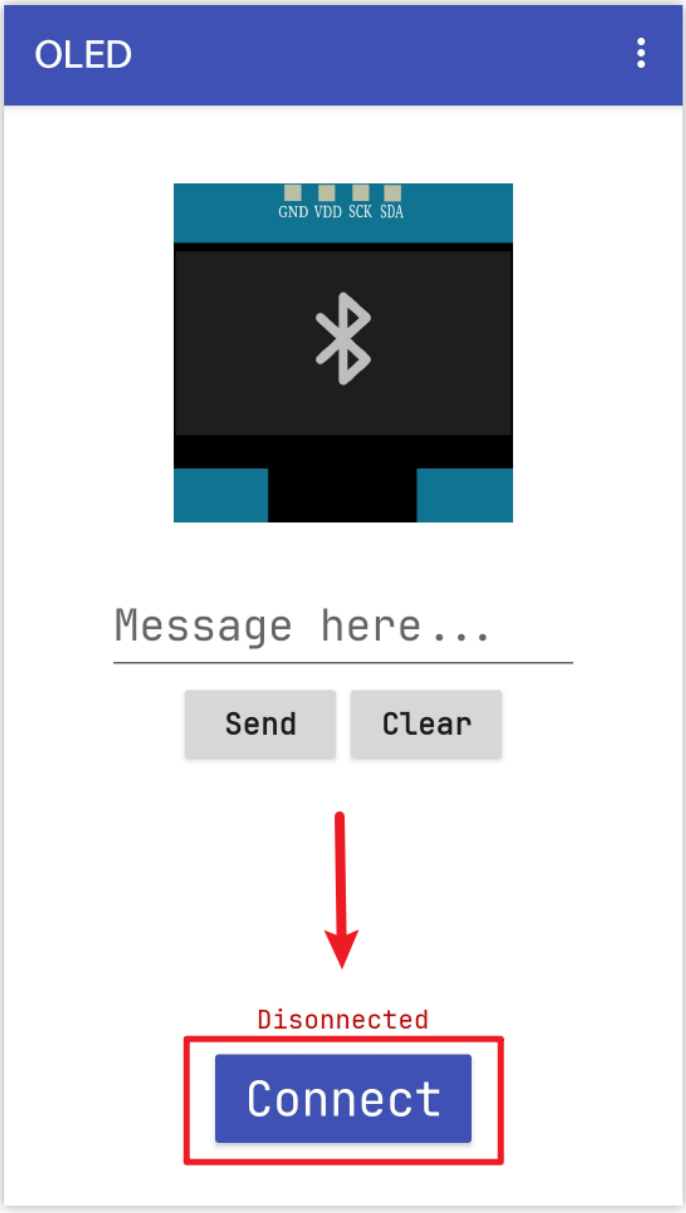


3. Nachdem Sie darauf geklickt haben, stimmen Sie der **Pairing-Anfrage** im Popup-Fenster zu. Sollte ein Pairing-Code erforderlich sein, geben Sie „1234“ ein.



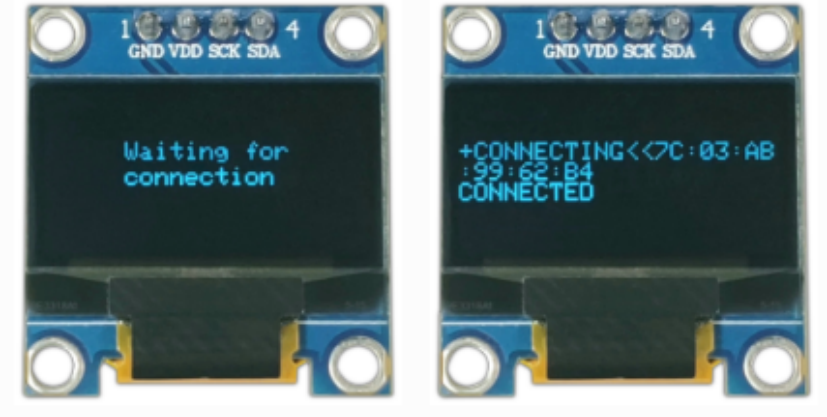
4. Öffnen Sie nun die frisch installierte **OLED**-App.

5. In der App tippen Sie auf die **Connect**-Schaltfläche, um eine Verbindung zwischen der App und dem Bluetooth-Modul herzustellen.



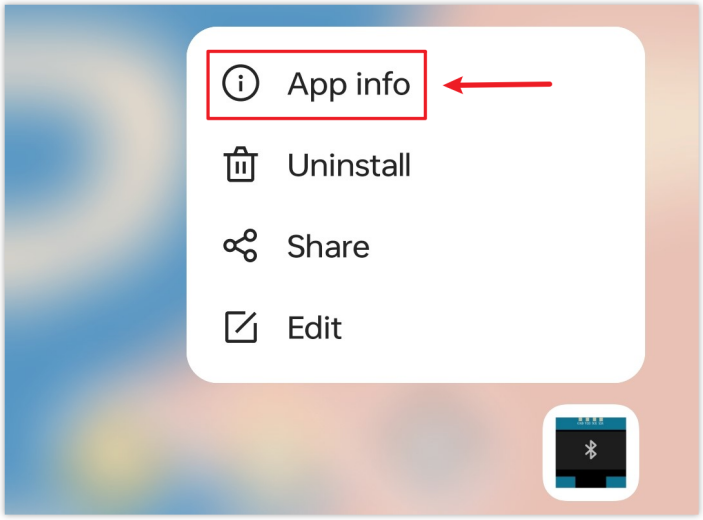
6. Diese Seite listet alle bereits gekoppelten Bluetooth-Geräte auf. Wählen Sie die Option `xx.xx.xx.xx.xx.xx JDY-31-SPP` aus der Liste aus. Der Name jedes Geräts ist neben seiner MAC-Adresse aufgeführt.

Nach erfolgreicher Verbindung wechselt das OLED-Display von „Waiting for connection“ zur Anzeige der MAC-Adresse des verbundenen Geräts.

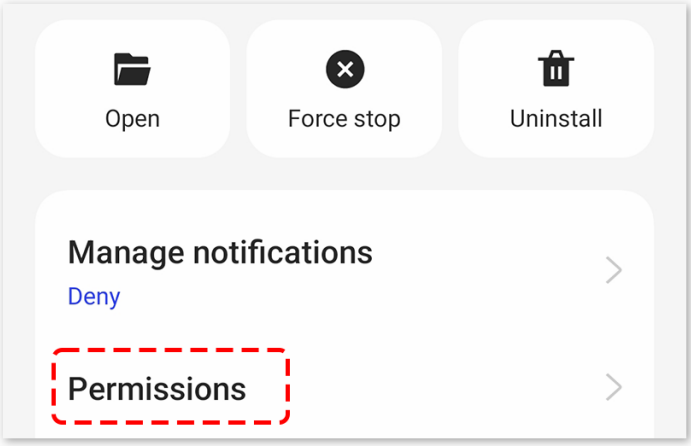


7. Sollten Sie auf der angezeigten Seite keine Geräte sehen, könnte es daran liegen, dass die App nicht berechtigt ist, in der Nähe befindliche Geräte zu scannen. In einem solchen Fall müssen Sie die Einstellungen manuell anpassen.

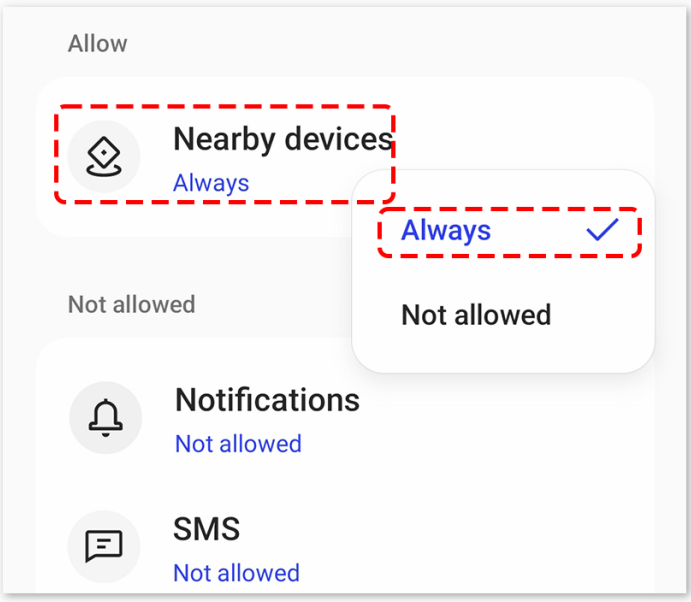
    - Um die **App-Info**-Seite zu öffnen, halten Sie das App-Symbol lange gedrückt und wählen Sie es aus. Falls Sie einen alternativen Weg kennen, diese Seite zu erreichen, nutzen Sie diesen.



- Navigieren Sie zur **Berechtigungen**-Seite.

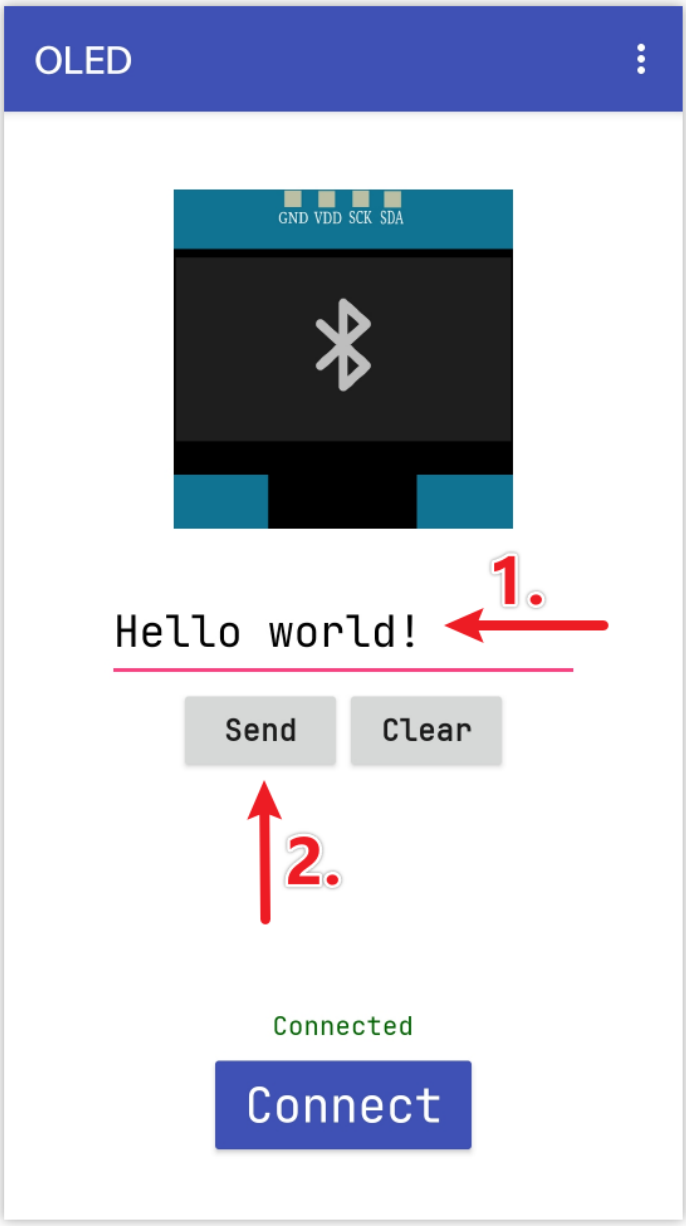


- Um der App das Scannen von Geräten in der Nähe zu ermöglichen, gehen Sie zu **Nahegelegene Geräte** und wählen Sie **Immer**.



- Starten Sie anschließend die App neu und wiederholen Sie die Schritte 5 und 6, um eine erfolgreiche Bluetooth-Verbindung herzustellen.

8. Nach erfolgreicher Verbindung werden Sie zur Hauptseite weitergeleitet. Tragen Sie Ihre gewünschte Nachricht in das vorgesehene Textfeld ein und betätigen Sie die Sendetaste, um sie auf dem OLED-Display darzustellen.



5. Code-Erklärung

1. **Einrichtung der Bluetooth-Kommunikation**:

   In diesem Abschnitt wird die `SoftwareSerial`-Bibliothek eingebunden und die digitalen Pins für die Bluetooth-Kommunikation konfiguriert. Die Standard-`Serial`-Schnittstelle dient zur Fehlersuche, während `bleSerial` speziell für die Bluetooth-Kommunikation vorgesehen ist.

```
#include <SoftwareSerial.h>
const int bluetoothTx = 3;
const int bluetoothRx = 4;
SoftwareSerial bleSerial(bluetoothTx, bluetoothRx);
```

2. **Einrichtung des OLED-Displays**:

Hier werden die benötigten Bibliotheken und Konstanten zur Initialisierung und Verwaltung des OLED-Displays deklariert.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

3. **Initialisierung**:

   In der Funktion `setup()` werden die seriellen Kommunikationswege initialisiert. Das OLED-Display wird gestartet, und die Anfangsmeldung „Waiting for connection“ wird angezeigt.

```
void setup() {
   Serial.begin(9600);
   bleSerial.begin(9600);
   if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
      Serial.println(F("SSD1306 allocation failed"));
      for (;;)
         ;
   }
   display.clearDisplay();
   display.setTextColor(WHITE);
   display.setTextSize(1);
   display.setCursor(32, 20);
   display.println("Waiting for");
   display.setCursor(32, 30);
   display.println("connection");
   display.display();
}
```

4. **Hauptschleife**:

   Innerhalb der `loop()`-Funktion prüft der Code ständig auf eingehende Daten vom Bluetooth-Modul. Sobald eine vollständige Nachricht (die mit einem ‚#‘ endet, fügt die App automatisch ein ‚#‘ am Ende der vom Benutzer gesendeten Nachricht hinzu) empfangen wird, wird diese auf dem OLED angezeigt. Zudem wird die empfangene Nachricht zur Fehlersuche im seriellen Monitor ausgegeben.

```
void loop() {
   while (bleSerial.available() > 0) {
      character = bleSerial.read();
      message = message + character;
      if (character == '#') {
         message = message.substring(0, message.length() - 1);
         Serial.println();
         Serial.print("DEBUG:");
         Serial.println(message);
         display.clearDisplay();
         display.setTextColor(WHITE);
         display.setTextSize(1);
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
            display.setCursor(0, 20);
            display.println(message);
            display.display();
            message = "";
            delay(200);
        }
    }
}
```

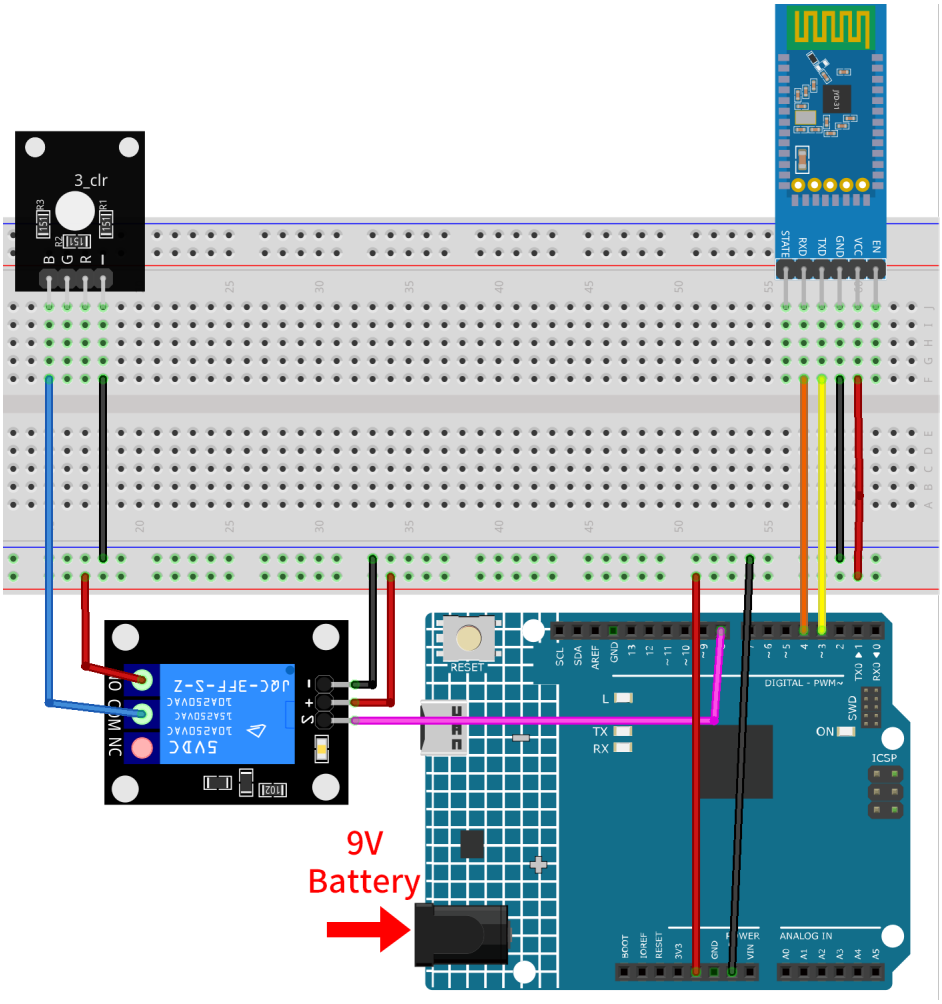
2.5.17 Bluetooth-Fernrelais

Dieses Projekt nutzt eine mit MIT App Inventor erstellte Android-App, um ein Relaismodul über das JDY-31 Bluetooth-Modul, welches an einen Arduino Uno angeschlossen ist, fernzusteuern. Wenn in der App ein Button gedrückt wird, sendet sie einen einfachen Befehl (‚1' oder ‚0') an den Arduino. Erhält der Arduino den Befehl ‚1' via Bluetooth, wird das Relais aktiviert; bei Erhalt des Befehls ‚0' wird es deaktiviert. Dies bietet eine benutzerfreundliche Schnittstelle auf dem Smartphone zur Steuerung von an das Relais angeschlossenen Geräten.

Die Android-Anwendung wird auf einer kostenlosen webbasierten Plattform namens erstellt. Das Projekt bietet eine hervorragende Gelegenheit, sich mit der Schnittstelle zwischen einem Arduino und einem Smartphone vertraut zu machen.

1. Schaltung aufbauen

Warnung: Das folgende Beispiel zeigt, wie ein Relais zur Steuerung eines LED-Moduls verwendet wird. **Obwohl es möglich ist, das Relais in realen Anwendungen mit anderen Geräten zu verbinden, ist bei der Arbeit mit HOHER Wechselspannung äußerste Vorsicht geboten. Unachgemäße oder fehlerhafte Verwendung kann zu schweren Verletzungen oder gar zum Tod führen. Dieses Projekt richtet sich daher an Personen, die sich mit HOHER Wechselspannung auskennen und diese sicher handhaben können. Sicherheit hat immer Vorrang.**



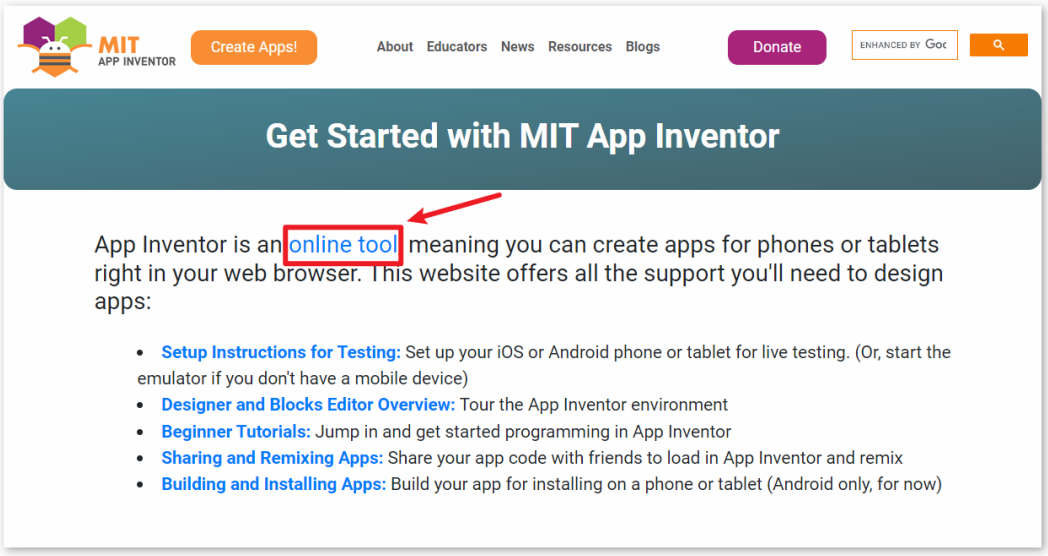
- *Arduino UNO R4 Minima-Platine*
- *JDY-31 Bluetooth-Modul*
- *5V-Relaismodul*
- *RGB-Modul*

2. Android-App erstellen

Die Android-Anwendung wird mit einer kostenlosen Webanwendung namens entwickelt. MIT App Inventor eignet sich hervorragend als Einstieg in die Android-Entwicklung, dank seiner intuitiven Drag-and-Drop-Funktionen zur Erstellung einfacher Applikationen.

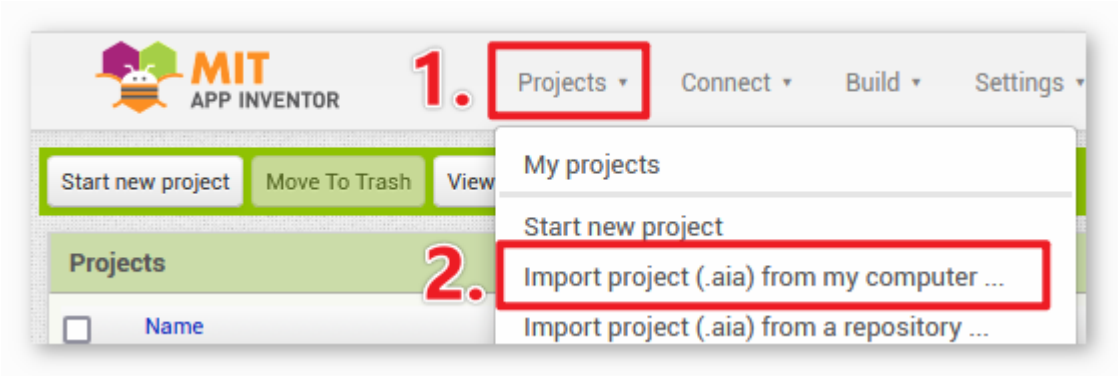
Nun legen wir los.

1. Gehen Sie zu und klicken Sie auf „Online-Tool“ zum Einloggen. Ein Google-Konto ist zur Registrierung bei MIT App Inventor erforderlich.

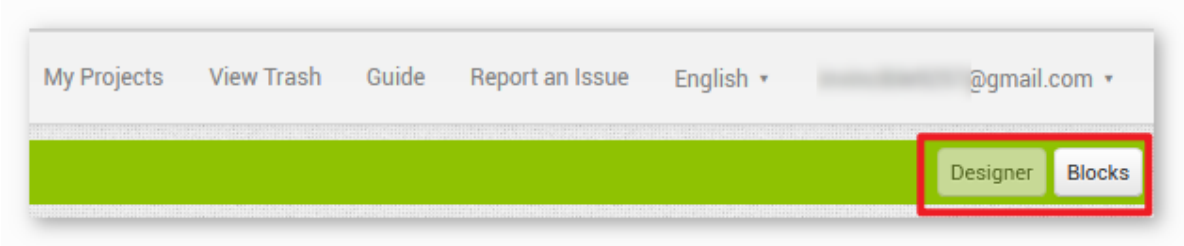


2. Nach dem Einloggen navigieren Sie zu **Projects** -> **Import project (.aia) from my computer**. Laden Sie anschließend die Datei `RemoteRelay.aia` hoch, die im Pfad `ultimate-sensor-kit\iot_project\bluetooth\08-Bluetooth_remote_relay` zu finden ist.

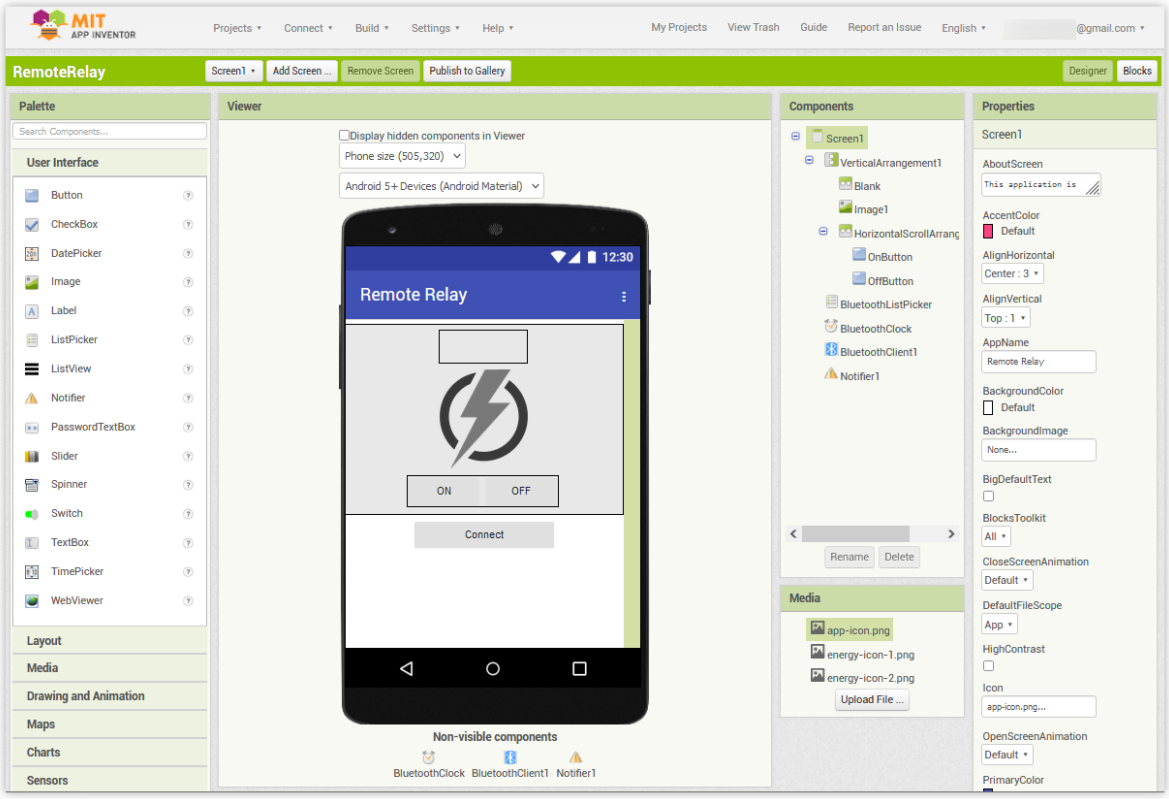
   Direkter Download hier: `RemoteRelay.aia`



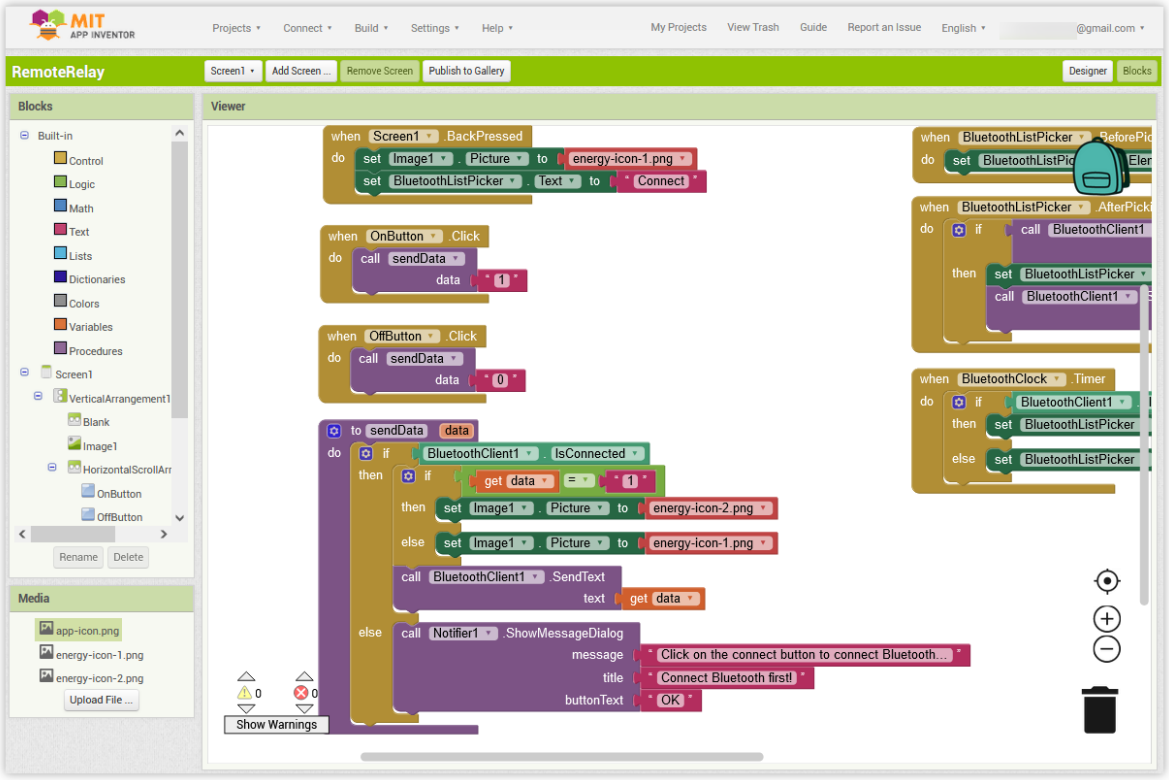
3. Nach dem Hochladen der `.aia`-Datei sehen Sie die Anwendung in der MIT App Inventor-Software. Dies ist eine vorkonfigurierte Vorlage. Nachdem Sie sich mit MIT App Inventor vertraut gemacht haben, können Sie diese Vorlage entsprechend modifizieren.
4. In MIT App Inventor gibt es zwei Hauptbereiche: den **Designer** und die **Blocks**. Zwischen diesen beiden Bereichen können Sie oben rechts auf der Seite umschalten.



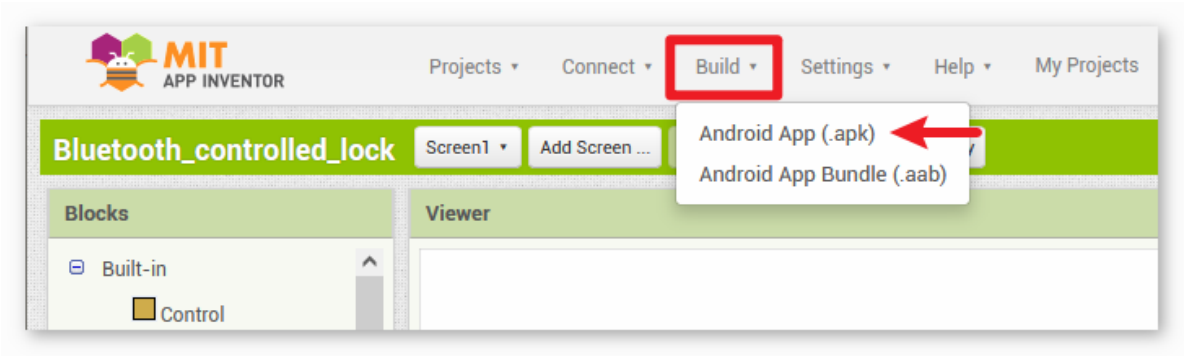
5. Der **Designer** ermöglicht das Hinzufügen von Buttons, Texten, Bildschirmen und die allgemeine ästhetische Gestaltung Ihrer Anwendung.



6. Als Nächstes gibt es den Bereich **Blocks**. Hier können Sie spezielle Funktionalitäten für Ihre App programmieren und jedes Element in der Benutzeroberfläche der App entsprechend konfigurieren.



7. Um die Anwendung auf einem Smartphone zu installieren, navigieren Sie zur Registerkarte **Build**.



- Hier können Sie eine `.apk`-Datei generieren. Nach der Auswahl dieser Option wird eine Seite angezeigt, auf der Sie zwischen dem Herunterladen einer `.apk`-Datei oder dem Scannen eines QR-Codes zur Installation wählen können. Folgen Sie der Installationsanleitung, um die Installation der Anwendung abzuschließen.

  Die vorab kompilierte APK können Sie hier herunterladen: `RemoteRelay.apk`

- Wenn Sie diese App im Google Play Store oder einem anderen App-Marktplatz hochladen möchten, können Sie eine `.aab`-Datei generieren.

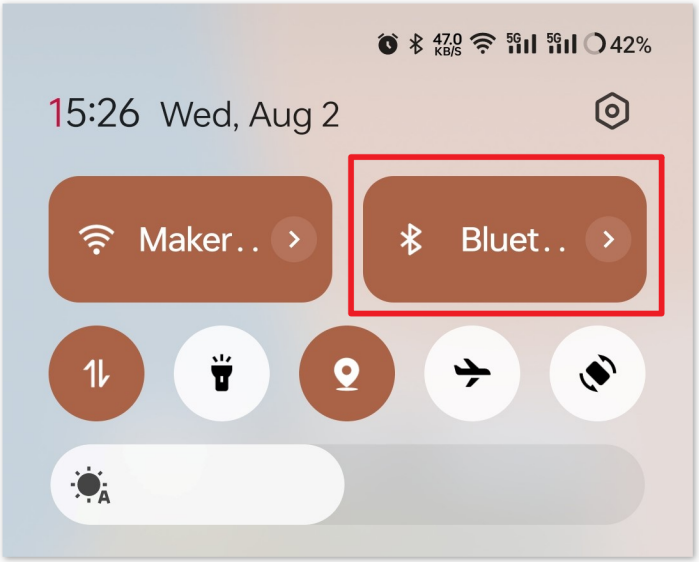
3. Den Code hochladen

1. Öffnen Sie die Datei `08-Bluetooth_remote_relay.ino` im Pfad `ultimate-sensor-kit\iot_project\bluetooth\08-Bluetooth_remote_relay` oder kopieren Sie den Code in die **Arduino IDE**.
2. Nach der Auswahl des richtigen Boards und Ports klicken Sie auf die Schaltfläche **Hochladen**.
3. Öffnen Sie den seriellen Monitor (Baudrate auf **9600** einstellen), um Debug-Meldungen anzuzeigen.

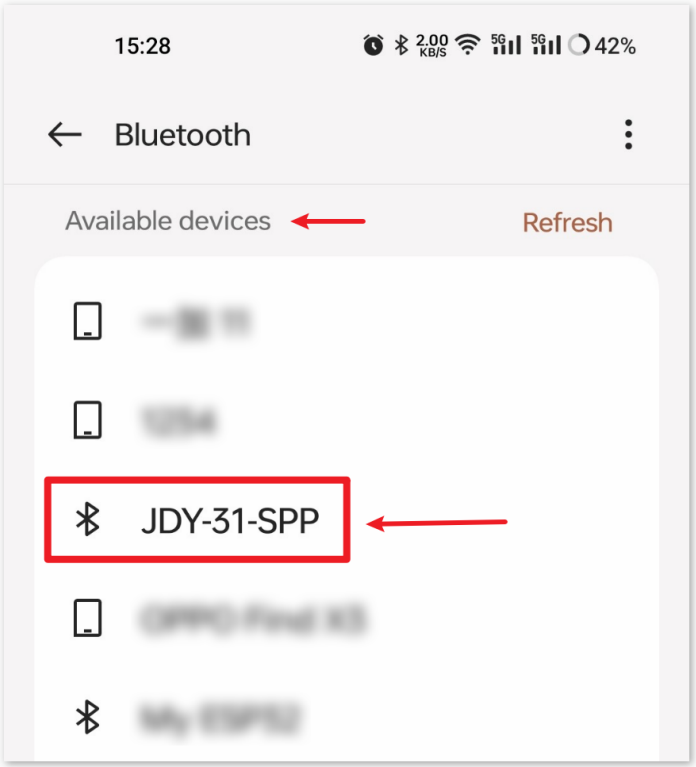
4. Verbindung zwischen App und Bluetooth-Modul herstellen

Vergewissern Sie sich, dass die zuvor erstellte Anwendung auf Ihrem Smartphone installiert ist.

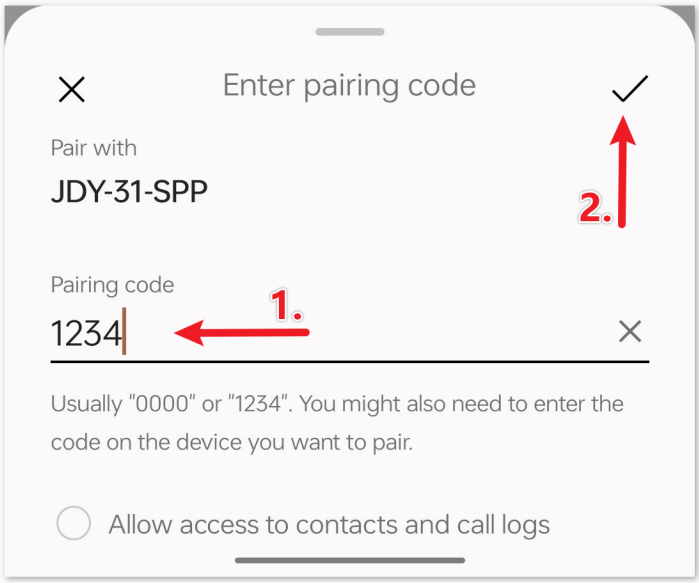
1. Aktivieren Sie zunächst **Bluetooth** auf Ihrem Smartphone.



2. Navigieren Sie zu den **Bluetooth-Einstellungen** auf Ihrem Smartphone und suchen Sie nach Namen wie **JDY-31-SPP**.

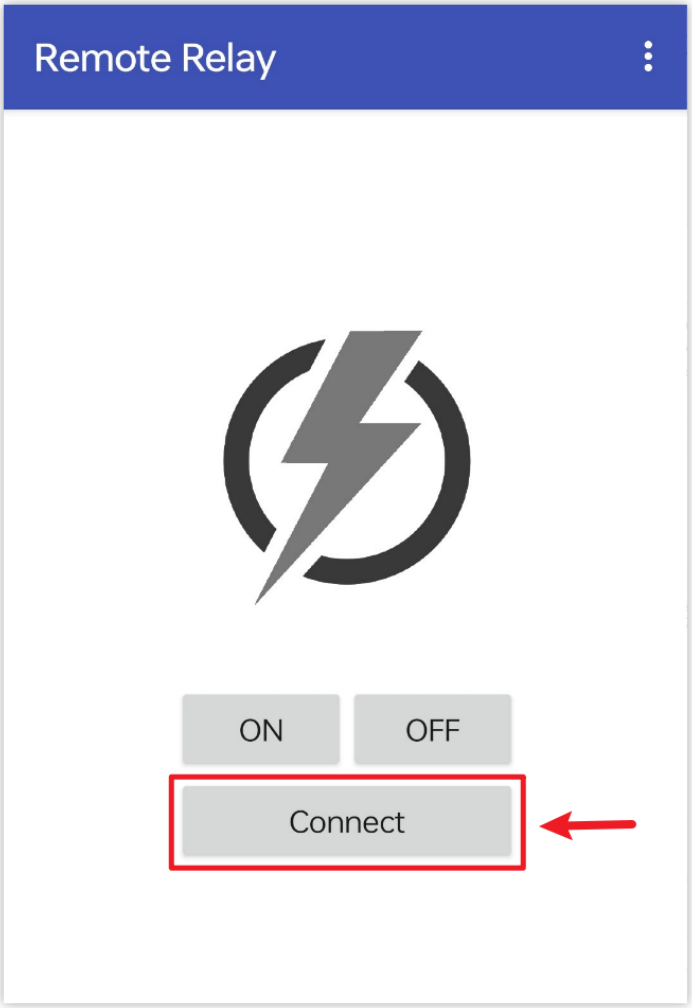


3. Klicken Sie darauf und stimmen Sie der **Kopplungsanfrage** im Popup-Fenster zu. Falls ein Kopplungscode erforderlich ist, geben Sie „1234“ ein.



4. Öffnen Sie nun die neu installierte **Remote Relay**-App.

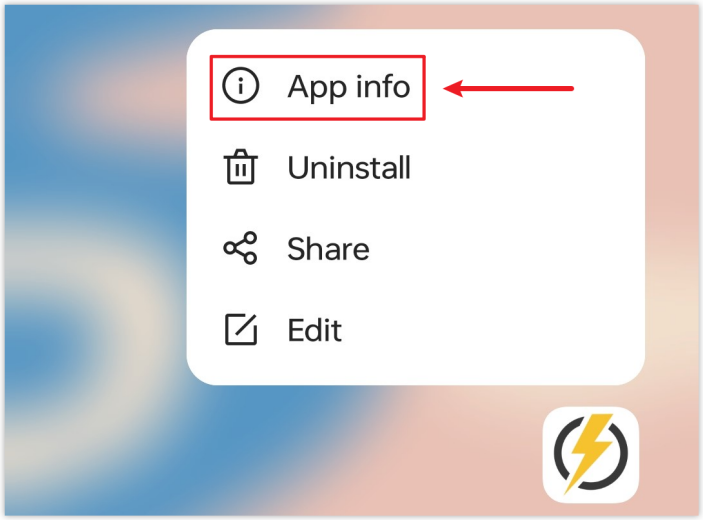
5. In der App klicken Sie auf die Schaltfläche **Connect**, um eine Verbindung zwischen der App und dem Bluetooth-Modul herzustellen.



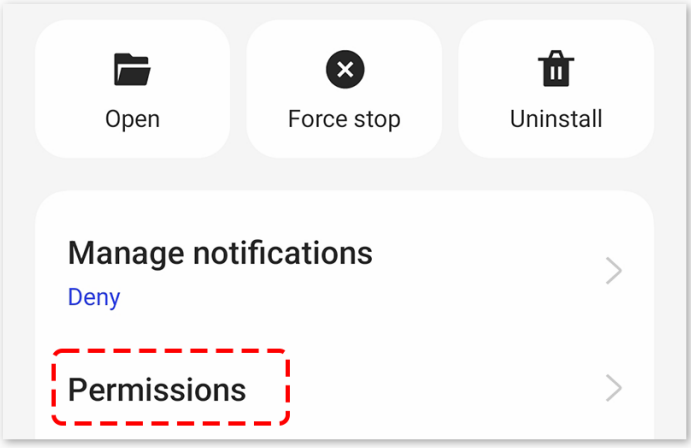
6. Diese Seite zeigt eine Liste aller gekoppelten Bluetooth-Geräte an. Wählen Sie die Option `xx.xx.xx.xx.xx.xx JDY-31-SPP` aus der Liste aus. Der Name jedes Geräts wird neben seiner MAC-Adresse aufgelistet.

7. Falls Sie auf der oben angezeigten Seite keine Geräte sehen, könnte dies daran liegen, dass der App die Berechtigung zum Scannen von Geräten in der Nähe fehlt. In diesem Fall müssen Sie die Einstellungen manuell anpassen.

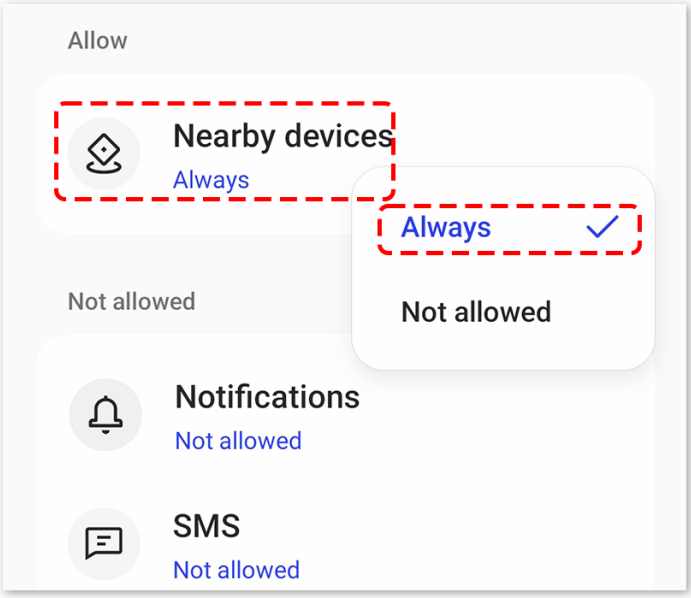
    - Um zur Seite **App-Informationen** zu gelangen, halten Sie das App-Symbol lange gedrückt und wählen Sie es aus. Alternativ können Sie jede andere Methode verwenden, um diese Seite zu erreichen.



    - Navigieren Sie zur Seite **Berechtigungen**.



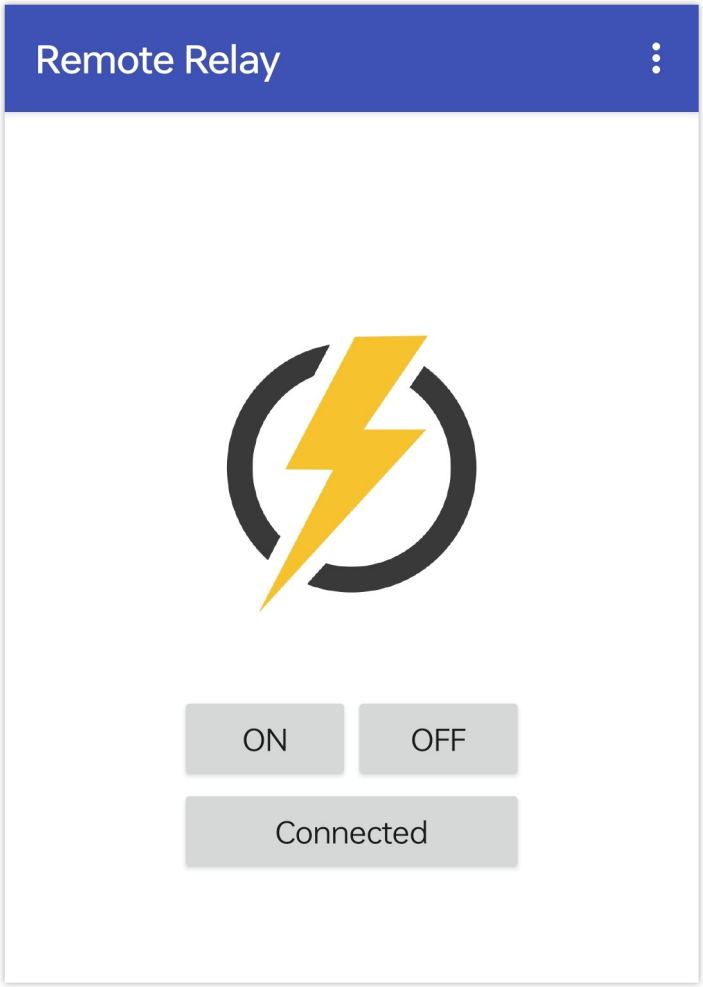
- Um der App das Scannen von Geräten in der Nähe zu ermöglichen, gehen Sie zu **Geräte in der Nähe** und wählen **Immer**.



- 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 zur Hauptseite weitergeleitet. Klicken Sie auf „ON“ oder „OFF“, um das Relais ein- oder auszuschalten.

Bemerkung: Wenn die MAC-Adresse des Bluetooth „1“ enthält, wird das Relais bei der ersten erfolgreichen Bluetooth-Verbindung eingeschaltet und dann schnell wieder ausgeschaltet. Denn beim Herstellen der Bluetooth-Verbindung wird die MAC-Adresse an den Arduino gesendet. Der Arduino erkennt „1“ und schaltet das Relais ein. Nach der Bluetooth-Initialisierung sendet die App eine 0 an den Arduino über Bluetooth, um sicherzustellen, dass der Anfangszustand des Relais nach der Verbindung geschlossen ist.



5. Code-Erläuterung

1. **Bibliothek und globale Variableninitialisierung**

```
#include <SoftwareSerial.h>

const int bluetoothTx = 3;
const int bluetoothRx = 4;
SoftwareSerial bleSerial(bluetoothTx, bluetoothRx);

const int relayPin = 8;
```

Dieses Segment bindet die `SoftwareSerial`-Bibliothek ein und initialisiert die globalen Variablen. Die Pins 3 und 4 sind für die Datenübertragung und den Datenempfang mit dem Bluetooth-Modul definiert. Zudem ist das Relaismodul an Pin 8 angeschlossen.

2. **Funktion setup()**

```
void setup() {
  Serial.begin(9600);
  bleSerial.begin(9600);
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  pinMode(relayPin, OUTPUT);
}
```

Die Funktion startet den seriellen Monitor und die Kommunikation mit dem Bluetooth-Modul mit einer Baudrate von 9600. Sie legt außerdem den `relayPin` als Ausgangspin fest.

3. **Funktion loop()**

```
void loop() {
  if (bleSerial.available() > 0) {
    char message = bleSerial.read();
    // Serial.println(message);  //for debug

    if (message == '1') {
      digitalWrite(relayPin, HIGH);
      Serial.println("On");
    } else if (message == '0') {
      digitalWrite(relayPin, LOW);
      Serial.println("Off");
    }
  }
}
```

Die `loop()`-Funktion wird fortlaufend ausgeführt. Sie prüft, ob eine Nachricht vom Bluetooth-Modul empfangen wurde. Falls ja, liest sie das Zeichen aus. Abhängig vom empfangenen Zeichen (‚1' oder ‚0') wird das Relais aktiviert oder deaktiviert und eine Bestätigungsnachricht („Eingeschaltet" oder „Ausgeschaltet") an den seriellen Monitor gesendet.

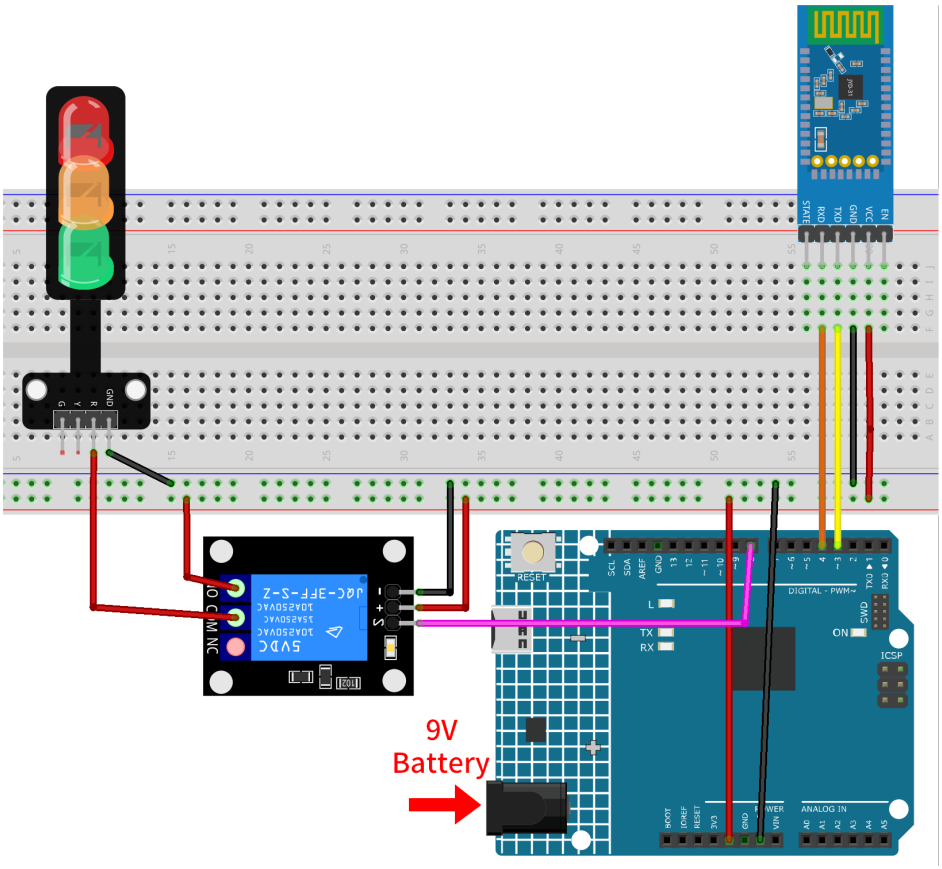
2.5.18 Bluetooth Sprachgesteuertes Relais

Dieses Projekt kombiniert eine Android-App, die mit dem MIT App Inventor entwickelt wurde, mit einem Arduino Uno Board. Die App bietet eine Spracheingabefunktion. Wenn die Spracheingabe des Benutzers das Wort „an" enthält, sendet die App eine „1"-Nachricht über Bluetooth an den Arduino, um das Relais einzuschalten. Enthält die Spracheingabe hingegen das Wort „aus", wird eine „0"-Nachricht gesendet, die den Arduino anweist, das Relais auszuschalten. Nach Erhalt dieser Nachrichten verarbeitet der Arduino sie entsprechend und schaltet das Relais ein oder aus.

Die Android-Anwendung wird mithilfe einer kostenlosen webbasierten Plattform namens erstellt. Dieses Projekt bietet eine hervorragende Möglichkeit, sich mit der Schnittstelle zwischen einem Arduino und einem Smartphone vertraut zu machen.

1. Schaltung aufbauen

Warnung: Im folgenden Beispiel wird gezeigt, wie ein Relais zur Steuerung eines Verkehrslichtmoduls verwendet wird. **Sie können das Relais in realen Anwendungen auch mit anderen Geräten verbinden. Dabei ist jedoch äußerste Vorsicht im Umgang mit HOHER Wechselspannung geboten. Unsachgemäße oder fehlerhafte Verwendung kann zu schweren Verletzungen oder gar zum Tod führen. Das Projekt ist daher nur für Personen geeignet, die sich mit HOHER Wechselspannung auskennen. Sicherheit hat immer Vorrang.**



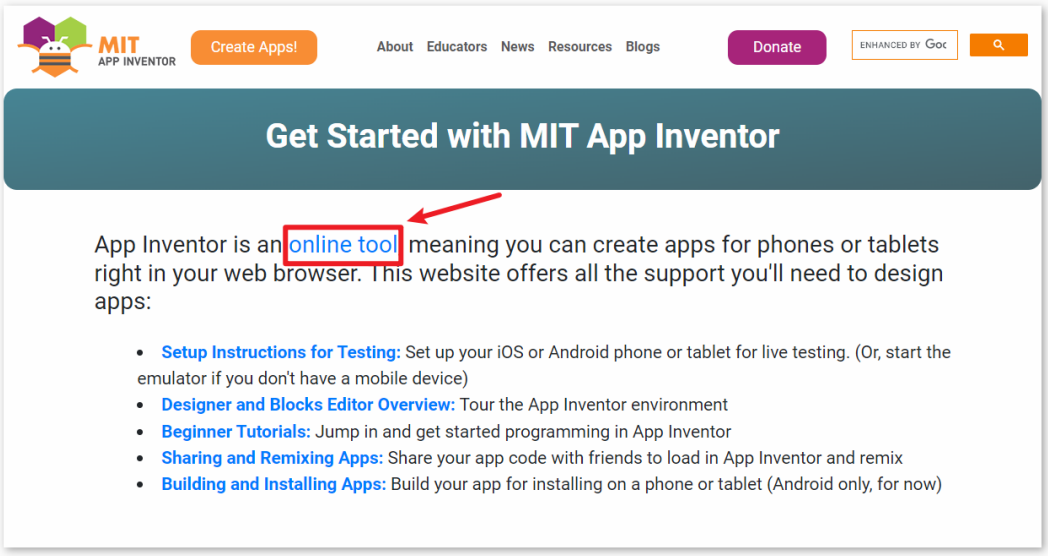
- *Arduino UNO R4 Minima-Platine*
- *JDY-31 Bluetooth-Modul*
- *5V-Relaismodul*
- *Ampel-Modul*

2. Android-App erstellen

Die Android-Anwendung wird mit einer kostenlosen Webanwendung namens entwickelt. Der MIT App Inventor dient als hervorragender Einstieg in die Android-Entwicklung dank seiner intuitiven Drag-and-Drop-Funktionen, die die Erstellung einfacher Anwendungen ermöglichen.

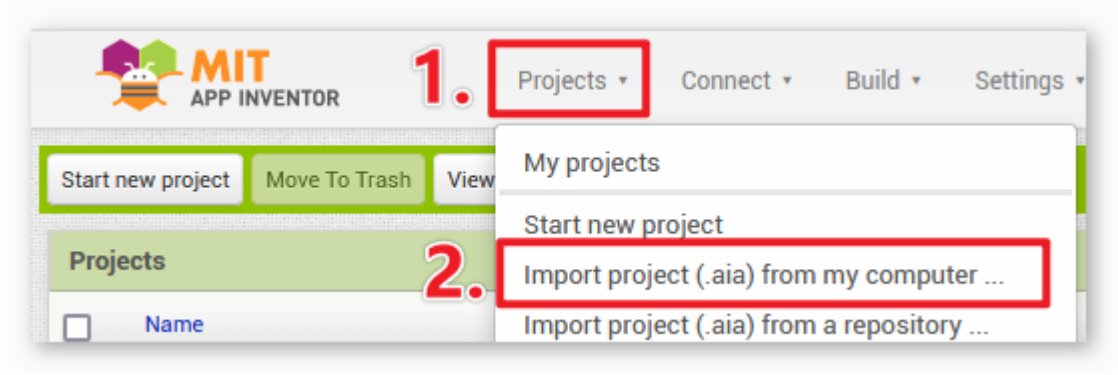
Jetzt geht's los.

1. Öffnen Sie und klicken Sie auf „Online-Tool", um sich anzumelden. Für die Registrierung bei MIT App Inventor benötigen Sie ein Google-Konto.



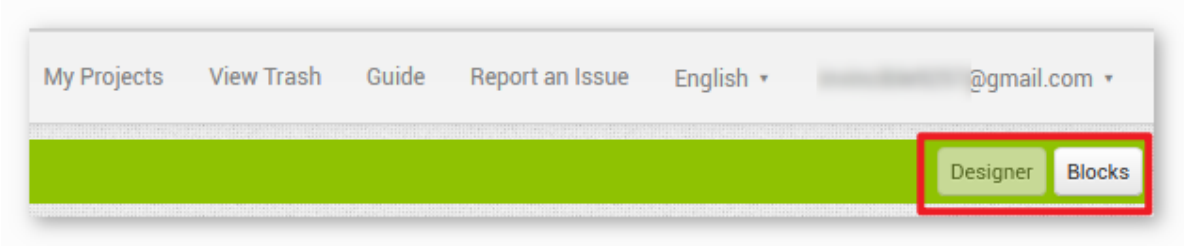
2. Nach dem Login navigieren Sie zu **Projects** -> **Import project (.aia) from my computer**. Laden Sie dann die `VoiceControl.aia`-Datei hoch, die im Pfad `ultimate-sensor-kit\iot_project\bluetooth\09-Bluetooth_voice_control_relay` zu finden ist.

   Hier können Sie auch direkt herunterladen: `VoiceControl.aia`

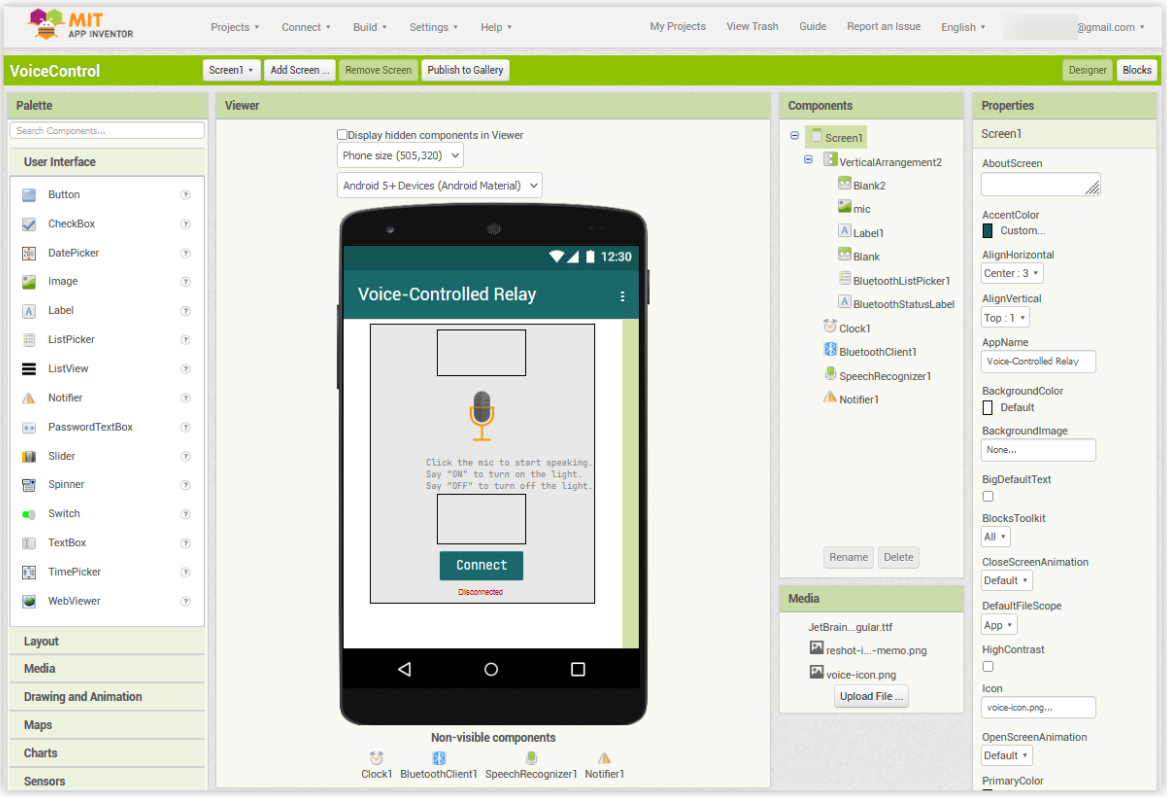


3. Nach dem Hochladen der `.aia`-Datei erscheint die Anwendung in der MIT App Inventor-Software. Dies ist eine vorkonfigurierte Vorlage, die Sie nach dem Kennenlernen des MIT App Inventors nach Ihren Wünschen anpassen können.

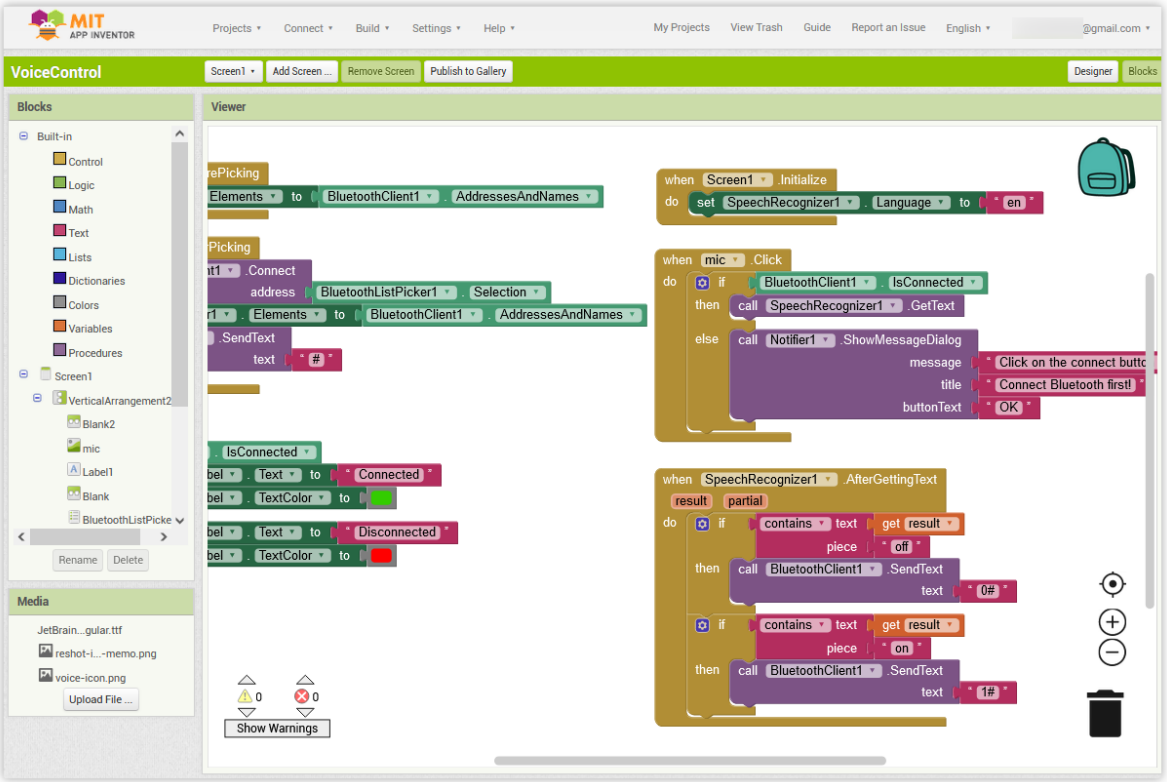
4. Im MIT App Inventor gibt es zwei Hauptbereiche: den **Designer** und die **Blocks**. Sie können oben rechts auf der Seite zwischen diesen beiden Bereichen wechseln.



5. Der **Designer** ermöglicht Ihnen, Schaltflächen, Text, Bildschirme und das allgemeine Erscheinungsbild Ihrer Anwendung zu gestalten.

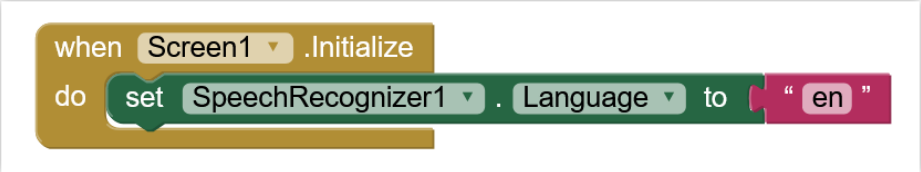


6. Als nächstes kommt der Bereich **Blocks**. Hier können Sie individuelle Funktionen für Ihre App erstellen und jedes Element auf der Benutzeroberfläche der App programmieren, um die gewünschten Funktionen zu erreichen.

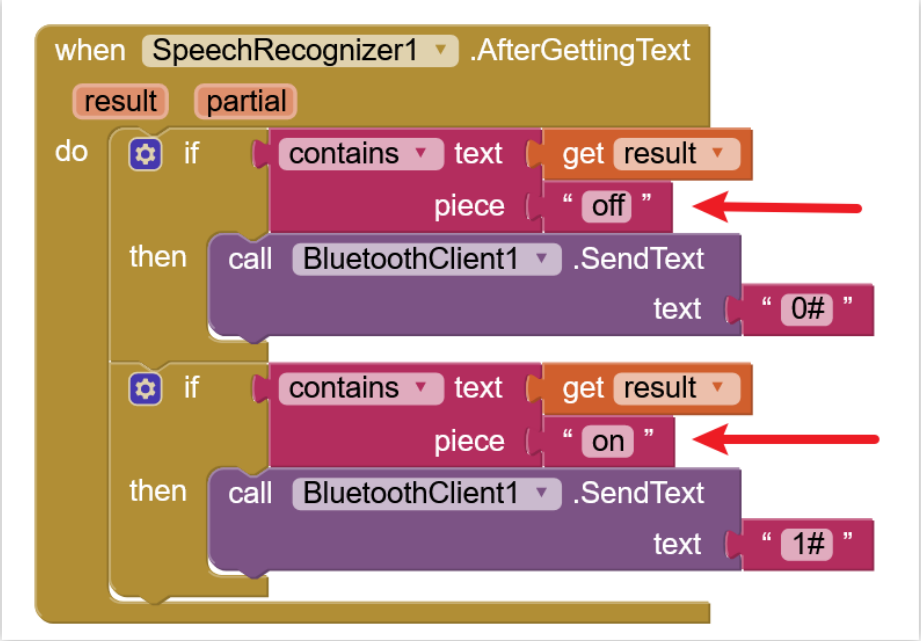


In diesem Projekt verwenden wir Englisch als Beispiel für die Spracherkennung. Wenn Sie eine Erkennung in einer anderen Sprache wünschen, müssen Sie den unten stehenden Codeblock anpassen und die APK selbst kompilieren.

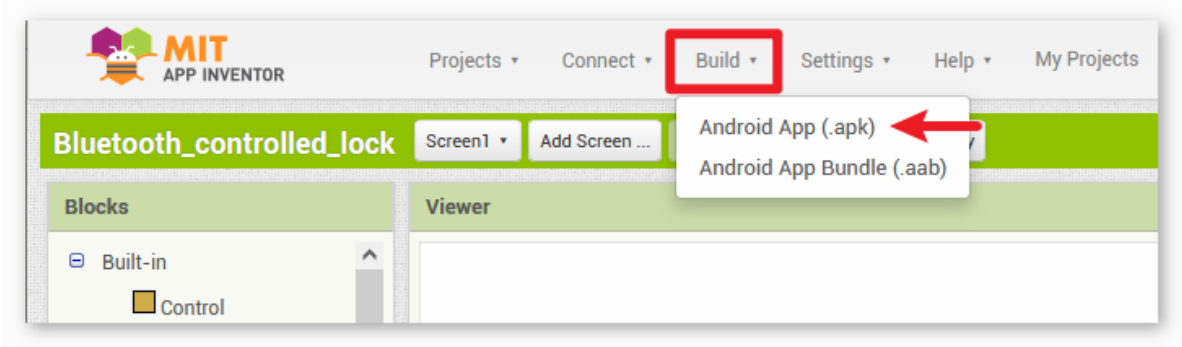
Zunächst müssen Sie `SpeechRecognizer1.Language` auf den **Sprachcode** der gewünschten Sprache setzen. Sprachen werden durch einen Sprachcode mit einer optionalen Regionskennung angegeben, wie zum Beispiel `en`, `de` oder `ja`. Der Sprachcode kann unter gefunden werden.



Danach müssen Sie die entsprechende Bedingung anpassen. Der durch den Pfeil im folgenden Bild gekennzeichnete Teil.



7. Um die Anwendung auf einem Smartphone zu installieren, wechseln Sie zur Registerkarte **Buildn**.



- 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 für die

Installation wählen können. Befolgen Sie die Installationsanleitung, um die Installation der Anwendung abzuschließen.

Sie können auch unsere vorkompilierte APK hier herunterladen: `VoiceControl.apk`

- Wenn Sie diese App im Google Play Store oder einem anderen App-Marktplatz veröffentlichen möchten, können Sie eine `.aab`-Datei generieren.

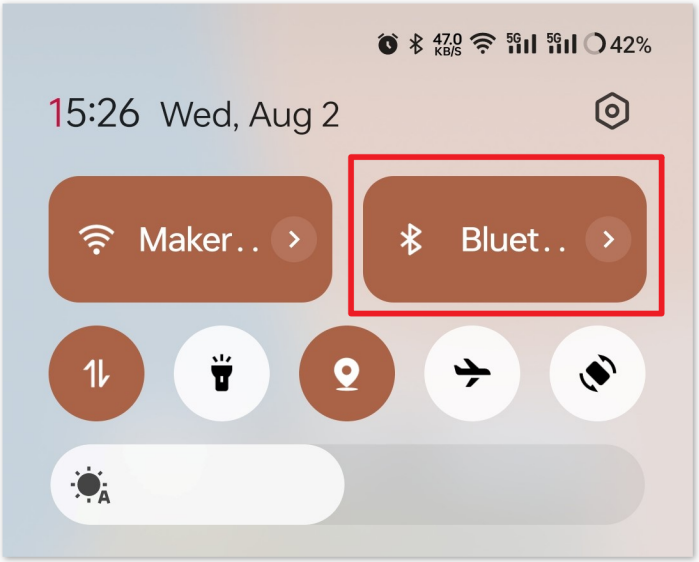
3. Den Code hochladen

1. Öffnen Sie die Datei `09-Bluetooth_voice_control_relay.ino` im Verzeichnis `ultimate-sensor-kit\iot_project\bluetooth\09-Bluetooth_voice_control_relay` oder fügen Sie den Code in die **Arduino IDE** ein.
2. Wählen Sie das passende Board und den entsprechenden Port aus und klicken Sie anschließend auf die Schaltfläche **Hochladen**.
3. Öffnen Sie den Seriellen Monitor und setzen Sie die Baudrate auf **9600**, um Debug-Meldungen einzusehen.

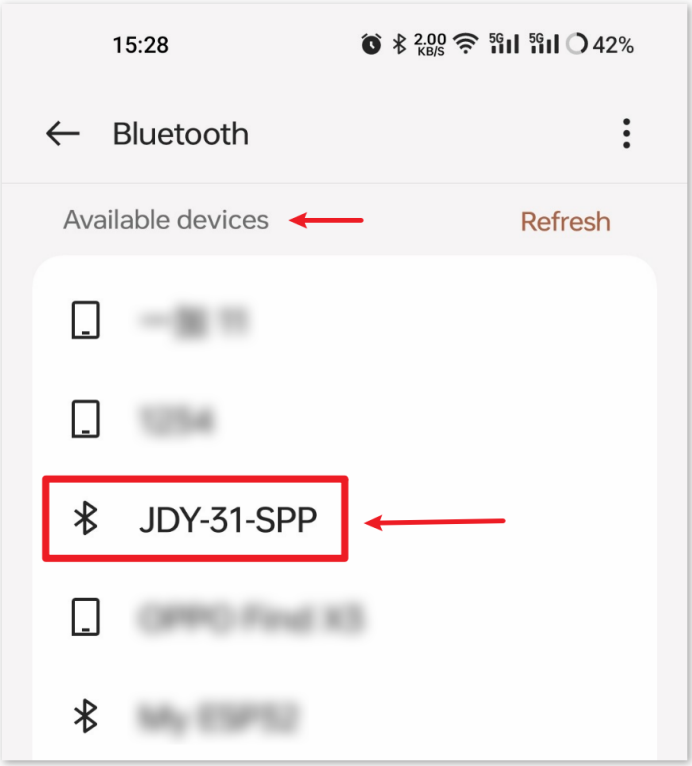
4. Verbindung der App mit dem Bluetooth-Modul

Stellen Sie sicher, dass die zuvor erstellte App auf Ihrem Smartphone installiert ist.

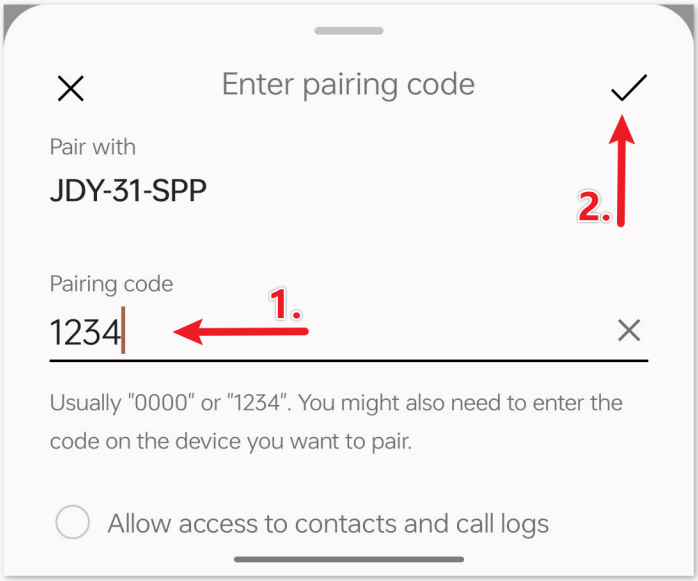
1. Aktivieren Sie zunächst **Bluetooth** auf Ihrem Smartphone.



2. Navigieren Sie zu den **Bluetooth-Einstellungen** auf Ihrem Smartphone und suchen Sie nach Gerätenamen wie **JDY-31-SPP**.

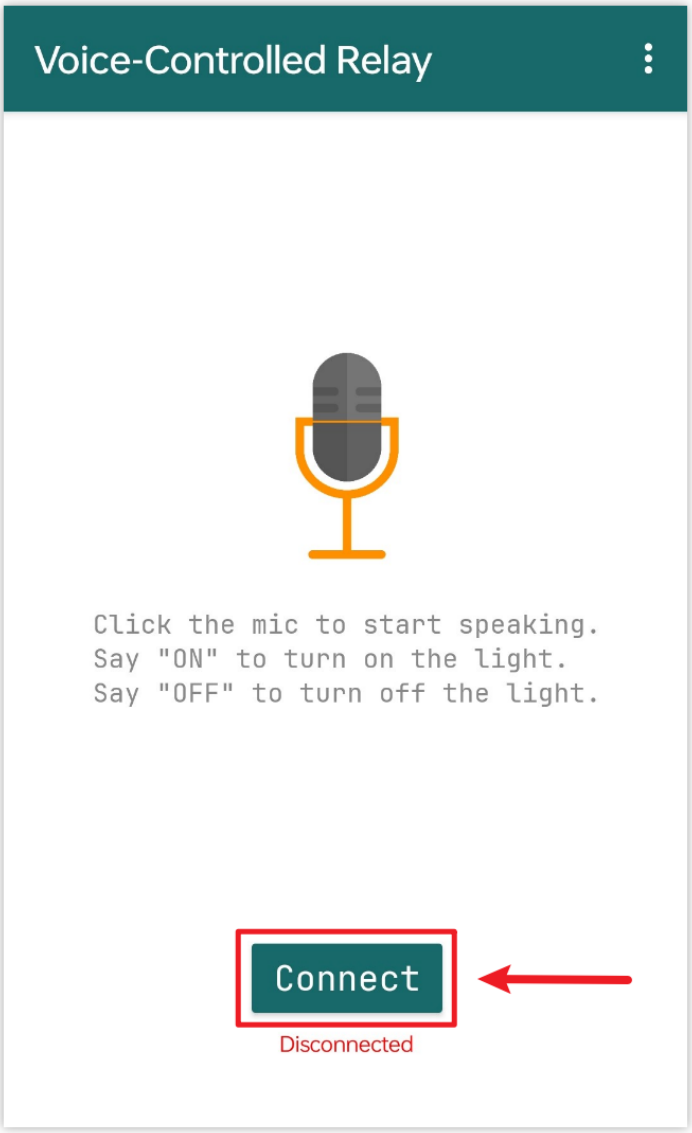


3. Nach dem Anklicken bestätigen Sie die **Kopplungsanfrage** im aufpoppenden Fenster. Falls ein Kopplungscode erforderlich ist, geben Sie „1234“ ein.



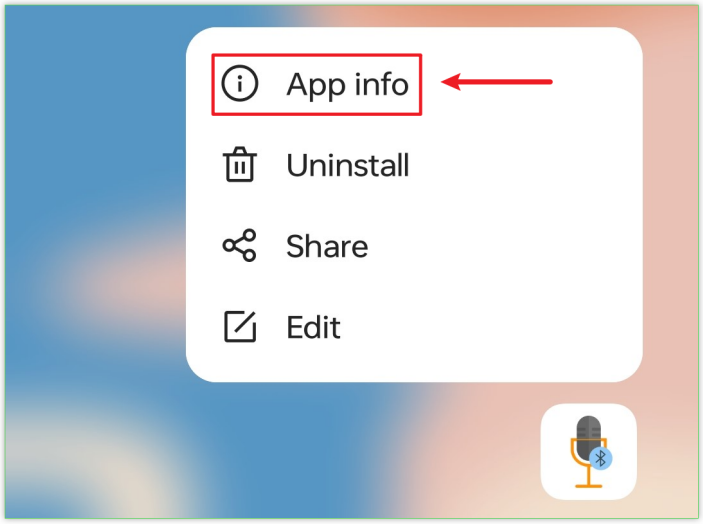
4. Öffnen Sie nun die frisch installierte **Voice-Controlled Relay**-App.

5. In der App klicken Sie auf die Schaltfläche **Connect**, um die Verbindung zwischen der App und dem Bluetooth-Modul herzustellen.

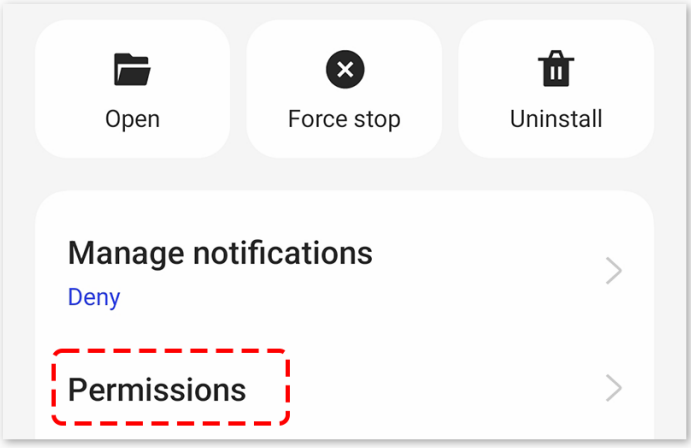


6. Diese Seite zeigt eine Liste aller gekoppelten Bluetooth-Geräte. Wählen Sie die Option `xx.xx.xx.xx.xx.xx JDY-31-SPP` aus der Liste. Der Gerätename wird neben der jeweiligen MAC-Adresse angezeigt.

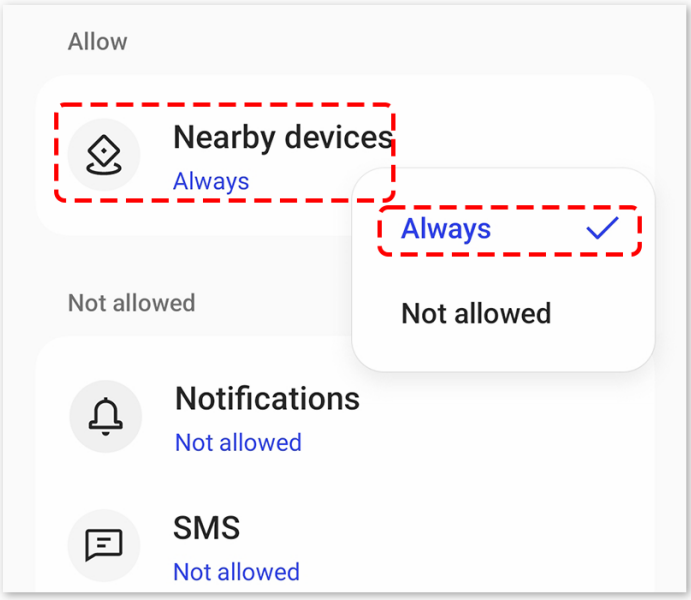
7. Sollten keine Geräte angezeigt werden, könnte dies daran liegen, dass der App die nötigen Berechtigungen fehlen. In diesem Fall müssen Sie die Einstellungen manuell anpassen.
    - Um zur Seite **App-Informationen** zu gelangen, halten Sie das App-Symbol gedrückt und wählen es aus.



    - Navigieren Sie zur **Berechtigungsseite**.

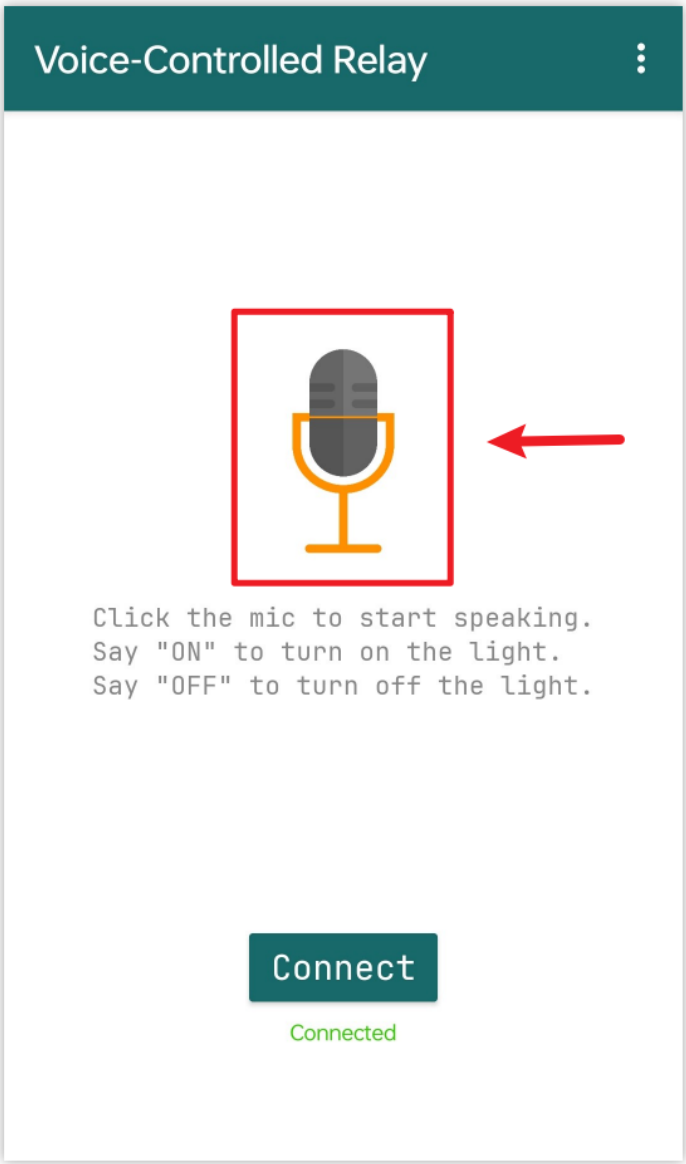


- Aktivieren Sie unter **Nahegelegene Geräte** die Option **Immer**, damit die App nach Geräten in der Umgebung suchen kann.



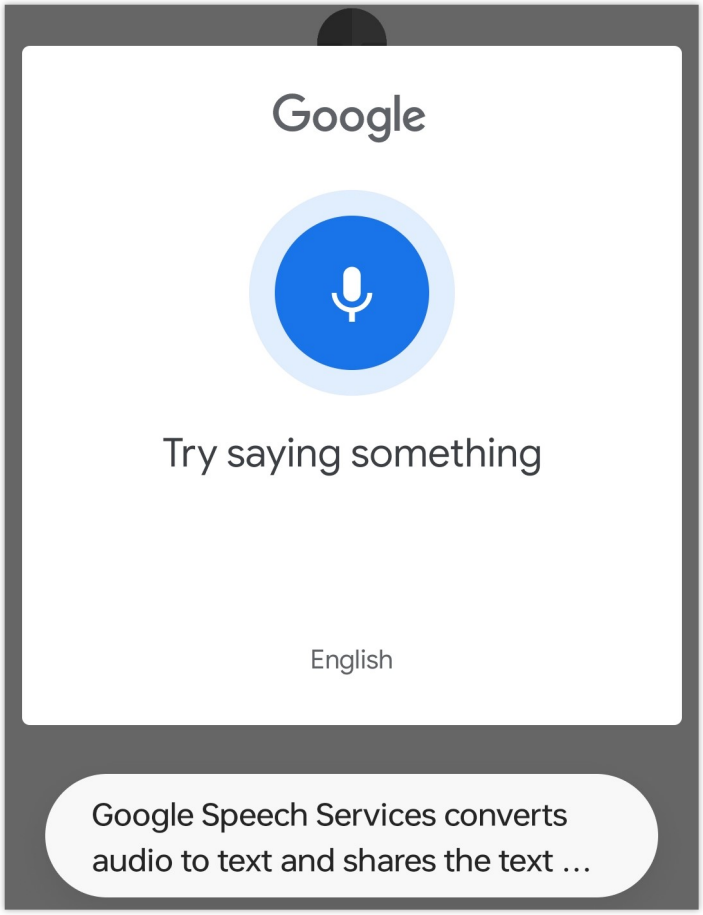
- Starten Sie die App neu und wiederholen Sie die Schritte 5 und 6 für eine erfolgreiche Bluetooth-Verbindung.

8. Nach erfolgreicher Verbindung gelangen Sie zur Hauptseite. Dort können Sie das Relais über die Schaltflächen „ON“ oder „OFF“ steuern.



Zwar können Sie das Relais auch mit kurzen Sprachbefehlen wie „on“ oder „off“ steuern, empfehlenswert sind jedoch vollständige Sätze wie „turn on the light“, um Fehlinterpretationen zu vermeiden.

Die Spracherkennung basiert auf Googles Spracherkennungsmotor. Daher könnte es notwendig sein, im Voraus zu installieren. Bei den meisten Android-Smartphones ist diese Funktion jedoch bereits vorinstalliert.



5. Code-Erläuterung

1. Kommunikation mit dem Bluetooth-Modul einrichten

```
#include <SoftwareSerial.h>
const int bluetoothTx = 3;                              // bluetooth tx to 3 pin
const int bluetoothRx = 4;                              // bluetooth rx to 4 pin
SoftwareSerial bleSerial(bluetoothTx, bluetoothRx);  // Declare SoftwareSerial␣
↪object for Bluetooth communication
```

Dieser Abschnitt initialisiert die Bluetooth-Kommunikation mit Hilfe der SoftwareSerial-Bibliothek. Diese Bibliothek ermöglicht dem Arduino, einen zusätzlichen seriellen Port zu nutzen. Der „TX“-Pin des Bluetooth-Moduls ist mit Pin 3 und der „RX“-Pin mit Pin 4 des Arduino verbunden.

2. Variablen und Steuer-Pin für das Relais definieren

```
char character;  // Character received from Bluetooth serial
String message;  // Stores the complete message from Bluetooth
const int relayPin = 8;
```

In diesem Abschnitt deklarieren wir Variablen, um einzelne Zeichen (`character`) und die komplette Nachricht (`message`) vom Bluetooth zu speichern. Der `relayPin` wird auf Pin 8 initialisiert, der zur Steuerung des Relais verwendet wird.

3. Serielle Kommunikation initialisieren und den Modus des Relais-Pins festlegen

```
void setup() {
  Serial.begin(9600);
  bleSerial.begin(9600);
  pinMode(relayPin, OUTPUT);
}
```

In der `setup()`-Funktion initialisieren wir den Standard-Seriell-Port und den Bluetooth-Seriell-Port mit einer Baudrate von 9600. Zudem setzen wir den `relayPin` als Ausgang.

4. Bluetooth-Nachrichten lesen und das Relais steuern

```
void loop() {
  while (bleSerial.available() > 0) {
    character = bleSerial.read();
    message = message + character;
    if (character == '#') {
      message = message.substring(0, message.length() - 1);
      Serial.println();
      Serial.print("DEBUG:");
      Serial.println(message);
      if (message == "1") {
        digitalWrite(relayPin, HIGH);
        Serial.println("On");
      } else if (message == "0") {
        digitalWrite(relayPin, LOW);
        Serial.println("Off");
      }
      message = "";
      delay(200);
    }
  }
}
```

Die `loop()`-Funktion überprüft kontinuierlich auf eingehende Nachrichten von Bluetooth. Bei Erhalt einer Nachricht wird jedes Zeichen zur `message`-Zeichenfolge hinzugefügt. Sobald das Zeichen # erkannt wird, gilt die Nachricht als vollständig. Wir entfernen dann das #, geben eine Debug-Nachricht aus und prüfen den Inhalt. Wenn dieser „1“ lautet, wird das Relais eingeschaltet; bei „0“ wird es ausgeschaltet. Anschließend wird die `message`-Zeichenfolge geleert und kurz gewartet, bevor nach der nächsten Nachricht gesucht wird.

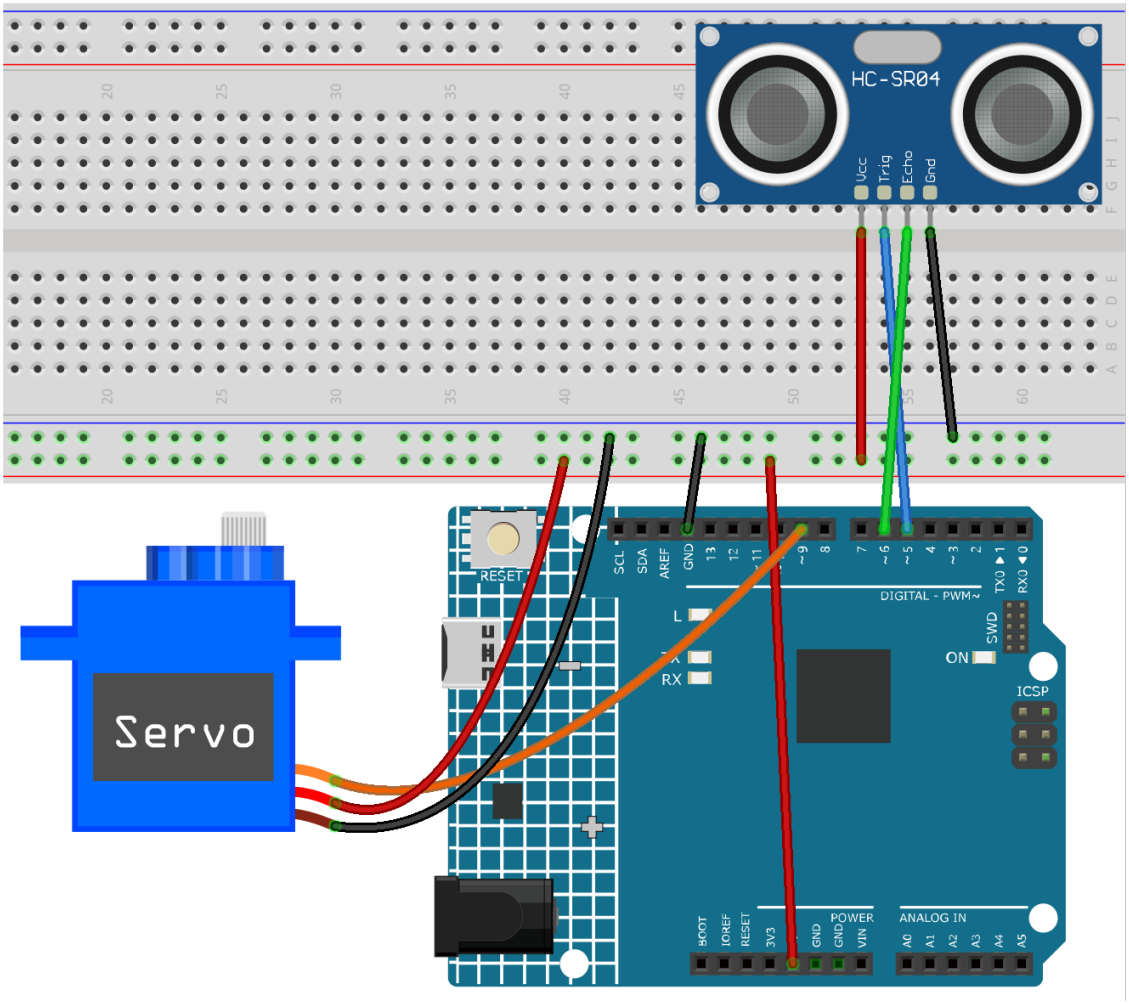
2.6 Spaßprojekte

In diesem Kapitel finden Sie einige unterhaltsame Projekte. Diese Projekte nutzen mehrere elektronische Komponenten und verdeutlichen die grundlegende Logik, wie die meisten Programme mit der Realität interagieren.

2.6.1 Intelligenter Mülleimer

Dieses Projekt beschäftigt sich mit der Idee eines intelligenten Mülleimers. Das Hauptziel ist, dass sich der Deckel des Mülleimers automatisch öffnet, wenn sich ein Objekt in einem bestimmten Abstand (in diesem Fall 20 cm) nähert. Diese Funktionalität wird durch die Kombination eines Ultraschall-Entfernungssensors mit einem Servomotor erreicht. Die Entfernung zwischen dem Objekt und dem Sensor wird kontinuierlich gemessen. Nähert sich das Objekt ausreichend, wird der Servomotor aktiviert und der Deckel geöffnet.

1. Schaltungsaufbau



- *Arduino UNO R4 Minima-Platine*
- *Ultraschall-Sensormodul (HC-SR04)*
- *Servomotor (SG90)*

2. Programmcode

1. Öffnen Sie die Datei `01-Smart_trashcan.ino` im Verzeichnis `ultimate-sensor-kit\fun_project\01-Smart_trashcan`, oder kopieren Sie diesen Code in die **Arduino IDE**.

3. Code-Erklärung

Das Projekt basiert auf der Echtzeitüberwachung der Entfernung zwischen einem Objekt und einem Mülleimer. Ein Ultraschallsensor misst diese Entfernung kontinuierlich. Nähert sich ein Objekt auf weniger als 20 cm, interpretiert der Mülleimer dies als Absicht, Abfall zu entsorgen, und öffnet automatisch seinen Deckel. Diese Automatisierung macht einen herkömmlichen Mülleimer smarter und komfortabler.

1. **Erstkonfiguration und Variablendeklaration**

   An dieser Stelle binden wir die `Servo`-Bibliothek ein und definieren die Konstanten und Variablen, die wir verwenden werden. Die Pins für den Servomotor und den Ultraschallsensor werden deklariert. Zudem steht uns ein Array `averDist` zur Verfügung, in dem die drei Entfernungsmessungen gespeichert werden.

```
#include <Servo.h>
Servo servo;
const int servoPin = 9;
const int openAngle = 0;
const int closeAngle = 90;
const int trigPin = 5;
const int echoPin = 6;
long distance, averageDistance;
long averDist[3];
const int distanceThreshold = 20;
```

2. setup() Funktion

   Die `setup()`-Funktion initialisiert die serielle Kommunikation, konfiguriert die Pins des Ultraschallsensors und bringt den Servomotor in die geschlossene Position.

```
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  servo.attach(servoPin);
  servo.write(closeAngle);
  delay(100);
}
```

3. loop() Funktion

   Die `loop()`-Funktion ist dafür verantwortlich, die Entfernung kontinuierlich zu messen, den Durchschnitt zu berechnen und auf dieser Grundlage zu entscheiden, ob der Deckel des Mülleimers geöffnet oder geschlossen werden soll.

```
void loop() {
  for (int i = 0; i <= 2; i++) {
    distance = readDistance();
    averDist[i] = distance;
    delay(10);
  }
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  averageDistance = (averDist[0] + averDist[1] + averDist[2]) / 3;
  Serial.println(averageDistance);
  if (averageDistance <= distanceThreshold) {
    servo.write(openAngle);
    delay(3500);
  } else {
    servo.write(closeAngle);
    delay(1000);
  }
}
```

4. Entfernungslesefunktion

   Diese Funktion, `readDistance()`, interagiert tatsächlich mit dem Ultraschallsensor. Sie sendet einen Impuls aus und wartet auf ein Echo. Die Zeit bis zum Eintreffen des Echos wird verwendet, um die Entfernung zwischen dem Sensor und einem Objekt davor zu berechnen.

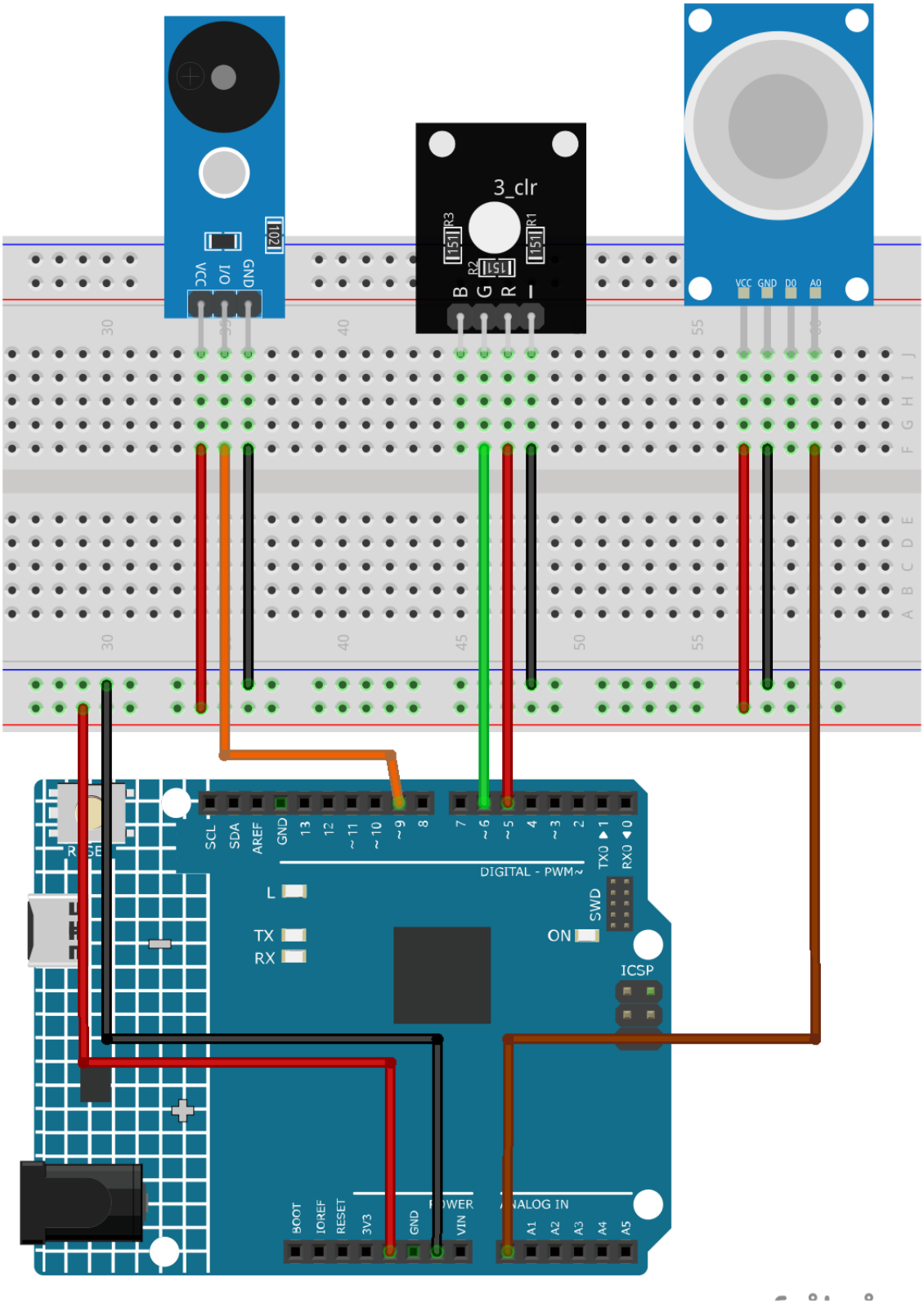
   Für weitere Informationen zum Prinzip des Ultraschallsensors siehe *Funktionsprinzip*.

```
float readDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  float distance = pulseIn(echoPin, HIGH) / 58.00;
  return distance;
}
```

2.6.2 Gasleck-Alarm

Dieses Projekt dreht sich um die Simulation eines Gaslecks mittels eines Arduino Uno Boards. Durch die Integration eines MQ-2 Gassensors und einer RGB-LED erfasst die Demonstration fortlaufend die Gaskonzentration. Überschreitet diese einen voreingestellten Grenzwert, wird ein Alarm (Summer) ausgelöst und die RGB-LED leuchtet rot auf. Bleibt die Konzentration unter dem Grenzwert, bleibt der Alarm inaktiv und die LED zeigt Grün. Es ist wichtig zu betonen, dass diese Demonstration rein illustrativ ist und echte Gasleck-Detektionssysteme nicht ersetzen sollte.

1. Schaltungsaufbau



- *Arduino UNO R4 Minima-Platine*

- *Gas-/Rauch-Sensormodul (MQ2)*
- *RGB-Modul*
- *Passives Summermodul*

2. Code

1. Öffnen Sie die Datei `02-Gas_leak_alarm.ino` im Verzeichnis `ultimate-sensor-kit\fun_project\02-Gas_leak_alarm` oder kopieren Sie den Code in die **Arduino IDE**.

3. Code-Erklärung

Das Kernprinzip des Projekts besteht darin, kontinuierlich die Gaskonzentration zu überwachen. Wenn die gemessene Konzentration einen bestimmten Schwellenwert überschreitet, wird der Alarm ausgelöst und die LED wechselt zu Rot. Dies dient als simulierter Warnmechanismus für potenziell gefährliche Zustände. Fällt die Konzentration unter den Schwellenwert, wird der Alarm deaktiviert und die LED wechselt zu Grün, was auf eine sichere Umgebung hinweist.

1. Konstanten und Variablen definieren

   Diese Zeilen deklarieren und initialisieren die Pinnummern für verschiedene Komponenten. Der `sensorPin` gibt den analogen Pin an, an dem der MQ-2-Gassensor angeschlossen ist. `sensorValue` ist eine Ganzzahl, die den analogen Ausgang des Sensors speichert. Der `buzzerPin` kennzeichnet den digitalen Pin für den Summer. Schließlich sind `RPin` und `GPin` die Pins für die roten und grünen Kanäle der RGB-LED.

```
// Define the pin numbers for the Gas Sensor
const int sensorPin = A0;
int sensorValue;

// Define the pin number for the buzzer
const int buzzerPin = 9;

// Define pin numbers for the RGB LED
const int RPin = 5;  // R channel of RGB LED
const int GPin = 6;  // G channel of RGB LED
```

2. Initialisierung in `setup()`

   Die Funktion `setup()` initialisiert die benötigten Einstellungen. Die serielle Kommunikation beginnt mit einer Baudrate von 9600, was die Anzeige der Sensorwerte im Serial Monitor ermöglicht. Die Pins für Summer und RGB-LED werden als `OUTPUT` festgelegt, um Signale an externe Komponenten zu senden.

```
void setup() {
  Serial.begin(9600);  // Start serial communication at 9600 baud rate

  // Initialize the buzzer and RGB LED pins as output
  pinMode(buzzerPin, OUTPUT);
  pinMode(RPin, OUTPUT);
  pinMode(GPin, OUTPUT);
}
```

3. Hauptprogrammschleife: Sensorauslesen und Alarm auslösen

   Die Funktion `loop()` liest kontinuierlich den Ausgang des Gassensors aus. Der Wert wird dann im Serial Monitor zur Beobachtung angezeigt. Je nach Sensorwert können zwei Szenarien eintreten:

- Überschreitet der Wert 300, wird der Summer mittels `tone()` aktiviert und die RGB-LED leuchtet rot.
- Liegt der Wert unter 300, wird der Summer mit `noTone()` stummgeschaltet und die LED leuchtet grün.

Zuletzt wird eine Verzögerung von 50 Millisekunden eingeführt, bevor die nächste Iteration der Schleife beginnt, um die Auslesefrequenz zu regulieren und die CPU-Last zu verringern.

```
void loop() {
  // Read the analog value of the gas sensor
  sensorValue = analogRead(sensorPin);

  // Print the sensor value to the serial monitor
  Serial.print("Analog output: ");
  Serial.println(sensorValue);

  // If the sensor value exceeds the threshold, trigger the alarm and make the RGB␣
↪LED red
  if (sensorValue > 300) {
    tone(buzzerPin, 500, 300);
    digitalWrite(GPin, LOW);
    digitalWrite(RPin, HIGH);
  } else {
    // If the sensor value is below the threshold, turn off the alarm and make the␣
↪RGB LED green
    noTone(buzzerPin);
    digitalWrite(RPin, LOW);
    digitalWrite(GPin, HIGH);
  }

  // Wait for 50 milliseconds before the next loop iteration
  delay(50);
}
```

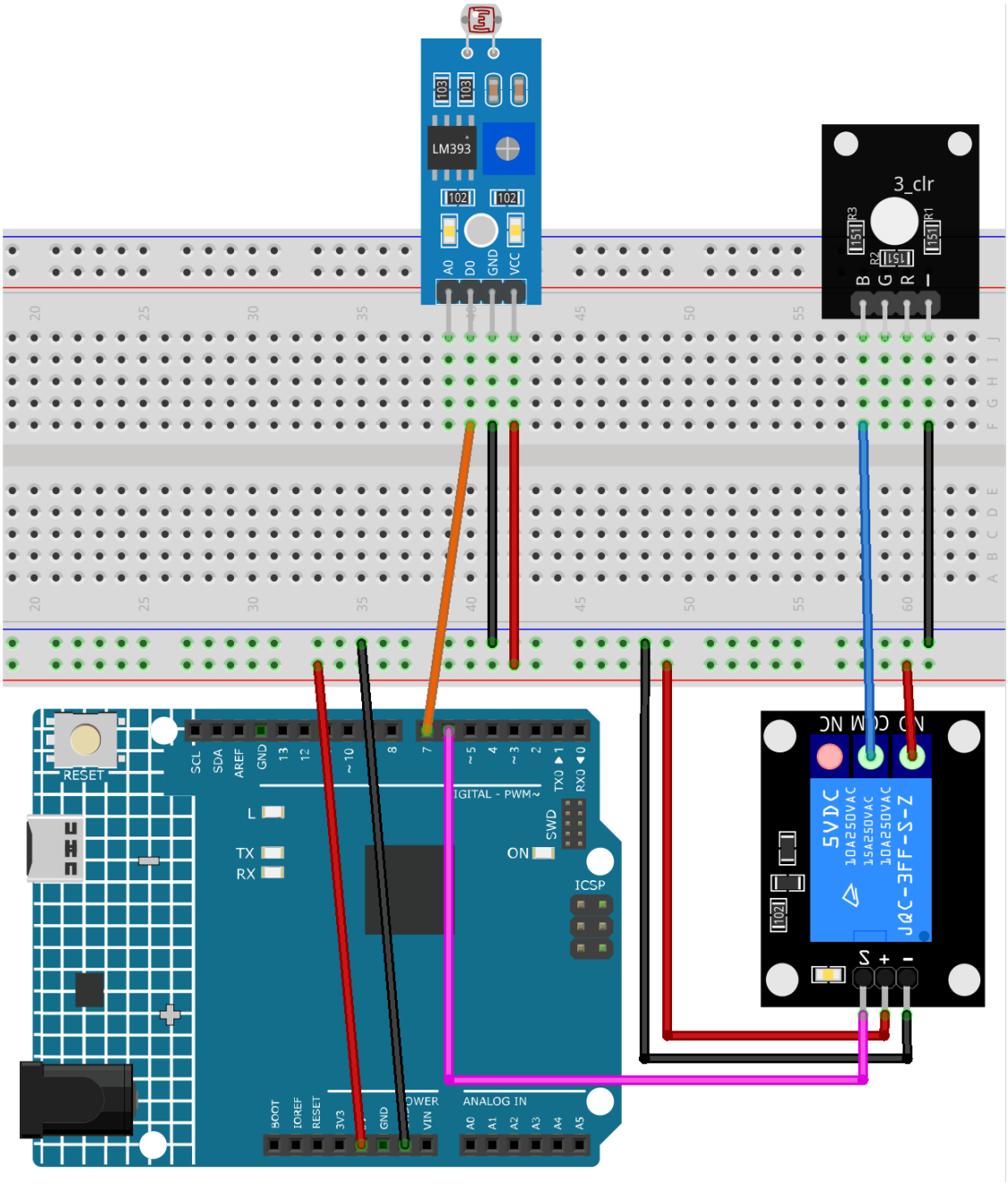
2.6.3 Lichtsteuerungsschalter

Dieses Projekt ist ein System zur Lichtsteuerung mittels Schalter. Der Hauptgedanke ist die Verwendung eines Fotowiderstand-Sensormoduls, um die Umgebungshelligkeit zu erfassen und daraufhin ein Relaismodul zu steuern. Sobald die Umgebungshelligkeit unter einen bestimmten Grenzwert fällt, wird das Relais aktiviert. Liegt die Helligkeit über dem Grenzwert, wird das Relais deaktiviert.

Warnung: In dieser Demonstration verwenden wir ein Relais, um ein RGB-LED-Modul zu steuern. Für reale Einsatzszenarien ist diese Vorgehensweise jedoch möglicherweise nicht optimal.

Während das Relais in tatsächlichen Anwendungen auch zur Steuerung anderer Geräte verwendet werden kann, ist beim Umgang mit HOHER Wechselspannung äußerste Vorsicht geboten. Unachgemäße oder fehlerhafte Anwendung kann zu schweren Verletzungen oder gar zum Tod führen. Dieses Projekt richtet sich daher an Personen, die mit HOHER Wechselspannung vertraut und sachkundig sind. Sicherheit hat immer Vorrang.

1. Schaltungsaufbau



- *Arduino UNO R4 Minima-Platine*
- cpn_photoresistor
- *5V-Relaismodul*
- *RGB-Modul*

2. Code

1. Öffnen Sie die Datei `03-fun_Light_control_switch.ino` im Verzeichnis `ultimate-sensor-kit\fun_project\03-fun_Light_control_switch` oder kopieren Sie den Code in die **Arduino IDE**.

3. Code-Erklärung

Das Kernprinzip dieses Projekts ist die Verwendung eines Fotowiderstand-Sensormoduls zur Erfassung der Umgebungshelligkeit. Fotowiderstände ändern ihren Widerstand je nach Lichteinfall. Diese Eigenschaft wird im Sensormodul genutzt, um ein digitales Ausgangssignal zu erzeugen. Fällt die Helligkeit unter den eingestellten Grenzwert, sendet der Sensor ein HIGH-Signal an das Arduino, das wiederum das Relais aktiviert.

Bemerkung: Der Fotowiderstand-Sensor verfügt über ein Potentiometer, mit dem der Grenzwert für die Ausgabe von HIGH bzw. LOW eingestellt werden kann. Je nach gewünschtem Helligkeitsniveau für die Schaltung muss dieser Wert eventuell angepasst werden.

---

1. **Konstanten und Pins definieren**

   An dieser Stelle legen wir die Pins fest, die für das Relais und den Sensor verwendet werden. Wir verwenden das Schlüsselwort `const`, da diese Pinnummern während der Laufzeit des Programms unverändert bleiben.

```
const int RelayPin = 6;
const int sensorPin = 7;
```

2. **Initialisierung in der setup()-Funktion**

   Die `setup()`-Funktion wird einmalig beim Start des Programms ausgeführt. Hier legen wir fest, dass `RelayPin` ein Ausgang ist, da wir Signale senden werden, um das Relais zu steuern. Außerdem starten wir die serielle Kommunikation mit einer Baudrate von 9600 für Debugging-Zwecke.

```
void setup() {
  // Set RelayPin as an output pin
  pinMode(RelayPin, OUTPUT);
  // Start the Serial communication for debugging
  Serial.begin(9600);
}
```

3. **Sensordaten lesen und Relais steuern**

   Die Hauptlogik findet in der `loop()`-Funktion statt. Hier wird der Wert des Fotowiderstand-Sensors wiederholt ausgelesen. Liefert der Sensor einen Wert von 1 (was auf ein Unterschreiten des Helligkeitsgrenzwerts hindeutet), wird das Relais durch Setzen von `RelayPin` auf HIGH aktiviert. Andernfalls wird es durch Setzen auf LOW deaktiviert.

```
void loop() {
  // Read the value from the photoresistance sensor module
  const int sensorValue = digitalRead(sensorPin);
  // If the light level is lower than the threshold (sensor value equals 1),
  // switch the relay module ON.
  if (sensorValue == 1) {
    digitalWrite(RelayPin, HIGH);
  } else
  // If the light level is higher than the threshold (sensor value equal 0),
```

(Fortsetzung auf der nächsten Seite)

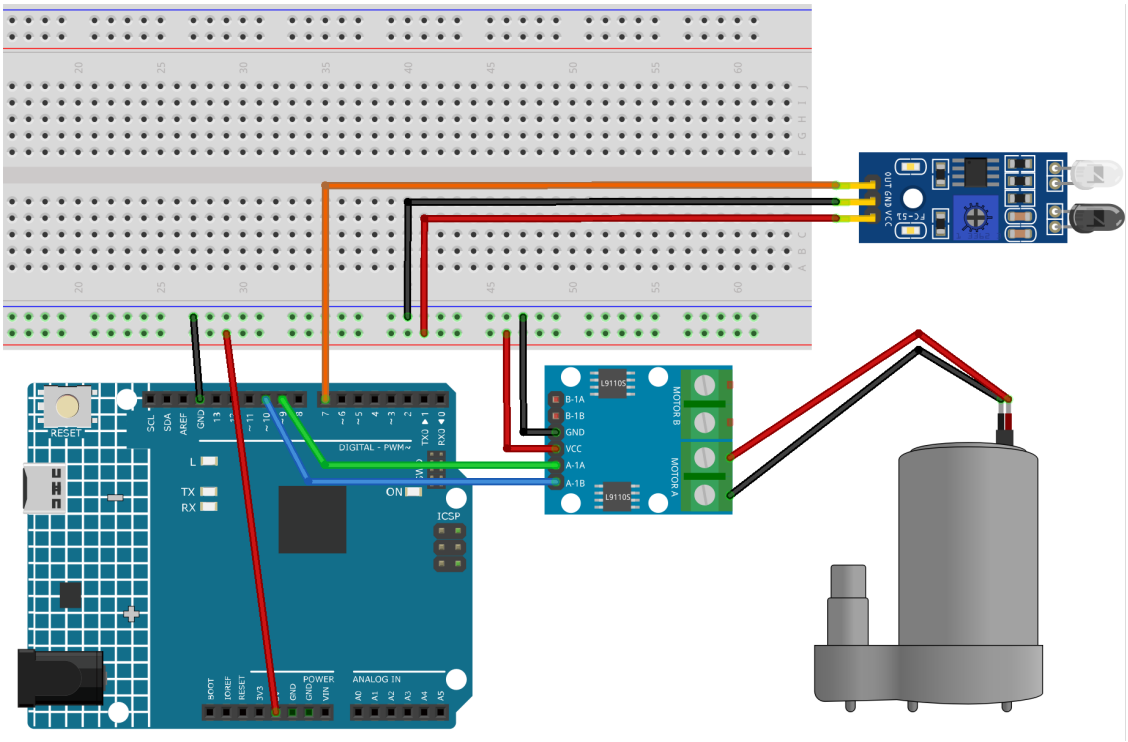
(Fortsetzung der vorherigen Seite)

```
  // switch the relay module OFF.
  {
    digitalWrite(RelayPin, LOW);
  }
}
```

2.6.4 Automatischer Seifenspender

Das Projekt „Automatischer Seifenspender“ nutzt ein Arduino Uno Board zusammen mit einem Infrarot-Hindernisvermeidungssensor und einer Wasserpumpe. Der Sensor erkennt die Anwesenheit eines Objekts wie einer Hand, was die Wasserpumpe aktiviert, um Seife zu spenden.

1. Schaltkreis aufbauen



- *Arduino UNO R4 Minima-Platine*
- *IR-Hindernisvermeidungssensor-Modul*
- *Kreiselpumpe*

2. Programmcode

1. Öffnen Sie die Datei `04-Automatic_soap_dispenser.ino` im Pfad `ultimate-sensor-kit\fun_project\04-Automatic_soap_dispenser` oder kopieren Sie diesen Code in die **Arduino IDE**.

3. Code-Erklärung

Die Hauptidee dieses Projekts besteht darin, ein berührungsloses Seifenspendersystem zu schaffen. Der Infrarot-Hindernisvermeidungssensor erkennt, wenn sich ein Objekt (wie eine Hand) nähert. Bei Erkennung eines Objekts sendet der Sensor ein Signal an den Arduino, der daraufhin die Wasserpumpe aktiviert, um Seife zu spenden. Die Pumpe bleibt für eine kurze Zeit aktiv, spendet Seife und schaltet dann ab.

1. **Definieren der Pins für den Sensor und die Pumpe**

   In diesem Code-Ausschnitt definieren wir die Arduino-Pins, die mit dem Sensor und der Pumpe verbunden sind. Pin 7 ist als Sensor-Pin definiert und wir verwenden die Variable `sensorValue`, um die von diesem Sensor gelesenen Daten zu speichern. Für die Wasserpumpe verwenden wir zwei Pins, 9 und 10.

```
const int sensorPin = 7;
int sensorValue;
const int pump1A = 9;
const int pump1B = 10;
```

2. **Einrichten des Sensors und der Pumpe**

   In der Funktion `setup()` legen wir die Modi für die verwendeten Pins fest. Der Sensor-Pin ist als `INPUT` eingestellt, da er Daten vom Sensor empfangen wird. Die Pumpen-Pins sind als `OUTPUT` eingestellt, da sie Befehle an die Pumpe senden werden. Wir stellen sicher, dass der Pin `pump1B` in einem `LOW`-Zustand (aus) startet und beginnen die serielle Kommunikation mit einer Baudrate von 9600.

```
void setup() {
  pinMode(sensorPin, INPUT);
  pinMode(pump1A, OUTPUT);
  pinMode(pump1B, OUTPUT);
  digitalWrite(pump1B, LOW);
  Serial.begin(9600);
}
```

3. **Kontinuierliche Überprüfung des Sensors und Steuerung der Pumpe**

   In der Funktion `loop()` liest der Arduino ständig den Wert vom Sensor mit `digitalRead()` und weist ihn `sensorValue()` zu. Dieser Wert wird dann zum Seriellen Monitor für Debugging-Zwecke ausgegeben. Wenn der Sensor ein Objekt erkennt, wird `sensorValue()` 0 sein. In diesem Fall wird `pump1A` auf `HIGH` gesetzt, wodurch die Pumpe aktiviert wird, und eine Verzögerung von 700 Millisekunden ermöglicht es der Pumpe, Seife zu spenden. Die Pumpe wird dann durch Setzen von `pump1A` auf `LOW` deaktiviert, und eine Verzögerung von 1 Sekunde gibt dem Benutzer Zeit, seine Hand wegzubewegen, bevor der Zyklus sich wiederholt.

```
void loop() {
  sensorValue = digitalRead(sensorPin);
  Serial.println(sensorValue);
  if (sensorValue == 0) {
    digitalWrite(pump1A, HIGH);
    delay(700);
    digitalWrite(pump1A, LOW);
    delay(1000);
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  }
}
```

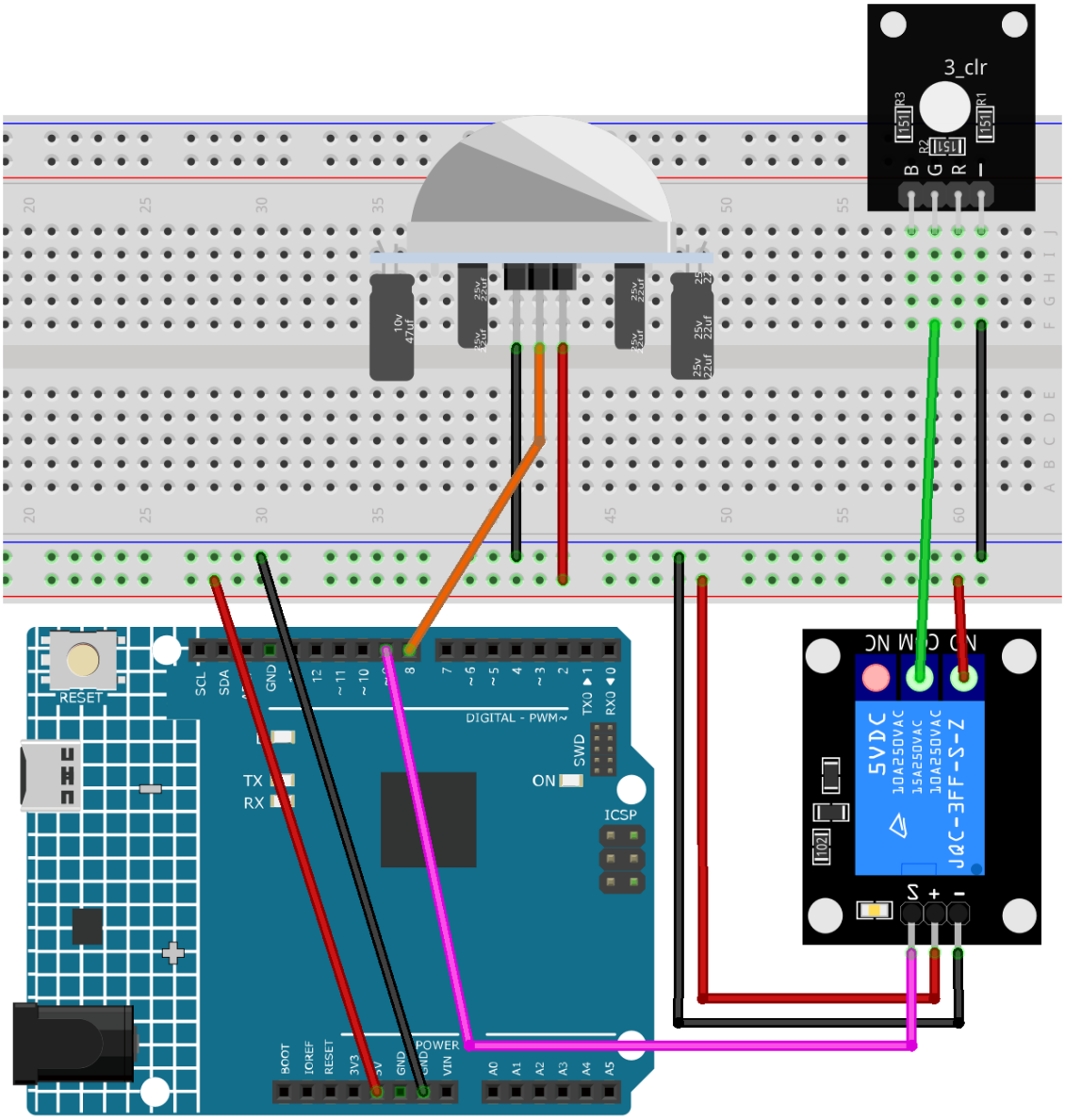
2.6.5 Bewegungsgesteuertes Relais

Dieses Arduino-Projekt hat zum Ziel, eine mit einem Relais gesteuerte Lampe mithilfe eines passiven Infrarot-(PIR)-Sensors zu steuern. Sobald der PIR-Sensor eine Bewegung erkennt, wird das Relais aktiviert und die Lampe eingeschaltet. Die Lampe bleibt für 5 Sekunden nach der zuletzt erkannten Bewegung eingeschaltet.

Warnung: Zur Demonstration verwenden wir ein Relais zur Steuerung eines RGB-LED-Moduls. In realen Anwendungsfällen ist diese Herangehensweise jedoch eventuell nicht die praktikabelste.

Beim Anschluss des Relais an andere Geräte in realen Anwendungen ist äußerste Vorsicht im Umgang mit HOHER Wechselspannung geboten. Unfachgemäße oder falsche Handhabung kann zu schweren Verletzungen oder sogar zum Tod führen. Daher richtet sich dieses Projekt an Personen, die sich mit HOHER Wechselspannung auskennen. Sicherheit hat stets oberste Priorität.

1. Schaltkreis aufbauen



- *Arduino UNO R4 Minima-Platine*
- cpn_pir_motion
- *5V-Relaismodul*
- *RGB-Modul*

2. Code

1. Öffnen Sie die Datei `05-Motion_triggered_relay.ino` im Verzeichnis `ultimate-sensor-kit\fun_project\05-Motion_triggered_relay` oder kopieren Sie diesen Code in die **Arduino IDE**.

3. Code-Erklärung

Das Projekt basiert auf der Fähigkeit des PIR-Bewegungssensors, Bewegungen zu erkennen. Bei erkannter Bewegung sendet der Sensor ein Signal an den Arduino, der das Relais aktiviert, wodurch wiederum eine Lampe eingeschaltet wird. Die Lampe bleibt für eine festgelegte Zeit (in diesem Fall 5 Sekunden) nach der letzten erkannten Bewegung an, sodass der Bereich kurzzeitig beleuchtet bleibt, auch wenn keine Bewegung mehr erfolgt.

1. **Initiale Einrichtung und Variablendeklarationen**

   In diesem Abschnitt werden Konstanten und Variablen definiert, die im gesamten Code verwendet werden. Wir legen die Pins für das Relais und den PIR-Sensor sowie eine Zeitverzögerungskonstante für die Bewegung fest. Außerdem gibt es eine Variable zur Verfolgung des letzten Zeitpunkts einer erkannten Bewegung und ein Flag zur Überwachung, ob eine Bewegung erkannt wurde.

```
// Define the pin number for the relay
const int relayPin = 9;

// Define the pin number for the PIR sensor
const int pirPin = 8;

// Motion delay threshold in milliseconds
const unsigned long MOTION_DELAY = 5000;

unsigned long lastMotionTime = 0;  // Timestamp of the last motion detection
bool motionDetected = false;       // Flag to track if motion is detected
```

2. **Konfiguration der Pins in der setup() Funktion**

   In der `setup()` Funktion konfigurieren wir die Pinmodi für das Relais und den PIR-Sensor und initialisieren das Relais so, dass es zu Beginn ausgeschaltet ist.

```
void setup() {
  pinMode(relayPin, OUTPUT);    // Set relayPin as an output pin
  pinMode(pirPin, INPUT);       // Set the PIR pin as an input
  digitalWrite(relayPin, LOW);  // Turn off the relay initially
}
```

3. **Hauptlogik in der loop() Funktion**

   Die `loop()` Funktion enthält die Hauptlogik. Wenn der PIR-Sensor eine Bewegung erkennt, sendet er ein `HIGH` Signal, schaltet das Relais ein und aktualisiert die `lastMotionTime`. Wenn innerhalb der festgelegten Verzögerung (in diesem Fall 5 Sekunden) keine Bewegung mehr erkannt wird, wird das Relais ausgeschaltet.

   Diese Methode gewährleistet, dass die Lampe auch bei sporadischen oder kurzen Bewegungen für mindestens 5 Sekunden nach der letzten erkannten Bewegung eingeschaltet bleibt und somit eine gleichmäßige Beleuchtungsdauer erreicht wird.

```
void loop() {
  if (digitalRead(pirPin) == HIGH) {
    lastMotionTime = millis();     // Update the last motion time
```

(Fortsetzung auf der nächsten Seite)

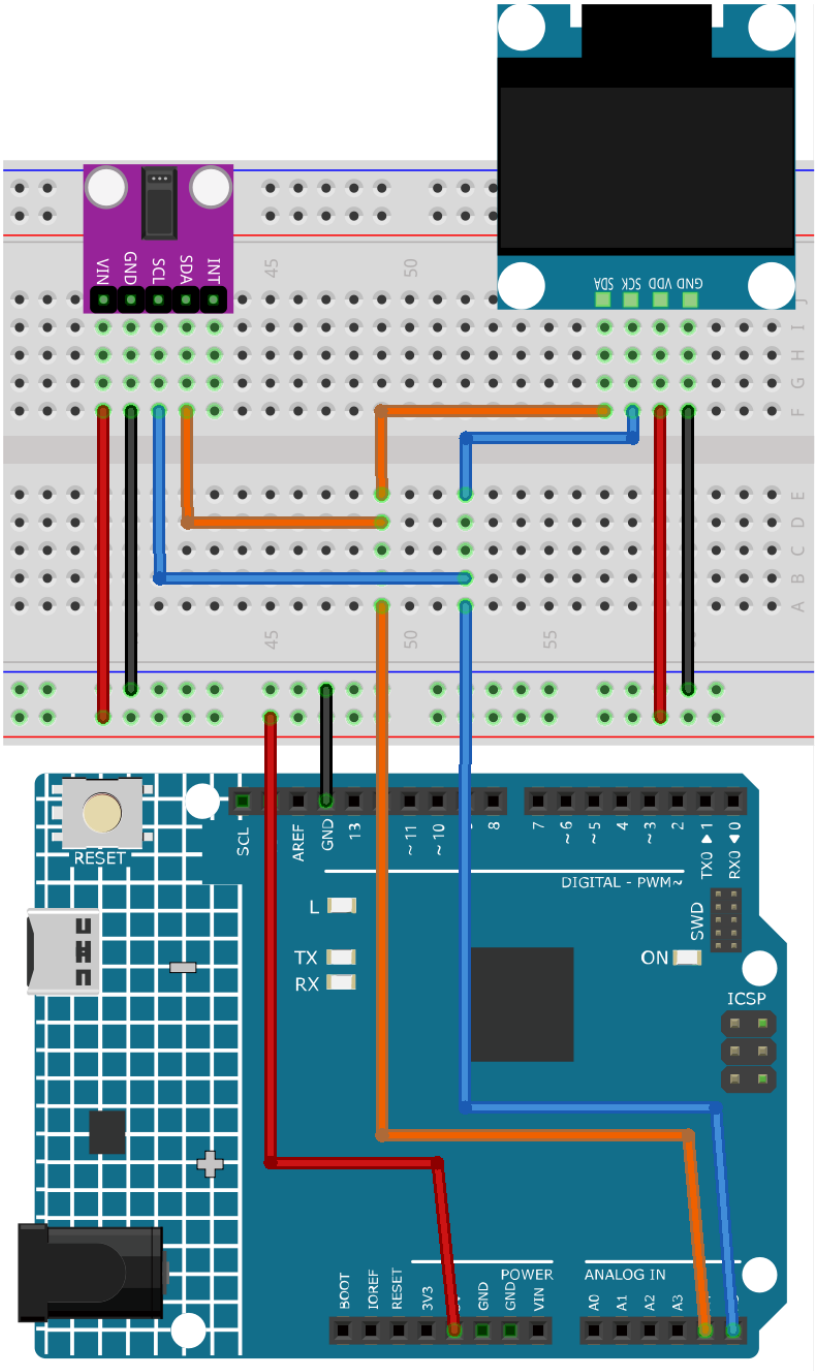
(Fortsetzung der vorherigen Seite)

```
    digitalWrite(relayPin, HIGH);  // Turn on the relay (and hence the light)
    motionDetected = true;
  }

  // If motion was detected earlier and 5 seconds have elapsed, turn off the relay
  if (motionDetected && (millis() - lastMotionTime >= MOTION_DELAY)) {
    digitalWrite(relayPin, LOW);  // Turn off the relay
    motionDetected = false;
  }
}
```

2.6.6 Herzfrequenzmonitor

Dieses Arduino-Projekt dient dem Aufbau eines einfachen Herzfrequenzmonitors mit einem MAX30102-Pulsoximeter-Sensor und einem SSD1306-OLED-Display. Der Code erfasst die Herzfrequenz durch Messung der Zeitintervalle zwischen den Herzschlägen. Nach vier Messungen wird deren Durchschnitt berechnet und auf dem OLED-Bildschirm angezeigt. Erkennt der Sensor keinen Finger, wird dem Benutzer eine Aufforderung angezeigt, den Finger korrekt auf dem Sensor zu positionieren.

1. Schaltung aufbauen



- *Arduino UNO R4 Minima-Platine*
- *Pulsoximeter und Herzfrequenzsensor (MAX30102)*
- *OLED-Display-Modul*

2. Code

1. Öffnen Sie die Datei `06-Heart_rate_monitor.ino` im Verzeichnis `ultimate-sensor-kit\fun_project\06-Heart_rate_monitor` oder kopieren Sie diesen Code in die **Arduino IDE**.

Bemerkung: Um die Bibliothek zu installieren, verwenden Sie den Arduino Library Manager und suchen nach **„SparkFun MAX3010x“** und installieren Sie diese.

---

3. Code-Erklärung

Der Hauptgedanke dieses Projekts besteht darin, die Pulsation des Blutflusses durch einen Finger mithilfe des MAX30102-Sensors zu erfassen. Durch die Blutzirkulation im Körper verändert sich das Blutvolumen in den Gefäßen der Fingerspitze geringfügig. Der Sensor erkennt diese minimalen Veränderungen, indem er Licht durch den Finger sendet und misst, wie viel davon absorbiert oder reflektiert wird. Die Zeitintervalle zwischen den einzelnen Pulsschlägen dienen zur Berechnung der Herzfrequenz in Schlägen pro Minute (BPM). Dieser Wert wird dann aus vier Messungen gemittelt und auf dem OLED-Display angezeigt.

1. **Einbindung von Bibliotheken und erste Deklarationen**:

   Der Code beginnt mit der Einbindung der erforderlichen Bibliotheken für das OLED-Display, den MAX30102-Sensor und die Herzfrequenzberechnung. Zusätzlich werden die Konfiguration für das OLED-Display und die Variablen für die Herzfrequenzberechnung deklariert.

   Bemerkung: Um die Bibliothek zu installieren, verwenden Sie den Arduino Library Manager und suchen nach **„SparkFun MAX3010x“** und installieren Sie diese.

   ```
   #include <Adafruit_GFX.h>  // OLED  libraries
   #include <Adafruit_SSD1306.h>
   #include <Wire.h>
   #include "MAX30105.h"   // MAX3010x library
   #include "heartRate.h"  // Heart rate  calculating algorithm

   // ... Variables and OLED configuration
   ```

   In diesem Projekt haben wir auch einige Bitmaps erstellt. Das Schlüsselwort „PROGMEM“ zeigt an, dass das Array im Programmspeicher des Arduino-Mikrocontrollers gespeichert ist. Die Verwendung von Programmspeicher (PROGMEM) anstelle von RAM ist bei großen Datenmengen sinnvoll, die sonst zu viel RAM belegen würden.

   ```
   static const unsigned char PROGMEM beat1_bmp[] = {...}

   static const unsigned char PROGMEM beat2_bmp[] = {...}
   ```

2. **Setup-Funktion**:

   Initialisiert die I2C-Kommunikation, startet die serielle Kommunikation, aktiviert das OLED-Display und konfiguriert den MAX30102-Sensor.

   ```
   void setup() {
       Wire.setClock(400000);
       Serial.begin(9600);
   ```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
    // ... Rest of the setup code
```

3. **Hauptschleife**:

   Hier befindet sich die Kernfunktionalität. Der IR-Wert wird vom Sensor gelesen. Wenn ein Finger erkannt wird (IR-Wert über 50.000), prüft das Programm, ob ein Herzschlag erfasst wurde. Bei Erkennung eines Herzschlags aktualisiert das OLED-Display die BPM und die Zeit zwischen den Herzschlägen wird zur BPM-Berechnung herangezogen. Andernfalls wird der Benutzer aufgefordert, seinen Finger auf den Sensor zu legen.

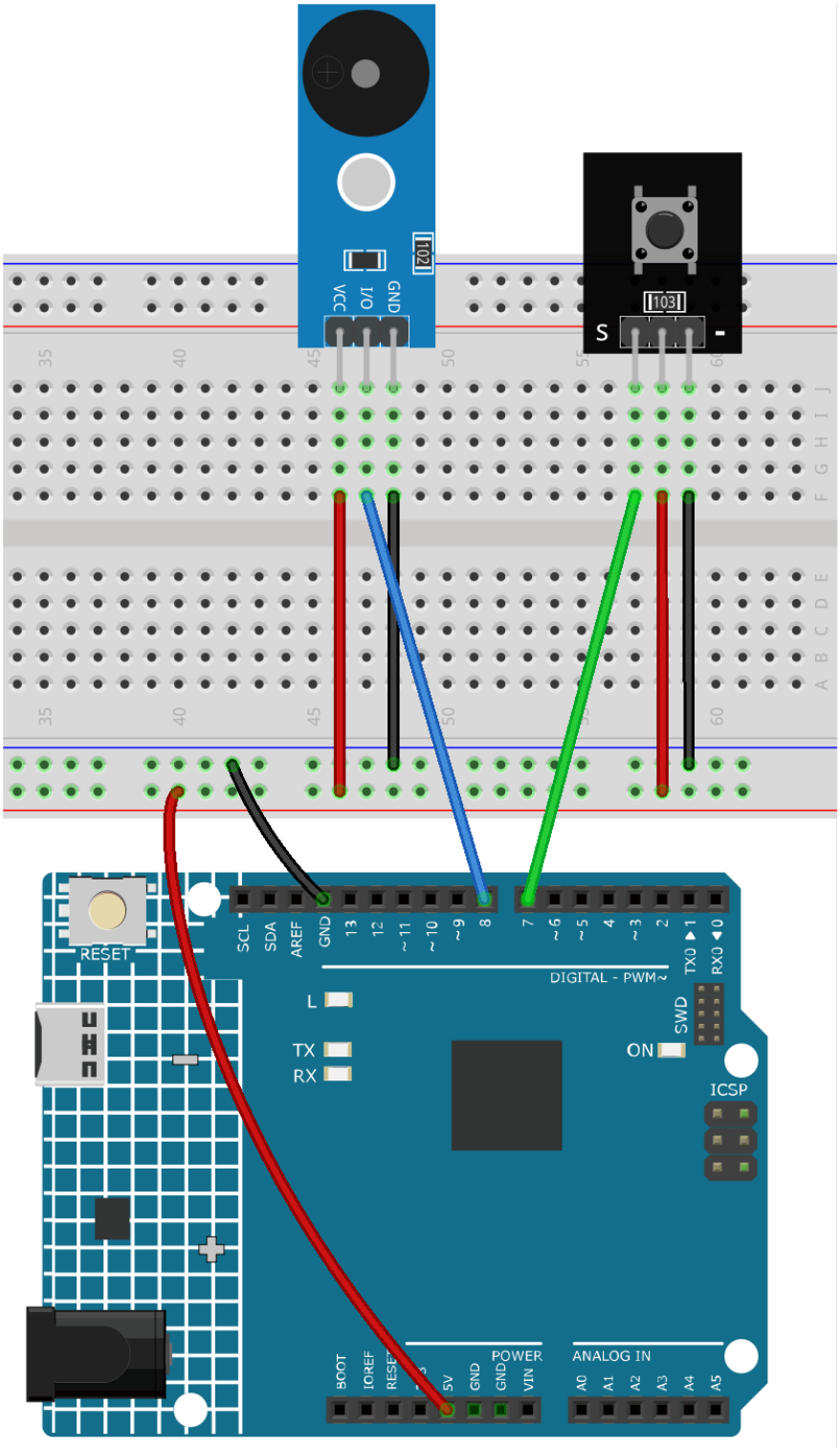
   Um einen dynamischen visuellen Effekt zu erzielen, haben wir auch zwei Bitmaps mit Herzschlägen vorbereitet. Durch das Wechseln zwischen diesen beiden Bitmaps wird der Effekt erzeugt.

```
void loop() {
  // Get IR value from sensor
  long irValue = particleSensor.getIR();

  //If a finger is detected
  if (irValue > 50000) {

    // Check if a beat is detected
    if (checkForBeat(irValue) == true) {

      // Update OLED display
      // Calculate the BPM

      // Calculate the average BPM
      //Print the IR value, current BPM value, and average BPM value to the serial␣
↪monitor

      // Update OLED display

    }
  }
  else {
    // ... Prompt to place the finger on the sensor
  }
}
```

2.6.7 Türklingel

Das Projekt „Türklingel“ zielt darauf ab, die Funktionalität einer echten Türklingel zu simulieren. Bei Betätigung eines Knopfes spielt der Arduino eine vordefinierte Melodie über ein passives Summermodul ab.

1. Schaltungsaufbau



- *Arduino UNO R4 Minima-Platine*
- *Tastenmodul*

- *Passives Summermodul*

2. Programmcode

1. Öffnen Sie die Datei `07-Doorbell.ino` im Pfad `ultimate-sensor-kit\fun_project\07-Doorbell`, oder kopieren Sie diesen Code in die **Arduino IDE**.

3. Code-Erläuterung

Die grundlegende Idee dieses Projekts ist die Verwendung eines Arduino Uno Boards zur Erfassung eines Knopfdrucks, der wiederum eine Melodie auf dem passiven Summer abspielt. Die Melodie besteht aus einer Abfolge von Noten (definiert durch ihre Tonhöhen) und deren Dauer.

1. Einbindung erforderlicher Bibliotheken und globale Variablen

```
#include "pitches.h"  // This library provides the frequency values for musical␣
↪notes.

const int buttonPin = 7;  // Button connected to digital pin 7
const int buzzerPin = 8;  // Buzzer connected to digital pin 8

// Arrays to define the melody and the corresponding note durations
int melody[] = {...};
int noteDurations[] = {...};
```

   In diesem Abschnitt werden die für Musiknoten erforderliche Bibliothek eingebunden und die Pins für unsere Komponenten festgelegt. Zusätzlich werden die Melodie und ihre Dauern in Arrays definiert.

2. Initialisierung des Knopfs und Start der seriellen Kommunikation

```
void setup() {
  Serial.begin(9600);           // Start serial communication at 9600 baud rate
  pinMode(buttonPin, INPUT);    // Set the button pin as an input
}
```

   In der `setup()` Funktion starten wir die serielle Kommunikation und konfigurieren den buttonPin als Eingang.

3. Überwachung des Knopfdrucks zur Wiedergabe der Melodie

```
void loop() {
  int buttonState = digitalRead(buttonPin);  // Read the state of the button

  if (buttonState == LOW) {                  // Check if the button is pressed
    Serial.println("Button pressed");        // Send a message to serial monitor
    buzzer();                                // Play the buzzer melody
  }
}
```

   In dieser Schleife überprüfen wir kontinuierlich den Zustand des Knopfs. Bei Betätigung wird eine Nachricht an den seriellen Monitor gesendet und die Funktion `buzzer()` aufgerufen, die die Melodie abspielt.

4. Die Funktion `buzzer()` zur Wiedergabe der Melodie

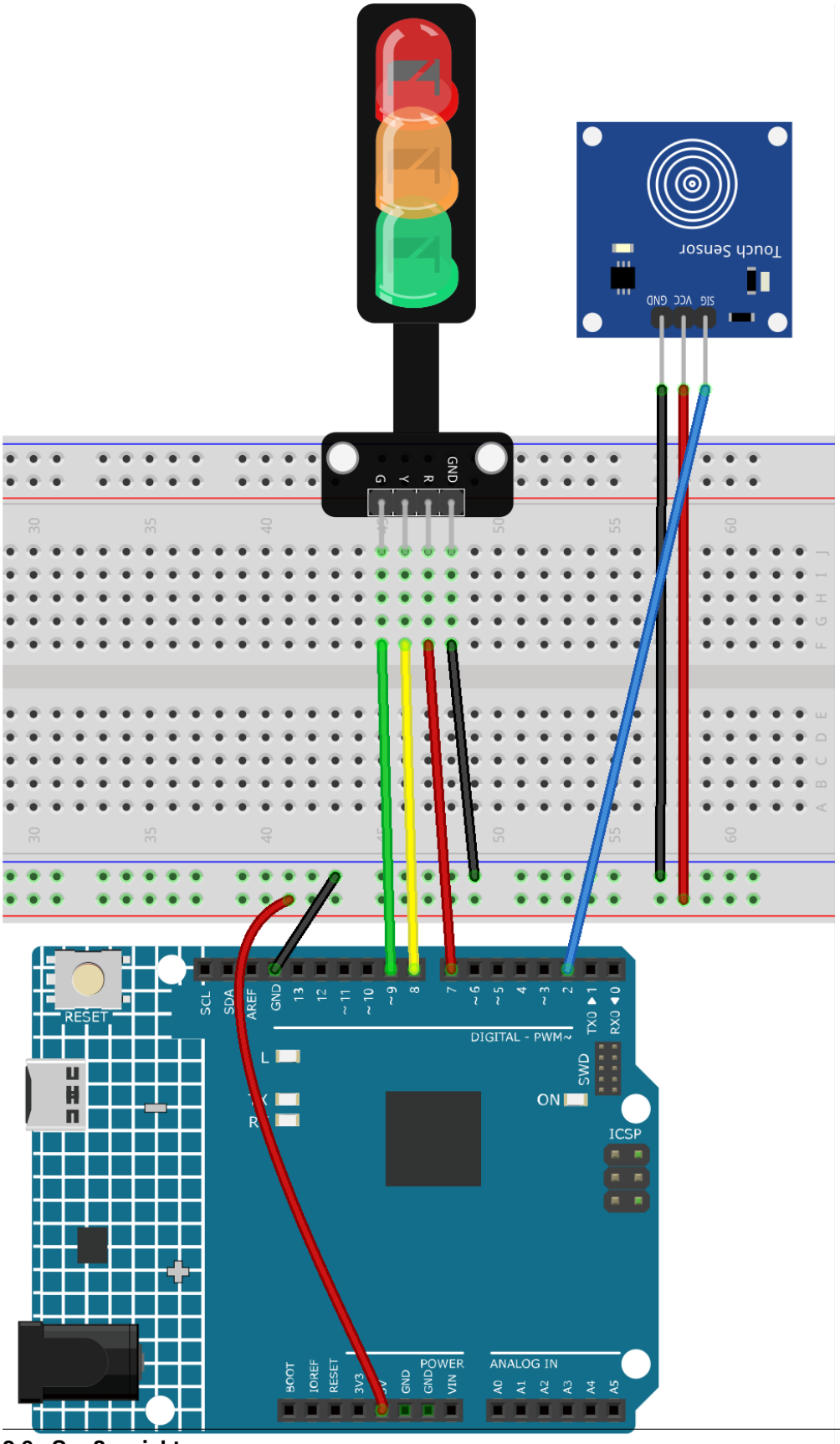
```
void buzzer() {
  int size = sizeof(noteDurations) / sizeof(int);   // Calculate the number of notes

  for (int thisNote = 0; thisNote < size; thisNote++) {
    int noteDuration = 1000 / noteDurations[thisNote];   // Calculate note's play␣
↪duration
    tone(buzzerPin, melody[thisNote], noteDuration);    // Play the note on the␣
↪buzzer

    int pauseBetweenNotes = noteDuration * 1.30;        // Calculate pause between␣
↪notes
    delay(pauseBetweenNotes);                           // Introduce the pause
    noTone(buzzerPin);                                  // Stop playing the note
  }
}
```

In der Funktion `buzzer()` werden die Noten der Melodie nacheinander abgespielt. Die `tone()` Funktion erzeugt einen Ton auf dem Summer für eine festgelegte Dauer. Nach dem Abspielen jeder Note folgt eine kurze Pause, bevor die nächste Note abgespielt wird.

2.6.8 Berührungsaktivierte Ampel

Das Projekt beinhaltet die Erstellung einer einfachen Ampelsteuerung mit einem Berührungssensor und einem Ampel-LED-Modul. Wird der Sensor berührt, durchläuft das LED-Modul folgende Sequenz: Rot -> Gelb -> Grün.

1. Schaltungsaufbau



- *Arduino UNO R4 Minima-Platine*
- *Berührungs-Sensormodul*
- *Ampel-Modul*

2. Code

1. Öffnen Sie die Datei `08-Touch_toggle_light.ino` im Verzeichnis `ultimate-sensor-kit\fun_project\08-Touch_toggle_light`, oder kopieren Sie diesen Code in die **Arduino IDE**.

3. Codeerklärung

Dieses Projekt basiert auf einem einfachen Prinzip: Wird eine Berührung am Sensor erkannt, leuchtet die nächste LED in der Sequenz (Rot -> Gelb -> Grün) auf. Der aktuelle Status der LEDs wird durch die Variable `currentLED` verwaltet.

1. Definition der Pins und Anfangswerte

```
const int touchSensorPin = 2;  // touch sensor pin
const int rledPin = 9;         // red LED pin
const int yledPin = 8;         // yellow LED pin
const int gledPin = 7;         // green LED pin
int lastTouchState;     // the previous state of touch sensor
int currentTouchState;  // the current state of touch sensor
int currentLED = 0;     // current LED 0->Red, 1->Yellow, 2->Green
```

   These lines define the pins that we connect the components to on the Arduino board and initialize the states for touch and LEDs.

2. setup() Funktion

```
void setup() {
  Serial.begin(9600);              // initialize serial
  pinMode(touchSensorPin, INPUT);  // configure touch sensor pin as input
  // set LED pins as outputs
  pinMode(rledPin, OUTPUT);
  pinMode(yledPin, OUTPUT);
  pinMode(gledPin, OUTPUT);
  currentTouchState = digitalRead(touchSensorPin);
}
```

   Diese Funktion wird einmal ausgeführt, wenn der Arduino eingeschaltet oder zurückgesetzt wird. Hier wird der Berührungssensor als Eingang und die LEDs als Ausgänge konfiguriert. Die serielle Kommunikation wird für die Fehlerdiagnose gestartet und der Anfangszustand des Berührungssensors wird gelesen.

3. loop() Funktion

```
void loop() {
  lastTouchState = currentTouchState;               // save the last state
  currentTouchState = digitalRead(touchSensorPin);  // read new state
  if (lastTouchState == LOW && currentTouchState == HIGH) {
    Serial.println("The sensor is touched");
    turnAllLEDsOff();  // Turn off all LEDs
    // switch on the next LED in sequence
    switch (currentLED) {
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
      case 0:
        digitalWrite(rledPin, HIGH);
        currentLED = 1;
        break;
      case 1:
        digitalWrite(yledPin, HIGH);
        currentLED = 2;
        break;
      case 2:
        digitalWrite(gledPin, HIGH);
        currentLED = 0;
        break;
    }
  }
}
```

In der Hauptschleife wird der aktuelle Zustand des Berührungssensors gelesen und mit dem vorherigen verglichen. Wird eine Berührung erkannt (Übergang von LOW zu HIGH), werden alle LEDs ausgeschaltet und die nächste in der Sequenz eingeschaltet.

4. Funktion zum Ausschalten der LEDs

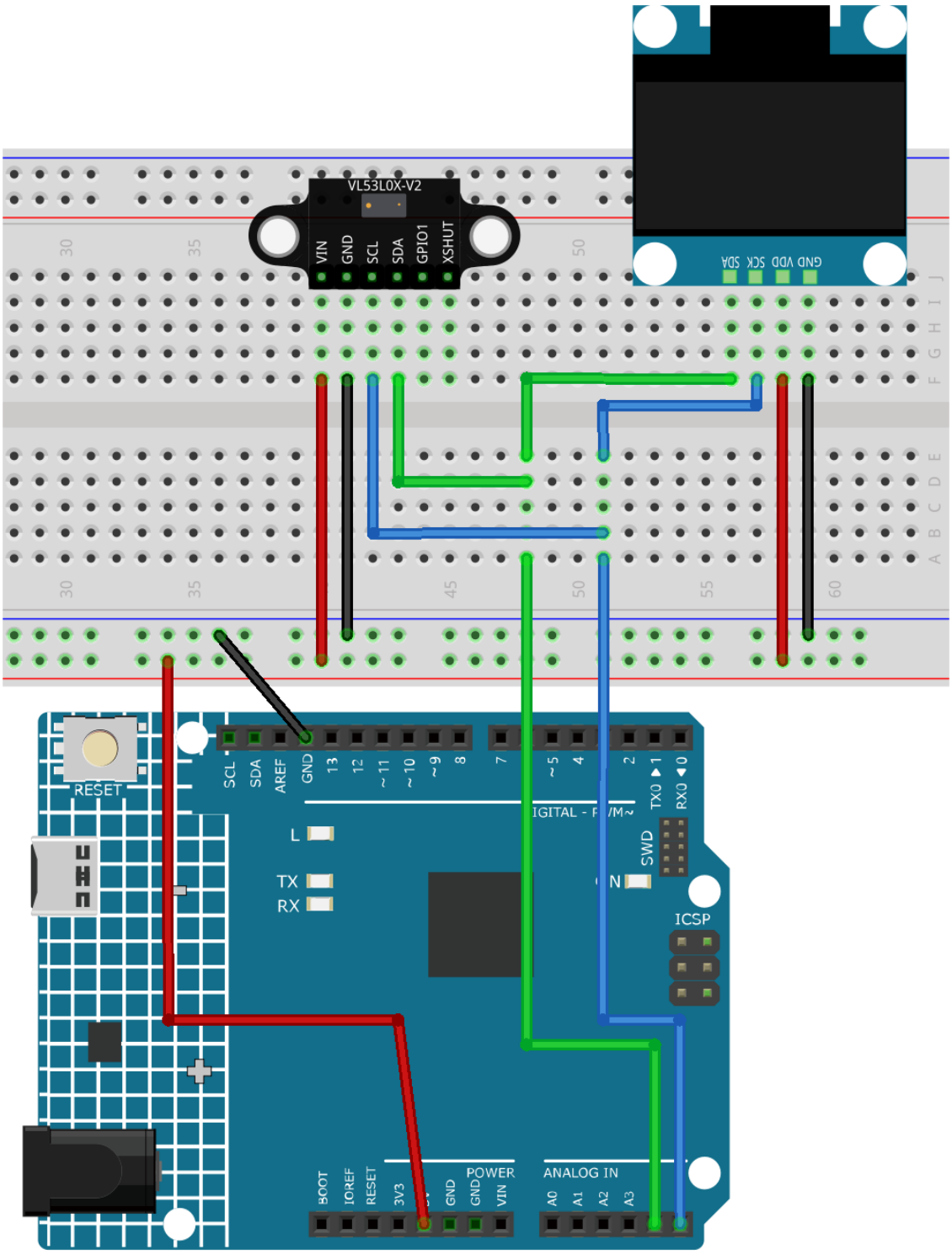
```
void turnAllLEDsOff() {
  digitalWrite(rledPin, LOW);
  digitalWrite(yledPin, LOW);
  digitalWrite(gledPin, LOW);
}
```

Diese Funktion schaltet alle LEDs aus, indem ihre Pins auf LOW gesetzt werden.

2.6.9 ToF Entfernungsmonitor

Dieses Projekt ist darauf ausgelegt, die Entfernung zu einem Objekt mit Hilfe des VL53L0X Time-of-Flight (ToF) Micro-LIDAR Entfernungssensors zu messen und anzuzeigen. Die gemessene Entfernung in Millimetern wird auf einem OLED-Display dargestellt und zusätzlich auf dem seriellen Monitor ausgegeben. Der VL53L0X kann einen Bereich von etwa 50mm bis 1200mm abdecken.

1. Schaltungsaufbau



- *Arduino UNO R4 Minima-Platine*
- *Time-of-Flight Mikro-LIDAR-Entfernungssensor (VL53L0X)*
- *OLED-Display-Modul*

2. Programmcode

1. Öffnen Sie die Datei `09-ToF_distance_monitor.ino` im Verzeichnis `ultimate-sensor-kit\fun_project\09-ToF_distance_monitor` oder kopieren Sie diesen Code in die **Arduino IDE**.

Bemerkung: Verwenden Sie den Arduino-Bibliotheksmanager und suchen Sie nach **„Adafruit_VL53L0X“** und installieren Sie diese.

---

3. Code-Erläuterung

Das Projekt nutzt den VL53L0X Time-of-Flight-Sensor, um Entfernungen durch Messung der Zeit, die das Licht benötigt, um zu einem Objekt und zurück zum Sensor zu gelangen, zu ermitteln. Das OLED-Display zeigt dann die gemessene Entfernung in Millimetern an. Über die serielle Kommunikation werden ebenfalls die Messwerte ausgegeben, was das Monitoring und Debugging erleichtert. Sowohl das OLED-Display als auch der VL53L0X-Sensor kommunizieren mit dem Arduino über das I2C-Protokoll.

1. Einbinden der notwendigen Bibliotheken und Initialisieren der Komponenten

---

Bemerkung: Verwenden Sie den Arduino-Bibliotheksmanager und suchen Sie nach **„Adafruit_VL53L0X“** und installieren Sie diese.

```
#include <Wire.h>
#include "Adafruit_VL53L0X.h"
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Initialize the OLED display module with a resolution of 128x64
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire, -1);

// Initialize the VL53L0X distance sensor
Adafruit_VL53L0X lox = Adafruit_VL53L0X();
```

- Die erforderlichen Bibliotheken für die Handhabung der I2C-Kommunikation, des Abstandssensors, des SPI-Protokolls und des OLED-Displays sind enthalten.
- Das OLED-Display und der VL53L0X-Abstandssensor werden initialisiert.

2. Initialisierung der seriellen Kommunikation und Vorbereitung des Displays sowie des VL53L0X-Entfernungssensors.

```
void setup() {
  Serial.begin(9600);

  // Start the OLED display with I2C address 0x3C
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.display();
  delay(1000);

  // Begin I2C communication
  Wire.begin();
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  // Start the VL53L0X distance sensor, halt if initialization fails
  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while (1)
      ;
  }

  // Set OLED display text size and color
  display.setTextSize(3);
  display.setTextColor(WHITE);
}
```

- Starte die serielle Kommunikation mit einer Baudrate von 9600.
- Initialisiere das OLED-Display mit seiner I2C-Adresse.
- Beginne die I2C-Kommunikation.
- Überprüfe, ob der VL53L0X-Distanzsensor ordnungsgemäß initialisiert ist. Wenn nicht, wird eine Fehlermeldung angezeigt und der Arduino tritt in eine Endlosschleife ein.
- Setze Textgröße und Farbe für das OLED-Display.

3. Hauptprogrammschleife zur Entfernungsvermessung und Anzeige des Ergebnisses

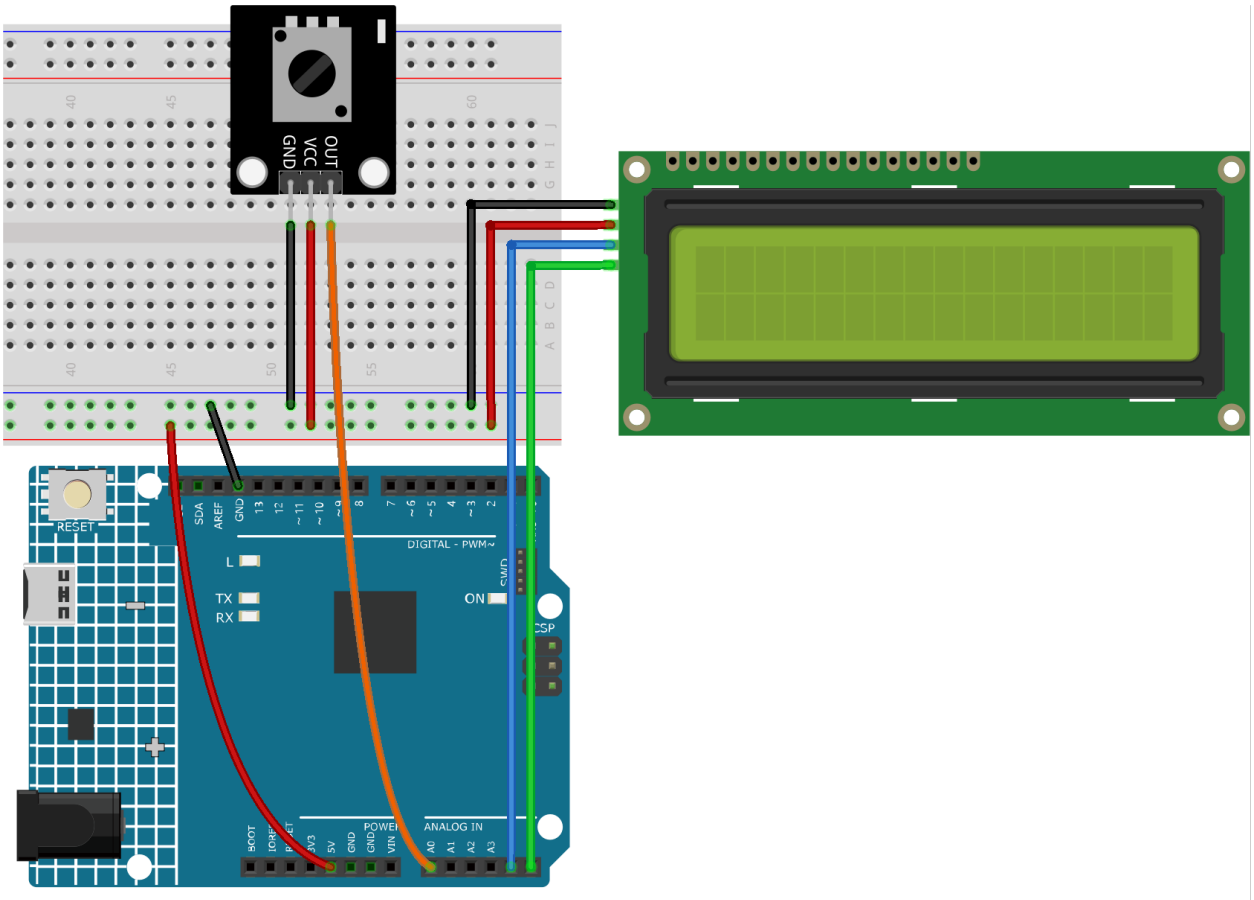
```
void loop() {
  VL53L0X_RangingMeasurementData_t measure;

  lox.rangingTest(&measure, false);  // pass in 'true' to get debug data printout

  // If there are no phase failures, display the measured distance
  if (measure.RangeStatus != 4) {
    display.clearDisplay();
    display.setCursor(12, 22);
    display.print(measure.RangeMilliMeter);
    display.print("mm");
    display.display();
    Serial.println();
    delay(50);
  } else {
    display.display();
    display.clearDisplay();
    return;
  }
}
```

- Erstelle eine Variable, um die Messdaten zu speichern.
- Führe eine Messung mit dem VL53L0X-Sensor durch.
- Überprüfe, ob die Messung gültig ist (d.h. keine Phasenausfälle vorliegen).
- Wenn die Messung gültig ist, lösche den OLED-Bildschirm, setze die Cursorposition und zeige den gemessenen Abstand an.
- Andernfalls aktualisiere das Display und lösche es für die nächste Lesung.

2.6.10 Potentiometer-Skalenwert

Dieses Projekt wurde entwickelt, um den Wert eines Potentiometers auszulesen und diesen Wert auf einem LCD 1620 mit einer I2C-Schnittstelle anzuzeigen. Der Wert wird auch an den seriellen Monitor gesendet, um ihn in Echtzeit anzusehen. Ein besonderes Merkmal dieses Projekts ist die visuelle Darstellung des Potentiometerwerts auf dem LCD, die einen Balken anzeigt, dessen Länge dem Wert entspricht.

1. Schalten Sie die Schaltung



- *Arduino UNO R4 Minima-Platine*
- *Potentiometer-Modul*
- *I2C LCD 1602*

2. Code

1. Öffnen Sie die Datei `10-Potentiometer_scale_value.ino` unter dem Pfad `ultimate-sensor-kit\fun_project\10-Potentiometer_scale_value`, oder kopieren Sie diesen Code in die **Arduino IDE**.

3. Code-Erklärung

Das Projekt funktioniert, indem es kontinuierlich den Wert von einem angeschlossenen Potentiometer liest. Dieser Wert wird dann auf eine kleinere Skala (0-16) abgebildet und sowohl numerisch als auch visuell auf dem LCD dargestellt. Durch Überprüfung des Unterschieds zwischen aufeinanderfolgenden Messwerten stellt der Code sicher, dass nur signifikante Änderungen auf dem Display widergespiegelt werden, wodurch das Flackern reduziert wird. Dies hilft, unerwünschte visuelle Effekte durch häufiges Aktualisieren des LCD-Bildschirms zu vermeiden.

1. **Bibliothek einbinden und initialisieren**:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

   Hier werden die benötigten Bibliotheken (`Wire` für die I2C-Kommunikation und `LiquidCrystal_I2C` für das LCD) eingebunden. Ein LCD-Objekt wird mit der I2C-Adresse `0x27` erstellt und ist definiert mit `16` Spalten und `2` Zeilen.

2. **Variablendeklaration**:

```
int lastRead = 0;     // Previous potentiometer value
int currentRead = 0;  // Current potentiometer value
```

   `lastRead` speichert den zuletzt gelesenen Potentiometerwert. `currentRead` speichert den aktuellen Wert des Potentiometers.

3. **setup() Funktion**:

```
void setup() {
  lcd.init();          // Initialize the LCD
  lcd.backlight();     // Turn on the LCD backlight
  Serial.begin(9600);  // Start serial communication at 9600 baud rate
}
```

   Das LCD wird initialisiert, seine Hintergrundbeleuchtung eingeschaltet und die serielle Kommunikation mit einer Baudrate von `9600` gestartet.

4. **Hauptloop**:

```
void loop() {
  int currentRead = analogRead(A0);
  int barLength = map(currentRead, 0, 1023, 0, 16);
  if (abs(lastRead - currentRead) > 2) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Wert:");
    lcd.setCursor(7, 0);
    lcd.print(currentRead);
    Serial.println(currentRead);
    for (int i = 0; i < barLength; i++) {
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
      lcd.setCursor(i, 1);
      lcd.print(char(255));
    }
  }
  lastRead = currentRead;
  delay(200);
}
```

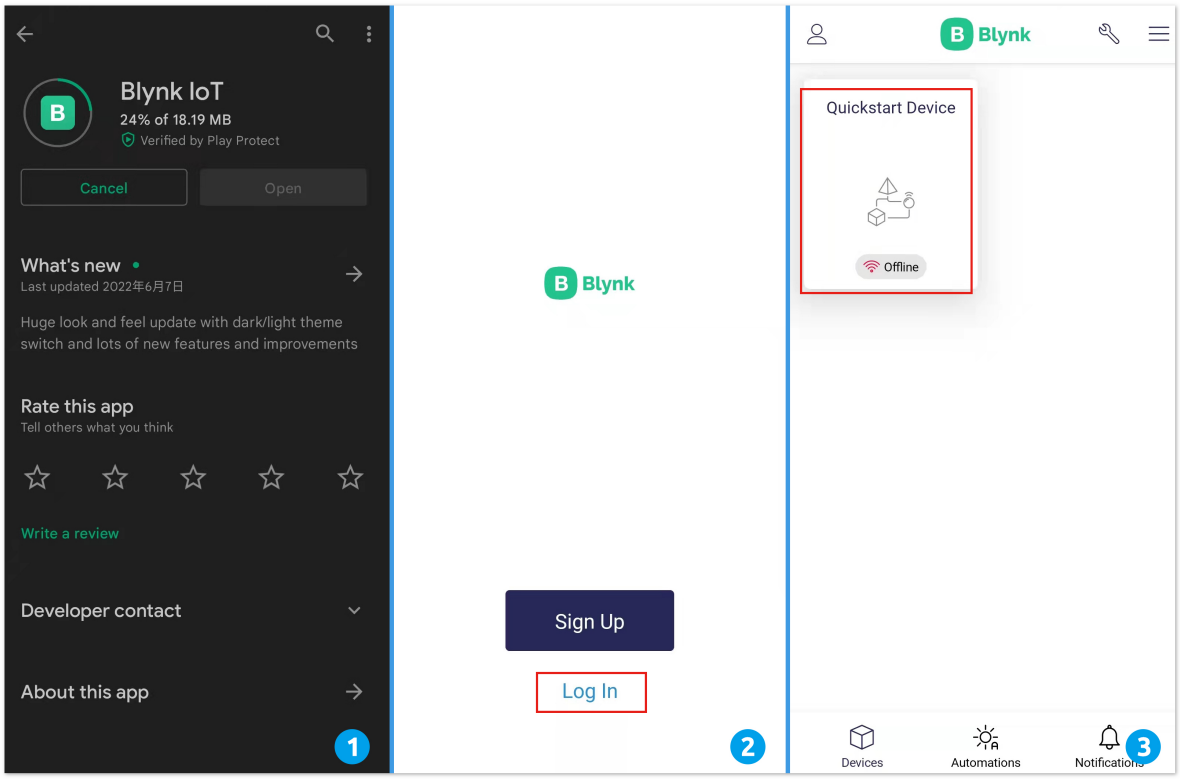
- Der Potentiometerwert wird gelesen und auf eine Balkenlänge (0-16) abgebildet.
- Wenn der Unterschied zwischen dem letzten und dem aktuellen Wert größer als 2 ist, wird das LCD aktualisiert.
- Der Wert wird in der ersten Zeile und ein Balken (basierend auf dem abgebildeten Wert) in der zweiten Zeile angezeigt.
- Der Wert wird auch an den seriellen Monitor gesendet.
- Vor der nächsten Iteration wird `lastRead` aktualisiert, und es wird eine Verzögerung von `200ms` für die Stabilität eingeführt.

2.7 FAQ

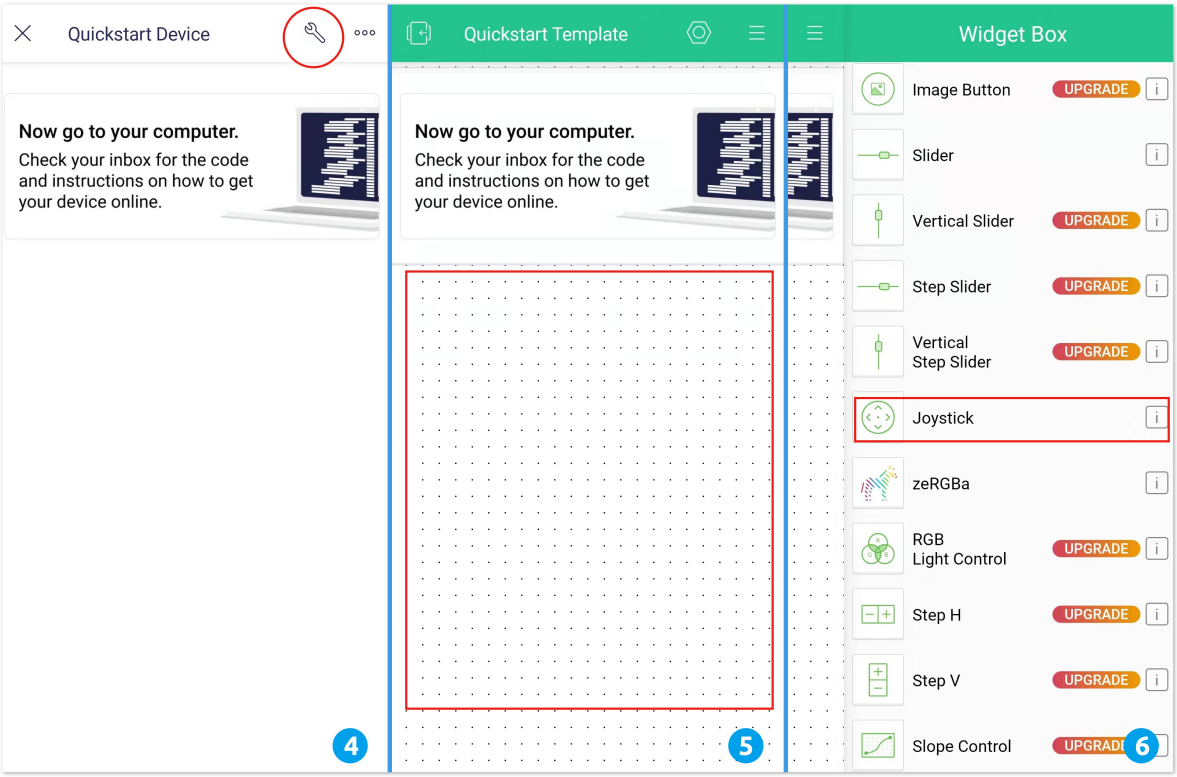
2.7.1 Wie verwendet man Blynk auf dem Mobilgerät?

Bemerkung: Da Datenströme nur im Webinterface von Blynk erstellt werden können, müssen Sie sich auf verschiedene Projekte beziehen, um Datenströme im Web zu erstellen. Befolgen Sie dann das untenstehende Tutorial, um Widgets in der Blynk-App auf Ihrem Mobilgerät zu erstellen.

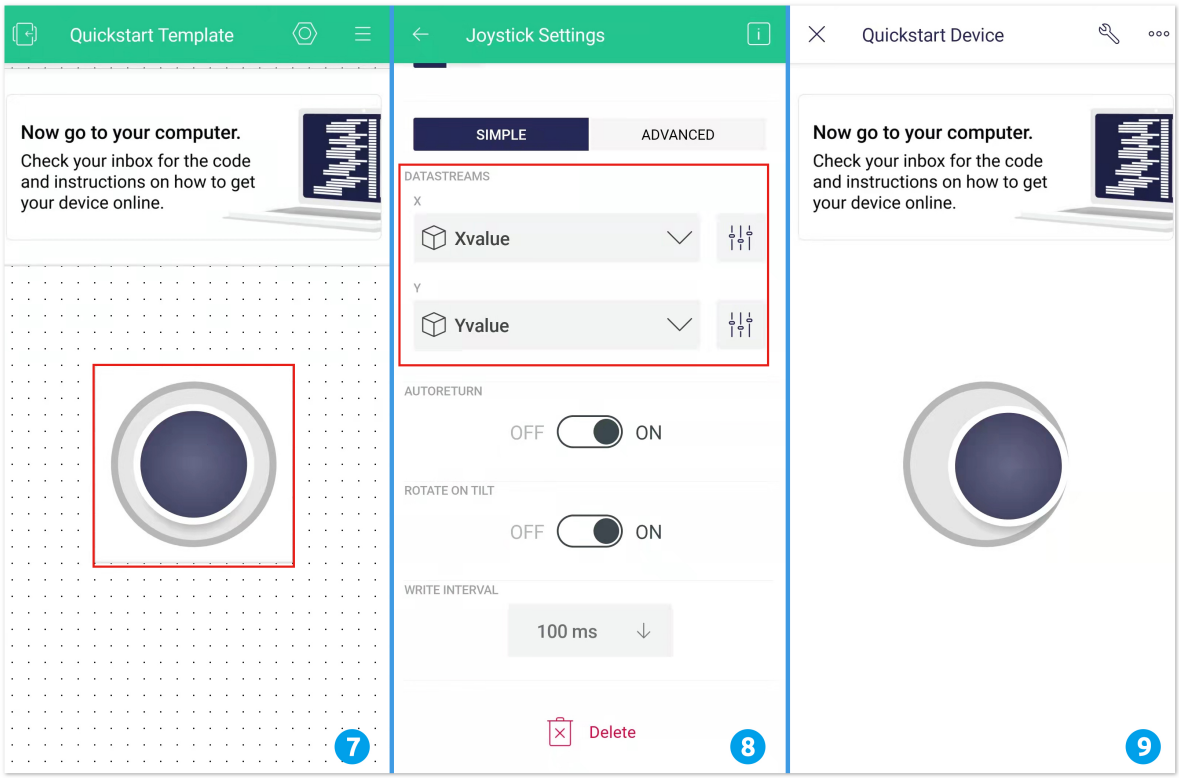
1. Öffnen Sie den Google Play Store oder den APP Store auf Ihrem Mobilgerät und suchen Sie nach „Blynk IoT“ (nicht Blynk(Legacy)), um die App herunterzuladen.
2. Melden Sie sich nach dem Start der App an. Dieses Konto sollte identisch mit dem Konto sein, das Sie im Web-Client verwenden.
3. Navigieren Sie zum **Dashboard** (falls Sie noch keines haben, erstellen Sie eines). Sie werden feststellen, dass die **Dashboards** für Mobilgeräte und Web unabhängig voneinander sind.



4. Klicken Sie auf das **Bearbeiten**-Symbol.
5. Tippen Sie auf einen leeren Bereich.
6. Wählen Sie das gleiche Widget aus, das Sie auch auf der Webseite verwendet haben, zum Beispiel ein **Joystick**-Widget.



7. Nun erscheint das **Joystick**-Widget im leeren Bereich. Klicken Sie darauf.
8. Die Einstellungen für den **Joystick** öffnen sich. Wählen Sie die Datenströme **X-Wert** und **Y-Wert** aus, die Sie gerade auf der Webseite festgelegt haben. Beachten Sie, dass jedes Widget einem anderen Datenstrom in jedem Projekt entspricht.
9. Kehren Sie zur **Dashboard**-Seite zurück. Jetzt können Sie den **Joystick** nach Belieben bedienen.

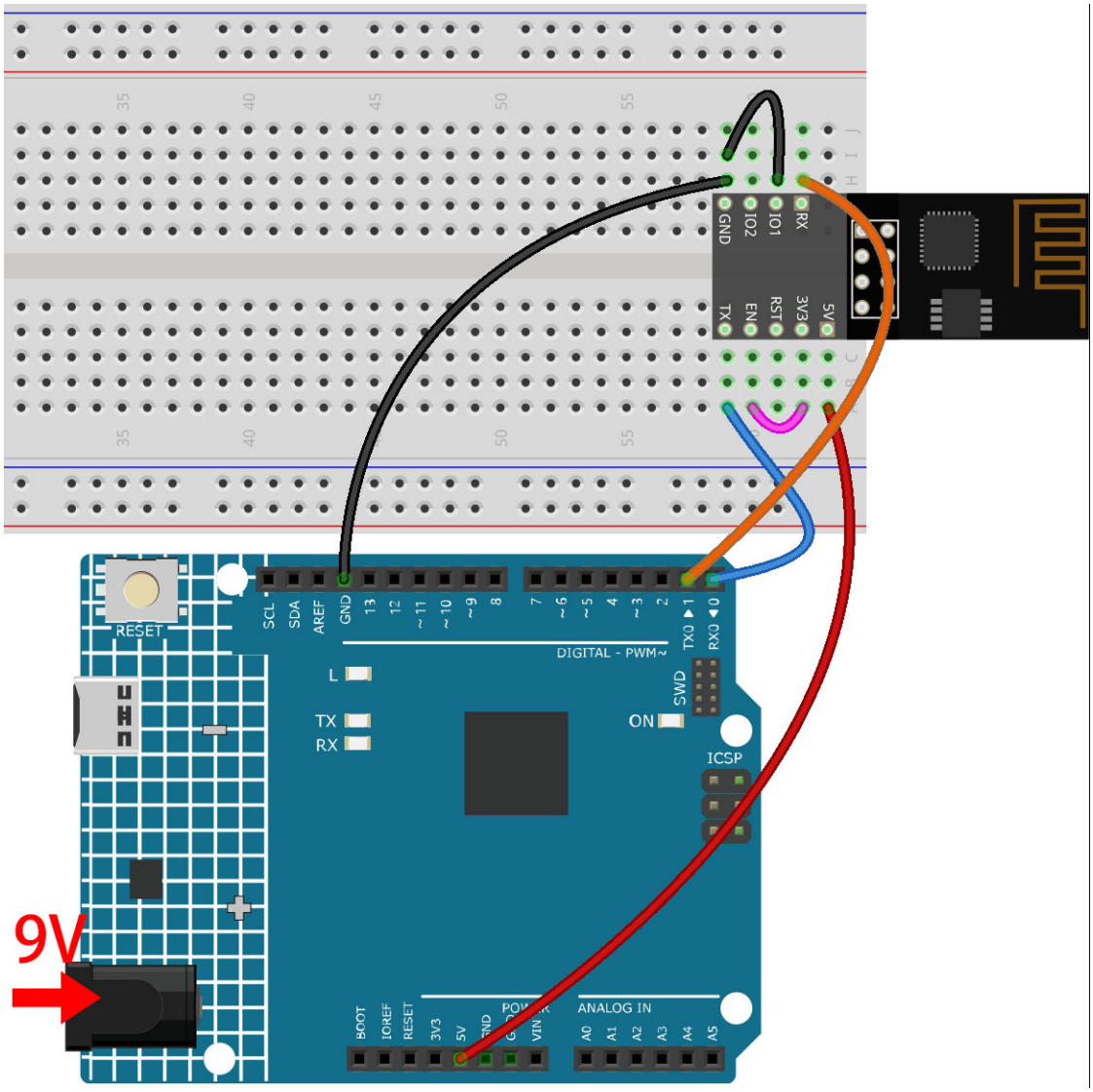


2.7.2 Wie man die Firmware für das ESP8266-Modul erneut aufspielt

Firmware mit R4 erneut aufspielen

1. Schaltkreis aufbauen

Verbinden Sie das ESP8266 mit dem Arduino UNO R4 Board.



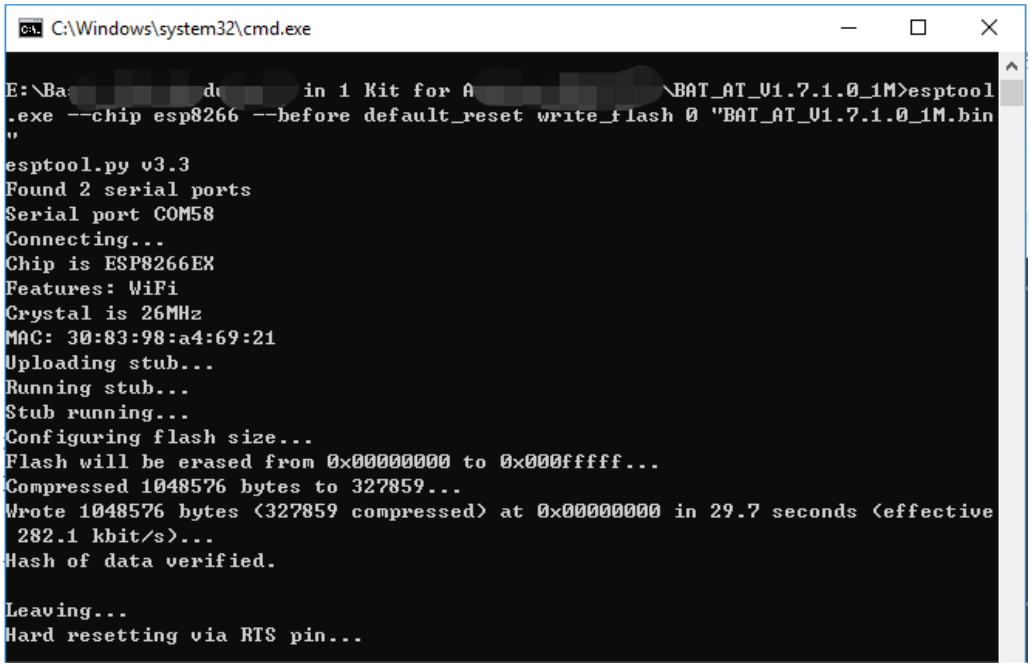
2. Folgenden Code auf R4 hochladen

```
void setup() {
    Serial.begin(115200);
    Serial1.begin(115200);
}

void loop() {
    if (Serial.available()) {      // If anything comes in Serial (USB),
        Serial1.write(Serial.read());   // read it and send it out Serial1 (pins 0 & 1)
    }
        if (Serial1.available()) {     // If anything comes in Serial1 (pins 0 & 1)
        Serial.write(Serial1.read());   // read it and send it out Serial (USB)
    }
}
```

3. Firmware aufspielen

- Wenn Sie **Windows** verwenden, folgen Sie den untenstehenden Schritten, um die Firmware aufzuspielen.

1. Firmware und Brenn-Tool herunterladen.
   - `ESP8266 Firmware`
2. Nach dem Entpacken erscheinen vier Dateien.
   - `BAT_AT_V1.7.1.0_1M.bin`: Die auf das ESP8266-Modul zu brennende Firmware.
   - `esptool.exe`: Kommandozeilen-Utility für Windows.
   - `install_r3.bat`: Befehlssatz für Windows, Doppelklick startet alle darin enthaltenen Befehle.
   - `install_r4.bat`: Wie `install_r3.bat`, jedoch speziell für das UNO R4 Board.
3. Doppelklicken Sie auf `install_r4.bat`, um den Brennvorgang zu starten. Wenn die folgende Aufforderung erscheint, wurde die Firmware erfolgreich installiert.



Bemerkung: Sollte der Brennvorgang scheitern, überprüfen Sie die folgenden Punkte:

- Setzen Sie das ESP8266-Modul zurück, indem Sie den RST-Anschluss am ESP8266-Adapter mit GND verbinden und wieder trennen.
- Überprüfen Sie die Verkabelung.
- Stellen Sie sicher, dass Ihr Computer das Board erkannt hat und der Port frei ist.
- Öffnen Sie die install.bat-Datei erneut.

- Zum Aufspielen der Firmware auf einem **Mac OS**-System folgen Sie diesen Schritten:

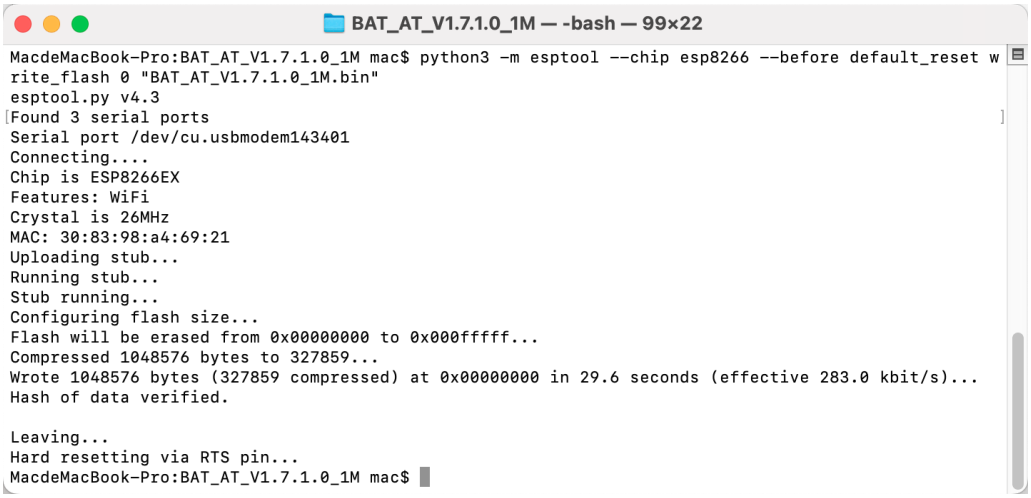
1. Installieren Sie mit den folgenden Befehlen Esptool. Esptool ist ein Python-basiertes, Open-Source-, plattformunabhängiges Hilfsprogramm zur Kommunikation mit dem ROM-Bootloader in Espressif-Chips.

```
python3 -m pip install --upgrade pip
python3 -m pip install esptool
```

2. Falls Esptool korrekt installiert ist, gibt ein Befehl wie `python3 -m esptool` eine Meldung wie [usage: esptool] aus.
3. Firmware herunterladen.
   - `ESP8266 Firmware`
4. Nach dem Entpacken erscheinen vier Dateien.
   - `BAT_AT_V1.7.1.0_1M.bin`: Die auf das ESP8266-Modul zu brennende Firmware.
   - `esptool.exe`: Kommandozeilen-Utility für Windows.
   - `install_r3.bat`: Befehlssatz für Windows.
   - `install_r4.bat`: Wie `install_r3.bat`, jedoch speziell für das UNO R4 Board.
5. Öffnen Sie ein Terminal, navigieren Sie mit `cd` in den gerade heruntergeladenen Firmware-Ordner und führen Sie die folgenden Befehle aus, um die vorhandene Firmware zu löschen und die neue Firmware aufzuspielen.

```
python3 -m esptool --chip esp8266 --before no_reset_no_sync erase_flash
python3 -m esptool --chip esp8266 --before no_reset_no_sync write_flash␣
↪0 "BAT_AT_V1.7.1.0_1M.bin"
```

6. Wenn die folgende Aufforderung erscheint, wurde die Firmware erfolgreich installiert.



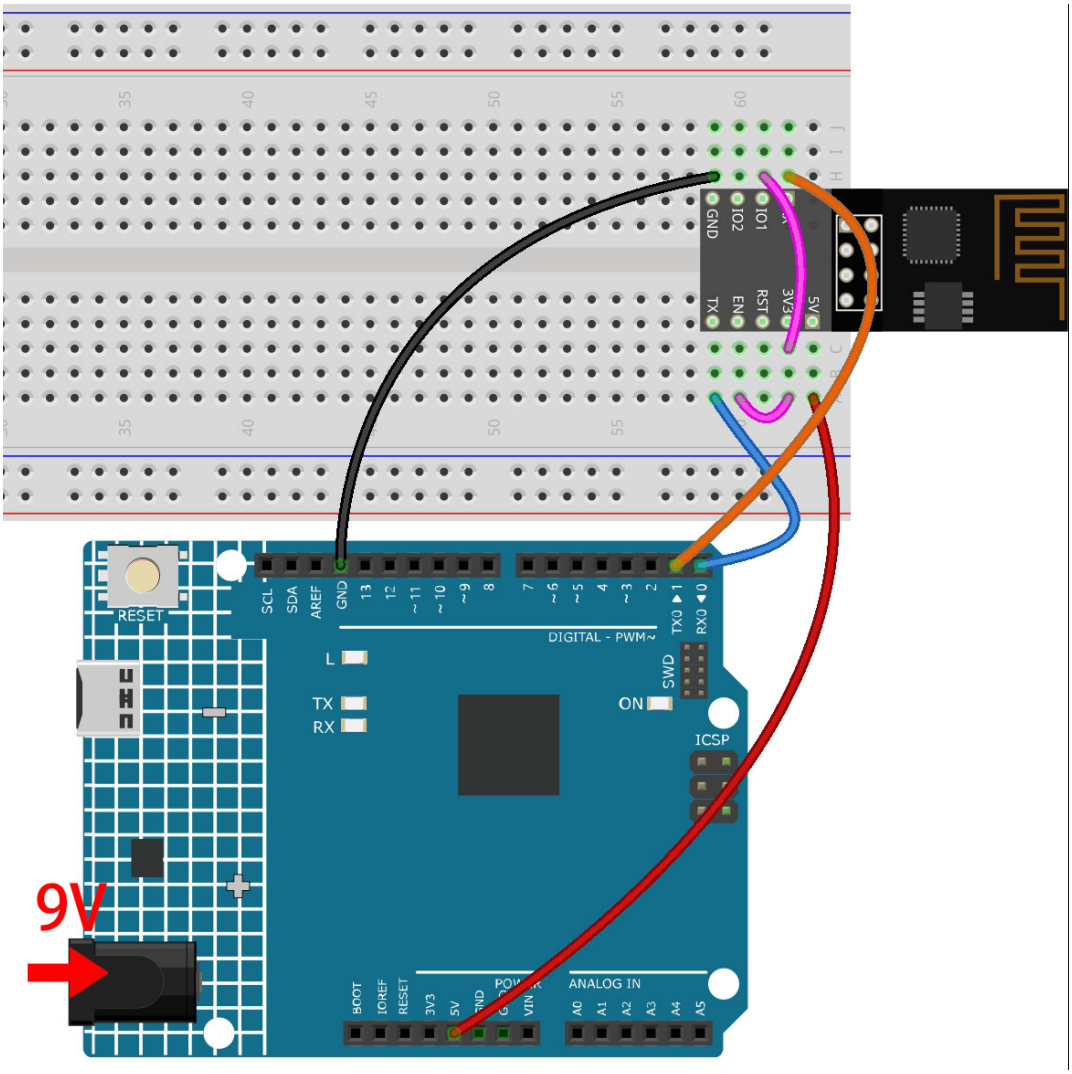
Bemerkung: Sollte der Brennvorgang scheitern, überprüfen Sie die folgenden Punkte:

- Setzen Sie das ESP8266-Modul zurück, indem Sie den RST-Anschluss am ESP8266-Adapter mit GND verbinden und wieder trennen.
- Überprüfen Sie die Verkabelung.
- Stellen Sie sicher, dass Ihr Computer das Board erkannt hat und der Port frei ist.
- Öffnen Sie die install.bat-Datei erneut.

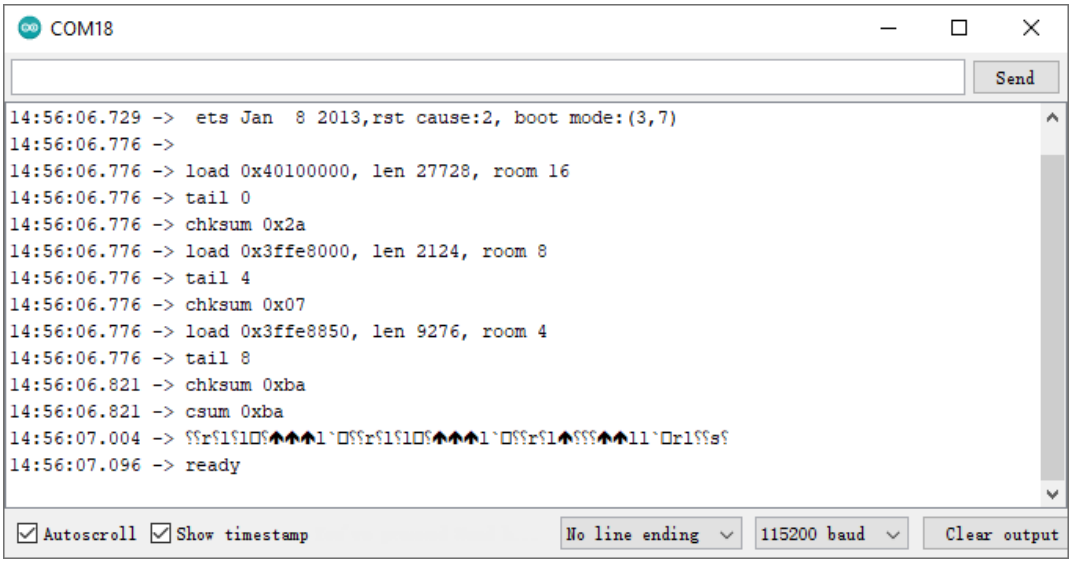
---

4. Test

1. Ergänzend zur ursprünglichen Verkabelung verbinden Sie IO1 mit 3V3.



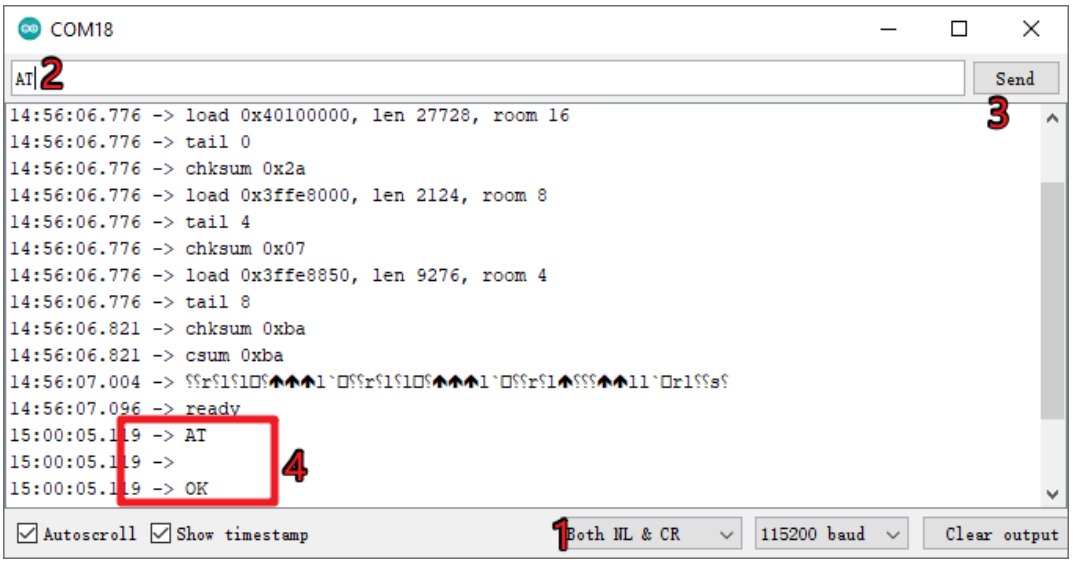
2. Wenn Sie das Lupensymbol (Serial Monitor) in der oberen rechten Ecke anklicken und die Baudrate auf **115200** einstellen, sollten Sie Informationen über das ESP8266-Modul sehen.



Bemerkung:

- Erscheint keine Meldung `ready`, können Sie versuchen, das ESP8266-Modul zurückzusetzen (RST mit GND verbinden) und den Serial Monitor erneut zu öffnen.

3. Klicken Sie auf **NEWLINE DROPDOWN BOX**, wählen Sie in den Dropdown-Optionen `both NL & CR`, geben Sie `AT` ein. Wenn OK zurückkommt, besteht eine erfolgreiche Verbindung zwischen dem ESP8266 und dem R3-Board.

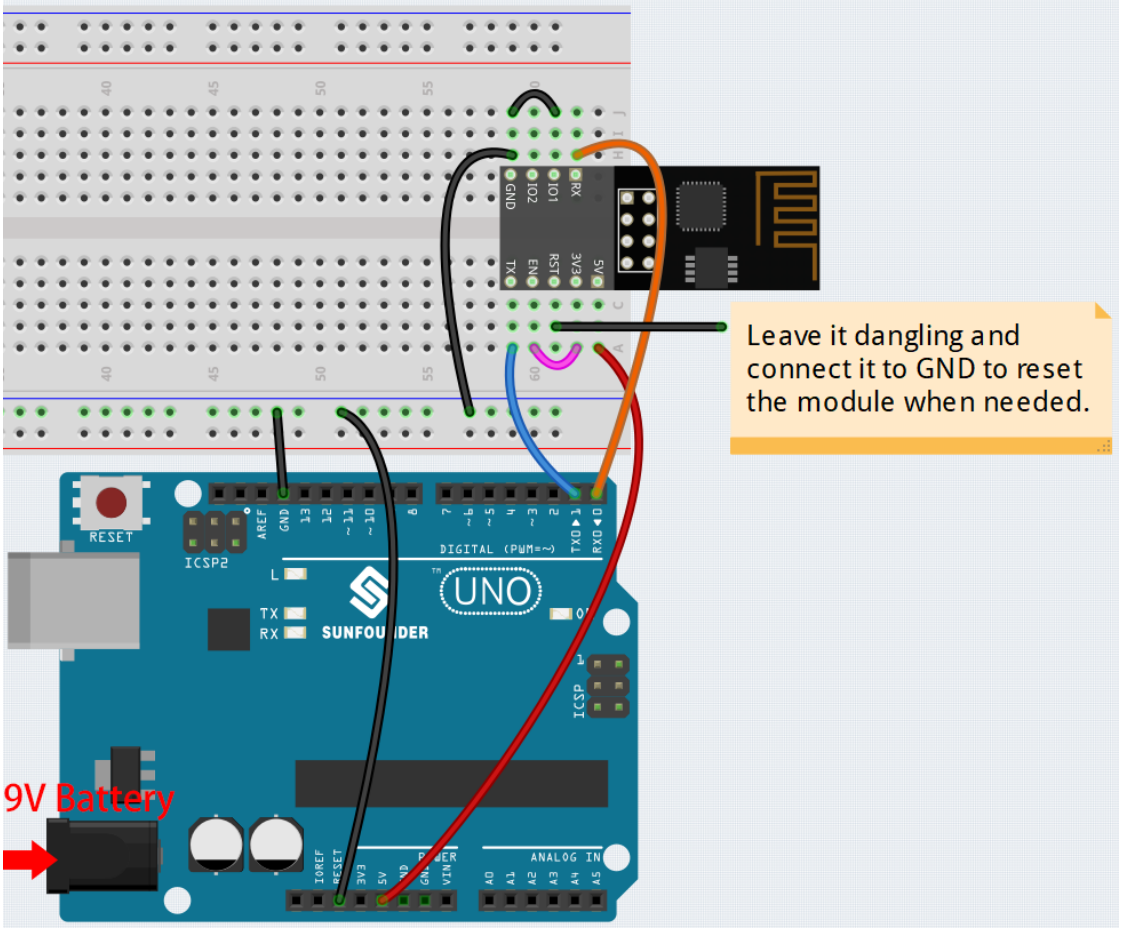


Jetzt können Sie fortfahren und mit *1.1 Konfiguration des ESP8266* den Arbeitsmodus und die Baudrate des ESP8266-Moduls einstellen.

Firmware mit R3 neu aufspielen

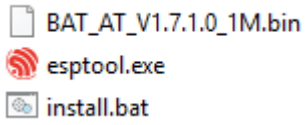
1. Schaltung aufbauen

Verbinden Sie das ESP8266-Modul mit dem SunFounder R3 Board.

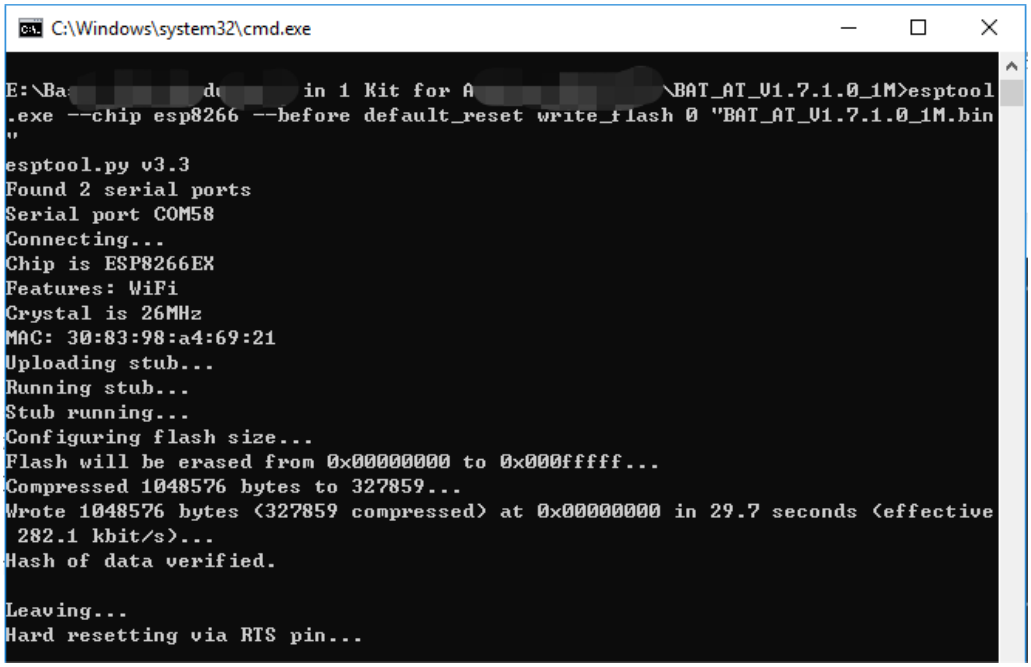


2. Firmware brennen

- Wenn Sie **Windows** verwenden, gehen Sie wie folgt vor:
  1. Firmware und Brenn-Tool herunterladen.
     - `ESP8266 Firmware`
  2. Nach dem Entpacken stehen vier Dateien zur Verfügung.

     - `BAT_AT_V1.7.1.0_1M.bin`: Die auf das ESP8266-Modul zu brennende Firmware.
     - `esptool.exe`: Ein Kommandozeilen-Programm für Windows.
     - `install_r3.bat`: Befehlsdatei für Windows. Ein Doppelklick führt alle enthaltenen Befehle aus.
     - `install_r4.bat`: Gleiche Funktion wie `install_r3.bat`, jedoch für das UNO R4 Board.
  3. Doppelklicken Sie auf `install_r3.bat`, um den Brennvorgang zu starten. Erscheint folgende Meldung, war der Vorgang erfolgreich.



Bemerkung: Sollte der Brennvorgang fehlschlagen, überprüfen Sie bitte:

- Setzen Sie das ESP8266-Modul zurück (RST am ESP8266-Adapter mit GND verbinden und wieder trennen).
- Überprüfen Sie die Verkabelung.
- Stellen Sie sicher, dass Ihr Computer das Board erkannt hat und der Port frei ist.
- Öffnen Sie die Datei install.bat erneut.

- Für **Mac OS**-Benutzer gilt:

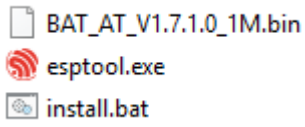
  1. Mit den folgenden Befehlen installieren Sie Esptool. Esptool ist ein plattformübergreifendes Python-Programm zur Kommunikation mit dem ROM-Bootloader von Espressif-Chips.

     ```
     python3 -m pip install --upgrade pip
     python3 -m pip install esptool
     ```

  2. Wenn Esptool korrekt installiert ist, sollte der Befehl `python3 -m esptool` eine Meldung wie [usage: esptool] ausgeben.
  3. Firmware herunterladen.

     - `ESP8266 Firmware`

  4. Nach dem Entpacken stehen drei Dateien zur Verfügung.

     - `BAT_AT_V1.7.1.0_1M.bin`: Die auf das ESP8266-Modul zu brennende Firmware.
     - `esptool.exe`: Ein Kommandozeilen-Programm für Windows.

- `install_r3.bat`: Befehlsdatei für Windows.
- `install_r4.bat`: Gleiche Funktion wie `install_r3.bat`, jedoch für das UNO R4 Board.

5. Öffnen Sie ein Terminal und verwenden Sie den Befehl `cd`, um in den Firmware-Ordner zu wechseln, den Sie gerade heruntergeladen haben. Führen Sie dann den folgenden Befehl aus, um die vorhandene Firmware zu löschen und die neue Firmware neu zu brennen.

```
python3 -m esptool --chip esp8266 --before default_reset erase_flash
python3 -m esptool --chip esp8266 --before default_reset write_flash 0
↪"BAT_AT_V1.7.1.0_1M.bin"
```

6. Erscheint folgende Meldung, war der Vorgang erfolgreich.

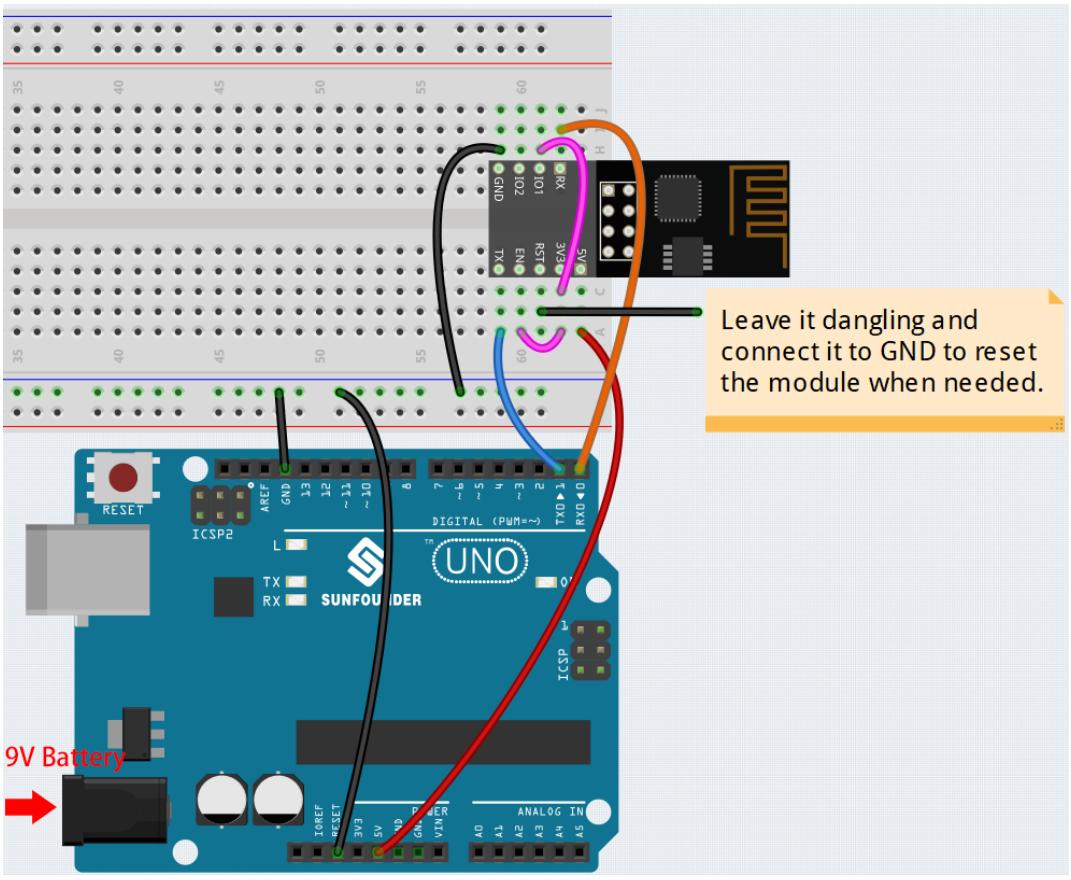
Bemerkung: Sollte der Brennvorgang fehlschlagen, überprüfen Sie bitte:

- Setzen Sie das ESP8266-Modul zurück (RST am ESP8266-Adapter mit GND verbinden und wieder trennen).
- Überprüfen Sie die Verkabelung.
- Stellen Sie sicher, dass Ihr Computer das Board erkannt hat und der Port frei ist.
- Öffnen Sie die Datei install.bat erneut.

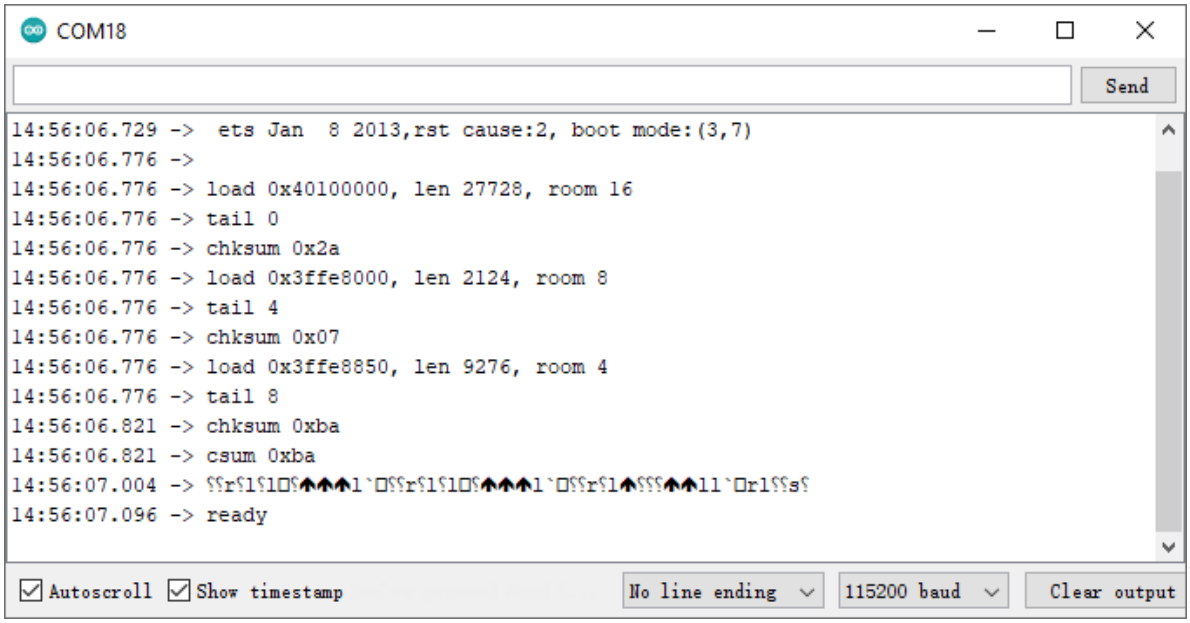
---

3. Test

1. Auf Grundlage der ursprünglichen Verkabelung, verbinden Sie IO1 mit 3V3.



2. Klicken Sie auf das Lupensymbol (Serial Monitor) in der oberen rechten Ecke und stellen Sie die Baudrate auf **115200** ein. Nun sollten Sie Informationen zum ESP8266-Modul sehen.

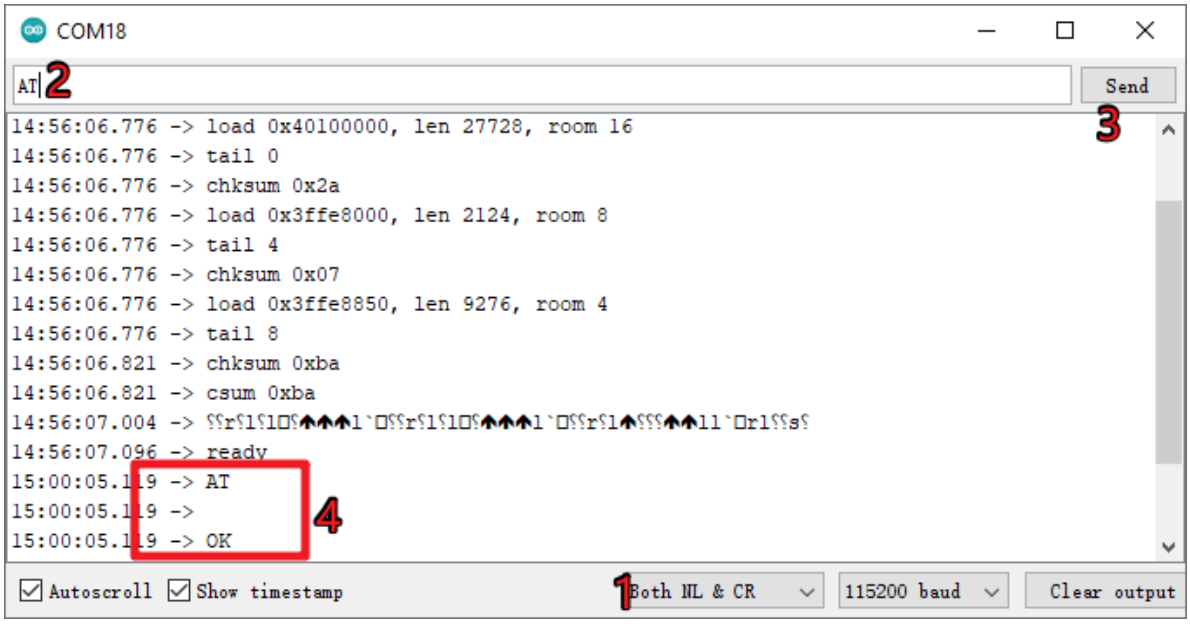


Bemerkung:

- Sollte die Meldung `ready` nicht erscheinen, können Sie versuchen, das ESP8266-Modul zurückzusetzen

(RST mit GND verbinden) und den Serial Monitor erneut zu öffnen.

3. Wählen Sie im **NEWLINE DROPDOWN BOX** die Option `both NL & CR` aus und geben Sie `AT` ein. Erscheint die Meldung OK, ist die Verbindung zwischen ESP8266 und dem R3-Board erfolgreich hergestellt.



Jetzt können Sie die *1.1 Konfiguration des ESP8266* befolgen, um den Arbeitsmodus und die Baudrate des ESP8266-Moduls einzustellen.

2.8 Vielen Dank

Ein herzlicher Dank geht an die Gutachter, die unsere Produkte bewertet haben, an die Experten, die Vorschläge für das Tutorial gemacht haben, und an die Nutzer, die uns kontinuierlich folgen und unterstützen. Ihre wertvollen Anregungen sind für uns Ansporn, immer bessere Produkte zu bieten!

Bei Fragen oder weiteren interessanten Ideen können Sie gerne eine E-Mail an service@sunfounder.com.

KAPITEL 3

Urheberrechtshinweis