

---

# **SunFounder Kepler Kit for Raspberry Pi Pico W**

***Release 1.0***

**[www.sunfounder.com](http://www.sunfounder.com)**

**15.01.2024**





---

## Inhaltsverzeichnis

---

<b>1</b>	<b>Raspberry Pi Pico W</b>	<b>3</b>
1.1	Merkmale . . . . .	4
1.2	Pico's Pins . . . . .	5
<b>2</b>	<b>Was ist in diesem Kit enthalten?</b>	<b>7</b>
2.1	Steckbrett . . . . .	8
2.2	Jumperkabel . . . . .	9
2.3	Widerstand . . . . .	10
2.4	Transistor . . . . .	13
2.5	Kondensator . . . . .	15
2.6	Diode . . . . .	16
2.7	Li-Po-Lademodul . . . . .	17
2.8	74HC595 . . . . .	19
2.9	TA6586 - Motorsteuerungs-Chip . . . . .	21
2.10	LED . . . . .	23
2.11	RGB-LED . . . . .	24
2.12	LED-Balkendiagramm . . . . .	27
2.13	7-Segment-Anzeige . . . . .	28
2.14	4-stellige 7-Segment-Anzeige . . . . .	30
2.15	LED-Punktmatrix . . . . .	32
2.16	I2C LCD1602 . . . . .	34
2.17	WS2812 RGB 8-LED-Streifen . . . . .	36
2.18	Summer . . . . .	37
2.19	Gleichstrommotor . . . . .	39
2.20	Servo . . . . .	41
2.21	DC-Wasserpumpe . . . . .	43
2.22	Relais . . . . .	44
2.23	Taster . . . . .	46
2.24	Mikroschalter . . . . .	47
2.25	Schiebeschalter . . . . .	49
2.26	Potentiometer . . . . .	50
2.27	Infrarotempfänger . . . . .	52
2.28	Joystick-Modul . . . . .	54
2.29	4x4 Tastenfeld . . . . .	55
2.30	MPR121 Modul . . . . .	57
2.31	MFRC522 Modul . . . . .	58

2.32	Fotowiderstand . . . . .	59
2.33	Thermistor . . . . .	60
2.34	Neigungsschalter . . . . .	62
2.35	Reedschalter . . . . .	63
2.36	PIR-Bewegungssensormodul . . . . .	64
2.37	Wasserspiegelsensor-Modul . . . . .	67
2.38	Ultraschallmodul . . . . .	68
2.39	DHT11 Temperatur- und Feuchtigkeitssensor . . . . .	69
2.40	MPU6050 Modul . . . . .	71
<b>3</b>	<b>Elektrischer Schaltkreis</b>	<b>75</b>
3.1	Hallo, Steckplatine! . . . . .	77
3.2	Vorsicht vor Kurzschlüssen . . . . .	79
3.3	Ausrichtung des Schaltkreises . . . . .	79
3.4	Schutz des Schaltkreises . . . . .	80
<b>4</b>	<b>Für MicroPython-Nutzer</b>	<b>83</b>
4.1	1.1 Einführung in MicroPython . . . . .	83
4.2	1.2 Installation der Thonny IDE . . . . .	84
4.3	1.3 Installation von MicroPython auf Ihrem Pico . . . . .	86
4.4	1.4 Bibliotheken auf den Pico hochladen . . . . .	90
4.5	1.5 Schnelle Einführung in Thonny . . . . .	93
4.6	1.6 (Optional) Grundlegende Syntax von MicroPython . . . . .	102
4.7	2.1 Hallo, LED! . . . . .	130
4.8	2.2 Anzeige des Pegels . . . . .	135
4.9	2.3 Abklingende LED . . . . .	140
4.10	2.4 Farbenfrohes Licht . . . . .	144
4.11	2.5 Tastenwert auslesen . . . . .	148
4.12	2.6 Neige es! . . . . .	152
4.13	2.7 Nach Links und Rechts Schalten . . . . .	155
4.14	2.8 Sanft Drücken . . . . .	158
4.15	2.9 Fühle den Magnetismus . . . . .	160
4.16	2.10 Bewegungserkennung beim Menschen . . . . .	163
4.17	2.11 Den Drehregler betätigen . . . . .	169
4.18	2.12 Das Licht spüren . . . . .	173
4.19	2.13 Thermometer . . . . .	176
4.20	2.14 Wasserstand erfühlen . . . . .	180
4.21	2.15 Zwei Arten von Transistoren . . . . .	183
4.22	2.16 Steuerung eines weiteren Stromkreises . . . . .	188
4.23	3.1 Piepton . . . . .	192
4.24	3.2 Eigener Ton . . . . .	195
4.25	3.3 RGB LED-Streifen . . . . .	199
4.26	3.4 Flüssigkristallanzeige . . . . .	202
4.27	3.5 Kleiner Ventilator . . . . .	206
4.28	3.6 Pumpensteuerung . . . . .	208
4.29	3.7 Schwingender Servo . . . . .	211
4.30	4.1 Den Joystick umschalten . . . . .	215
4.31	4.2 4x4 Tastenfeld . . . . .	218
4.32	4.3 Elektroden-Tastatur . . . . .	223
4.33	5.1 Mikrochip - 74HC595 . . . . .	226
4.34	5.2 Nummernanzeige . . . . .	230
4.35	5.3 Zeitmesser . . . . .	235
4.36	5.4 8x8 Pixel-Grafik . . . . .	240
4.37	6.1 Abstandsmessung . . . . .	246

4.38	6.2 Temperatur - Feuchtigkeit . . . . .	250
4.39	6.3 6-Achsen-Bewegungsverfolgung . . . . .	255
4.40	6.4 Infrarot-Fernbedienung . . . . .	258
4.41	6.5 Funkfrequenz-Identifikation . . . . .	264
4.42	7.1 Licht-Theremin . . . . .	268
4.43	7.2 Raumtemperaturmessgerät . . . . .	271
4.44	7.3 Alarmanlagenlampe . . . . .	274
4.45	7.4 Personen Zähler . . . . .	278
4.46	7.5 SPIEL - 10 Sekunden . . . . .	283
4.47	7.6 Ampel . . . . .	288
4.48	7.7 Zahlenraten . . . . .	292
4.49	7.8 RFID-Musikplayer . . . . .	298
4.50	7.9 Frucht-Klavier . . . . .	303
4.51	7.10 Einparkhilfe . . . . .	307
4.52	7.11 Somatosensorische Steuerung . . . . .	312
4.53	7.12 Digitaler Wasserwaage . . . . .	317
<b>5</b>	<b>IoT-Projekte</b>	<b>323</b>
5.1	1. Zugang zum Netzwerk . . . . .	323
5.2	2. Folgen Sie dem @CheerLights . . . . .	331
5.3	3. Sicherheitssystem über @IFTTT . . . . .	335
5.4	4. Echtzeit-Wetterdaten von @OpenWeatherMap . . . . .	348
5.5	5. Cloud-Rufsystem mit @MQTT . . . . .	357
5.6	6. Cloud-Player mit @MQTT . . . . .	362
5.7	7. Ein Webserver einrichten . . . . .	368
5.8	8. Web-App mit @Anvil erstellen . . . . .	376
5.9	9. Spiel mit dem @SunFounder Controller . . . . .	395
5.10	10. Pflanzenüberwachung mit dem @SunFounder Controller . . . . .	404
<b>6</b>	<b>Für Arduino-Nutzer</b>	<b>409</b>
6.1	1.1 Arduino IDE installieren (Wichtig) . . . . .	409
6.2	1.2 Vorstellung der Arduino IDE . . . . .	415
6.3	1.3 Raspberry Pi Pico Weinrichten (Wichtig) . . . . .	416
6.4	1.4 Bibliotheken installieren (Wichtig) . . . . .	423
6.5	2.1 - Hallo, LED! . . . . .	425
6.6	2.2 - Pegelanzeige . . . . .	428
6.7	2.3 - Verblässende LED . . . . .	432
6.8	2.4 - Farbenfrohes Licht . . . . .	436
6.9	2.5 - Tastenwert auslesen . . . . .	440
6.10	2.6 - Kipp es! . . . . .	444
6.11	2.7 - Links und Rechts Umschalten . . . . .	446
6.12	2.8 - Sanft Drücken . . . . .	449
6.13	2.9 - Magnetismus spüren . . . . .	451
6.14	2.10 - Menschliche Bewegung erfassen . . . . .	453
6.15	2.11 - Drehen Sie den Knopf . . . . .	457
6.16	2.12 - Das Licht erfassen . . . . .	461
6.17	2.13 - Thermometer . . . . .	463
6.18	2.14 - Den Wasserstand erfühlen . . . . .	467
6.19	2.15 - Zwei Arten von Transistoren . . . . .	470
6.20	2.16 - Steuern eines weiteren Stromkreises . . . . .	474
6.21	3.1 - Piepton . . . . .	478
6.22	3.2 - Individueller Ton . . . . .	481
6.23	3.3 WS2812 RGB-Strip . . . . .	484
6.24	3.4 - Flüssigkristallanzeige . . . . .	487

6.25	3.5 - Kleiner Ventilator . . . . .	491
6.26	3.6 - Pumpensteuerung . . . . .	494
6.27	3.7 - Schwingender Servo . . . . .	496
6.28	4.1 - Den Joystick bedienen . . . . .	500
6.29	4.2 - 4x4 Tastenfeld . . . . .	503
6.30	4.3 - Elektroden-Tastatur . . . . .	506
6.31	5.1 Mikrochip - 74HC595 . . . . .	509
6.32	5.2 - Zahlenanzeige . . . . .	512
6.33	5.3 - Zeitmesser . . . . .	515
6.34	5.4 - 8x8 Pixelgrafik . . . . .	519
6.35	6.1 - Abstandsmessung . . . . .	524
6.36	6.2 - Temperatur - Feuchtigkeit . . . . .	527
6.37	6.3 - 6-Achsen-Bewegungsverfolgung . . . . .	531
6.38	6.4 - IR-Fernbedienung . . . . .	535
6.39	6.5 - Funkfrequenz-Identifikation . . . . .	538
<b>7</b>	<b>Für Piper Make</b>	<b>543</b>
7.1	1.1 Einrichtung des Pico . . . . .	543
7.2	1.2 Schnellstartanleitung für Piper Make . . . . .	547
7.3	1.3 Wie speichert oder importiert man Code? . . . . .	554
7.4	2.1 LED Blinken Lassen . . . . .	556
7.5	2.2 Taster . . . . .	558
7.6	2.3 Serviceklingel . . . . .	561
7.7	2.4 Regenbogenlicht . . . . .	563
7.8	2.5 Schlagzeug-Set . . . . .	567
7.9	2.6 Intelligenter Wassertank . . . . .	570
7.10	2.7 Schwenk-Servo . . . . .	574
7.11	2.8 Lichtintensitätsanzeige . . . . .	577
7.12	2.9 Glückskatze . . . . .	580
7.13	2.10 Fließende LEDs . . . . .	584
7.14	2.11 Rückfahrssystem . . . . .	591
7.15	2.12 Intelligenter Ventilator . . . . .	595
7.16	2.13 Reaktionsspiel . . . . .	600
<b>8</b>	<b>Videokurse</b>	<b>607</b>
<b>9</b>	<b>FAQ</b>	<b>609</b>
9.1	Arduino . . . . .	609
9.2	MicroPython . . . . .	609
9.3	Piper Make . . . . .	611
<b>10</b>	<b>Urheberrechtshinweis</b>	<b>613</b>

Danke, dass Sie sich für unser Kepler 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.

Vielen Dank, dass Sie sich für das SunFounder Kepler Kit entschieden haben.

Dies ist ein Lernkit, das auf dem Raspberry Pi Pico W basiert.

Der Raspberry Pi Pico W verfügt über integrierte Einzelband-2,4-GHz-WLAN-Schnittstellen (802.11n) mittels des Infineon CYW4343 und behält gleichzeitig den Pico-Formfaktor bei. Neben den grundlegenden GPIO-Funktionen ermöglicht er auch die Netzwerkanbindung, sodass er für diverse IoT-Projekte einsetzbar ist. Beispielsweise lässt sich IFTTT für ein Sicherheitssystem nutzen oder ein Cloud-Player und ein Cloud-Klingelsystem mit MQTT realisieren.

Das Kit enthält eine breite Palette an Komponenten wie Displays, Tonausgabe, Treiber, Steuerungen und Sensoren, wodurch Sie ein umfassendes Verständnis für elektronische Bauteile erhalten.

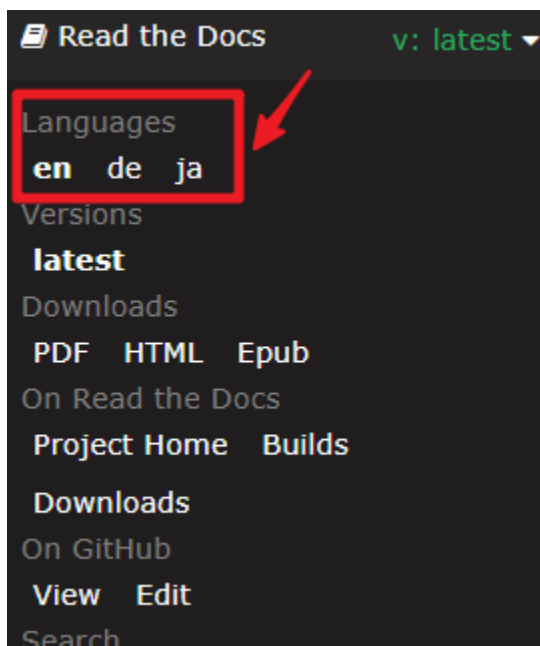
Darüber hinaus stehen Ihnen drei Programmiersprachen zur Verfügung: MicroPython, C/C++ (Arduino) und Piper Make. Um den Einstieg zu erleichtern, bietet jede Sprache gezielte und interessante Projekte, sodass Sie je nach Bedarf wählen können.

Bei Interesse an weiteren Projekten, die wir bisher nicht anbieten, können Sie uns gerne per E-Mail kontaktieren. Wir werden unsere Online-Tutorials so schnell wie möglich aktualisieren.

Hier ist die E-Mail-Adresse: [service@sunfounder.com](mailto:service@sunfounder.com).

## Zur Anzeigesprache

Dieses Dokument ist auch in anderen Sprachen verfügbar. Um die Anzeigesprache zu wechseln, klicken Sie bitte auf das Icon **Read the Docs**, das sich in der unteren linken Ecke der Seite befindet.

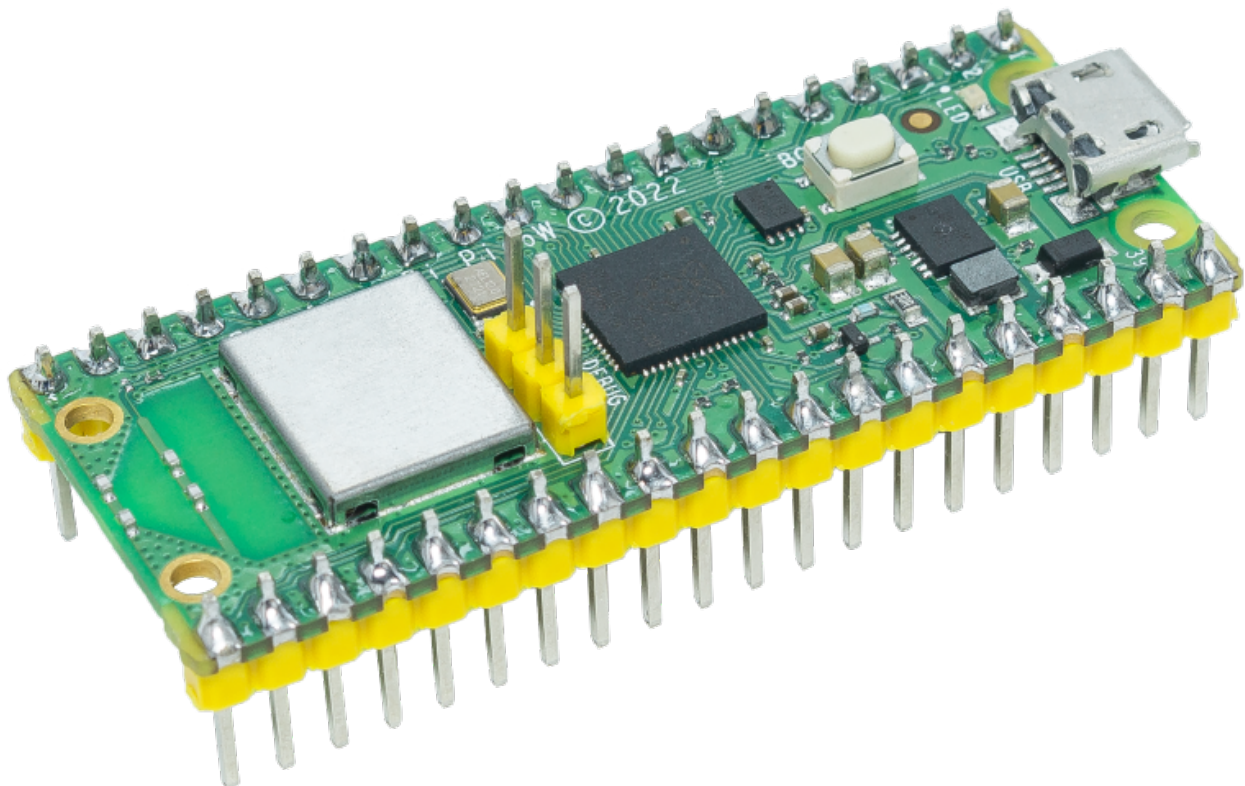


Quellcode

- SunFounder Kepler Kit
- Oder sehen Sie sich den Code unter [Kepler Kit - GitHub](#) an.

### Inhalt

## Raspberry Pi Pico W



Mit dem Raspberry Pi Pico W bringt die erfolgreiche Raspberry Pi Pico-Produktreihe nun auch drahtlose Konnektivität ins Spiel. Basierend auf unserer RP2040-Siliziumplattform stehen Pico-Produkte für hohe Leistung, geringe Kosten und einfache Handhabung im Mikrocontroller-Bereich.

Der Raspberry Pi Pico W bietet eine 2,4 GHz 802.11 b/g/n WLAN-Unterstützung, eine integrierte Antenne und eine modulare Konformitätszertifizierung. Er kann sowohl im Station- als auch im Access-Point-Modus betrieben werden. Sowohl C- als auch MicroPython-Entwickler haben vollen Zugriff auf die Netzwerkfunktionalität. Der Raspberry Pi

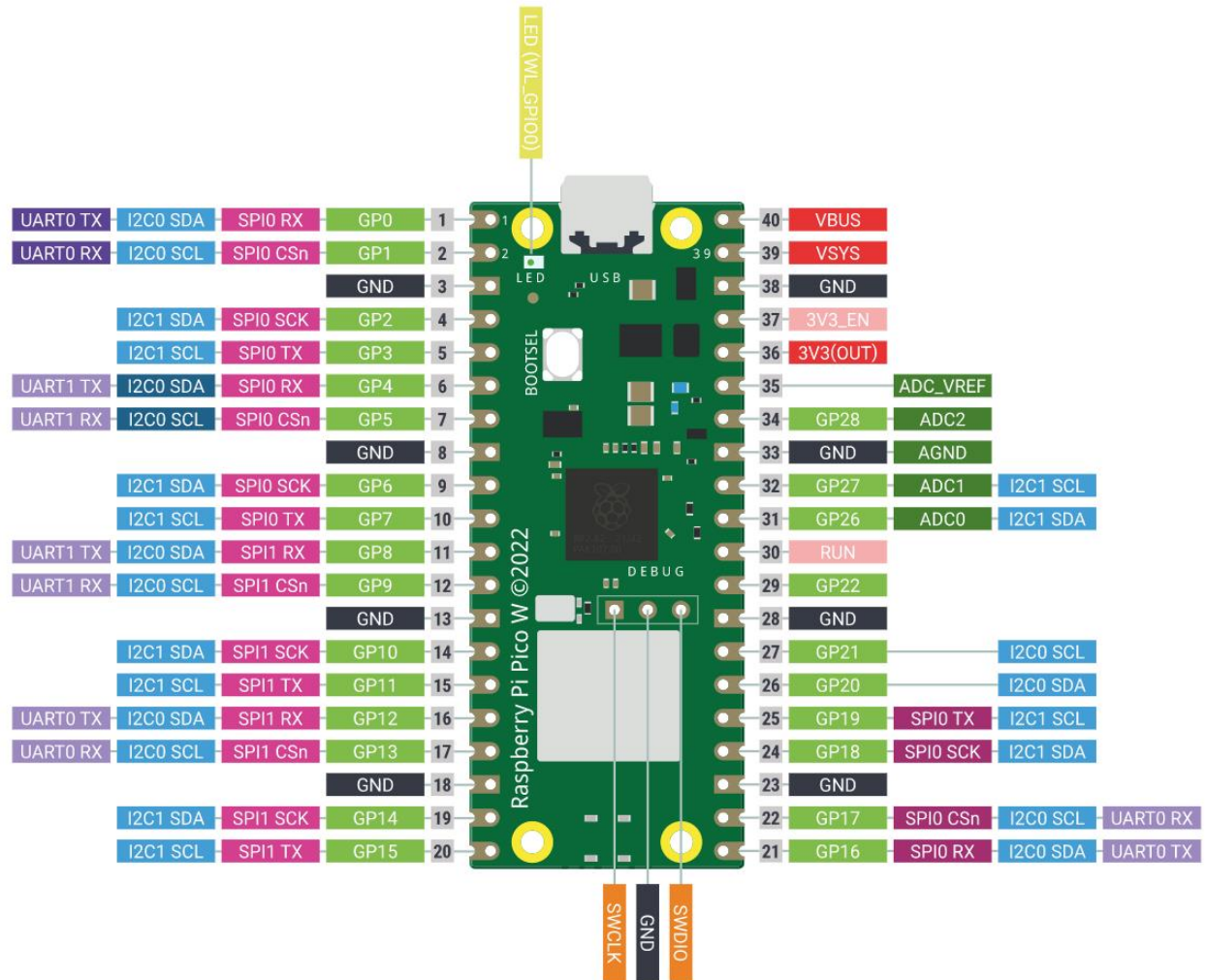
Pico W kombiniert den RP2040 mit 2 MB Flash-Speicher und einem Spannungsversorgungschip, der Eingangsspannungen von 1,8 bis 5,5 V unterstützt. Er bietet 26 GPIO-Pins, von denen drei als analoge Eingänge fungieren können, auf Durchstecklöchern im 0,1-Zoll-Raster mit gegossenen Kanten. Der Raspberry Pi Pico W ist sowohl einzeln als auch in 480er-Spulen für die automatisierte Montage erhältlich.

### 1.1 Merkmale

- Abmessungen: 21 mm x 51 mm
- RP2040-Mikrocontroller-Chip, entwickelt von Raspberry Pi in Großbritannien
- Dual-Core Arm Cortex-M0+ Prozessor, flexible Taktung bis zu 133 MHz
- 264 kB integrierter SRAM
- 2 MB Onboard-QSPI-Flash
- 2,4 GHz 802.11n WLAN
- 26 multifunktionale GPIO-Pins, inklusive 3 analoger Eingänge
- 2 x UART, 2 x SPI, 2 x I2C, 16 x PWM-Kanäle
- 1 x USB 1.1 mit Host- und Device-Unterstützung
- 8 x programmierbare I/O (PIO) Zustandsmaschinen für benutzerdefinierte Peripheriegeräte
- Unterstützte Eingangsspannung 1,8-5,5 V DC
- Betriebstemperatur -20°C bis +70°C
- Lötfähiges Modul für direkte Montage auf Trägerplatinen
- Drag-and-drop-Programmierung über USB-Massenspeicher
- Energiesparmodi im Schlaf- und Ruhezustand
- Präzise On-Chip-Uhr
- Temperatursensor
- Beschleunigte Ganzzahl- und Gleitkomma-Bibliotheken auf dem Chip



## 1.2 Pico's Pins



Name	Beschreibung	Funktion
GP0-GP28	Allzweck-Ein-/Ausgabepins	Können als Ein- oder Ausgang fungieren und haben keine festgelegte Funktion.
GND	0-Volt-Erde	Mehrere GND-Pins am Pico W erleichtern die Verdrahtung.
RUN	Aktiviert oder deaktiviert Ihren Pico	Starten und Stoppen Ihres Pico W von einem anderen Mikrocontroller aus.
GPxx_ADCx	Allzweck-Ein-/Ausgang oder analoger Eingang	Können sowohl als analoger als auch als digitaler Ein- oder Ausgang verwendet werden – jedoch nicht gleichzeitig.
ADC_VREF	Analog-Digital-Wandler (ADC) Spannungsreferenz	Ein spezieller Eingangspin, der eine Referenzspannung für analoge Eingänge festlegt.
AGND	Analog-Digital-Wandler (ADC) 0-Volt-Erde	Eine spezielle Erdverbindung zur Verwendung mit dem ADC_VREF-Pin.
3V3(O)	3,3-Volt-Stromversorgung	Eine 3,3-Volt-Stromquelle, dieselbe Spannung, bei der Ihr Pico W intern betrieben wird, generiert aus dem VSYS-Eingang.
3V3(E)	Aktiviert oder deaktiviert die Stromversorgung	Schaltet die 3V3(O)-Stromversorgung ein oder aus und kann auch Ihren Pico W ausschalten.
VSYS	2-5-Volt-Stromversorgung	Ein direkt mit der internen Stromversorgung Ihres Pico verbundener Pin, der nicht abgeschaltet werden kann, ohne den Pico W ebenfalls auszuschalten.
VBUS	5-Volt-Stromversorgung	Eine 5-Volt-Stromquelle aus dem Micro-USB-Anschluss Ihres Pico, die zur Stromversorgung von Hardware dient, die mehr als 3,3 V benötigt.

Alles, was Sie für den Einstieg mit Ihrem Raspberry Pi Pico W benötigen, finden Sie [hier](#)

Oder klicken Sie auf die folgenden Links:

- [Raspberry Pi Pico W Produktübersicht](#)
- [Raspberry Pi Pico W Datenblatt](#)
- [Erste Schritte mit dem Raspberry Pi Pico: C/C++-Entwicklung](#)
- [Raspberry Pi Pico C/C++ SDK](#)
- [API-Ebene Doxygen-Dokumentation für das Raspberry Pi Pico C/C++ SDK](#)
- [Raspberry Pi Pico Python SDK](#)
- [Raspberry Pi RP2040 Datenblatt](#)
- [GitHub-Repository für die Raspberry Pi Pico Firmware](#)
- [Community-Forum](#)

## KAPITEL 2

---

### Was ist in diesem Kit enthalten?

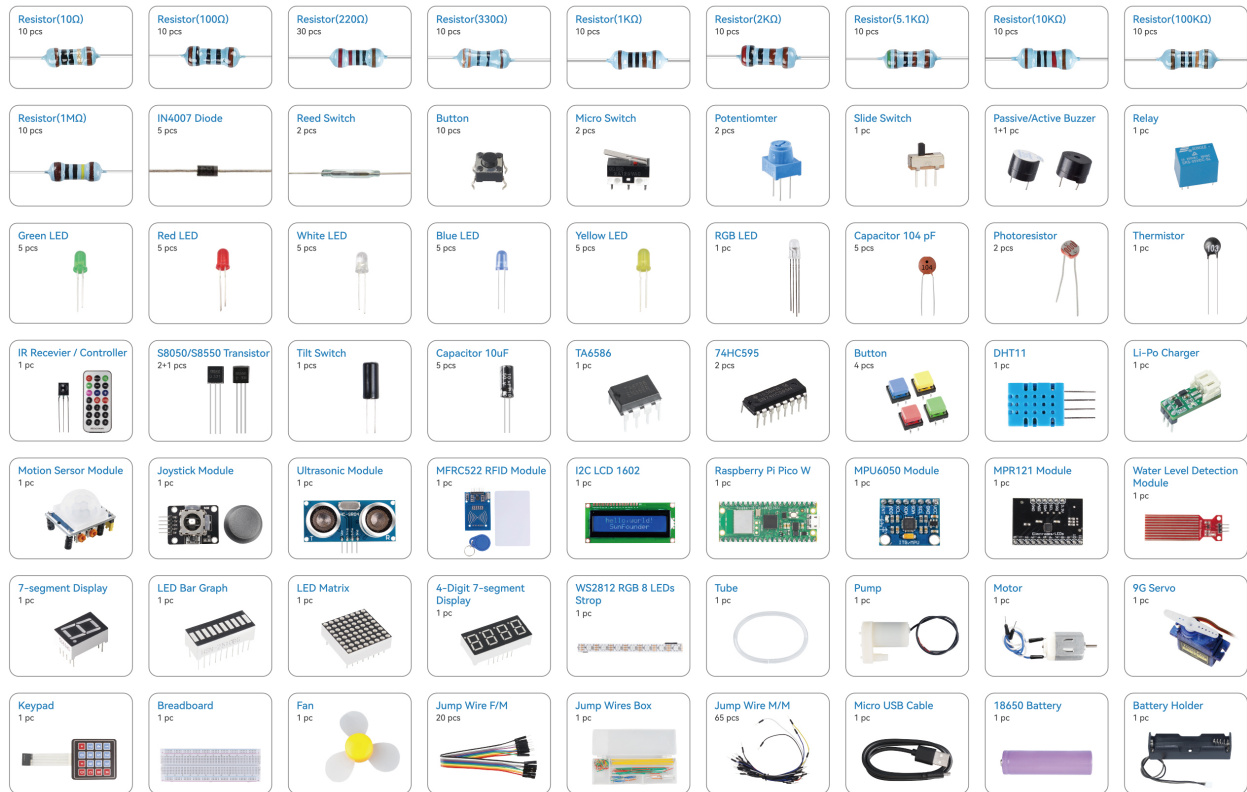
---

Nachfolgend finden Sie eine Liste des Kits, damit Sie den Inhalt überprüfen können, sobald Sie es erhalten haben.

Einige Komponenten im Kit sind sehr klein und sehen gleich aus. Es kann vorkommen, dass unser Personal beim Verpacken des Kits etwas übersieht oder versehentlich falsche Teile beilegt. Sollten Sie einen fehlenden oder falschen Bestandteil feststellen, können Sie uns gerne den Namen der betreffenden Komponente mitteilen.

Hier ist die E-Mail: [service@sunfounder.com](mailto:service@sunfounder.com).

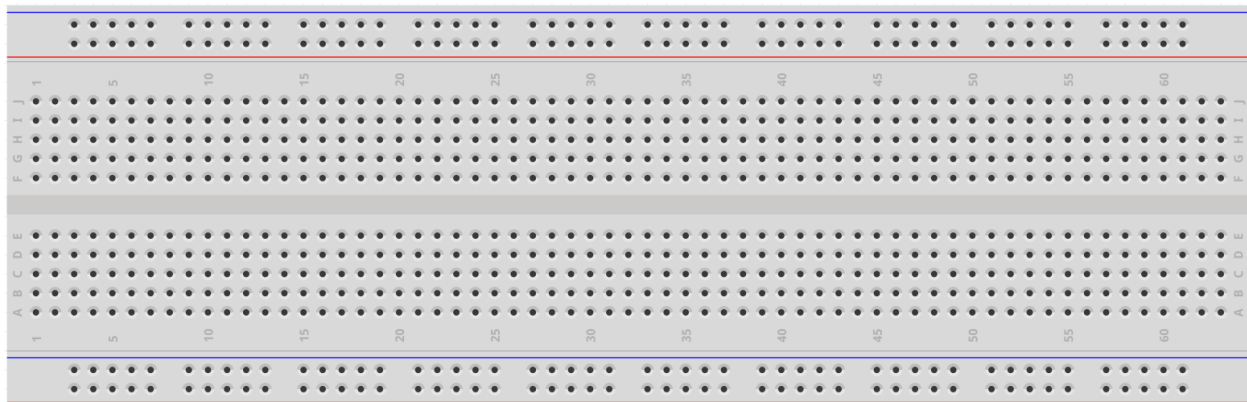
## Packing List



Online Tutorials: <https://kepler-kit.rtfid.io>

## Grundlagen

### 2.1 Steckbrett

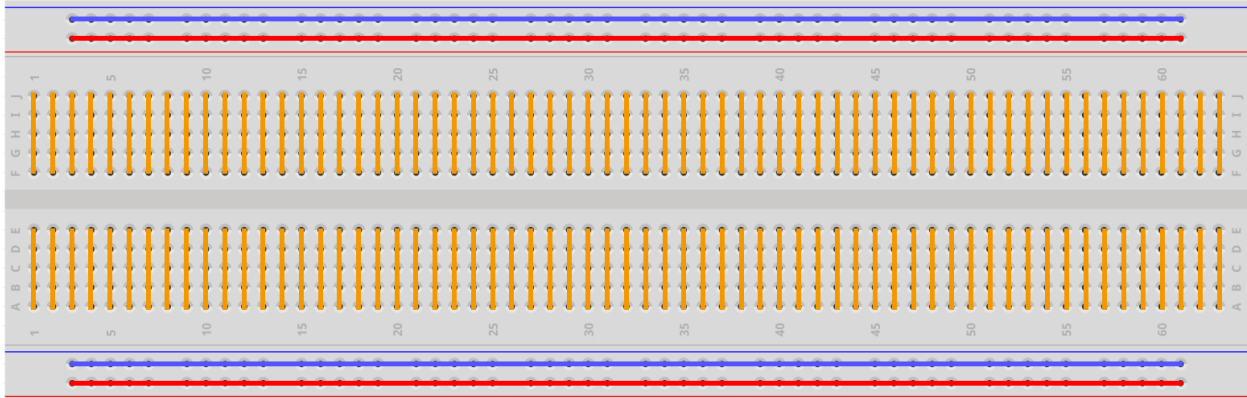


Ein Steckbrett dient als Grundlage für den Prototypenbau in der Elektronik. Ursprünglich bezeichnete der Begriff tatsächlich ein Schneidebrett aus Holz, das zum Brot schneiden genutzt wurde. In den 1970er Jahren kam das lötfreie Steckbrett (auch bekannt als Steckplatine oder Terminal-Array-Platine) auf den Markt und heutzutage wird der Begriff „Steckbrett“ vorwiegend in diesem Kontext verwendet.

Es wird eingesetzt, um Schaltungen schnell zu bauen und zu testen, bevor ein endgültiges Schaltungsdesign festgelegt wird. Das Steckbrett verfügt über zahlreiche Löcher, in die die oben genannten Komponenten wie ICs und Widerstände

sowie Verbindungskabel eingesteckt werden können. Das Steckbrett ermöglicht das einfache Ein- und Ausstecken von Komponenten.

Das Bild zeigt den internen Aufbau eines Steckbretts. Obwohl die Löcher im Steckbrett scheinbar voneinander unabhängig sind, sind sie intern tatsächlich durch Metallstreifen miteinander verbunden.



Falls Sie mehr über Steckbretter erfahren möchten, verweisen wir auf: [Wie benutzt man ein Steckbrett - Science Buddies](#)

### Beispiel

- *Hallo, Steckplatine!*

## 2.2 Jumperkabel

Kabel, die zwei Anschlusspunkte miteinander verbinden, werden als Jumperkabel bezeichnet. Es gibt verschiedene Arten von Jumperkabeln. In diesem Abschnitt konzentrieren wir uns auf die Kabel, die in Steckbrettern verwendet werden. Unter anderem dienen sie dazu, elektrische Signale von beliebigen Punkten auf dem Steckbrett zu den Ein-/Ausgangspins eines Mikrocontrollers zu übertragen.

Jumperkabel werden eingesetzt, indem ihre „Endstecker“ in die dafür vorgesehenen Schlitze im Steckbrett eingeführt werden. Unter der Oberfläche des Steckbretts befinden sich mehrere Reihen von parallelen Platten, die die Schlitze in Gruppen von Reihen oder Spalten miteinander verbinden, je nach Bereich. Die „Endstecker“ werden ohne Löten in die spezifischen Schlitze eingeführt, die im jeweiligen Prototypen miteinander verbunden werden sollen.

Es gibt drei Typen von Jumperkabeln: Female-to-Female, Male-to-Male und Male-to-Female. Ein Male-to-Female-Kabel zeichnet sich durch einen vorstehenden Stift an einem Ende und eine eingelassene Buchse am anderen Ende aus. Male-to-Male bedeutet, dass beide Enden männlich sind, und Female-to-Female, dass beide Enden weiblich sind.

Male-to-Female



Male-to-Male



Female-to-Female




---

### Bemerkung:

- In einem Projekt können mehrere Typen von Jumperkabeln verwendet werden.
  - Die Farbe der Jumperkabel hat keine Auswirkung auf deren Funktion; sie dient lediglich dazu, die Verbindungen zwischen den einzelnen Schaltkreisen besser identifizieren zu können.
-

## 2.3 Widerstand



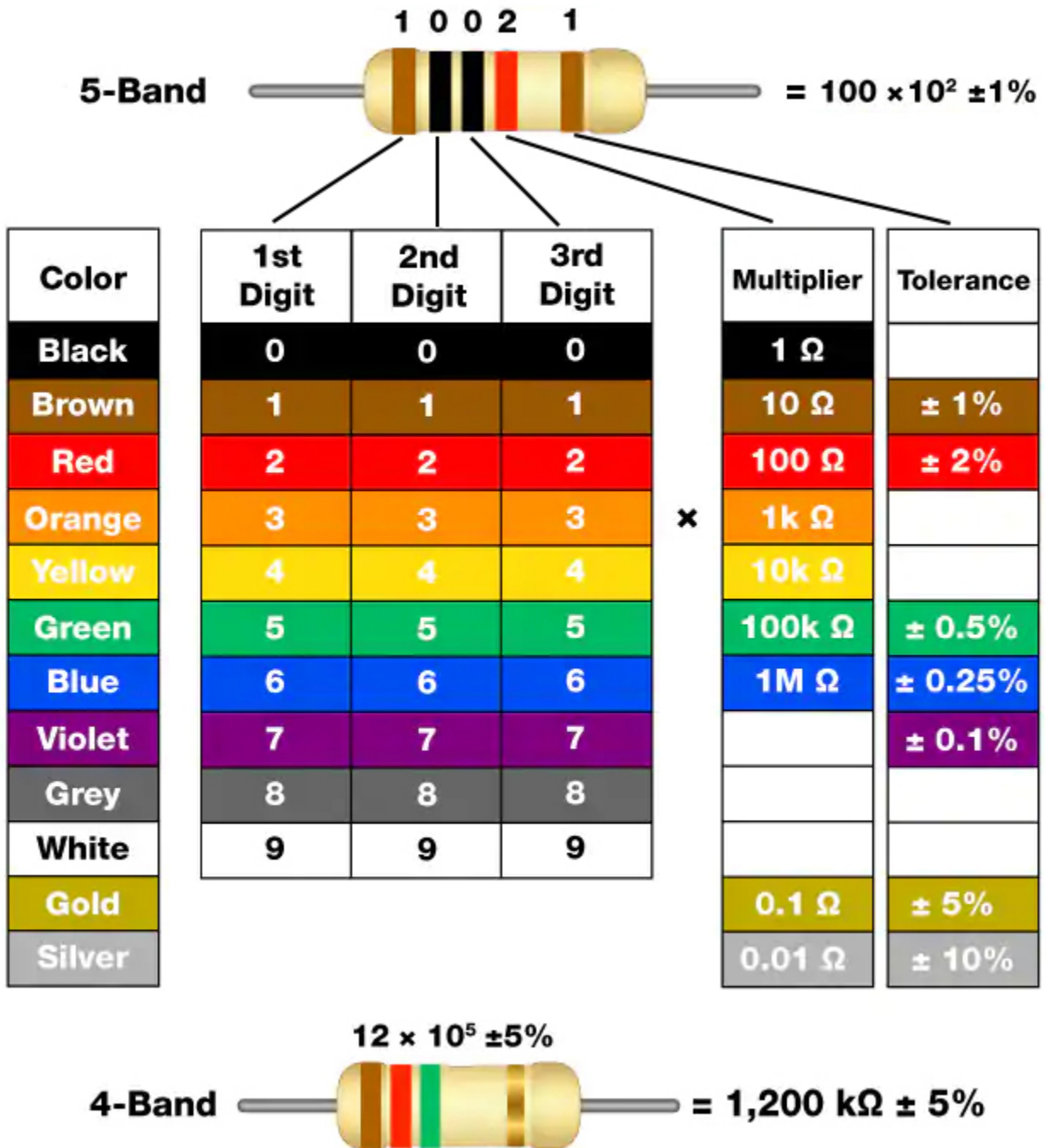
Ein Widerstand ist ein elektronisches Bauelement, das den Strom in einem Zweig begrenzen kann. Ein Festwiderstand ist eine Form von Widerstand, dessen Widerstandswert nicht verändert werden kann, während der Widerstand eines Potentiometers oder eines variablen Widerstands einstellbar ist.

Es gibt zwei allgemein verwendete Schaltsymbole für Widerstände. Normalerweise ist der Widerstandswert darauf gekennzeichnet. Wenn Sie also diese Symbole in einer Schaltung sehen, handelt es sich um einen Widerstand.



ist die Einheit des elektrischen Widerstands, und größere Einheiten umfassen K, M usw. Die Beziehung zwischen ihnen lässt sich wie folgt darstellen:  $1\text{ M} = 1000\text{ K}$ ,  $1\text{ K} = 1000$ . In der Regel ist der Widerstandswert darauf markiert.

Bevor man einen Widerstand verwendet, muss man seinen Widerstandswert kennen. Es gibt zwei Methoden: Man kann die Farbringe auf dem Widerstand ablesen oder einen Multimeter verwenden, um den Widerstand zu messen. Die erste Methode wird empfohlen, da sie bequemer und schneller ist.



Wie auf der Karte gezeigt, steht jede Farbe für eine Nummer.

Schwarz	Braun	Rot	Orange	Gelb	Grün
0	1	2	3	4	5

Blau	Violett	Grau	Weiß	Gold	Silber
6	7	8	9	0.1	0.01

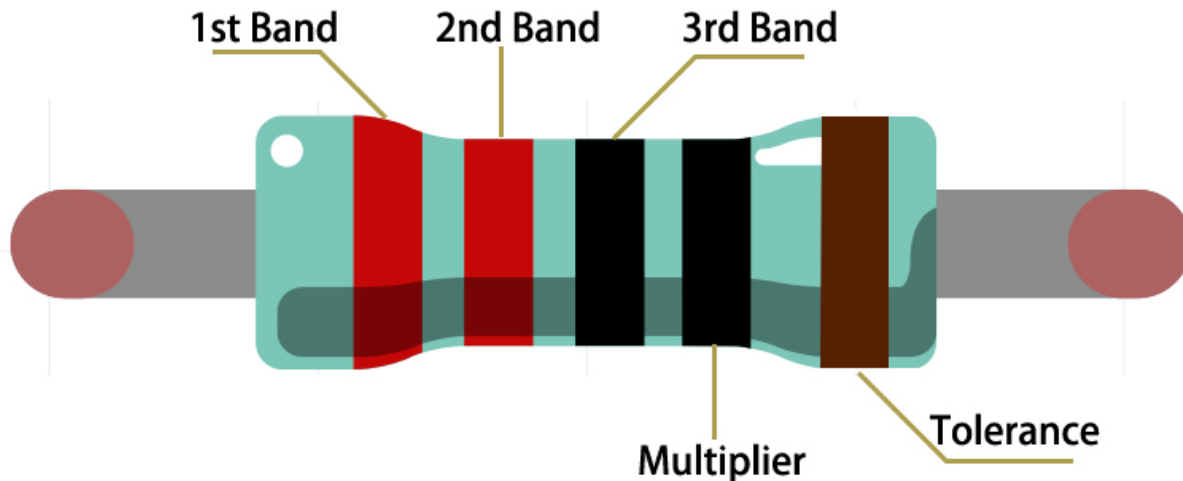


Vier- und fünfbandige Widerstände werden häufig verwendet und haben jeweils vier bzw. fünf farbige Ringe.

Normalerweise ist es nicht sofort ersichtlich, an welchem Ende man beginnen sollte, die Farben abzulesen. Ein Hinweis ist, dass der Abstand zwischen dem 4. und 5. Ring vergleichsweise größer ist.

Daher kann man die Lücke zwischen den beiden farbigen Ringen an einem Ende des Widerstands betrachten; ist sie größer als alle anderen Lücken, dann kann man von der gegenüberliegenden Seite ablesen.

Sehen wir uns an, wie man den Widerstandswert eines 5-bändigen Widerstands abliest, wie unten gezeigt.



Für diesen Widerstand sollte der Widerstandswert von links nach rechts abgelesen werden. Der Wert sollte in folgendem Format vorliegen: 1. Band 2. Band 3. Band x  $10^{\text{Multiplikator}}$  () und der zulässige Fehler beträgt  $\pm \text{Toleranz}\%$ . So beträgt der Widerstandswert dieses Widerstands 2(rot) 2(rot) 0(schwarz) x  $10^0(\text{schwarz}) = 220$  , und der zulässige Fehler beträgt  $\pm 1\%$  (braun).

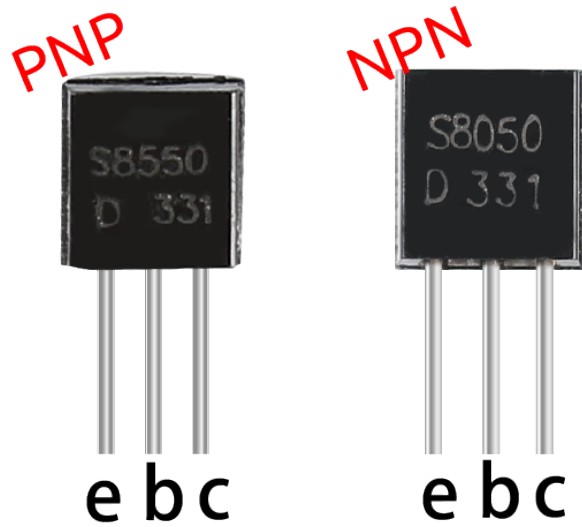
Tab. 1: Übliche Widerstands-Farbcodes

Widerstand	Farbcode
10	braun schwarz schwarz silber braun
100	braun schwarz schwarz schwarz braun
220	rot rot schwarz schwarz braun
330	orange orange schwarz schwarz braun
1k	braun schwarz schwarz braun braun
2k	rot schwarz schwarz braun braun
5.1k	grün braun schwarz braun braun
10k	braun schwarz schwarz rot braun
100k	braun schwarz schwarz orange braun
1M	braun schwarz schwarz grün braun

Mehr über Widerstände erfahren Sie auf der Wikipedia-Seite: [Widerstand – Wikipedia](#).



## 2.4 Transistor



Ein Transistor ist ein Halbleiterbauelement, das Strom mittels Strom steuert. Seine Hauptfunktionen bestehen darin, schwache Signale zu verstärken und als berührungsloser Schalter zu agieren.

Ein Transistor besteht aus einer dreischichtigen Struktur aus P-Typ und N-Typ Halbleitern. Diese bilden die drei internen Regionen: Die dünnere Mitte ist die Basisregion; die beiden anderen sind entweder N-Typ oder P-Typ – die kleinere Region mit einer hohen Anzahl an Ladungsträgern ist die Emitterregion, während die andere als Kollektorregion fungiert. Diese Anordnung ermöglicht die Verstärkerfunktion des Transistors. Aus diesen drei Regionen gehen jeweils drei Pole hervor: Basis (b), Emitter (e) und Kollektor (c). Sie bilden zwei P-N-Übergänge, nämlich den Emitter- und den Kollektorübergang. Die Pfeilrichtung im Transistorsymbol gibt die Richtung des Emitterübergangs an.

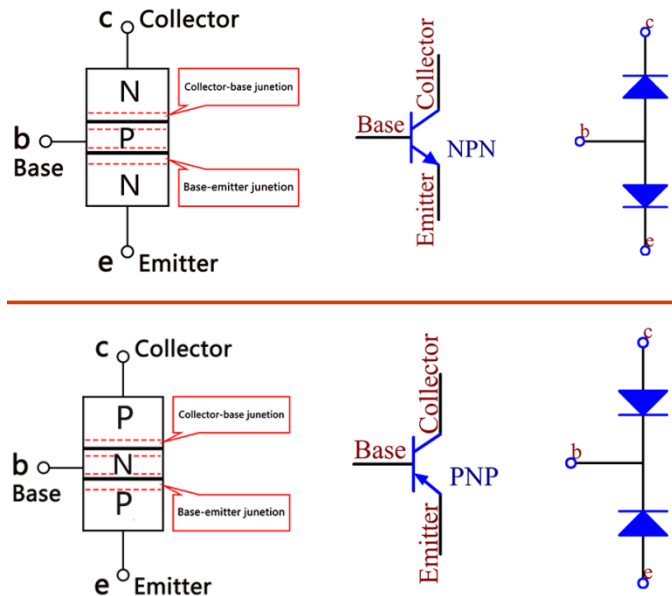
- [P-N-Übergang – Wikipedia](#)

Je nach Halbleitertyp lassen sich Transistoren in zwei Gruppen einteilen: NPN und PNP. Aus den Abkürzungen geht hervor, dass der erstere aus zwei N-Typ und einem P-Typ Halbleiter besteht, während der letztere das Gegenteil darstellt. Siehe die untenstehende Abbildung.

---

**Bemerkung:** Der s8550 ist ein PNP-Transistor und der s8050 ein NPN-Transistor. Sie sehen sehr ähnlich aus; es ist also wichtig, die Beschriftungen genau zu prüfen.

---



Ein NPN-Transistor wird durch ein Hochpegelsignal aktiviert, während für einen PNP-Transistor ein Lowpegelsignal erforderlich ist. Beide Transistortypen werden häufig als berührungslose Schalter eingesetzt, wie in diesem Experiment.

- [S8050 Transistor Datenblatt](#)
- [S8550 Transistor Datenblatt](#)

Richten Sie die beschriftete Seite zu sich und die Pins nach unten. Die Pins von links nach rechts sind Emitter (e), Basis (b) und Kollektor (c).



#### Bemerkung:

- Die Basis dient als Steuereinheit für die größere Stromquelle.
- Im NPN-Transistor ist der Kollektor die größere Stromquelle und der Emitter der Ausgang, im PNP-Transistor ist es genau umgekehrt.

#### Beispiel

- *2.15 Zwei Arten von Transistoren* (Für MicroPython-Nutzer)
- *2.16 Steuerung eines weiteren Stromkreises* (Für MicroPython-Nutzer)
- *3.1 Piepton* (Für MicroPython-Nutzer)
- *3.2 Eigener Ton* (Für MicroPython-Nutzer)
- *7.1 Licht-Theremin* (Für MicroPython-Nutzer)
- *7.3 Alarmanlagenlampe* (Für MicroPython-Nutzer)
- *7.8 RFID-Musikplayer* (Für MicroPython-Nutzer)
- *7.9 Frucht-Klavier* (Für MicroPython-Nutzer)
- *7.10 Einparkhilfe* (Für MicroPython-Nutzer)
- *3.1 - Piepton* (Für Arduino-Nutzer)
- *3.2 - Individueller Ton* (Für Arduino-Nutzer)
- *2.15 - Zwei Arten von Transistoren* (Für Arduino-Nutzer)
- *2.16 - Steuern eines weiteren Stromkreises* (Für Arduino-Nutzer)
- *2.3 Serviceklingel* (Für Piper Make-Nutzer)
- *2.11 Rückfahrsystem* (Für Piper Make-Nutzer)
- *2.13 Reaktionsspiel* (Für Piper Make-Nutzer)

## 2.5 Kondensator



Die Kapazität gibt an, wie viel elektrische Ladung bei einer gegebenen Spannung gespeichert werden kann. Sie wird mit C bezeichnet und ihre internationale Einheit ist das Farad (F). In einem elektrischen Feld bewegen sich elektrische Ladungen grundsätzlich durch die wirkende Kraft. Ist zwischen den Leitern jedoch ein Medium vorhanden, wird die Bewegung der Ladungen behindert und sie sammeln sich an den Leitern an.

Diese Ansammlung von elektrischen Ladungen wird als Kapazität bezeichnet. Kondensatoren sind aufgrund ihrer vielfältigen Einsatzmöglichkeiten in der Elektronik allgegenwärtig. Sie finden Anwendung in Gleichstrom-Isolierung, Kopplung, Bypass, Filterung, Schwingkreisen, Energieumwandlung und Steuerschaltungen. Es gibt verschiedene Arten von Kondensatoren, wie etwa Elektrolytkondensatoren und Festkörperkondensatoren.

Je nach Materialbeschaffenheit unterscheidet man zudem Aluminium-Elektrolytkondensatoren, Folienkondensatoren, Tantalkondensatoren, Keramikkondensatoren und Superkondensatoren.

In diesem Bausatz werden Keramik- und Elektrolytkondensatoren verwendet.

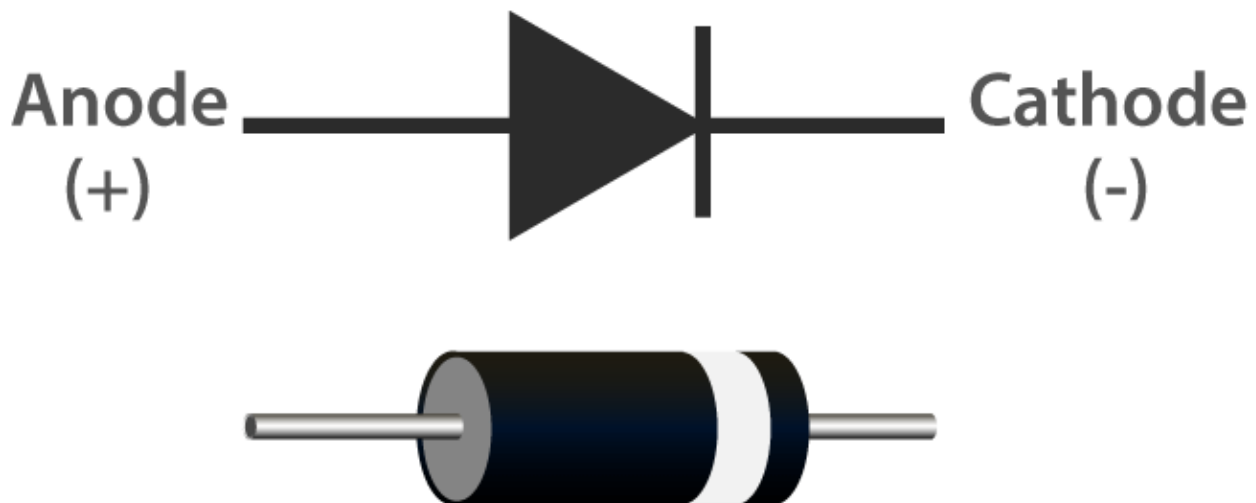
- [Keramikkondensator – Wikipedia](#)
- [Elektrolytkondensator – Wikipedia](#)

Auf den Keramikkondensatoren finden sich Beschriftungen wie 103 oder 104, die den Kapazitätswert angeben. Dabei steht 103 für  $10 \times 10^3 \text{ pF}$  und 104 für  $10 \times 10^4 \text{ pF}$ .

### Einheitenumrechnung

$$1\text{F}=10^3\text{mF}=10^6\mu\text{F}=10^9\text{nF}=10^{12}\text{pF}$$

## 2.6 Diode



Eine Diode ist ein elektronisches Bauelement mit zwei Elektroden, das den Stromfluss nur in einer Richtung zulässt. Diese Funktion wird oft als „Gleichrichtung“ bezeichnet. Somit kann die Diode als elektronisches Äquivalent eines Rückschlagventils betrachtet werden.

Die beiden Anschlüsse einer Diode sind polarisiert; das positive Ende wird Anode und das negative Ende Kathode genannt. Die Kathode ist üblicherweise aus Silber oder durch einen Farbring gekennzeichnet. Eine der Schlüsseleigenschaften von Dioden ist die Steuerung der Stromrichtung — der Strom fließt von der Anode zur Kathode. Das Verhalten einer Diode ist ähnlich dem eines Rückschlagventils. Eine wesentliche Charakteristik ist die nicht-lineare Strom-Spannungs-Kennlinie. Wird eine höhere Spannung an der Anode angelegt, fließt Strom von der Anode zur Kathode; dieser Zustand wird als Vorwärtsbias bezeichnet. Im umgekehrten Fall, also wenn die höhere Spannung an der Kathode anliegt, leitet die Diode keinen Strom; dieser Zustand wird als Rückwärtsbias bezeichnet.

Aufgrund ihrer unidirektionalen Leitfähigkeit findet die Diode in nahezu allen komplexeren elektronischen Schaltungen Verwendung. Sie war eines der ersten Halbleiterbauelemente und ihre Anwendungsgebiete sind vielfältig.

In der Realität weisen Dioden jedoch keine perfekte Ein- und Ausschaltcharakteristik auf, sondern vielmehr komplexe nicht-lineare elektronische Eigenschaften, die von der spezifischen Diodentechnologie abhängen.

Eine Diode ist ein p-n-Übergang, gebildet durch einen p-Typ- und einen n-Typ-Halbleiter. An der Grenzfläche bildet sich eine Raumladungszone mit einem Eigenfeld aus. In elektrischem Gleichgewicht gleichen sich die Drift- und Diffusionsströme aus. Bei Vorwärtsbias verstärken sich die externen und das Eigenfeld, wodurch die Leitfähigkeit zunimmt. Bei Rückwärtsbias wird ein Sättigungsstrom  $I_0$  erzeugt, der von der angelegten Spannung unabhängig ist.

### 1. Vorwärtscharakteristik

Bei Anlegen einer Vorwärtsspannung bleibt der Strom anfangs nahezu null, da die Spannung die Raumladungszone nicht überwinden kann. Dieser Bereich wird als Sperrbereich bezeichnet. Erst bei Überschreitung dieser Spannung beginnt der Strom stark anzusteigen.

### 2. Rückwärtscharakteristik

Bei Anlegen einer Rückwärtsspannung bleibt der Stromfluss gering und wird als Sättigungs- oder Leckstrom bezeichnet, der stark temperaturabhängig ist.

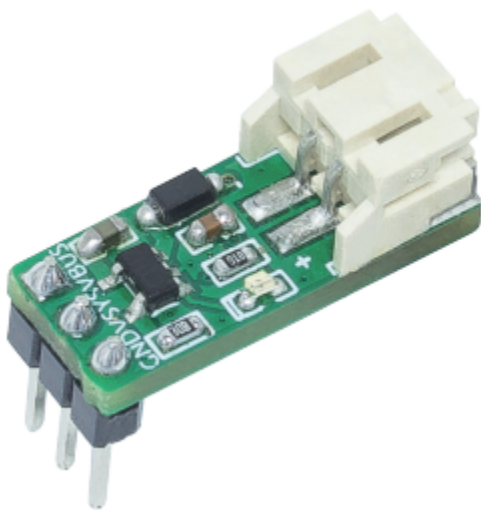
### 3. Durchbruch

Überschreitet die Rückwärtsspannung einen bestimmten Wert, steigt der Stromfluss schlagartig an, was als elektrischer Durchbruch bekannt ist. Die dafür erforderliche Spannung wird als Durchbruchspannung bezeichnet.

Frühe Dioden bestanden aus „Katzenfaden“-Kristallen und Vakuumröhren. Heutige Dioden verwenden Halbleitermaterialien wie Silizium oder Germanium.

- [P-N-Übergang - Wikipedia](#)
- [Diode - Wikipedia](#)

## 2.7 Li-Po-Lademodul



Dies ist ein Li-Po-Lademodul, konzipiert für Raspberry Pi Pico/Pico H/Pico W. Einfach einstecken und den Pico auf dem Steckbrett wie unten gezeigt positionieren. Danach die Batterie am anderen Ende anschließen und schon kann es losgehen.

Wenn das Pico W über ein USB-Kabel an einen Computer oder eine Steckdose angeschlossen wird, leuchtet die Kontrollleuchte auf dem Li-Po-Lademodul auf. Dies signalisiert, dass die Batterie gleichzeitig geladen wird. Wird das USB-Kabel entfernt, wird das Pico W von der Batterie versorgt, sodass Ihr Projekt weiterhin läuft.

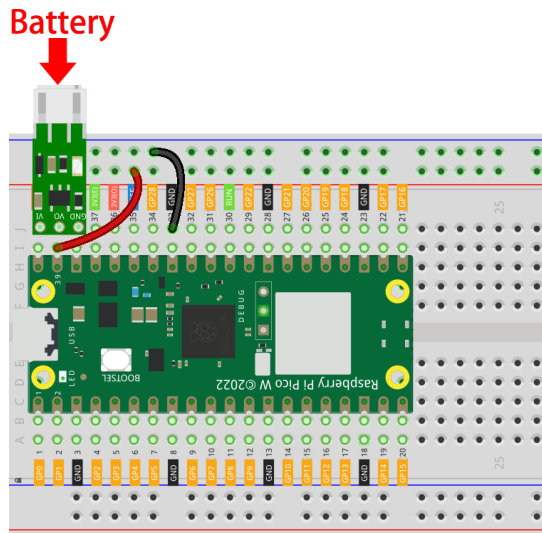
---

**Bemerkung:** Bei einigen leistungsschwachen Computern kann es vorkommen, dass das Pico W nicht erkannt wird, wenn es mit dem angeschlossenen Lademodul an den Computer angeschlossen wird.

Der Grund dafür ist, dass die USB-Port-Spannung beim Laden der Batterie abfällt, was dazu führt, dass die Stromversorgung des Pico W nicht ausreicht, um vom Computer erkannt zu werden.

In diesem Fall muss das Li-Po-Lademodul entfernt und das Pico W erneut eingesteckt werden.

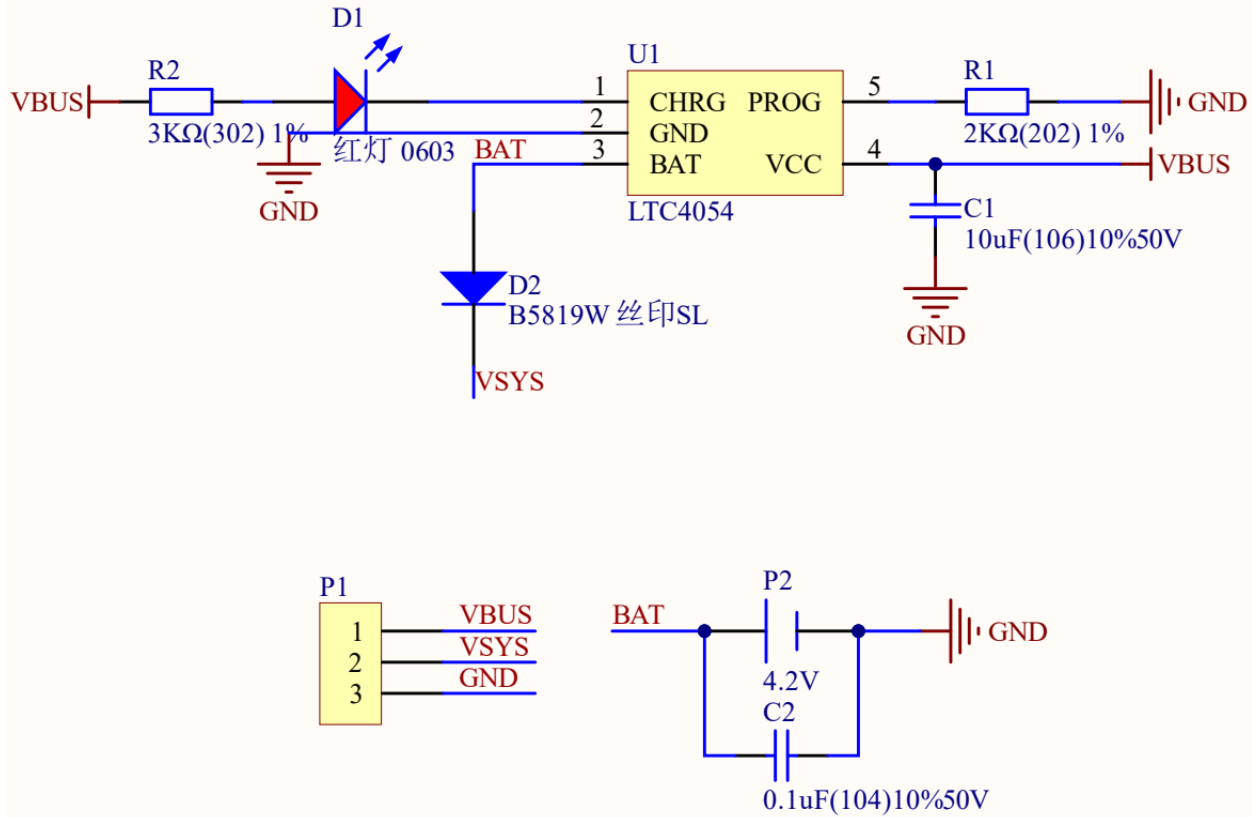
---



### Eigenschaften

- Eingangsspannung: 5V
- Ausgangsspannung: 3,3V
- Größe: 20mmx7mm
- Schnittstellenmodell: PH2.0
- Es gibt einen passenden 1A-Batteriehalter sowie einen 800mAh 18650, der zusammen verwendet werden kann.

### Schaltplan



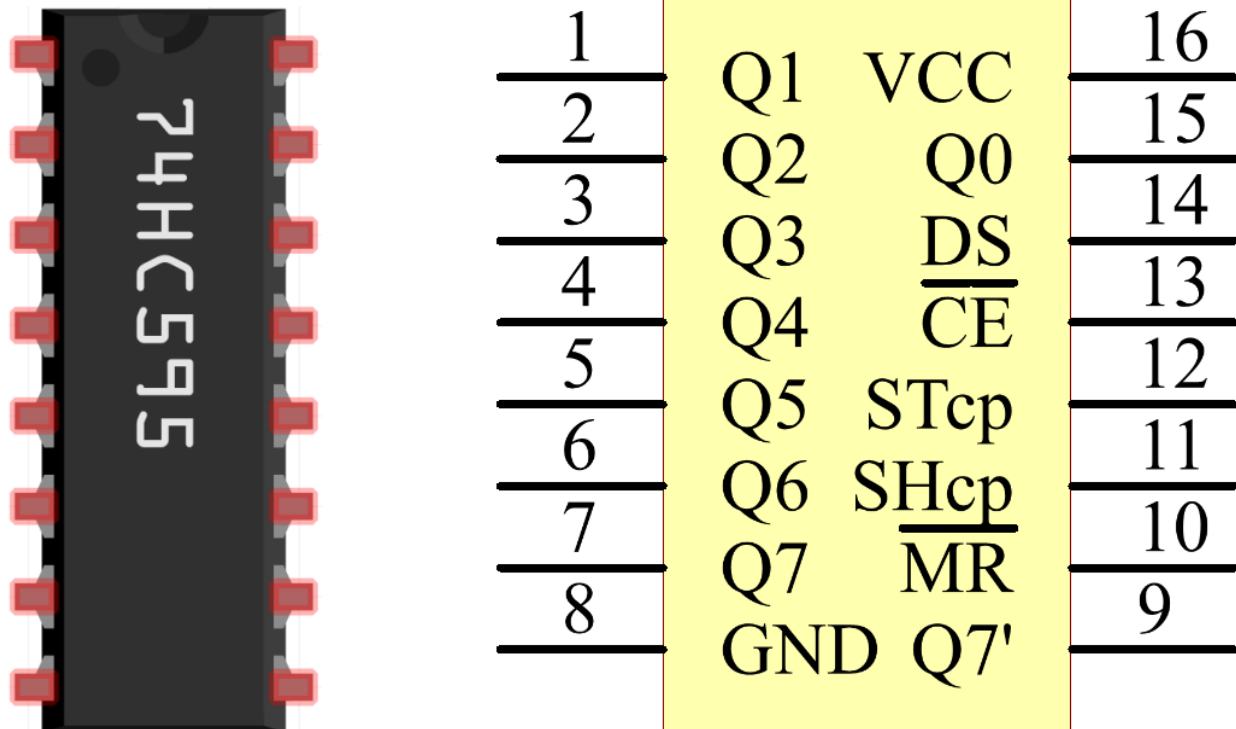
## Chips

### 2.8 74HC595



Der 74HC595 besteht aus einem 8-Bit-Schieberegister und einem Speicherregister mit dreizuständigen parallelen Ausgängen. Er wandelt serielle Eingaben in parallele Ausgaben um, sodass IO-Ports eines MCU gespart werden können.

- Bei hohem Pegel an MR (Pin10) und niedrigem Pegel an OE (Pin13) wird die Daten am steigenden Flanken von SHcp eingelesen und durchlaufen das Speicherregister über die steigende Flanke von SHcp.
- Wenn die beiden Taktgeber miteinander verbunden sind, liegt das Schieberegister immer einen Taktimpuls vor dem Speicherregister.
- Im Speicherregister gibt es einen seriellen Schieberegistereingangspin (Ds), einen seriellen Ausgangspin (Q) und eine asynchrone Reset-Taste (niedriger Pegel).
- Das Speicherregister gibt einen Bus mit parallelem 8-Bit und in drei Zuständen aus.
- Ist OE aktiviert (niedriger Pegel), werden die Daten im Speicherregister zum Bus (Q0 ~ Q7) ausgegeben.
- [74HC595 Datenblatt](#)



Pins des 74HC595 und ihre Funktionen:

- **Q0-Q7**: 8-Bit-parallele Daten-Ausgangspins, direkt für die Steuerung von 8 LEDs oder 8 Pins einer 7-Segment-Anzeige geeignet.
- **Q7'**: Serieller Ausgangspin, verbunden mit DS eines weiteren 74HC595 zur seriellen Verknüpfung mehrerer 74HC595.
- **MR**: Reset-Pin, aktiv bei niedrigem Pegel.
- **SHcp**: Zeitsequenz-Eingang des Schieberegisters. An der steigenden Flanke rückt die Daten im Schieberegister jeweils um ein Bit vor, d.h. Daten in Q1 bewegen sich zu Q2 usw. An der fallenden Flanke bleiben die Daten unverändert.
- **STcp**: Zeitsequenz-Eingang des Speicherregisters. An der steigenden Flanke wandern die Daten aus dem Schieberegister ins Speicherregister.
- **OE**: Enable-Pin für den Ausgang, aktiv bei niedrigem Pegel.
- **DS**: Serieller Dateneingangspin.
- **VCC**: Positive Versorgungsspannung.
- **GND**: Masse.

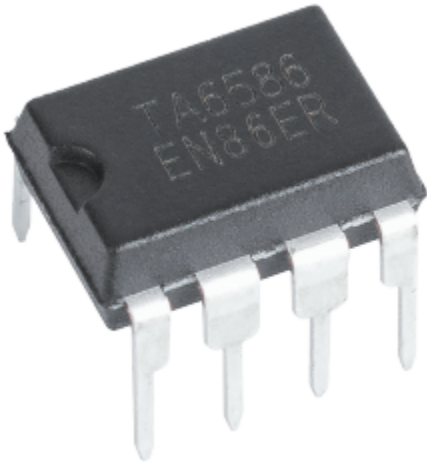
### Beispiel

- [5.1 Mikrochip - 74HC595](#) (Für MicroPython-Anwender)
- [5.2 Nummernanzeige](#) (Für MicroPython-Anwender)
- [5.3 Zeitmesser](#) (Für MicroPython-Anwender)
- [5.4 8x8 Pixel-Grafik](#) (Für MicroPython-Anwender)
- [7.4 Personen Zähler](#) (Für MicroPython-Anwender)
- [7.5 SPIEL - 10 Sekunden](#) (Für MicroPython-Anwender)



- *7.6 Ampel* (Für MicroPython-Anwender)
- *7.12 Digitaler Wasserwaage* (Für MicroPython-Anwender)
- *5.1 Mikrochip - 74HC595* (Für Arduino-Anwender)
- *5.2 - Zahlenanzeige* (Für Arduino-Anwender)
- *5.3 - Zeitmesser* (Für Arduino-Anwender)
- *5.4 - 8x8 Pixelgrafik* (Für Arduino-Anwender)

## 2.9 TA6586 - Motorsteuerungs-Chip



Der TA6586 ist ein monolithischer IC, der zur Steuerung bidirektionaler Gleichstrommotoren entwickelt wurde. Er verfügt über zwei Logikeingangspins zur Kontrolle der Fahrtrichtung, vorwärts und rückwärts. Der Schaltkreis zeichnet sich durch gute Störfestigkeit, geringen Ruhestrom und niedrigen Ausgangssättigungsdruckabfall aus. Ein integrierte Klemmdiode kehrt den Einfluss des Freisetzens des induktiven Laststroms um, wodurch der IC beim Steuern von Relais, Gleichstrommotoren, Schrittmotoren oder beim Einsatz in Schaltnetzteilen sicher und zuverlässig ist. Der TA6586 eignet sich für Spielzeugfahrzeuge, ferngesteuerte Flugzeugantriebe, automatische Ventilmotoren, elektromagnetische Schlossantriebe, Präzisionsinstrumente und andere Schaltungen.

### Eigenschaften

- Niedriger Ruhestrom: 2uA
- Weiter Versorgungsspannungsbereich
- Integrierte Bremsfunktion
- Thermischer Überlastschutz
- Überstrombegrenzung und Kurzschlusschutz
- DIP8 bleifreies Gehäuse.

### Pin-Funktion

Pin NO	Name	Function
1	BI	Backward input
2	FI	Forward input
3	GND	Ground
4	Vcc	Vcc
5, 6	FO	Forward output
7, 8	BO	Backward output

#### Eingangswahrheitstabelle

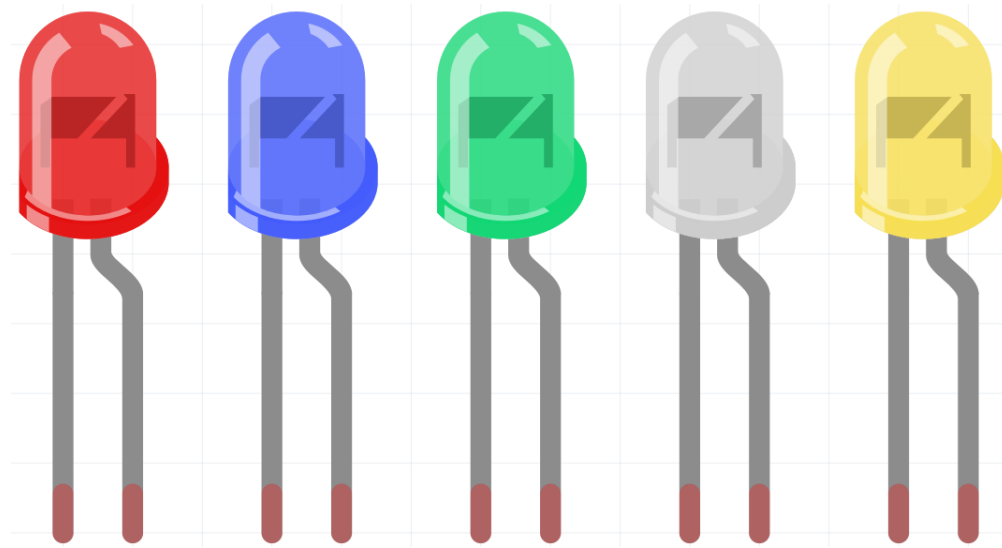
2pin Finput	1pin Binput	5,6pin Foutput	7,8pin Boutput
H	L	H	L
L	H	L	H
H	H	L	L
L	L	Open	Open

#### Beispiel

- [3.5 Kleiner Ventilator](#) (Für MicroPython-Anwender)
- [3.5 - Kleiner Ventilator](#) (Für Arduino-Anwender)
- [3.6 Pumpensteuerung](#) (Für MicroPython-Anwender)
- [3.6 - Pumpensteuerung](#) (Für Arduino-Anwender)
- [2.12 Intelligenter Ventilator](#) (Für Piper Make-Anwender)

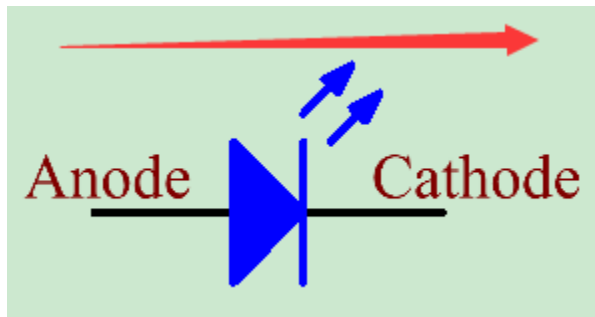
#### Anzeigeelemente

## 2.10 LED



Halbleiterlichtemittierende Dioden sind Bauteile, die elektrische Energie durch PN-Übergänge in Lichtenergie umwandeln können. Je nach Wellenlänge können sie in Laserdioden, infrarotemittierende Dioden und sichtbare lichtemittierende Dioden eingeteilt werden, wobei die letztere allgemein als lichtemittierende Diode (LED) bekannt ist.

Dioden haben eine unidirektionale Leitfähigkeit, sodass der Stromfluss so verläuft, wie im Schaltzeichensymbol durch den Pfeil angezeigt wird. Das Anodenende sollte positiv und das Kathodenende negativ polarisiert werden. Dann leuchtet die LED.



Eine LED hat zwei Anschlüsse. Der längere ist die Anode und der kürzere die Kathode. Achten Sie darauf, sie nicht umgekehrt anzuschließen. LEDs haben eine festgelegte Vorwärtsspannung. Sie können nicht direkt an eine Schaltung angeschlossen werden, da die Versorgungsspannung diese Vorwärtsspannung übersteigen und die LED beschädigen könnte. Die Vorwärtsspannung von roten, gelben und grünen LEDs beträgt 1,8 V, während die von weißen LEDs 2,6 V beträgt. Die meisten LEDs können einen maximalen Strom von 20 mA vertragen, daher sollte ein strombegrenzender Widerstand in Reihe geschaltet werden.

Die Formel für den Widerstandswert lautet:

$$R = (V_{\text{supply}} - V_D) / I$$

Dabei steht **R** für den Widerstandswert des strombegrenzenden Widerstands, **V<sub>supply</sub>** für die Versorgungsspannung, **V<sub>D</sub>** für den Spannungsabfall und **I** für den Arbeitsstrom der LED.

Hier finden Sie eine detaillierte Einführung in die LED: [LED - Wikipedia](#).

### Beispiel

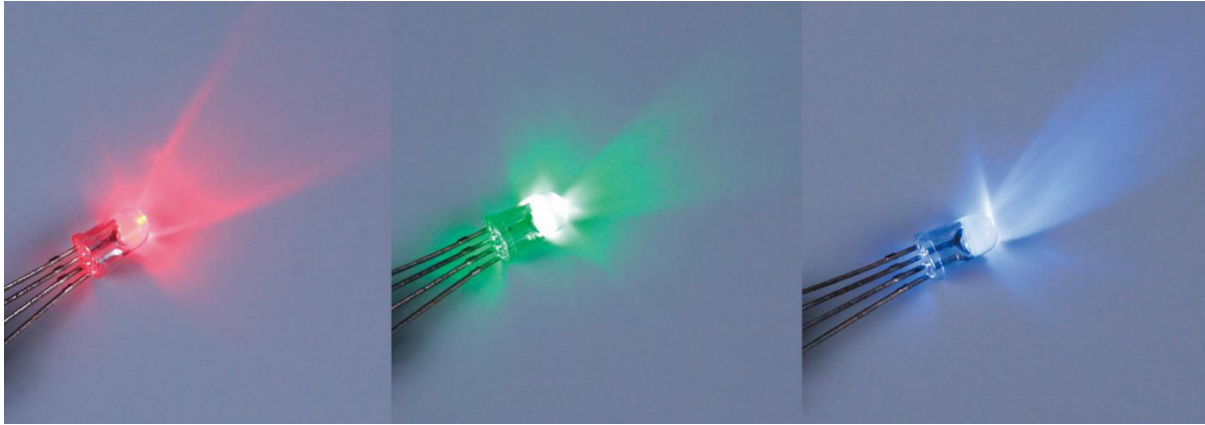
- [2.1 Hallo, LED!](#) (Für MicroPython-Nutzer)

- [2.3 Abklingende LED](#) (Für MicroPython-Nutzer)
- [7.3 Alarmanlagenlampe](#) (Für MicroPython-Nutzer)
- [7.6 Ampel](#) (Für MicroPython-Nutzer)
- [7.10 Einparkhilfe](#) (Für MicroPython-Nutzer)
- [2.1 - Hallo, LED!](#) (Für Arduino-Nutzer)
- [2.3 - Verblässende LED](#) (Für Arduino-Nutzer)
- [2.1 LED Blinken Lassen](#) (Für Piper Make-Nutzer)
- [2.2 Taster](#) (Für Piper Make-Nutzer)
- [2.3 Serviceklingel](#) (Für Piper Make-Nutzer)
- [2.11 Rückfahrssystem](#) (Für Piper Make-Nutzer)
- [2.13 Reaktionsspiel](#) (Für Piper Make-Nutzer)

## 2.11 RGB-LED

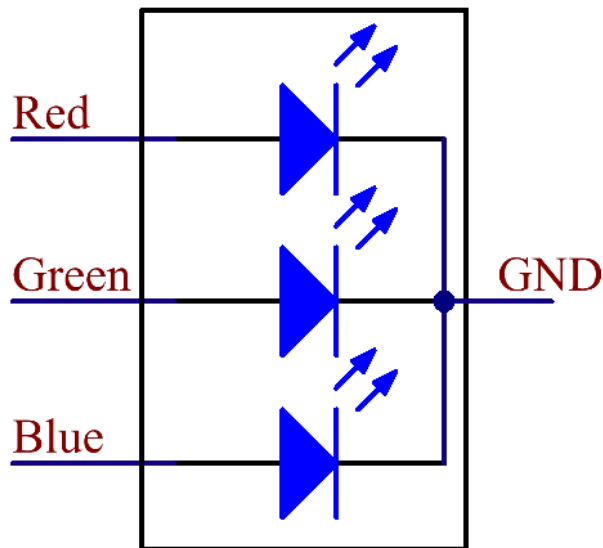


RGB-LEDs emittieren Licht in verschiedenen Farben. Eine RGB-LED kombiniert drei LEDs in den Farben Rot, Grün und Blau in einem transparenten oder halbtransparenten Kunststoffgehäuse. Durch Anpassung der Eingangsspannung der drei Pins und deren Überlagerung können statistisch bis zu 16.777.216 verschiedene Farben erzeugt werden.

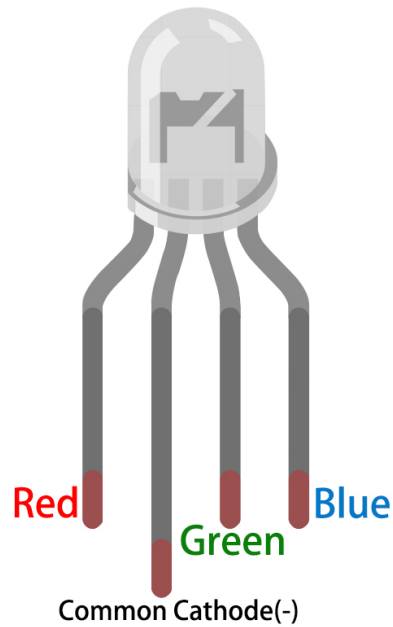


RGB-LEDs können in Gemeinsam-Anode und Gemeinsam-Kathode unterteilt werden. In diesem Bausatz wird die letztere verwendet. Der Begriff **Gemeinsam-Kathode**, oder GK, bedeutet, dass die Kathoden der drei LEDs verbunden sind. Sobald sie mit GND verbunden und die drei Pins eingesteckt sind, leuchtet die LED in der entsprechenden Farbe auf.

Das Schaltungssymbol ist als Abbildung dargestellt.



Eine RGB-LED hat 4 Pins: Der längste Pin ist der Gemeinsam-Kathode-Pin, der in der Regel mit GND verbunden ist. Der linke Pin neben dem längsten Pin ist Rot, und die beiden Pins rechts davon sind Grün und Blau.



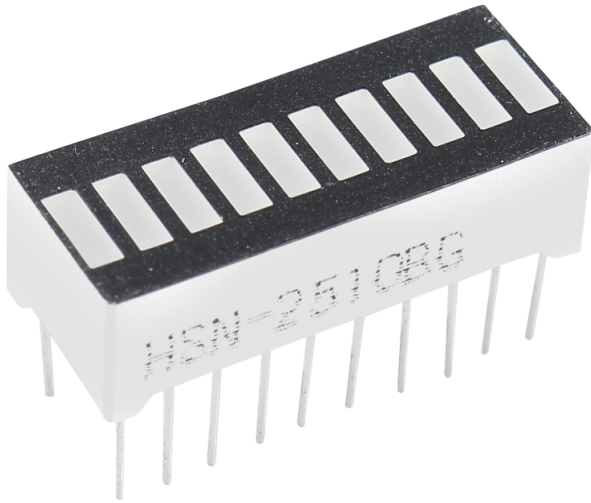
#### Funktionen

- Farbe: Drei-Farben (Rot/Grün/Blau)
- Gemeinsam-Kathode
- 5mm Klare Runde Linse
- Vorwärtsspannung: Rot: DC 2,0 - 2,2V; Blau&Grün: DC 3,0 - 3,2V (IF=20mA)
- 0,06 Watt DIP RGB-LED
- Leuchtkraft bis zu +20% heller
- Betrachtungswinkel: 30°

#### Beispiel

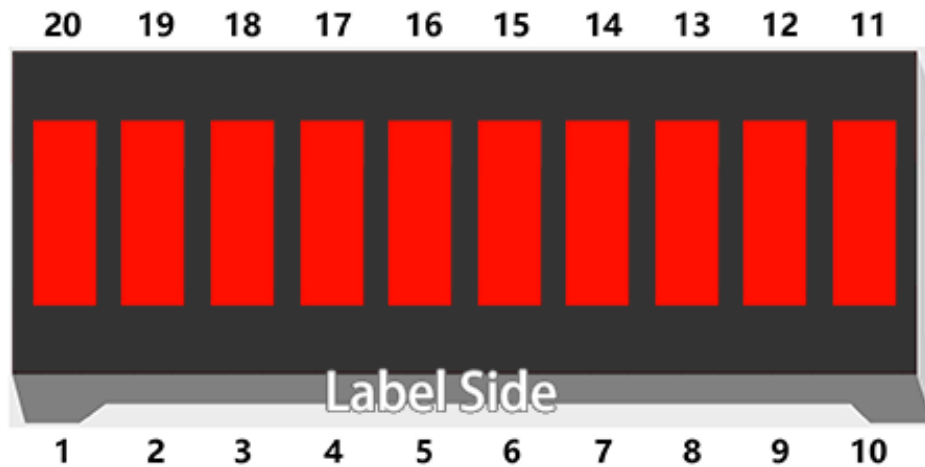
- [2.4 Farbenfrohes Licht](#) (Für MicroPython-Nutzer)
- [7.9 Frucht-Klavier](#) (Für MicroPython-Nutzer)
- [2.4 - Farbenfrohes Licht](#) (Für Arduino-Nutzer)
- [2.4 Regenbogenlicht](#) (Für Piper Make-Nutzer)

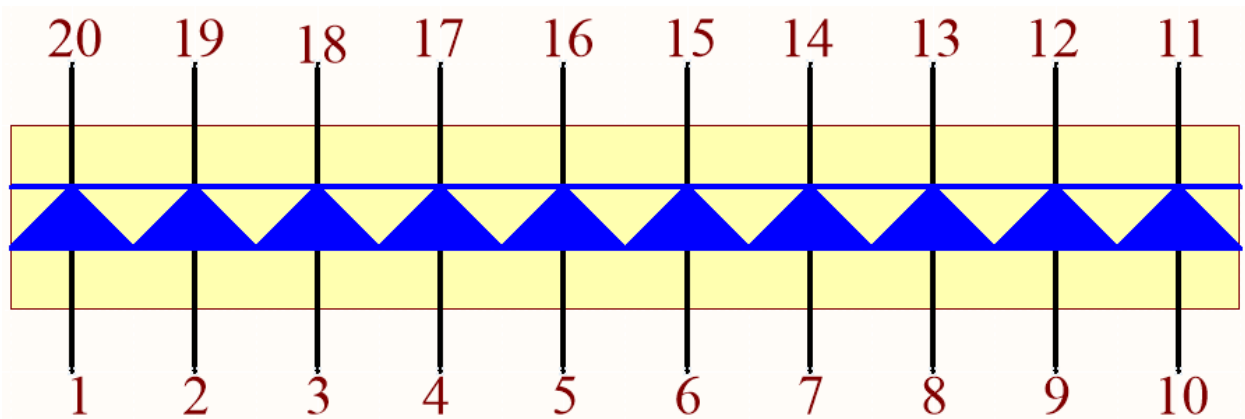
## 2.12 LED-Balkendiagramm



Ein LED-Balkendiagramm ist ein LED-Array, das zur Verbindung mit elektronischen Schaltungen oder Mikrocontrollern verwendet wird. Es ist genauso einfach, ein LED-Balkendiagramm mit der Schaltung zu verbinden, wie 10 einzelne LEDs mit 10 Ausgangspins zu verbinden. In der Regel können LED-Balkendiagramme als Batteriestandsanzeige, in Audiogeräten oder in industriellen Steuerungspanels eingesetzt werden. Darüber hinaus gibt es zahlreiche weitere Anwendungsmöglichkeiten für LED-Balkendiagramme.

Im Folgenden ist das interne Schaltbild des LED-Balkendiagramms dargestellt. Allgemein gilt: Die Seite mit der Beschriftung ist die Anode, die gegenüberliegende Seite die Kathode.

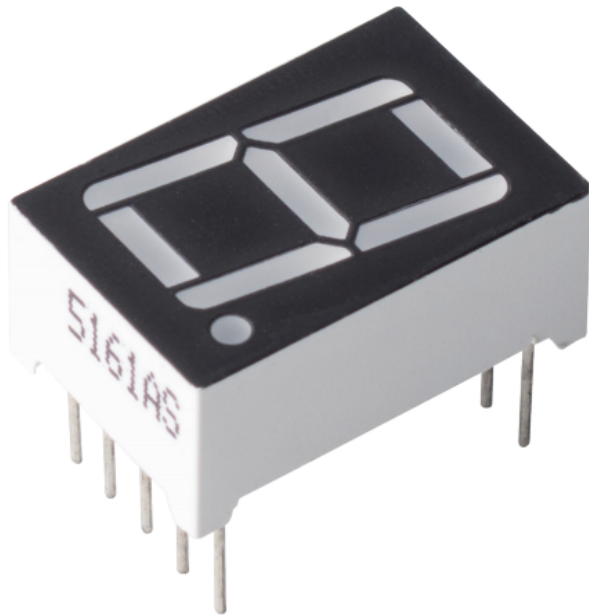




### Beispiel

- *2.2 Anzeige des Pegels* (Für MicroPython-Nutzer)
- *2.2 - Pegelanzeige* (Für Arduino-Nutzer)
- *2.8 Lichtintensitätsanzeige* (Für Piper Make-Nutzer)

## 2.13 7-Segment-Anzeige

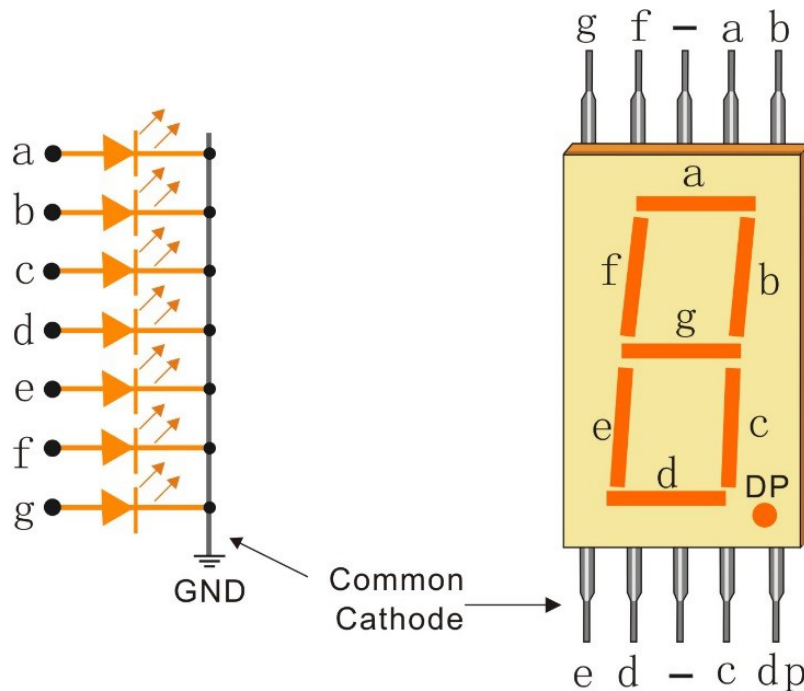


Eine 7-Segment-Anzeige ist ein acht-förmiges Bauteil, das aus 7 LEDs besteht. Jede dieser LEDs wird als Segment bezeichnet - sobald es aktiviert ist, bildet ein Segment einen Teil einer darzustellenden Zahl.

Es gibt zwei Arten von Pin-Verbindungen: Gemeinsame Kathode (Common Cathode, CC) und Gemeinsame Anode (Common Anode, CA). Wie der Name bereits verrät, sind bei einer CC-Anzeige alle Kathoden der 7 LEDs verbunden, während bei einer CA-Anzeige alle Anoden der 7 Segmente miteinander verbunden sind.

In diesem Bausatz verwenden wir die 7-Segment-Anzeige mit gemeinsamer Kathode, hier ist das entsprechende elektronische Symbol.





Jedes der LEDs in der Anzeige hat eine positionelle Bezeichnung und einen der Anschlusspins, der aus dem rechteckigen Kunststoffgehäuse herausgeführt wird. Diese LED-Pins sind von „a“ bis „g“ beschriftet und repräsentieren jeweils eine einzelne LED. Die anderen LED-Pins sind zusammengeführt und bilden einen gemeinsamen Pin. Durch die Vorwärtvorspannung der entsprechenden Pins der LED-Segmente in einer bestimmten Reihenfolge leuchten einige Segmente auf, während andere gedimmt bleiben, sodass der entsprechende Charakter auf der Anzeige dargestellt wird.

- [7-Segment-Anzeige - Wikipedia](#)

### Anzeigecodes

Um Ihnen zu helfen, zu verstehen, wie 7-Segment-Anzeigen (Gemeinsame Kathode) Zahlen darstellen, haben wir die folgende Tabelle erstellt. Die Zahlen sind die Zahlen von 0 bis F, die auf der 7-Segment-Anzeige dargestellt werden; (DP) GFEDCBA bezieht sich auf das jeweilige LED-Set, das auf 0 oder 1 gesetzt ist. Zum Beispiel bedeutet 00111111, dass DP und G auf 0 gesetzt sind, während die anderen auf 1 gesetzt sind. Daher wird die Zahl 0 auf der 7-Segment-Anzeige angezeigt, während der HEX-Code der entsprechenden Hexadezimalzahl entspricht.

Tab. 2: Glyph Code

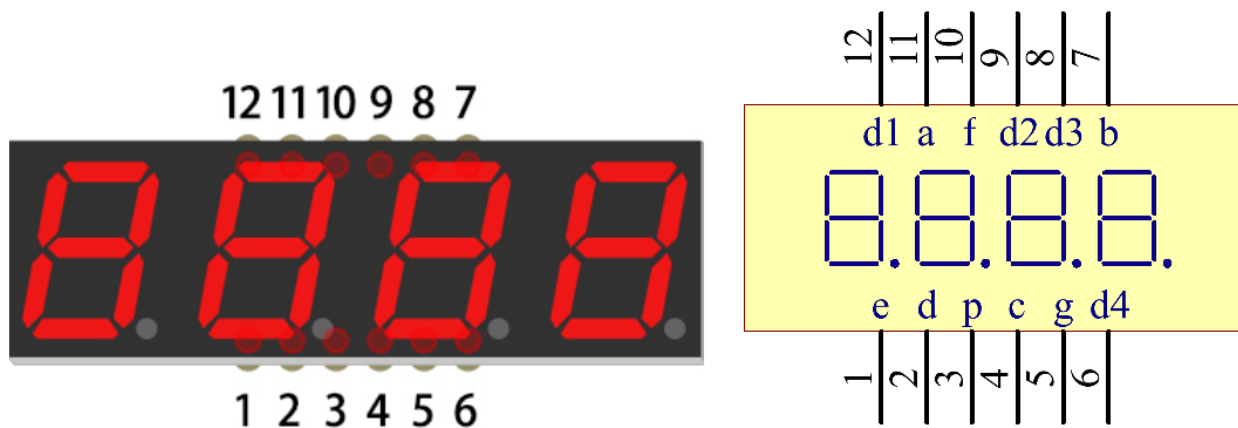
Zahlen	Binärer Code	Hex-Code
0	00111111	0x3f
1	00000110	0x06
2	01011011	0x5b
3	01001111	0x4f
4	01100110	0x66
5	01101101	0x6d
6	01111101	0x7d
7	00000111	0x07
8	01111111	0x7f
9	01101111	0x6f
A	01110111	0x77
B	01111100	0x7c
C	00111001	0x39
D	01011110	0x5e
E	01111001	0x79
F	01110001	0x71

**Beispiel**

- 5.2 *Nummernanzeige* (Für MicroPython-Nutzer)
- 5.2 - *Zahlenanzeige* (Für Arduino-Nutzer)

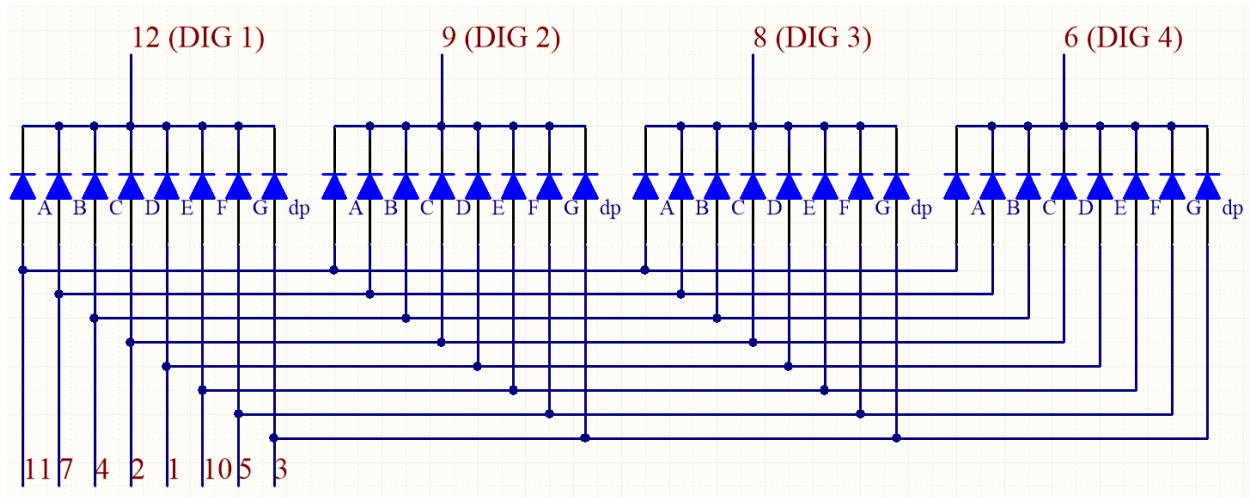
## 2.14 4-stellige 7-Segment-Anzeige

Eine 4-stellige 7-Segment-Anzeige besteht aus vier eigenständigen 7-Segment-Displays.



Jedes der 7-Segment-Displays in der 4-stelligen Anzeige arbeitet unabhängig voneinander. Auf dem Prinzip der Trägheit des menschlichen Auges basierend, werden die Zeichen auf jedem 7-Segment schnell nacheinander angezeigt, um einen durchgehenden Text darzustellen.

Zum Beispiel: Wenn „1234“ angezeigt wird, erscheint die „1“ auf dem ersten 7-Segment, während die „234“ nicht dargestellt werden. Nach einer kurzen Zeit wird die „2“ auf dem zweiten 7-Segment angezeigt, während die anderen inaktiv bleiben. Dieser Vorgang wiederholt sich sehr schnell (typischerweise in 5 ms) für alle vier Displays. Aufgrund des Nachleuchteffekts und der visuellen Trägheit nehmen wir alle vier Zeichen gleichzeitig wahr.



### Anzeigencodes

Um das Verständnis für die Darstellung von Zahlen auf 7-Segment-Anzeigen (Common Cathode) zu erleichtern, haben wir die folgende Tabelle erstellt. Die Zahlen repräsentieren die auf dem 7-Segment-Display dargestellten Werte von 0-F; (DP) GFEDCBA bezieht sich auf die entsprechenden LEDs, die auf 0 oder 1 gesetzt sind. Zum Beispiel bedeutet 00111111, dass DP und G auf 0 und alle anderen auf 1 gesetzt sind. Daraus ergibt sich, dass die Zahl 0 auf dem Display angezeigt wird, während der HEX-Code der entsprechenden hexadezimalen Nummer entspricht.

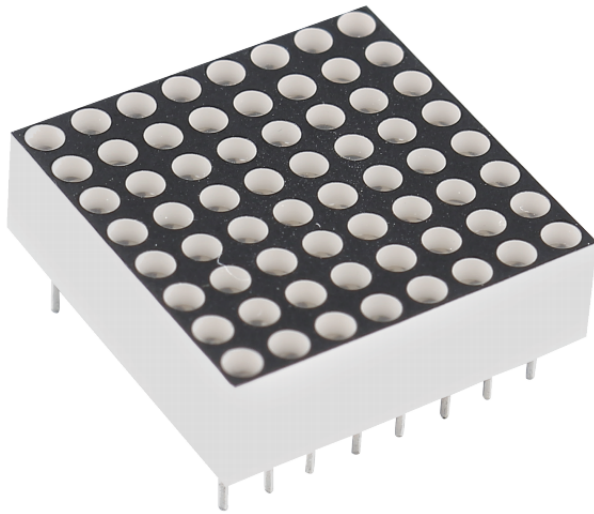
Tab. 3: Glyph Code

Zahlen	Binärcode	Hex-Code
0	00111111	0x3f
1	00000110	0x06
2	01011011	0x5b
3	01001111	0x4f
4	01100110	0x66
5	01101101	0x6d
6	01111101	0x7d
7	00000111	0x07
8	01111111	0x7f
9	01101111	0x6f
A	01110111	0x77
B	01111100	0x7c
C	00111001	0x39
D	01011110	0x5e
E	01111001	0x79
F	01110001	0x71

### Beispiel

- [5.3 Zeitmesser](#) (Für MicroPython-Nutzer)
- [7.4 Personen Zähler](#) (Für MicroPython-Nutzer)
- [7.5 SPIEL - 10 Sekunden](#) (Für MicroPython-Nutzer)
- [7.6 Ampel](#) (Für MicroPython-Nutzer)
- [5.3 - Zeitmesser](#) (Für Arduino-Nutzer)

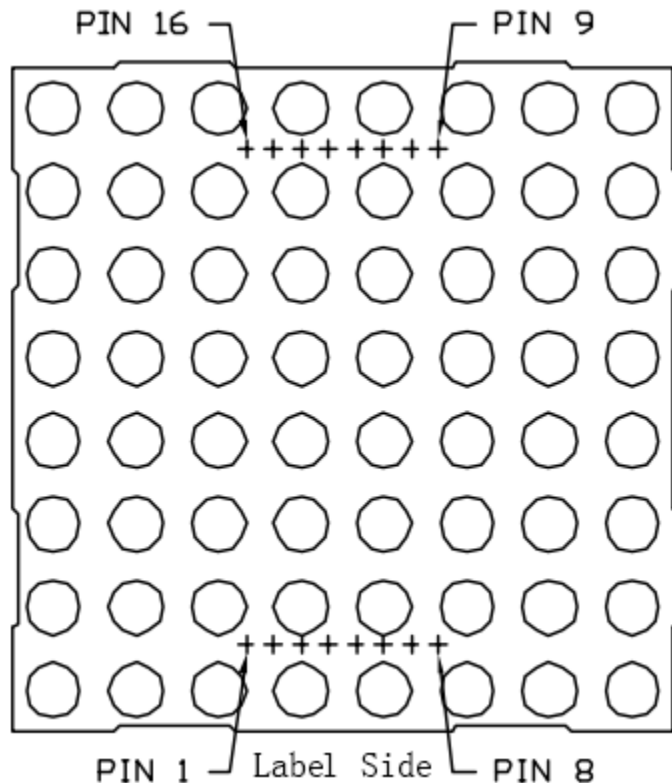
## 2.15 LED-Punktmatrix



Generell lässt sich die LED-Punktmatrix in zwei Typen unterteilen: Gemeinsame Kathode (CC) und gemeinsame Anode (CA). Optisch sehen beide Typen ähnlich aus, der Unterschied liegt jedoch im Inneren. Dies lässt sich durch einen Test feststellen. Im vorliegenden Kit wird ein CA-Modell verwendet, das seitlich mit 788BS beschriftet ist.

Siehe dazu die untenstehende Abbildung. Die Pins sind an den beiden Enden der Rückseite angeordnet. Orientieren Sie sich an der beschrifteten Seite: Die Pins an diesem Ende sind die Pins 1-8, am anderen Ende sind es die Pins 9-16.

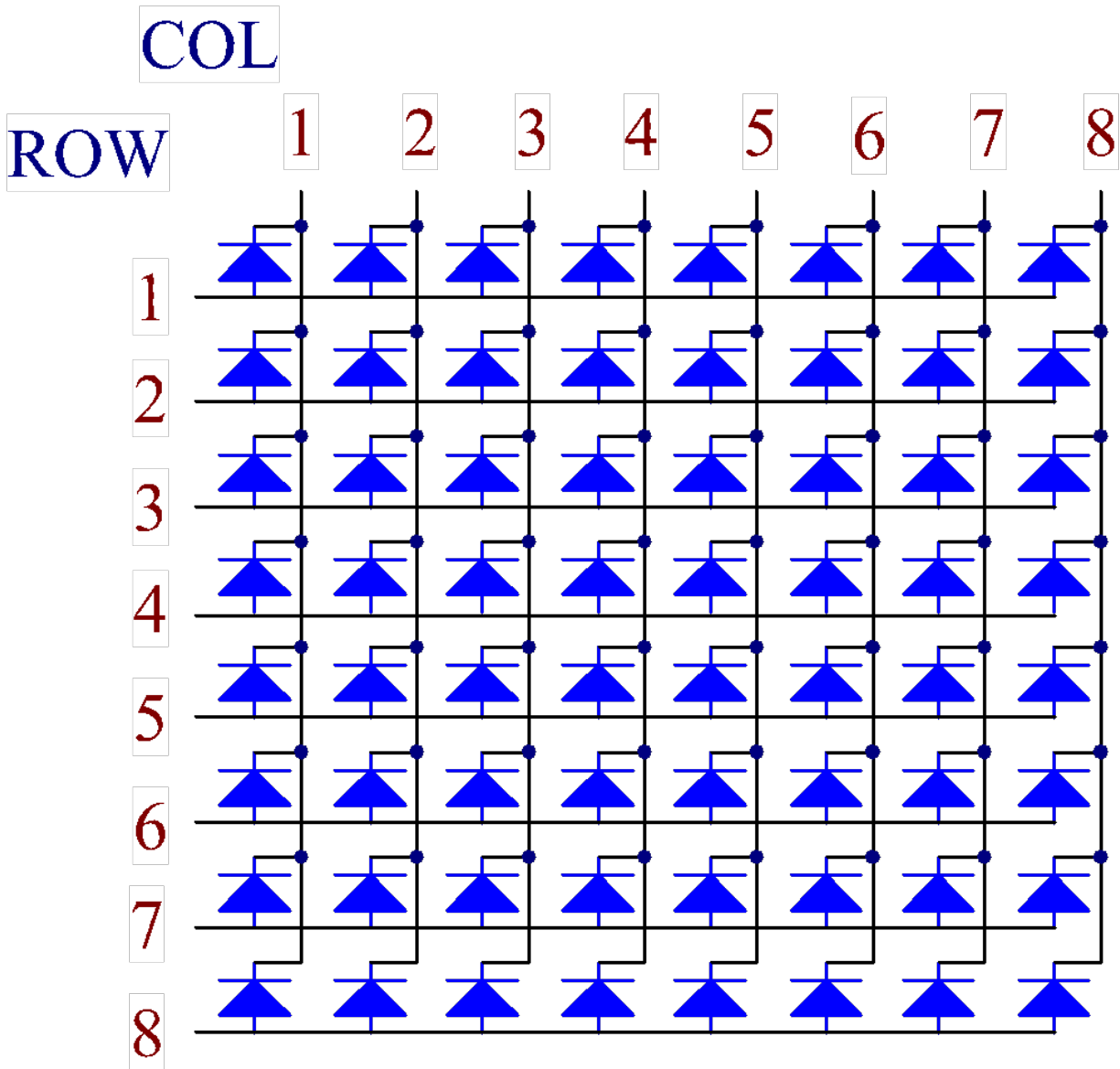
Die äußere Ansicht:



Die folgenden Abbildungen zeigen den internen Aufbau. In einer CA-LED-Punktmatrix repräsentiert die Zeile (ROW)

die Anode der LED, während die Spalte (COL) die Kathode ist; bei einer CC-Matrix ist es umgekehrt. Gemeinsam ist beiden Typen: Die Pins 13, 3, 4, 10, 6, 11, 15 und 16 sind jeweils COL, während die Pins 9, 14, 8, 12, 1, 7, 2 und 5 alle ROW sind. Möchten Sie die erste LED in der linken oberen Ecke einschalten, setzen Sie bei einer CA-Matrix Pin 9 auf High und Pin 13 auf Low; bei einer CC-Matrix setzen Sie Pin 13 auf High und Pin 9 auf Low. Um die gesamte erste Spalte aufzuhellen, setzen Sie bei CA Pin 13 auf Low und die ROW-Pins 9, 14, 8, 12, 1, 7, 2 und 5 auf High; bei CC setzen Sie Pin 13 auf High und die ROW-Pins auf Low. Die nachfolgenden Abbildungen sollten zur weiteren Veranschaulichung dienen.

Die innere Ansicht:



Zuordnung der Pinnummern zu den oben genannten Reihen und Spalten:

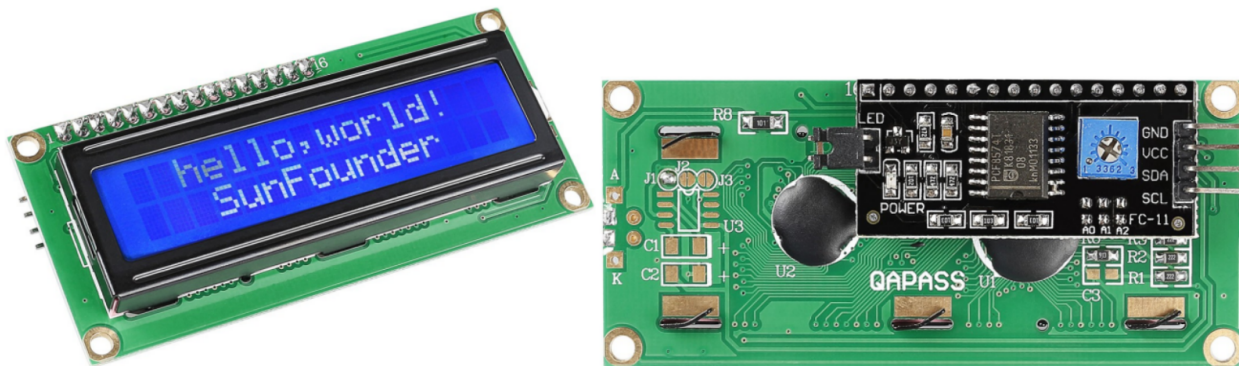
COL	1	2	3	4	5	6	7	8
Pin No.	13	3	4	10	6	11	15	16
ROW	1	2	3	4	5	6	7	8
Pin No.	9	14	8	12	1	7	2	5

Darüber hinaus werden hier zwei 74HC595-Chips verwendet. Einer steuert die Reihen der LED-Punktmatrix, der andere die Spalten.

### Beispiel

- [5.4 8x8 Pixel-Grafik](#) (Für MicroPython-Nutzer)
- [7.12 Digitaler Wasserwaage](#) (Für MicroPython-Nutzer)
- [5.4 - 8x8 Pixelgrafik](#) (Für Arduino-Nutzer)

## 2.16 I2C LCD1602



- **GND:** Masse
- **VCC:** Spannungsversorgung, 5V.
- **SDA:** Serielle Datenleitung. Über einen Pull-up-Widerstand mit VCC verbinden.
- **SCL:** Serielle Taktleitung. Über einen Pull-up-Widerstand mit VCC verbinden.

Wie allgemein bekannt ist, bereichern LCDs und andere Anzeigen zwar die Mensch-Maschine-Interaktion, haben jedoch eine gemeinsame Schwäche. Wenn sie an einen Controller angeschlossen werden, werden mehrere IO-Ports des Controllers belegt, der über nicht viele externe Anschlüsse verfügt. Dies schränkt auch andere Funktionen des Controllers ein.

Daher wurde das LCD1602 mit einem I2C-Modul entwickelt, um dieses Problem zu lösen. Das I2C-Modul verfügt über einen integrierten PCF8574 I2C-Chip, der I2C-Serien-Daten in parallele Daten für das LCD-Display umwandelt.

- [PCF8574 Datenblatt](#)

### I2C-Adresse

Die Standardadresse ist im Grunde 0x27, in einigen Fällen kann sie jedoch auch 0x3F sein.

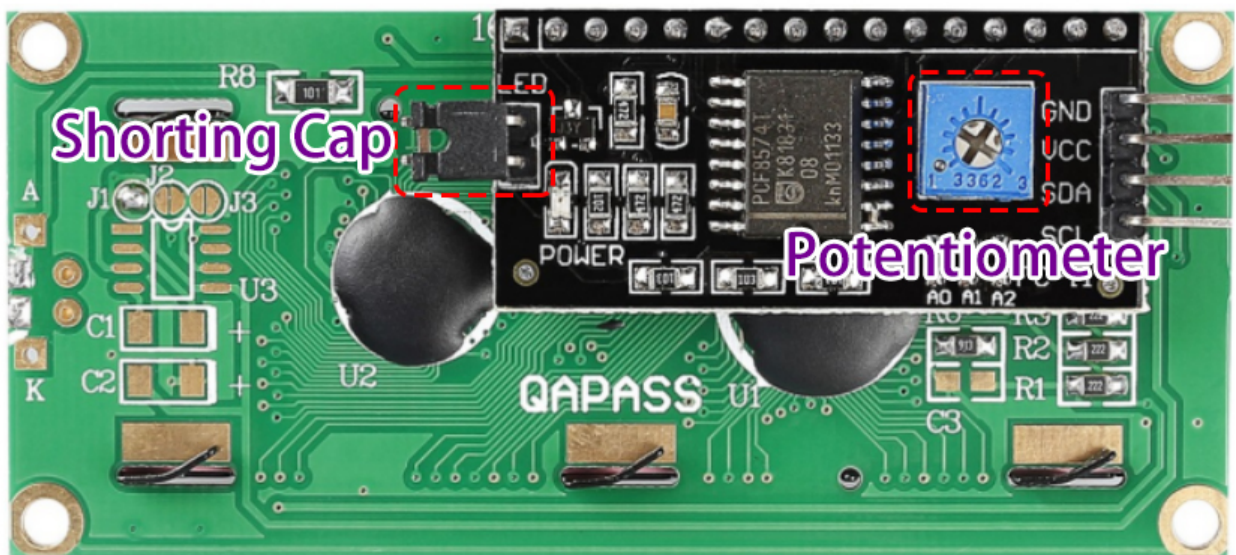
Beispielhaft für die Standardadresse 0x27 lässt sich die Geräteadresse durch das Kurzschließen der A0/A1/A2-Pads modifizieren; im Ausgangszustand sind A0/A1/A2 auf 1, und wenn das Pad kurzgeschlossen wird, sind A0/A1/A2 auf 0.

## Slave Address

Slave Address								
0	0	1	0	0	A2	A1	A0	
0	0	1	0	0	1	1	1	0x27
0	0	1	0	0	1	1	0	0x26
0	0	1	0	0	1	0	1	0x25
0	0	1	0	0	0	1	1	0x23
.....								
0	0	1	0	0	0	0	0	0x20

### Hintergrundbeleuchtung/Kontrast

Die Hintergrundbeleuchtung kann durch Aufsetzen einer Jumper-Kappe aktiviert werden, zum Deaktivieren einfach die Jumper-Kappe abziehen. Das blaue Potentiometer auf der Rückseite dient zur Kontrastanpassung (Verhältnis der Helligkeit zwischen dem hellsten Weiß und dem dunkelsten Schwarz).



- **Kurzschlusskappe:** Die Hintergrundbeleuchtung kann mit dieser Kappe aktiviert werden, zum Deaktivieren



einfach diese Kappe abziehen.

- **Potentiometer:** Dient zur Kontrasteinstellung (der Klarheit der angezeigten Texte), die im Uhrzeigersinn erhöht und gegen den Uhrzeigersinn verringert wird.

### Beispiel

- [3.4 Flüssigkristallanzeige](#) (Für MicroPython-Nutzer)
- [7.2 Raumtemperaturmessgerät](#) (Für MicroPython-Nutzer)
- [7.7 Zahlenraten](#) (Für MicroPython-Nutzer)
- [3.4 - Flüssigkristallanzeige](#) (Für Arduino-Nutzer)

## 2.17 WS2812 RGB 8-LED-Streifen



Der WS2812 RGB 8-LED-Streifen besteht aus 8 RGB-LEDs und lässt sich mit nur einem Pin steuern. Jede RGB-LED enthält einen WS2812-Chip und kann individuell angesteuert werden. Der Streifen ermöglicht eine Helligkeitsanzeige mit 256 Stufen und eine echte Farbanzeige mit 16.777.216 Farben. Er verfügt zudem über eine intelligente digitale Schnittstelle mit Datenhalte- und Signalformungsschaltung, um die Farbkonsistenz der Pixel zu gewährleisten.

Der Streifen ist flexibel, kann nach Belieben verlängert, gebogen und geschnitten werden. Die Rückseite ist mit einem Klebeband versehen, sodass der Streifen auch auf unebenen Flächen angebracht und in beengten Räumen installiert werden kann.

### Funktionen

- Betriebsspannung: DC5V
- IC: Ein IC steuert eine RGB-LED
- Verbrauch: 0,3 W pro LED
- Arbeitstemperatur: -15 bis 50 Grad Celsius
- Farbe: Vollfarb-RGB
- RGB-Typ: 5050RGB (Integrierter IC WS2812B)
- Streifendicke: 2 mm
- Jede LED ist einzeln steuerbar

### Einführung in WS2812B

- [WS2812B Datenblatt](#)

WS2812B ist eine intelligent steuerbare LED-Lichtquelle, bei der Schaltkreis und RGB-Chip in einem 5050-Gehäuse integriert sind. Es enthält eine intelligente digitale Schnittstelle mit Datenhalteschaltung und Signalformungsverstärkung. Zudem ist ein präziser interner Oszillator sowie eine mit 12V programmierbare Konstantstromsteuerung enthalten, die für eine gleichbleibende Farbqualität der Pixel sorgt.

Das Datenübertragungsprotokoll verwendet den Einzel-NZR-Kommunikationsmodus. Nach dem Einschalten der Pixel empfängt der DIN-Port Daten vom Controller. Das erste Pixel sammelt die ersten 24 Bit Daten und sendet sie an den internen Datenlatch. Die restlichen Daten werden durch die interne Signalformungs- und Verstärkungsschaltung an das nächste kaskadierende Pixel über den DO-Port weitergeleitet.



Durch die niedrige Betriebsspannung ist die LED umweltfreundlich und energiesparend. Sie bietet hohe Helligkeit, einen großen Streuwinkel, gute Konsistenz, geringen Stromverbrauch und eine lange Lebensdauer. Die Integration des Steuerchips in die LED vereinfacht die Schaltung und erleichtert die Installation.

### Beispiel

- [3.3 RGB LED-Streifen](#) (Für MicroPython-Nutzer)
- [7.8 RFID-Musikplayer](#) (Für MicroPython-Nutzer)
- [3.3 WS2812 RGB-Strip](#) (Für Arduino-Nutzer)
- [2.10 Fließende LEDs](#) (Für Piper Make-Nutzer)

### Akustische Elemente

## 2.18 Summer

Als eine Art von elektronischem Summer mit integrierter Struktur, die von Gleichstrom versorgt werden, finden Summer weitreichende Anwendung in Computern, Druckern, Kopierern, Alarmanlagen, elektronischem Spielzeug, Kfz-Elektronik, Telefonen, Zeitgebern und anderen elektronischen Produkten oder stimmlichen Geräten.

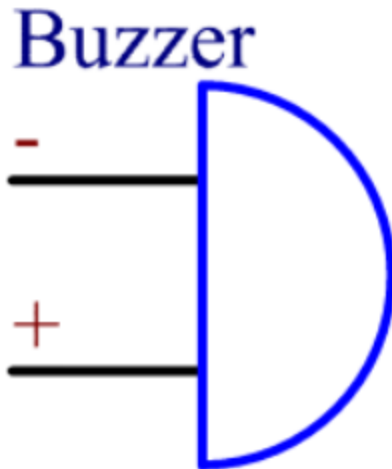
Summer lassen sich in aktive und passive Modelle unterteilen (siehe nachstehendes Bild). Wenn die Anschlusspins des Summers nach oben zeigen, ist der Summer mit einer grünen Platine ein passiver Summer, während der mit schwarzem Klebeband umwickelte ein aktiver Summer ist.



Unterschied zwischen einem aktiven und einem passiven Summer:

Ein aktiver Summer verfügt über eine integrierte Oszillationsquelle und gibt beim Anlegen einer Spannung einen Ton ab. Ein passiver Summer hingegen besitzt keine solche Quelle und kann daher nicht mit Gleichstromsignalen betrieben werden; stattdessen müssen Sie ihn mit Rechteckwellen antreiben, deren Frequenz zwischen 2K und 5K liegt. Aktive Summer sind aufgrund der mehreren integrierten Oszillationskreise meist teurer als passive.

Im Folgenden ist das elektrische Symbol eines Summers dargestellt. Er besitzt zwei Anschlusspins für den positiven und den negativen Pol. Ein Pluszeichen auf der Oberfläche kennzeichnet die Anode, der andere Pin ist die Kathode.



An den Pins des Summers können Sie erkennen, dass der längere der Anodenanschluss und der kürzere der Kathodenanschluss ist. Bitte verwechseln Sie diese nicht beim Anschließen, da sonst der Summer keinen Ton abgibt.

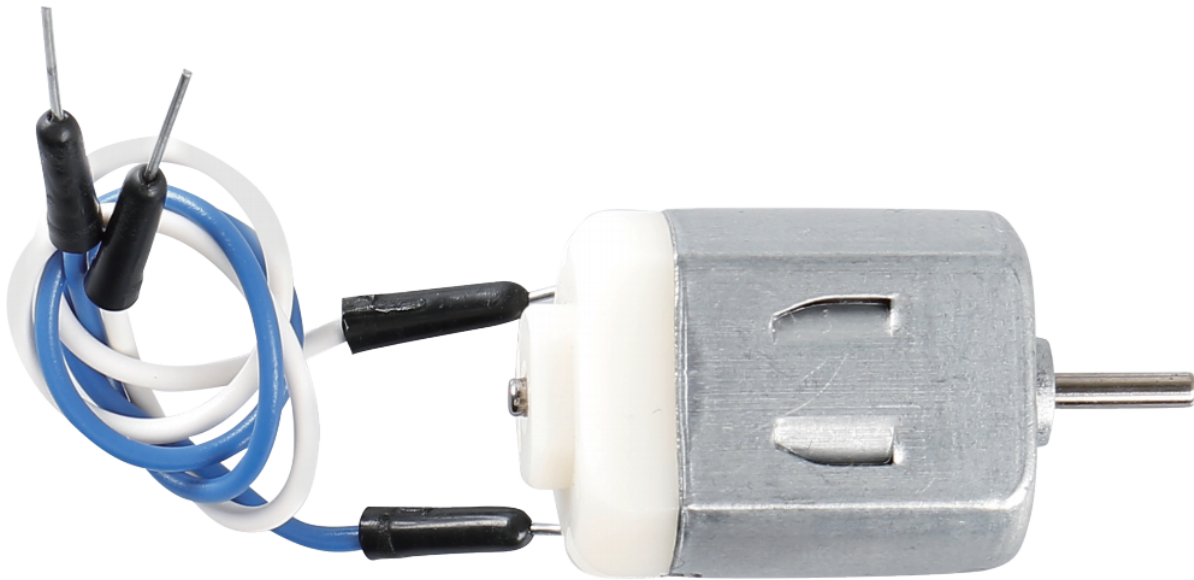
[Summer - Wikipedia](#)

### Beispiel

- [3.1 Piepton](#) (Für MicroPython-Nutzer)
- [3.2 Eigener Ton](#) (Für MicroPython-Nutzer)
- [7.1 Licht-Theremin](#) (Für MicroPython-Nutzer)
- [7.3 Alarmanlagenlampe](#) (Für MicroPython-Nutzer)
- [7.8 RFID-Musikplayer](#) (Für MicroPython-Nutzer)
- [7.9 Frucht-Klavier](#) (Für MicroPython-Nutzer)
- [7.10 Einparkhilfe](#) (Für MicroPython-Nutzer)
- [3.1 - Piepton](#) (Für Arduino-Nutzer)
- [3.2 - Individueller Ton](#) (Für Arduino-Nutzer)
- [2.3 Serviceklingel](#) (Für Piper Make-Nutzer)
- [2.11 Rückfahrssystem](#) (Für Piper Make-Nutzer)
- [2.13 Reaktionsspiel](#) (Für Piper Make-Nutzer)

### Aktuatoren

## 2.19 Gleichstrommotor

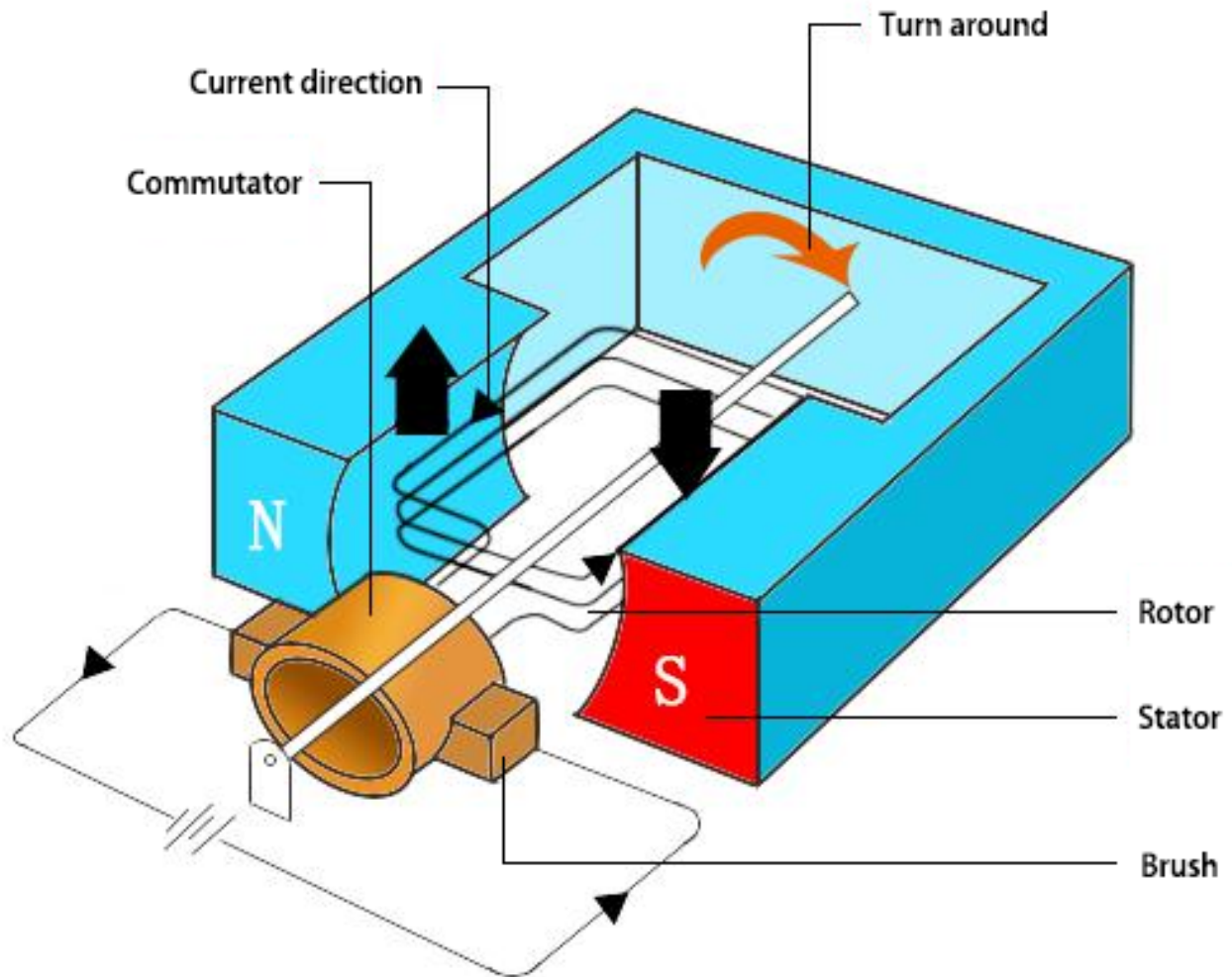


Hierbei handelt es sich um einen 3V-Gleichstrommotor. Wenn Sie an beide der 2 Anschlüsse ein hohes und ein niedriges Signal anlegen, beginnt der Motor zu drehen.

- **Größe:** 25\*20\*15MM
- **Betriebsspannung:** 1-6V
- **Leerlaufstrom** (3V): 70m
- **Leerlaufdrehzahl** (3V): 13000 U/min
- **Blockierstrom** (3V): 800mA
- **Wellendurchmesser:** 2mm

Ein Gleichstrommotor ist ein kontinuierlicher Aktuator, der elektrische Energie in mechanische Energie umwandelt. Durch ihre Fähigkeit zur kontinuierlichen Winkelrotation treiben Gleichstrommotoren Rotationspumpen, Ventilatoren, Kompressoren, Laufräder und andere Geräte an.

Ein Gleichstrommotor besteht aus zwei Hauptkomponenten: dem festen Teil des Motors, genannt **Stator**, und dem beweglichen Innenbereich des Motors, bekannt als **Rotor** (oder **Anker** eines Gleichstrommotors). Der Schlüssel zur Bewegungserzeugung liegt in der Positionierung des Ankers im Magnetfeld des Permanentmagneten, dessen Feld sich von Nordpol zu Südpol erstreckt. Die Wechselwirkung zwischen diesem Magnetfeld und den bewegten geladenen Teilchen (durch den stromführenden Draht erzeugt) resultiert im Drehmoment, das den Anker rotieren lässt.



Der Strom fließt vom positiven Pol der Batterie durch die Schaltung, über die Kupferbürsten zum Kommutator und dann zum Anker. Aufgrund der zwei Lücken im Kommutator kehrt dieser Fluss bei jeder vollständigen Rotation um. Diese kontinuierliche Umkehrung wandelt die Gleichstromversorgung der Batterie im Wesentlichen in Wechselstrom um, sodass der Anker das Drehmoment zur richtigen Zeit in die richtige Richtung erhält, um die Rotation aufrechtzuerhalten.

- [Gleichstrommotor - MagLab](#)
- [Flemings linke Handregel für Motoren - Wikipedia](#)

#### Beispiel

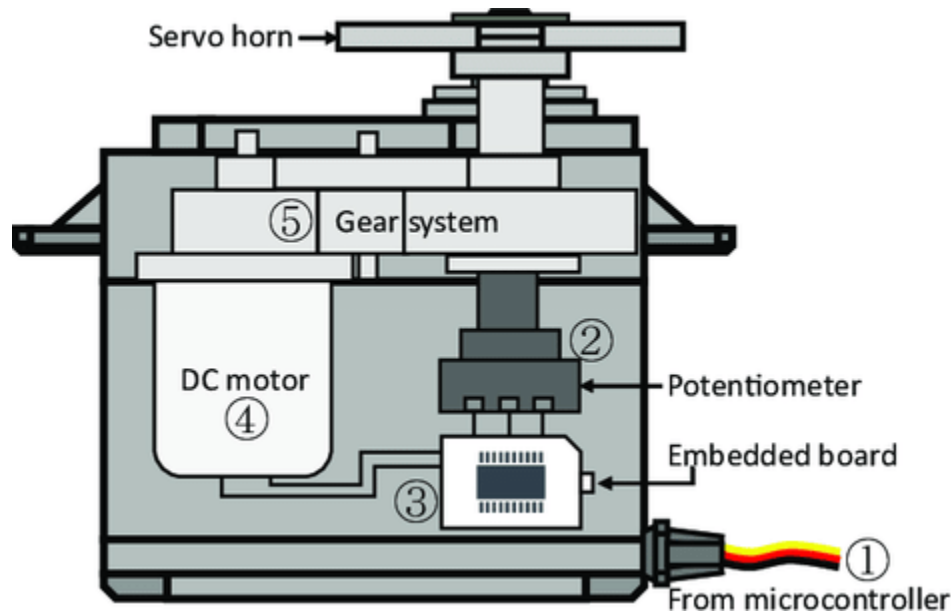
- [3.5 Kleiner Ventilator](#) (Für MicroPython-Nutzer)
- [3.5 - Kleiner Ventilator](#) (Für Arduino-Nutzer)
- [2.12 Intelligenter Ventilator](#) (Für Piper Make-Nutzer)

## 2.20 Servo

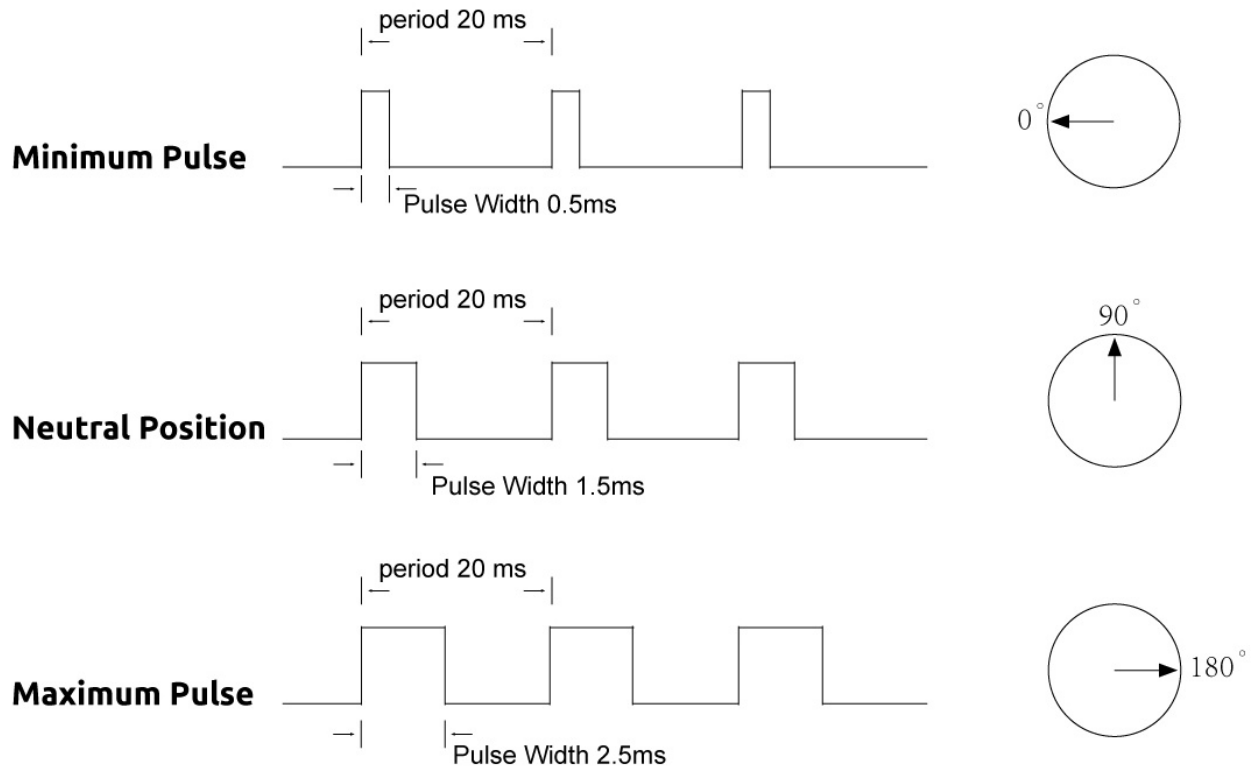


Ein Servo setzt sich in der Regel aus den folgenden Teilen zusammen: Gehäuse, Welle, Getriebesystem, Potentiometer, Gleichstrommotor und eingebettete Platine.

Die Funktionsweise ist wie folgt: Der Mikrocontroller sendet PWM-Signale an den Servo, welche von der eingebetteten Platine im Servo über den Signaleingang empfangen werden. Die Platine steuert daraufhin den internen Motor an, der das Getriebe in Bewegung setzt und so die Welle antreibt. Die Welle und das Potentiometer des Servos sind miteinander verbunden. Wenn die Welle sich dreht, wird das Potentiometer mitbewegt, das dann eine Spannung an die Platine sendet. Basierend auf dieser Spannung bestimmt die Platine die Drehrichtung und -geschwindigkeit und stoppt die Welle exakt in der vorgegebenen Position.



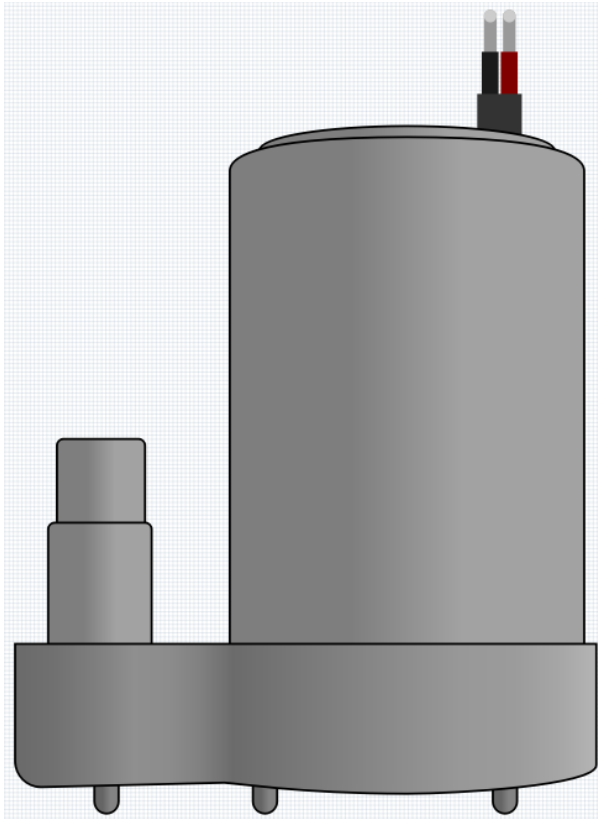
Der Winkel wird durch die Dauer eines Pulses bestimmt, der auf das Steuerkabel aufgebracht wird. Dies wird als Pulsweitenmodulation bezeichnet. Der Servo erwartet alle 20 ms einen Puls. Die Länge dieses Pulses bestimmt, wie weit sich der Motor dreht. Ein 1,5 ms langer Puls beispielsweise bringt den Motor in die 90-Grad-Position (Neutralstellung). Wird ein kürzerer Puls als 1,5 ms an den Servo gesendet, dreht dieser seine Ausgangswelle um eine bestimmte Anzahl von Grad gegen den Uhrzeigersinn von der Neutralposition aus. Ein Puls, der länger als 1,5 ms ist, bewirkt das Gegenteil. Die minimale und maximale Pulsbreite, die den Servo zu einer gültigen Position steuert, hängt vom jeweiligen Servo ab. Üblicherweise beträgt die minimale Pulsbreite etwa 0,5 ms und die maximale etwa 2,5 ms.



#### Beispiel

- *3.7 Schwingender Servo* (Für MicroPython-Nutzer)
- *7.11 Somatosensorische Steuerung* (Für MicroPython-Nutzer)
- *3.7 - Schwingender Servo* (Für Arduino-Nutzer)
- *2.6 Intelligenter Wassertank* (Für Piper Make-Nutzer)
- *2.7 Schwenk-Servo* (Für Piper Make-Nutzer)
- *2.9 Glückskatze* (Für Piper Make-Nutzer)

## 2.21 DC-Wasserpumpe



Diese Pumpe funktioniert im Grunde genommen wie ein Gleichstrommotor, betrieben mit einer Spannung von 3V und einem Strom von 100mA. Sobald sie eingeschaltet ist, saugt die Pumpe Wasser über das untere Ende ihres Kunststoffgehäuses an und drückt es durch das Auslassrohr hinaus. Sie muss stets vollständig im Wasser eingetaucht sein, um korrekt zu arbeiten. Eine Umkehrung der Polarität führt nicht dazu, dass sie als Wassereinzugsgerät fungiert; sie pumpt weiterhin nur Wasser aus!

Diese Tauchpumpe eignet sich hervorragend für Anfänger, die ein Springbrunnen- oder Pflanzenbewässerungsprojekt realisieren möchten, da sie extrem benutzerfreundlich ist.

### Funktionen

- **Spannungsbereich:** DC 3 ~ 4,5V
- **Betriebsstrom:** 120 ~ 180mA
- **Leistung:** 0,36 ~ 0,91W
- **Maximale Förderhöhe:** 0,35 ~ 0,55M
- **Maximale Durchflussrate:** 80 ~ 100 L/H
- **Kontinuierliche Betriebsdauer:** 100 Stunden
- **Wasserdichtheitsklasse:** IP68
- **Antriebsart:** DC, magnetischer Antrieb
- **Material:** Technischer Kunststoff
- **Außendurchmesser des Auslasses:** 7,8 mm



- **Innendurchmesser des Auslasses:** 6,5 mm
- Es handelt sich um eine Tauchpumpe, die auch so verwendet werden sollte. Bei ungetauchtem Betrieb besteht Überhitzungsgefahr.
- Sie ist mit einem 25 cm langen Steckdraht ausgestattet, der sich einfach in ein Steckbrett einsetzen lässt.

#### Beispiel

- *3.6 Pumpensteuerung* (Für MicroPython-Nutzer)
- *3.6 - Pumpensteuerung* (Für Arduino-Nutzer)

## 2.22 Relais



Wie bekannt ist, dient ein Relais dazu, eine Verbindung zwischen zwei oder mehr Punkten oder Geräten herzustellen, die auf ein eingegebenes Signal reagieren. Anders ausgedrückt, bieten Relais eine Isolation zwischen dem Controller und dem Gerät, da diese sowohl mit Wechselstrom (AC) als auch mit Gleichstrom (DC) betrieben werden können. Da sie jedoch Signale von einem Mikrocontroller erhalten, der mit Gleichstrom arbeitet, ist ein Relais erforderlich, um die Lücke zu schließen. Relais sind besonders nützlich, wenn man einen großen Strom oder eine hohe Spannung mit einem kleinen elektrischen Signal steuern muss.

Ein Relais besteht aus fünf Hauptkomponenten:

**Elektromagnet** - Er besteht aus einem Eisenkern, der von einer Spule umwickelt ist. Wenn Strom durchfließt, wird er magnetisch. Deshalb wird er als Elektromagnet bezeichnet.

**Anker** - Der bewegliche magnetische Streifen wird als Anker bezeichnet. Wenn Strom durch die Spule fließt, wird sie magnetisiert und erzeugt ein Magnetfeld, das dazu dient, die normalerweise offenen (N/O) oder normalerweise geschlossenen (N/C) Kontakte herzustellen oder zu trennen. Der Anker kann sowohl mit Gleichstrom (DC) als auch mit Wechselstrom (AC) bewegt werden.

**Feder** - Wenn kein Strom durch die Spule des Elektromagneten fließt, zieht die Feder den Anker zurück, sodass der Stromkreis nicht geschlossen werden kann.

**Elektrische Kontakte** - Es gibt zwei Kontaktstellen:

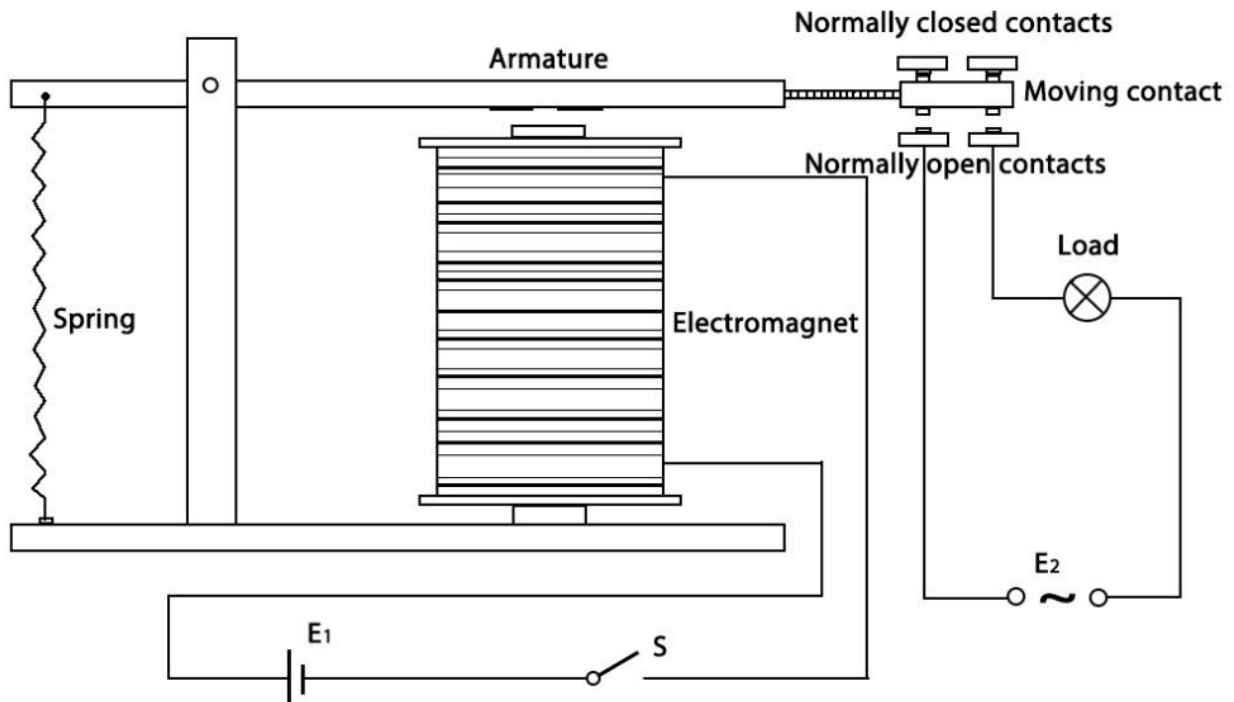
- Normalerweise offen - verbunden, wenn das Relais aktiviert ist, und getrennt, wenn es inaktiv ist.
- Normalerweise geschlossen - nicht verbunden, wenn das Relais aktiviert ist, und verbunden, wenn es inaktiv ist.

**Gehäuse** - Relais sind zum Schutz mit Kunststoff ummantelt.

Das Funktionsprinzip eines Relais ist einfach. Wenn Strom an das Relais angelegt wird, fließt der Strom durch die Steuerspule; daraufhin beginnt der Elektromagnet sich zu magnetisieren. Der Anker wird dann zur Spule hingezogen, und



der bewegliche Kontakt zieht mit und verbindet sich mit den normalerweise offenen Kontakten. Somit wird der Laststromkreis eingeschaltet. Um den Stromkreis wieder zu unterbrechen, wird der bewegliche Kontakt durch die Kraft der Feder zu den normalerweise geschlossenen Kontakten zurückgezogen. Auf diese Weise kann das Ein- und Ausschalten des Relais den Zustand eines Laststromkreises steuern.



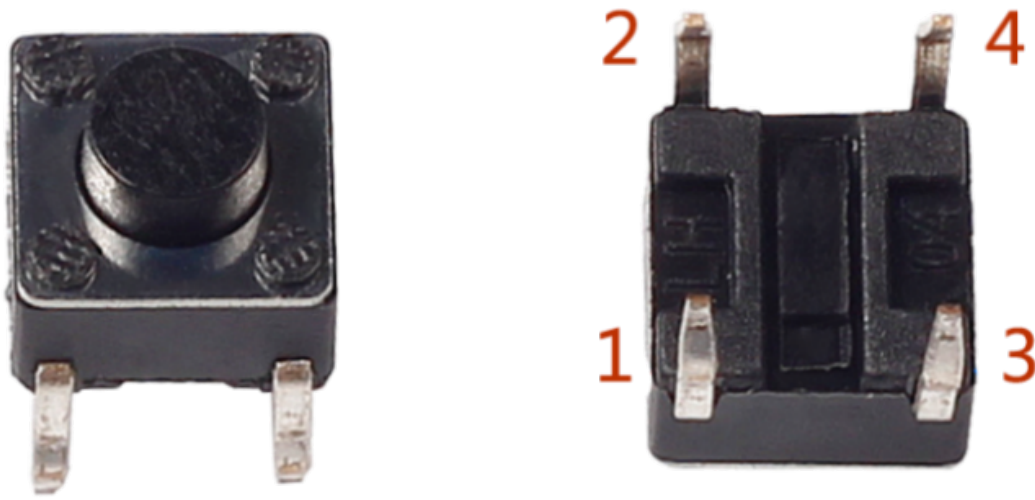
- [Relais - Wikipedia](#)

### Beispiel

- [2.16 Steuerung eines weiteren Stromkreises](#) (Für MicroPython-Nutzer)
- [2.16 - Steuern eines weiteren Stromkreises](#) (Für Arduino-Nutzer)

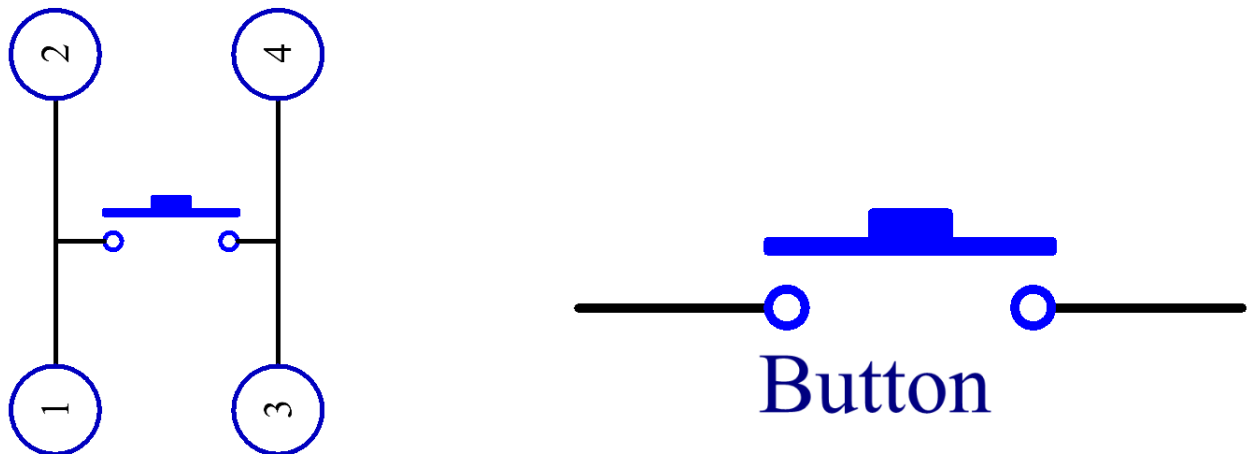
### Steuerelemente

## 2.23 Taster

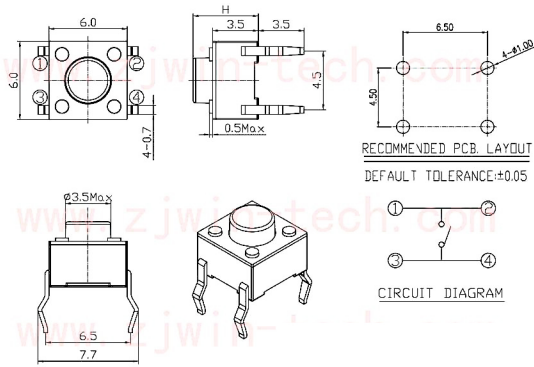


Taster sind ein häufig verwendeter Bauteil zur Steuerung elektronischer Geräte. Sie dienen meist als Schalter, um elektrische Stromkreise zu schließen oder zu unterbrechen. Obwohl Taster in verschiedenen Formen und Größen erhältlich sind, handelt es sich bei dem hier vorgestellten Modell um einen 6mm-Mini-Taster, wie auf den folgenden Bildern zu sehen ist. Pin 1 ist mit Pin 2 und Pin 3 mit Pin 4 verbunden. Man muss also lediglich entweder Pin 1 mit Pin 3 oder Pin 2 mit Pin 4 verbinden.

Nachfolgend ist die interne Struktur eines Tasters dargestellt. Das Symbol rechts unten wird üblicherweise verwendet, um einen Taster in Schaltkreisen zu kennzeichnen.



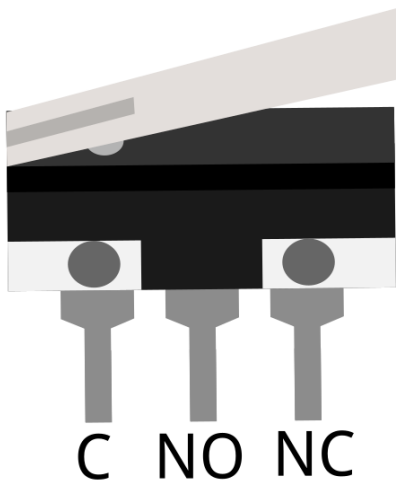
Da Pin 1 mit Pin 2 und Pin 3 mit Pin 4 verbunden sind, werden beim Drücken des Tasters alle 4 Pins miteinander verbunden, wodurch der Stromkreis geschlossen wird.



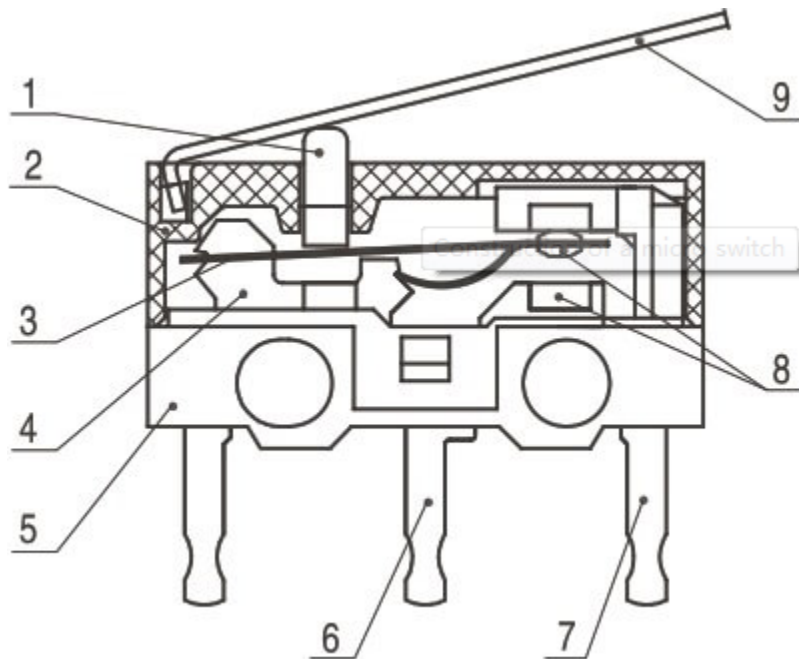
### Beispiel

- [2.5 Tastenwert auslesen](#) (Für MicroPython-Anwender)
- [2.5 - Tastenwert auslesen](#) (Für Arduino-Anwender)
- [2.2 Taster](#) (Für Piper Make-Anwender)
- [2.4 Regenbogenlicht](#) (Für Piper Make-Anwender)
- [2.5 Schlagzeug-Set](#) (Für Piper Make-Anwender)
- [2.13 Reaktionsspiel](#) (Für Piper Make-Anwender)

## 2.24 Mikroschalter



Der Aufbau eines Mikroschalters ist wirklich simpel. Die Hauptkomponenten des Schalters sind:



- 1. Betätigungsstößel (Aktuator)
- 2. Abdeckung
- 3. Bewegliches Teil
- 4. Halterung
- 5. Gehäuse
- 6. NO-Klemme: normalerweise offen
- 7. NC-Klemme: normalerweise geschlossen
- 8. Kontakt
- 9. Bewegungsarm

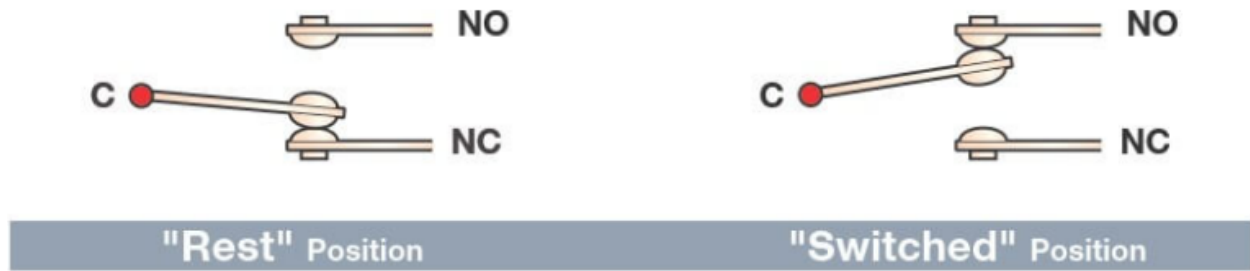
Nachdem der Mikroschalter physischen Kontakt mit einem Objekt hergestellt hat, ändert er die Position seiner Kontakte. Das grundlegende Funktionsprinzip ist wie folgt.

Wenn der Betätigungsstößel in der Ausgangs- oder Ruheposition ist:

- Der normalerweise geschlossene Stromkreis ist stromführend.
- Der normalerweise offene Stromkreis ist elektrisch isoliert.

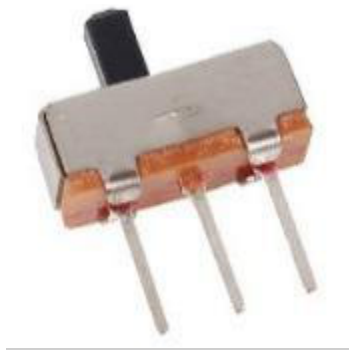
Wenn der Betätigungsstößel gedrückt oder umgeschaltet wird:

- Der normalerweise geschlossene Stromkreis ist unterbrochen.
- Der normalerweise offene Stromkreis ist geschlossen.

**Beispiel**

- 2.8 *Sanft Drücken* (Für MicroPython-Anwender)
- 2.8 - *Sanft Drücken* (Für Arduino-Anwender)
- 2.3 *Serviceklingel* (Für Piper Make-Anwender)

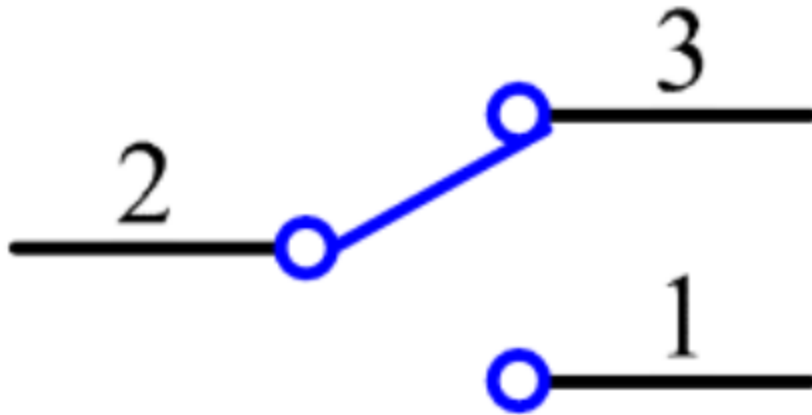
## 2.25 Schiebeschalter



Ein Schiebeschalter funktioniert, wie der Name schon sagt, durch das Verschieben der Schalteiste, um den Stromkreis zu schließen oder zu unterbrechen und weitere Schaltkreise umzuschalten. Gebräuchliche Typen sind SPDT, SPTT, DPDT, DPTT usw. Der Schiebeschalter wird häufig in Niederspannungsschaltkreisen eingesetzt. Er zeichnet sich durch Flexibilität und Stabilität aus und findet breite Anwendung in elektrischen Instrumenten und elektronischem Spielzeug. Funktionsweise: Der mittlere Pin dient als fester Anschlusspunkt. Wenn Sie den Schieber nach links ziehen, werden die beiden linken Pins miteinander verbunden; ziehen Sie ihn nach rechts, werden die beiden rechten Pins verbunden. So fungiert er als Schalter, der Schaltkreise verbindet oder trennt. Siehe nachstehende Abbildung:



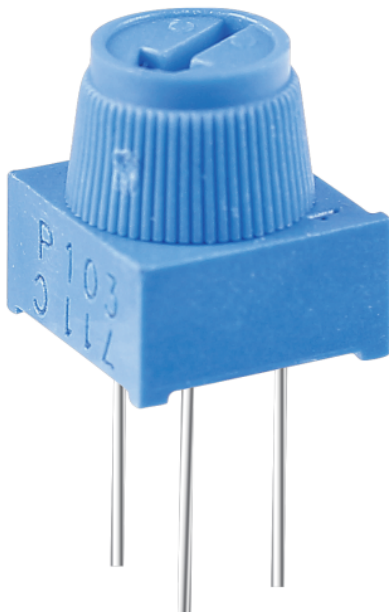
Das Schaltzeichen des Schiebeschalters ist unten dargestellt. Der Pin2 in der Abbildung bezieht sich auf den mittleren Pin.



### Beispiel

- *2.7 Nach Links und Rechts Schalten* (Für MicroPython-Anwender)
- *7.3 Alarmanlagenlampe* (Für MicroPython-Anwender)
- *2.7 - Links und Rechts Umschalten* (Für Arduino-Anwender)
- *2.5 Schlagzeug-Set* (Für Piper Make-Anwender)

## 2.26 Potentiometer



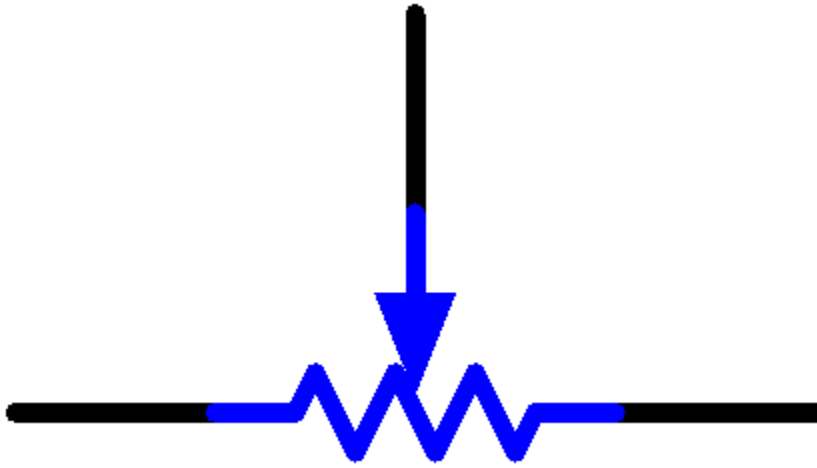
Ein Potentiometer ist ebenfalls ein Widerstandselement mit drei Anschlüssen, dessen Widerstandswert nach einer bestimmten Regelung variiert werden kann.

Potentiometer gibt es in verschiedenen Formen, Größen und Werten, sie haben jedoch alle folgende Gemeinsamkeiten:

- Sie verfügen über drei Anschlüsse (oder Kontaktpunkte).
- Sie besitzen einen Drehknopf, eine Schraube oder einen Schieber, mit dem der Widerstand zwischen dem mittleren und einem der äußeren Anschlüsse verändert werden kann.

- 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 Schaltzeichen für ein Potentiometer.



Die Funktionen des Potentiometers im Stromkreis sind wie folgt:

1. Als Spannungsteiler

Das Potentiometer ist ein stufenlos einstellbarer Widerstand. Wenn Sie die Achse oder den Schiebegriff des Potentiometers verstellen, gleitet der bewegliche Kontakt über den Widerstand. An dieser Stelle kann eine Spannung abhängig von der am Potentiometer angelegten Spannung und dem Drehwinkel oder dem Verfahrweg des beweglichen Arms ausgegeben werden.

2. Als Rheostat

Wenn das Potentiometer als Rheostat verwendet wird, verbinden Sie den mittleren Pin mit einem der beiden anderen Pins im Stromkreis. So erhalten Sie einen stufenlos und kontinuierlich veränderbaren Widerstandswert innerhalb des Verfahrwegs des beweglichen Kontakts.

3. Als Stromregler

Wenn das Potentiometer als Stromregler fungiert, muss der Schiebekontakt als einer der Ausgangsanschlüsse verbunden sein.

Wenn Sie mehr über Potentiometer erfahren möchten, siehe: [Potentiometer - Wikipedia](#)

**Beispiel**

- [2.11 Den Drehregler betätigen](#) (Für MicroPython-Anwender)
- [2.11 - Drehen Sie den Knopf](#) (Für Arduino-Anwender)
- [2.7 Schwenk-Servo](#) (Für Piper Make-Anwender)

## 2.27 Infrarotempfänger

### 2.27.1 IR-Empfänger



- S: Signalausgang
- +: VCC
- -: GND

Ein Infrarotempfänger ist eine Komponente, die Infrarotsignale empfangen kann. Er ist in der Lage, eigenständig Infrarotstrahlung zu detektieren und TTL-kompatible Signale auszugeben. Von der Größe her entspricht er einem herkömmlichen Transistor im Kunststoffgehäuse und eignet sich für alle Anwendungsgebiete von Infrarot-Fernbedienungen bis zu Infrarot-Datenübertragung.

Infrarot (IR) ist eine populäre, kosteneffiziente und einfach zu verwendende Technologie für drahtlose Kommunikation. Infrarotlicht liegt in einem Wellenlängenbereich, der für das menschliche Auge unsichtbar ist, was es ideal für drahtlose Übertragungen macht. Häufig wird eine 38-kHz-Modulation für die Infrarotkommunikation verwendet.

- Verwendet den hochsensiblen HX1838 IR-Empfänger-Sensor
- Für den Einsatz in Fernbedienungen geeignet
- Stromversorgung: 3,3 bis 5V
- Digitale Schnittstelle
- Modulationsfrequenz: 38 kHz



## 2.27.2 Fernbedienung



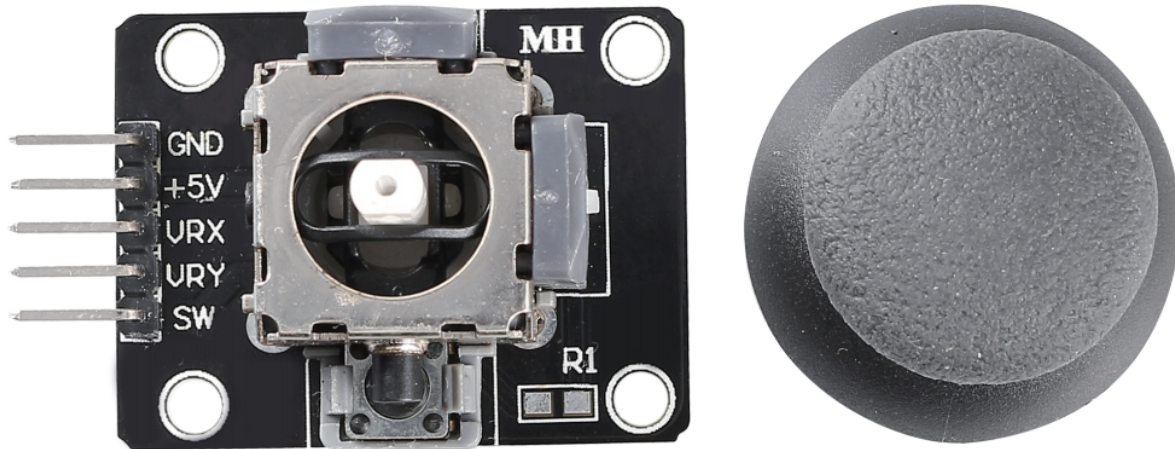
Diese kompakte Infrarot-Fernbedienung verfügt über 21 Funktionstasten und eine Übertragungsreichweite von bis zu 8 Metern. Sie ist optimal zur Steuerung diverser Geräte im Kinderzimmer einsetzbar.

- Abmessungen: 85x39x6mm
- Reichweite: 8 bis 10m
- Batterietyp: 3V Knopfzellen-Lithium-Mangan-Batterie
- Trägerfrequenz für Infrarot: 38 kHz
- Oberflächenmaterial: 0,125 mm PET
- Effektive Nutzungsdauer: über 20.000 Betätigungen

### Beispiele

- [6.4 Infrarot-Fernbedienung](#) (Für MicroPython-Anwender)
- [6.4 - IR-Fernbedienung](#) (Für Arduino-Anwender)

## 2.28 Joystick-Modul

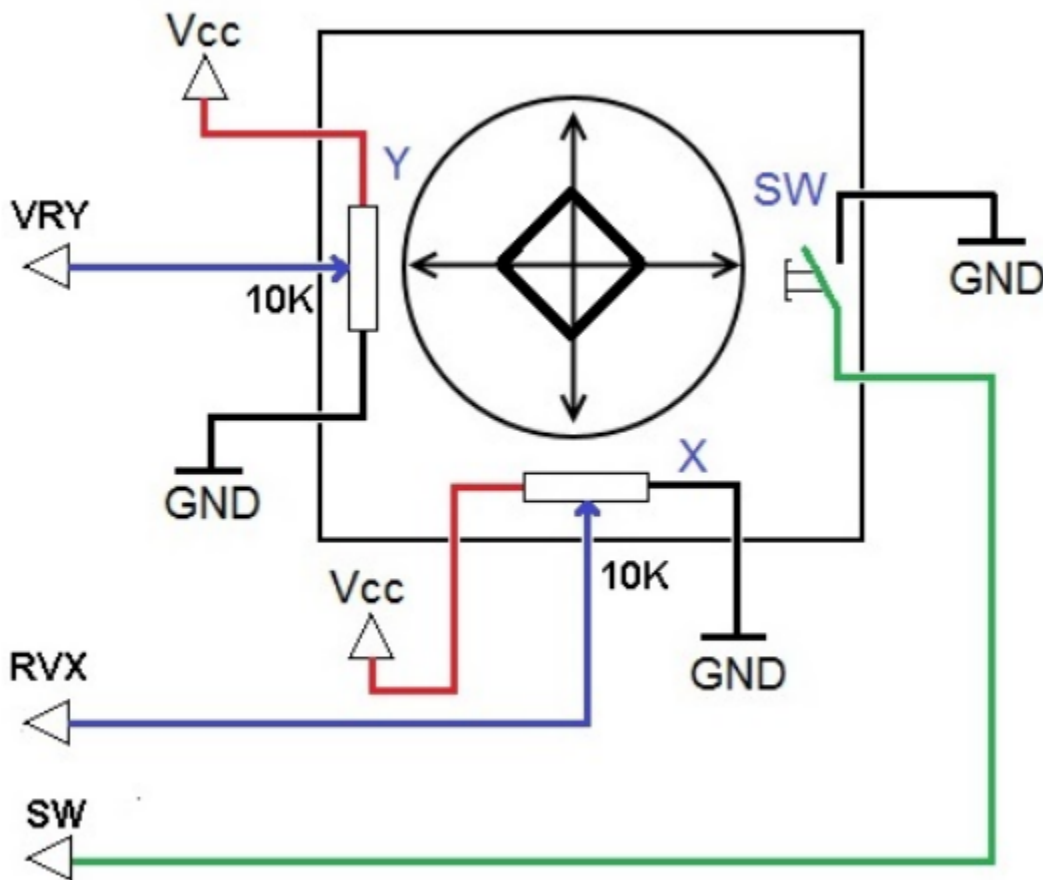


Die grundlegende Idee eines Joysticks besteht darin, die Bewegungen eines Sticks in elektronische Informationen umzuwandeln, die ein Computer verarbeiten kann.

Um dem Computer eine vollständige Bewegungspalette zu übermitteln, muss ein Joystick die Position des Sticks entlang zweier Achsen messen – der X-Achse (von links nach rechts) und der Y-Achse (von oben nach unten). Wie in der elementaren Geometrie geben die X-Y-Koordinaten die genaue Position des Sticks an.

Zur Bestimmung der Position des Sticks überwacht das Joystick-Steuersystem einfach die Lage jedes Schafts. Das herkömmliche analoge Joystick-Design verwendet hierzu zwei Potenziometer oder variable Widerstände.

Der Joystick verfügt auch über einen digitalen Eingang, der betätigt wird, wenn der Joystick nach unten gedrückt wird.



- Joystick - Wikipedia

#### Beispiel

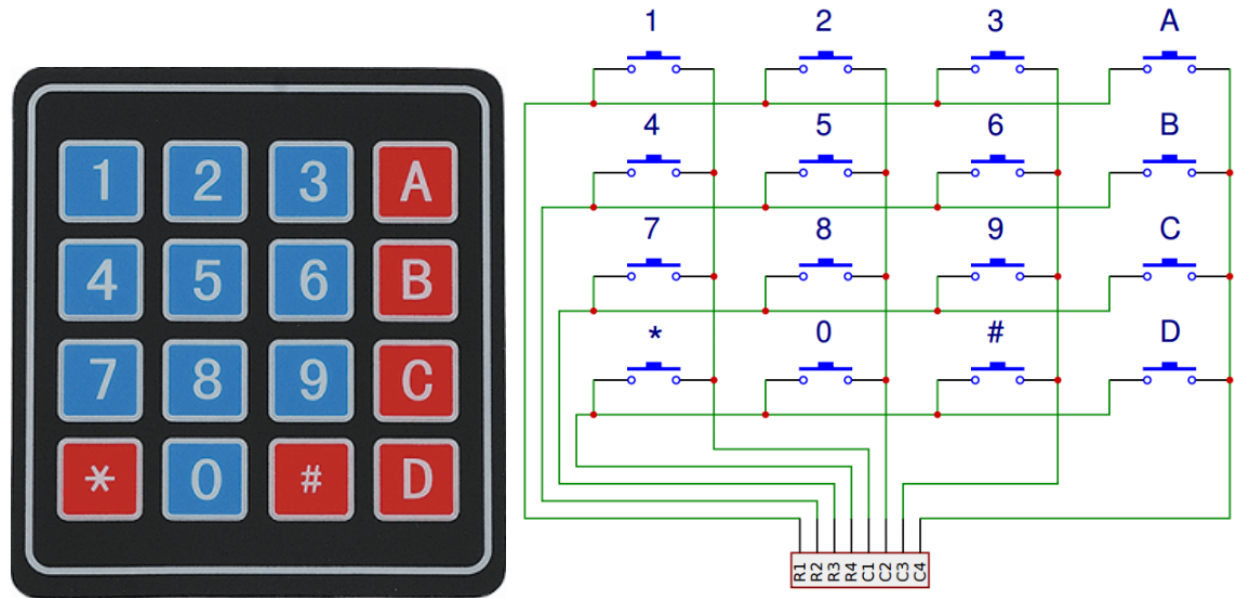
- [4.1 Den Joystick umschalten](#) (Für MicroPython-Nutzer)
- [4.1 - Den Joystick bedienen](#) (Für Arduino-Nutzer)

## 2.29 4x4 Tastenfeld

Im Mikrocontroller-System, wenn mehrere Tasten wie beispielsweise in elektronischen Codeschlössern oder Telefon-Tastensfeldern verwendet werden, gibt es in der Regel mindestens 12 bis 16 Tasten. Üblicherweise kommt dabei ein Matrix-Tastensfeld zum Einsatz.

Ein Matrix-Tastensfeld wird auch Reihen-Tastensfeld genannt. Es verfügt über vier I/O-Leitungen als Reihen und vier I/O-Leitungen als Spalten. An jedem Schnittpunkt der Reihen- und Spaltenleitungen ist eine Taste angebracht. Somit beträgt die Anzahl der Tasten auf der Tastatur  $4 \times 4$ . Diese Reihen- und Spaltenstruktur kann die Auslastung der I/O-Ports in einem Mikrocontroller-System effektiv verbessern.

Die Kontakte werden über eine Stiftleiste erreicht, die sich für die Verbindung mit einem Flachbandkabel oder zur Einsteckmontage in eine Leiterplatte eignet. Bei einigen Tastensfeldern stellt jede Taste eine separate Verbindung im Header her, während alle Tasten einen gemeinsamen Masseanschluss teilen.



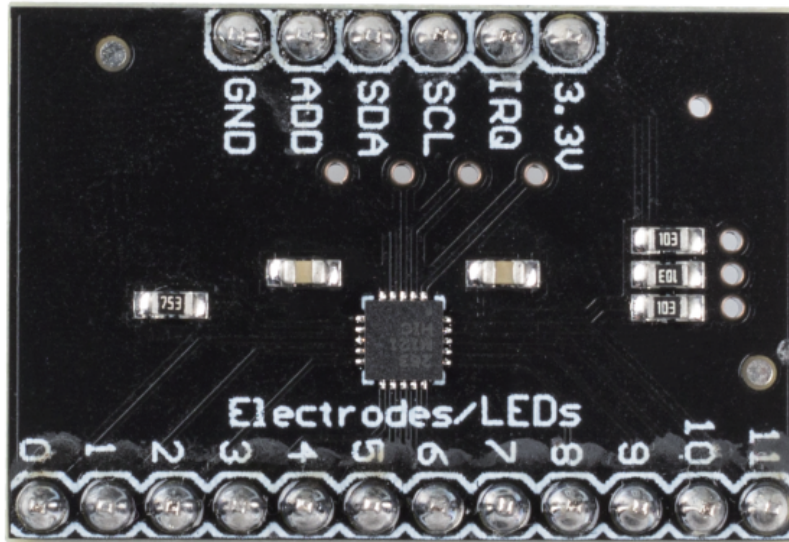
Häufiger sind die Tasten matrixcodiert, was bedeutet, dass jede Taste ein einzigartiges Paar von Leitern in einer Matrix verbindet. Diese Konfiguration eignet sich für die Abfrage durch einen Mikrocontroller, der so programmiert werden kann, dass er nacheinander einen Ausgangspuls an jede der vier horizontalen Leitungen sendet. Während jedes Pulses prüft er die verbleibenden vier vertikalen Leitungen sequenziell, um festzustellen, welche davon, falls überhaupt, ein Signal trägt. Pull-up- oder Pull-down-Widerstände sollten zu den Eingangsleitungen hinzugefügt werden, um unvorhersehbares Verhalten der Mikrocontroller-Eingänge zu verhindern, wenn kein Signal anliegt.

- [Tastenfeld - Wikipedia](#)

### Beispiel

- [4.2 4x4 Tastenfeld](#) (Für MicroPython-Nutzer)
- [7.7 Zahlenraten](#) (Für MicroPython-Nutzer)
- [4.2 - 4x4 Tastenfeld](#) (Für Arduino-Nutzer)

## 2.30 MPR121 Modul



- **3.3V:** Spannungsversorgung
- **IRQ:** Open-Collector-Interrupt-Ausgangspin, aktiv niedrig
- **SCL:** I2C-Takt
- **SDA:** I2C-Daten
- **ADD:** I2C-Adressauswahl-Eingangspin. Verbinden Sie den ADDR-Pin mit der VSS-, VDD-, SDA- oder SCL-Leitung. Die resultierenden I2C-Adressen sind jeweils 0x5A, 0x5B, 0x5C und 0x5D
- **GND:** Masse
- **0~11:** Elektrode 0~11, eine Berührungssensor-Elektrode. Üblicherweise können Elektroden einfache Metallstücke oder Drähte sein. Je nach Kabellänge oder Material der Unterlage kann jedoch die Auslösung des Sensors erschwert werden. Aus diesem Grund ermöglicht der MPR121 die Konfiguration der Auslöse- und Rücksetzkriterien für jede Elektrode.

### MPR121 ÜBERSICHT

Der MPR121 ist die zweite Generation von kapazitiven Berührungssensoren nach der Einführung der MPR03x-Serie. Der MPR121 bietet erweiterte interne Intelligenz; zu den wichtigsten Neuerungen gehören eine erhöhte Anzahl von Elektroden, eine hardwarekonfigurierbare I2C-Adresse, ein erweitertes Filtersystem mit Entprellung und vollkommen unabhängige Elektroden mit integrierter Auto-Konfiguration. Zudem verfügt das Gerät über einen 13. simulierten Sensor-Kanal, der speziell für die Nahbereichserkennung über die multiplexten Sensoreingänge genutzt wird.

- [MPR121 Datenblatt](#)

### Funktionen

- **Niedriger Energieverbrauch**
  - Betriebsspannung von 1,71 V bis 3,6 V
  - 29 A Versorgungsstrom bei einem Abtastintervall von 16 ms
  - 3 A Stromaufnahme im Stop-Modus
- **12 kapazitive Sensoreingänge**
  - 8 Eingänge multifunktional als LED-Treiber und GPIO verwendbar

- **Vollständige Berührungserkennung**
  - Auto-Konfiguration für jeden Sensoreingang
  - Auto-Kalibrierung für jeden Sensoreingang
  - Berührungs-/Freigabeschwelle und Entprellung für die Berührungserkennung
- I2C-Schnittstelle mit Interrupt-Ausgang
- 3 mm x 3 mm x 0,65 mm 20-poliges QFN-Gehäuse
- Betriebstemperaturbereich von -40°C bis +85°C

#### Beispiel

- [4.3 Elektroden-Tastatur](#) (Für MicroPython-Nutzer)
- [7.9 Frucht-Klavier](#) (Für MicroPython-Nutzer)
- [4.3 - Elektroden-Tastatur](#) (Für Arduino-Nutzer)

## 2.31 MFRC522 Modul



Der MFRC522 ist eine Art integrierter Lese- und Schreibkartenchip, der üblicherweise im 13,56-MHz-Radiofrequenzbereich eingesetzt wird. Herausgebracht von NXP, handelt es sich um einen kostengünstigen, niederspannungs- und kompakten kontaktlosen Kartenchip, der sich optimal für intelligente Instrumente und tragbare Handgeräte eignet.

Das MFRC522 Modul nutzt fortschrittliche Modulations- und Demodulationskonzepte, die alle Arten von 13,56-MHz-passiven, kontaktlosen Kommunikationsmethoden und -protokollen vollständig abdecken. Darüber hinaus unterstützt es den schnellen CRYPTO1-Verschlüsselungsalgorithmus zur Verifizierung von MIFARE-Produkten. Der MFRC522 ermöglicht zudem Hochgeschwindigkeitskommunikation mit MIFARE-Serienprodukten und bietet eine bidirektionale Datenübertragungsrate von bis zu 424 kbit/s. Als neues Mitglied der 13,56-MHz-Hochintegrierten-Lesekarten-Serie ähnelt der MFRC522 den existierenden Modellen MF RC500 und MF RC530, weist jedoch auch signifikante Unterschiede auf. Die Kommunikation mit der Host-Maschine erfolgt seriell, was den Verdrahtungsaufwand reduziert. Wahlweise stehen SPI, I2C und serieller UART-Modus (vergleichbar mit RS232) zur Verfügung, was die Anschlussmöglichkeiten minimiert, Platz auf der Leiterplatte spart und die Kosten senkt.

- [MFRC522 Datenblatt](#)

#### Beispiel

- [6.5 Funkfrequenz-Identifikation](#) (Für MicroPython-Nutzer)
- [7.8 RFID-Musikplayer](#) (Für MicroPython-Nutzer)



- [6.5 - Funkfrequenz-Identifikation](#) (Für Arduino-Nutzer)

## Sensoren

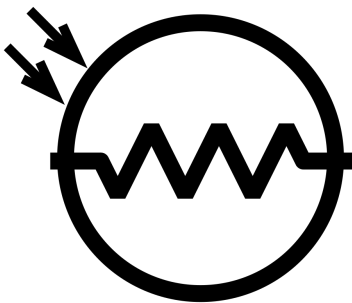
### 2.32 Fotowiderstand



Ein Fotowiderstand oder auch Fotodiode ist ein lichtgesteuerter variabler Widerstand. Die Widerstandsfähigkeit eines Fotowiderstands nimmt ab, sobald die Intensität des einfallenden Lichts zunimmt; anders ausgedrückt zeigt er eine Fotoleitfähigkeit.

Fotowiderstände finden Anwendung in lichtempfindlichen Detektorschaltungen sowie in licht- und dunkelaktivierten Schaltkreisen als halbleitender Widerstand. Im Dunkeln kann der Widerstand eines Fotowiderstands mehrere Megaohm (M) betragen, während er bei Lichteinfall auf wenige hundert Ohm sinken kann.

Hier ist das Elektroniksymbol für einen Fotowiderstand.



- [Fotowiderstand - Wikipedia](#)

#### Beispiel

- [2.12 Das Licht spüren](#) (Für MicroPython-Nutzer)
- [7.1 Licht-Theremin](#) (Für MicroPython-Nutzer)
- [2.12 - Das Licht erfassen](#) (Für Arduino-Nutzer)
- [2.8 Lichtintensitätsanzeige](#) (Für Piper Make-Nutzer)

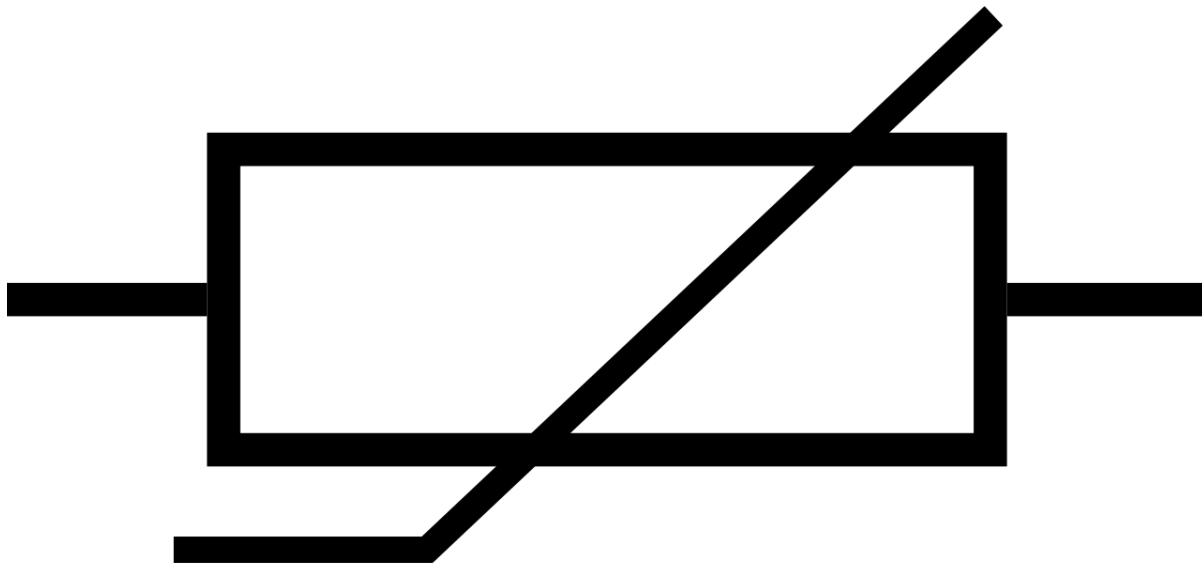
## 2.33 Thermistor



Ein Thermistor ist eine spezielle Form eines Widerstands, dessen Widerstandswert stark temperaturabhängig ist, viel stärker als bei Standardwiderständen. Der Begriff setzt sich aus den Wörtern „thermal“ und „resistor“ zusammen. Thermistoren werden häufig als Einschaltstrombegrenzer, Temperatursensoren (in der Regel vom NTC-Typ mit negativem Temperaturkoeffizienten), selbstzurücksetzende Überstromschutzvorrichtungen und selbstregelnde Heizelemente (in der Regel vom PTC-Typ mit positivem Temperaturkoeffizienten) eingesetzt.

- [Thermistor - Wikipedia](#)

Hier ist das Elektroniksymbol für einen Thermistor.



Es gibt zwei grundlegend unterschiedliche Arten von Thermistoren:

- Bei NTC-Thermistoren nimmt der Widerstand mit steigender Temperatur ab. Dies ist in der Regel auf eine Zunahme der durch thermische Anregung angeregten Leitungselektronen zurückzuführen. NTCs werden häufig als



Temperatursensoren eingesetzt oder in Reihe zu einer Schaltung als Einschaltstrombegrenzer.

- Bei PTC-Thermistoren steigt der Widerstand mit zunehmender Temperatur, in der Regel durch eine Zunahme der thermischen Gitteranregungen, insbesondere von Verunreinigungen und Unvollkommenheiten. PTC-Thermistoren werden oft in Reihe zu einer Schaltung eingesetzt und dienen als rückstellbare Sicherungen gegen Überstrombedingungen.

In diesem Bausatz verwenden wir einen NTC-Thermistor. Jeder Thermistor hat einen Nennwiderstand. Hier beträgt dieser 10 kOhm, gemessen bei einer Temperatur von 25 Grad Celsius.

Hier ist die Beziehung zwischen dem Widerstand und der Temperatur:

$$R_T = R_N * \exp(B * (1/T_K - 1/T_N))$$

- **$R_T$**  ist der Widerstand des NTC-Thermistors bei der Temperatur  $T_K$ .
- **$R_N$**  ist der Widerstand des NTC-Thermistors bei der Nenntemperatur  $T_N$ . Hier beträgt der Zahlenwert von  $R_N$  10k.
- **$T_K$**  ist eine Temperatur in Kelvin, die Einheit ist K. Hier beträgt der Zahlenwert von  $T_K$  273,15 + Grad Celsius.
- **$T_N$**  ist eine Nenntemperatur in Kelvin; die Einheit ist ebenfalls K. Hier beträgt der Zahlenwert von  $T_N$  273,15+25.
- **$B$ (Beta)**, die Materialkonstante des NTC-Thermistors, wird auch als Temperaturkoeffizient bezeichnet und hat den Zahlenwert 3950.
- **exp** steht für die Exponentialfunktion, deren Basis die natürliche Zahl  $e$  ist und die ungefähr 2,7 beträgt.

Diese Beziehung wird durch die Formel  $T_K = 1 / (\ln(R_T / R_N) / B + 1 / T_N)$  hergeleitet, wobei die resultierende Temperatur in Kelvin um 273,15 reduziert wird, um Grad Celsius zu erhalten.

Diese Beziehung ist eine empirische Formel und gilt nur, wenn Temperatur und Widerstand innerhalb des wirksamen Bereichs liegen.

### Beispiel

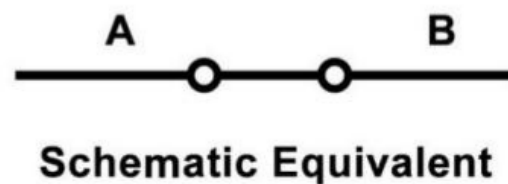
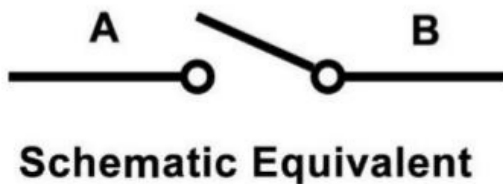
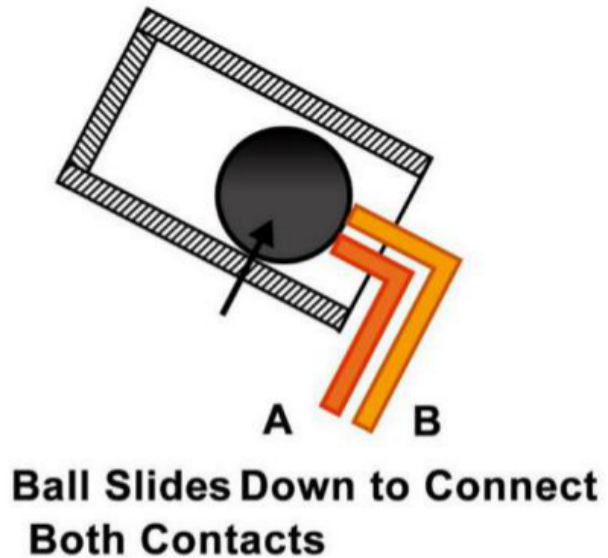
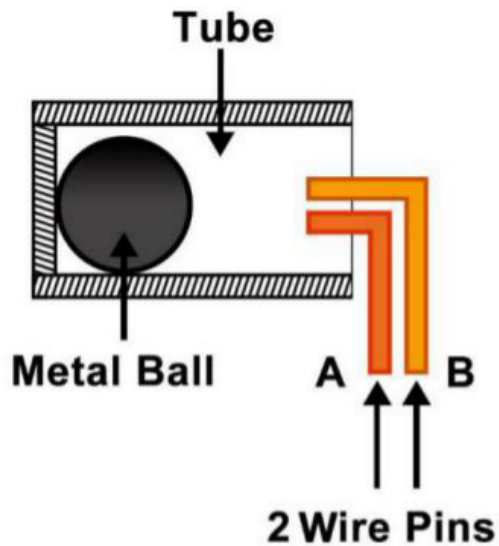
- [2.13 Thermometer](#) (Für MicroPython-Nutzer)
- [7.2 Raumtemperaturmessgerät](#) (Für MicroPython-Nutzer)
- [2.13 - Thermometer](#) (Für Arduino-Nutzer)

## 2.34 Neigungsschalter



Der hier verwendete Neigungsschalter ist ein Kugeltyp mit einer Metallkugel im Inneren. Er dient zur Erfassung kleiner Neigungswinkel.

Das Funktionsprinzip ist denkbar einfach. Wenn der Schalter in einem bestimmten Winkel geneigt wird, rollt die innenliegende Kugel herunter und berührt die beiden an die äußeren Pins angeschlossenen Kontakte, wodurch der Schaltkreis ausgelöst wird. Wird der Schalter nicht geneigt, bleibt die Kugel von den Kontakten entfernt und der Schaltkreis wird unterbrochen.



- [SW520D Neigungsschalter Datenblatt](#)

#### Beispiel

- [2.6 Neige es!](#) (Für MicroPython-Nutzer)
- [7.5 SPIEL - 10 Sekunden](#) (Für MicroPython-Nutzer)
- [2.6 - Kipp es!](#) (Für Arduino-Nutzer)
- [2.10 Fließende LEDs](#) (Für Piper Make-Nutzer)

## 2.35 Reedschalter



Der Reedschalter ist ein elektrischer Schalter, der durch ein angelegtes Magnetfeld betätigt wird. Er wurde 1936 von Walter B. Ellwood von den Bell Telephone Laboratories erfunden und am 27. Juni 1940 in den Vereinigten Staaten unter der Patentnummer 2264746 patentiert.

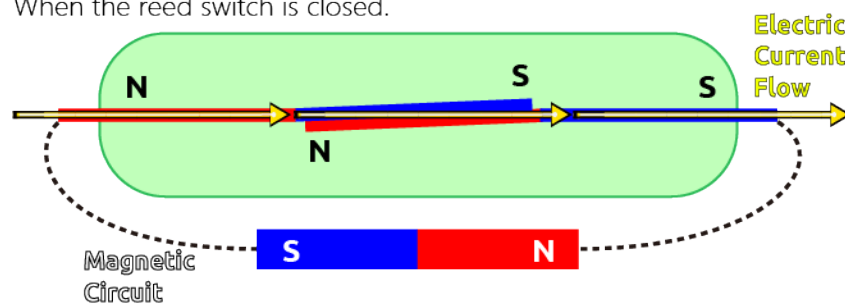
Das Funktionsprinzip eines Reedschalters ist denkbar einfach. Zwei Zungen (üblicherweise aus Eisen und Nickel, zwei Metallen) überlappen an den Endpunkten und sind in einem Glasröhrchen versiegelt. Die beiden Zungen sind durch einen kleinen Spalt von nur wenigen Mikrometern getrennt. Das Glasröhrchen ist mit einem hochreinen inerten Gas (wie Stickstoff) gefüllt; einige Reedschalter sind so konstruiert, dass sie ein Vakuum im Inneren haben, um ihre Hochspannungseigenschaften zu verbessern.

Die Zunge fungiert als magnetischer Flussleiter. Die beiden Zungen sind im Ruhezustand nicht in Kontakt; beim Durchlaufen eines Magnetfelds, erzeugt durch einen Permanentmagneten oder eine elektromagnetische Spule, nehmen die beiden Zungen an ihren Endpunkten unterschiedliche Polaritäten an. Wenn die magnetische Kraft die Federkraft der Zungen selbst übersteigt, ziehen sie sich zusammen und schließen den Schaltkreis. Wenn das Magnetfeld abnimmt oder verschwindet, trennen sich die Zungen aufgrund ihrer eigenen Elastizität, und die Kontaktflächen öffnen den Schaltkreis.

When the reed switch is open.



When the reed switch is closed.



- [Reedschalter - Wikipedia](#)

### Beispiel

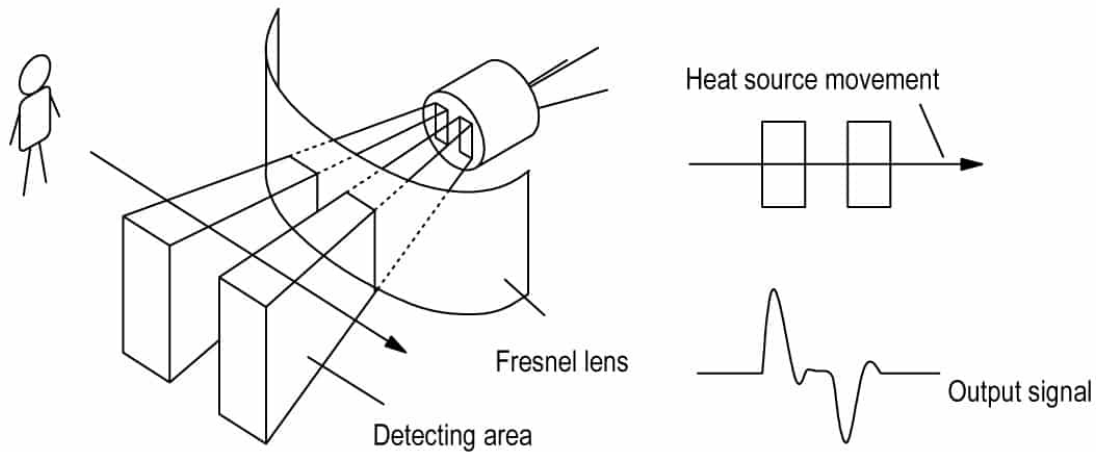
- [2.9 Fühle den Magnetismus](#) (Für MicroPython-Nutzer)
- [2.9 - Magnetismus spüren](#) (Für Arduino-Nutzer)

## 2.36 PIR-Bewegungssensormodul

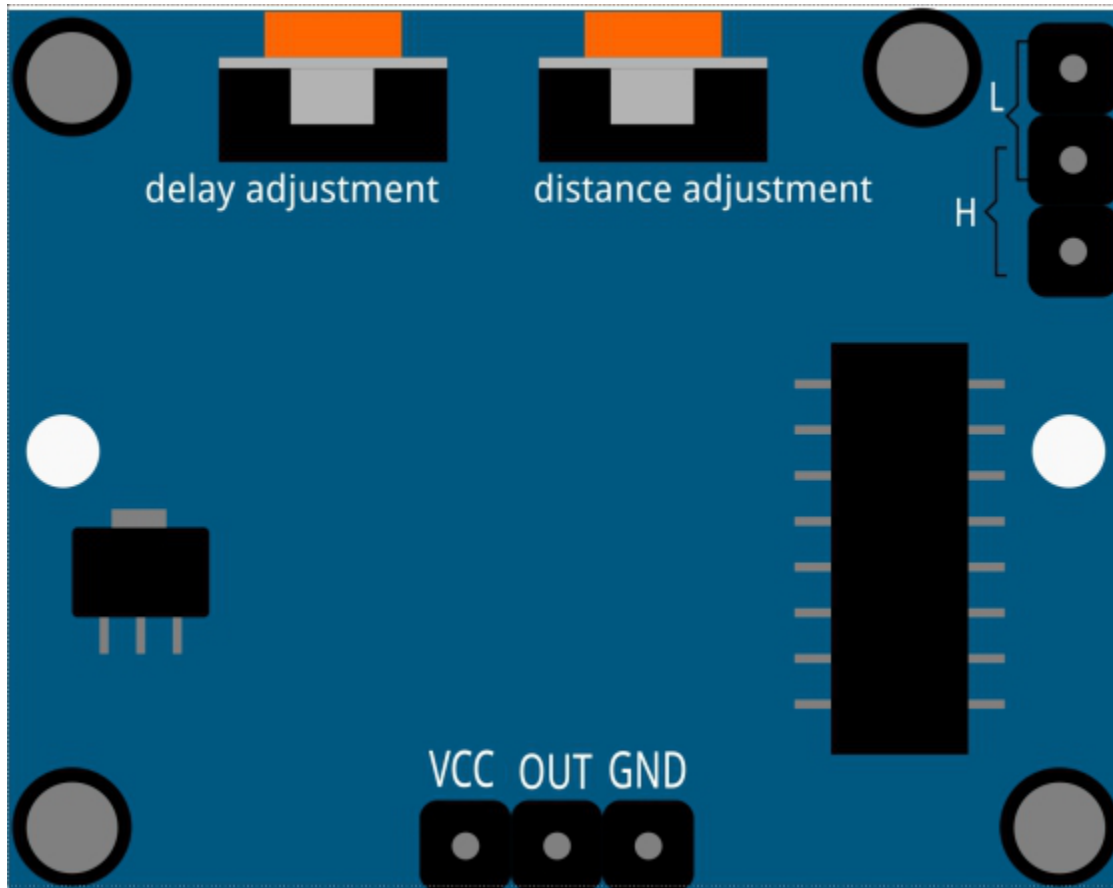


Der PIR-Sensor erkennt infrarote Wärmestrahlung, die zur Erfassung der Anwesenheit von Organismen genutzt werden kann, die infrarote Wärmestrahlung abgeben.

Der PIR-Sensor ist in zwei Schlitze unterteilt, die mit einem Differenzverstärker verbunden sind. Solange sich ein unbewegtes Objekt vor dem Sensor befindet, empfangen beide Schlitze dieselbe Menge an Strahlung, und die Ausgangsspannung ist null. Bewegt sich jedoch ein Objekt vor dem Sensor, nimmt einer der Schlitze mehr Strahlung auf als der andere, was dazu führt, dass die Ausgangsspannung ansteigt oder abfällt. Diese Veränderung der Ausgangsspannung resultiert aus der erkannten Bewegung.



Nach der Verkabelung des Sensormoduls erfolgt eine einminütige Initialisierung. Während dieser Phase kann das Modul 0 bis 3 Mal in Abständen ein Signal ausgeben. Danach befindet sich das Modul im Standby-Modus. Bitte vermeiden Sie Licht- und andere Störquellen im Erfassungsbereich des Moduls, um Fehlfunktionen zu verhindern. Idealerweise sollte das Modul auch nicht bei starkem Wind betrieben werden, da dieser ebenfalls den Sensor beeinträchtigen kann.



### Entfernungsanpassung

Durch Drehen des Entfernungsmesspotentiometers im Uhrzeigersinn wird der Erfassungsbereich vergrößert; der maximale Bereich liegt bei etwa 0-7 Metern. Dreht man es gegen den Uhrzeigersinn, verringert sich der Bereich auf ungefähr 0-3 Meter.

### Verzögerungsanpassung

Drehen Sie das Potentiometer für die Verzögerungsanpassung im Uhrzeigersinn, verlängert sich die Sensing-Verzögerung bis zu einem Maximum von 300 Sekunden. In die entgegengesetzte Richtung verkürzt sich die Verzögerung bis zu einem Minimum von 5 Sekunden.

### Zwei Auslösemodi

Die Auswahl unterschiedlicher Modi erfolgt durch das Setzen der Jumperkappe.

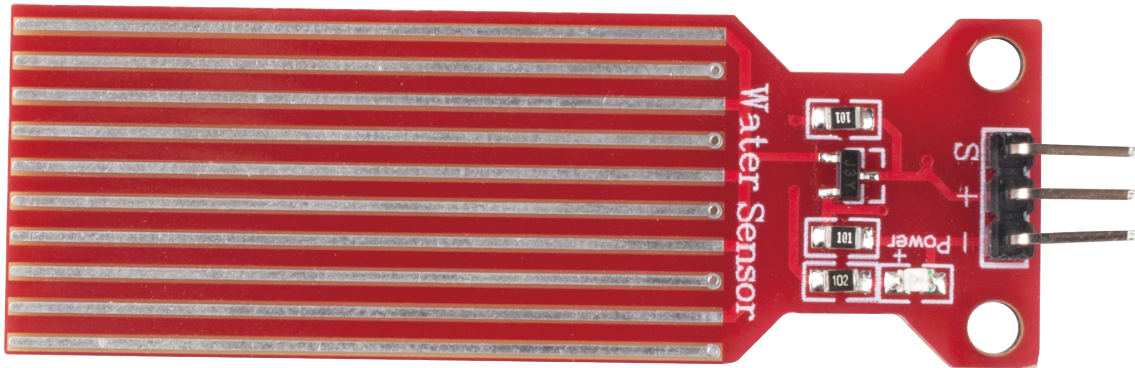
- **H:** Wiederholbarer Auslösemodus: Nach der Erfassung einer menschlichen Präsenz gibt das Modul ein Hochsignal aus. Während der anschließenden Verzögerungsperiode bleibt das Ausgangssignal bei erneutem Betreten des Erfassungsbereichs hoch.
- **L:** Einmaliger Auslösemodus: Das Modul gibt ein Hochsignal aus, wenn es eine menschliche Präsenz erfasst. Nach Ablauf der Verzögerungszeit wechselt das Ausgangssignal automatisch auf ein Niedrigsignal.

### Beispiel

- [2.10 Bewegungserkennung beim Menschen](#) (Für MicroPython-Nutzer)
- [7.4 Personen Zähler](#) (Für MicroPython-Nutzer)
- [2.10 - Menschliche Bewegung erfassen](#) (Für Arduino-Nutzer)

- [2.9 Glückskatze](#) (Für Piper Make-Nutzer)

## 2.37 Wasserspiegelsensor-Modul



Der Wasserspiegelsensor übermittelt das erfasste Wasserspiegelsignal an die Steuereinheit. Der in der Steuereinheit integrierte Computer vergleicht das gemessene Signal mit dem voreingestellten Sollwert, ermittelt die Abweichung und gibt entsprechend „Ein-“ oder „Aus“-Befehle an das Wassernachspeise-Elektroventil. So wird sichergestellt, dass der Behälter den eingestellten Wasserspiegel erreicht.

Der Wasserspiegelsensor verfügt über zehn freiliegende Kupferbahnen, von denen fünf für die Stromversorgung und fünf für die Sensorik zuständig sind. Bei Überflutung werden diese Bahnen durch das Wasser überbrückt. Auf der Platine befindet sich eine Betriebs-LED, die aufleuchtet, wenn die Platine mit Strom versorgt wird.

Die Kombination dieser Bahnen wirkt wie ein variabler Widerstand, dessen Widerstandswert sich je nach Wasserspiegel ändert. Genauer gesagt, je mehr Wasser den Sensor bedeckt, desto besser ist die Leitfähigkeit und desto geringer der Widerstand. Umgekehrt steigt der Widerstand, wenn die Leitfähigkeit abnimmt. Anschließend wird das Ausgangsspannungssignal vom Sensor verarbeitet und an den Mikrocontroller übermittelt, um uns bei der Bestimmung des Wasserspiegels zu unterstützen.

**Warnung:** Der Sensor darf nicht vollständig unter Wasser getaucht werden. Lassen Sie nur den Bereich, in dem sich die zehn Bahnen befinden, mit Wasser in Kontakt kommen. Zudem beschleunigt die Stromversorgung des Sensors in einer feuchten Umgebung die Korrosion der Sonde und verkürzt die Lebensdauer des Sensors. Daher empfehlen wir, den Sensor nur während der Messungen mit Strom zu versorgen.

### Beispiel

- [2.14 Wasserstand erfühlen](#) (Für MicroPython-Anwender)
- [2.14 - Den Wasserstand erfühlen](#) (Für Arduino-Anwender)
- [2.6 Intelligenter Wassertank](#) (Für Piper Make-Anwender)

## 2.38 Ultraschallmodul



- **TRIG:** Trigger-Impulseingang
- **ECHO:** Echo-Impulsausgang
- **GND:** Masse
- **VCC:** 5V Versorgungsspannung

Es handelt sich hierbei um den HC-SR04 Ultraschall-Distanzsensor, der eine berührungslose Messung von 2 cm bis 400 cm mit einer Reichweitengenauigkeit von bis zu 3 mm ermöglicht. Das Modul umfasst einen Ultraschall-Sender, einen Empfänger und eine Steuerschaltung.

Zur Inbetriebnahme sind lediglich 4 Pins anzuschließen: VCC (Stromversorgung), Trig (Trigger), Echo (Empfang) und GND (Masse). Dies macht das Modul besonders benutzerfreundlich für Ihre Messprojekte.

### Merkmale

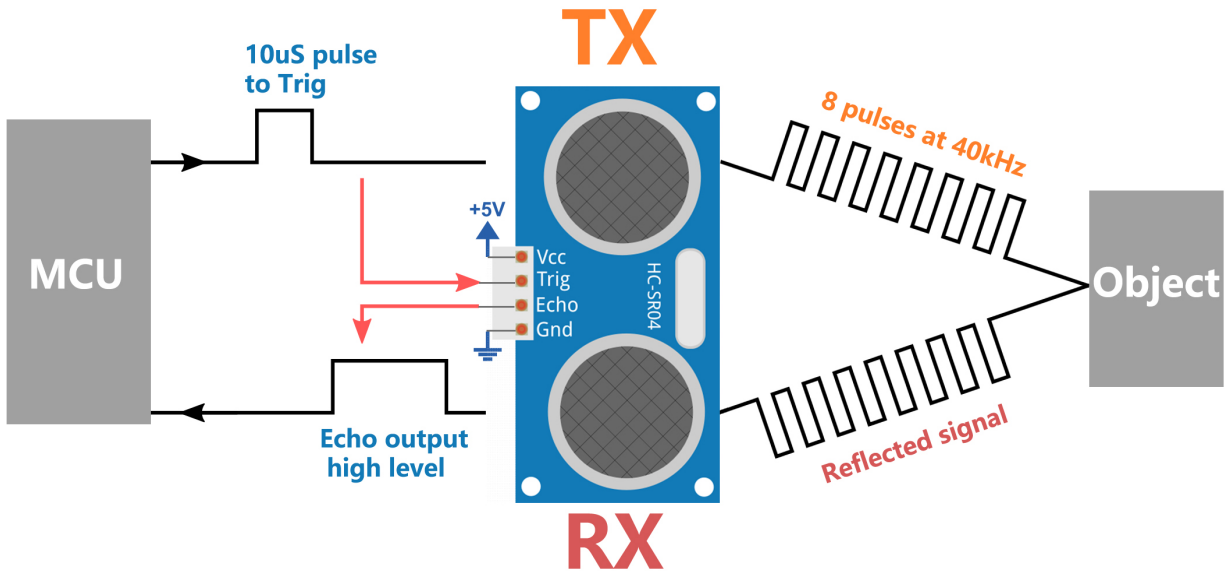
- Betriebsspannung: DC5V
- Arbeitsstrom: 16mA
- Arbeitsfrequenz: 40Hz
- Maximale Reichweite: 500cm
- Minimale Reichweite: 2cm
- Trigger-Eingangssignal: 10uS TTL-Impuls
- Echo-Ausgangssignal: TTL-Pegelsignal und reichweitenproportionales Signal
- Anschluss: XH2.54-4P
- Abmessungen: 46x20.5x15 mm

### Funktionsprinzip

Die Grundlagen sind wie folgt:

- Auslösung über IO mit einem mindestens 10us hohen Pegelsignal.
- Das Modul sendet eine 8-Zyklus-Ultraschall-Burst bei 40 kHz aus und prüft, ob ein Pulssignal zurückkommt.
- Echo gibt ein Hochpegelsignal aus, wenn ein Signal zurückkommt; die Dauer dieses Hochpegels entspricht der Zeit von der Aussendung bis zur Rückkehr.
- Distanz = (Hochpegelzeit x Schallgeschwindigkeit (340M/S)) / 2





Formel:

- $us / 58 = \text{Entfernung in Zentimetern}$
- $us / 148 = \text{Entfernung in Zoll}$
- $\text{Distanz} = \text{Hochpegelzeit} \times \text{Schallgeschwindigkeit (340M/S)} / 2$

**Bemerkung:** Dieses Modul sollte nicht unter Spannung angeschlossen werden. Falls notwendig, sollte zuerst der GND-Pin des Moduls verbunden werden, um eine Beeinträchtigung der Funktionalität zu vermeiden.

Die Fläche des zu messenden Objekts sollte mindestens 0,5 Quadratmeter betragen und möglichst flach sein. Andernfalls werden die Messergebnisse beeinträchtigt.

#### Beispiel

- [6.1 Abstandsmessung](#) (Für MicroPython-Anwender)
- [7.10 Einparkhilfe](#) (Für MicroPython-Anwender)
- [6.1 - Abstandsmessung](#) (Für Arduino-Anwender)
- [2.11 Rückfahrsystem](#) (Für Piper Make-Anwender)

## 2.39 DHT11 Temperatur- und Feuchtigkeitssensor

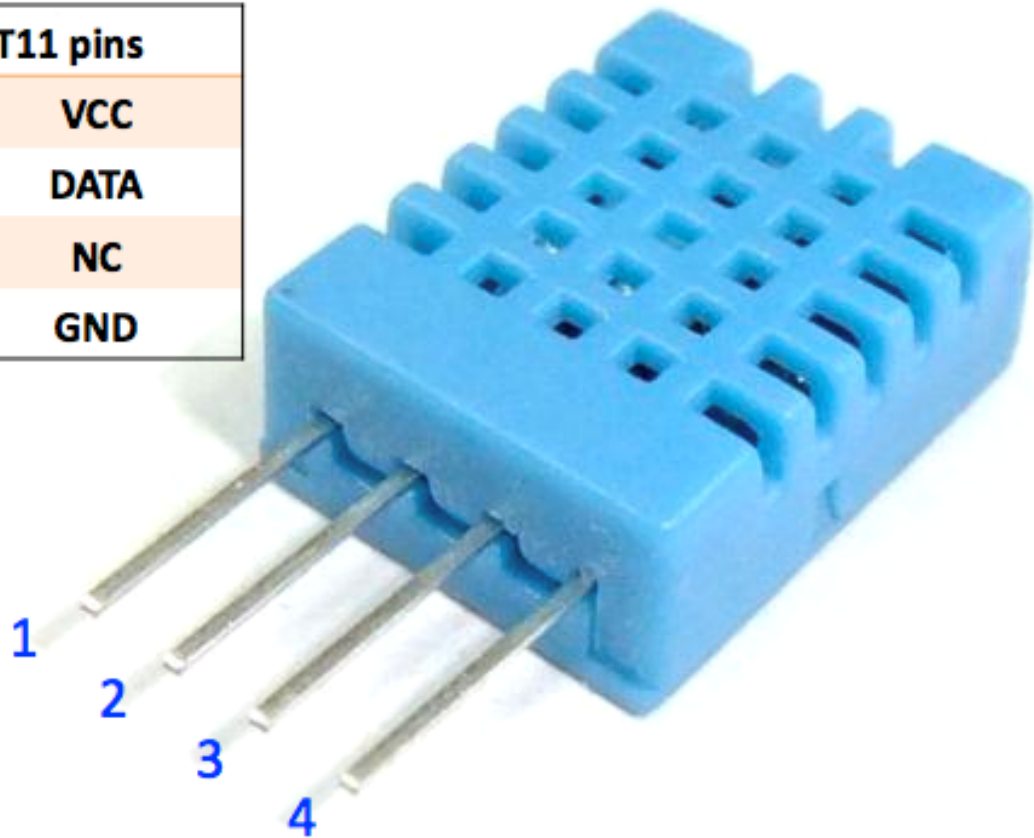
Der digitale Temperatur- und Feuchtigkeitssensor DHT11 ist ein Verbundsensor, der kalibrierte digitale Signalausgaben für Temperatur und Feuchtigkeit liefert. Dank der Anwendung von speziellen digitalen Modulen und Sensortechnologie für Temperatur und Feuchtigkeit zeichnet sich das Produkt durch hohe Zuverlässigkeit und ausgezeichnete Langzeitstabilität aus.

Der Sensor enthält eine resistive Feuchtigkeitmesskomponente und ein NTC-Temperaturmessgerät, die mit einem leistungsfähigen 8-Bit-Mikrocontroller verbunden sind.

Zur Verfügung stehen nur drei Pins: VCC, GND und DATA. Der Kommunikationsprozess beginnt mit dem Senden von Startsignalen über die DATA-Leitung an DHT11. Daraufhin empfängt DHT11 die Signale und sendet ein Antwortsignal zurück. Der Host empfängt das Antwortsignal und beginnt dann, 40-Bit-Temperatur- und Feuchtigkeits-

daten zu empfangen (8-Bit-Feuchtigkeitsganzzahl + 8-Bit-Feuchtigkeitsdezimal + 8-Bit-Temperaturganzzahl + 8-Bit-Temperaturdezimal + 8-Bit-Prüfsumme).

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



### Funktionen

1. Messbereich für Feuchtigkeit: 20 - 90 %RH
2. Messbereich für Temperatur: 0 - 60°C
3. Ausgabe digitaler Signale für Temperatur und Feuchtigkeit
4. Betriebsspannung: DC 5V; PCB-Größe: 2,0 x 2,0 cm
5. Genauigkeit der Feuchtigkeitsmessung:  $\pm 5$  %RH
6. Genauigkeit der Temperaturmessung:  $\pm 2^\circ\text{C}$

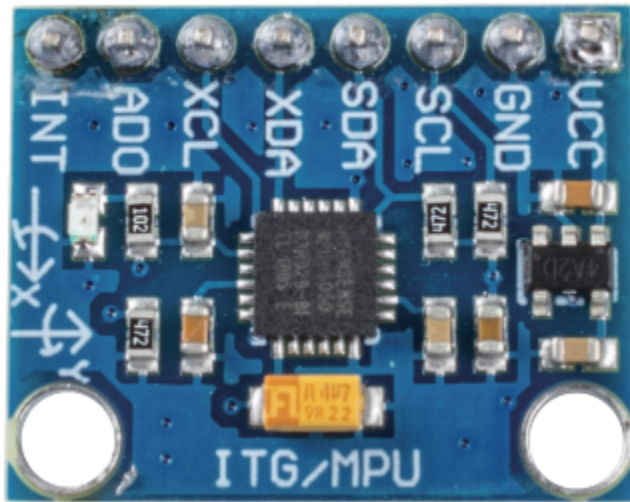
- [DHT11 Datenblatt](#)

### Beispiel

- [6.2 Temperatur - Feuchtigkeit](#) (Für MicroPython-Anwender)
- [6.2 - Temperatur - Feuchtigkeit](#) (Für Arduino-Anwender)

## 2.40 MPU6050 Modul

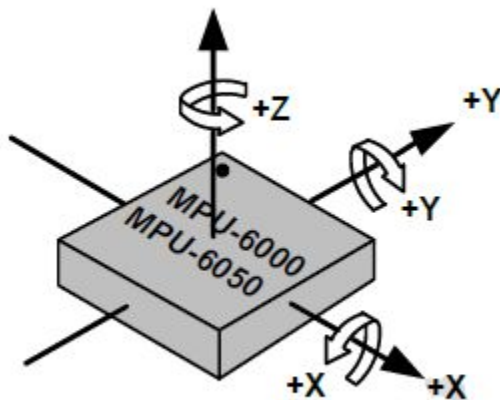
### MPU6050



Der MPU-6050 ist ein 6-Achsen-Motion-Tracking-Gerät, das einen 3-Achsen-Gyroskop mit einem 3-Achsen-Beschleunigungsmesser kombiniert.

Die drei Koordinatensysteme sind wie folgt definiert:

Legen Sie das MPU6050 flach auf den Tisch, sodass die Seite mit dem Aufkleber 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 die Y-Achse definiert.

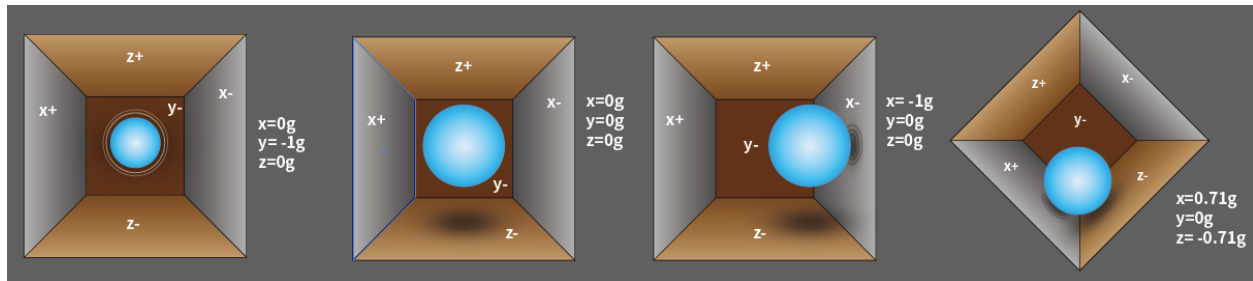


### 3-Achsen-Beschleunigungsmesser

Der Beschleunigungsmesser funktioniert nach dem Prinzip des piezoelektrischen Effekts, der Fähigkeit bestimmter Materialien, eine elektrische Ladung in Reaktion auf mechanischen Druck 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, wird die Kugel aufgrund der Schwerkraft in Richtung der Neigung bewegt. Die Wand, gegen die die Kugel prallt, erzeugt winzige piezoelektrische Ströme. Jedes

Paar gegenüberliegender Wände in einem Quader entspricht einer Achse im 3D-Raum: X, Y und Z. Abhängig von dem erzeugten Strom der piezoelektrischen Wände können wir die Neigungsrichtung und deren Ausmaß bestimmen.



Das MPU6050 kann dazu verwendet werden, die Beschleunigung auf jeder Koordinatenachse zu erkennen (im ruhenden Desktop-Zustand beträgt die Z-Achsen-Beschleunigung 1 Gravitationskraft und die X- und Y-Achsen sind 0). Bei Neigung oder in einem Zustand der Schwerelosigkeit/Überlastung ändert sich der entsprechende Messwert.

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

Die Beschleunigung wird folgendermaßen in einen Beschleunigungswert umgewandelt:

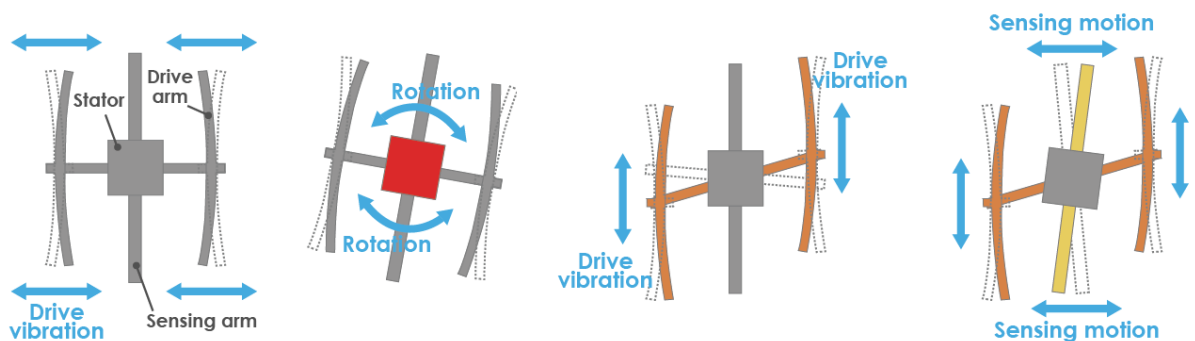
Beschleunigung = (Rohdaten der Beschleunigungsachsen / 65536 \* voller Skalenbeschleunigungsbereich) g

Am Beispiel der X-Achse: Wenn die Rohdaten der Beschleunigung der X-Achse 16384 betragen und der Bereich als +/-2g ausgewählt ist:

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

### 3-Achsen-Gyroskop

Gyroskope funktionieren nach dem Prinzip der Coriolis-Beschleunigung. Stellen Sie sich eine gabelähnliche Struktur vor, die ständig in Bewegung ist. Sie wird durch piezoelektrische Kristalle an Ort und Stelle gehalten. Wenn Sie diese Anordnung zu neigen versuchen, erfahren die Kristalle eine Kraft in Richtung der Neigung. Diese wird durch die Trägheit der beweglichen Gabel erzeugt. Die Kristalle erzeugen dadurch einen Strom, der mit dem piezoelektrischen Effekt in Einklang steht, und dieser Strom wird verstärkt.



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.

Das Gyroskop hat ebenfalls vier Messbereiche: +/- 250, +/- 500, +/- 1000, +/- 2000. Die Berechnungsmethode und die Beschleunigung sind im Wesentlichen konsistent.

Die Formel zur Umwandlung des Messwerts in die Winkelgeschwindigkeit lautet:

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

Anhand der X-Achse: Wenn die Rohdaten der X-Achse des Beschleunigungsmessers 16384 betragen und der Bereich +/- 250°/s beträgt:

**Winkelgeschwindigkeit entlang der X-Achse =  $(16384 / 65536 * 500)^\circ/\text{s} = 125^\circ/\text{s}$**

**Beispiel**

- [6.3 6-Achsen-Bewegungsverfolgung](#) (Für MicroPython-Nutzer)
- [7.11 Somatosensorische Steuerung](#) (Für MicroPython-Nutzer)
- [7.12 Digitaler Wasserwaage](#) (Für MicroPython-Nutzer)
- [6.3 - 6-Achsen-Bewegungsverfolgung](#) (Für Arduino-Nutzer)



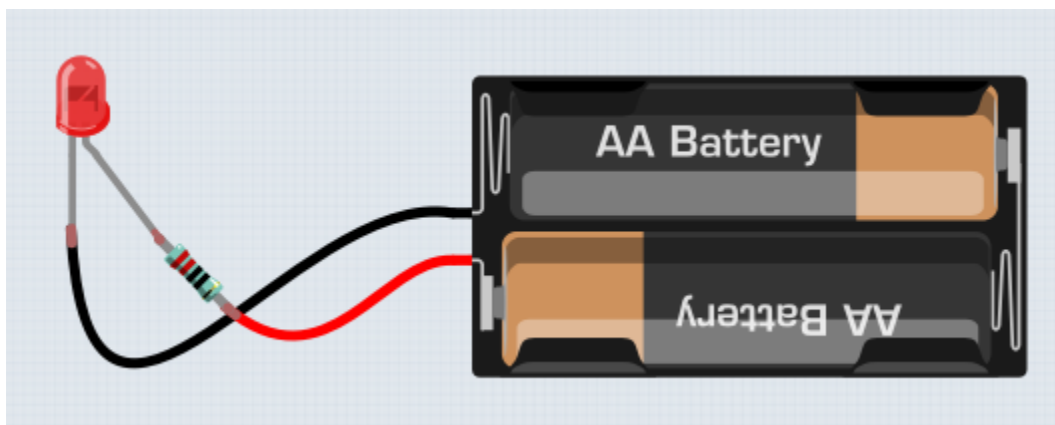
## Elektrischer Schaltkreis

Viele Dinge, die du täglich benutzt, werden elektrisch betrieben, sei es die Beleuchtung in deinem Zuhause oder der Computer, auf dem du gerade dies liest.

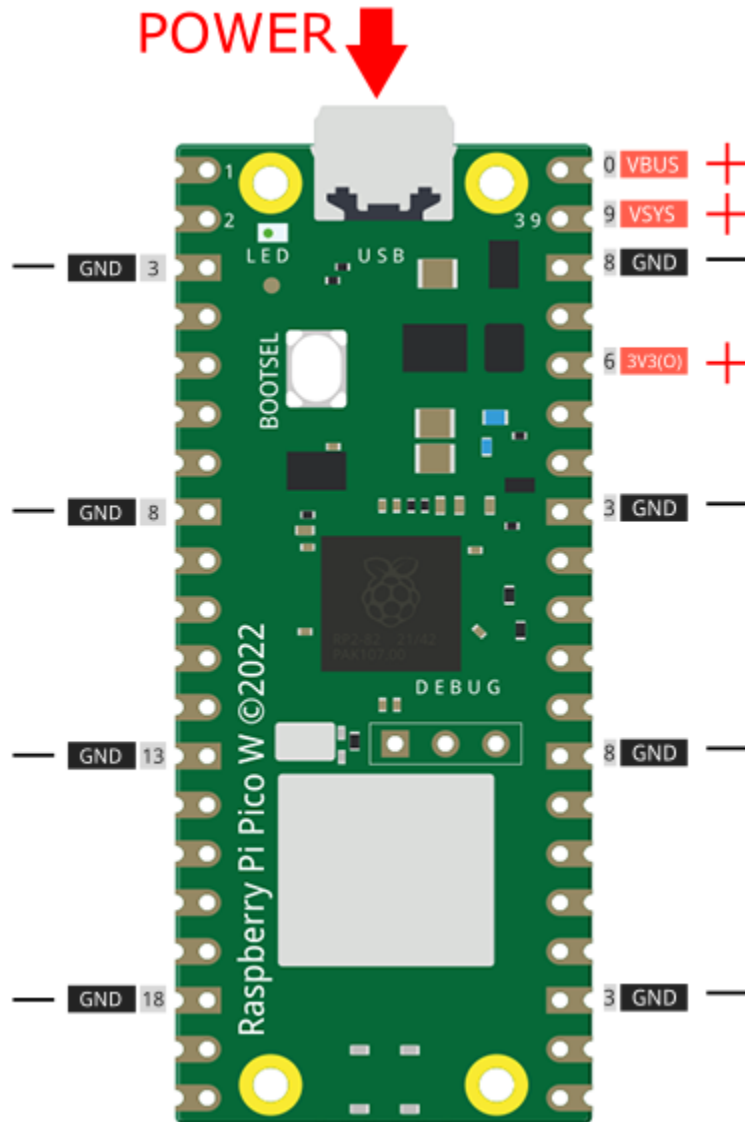
Um Elektrizität nutzen zu können, muss ein elektrischer Schaltkreis geschaltet werden. Ein solcher besteht aus Metallleitungen sowie elektrischen und elektronischen Bauteilen.

Jeder Schaltkreis benötigt eine Stromquelle. In deinem Haushalt werden die meisten Geräte (z.B. Fernseher, Lampen) über Steckdosen mit Energie versorgt. Viele kleinere, mobile Schaltkreise (z.B. elektronisches Spielzeug, Mobiltelefone) hingegen laufen auf Batterien. Eine Batterie hat zwei Anschlüsse: einen positiven, der mit einem Pluszeichen (+) markiert ist, und einen negativen, der durch ein Minuszeichen (-) symbolisiert wird, jedoch meist nicht auf der Batterie vermerkt ist.

Damit ein Stromfluss entsteht, muss ein leitfähiger Pfad den positiven mit dem negativen Anschluss der Batterie verbinden. Man spricht dann von einem geschlossenen Schaltkreis (im Gegensatz dazu steht der offene Schaltkreis, bei dem die Verbindung getrennt ist). Elektrischer Strom durchfließt dann Geräte wie Lampen und bringt sie zum Leuchten.



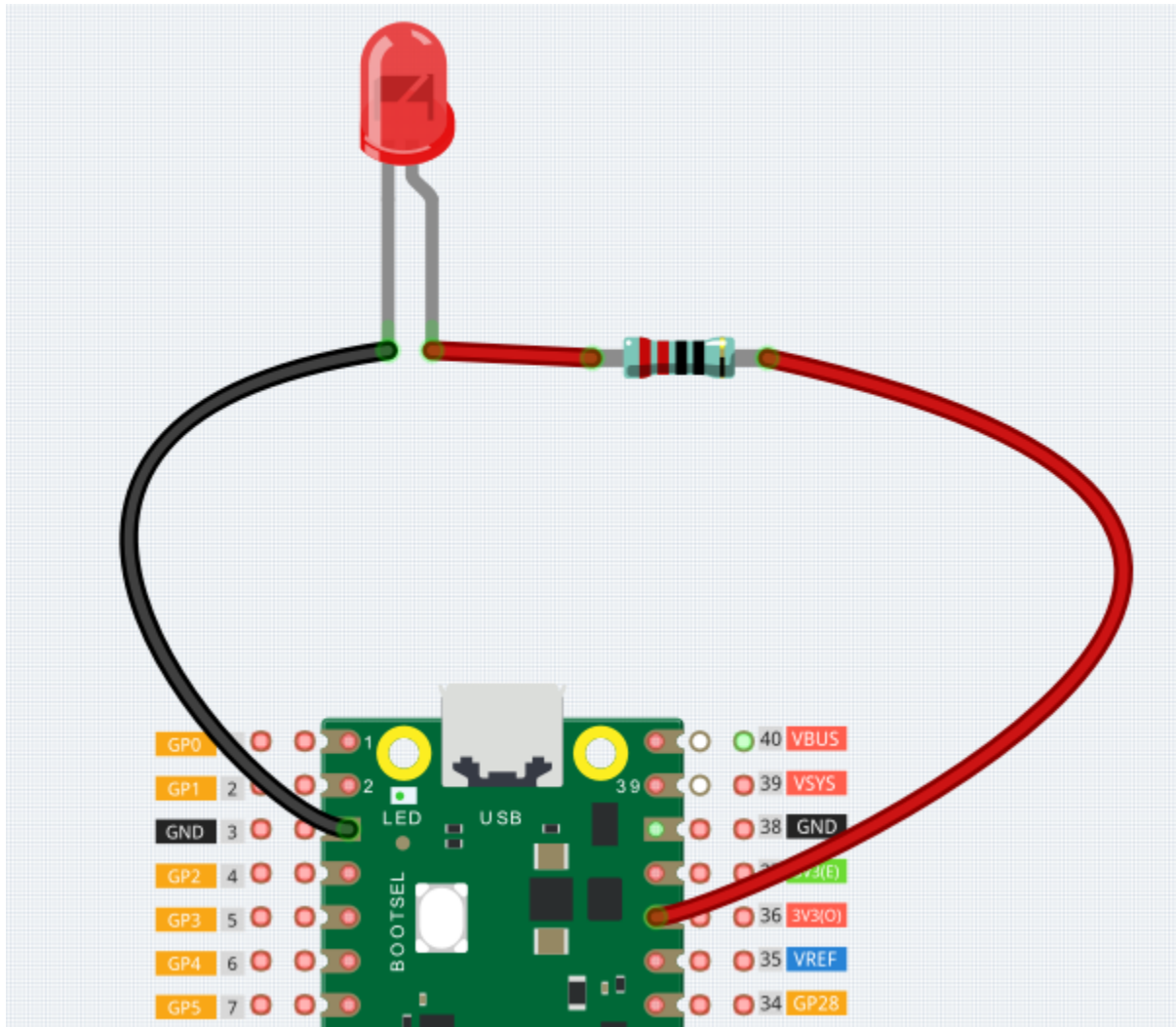
Ein Pico W hat einige Ausgangspins für die Stromversorgung (positiv) und einige Massepins (negativ). Diese Pins können als positive und negative Anschlüsse für die Stromversorgung verwendet werden, indem der Pico W an eine Energiequelle angeschlossen wird.



Mit Elektrizität lassen sich Werke mit Licht, Ton und Bewegung realisieren. Man kann eine LED zum Leuchten bringen, indem man den längeren Pin an den positiven und den kürzeren an den negativen Anschluss anschließt. Ohne Vorkehrungen würde die LED jedoch schnell kaputtgehen, weshalb ein 220\*-Ohm-Widerstand im Schaltkreis eingefügt werden muss.

Die darzustellende Schaltung sieht wie folgt aus.





Vielleicht fragst du dich jetzt: Wie baue ich diesen Schaltkreis zusammen? Halte ich die Kabel mit der Hand oder klebe ich die Pins und Kabel fest?

In diesem Fall sind steckbare Experimentierplatten (Breadboards) deine besten Helfer.

### 3.1 Hallo, Steckplatine!

Eine Steckplatine ist eine rechteckige Kunststoffplatte mit vielen kleinen Löchern. Diese erlauben es uns, elektronische Bauteile einfach einzustecken und elektrische Schaltungen zu bauen. Die Bauteile werden nicht permanent fixiert, sodass wir bei einem Fehler den Schaltkreis einfach reparieren oder von vorn beginnen können.

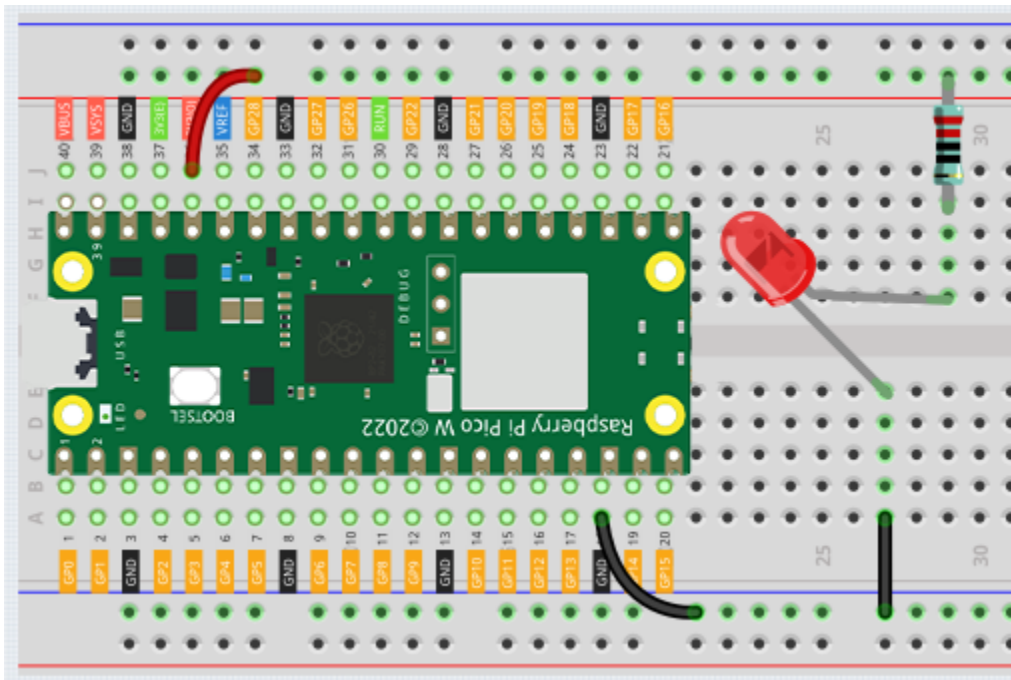
**Bemerkung:** Spezielle Werkzeuge sind für die Verwendung von Steckplatten nicht erforderlich. Da jedoch viele elektronische Bauteile sehr klein sind, können Pinzetten hilfreich sein, um kleinere Teile besser greifen zu können.

Im Internet finden sich zahlreiche Informationen zu Steckplatten.

- [Wie benutzt man eine Steckplatine - Science Buddies](#)
- [Was ist eine STECKPLATINE? - Makezine](#)

Es gibt ein paar Dinge, die du über Steckplatinen wissen solltest.

1. Jede halbe Reihengruppe (wie die Spalte A-E in Reihe 1 oder Spalte F-J in Reihe 3) ist intern verbunden. Wenn also ein elektrisches Signal an A1 anliegt, kann es auch an B1, C1, D1, E1, jedoch nicht an F1 oder A2 austreten.
2. In den meisten Fällen werden beide Seiten der Steckplatine als Stromschienen verwendet, und die Löcher in jeder Spalte (etwa 50 Löcher) sind miteinander verbunden. Üblicherweise werden positive Stromanschlüsse in der Nähe des roten Drahts und negative in der Nähe des blauen Drahts angeschlossen.
3. In einem Schaltkreis fließt der Strom vom positiven zum negativen Pol, nachdem er die Last durchquert hat. In diesem Fall könnte ein Kurzschluss auftreten.



Lassen Sie uns nun den Schaltkreis entsprechend der Stromflussrichtung aufbauen!

1. In diesem Schaltkreis verwenden wir den 3V3-Pin des Pico W-Boards, um die LED mit Strom zu versorgen. Verwenden Sie ein Steckbrückenkabel (M2M), um es mit der roten Stromschiene zu verbinden.
2. Um die LED zu schützen, muss der Strom durch einen 220-Ohm-Widerstand fließen. Verbinden Sie ein Ende (es ist egal welches) des Widerstands mit der roten Stromschiene und das andere mit einer freien Reihe der Steckplatine (in meiner Schaltung Reihe 24).

---

**Bemerkung:** Die Farbringe des 220-Ohm-Widerstands sind rot, rot, schwarz, schwarz und braun.

---

3. Wenn Sie die LED aufheben, sehen Sie, dass einer der Anschlüsse länger ist als der andere. Verbinden Sie den längeren Anschluss mit derselben Reihe wie der Widerstand und den kürzeren Anschluss mit einer Reihe auf der gegenüberliegenden Seite der Mittellücke auf der Steckplatine.

---

**Bemerkung:** Der längere Anschluss ist die Anode und repräsentiert die positive Seite des Schaltkreises; der kürzere ist die Kathode und steht für die negative Seite.

Die Anode sollte über einen Widerstand mit dem GPIO-Pin verbunden werden; die Kathode sollte mit dem GND-Pin verbunden werden.

---

4. Verwenden Sie ein Steckbrückenkabel (M2M), um den kürzeren LED-Anschluss mit der negativen Stromschiene der Steckplatine zu verbinden.
5. Verbinden Sie den GND-Pin des Pico W mit der negativen Stromschiene über ein Steckbrückenkabel.

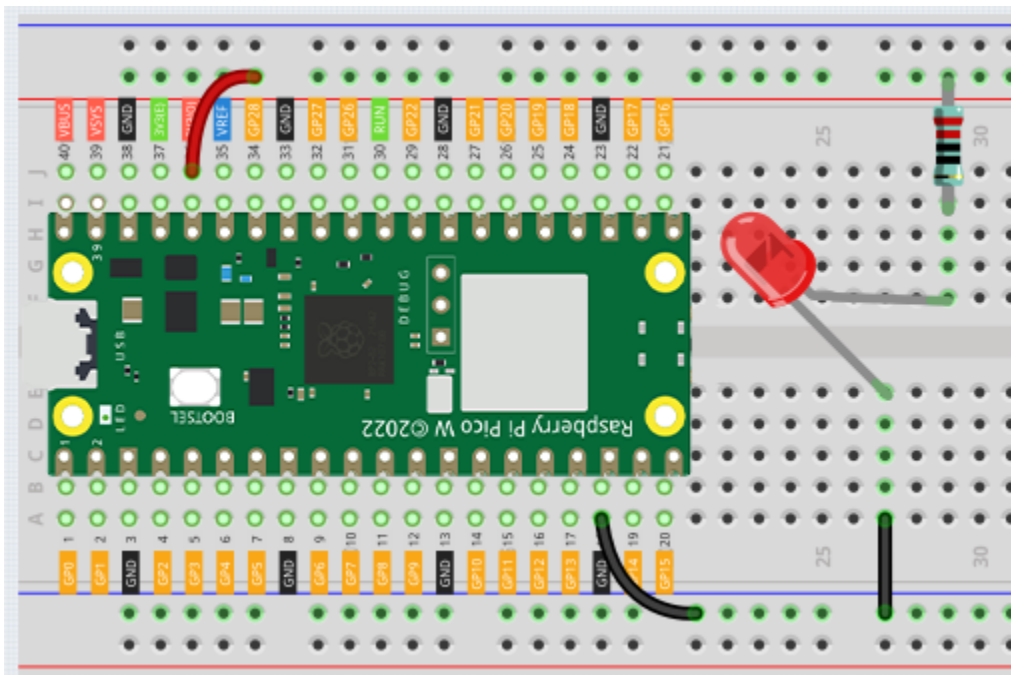
## 3.2 Vorsicht vor Kurzschlüssen

Kurzschlüsse können entstehen, wenn zwei Bauteile, die eigentlich nicht miteinander verbunden sein sollten, „versehentlich“ Kontakt aufnehmen. Dieses Set beinhaltet Widerstände, Transistoren, Kondensatoren, LEDs und mehr, deren lange Metallstifte sich berühren und einen Kurzschluss verursachen können. Manche Schaltkreise funktionieren einfach nicht mehr richtig, wenn ein Kurzschluss auftritt. In Einzelfällen kann ein Kurzschluss die Bauteile dauerhaft beschädigen, insbesondere wenn die Stromversorgung und die Masseleitung betroffen sind. Das kann dazu führen, dass der Schaltkreis stark erhitzt, das Plastik auf der Steckplatine schmilzt und sogar die Bauteile verbrennen!

Stellen Sie deshalb immer sicher, dass die Stifte der elektronischen Bauteile auf der Steckplatine einander nicht berühren.

## 3.3 Ausrichtung des Schaltkreises

Schaltkreise haben eine Orientierung, die bei bestimmten elektronischen Bauteilen eine entscheidende Rolle spielt. Einige Geräte haben eine Polarität, d.h., sie müssen gemäß ihrer positiven und negativen Pole korrekt angeschlossen werden. Falsch ausgerichtete Schaltkreise funktionieren nicht einwandfrei.



Wenn Sie die LED in diesem einfachen Schaltkreis, den wir zuvor gebaut haben, umdrehen, werden Sie feststellen, dass sie nicht mehr funktioniert.

Im Gegensatz dazu haben manche Bauteile keine Ausrichtung, wie zum Beispiel die Widerstände in diesem Schaltkreis. Diese können Sie umkehren, ohne den normalen Betrieb von LEDs zu beeinträchtigen.

Die meisten Komponenten und Module mit Beschriftungen wie „+“, „-“, „GND“, „VCC“ oder unterschiedlich langen Pins müssen auf eine spezielle Weise an den Schaltkreis angeschlossen werden.

### 3.4 Schutz des Schaltkreises

Stromstärke ist die Geschwindigkeit, mit der Elektronen an einem Punkt in einem vollständigen elektrischen Stromkreis fließen. Einfach ausgedrückt: Strom = Fluss. Ein Ampere (Amper), oder kurz Amp, ist die internationale Einheit für die Messung der Stromstärke. Sie drückt die Menge der Elektronen (manchmal als „elektrische Ladung“ bezeichnet) aus, die an einem Punkt im Stromkreis über eine bestimmte Zeit hinweg fließen.

Die treibende Kraft (Spannung) hinter dem Stromfluss wird als Spannung bezeichnet und in Volt (V) gemessen.

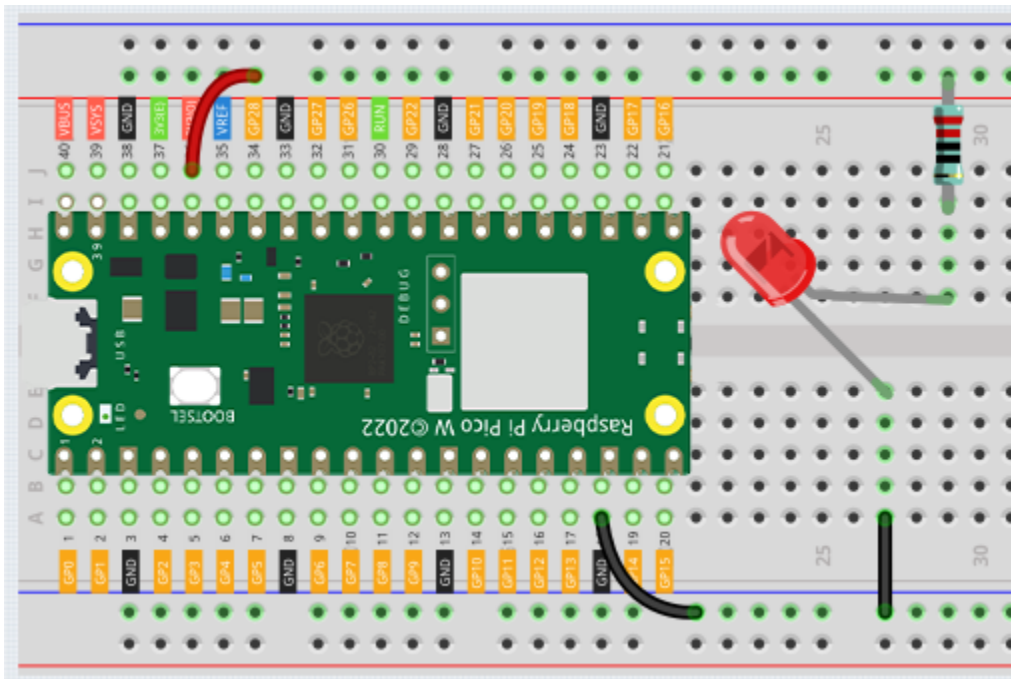
Widerstand (R) ist die Eigenschaft des Materials, die den Stromfluss einschränkt, und wird in Ohm ( $\Omega$ ) gemessen.

Laut Ohmschem Gesetz (solange die Temperatur konstant bleibt), sind Stromstärke, Spannung und Widerstand proportional zueinander. Die Stromstärke eines Schaltkreises ist proportional zu seiner Spannung und umgekehrt proportional zu seinem Widerstand.

Daher gilt: Stromstärke (I) = Spannung (V) / Widerstand (R).

- [Ohmsches Gesetz - Wikipedia](#)

Zum Ohmschen Gesetz können wir ein einfaches Experiment durchführen.



Wenn Sie den Draht, der 3V3 mit 5V verbindet (d.h. VBUS, der 40. Pin des Pico W), ändern, wird die LED heller. Wenn Sie den Widerstand von 220 Ohm auf 1000 Ohm ändern (Farbring: braun, schwarz, schwarz, braun, braun), werden Sie feststellen, dass die LED dunkler wird. Je größer der Widerstand, desto dunkler die LED.

**Bemerkung:** Für eine Einführung in Widerstände und wie man den Widerstand berechnet, siehe [Widerstand](#).

Die meisten vorgepackten Module benötigen lediglich Zugang zur richtigen Spannung (meistens 3,3V oder 5V), wie beispielsweise Ultraschallmodule.

In Ihren selbstgebaute Schaltungen sollten Sie jedoch auf die Versorgungsspannung und den Einsatz von Widerständen für elektrische Geräte achten.

Als Beispiel verbrauchen LEDs normalerweise 20mA und haben einen Spannungsabfall von etwa 1,8V. Laut Ohmschem Gesetz benötigen wir bei einer 5V-Stromversorgung mindestens einen 160-Ohm-Widerstand  $((5-1,8)/20\text{mA})$ ,

um die LED nicht durchbrennen zu lassen.



---

## Für MicroPython-Nutzer

---

In diesem Abschnitt erfahren Sie die Geschichte von MicroPython, wie Sie MicroPython auf dem Pico W installieren können, die Grundlagen der Syntax und eine Reihe interessanter sowie praktischer Projekte, die Ihnen beim schnellen Erlernen von MicroPython helfen.

Wir empfehlen, die Kapitel in der vorgeschlagenen Reihenfolge zu lesen.

### 1. Einstieg

## 4.1 1.1 Einführung in MicroPython

MicroPython ist eine in C geschriebene Software-Implementierung einer Programmiersprache, die weitgehend mit Python 3 kompatibel ist und für den Betrieb auf Mikrocontrollern optimiert wurde.

MicroPython besteht aus einem Python-Compiler für Bytecode und einem Laufzeitinterpreter dieses Bytecodes. Dem Benutzer wird eine interaktive Eingabeaufforderung (die REPL) zur Verfügung gestellt, über die sofort unterstützte Befehle ausgeführt werden können. Darüber hinaus sind ausgewählte Python-Basisbibliotheken enthalten. MicroPython bietet Module, die dem Programmierer Zugang zur Low-Level-Hardware ermöglichen.

- Referenz: [MicroPython - Wikipedia](#)

### 4.1.1 Die Geschichte beginnt hier

Die Dinge änderten sich 2013, als Damien George eine Crowdfunding-Kampagne (Kickstarter) startete.

Damien war ein Student der Universität Cambridge und ein begeisterter Roboterprogrammierer. Er wollte die Welt von Python von einer Gigabyte-Maschine auf einen Kilobyte verkleinern. Seine Kickstarter-Kampagne sollte seine Entwicklungsarbeit unterstützen, während er seinen Prototyp in eine fertige Implementierung verwandelte.

MicroPython wird von einer vielfältigen Pythonista-Gemeinschaft unterstützt, die ein großes Interesse am Erfolg des Projekts hat.

Neben dem Testen und Unterstützen des Codebasis stellten die Entwickler Tutorials, Codebibliotheken und Hardware-Portierungen bereit, sodass Damien sich auf andere Aspekte des Projekts konzentrieren konnte.

- Referenz: [realpython](#)

### 4.1.2 Warum MicroPython?

Obwohl die ursprüngliche Kickstarter-Kampagne MicroPython als Entwicklungsboard „pyboard“ mit STM32F4 herausbrachte, unterstützt MicroPython viele ARM-basierte Produktarchitekturen. Zu den hauptsächlich unterstützten Ports gehören ARM Cortex-M (viele STM32-Boards, TI CC3200/WiPy, Teensy-Boards, Nordic nRF-Serie, SAMD21 und SAMD51), ESP8266, ESP32, 16-Bit-PIC, Unix, Windows, Zephyr und JavaScript. Zweitens ermöglicht MicroPython schnelles Feedback, da man REPL für interaktive Befehlseingaben nutzen und sofortige Antworten erhalten kann. Man kann sogar Code anpassen und ihn sofort ausführen, anstatt den Code-Kompilieren-Hochladen-Ausführen-Zyklus zu durchlaufen.

Während Python dieselben Vorteile bietet, sind einige Mikrocontrollerboards wie der Raspberry Pi Pico klein, einfach und haben wenig Speicher, um die Python-Sprache überhaupt auszuführen. Deshalb hat sich MicroPython weiterentwickelt, hat die wichtigsten Python-Funktionen beibehalten und eine Reihe neuer Funktionen hinzugefügt, um mit diesen Mikrocontrollerboards zu arbeiten.

Als Nächstes lernen Sie, wie Sie MicroPython auf den Raspberry Pi Pico installieren können.

- Referenz: [MicroPython - Wikipedia](#)
- Referenz: [realpython](#)

## 4.2 1.2 Installation der Thonny IDE

Bevor Sie mit der Programmierung des Pico mit MicroPython beginnen können, benötigen Sie eine integrierte Entwicklungsumgebung (IDE). Hier empfehlen wir Thonny. Thonny bringt Python 3.7 bereits integriert mit, sodass nur ein einfacher Installer benötigt wird und Sie sofort mit dem Programmieren beginnen können.

---

**Bemerkung:** Da der Raspberry Pi Pico Interpreter nur mit der Thonny-Version 3.3.3 oder höher funktioniert, können Sie dieses Kapitel überspringen, wenn Sie diese Version bereits haben; anderenfalls aktualisieren oder installieren Sie sie bitte.

---

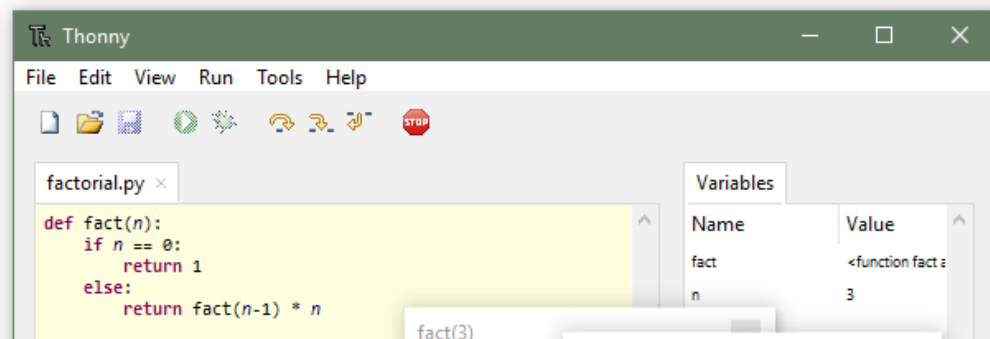
1. Den Download finden Sie auf der Website. Sobald Sie die Seite öffnen, sehen Sie oben rechts ein hellgraues Kästchen. Klicken Sie auf den für Ihr Betriebssystem zutreffenden Link.

**Thonny**  
Python IDE for beginners



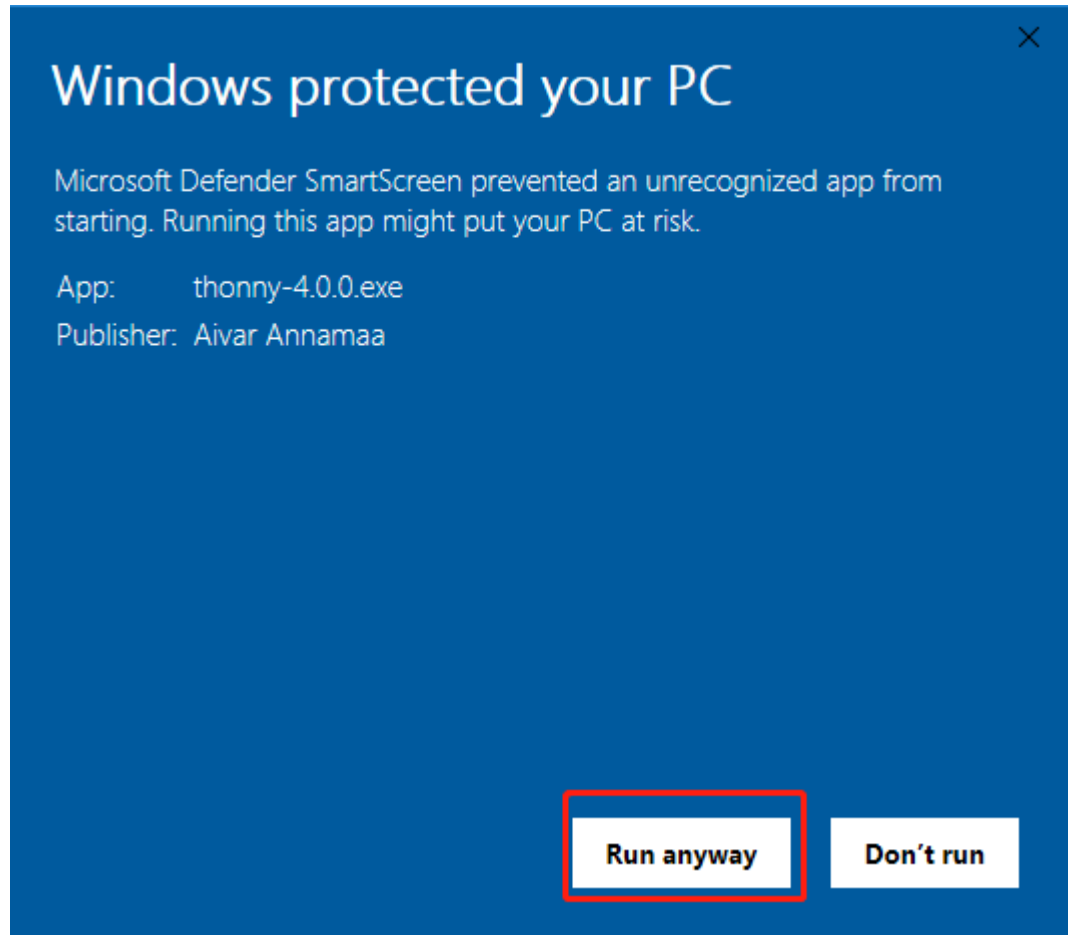
Download version **4.0.0** for

**Windows • Mac • Linux**





- Die Installationsprogramme sind mit einem neuen Zertifikat signiert, das noch keinen guten Ruf hat. Möglicherweise müssen Sie Browser-Warnmeldungen bestätigen (z.B. in Chrome „Behalten“ anstelle von „Verwerfen“ wählen) sowie die Warnung des Windows Defenders (**Mehr Infos** **Trotzdem ausführen**).



- Klicken Sie anschließend auf **Weiter** und **Installieren**, um die Installation von Thonny abzuschließen.

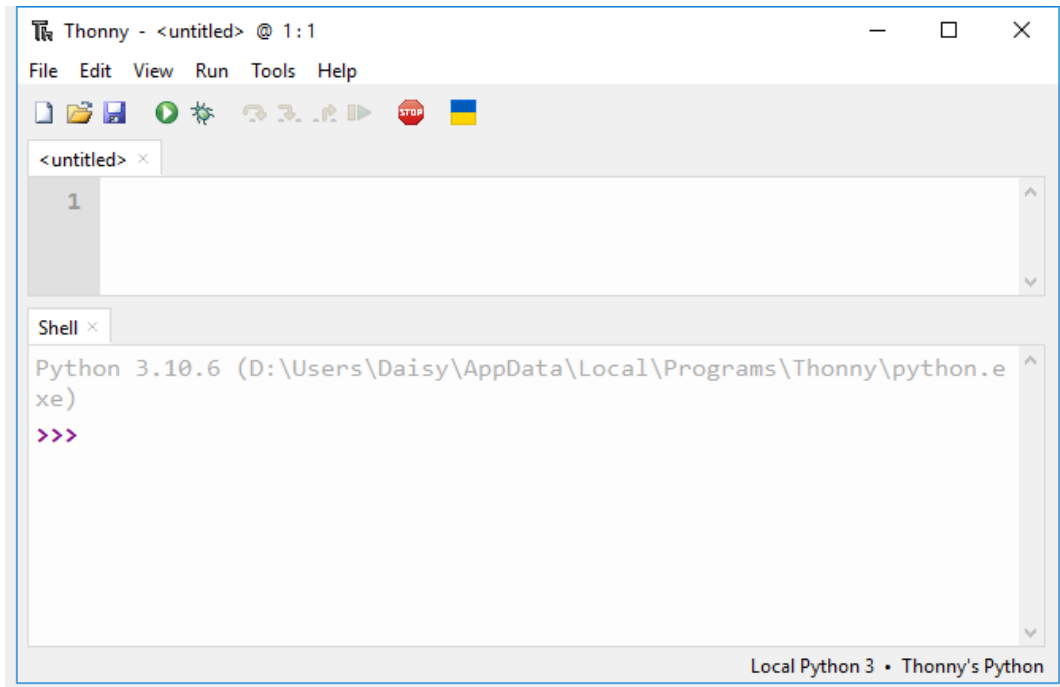


## 4.3 1.3 Installation von MicroPython auf Ihrem Pico

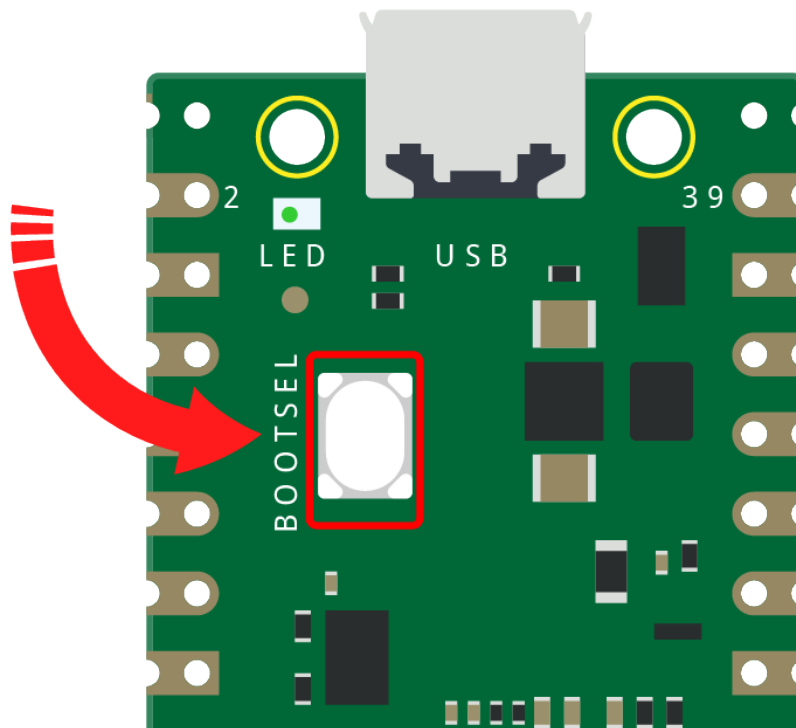
Jetzt geht es darum, MicroPython auf dem Raspberry Pi Pico zu installieren. Die Thonny IDE ermöglicht Ihnen eine äußerst komfortable Installation mit nur einem Klick.

**Bemerkung:** Falls Sie Thonny nicht aktualisieren möchten, können Sie die offizielle der Raspberry Pi Foundation nutzen, indem Sie eine `rp2_pico_xxxx.uf2` Datei per Drag-and-Drop auf den Raspberry Pi Pico ziehen.

1. Öffnen Sie die Thonny IDE.

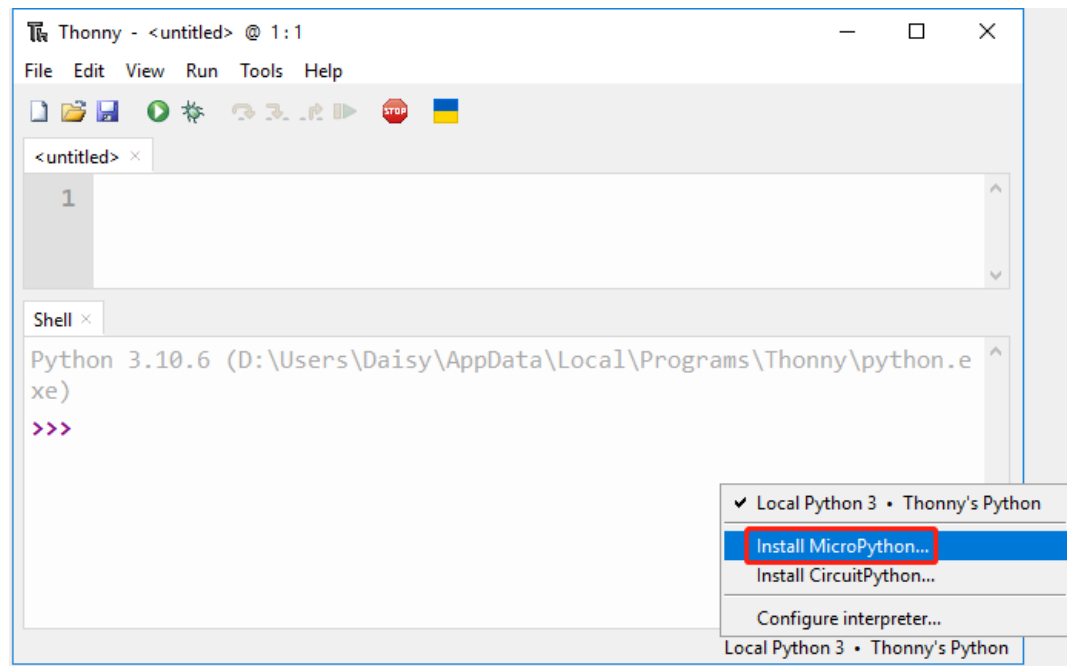


2. Halten Sie die **BOOTSEL**-Taste gedrückt und schließen Sie den Pico über ein Micro-USB-Kabel an den Computer an. Lassen Sie die **BOOTSEL**-Taste los, sobald Ihr Pico als Massenspeichergerät mit dem Namen **RPI-RP2** erkannt wird.

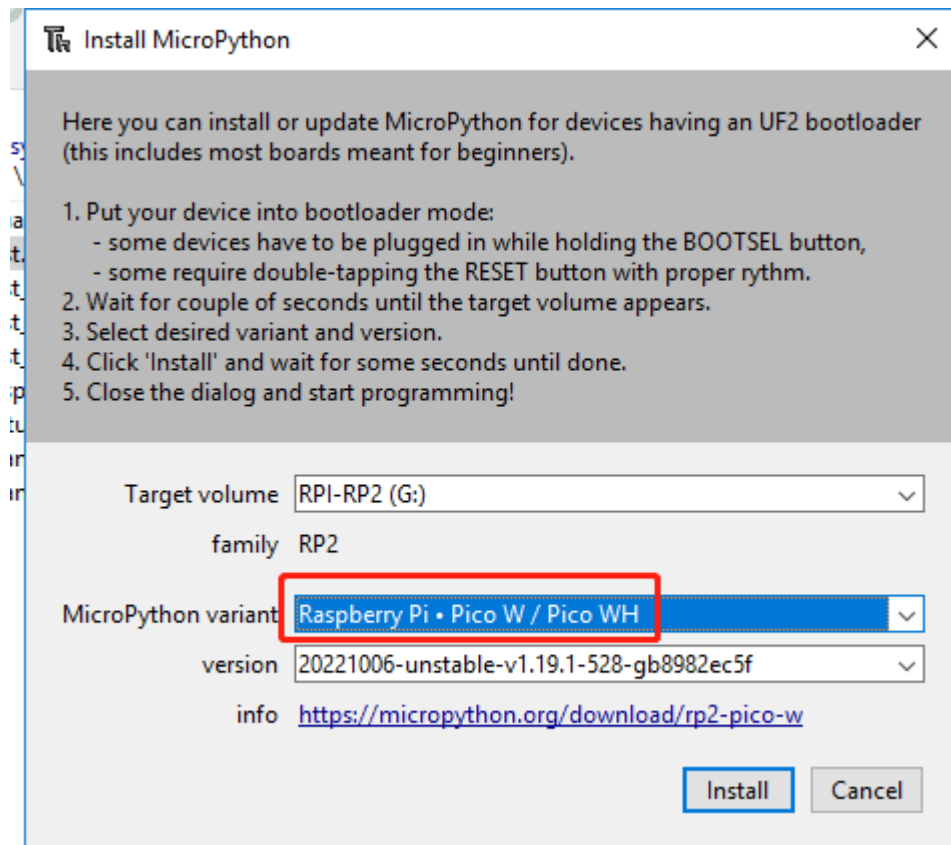


3. Klicken Sie in der unteren rechten Ecke auf die Schaltfläche zur Interpreter-Auswahl und wählen Sie **MicroPython** installieren.

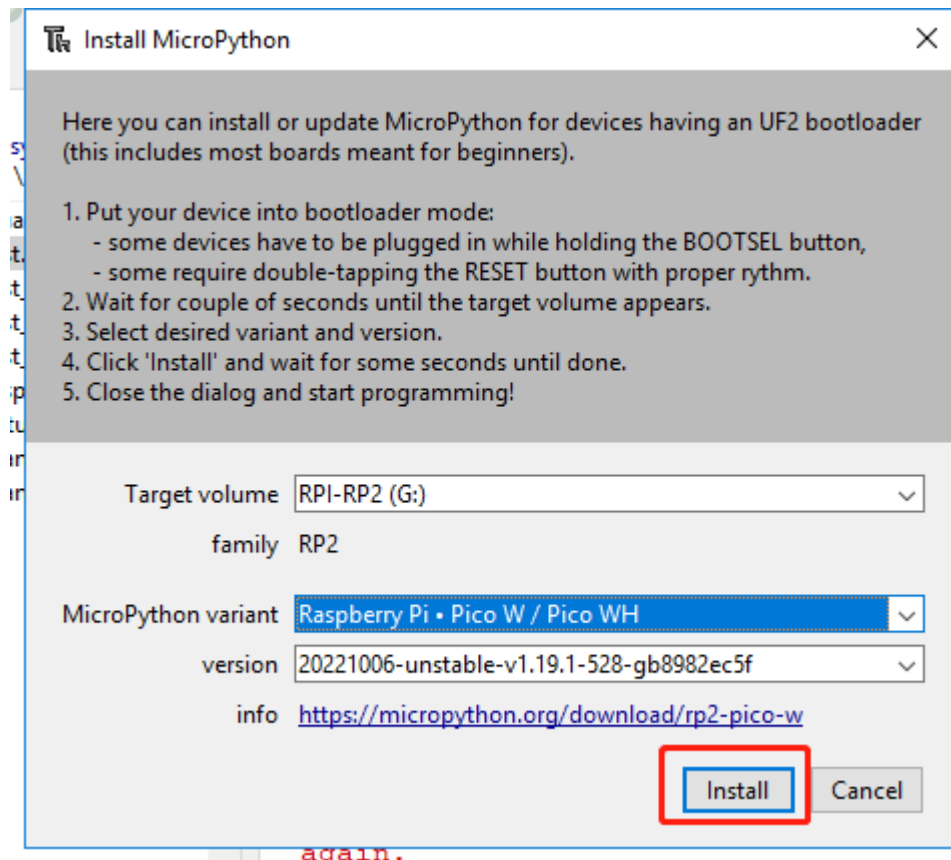
**Bemerkung:** Falls diese Option in Ihrer Thonny-Version nicht verfügbar ist, aktualisieren Sie bitte auf die neueste Version.



4. Im Feld **Ziellaufwerk** erscheint automatisch das Laufwerk des gerade angeschlossenen Pico, und im Feld **MicroPython-Variante** wählen Sie **Raspberry Pi Pico/Pico H**.



5. Klicken Sie auf die Schaltfläche **Installieren**, warten Sie, bis die Installation abgeschlossen ist, und schließen Sie dann diese Seite.

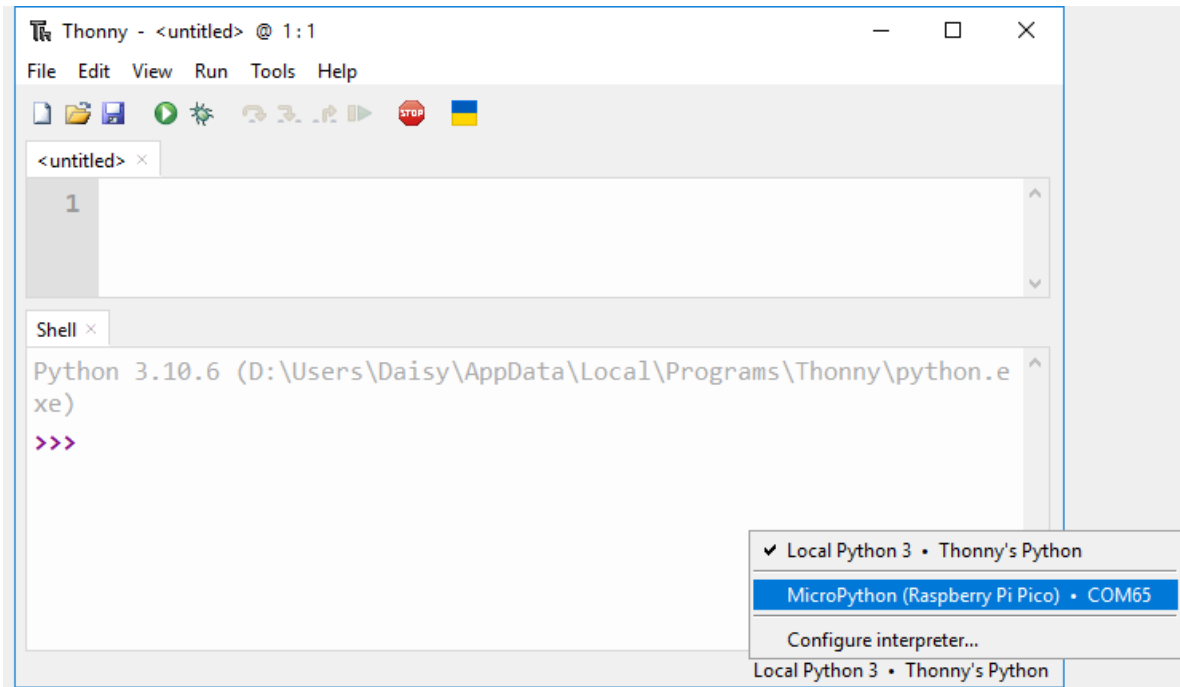


Herzlichen Glückwunsch, Ihr Raspberry Pi Pico ist jetzt einsatzbereit.

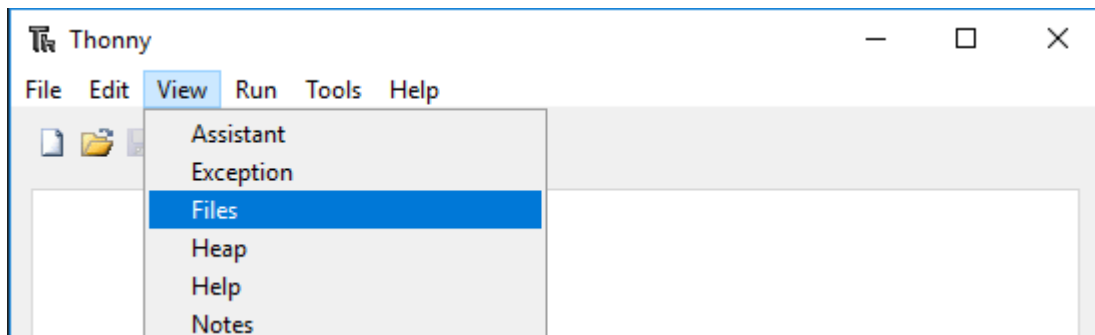
## 4.4 1.4 Bibliotheken auf den Pico hochladen

Für einige Projekte sind zusätzliche Bibliotheken erforderlich. In diesem Abschnitt wird erläutert, wie diese Bibliotheken zunächst auf den Raspberry Pi Pico W hochgeladen werden können, um später den Code direkt auszuführen.

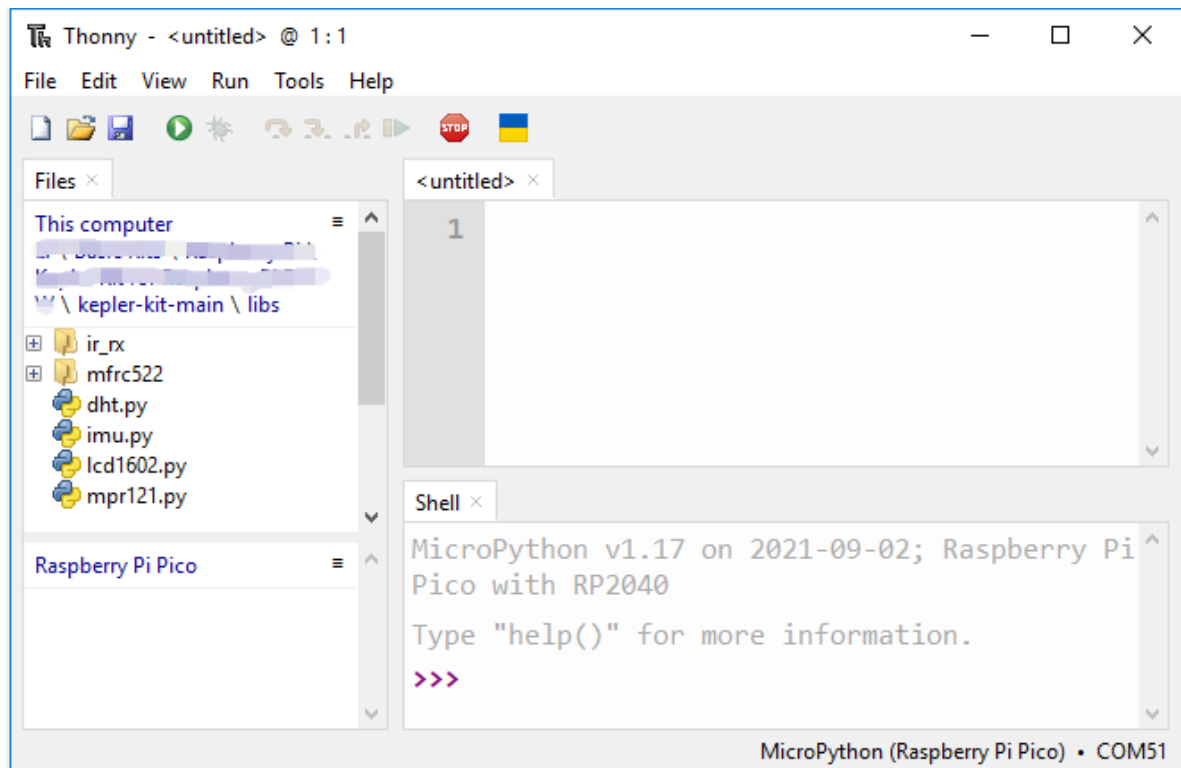
1. Laden Sie den zugehörigen Code über den folgenden Link herunter.
  - SunFounder Kepler Kit
2. Öffnen Sie die Thonny IDE und schließen Sie den Pico mit einem Mikro-USB-Kabel an Ihren Computer an. Klicken Sie dann in der unteren rechten Ecke auf den Interpreter „MicroPython (Raspberry Pi Pico).COMXX“.



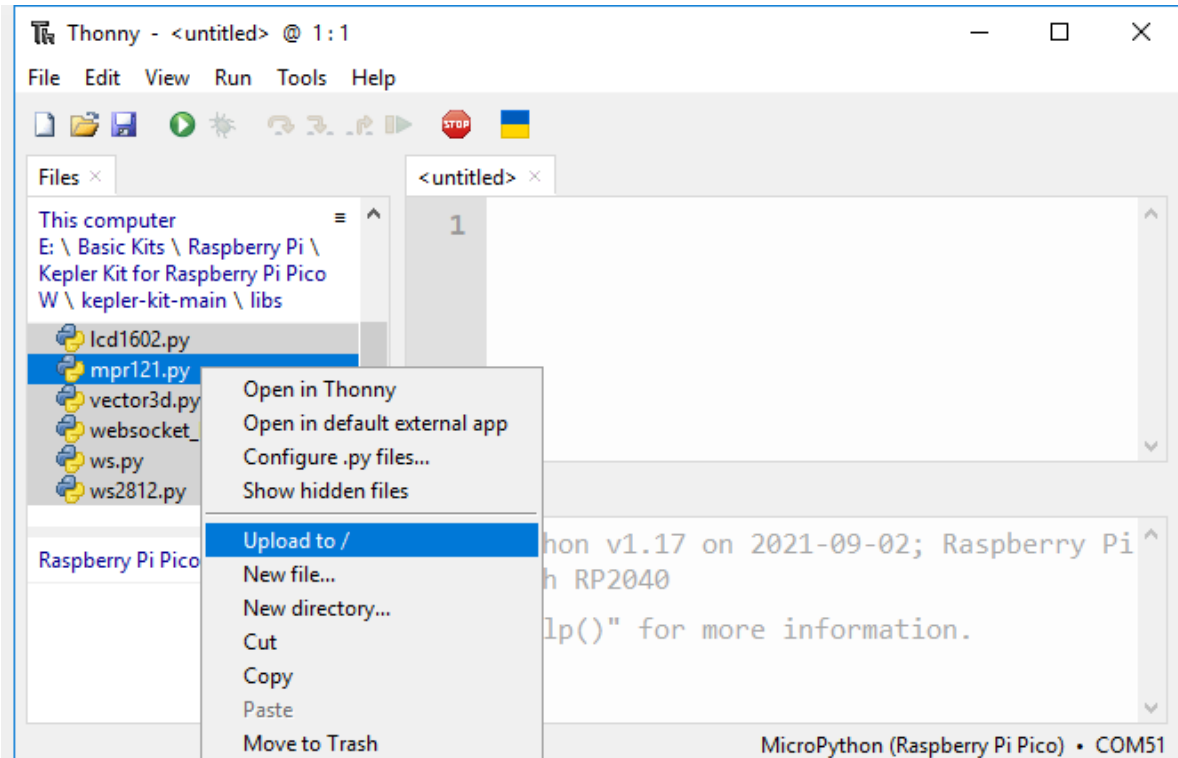
3. Navigieren Sie in der oberen Menüleiste zu **Ansicht** -> **Dateien**.



4. Wechseln Sie den Pfad zu dem Ordner, in dem Sie zuvor das [Code-Paket](#) heruntergeladen haben. Anschließend navigieren Sie zum Ordner kepler-kit-main/libs.

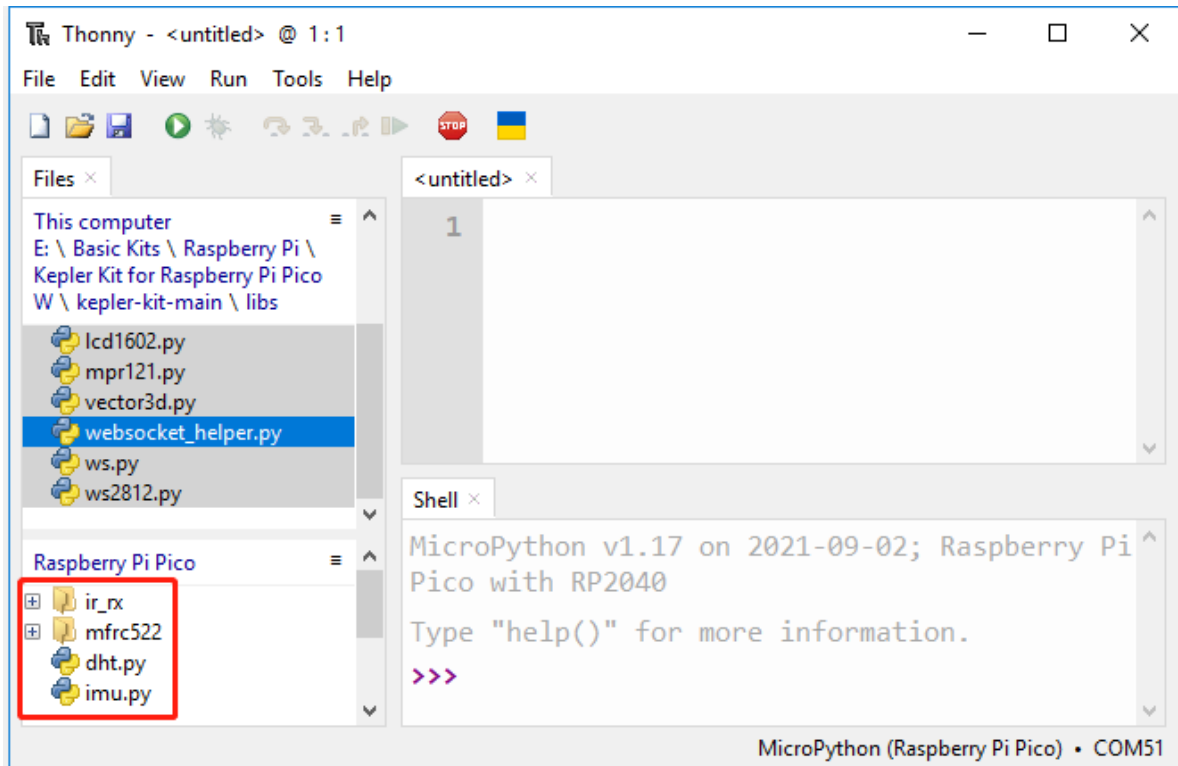


5. Markieren Sie alle Dateien und Ordner im Verzeichnis `libs/`, klicken Sie mit der rechten Maustaste und wählen Sie **Hochladen zu** aus. Der Upload kann einige Zeit dauern.



6. Nun sollten Sie die gerade hochgeladenen Dateien auf Ihrem Laufwerk Raspberry Pi Pico sehen können.





## 4.5 1.5 Schnelle Einführung in Thonny

### 4.5.1 Code direkt öffnen und ausführen

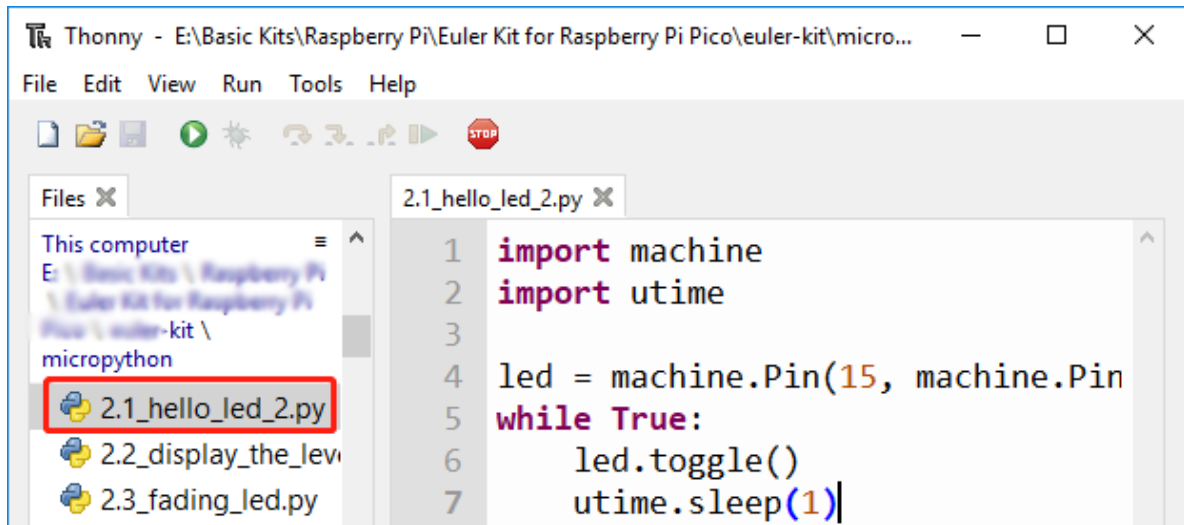
Der Codeabschnitt in den Projekten informiert Sie genau darüber, welcher Code verwendet wird. Doppelklicken Sie daher auf die .py-Datei mit der Seriennummer im Pfad kepler-kit-main/micropython/, um sie zu öffnen.

Vorher müssen Sie jedoch das Paket herunterladen und die Bibliothek hochladen, wie in [1.4 Bibliotheken auf den Pico hochladen](#) beschrieben.

1. Code öffnen.

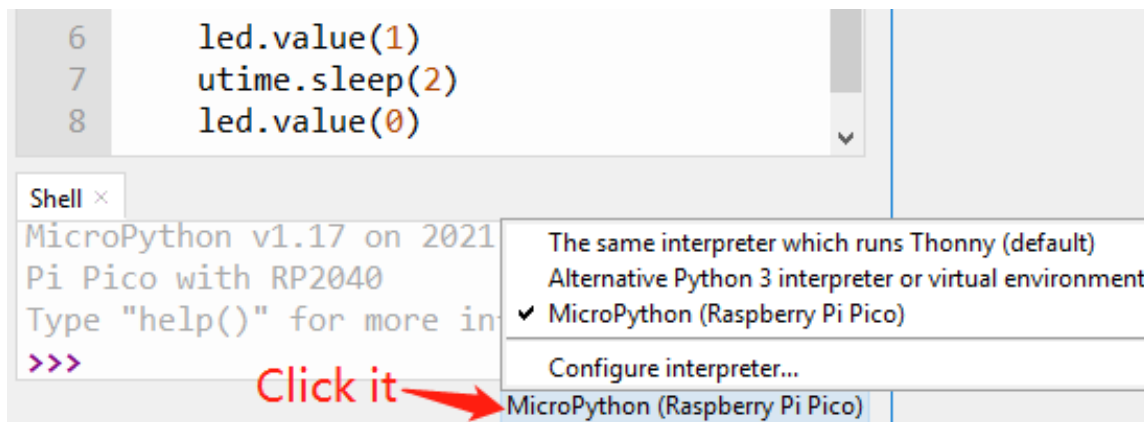
Zum Beispiel 2.1\_hello\_led.py.

Wenn Sie darauf doppelklicken, öffnet sich ein neues Fenster rechts. Sie können gleichzeitig mehrere Codes öffnen.



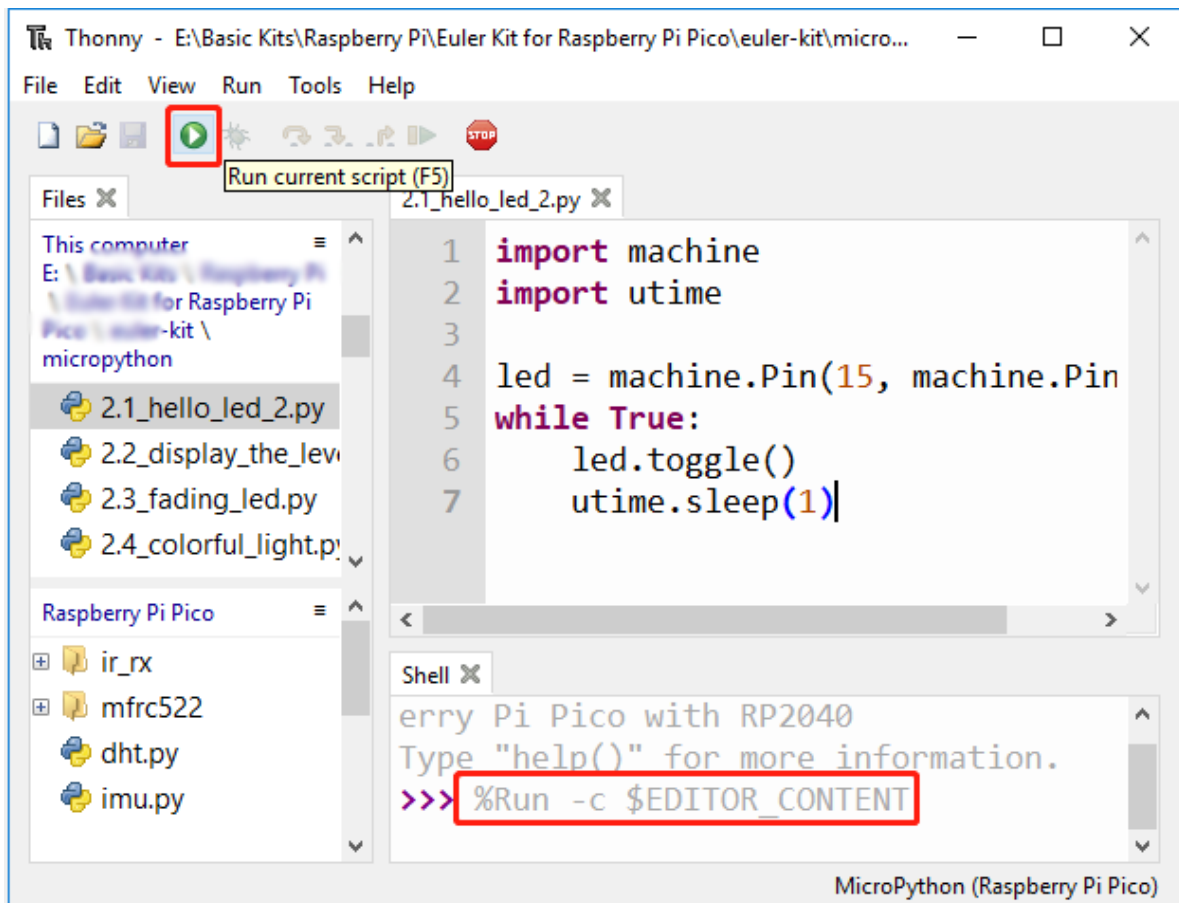
## 2. Richtigen Interpreter auswählen

Verbinden Sie den Pico W mit einem Mikro-USB-Kabel mit Ihrem Computer und wählen Sie den Interpreter „MicroPython (Raspberry Pi Pico)“ aus.



## 3. Code ausführen

Um das Skript auszuführen, klicken Sie auf die Schaltfläche **Aktuelles Skript ausführen** oder drücken Sie F5.



Falls der Code Informationen enthält, die ausgegeben werden müssen, erscheinen diese in der Shell; ansonsten erscheint nur die folgende Information.

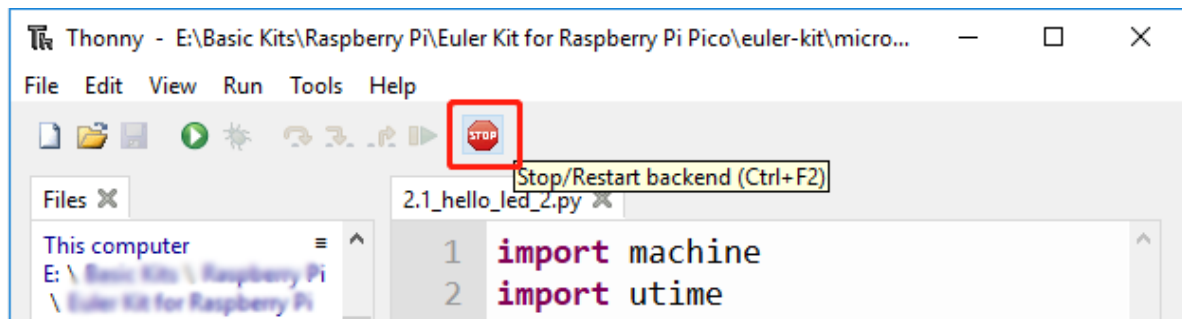
Klicken Sie auf **Ansicht -> Bearbeiten**, um das Shell-Fenster zu öffnen, falls es in Ihrem Thonny nicht angezeigt wird.

```
MicroPython vx.xx on xxxx-xx-xx; Raspberry Pi Pico W mit RP2040

Geben Sie "help()" für weitere Informationen ein.
>>> %Run -c $EDITOR_CONTENT
```

- Die erste Zeile zeigt die Version von MicroPython, das Datum und Informationen zu Ihrem Gerät.
- Die zweite Zeile fordert Sie auf, „help()“ einzugeben, um Hilfe zu erhalten.
- Die dritte Zeile ist ein Befehl von Thonny, der dem MicroPython-Interpreter auf Ihrem Pico W sagt, den Inhalt des Skriptbereichs - „EDITOR\_CONTENT“ - auszuführen.
- Falls nach der dritten Zeile eine Nachricht erscheint, handelt es sich normalerweise um eine Ausgabe, die Sie in MicroPython drucken lassen, oder um eine Fehlermeldung des Codes.

#### 4. Ausführung stoppen

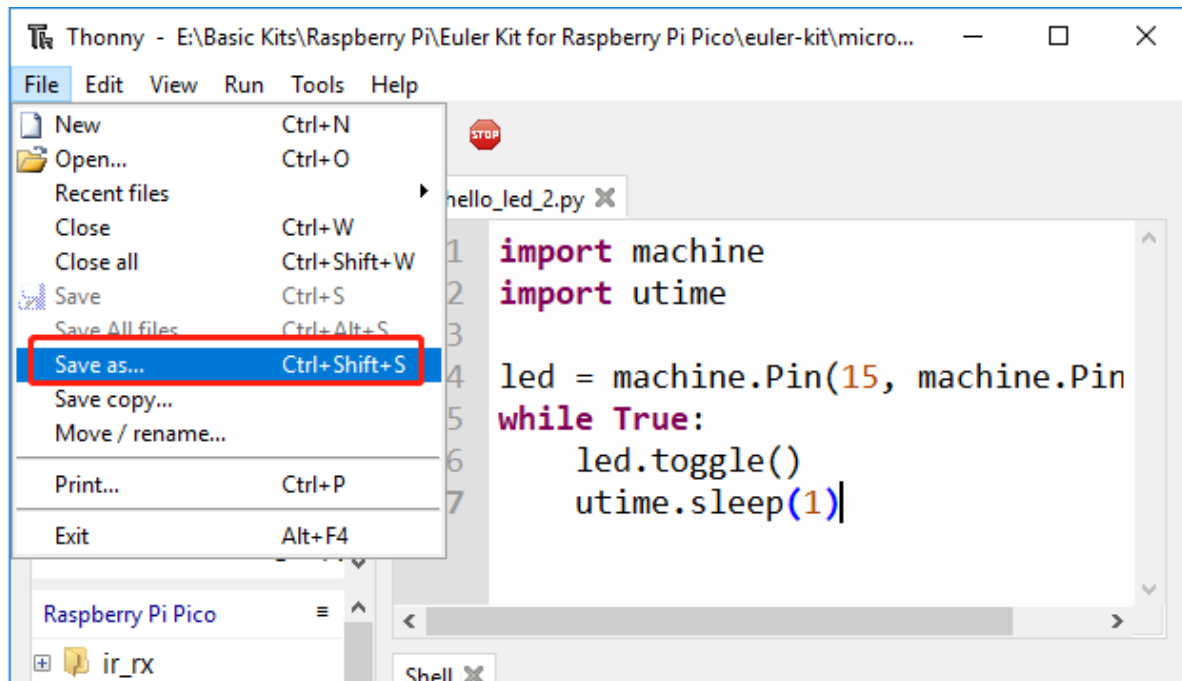


Um den laufenden Code zu stoppen, klicken Sie auf die Schaltfläche **Stop/Backend neu starten**. Der Befehl `%RUN -c $EDITOR_CONTENT` verschwindet nach dem Anhalten.

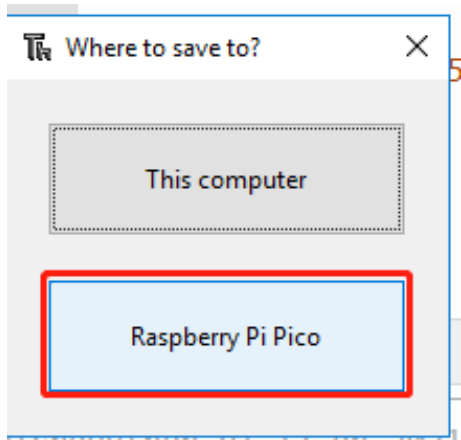
##### 5. Speichern oder Speichern unter

Änderungen am geöffneten Beispiel können durch Drücken von **Strg+S** oder durch Klicken auf die Schaltfläche **Speichern** in Thonny gespeichert werden.

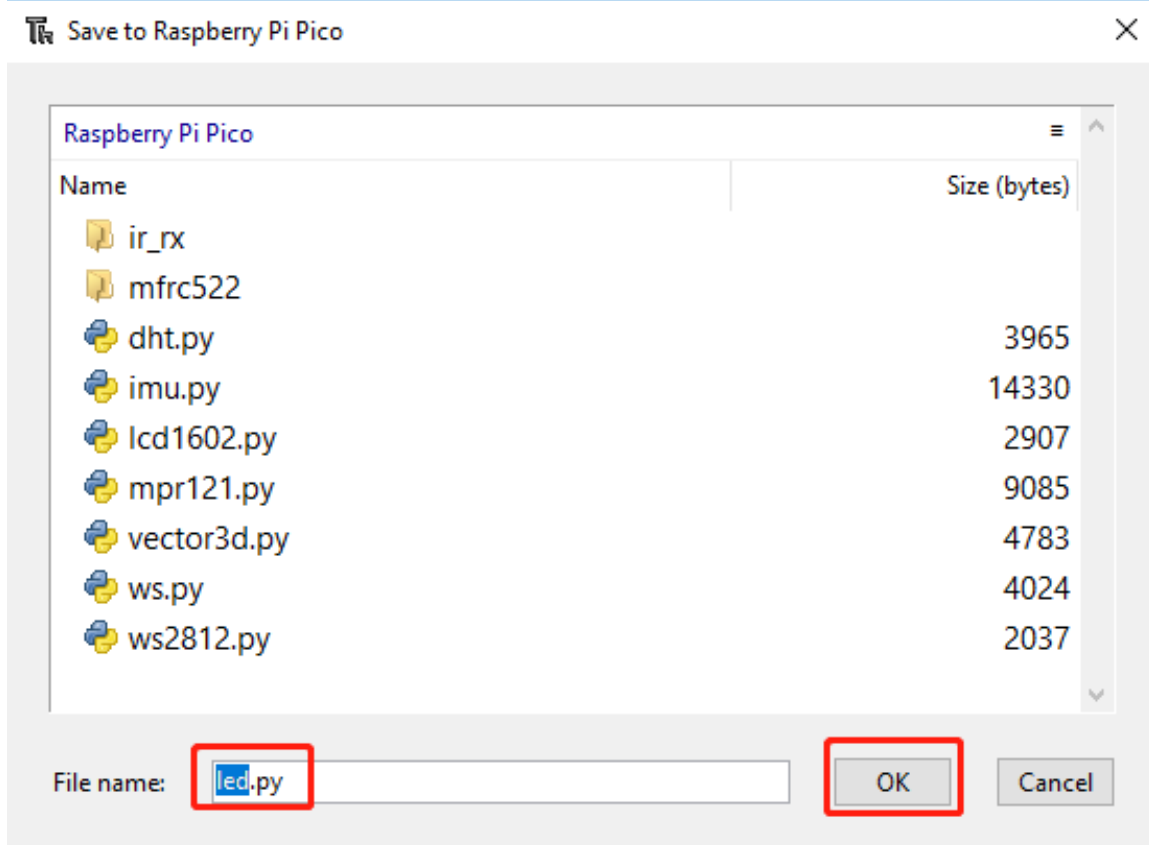
Der Code kann als separate Datei innerhalb des Laufwerks Raspberry Pi Pico W gespeichert werden, indem Sie auf **Datei -> Speichern unter** klicken.



Wählen Sie **Raspberry Pi Pico** aus.



Klicken Sie nach Eingabe des Dateinamens und der Erweiterung `.py` auf **OK**. Auf dem Laufwerk des Raspberry Pi Pico W sehen Sie dann Ihre gespeicherte Datei.



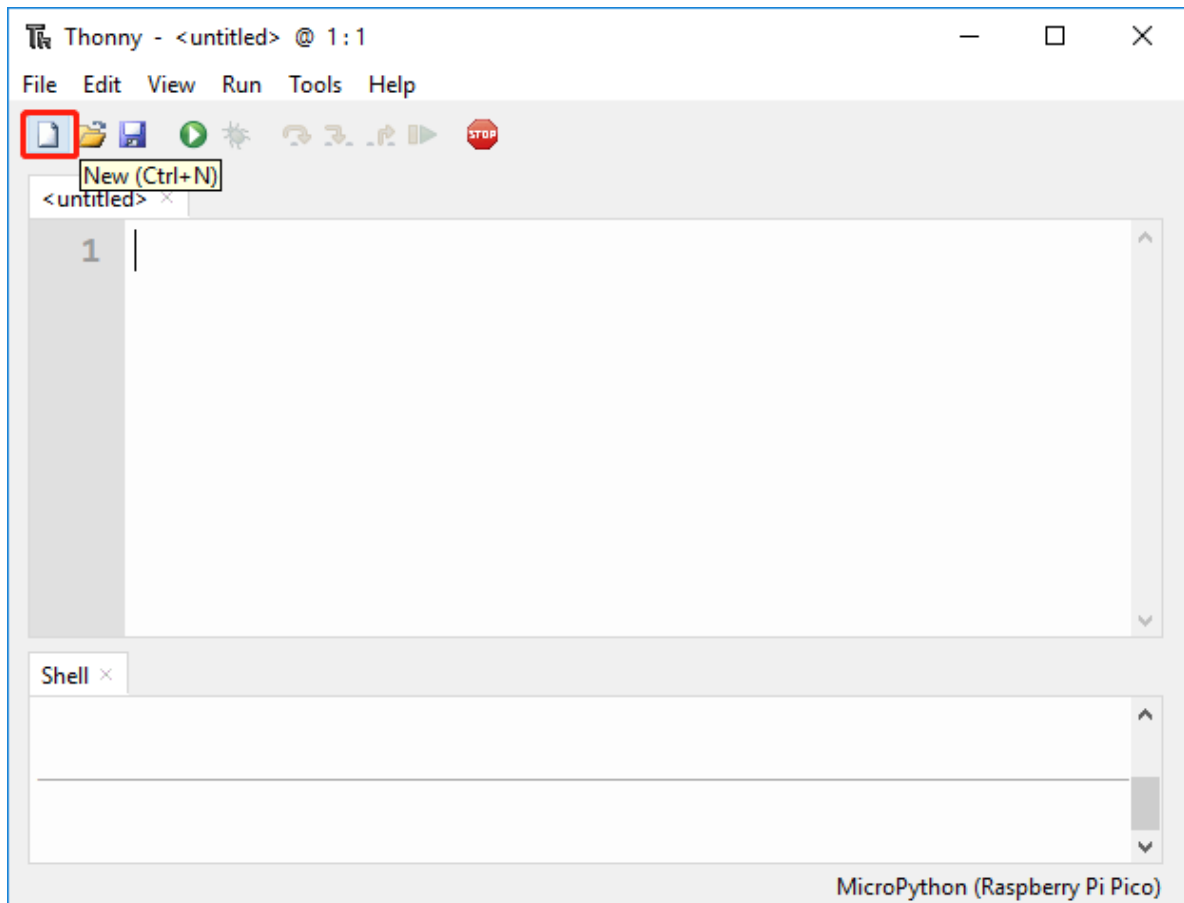
**Bemerkung:** Unabhängig davon, welchen Namen Sie Ihrem Code geben, ist es ratsam, die Art des Codes zu beschreiben, anstatt ihm einen bedeutungslosen Namen wie `abc.py` zu geben. Wenn Sie den Code als `main.py` speichern, wird er automatisch ausgeführt, sobald die Stromversorgung eingeschaltet ist.

## 4.5.2 Datei erstellen und ausführen

Der Code wird direkt im Codebereich angezeigt. Sie können ihn in Thonny kopieren und wie folgt ausführen.

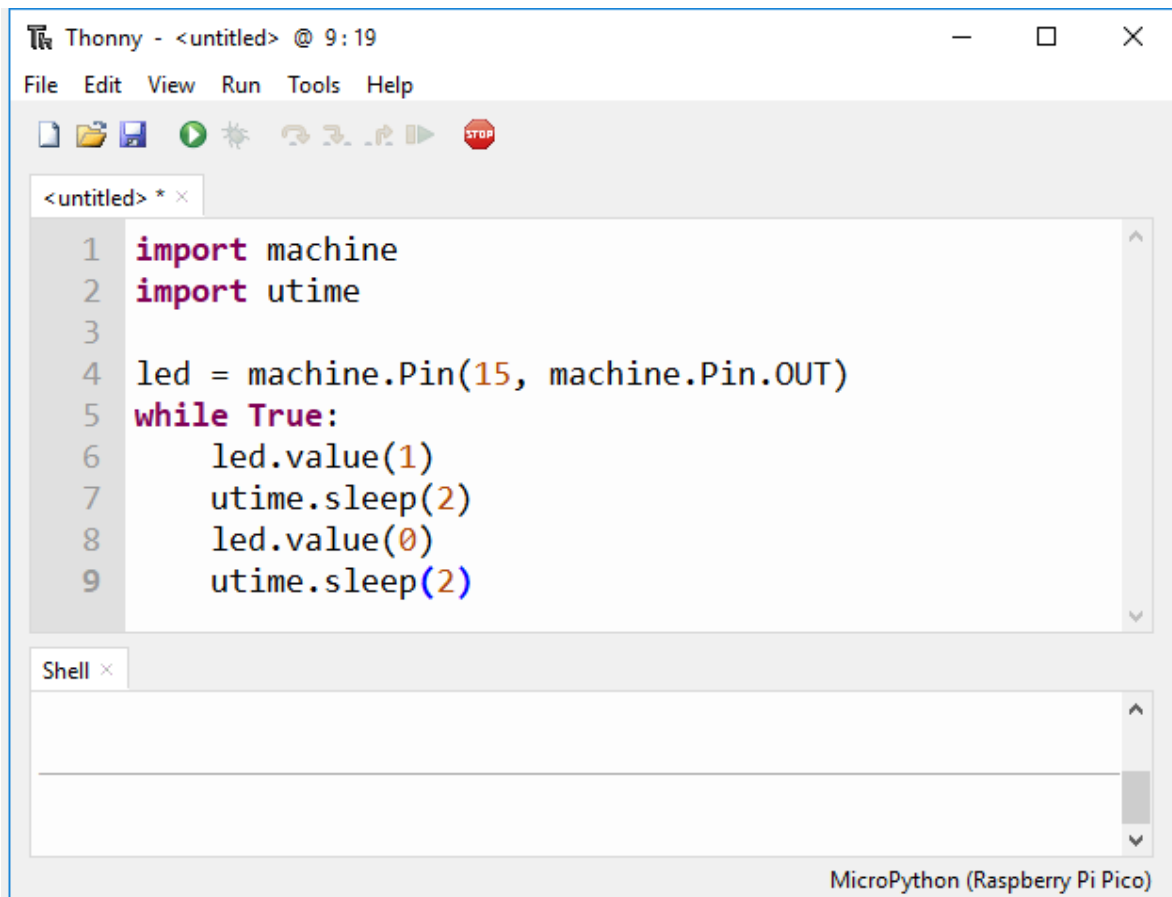
### 1. Neue Datei erstellen

Öffnen Sie die Thonny IDE und klicken Sie auf die Schaltfläche **Neu**, um eine neue leere Datei zu erstellen.



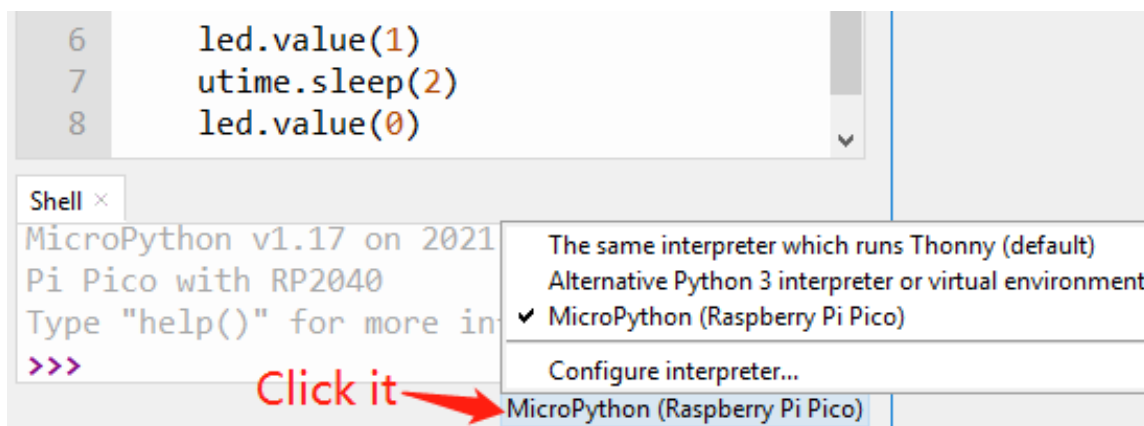
### 2. Code kopieren

Kopieren Sie den Code aus dem Projekt in die Thonny IDE.



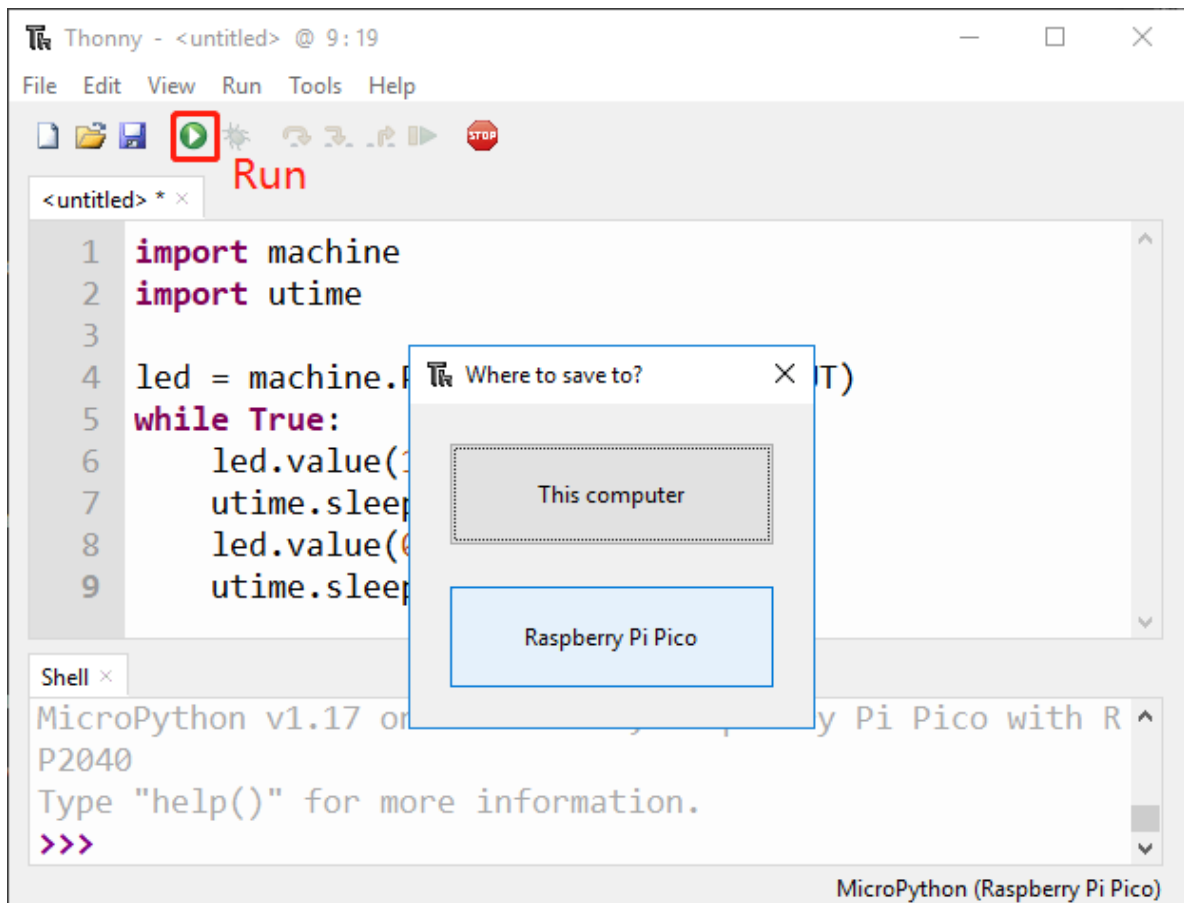
### 3. Richtigen Interpreter auswählen

Schließen Sie den Pico W mit einem Mikro-USB-Kabel an Ihren Computer an und wählen Sie im unteren rechten Eck den Interpreter „MicroPython (Raspberry Pi Pico)“ aus.



### 4. Code ausführen und speichern

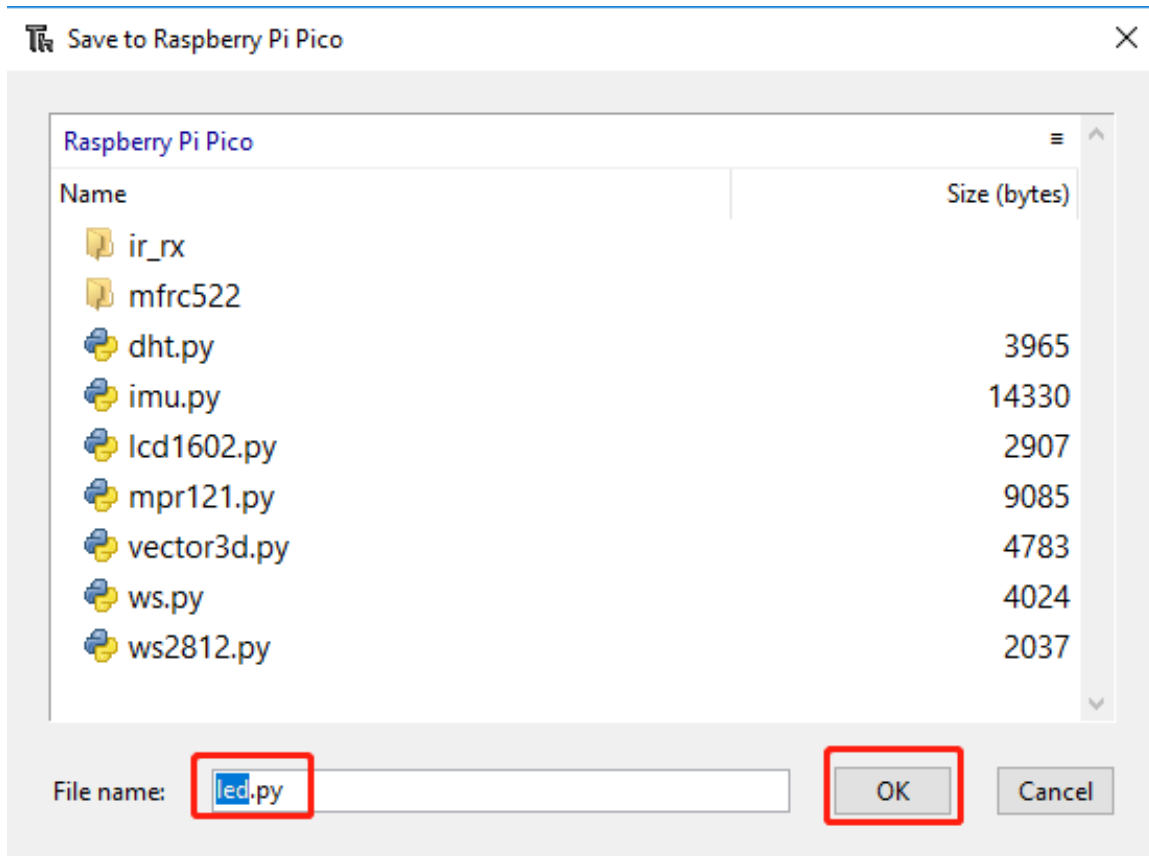
Klicken Sie auf **Aktuelles Skript ausführen** oder drücken Sie einfach F5, um den Code auszuführen. Falls Ihr Code noch nicht gespeichert ist, erscheint ein Fenster, in dem Sie wählen können, ob Sie ihn auf **diesem Computer** oder dem **Raspberry Pi Pico** speichern möchten.



**Bemerkung:** Thonny speichert Ihr Programm auf dem Raspberry Pi Pico, wenn Sie dies anweisen. Wenn Sie den Pico W abziehen und an einen anderen Computer anschließen, bleibt Ihr Programm erhalten.

Klicken Sie nach der Auswahl des Speicherorts und der Benennung der Datei sowie der Hinzufügung der Erweiterung **.py** auf OK.





**Bemerkung:** Unabhängig vom Namen, den Sie Ihrem Code geben, ist es am besten, seine Art zu beschreiben und ihm keinen sinnlosen Namen wie `abc.py` zu geben. Wenn Sie den Code als `main.py` speichern, wird er automatisch ausgeführt, sobald die Stromversorgung eingeschaltet ist.

Sobald Ihr Programm gespeichert ist, wird es automatisch ausgeführt und die folgenden Informationen werden im Shell-Bereich angezeigt.

Klicken Sie auf **Ansicht -> Bearbeiten**, um das Shell-Fenster zu öffnen, falls es in Ihrem Thonny nicht angezeigt wird.

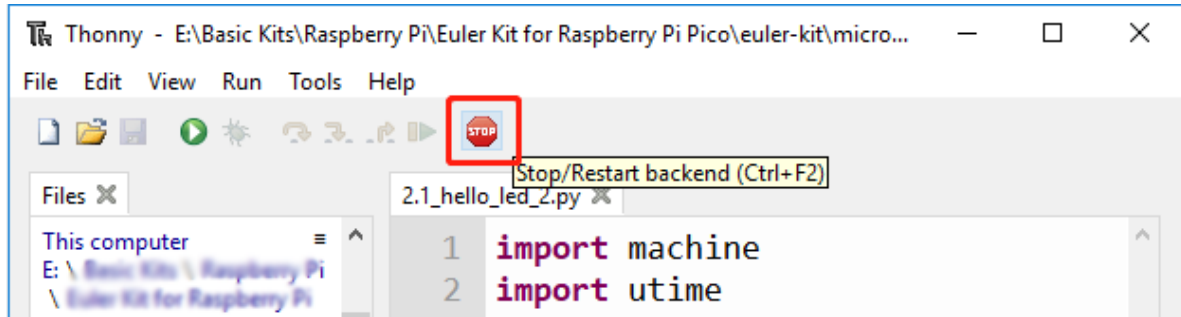
```
MicroPython vx.xx.x on xxxx-xx-xx; Raspberry Pi Pico W mit RP2040
```

```
Geben Sie "help()" für weitere Informationen ein.
```

```
>>> %Run -c $EDITOR_CONTENT
```

- Die erste Zeile zeigt die Version von MicroPython, das Datum und Informationen zu Ihrem Gerät.
- Die zweite Zeile fordert Sie auf, „help()“ einzugeben, um Hilfe zu erhalten.
- Die dritte Zeile ist ein Befehl von Thonny, der dem MicroPython-Interpreter auf Ihrem Pico W sagt, den Inhalt des Skriptbereichs - „EDITOR\_CONTENT“ - auszuführen.
- Falls nach der dritten Zeile eine Nachricht erscheint, handelt es sich normalerweise um eine Ausgabe, die Sie in MicroPython drucken lassen, oder um eine Fehlermeldung des Codes.

#### 5. Ausführung stoppen



Um den laufenden Code zu stoppen, klicken Sie auf die Schaltfläche **Stop/Backend neu starten**. Der Befehl `%Run -c $EDITOR_CONTENT` verschwindet nach dem Stoppen.

#### 6. Datei öffnen

Es gibt zwei Möglichkeiten, eine gespeicherte Code-Datei zu öffnen.

- Die erste Möglichkeit besteht darin, auf das Öffnen-Symbol in der Thonny-Symboleiste zu klicken. Genau wie beim Speichern eines Programms werden Sie gefragt, ob Sie es von **diesem Computer** oder dem **Raspberry Pi Pico** öffnen möchten. Wählen Sie beispielsweise **Raspberry Pi Pico**, erscheint eine Liste aller Programme, die Sie auf dem Pico W gespeichert haben.
- Die zweite Möglichkeit besteht darin, die Dateivorschau direkt zu öffnen, indem Sie auf **Ansicht-> Datei->** klicken und dann die entsprechende `.py`-Datei doppelklicken, um sie zu öffnen.

## 4.6 1.6 (Optional) Grundlegende Syntax von MicroPython

### 4.6.1 Einrückung

Die Einrückung bezieht sich auf die Leerzeichen am Anfang einer Codezeile. Wie bei standardmäßigen Python-Programmen werden auch MicroPython-Programme üblicherweise von oben nach unten ausgeführt: Es durchläuft jede Zeile der Reihe nach, führt sie im Interpreter aus und fährt dann mit der nächsten Zeile fort. Genauso, als würden Sie sie Zeile für Zeile in der Shell eingeben. Ein Programm, das einfach nur die Befehlsliste Zeile für Zeile durchgeht, ist allerdings nicht sehr intelligent. Daher hat auch MicroPython, genau wie Python, eine eigene Methode zur Steuerung der Reihenfolge seiner Programmausführung: die Einrückung.

Vor einem `print()`-Aufruf muss mindestens ein Leerzeichen gesetzt werden, sonst erscheint die Fehlermeldung „Ungültige Syntax“. Es wird allgemein empfohlen, Leerzeichen durch einheitliches Drücken der Tab-Taste zu standardisieren.

```
if 8 > 5:
print("Eight is greater than Five!")
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 2
SyntaxError: invalid syntax
```

In einem Codeblock müssen Sie die gleiche Anzahl an Leerzeichen verwenden, sonst wird Python einen Fehler ausgeben.

```
if 8 > 5:
print("Eight is greater than Five!")
    print("Eight is greater than Five")
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 2
SyntaxError: invalid syntax
```

## 4.6.2 Kommentare

Kommentare im Code helfen uns, den Code zu verstehen, machen den gesamten Code lesefreundlicher und ermöglichen es, während des Testens Teile des Codes auszukommentieren, sodass diese Teile nicht ausgeführt werden.

### Einzeilige Kommentare

Einzeilige Kommentare in MicroPython beginnen mit einem #, und der darauf folgende Text wird bis zum Ende der Zeile als Kommentar betrachtet. Kommentare können vor oder nach dem Code platziert werden.

```
print("Hallo Welt") # Das ist ein Kommentar
```

```
>>> %Run -c $EDITOR_CONTENT
Hallo Welt
```

Kommentare sind nicht zwangsläufig Text, der dazu dient, den Code zu erklären. Sie können auch Teile des Codes auskommentieren, um zu verhindern, dass MicroPython den Code ausführt.

```
#print("Wird nicht ausgeführt!")
print("Hallo Welt") # Das ist ein Kommentar
```

```
>>> %Run -c $EDITOR_CONTENT
Hallo Welt
```

### Mehrzeilige Kommentare

Wenn Sie mehrere Zeilen kommentieren möchten, können Sie mehrere # Zeichen verwenden.

```
# Das ist ein Kommentar
# der über mehrere
# Zeilen geht
print("Hallo Welt!")
```

```
>>> %Run -c $EDITOR_CONTENT
Hallo Welt!
```

Alternativ können Sie auch mehrzeilige Zeichenketten verwenden.

Da MicroPython Zeichenfolgen, die keiner Variablen zugewiesen werden, ignoriert, können Sie mehrzeilige Zeichenketten (Dreifach-Anführungszeichen) zum Code hinzufügen und dort Kommentare einfügen:

```
"""
Das ist ein Kommentar
der über mehrere
Zeilen geht
"""
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
"""  
print("Hello, World!")
```

```
>>> %Run -c $EDITOR_CONTENT  
Hello, World!
```

Solange die Zeichenfolge keiner Variablen zugewiesen ist, wird MicroPython sie nach dem Lesen des Codes ignorieren und so behandeln, als hätten Sie einen mehrzeiligen Kommentar verfasst.

### 4.6.3 Print()

Die Funktion `print()` gibt die angegebene Nachricht auf dem Bildschirm oder einem anderen Standardausgabegerät aus. Die Nachricht kann eine Zeichenkette oder ein beliebiges anderes Objekt sein. Das Objekt wird vor der Ausgabe auf dem Bildschirm in eine Zeichenkette umgewandelt.

Mehrere Objekte ausgeben:

```
print("Willkommen!", "Viel Spaß!")
```

```
>>> %Run -c $EDITOR_CONTENT  
Willkommen! Viel Spaß!
```

Tupel ausgeben:

```
x = ("Birne", "Apfel", "Traube")  
print(x)
```

```
>>> %Run -c $EDITOR_CONTENT  
( 'Birne', 'Apfel', 'Traube' )
```

Zwei Nachrichten ausgeben und das Trennzeichen festlegen:

```
print("Hallo", "wie geht's?", sep="---")
```

```
>>> %Run -c $EDITOR_CONTENT  
Hallo---wie geht's?
```

### 4.6.4 Variablen

Variablen dienen als Behälter zur Speicherung von Datenwerten.

Eine Variable zu erstellen ist sehr einfach. Sie muss lediglich benannt und ihr ein Wert zugewiesen werden. Der Datentyp der Variable muss bei der Zuweisung nicht angegeben werden, da die Variable eine Referenz ist und über die Zuweisung auf Objekte verschiedener Datentypen zugreift.

Die Benennung von Variablen muss folgende Regeln beachten:

- Variablennamen dürfen nur Zahlen, Buchstaben und Unterstriche enthalten
- Das erste Zeichen des Variablennamens muss ein Buchstabe oder Unterstrich sein
- Variablennamen sind groß- und kleinschreibungsempfindlich

## Variable erstellen

In MicroPython gibt es keinen Befehl zur Deklaration von Variablen. Variablen werden erstellt, indem ihnen zum ersten Mal ein Wert zugewiesen wird. Es ist keine spezielle Typdeklaration erforderlich, und der Typ kann sogar nach dem Festlegen der Variable geändert werden.

```
x = 8      # x ist vom Typ int
x = "Lily" # x ist nun vom Typ str
print(x)
```

```
>>> %Run -c $EDITOR_CONTENT
Lily
```

## Typumwandlung (Casting)

Wenn Sie den Datentyp für die Variable spezifizieren möchten, können Sie dies durch Typumwandlung (Casting) tun.

```
x = int(5)    # x wird 5 sein
y = str(5)    # y wird '5' sein
z = float(5)  # z wird 5.0 sein
print(x, y, z)
```

```
>>> %Run -c $EDITOR_CONTENT
5 5 5.0
```

## Den Typ abfragen

Sie können den Datentyp einer Variable mit der Funktion `type()` abfragen.

```
x = 5
y = "Hallo"
z = 5.0
print(type(x), type(y), type(z))
```

```
>>> %Run -c $EDITOR_CONTENT
<class 'int'> <class 'str'> <class 'float'>
```

## Einfache oder doppelte Anführungszeichen?

In MicroPython können einfache oder doppelte Anführungszeichen verwendet werden, um String-Variablen zu definieren.

```
x = "Hallo"
# ist dasselbe wie
x = 'Hallo'
```

## Groß- und Kleinschreibung

Variablennamen sind groß- und kleinschreibungsempfindlich.

```
a = 5
A = "Lily"
# A wird a nicht überschreiben
print(a, A)
```

```
>>> %Run -c $EDITOR_CONTENT
5 Lily
```

### 4.6.5 If-Else-Anweisungen

Entscheidungsfindung ist erforderlich, wenn ein bestimmter Code nur bei Erfüllung einer bestimmten Bedingung ausgeführt werden soll.

#### if

```
if Testausdruck:
    Anweisung(en)
```

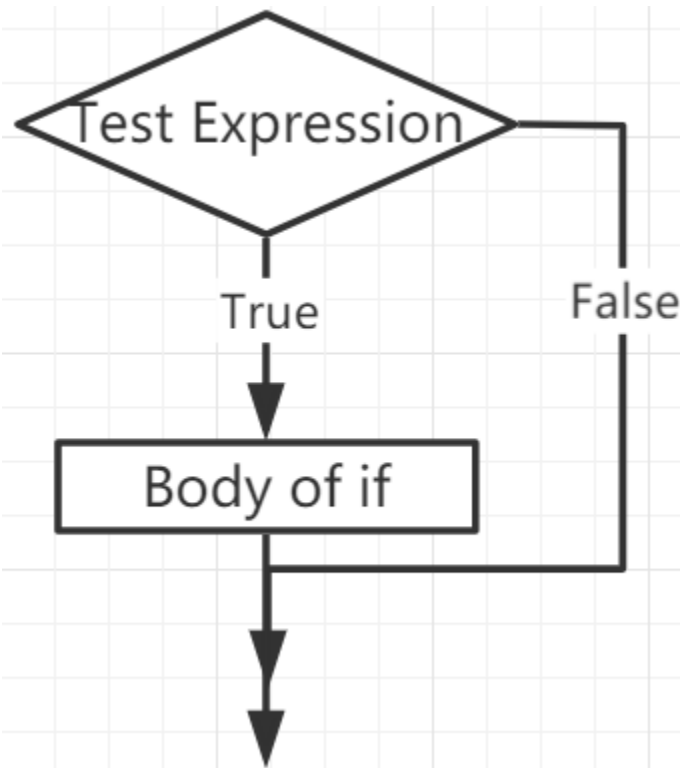
In diesem Fall wertet das Programm den Testausdruck aus und führt die Anweisung nur aus, wenn der Testausdruck wahr ist.

Ist der Testausdruck falsch, werden die Anweisung(en) nicht ausgeführt.

In MicroPython signalisiert die Einrückung den Körper der if-Anweisung. Der Körper beginnt mit einer Einrückung und endet mit der ersten nicht eingerückten Zeile.

Python interpretiert Werte ungleich Null als „True“. None und 0 werden als „False“ interpretiert.

#### Flussdiagramm für if-Anweisungen



### Beispiel

```
num = 8
if num > 0:
    print(num, "ist eine positive Zahl.")
print("Ende mit dieser Zeile")
```

```
>>> %Run -c $EDITOR_CONTENT
8 ist eine positive Zahl.
Ende mit dieser Zeile
```

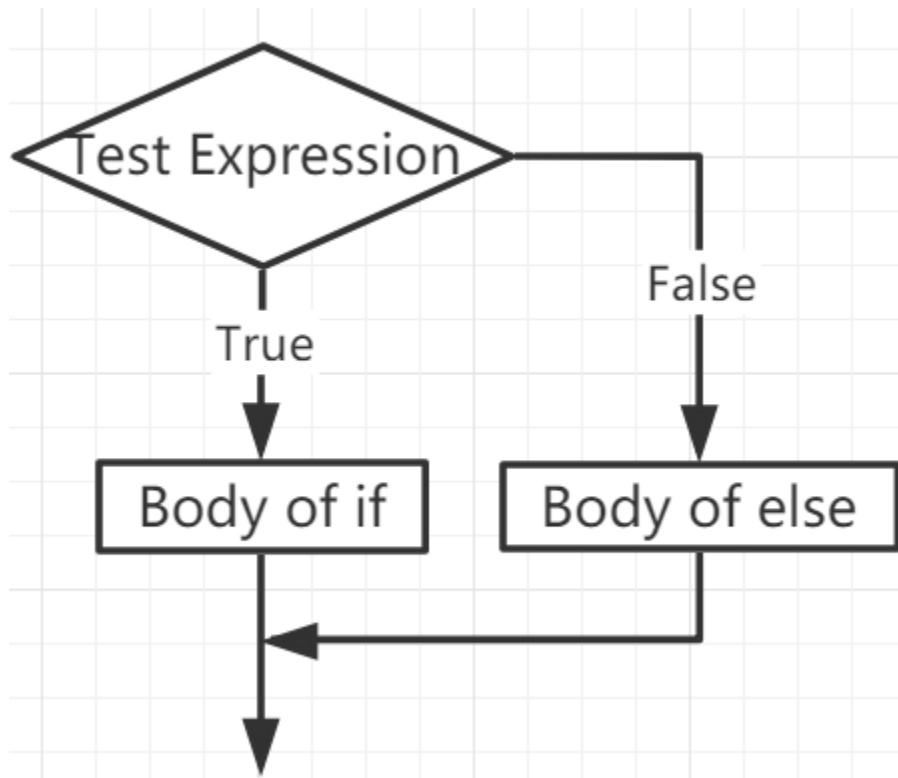
### if...else

```
if Testausdruck:
    Körper von if
else:
    Körper von else
```

Die if...else-Anweisung wertet den Testausdruck aus und führt den Körper von if nur aus, wenn die Testbedingung True ist.

Ist die Bedingung False, wird der Körper von else ausgeführt. Einrückungen dienen zur Abgrenzung der Blöcke.

### Flussdiagramm für if...else-Anweisungen



### Beispiel

```
num = -8
if num > 0:
    print(num, "ist eine positive Zahl.")
else:
    print(num, "ist eine negative Zahl.")
```

```
>>> %Run -c $EDITOR_CONTENT
-8 ist eine negative Zahl.
```

### if...elif...else

```
if Testausdruck:
    Körper von if
elif Testausdruck:
    Körper von elif
else:
    Körper von else
```

Elif steht für *else if*. Damit können wir mehrere Ausdrücke prüfen.

Ist die Bedingung des `if` falsch, wird die Bedingung des nächsten `elif`-Blocks geprüft und so weiter.

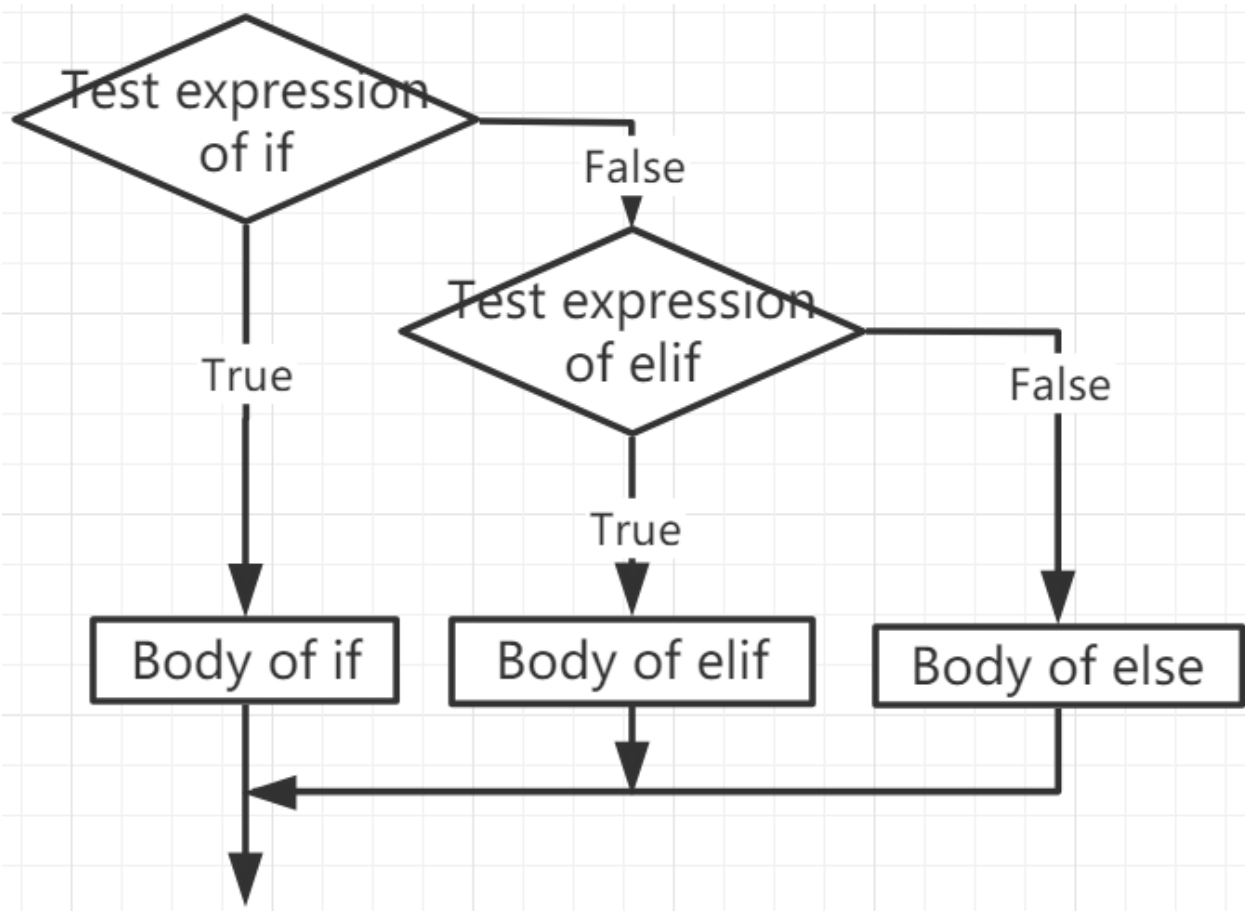
Sind alle Bedingungen *falsch*, wird der Körper von `else` ausgeführt.

Nur einer der `if...elif...else`-Blöcke wird je nach Bedingung ausgeführt.

Der `if`-Block kann nur einen `else`-Block haben, jedoch mehrere `elif`-Blöcke.



## Flussdiagramm für if...elif...else-Anweisungen



## Beispiel

```
x = 10
y = 9

if x > y:
    print("x ist größer als y")
elif x == y:
    print("x und y sind gleich")
else:
    print("y ist größer als x")
```

```
>>> %Run -c $EDITOR_CONTENT
x ist größer als y
```

## Verschachtelte if-Anweisungen

Wir können eine if-Anweisung in eine andere if-Anweisung einbetten; das nennen wir dann eine verschachtelte if-Anweisung.

### Beispiel

```
x = 67

if x > 10:
    print("Über zehn,")
    if x > 20:
        print("und auch über 20!")
    else:
        print("aber nicht über 20.")
```

```
>>> %Run -c $EDITOR_CONTENT
Über zehn,
und auch über 20!
```

## 4.6.6 While-Schleifen

Das `while`-Statement wird verwendet, um ein Programm in einer Schleife auszuführen. Dies geschieht unter bestimmten Bedingungen, um wiederholt die gleiche Aufgabe abzuarbeiten.

Die Grundform lautet:

```
while Testausdruck:
    Schleifenkörper
```

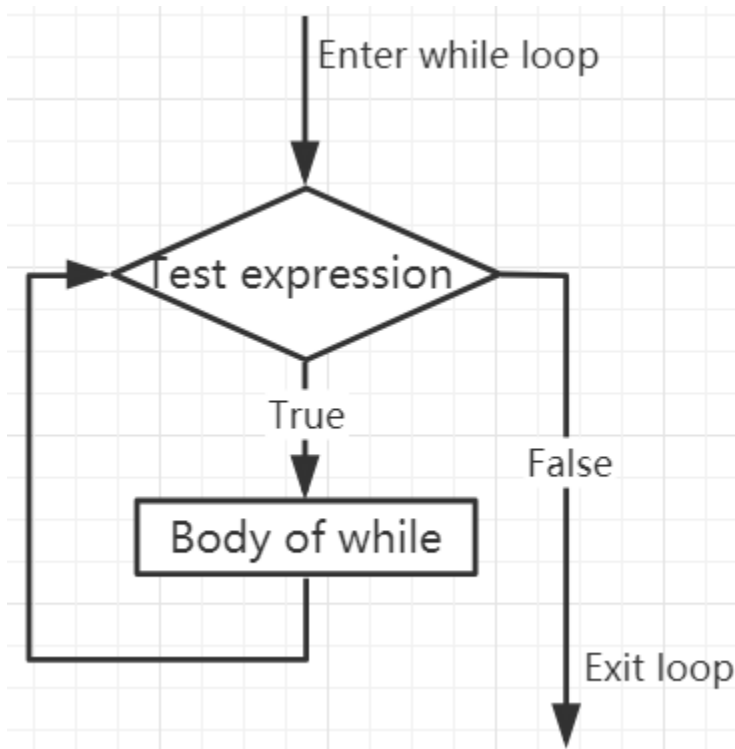
In der `while`-Schleife wird zunächst der Testausdruck überprüft. Nur wenn der Testausdruck den Wert `True` ergibt, wird der Schleifenkörper ausgeführt. Nach einer Iteration wird der Testausdruck erneut überprüft. Dieser Prozess wiederholt sich, bis der Testausdruck den Wert `False` ergibt.

In MicroPython wird der Körper der `while`-Schleife durch Einrückung bestimmt.

Der Körper beginnt mit einer Einrückung und endet mit der ersten nicht eingerückten Zeile.

Python interpretiert jeden von Null verschiedenen Wert als `True`. `None` und `0` werden als `False` interpretiert.

### Flussdiagramm der while-Schleife



```
x = 10  
  
while x > 0:  
    print(x)  
    x -= 1
```

```
>>> %Run -c $EDITOR_CONTENT  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1
```

### Break-Anweisung

Mit der Break-Anweisung können wir die Schleife abbrechen, selbst wenn die While-Bedingung wahr ist:

```
x = 10  
  
while x > 0:  
    print(x)  
    if x == 6:
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
break
x -= 1
```

```
>>> %Run -c $EDITOR_CONTENT
10
9
8
7
6
```

### While-Schleife mit Else

Ähnlich wie die `if`-Schleife kann auch die `while`-Schleife einen optionalen `else`-Block haben.

Wenn die Bedingung in der `while`-Schleife als `False` bewertet wird, wird der `else`-Teil ausgeführt.

```
x = 10

while x > 0:
    print(x)
    x -= 1
else:
    print("Spiel beendet")
```

```
>>> %Run -c $EDITOR_CONTENT
10
9
8
7
6
5
4
3
2
1
Spiel beendet
```

### 4.6.7 For-Schleifen

Die `for`-Schleife kann über jede Sequenz von Elementen iterieren, beispielsweise über eine Liste oder einen String.

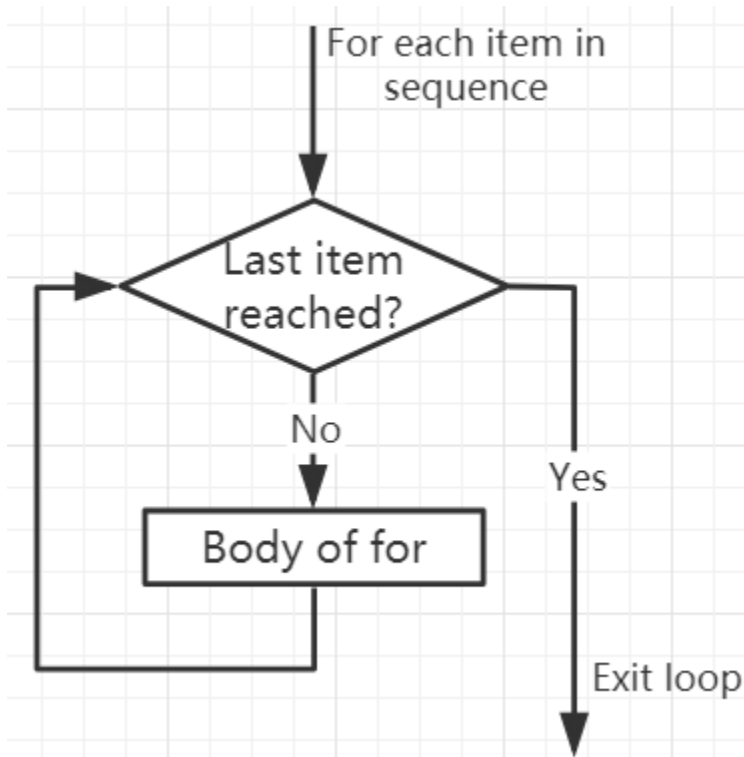
Die Syntax einer `for`-Schleife ist wie folgt:

```
for val in sequenz:
    Schleifenkörper
```

Hierbei ist `val` eine Variable, die in jeder Iteration den Wert des Elements aus der Sequenz annimmt.

Die Schleife wird solange ausgeführt, bis das letzte Element der Sequenz erreicht ist. Durch Einrückung wird der Schleifenkörper vom restlichen Code abgegrenzt.

#### Flussdiagramm der `for`-Schleife



```

zahlen = [1, 2, 3, 4]
summe = 0

for val in zahlen:
    summe = summe+val

print("Die Summe beträgt", summe)

```

```

>>> %Run -c $EDITOR_CONTENT
Die Summe beträgt 10

```

### Die break-Anweisung

Mit der break-Anweisung kann die Schleife beendet werden, bevor alle Elemente durchlaufen wurden:

```

zahlen = [1, 2, 3, 4]
summe = 0

for val in zahlen:
    summe = summe+val
    if summe == 6:
        break
print("Die Summe beträgt", summe)

```

```

>>> %Run -c $EDITOR_CONTENT
Die Summe beträgt 6

```

## Die continue-Anweisung

Mit der `continue`-Anweisung kann die aktuelle Iteration der Schleife beendet und mit der nächsten fortgefahren werden:

```
zahlen = [1, 2, 3, 4]

for val in zahlen:
    if val == 3:
        continue
    print(val)
```

```
>>> %Run -c $EDITOR_CONTENT
1
2
4
```

## Die range()-Funktion

Mit der `range()`-Funktion kann eine Zahlenreihe erzeugt werden. `range(6)` erzeugt Zahlen von 0 bis 5 (6 Zahlen insgesamt).

Man kann auch Start, Ende und Schrittgröße definieren: `range(start, stop, step_size)`. Wenn nicht angegeben, wird die Schrittgröße standardmäßig auf 1 gesetzt.

In gewissem Sinne ist das von `range` zurückgegebene Objekt „faul“, da es bei seiner Erstellung nicht jede darin enthaltene Zahl generiert. Es ist jedoch kein Iterator, da es die Operationen `in`, `len` und `__getitem__` unterstützt.

Diese Funktion speichert nicht alle Werte im Speicher; das wäre ineffizient. Sie speichert nur Start, Ende und Schrittgröße und generiert die nächste Zahl bei Bedarf.

Um diese Funktion zu zwingen, alle Elemente auszugeben, kann man die `list()`-Funktion verwenden.

```
print(range(6))

print(list(range(6)))

print(list(range(2, 6)))

print(list(range(2, 10, 2)))
```

```
>>> %Run -c $EDITOR_CONTENT
range(0, 6)
[0, 1, 2, 3, 4, 5]
[2, 3, 4, 5]
[2, 4, 6, 8]
```

Man kann `range()` in einer `for`-Schleife verwenden, um über eine Zahlenreihe zu iterieren. Das kann in Kombination mit der `len()`-Funktion genutzt werden, um mit dem Index über eine Sequenz zu iterieren.

```
früchte = ['Birne', 'Apfel', 'Traube']

for i in range(len(früchte)):
    print("Ich mag", früchte[i])
```

```
>>> %Run -c $EDITOR_CONTENT
Ich mag Birne
Ich mag Apfel
Ich mag Traube
```

## Else in For-Schleife

Die for-Schleife kann auch einen optionalen else-Block haben. Wenn die Elemente der für die Schleife verwendeten Sequenz aufgebraucht sind, wird der else-Teil ausgeführt.

Das break-Schlüsselwort kann verwendet werden, um die for-Schleife zu beenden. In diesem Fall wird der else-Teil ignoriert.

Daher wird der else-Teil der for-Schleife ausgeführt, wenn kein Abbruch erfolgt.

```
for val in range(5):
    print(val)
else:
    print("Fertig")
```

```
>>> %Run -c $EDITOR_CONTENT
0
1
2
3
4
Fertig
```

Der Else-Block wird NICHT ausgeführt, wenn die Schleife durch eine break-Anweisung gestoppt wird.

```
for val in range(5):
    if val == 2: break
    print(val)
else:
    print("Fertig")
```

```
>>> %Run -c $EDITOR_CONTENT
0
1
```

## 4.6.8 Funktionen

In MicroPython ist eine Funktion eine Gruppe von verwandten Anweisungen, die eine bestimmte Aufgabe ausführen.

Funktionen helfen dabei, unser Programm in kleinere, modulare Blöcke zu zerlegen. Je umfangreicher unser Projekt wird, desto übersichtlicher und handhabbarer wird es durch die Verwendung von Funktionen.

Zudem vermeiden Funktionen Duplikate und machen den Code wiederverwendbar.

## Eine Funktion erstellen

```
def funktions_name(parameter):  
    """Dokumentationszeichenfolge"""  
    Anweisung(en)
```

- Eine Funktion wird mit dem Schlüsselwort `def` definiert.
- Ein Funktionsname zur eindeutigen Identifizierung der Funktion. Die Benennung von Funktionen und Variablen folgt denselben Regeln:
  - Darf nur Zahlen, Buchstaben und Unterstriche enthalten.
  - Das erste Zeichen muss ein Buchstabe oder Unterstrich sein.
  - Groß- und Kleinschreibung wird unterschieden.
- Parameter (Argumente), über die Werte an eine Funktion übergeben werden. Diese sind optional.
- Der Doppelpunkt (`:`) markiert das Ende der Funktionskopfzeile.
- Eine optionale Dokumentationszeichenfolge, die in der Regel durch dreifache Anführungszeichen mehrzeilig gestaltet werden kann, dient zur Beschreibung der Funktion.
- Eine oder mehrere gültige MicroPython-Anweisungen, die den Funktionskörper bilden. Die Anweisungen müssen die gleiche Einrückungsebene haben (in der Regel 4 Leerzeichen).
- Jede Funktion benötigt mindestens eine Anweisung. Sollte aus irgendeinem Grund eine Funktion keine Anweisung enthalten, verwenden Sie bitte die Anweisung `pass`, um Fehler zu vermeiden.
- Eine optionale `return`-Anweisung, um einen Wert aus der Funktion zurückzugeben.

## Eine Funktion aufrufen

Um eine Funktion aufzurufen, fügen Sie Klammern hinter den Funktionsnamen.

```
def meine_funktion():  
    print("Deine erste Funktion")  
  
meine_funktion()
```

```
>>> %Run -c $EDITOR_CONTENT  
Deine erste Funktion
```

## Die return-Anweisung

Die `return`-Anweisung wird verwendet, um eine Funktion zu verlassen und an die Stelle zurückzukehren, von der aus sie aufgerufen wurde.

### Syntax von `return`

```
return [Ausdrucksliste]
```

Die Anweisung kann einen Ausdruck enthalten, der ausgewertet wird und einen Wert zurückgibt. Wenn in der Anweisung kein Ausdruck enthalten ist oder die `return`-Anweisung in der Funktion selbst nicht vorhanden ist, gibt die Funktion ein `None`-Objekt zurück.



```
def meine_funktion():
    print("Deine erste Funktion")

print(meine_funktion())
```

```
>>> %Run -c $EDITOR_CONTENT
Deine erste Funktion
None
```

In diesem Fall ist None der Rückgabewert, da die return-Anweisung nicht verwendet wird.

## Argumente

Informationen können der Funktion als Argumente übergeben werden.

Geben Sie die Argumente in Klammern hinter dem Funktionsnamen an. Sie können so viele Argumente hinzufügen wie nötig, trennen Sie diese einfach durch Kommas.

```
def welcome(name, msg):
    """This is a welcome function for
    the person with the provided message"""
    print("Hello", name + ', ' + msg)

welcome("Lily", "Welcome to China!")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello Lily, Welcome to China!
```

## Anzahl der Argumente

Standardmäßig muss eine Funktion mit der korrekten Anzahl an Argumenten aufgerufen werden. Das heißt, wenn Ihre Funktion zwei Parameter erwartet, müssen Sie die Funktion auch mit genau zwei Argumenten aufrufen, weder mehr noch weniger.

```
def welcome(name, msg):
    """This is a welcome function for
    the person with the provided message"""
    print("Hello", name + ', ' + msg)

welcome("Lily", "Welcome to China!")
```

Hier hat die Funktion welcome() zwei Parameter.

Da wir diese Funktion mit zwei Argumenten aufgerufen haben, wird sie fehlerfrei ausgeführt.

Wird sie jedoch mit einer abweichenden Anzahl an Argumenten aufgerufen, gibt der Interpreter eine Fehlermeldung aus.

Folgende Aufrufe der Funktion, die entweder ein oder gar kein Argument enthalten, erzeugen jeweils eine entsprechende Fehlermeldung.

```
welcome("Lily")Only one argument
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 6, in <module>
TypeError: function takes 2 positional arguments but 1 were given
```

```
welcome()No arguments
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 6, in <module>
TypeError: function takes 2 positional arguments but 0 were given
```

### Standardargumente

In MicroPython können wir den Zuweisungsoperator (=) verwenden, um einen Standardwert für den Parameter festzulegen.

Wenn wir die Funktion ohne Argument aufrufen, wird der Standardwert verwendet.

```
def welcome(name, msg = "Welcome to China!"):
    """This is a welcome function for
    the person with the provided message"""
    print("Hello", name + ', ' + msg)
welcome("Lily")
```

```
>>> %Run -c $EDITOR_CONTENT
Hello Lily, Welcome to China!
```

In dieser Funktion ist der Parameter `name` zwingend erforderlich, da er keinen Standardwert hat.

Andererseits ist der Standardwert des Parameters `msg` „Willkommen in China!“. Daher ist er beim Aufruf der Funktion optional. Wird ein Wert angegeben, überschreibt dieser den Standardwert.

In der Funktion können beliebig viele Argumente einen Standardwert haben. Sobald jedoch ein Argument einen Standardwert hat, müssen alle folgenden Argumente ebenfalls Standardwerte haben.

Das bedeutet, dass Standardargumente immer am Ende der Parameterliste stehen müssen.

Zum Beispiel, wenn wir die obenstehende Funktionsdeklaration wie folgt definieren:

```
def welcome(name = "Lily", msg):
```

Dann erhalten wir die folgende Fehlermeldung:

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
SyntaxError: non-default argument follows default argument
```

## Schlüsselwortargumente

Wenn wir eine Funktion mit bestimmten Werten aufrufen, werden diese Werte basierend auf ihrer Position den Argumenten zugewiesen.

Beispielsweise wird im oben erwähnten Fall der Funktion `welcome()`, wenn wir sie mit `welcome(„Lily“, „Willkommen in China“)` aufrufen, der Wert „Lily“ dem Parameter `name` und entsprechend „Willkommen in China“ dem Parameter `msg` zugewiesen.

MicroPython ermöglicht das Aufrufen von Funktionen mit Schlüsselwortargumenten. Bei dieser Art des Aufrufs kann die Reihenfolge der Argumente variiert werden.

```
# Schlüsselwortargumente
welcome(name = "Lily", msg = "Willkommen in China!")

# Schlüsselwortargumente (in unterschiedlicher Reihenfolge)
welcome(msg = "Willkommen in China!", name = "Lily")

# Ein Positionsargument, ein Schlüsselwortargument
welcome("Lily", msg = "Willkommen in China!")
```

Wie zu sehen ist, können Positionsargumente und Schlüsselwortargumente in Funktionsaufrufen gemischt werden. Es ist jedoch wichtig, dass die Schlüsselwortargumente immer nach den Positionsargumenten stehen.

Ein Positionsargument nach einem Schlüsselwortargument führt zu einem Fehler.

Zum Beispiel resultiert der folgende Funktionsaufruf in einem Fehler:

```
welcome(name="Lily", "Willkommen in China!")
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<stdin>", line 5, in <module>
SyntaxError: positional argument follows keyword argument
```

## Beliebige Argumente

Manchmal wissen wir im Voraus nicht, wie viele Argumente an die Funktion übergeben werden.

In der Funktionsdefinition können wir ein Sternchen (\*) vor dem Parameternamen setzen.

```
def welcome(*names):
    """Diese Funktion begrüßt alle Personen
    im Namens-Tupel"""
    for name in names:
        print("Willkommen in China!", name)

welcome("Lily", "John", "Wendy")
```

```
>>> %Run -c $EDITOR_CONTENT
Willkommen in China! Lily
Willkommen in China! John
Willkommen in China! Wendy
```

Hier haben wir die Funktion mit mehreren Argumenten aufgerufen, die in ein Tupel verpackt und dann an die Funktion übergeben werden.

Innerhalb der Funktion verwenden wir eine Schleife, um alle Argumente abzurufen.

### Rekursion

In Python ist es bekanntlich möglich, dass eine Funktion andere Funktionen aufruft. Sie kann sogar sich selbst aufrufen. Solche Konstrukte werden als rekursive Funktionen bezeichnet.

Dies hat den Vorteil, dass man durch Daten iterieren kann, um ein Ergebnis zu erreichen.

Entwickler sollten bei der Verwendung von Rekursion sehr vorsichtig sein, da leicht eine Funktion entstehen kann, die niemals endet oder übermäßig viel Speicher bzw. Prozessorleistung verbraucht. Bei korrekter Implementierung kann Rekursion jedoch ein sehr effizienter und mathematisch eleganter Ansatz zur Programmierung sein.

```
def rec_func(i):  
    if(i > 0):  
        result = i + rec_func(i - 1)  
        print(result)  
    else:  
        result = 0  
    return result  
  
rec_func(6)
```

```
>>> %Run -c $EDITOR_CONTENT  
1  
3  
6  
10  
15  
21
```

In diesem Beispiel ruft `rec_func()` sich selbst auf („Rekursion“). Wir verwenden die Variable `i` als Datenwert, der bei jedem Rekursionsschritt um 1 verringert wird. Wenn die Bedingung nicht größer als 0 ist (also 0), endet die Rekursion.

Für neue Entwickler kann es etwas Zeit in Anspruch nehmen, die Funktionsweise zu verstehen; der beste Weg zur Überprüfung ist das Ausprobieren und Anpassen.

#### Vorteile der Rekursion

- Rekursive Funktionen machen den Code sauber und elegant.
- Komplexe Aufgaben können durch Rekursion in einfachere Teilprobleme zerlegt werden.
- Die Erzeugung von Sequenzen ist mit Rekursion einfacher als mit verschachtelten Schleifen.

#### Nachteile der Rekursion

- Manchmal ist die Logik hinter der Rekursion schwer nachzuvollziehen.
- Rekursive Aufrufe sind ressourcenintensiv, da sie viel Speicher und Zeit verbrauchen.
- Rekursive Funktionen sind schwer zu debuggen.

## 4.6.9 Datentypen

### Eingebaute Datentypen

MicroPython unterstützt die folgenden Datentypen:

- Texttyp: str
- Numerische Typen: int, float, complex
- Sequenztypen: list, tuple, range
- Abbildungstyp: dict
- Mengentypen: set, frozenset
- Boolescher Typ: bool
- Binärtypen: bytes, bytearray, memoryview

### Den Datentyp ermitteln

Mit der Funktion `type()` können Sie den Datentyp eines beliebigen Objekts herausfinden:

```
a = 6.8
print(type(a))
```

```
>>> %Run -c $EDITOR_CONTENT
<class 'float'>
```

### Datentyp setzen

In MicroPython muss der Datentyp nicht explizit festgelegt werden; er wird automatisch bei der Wertzuweisung an eine Variable bestimmt:

```
x = "willkommen"
y = 45
z = ["Apfel", "Banane", "Kirsche"]

print(type(x))
print(type(y))
print(type(z))
```

```
>>> %Run -c $EDITOR_CONTENT
<class 'str'>
<class 'int'>
<class 'list'>
>>>
```

## Spezifischen Datentyp festlegen

Möchten Sie den Datentyp explizit angeben, können Sie die folgenden Konstruktorfunktionen verwenden:

Beispiel	Datentyp
<code>x = int(20)</code>	int
<code>x = float(20.5)</code>	float
<code>x = complex(1j)</code>	complex
<code>x = str(„Hallo Welt“)</code>	str
<code>x = list((„Apfel“, „Banane“, „Kirsche“))</code>	list
<code>x = tuple((„Apfel“, „Banane“, „Kirsche“))</code>	tuple
<code>x = range(6)</code>	range
<code>x = dict(name=„John“, age=36)</code>	dict
<code>x = set((„Apfel“, „Banane“, „Kirsche“))</code>	set
<code>x = frozenset((„Apfel“, „Banane“, „Kirsche“))</code>	frozenset
<code>x = bool(5)</code>	bool
<code>x = bytes(5)</code>	bytes
<code>x = bytearray(5)</code>	bytearray
<code>x = memoryview(bytes(5))</code>	memoryview

Einige von ihnen können Sie ausgeben, um das Ergebnis zu sehen.

```
a = float(20.5)
b = list(("Apfel", "Banane", "Kirsche"))
c = bool(5)

print(a)
print(b)
print(c)
```

```
>>> %Run -c $EDITOR_CONTENT
20.5
['Apfel', 'Banane', 'Kirsche']
Wahr
>>>
```

## Typumwandlung

Mit den Methoden `int()`, `float()` und `complex()` können Sie von einem Typ in einen anderen konvertieren. In Python erfolgt das Casting mithilfe von Konstruktorfunktionen:

- `int()` - erstellt eine Ganzzahl aus einem Ganzzahl-, Fließkomma- oder Zeichenliteral (vorausgesetzt, die Zeichenfolge stellt eine ganze Zahl dar)
- `float()` - erstellt eine Fließkommazahl aus einem Ganzzahl-, Fließkomma- oder Zeichenliteral (vorausgesetzt, die Zeichenfolge stellt eine Fließkommazahl oder eine ganze Zahl dar)
- `str()` - erstellt eine Zeichenfolge aus einer Vielzahl von Datentypen, einschließlich Zeichenfolgen, Ganzzahl- und Fließkommaliteralen

```
a = float("5")
b = int(3.7)
c = str(6.0)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
print(a)
print(b)
print(c)
```

Hinweis: Komplexe Zahlen können nicht in einen anderen Zahlenwert umgewandelt werden.

## 4.6.10 Operatoren

Operatoren dienen zur Durchführung von Operationen auf Variablen und Werten.

- *Arithmetische Operatoren*
- *Zuweisungsoperatoren*
- *Vergleichsoperatoren*
- *Logische Operatoren*
- *Identitätsoperatoren*
- *Mitgliedschaftsoperatoren*
- *Bitweise Operatoren*

### Arithmetische Operatoren

Mit arithmetischen Operatoren können Sie gängige mathematische Operationen durchführen.

Operator	Bezeichnung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	Modulo
**	Potenzierung
//	Ganzzahlige Division

```
x = 5
y = 3

a = x + y
b = x - y
c = x * y
d = x / y
e = x % y
f = x ** y
g = x // y

print(a)
print(b)
print(c)
print(d)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
print(e)
print(f)
print(g)
```

```
>>> %Run -c $EDITOR_CONTENT
8
2
15
1.666667
2
125
1
8
2
15
>>>
```

## Zuweisungsoperatoren

Zuweisungsoperatoren werden verwendet, um Werte Variablen zuzuweisen.

Operator	Beispiel	Entsprechend
=	a = 6	a = 6
+=	a += 6	a = a + 6
-=	a -= 6	a = a - 6
*=	a *= 6	a = a * 6
/=	a /= 6	a = a / 6
%=	a %= 6	a = a % 6
**=	a **= 6	a = a ** 6
//=	a //= 6	a = a // 6
&=	a &= 6	a = a & 6
=	a  = 6	a = a   6
^=	a ^= 6	a = a ^ 6
>>=	a >>= 6	a = a >> 6
<<=	a <<= 6	a = a << 6

```
a = 6
a *= 6
print(a)
```

```
>>> %Run test.py
36
>>>
```



## Vergleichsoperatoren

Vergleichsoperatoren werden verwendet, um zwei Werte miteinander zu vergleichen.

Operator	Bezeichnung
<code>==</code>	Gleich
<code>!=</code>	Ungleich
<code>&lt;</code>	Kleiner als
<code>&gt;</code>	Größer als
<code>&gt;=</code>	Größer oder gleich
<code>&lt;=</code>	Kleiner oder gleich

```
a = 6
b = 8

print(a > b)
```

```
>>> %Run test.py
False
>>>
```

Gibt **False** zurück, weil **a** kleiner als **b** ist.

## Logische Operatoren

Logische Operatoren werden verwendet, um Bedingungsanweisungen zu kombinieren.

Operator	Beschreibung
<code>and</code>	Gibt True zurück, wenn beide Aussagen wahr sind
<code>or</code>	Gibt True zurück, wenn eine der Aussagen wahr ist
<code>not</code>	Kehrt das Ergebnis um, gibt False zurück, wenn das Ergebnis wahr ist

```
a = 6
print(a > 2 and a < 8)
```

```
>>> %Run -c $EDITOR_CONTENT
True
>>>
```

## Identitätsoperatoren

Identitätsoperatoren dienen zum Vergleich von Objekten, nicht ob sie gleich sind, sondern ob es sich tatsächlich um dasselbe Objekt mit demselben Speicherort handelt.

Operator	Beschreibung
<code>is</code>	Gibt True zurück, wenn beide Variablen dasselbe Objekt sind
<code>is not</code>	Gibt True zurück, wenn beide Variablen nicht dasselbe Objekt sind

```
a = ["hello", "welcome"]
b = ["hello", "welcome"]
c = a

print(a is c)
# Gibt True zurück, da c dasselbe Objekt wie a ist

print(a is b)
# Gibt False zurück, da a nicht dasselbe Objekt wie b ist, auch wenn sie denselben
↪ Inhalt haben

print(a == b)
# Gibt True zurück, da a gleich b ist
```

```
>>> %Run -c $EDITOR_CONTENT
True
False
True
>>>
```

## Mitgliedschaftsoperatoren

Mitgliedschaftsoperatoren werden verwendet, um zu testen, ob eine Sequenz in einem Objekt enthalten ist.

Operator	Beschreibung
in	Gibt True zurück, wenn eine Sequenz mit dem angegebenen Wert im Objekt vorhanden ist
not in	Gibt True zurück, wenn eine Sequenz mit dem angegebenen Wert nicht im Objekt vorhanden ist

```
a = ["hello", "welcome", "Goodmorning"]

print("welcome" in a)
```

```
>>> %Run -c $EDITOR_CONTENT
True
>>>
```

## Bitweise Operatoren

Bitweise Operatoren werden zum Vergleichen von (binären) Zahlen verwendet.

Operator	Name	Beschreibung
&	UND	Setzt jedes Bit auf 1, wenn beide Bits 1 sind
	ODER	Setzt jedes Bit auf 1, wenn eines von zwei Bits 1 ist
^	XOR	Setzt jedes Bit auf 1, wenn nur eines von zwei Bits 1 ist
~	NICHT	Kehrt alle Bits um
<<	Zero-fill Linksschiebung	Verschiebt nach links, indem von rechts Nullen eingefügt werden und die am weitesten links stehenden Bits herausfallen
>>	Signierte Rechtsschiebung	Verschiebt nach rechts, indem Kopien des am weitesten links stehenden Bits von links eingefügt werden und die am weitesten rechts stehenden Bits herausfallen

```
num = 2

print(num & 1)
print(num | 1)
print(num << 1)
```

```
>>> %Run -c $EDITOR_CONTENT
0
3
4
>>>
```

#### 4.6.11 Listen

Listen dienen dazu, mehrere Elemente in einer einzigen Variable zu speichern und werden mit eckigen Klammern erstellt:

```
B_Liste = ["Blossom", "Bubbles", "Buttercup"]
print(B_Liste)
```

Listenelemente sind veränderbar, geordnet und erlauben doppelte Werte. Die Elemente der Liste sind indiziert, wobei das erste Element den Index [0], das zweite den Index [1] usw. hat.

```
C_Liste = ["Rot", "Blau", "Grün", "Blau"]
print(C_Liste)           # Duplikate erlaubt
print(C_Liste[0])
print(C_Liste[1])        # geordnet
C_Liste[2] = "Lila"      # veränderbar
print(C_Liste)
```

```
>>> %Run -c $EDITOR_CONTENT
['Rot', 'Blau', 'Grün', 'Blau']
Rot
Blau
['Rot', 'Blau', 'Lila', 'Blau']
```

Eine Liste kann unterschiedliche Datentypen enthalten:

```
A_Liste = ["Banane", 255, False, 3.14]
print(A_Liste)
```

```
>>> %Run -c $EDITOR_CONTENT  
['Banane', 255, False, 3.14]
```

### Listenlänge

Um herauszufinden, wie viele Elemente in der Liste enthalten sind, verwenden Sie die Funktion `len()`.

```
A_Liste = ["Banane", 255, False, 3.14]  
print(len(A_Liste))
```

```
>>> %Run -c $EDITOR_CONTENT  
4
```

### Listenelemente überprüfen

Drucken Sie das zweite Element der Liste aus:

```
A_Liste = ["Banane", 255, False, 3.14]  
print(A_Liste[1])
```

```
>>> %Run -c $EDITOR_CONTENT  
[255]
```

Drucken Sie das letzte Element der Liste aus:

```
A_Liste = ["Banane", 255, False, 3.14]  
print(A_Liste[-1])
```

```
>>> %Run -c $EDITOR_CONTENT  
[3.14]
```

Drucken Sie das zweite und dritte Element aus:

```
A_Liste = ["Banane", 255, False, 3.14]  
print(A_Liste[1:3])
```

```
>>> %Run -c $EDITOR_CONTENT  
[255, False]
```

### Listen-Elemente ändern

Ändere das zweite und dritte Element:

```
A_Liste = ["Banane", 255, False, 3.14]  
A_Liste[1:3] = [True, "Orange"]  
print(A_Liste)
```

```
>>> %Run -c $EDITOR_CONTENT  
['Banane', True, 'Orange', 3.14]
```

Ersetze das zweite Element durch zwei Werte:

```
A_Liste = ["Banane", 255, False, 3.14]
A_Liste[1:2] = [True, "Orange"]
print(A_Liste)
```

```
>>> %Run -c $EDITOR_CONTENT
['Banane', True, 'Orange', False, 3.14]
```

### Listenelemente hinzufügen

Mit der append()-Methode ein Element hinzufügen:

```
C_Liste = ["Rot", "Blau", "Grün"]
C_Liste.append("Orange")
print(C_Liste)
```

```
>>> %Run -c $EDITOR_CONTENT
['Rot', 'Blau', 'Grün', 'Orange']
```

Ein Element an der zweiten Position einfügen:

```
C_Liste = ["Rot", "Blau", "Grün"]
C_Liste.insert(1, "Orange")
print(C_Liste)
```

```
>>> %Run -c $EDITOR_CONTENT
['Rot', 'Orange', 'Blau', 'Grün']
```

### Listenelemente entfernen

Die remove()-Methode entfernt das angegebene Element.

```
C_Liste = ["Rot", "Blau", "Grün"]
C_Liste.remove("Blau")
print(C_Liste)
```

```
>>> %Run -c $EDITOR_CONTENT
['Rot', 'Grün']
```

Die pop()-Methode entfernt das Element am angegebenen Index. Wenn kein Index angegeben wird, entfernt die pop()-Methode das letzte Element.

```
A_Liste = ["Banane", 255, False, 3.14, True, "Orange"]
A_Liste.pop(1)
print(A_Liste)
A_Liste.pop()
print(A_Liste)
```

```
>>> %Run -c $EDITOR_CONTENT
255
['Banane', False, 3.14, True, 'Orange']
'Orange'
['Banane', False, 3.14, True]
```

Das Schlüsselwort `del` entfernt ebenfalls den angegebenen Index:

```
C_Liste = ["Rot", "Blau", "Grün"]
del C_Liste[1]
print(C_Liste)
```

```
>>> %Run -c $EDITOR_CONTENT
['Rot', 'Grün']
```

Die `clear()`-Methode leert die Liste. Die Liste bleibt bestehen, hat aber keinen Inhalt mehr.

```
C_Liste = ["Rot", "Blau", "Grün"]
C_Liste.clear()
print(C_Liste)
```

```
>>> %Run -c $EDITOR_CONTENT
[]
```

## 2. Ausgabe & Eingabe

### 4.7 2.1 Hallo, LED!

Genau wie das Ausgeben von „Hallo, Welt!“ der erste Schritt beim Erlernen der Programmierung ist, stellt die Ansteuerung einer LED mit einem Programm die traditionelle Einführung in die physische Programmierung dar.

- *LED*

#### Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten:

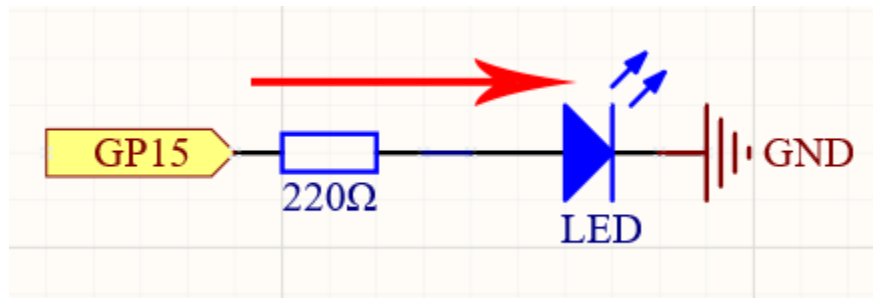
Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler-Set	450+	

Die Einzelteile können auch über die unten stehenden Links separat erworben werden.

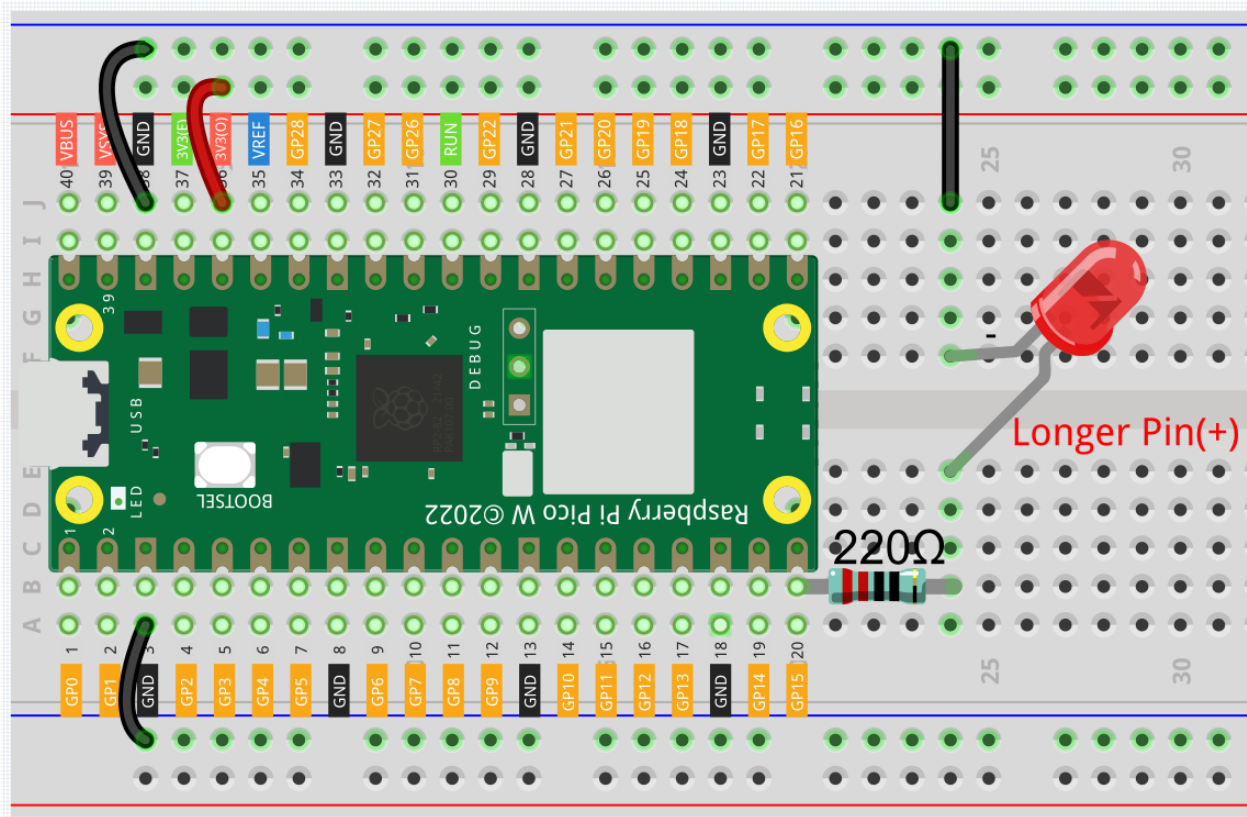
SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1 (220)	
6	<i>LED</i>	1	

### Schaltplan



Dieser Schaltkreis funktioniert nach einem einfachen Prinzip, die Stromrichtung ist in der Abbildung dargestellt. Die LED leuchtet auf, wenn GP15 eine hohe Ausgangsspannung (3,3 V) liefert, und erlischt, wenn GP15 eine niedrige Ausgangsspannung (0 V) liefert.

### Verdrahtung



Um den Schaltkreis aufzubauen, folgen wir der Stromrichtung!

1. Die LED wird über den GP15-Pin der Pico-W-Platine mit Strom versorgt; hier beginnt der Schaltkreis.
2. Um die LED zu schützen, muss der Strom durch einen 220-Ohm-Widerstand fließen. Ein Ende des Widerstands sollte in die gleiche Reihe wie der Pico-W GP15-Pin (Reihe 20 in meinem Schaltkreis) eingefügt werden, das andere Ende in eine freie Reihe des Steckbretts (Reihe 24).

---

**Bemerkung:** Der Farbring des 220-Ohm-Widerstands ist rot, rot, schwarz, schwarz und braun.

---

3. Wenn Sie die LED aufnehmen, stellen Sie fest, dass einer der Anschlüsse länger ist als der andere. Verbinden Sie den längeren Anschluss mit der gleichen Reihe wie der Widerstand und den kürzeren Anschluss mit der Reihe gegenüber der Mittellücke auf dem Steckbrett.

---

**Bemerkung:** Der längere Anschluss ist die Anode und repräsentiert die positive Seite des Schaltkreises; der kürzere ist die Kathode und steht für die negative Seite.

Die Anode muss über einen Widerstand mit dem GPIO-Pin verbunden sein; die Kathode muss mit dem GND-Pin verbunden sein.

---

4. Verwenden Sie ein Stecker-zu-Stecker (M2M) Verbindungskabel, um den kurzen Anschluss der LED mit der negativen Stromschiene des Steckbretts zu verbinden.
5. Verbinden Sie den GND-Pin des Pico W mit der negativen Stromschiene des Steckbretts mithilfe eines Verbindungskabels.

**Code**



**Bemerkung:**

- Öffnen Sie die Datei `2.1_hello_led.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5, um es auszuführen.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
- Für detaillierte Anleitungen beziehen Sie sich bitte auf *Code direkt öffnen und ausführen*.

```
import machine
import utime

led = machine.Pin(15, machine.Pin.OUT)
while True:
    led.value(1)
    utime.sleep(2)
    led.value(0)
    utime.sleep(2)
```

Nachdem der Code ausgeführt wurde, sehen Sie die LED blinken.

**Wie funktioniert es?**

Die Bibliothek „machine“ wird benötigt, um GPIO zu nutzen.

```
import machine
```

Diese Bibliothek enthält alle Anweisungen, die zur Kommunikation zwischen MicroPython und Pico W erforderlich sind. Fehlt diese Zeile im Code, können wir keine GPIOs steuern.

Das Nächste, was auffällt, ist diese Zeile:

```
led = machine.Pin(15, machine.Pin.OUT)
```

Hier wird das Objekt `led` definiert. Technisch gesehen könnte es jeden beliebigen Namen haben, wie `x`, `y`, `Banane` oder `Michael_Jackson`. Um die Lesbarkeit des Programms zu gewährleisten, sollte ein beschreibender Name gewählt werden.

Im zweiten Teil dieser Zeile (dem Teil nach dem Gleichheitszeichen) rufen wir die Funktion „Pin“ aus der „machine“-Bibliothek auf. Diese dient dazu, den GPIO-Pins des Pico W zu sagen, was sie tun sollen. Die Funktion „Pin“ hat zwei Parameter: Der erste (15) gibt an, welcher Pin eingestellt werden soll; der zweite Parameter (`machine.Pin.OUT`) gibt an, dass der Pin als Ausgang und nicht als Eingang fungieren soll.

Der oben stehende Code hat den Pin „eingestellt“, aber er wird die LED noch nicht zum Leuchten bringen. Dazu müssen wir den Pin auch „nutzen“.

```
led.value(1)
```

Der GP15-Pin wurde zuvor eingerichtet und heißt `led`. Die Funktion dieser Anweisung besteht darin, den Wert von `led` auf 1 zu setzen.

Um GPIO zu verwenden, sind folgende Schritte notwendig:

- **„machine“-Bibliothek importieren:** Dies ist ein notwendiger Schritt und wird nur einmal durchgeführt.
- **GPIO einstellen:** Vor der Nutzung muss jeder Pin konfiguriert werden.

- **Verwenden:** Durch Zuweisung eines Wertes wird der Arbeitszustand des Pins verändert.

Folgt man diesen Anweisungen, erhält man beispielsweise folgenden Code:

```
import machine
led = machine.Pin(15, machine.Pin.OUT)
led.value(1)
```

Führt man diesen aus, wird die LED leuchten.

Als nächstes fügen wir die „Ausschalt“-Anweisung hinzu:

```
import machine
led = machine.Pin(15, machine.Pin.OUT)
led.value(1)
led.value(0)
```

Gemäß dem Code sollte das Programm die LED zunächst einschalten und dann wieder ausschalten. In der Praxis stellt sich jedoch heraus, dass dies nicht der Fall ist. Die LED leuchtet nicht, da die beiden Zeilen sehr schnell nacheinander ausgeführt werden, schneller als das menschliche Auge reagieren kann. Das lässt sich beheben, indem das Programm verlangsamt wird.

Für diesen Zweck sollte die zweite Zeile des Programms wie folgt aussehen:

```
import utime
```

Ähnlich wie bei `machine` importieren wir hier die Bibliothek `utime`, die Zeitfunktionen verwaltet. Nun fügen wir zwischen `led.value(1)` und `led.value(0)` eine Verzögerung von 2 Sekunden ein.

```
utime.sleep(2)
```

So sollte der Code jetzt aussehen. Beim Ausführen wird nun zuerst die LED eingeschaltet und dann ausgeschaltet:

```
import machine
import utime
led = machine.Pin(15, machine.Pin.OUT)
led.value(1)
utime.sleep(2)
led.value(0)
```

Zuletzt soll die LED blinken. Dafür erstellen wir eine Schleife und ändern das Programm entsprechend.

```
import machine
import utime

led = machine.Pin(15, machine.Pin.OUT)
while True:
    led.value(1)
    utime.sleep(2)
    led.value(0)
    utime.sleep(2)
```

- *While-Schleifen*

### Weitere Informationen

Normalerweise gibt es eine API-Dokumentation, die mit der Bibliothek verknüpft ist. Diese enthält alle notwendigen Informationen für die Verwendung der Bibliothek, einschließlich detaillierter Beschreibungen von Funktionen, Klassen, Rückgabetypen, Parameterarten usw.

In diesem Artikel haben wir MicroPythons `machine` und `utime` Bibliotheken verwendet; weitere Verwendungsmöglichkeiten finden Sie hier:

- `machine.Pin`
- `utime`

Um dieses Beispiel des LED-Blinkens zu verstehen, lesen Sie bitte die API-Dokumentation!

#### Bemerkung:

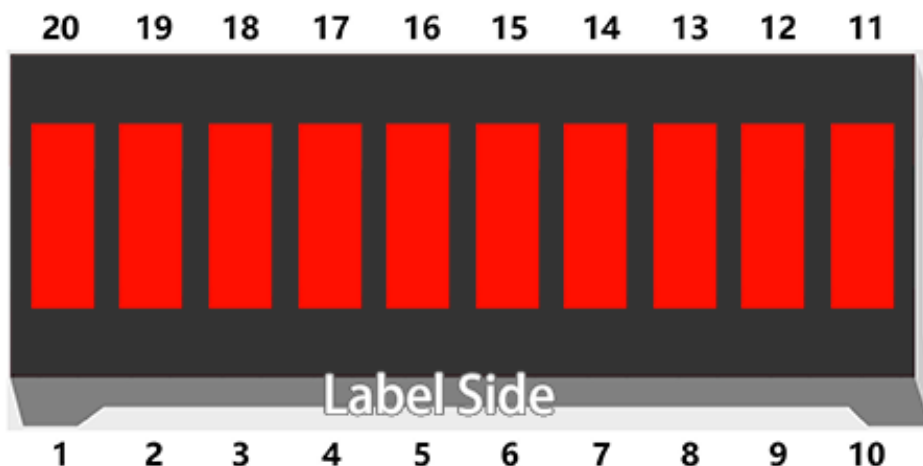
- Öffnen Sie die Datei `2.1_hello_led_2.py` im Ordner `kepler-kit-main/micropython` oder kopieren Sie den Code in Thonny. Dann klicken Sie auf „Aktuelles Skript ausführen“ oder drücken einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
- Für detaillierte Anleitungen beachten Sie bitte *Code direkt öffnen und ausführen*.

```
import machine
import utime

led = machine.Pin(15, machine.Pin.OUT)
while True:
    led.toggle()
    utime.sleep(1)
```

## 4.8 2.2 Anzeige des Pegels

Das erste Projekt besteht einfach darin, eine LED blinken zu lassen. Für dieses Projekt verwenden wir die LED-Balkenanzeige, die aus 10 LEDs in einem Kunststoffgehäuse besteht und normalerweise zur Anzeige von Leistung oder Lautstärke verwendet wird.



- *LED-Balkendiagramm*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

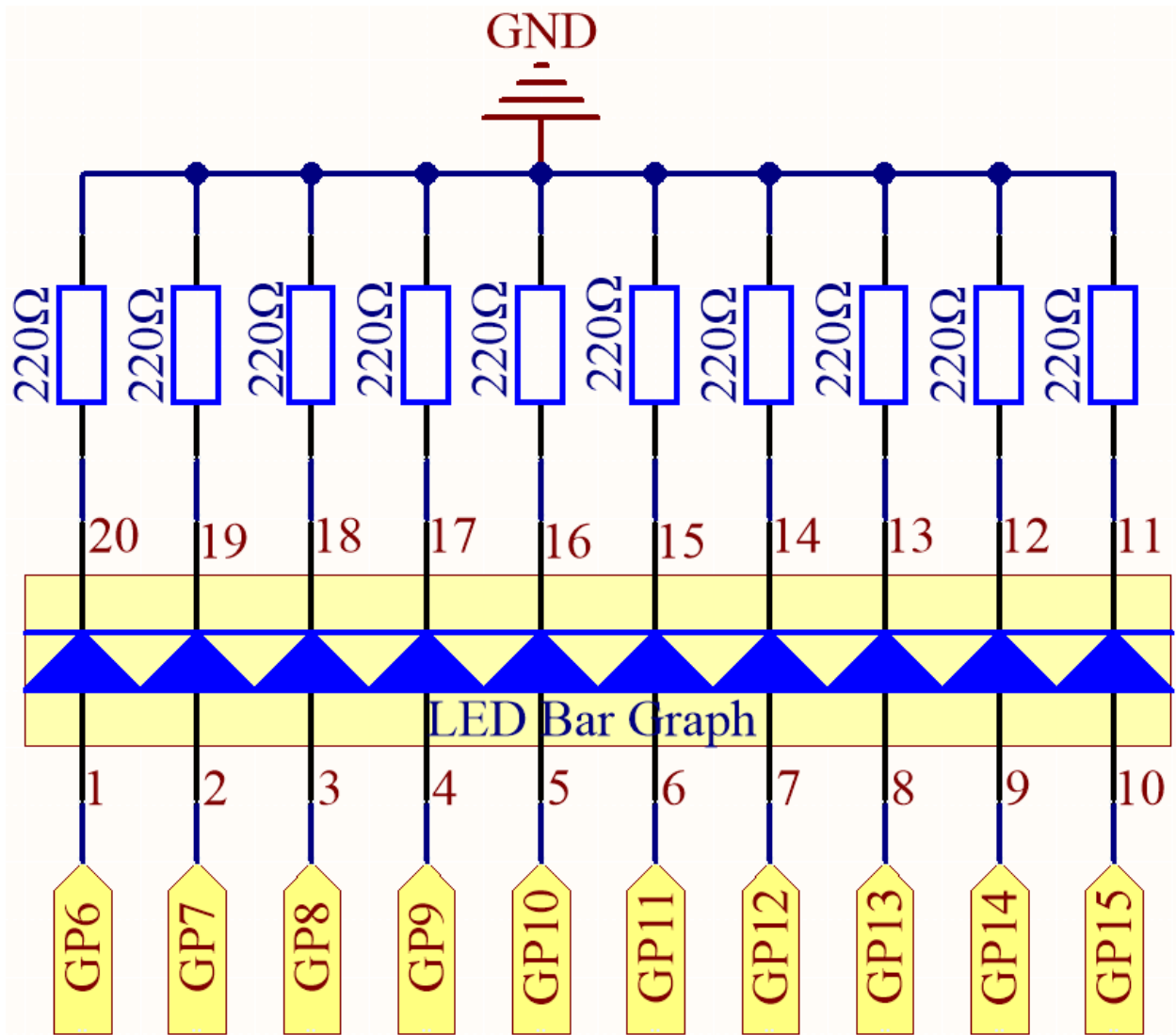
Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die Teile auch einzeln über die untenstehenden Links kaufen.

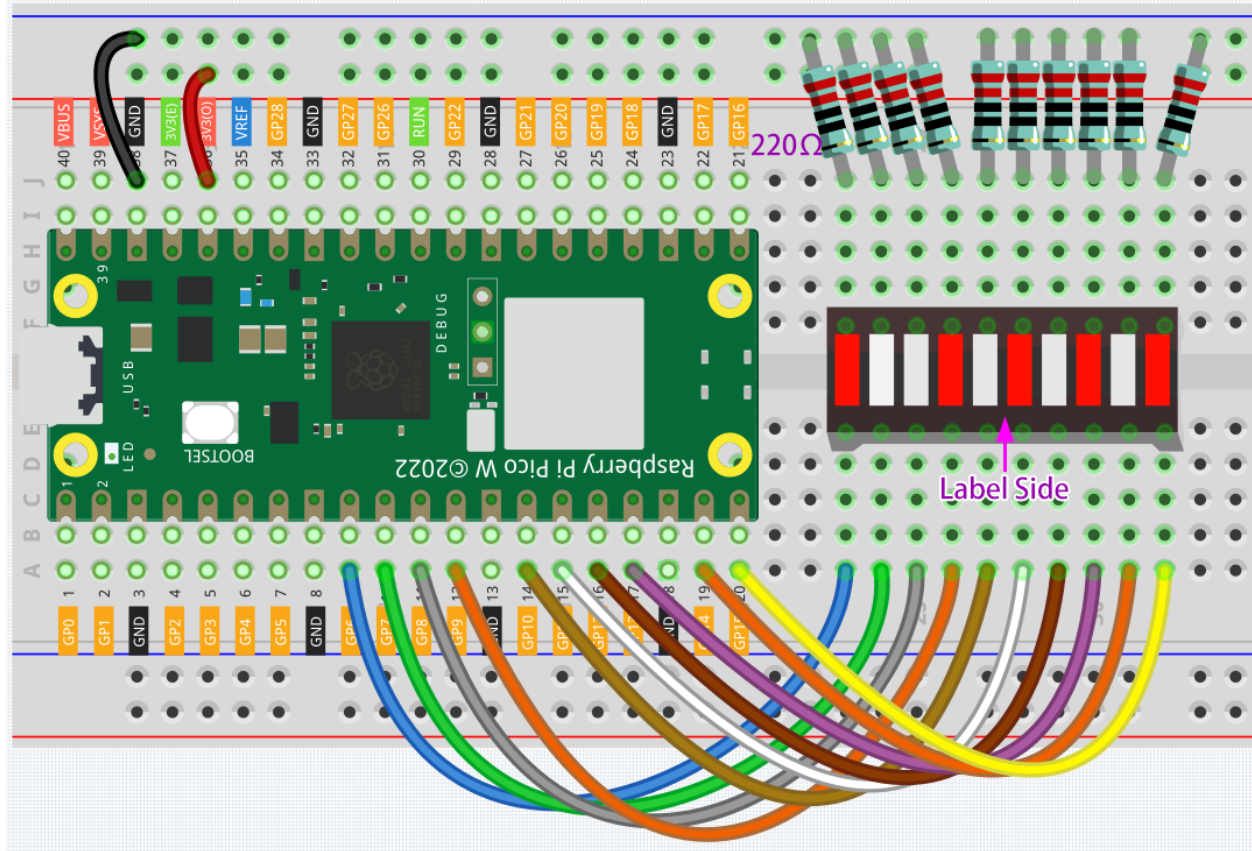
SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	10 (220)	
6	<i>LED-Balkendiagramm</i>	1	

### Schaltplan



In der LED-Balkenanzeige gibt es 10 LEDs, von denen jede individuell gesteuert werden kann. Die Anode jeder LED ist mit GP6\*GP15 verbunden, die Kathode über einen 220-Ohm-Widerstand mit GND.

#### Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `2.2_display_the_level.py` im Ordner `kepler-kit-main/micropython` oder kopieren Sie den Code in Thonny und klicken Sie dann auf „Run Current Script“ oder drücken einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
- Für detaillierte Anleitungen beachten Sie bitte *Code direkt öffnen und ausführen*.

```
import machine
import utime

pin = [6,7,8,9,10,11,12,13,14,15]
led= []
for i in range(10):
    led.append(None)
    led[i] = machine.Pin(pin[i], machine.Pin.OUT)

while True:
    for i in range(10):
        led[i].toggle()
        utime.sleep(0.2)
```

Auf der LED-Balkenanzeige sehen Sie, dass die LEDs in einer Sequenz aufleuchten und wieder erlöschen, während

das Programm läuft.

### Wie funktioniert das?

Die LED-Balkenanzeige besteht aus zehn LEDs, die von zehn Pins gesteuert werden. Das bedeutet, dass wir diese Pins definieren müssen. Es wäre zu mühsam, sie einzeln zu definieren. Daher verwenden wir hier **Lists**.

**Bemerkung:** Python-Listen sind einer der vielseitigsten Datentypen, die es uns ermöglichen, mit mehreren Elementen gleichzeitig zu arbeiten. Sie werden durch das Platzieren von Elementen in eckigen Klammern `[]` erstellt, die durch Kommas getrennt sind.

```
pin = [6,7,8,9,10,11,12,13,14,15]
```

Mit dieser Codezeile wird eine Liste `pin` definiert, die die zehn Elemente `6,7,8,9,10,11,12,13,14,15` enthält. Wir können den Index-Operator `[]` verwenden, um auf ein Element in einer Liste zuzugreifen. In Python beginnen die Indizes bei 0. Daher wird eine Liste mit 10 Elementen einen Index von 0 bis 9 haben. Bei dieser Liste als Beispiel ist `pin[0]` gleich 6 und `pin[4]` gleich 10.

Als Nächstes deklarieren Sie eine leere Liste `led`, die zur Definition von zehn LED-Objekten verwendet wird.

```
led = []
```

Aufgrund der Länge der Liste, die 0 beträgt, funktionieren direkte Operationen auf dem Array, wie zum Beispiel das Drucken von `led[0]`, nicht. Es gibt neue Elemente, die wir hinzufügen müssen.

```
led.append(None)
```

Durch diese `append()` Methode hat die Liste `led` ihr erstes Element erhalten, mit einer Länge von 1, und `led[0]` wird zu einem gültigen Element, obwohl sein aktueller Wert `None` ist (was für Null steht).

Der nächste Schritt besteht darin, `led[0]`, die an Pin 6 angeschlossene LED, als das erste LED-Objekt zu definieren.

```
led[0] = machine.Pin(6, machine.Pin.OUT)
```

Das erste LED-Objekt ist nun definiert.

Wie Sie sehen können, haben wir die zehn Pin-Nummern als Liste **pin** erstellt, die wir in diese Zeile einfügen können, um Massenoperationen zu erleichtern.

```
led[0] = machine.Pin(pin[0], machine.Pin.OUT)
```

Verwenden Sie eine `for`-Schleife, damit alle 10 Pins den obigen Befehl ausführen.

```
import machine

pin = [6,7,8,9,10,11,12,13,14,15]
led = []
for i in range(10):
    led.append(None)
    led[i] = machine.Pin(pin[i], machine.Pin.OUT)
```

- *Listen*
- *For-Schleifen*

Verwenden Sie eine weitere `for`-Schleife, um die zehn LEDs auf der LED-Balkenanzeige nacheinander umzuschalten.

```
for i in range(10):
    led[i].toggle()
    utime.sleep(0.2)
```

Der Code wird abgeschlossen, indem der obige Codeblock in eine While-Schleife eingefügt wird.

```
import machine
import utime

pin = [6,7,8,9,10,11,12,13,14,15]
led = []
for i in range(10):
    led.append(None)
    led[i] = machine.Pin(pin[i], machine.Pin.OUT)

while True:
    for i in range(10):
        led[i].toggle()
        utime.sleep(0.2)
```

## 4.9 2.3 Abklingende LED

Bislang haben wir lediglich zwei Arten von Ausgangssignalen verwendet: hohes und niedriges Signalniveau, auch als EIN und AUS bezeichnet. Das nennen wir digitalen Ausgang. In der Praxis arbeiten jedoch viele Geräte nicht ausschließlich mit diesen zwei Zuständen, sondern benötigen differenziertere Steuerungen, wie etwa die Geschwindigkeitsregelung eines Motors oder die Helligkeitsanpassung einer Schreibtischlampe. Früher wurden dafür Schieberegler zur Widerstandseinstellung genutzt, was allerdings als unzuverlässig und ineffizient gilt. Deshalb hat sich die Pulsweitenmodulation (PWM) als praktikable Lösung für solche komplexen Anforderungen etabliert.

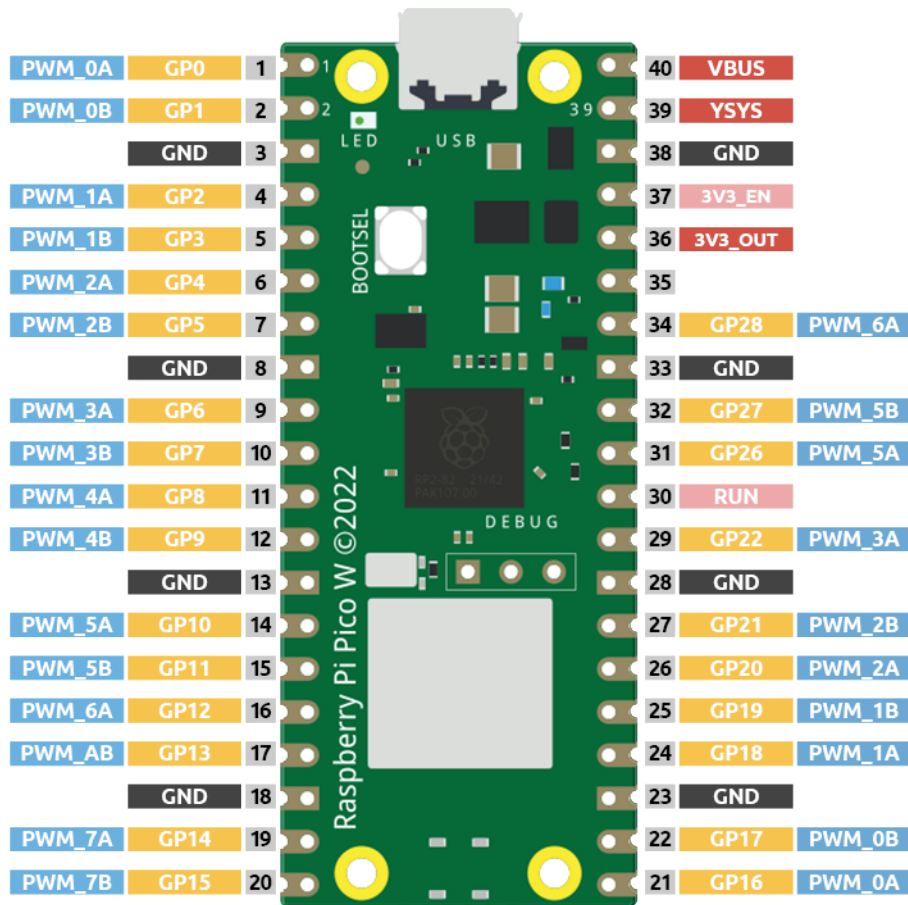
Ein Puls ist im Kontext digitaler Ausgänge ein Signal, das sowohl ein hohes als auch ein niedriges Niveau enthält. Die Breite dieser Pulse kann durch Veränderung der Schaltfrequenz zwischen EIN und AUS angepasst werden.

Innerhalb einer kurzen Zeitspanne – etwa 20 Millisekunden, die der visuellen Speicherzeit der meisten Menschen entspricht – lässt sich eine LED ein- und wieder ausschalten, ohne dass uns das Auge den Wechsel als solchen wahrnimmt. Allerdings erscheint die Helligkeit der LED dann etwas reduziert. Während dieser Phase gilt: Je länger die LED leuchtet, desto heller erscheint sie. Mit anderen Worten, je breiter der Puls im gegebenen Zyklus ist, desto größer ist die vom Mikrocontroller ausgegebene elektrische Signalstärke. Das ist der Mechanismus, mit dem PWM die Helligkeit der LED (oder die Geschwindigkeit des Motors) steuert.

- [Pulsweitenmodulation – Wikipedia](#)

Beim Einsatz von PWM mit dem Pico W gibt es einige Besonderheiten zu beachten. Betrachten wir hierzu folgendes Bild:





Der Pico W bietet für jeden GPIO-Pin Unterstützung für PWM, allerdings sind nur 16 unabhängige PWM-Ausgänge (anstatt 30) verfügbar. Diese sind zwischen GP0 und GP15 auf der linken Seite verteilt, und die PWM-Ausgabe der rechten GPIO-Pins ist identisch mit der der linken.

Es ist ratsam, im Code nicht denselben PWM-Kanal für unterschiedliche Anwendungen zu nutzen. So sind beispielsweise GP0 und GP16 beide PWM\_0A.

Nachdem wir nun das nötige Verständnis haben, wollen wir den Effekt der abklingenden LED erzielen.

- *LED*

### Erforderliche Bauteile

Für dieses Projekt benötigen wir die nachstehend aufgeführten Bauteile.

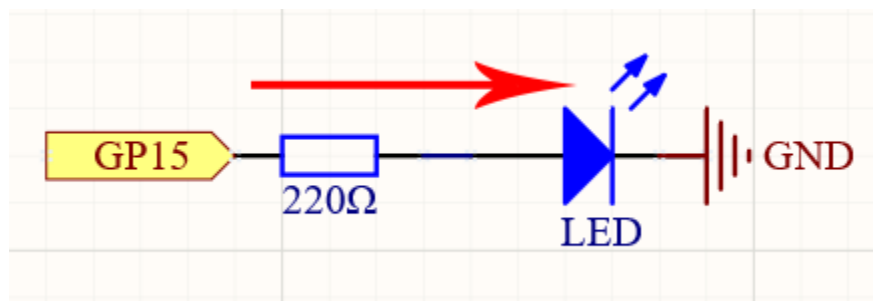
Ein komplettes Set ist definitiv praktisch; hier ist der entsprechende Link:

Name	Bestandteile des Sets	Link
Kepler Set	450+	

Alternativ können die Komponenten auch einzeln erworben werden.

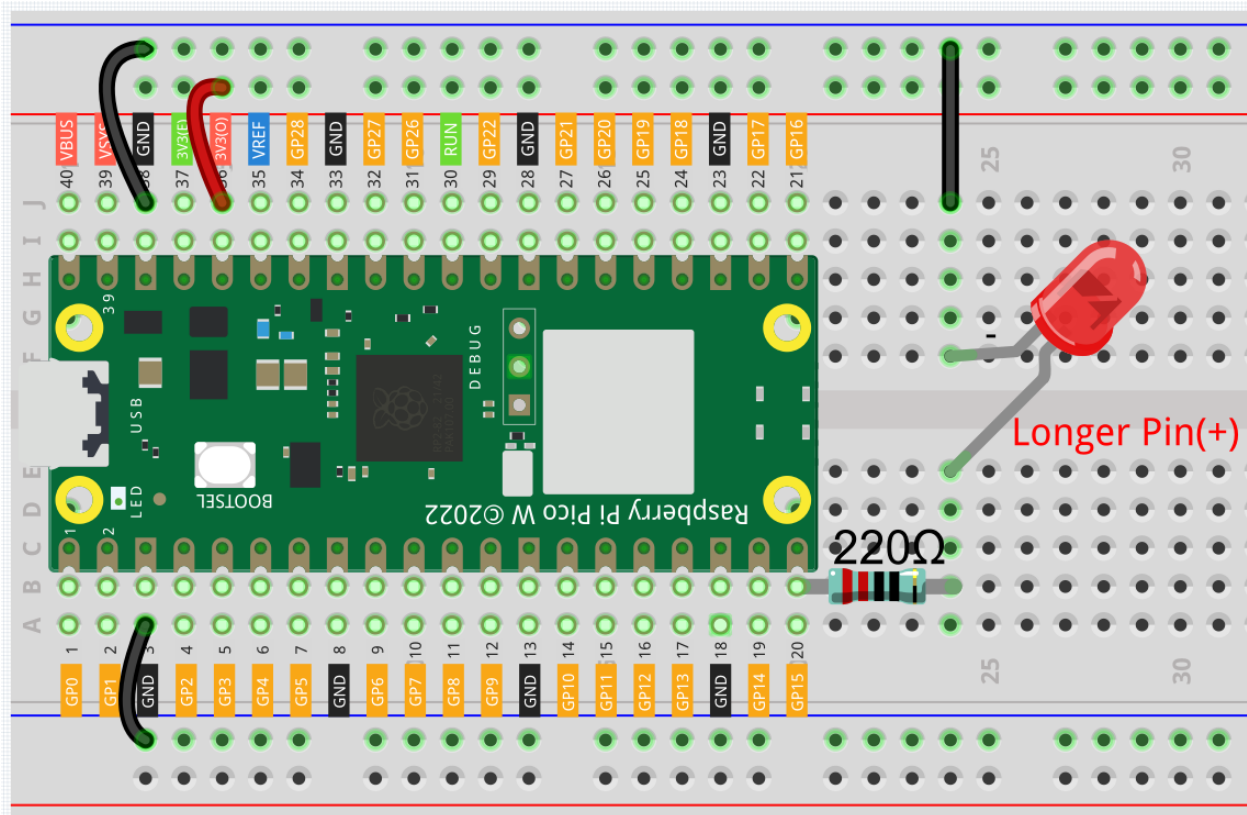
SN	Bauteil	Anzahl	Link
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1 (220)	
6	<i>LED</i>	1	

### Schaltplan



Dieses Projekt verwendet die gleiche Schaltung wie das erste Projekt [2.1 Hallo, LED!](#), jedoch unterscheidet sich die Art des Signals. Während das erste Projekt digitale Hoch- und Tiefpegel (0&1) direkt von GP15 ausgibt, um die LEDs ein- oder auszuschalten, wird in diesem Projekt ein PWM-Signal von GP15 zur Helligkeitssteuerung der LED verwendet.

### Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `2.3_fading_led.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den Interpreter „MicroPython (Raspberry Pi Pico)“ in der rechten unteren Ecke auszuwählen.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
import utime

led = machine.PWM(machine.Pin(15))
led.freq(1000)

for brightness in range(0,65535,50):
    led.duty_u16(brightness)
    utime.sleep_ms(10)
led.duty_u16(0)
```

Mit dem Ausführen des Codes wird die LED allmählich heller.

### Funktionsweise

Hier ändern wir die Helligkeit der LED, indem wir den Tastgrad des PWM-Ausgangs von GP15 variieren. Werfen wir einen Blick auf diese Codezeilen.

```
import machine
import utime

led = machine.PWM(machine.Pin(15))
led.freq(1000)

for brightness in range(0,65535,50):
    led.duty_u16(brightness)
    utime.sleep_ms(10)
led.duty_u16(0)
```

- `led = machine.PWM(machine.Pin(15))` definiert den Pin GP15 als PWM-Ausgang.
- Mit der Zeile `led.freq(1000)` wird die PWM-Frequenz eingestellt, in diesem Fall auf 1000 Hz, was bedeutet, dass ein Zyklus 1 ms (1/1000) dauert.
- `led.duty_u16()` legt den Tastgrad fest, der als 16-Bit-Ganzzahl ( $2^{16}=65536$ ) repräsentiert ist. Eine 0 steht für einen Tastgrad von 0%, d.h. der Pin bleibt während des gesamten Zyklus auf niedrigem Pegel. Der Wert 65535 entspricht einem Tastgrad von 100%, was bedeutet, dass der Ausgang durchgehend auf hohem Pegel ist und das Ergebnis '1' ist. Bei einem Wert von 32768 ist der Ausgang zur Hälfte der Zeit auf hohem Pegel, wodurch die LED nur halb so hell leuchtet.

## 4.10 2.4 Farbenfrohes Licht

Wie wir wissen, kann Licht überlagert werden. Zum Beispiel ergibt die Kombination von blauem und grünem Licht Zyanlicht, während rotes und grünes Licht Gelblicht erzeugen. Dieses Prinzip wird als „Additive Farbmischung“ bezeichnet.

- [Additive Farbe – Wikipedia](#)

Basierend auf dieser Methode können wir die drei Grundfarben verwenden, um sichtbares Licht jeder beliebigen Farbe in verschiedenen Verhältnissen zu mischen. Beispielsweise kann Orange durch mehr Rot und weniger Grün erzeugt werden.

In diesem Kapitel werden wir die Geheimnisse der additiven Farbmischung mit einer RGB-LED erkunden!

Eine RGB-LED entspricht im Grunde einer Kapselung von roten, grünen und blauen LEDs unter einer einzigen Lampenabdeckung. Diese drei LEDs teilen sich einen gemeinsamen Kathodenpin. Da jedem Anodenpin ein elektrisches Signal zugeführt wird, kann das Licht der entsprechenden Farbe dargestellt werden. Durch die Änderung der Signalintensität jeder Anode können vielfältige Farben erzeugt werden.

- [RGB-LED](#)

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

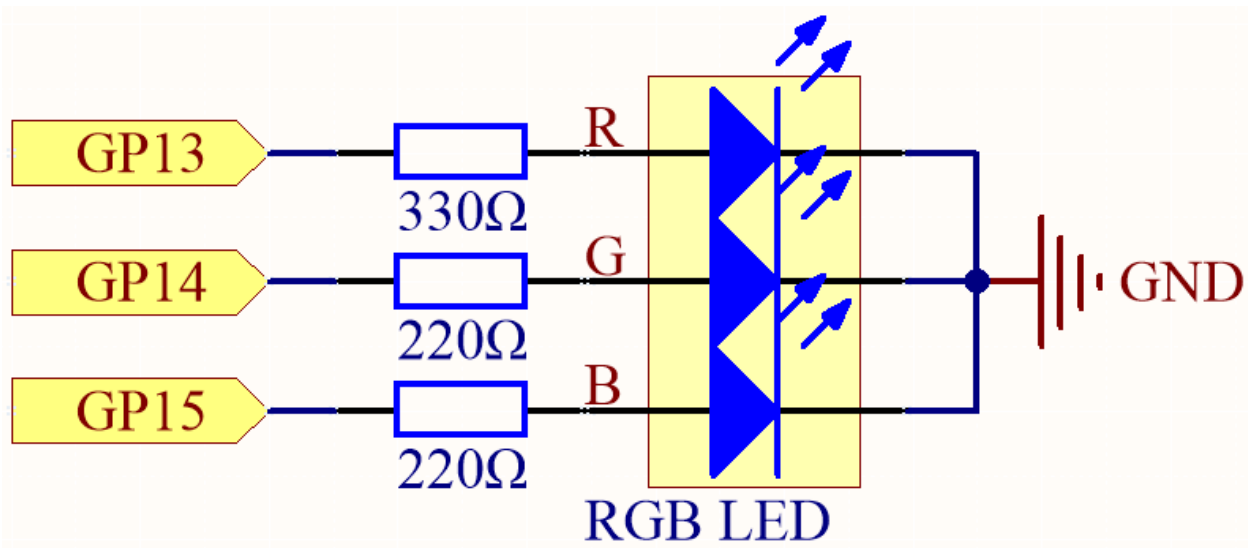
Ein vollständiges Set ist definitiv praktisch. Hier ist der Link dazu:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die Teile auch einzeln über die untenstehenden Links erwerben.

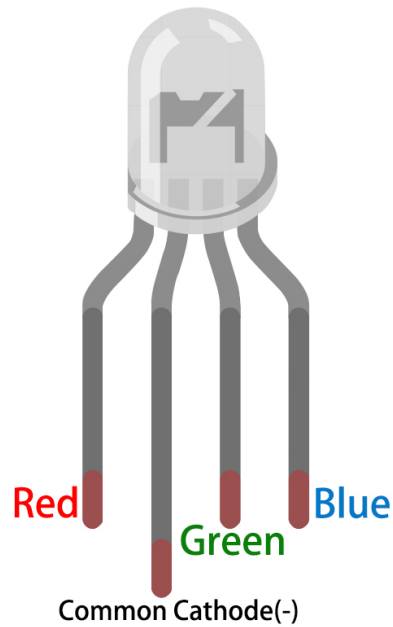
SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	3(1-330, 2-220)	
6	<i>RGB-LED</i>	1	

### Schaltplan

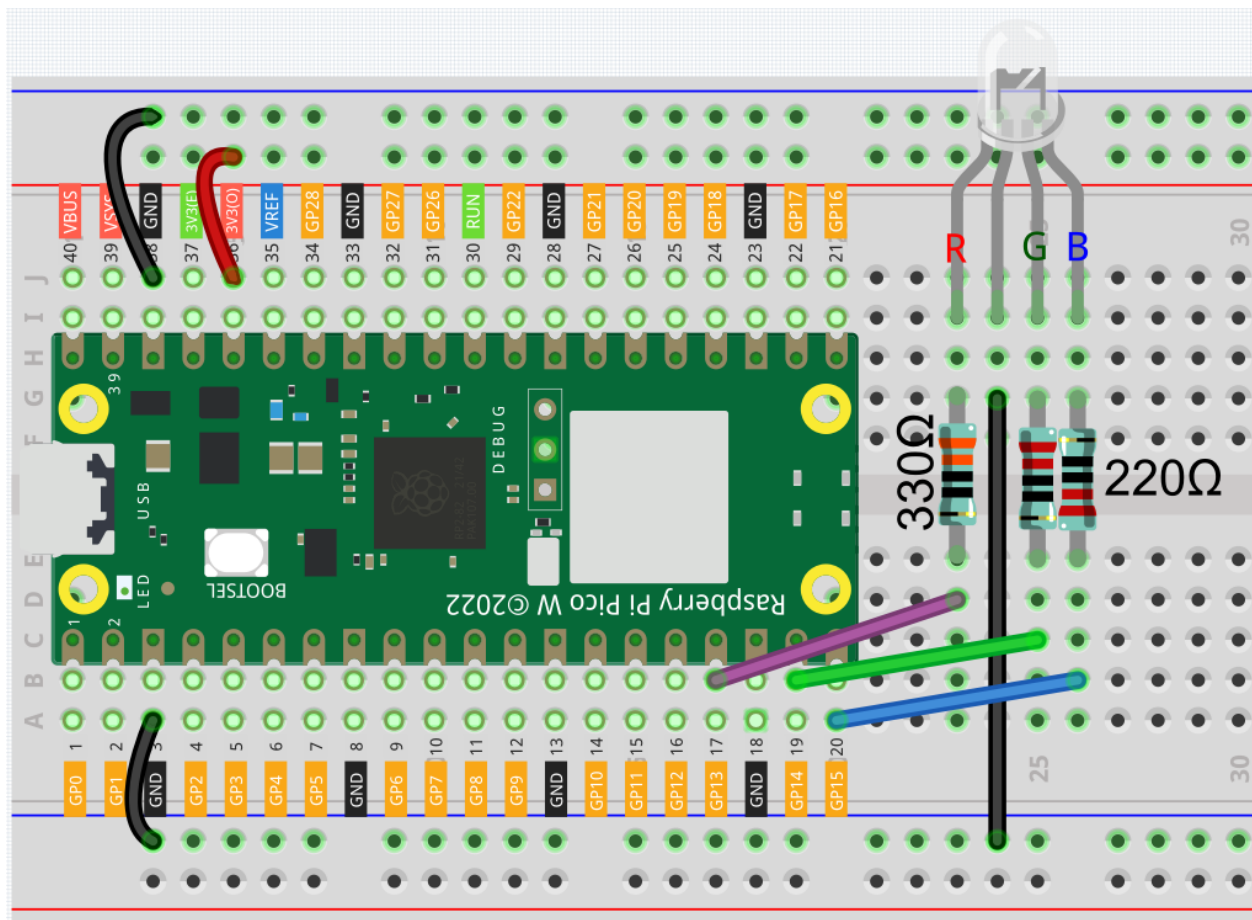


Die PWM-Pins GP13, GP14 und GP15 steuern jeweils die Rot-, Grün- und Blau-Pins der RGB-LED. Der gemeinsame Kathodenpin wird mit GND verbunden. So kann die RGB-LED durch Überlagerung von Licht mit verschiedenen PWM-Werten auf diesen Pins eine bestimmte Farbe anzeigen.

### Verdrahtung



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



Code

**Bemerkung:**

- Öffnen Sie die Datei `2.4_colorful_light.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, im unteren rechten Bereich auf den „MicroPython (Raspberry Pi Pico)“-Interpreter zu klicken.
- Für detaillierte Anleitungen siehe *Code direkt öffnen und ausführen*.

```
import machine
import utime

red = machine.PWM(machine.Pin(13))
green = machine.PWM(machine.Pin(14))
blue = machine.PWM(machine.Pin(15))
red.freq(1000)
green.freq(1000)
blue.freq(1000)

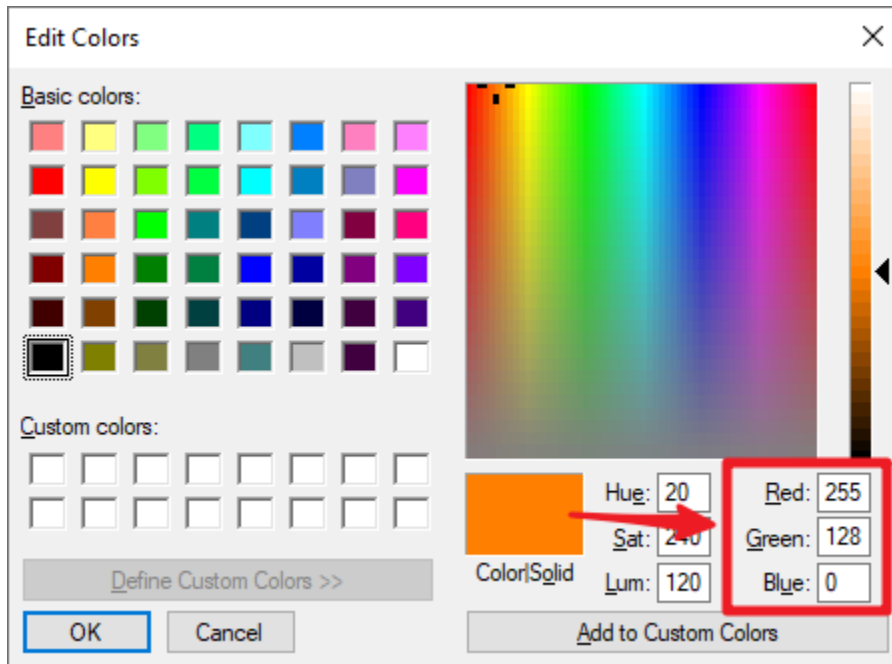
def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def color_to_duty(rgb_value):
    rgb_value = int(interval_mapping(rgb_value, 0, 255, 0, 65535))
    return rgb_value

def color_set(red_value, green_value, blue_value):
    red.duty_u16(color_to_duty(red_value))
    green.duty_u16(color_to_duty(green_value))
    blue.duty_u16(color_to_duty(blue_value))

color_set(255, 128, 0)
```

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



Tragen Sie den RGB-Wert in `color_set()` ein, um die gewünschten Farben mit der RGB-LED darzustellen.

#### Wie funktioniert es?

Um die drei Grundfarben gemeinsam nutzen zu können, haben wir eine `color_set()` Funktion definiert.

Aktuell verwenden Pixel in Computerhardware meist eine 24-Bit-Darstellung. Jede Grundfarbe wird in 8 Bit unterteilt, und der Farbwertbereich liegt zwischen 0 und 255. Es gibt 256 mögliche Kombinationen für jede der drei Grundfarben (vergessen Sie nicht, 0 zu zählen!), also  $256 \times 256 \times 256 = 16.777.216$  Farben. Die `color_set()` Funktion verwendet ebenfalls die 24-Bit-Notation, um die Farbauswahl zu vereinfachen.

Da der Wertebereich von `duty_u16()` 0~65535 beträgt (anstelle von 0 bis 255), wenn die Ausgangssignale über PWM zur RGB-LED gesendet werden, haben wir die Funktionen `color_to_duty()` und `interval_mapping()` definiert, um die Farbwerte auf die Tastverhältniswerte abzubilden.

## 4.11 2.5 Tastenwert auslesen

Diese Pins haben sowohl Eingabe- als auch Ausgabefunktionen, wie es der Name GPIO (General-purpose input/output) bereits andeutet. Bisher haben wir die Ausgabefunktion genutzt; in diesem Kapitel verwenden wir die Eingabefunktion, um den Tastenwert einzulesen.

- *Taster*

#### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Ein vollständiges Set zu kaufen ist natürlich bequem, hier ist der Link:

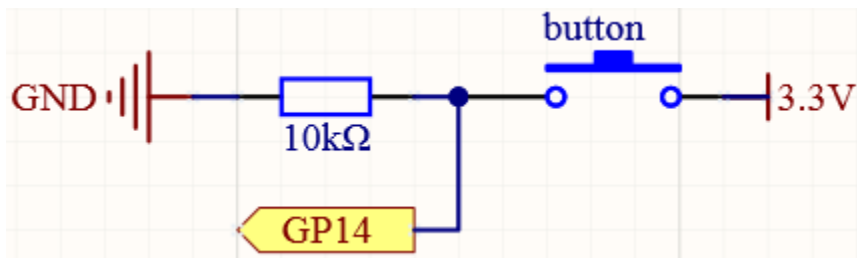
Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Alternativ können Sie die Komponenten auch einzeln über die folgenden Links erwerben.



SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Taster</i>	1	

### Schaltplan



Solange eine Seite des Tastenpins mit 3,3 V verbunden ist und der andere Pin mit GP14 verbunden ist, wird GP14 hoch sein, wenn die Taste gedrückt wird. Wenn die Taste jedoch nicht gedrückt ist, befindet sich GP14 in einem unbestimmten Zustand und kann hoch oder niedrig sein. Um ein stabiles niedriges Signalniveau zu erhalten, wenn die Taste nicht gedrückt ist, muss GP14 über einen 10K-Pull-down-Widerstand erneut mit GND verbunden werden.

### Verdrahtung

Solange die Taste nicht gedrückt ist, sind die linken und rechten Pins voneinander unabhängig, und der Strom kann

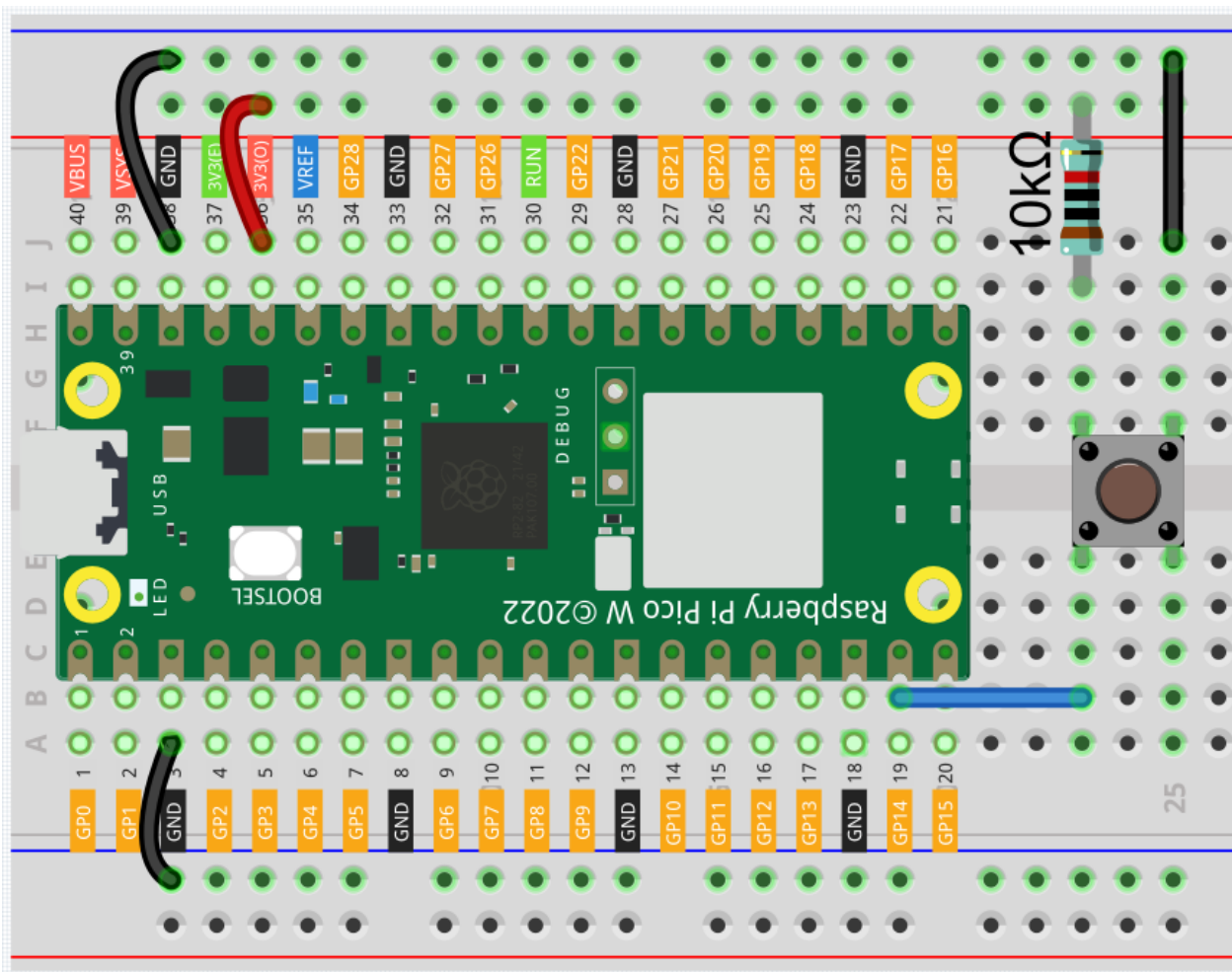
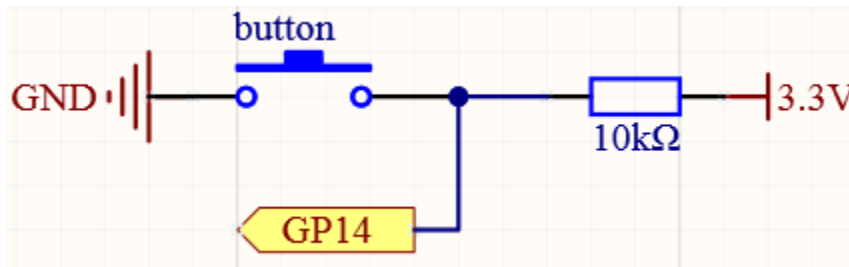
(Fortsetzung der vorherigen Seite)

```
button = machine.Pin(14, machine.Pin.IN)
while True:
    if button.value() == 1:
        print("You pressed the button!")
        utime.sleep(1)
```

Sobald der Code ausgeführt wird, wird „Sie haben die Taste gedrückt!“ in der Shell ausgegeben.

### Pull-Up-Arbeitsmodus

Der nächste Abschnitt behandelt die Verdrahtung und den Code, wenn Sie den Taster im Pull-Up-Modus verwenden.



Der einzige Unterschied, den Sie im Vergleich zum Pull-Down-Modus feststellen werden, ist, dass der 10K-Widerstand mit 3,3 V und die Taste mit GND verbunden ist. Dadurch erhält GP14 ein niedriges Signalelveau, wenn die Taste

gedrückt wird, was das Gegenteil vom Pull-Down-Modus ist. Ändern Sie also diesen Code einfach zu `if button.value() == 0:`.

Weitere Referenzen finden Sie hier:

- [machine.Pin](#)

## 4.12 2.6 Neige es!



Der Kippschalter ist ein Gerät mit zwei Anschlüssen und einer Metallkugel in der Mitte. Wenn der Schalter aufrecht ist, sind die beiden Pins verbunden; wird er geneigt, sind sie getrennt.

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

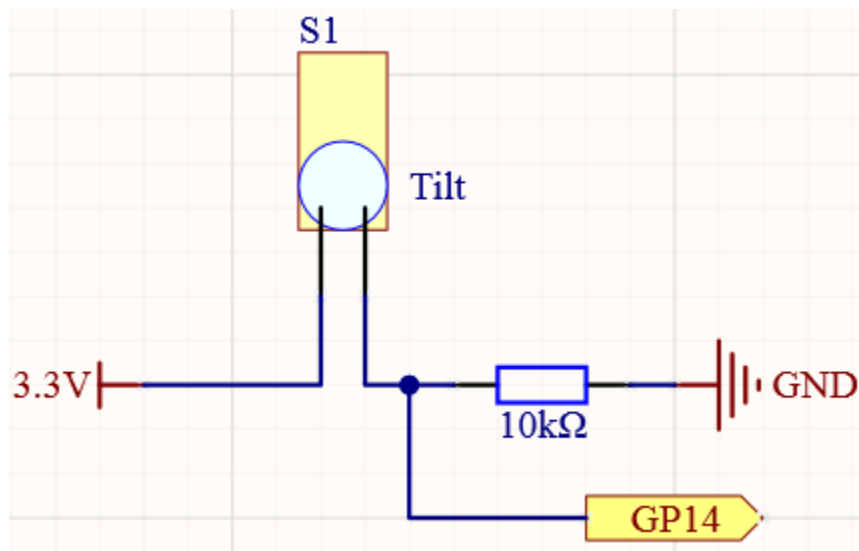
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
Kepler-Kit	450+	

Sie können die Teile auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Neigungsschalter</i>	1	

### Schaltplan

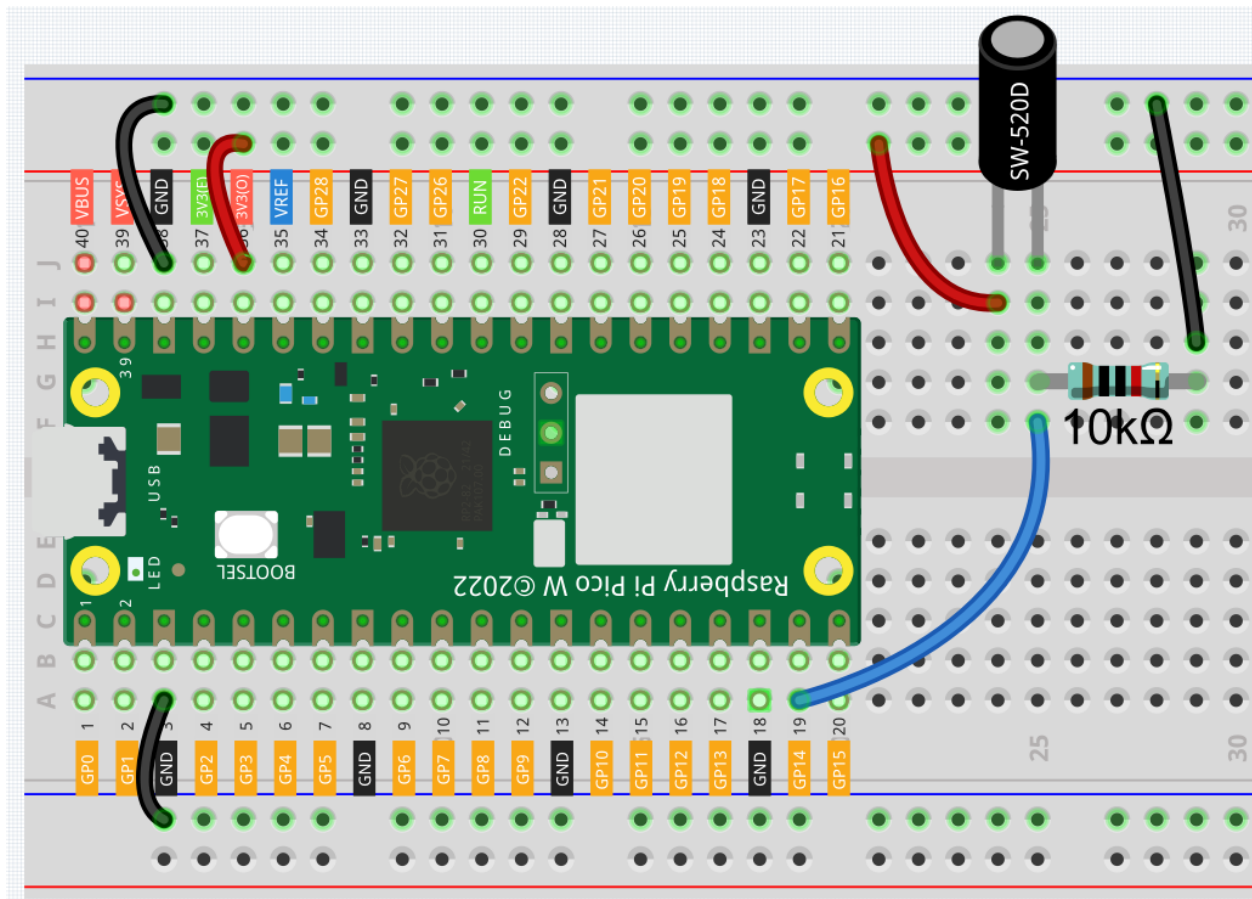


Wenn der Schalter aufrecht steht, wird GP14 hoch; neigen Sie ihn, wird GP14 niedrig.

Der 10K-Widerstand dient dazu, GP14 in einem stabilen Niedrigzustand zu halten, wenn der Kippschalter geneigt ist.

- *Neigungsschalter*

### Verdrahtung



## Code

**Bemerkung:**

- Öffnen Sie die Datei `2.6_tilt_switch.py` im Pfad `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny, und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der unteren rechten Ecke auszuwählen.
- Für detaillierte Anleitungen verweisen wir auf [Code direkt öffnen und ausführen](#).

```
import machine
import utime
button = machine.Pin(14, machine.Pin.IN)
while True:
    if button.value() == 0:
        print("The switch works!")
        utime.sleep(1)
```

Nach dem Ausführen des Programms wird im Shell „The switch works!“ angezeigt, wenn Sie das Breadboard (Kippschalter) neigen.

## 4.13 2.7 Nach Links und Rechts Schalten



Der Schiebeschalter ist ein Gerät mit drei Anschlüssen, wobei der mittlere Anschluss (Pin 2) der gemeinsame Anschluss ist. Wird der Schalter nach links geschaltet, sind die beiden linken Pins miteinander verbunden. Wird er nach rechts geschaltet, sind die beiden rechten Pins verbunden.

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

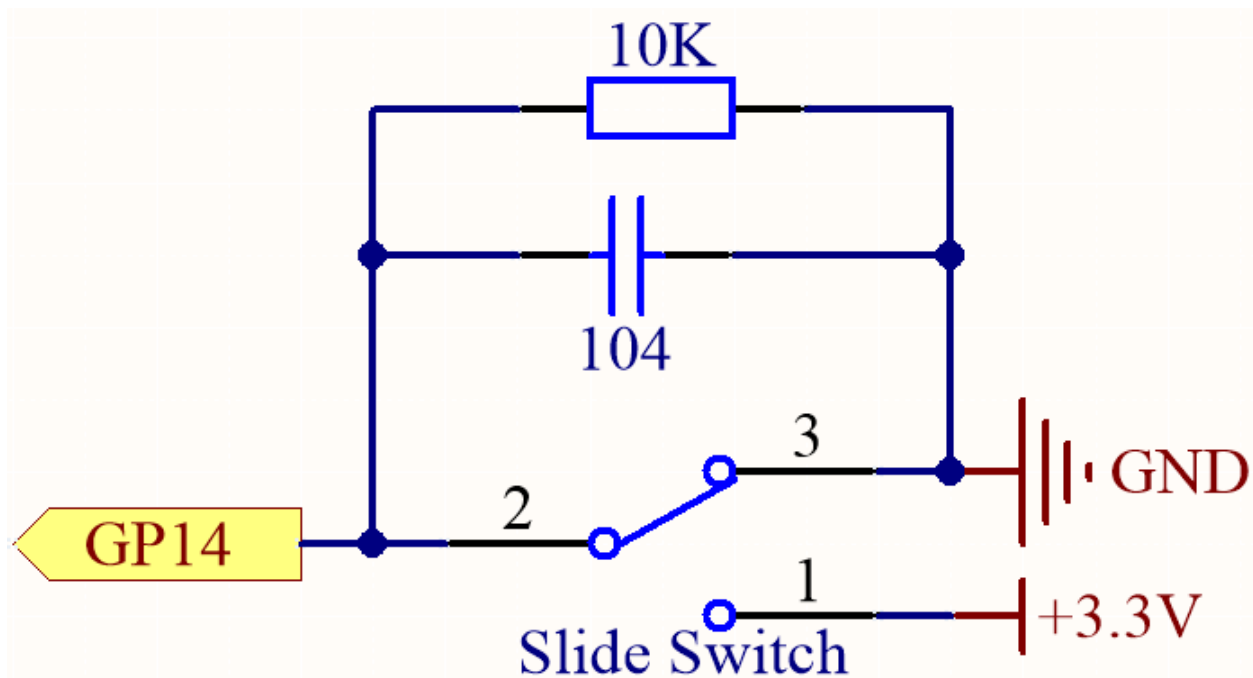
Ein komplettes Kit zu kaufen, ist definitiv praktisch, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
Kepler-Kit	450+	

Sie können die Bauteile auch einzeln über die folgenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Kondensator</i>	1(104)	
7	<i>Schiebeschalter</i>	1	

### Schaltplan



Bei einer Schaltung nach rechts oder links wird GP14 einen unterschiedlichen Pegel erreichen.

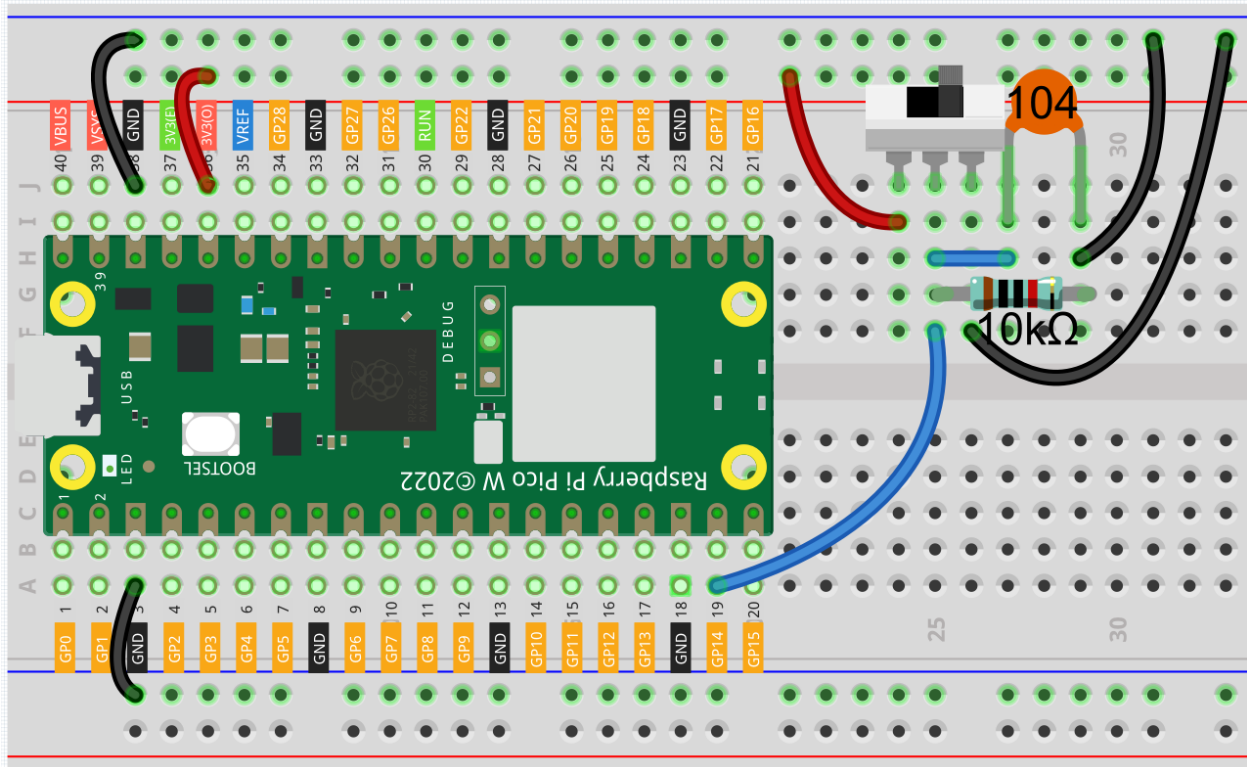
Der 10K-Widerstand dient dazu, GP14 während des Umschaltens auf einem niedrigen Pegel zu halten (nicht ganz nach links und nicht ganz nach rechts geschaltet).

Der 104-Keramikkondensator wird hier eingesetzt, um Rauschen zu eliminieren.

- *Schiebeschalter*
- *Kondensator*

#### Verdrahtung





## Code

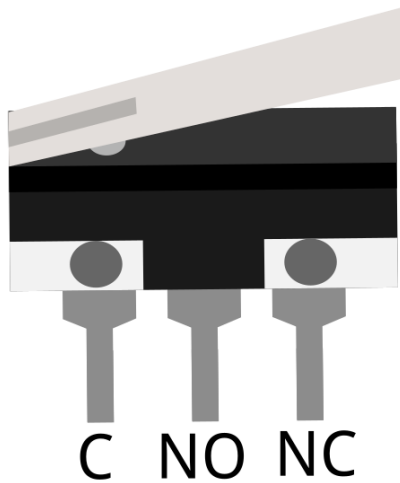
### Bemerkung:

- Öffnen Sie die Datei `2.7_slide_switch.py` im Ordner `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny, und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der unteren rechten Ecke auszuwählen.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
import utime
button = machine.Pin(14, machine.Pin.IN)
while True:
    if button.value() == 0:
        print("The switch works!")
        utime.sleep(1)
```

Nach dem Ausführen des Programms erscheint im Shell die Meldung „The switch works!“, wenn der Schiebeschalter nach rechts geschaltet wird.

## 4.14 2.8 Sanft Drücken



Ein Mikroschalter ist ebenfalls ein Gerät mit 3 Anschlüssen. Die Reihenfolge der Anschlüsse ist C (gemeinsamer Pin), NO (normalerweise offen) und NC (normalerweise geschlossen).

Wenn der Mikroschalter nicht gedrückt ist, sind 1 (C) und 3 (NC) miteinander verbunden. Wird er gedrückt, sind 1 (C) und 2 (NO) miteinander verbunden.

- *Mikroschalter*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

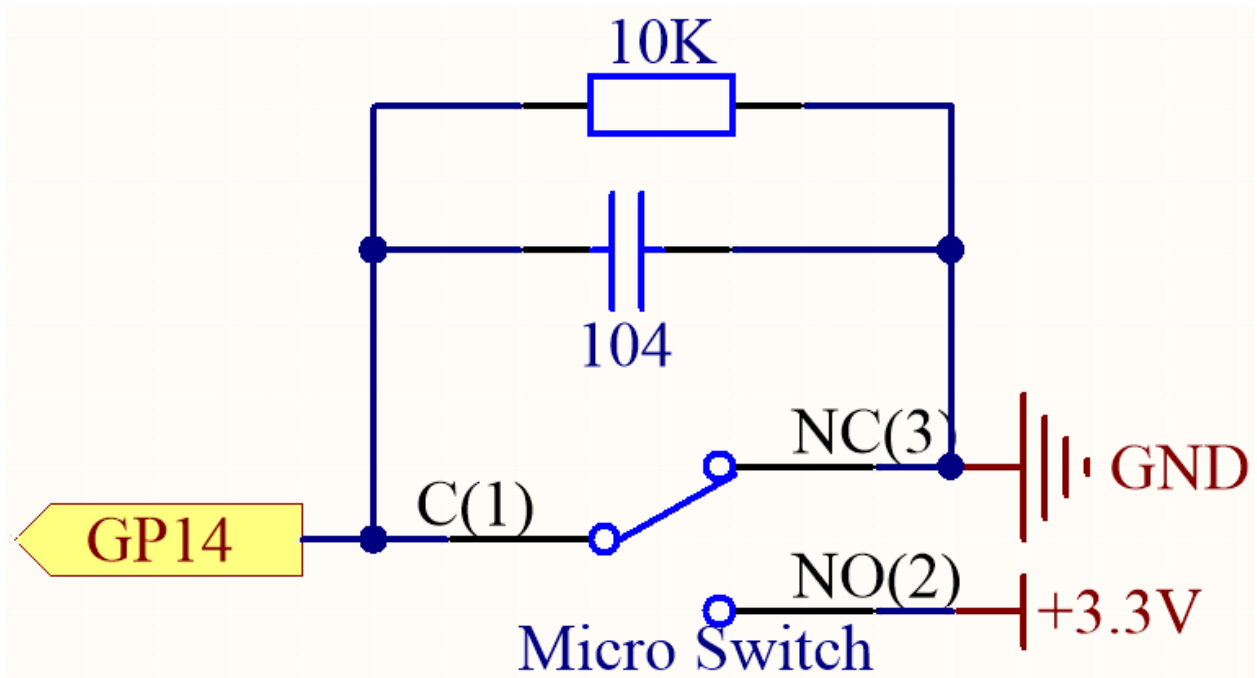
Ein Komplettsset zu kaufen ist definitiv praktisch, hier ist der Link:

Name	ARTIKEL IM SET	LINK
Kepler-Set	450+	

Sie können die einzelnen Teile auch über die untenstehenden Links kaufen.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Kondensator</i>	1(104)	
7	<i>Mikroschalter</i>	1	

### Schaltplan

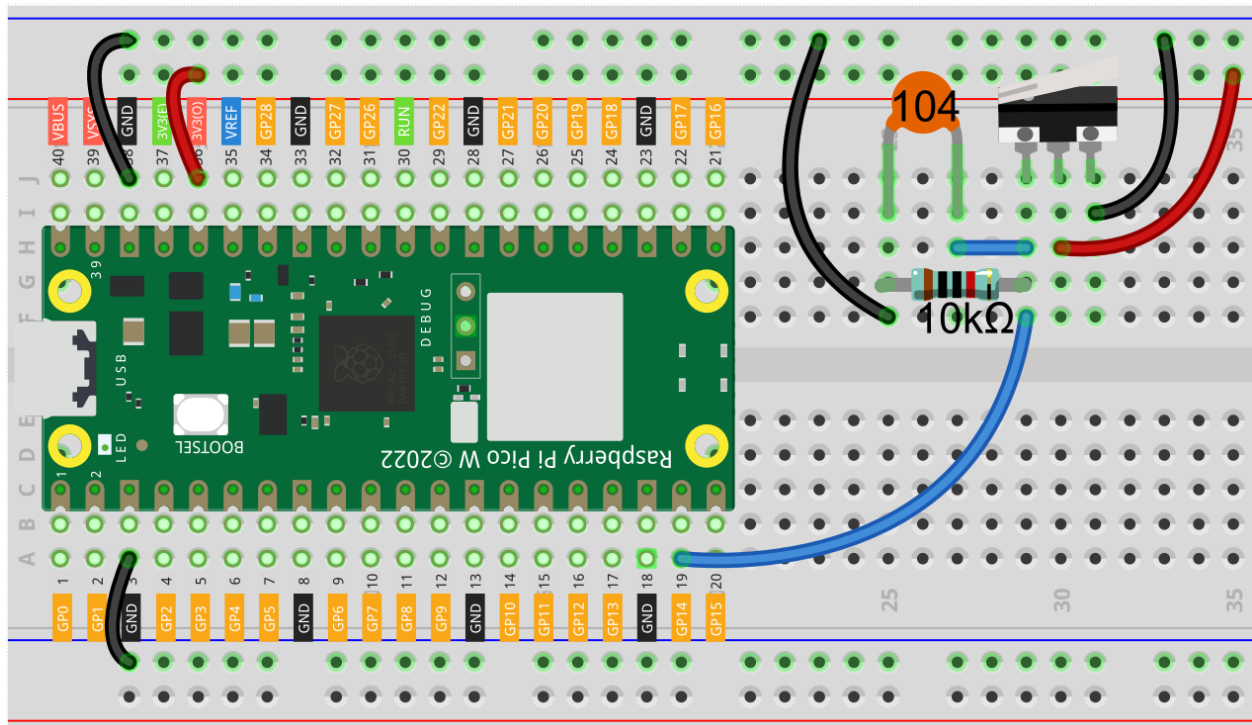


Standardmäßig ist GP14 niedrig und wird hoch, wenn der Schalter gedrückt wird.

Der 10K-Widerstand dient dazu, GP14 während des Drückens niedrig zu halten.

Der 104 Keramikkondensator wird hier verwendet, um Rauschen zu eliminieren.

#### Verdrahtung



#### Code

### Bemerkung:

- Öffnen Sie die Datei `2.8_micro_switch.py` im Pfad `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken Sie auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, im unteren rechten Bereich auf den „MicroPython (Raspberry Pi Pico)“-Interpreter zu klicken.
- Für detaillierte Anleitungen siehe *Code direkt öffnen und ausführen*.

```
import machine
import utime
button = machine.Pin(14, machine.Pin.IN)
while True:
    if button.value() == 1:
        print("The switch works!")
        utime.sleep(1)
```

Nachdem das Programm gelaufen ist, erscheint „Der Schalter funktioniert!“ im Shell-Fenster, wenn Sie den Schiebeschalter nach rechts bewegen.

## 4.15 2.9 Fühle den Magnetismus

Der gebräuchlichste Typ eines Reed-Schalters besteht aus einem Paar magnetisierbarer, flexibler Metallzungen, deren Enden bei geöffnetem Schalter durch eine kleine Lücke getrennt sind.

Ein Magnetfeld, erzeugt entweder durch einen Elektromagneten oder einen Permanentmagneten, führt dazu, dass sich die Zungen anziehen und somit einen elektrischen Stromkreis schließen. Die Federkraft der Zungen bewirkt, dass sie sich trennen und den Stromkreis unterbrechen, sobald das Magnetfeld erlischt.

Ein typisches Anwendungsbeispiel für einen Reed-Schalter ist die Überwachung des Öffnens von Türen oder Fenstern in einer Alarmanlage.

- *Reedschalter*

### Benötigte Komponenten

Für dieses Projekt werden die folgenden Komponenten benötigt.

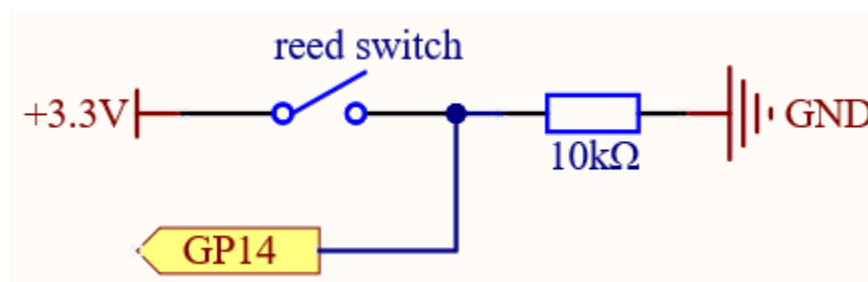
Ein Komplettsatz zu kaufen, ist definitiv praktisch. Hier ist der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Die Teile können auch einzeln über die folgenden Links erworben werden.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Reedschalter</i>	1	

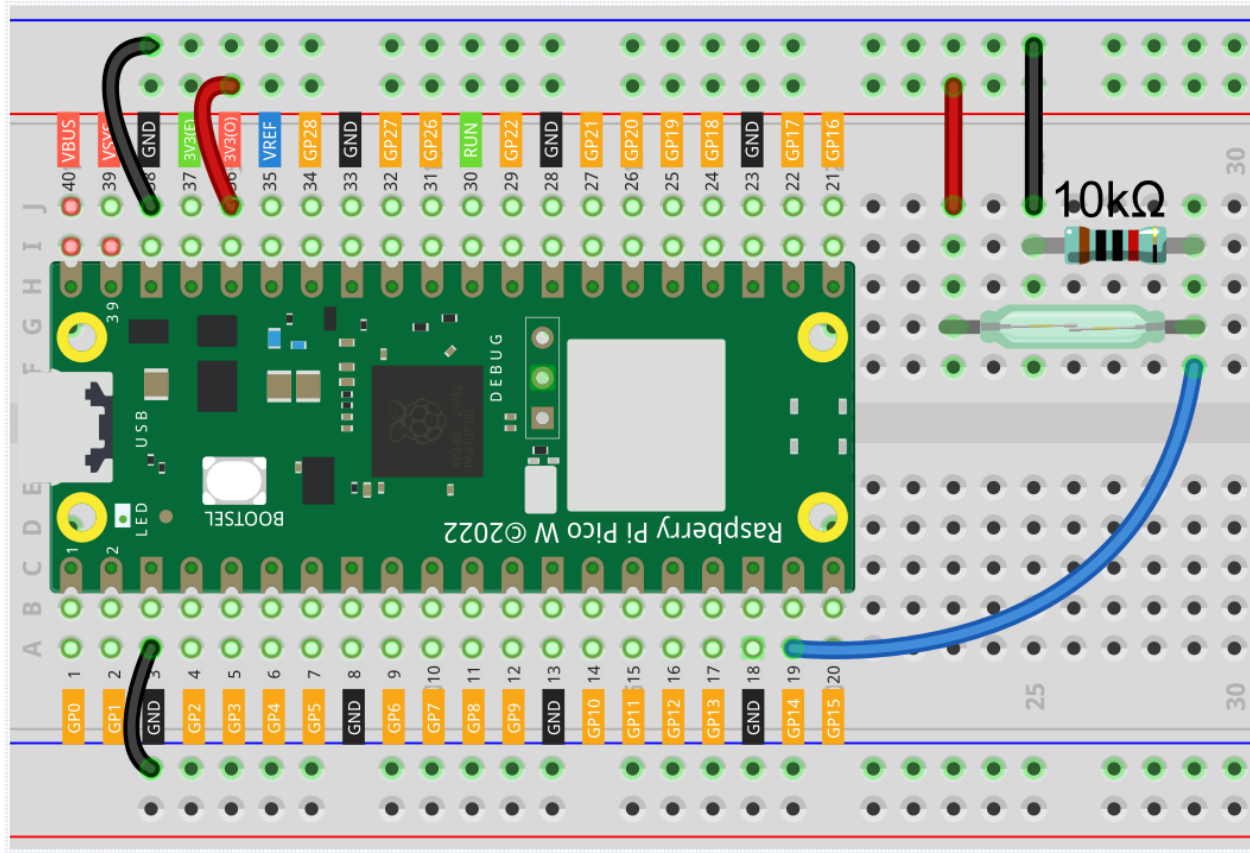
### Schaltplan



Standardmäßig ist GP14 niedrig; der Wert wird hoch, sobald ein Magnet in der Nähe des Reed-Schalters ist.

Der 10K-Widerstand dient dazu, GP14 auf einem konstant niedrigen Level zu halten, wenn kein Magnet in der Nähe ist.

### Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `2.9_feel_the_magnetism.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken Sie auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
- Für ausführliche Anleitungen beachten Sie bitte *Code direkt öffnen und ausführen*.

```
import machine
import utime
reed = machine.Pin(14, machine.Pin.IN)
while True:
    if reed.value() == 1:
        print("There are magnets here!!")
        utime.sleep(1)
```

Läuft der Code, wird GP14 hoch, wenn ein Magnet in der Nähe des Reed-Schalters ist, ansonsten niedrig. Ganz wie der Knopf im Kapitel [2.5 Tastenwert auslesen](#).

### Mehr erfahren

Dieses Mal haben wir eine flexible Art der Schalterbenutzung erprobt: Unterbrechungsanfragen, auch IRQs genannt.

Stellen Sie sich zum Beispiel vor, Sie lesen Seite für Seite ein Buch, als wäre ein Programm einen Thread am Ausführen. Plötzlich kommt jemand und stellt Ihnen eine Frage, unterbricht also Ihre Lektüre. Diese Person führt die Unterbrechungsanfrage aus: Sie sollen kurz stoppen, die Frage beantworten und dann Ihre Lektüre fortsetzen.

Die Unterbrechungsanfragen in MicroPython funktionieren ähnlich, sie erlauben bestimmten Aktionen, das Hauptprogramm zu unterbrechen.

---

**Bemerkung:**

- Öffnen Sie die Datei `2.9_feel_the_magnetism_irq.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken Sie auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
  - Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
  - Für ausführliche Anleitungen beachten Sie bitte *Code direkt öffnen und ausführen*.
- 

```
import machine
import utime

reed_switch = machine.Pin(14, machine.Pin.IN)

def detected(pin):
    print("Magnet!")

reed_switch.irq(trigger=machine.Pin.IRQ_RISING, handler=detected)
```

Zunächst wird eine Callback-Funktion `detected(pin)` definiert, die als Unterbrechungsbehandler dient. Sie wird ausgeführt, wenn eine Unterbrechungsanfrage ausgelöst wird. Dann wird im Hauptprogramm eine Unterbrechungsanfrage eingerichtet, die aus zwei Teilen besteht: dem `trigger` und dem `handler`.

Im Programm ist `trigger IRQ_RISING`, was bedeutet, dass der Wert des Pins von niedrig auf hoch wechselt (also beim Tastendruck).

`handler` ist `detected`, die vorher definierte Callback-Funktion.

- `machine.Pin.irq` - *Micropython Docs* <<https://docs.micropython.org/en/latest/library/machine.Pin.html#machine.Pin.irq>>

## 4.16 2.10 Bewegungserkennung beim Menschen

Ein passiver Infrarotsensor (PIR-Sensor) ist ein weit verbreiteter Sensor, der Infrarotstrahlung (IR) von Objekten in seinem Sichtfeld misst. Einfach ausgedrückt: Er nimmt die von einem Körper abgestrahlte Infrarotstrahlung auf und erkennt so die Bewegung von Menschen und anderen Tieren. Konkret informiert er die Hauptsteuerplatine darüber, dass jemand den Raum betreten hat.

*PIR-Bewegungssensormodul*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

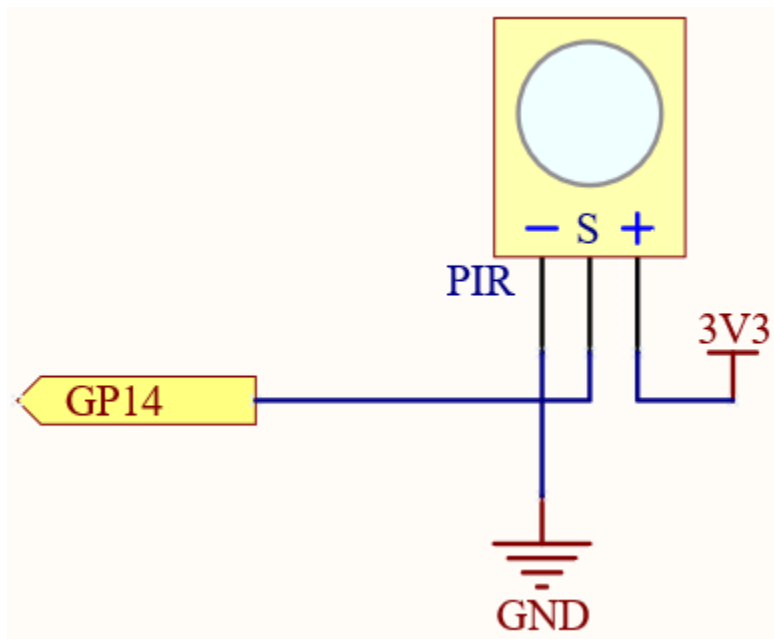
Ein Komplettsset zu kaufen ist sicherlich bequem, hier ist der Link:

Bezeichnung	ELEMENTE IM SET	LINK
Kepler-Set	450+	

Sie können die Teile auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>PIR-Bewegungssensormodul</i>	1	

### Schaltplan



Wenn das PIR-Modul eine vorbeigehende Person erkennt, wird GP14 auf „hoch“ gesetzt, andernfalls bleibt er „niedrig“.

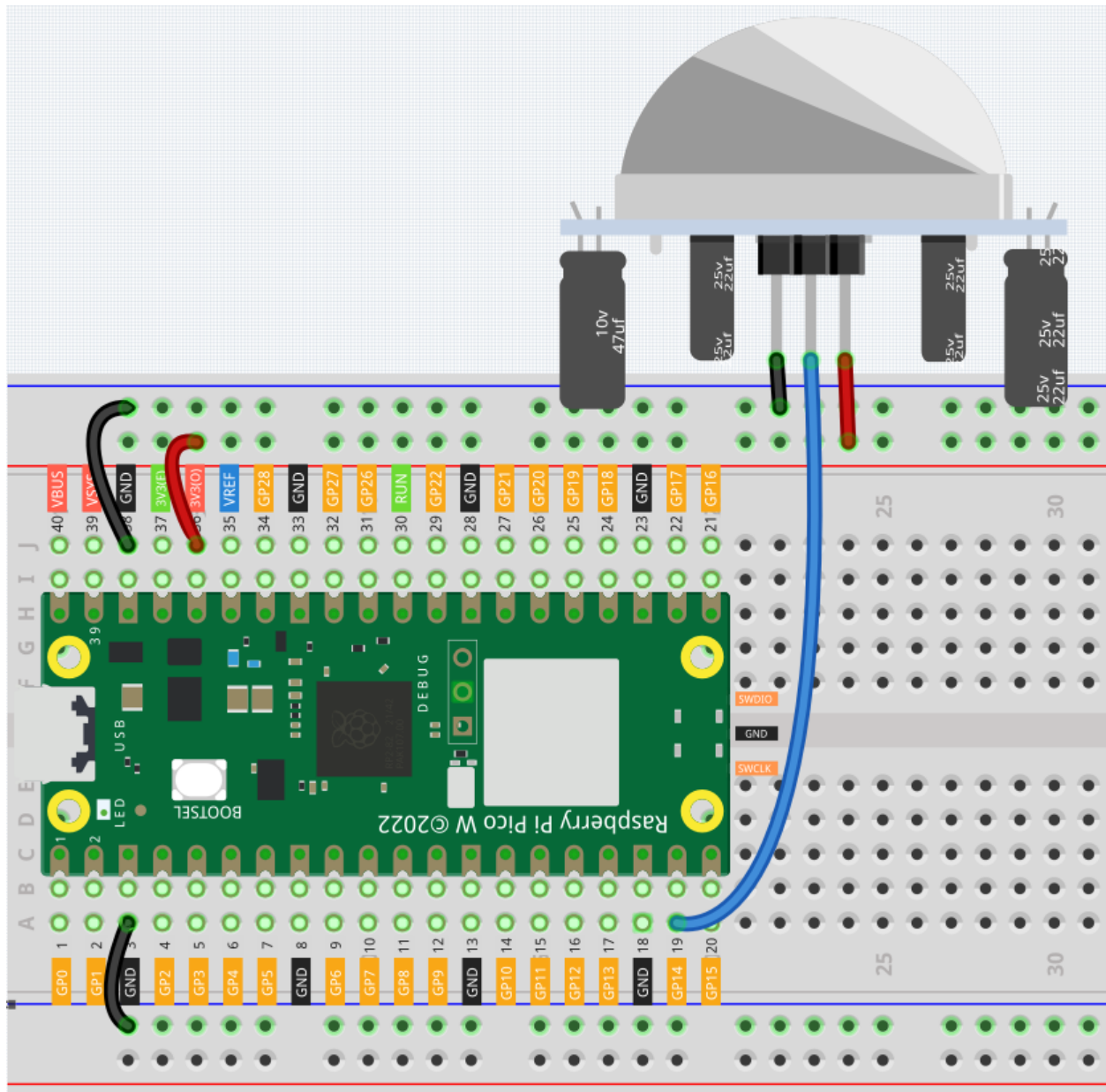
**Bemerkung:** Das PIR-Modul verfügt über zwei Potentiometer: eines zur Einstellung der Empfindlichkeit, das andere zur Anpassung der Erfassungsreichweite. Für optimale Ergebnisse sollten beide Potentiometer ganz nach links gedreht werden.





---

## Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `2.10_detect_human_movement.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Anschließend klicken Sie auf „Aktuelles Skript ausführen“ oder drücken einfach F5.
- Vergessen Sie nicht, den Interpreter „MicroPython (Raspberry Pi Pico)“ in der unteren rechten Ecke auszuwählen.
- Detaillierte Anleitungen finden Sie unter [Code direkt öffnen und ausführen](#).

```
import machine
import utime

pir_sensor = machine.Pin(14, machine.Pin.IN)

def motion_detected(pin):
    print("Somebody here!")

pir_sensor.irq(trigger=machine.Pin.IRQ_RISING, handler=motion_detected)
```

Nach dem Ausführen des Programms wird in der Shell „Somebody here!“ ausgegeben, wenn das PIR-Modul eine nahe Person erkennt.

### Weitere Informationen

Das PIR-Modul ist sehr empfindlich. Um es an die Einsatzumgebung anzupassen, sind Einstellungen erforderlich. Richten Sie die Seite mit den beiden Potentiometern auf sich aus und drehen Sie beide Potentiometer ganz nach links. Setzen Sie die Jumperkappe auf den Pin mit L und den mittleren Pin.

### Bemerkung:

- Öffnen Sie die Datei `2.10_pir_adjustment.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Anschließend klicken Sie auf „Aktuelles Skript ausführen“ oder drücken einfach F5.
- Vergessen Sie nicht, den Interpreter „MicroPython (Raspberry Pi Pico)“ in der unteren rechten Ecke auszuwählen.
- Detaillierte Anleitungen finden Sie unter [Code direkt öffnen und ausführen](#).

```
import machine
import utime

pir_sensor = machine.Pin(14, machine.Pin.IN)

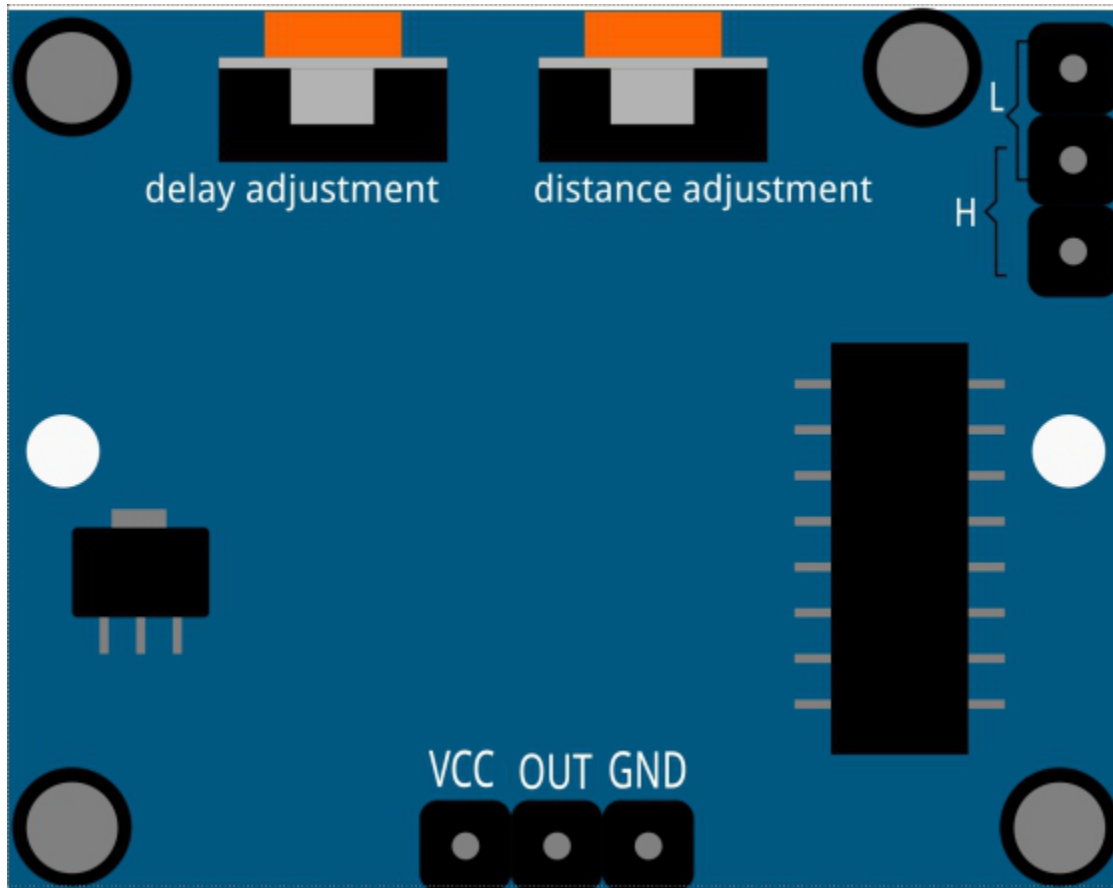
global timer_delay
timer_delay = utime.ticks_ms()
print("start")

def pir_in_high_level(pin):
    global timer_delay
    pir_sensor.irq(trigger=machine.Pin.IRQ_FALLING, handler=pir_in_low_level)
    intervals = utime.ticks_diff(utime.ticks_ms(), timer_delay)
    timer_delay = utime.ticks_ms()
    print("the dormancy duration is " + str(intervals) + "ms")

def pir_in_low_level(pin):
    global timer_delay
    pir_sensor.irq(trigger=machine.Pin.IRQ_RISING, handler=pir_in_high_level)
    intervals2 = utime.ticks_diff(utime.ticks_ms(), timer_delay)
    timer_delay = utime.ticks_ms()
    print("the duration of work is " + str(intervals2) + "ms")

pir_sensor.irq(trigger=machine.Pin.IRQ_RISING, handler=pir_in_high_level)
```

Lassen wir uns die Anpassungsmethode anhand der experimentellen Ergebnisse analysieren.



### 1. Auslösemodus

Werfen wir einen Blick auf die Pins mit der Jumperkappe in der Ecke. Sie ermöglichen dem PIR, in den wiederholbaren oder nicht wiederholbaren Auslösemodus zu wechseln.

Aktuell ist unsere Jumperkappe mit dem mittleren Pin und dem L-Pin verbunden, was den PIR in den nicht wiederholbaren Auslösemodus versetzt. In diesem Modus sendet der PIR bei Erkennung einer Bewegung ein Hochpegelsignal für etwa 2,8 Sekunden an die Hauptsteuerplatine. Anhand der ausgegebenen Daten können wir erkennen, dass die Arbeitsdauer stets rund 2800 ms beträgt.

Als Nächstes ändern wir die Position der unteren Jumperkappe und verbinden sie mit dem mittleren Pin und dem H-Pin, um den PIR in den wiederholbaren Auslösemodus zu versetzen. In diesem Modus sendet der PIR ein Hochpegelsignal an die Hauptsteuerplatine, solange innerhalb des Erfassungsbereichs eine Bewegung stattfindet (beachten Sie, dass es sich um eine Bewegung handelt, nicht um ein statisches Verharren vor dem Sensor). In den ausgegebenen Daten ist die Arbeitsdauer ein variabler Wert.

### 2. Verzögerungsanpassung

Das linke Potentiometer dient zur Einstellung des Intervalls zwischen zwei Arbeitszyklen.

Derzeit haben wir es ganz gegen den Uhrzeigersinn gedreht, was dazu führt, dass der PIR nach Beendigung der Hochpegel-Arbeit eine Ruhezeit von etwa 5 Sekunden einlegen muss. In dieser Zeit erfasst der PIR keine Infrarotstrahlung im Zielbereich mehr. Anhand der ausgegebenen Daten können wir erkennen, dass die Ruhezeit immer mindestens 5000 ms beträgt.

Drehen wir das Potentiometer im Uhrzeigersinn, verlängert sich auch die Ruhezeit. Wenn es ganz im Uhrzeigersinn gedreht wird, kann die Ruhezeit bis zu 300 Sekunden betragen.

### 3. Entfernungsanpassung

Das mittlere Potentiometer dient zur Einstellung des Erfassungsbereichs des PIR.

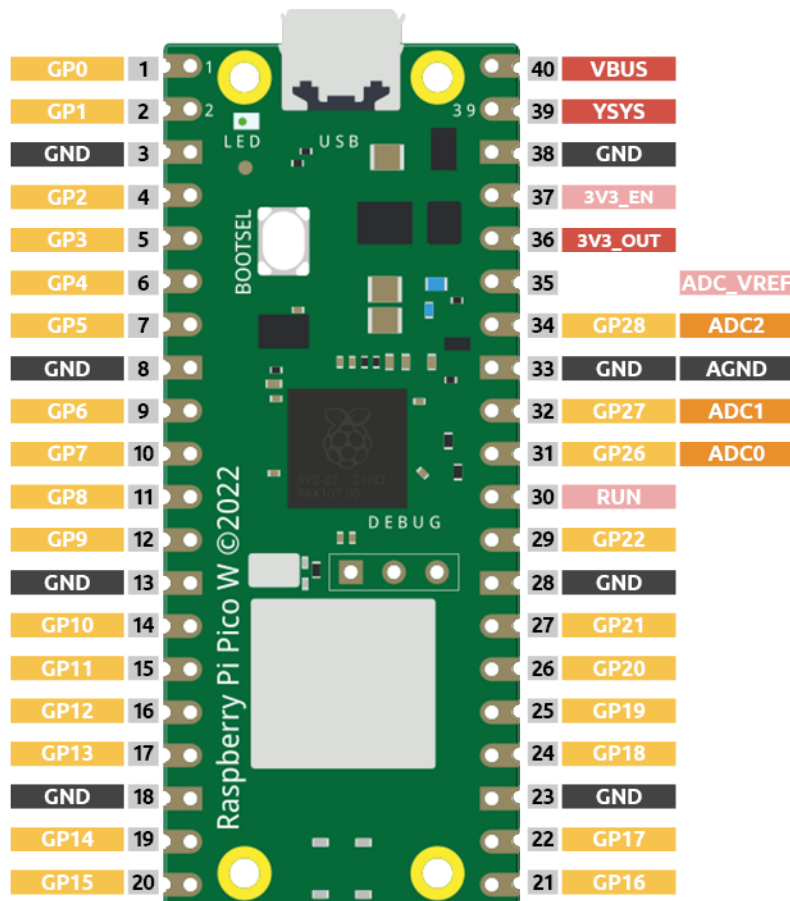
Drehen Sie den Knopf des Entfernungsanpassungspotentiometers **im Uhrzeigersinn**, um den Erfassungsbereich zu erweitern. Der maximale Erfassungsbereich beträgt etwa 0 bis 7 Meter. Wird es **gegen den Uhrzeigersinn** gedreht, verringert sich der Erfassungsbereich, und der minimale Erfassungsbereich beträgt etwa 0 bis 3 Meter.

## 4.17 2.11 Den Drehregler betätigen

In den vorherigen Projekten haben wir den digitalen Eingang des Pico W genutzt. Beispielsweise kann ein Knopf den Pin von einem niedrigen (aus) zu einem hohen Pegel (ein) ändern. Das ist ein binärer Arbeitszustand.

Der Pico W ist jedoch auch in der Lage, eine andere Art von Eingangssignal zu empfangen: den analogen Eingang. Dieser kann sich in einem Zustand von vollständig geschlossen bis vollständig geöffnet befinden und bietet ein Spektrum an möglichen Werten. Der analoge Eingang erlaubt es dem Mikrocontroller, die Lichtintensität, Schallintensität, Temperatur, Feuchtigkeit usw. der physischen Welt zu erfassen.

Normalerweise benötigt ein Mikrocontroller zusätzliche Hardware für den analogen Eingang - den Analog-Digital-Umsetzer (ADC). Der Pico W verfügt jedoch bereits über einen integrierten ADC, den wir direkt nutzen können.



Der Pico W besitzt drei GPIO-Pins, die für analogen Eingang genutzt werden können: GP26, GP27, GP28, also analoge Kanäle 0, 1 und 2. Zudem gibt es einen vierten analogen Kanal, der mit dem eingebauten Temperatursensor verbunden ist und hier nicht weiter erläutert wird.

In diesem Projekt versuchen wir, den analogen Wert eines Potentiometers auszulesen.

- *Potentiometer*

### Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

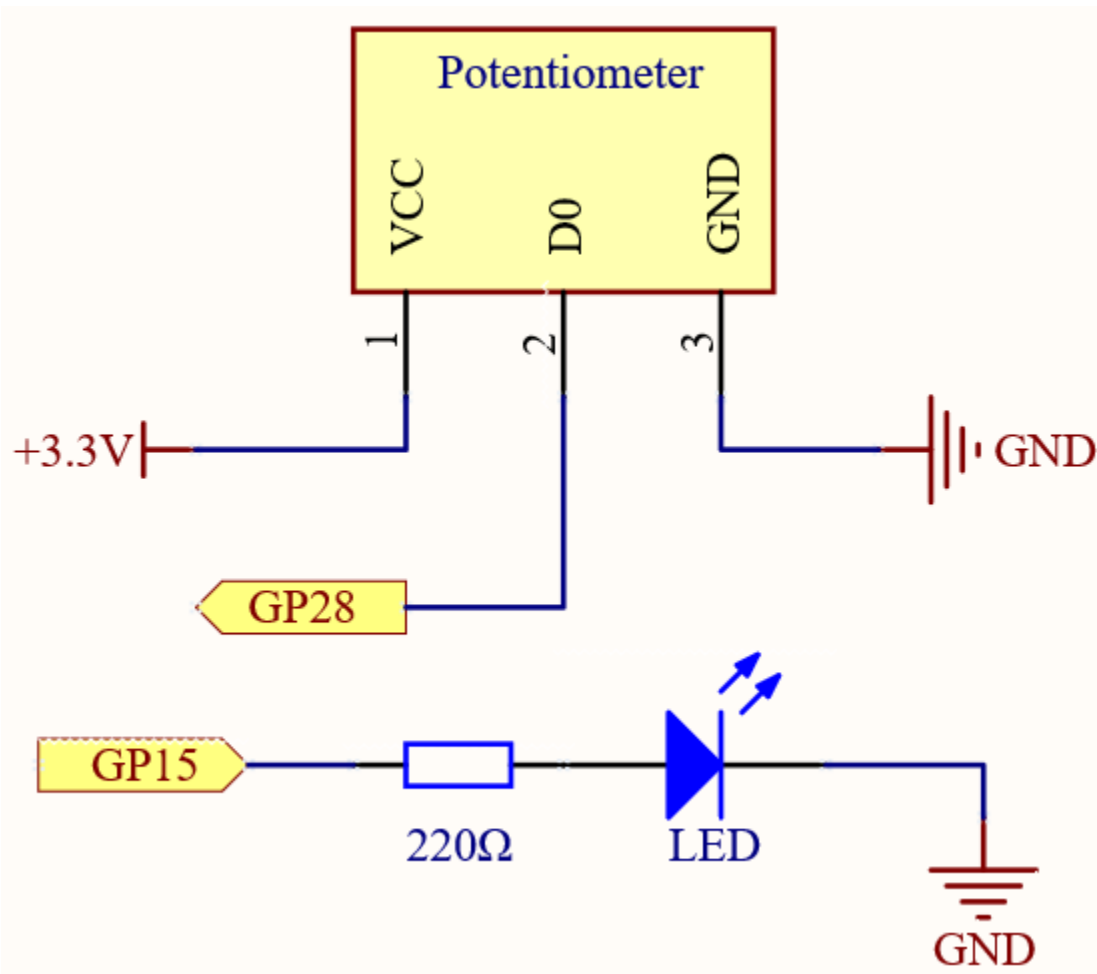
Es ist sicherlich praktisch, ein komplettes Kit zu kaufen, hier ist der Link:

Name	KOMPONENTEN IN DIESEM KIT	LINK
Kepler Kit	450+	

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

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1 (220)	
6	<i>LED</i>	1	
7	<i>Potentiometer</i>	1	

### Schaltplan



Das Potentiometer ist ein analoges Bauteil, das in zwei verschiedene Richtungen gedreht werden kann.

Schließen Sie den mittleren Pin des Potentiometers an den analogen Pin GP28 an. Der Raspberry Pi Pico W verfügt über einen mehrkanaligen, 16-bit Analog-Digital-Umsetzer. Das bedeutet, dass die Eingangsspannung zwischen 0 und der Betriebsspannung (3,3V) auf einen Ganzzahlwert zwischen 0 und 65535 abgebildet wird. Der Wertebereich von GP28 reicht also von 0 bis 65535.

Die Berechnungsformel lautet wie folgt:

$$(V_p/3.3V) \times 65535 = A_p$$

Programmieren Sie dann den Wert von GP28 (Potentiometer) als PWM-Wert von GP15 (LED). Sie werden feststellen, dass die Helligkeit der LED sich gleichzeitig ändert, wenn Sie das Potentiometer drehen.

#### Verdrahtung







**Funktionsweise**

```
potentiometer = machine.ADC(28)
```

Zugriff auf den ADC, der mit einer durch die ID identifizierten Quelle verbunden ist. In diesem Beispiel handelt es sich um GP28.

```
potentiometer.read_u16()
```

Führt eine analoge Messung durch und gibt einen Ganzzahlwert im Bereich von 0 bis 65535 zurück. Der Rückgabewert stellt die rohe Messung dar, die vom ADC erfasst und so skaliert wurde, dass der Mindestwert 0 und der Höchstwert 65535 beträgt.

- [machine.ADC - MicroPython Dokumentation](#)

## 4.18 2.12 Das Licht spüren

Der Fotowiderstand ist ein typisches Bauelement für analoge Eingänge und funktioniert sehr ähnlich wie ein Potentiometer. Sein Widerstandswert ist abhängig von der Lichtintensität: Je stärker das Licht, desto geringer der Widerstandswert und umgekehrt.

- *Fotowiderstand*

**Benötigte Komponenten**

Für dieses Projekt benötigen wir die folgenden Bauteile.

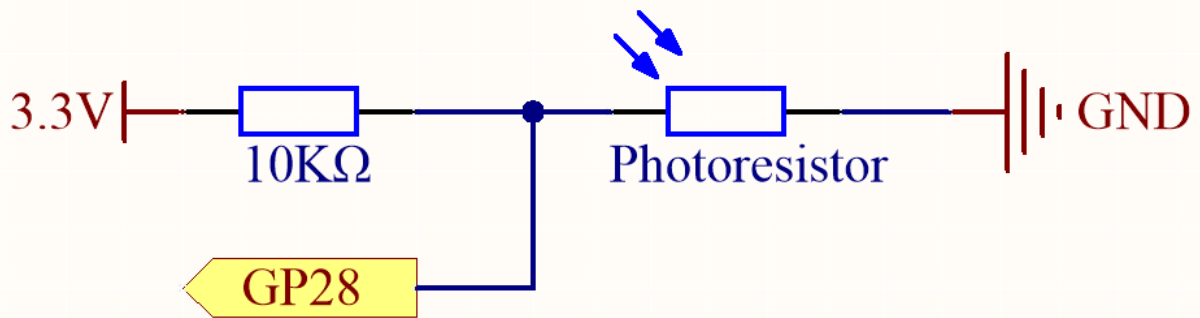
Es ist definitiv praktisch, ein komplettes Set zu erwerben. Hier ist der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler-Set	450+	

Sie können die Komponenten auch einzeln über die folgenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i><a href="#">Raspberry Pi Pico W</a></i>	1	
2	Micro-USB-Kabel	1	
3	<i><a href="#">Steckbrett</a></i>	1	
4	<i><a href="#">Jumperkabel</a></i>	Mehrere	
5	<i><a href="#">Widerstand</a></i>	1 (10K)	
6	<i><a href="#">Fotowiderstand</a></i>	1	

**Schaltplan**



In dieser Schaltung sind der 10K-Widerstand und der Fotowiderstand in Reihe geschaltet; der durch sie fließende Strom ist identisch. Der 10K-Widerstand dient als Schutz, und GP28 liest den umgewandelten Spannungswert des Fotowiderstands.

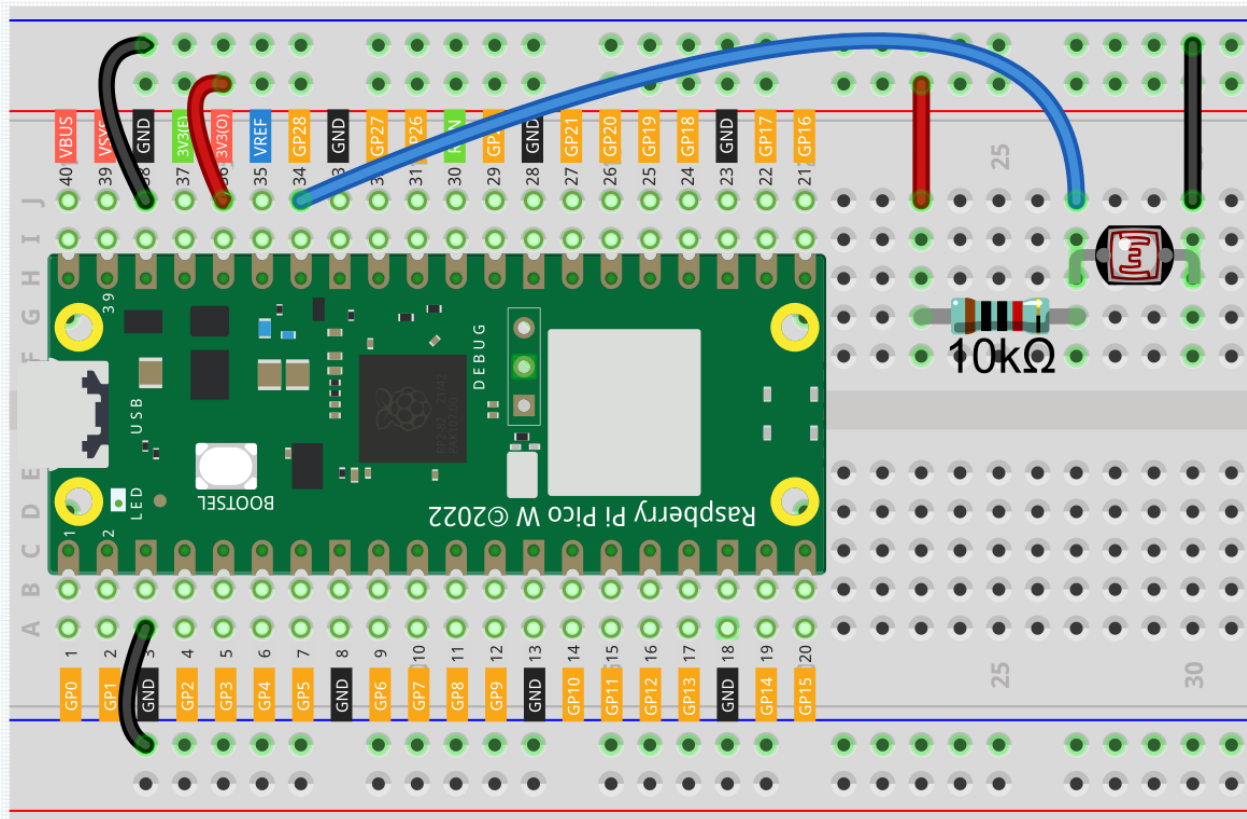
Wenn das Licht intensiver wird, sinkt der Widerstand des Fotowiderstands, und somit auch seine Spannung. Daraufhin wird auch der Wert von GP28 niedriger. Sollte das Licht ausreichend stark sein, nähert sich der Widerstand des Fotowiderstands dem Wert 0 an, und der Wert von GP28 wird ebenfalls nahe 0 liegen. In diesem Fall spielt der 10K-Widerstand eine schützende Rolle, um einen Kurzschluss zwischen 3,3V und GND zu vermeiden.

Platziert man den Fotowiderstand in einer dunklen Umgebung, wird der Wert von GP28 ansteigen. In völliger Dunkelheit wäre der Widerstand des Fotowiderstands unendlich, und seine Spannung wäre nahe 3,3V (der 10K-Widerstand ist vernachlässigbar), und der Wert von GP28 würde sich dem Maximalwert von 65535 annähern.

Die Berechnungsformel lautet wie folgt:

$$(V_p/3.3V) \times 65535 = A_p$$

### Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `2.12_feel_the_light.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und führen Sie ihn mit „Aktuelles Skript ausführen“ oder einfach mit F5 aus.
- Vergewissern Sie sich, dass der Interpreter „MicroPython (Raspberry Pi Pico)“ in der unteren rechten Ecke ausgewählt ist.
- Für ausführliche Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
import utime

photoresistor = machine.ADC(28)

while True:
    light_value = photoresistor.read_u16()
    print(light_value)
    utime.sleep_ms(10)
```

Nachdem das Programm läuft, werden die Werte des Fotowiderstands in der Shell ausgegeben. Man kann die Werte verändern, indem man eine Taschenlampe darauf richtet oder den Fotowiderstand mit der Hand abdeckt.

## 4.19 2.13 Thermometer

Ein Thermometer ist ein Gerät, das die Temperatur oder einen Temperaturgradienten misst (das Maß für die Wärme oder Kälte eines Objekts). Ein Thermometer hat zwei wichtige Elemente: (1) einen Temperatursensor (z.B. die Birne eines Quecksilber-Thermometers oder den pyrometrischen Sensor in einem Infrarot-Thermometer), bei dem eine Veränderung mit einer Temperaturänderung eintritt; und (2) eine Möglichkeit, diese Änderung in einen numerischen Wert umzuwandeln (z.B. die sichtbare Skala, die auf einem Quecksilber-Thermometer markiert ist, oder die digitale Anzeige bei einem Infrarot-Modell). Thermometer werden in Technik und Industrie zur Prozessüberwachung, in der Meteorologie, in der Medizin und in der wissenschaftlichen Forschung weit verbreitet verwendet.

Ein Thermistor ist eine Art Temperatursensor, dessen Widerstand stark temperaturabhängig ist, und es gibt zwei Typen: Negative Temperaturkoeffizient (NTC) und Positiver Temperaturkoeffizient (PTC), auch bekannt als NTC und PTC. Der Widerstand des PTC-Thermistors steigt mit der Temperatur, während der Zustand des NTC dem des PTC entgegengesetzt ist.

In diesem Experiment verwenden wir einen **NTC-Thermistor**, um ein Thermometer herzustellen.

- *Thermistor*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

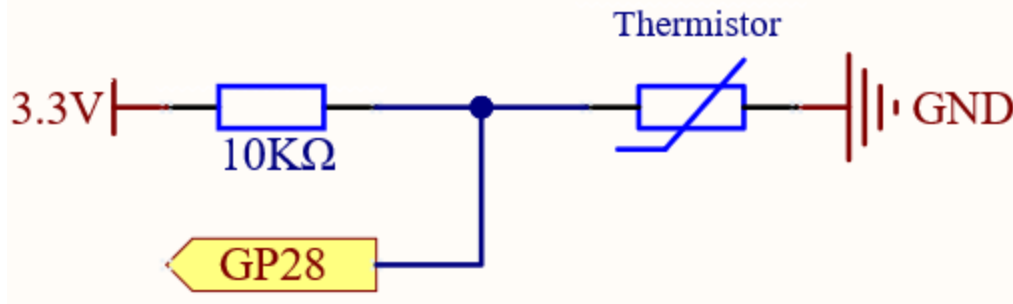
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
Kepler Kit	450+	

Sie können diese auch einzeln über die untenstehenden Links kaufen.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro USB Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Thermistor</i>	1	

### Schaltplan



In dieser Schaltung sind der 10K Widerstand und der Thermistor in Reihe geschaltet, und der durch sie fließende Strom ist derselbe. Der 10K Widerstand dient als Schutz, und der GP28 liest den Wert nach der Spannungsumwandlung des Thermistors.

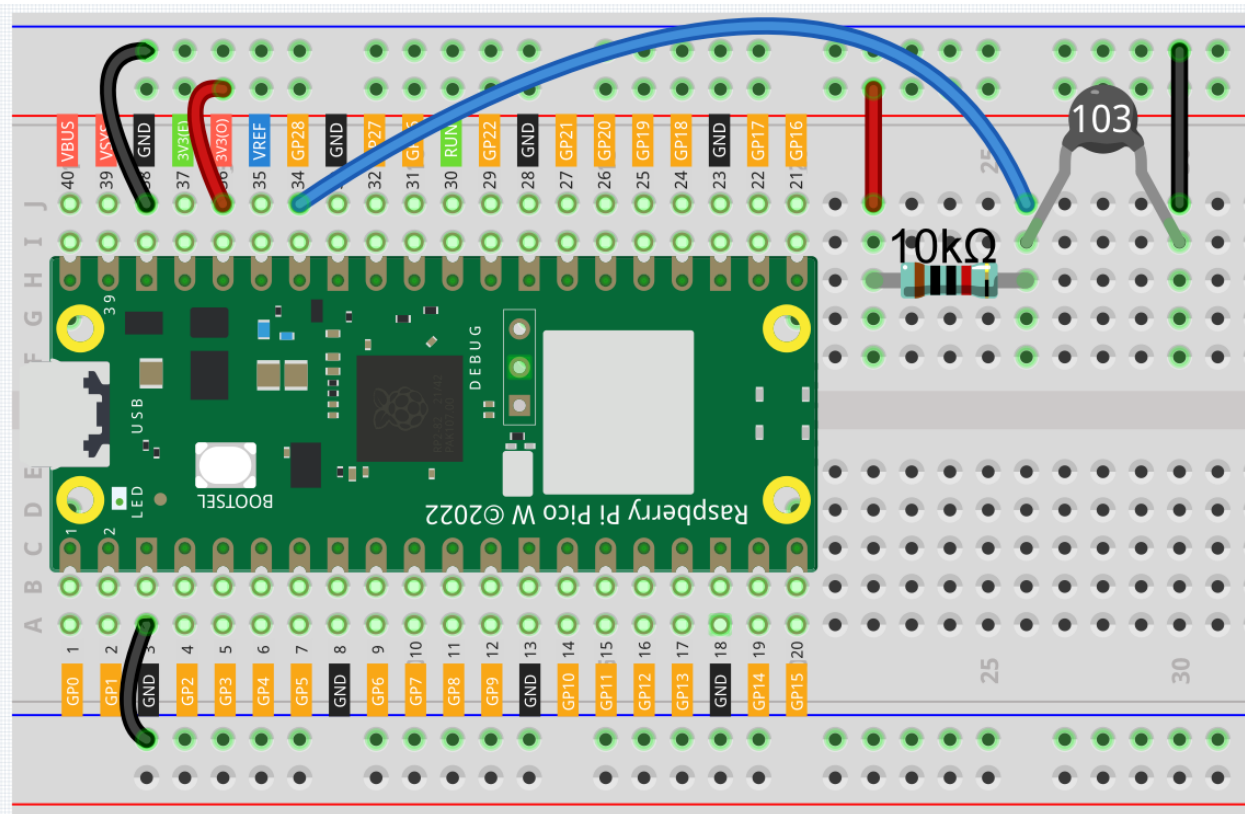
Wenn die Temperatur steigt, verringert sich der Widerstandswert des NTC-Thermistors, dann sinkt seine Spannung, sodass der Wert von GP28 abnimmt; Wenn die Temperatur hoch genug ist, wird der Widerstand des Thermistors nahezu 0 sein, und der Wert von GP28 wird nahezu 0 sein. In dieser Zeit spielt der 10K Widerstand eine schützende Rolle, sodass 3,3V und GND nicht miteinander verbunden sind, was zu einem Kurzschluss führt.

Wenn die Temperatur sinkt, wird der Wert von GP28 steigen. Wenn die Temperatur niedrig genug ist, wird der Widerstand des Thermistors unendlich sein, und seine Spannung wird nahe 3,3V liegen (der 10K Widerstand ist vernachlässigbar), und der Wert von GP28 wird nahe dem Maximalwert von 65535 sein.

Die Berechnungsformel ist unten dargestellt.

$$(V_p/3.3V) \times 65535 = A_p$$

### Verdrahtung



---

### Bemerkung:

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

### Code

---

### Bemerkung:

- Öffnen Sie die Datei `2.13_thermometer.py` unter dem Pfad `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny, dann klicken Sie auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5, um es auszuführen.
  - Vergessen Sie nicht, in der rechten unteren Ecke den „MicroPython (Raspberry Pi Pico)“-Interpreter zu wählen.
  - Detaillierte Anleitungen finden Sie unter [Code direkt öffnen und ausführen](#).
- 

```
import machine
import utime
import math

thermistor = machine.ADC(28)

while True:
    temperature_value = thermistor.read_u16()
    Vr = 3.3 * float(temperature_value) / 65535
    Rt = 10000 * Vr / (3.3 - Vr)
    temp = 1/(((math.log(Rt / 10000)) / 3950) + (1 / (273.15+25)))
    Cel = temp - 273.15
    Fah = Cel * 1.8 + 32
    print ('Celsius: %.2f C   Fahrenheit: %.2f F' % (Cel, Fah))
    utime.sleep_ms(200)
```

Nachdem das Programm ausgeführt wurde, gibt die Shell die Temperaturen in Celsius und Fahrenheit aus.

### Wie funktioniert es?

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

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

Die Temperatur in Celsius oder Fahrenheit wird mittels Programmierung ausgegeben.

```
import math
```

Hierbei handelt es sich um eine numerische Bibliothek, die eine Reihe von Funktionen zur Berechnung gängiger mathematischer Operationen und Transformationen deklariert.

- `math`

```
temperature_value = thermistor.read_u16()
```

Diese Funktion wird verwendet, um den Wert des Thermistors auszulesen.

```
Vr = 3.3 * float(temperature_value) / 65535
Rt = 10000 * Vr / (3.3 - Vr)
temp = 1/(((math.log(Rt / 10000)) / 3950) + (1 / (273.15+25)))
Cel = temp - 273.15
Fah = Cel * 1.8 + 32
print ('Celsius: %.2f C Fahrenheit: %.2f F' % (Cel, Fah))
utime.sleep_ms(200)
```

Diese Funktion wird verwendet, um den Wert des Thermistors auszulesen.

```
Vr = 3.3 * float(temperature_value) / 65535
Rt = 10000 * Vr / (3.3 - Vr)
```

In den beiden obigen Codezeilen wird zuerst die Spannung anhand des gelesenen analogen Wertes berechnet und anschließend Rt (der Widerstand des Thermistors) ermittelt.

```
temp = 1/(((math.log(Rt / 10000)) / 3950) + (1 / (273.15+25)))
```

**Bemerkung:** Hier ist die Beziehung zwischen Widerstand und Temperatur:

**RT = RN expB(1/TK – 1/TN)**

- RT ist der Widerstand des NTC-Thermistors bei einer Temperatur von TK.
- RN ist der Widerstand des NTC-Thermistors bei der Nenntemperatur TN. Hier beträgt der Zahlenwert von RN 10k.
- TK ist eine Kelvin-Temperatur, deren Einheit K ist. Hier beträgt der Zahlenwert von TK 273,15 + Grad Celsius.
- TN ist eine Nenn-Kelvin-Temperatur; die Einheit ist auch K. Hier beträgt der Zahlenwert von TN 273,15+25.
- Und B(beta), die Materialkonstante des NTC-Thermistors, wird auch als Wärmeempfindlichkeitsindex bezeichnet und hat einen Zahlenwert von 3950.
- exp ist die Abkürzung für exponentiell, und die Basiszahl e ist eine natürliche Zahl und beträgt ungefähr 2,7.

Wandeln Sie diese Formel  $TK=1/(\ln(RT/RN)/B+1/TN)$  um, um eine Kelvin-Temperatur zu erhalten, die minus 273,15 Grad Celsius entspricht.

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

Dieser Code bezieht sich darauf, Rt in die Formel  $TK=1/(\ln(RT/RN)/B+1/TN)$  einzusetzen, um die Kelvin-Temperatur zu erhalten.

```
temp = temp - 273.15
```

Umwandlung der Kelvin-Temperatur in Grad Celsius.

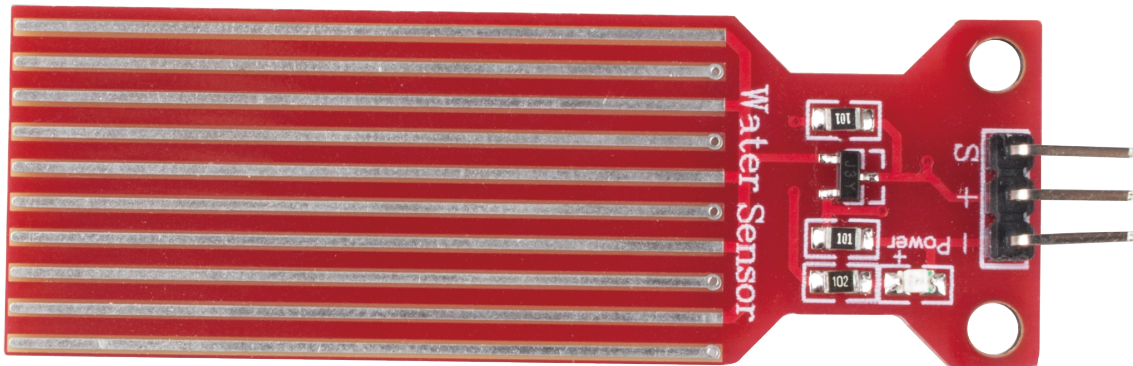
```
Fah = Cel * 1.8 + 32
```

Umwandlung des Celsius-Grades in Fahrenheit.

```
print ('Celsius: %.2f °C Fahrenheit: %.2f ' % (Cel, Fah))
```

Geben Sie Grad Celsius, Fahrenheit und deren Einheiten in der Shell aus.

## 4.20 2.14 Wasserstand erfühlen



Der Wasserstandssensor ist zur Wasserdetektion konzipiert und kann vielfältig zur Messung von Regenfällen, Wasserstand und sogar Flüssigkeitsaustritt eingesetzt werden.

Er misst den Wasserstand, indem er eine Reihe von freiliegenden parallelen Drahtspuren verwendet, um die Größe der Wassertropfen/das Volumen zu messen. Das Wasserstandsvolumen lässt sich leicht in ein analoges Signal umwandeln, und der ausgegebene Analogwert kann direkt vom Hauptsteuerboard abgelesen werden, um den Wasserstandsalarm auszulösen.

**Warnung:** Der Sensor darf nicht vollständig unter Wasser getaucht werden. Lassen Sie nur den Teil, an dem sich die zehn Spuren befinden, mit Wasser in Kontakt kommen. Zudem wird das Einschalten des Sensors in einer feuchten Umgebung die Korrosion der Sonde beschleunigen und die Lebensdauer des Sensors verkürzen. Es wird daher empfohlen, den Sensor nur beim Ablesen einzuschalten.

- *Wasserspiegelsensor-Modul*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

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

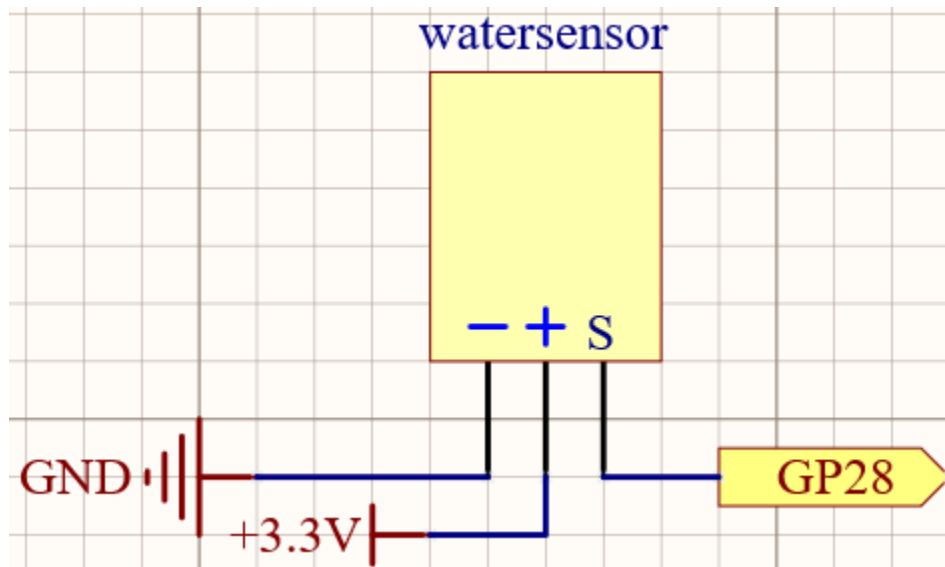
Name	ARTIKEL IN DIESEM KIT	LINK
Kepler Kit	450+	

Sie können die Komponenten auch einzeln über die untenstehenden Links erwerben.

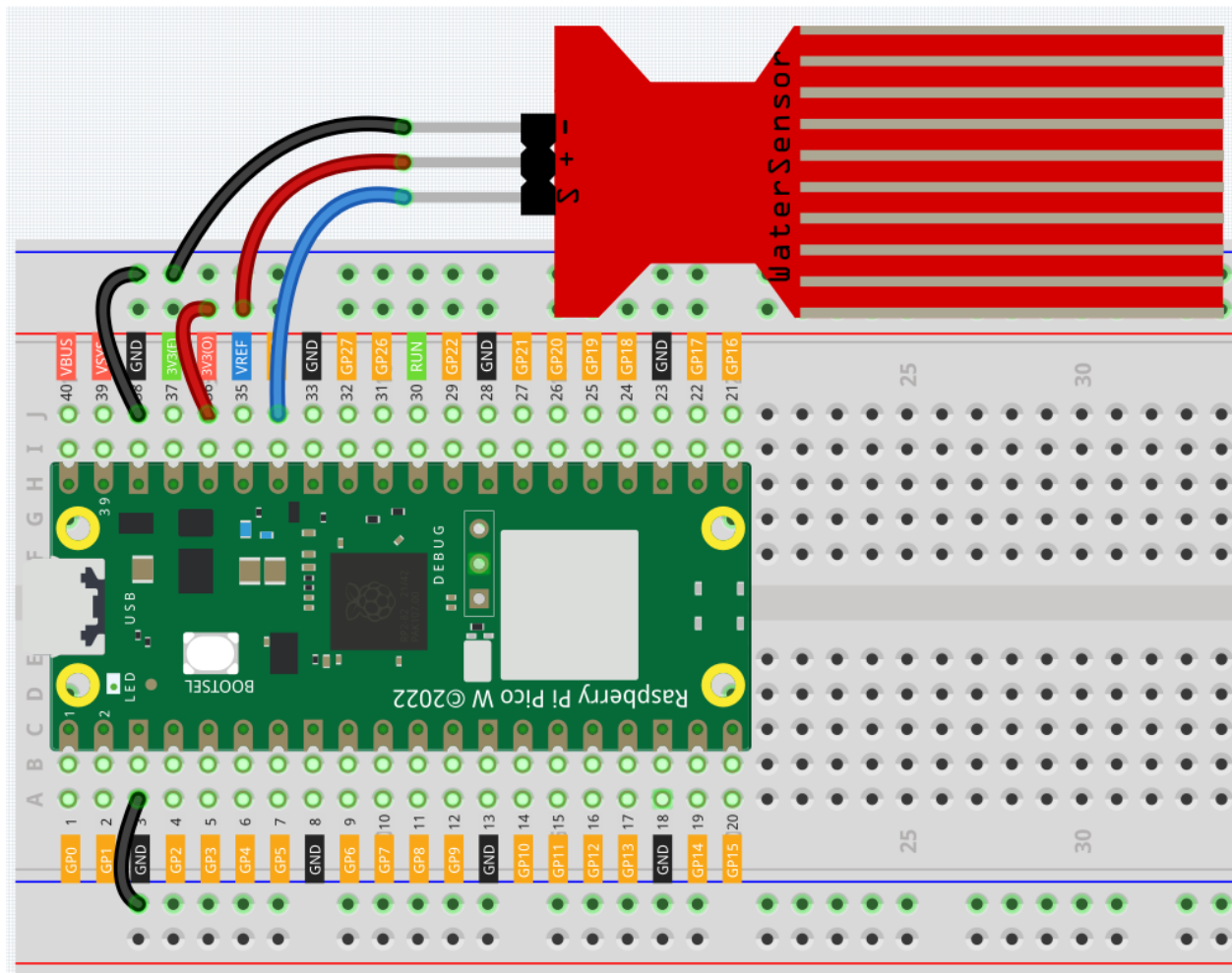


SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Wasserspiegelsensor-Modul</i>	1	

### Schaltplan



### Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `2.14_feel_the_water_level.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie anschließend auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
- Für ausführliche Tutorials verweisen wir auf [Code direkt öffnen und ausführen](#).

```
import machine
import utime

sensor = machine.ADC(28)

while True:
    value=sensor.read_u16()
    print(value)
    utime.sleep_ms(200)
```

Nachdem das Programm ausgeführt wurde, tauchen Sie das Wasserstandsmodul langsam ins Wasser. Mit zunehmender Tiefe wird die Shell einen höheren Wert ausgeben.

### Mehr erfahren

Es gibt eine Möglichkeit, das Analogeingangsmodul als digitales Modul zu verwenden.

Zunächst ermitteln Sie den Wert des Wasserstandssensors in einer trockenen Umgebung und verwenden diesen als Schwellenwert. Dann führen Sie die Programmierung durch und lesen den Wert des Wasserstandssensors erneut ab. Weicht der Messwert des Sensors deutlich vom Wert in einer trockenen Umgebung ab, ist er Flüssigkeiten ausgesetzt. Mit anderen Worten: Platzieren Sie dieses Gerät in der Nähe eines Wasserrohrs, kann es feststellen, ob ein Leck im Rohr vorliegt.

---

### Bemerkung:

- Öffnen Sie die Datei `2.14_water_level_threshold.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie anschließend auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
  - Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
  - Für ausführliche Tutorials verweisen wir auf *[Code direkt öffnen und ausführen](#)*.
- 

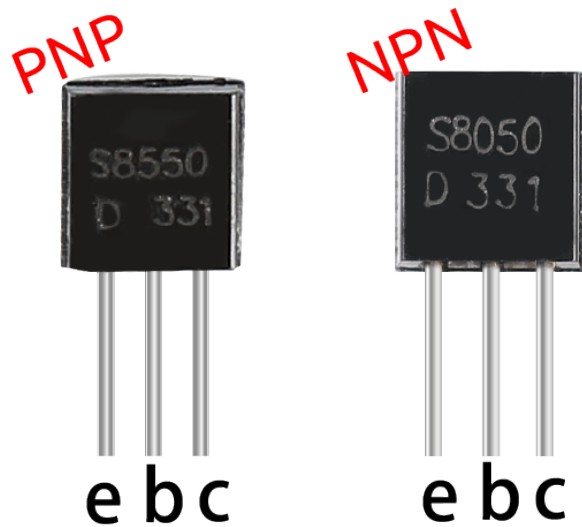
```
import machine
import utime

sensor = machine.ADC(28)
threshold = 30000 #Dieser Wert muss an die Umgebung angepasst werden.

while True:
    value=sensor.read_u16()
    if value > threshold :
        print("Liquid leakage!")
    utime.sleep_ms(200)
```

## 4.21 2.15 Zwei Arten von Transistoren

Dieses Set ist mit zwei Arten von Transistoren ausgestattet, dem S8550 und dem S8050, wobei der erstere ein PNP- und der letztere ein NPN-Typ ist. Sie sehen sehr ähnlich aus, daher ist eine genaue Kontrolle der Beschriftungen erforderlich. Wird ein High-Level-Signal durch einen NPN-Transistor geleitet, wird dieser aktiviert. Ein PNP-Transistor hingegen benötigt ein Low-Level-Signal zur Ansteuerung. Beide Transistortypen werden häufig in berührungslosen Schaltungen verwendet, wie auch in diesem Experiment.



Verwenden wir LED und Schalter, um den Umgang mit Transistoren zu verstehen!

### Transistor

#### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

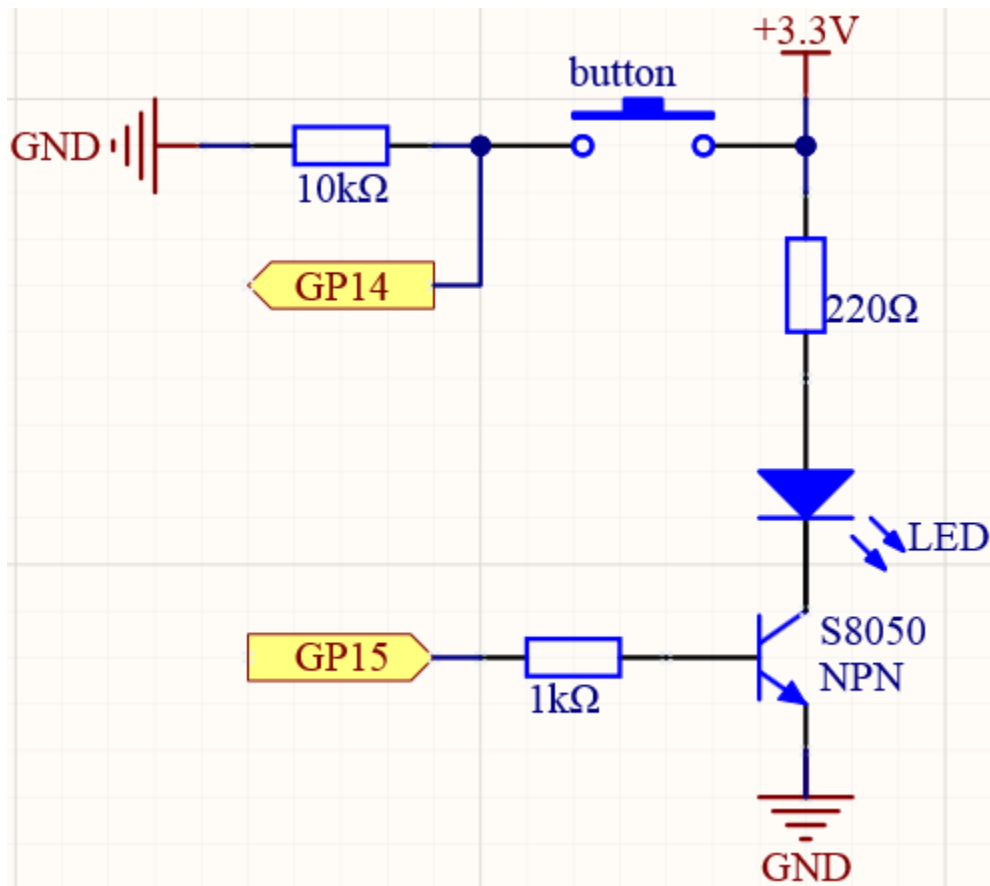
Es ist natürlich praktisch, gleich ein ganzes Set zu kaufen. Hier ist der Link dazu:

Name	ARTIKEL IM SET	LINK
Kepler-Set	450+	

Die Komponenten können auch einzeln über die folgenden Links erworben werden.

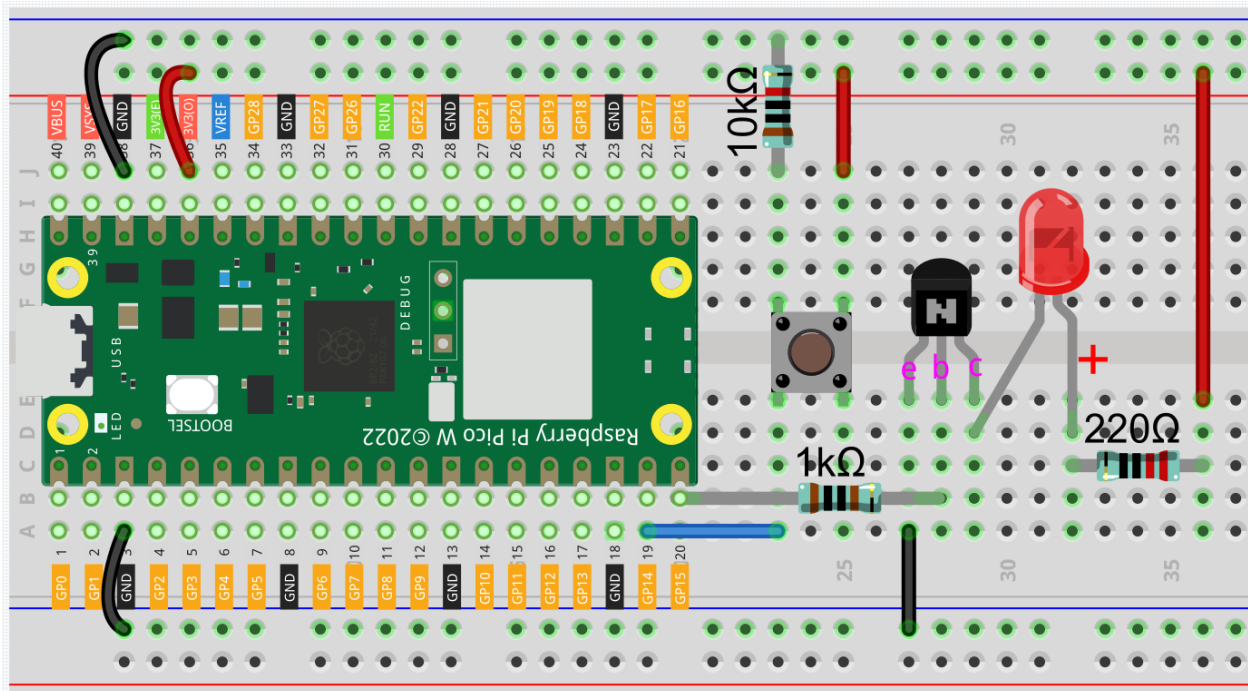
SN	KOMPONENTE	AN-ZAHL	LINK
1	<a href="#">Raspberry Pi Pico W</a>	1	
2	Micro-USB-Kabel	1	
3	<a href="#">Steckbrett</a>	1	
4	<a href="#">Jumperkabel</a>	Mehrere	
5	<a href="#">Widerstand</a>	3(220, 1K, 10K)	
6	<a href="#">LED</a>	1	
7	<a href="#">Taster</a>	1	
8	<a href="#">Transistor</a>	1(S8050/S8550)	

#### Anschluss des NPN (S8050) Transistors

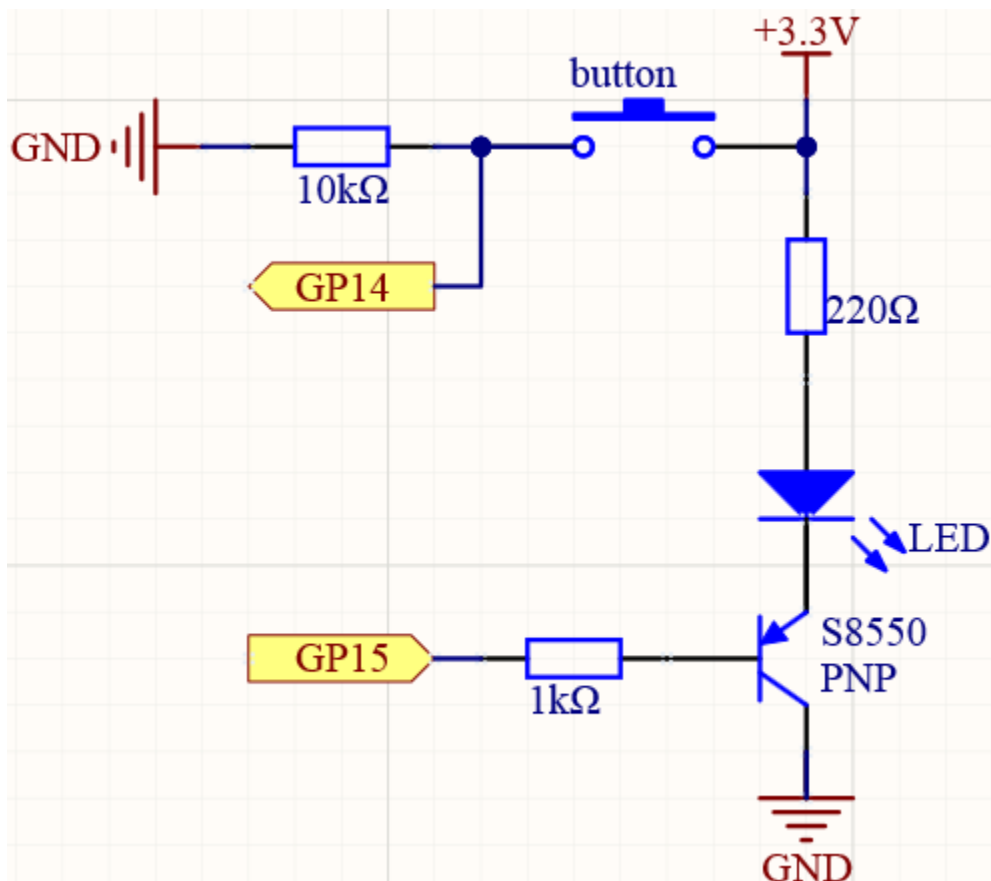


In dieser Schaltung wird GP14 auf High gesetzt, wenn der Taster gedrückt wird.

Programmiert man GP15 auf einen hohen Ausgangspegel, so wird nach einem 1k-Strombegrenzungswiderstand (zum Schutz des Transistors) der S8050 (NPN-Transistor) zum Leiten gebracht und die LED leuchtet auf.



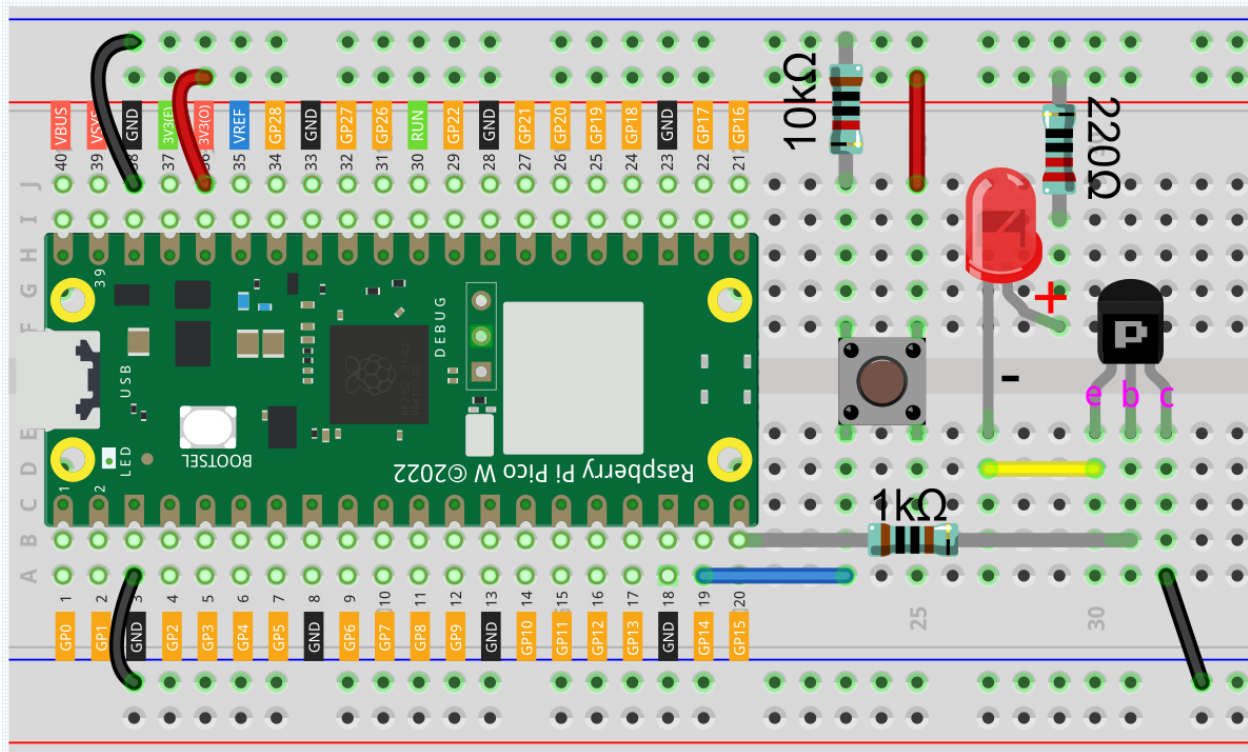
Anschluss des PNP(S8550) Transistors



In dieser Schaltung ist GP14 standardmäßig auf Low und ändert auf High, wenn der Taster gedrückt wird.

Programmiert man GP15 auf einen niedrigen Ausgangspegel, so wird nach einem 1k-Strombegrenzungswiderstand (zum Schutz des Transistors) der S8550 (PNP-Transistor) zum Leiten gebracht und die LED leuchtet auf.

Der einzige Unterschied, den Sie zwischen dieser und der vorherigen Schaltung bemerken werden, besteht darin, dass in der vorherigen Schaltung die Kathode der LED mit dem **Kollektor** des **S8050 (NPN-Transistor)** und in dieser mit dem **Emitter** des **S8550 (PNP-Transistor)** verbunden ist.



## Code

### Bemerkung:

- Öffnen Sie die Datei `2.15_transistor.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny, und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, auf den „MicroPython (Raspberry Pi Pico)“-Interpreter in der unteren rechten Ecke zu klicken.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
button = machine.Pin(14, machine.Pin.IN)
signal = machine.Pin(15, machine.Pin.OUT)

while True:
    button_status = button.value()
    if button_status == 1:
        signal.value(1)
    elif button_status == 0:
        signal.value(0)
```

Beide Transistortypen können mit dem gleichen Code gesteuert werden. Drücken wir den Taster, sendet der Pico W ein High-Level-Signal an den Transistor; lassen wir los, wird ein Low-Level-Signal gesendet. Wir sehen, dass sich in den beiden Schaltungen diametral entgegengesetzte Phänomene ergeben.

- Die Schaltung mit dem S8050 (NPN-Transistor) leuchtet auf, wenn der Taster gedrückt wird, das heißt, sie empfängt einen High-Level-Stromkreis;
- Die Schaltung mit dem S8550 (PNP-Transistor) leuchtet auf, wenn sie losgelassen wird, das heißt, sie empfängt einen Low-Level-Stromkreis.

## 4.22 2.16 Steuerung eines weiteren Stromkreises

Im Alltag können wir einen Schalter betätigen, um eine Lampe ein- oder auszuschalten. Doch was ist, wenn Sie die Lampe mit Pico W so steuern möchten, dass sie sich automatisch nach zehn Minuten ausschaltet?

Ein Relais kann Ihnen dabei helfen.

Ein Relais ist tatsächlich eine spezielle Art von Schalter, der von einer Seite des Stromkreises (in der Regel ein Niederspannungsstromkreis) gesteuert wird und dazu dient, die andere Seite des Stromkreises (in der Regel ein Hochspannungsstromkreis) zu steuern. Das macht es praktisch, unsere Haushaltsgeräte umzurüsten, damit sie durch ein Programm gesteuert, zu intelligenten Geräten gemacht oder sogar mit dem Internet verbunden werden können.

**Warnung:** Das Modifizieren von Elektrogeräten ist mit großen Gefahren verbunden. Versuchen Sie es nicht leichtfertig und führen Sie es nur unter Anleitung von Fachleuten durch.

- *Relais*

In diesem Projekt verwenden wir als Beispiel nur einen einfachen, durch ein Steckbrett-Strommodul betriebenen Stromkreis, um zu zeigen, wie man ihn mit einem Relais steuert.

- `cpn_power_module`

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

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

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

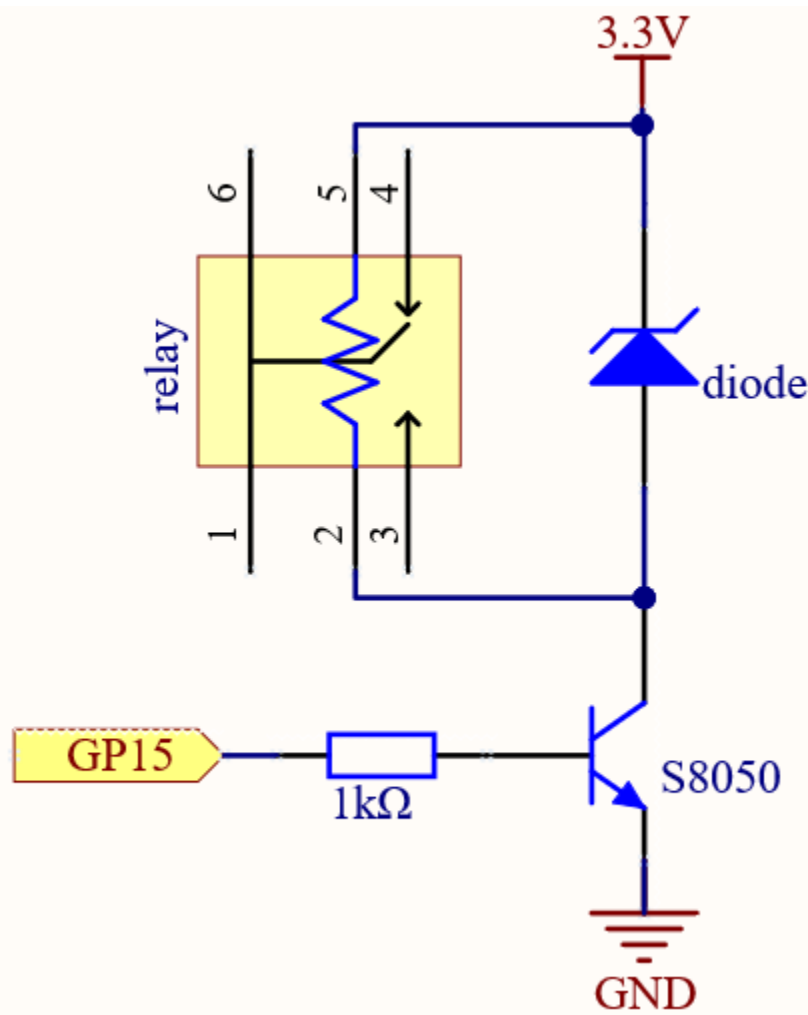
Sie können die Komponenten auch einzeln über die untenstehenden Links kaufen.



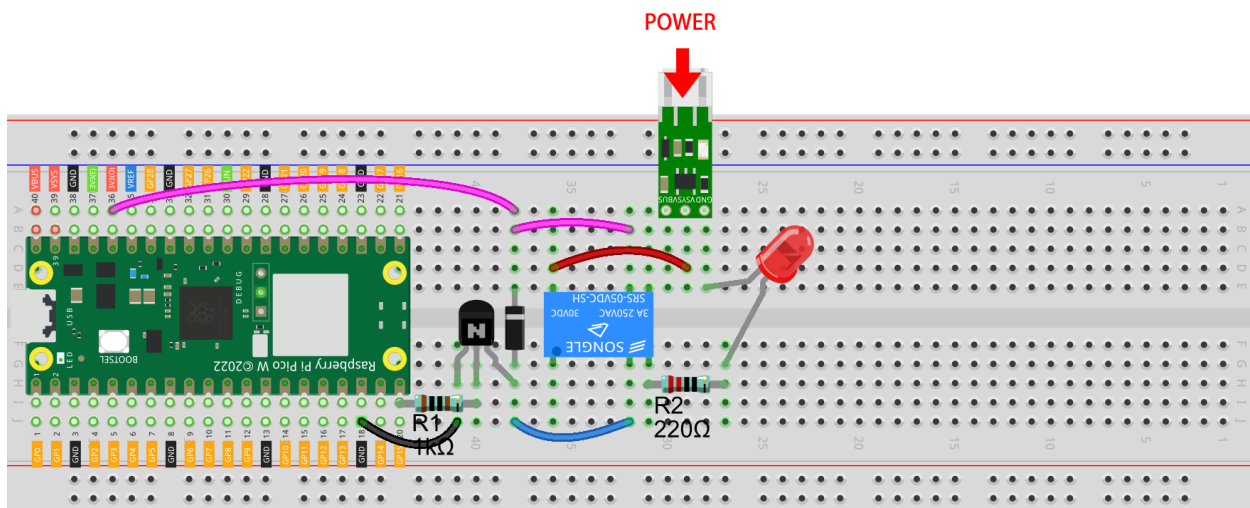
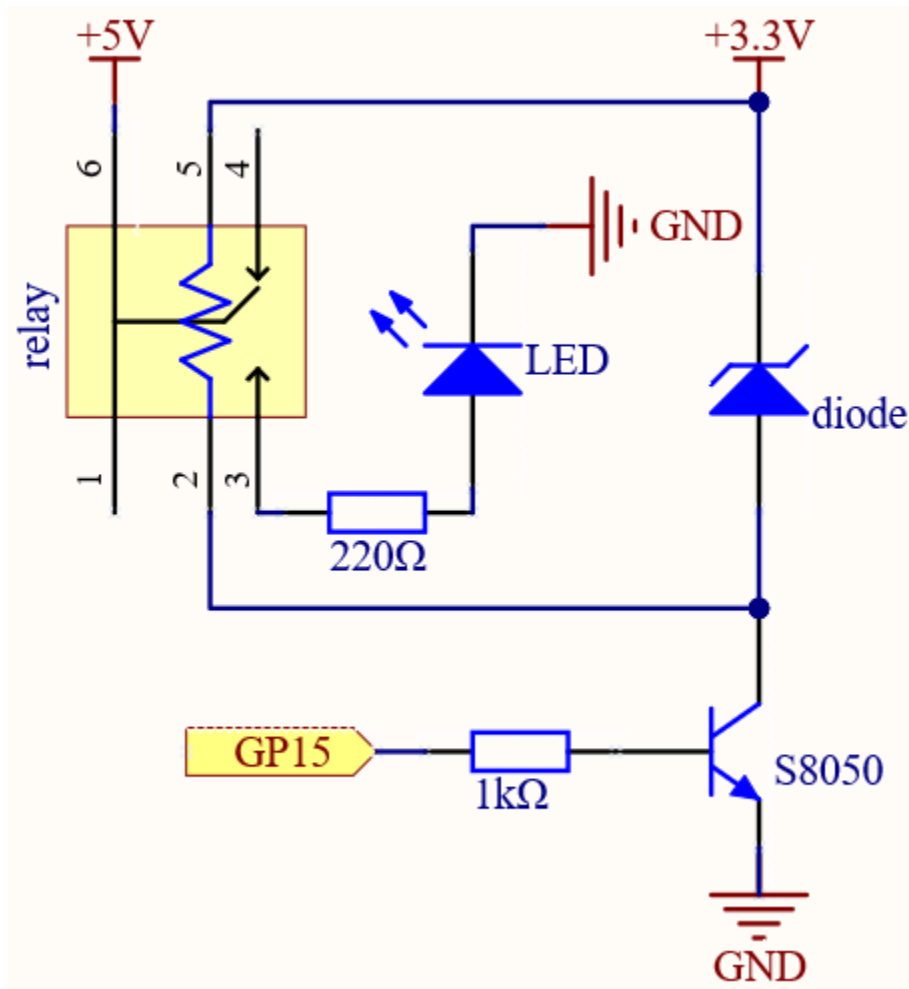
SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1 (S8050)	
6	<i>Diode</i>	1	
7	<i>Relais</i>	1	

### Verdrahtung

Zuerst erstellen wir einen Niederspannungsstromkreis zur Steuerung eines Relais. Das Ansteuern des Relais erfordert einen hohen Strom, daher ist ein Transistor erforderlich. Hier verwenden wir den S8050.







Jetzt kann das Relais den Laststromkreis ein- und ausschalten.

Code

Bemerkung:

- Öffnen Sie die Datei `2.16_control_another_circuit.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny, und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5, um es auszuführen.
  - Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
  - Für detaillierte Anleitungen siehe *Code direkt öffnen und ausführen*.
- 

```
import machine
import utime

relay = machine.Pin(15, machine.Pin.OUT)
while True:
    relay.value(1)
    utime.sleep(2)
    relay.value(0)
    utime.sleep(2)
```

Wenn der Code ausgeführt wird, wechselt das Relais alle zwei Sekunden den Betriebszustand des gesteuerten Stromkreises. Sie können eine der Zeilen manuell auskommentieren, um die Korrespondenz zwischen dem Relaisschaltkreis und dem Laststromkreis weiter zu klären.

### Weitere Informationen

Pin 3 des Relais ist normalerweise offen und schließt nur, wenn die Kontaktpule in Betrieb ist; Pin 4 ist normalerweise geschlossen und schließt, wenn die Kontaktpule erregt wird. Pin 1 ist mit Pin 6 verbunden und ist der gemeinsame Anschluss des Laststromkreises.

Indem man ein Ende des Laststromkreises von Pin 3 auf Pin 4 wechselt, erhält man genau den gegenteiligen Betriebszustand.

### 3. Ton & Anzeige & Bewegung

## 4.23 3.1 Piepton

Der aktive Summer ist ein typisches digitales Ausgabegerät, dessen Anwendung genauso einfach ist wie das Einschalten einer LED!

- *Summer*

### Erforderliche Komponenten

Für dieses Projekt benötigen wir folgende Bauteile.

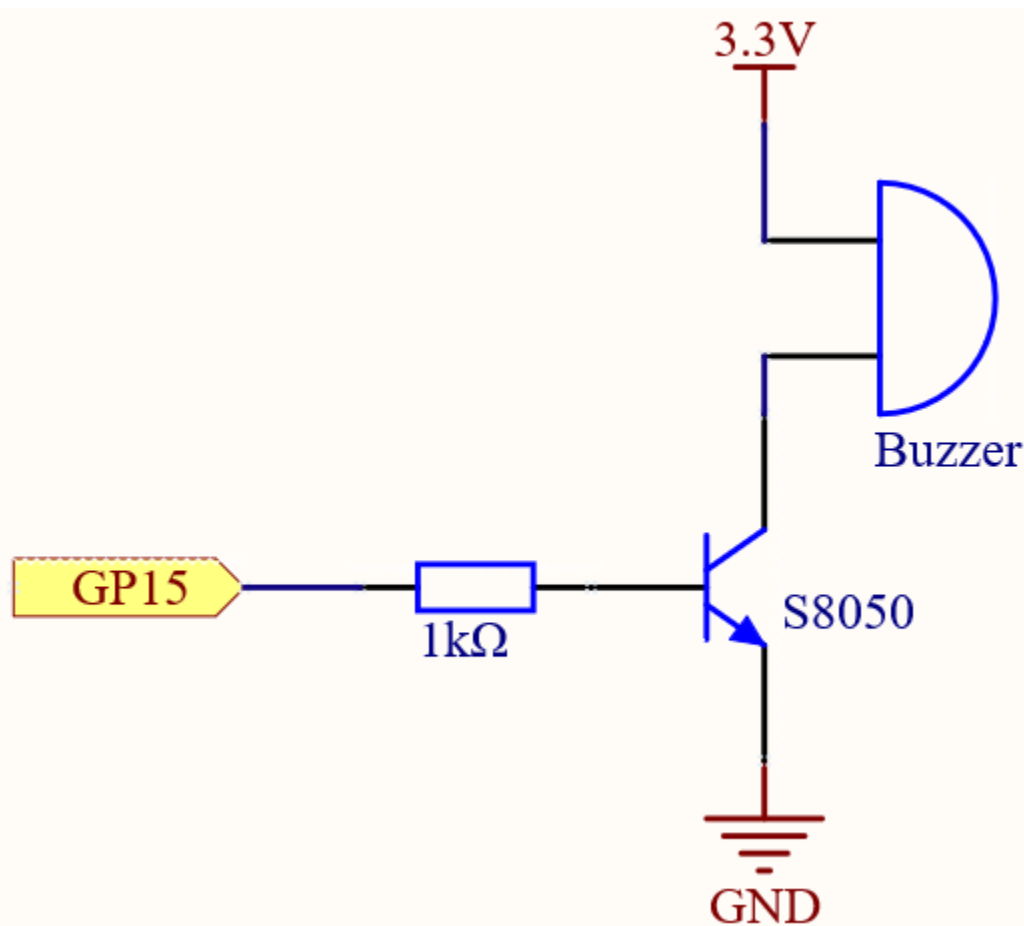
Ein Gesamtpaket zu kaufen ist natürlich bequemer, hier ist der Link dazu:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die Bauteile aber auch einzeln unter den folgenden Links erwerben.

SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1 (S8050)	
6	<i>Widerstand</i>	1 (1K)	
7	Aktiver <i>Summer</i>	1	

### Schaltbild



Wird der GP15-Ausgang auf „High“ gesetzt, wird der S8050 (NPN-Transistor) nach dem 1K-Strombegrenzungswiderstand (zum Schutz des Transistors) leitend und der Summer gibt einen Ton ab.

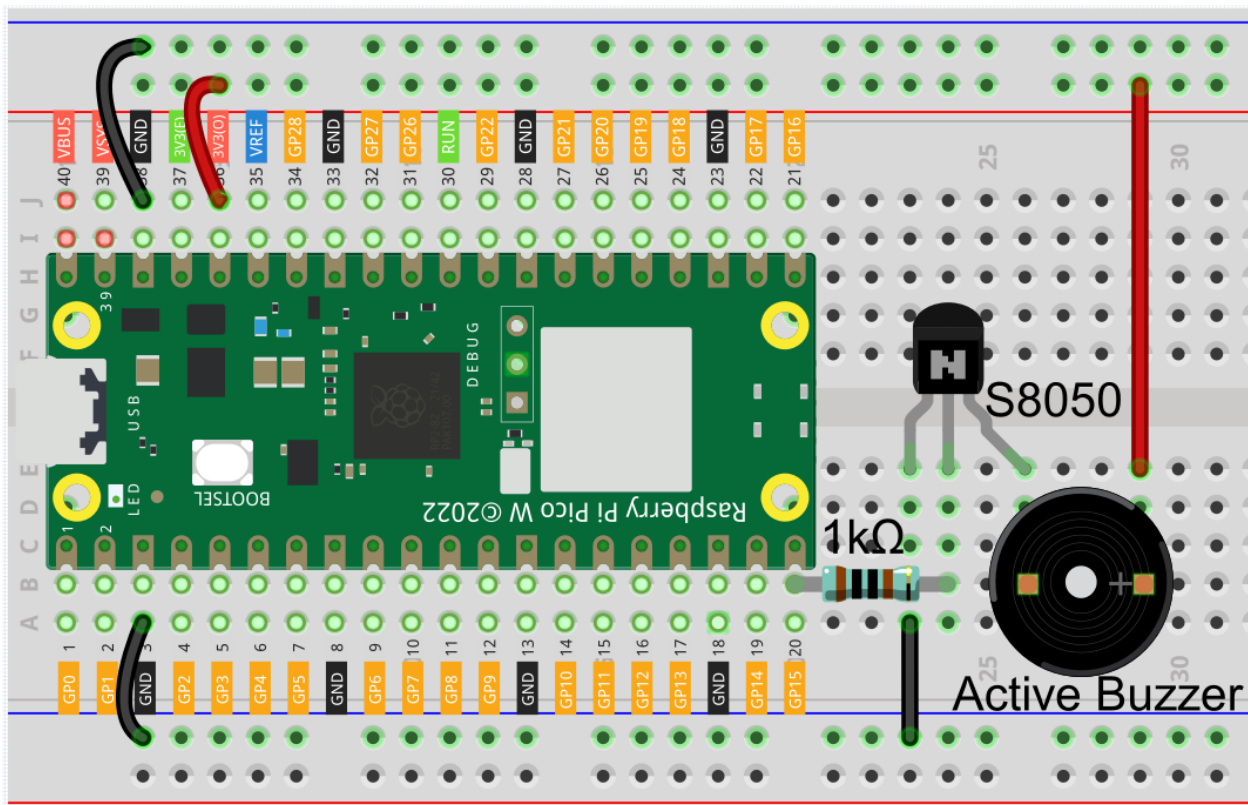
Die Aufgabe des S8050 (NPN-Transistor) besteht darin, den Strom zu verstärken, damit der Summer lauter tönt. Tatsächlich können Sie den Summer auch direkt an GP15 anschließen, werden dann aber feststellen, dass der Ton leiser ist.

### Verdrahtung

Im Kit sind zwei verschiedene Summertypen enthalten. Wir verwenden den aktiven Summer. Drehen Sie beide um, der versiegelte Rücken (nicht die freiliegende Leiterplatte) ist der, den wir benötigen.



Für den Betrieb des Summers ist ein Transistor erforderlich, hier verwenden wir den S8050 (NPN-Transistor).



### Code

#### Bemerkung:

- Öffnen Sie die Datei 3.1\_beep.py im Ordner kepler-kit-main/micropython oder kopieren Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.

- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
import utime

buzzer = machine.Pin(15, machine.Pin.OUT)
while True:
    for i in range(4):
        buzzer.value(1)
        utime.sleep(0.3)
        buzzer.value(0)
        utime.sleep(0.3)
    utime.sleep(1)
```

Nach dem Ausführen des Codes hören Sie jede Sekunde einen Piepton.

## 4.24 3.2 Eigener Ton

Im vorherigen Projekt haben wir einen aktiven Summer verwendet, diesmal setzen wir einen passiven Summer ein.

Wie der aktive Summer arbeitet auch der passive Summer mit dem Phänomen der elektromagnetischen Induktion. Der Unterschied besteht darin, dass ein passiver Summer keine eigene Schwingungsquelle hat. Daher wird er bei Verwendung von Gleichstromsignalen keinen Ton erzeugen. Dies ermöglicht es dem passiven Summer jedoch, seine eigene Schwingungsfrequenz anzupassen und unterschiedliche Töne wie „do, re, mi, fa, sol, la, ti“ auszusenden.

Lassen Sie den passiven Summer eine Melodie erklingen!

- [Summer](#)

### Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

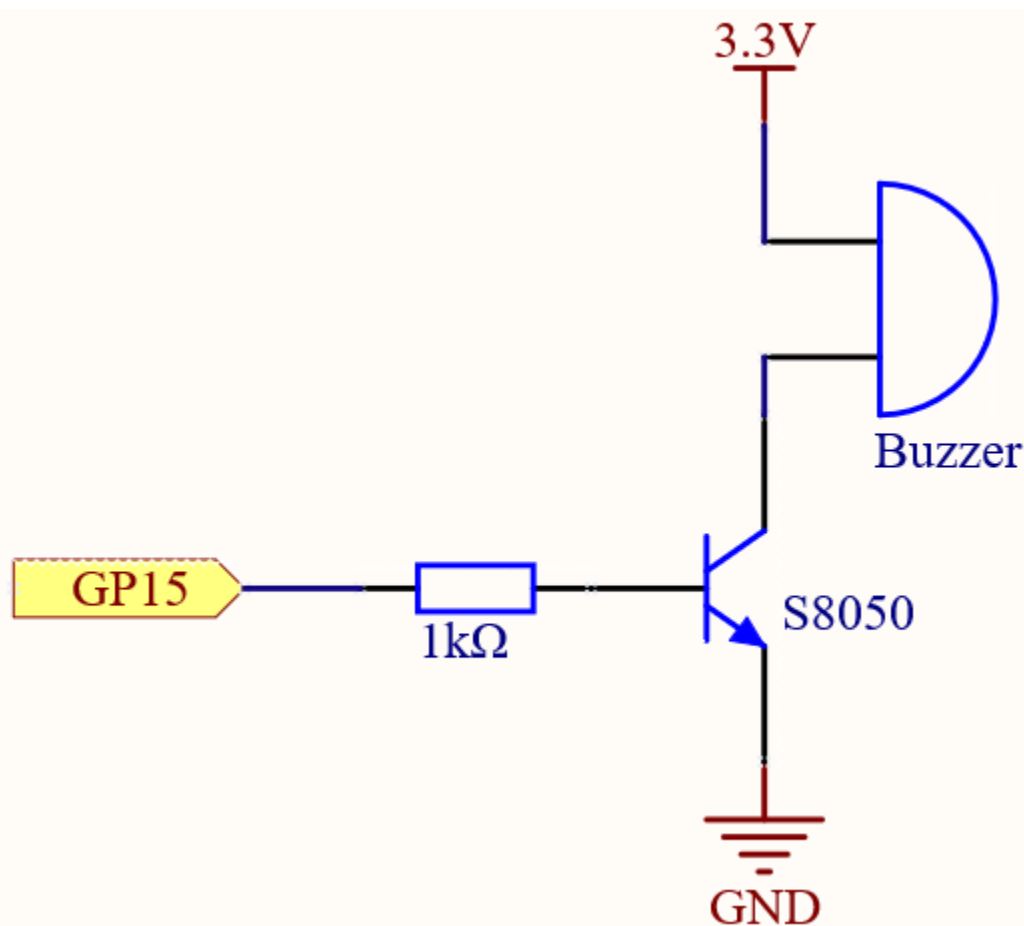
Ein Gesamtpaket zu kaufen ist natürlich praktisch, hier ist der Link dazu:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Die Komponenten können auch einzeln über die unten stehenden Links erworben werden.

SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1(S8050)	
6	<i>Widerstand</i>	1(1K)	
7	Passive <i>Summer</i>	1	

### Schaltbild



Wird der GP15-Ausgang auf „High“ gesetzt, wird nach dem 1K-Strombegrenzungswiderstand (zum Schutz des Transistors) der S8050 (NPN-Transistor) leitend, und der Summer gibt einen Ton ab.

Die Rolle des S8050 (NPN-Transistor) besteht darin, den Strom zu verstärken und somit den Ton des Summers zu erhöhen. Tatsächlich können Sie den Summer auch direkt an GP15 anschließen, dann werden Sie allerdings feststellen, dass der Ton leiser ist.

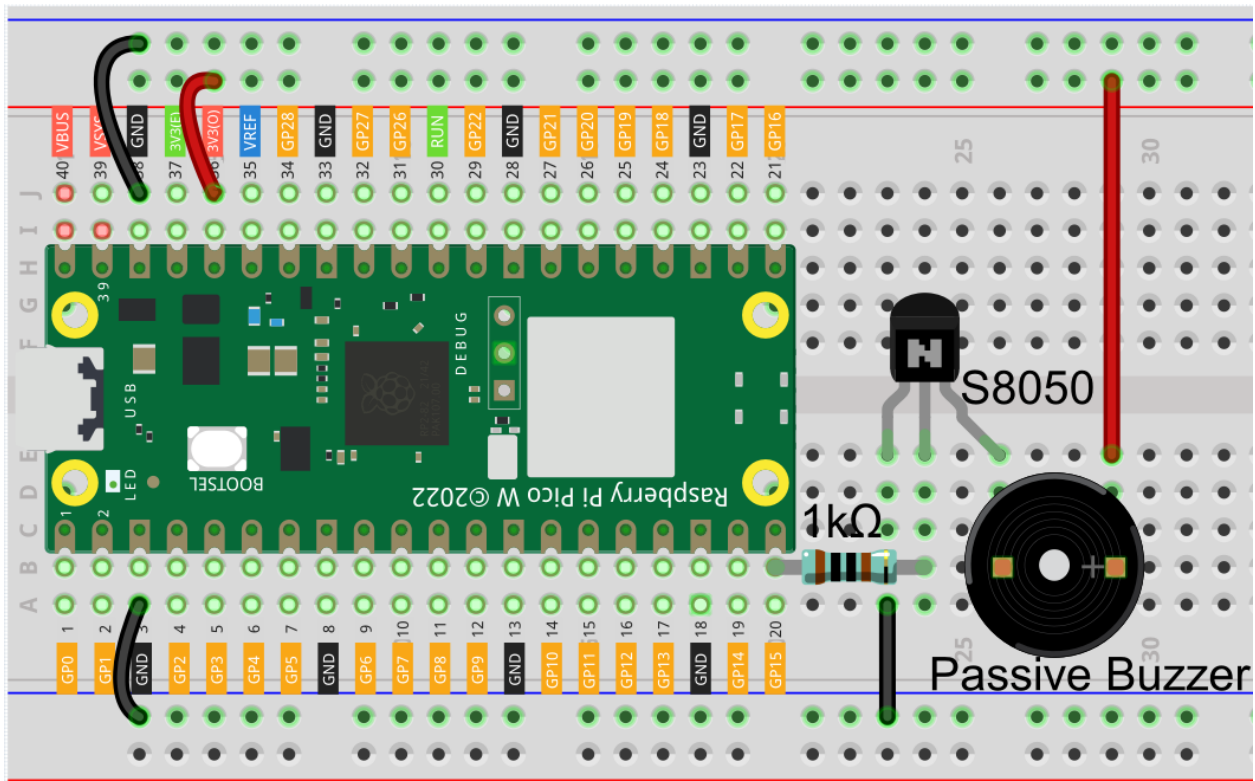


## Verdrahtung



Im Kit sind zwei verschiedene Summertypen enthalten; wir verwenden einen passiven Summer (den mit der freilegenden Leiterplatte auf der Rückseite).

Für den Betrieb des Summers ist ein Transistor erforderlich, hier verwenden wir den S8050.



## Code

**Bemerkung:**

- Öffnen Sie die Datei `3.2_custom_tone.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, unten rechts den Interpreter „MicroPython (Raspberry Pi Pico)“ auszuwählen.
- Für ausführliche Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
import utime

buzzer = machine.PWM(machine.Pin(15))

def tone(pin, frequency, duration):
    pin.freq(frequency)
    pin.duty_u16(30000)
    utime.sleep_ms(duration)
    pin.duty_u16(0)

tone(buzzer, 440, 250)
utime.sleep_ms(500)
tone(buzzer, 494, 250)
utime.sleep_ms(500)
tone(buzzer, 523, 250)
```

### Wie funktioniert es?

Wenn der passive Summer ein digitales Signal erhält, kann er nur die Membran bewegen, ohne einen Ton zu erzeugen. Daher verwenden wir die Funktion `tone()` um ein PWM-Signal zu generieren und den passiven Summer zum Klingen zu bringen.

Diese Funktion hat drei Parameter:

- **pin**, der GPIO-Pin, der den Summer steuert.
- **Frequenz**, die Tonhöhe des Summers wird durch die Frequenz bestimmt. Je höher die Frequenz, desto höher die Tonhöhe.
- **Dauer**, die Dauer des Tons.

Wir nutzen die Funktion `duty_u16()` um den Tastgrad auf 30000 (etwa 50%) zu setzen. Es können auch andere Werte sein; wichtig ist nur, ein diskontinuierliches elektrisches Signal zu erzeugen.

### Mehr erfahren

Wir können den spezifischen Ton gemäß der Grundfrequenz des Klaviers simulieren, um ein vollständiges Musikstück zu spielen.

- [Frequenzen der Klaviertasten - Wikipedia](#)

---

### Bemerkung:

- Öffnen Sie die Datei `3.2_custom_tone_2.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, unten rechts den Interpreter „MicroPython (Raspberry Pi Pico)“ auszuwählen.
- Für ausführliche Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
import utime
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

NOTE_C4 = 262
NOTE_G3 = 196
NOTE_A3 = 220
NOTE_B3 = 247

melody = [NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, NOTE_B3, NOTE_C4]

buzzer = machine.PWM(machine.Pin(15))

def tone(pin, frequency, duration):
    pin.freq(frequency)
    pin.duty_u16(30000)
    utime.sleep_ms(duration)
    pin.duty_u16(0)

for note in melody:
    tone(buzzer, note, 250)
    utime.sleep_ms(150)

```

## 4.25 3.3 RGB LED-Streifen

WS2812 ist eine intelligente LED-Lichtquelle, bei der die Steuerschaltung und der RGB-Chip in einem 5050-Komponentenpaket integriert sind. Sie enthält eine intelligente digitale Port-Datenverriegelung und eine Signalformungsverstärkungs-Antriebsschaltung. Zusätzlich verfügt sie über einen präzisen internen Oszillator und einen programmierbaren Konstantstromregler, der effektiv die Farbkonsistenz der einzelnen Pixel gewährleistet.

Das Datenübertragungsprotokoll verwendet den einzelnen NZR-Kommunikationsmodus. Nach dem Einschalten des Pixels empfängt der DIN-Port Daten vom Controller. Das erste Pixel sammelt die ersten 24-Bit-Daten und sendet sie an die interne Datenverriegelung. Die weiteren, durch die interne Signalformungsverstärkung geformten Daten werden durch den DO-Port zum nächsten Kaskadenpixel gesendet. Nach der Übertragung für jedes Pixel verringert sich das Signal um 24 Bit. Das Pixel verwendet die automatische Signalumformungstechnologie, wodurch die Anzahl der kaskadierten Pixel nur von der Geschwindigkeit der Signalübertragung abhängt.

- [WS2812 RGB 8-LED-Streifen](#)

### Benötigte Komponenten

Für dieses Projekt werden folgende Komponenten benötigt.

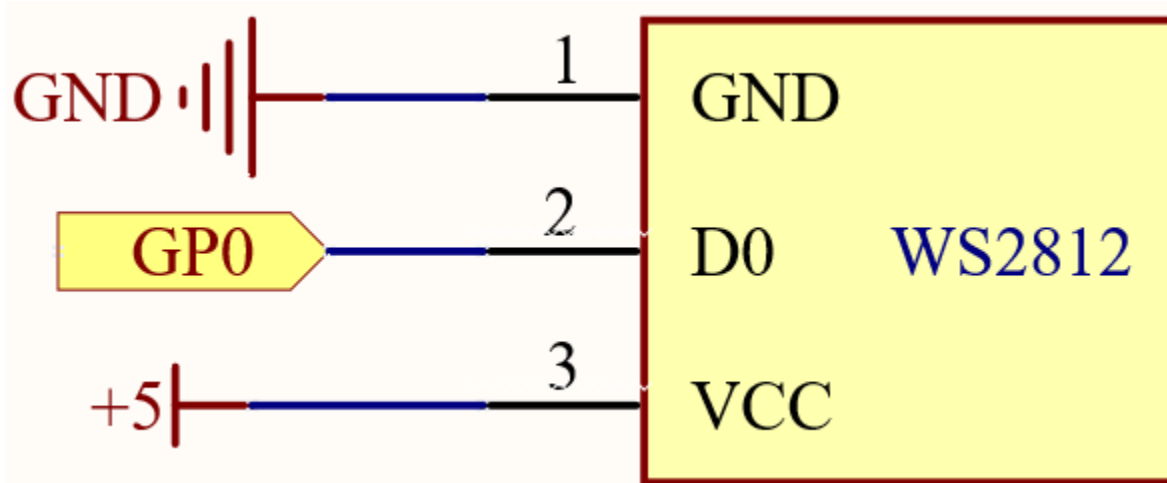
Ein vollständiges Kit ist definitiv praktisch, hier ist der Link:

Bezeichnung	TEILE IM KIT	LINK
Kepler Kit	450+	

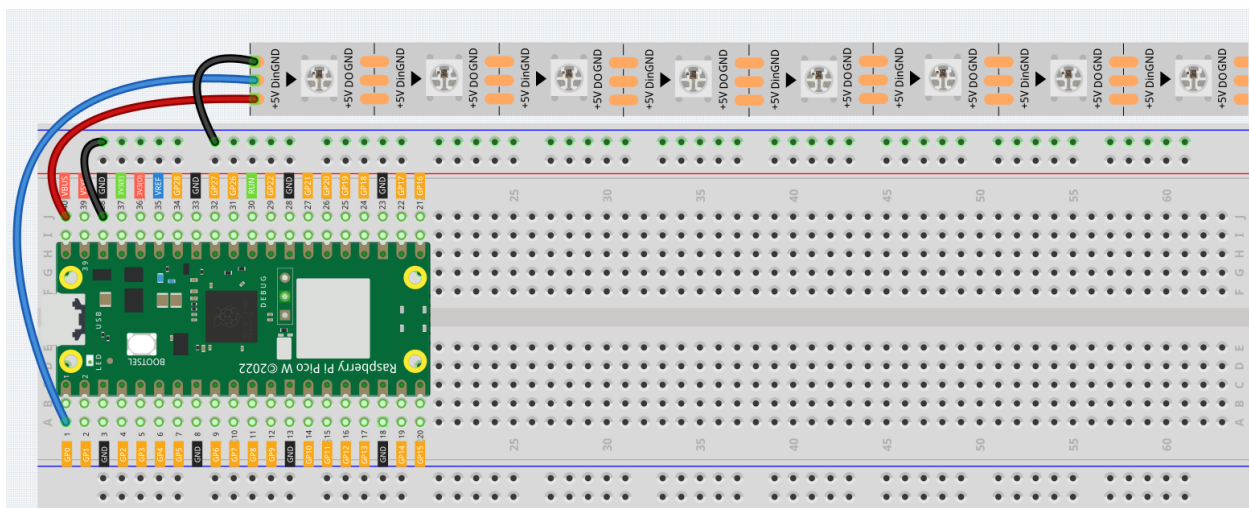
Sie können die Komponenten auch einzeln über die folgenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>WS2812 RGB 8-LED-Streifen</i>	1	

### Schaltplan



### Verkabelung



**Warnung:** Ein Punkt, den Sie beachten müssen, ist der Strom.

Obwohl der LED-Streifen mit beliebig vielen LEDs am Pico W betrieben werden kann, ist die Leistung seines

VBUS-Pins begrenzt. Hier verwenden wir acht LEDs, was sicher ist. Wenn Sie jedoch mehr LEDs verwenden möchten, benötigen Sie eine separate Stromversorgung.

## Code

### Bemerkung:

- Öffnen Sie die Datei `3.3_rgb_led_strip.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken Sie auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den Interpreter „MicroPython (Raspberry Pi Pico)“ in der unteren rechten Ecke auszuwählen.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).
- Hier benötigen Sie die Bibliothek `ws2812.py`, prüfen Sie, ob sie auf dem Pico W hochgeladen wurde. Eine detaillierte Anleitung finden Sie unter [1.4 Bibliotheken auf den Pico hochladen](#).

```
import machine
from ws2812 import WS2812

ws = WS2812(machine.Pin(0),8)

ws[0] = [64,154,227]
ws[1] = [128,0,128]
ws[2] = [50,150,50]
ws[3] = [255,30,30]
ws[4] = [0,128,255]
ws[5] = [99,199,0]
ws[6] = [128,128,128]
ws[7] = [255,100,0]
ws.write()
```

Wählen Sie einige Ihrer Lieblingsfarben aus und zeigen Sie sie auf dem RGB-LED-Streifen an!

### Wie funktioniert das?

In der `ws2812`-Bibliothek haben wir alle relevanten Funktionen in die Klasse `WS2812` integriert.

Sie können den RGB-LED-Streifen mit dem folgenden Befehl nutzen.

```
from ws2812 import WS2812
```

Deklarieren Sie ein `WS2812`-Objekt mit dem Namen „ws“, das an den „Pin“ angeschlossen ist, auf dem sich „Anzahl“ RGB-LEDs befinden.

```
ws = WS2812(pin, number)
```

`ws` ist ein Array-Objekt, dessen Elemente den einzelnen RGB-LEDs auf dem `WS2812`-Streifen entsprechen, beispielsweise ist `ws[0]` die erste und `ws[7]` die achte.

Sie können jeder RGB-LED Farbwerte zuweisen. Diese Werte müssen eine 24-Bit-Farbe sein (dargestellt durch sechs Hexadezimalziffern) oder eine Liste von drei 8-Bit-RGB-Werten.

Beispiel: Der rote Wert ist „`0xFF0000`“ oder „`[255,0,0]`“.

```
ws[i] = color value
```

Verwenden Sie dann diesen Befehl, um die Farbe für den LED-Streifen zu setzen und ihn zum Leuchten zu bringen.

```
ws.write()
```

Sie können auch direkt den folgenden Befehl verwenden, um alle LEDs in derselben Farbe leuchten zu lassen.

```
ws.write_all(color value)
```

### Mehr erfahren

Wir können zufällig Farben generieren und ein buntes, fließendes Licht erzeugen.

---

#### Bemerkung:

- Öffnen Sie die Datei `3.3_rgb_led_strip_2.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken Sie auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
  - Vergessen Sie nicht, den Interpreter „MicroPython (Raspberry Pi Pico)“ in der unteren rechten Ecke auszuwählen.
  - Für detaillierte Anleitungen siehe *[Code direkt öffnen und ausführen](#)*.
- 

```
import machine
from ws2812 import WS2812
import utime
import urandom

ws = WS2812(machine.Pin(0), 8)

def flowing_light():
    for i in range(7, 0, -1):
        ws[i] = ws[i-1]
    ws[0] = int(urandom.uniform(0, 0xFFFFFF))
    ws.write()
    utime.sleep_ms(80)

while True:
    flowing_light()
    print(ws[0])
```

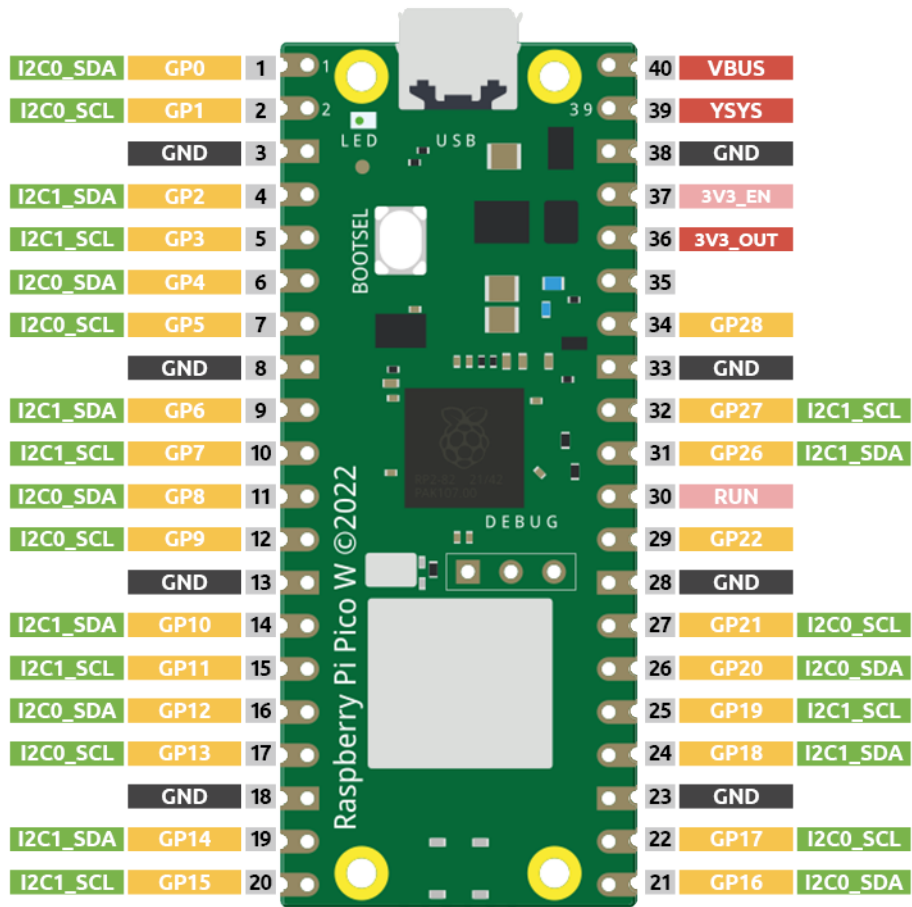
## 4.26 3.4 Flüssigkristallanzeige

LCD1602 ist eine Zeichen-Flüssigkristallanzeige, die gleichzeitig 32 (16\*2) Zeichen anzeigen kann.

Wie wir alle wissen, haben LCDs und andere Displays, obwohl sie die Mensch-Maschine-Interaktion erheblich bereichern, eine gemeinsame Schwachstelle. Wenn sie an einen Controller angeschlossen sind, werden mehrere IOs des Controllers belegt, der nicht so viele externe Anschlüsse hat. Das schränkt auch andere Funktionen des Controllers ein. Deshalb wurde LCD1602 mit einem I2C-Bus entwickelt, um dieses Problem zu lösen.

- *[I2C LCD1602](#)*

- Inter-Integrated Circuit - Wikipedia



Hier verwenden wir die I2C0-Schnittstelle, um den LCD1602 zu steuern und Text anzuzeigen.

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

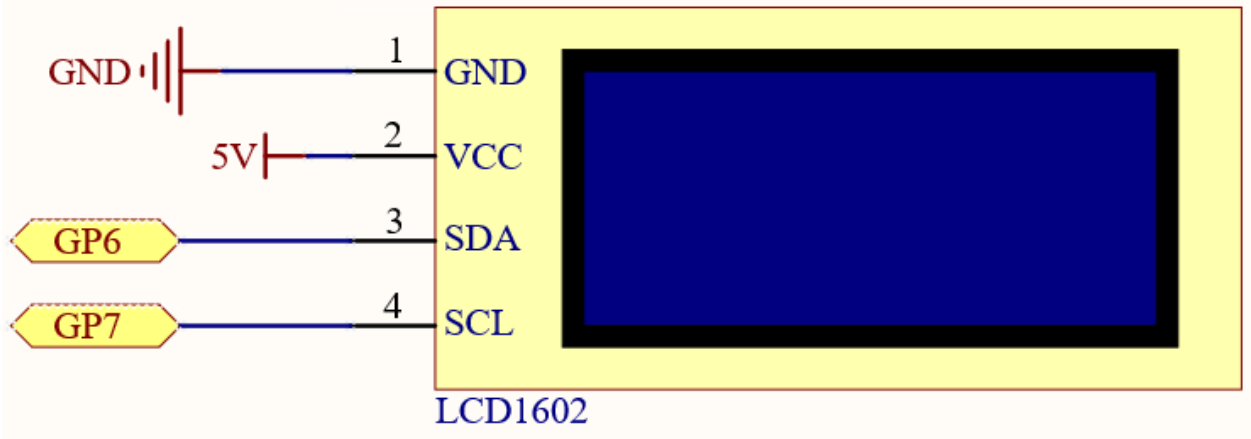
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
Kepler Kit	450+	

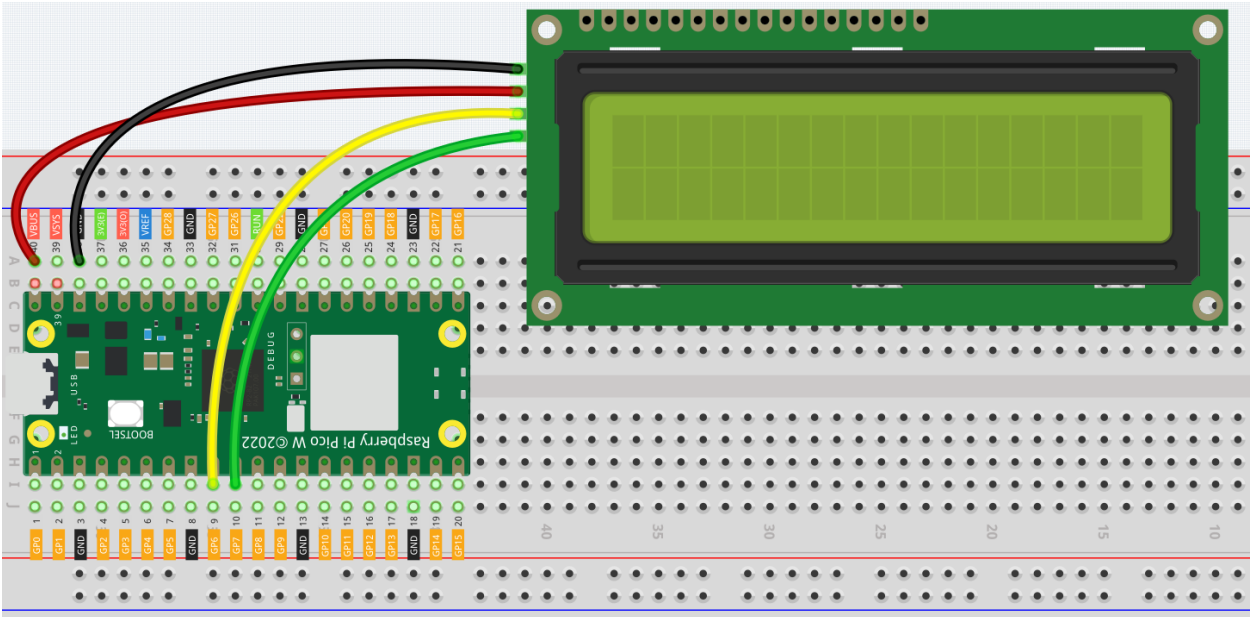
Sie können diese auch separat über die untenstehenden Links kaufen.

SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro USB Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>I2C LCD1602</i>	1	

Schaltplan



Verkabelung



Code



---

**Bemerkung:**

- Öffnen Sie die Datei `3.4_liquid_crystal_display.py` im Pfad `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny, dann klicken Sie auf „Aktuelles Skript ausführen“ oder drücken einfach F5.
  - Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der unteren rechten Ecke auszuwählen.
  - Für detaillierte Anleitungen siehe *Code direkt öffnen und ausführen*.
  - Hier benötigen Sie die Bibliothek `lcd1602.py`. Bitte überprüfen Sie, ob sie auf Pico W hochgeladen wurde. Eine detaillierte Anleitung finden Sie unter *1.4 Bibliotheken auf den Pico hochladen*.
- 

```
from lcd1602 import LCD
import utime

lcd = LCD()
string = " Hallo!\n"
lcd.message(string)
utime.sleep(2)
string = "    Sunfounder!"
lcd.message(string)
utime.sleep(2)
lcd.clear()
```

Nachdem das Programm ausgeführt wurde, erscheinen nacheinander zwei Textzeilen auf dem LCD und verschwinden dann wieder.

---

**Bemerkung:** Wenn der Code läuft und der Bildschirm leer bleibt, können Sie das Potentiometer auf der Rückseite drehen, um den Kontrast zu erhöhen.

---

**Wie funktioniert das?**

In der `lcd1602`-Bibliothek integrieren wir die relevanten Funktionen von `lcd1602` in die `LCD`-Klasse.

Importieren der `lcd1602`-Bibliothek

```
from lcd1602 import LCD
```

Deklarieren eines Objekts der `LCD`-Klasse und nennen es `lcd`.

```
lcd = LCD()
```

Mit dieser Anweisung wird der Text auf dem LCD angezeigt. Es sollte beachtet werden, dass das Argument ein String sein muss. Wenn wir eine Ganzzahl oder Fließkommazahl übergeben wollen, müssen wir die Umwandlungsanweisung `str()` verwenden.

```
lcd.message(string)
```

Wenn Sie diese Anweisung mehrmals aufrufen, überlagert `lcd` die Texte. Dafür muss die folgende Anweisung verwendet werden, um die Anzeige zu löschen.

```
lcd.clear()
```

## 4.27 3.5 Kleiner Ventilator

Nun verwenden wir den TA6586, um den Gleichstrommotor im Uhrzeigersinn und gegen den Uhrzeigersinn anzutreiben. Da der Gleichstrommotor einen vergleichsweise hohen Strom benötigt, setzen wir hier aus Sicherheitsgründen ein Netzteilmodul zur Stromversorgung des Motors ein.

- *Gleichstrommotor*
- *TA6586 - Motorsteuerungs-Chip*
- *cpn\_power\_module*

### Benötigte Bauteile

Für dieses Projekt benötigen wir die folgenden Komponenten.

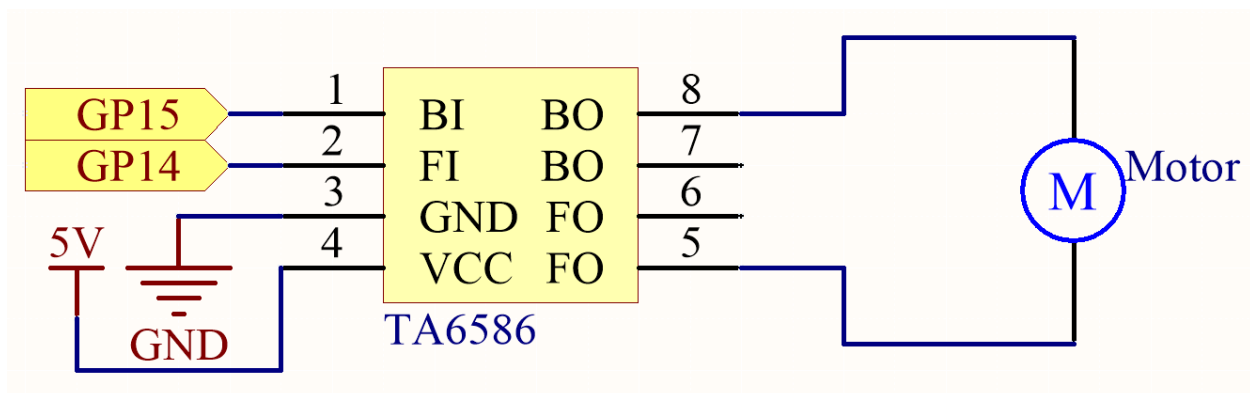
Es ist durchaus praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Bezeichnung	TEILE IN DIESEM KIT	LINK
Kepler Kit	450+	

Sie können die Bauteile auch einzeln über die unten stehenden Links erwerben.

SN	BAUTEIL	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>TA6586 - Motorsteuerungs-Chip</i>	1	
6	<i>Gleichstrommotor</i>	1	
7	<i>Li-Po-Lademodul</i>	1	
8	18650 Batterie	1	
9	Batteriehalter	1	

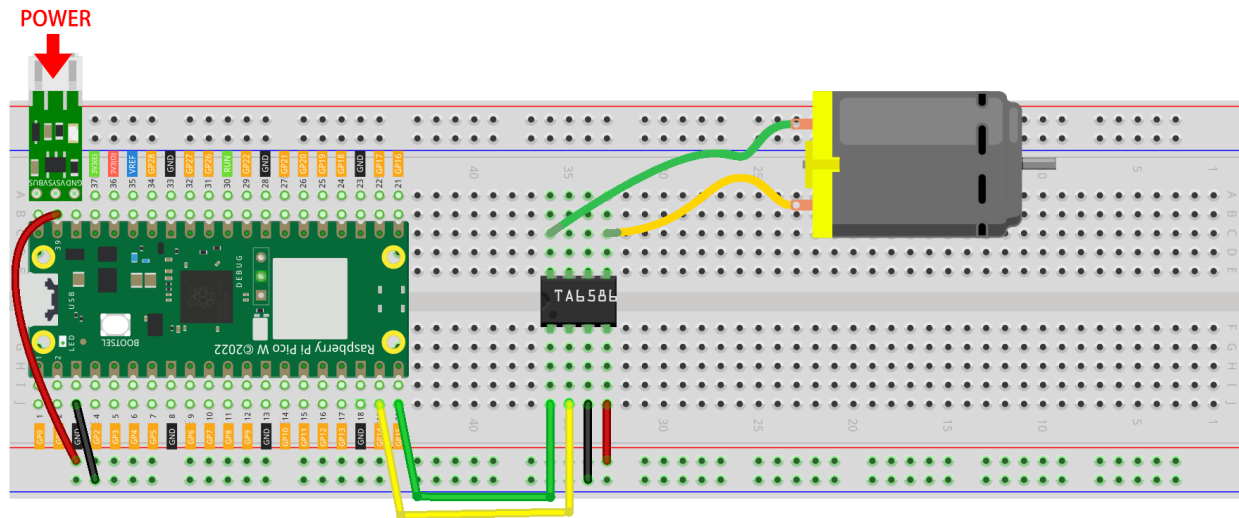
### Schaltplan



## Verdrahtung

### Bemerkung:

- Da Gleichstrommotoren einen hohen Strom benötigen, verwenden wir hier aus Sicherheitsgründen ein Li-Po-Ladegerätmodul zur Stromversorgung des Motors.
- Stellen Sie sicher, dass Ihr Li-Po-Ladegerätmodul wie im Schaltplan gezeigt angeschlossen ist. Andernfalls besteht die Gefahr eines Kurzschlusses, der Ihre Batterie und Schaltung beschädigen könnte.



### Code

### Bemerkung:

- Öffnen Sie die Datei `3.5_small_fan.py` im Pfad `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny, und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der unteren rechten Ecke auszuwählen.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
import utime

motor1A = machine.Pin(14, machine.Pin.OUT)
motor2A = machine.Pin(15, machine.Pin.OUT)

def clockwise():
    motor1A.high()
    motor2A.low()

def anticlockwise():
    motor1A.low()
    motor2A.high()
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

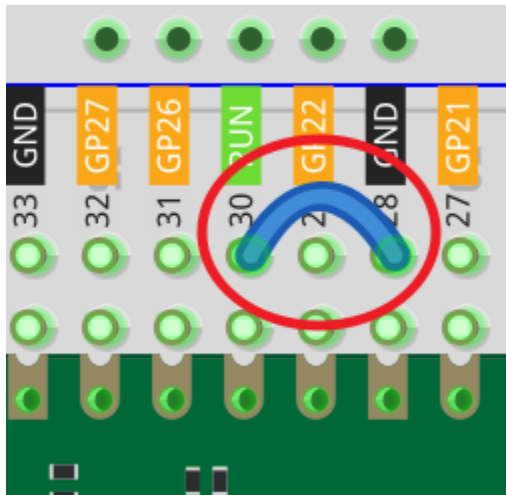
```
def stopMotor():
    motor1A.low()
    motor2A.low()

while True:
    clockwise()
    utime.sleep(1)
    stopMotor()
    utime.sleep(1)
    anticlockwise()
    utime.sleep(1)
    stopMotor()
    utime.sleep(1)
```

Sobald das Programm läuft, wird der Motor in einem regelmäßigen Muster hin und her drehen.

#### Bemerkung:

- Wenn der Motor sich nach dem Klicken auf die Stop-Taste weiterdreht, müssen Sie zu diesem Zeitpunkt den **RUN**-Pin am Pico W mit einem Draht auf GND zurücksetzen und dann diesen Draht entfernen, um den Code erneut auszuführen.
- Dies liegt daran, dass der Motor mit zu hohem Strom arbeitet, was dazu führen kann, dass der Pico W die Verbindung zum Computer verliert.



## 4.28 3.6 Pumpensteuerung

Kleine Kreispumpen eignen sich hervorragend für Projekte zur automatischen Pflanzenbewässerung. Sie können auch zum Erstellen kleiner intelligenter Wasserspiele verwendet werden.

Das Antriebselement ist ein Elektromotor, der genau wie ein herkömmlicher Motor betrieben wird.

- *DC-Wasserpumpe*
- *Gleichstrommotor*

- *TA6586 - Motorsteuerungs-Chip*
- `cpn_power_module`

#### Bemerkung:

1. Schließen Sie den Schlauch an den Motorausgang an, tauchen Sie die Pumpe ins Wasser und schalten Sie sie ein.
2. Achten Sie darauf, dass der Wasserstand stets über dem Motor liegt. Ein Leerlauf kann den Motor durch Hitze-generierung beschädigen und Lärm erzeugen.
3. Wenn Sie Pflanzen bewässern, sollten Sie das Einsaugen von Erde vermeiden, da dies die Pumpe verstopfen kann.
4. Sollte kein Wasser aus dem Schlauch kommen, könnte Restwasser im Schlauch die Luftzirkulation blockieren und muss zuerst abgelassen werden.

#### Benötigte Bauteile

Für dieses Projekt werden die folgenden Bauteile benötigt.

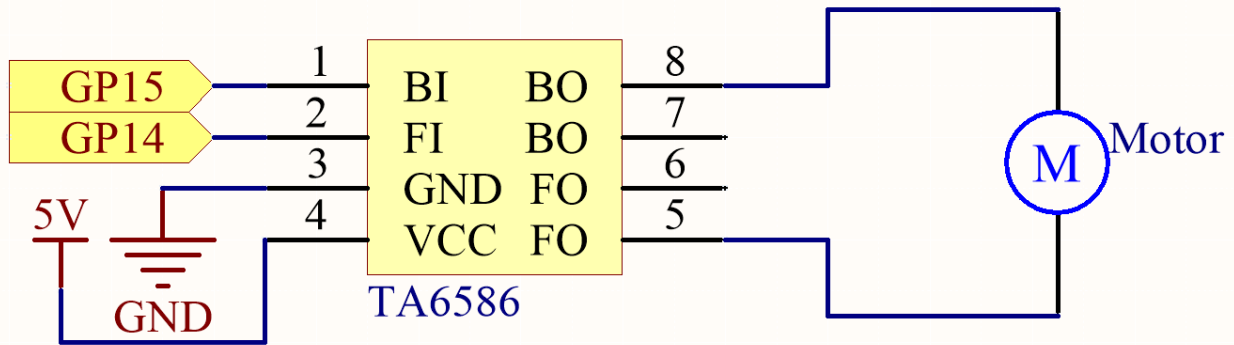
Es ist definitiv praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Bezeichnung	BAUTEILE IN DIESEM KIT	LINK
Kepler Kit	450+	

Sie können die Bauteile auch einzeln über die untenstehenden Links erwerben.

SN	BAUTEIL	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>TA6586 - Motorsteuerungs-Chip</i>	1	
6	<i>Li-Po-Lademodul</i>	1	
7	18650 Batterie	1	
8	Batteriehalter	1	
9	<i>DC-Wasserpumpe</i>	1	

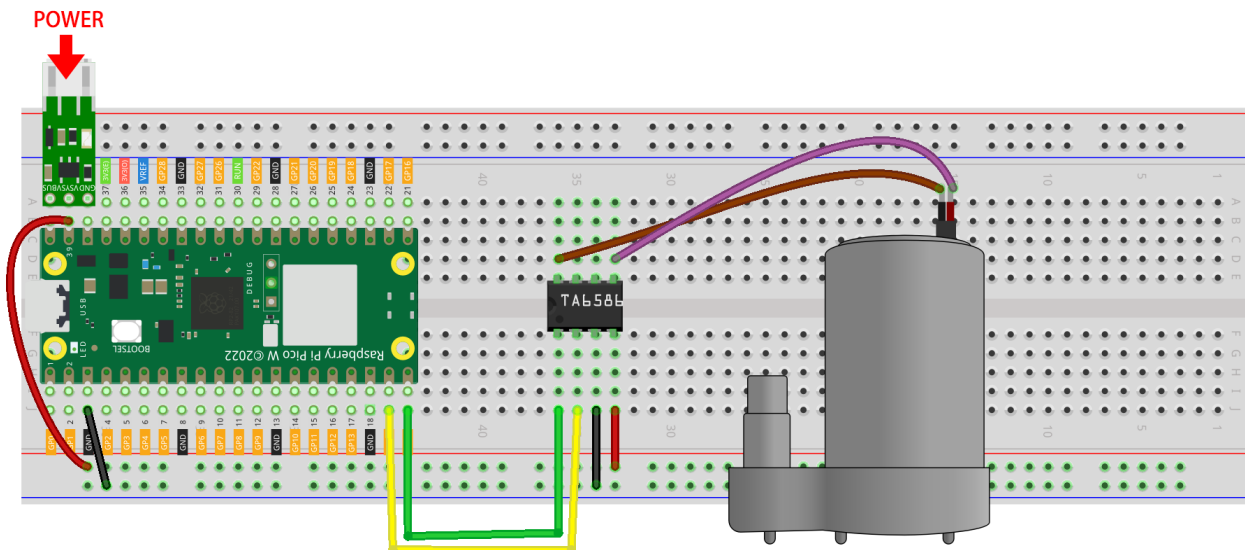
#### Schaltplan



## Verkabelung

### Bemerkung:

- Da die Pumpe einen hohen Strombedarf hat, nutzen wir ein Li-Po-Ladegerät-Modul, um den Motor aus Sicherheitsgründen mit Strom zu versorgen.
- Stellen Sie sicher, dass Ihr Li-Po-Ladegerät-Modul wie im Diagramm gezeigt angeschlossen ist. Andernfalls könnte es zu einem Kurzschluss kommen, der Ihre Batterie und Schaltung beschädigt.



## Code

### Bemerkung:

- Öffnen Sie die Datei `3.6_pumping.py` im Pfad `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny, dann klicken Sie auf „Aktuelles Skript ausführen“ oder drücken einfach F5.
- Vergessen Sie nicht, unten rechts auf den „MicroPython (Raspberry Pi Pico)“-Interpreter zu klicken.
- Für detaillierte Anleitungen verweisen wir auf [Code direkt öffnen und ausführen](#).

```
import machine
import utime

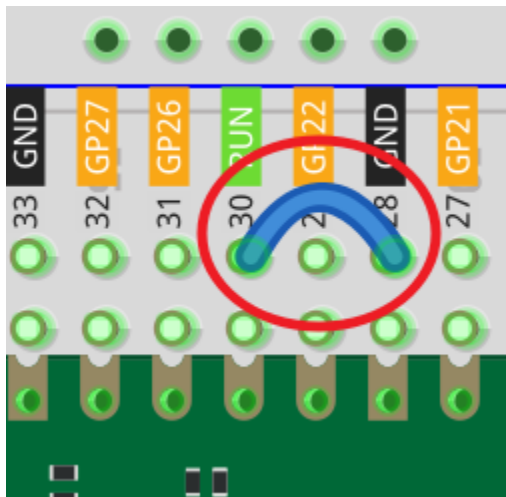
motor1A = machine.Pin(14, machine.Pin.OUT)
motor2A = machine.Pin(15, machine.Pin.OUT)

while True:
    motor1A.high()
    motor2A.low()
```

Nachdem der Code ausgeführt wurde, beginnt die Pumpe zu arbeiten und gleichzeitig fließt Wasser aus dem Schlauch.

#### Bemerkung:

- Sollte der Motor nach dem Klicken auf den Stop-Button immer noch laufen, müssen Sie zu diesem Zeitpunkt den **RUN**-Pin am Pico W mit einem Draht auf GND zurücksetzen und dann diesen Draht entfernen, um den Code erneut auszuführen.
- Dies liegt daran, dass der Motor mit zu hohem Strom arbeitet, was dazu führen kann, dass der Pico W die Verbindung zum Computer verliert.



## 4.29 3.7 Schwingender Servo

In diesem Bausatz befinden sich neben LEDs und passivem Summer auch ein durch PWM-Signale gesteuertes Gerät, nämlich der Servo.

Der Servo ist ein Positions- (Winkel-) Servo, ideal geeignet für Steuerungssysteme, die konstante Winkeländerungen erfordern und aufrechterhalten können. Er findet häufig Anwendung in hochwertigen ferngesteuerten Spielzeugen, wie etwa Flugzeug-, U-Boot-Modellen und ferngesteuerten Robotern.

Versuchen wir nun, den Servo zum Schwingen zu bringen!

- *Servo*

#### Benötigte Bauteile

Für dieses Projekt benötigen wir folgende Komponenten.

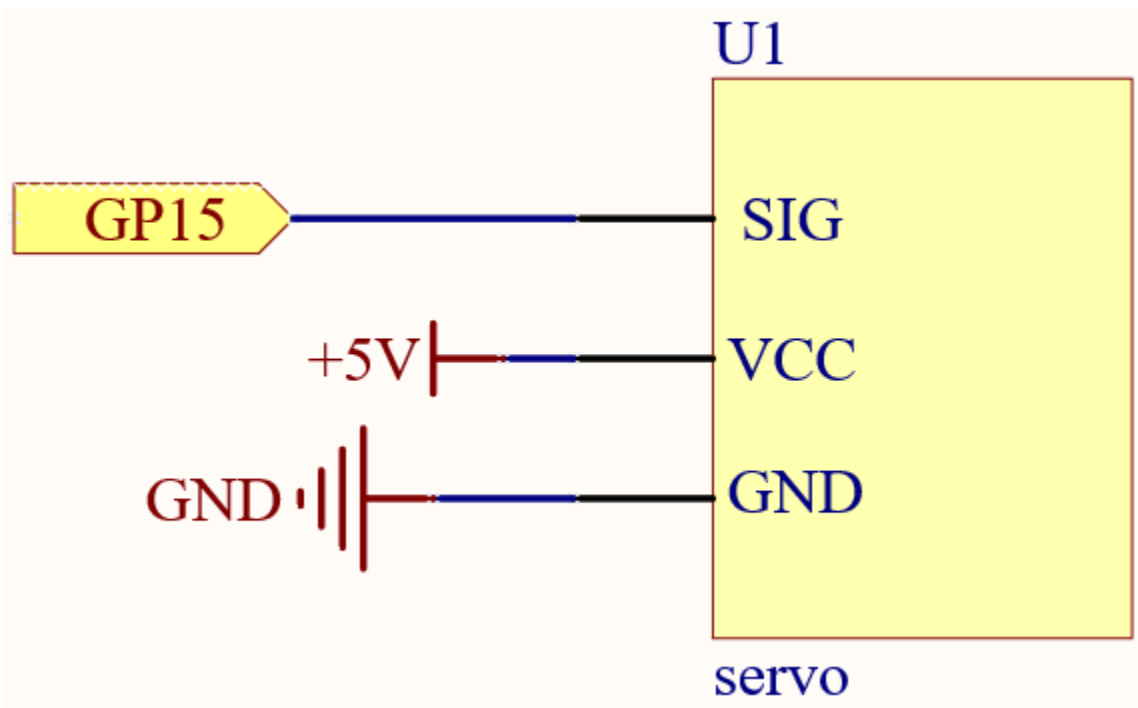
Es ist definitiv praktisch, das gesamte Set zu kaufen. Hier ist der Link:

Bezeichnung	BAUTEILE IM SET	LINK
Kepler Kit	450+	

Die Bauteile können auch einzeln über die nachstehenden Links erworben werden.

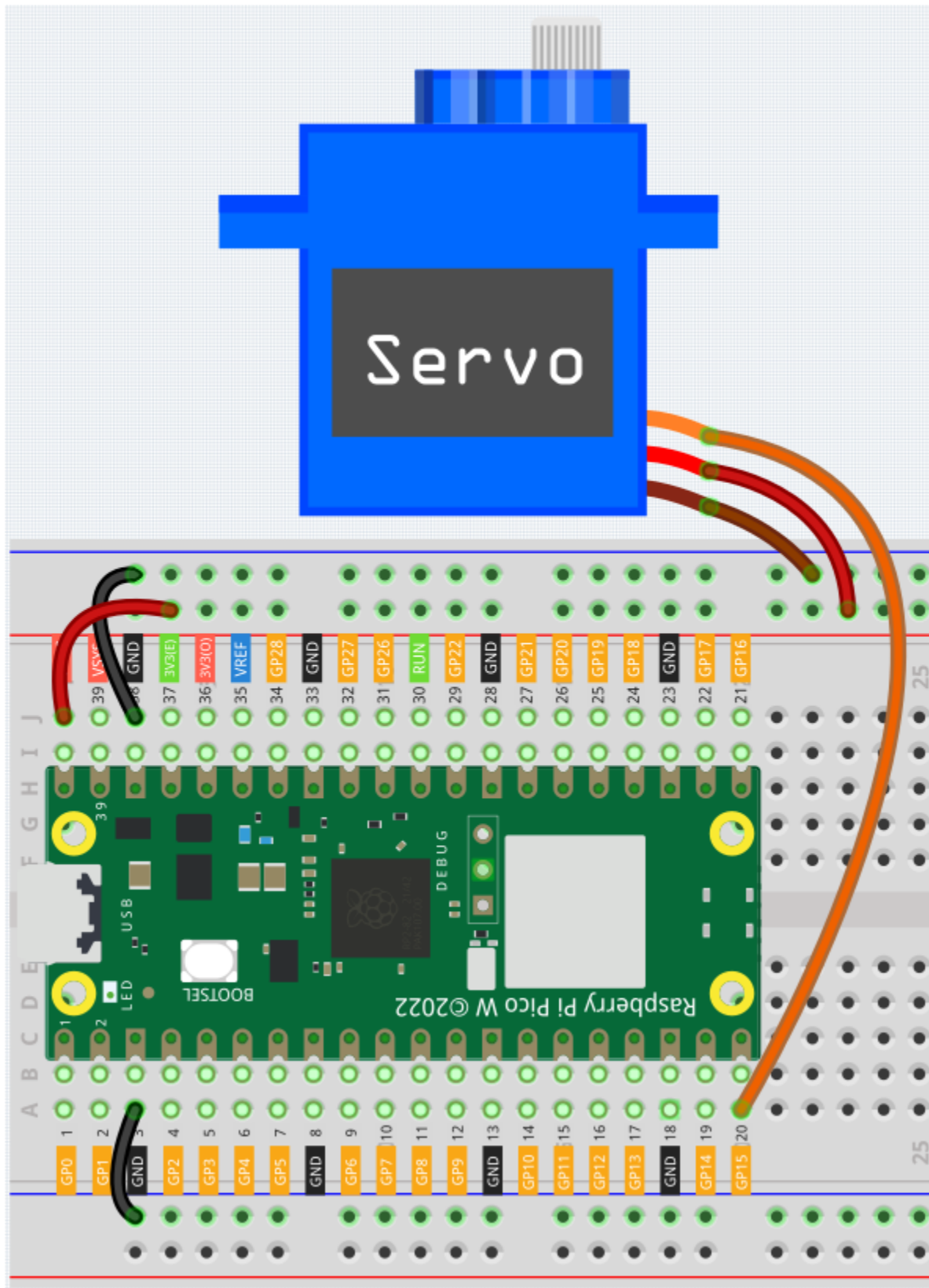
SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Servo</i>	1	

### Schaltplan



### Verdrahtung





- Das orangefarbene Kabel ist das Signal- und wird an GP15 angeschlossen.

- Das rote Kabel ist VCC und wird an VBUS(5V) angeschlossen.
- Das braune Kabel ist GND und wird an GND angeschlossen.

### Code

---

#### Bemerkung:

- Öffnen Sie die Datei `3.7_swinging_servo.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie den Code in Thonny und klicken Sie auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
  - Vergessen Sie nicht, im unteren rechten Eck auf den Interpreter „MicroPython (Raspberry Pi Pico)“ zu klicken.
  - Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).
- 

```
import machine
import utime

servo = machine.PWM(machine.Pin(15))
servo.freq(50)

def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def servo_write(pin,angle):
    pulse_width=interval_mapping(angle, 0, 180, 0.5,2.5)
    duty=int(interval_mapping(pulse_width, 0, 20, 0,65535))
    pin.duty_u16(duty)

while True:
    for angle in range(180):
        servo_write(servo,angle)
        utime.sleep_ms(20)
    for angle in range(180,-1,-1):
        servo_write(servo,angle)
        utime.sleep_ms(20)
```

Während das Programm läuft, sehen wir den Servoarm, der zwischen 0° und 180° hin- und herschwingt.

Das Programm wird durch die Schleife `while True` ständig ausgeführt, daher müssen wir den Stopp-Button drücken, um es zu beenden.

#### Wie funktioniert es?

Wir haben die Funktion `servo_write()` definiert, um den Servo zu steuern.

Diese Funktion hat zwei Parameter:

- `pin`, der GPIO-Pin, der den Servo steuert.
- `Angle`, der Ausgangswinkel der Welle.

In dieser Funktion wird `interval_mapping()` aufgerufen, um den Winkelbereich von 0 ~ 180 Grad auf die Pulsdauer von 0,5 ~ 2,5 ms abzubilden.

```
pulse_width=interval_mapping(angle, 0, 180, 0.5,2.5)
```

Warum genau 0,5 ~ 2,5 ms? Das ist durch den Arbeitsmodus des Servos bestimmt.

*Servo*

Anschließend wird die Pulsdauer von der Periode in die Tastverhältnis umgewandelt. Da `duty_u16()` keine Dezimalstellen akzeptiert, verwenden wir `int()`, um das Tastverhältnis in einen Ganzzahltyp umzuwandeln.

```
duty=int(interval_mapping(pulse_width, 0, 20, 0,65535))
```

Schließlich wird der Tastverhältniswert in `duty_u16()` geschrieben.

**4. Steuerung****4.30 4.1 Den Joystick umschalten**

Wenn du viele Videospiele spielst, solltest du mit dem Joystick bestens vertraut sein. Er wird normalerweise verwendet, um die Spielfigur zu bewegen, den Bildschirm zu drehen usw.

Das Grundprinzip hinter der Fähigkeit des Joysticks, dem Computer unsere Aktionen mitzuteilen, ist sehr einfach. Man kann ihn sich als zwei senkrecht zueinander stehende Potentiometer vorstellen. Diese beiden Potentiometer messen den analogen Wert des Joysticks in vertikaler und horizontaler Richtung, was in einem Wert (x,y) in einem ebenen rechtwinkligen Koordinatensystem resultiert.

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

- *Joystick-Modul*

**Benötigte Komponenten**

Für dieses Projekt benötigen wir folgende Komponenten.

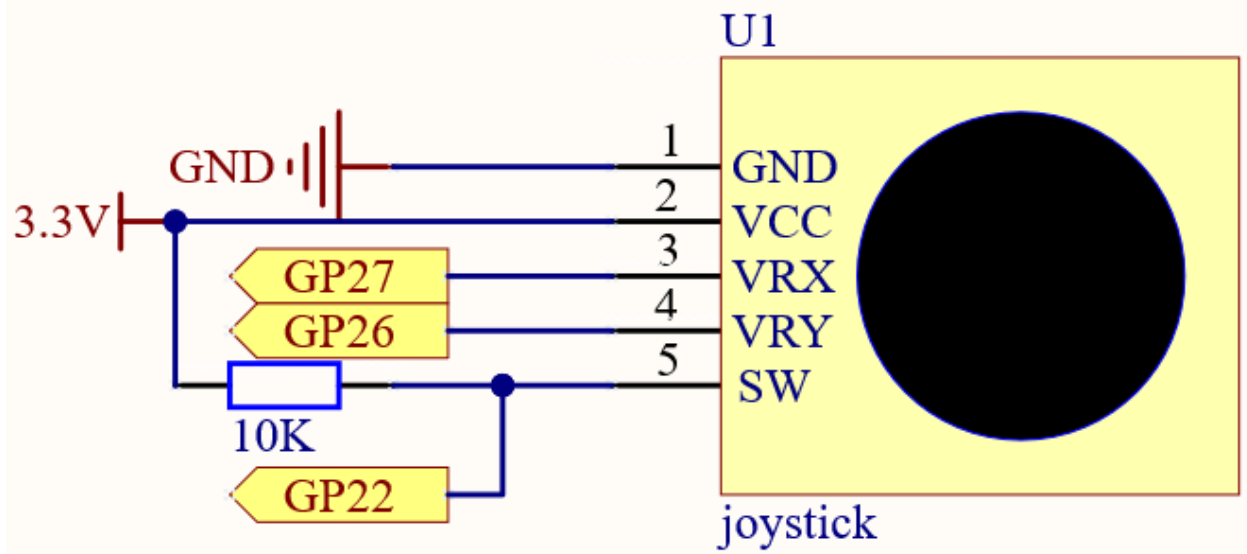
Es ist definitiv praktisch, ein ganzes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Set	450+	

Sie können die Teile auch einzeln über die untenstehenden Links kaufen.

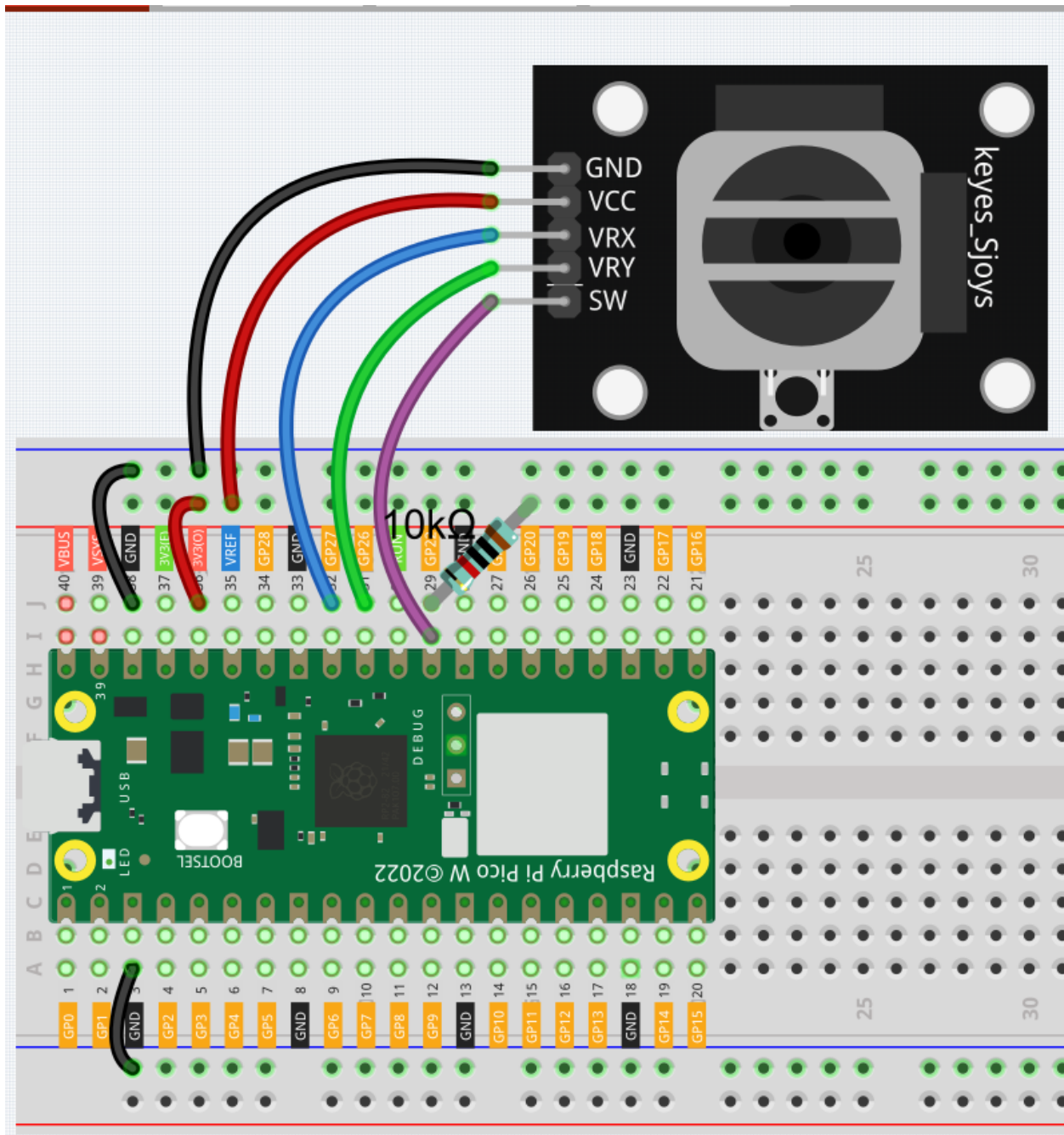
SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1 (10K)	
6	<i>Joystick-Modul</i>	1	

**Schaltplan**



Der SW-Pin ist mit einem 10K Pull-up-Widerstand verbunden, um ein stabiles High-Level am SW-Pin (Z-Achse) zu erhalten, wenn der Joystick nicht gedrückt ist; sonst wäre der SW in einem unbestimmten Zustand und der Ausgangswert könnte zwischen 0/1 variieren.

#### Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `4.1_toggle_the_joystick.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken dann auf „Aktuelles Skript ausführen“ oder drücken einfach F5.
- Vergessen Sie nicht, den Interpreter „MicroPython (Raspberry Pi Pico)“ in der unteren rechten Ecke auszuwählen.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
import utime

x_joystick = machine.ADC(27)
y_joystick = machine.ADC(26)
z_switch = machine.Pin(22,machine.Pin.IN)

while True:
    x_value = x_joystick.read_u16()
    y_value = y_joystick.read_u16()
    z_value = z_switch.value()
    print(x_value,y_value,z_value)
    utime.sleep_ms(200)
```

Nachdem das Programm ausgeführt wurde, gibt die Shell die Werte x, y und z des Joysticks aus.

- Die Werte der x- und y-Achse sind analoge Werte, die zwischen 0 und 65535 variieren.
- Die Z-Achse ist ein digitaler Wert mit einem Status von 1 oder 0.

## 4.31 4.2 4x4 Tastenfeld

Das 4x4 Tastenfeld, auch Matrix-Tastenfeld genannt, besteht aus einer Anordnung von 16 Tasten in einer einzigen Einheit.

Solche Tastenfelder findet man hauptsächlich in Geräten, die digitale Eingaben erfordern, wie Taschenrechner, Fernbedienungen, Tastentelefone, Verkaufsautomaten, Geldautomaten, Zahlenschlösser und digitale Türschlösser.

In diesem Projekt lernen wir, wie man ermittelt, welche Taste gedrückt wurde und den entsprechenden Tastenwert erhält.

- [\*4x4 Tastenfeld\*](#)
- [E.161 – Wikipedia](#)

### Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

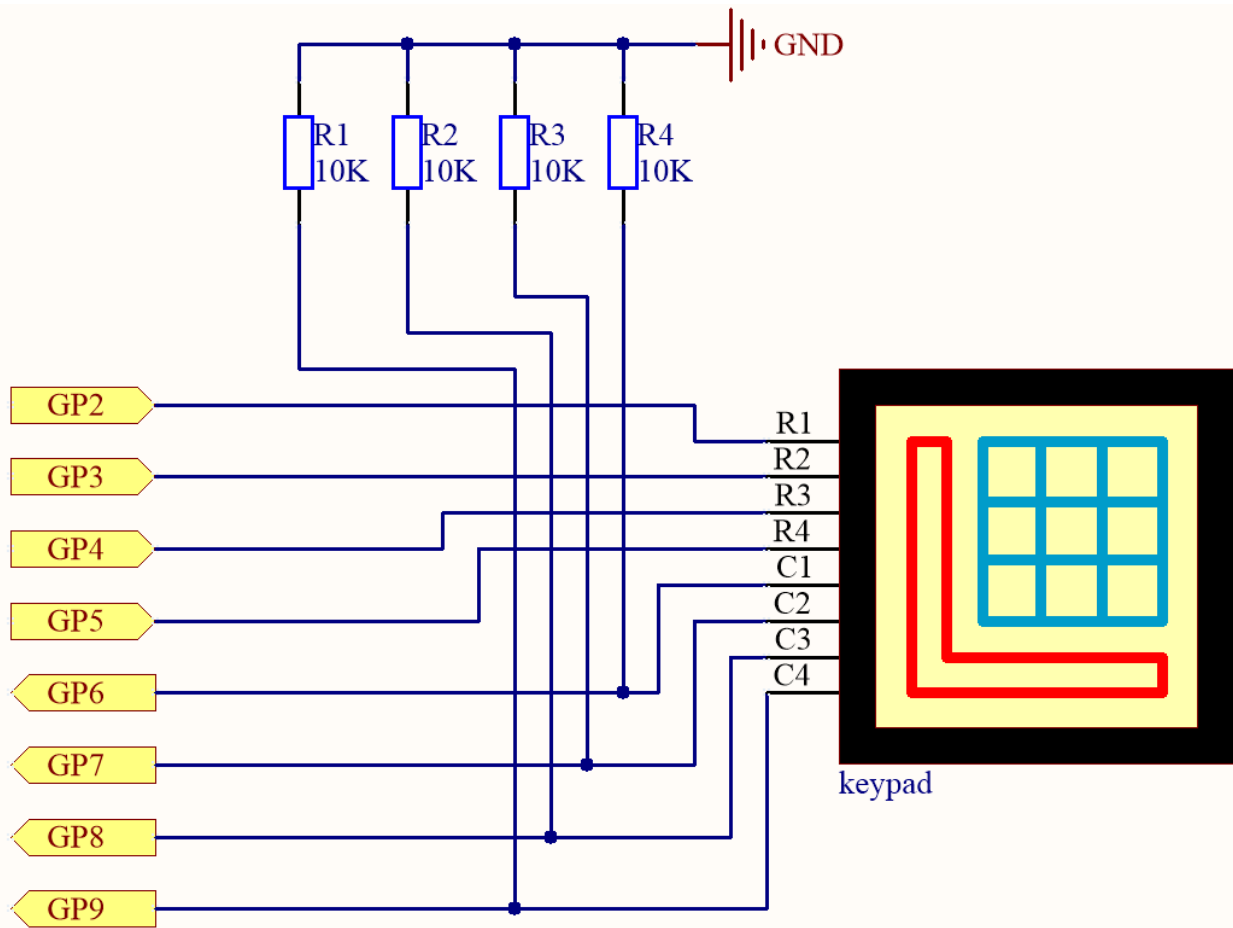
Ein Komplettsset ist definitiv praktisch. Hier ist der Link:

Bezeichnung	ARTIKEL IM SET	LINK
Kepler Set	450+	

Die einzelnen Komponenten können auch über die untenstehenden Links erworben werden.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	4(10K)	
6	<i>4x4 Tastenfeld</i>	1	

### Schaltplan



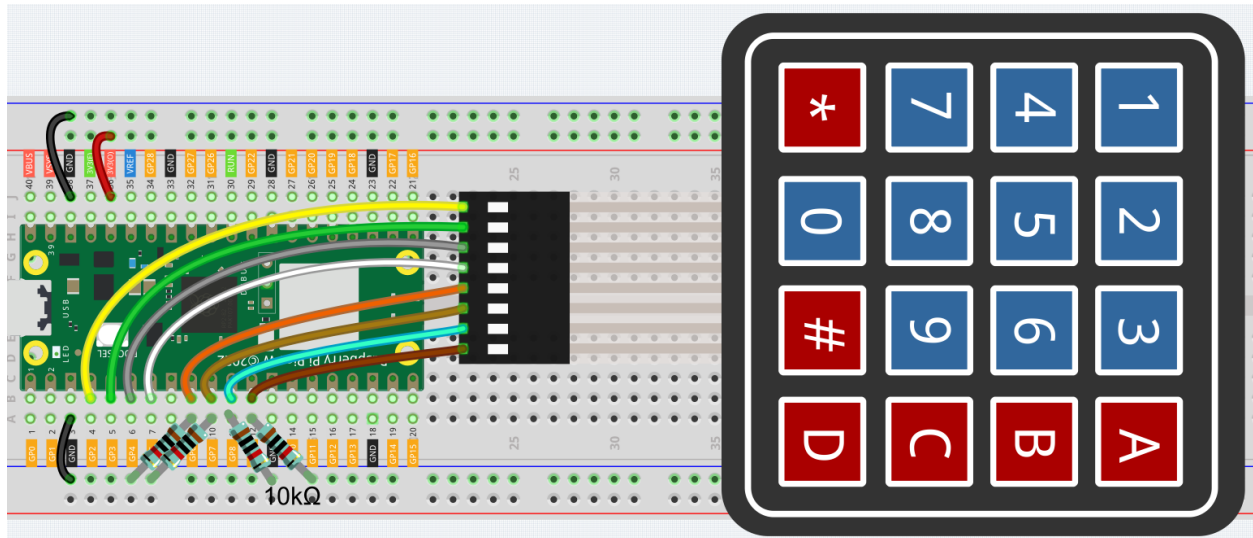
An jede der vier Spalten des Matrix-Tastenfelds ist ein Pull-Down-Widerstand angeschlossen, sodass die Anschlüsse G6 bis G9 auf einem stabilen niedrigen Pegel sind, wenn keine Taste gedrückt ist.

Die Reihen des Tastenfelds (G2 bis G5) sind so programmiert, dass sie auf High gehen; wird einer der Anschlüsse G6 bis G9 als High gelesen, wissen wir, welche Taste gedrückt wurde.

Zum Beispiel, wenn G6 als High gelesen wird, dann wurde die Nummerntaste 1 gedrückt; denn die Steuere pins der Nummerntaste 1 sind G2 und G6, wenn die Taste 1 gedrückt wird, werden G2 und G6 miteinander verbunden und G6

ist ebenfalls High.

## Verdrahtung



Um die Verdrahtung zu vereinfachen, sind im obigen Schaltplan die Reihen und Spalten des Matrix-Tastenfelds sowie die 10K-Widerstände gleichzeitig in die Löcher eingesteckt, in denen sich G6 bis G9 befinden.

## Code

### Bemerkung:

- Öffnen Sie die Datei 4\_2\_4x4\_keypad.py im Verzeichnis kepler-kit-main/micropython oder kopieren Sie den Code in Thonny, und klicken Sie auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, im rechten unteren Eck den „MicroPython (Raspberry Pi Pico)“-Interpreter auszuwählen.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
import time

characters = [["1", "2", "3", "A"], ["4", "5", "6", "B"], ["7", "8", "9", "C"], ["*", "0", "#", "D"]]

pin = [2, 3, 4, 5]
row = []
for i in range(4):
    row.append(None)
    row[i] = machine.Pin(pin[i], machine.Pin.OUT)

pin = [6, 7, 8, 9]
col = []
for i in range(4):
    col.append(None)
    col[i] = machine.Pin(pin[i], machine.Pin.IN)

def readKey():
    key = []
```

(Fortsetzung auf der nächsten Seite)



(Fortsetzung der vorherigen Seite)

```

    for i in range(4):
        row[i].high()
        for j in range(4):
            if(col[j].value() == 1):
                key.append(characters[i][j])
        row[i].low()
    if key == [] :
        return None
    else:
        return key

last_key = None
while True:
    current_key = readKey()
    if current_key == last_key:
        continue
    last_key = current_key
    if current_key != None:
        print(current_key)
    time.sleep(0.1)

```

Nachdem das Programm ausgeführt wurde, wird die Shell die Tasten ausgeben, die Sie auf dem Keypad gedrückt haben.

#### Funktionsweise

```

import machine
import time

characters = [
    ["1", "2", "3", "A"], ["4", "5", "6", "B"], ["7", "8", "9", "C"], ["*", "0", "#", "D"]
]

pin = [2, 3, 4, 5]
row = []
for i in range(4):
    row.append(None)
    row[i] = machine.Pin(pin[i], machine.Pin.OUT)

pin = [6, 7, 8, 9]
col = []
for i in range(4):
    col.append(None)
    col[i] = machine.Pin(pin[i], machine.Pin.IN)

```

Definiert jede Taste der Matrix-Tastatur im Array characters[] und legt die Pins für jede Reihe und Spalte fest.

```

last_key = None
while True:
    current_key = readKey()
    if current_key == last_key:
        continue
    last_key = current_key
    if current_key != None:
        print(current_key)
    time.sleep(0.1)

```

Dies ist der Teil der Hauptfunktion, der den Wert der gedrückten Taste liest und ausgibt.

Die Funktion `readKey()` liest den Zustand jeder Taste aus.

Die Anweisungen `if current_key != None` und `if current_key == last_key` dienen dazu, festzustellen, ob eine Taste gedrückt ist und wie der Zustand der gedrückten Taste ist. (Wenn Sie beispielsweise ,3‘ drücken, während Sie ,1‘ drücken, ist die Bewertung gültig.)

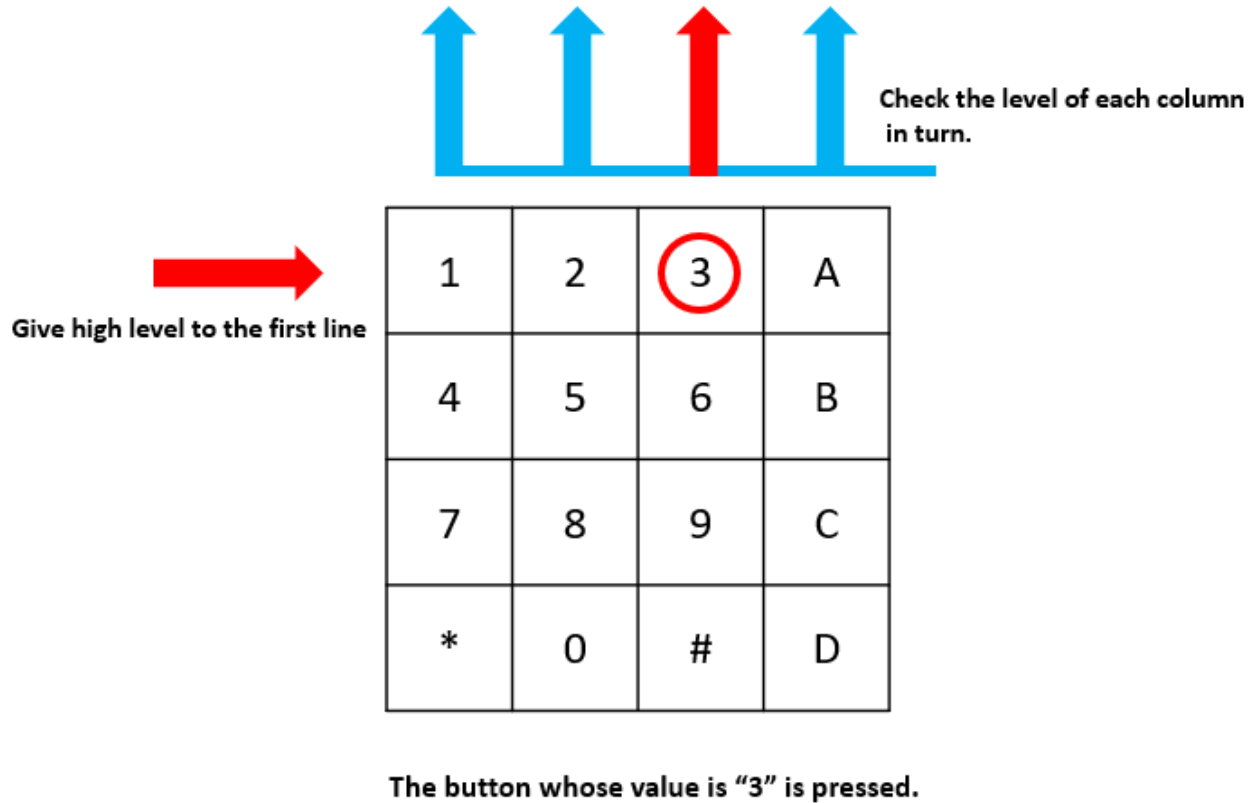
Gibt den Wert der aktuell gedrückten Taste aus, wenn die Bedingung gültig ist.

Die Anweisung `last_key = current_key` speichert den Zustand jeder Auswertung in einem Array `last_key`, um die nächste Runde der bedingten Bewertung zu erleichtern.

```
def readKey():
    key = []
    for i in range(4):
        row[i].high()
        for j in range(4):
            if(col[j].value() == 1):
                key.append(characters[i][j])
        row[i].low()
    if key == [] :
        return None
    else:
        return key
```

Diese Funktion setzt jede Reihe der Matrix-Tastatur nacheinander auf ein hohes Niveau. Wenn eine Taste gedrückt wird, erhält die entsprechende Spalte ein hohes Niveau. Nach Durchlaufen der zweistufigen Schleife wird der Wert der Taste, deren Zustand 1 ist, im Array `key` gespeichert.

Wenn Sie die Taste ,3‘ drücken:



`row[0]` wird auf ein hohes Niveau gesetzt und `col[2]` erhält ebenfalls ein hohes Niveau.

`col[0]`, `col[1]`, `col[3]` erhalten ein niedriges Niveau.

Es gibt vier Zustände: 0, 0, 1, 0; und wir schreiben ,3' in `pressed_keys`.

Wenn `row[1]`, `row[2]`, `row[3]` auf ein hohes Niveau gesetzt werden, erhalten `col[0] ~ col[4]` ein niedriges Niveau.

Die Schleife stoppt und gibt `key = ,3'` zurück.

Wenn Sie die Tasten ,1' und ,3' drücken, wird `key = [,1', '3']` zurückgegeben.

## 4.32 4.3 Elektroden-Tastatur

Der MPR121 ist eine hervorragende Option, wenn Sie eine Vielzahl von Berührungsschaltern in Ihr Projekt integrieren möchten. Er verfügt über Elektroden, die mit Leitern erweitert werden können. Wenn Sie die Elektroden an einer Banane befestigen, können Sie diese in einen Berührungsschalter verwandeln.

- *MPR121 Modul*

### Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

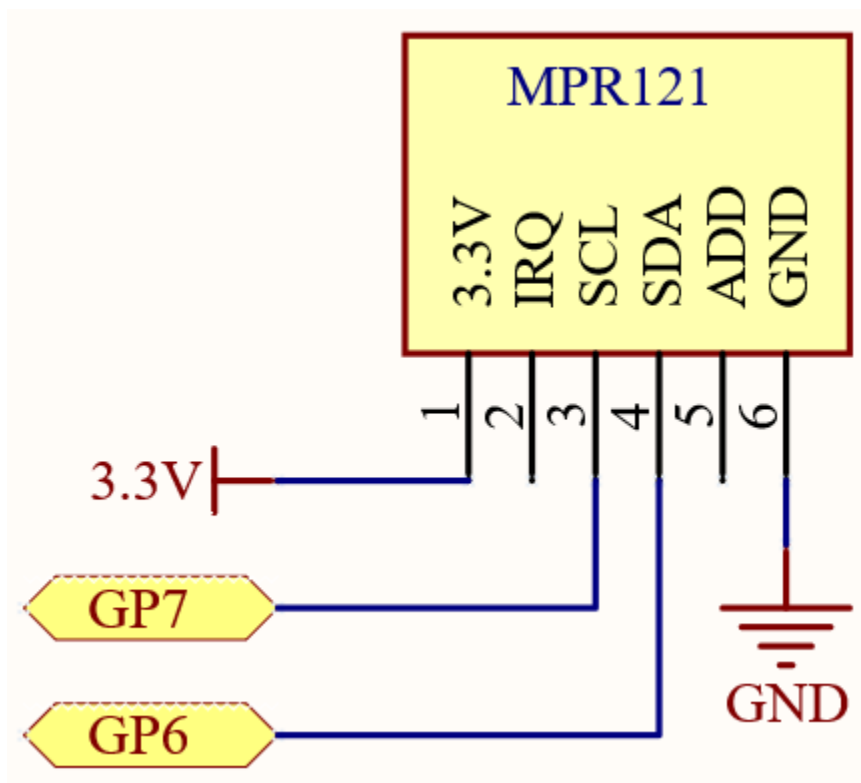
Es ist durchaus praktisch, gleich ein ganzes Kit zu kaufen, hier ist der Link:

Bezeichnung	ARTIKEL IM KIT	LINK
Kepler-Kit	450+	

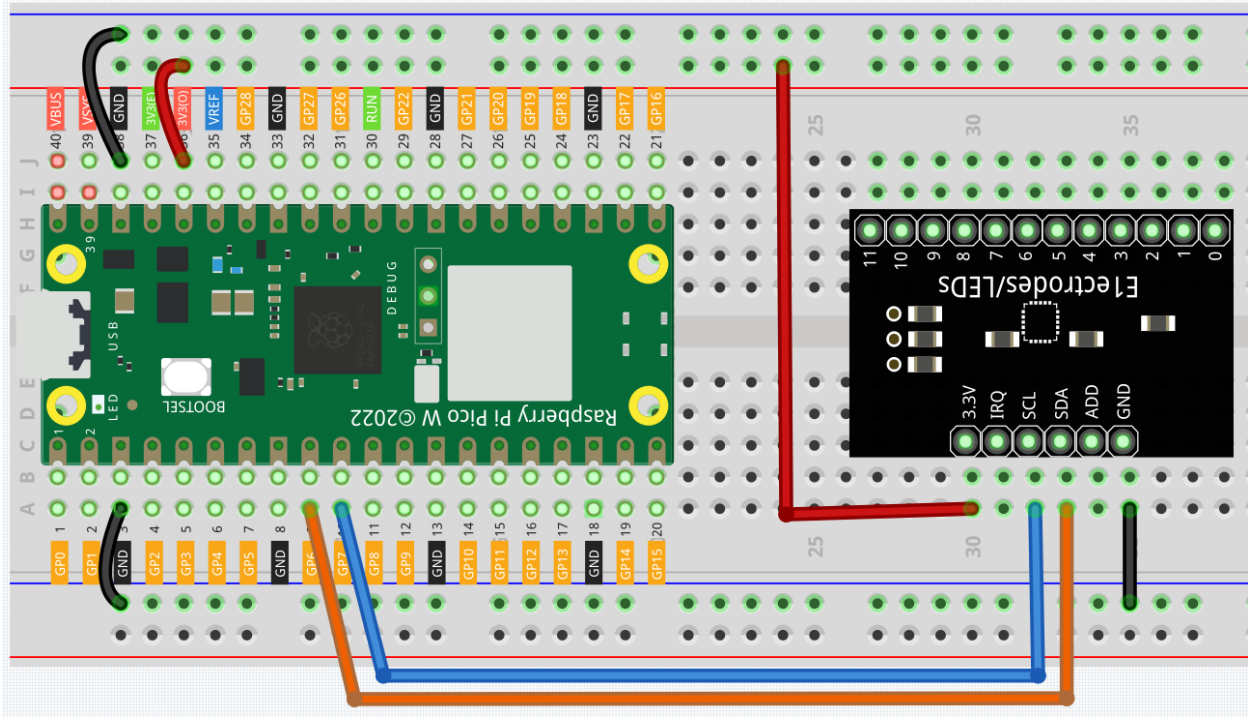
Sie können die Komponenten auch einzeln über die unten stehenden Links beziehen.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>MPR121 Modul</i>	1	

### Schaltplan



### Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `4.3_electrode_keyboard.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Anschließend klicken Sie auf „Aktuelles Skript ausführen“ oder drücken einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
- Für detaillierte Anleitungen verweisen wir auf *Code direkt öffnen und ausführen*.
- Sie benötigen die Bibliothek `mpr121.py`. Prüfen Sie, ob diese bereits auf dem Pico W hochgeladen wurde. Eine ausführliche Anleitung finden Sie unter *1.4 Bibliotheken auf den Pico hochladen*.

```
from mpr121 import MPR121
from machine import Pin, I2C
import time

i2c = I2C(1, sda=Pin(6), scl=Pin(7))
mpr = MPR121(i2c)

# Überprüfen aller Tasten
while True:
    value = mpr.get_all_states()
    if len(value) != 0:
        print(value)
        time.sleep_ms(100)
```

Nachdem das Programm ausgeführt wurde, können Sie die zwölf Elektroden am MPR121 berühren, und die berührten

Elektroden werden ausgegeben.

Sie können die Elektroden erweitern, um andere Leiter wie Obst, Draht, Folie usw. anzuschließen. Dies eröffnet Ihnen weitere Möglichkeiten zur Auslösung dieser Elektroden.

### Funktionsweise?

In der mpr121-Bibliothek haben wir die Funktionalität in die Klasse MPR121 integriert.

```
from mpr121 import MPR121
```

Der MPR121 ist ein I2C-Modul, für dessen Initialisierung ein Satz I2C-Pins definiert werden muss. Zu diesem Zeitpunkt werden die Anfangszustände der Elektroden des Moduls aufgezeichnet. Falls die Elektroden erweitert werden, muss das Beispiel erneut ausgeführt werden, um die Anfangswerte zurückzusetzen.

```
from machine import Pin, I2C
i2c = I2C(1, sda=Pin(6), scl=Pin(7))
mpr = MPR121(i2c)
```

- [Inter-Integrated Circuit - Wikipedia](#)

Verwenden Sie dann `mpr.get_all_states()` um zu lesen, ob die Elektroden ausgelöst werden. Wenn die Elektroden 2 und 3 ausgelöst werden, wird der Wert `[2, 3]` erzeugt.

```
while True:
    value = mpr.get_all_states()
    if len(value) != 0:
        print(value)
        time.sleep_ms(100)
```

Sie können auch `mpr.is_touched(electrode)` verwenden, um eine bestimmte Elektrode zu überprüfen. Wenn sie ausgelöst wird, gibt die Methode `True` zurück, andernfalls `False`.

```
while True:
    value = mpr.is_touched(0)
    print(value)
    time.sleep_ms(100)
```

## 5. Mikrochip

### 4.33 5.1 Mikrochip - 74HC595

Ein integrierter Schaltkreis (integrated circuit, kurz IC) ist eine Art Miniatur-Elektronikbauelement oder -komponente, das in Schaltkreisen durch das Kürzel „IC“ dargestellt wird.

Mit Hilfe eines speziellen Verfahrens werden Transistoren, Widerstände, Kondensatoren, Spulen und andere für einen Schaltkreis erforderliche Bauelemente und Verbindungen auf einem oder mehreren kleinen Halbleiter-Wafern oder dielektrischen Substraten angefertigt. Anschließend werden diese in ein Gehäuse eingebettet und ergeben somit eine Mikrostruktur mit den erforderlichen Schaltkreisfunktionen. Alle Bauteile sind als eine Einheit strukturiert, was einen großen Schritt in Richtung Miniaturisierung, geringem Stromverbrauch, Intelligenz und hoher Zuverlässigkeit von elektronischen Bauelementen bedeutet.

Die Erfinder der integrierten Schaltkreise sind Jack Kilby (integrierte Schaltkreise auf Germaniumbasis (Ge)) und Robert Norton Noyce (integrierte Schaltkreise auf Siliziumbasis (Si)).

Dieses Kit enthält einen IC, den 74HC595, der die Nutzung von GPIO-Pins erheblich reduziert. Konkret kann er 8 Pins für die digitale Signalausgabe ersetzen, indem ein 8-Bit-Binärzahl geschrieben wird.

- [Binärzahl – Wikipedia](#)
- [74HC595](#)

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

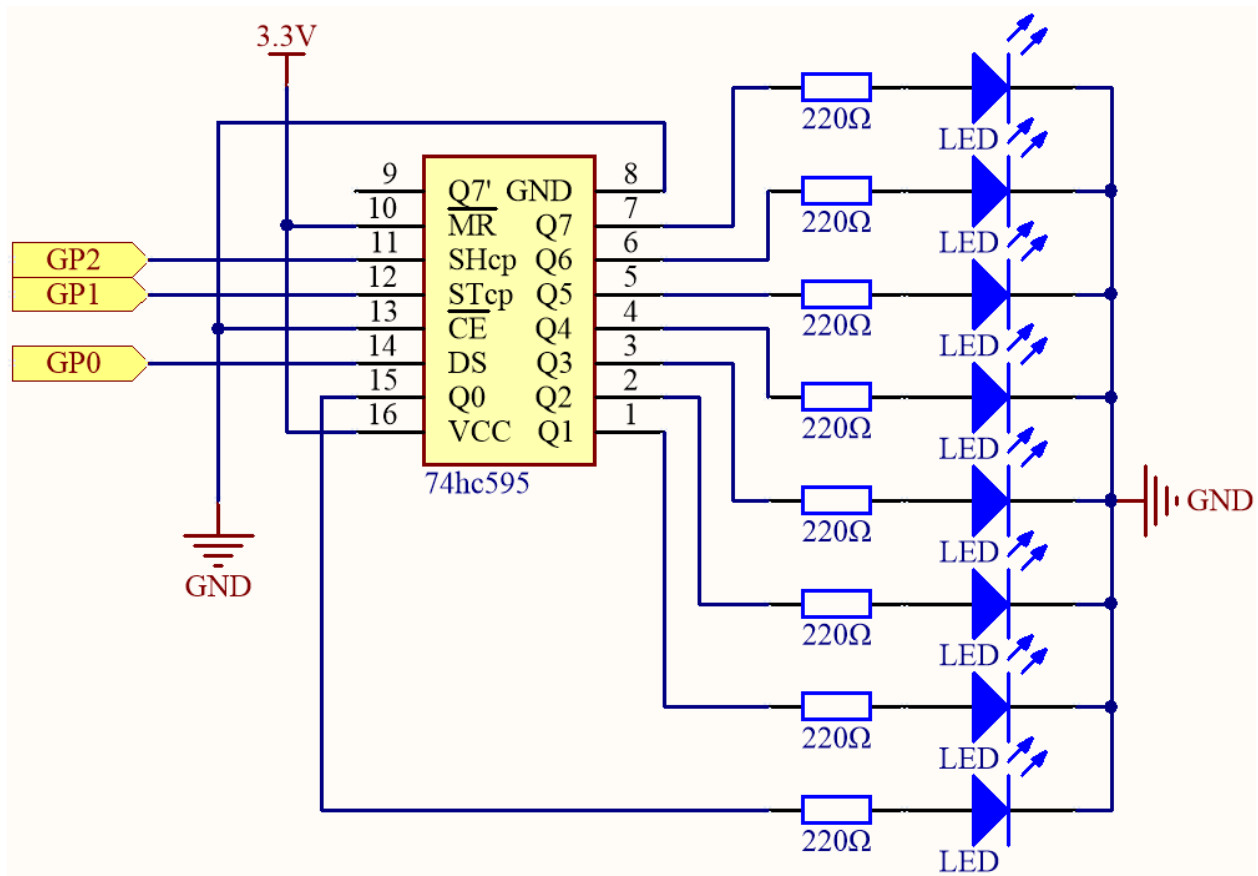
Es ist definitiv praktisch, ein komplettes Kit zu kaufen, hier ist der Link:

Bezeichnung	KOMPONENTEN IN DIESEM KIT	LINK
Kepler-Kit	450+	

Sie können die Komponenten auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	MEN-GE	LINK
1	<a href="#">Raspberry Pi Pico W</a>	1	
2	Micro-USB-Kabel	1	
3	<a href="#">Steckbrett</a>	1	
4	<a href="#">Jumperkabel</a>	Mehrere	
5	<a href="#">Widerstand</a>	8 (220)	
6	<a href="#">LED</a>	8	
7	<a href="#">74HC595</a>	1	

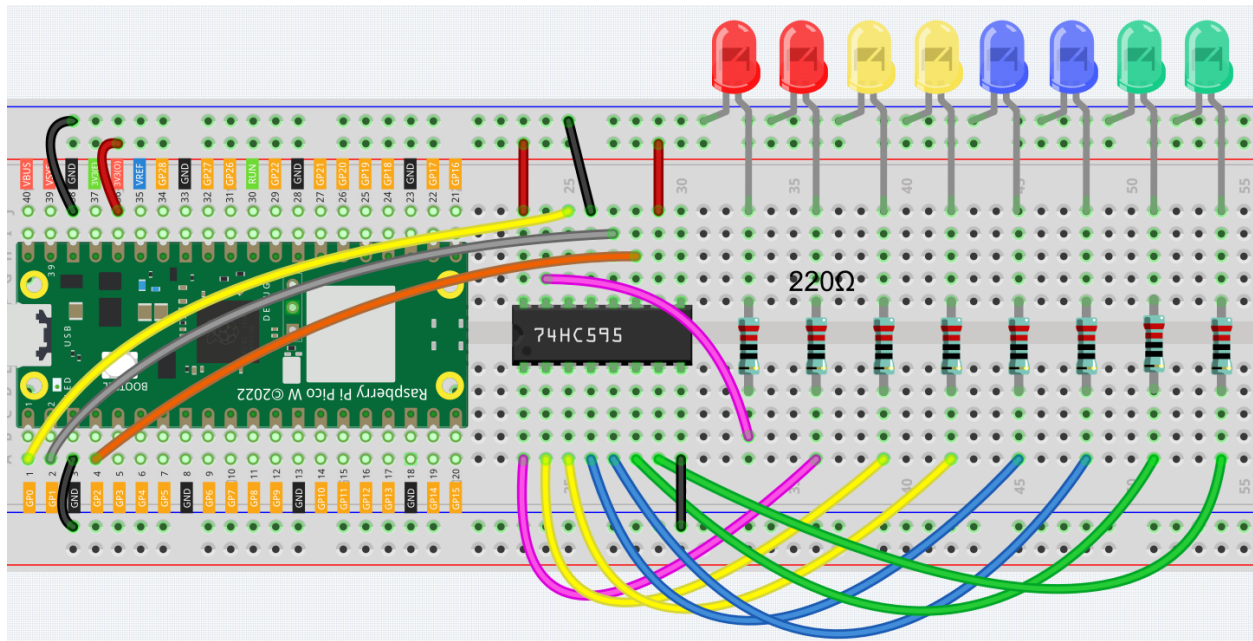
### Schaltplan



- Wenn MR (Pin 10) auf hohem und OE (Pin 13) auf niedrigem Pegel ist, wird das Daten in der aufsteigenden Flanke von SHcp eingegeben und durch die aufsteigende Flanke von SHcp ins Speicherregister übertragen.
- Sind die beiden Taktgeber verbunden, ist das Schieberegister immer einen Puls vor dem Speicherregister.
- Im Speicherregister gibt es einen seriellen Eingangspin (Ds), einen seriellen Ausgangspin (Q) und eine asynchrone Zurücksetztaste (Low-Pegel).
- Das Speicherregister gibt einen Bus mit einem parallelen 8-Bit und in drei Zuständen aus.
- Ist OE aktiviert (Low-Pegel), werden die Daten im Speicherregister auf den Bus (Q0 ~ Q7) ausgegeben.

### Verkabelung





## Code

### Bemerkung:

- Öffnen Sie die Datei 5.1\_microchip\_74hc595.py im Pfad kepler-kit-main/micropython oder kopieren Sie diesen Code in Thonny und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, im rechten unteren Eck den Interpreter „MicroPython (Raspberry Pi Pico)“ auszuwählen.
- Für detaillierte Informationen über die Funktionsweise des Codes und die zu verwendenden Bibliotheken, verweisen wir auf den Kommentarbereich im Code.

```
import machine
import time

sdi = machine.Pin(0,machine.Pin.OUT)
rclk = machine.Pin(1,machine.Pin.OUT)
srclk = machine.Pin(2,machine.Pin.OUT)

def hc595_shift(dat):
    rclk.low()
    time.sleep_ms(5)
    for bit in range(7, -1, -1):
        srclk.low()
        time.sleep_ms(5)
        value = 1 & (dat >> bit)
        sdi.value(value)
        time.sleep_ms(5)
        srclk.high()
        time.sleep_ms(5)
    time.sleep_ms(5)
    rclk.high()
    time.sleep_ms(5)
```

(Fortsetzung auf der nächsten Seite)

```

num = 0

for i in range(16):
    if i < 8:
        num = (num<<1) + 1
    elif i>=8:
        num = (num & 0b01111111)<<1
    hc595_shift(num)
    print("{:0>8b}".format(num))
    time.sleep_ms(200)

```

Wenn das Programm läuft, wird num als achtbitige Binärzahl in den 74HC595-Chip geschrieben, um das Ein- und Ausschalten der 8 LEDs zu steuern. Den aktuellen Wert von num können wir im Shell-Fenster einsehen.

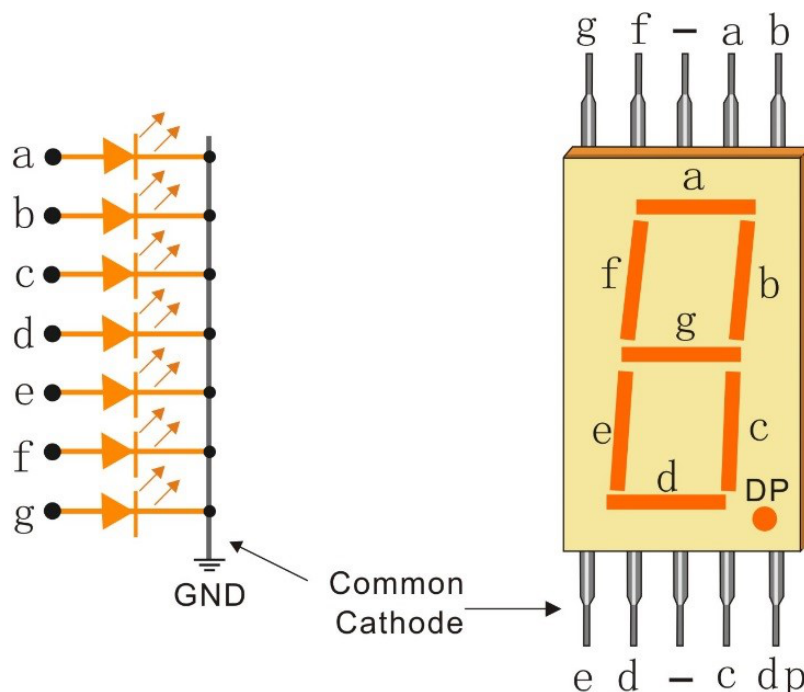
#### Wie funktioniert es?

Die Funktion `hc595_shift()` bewirkt, dass der 74HC595 acht digitale Signale ausgibt. Dabei wird das letzte Bit der Binärzahl an Q0 und das erste Bit an Q7 ausgegeben. Mit anderen Worten: Schreibt man die Binärzahl „00000001“, so gibt Q0 ein hohes Signal aus, während Q1 bis Q7 ein niedriges Signal ausgeben.

## 4.34 5.2 Nummernanzeige

7-Segment-Anzeigen begegnen uns im Alltag an vielen Stellen. Sie finden beispielsweise in Klimaanlage Verwendung zur Temperaturanzeige oder in Verkehrsleitsystemen als Timer.

Im Grunde genommen handelt es sich bei einer 7-Segment-Anzeige um ein aus 8 LEDs bestehendes Paket, von denen 7 in Streifenform ein „8“ bilden, während eine etwas kleinere punktförmige LED als Dezimalpunkt dient. Diese LEDs sind als a, b, c, d, e, f, g und dp markiert. Sie haben eigene Anodenstifte und teilen sich Kathoden. Ihre Stiftpositionen sind in der untenstehenden Abbildung dargestellt.



Das bedeutet, dass zur vollständigen Steuerung 8 digitale Signale gleichzeitig benötigt werden, und der 74HC595 kann dies leisten.

- *7-Segment-Anzeige*

### **Benötigte Komponenten**

Für dieses Projekt benötigen wir die folgenden Komponenten.

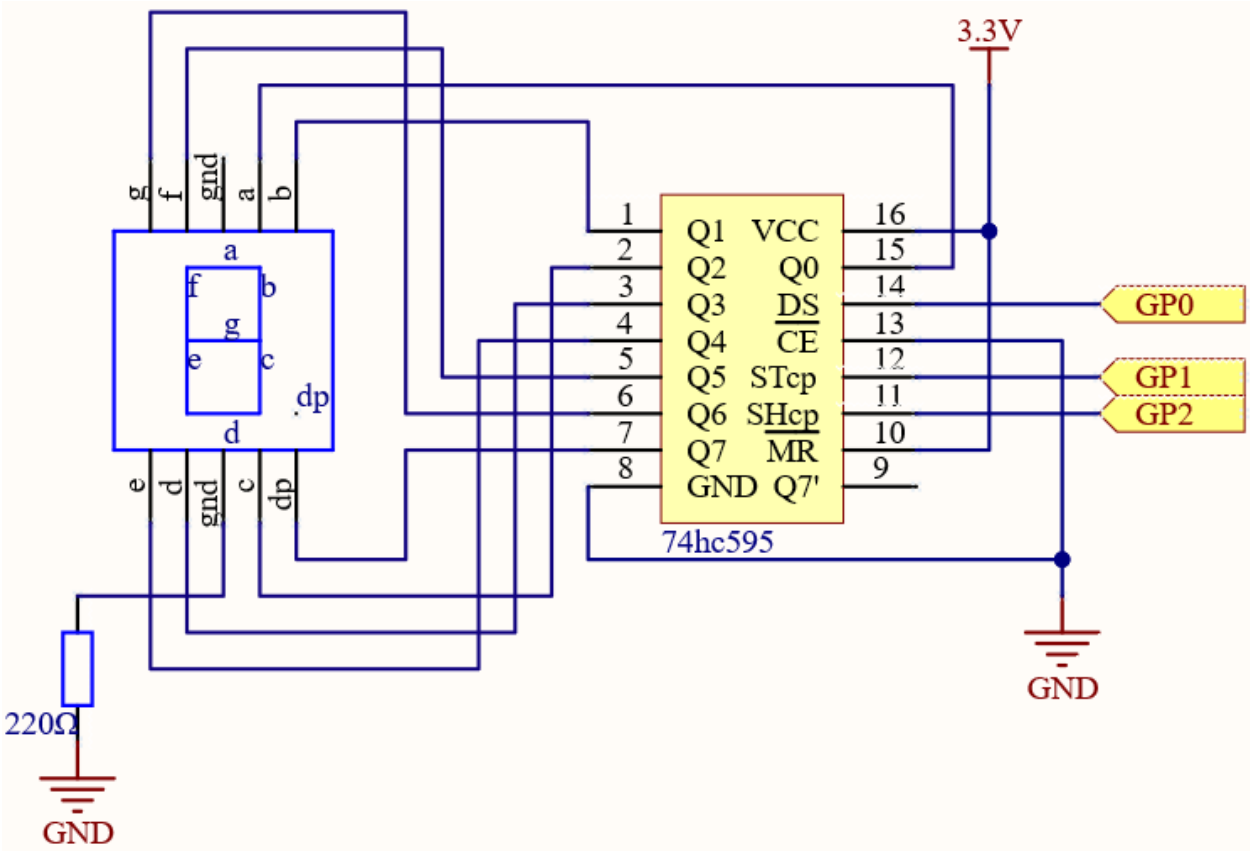
Ein Komplettsset ist natürlich besonders praktisch, hier der Link:

Bezeichnung	TEILE IM SET	LINK
Kepler-Kit	450+	

Die Komponenten können natürlich auch einzeln über die untenstehenden Links erworben werden.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1 (220)	
6	<i>7-Segment-Anzeige</i>	1	
7	<i>74HC595</i>	1	

### **Schaltplan**

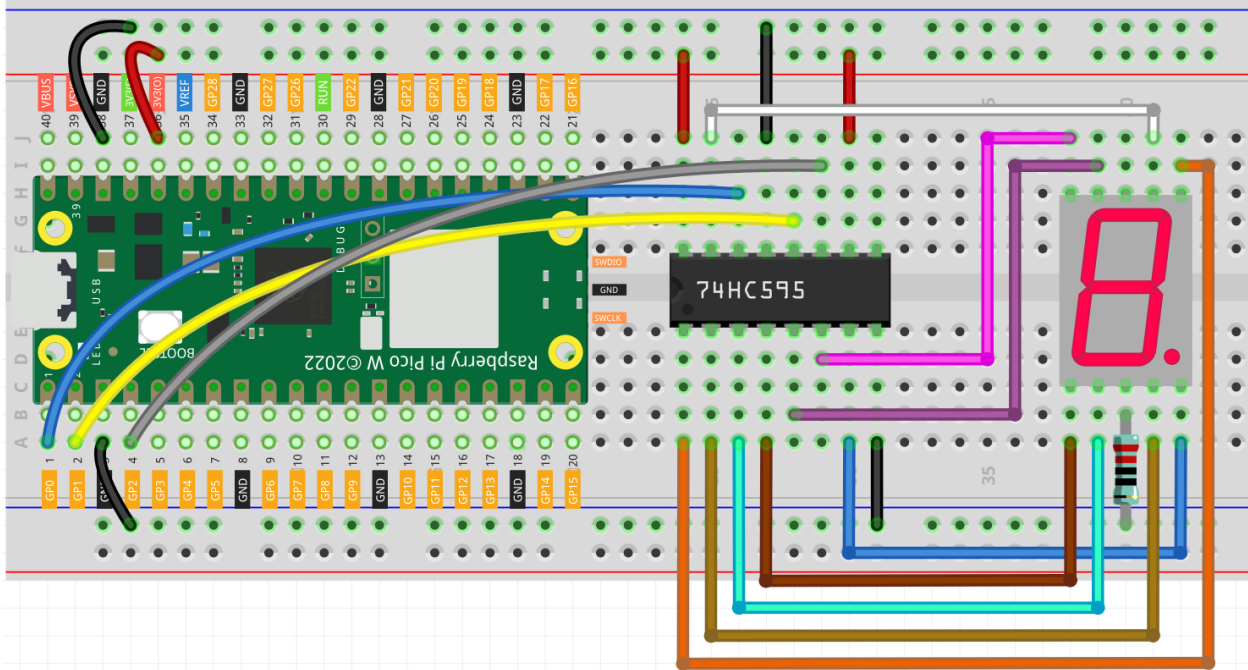


Hier ist das Verdrahtungsprinzip im Wesentlichen das gleiche wie bei [5.1 Mikrochip - 74HC595](#). Der einzige Unterschied besteht darin, dass Q0-Q7 an die Pins a ~ g der 7-Segment-Anzeige angeschlossen sind.

Tab. 1: Verdrahtung

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

**Verdrahtung**



## Code

### Bemerkung:

- Öffnen Sie die Datei `5.2_number_display.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5, um es auszuführen.
- Vergessen Sie nicht, im unteren rechten Eck den Interpreter „MicroPython (Raspberry Pi Pico)“ auszuwählen.
- Für detaillierte Anleitungen verweisen wir auf [Code direkt öffnen und ausführen](#).

```
import machine
import time

SEGCODE = [0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f]

sdi = machine.Pin(0, machine.Pin.OUT)
rclk = machine.Pin(1, machine.Pin.OUT)
srclk = machine.Pin(2, machine.Pin.OUT)

def hc595_shift(dat):
    rclk.low()
    time.sleep_ms(5)
    for bit in range(7, -1, -1):
        srclk.low()
        time.sleep_ms(5)
        value = 1 & (dat >> bit)
        sdi.value(value)
        time.sleep_ms(5)
        srclk.high()
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

    time.sleep_ms(5)
    time.sleep_ms(5)
    rclk.high()
    time.sleep_ms(5)

while True:
    for num in range(10):
        hc595_shift(SEGCODE[num])
        time.sleep_ms(500)

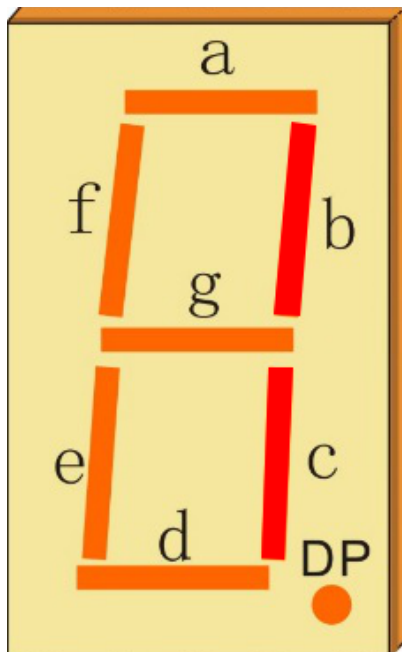
```

Während das Programm läuft, werden die Zahlen 0 bis 9 sequenziell auf der LED-Segmentanzeige dargestellt.

### Wie funktioniert das?

hc595\_shift() wird 74HC595 dazu bringen, 8 digitale Signale auszugeben. Es gibt das letzte Bit der Binärzahl an Q0 aus und das Ausgangssignal des ersten Bits an Q7. Mit anderen Worten, wenn die Binärzahl „00000001“ geschrieben wird, gibt Q0 ein hohes Signal aus und Q1~Q7 geben niedrige Signale aus.

Angenommen, die 7-Segment-Anzeige zeigt die Zahl „1“ an, müssen wir ein hohes Signal für b und c schreiben und ein niedriges Signal für a, d, e, f, g und dg schreiben.



Das bedeutet, die Binärzahl „00000110“ muss geschrieben werden. Zur besseren Lesbarkeit verwenden wir die hexadezimale Schreibweise „0x06“.

- [Hexadezimal](#)
- [Binär-Hex-Konverter](#)

In gleicher Weise können auch andere Zahlen auf der LED-Segmentanzeige dargestellt werden. Die nachfolgende Tabelle zeigt die entsprechenden Codes.

Tab. 2: Glyph-Code

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

Geben Sie diese Codes in `hc595_shift()` ein, um die entsprechenden Zahlen auf der LED-Segmentanzeige darzustellen.

## 4.35 5.3 Zeitmesser

Ein 4-stelliges 7-Segment-Display besteht aus vier zusammenarbeitenden 7-Segment-Anzeigen.

Das 4-stellige 7-Segment-Display arbeitet unabhängig voneinander. Es nutzt das Prinzip der visuellen Persistenz des menschlichen Auges, um die Zeichen jedes 7-Segments schnell in einer Schleife anzuzeigen und so fortlaufende Zeichenfolgen zu bilden.

Zum Beispiel, wenn „1234“ auf dem Display angezeigt wird, erscheint „1“ auf dem ersten 7-Segment, während „234“ nicht angezeigt wird. Nach einer kurzen Zeit zeigt das zweite 7-Segment „2“, die 1., 3. und 4. 7-Segment-Anzeige bleiben aus, und so weiter. Dieser Vorgang ist sehr kurz (typischerweise 5ms), und durch den optischen Nachleuchteffekt sowie das Prinzip der visuellen Persistenz sehen wir alle vier Zeichen gleichzeitig.

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

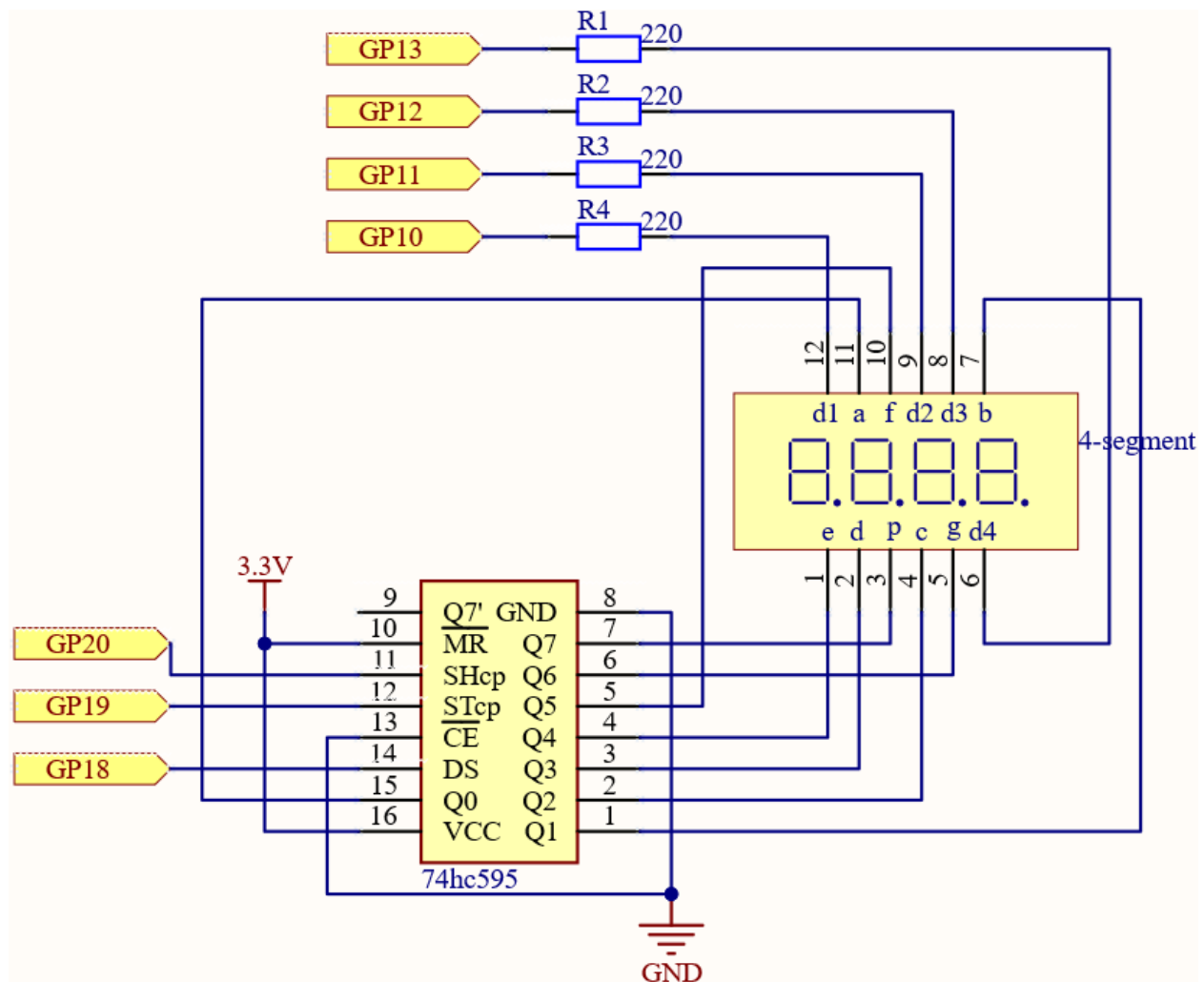
Ein komplettes Kit ist sicherlich praktisch, hier ist der Link dazu:

Bezeichnung	ARTIKEL IN DIESEM KIT	LINK
Kepler Kit	450+	

Sie können die Teile auch separat über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	mehrere	
5	<i>Widerstand</i>	4 (220)	
6	<i>4-stellige 7-Segment-Anzeige</i>	1	
7	<i>74HC595</i>	1	

### Schaltplan

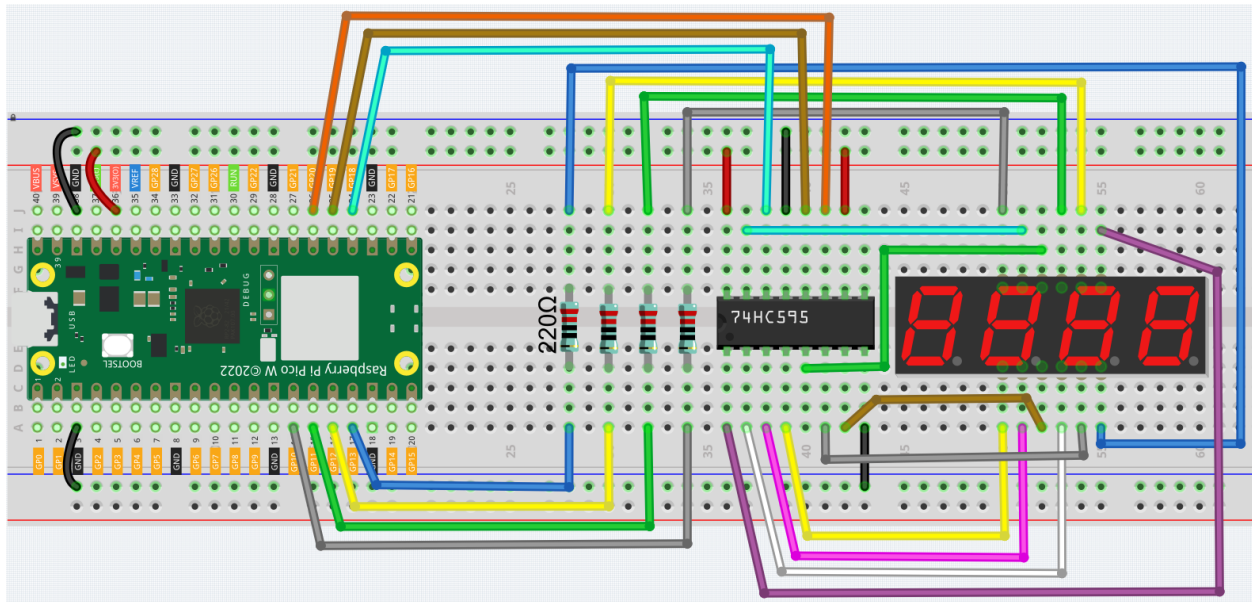


Der Verdrahtungsansatz ist im Grunde der gleiche wie bei [5.1 Mikrochip - 74HC595](#), der einzige Unterschied besteht darin, dass die Pins Q0-Q7 an die a ~ g Pins des 4-stelligen 7-Segment-Displays angeschlossen sind.

Anschließend wählen G10 ~ G13 aus, welches 7-Segment-Display aktiv sein soll.



## Verdrahtung



## Code

## Bemerkung:

- Öffnen Sie die Datei `5.3_time_counter.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
- Für detaillierte Anleitungen beziehen Sie sich bitte auf [Code direkt öffnen und ausführen](#).

```
import machine
import time

SEGCODE = [0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f]

sdi = machine.Pin(18,machine.Pin.OUT)
rclk = machine.Pin(19,machine.Pin.OUT)
srclk = machine.Pin(20,machine.Pin.OUT)

placePin = []
pin = [10,13,12,11]
for i in range(4):
    placePin.append(None)
    placePin[i] = machine.Pin(pin[i], machine.Pin.OUT)

timerStart=time.ticks_ms()

def timer1():
    return int((time.ticks_ms()-timerStart)/1000)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

def pickDigit(digit):
    for i in range(4):
        placePin[i].value(1)
    placePin[digit].value(0)

def clearDisplay():
    hc595_shift(0x00)

def hc595_shift(dat):
    rclk.low()
    time.sleep_us(200)
    for bit in range(7, -1, -1):
        srclk.low()
        time.sleep_us(200)
        value = 1 & (dat >> bit)
        sdi.value(value)
        time.sleep_us(200)
        srclk.high()
        time.sleep_us(200)
    time.sleep_us(200)
    rclk.high()
    time.sleep_us(200)

while True:
    count = timer1()
    #print(count)

    pickDigit(0)
    hc595_shift(SEGCODE[count%10])

    pickDigit(1)
    hc595_shift(SEGCODE[count%100//10])

    pickDigit(2)
    hc595_shift(SEGCODE[count%1000//100])

    pickDigit(3)
    hc595_shift(SEGCODE[count%10000//1000])

```

Nachdem das Programm ausgeführt wurde, verwandelt sich die 4-stellige 7-Segment-Anzeige in einen Zähler, und die Zahl erhöht sich jede Sekunde um 1.

### Wie funktioniert es?

Das Senden von Signalen an jede 7-Segment-Anzeige erfolgt auf die gleiche Weise wie bei [5.2 Nummernanzeige](#), indem die Funktion `hc595_shift()` verwendet wird. Der Kernpunkt der 4-stelligen 7-Segment-Anzeige besteht darin, selektiv jede 7-Segment-Anzeige zu aktivieren. Der damit verbundene Code ist wie folgt:

```

placePin = []
pin = [13, 12, 11, 10]
for i in range(4):
    placePin.append(None)
    placePin[i] = machine.Pin(pin[i], machine.Pin.OUT)

```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
def pickDigit(digit):
    for i in range(4):
        placePin[i].value(1)
    placePin[digit].value(0)

while True:

    hc595_shift(SEGCODE[count%10])
    pickDigit(0)

    hc595_shift(SEGCODE[count%100//10])
    pickDigit(1)

    hc595_shift(SEGCODE[count%1000//100])
    pickDigit(2)

    hc595_shift(SEGCODE[count%10000//1000])
    pickDigit(3)
```

An dieser Stelle werden vier Pins (GP10, GP11, GP12, GP13) verwendet, um jeden Bit der 4-stelligen 7-Segment-Anzeige individuell zu steuern. Wenn der Zustand dieser Pins 0 ist, ist das entsprechende 7-Segment-Display aktiv; bei Zustand 1 gilt das Gegenteil.

Hier wird die Funktion `pickDigit(digit)` verwendet, um alle vier Ziffern zu deaktivieren und dann eine bestimmte Ziffer einzeln zu aktivieren. Danach wird `hc595_shift()` verwendet, um den entsprechenden 8-Bit-Code für die 7-Segment-Anzeige zu schreiben.

Die 4-stellige 7-Segment-Anzeige muss kontinuierlich abwechselnd aktiviert werden, damit wir sie sehen können. Jedoch dürfen wir im Hauptprogramm keinen Code hinzufügen, der das Timing beeinflussen würde. Zu diesem Zweck ist die Verwendung der Funktion `time.ticks_ms()` aus der `time`-Bibliothek eine ausgezeichnete Methode.

```
import time

timerStart=time.ticks_ms()

def timer1():
    return int((time.ticks_ms()-timerStart)/1000)

while True:
    count = timer1()
```

Die Funktion `time.ticks_ms()` ermittelt eine (nicht explizite) Zeit, und wir speichern den ersten ermittelten Zeitwert als `timerStart`. Wenn später die Zeit benötigt wird, wird die Funktion `time.ticks_ms()` erneut aufgerufen, und der Wert wird von `timerStart` abgezogen, um die bisherige Laufzeit des Programms (in Millisekunden) zu ermitteln.

Abschließend wird dieser Zeitwert in die 4-stellige 7-Segment-Anzeige umgewandelt und ausgegeben, und das war's.

- [Time - MicroPython Docs](#)

## 4.36 5.4 8x8 Pixel-Grafik

Die LED-Matrix ist eine niedrigauflösende Punkt-Matrix-Anzeige und nutzt ein Array aus Leuchtdioden als Pixel für gemusterte Darstellungen.

Diese Anzeigen sind hell genug, um auch im Tageslicht im Freien sichtbar zu sein und finden sich in einigen Geschäften, Werbetafeln, Schildern und variablen Anzeigetafeln (wie beispielsweise in öffentlichen Verkehrsmitteln).

In diesem Bausatz wird eine 8x8 Punktmatrix mit 16 Pins verwendet. Die Anoden sind in Reihen und die Kathoden in Spalten (auf der Schaltungsebene) miteinander verbunden, um so die 64 LEDs zu steuern.

Um die erste LED zu beleuchten, sollte ein hohes Signal an Row1 und ein niedriges an Col1 angelegt werden. Für die zweite LED gilt dementsprechend ein hohes Signal an Row1 und ein niedriges an Col2, und so weiter. Jede LED lässt sich individuell steuern, indem der Stromfluss durch die jeweiligen Reihen und Spalten geregelt wird. Dadurch können Zeichen oder Bilder dargestellt werden.

- [LED-Punktmatrix](#)
- [74HC595](#)

### Erforderliche Komponenten

Für dieses Projekt benötigen wir folgende Komponenten:

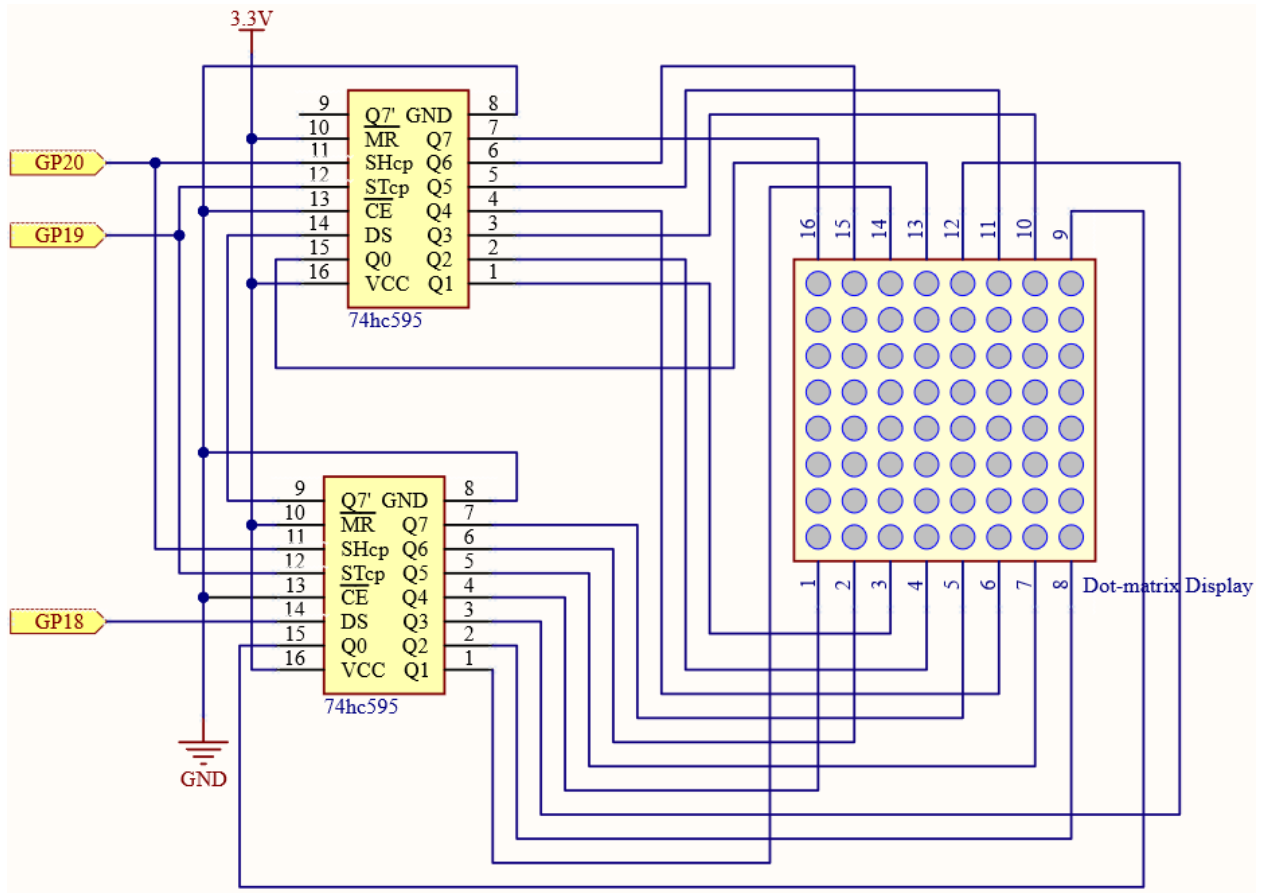
Es ist definitiv praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	KOMPONENTEN IM SET	LINK
Kepler Kit	450+	

Sie können die Komponenten auch einzeln über die folgenden Links erwerben.

SN	KOMPONENTE	MEN-GE	LINK
1	<a href="#">Raspberry Pi Pico W</a>	1	
2	Micro-USB-Kabel	1	
3	<a href="#">Steckbrett</a>	1	
4	<a href="#">Jumperkabel</a>	Mehrere	
5	<a href="#">LED-Punktmatrix</a>	1	
6	<a href="#">74HC595</a>	2	

### Schaltplan



Die 8x8 Punktmatrix wird durch zwei 74HC595-Chips gesteuert, wobei einer die Reihen und der andere die Spalten steuert. Beide Chips teilen sich die Ports GP18~GP20, was die I/O-Ports des Pico W Boards erheblich einspart.

Pico W muss eine 16-Bit-Binärzahl ausgeben, wobei die ersten 8 Bit an den 74HC595 für die Reihen und die letzten 8 Bit an den 74HC595 für die Spalten gehen, damit die Punktmatrix ein bestimmtes Muster anzeigen kann.

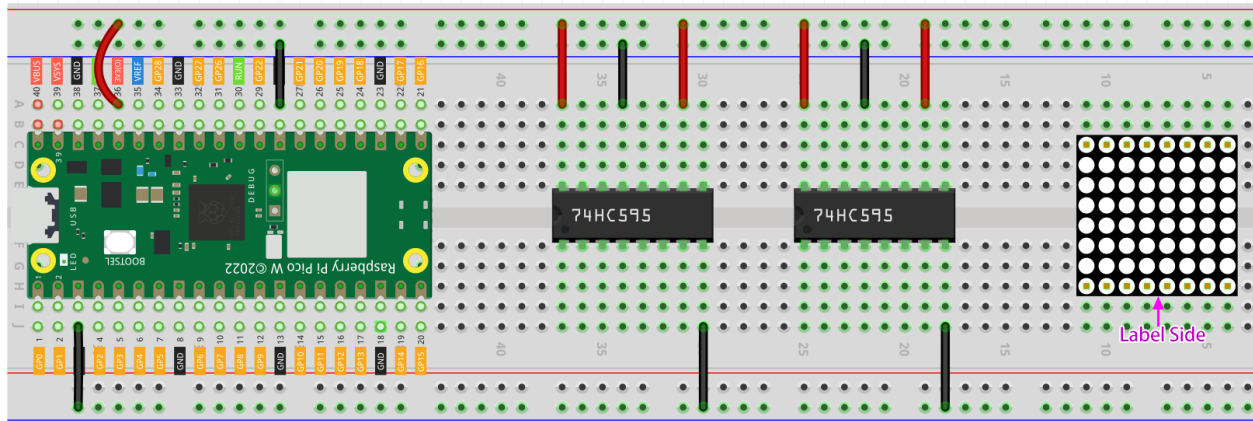
Q7': Serieller Ausgangspin, verbunden mit DS eines weiteren 74HC595, um mehrere 74HC595 in Serie zu schalten.

### Verdrahtung

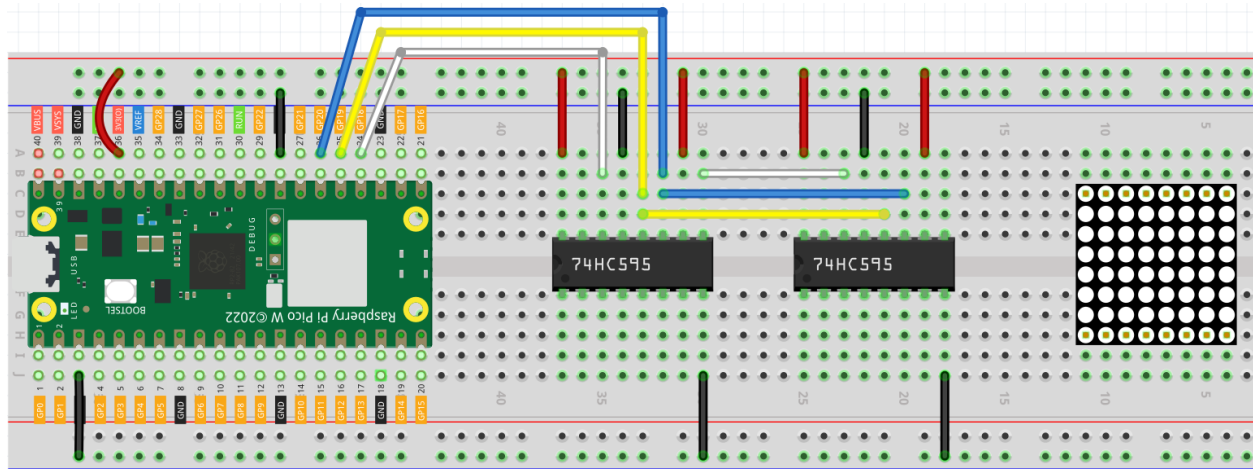
Bauen Sie die Schaltung auf. Da die Verdrahtung kompliziert ist, gehen wir schrittweise vor.

**Schritt 1:** Setzen Sie zunächst den Pico W, die LED-Punktmatrix und die beiden 74HC595-Chips in das Steckbrett ein. Verbinden Sie 3,3V und GND des Pico W mit den Löchern an beiden Seiten der Platine, und schließen Sie dann Pin 16 und Pin 10 der beiden 74HC595-Chips an VCC, Pin 13 und Pin 8 an GND an.

**Bemerkung:** In der oben stehenden Fritzing-Abbildung ist die beschriftete Seite unten.

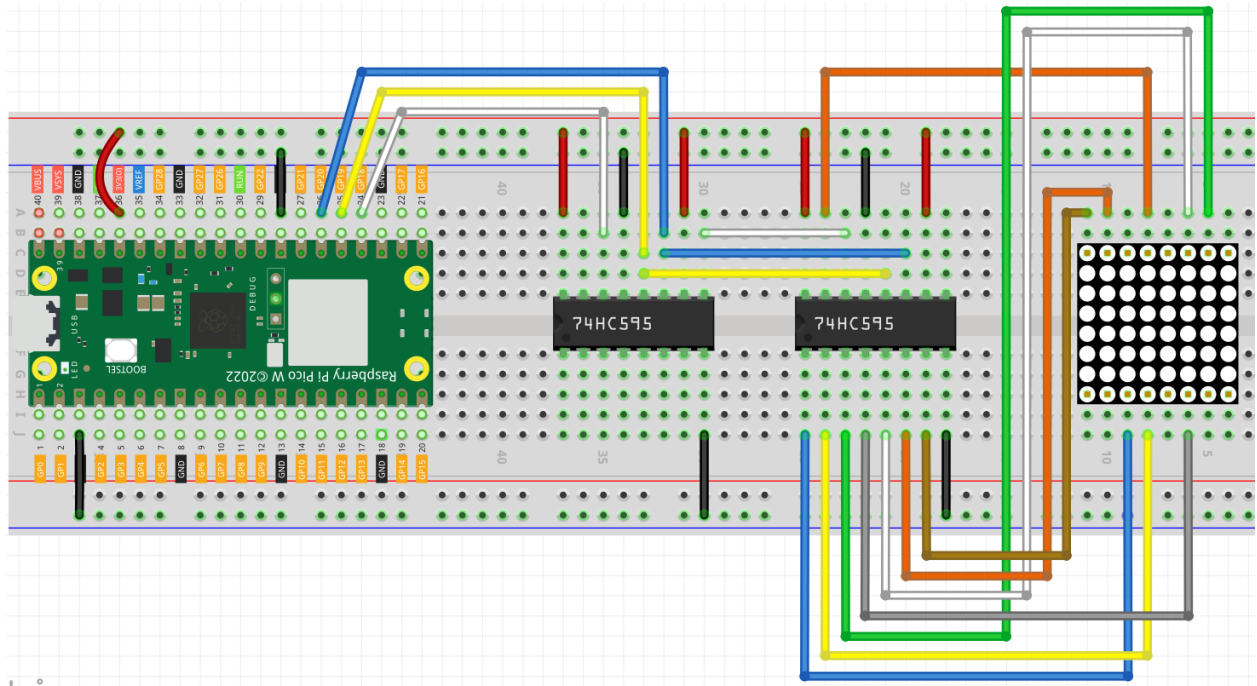


**Schritt 2:** Verbinden Sie Pin 11 der beiden 74HC595 miteinander und dann mit GP20; danach Pin 12 der beiden Chips und mit GP19; als Nächstes Pin 14 des linken 74HC595 mit GP18 und Pin 9 mit Pin 14 des zweiten 74HC595.



**Schritt 3:** Der 74HC595 auf der rechten Seite dient zur Steuerung der Spalten der LED-Punktmatrix. Untenstehende Tabelle zeigt die Zuordnung. Daher sind die Pins Q0-Q7 des 74HC595 jeweils mit den Pins 13, 3, 4, 10, 6, 11, 15 und 16 verbunden.

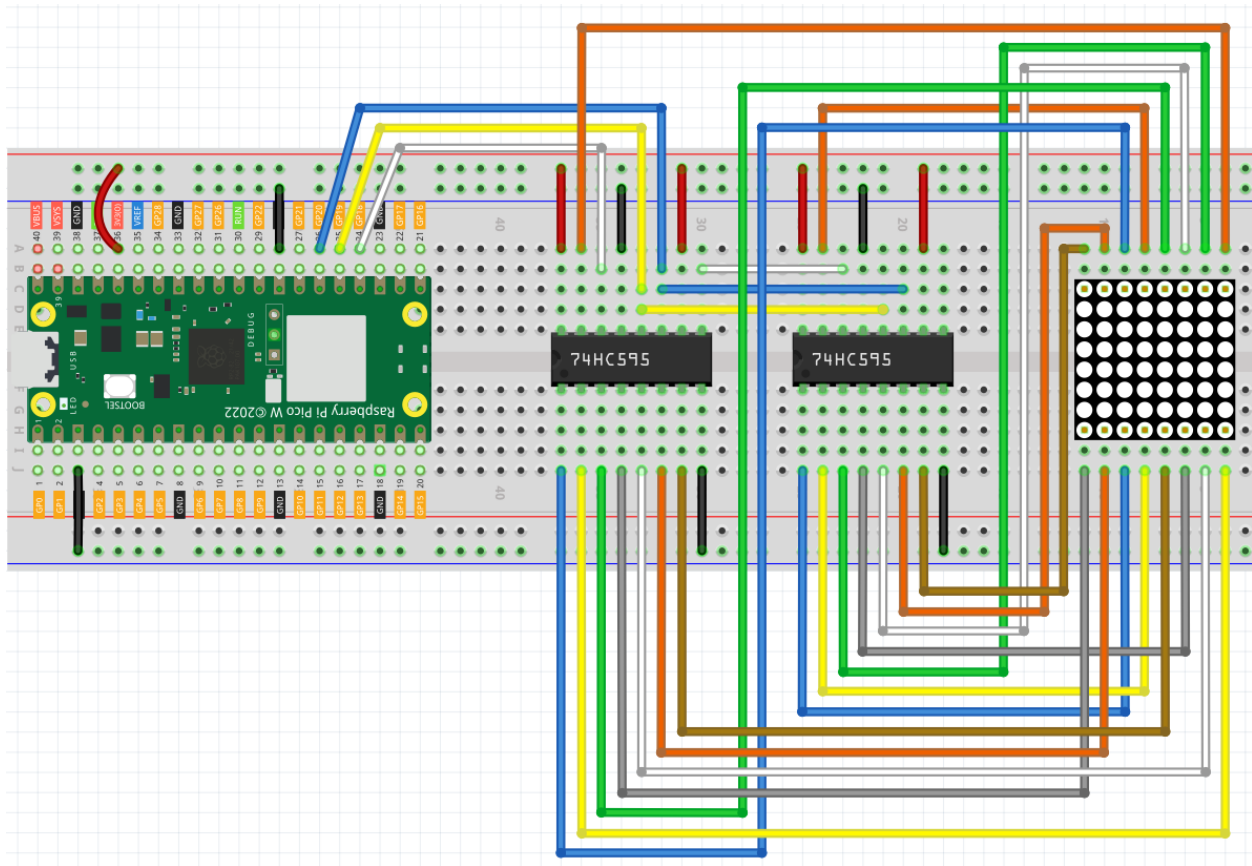
74HC595	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
LED Dot Matrix	13	3	4	10	6	11	15	16



**Schritt 4:** Verbinden Sie nun die Reihen der LED-Punktmatrix. Der 74HC595 auf der linken Seite steuert die Reihen der LED-Punktmatrix. Unterstehende Tabelle zeigt die Zuordnung. Wie man sieht, sind die Pins Q0-Q7 des linken 74HC595 jeweils mit den Pins 9, 14, 8, 12, 1, 7, 2 und 5 verbunden.

74HC595	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
LED Dot Matrix	9	14	8	12	1	7	2	5





## Code

### Bemerkung:

- Öffnen Sie die Datei `5.4_8x8_pixel_graphics.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny, und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der unteren rechten Ecke auszuwählen.
- Für detaillierte Anleitungen verweisen wir auf *Code direkt öffnen und ausführen*.

```
import machine
import time

sdi = machine.Pin(18,machine.Pin.OUT)
rclk = machine.Pin(19,machine.Pin.OUT)
srclk = machine.Pin(20,machine.Pin.OUT)

glyph = [0xFF,0xBB,0xD7,0xEF,0xD7,0xBB,0xFF,0xFF]

# Daten an 74HC595 senden
def hc595_in(dat):
    for bit in range(7, -1, -1):
        srclk.low()
```

(Fortsetzung auf der nächsten Seite)



(Fortsetzung der vorherigen Seite)

```

        time.sleep_us(30)
        sdi.value(1 & (dat >> bit))
        time.sleep_us(30)
        srclk.high()

def hc595_out():
    rclk.high()
    time.sleep_us(200)
    rclk.low()

while True:
    for i in range(0,8):
        hc595_in(glyph[i])
        hc595_in(0x80>>i)
        hc595_out()

```

Sobald das Programm läuft, wird ein x-Grafikmuster auf der 8x8-Punktmatrix angezeigt.

### Wie funktioniert es?

Hier nutzen wir zwei 74HC595-Chips, um die Signale für die Reihen und Spalten der Punkt-Matrix zu steuern. Die Methode zur Signalbereitstellung entspricht der Funktion `hc595_shift(dat)` aus vorherigen Kapiteln. Der Unterschied besteht jedoch darin, dass wir hier eine 16-Bit-Binärzahl auf einmal schreiben müssen. Daher teilen wir `hc595_shift(dat)` in zwei Funktionen auf: `hc595_in(dat)` und `hc595_out()`.

```

def hc595_in(dat):
    for bit in range(7, -1, -1):
        srclk.low()
        time.sleep_us(30)
        sdi.value(1 & (dat >> bit))
        time.sleep_us(30)
        srclk.high()

def hc595_out():
    rclk.high()
    time.sleep_us(200)
    rclk.low()

```

Anschließend rufen Sie `hc595_in(dat)` zweimal in der Hauptschleife auf, schreiben zwei 8-Bit-Binärzahlen und rufen dann `hc595_out()` auf, damit ein Muster angezeigt werden kann.

Beachten Sie jedoch, dass die LEDs in der Punkt-Matrix gemeinsame Pole verwenden. Die gleichzeitige Steuerung mehrerer Reihen bzw. Spalten würde sich gegenseitig beeinflussen. Daher ist es notwendig, eine Spalte (oder eine Reihe) nach der anderen zu aktivieren, den Vorgang 8-mal zu wiederholen und das Prinzip der Nachbildwirkung zu nutzen, um das menschliche Auge 8 Muster verschmelzen zu lassen.

```

while True:
    for i in range(0, 8):
        hc595_in(glyph[i])
        hc595_in(0x80 >> i)
        hc595_out()

```

In diesem Beispiel schachtelt die Hauptfunktion eine `for`-Schleife. Wenn `i` 1 ist, wird nur die erste Zeile aktiviert, und das Bild der ersten Zeile wird geschrieben. Und so weiter, bis alle 8 Ausgaben abgeschlossen sind.

Übrigens sollte, ähnlich wie beim 4-stelligen 7-Segment-Display, die Aktualisierungsrate aufrechterhalten werden, um ein Flackern zu vermeiden. Daher sollte zusätzliches `sleep()` in der Hauptschleife möglichst vermieden werden.

### Mehr erfahren

Versuchen Sie, `glyph` durch das folgende Array zu ersetzen und schauen Sie, was passiert!

```
glyph1 = [0xFF, 0xEF, 0xC7, 0xAB, 0xEF, 0xEF, 0xEF, 0xFF]
glyph2 = [0xFF, 0xEF, 0xEF, 0xEF, 0xAB, 0xC7, 0xEF, 0xFF]
glyph3 = [0xFF, 0xEF, 0xDF, 0x81, 0xDF, 0xEF, 0xFF, 0xFF]
glyph4 = [0xFF, 0xF7, 0xFB, 0x81, 0xFB, 0xF7, 0xFF, 0xFF]
glyph5 = [0xFF, 0xBB, 0xD7, 0xEF, 0xD7, 0xBB, 0xFF, 0xFF]
glyph6 = [0xFF, 0xFF, 0xF7, 0xEB, 0xDF, 0xBF, 0xFF, 0xFF]
```

Oder Sie könnten versuchen, Ihre eigenen Grafiken zu zeichnen.

### 6. Fortgeschritten

## 4.37 6.1 Abstandsmessung

Das Ultraschallsensormodul funktioniert nach dem Prinzip von Sonar- und Radarsystemen, um die Entfernung zu einem Objekt zu ermitteln.

- *Ultraschallmodul*

### Benötigte Komponenten

Für dieses Projekt werden die folgenden Komponenten benötigt.

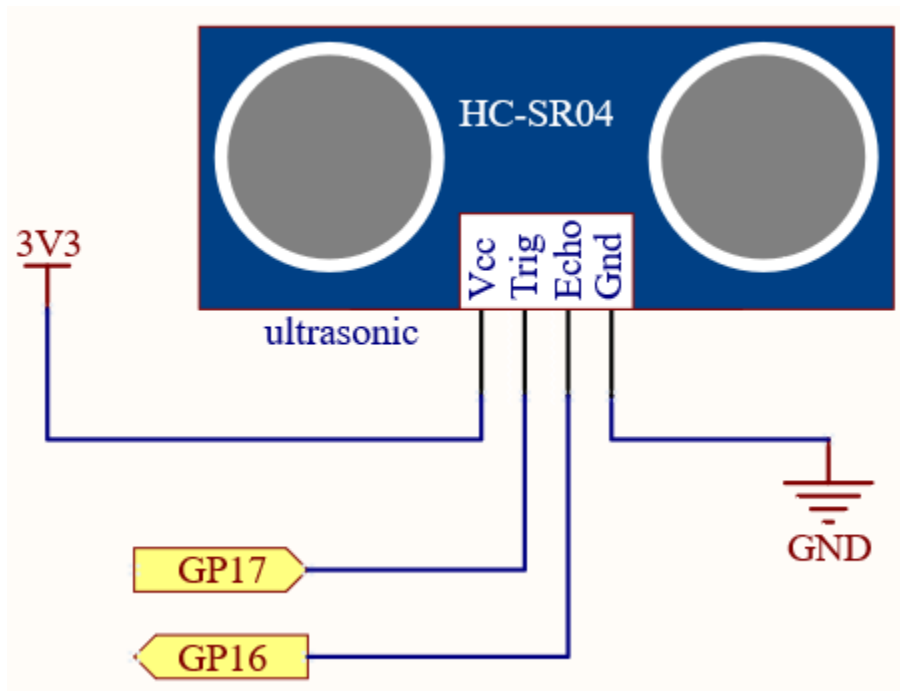
Ein Komplettsset zu kaufen ist definitiv praktisch, hier ist der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler-Set	450+	

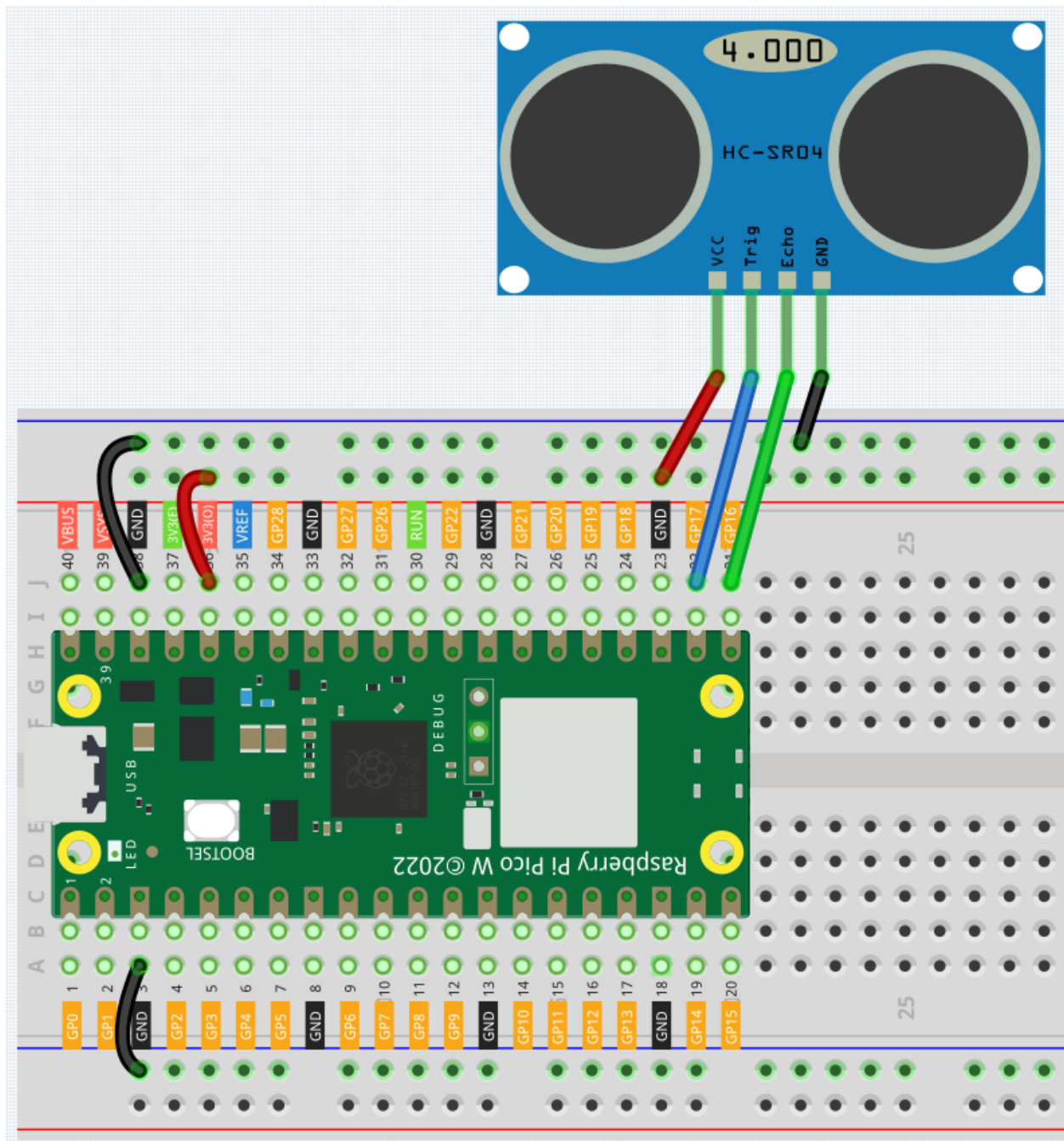
Sie können die Teile auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Ultraschallmodul</i>	1	

### Schaltplan



Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `6.1_measuring_distance.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
- Für detaillierte Anleitungen verweisen wir auf *Code direkt öffnen und ausführen*.

```

import machine
import time

TRIG = machine.Pin(17,machine.Pin.OUT)
ECHO = machine.Pin(16,machine.Pin.IN)

def distance():
    TRIG.low()
    time.sleep_us(2)
    TRIG.high()
    time.sleep_us(10)
    TRIG.low()
    while not ECHO.value():
        pass
    time1 = time.ticks_us()
    while ECHO.value():
        pass
    time2 = time.ticks_us()
    during = time.ticks_diff(time2,time1)
    return during * 340 / 2 / 10000

while True:
    dis = distance()
    print ('Distance: %.2f' % dis)
    time.sleep_ms(300)

```

Sobald das Programm läuft, wird die Shell den Abstand des Ultraschallsensors zum Hindernis vor ihm ausgeben.

### Funktionsweise

Ultraschallsensoren erzeugen hochfrequente Schallwellen (Ultraschallwellen), die von der Sendesonde ausgesendet werden. Trifft diese Ultraschallwelle auf ein Objekt, wird sie als Echo reflektiert und von der Empfangssonde detektiert. Durch die Berechnung der Zeit von der Aussendung bis zum Empfang lässt sich die Entfernung ermitteln. Auf diesem Prinzip basiert die Funktion `distance()`.

```

def distance():
    TRIG.low()
    time.sleep_us(2)
    TRIG.high()
    time.sleep_us(10)
    TRIG.low()
    while not ECHO.value():
        pass
    time1 = time.ticks_us()
    while ECHO.value():
        pass
    time2 = time.ticks_us()
    during = time.ticks_diff(time2,time1)
    return during * 340 / 2 / 10000

```

- Dabei dienen die ersten paar Zeilen dazu, eine 10µs Ultraschallwelle auszusenden.

```
TRIG.low()
time.sleep_us(2)
TRIG.high()
time.sleep_us(10)
TRIG.low()
```

- Anschließend wird das Programm angehalten und die aktuelle Zeit erfasst, sobald die Ultraschallwelle ausgesendet wurde.

```
while not ECHO.value():
    pass
time1 = time.ticks_us()
```

- Daraufhin wird das Programm erneut pausiert. Nachdem das Echo empfangen wurde, wird die aktuelle Zeit erneut erfasst.

```
while ECHO.value():
    pass
time2 = time.ticks_us()
```

- Abschließend wird anhand der Zeitdifferenz zwischen den beiden Erfassungen die Schallgeschwindigkeit (340 m/s) mit der Zeit multipliziert, um die doppelte Entfernung zwischen dem Ultraschallmodul und dem Hindernis zu erhalten (also einen Rundflug der Ultraschallwellen vom Modul zum Hindernis). Die Umrechnung in Zentimeter liefert den benötigten Rückgabewert.

```
during = time.ticks_diff(time2,time1)
return during * 340 / 2 / 10000
```

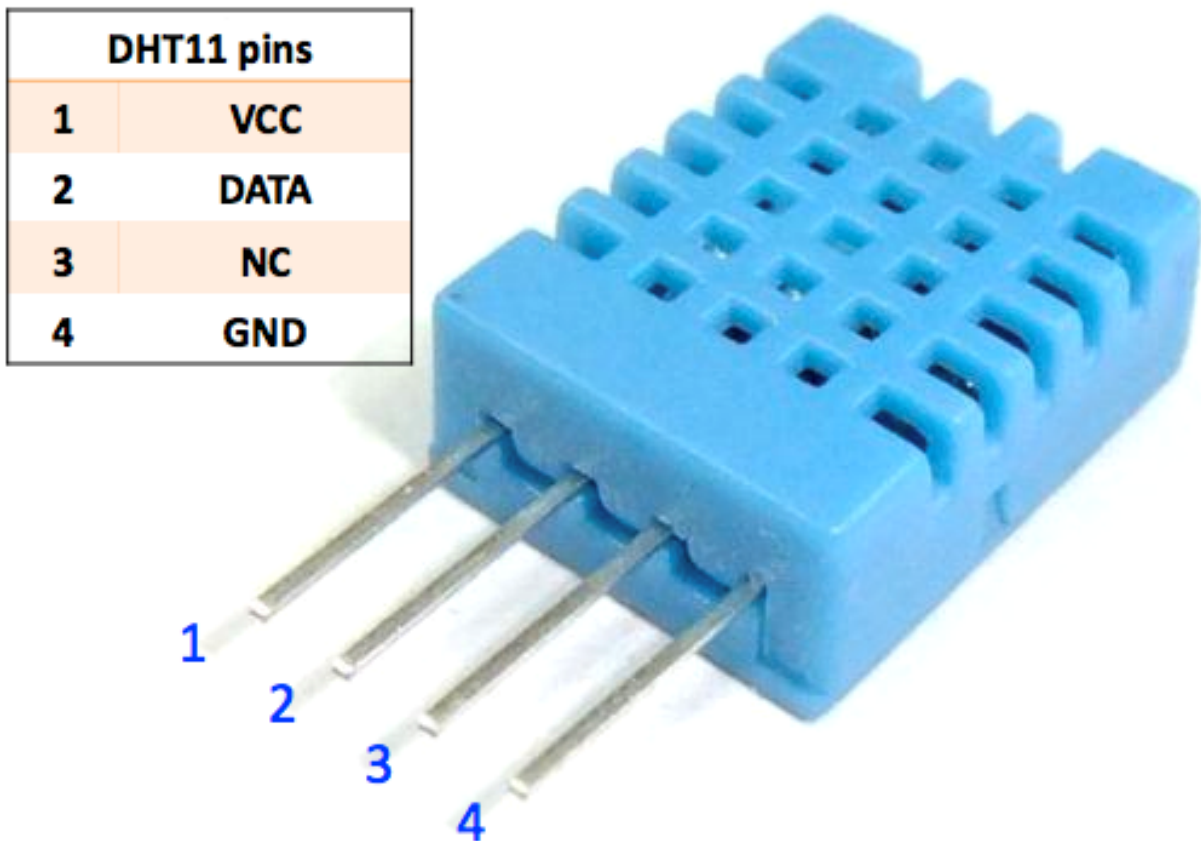
Beachten Sie, dass der Ultraschallsensor das Programm anhält, wenn er arbeitet, was zu Verzögerungen bei der Entwicklung komplexer Projekte führen kann.

## 4.38 6.2 Temperatur - Feuchtigkeit

Feuchtigkeit und Temperatur stehen sowohl im Hinblick auf die physikalische Größe selbst als auch im tatsächlichen Leben der Menschen in engem Zusammenhang. Die Temperatur und Feuchtigkeit unserer Umgebung beeinflussen direkt die Thermoregulationsfunktion und den Wärmeübertragungseffekt des menschlichen Körpers. Dies beeinflusst weiterhin die Denkaktivität und den geistigen Zustand und somit die Effizienz unserer Lern- und Arbeitsprozesse.

Temperatur ist eine der sieben grundlegenden physikalischen Größen im Internationalen Einheitensystem und dient zur Messung des Wärme- oder Kältegrades eines Objekts. Celsius ist eine der weltweit am häufigsten verwendeten Temperaturskalen und wird durch das Symbol „°C“ ausgedrückt.

Feuchtigkeit ist die Konzentration von Wasserdampf in der Luft. Die relative Luftfeuchtigkeit wird im Alltag häufig verwendet und in %RH ausgedrückt. Sie steht in engem Zusammenhang mit der Temperatur. Für ein bestimmtes Volumen von abgeschlossenem Gas gilt: Je höher die Temperatur, desto niedriger die relative Feuchtigkeit und umgekehrt.



Ein grundlegender digitaler Temperatur- und Feuchtigkeitssensor, der **DHT11**, ist in diesem Kit enthalten. Er verwendet einen kapazitiven Feuchtigkeitssensor und einen Thermistor, um die umgebende Luft zu messen und gibt ein digitales Signal an den Datenpins aus (analoge Eingangspins sind nicht erforderlich).

- *DHT11 Temperatur- und Feuchtigkeitssensor*

### Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten:

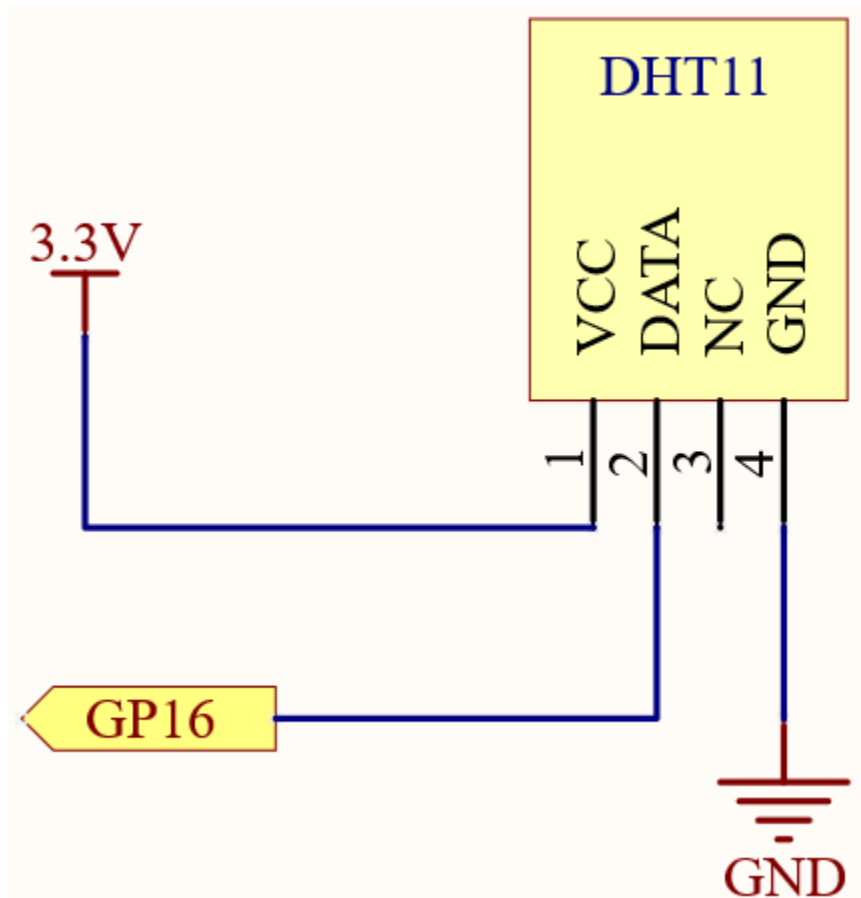
Es ist definitiv praktisch, ein gesamtes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler-Set	450+	

Sie können diese auch über die nachfolgenden Links einzeln erwerben:

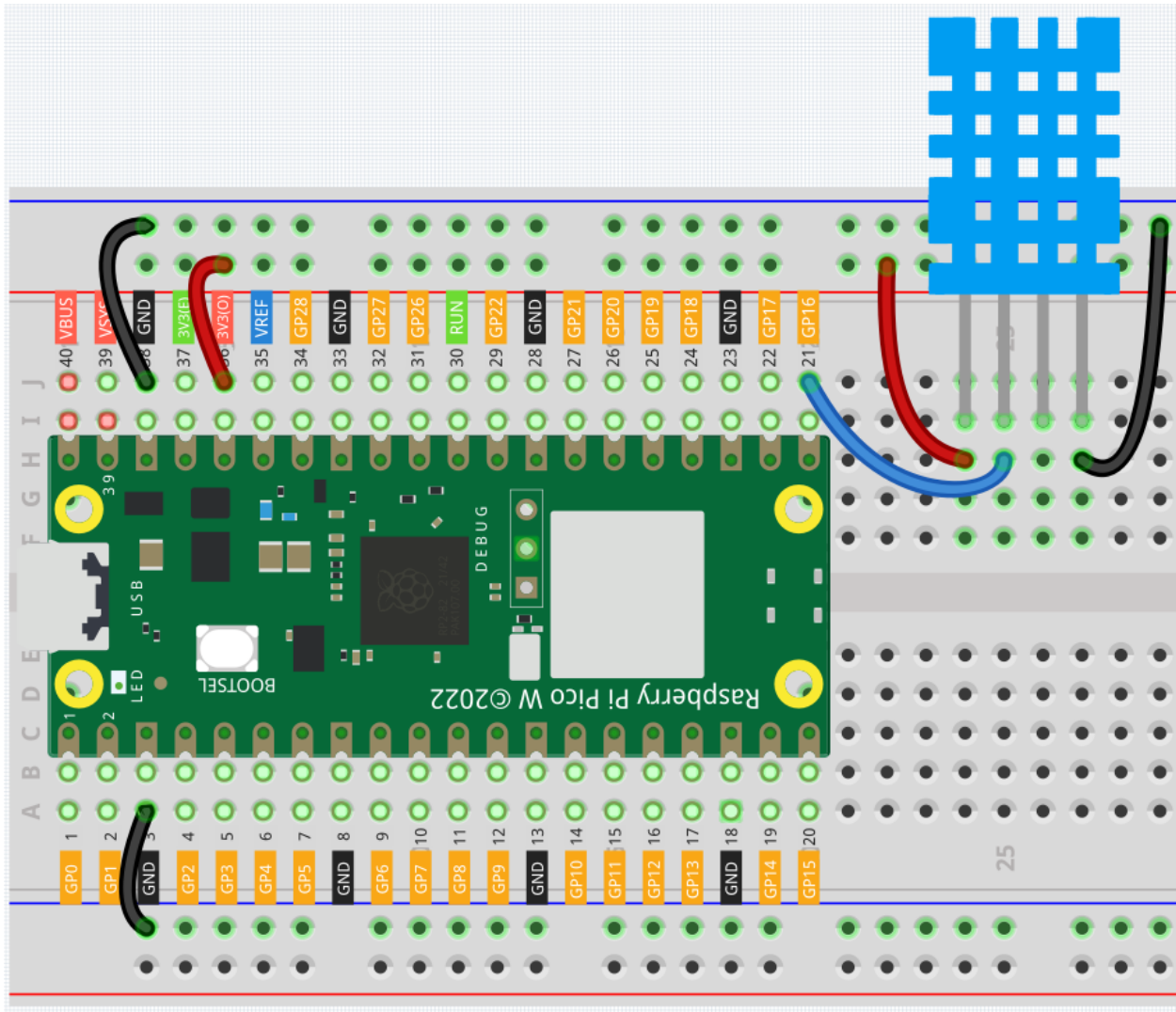
SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro USB Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>DHT11 Temperatur- und Feuchtigkeitssensor</i>	1	

### Schaltplan



### Verkabelung





### Code

#### Bemerkung:

- Öffnen Sie die Datei `6.2_temperature_humidity.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der rechten unteren Ecke auszuwählen.
- Detaillierte Anleitungen finden Sie unter [Code direkt öffnen und ausführen](#).
- Hier müssen Sie die Bibliothek `dht.py` verwenden. Bitte überprüfen Sie, ob sie auf Pico W hochgeladen wurde. Eine detaillierte Anleitung finden Sie unter [1.4 Bibliotheken auf den Pico hochladen](#).

```
from machine import Pin, I2C
import utime as time
from dht import DHT11, InvalidPulseCount
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

pin = Pin(16, Pin.IN, Pin.PULL_UP)
sensor = DHT11(pin)
time.sleep(5) # initial delay

while True:
    try:
        sensor.measure()
        string = "Temperature:{}\nHumidity: {}".format(sensor.temperature, sensor.
↪humidity)
        print(string)
        time.sleep(4)

    except InvalidPulseCount as e:
        print('Bad pulse count - retrying ...')

```

Nachdem der Code ausgeführt wurde, werden Temperatur und Feuchtigkeit kontinuierlich in der Shell ausgegeben. Mit fortlaufender Programmausführung werden diese Werte immer genauer.

### Wie funktioniert es?

In der dht-Bibliothek haben wir die relevante Funktionalität in die Klasse DHT11 integriert.

```

from dht import DHT11, InvalidPulseCount

```

Initialisieren Sie das DHT11-Objekt. Für dieses Gerät wird nur ein digitaler Eingang benötigt.

```

pin = Pin(16, Pin.IN, Pin.PULL_UP)
sensor = DHT11(pin)

```

Verwenden Sie `sensor.measure()`, um die aktuelle Temperatur und Feuchtigkeit zu lesen, die in `sensor.temperature` und `sensor.humidity` gespeichert werden. Diese Werte werden dann ausgegeben. Die Abtastrate des DHT11 beträgt 1HZ, daher wird in der Schleife ein `time.sleep(1)` benötigt.

```

while True:
    try:
        sensor.measure()
        string = "Temperature:{}\nHumidity: {}".format(sensor.temperature, sensor.
↪humidity)
        print(string)
        time.sleep(4)

    except InvalidPulseCount as e:
        print('Bad pulse count - retrying ...')

```

## 4.39 6.3 6-Achsen-Bewegungsverfolgung

Der MPU-6050 ist ein 6-Achsen-Sensor, der einen 3-Achsen-Gyroskop und einen 3-Achsen-Beschleunigungsmesser kombiniert.

Ein Beschleunigungsmesser ist ein Instrument zur Messung der Eigenbeschleunigung. Ein am Erdboden ruhender Beschleunigungsmesser misst beispielsweise eine Beschleunigung in Richtung der Erdanziehung von etwa  $g \approx 9.81 \text{ m/s}^2$ .

Beschleunigungsmesser finden in Industrie und Wissenschaft vielfältige Anwendung, beispielsweise in Trägheitsnavigationssystemen für Flugzeuge und Raketen, zur Bilddarstellung in Tablets und Digitalkameras und vieles mehr.

Gyroskope dienen der Messung der Orientierung und der Winkelgeschwindigkeit eines Geräts. Einsatzgebiete für Gyroskope umfassen Anti-Kipp- und Airbagsysteme für Automobile, Bewegungssensoren für intelligente Geräte, Lagestabilisierungssysteme für Drohnen und vieles mehr.

- *MPU6050 Modul*

### Erforderliche Komponenten

Für dieses Projekt benötigen wir folgende Bauteile.

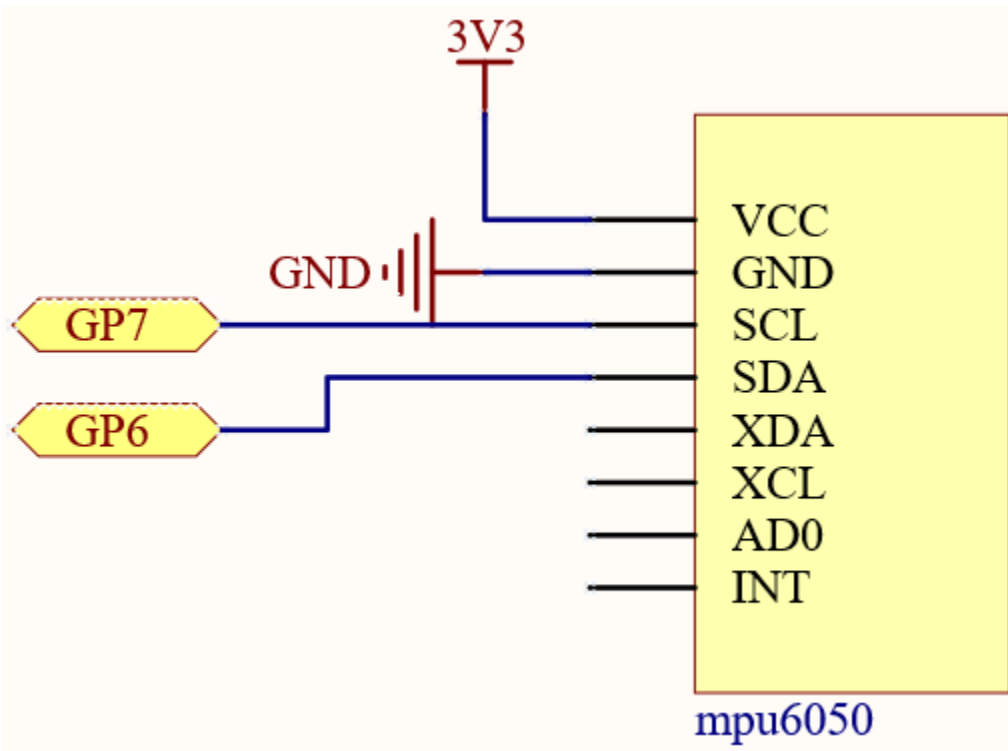
Ein komplettes Kit zu erwerben, ist natürlich praktisch. Hier ist der Link:

Bezeichnung	KOMPONENTEN IM KIT	LINK
Kepler-Kit	450+	

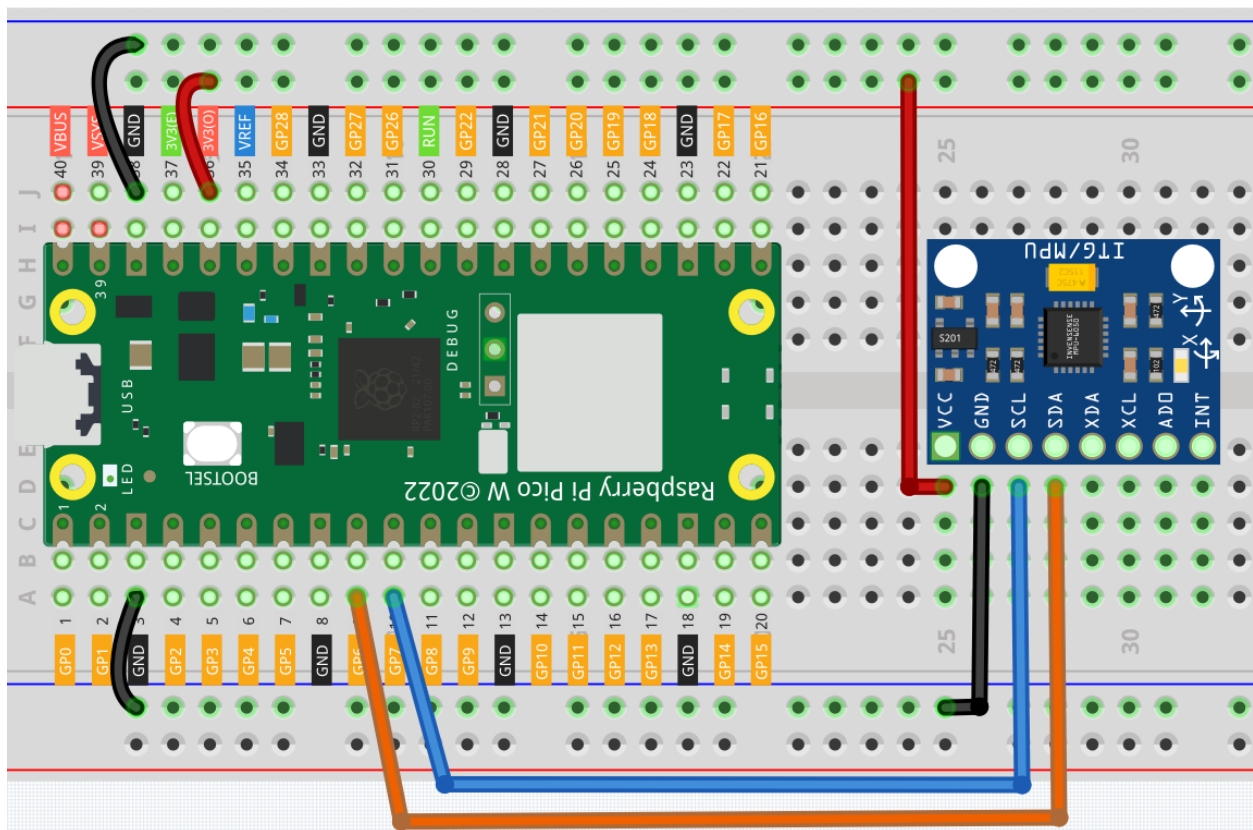
Die Komponenten können auch einzeln über die folgenden Links bezogen werden:

SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>MPU6050 Modul</i>	1	

### Schaltplan



## Verkabelung



## Code

**Bemerkung:**

- Öffnen Sie die Datei `6.3_6axis_motion_tracking.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie den Code in Thonny. Dann klicken Sie auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Wählen Sie im rechten unteren Eck den „MicroPython (Raspberry Pi Pico)“-Interpreter.
- Für detaillierte Anleitungen siehe *Code direkt öffnen und ausführen*.
- Hier benötigen Sie die Bibliotheken `imu.py` und `vector3d.py`. Stellen Sie sicher, dass diese auf dem Pico W hochgeladen wurden. Eine detaillierte Anleitung finden Sie unter *1.4 Bibliotheken auf den Pico hochladen*.

```
from imu import MPU6050
from machine import I2C, Pin
import time

i2c = I2C(1, sda=Pin(6), scl=Pin(7), freq=400000)
mpu = MPU6050(i2c)

while True:
    print("x: %s, y: %s, z: %s"%(mpu.accel.x, mpu.accel.y, mpu.accel.z))
    time.sleep(0.1)
    print("A: %s, B: %s, Y: %s"%(mpu.gyro.x, mpu.gyro.y, mpu.gyro.z))
    time.sleep(0.1)
```

Nach dem Ausführen des Programms sehen Sie die Werte des 3-Achsen-Beschleunigungsmessers und des 3-Achsen-Gyroskops in der Ausgabe rotieren. Drehen Sie den MPU6050 beliebig, und Sie werden feststellen, dass sich die Werte entsprechend ändern. Um die Änderungen besser erkennen zu können, können Sie eine der Ausgabelinien auskommentieren und sich auf einen Datensatz konzentrieren.

Die Einheit des Beschleunigungswerts ist „m/s<sup>2</sup>“ und die Einheit des Gyroskopwerts ist „°/s“.

**Wie funktioniert es?**

In der `imu`-Bibliothek haben wir die relevanten Funktionen in der Klasse `MPU6050` integriert. Der MPU6050 ist ein I2C-Modul und erfordert für die Initialisierung definierte I2C-Pins.

```
from imu import MPU6050
from machine import I2C, Pin

i2c = I2C(1, sda=Pin(6), scl=Pin(7), freq=400000)
mpu = MPU6050(i2c)
```

In der Folge können Sie Echtzeit-Beschleunigungs- und Winkelgeschwindigkeitswerte in `mpu.accel.x`, `mpu.accel.y`, `mpu.accel.z`, `mpu.gyro.x`, `mpu.gyro.y`, `mpu.gyro.z` abrufen.

```
while True:
    print("x: %s, y: %s, z: %s"%(mpu.accel.x, mpu.accel.y, mpu.accel.z))
    time.sleep(0.1)
    print("A: %s, B: %s, Y: %s"%(mpu.gyro.x, mpu.gyro.y, mpu.gyro.z))
    time.sleep(0.1)
```

## 4.40 6.4 Infrarot-Fernbedienung

In der Unterhaltungselektronik dienen Fernbedienungen zur Steuerung von Geräten wie Fernsehern und DVD-Playern. In einigen Fällen ermöglichen sie die Bedienung von Geräten, die außerhalb der Reichweite liegen, wie beispielsweise Zentral-Klimaanlagen.

Der IR-Empfänger ist eine Komponente mit einer auf Infrarotlicht abgestimmten Fotodiode. Er kommt fast immer bei der Erkennung von Fernbedienungssignalen zum Einsatz - jeder Fernseher und DVD-Player ist an der Frontseite mit einem solchen Modul ausgestattet, um das IR-Signal von der Fernbedienung zu empfangen. In der Fernbedienung selbst befindet sich eine dazu passende IR-LED, die IR-Impulse aussendet, um den Fernseher ein- oder auszuschalten oder den Sender zu wechseln.

- *Infrarotempfänger*

### Benötigte Bauteile

Für dieses Projekt benötigen wir folgende Komponenten.

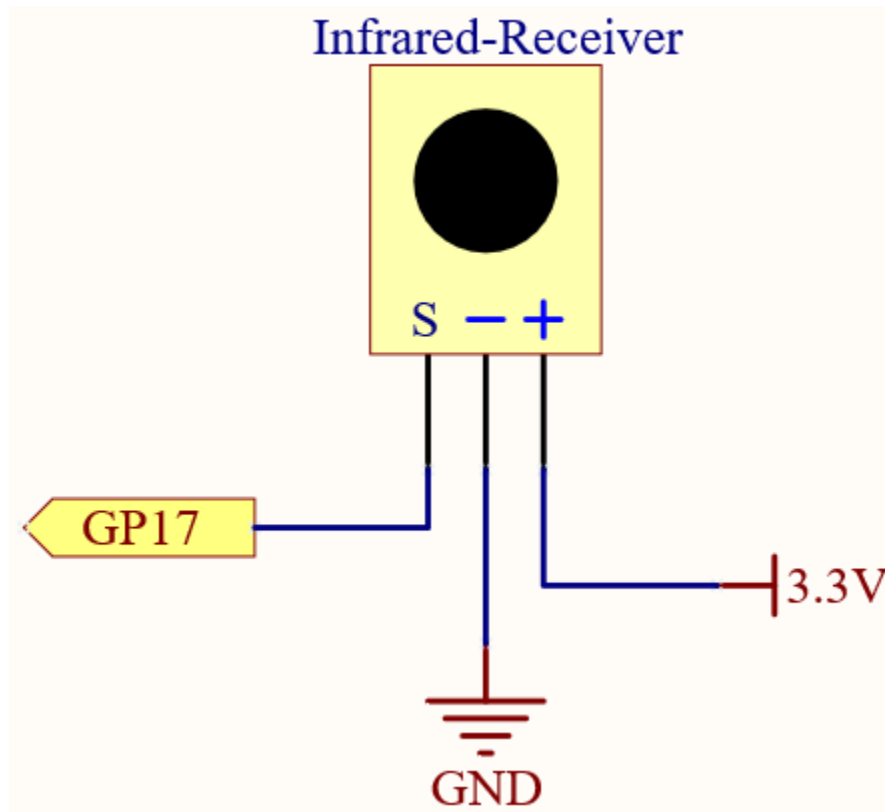
Ein Komplett-Kit ist natürlich praktisch, hier der Link dazu:

Bezeichnung	KOMPONENTEN IM KIT	LINK
Kepler-Kit	450+	

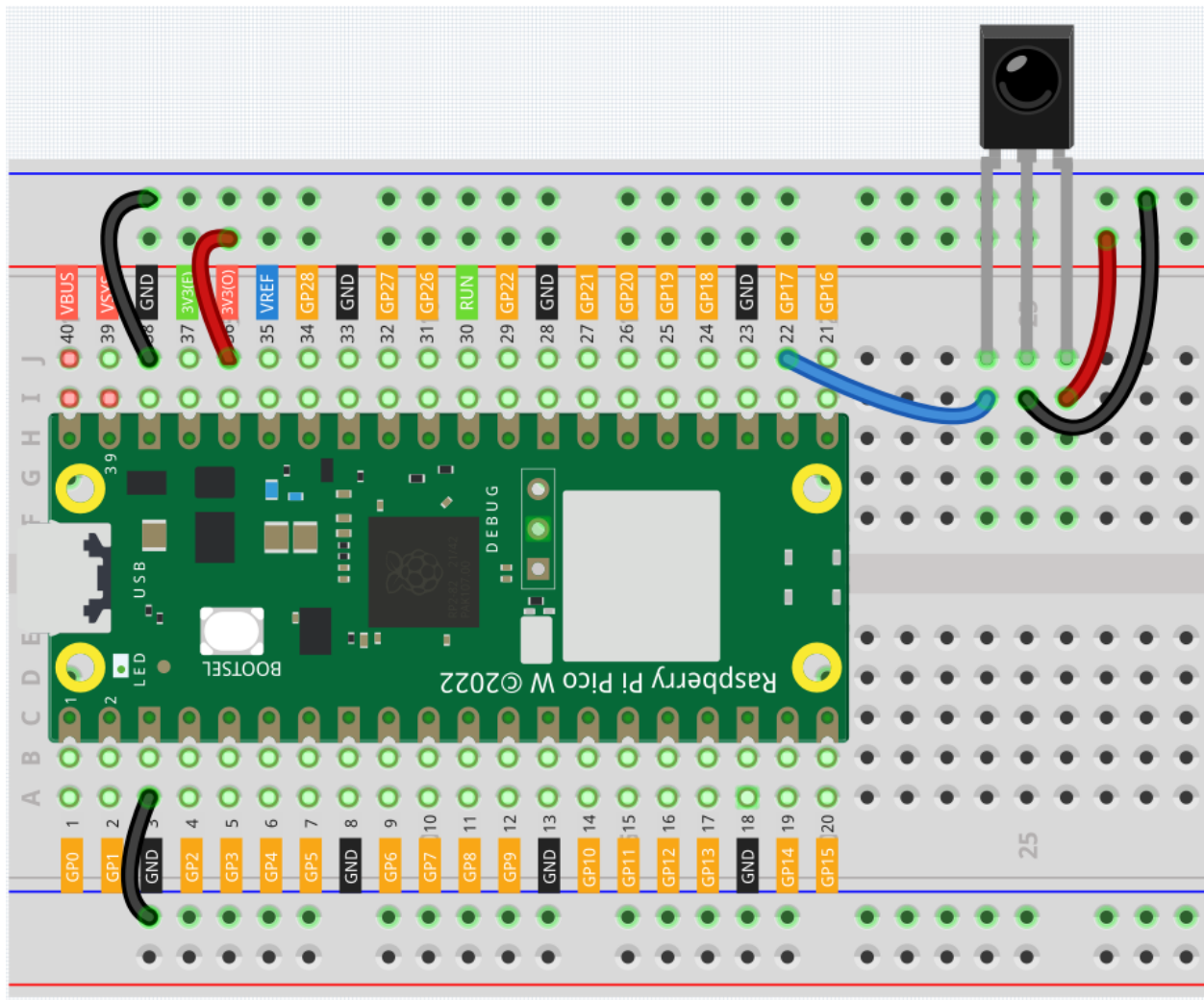
Die einzelnen Komponenten können auch über die untenstehenden Links erworben werden.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Mikro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Infrarotempfänger</i>	1	

### Schaltplan



Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `6.4_ir_remote_control.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie den Code in Thonny und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Wählen Sie in der unteren rechten Ecke den Interpreter „MicroPython (Raspberry Pi Pico)“ aus.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).
- Die benötigten Bibliotheken finden Sie im Ordner `ir_rx`. Vergewissern Sie sich, dass diese auf den Pico hochgeladen wurden. Detaillierte Anleitungen finden Sie unter [1.4 Bibliotheken auf den Pico hochladen](#).

```
import time
from machine import Pin, freq
from ir_rx.print_error import print_error
from ir_rx.nec import NEC_8
```

(Fortsetzung auf der nächsten Seite)



(Fortsetzung der vorherigen Seite)

```

pin_ir = Pin(17, Pin.IN)

def decodeKeyValue(data):
    if data == 0x16:
        return "0"
    if data == 0x0C:
        return "1"
    if data == 0x18:
        return "2"
    if data == 0x5E:
        return "3"
    if data == 0x08:
        return "4"
    if data == 0x1C:
        return "5"
    if data == 0x5A:
        return "6"
    if data == 0x42:
        return "7"
    if data == 0x52:
        return "8"
    if data == 0x4A:
        return "9"
    if data == 0x09:
        return "+"
    if data == 0x15:
        return "-"
    if data == 0x7:
        return "EQ"
    if data == 0x0D:
        return "U/SD"
    if data == 0x19:
        return "CYCLE"
    if data == 0x44:
        return "PLAY/PAUSE"
    if data == 0x43:
        return "FORWARD"
    if data == 0x40:
        return "BACKWARD"
    if data == 0x45:
        return "POWER"
    if data == 0x47:
        return "MUTE"
    if data == 0x46:
        return "MODE"
    return "ERROR"

# User callback
def callback(data, addr, ctrl):
    if data < 0: # NEC protocol sends repeat codes.
        pass
    else:

```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

        print(decodeKeyValue(data))

ir = NEC_8(pin_ir, callback) # Instantiate receiver
ir.error_function(print_error) # Show debug information

try:
    while True:
        pass
except KeyboardInterrupt:
    ir.close()

```

Die neue Fernbedienung besitzt ein Plastikteil am Ende, um die innenliegende Batterie zu isolieren. Um die Fernbedienung zu aktivieren, muss dieses Plastikteil entfernt werden. Sobald das Programm läuft und Sie eine Taste auf der Fernbedienung drücken, wird die gedrückte Taste in der Shell ausgegeben.

### Wie funktioniert es?

Das Programm mag auf den ersten Blick komplex erscheinen, erfüllt jedoch die Grundfunktionen des IR-Empfängers mit nur wenigen Codezeilen.

```

import time
from machine import Pin, freq
from ir_rx.nec import NEC_8

pin_ir = Pin(17, Pin.IN)

# Benutzerdefinierte Rückruffunktion
def callback(data, addr, ctrl):
    if data < 0: # NEC-Protokoll sendet Wiederholungscodes.
        pass
    else:
        print(decodeKeyValue(data))

ir = NEC_8(pin_ir, callback) # Empfänger instanziiieren

```

Hier wird ein `ir`-Objekt instanziiert, das ständig die vom IR-Empfänger empfangenen Signale liest.

Die Ergebnisse werden im `data`-Parameter der Rückruffunktion gespeichert.

- Rückruffunktion - Wikipedia

Falls der IR-Empfänger doppelte Werte erhält (z. B. durch gedrückt Halten einer Taste), wird `data < 0`, und diese Daten müssen gefiltert werden.

Ansonsten wäre `data` ein verwendbarer Wert, jedoch in unverständlichem Code, und die Funktion `decodeKeyValue(data)` dient zur Entschlüsselung.

```

def decodeKeyValue(data):
    if data == 0x16:
        return "0"
    if data == 0x0C:
        return "1"
    if data == 0x18:
        return "2"
    if data == 0x5E:

```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

    return "3"
if data == 0x08:
    return "4"
if data == 0x1C:
    return "5"
if data == 0x5A:
    return "6"
if data == 0x42:
    return "7"
if data == 0x52:
    return "8"
if data == 0x4A:
    return "9"
if data == 0x09:
    return "+"
if data == 0x15:
    return "-"
if data == 0x7:
    return "EQ"
if data == 0x0D:
    return "U/SD"
if data == 0x19:
    return "CYCLE"
if data == 0x44:
    return "PLAY/PAUSE"
if data == 0x43:
    return "FORWARD"
if data == 0x40:
    return "BACKWARD"
if data == 0x45:
    return "POWER"
if data == 0x47:
    return "MUTE"
if data == 0x46:
    return "MODE"
return "ERROR"

```

Falls wir die Taste **1** drücken, gibt der IR-Empfänger einen Wert wie **0x0C** aus, der entschlüsselt werden muss, um der spezifischen Taste zu entsprechen.

Es folgen einige Debug-Funktionen. Diese sind wichtig, stehen jedoch nicht im direkten Zusammenhang mit dem gewünschten Effekt, daher sind sie im Programm enthalten.

```

from ir_rx.print_error import print_error

ir.error_function(print_error) # Debug-Informationen anzeigen

```

Abschließend verwenden wir eine leere Schleife als Hauptprogramm und nutzen *try-except*, um das Programm mit Beendigung des *ir*-Objekts zu schließen.

```

try:
    while True:
        pass

```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
except KeyboardInterrupt:
    ir.close()
```

- [Try-Anweisung - Python-Dokumentation](#)

## 4.41 6.5 Funkfrequenz-Identifikation

Die Funkfrequenz-Identifikation (RFID) ist eine Technologie, die drahtlose Kommunikation zwischen einem Objekt (oder Tag) und einem abfragenden Gerät (oder Lesegerät) zur Identifizierung und Nachverfolgung nutzt. Die Übertragungsbereichsweite des Tags ist auf einige Meter begrenzt. Eine direkte Sichtlinie zwischen Lesegerät und Tag ist nicht zwingend erforderlich.

Die meisten Tags verfügen über einen integrierten Schaltkreis (IC) und eine Antenne. Neben der Datenspeicherung ermöglicht der Mikrochip die Kommunikation mit dem Lesegerät via Funkfrequenz (RF). Passive Tags haben keine eigenständige Energiequelle und sind für ihre Stromversorgung auf ein externes elektromagnetisches Signal des Lesegeräts angewiesen. Aktive Tags hingegen verfügen über eine unabhängige Energiequelle, etwa eine Batterie, was ihnen eine höhere Leistungsfähigkeit bei der Verarbeitung, Übertragung und Reichweite ermöglicht.

- [MFRC522 Modul](#)

### Benötigte Komponenten

Für dieses Projekt werden die folgenden Komponenten benötigt.

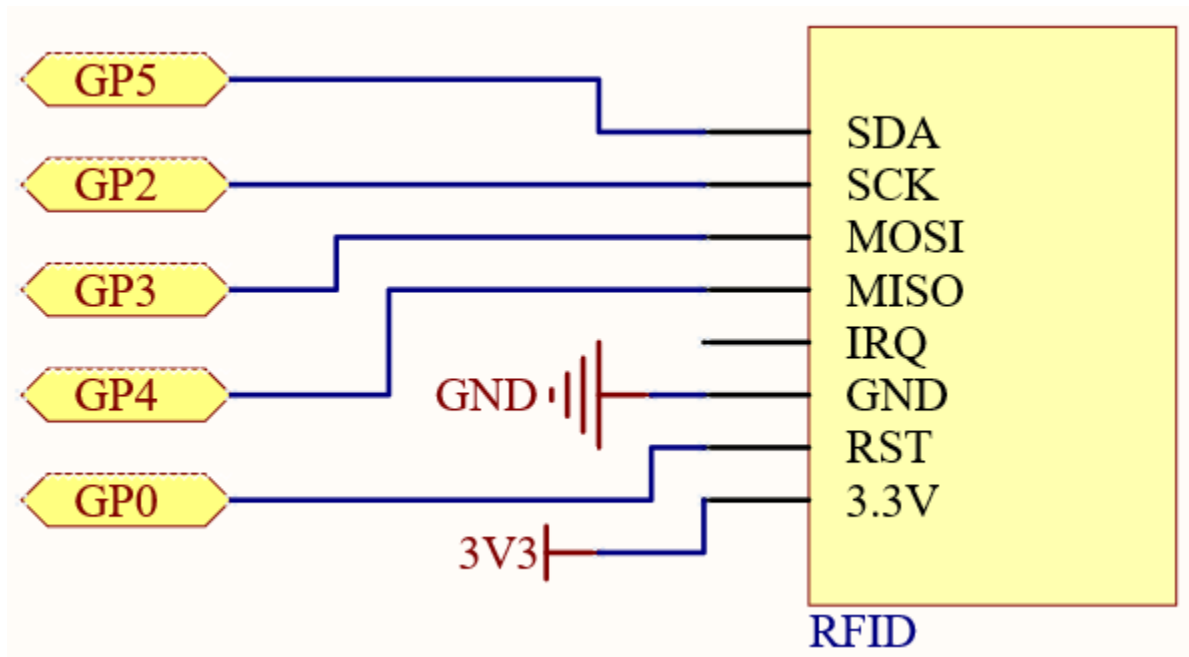
Ein Komplettsset ist natürlich praktisch, hier ist der Link:

Bezeichnung	ARTIKEL IN DIESEM KIT	LINK
Kepler Kit	450+	

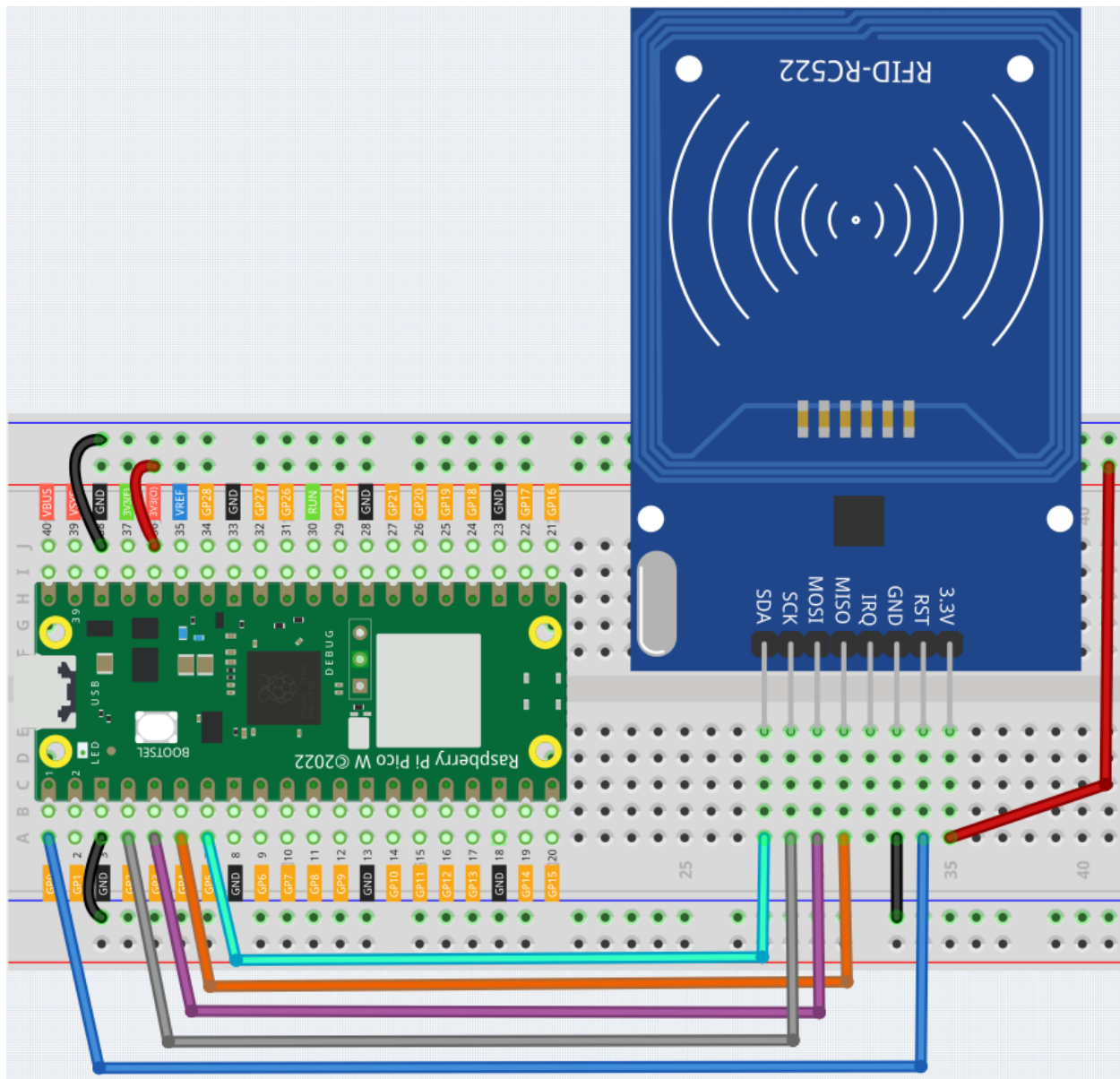
Die Komponenten können auch separat über die untenstehenden Links erworben werden.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<a href="#">Raspberry Pi Pico W</a>	1	
2	Micro-USB-Kabel	1	
3	<a href="#">Steckbrett</a>	1	
4	<a href="#">Jumperkabel</a>	mehrere	
5	<a href="#">MFRC522 Modul</a>	1	

### Schaltplan



Verkabelung



## Code

Bitte verwenden Sie die Bibliotheken im Ordner `mfr522`. Stellen Sie sicher, dass diese auf dem Pico W hochgeladen wurden. Eine detaillierte Anleitung finden Sie unter [1.4 Bibliotheken auf den Pico hochladen](#).

Die Hauptfunktion ist zweigeteilt:

- `6.5_rfid_write.py`: Dient dem Beschreiben der Karte (oder des Schlüssels).
- `6.5_rfid_read.py`: Dient dem Auslesen der Karte (oder des Schlüssels).

Öffnen Sie die Datei `6.5_rfid_write.py` im Pfad `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.

Nach dem Ausführen können Sie eine Nachricht im Shell-Fenster eingeben und die Karte (oder den Schlüssel) in die Nähe des MFRC522-Moduls halten, um die Nachricht darauf zu schreiben.

```

from mfrc522 import SimpleMFRC522

reader = SimpleMFRC522(spi_id=0,sck=2,miso=4,mosi=3,cs=5,rst=0)

def write():
    to_write = input("Please enter the message: ")
    print("Writing...Please place the card...")
    id, text = reader.write(to_write)
    print("ID: %s\nText: %s" % (id,text))

write()

```

Öffnen Sie die Datei 6.5\_rfid\_read.py im Pfad kepler-kit-main/micropython oder kopieren Sie diesen Code in Thonny und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5, um es auszuführen.

Nach dem Ausführen können Sie die auf der Karte (oder dem Schlüssel) gespeicherte Nachricht auslesen.

```

from mfrc522 import SimpleMFRC522

reader = SimpleMFRC522(spi_id=0,sck=2,miso=4,mosi=3,cs=5,rst=0)

def read():
    print("Reading...Please place the card...")
    id, text = reader.read()
    print("ID: %s\nText: %s" % (id,text))

read()

```

**Wie funktioniert es?**

```

from mfrc522 import SimpleMFRC522

reader = SimpleMFRC522(spi_id=0,sck=2,miso=4,mosi=3,cs=5,rst=0)

```

Instanziierung der Klasse SimpleMFRC522().

```
id, text = reader.read()
```

Diese Funktion dient dem Auslesen der Kartendaten. Bei erfolgreichem Auslesen werden ID und Text zurückgegeben.

```
id, text = reader.write("text")
```

Diese Funktion dient dem Beschreiben der Karte. Drücken Sie die **Eingabetaste**, um den Vorgang abzuschließen. text sind die auf die Karte zu schreibenden Informationen.

## 7. Unterhaltsame Projekte

## 4.42 7.1 Licht-Theremin

Ein Theremin ist ein elektronisches Musikinstrument, das keinen physischen Kontakt erfordert. Je nach Position der Hand des Spielers erzeugt es unterschiedliche Töne.

Üblicherweise besteht der Steuerungsbereich aus zwei Metallantennen, die die Position der Hände des Thereministen erfassen und Oszillatoren mit einer Hand und die Lautstärke mit der anderen steuern. Die elektrischen Signale vom Theremin werden verstärkt und an einen Lautsprecher gesendet.

Zwar können wir das gleiche Instrument mit Pico W nicht nachbilden, jedoch können wir mit einem Fotowiderstand und einem passiven Summer ein ähnliches Spielgefühl erzeugen.

- [Theremin - Wikipedia](#)

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

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

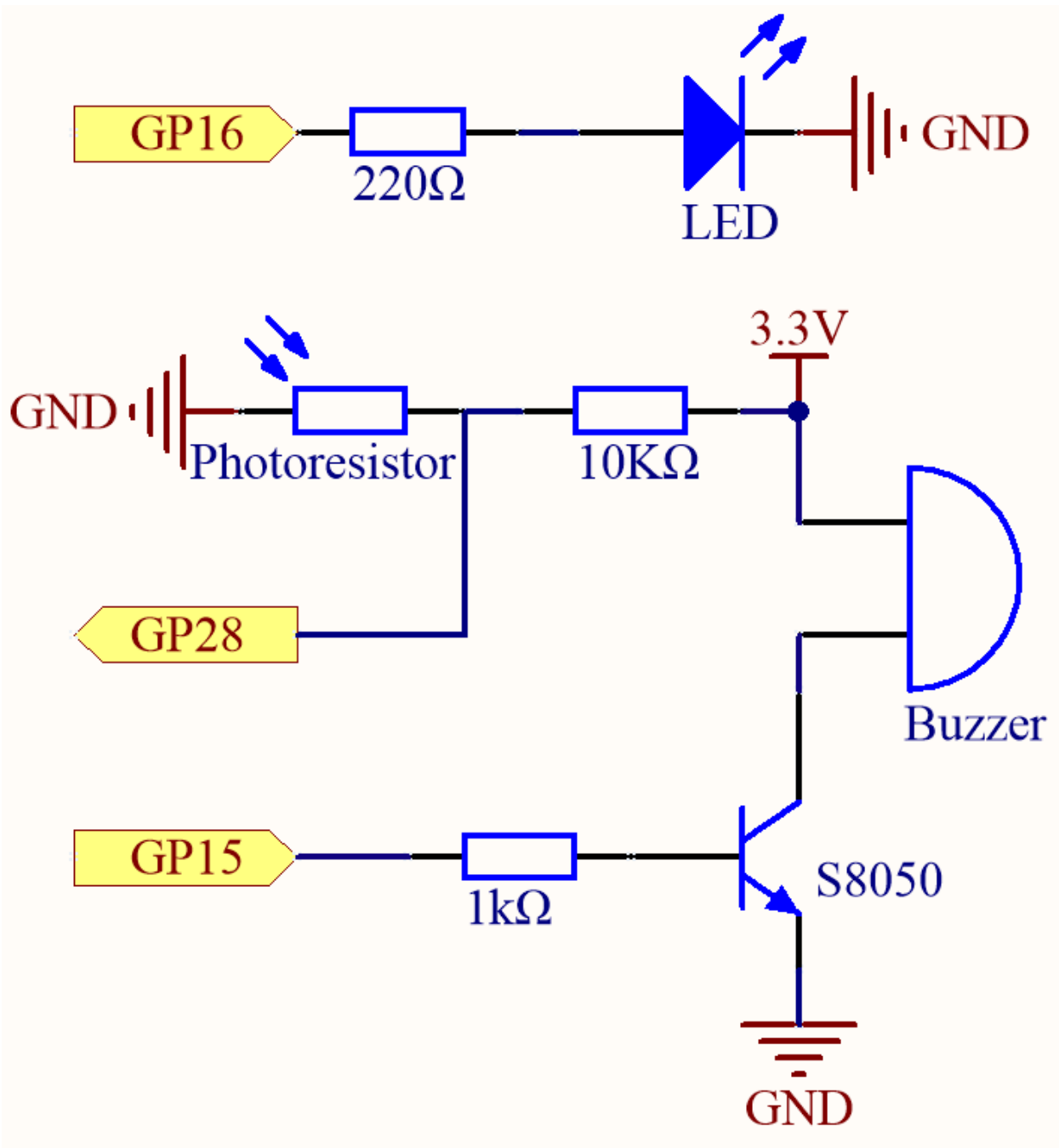
Name	ARTIKEL IN DIESEM KIT	LINK
Kepler Kit	450+	

Sie können die Teile auch einzeln über die unten stehenden Links erwerben.

SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>LED</i>	1	
6	<i>Transistor</i>	1(S8050)	
7	<i>Widerstand</i>	3(1K, 220, 10K)	
8	Aktiver <i>Summer</i>	1	
9	<i>Fotowiderstand</i>	1	

### Schaltplan



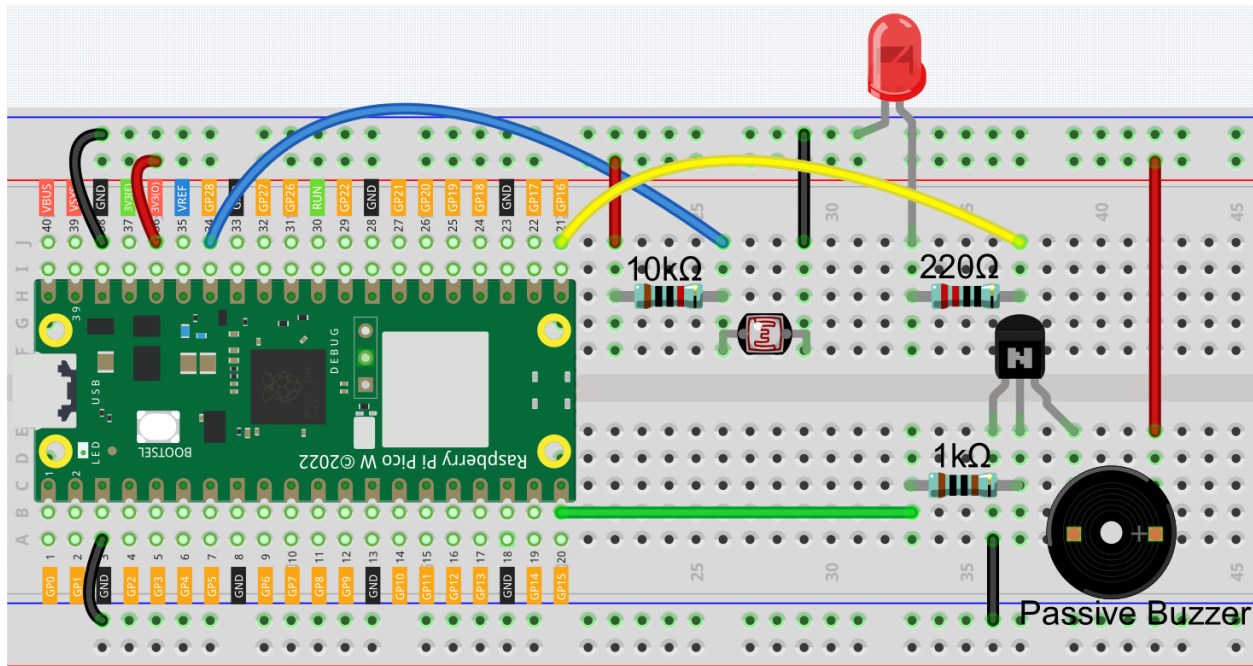


Bevor Sie mit dem Projekt beginnen, bewegen Sie Ihre Hand auf und ab über den Fotowiderstand, um den Lichtintensitätsbereich zu kalibrieren. Die mit GP16 verbundene LED dient zur Anzeige der Debugging-Zeit; sie leuchtet beim Debugging-Start und erlischt beim Debugging-Ende.

Wenn GP15 ein hohes Signal ausgibt, leitet der S8050 (NPN-Transistor) und der passive Summer ertönt.

Je stärker das Licht, desto kleiner ist der Wert an GP28; umgekehrt ist er größer, wenn das Licht schwächer ist. Durch Programmierung des Fotowiderstandswerts zur Beeinflussung der Frequenz des passiven Summers kann ein lichtempfindliches Gerät simuliert werden.

#### Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `7.1_light_thereimin.py` im Pfad `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken Sie auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der unteren rechten Ecke auszuwählen.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
import utime

led = machine.Pin(16, machine.Pin.OUT)
photoresistor = machine.ADC(28)
buzzer = machine.PWM(machine.Pin(15))

light_low=65535
light_high=0

def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def tone(pin,frequency,duration):
    pin.freq(frequency)
    pin.duty_u16(300000)
    utime.sleep_ms(duration)
    pin.duty_u16(0)

# calibrate the photoresistor max & min values.
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

timer_init_start = utime.ticks_ms()
led.value(1)
while utime.ticks_diff(utime.ticks_ms(), timer_init_start)<5000:
    light_value = photoresistor.read_u16()
    if light_value > light_high:
        light_high = light_value
    if light_value < light_low:
        light_low = light_value
led.value(0)

# play
while True:
    light_value = photoresistor.read_u16()
    pitch = int(interval_mapping(light_value,light_low,light_high,50,6000))
    if pitch > 50 :
        tone(buzzer,pitch,20)
    utime.sleep_ms(10)

```

Sobald das Programm startet, leuchtet die LED auf, und wir haben fünf Sekunden Zeit, um den Erfassungsbereich des Fotowiderstands zu kalibrieren.

Dies ist auf die verschiedenen Lichtverhältnisse zurückzuführen, unter denen das Gerät eingesetzt werden könnte (z. B. unterschiedliche Lichtintensitäten zu Mittag und in der Dämmerung) sowie auf die Höhe unserer Hände über dem Fotowiderstand. Sie müssen die maximale und minimale Höhe Ihrer Hand über dem Fotowiderstand festlegen, die zugleich die Höhe ist, in der Sie das Instrument spielen.

Nach Ablauf der fünf Sekunden erlischt die LED, und wir können unsere Hände über dem Fotowiderstand bewegen und spielen.

## 4.43 7.2 Raumtemperaturmessgerät

Mit einem Thermistor und einem I2C LCD1602 können wir ein Raumtemperaturmessgerät erstellen.

Dieses Projekt ist sehr einfach und basiert auf [2.13 Thermometer](#), wobei ein I2C LCD1602 zur Anzeige der Temperatur verwendet wird.

### Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

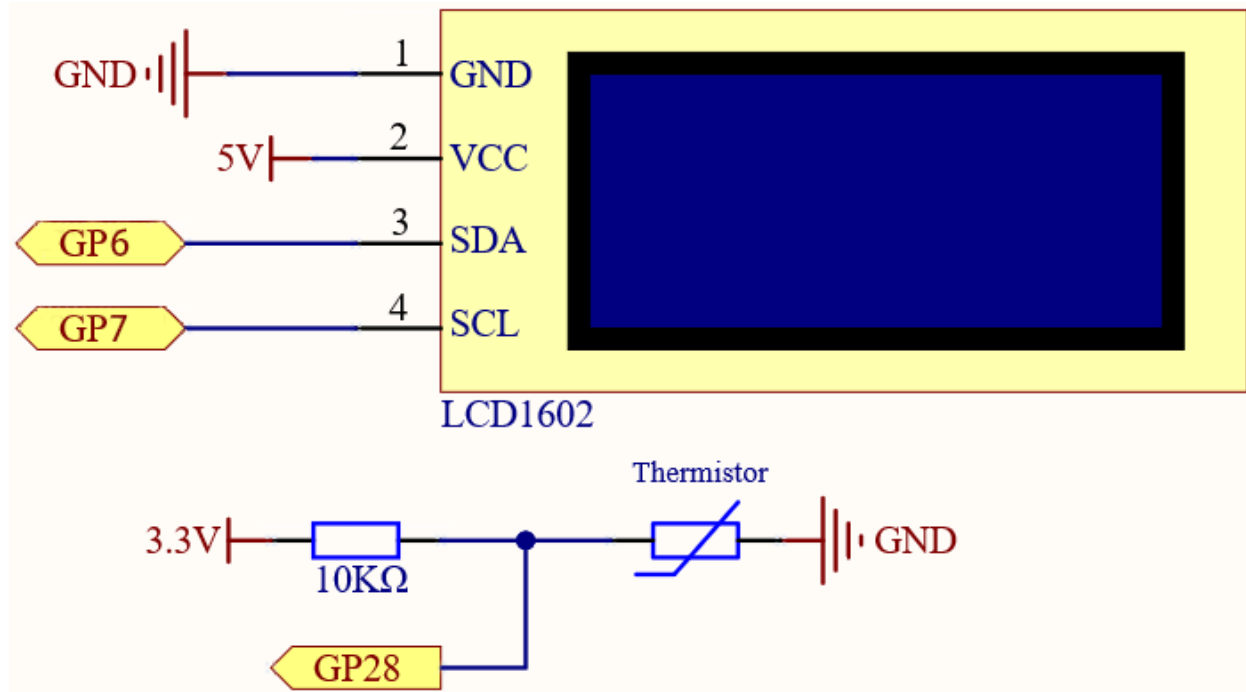
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
Kepler-Kit	450+	

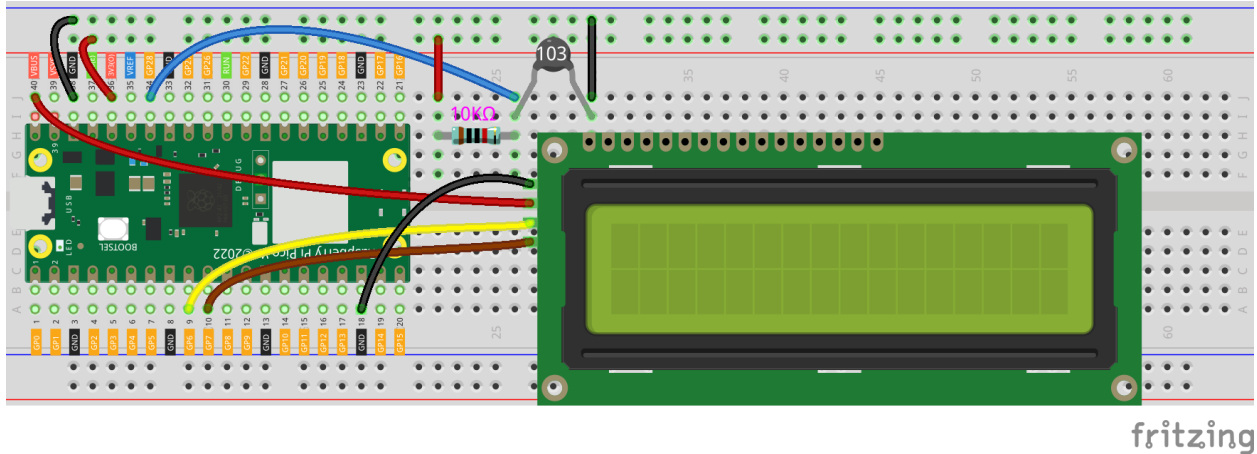
Sie können sie auch separat über die unten stehenden Links erwerben.

SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Thermistor</i>	1	
7	<i>I2C LCD1602</i>	1	

### Schaltplan



### Verkabelung



## Code

### Bemerkung:

- Öffnen Sie die Datei 7.2\_room\_temperature\_meter.py im Pfad kepler-kit-main/micropython oder kopieren Sie diesen Code in Thonny, und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, im unteren rechten Eck den „MicroPython (Raspberry Pi Pico)“-Interpreter auszuwählen.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
from lcd1602 import LCD
import machine
import utime
import math

thermistor = machine.ADC(28)
lcd = LCD()

while True:
    temperature_value = thermistor.read_u16()
    Vr = 3.3 * float(temperature_value) / 65535
    Rt = 10000 * Vr / (3.3 - Vr)
    temp = 1/(((math.log(Rt / 10000)) / 3950) + (1 / (273.15+25)))
    Cel = temp - 273.15
    #Fah = Cel * 1.8 + 32
    #print ('Celsius: %.2f C   Fahrenheit: %.2f F' % (Cel, Fah))
    #utime.sleep_ms(200)

    string = " Temperature is \n      " + str('{:.2f}'.format(Cel))+ " C"
    lcd.message(string)
    utime.sleep(1)
    lcd.clear()
```

Nach dem Ausführen des Programms wird die LCD die aktuelle Temperatur im Raum anzeigen.

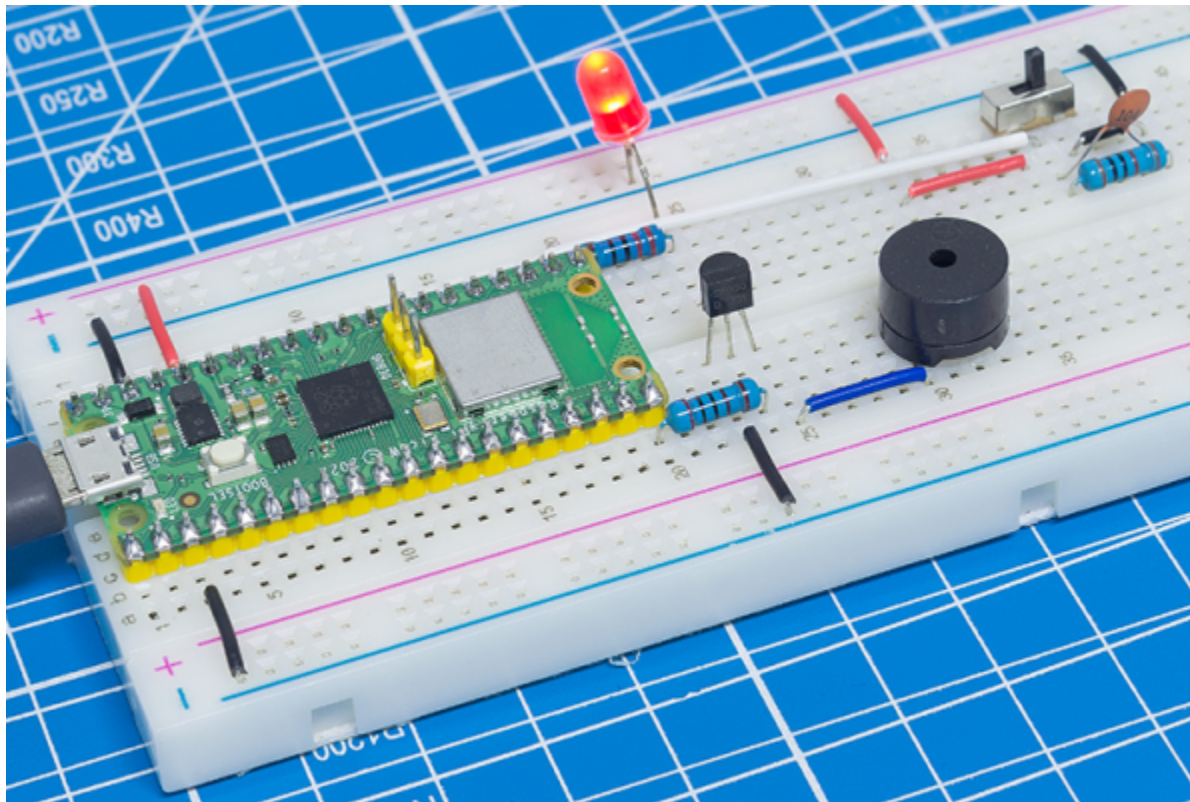
**Bemerkung:** Wenn der Code und die Verkabelung in Ordnung sind, aber das LCD dennoch keinen Inhalt anzeigt,

können Sie das Potentiometer auf der Rückseite drehen, um den Kontrast zu erhöhen.

### 4.44 7.3 Alarmanlagenlampe

Polizeilichter sind oft im wirklichen Leben (oder in Filmen) zu sehen. In der Regel werden sie zur Verkehrsregelung, als Warnmittel und als wichtiges Sicherheitszubehör für Polizeibeamte, Rettungsfahrzeuge, Feuerwehrautos und Baufahrzeuge eingesetzt. Wenn Sie ihre Lichter sehen oder ihre Sirene hören, sollten Sie vorsichtig sein, denn das bedeutet, dass Sie (oder die Menschen in Ihrer Umgebung) möglicherweise in Gefahr sind.

In diesem Projekt nutzen wir eine LED und einen Summer, um ein kleines Warnlicht zu schaffen, das über einen Schiebeschalter aktiviert wird.



#### Benötigte Komponenten

Für dieses Projekt werden folgende Komponenten benötigt.

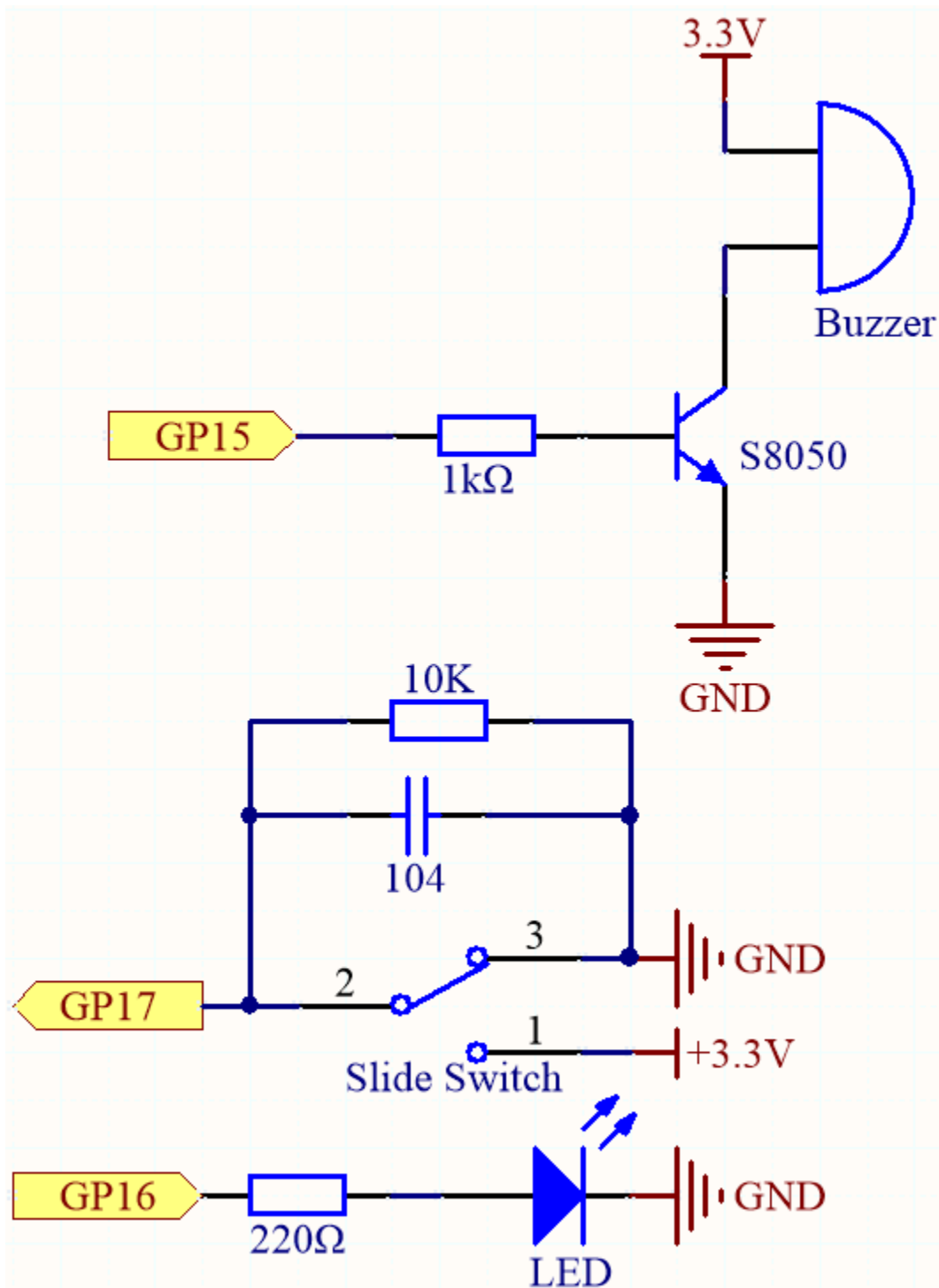
Es ist definitiv praktisch, ein komplettes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
Kepler Kit	450+	

Sie können die Teile auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	MEN- GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>LED</i>	1	
6	<i>Transistor</i>	1(S8050)	
7	<i>Widerstand</i>	3(1K, 220, 10K)	
8	Passiver <i>Summer</i>	1	
9	<i>Kondensator</i>	1(104)	
10	<i>Schiebeschalter</i>	1	

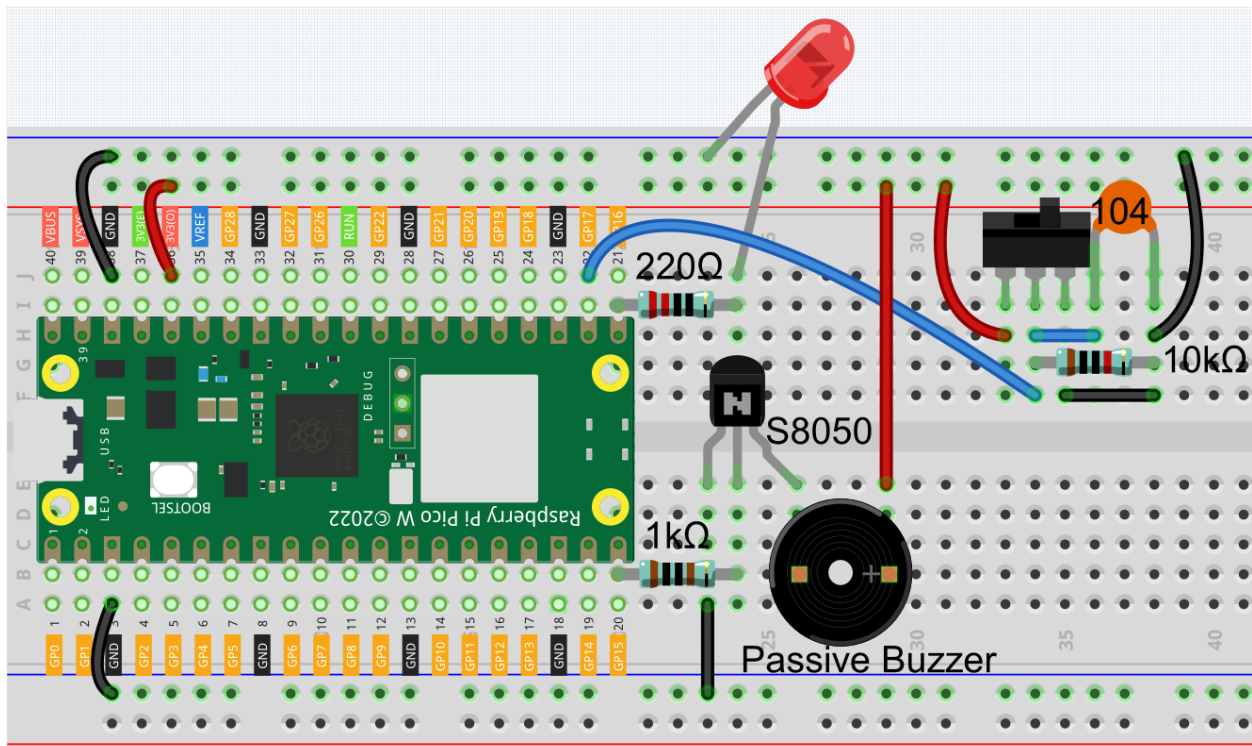
### Schaltplan



- GP17 ist mit dem mittleren Pin des Schiebeschalters verbunden, parallel dazu sind ein 10K-Widerstand und ein Kondensator (Filter) an GND angeschlossen. Dies ermöglicht dem Schiebeschalter, ein konstant hohes oder niedriges Signal auszugeben, wenn er nach links oder rechts bewegt wird.
- Sobald GP15 hoch ist, leitet der NPN-Transistor und der passive Summer beginnt zu tönen. Dieser Summer wird programmiert, um in der Frequenz allmählich anzusteigen und so einen Sirenenton zu erzeugen.
- An GP16 ist eine LED angeschlossen, die so programmiert ist, dass ihre Helligkeit periodisch wechselt, um eine Sirene zu simulieren.



## Verdrahtung



## Code

## Bemerkung:

- Öffnen Sie die Datei `7.3_alarm_siren_lamp.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, im unteren rechten Bereich auf den „MicroPython (Raspberry Pi Pico)“-Interpreter zu klicken.
- Für ausführliche Anleitungen beziehen Sie sich bitte auf [Code direkt öffnen und ausführen](#).

```
import machine
import time

buzzer = machine.PWM(machine.Pin(15))
led = machine.PWM(machine.Pin(16))
led.freq(1000)

switch = machine.Pin(17, machine.Pin.IN)

def noTone(pin):
    pin.duty_u16(0)

def tone(pin, frequency):
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

pin.freq(frequency)
pin.duty_u16(300000)

def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def toggle(pin):
    global bell_flag
    bell_flag = not bell_flag
    print(bell_flag)
    if bell_flag:
        switch.irq(trigger=machine.Pin.IRQ_FALLING, handler=toggle)
    else:
        switch.irq(trigger=machine.Pin.IRQ_RISING, handler=toggle)

bell_flag = False
switch.irq(trigger=machine.Pin.IRQ_RISING, handler=toggle)

while True:
    if bell_flag == True:
        for i in range(0,100,2):
            led.duty_u16(int(interval_mapping(i,0,100,0,65535)))
            tone(buzzer,int(interval_mapping(i,0,100,130,800)))
            time.sleep_ms(10)
        else:
            noTone(buzzer)
            led.duty_u16(0)

```

Sobald das Programm läuft, verschieben Sie den Schiebeschalter nach links (bei Ihnen kann es auch rechts sein, je nachdem, wie Ihr Schiebeschalter verdrahtet ist). Der Summer gibt dann einen aufsteigenden Warnton ab und die LED ändert entsprechend ihre Helligkeit; verschieben Sie den Schiebeschalter nach rechts und der Summer und die LED hören auf zu arbeiten.

## 4.45 7.4 Personen Zähler

In großen Einkaufszentren, Flughäfen, Bahnhöfen, Museen und anderen öffentlichen Räumen wie Ausstellungshallen ist die Erfassung der Besucherströme unerlässlich.

In Flughäfen und Bahnhöfen beispielsweise muss die Personenanzahl strikt überwacht werden, um für Sicherheit und einen reibungslosen Ablauf zu sorgen. Auch in Einkaufszentren lässt sich so herausfinden, zu welchen Zeiten besonders viele Besucher anwesend sind und welches Umsatzpotenzial pro Besucher besteht. Daraus können Verbrauchsgewohnheiten analysiert und der Umsatz gesteigert werden.

Personenzähler helfen dabei, den Betrieb dieser öffentlichen Einrichtungen effizient zu organisieren.

Ein einfacher Personenzähler wird hier mit einem PIR-Sensor und einer 4-stelligen 7-Segment-Anzeige realisiert.

### Benötigte Komponenten

Für dieses Projekt werden die folgenden Komponenten benötigt.

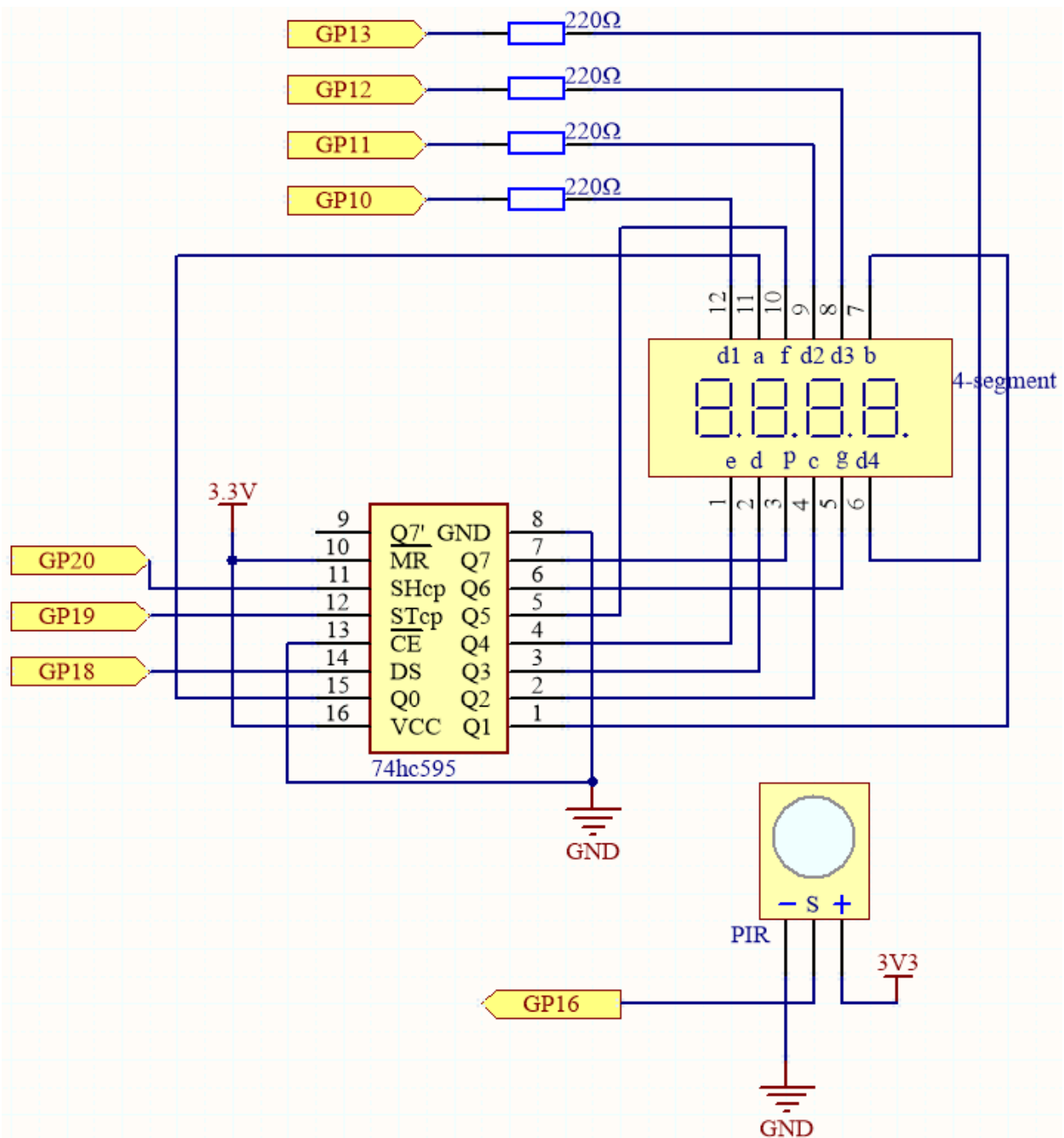
Ein Gesamtset ist sicherlich praktisch, hier ist der Link:

Bezeichnung	ELEMENTE IN DIESEM KIT	LINK
Kepler Kit	450+	

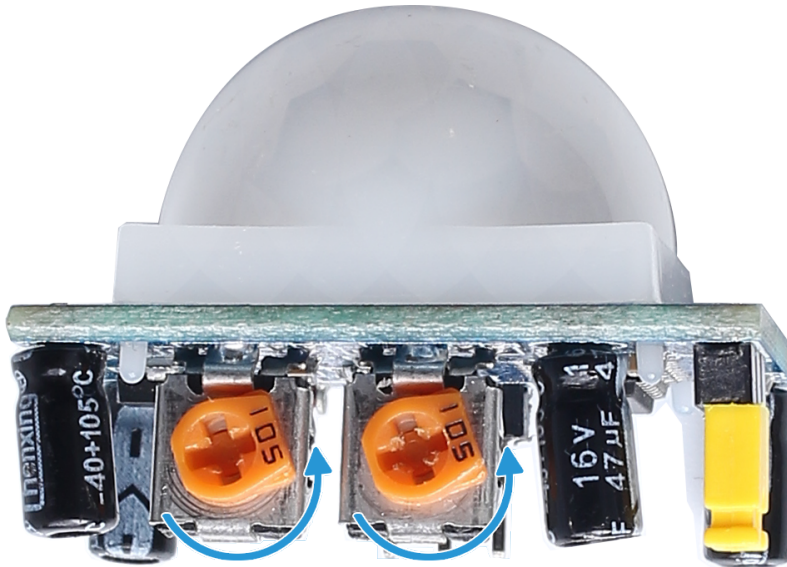
Die Komponenten können auch einzeln über die untenstehenden Links erworben werden.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	4(220)	
6	<i>4-stellige 7-Segment-Anzeige</i>	1	
7	<i>74HC595</i>	1	
8	<i>PIR-Bewegungssensormodul</i>	1	

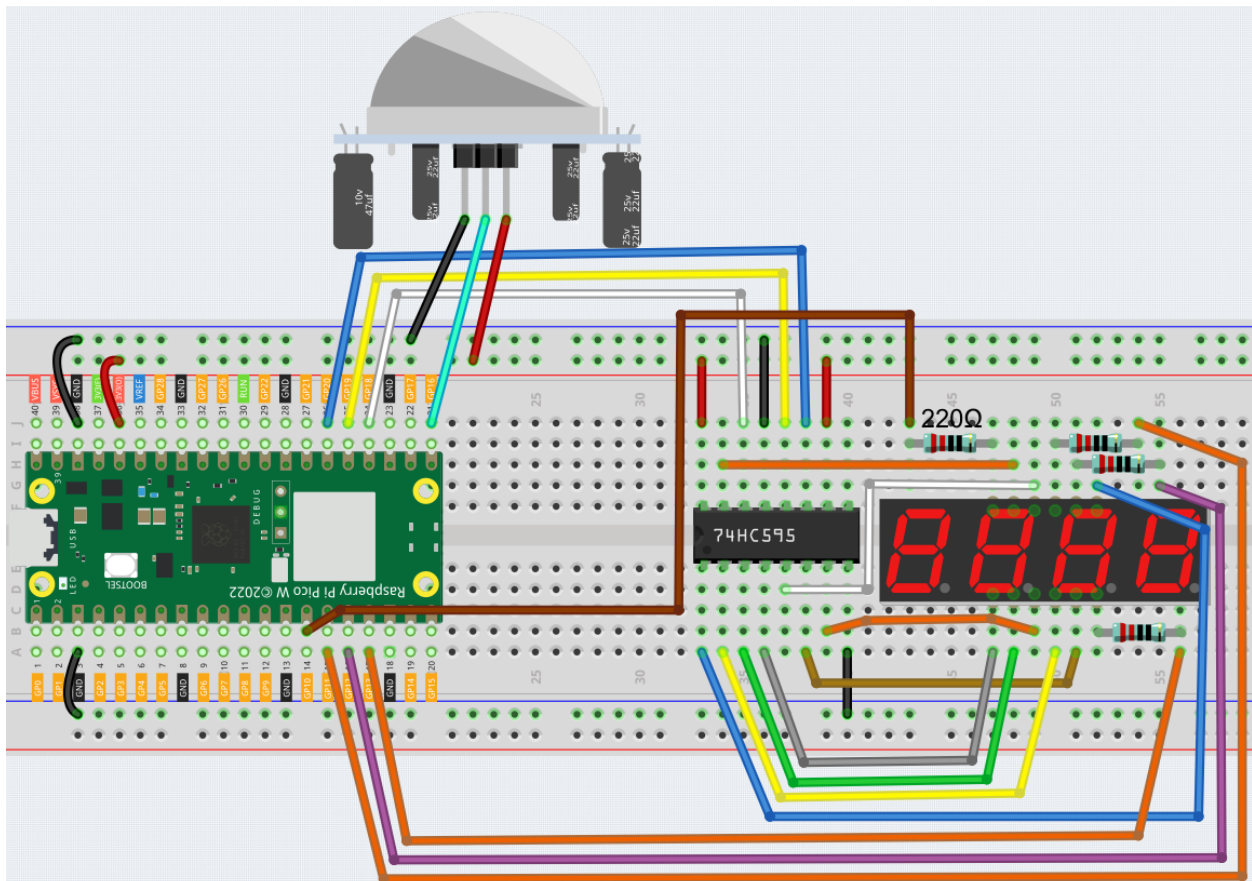
### Schaltplan



- Diese Schaltung basiert auf dem [5.3 Zeitmesser](#) und wird durch ein PIR-Modul ergänzt.
- Der PIR-Sensor sendet ein etwa 2,8 Sekunden langes High-Signal aus, wenn jemand vorbeigeht.
- Das PIR-Modul hat zwei Potentiometer: eines für die Empfindlichkeit und eines für die Erfassungsreichweite. Um das PIR-Modul optimal einzustellen, sollten beide Potentiometer vollständig gegen den Uhrzeigersinn gedreht werden.



## Verkabelung



## Code

### Bemerkung:

- Öffnen Sie die Datei `7.4_passenger_counter.py` im Pfad `kepler-kit-main/micropython` oder kopieren

Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.

- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der unteren rechten Ecke auszuwählen.
  - Für detaillierte Anleitungen siehe *Code direkt öffnen und ausführen*.
- 

```
import machine
import time

pir_sensor = machine.Pin(16, machine.Pin.IN)

SEGCODE = [0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f]

sdi = machine.Pin(18,machine.Pin.OUT)
rclk = machine.Pin(19,machine.Pin.OUT)
srclk = machine.Pin(20,machine.Pin.OUT)

placePin = []
pin = [10,13,12,11]
for i in range(4):
    placePin.append(None)
    placePin[i] = machine.Pin(pin[i], machine.Pin.OUT)

count = 0

def pickDigit(digit):
    for i in range(4):
        placePin[i].value(1)
    placePin[digit].value(0)

def clearDisplay():
    hc595_shift(0x00)

def hc595_shift(dat):
    rclk.low()
    time.sleep_us(200)
    for bit in range(7, -1, -1):
        srclk.low()
        time.sleep_us(200)
        value = 1 & (dat >> bit)
        sdi.value(value)
        time.sleep_us(200)
        srclk.high()
        time.sleep_us(200)
    time.sleep_us(200)
    rclk.high()

def motion_detected(pin):
    global count
    count = count+1
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

pir_sensor.irq(trigger=machine.Pin.IRQ_RISING, handler=motion_detected)

while True:
    #print(count)

    pickDigit(0)
    hc595_shift(SEGCODE[count%10])

    pickDigit(1)
    hc595_shift(SEGCODE[count%100//10])

    pickDigit(2)
    hc595_shift(SEGCODE[count%1000//100])

    pickDigit(3)
    hc595_shift(SEGCODE[count%10000//1000])

```

Beim Ausführen des Codes wird die Zahl auf der 4-stelligen 7-Segment-Anzeige um eins erhöht, sobald jemand vor dem PIR-Modul vorbeigeht.

## 4.46 7.5 SPIEL - 10 Sekunden

Um Ihre Konzentration auf die Probe zu stellen, bauen Sie mit mir ein Spielgerät. Kreieren Sie einen Zauberstab, indem Sie den Kipp-Schalter mit einem Stab verbinden. Wenn Sie den Zauberstab schütteln, beginnt die 4-stellige 7-Segment-Anzeige mit dem Zählen. Schütteln Sie ihn erneut, stoppt das Zählen. Um zu gewinnen, muss der angezeigte Zählerstand bei **10.00** stehen bleiben. Spielen Sie das Spiel mit Ihren Freunden, um herauszufinden, wer der Zeitmagier ist.

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

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

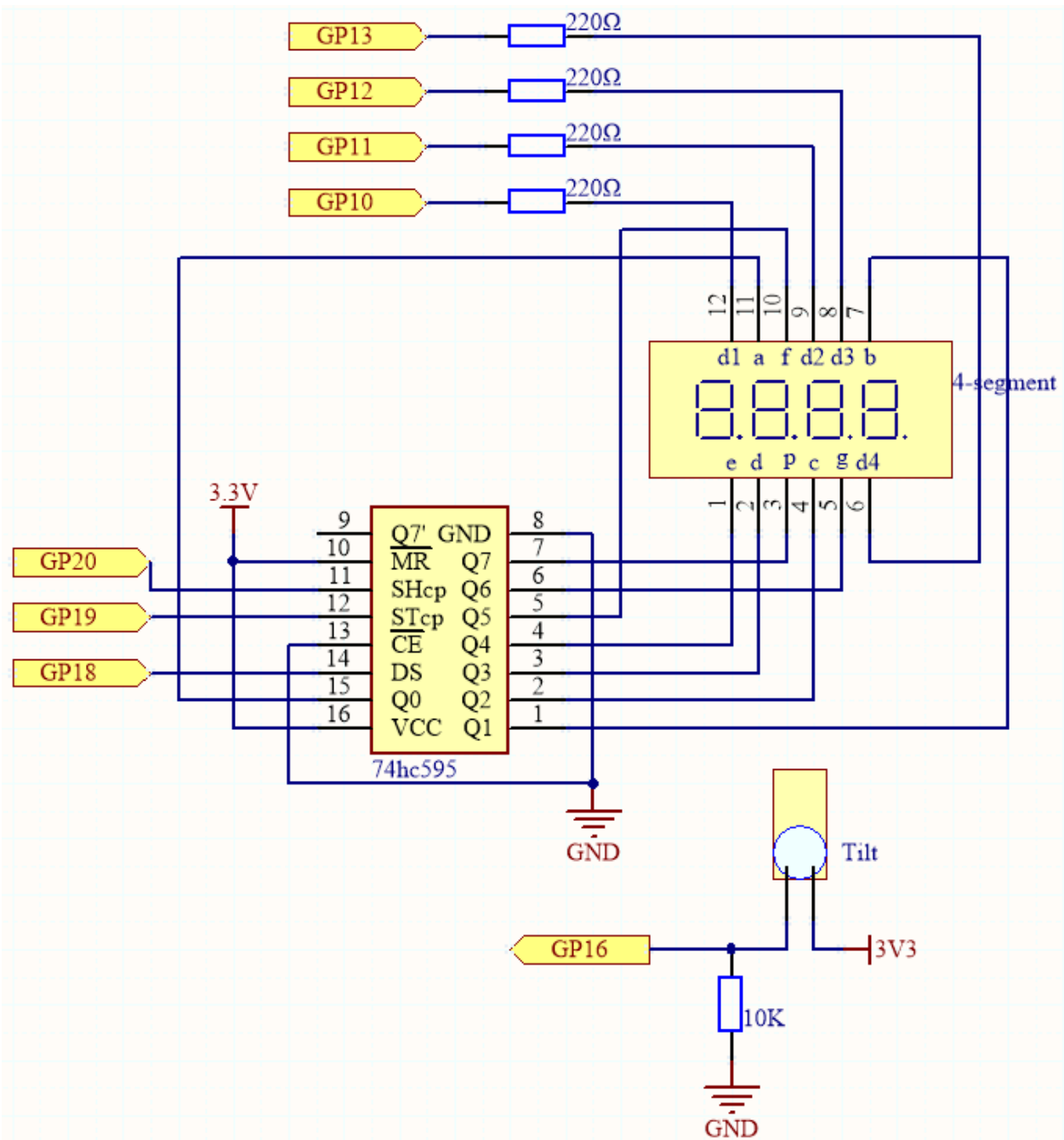
Name	ARTIKEL IN DIESEM KIT	LINK
Kepler Kit	450+	

Sie können die Teile auch separat unter den folgenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	5(4-220, 1-10K)	
6	<i>4-stellige 7-Segment-Anzeige</i>	1	
7	<i>74HC595</i>	1	
8	<i>Neigungsschalter</i>	1	

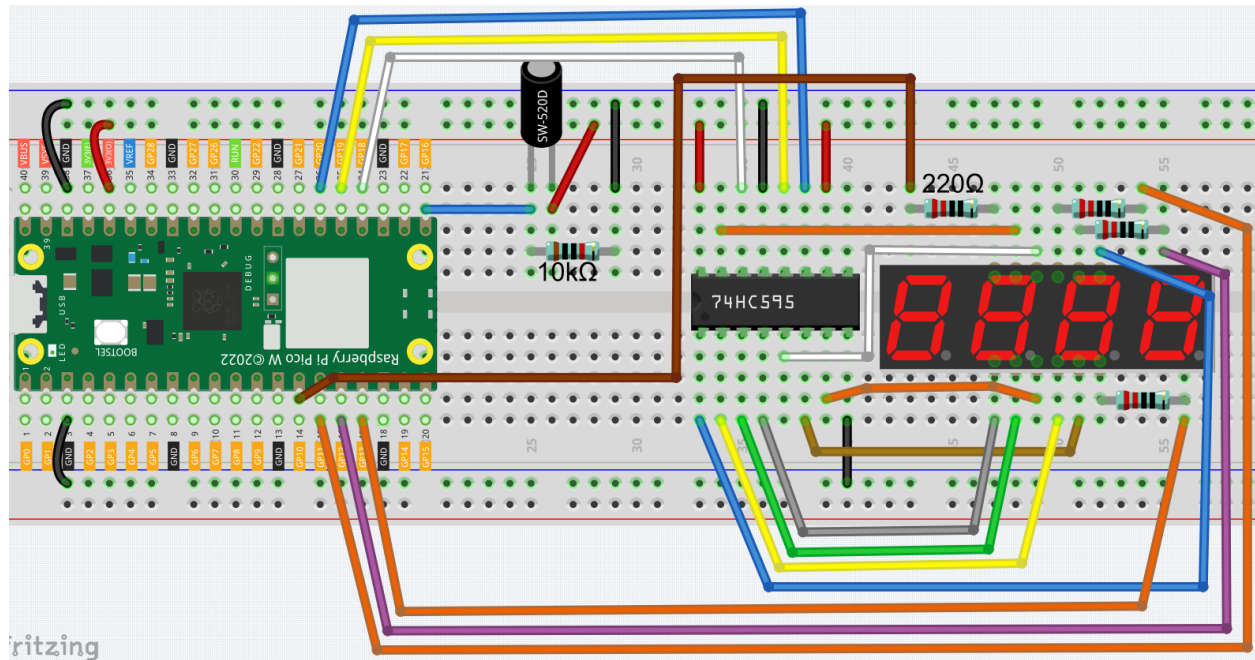
**Schaltplan**





- Diese Schaltung basiert auf [5.3 Zeitmesser](#), ergänzt durch einen Kipp-Schalter.
- GP16 ist hoch, wenn der Kipp-Schalter aufrecht ist; niedrig, wenn gekippt.

#### Verkabelung



## Code

### Bemerkung:

- Öffnen Sie die Datei 7.5\_game\_10\_second.py im Pfad kepler-kit-main/micropython oder kopieren Sie diesen Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der unteren rechten Ecke auszuwählen.
- Für detaillierte Anleitungen verweisen wir auf [Code direkt öffnen und ausführen](#).

```
import machine
import time

SEGCODE = [0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f]

sdi = machine.Pin(18, machine.Pin.OUT)
rclk = machine.Pin(19, machine.Pin.OUT)
srcclk = machine.Pin(20, machine.Pin.OUT)

placePin = []
pin = [10, 13, 12, 11]
for i in range(4):
    placePin.append(None)
    placePin[i] = machine.Pin(pin[i], machine.Pin.OUT)

def pickDigit(digit):
    for i in range(4):
        placePin[i].value(1)
    placePin[digit].value(0)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

def clearDisplay():
    hc595_shift(0x00)

def hc595_shift(dat):
    rclk.low()
    time.sleep_us(200)
    for bit in range(7, -1, -1):
        srclk.low()
        time.sleep_us(200)
        value = 1 & (dat >> bit)
        sdi.value(value)
        time.sleep_us(200)
        srclk.high()
        time.sleep_us(200)
    time.sleep_us(200)
    rclk.high()
    #time.sleep_us(200)

def display(num):

    pickDigit(0)
    hc595_shift(SEGCODE[num%10])

    pickDigit(1)
    hc595_shift(SEGCODE[num%100//10])

    pickDigit(2)
    hc595_shift(SEGCODE[num%1000//100]+0x80)

    pickDigit(3)
    hc595_shift(SEGCODE[num%10000//1000])

tilt_switch = machine.Pin(16,machine.Pin.IN)

count_flag = False

def shake(pin):
    global timeStart,count_flag
    count_flag = not count_flag
    if count_flag == True:
        timeStart = time.ticks_ms()

tilt_switch.irq(trigger=machine.Pin.IRQ_RISING, handler=shake)

count = 0
while True:
    if count_flag == True:
        count = int((time.ticks_ms()-timeStart)/10)
        display(count)

```

Die 4-stellige 7-Segment-Anzeige beginnt mit dem Zählen, sobald Sie den Zauberstab schütteln, und stoppt, wenn Sie ihn erneut schütteln. Sie gewinnen, wenn es Ihnen gelingt, den angezeigten Zählstand bei 10,00 zu halten. Das Spiel wird mit einem weiteren Schütteln fortgesetzt.

### 4.47 7.6 Ampel

Eine **Ampel** ist ein Signalgerät, das an Straßenkreuzungen, Fußgängerüberwegen und anderen Orten aufgestellt ist, um den Verkehrsfluss zu steuern.

Ampeln werden durch die **Wiener Übereinkommen über Straßenverkehrszeichen** standardisiert und geben den Verkehrsteilnehmern durch abwechselndes Leuchten von LEDs in drei Standardfarben das Recht auf Vorfahrt.

- **Rotes Licht:** Bei einem blinkenden roten Licht muss der Verkehr anhalten, vergleichbar mit einem Stoppschild.
- **Gelbes Licht:** Ein Warnsignal, dass das Licht bald auf Rot umschaltet. Gelbe Lichter werden in verschiedenen Ländern (Regionen) unterschiedlich interpretiert.
- **Grünes Licht:** Erlaubt es dem Verkehr, in die angezeigte Richtung zu fahren.

In diesem Projekt werden wir drei Farben von LEDs verwenden, um die Ampelphasen darzustellen, sowie eine 4-stellige 7-Segment-Anzeige, um die Dauer jeder Verkehrsphase anzuzeigen.

#### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

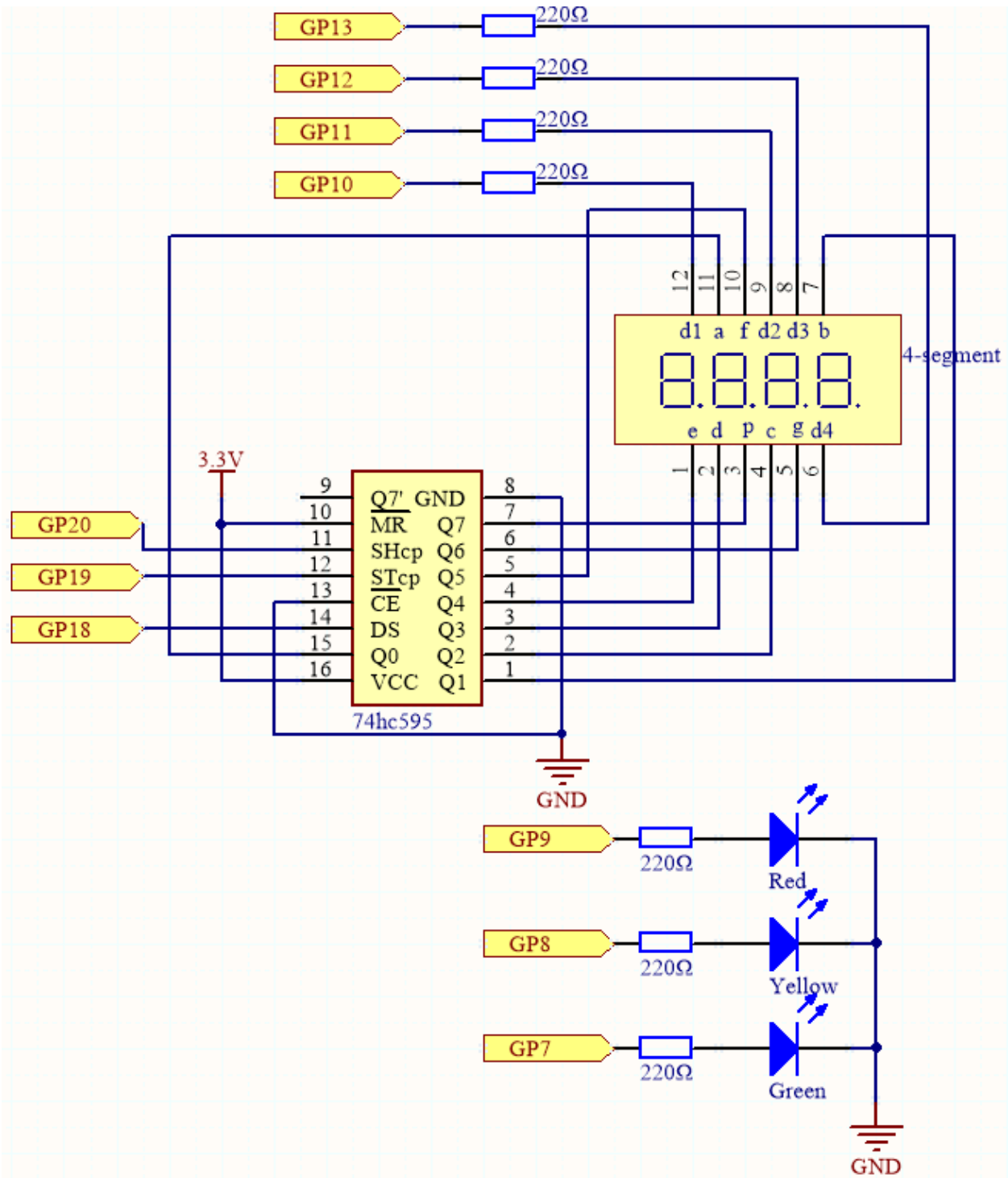
Es ist definitiv praktisch, ein ganzes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die Einzelteile auch über die unten stehenden Links separat erwerben.

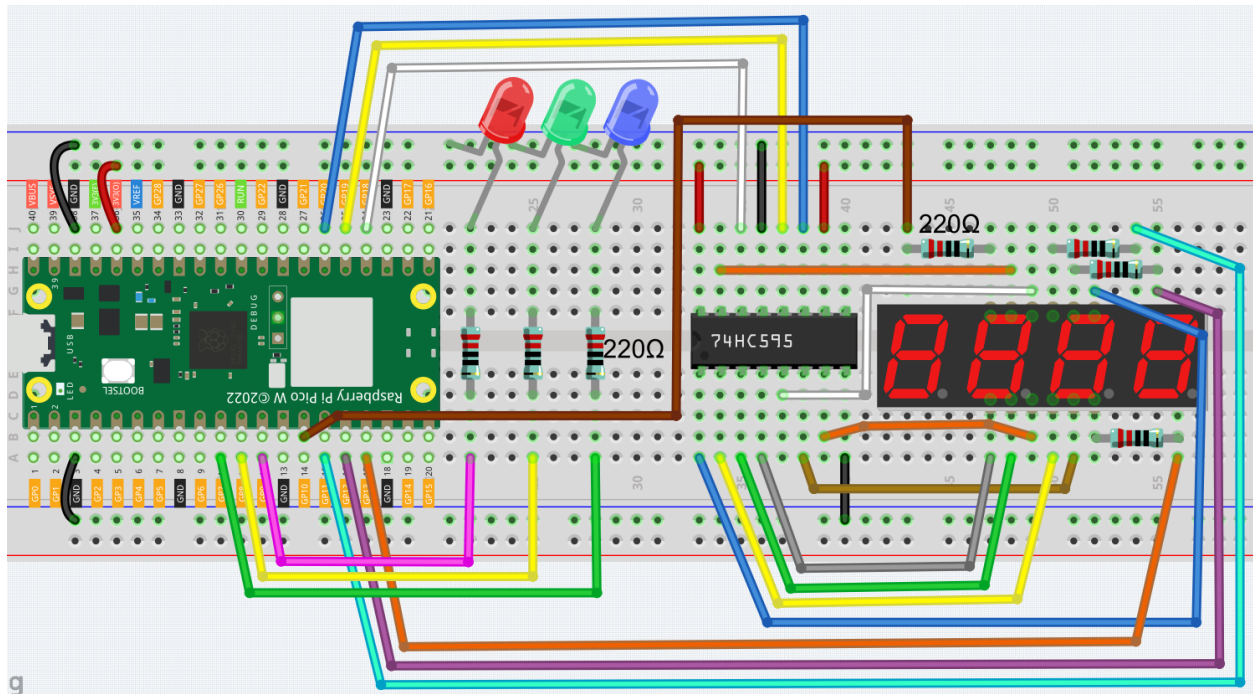
SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	7 (220)	
6	<i>4-stellige 7-Segment-Anzeige</i>	1	
7	<i>74HC595</i>	1	
8	<i>LED</i>	1	

#### Schaltplan



- Diese Schaltung basiert auf dem 5.3 *Zeitmesser*, ergänzt durch 3 LEDs.
- Die 3 roten, gelben und grünen LEDs sind jeweils an GP7~GP9 angeschlossen.

#### Verkabelung



## Code

### Bemerkung:

- Öffnen Sie die Datei `7.6_traffic_light.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny. Klicken Sie anschließend auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, im unteren rechten Eck den Interpreter „MicroPython (Raspberry Pi Pico)“ auszuwählen.
- Für detaillierte Anleitungen verweisen wir auf [Code direkt öffnen und ausführen](#).

```
import machine
import time
from machine import Timer

# [Green, Yellow, Red]
lightTime=[30, 5, 30]

# display
SEGCODE = [0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f]

sdi = machine.Pin(18,machine.Pin.OUT)
rclk = machine.Pin(19,machine.Pin.OUT)
srcclk = machine.Pin(20,machine.Pin.OUT)

placePin = []
pin = [10,13,12,11]
for i in range(4):
    placePin.append(None)
    placePin[i] = machine.Pin(pin[i], machine.Pin.OUT)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

def pickDigit(digit):
    for i in range(4):
        placePin[i].value(1)
    placePin[digit].value(0)

def clearDisplay():
    hc595_shift(0x00)

def hc595_shift(dat):
    rclk.low()
    time.sleep_us(200)
    for bit in range(7, -1, -1):
        srclk.low()
        time.sleep_us(200)
        value = 1 & (dat >> bit)
        sdi.value(value)
        time.sleep_us(200)
        srclk.high()
        time.sleep_us(200)
    time.sleep_us(200)
    rclk.high()

def display(num):

    pickDigit(0)
    hc595_shift(SEGCODE[num%10])

    pickDigit(1)
    hc595_shift(SEGCODE[num%100//10])

    pickDigit(2)
    hc595_shift(SEGCODE[num%1000//100])

    pickDigit(3)
    hc595_shift(SEGCODE[num%10000//1000])

# led
# 9Red, 8Yellow, 7Green
pin = [7, 8, 9]
led=[]
for i in range(3):
    led.append(None)
    led[i] = machine.Pin(pin[i], machine.Pin.OUT)

def lightup(state):
    for i in range(3):
        led[i].value(0)
    led[state].value(1)

# timer
counter = 0

```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
color_state= 0

def time_count(ev):
    global counter, color_state
    counter -= 1
    if counter <= 0:
        color_state = (color_state+1) % 3
        counter = lightTime[color_state]

tim = Timer(period=1000, mode=Timer.PERIODIC, callback=time_count)

while True:
    display(counter)
    lightup(color_state)
```

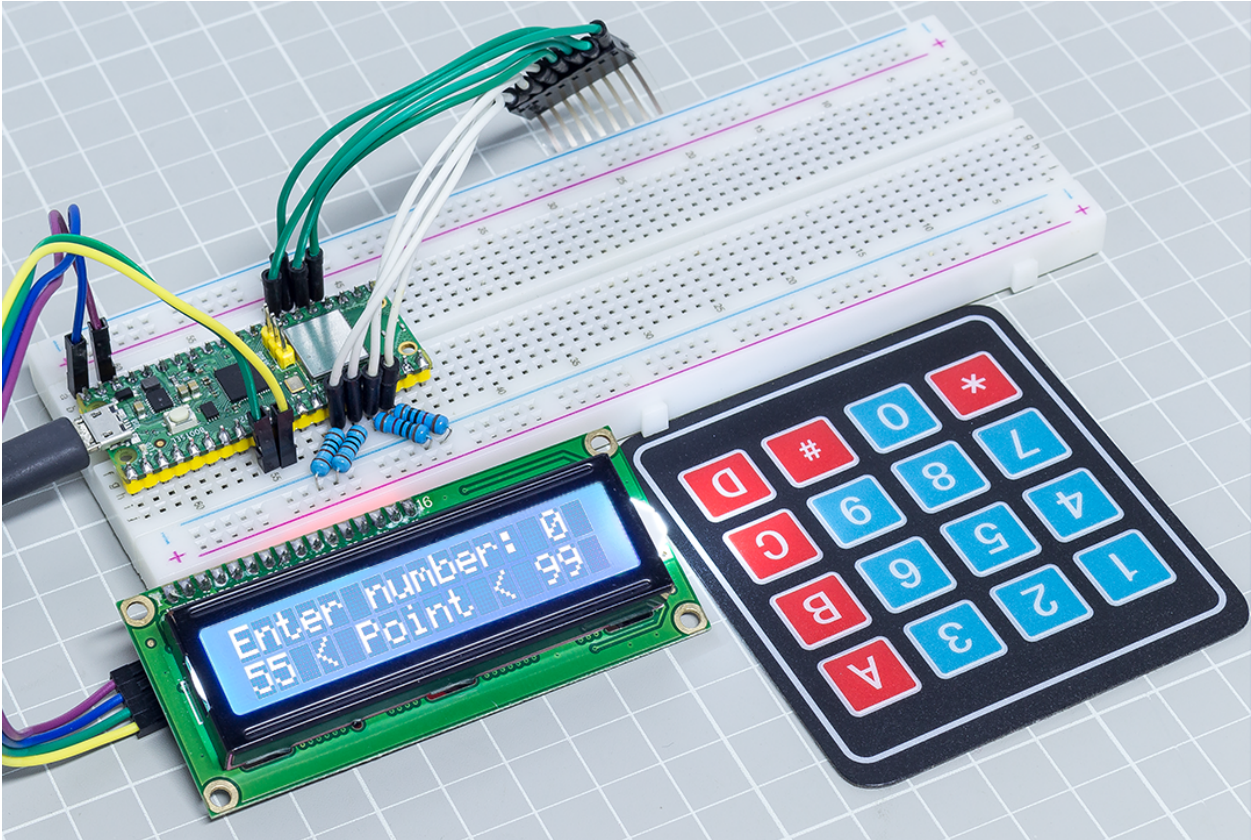
Wenn der Code ausgeführt wird, leuchtet die grüne LED für 30 Sekunden, die gelbe LED für 5 Sekunden und die rote LED für 30 Sekunden.

## 4.48 7.7 Zahlenraten

Zahlenraten ist ein unterhaltsames Partyspiel, bei dem Sie und Ihre Freunde Zahlen von 0 bis 99 eingeben. Mit jeder Eingabe schrumpft der Zahlenbereich, bis ein Spieler das Rätsel richtig löst. Dann ist dieser Spieler besiegt und wird bestraft.

Zum Beispiel, wenn die Glückszahl 51 ist, die für die Spieler unsichtbar ist, und Spieler 1 die Zahl 50 eingibt, ändert sich die Aufforderung zu 50 - 99; wenn Spieler 2 die Zahl 70 eingibt, ändert sich der Bereich zu 50 - 70; wenn Spieler 3 die Zahl 51 eingibt, hat er Pech gehabt. In diesem Fall werden die Zahlen über eine Tastatur eingegeben und die Ergebnisse auf einem LCD-Bildschirm angezeigt.





**Benötigte Komponenten**

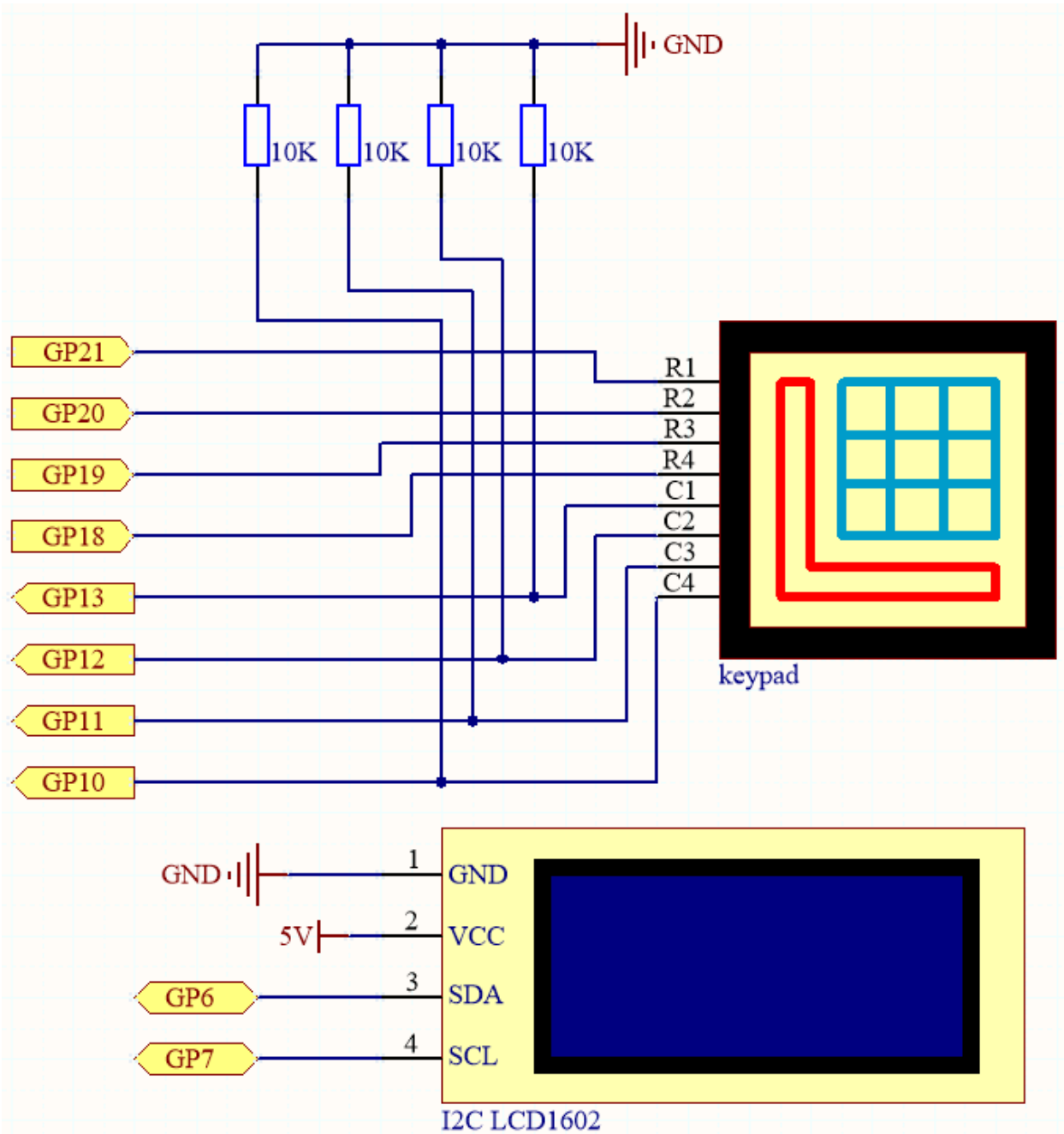
Für dieses Projekt benötigen wir folgende Komponenten.  
Ein Komplettsset zu kaufen ist definitiv praktisch, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
Kepler Kit	450+	

Sie können die Komponenten auch einzeln über die untenstehenden Links erwerben.

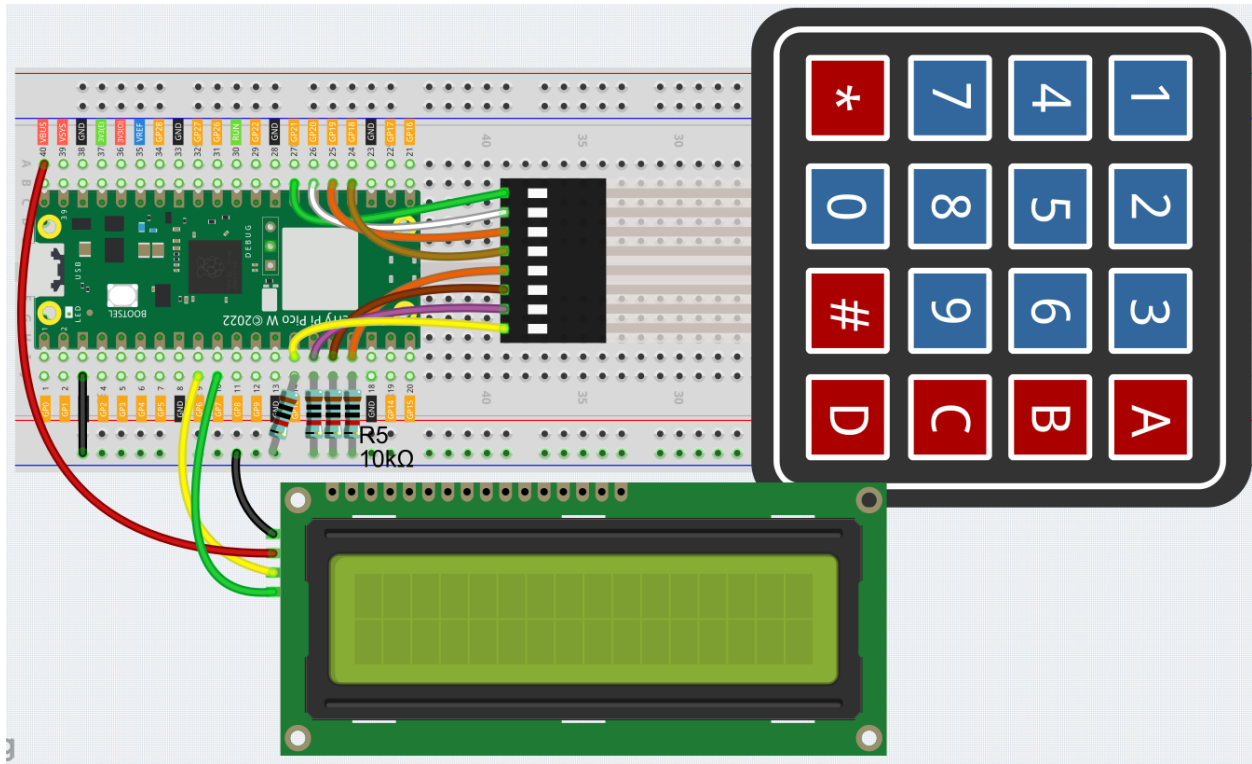
SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	4(10K)	
6	<i>4x4 Tastenfeld</i>	1	
7	<i>I2C LCD1602</i>	1	

## Schaltplan



Diese Schaltung basiert auf *4.2 4x4 Tastenfeld* und wird durch ein I2C LCD1602 ergänzt, um die gedrückten Tasten anzuzeigen.

## Verkabelung



Um die Verkabelung zu erleichtern, sind in der obigen Darstellung die Spalten und Reihen der Matrix-Tastatur sowie die 10K-Widerstände gleichzeitig in die Löcher eingefügt, wo G10 ~ G13 liegen.

### Code

#### Bemerkung:

- Öffnen Sie die Datei `7.7_game_guess_number.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5, um es auszuführen.
- Vergessen Sie nicht, im unteren rechten Eck den Interpreter „MicroPython (Raspberry Pi Pico)“ auszuwählen.
- Für detaillierte Anleitungen verweisen wir auf [Code direkt öffnen und ausführen](#).

```
from lcd1602 import LCD
import machine
import time
import urandom

# keypad function
characters = [["1", "2", "3", "A"], ["4", "5", "6", "B"], ["7", "8", "9", "C"], ["*", "0", "#", "D"]]

pin = [21, 20, 19, 18]
row = []
for i in range(4):
    row.append(None)
    row[i] = machine.Pin(pin[i], machine.Pin.OUT)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

pin = [13,12,11,10]
col = []
for i in range(4):
    col.append(None)
    col[i] = machine.Pin(pin[i], machine.Pin.IN)

def readKey():
    key = []
    for i in range(4):
        row[i].high()
        for j in range(4):
            if(col[j].value() == 1):
                key.append(characters[i][j])
        row[i].low()
    if key == [] :
        return None
    else:
        return key

# init/reset number
# reset the result as False for lcd show
def init_new_value():
    global pointValue,upper,count,lower
    pointValue = int(urandom.uniform(0, 99))
    print(pointValue)
    upper = 99
    lower = 0
    count = 0
    return False

# lcd show message
# If target, show game over.
# If not target, or not detected, show guess number.
def lcd_show(result):
    lcd.clear()
    if result == True:
        string = "GAME OVER!\n"
        string += "Point is " + str(pointValue)
    else :
        string = "Enter number: " + str(count) + "\n"
        string += str(lower)+ " < Point < " + str(upper)
    lcd.message(string)
    return

# detect number & reflesh show message
# if not target, reflesh number (upper or lower) and return False
# if target, return True
def number_processing():
    global upper,count,lower
    if count > pointValue:
        if count < upper:

```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

        upper = count
    elif count < pointValue:
        if count > lower:
            lower = count
    elif count == pointValue:
        return True
    count = 0
    return False

## start
lcd = LCD()
string = "Welcome!\n"
string = "Press A to Start!"
lcd.message(string)
result=init_new_value()

# read key & display
last_key = None
while True:
    current_key = readKey()
    if current_key == last_key:
        continue
    last_key = current_key
    if current_key != None:
        # print(current_key)
        if current_key ==["A"]: # reset number
            result=init_new_value()
        elif current_key=="D": # check
            result=number_processing()
        elif current_key[0] in list(["1","2","3","4","5","6","7","8","9","0"]) and count
↪ < 10: #check validity & limit digits
            count = count * 10 + int(current_key[0])
        lcd_show(result) # show
    time.sleep(0.1)

```

- Nach dem Ausführen des Codes drücken Sie A, um das Spiel zu starten. Eine zufällige Zahl point wird erzeugt, jedoch nicht auf dem LCD angezeigt. Ihre Aufgabe ist es, diese zu erraten.
- Die eingegebene Zahl wird am Ende der ersten Zeile angezeigt, bis die endgültige Berechnung abgeschlossen ist. (Drücken Sie D, um den Vergleich zu starten.)
- Der Zahlenbereich von point wird in der zweiten Zeile angezeigt. Ihre eingegebene Zahl muss in diesem Bereich liegen.
- Wenn Sie eine Zahl eingeben, verengt sich der Bereich; wenn Sie die Glückszahl zufällig oder unglücklicherweise erraten, erscheint GAME OVER!.

**Bemerkung:** Falls der Code und die Verkabelung in Ordnung sind, das LCD jedoch keinen Inhalt anzeigt, können Sie das Potentiometer auf der Rückseite drehen, um den Kontrast zu erhöhen.

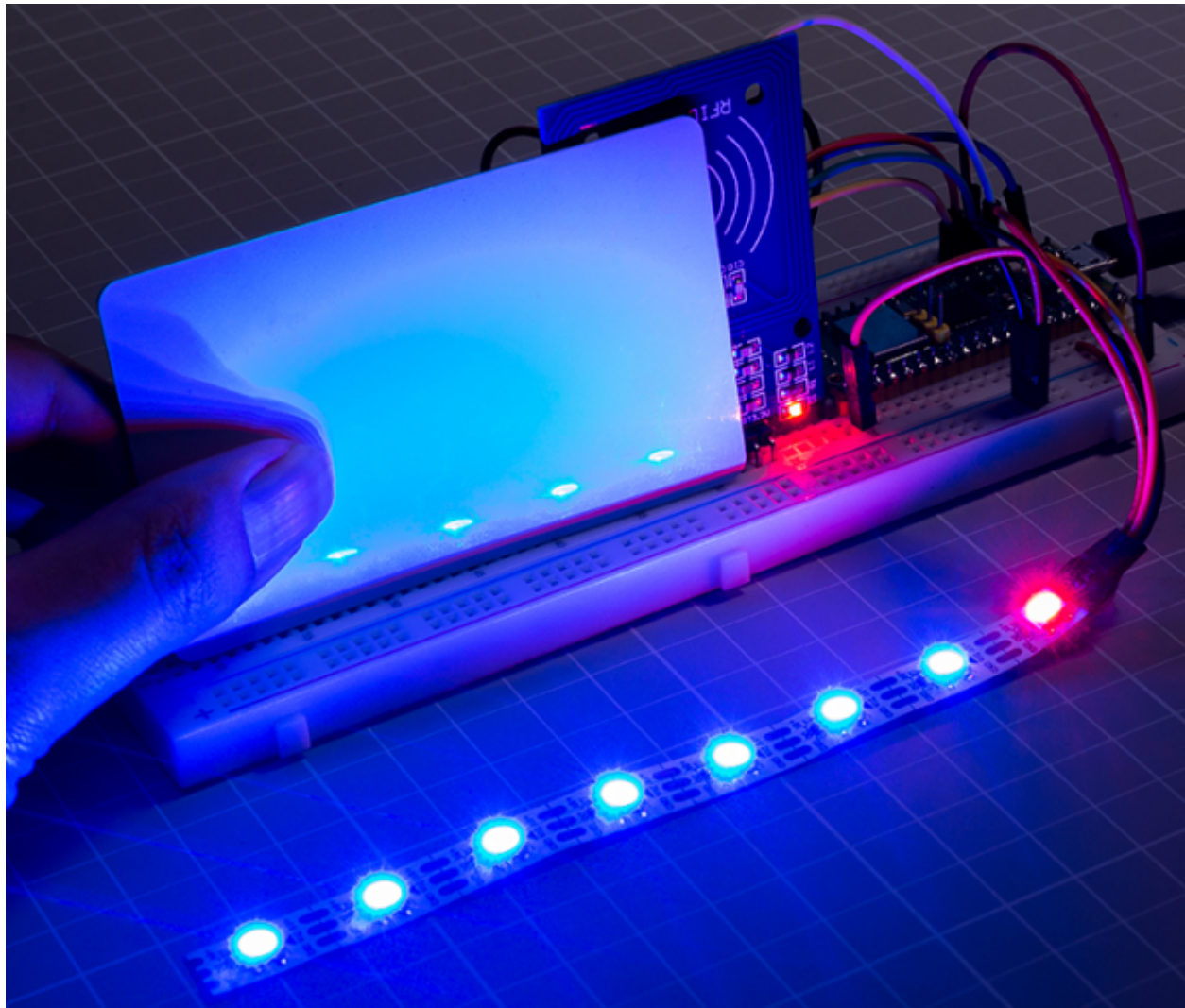


## 4.49 7.8 RFID-Musikplayer

In unserem vorherigen Projekt, *6.5 Funkfrequenz-Identifikation*, haben wir gelernt, dass das MFRC522-Modul uns erlaubt, bis zu 48 Zeichen Informationen auf die Karte (oder den Schlüssel) zu schreiben. Dies schließt sowohl den Schlüssel als auch die Identitätsinformationen ein, sowie die Noten für die Musik.

Zum Beispiel, wenn Sie EEFGGFEDCCDEEDD EEFGGFEDCCDEDDCC schreiben, wird der Summer die Melodie spielen, wenn die Karte (oder der Schlüssel) erneut gelesen wird. Es kann auch mit einem WS2812 ausgestattet werden, um beeindruckende Effekte darzustellen.

Sie können weitere Notenblätter im Internet finden oder sogar Ihre eigene Musik schreiben, diese auf die Karte (oder den Schlüssel) übertragen und sie mit Ihren Freunden teilen!



### Benötigte Bauteile

Für dieses Projekt benötigen wir die folgenden Komponenten.

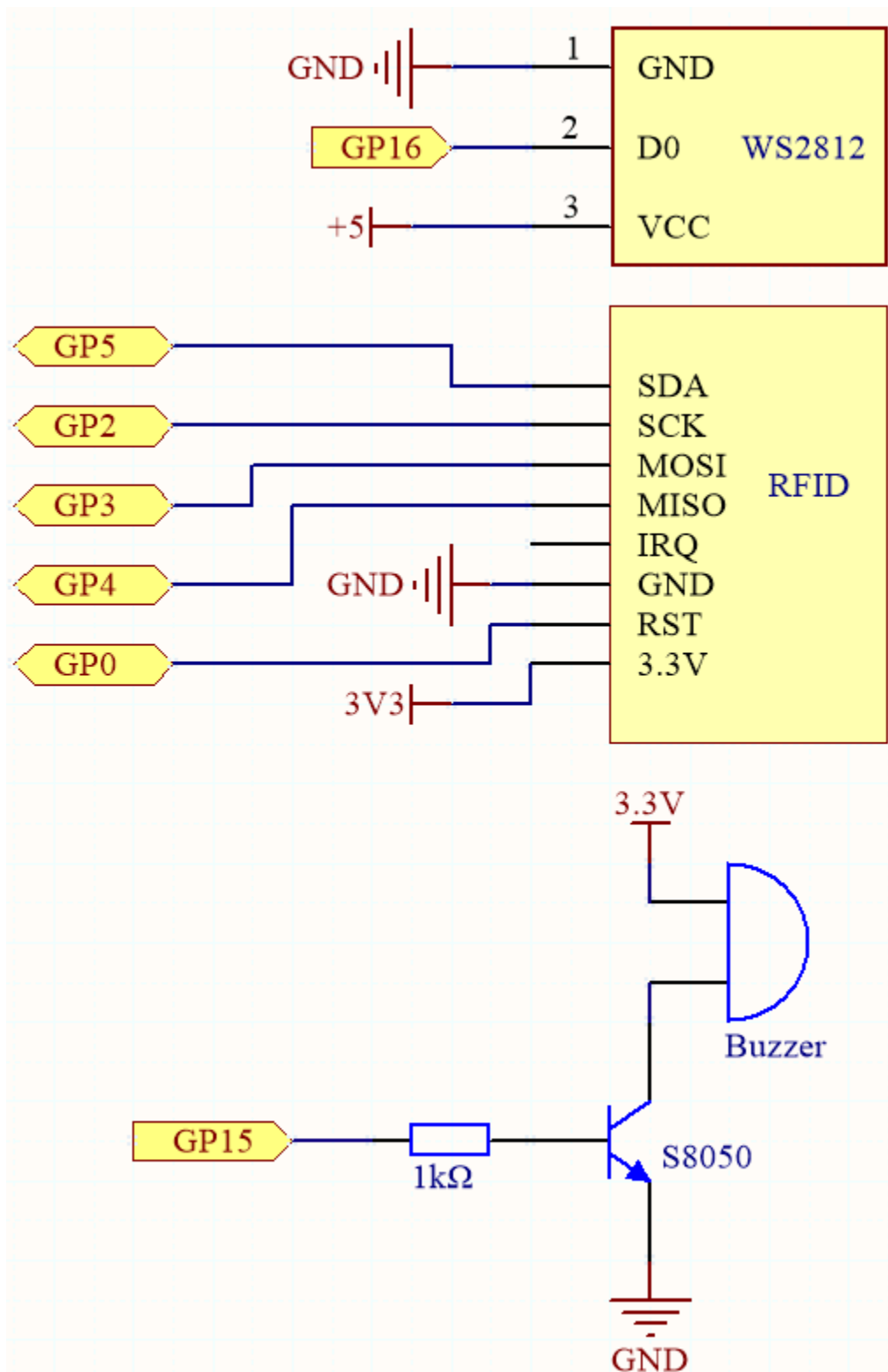
Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Bezeichnung	KOMPONENTEN IN DIESEM KIT	LINK
Kepler-Kit	450+	

Sie können die Bauteile auch einzeln über die unten stehenden Links erwerben.

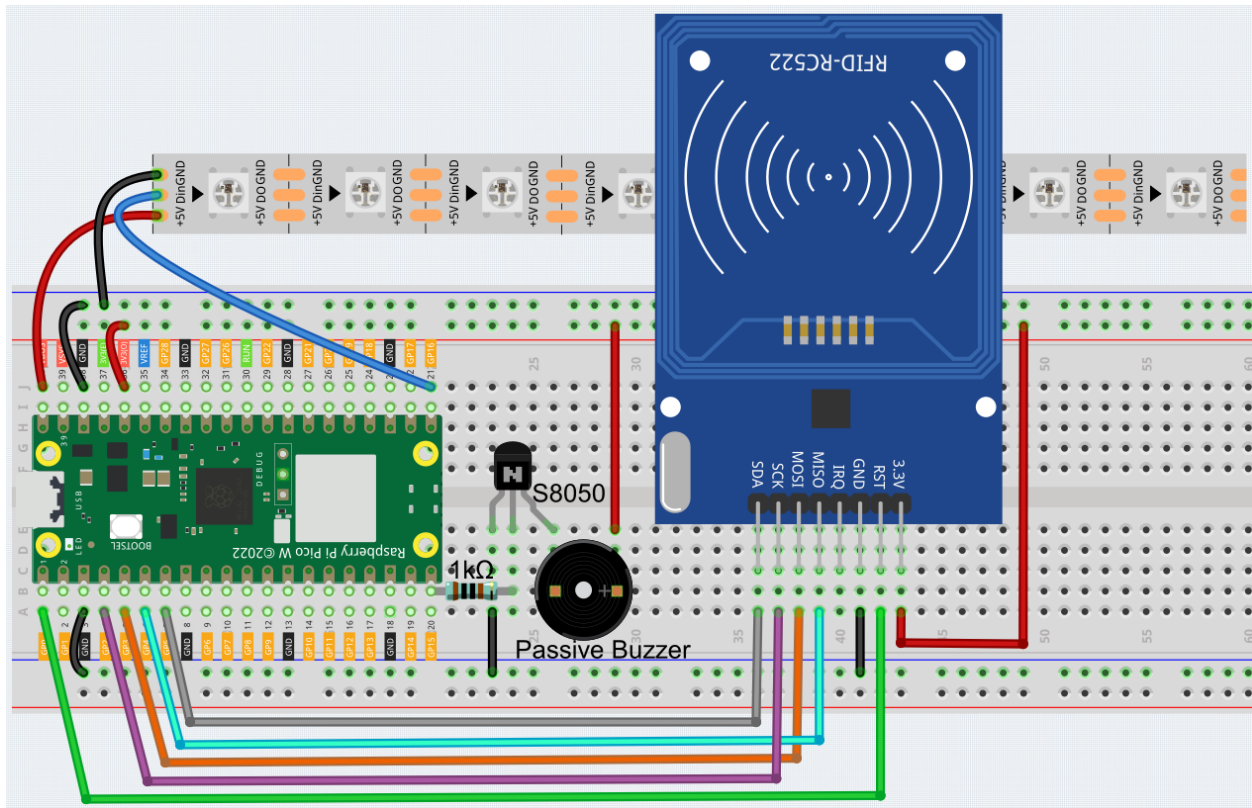
SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1(S8050)	
6	<i>Widerstand</i>	1(1K)	
7	Passiver Summer	1	
8	<i>MFRC522 Modul</i>	1	
9	<i>WS2812 RGB 8-LED-Streifen</i>	1	

## Schaltplan



Verdrahtung





### Code

1. Öffnen Sie die Datei 6.5\_rfid\_write.py im Verzeichnis kepler-kit-main/micropython und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
2. Nach dem Ausführen geben Sie EEFGGFEDCCDEEDD EEFGGFEDCCDEDCC im Shell ein und halten Sie dann die Karte (oder den Schlüssel) nahe am MFRC522-Modul. Auf diese Weise wird die Partitur der Ode an die Freude gespeichert.
3. Öffnen Sie die Datei 7.8\_rfid\_music\_player.py im Verzeichnis kepler-kit-main/micropython oder kopieren Sie diesen Code in Thonny, und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.

```
from mfrc522 import SimpleMFRC522
import machine
import time
from ws2812 import WS2812
import urandom

# ws2812
ws = WS2812(machine.Pin(16),8)

# mfrc522
reader = SimpleMFRC522(spi_id=0,sck=2,miso=4,mosi=3,cs=5,rst=0)

# buzzer
NOTE_C4 = 262
NOTE_D4 = 294
NOTE_E4 = 330
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

NOTE_F4 = 349
NOTE_G4 = 392
NOTE_A4 = 440
NOTE_B4 = 494
NOTE_C5 = 523

buzzer = machine.PWM(machine.Pin(15))
note=[NOTE_C4,NOTE_D4,NOTE_E4,NOTE_F4,NOTE_G4,NOTE_A4,NOTE_B4,NOTE_C5]

def tone(pin,frequency,duration):
    pin.freq(frequency)
    pin.duty_u16(30000)
    time.sleep_ms(duration)
    pin.duty_u16(0)

# lightup
def lumi(index):
    for i in range(8):
        ws[i] = 0x0000FF
    ws[index] = 0xFF0000 # int(urandom.uniform(0, 0xFFFFFF))
    ws.write()

# encode text to index
words=["C","D","E","F","G","A","B","N"]
def take_text(text):
    string=text.replace(' ','').upper()
    while len(string)>0:
        index=words.index(string[0])
        tone(buzzer,note[index],250)
        lumi(index)
        new_str=""
        for i in range(0, len(string)):
            if i != 0:
                new_str = new_str + string[i]
        string=new_str

# read card
def read():
    print("Reading...Please place the card...")
    id, text = reader.read()
    print("ID: %s\nText: %s" % (id,text))
    take_text(text)

read()

```

4. Wenn Sie die Karte (oder den Schlüssel) erneut nahe am MFRC522-Modul platzieren, wird der Summer die auf der Karte (oder dem Schlüssel) gespeicherte Musik abspielen und der RGB-Streifen wird in einer zufälligen Farbe leuchten.

## 4.50 7.9 Frucht-Klavier

Elektrische Leitfähigkeit findet man in vielen Metallgegenständen, ebenso wie im menschlichen Körper und in Früchten. Diese Eigenschaft kann genutzt werden, um ein amüsantes kleines Projekt zu schaffen: ein Frucht-Klavier. Mit anderen Worten, wir verwandeln Früchte in Tastaturen, die Musik abspielen können, einfach indem man sie berührt.



### Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

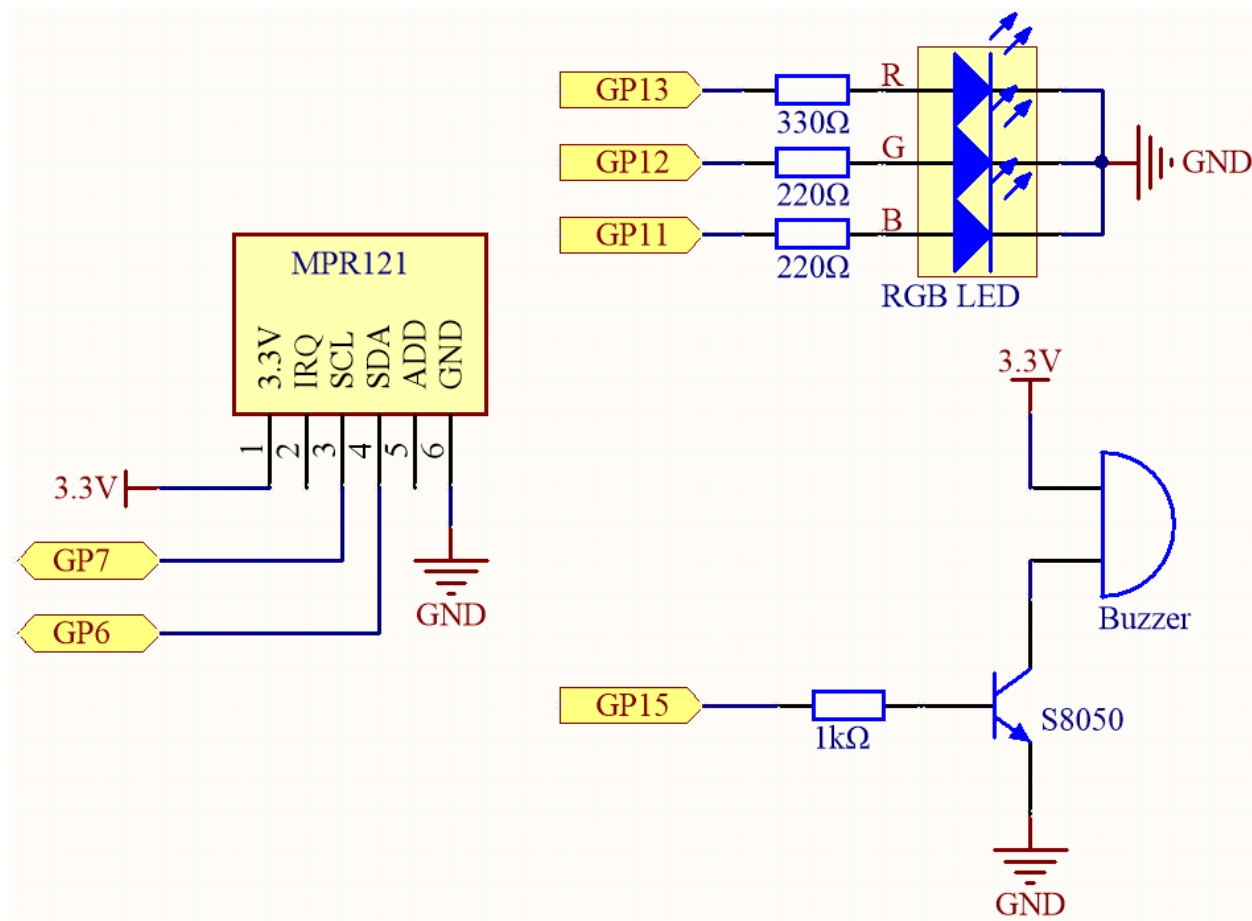
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	LINK
Kepler Kit	450+	

Sie können diese auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1(S8050)	
6	<i>Widerstand</i>	4(1-1K, 1-330, 2-220)	
7	Passiver Summer	1	
8	<i>RGB-LED</i>	1	
9	<i>MPR121 Modul</i>	1	

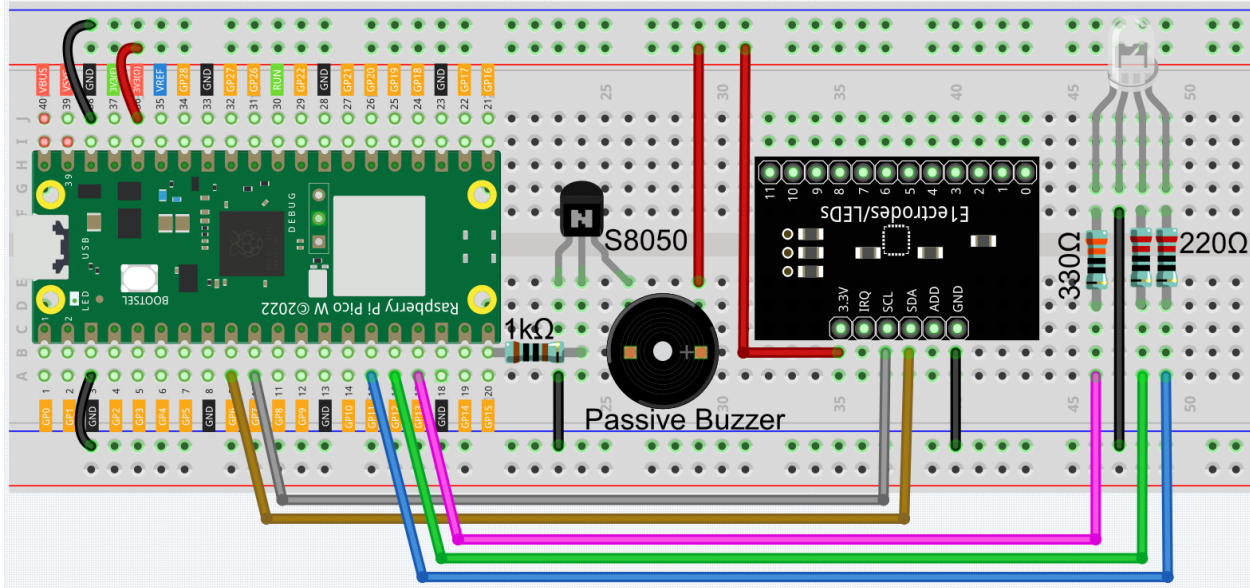
## Schaltplan



Um die Früchte in Klaviertasten zu verwandeln, müssen Sie noch die Elektroden am MPR121 mit der Frucht verbinden (z.B. in den Bananenstiel stecken).

Am Anfang wird der MPR121 initialisiert und jeder Elektrode wird ein Wert basierend auf der aktuellen Ladung zugewiesen. Wenn ein Leiter (zum Beispiel der menschliche Körper) eine Elektrode berührt, verschiebt und balanciert sich die Ladung. Das führt dazu, dass der Wert der Elektrode vom Ausgangswert abweicht und dem Hauptsteuerungsboard mitteilt, dass sie berührt wurde. Während dieses Vorgangs ist sicherzustellen, dass die Verkabelung jeder Elektrode stabil ist, damit ihre Ladung bei der Initialisierung ausgeglichen ist.

### Verkabelung



### Code

#### Bemerkung:

- Öffnen Sie die Datei `7.9_fruit_piano.py` im Pfad `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny, und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, den „MicroPython (Raspberry Pi Pico)“-Interpreter in der unteren rechten Ecke auszuwählen.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).
- Hier müssen Sie die Bibliothek `mpr121.py` verwenden. Prüfen Sie, ob sie auf dem Pico W hochgeladen wurde. Eine detaillierte Anleitung finden Sie unter [1.4 Bibliotheken auf den Pico hochladen](#).

```
from mpr121 import MPR121
from machine import Pin, I2C
import time
import urandom

# mpr121
i2c = I2C(1, sda=Pin(6), scl=Pin(7))
mpr = MPR121(i2c)

# buzzer
NOTE_A3 = 220
NOTE_B3 = 247
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

NOTE_C4 = 262
NOTE_D4 = 294
NOTE_E4 = 330
NOTE_F4 = 349
NOTE_G4 = 392
NOTE_A4 = 440
NOTE_B4 = 494
NOTE_C5 = 523
NOTE_D5 = 587
NOTE_E5 = 659

buzzer = machine.PWM(machine.Pin(15))
note = [NOTE_A3, NOTE_B3, NOTE_C4, NOTE_D4, NOTE_E4, NOTE_F4, NOTE_G4, NOTE_A4, NOTE_B4, NOTE_C5,
↪ NOTE_D5, NOTE_E5]

def tone(pin, frequency):
    pin.freq(frequency)
    pin.duty_u16(30000)

def noTone(pin):
    pin.duty_u16(0)

# rgb led
red = machine.PWM(machine.Pin(13))
green = machine.PWM(machine.Pin(12))
blue = machine.PWM(machine.Pin(11))
red.freq(1000)
green.freq(1000)
blue.freq(1000)

def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def lightup():
    red.duty_u16(int(urandom.uniform(0, 65535)))
    green.duty_u16(int(urandom.uniform(0, 65535)))
    blue.duty_u16(int(urandom.uniform(0, 65535)))

def dark():
    red.duty_u16(0)
    green.duty_u16(0)
    blue.duty_u16(0)

# main project
lastState=mpr.get_all_states()
touchMills=time.ticks_ms()
beat=500

while True:

```

(Fortsetzung auf der nächsten Seite)



(Fortsetzung der vorherigen Seite)

```

currentState=mpr.get_all_states()
if currentState != lastState:
    for i in range(12):
        if i in list(currentState) and not i in list(lastState):
            tone(buzzer,note[i])
            lightup()
            touchMills=time.ticks_ms()
if time.ticks_diff(time.ticks_ms(),touchMills)>=beat or len(currentState) == 0:
    noTone(buzzer)
    dark()
lastState = currentState

```

Berühren Sie die Früchte nicht, bevor das Programm ausgeführt wird, um eine fehlerhafte Referenz bei der Initialisierung zu vermeiden. Nachdem das Programm ausgeführt wurde, berühren Sie die Früchte sanft. Der Summer gibt den entsprechenden Ton ab und das RGB-Licht blinkt einmal zufällig auf.

## 4.51 7.10 Einparkhilfe

Dieses Projekt nutzt eine LED, einen Summer und ein Ultraschallmodul, um ein Einparkassistentensystem zu realisieren. Es lässt sich auf ein ferngesteuertes Auto setzen, um den realen Vorgang des Einparkens in eine Garage zu simulieren.

### Benötigte Komponenten

Für dieses Projekt werden die folgenden Komponenten benötigt.

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

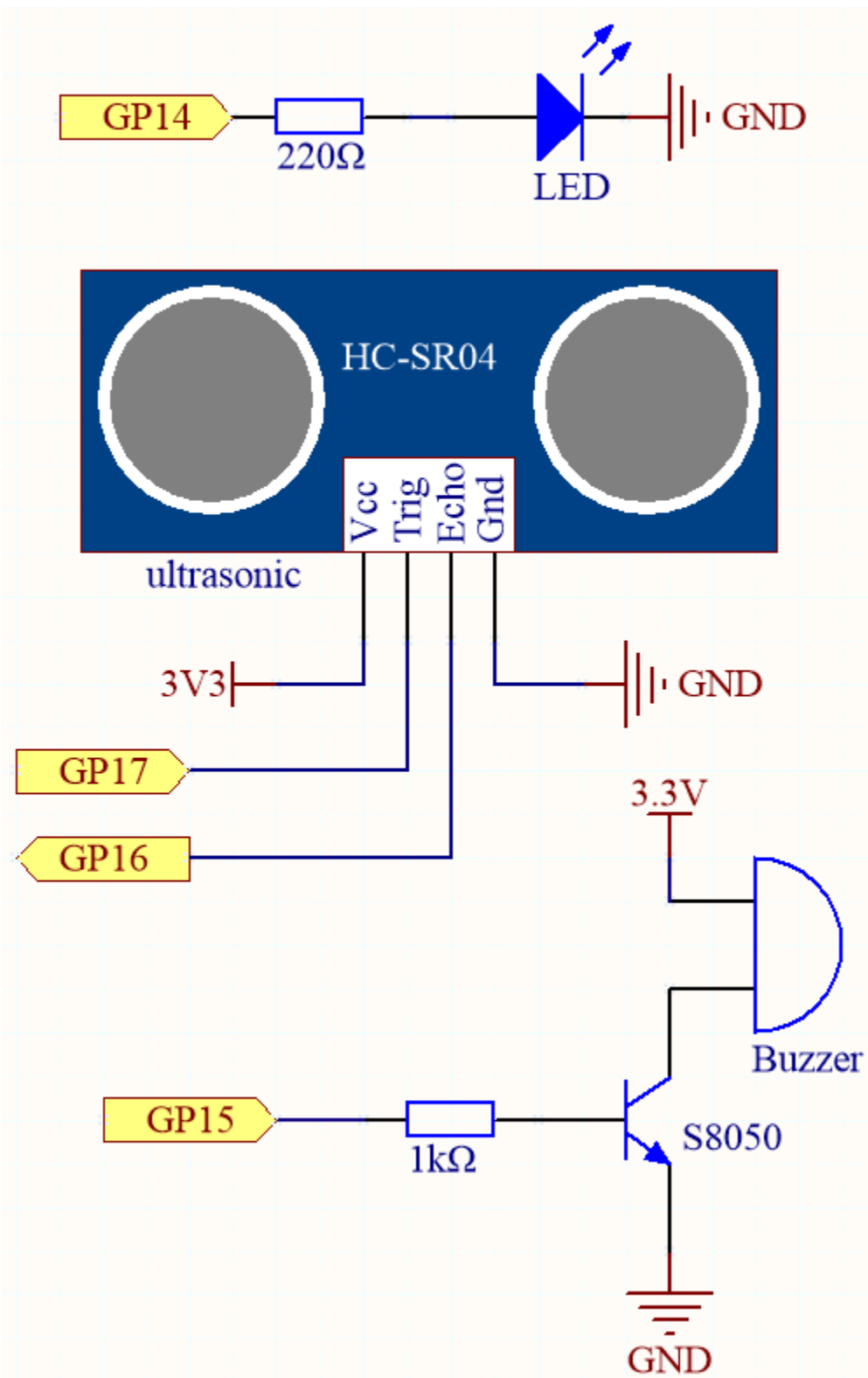
Bezeichnung	IN DIESEM SET ENTHALTENE ARTIKEL	LINK
Kepler-Set	450+	

Sie können die Teile auch einzeln über die untenstehenden Links erwerben.

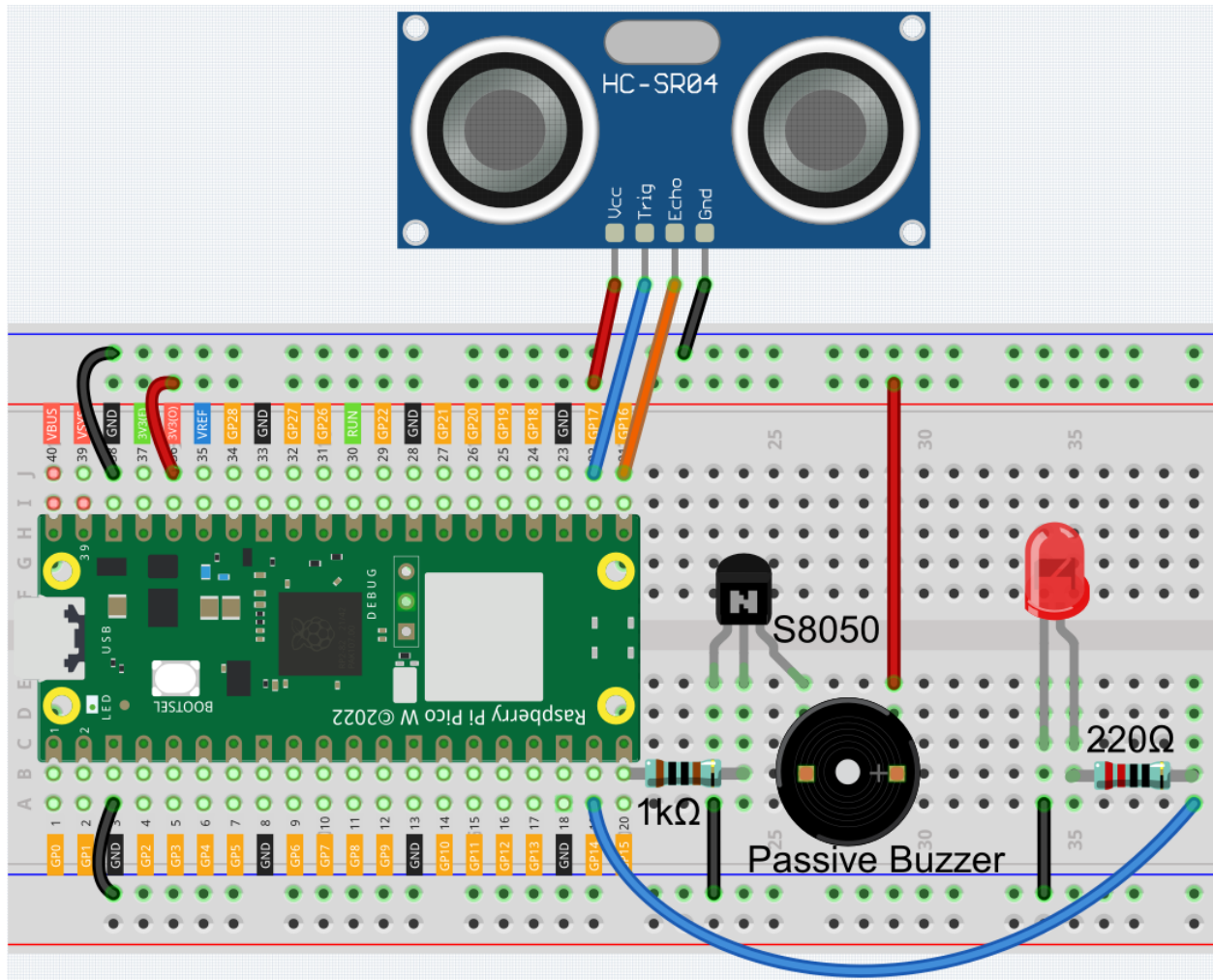
SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1 (S8050)	
6	<i>Widerstand</i>	2 (1K, 220)	
7	Passiver <i>Summer</i>	1	
8	<i>LED</i>	1	
9	<i>Ultraschallmodul</i>	1	

**Schaltplan**





Verkabelung



## Code

### Bemerkung:

- Öffnen Sie die Datei `7.10_reversing_aid.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergessen Sie nicht, im unteren rechten Eck den Interpreter „MicroPython (Raspberry Pi Pico)“ auszuwählen.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).

```
import machine
import time
import _thread

buzzer = machine.Pin(15, machine.Pin.OUT)
led = machine.Pin(14, machine.Pin.OUT)

TRIG = machine.Pin(17, machine.Pin.OUT)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

ECHO = machine.Pin(16,machine.Pin.IN)

dis = 100

def distance():
    timeout=10000*5/340
    TRIG.low()
    time.sleep_us(2)
    TRIG.high()
    time.sleep_us(10)
    TRIG.low()
    timeout_start = time.ticks_ms() # For timeout, re-read distance
    while not ECHO.value():
        waiting_time = time.ticks_ms()
        if waiting_time - timeout_start > timeout:
            return -1
    time1 = time.ticks_us()
    while ECHO.value():
        waiting_time = time.ticks_ms()
        if waiting_time - timeout_start > timeout:
            return -1
    time2 = time.ticks_us()
    during = time.ticks_diff(time2 ,time1)
    return during * 340 / 2 / 10000

def ultrasonic_thread():
    global dis
    while True:
        dis = distance()

_thread.start_new_thread(ultrasonic_thread, ())

def beep():
    buzzer.value(1)
    led.value(1)
    time.sleep(0.1)
    buzzer.value(0)
    led.value(0)
    time.sleep(0.1)

intervals = 100000000
previousMills=time.ticks_ms()
time.sleep(1)

while True:
    if dis<0:
        pass
    elif dis <= 10:
        intervals = 300
    elif dis <= 20:
        intervals =500
    elif dis <=50:

```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

        intervals =1000
    else:
        intervals = 2000
    if dis!=-1:
        print ('Distance: %.2f' % dis)
        time.sleep_ms(100)

    currentMills=time.ticks_ms()

    if time.ticks_diff(currentMills,previousMills)>=intervals:
        beep()
        previousMills=currentMills

```

- Sobald das Programm läuft, wird der Ultraschallsensor kontinuierlich die Entfernung zum vor Ihnen befindlichen Hindernis messen, und Sie können den genauen Entfernungswert in der Shell sehen.
- Je nach Entfernungswert ändern die LED und der Summer die Frequenz ihres Blinkens und Piepsens und signalisieren so die Annäherung an das Hindernis.
- Im Artikel [6.1 Abstandsmessung](#) wurde erwähnt, dass das Programm pausiert, während der Ultraschallsensor arbeitet.
- Um die Timing von LED und Summer nicht zu beeinträchtigen, haben wir in diesem Beispiel einen separaten Thread für die Entfernungsmessung erstellt.

## 4.52 7.11 Somatosensorische Steuerung

Wenn Sie viele Robotik-Filme gesehen haben, ist Ihnen diese Vorstellung wahrscheinlich nicht fremd. Der Protagonist bewegt sein Handgelenk und der riesige Roboter folgt der Bewegung; der Protagonist ballt die Faust, und auch der Roboter tut dies. Es sieht einfach cool aus.

Diese Technologie findet bereits breite Anwendung in Universitäten und Forschungsinstituten. Die Einführung von 5G wird ihre Einsatzmöglichkeiten erheblich erweitern. Ein typisches Beispiel hierfür ist die Fernbedienung des „Chirurgie-Roboters da Vinci“.

Ein Robotiksystem dieser Art besteht in der Regel aus zwei Modulen: einem Modul zur Erfassung menschlicher Bewegungen und einem Aktuator-Modul für den Roboterarm (in manchen Anwendungen gibt es auch ein Datenkommunikationsmodul).

Hier kommt der MPU6050 zum Einsatz, um menschliche Bewegungen zu erfassen (indem er an einem Handschuh befestigt wird), während ein Servomotor die Bewegung des Roboterarms simuliert.

### Benötigte Komponenten

Für dieses Projekt werden die folgenden Komponenten benötigt.

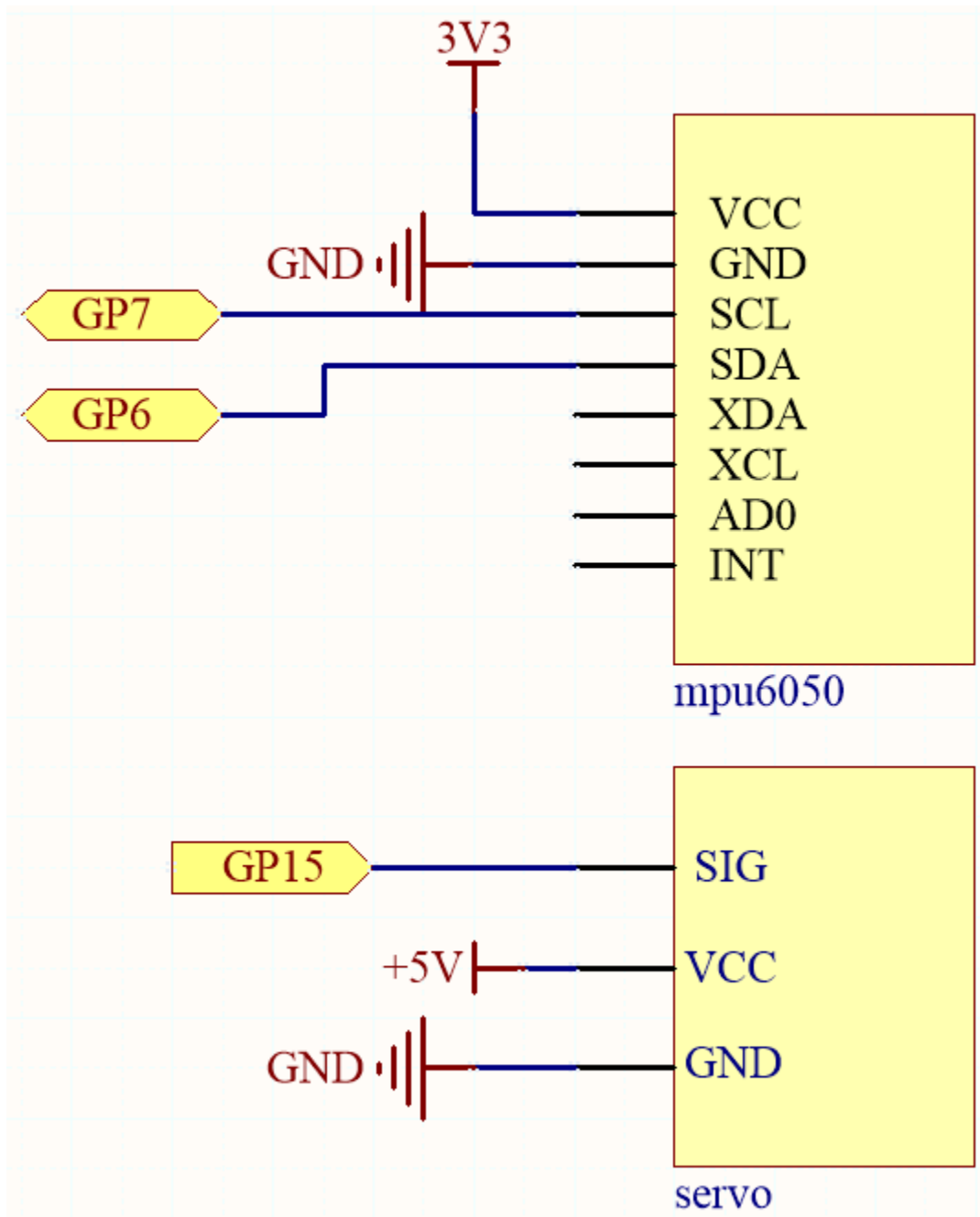
Ein Gesamtkit zu erwerben ist definitiv praktisch. Hier ist der Link:

Bezeichnung	ARTIKEL IN DIESEM KIT	LINK
Kepler-Set	450+	

Alternativ können die Komponenten auch einzeln über die unten stehenden Links gekauft werden.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>MPU6050 Modul</i>	1	
6	<i>Servo</i>	1	

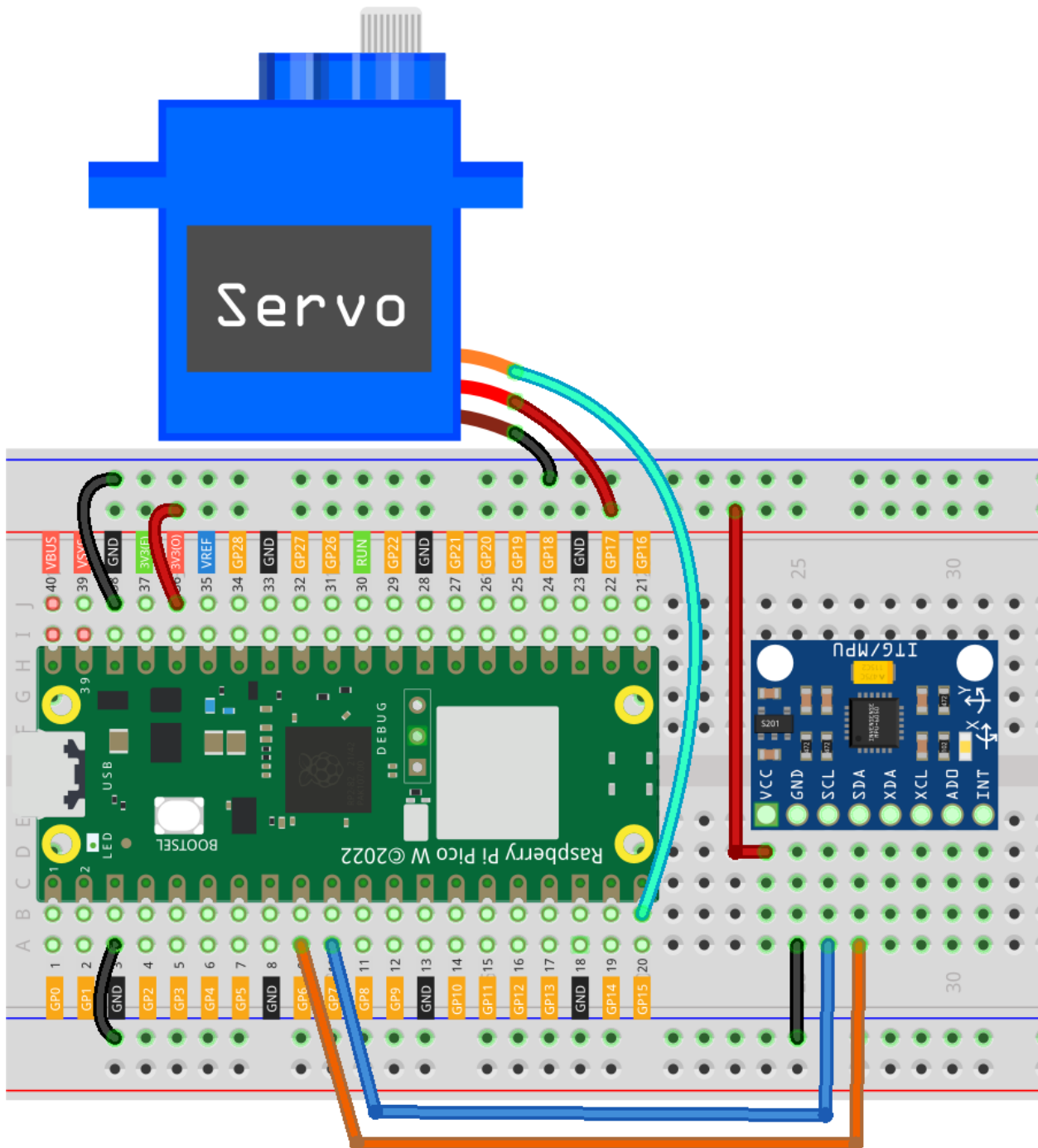
**Schaltplan**



Der MPU6050 berechnet den Neigungswinkel basierend auf den Beschleunigungswerten in jeder Richtung.

Das Programm steuert den Servomotor so, dass er den entsprechenden Auslenkwinkel gemäß dem sich ändernden Neigungswinkel ausführt.

#### Verdrahtung



## Code

### Bemerkung:

- Öffnen Sie die Datei `7.11_somatosensory_controller.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie den unten stehenden Code in Thonny. Klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergewissern Sie sich, dass der Interpreter „MicroPython (Raspberry Pi Pico)“ in der unteren rechten Ecke ausgewählt ist.

- Für detaillierte Anleitungen siehe *Code direkt öffnen und ausführen*.
  - Hier müssen Sie auch die Dateien `imu.py` und `vector3d.py` verwenden. Bitte überprüfen Sie, ob sie auf Pico W hochgeladen wurden. Detaillierte Anweisungen finden Sie unter *1.4 Bibliotheken auf den Pico hochladen*.
- 

```
from imu import MPU6050
from machine import I2C, Pin
import time
import math

# mpu6050
i2c = I2C(1, sda=Pin(6), scl=Pin(7), freq=400000)
mpu = MPU6050(i2c)

# servo
servo = machine.PWM(machine.Pin(15))
servo.freq(50)

def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

# get rotary angle
def dist(a,b):
    return math.sqrt((a*a)+(b*b))

def get_y_rotation(x,y,z):
    radians = math.atan2(x, dist(y,z))
    return -math.degrees(radians)

def get_x_rotation(x,y,z):
    radians = math.atan2(y, dist(x,z))
    return math.degrees(radians)

# servo work
def servo_write(pin,angle):
    pulse_width=interval_mapping(angle, 0, 180, 0.5,2.5)
    duty=int(interval_mapping(pulse_width, 0, 20, 0,65535))
    pin.duty_u16(duty)

times=25
while True:
    total=0
    for i in range(times):
        angle=get_y_rotation(mpu.accel.x, mpu.accel.y, mpu.accel.z) #get rotation value
        total+=angle
    average_angle=int(total/times) # make the value smooth
    servo_write(servo,interval_mapping(average_angle,-90,90,0,180))
```

Sobald das Programm läuft, wird der Servomotor sich nach links und rechts drehen, wenn Sie den MPU6050 neigen (oder Ihr Handgelenk bewegen, falls er an einem Handschuh montiert ist).



## 4.53 7.12 Digitaler Wasserwaage

Ein **Wasserwaage** ist ein Instrument, das dazu dient, zu zeigen, ob eine Fläche horizontal (waagrecht) oder vertikal (senkrecht) ist. Verschiedene Typen von Wasserwaagen werden von Zimmerleuten, Steinmetzen, Maurern und anderen Handwerkern im Baugewerbe, von Vermessungsingenieuren, Mühlenbauern und anderen Metallarbeitern sowie in einigen fotografischen und videografischen Arbeiten verwendet.

In diesem Projekt erstellen wir eine digitale Wasserwaage mit einem MPU6050 und einer 8x8 LED-Matrix. Wenn Sie den MPU6050 kippen, wird die Blase auf der LED-Matrix ebenfalls kippen.

### Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

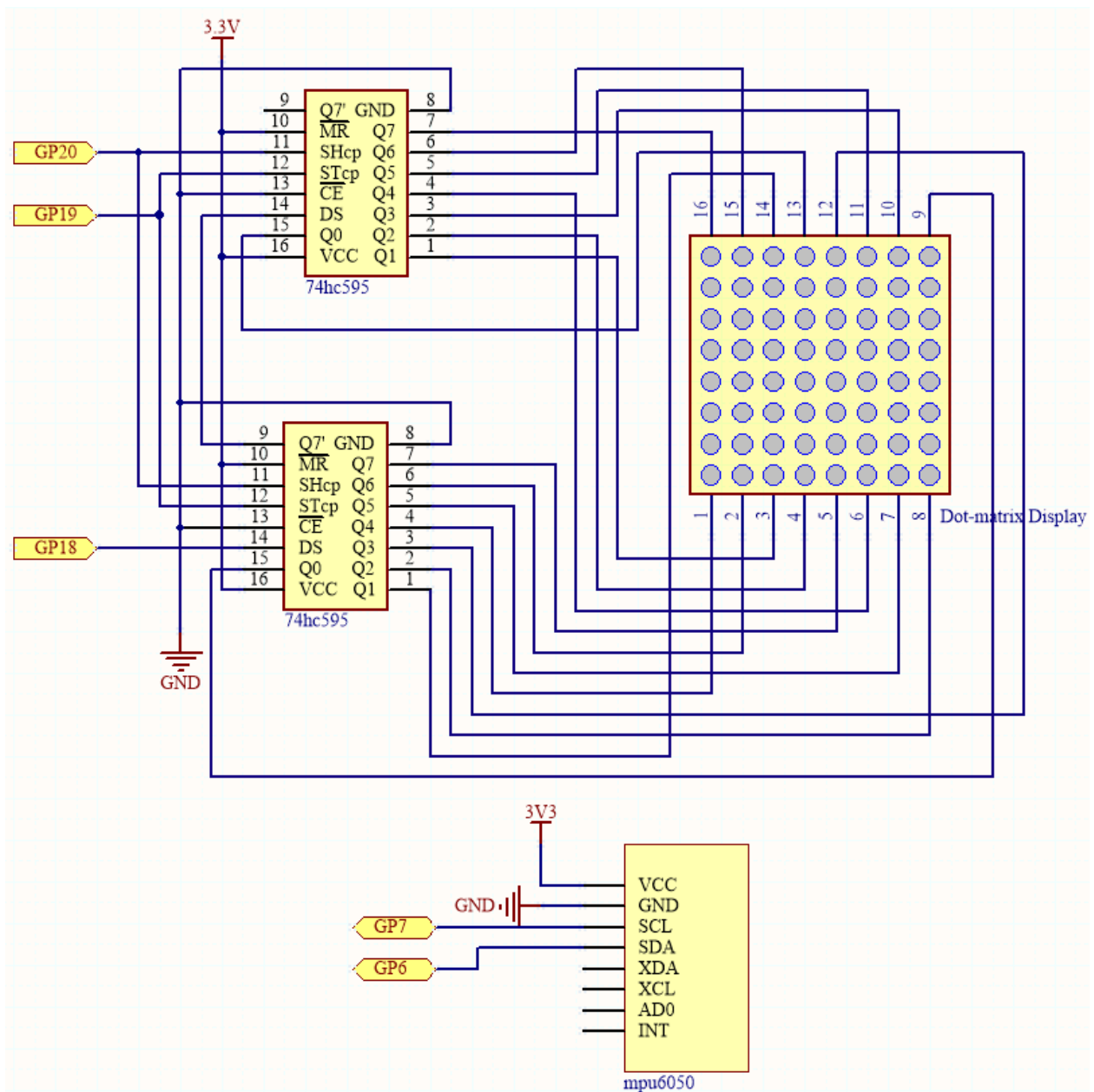
Es ist definitiv praktisch, ein gesamtes Kit zu kaufen. Hier ist der Link:

Bezeichnung	ARTIKEL IN DIESEM KIT	LINK
Kepler-Set	450+	

Die Komponenten können auch einzeln über die untenstehenden Links erworben werden.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>LED-Punktmatrix</i>	1	
6	<i>74HC595</i>	2	
7	<i>MPU6050 Modul</i>	1	

### Schaltplan

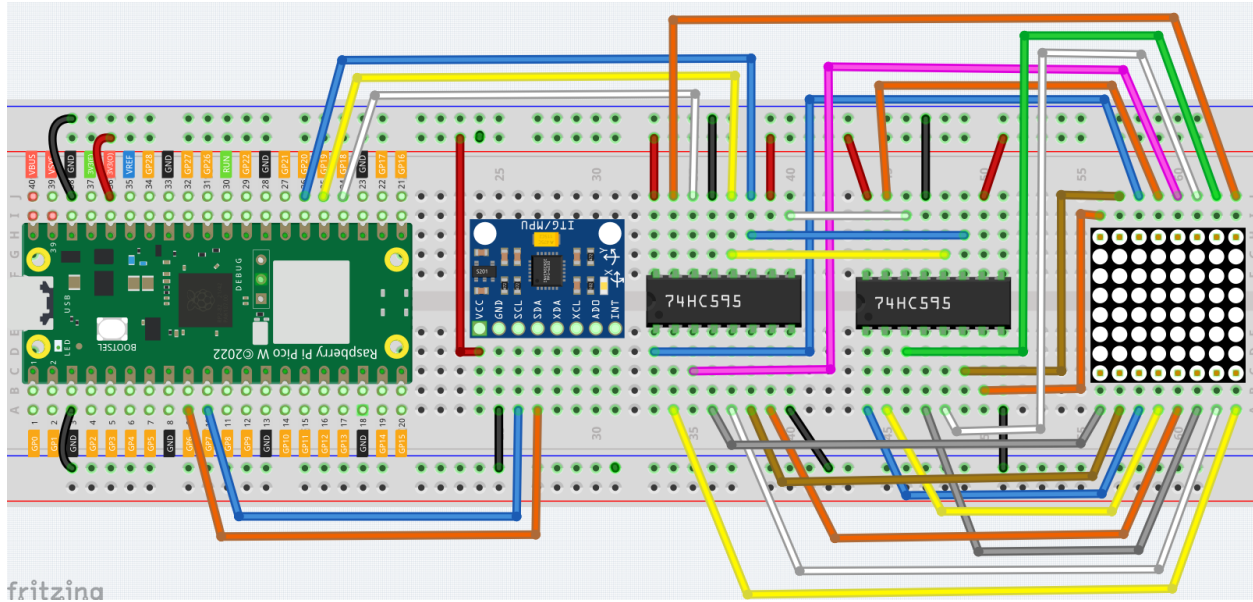


Der MPU6050 nimmt die Beschleunigungswerte in jeder Richtung auf und berechnet den Neigungswinkel.

Das Programm erzeugt dann auf Basis der Daten von den beiden 74HC595-Chips einen 2x2-Punkt auf der Punktmatrix.

Je nach Veränderung des Neigungswinkels sendet das Programm unterschiedliche Daten an die 74HC595-Chips, und die Position des Punkts ändert sich, was einen Blaseneffekt erzeugt.

## Verkabelung



### Code

#### Bemerkung:

- Öffnen Sie die Datei `7.12_digital_bubble_level.py` im Verzeichnis `kepler-kit-main/micropython` oder kopieren Sie diesen Code in Thonny und klicken Sie dann auf „Aktuelles Skript ausführen“ oder drücken Sie einfach F5.
- Vergewissern Sie sich, dass der Interpreter „MicroPython (Raspberry Pi Pico)“ in der unteren rechten Ecke ausgewählt ist.
- Für detaillierte Anleitungen siehe [Code direkt öffnen und ausführen](#).
- Hier müssen Sie auch die Dateien `imu.py` und `vector3d.py` verwenden. Bitte überprüfen Sie, ob sie auf dem Pico W hochgeladen wurden. Detaillierte Anweisungen finden Sie unter [1.4 Bibliotheken auf den Pico hochladen](#).

```
import machine
from machine import I2C, Pin
import time
import math
from imu import MPU6050

### mpu6050
i2c = I2C(1, sda=Pin(6), scl=Pin(7), freq=400000)
mpu = MPU6050(i2c)

# get rotary angle
def dist(a,b):
    return math.sqrt((a*a)+(b*b))

def get_y_rotation(x,y,z):
    radians = math.atan2(x, dist(y,z))
    return -math.degrees(radians)
```

(Fortsetzung auf der nächsten Seite)

```

def get_x_rotation(x,y,z):
    radians = math.atan2(y, dist(x,z))
    return math.degrees(radians)

def get_angle():
    y_angle=get_y_rotation(mpu.accel.x, mpu.accel.y, mpu.accel.z)
    x_angle=get_x_rotation(mpu.accel.x, mpu.accel.y, mpu.accel.z)
    return x_angle,y_angle

### led matrix display
sdi = machine.Pin(18,machine.Pin.OUT)
rclk = machine.Pin(19,machine.Pin.OUT)
srclk = machine.Pin(20,machine.Pin.OUT)

def hc595_in(dat):
    for bit in range(7,-1, -1):
        srclk.low()
        time.sleep_us(30)
        sdi.value(1 & (dat >> bit))
        time.sleep_us(30)
        srclk.high()

def hc595_out():
    rclk.high()
    time.sleep_us(200)
    rclk.low()

def display(glyph):
    for i in range(0,8):
        hc595_in(glyph[i])
        hc595_in(0x80>>i)
        hc595_out()

# data transformation
def matrix_2_glyph(matrix):
    glyph= [0 for i in range(8)] # glyph code for display()
    for i in range(8):
        for j in range(8):
            glyph[i]+=matrix[i][j]<<j
    return glyph

def clamp_number(val, min, max):
    return min if val < min else max if val > max else val

def interval_mapping(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

# Calculate the position of the bubble
sensitivity=4          # The higher the number, the more sensitive
matrix_range=7        # The size of the matrix is 8, so the coordinate range is 0~7
point_range=matrix_range-1    # The x, y value of the bubble's marker point (upper left,

```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

↪point) should be between 0-6
def bubble_position():
    x,y=get_angle()
    x=int(clamp_number(interval_mapping(x,-90,90,0-sensitivity,point_range+sensitivity),
↪0,point_range))
    y=int(clamp_number(interval_mapping(y,-90,90,point_range+sensitivity,0-sensitivity),
↪0,point_range))
    return [x,y]

# Drop the bubble into empty matrix
def drop_bubble(matrix,bubble):
    matrix[bubble[0]][bubble[1]]=0
    matrix[bubble[0]+1][bubble[1]]=0
    matrix[bubble[0]][bubble[1]+1]=0
    matrix[bubble[0]+1][bubble[1]+1]=0
    return matrix

while True:
    matrix= [[1 for i in range(8)] for j in range(8)] # empty matrix
    bubble=bubble_position() # bubble coordinate
    matrix=drop_bubble(matrix,bubble) # drop the bubble into empty matrix
    display(matrix_2_glyph(matrix)) # show matrix

```

Stellen Sie das Steckbrett auf eine ebene Fläche, nachdem Sie das Programm ausgeführt haben. Ein Punkt wird in der Mitte der LED-Matrix erscheinen (falls dies nicht der Fall ist, ist der MPU6050 möglicherweise nicht waagerecht). Wenn Sie das Steckbrett kippen, wird der Punkt in die Richtung wandern, in die Sie es gekippt haben.



Dieser Abschnitt führt Sie durch die Schritte, um das Pico W mit dem Netzwerk zu verbinden und spannende IoT-Projekte umzusetzen.

Bevor Sie jedoch in diesen Abschnitt einsteigen, sollten Sie unbedingt **1. Einstieg** im Abschnitt *Für MicroPython-Nutzer* abschließen. Dort erfahren Sie, wie Sie die Thonny-IDE installieren, die Micropython-Firmware für das Raspberry Pi Pico W aufspielen und die notwendigen Bibliotheken hochladen.

5.1 1. Zugang zum Netzwerk

Das Raspberry Pi Pico W ähnelt stark dem Raspberry Pi Pico und bietet die gleichen GPIOs, den gleichen Micro-USB-Anschluss und die gleiche Größe. Der einzige Unterschied ist der zusätzliche CYW43439 2,4-GHz-WLAN-Chip von Infineon. Jetzt schauen wir uns an, wie es sich mit unserem WLAN-Netzwerk verbindet.

Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

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

Name	ARTIKEL IN DIESEM SET	LINK
Kepler-Kit	450+	

Sie können diese auch separat über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	

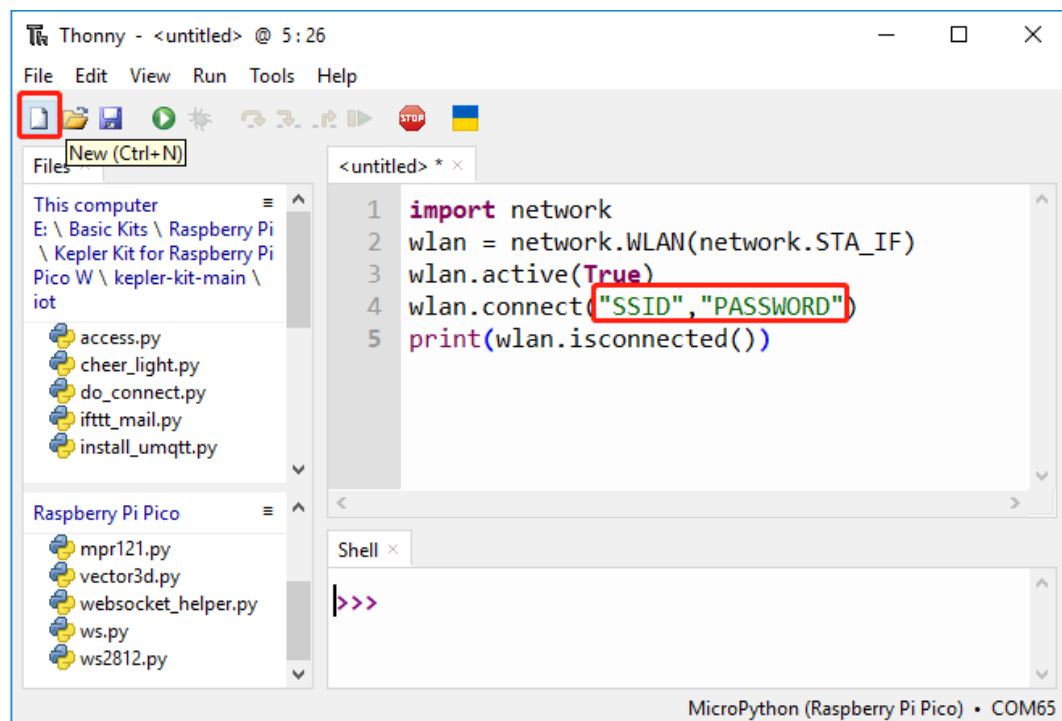
### 5.1.1 1. Internetverbindung herstellen

Mit nur fünf Zeilen MicroPython ist unser Raspberry Pi Pico W problemlos mit dem Internet verbunden.

Sie können den folgenden Code direkt in der Shell ausführen. Drücken Sie nach der Eingabe die Enter-Taste. Alternativ folgen Sie der untenstehenden Anleitung und legen eine neue .py-Datei an, um den Code auszuführen.

```
import network
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect("SSID", "PASSWORD")
print(wlan.isconnected())
```

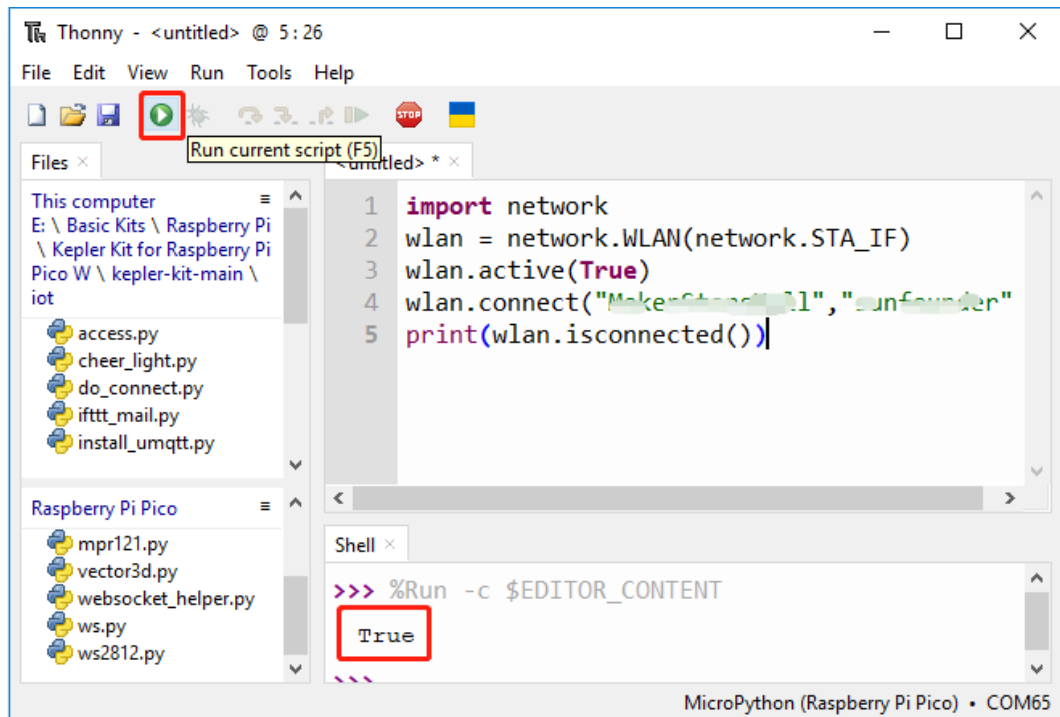
1. Erstellen Sie ein neues Skript, indem Sie in Thonny auf die Schaltfläche **Neu** klicken. Kopieren Sie dann den oben stehenden Code und ändern Sie SSID und PASSWORD entsprechend Ihren Angaben.



2. Um das Skript auszuführen, klicken Sie auf die Schaltfläche **Aktuelles Skript ausführen** oder drücken Sie F5. Bei erfolgreicher Verbindung wird true ausgegeben.

**Bemerkung:** Vergewissern Sie sich, dass das Raspberry Pi Pico W per USB-Kabel mit dem Computer verbunden ist. Wählen Sie dann unten rechts MicroPython (Raspberry Pi Pico).COMxxx als Interpreter.





### 5.1.2 2. Zeitüberschreitung und IP-Anzeige

Angehts potenziell schlechter Netzwerkbedingungen fügen wir dem Code eine Überprüfung für Zeitüberschreitungen hinzu.

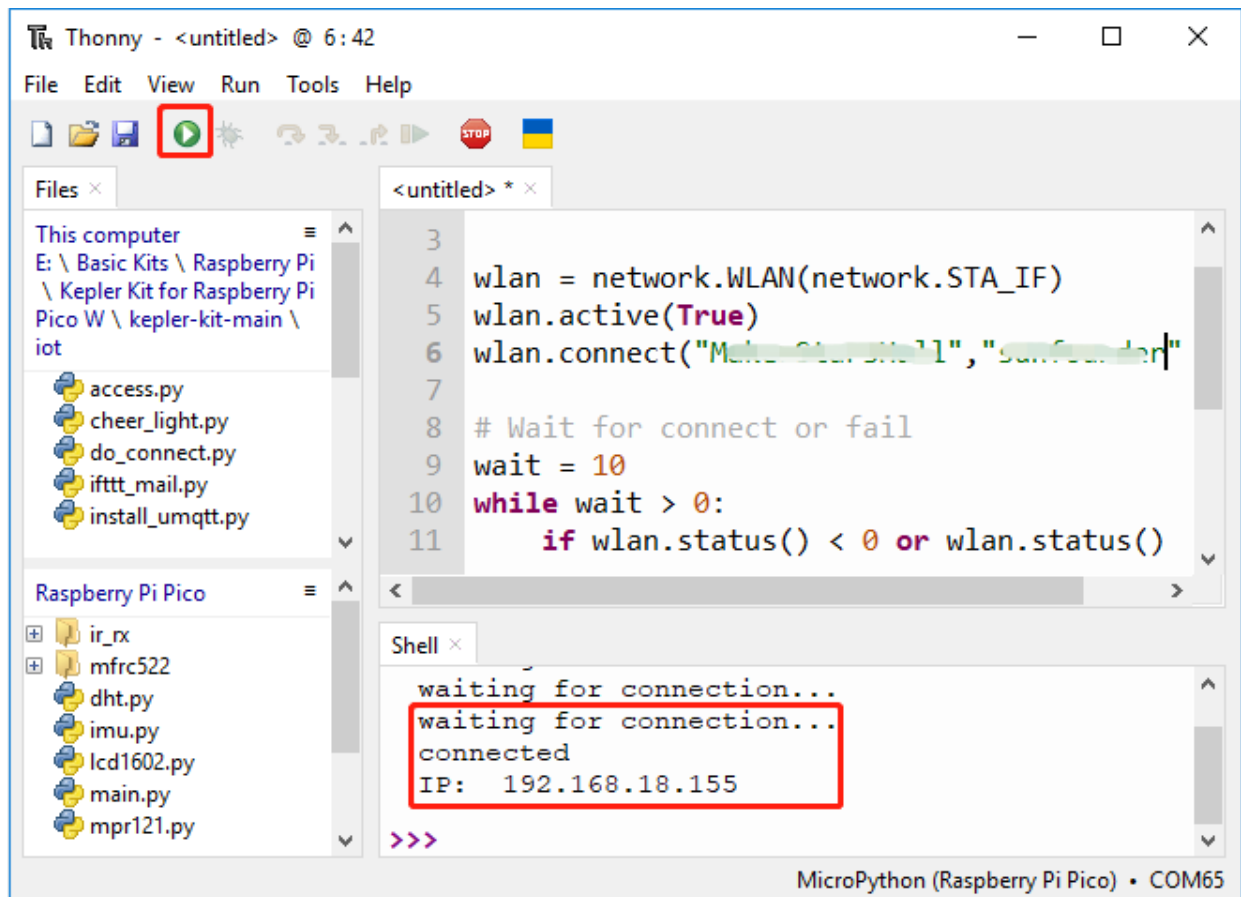
Ist die Verbindung erfolgreich, wird die IP-Adresse des Pico W nach dem Ausführen des Skripts angezeigt.

```
import network
import time

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect("SSID", "PASSWORD")

# Wait for connect or fail
wait = 10
while wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    wait -= 1
    print('waiting for connection...')
    time.sleep(1)

# Handle connection error
if wlan.status() != 3:
    raise RuntimeError('wifi connection failed')
else:
    print('connected')
    print('IP: ', wlan.ifconfig()[0])
```



- `wlan.status()` Funktion: Gibt den aktuellen Status der WLAN-Verbindung zurück. Die Rückgabewerte sind in der folgenden Tabelle aufgeführt.

Status	Wert	Beschreibung
STAT_IDLE	0	keine Verbindung und keine Aktivität,
STAT_CONNECTING	1	Verbindungsaufbau läuft,
STAT_WRONG_PASSWORD	-3	fehlgeschlagen wegen falschem Passwort,
STAT_NO_AP_FOUND	-2	fehlgeschlagen, weil kein Zugangspunkt antwortete,
STAT_CONNECT_FAIL	-1	fehlgeschlagen aus anderen Gründen,
STAT_GOT_IP	3	Verbindung erfolgreich.

- `wlan.ifconfig()` Funktion: Ermittelt IP-Adressen, Subnetzmasken, Gateways und DNS-Server. Bei direktem Aufruf wird ein 4-Tupel mit den genannten Informationen zurückgegeben. In diesem Fall zeigen wir nur die IP-Adresse an.
- [class WLAN – MicroPython Docs](#)

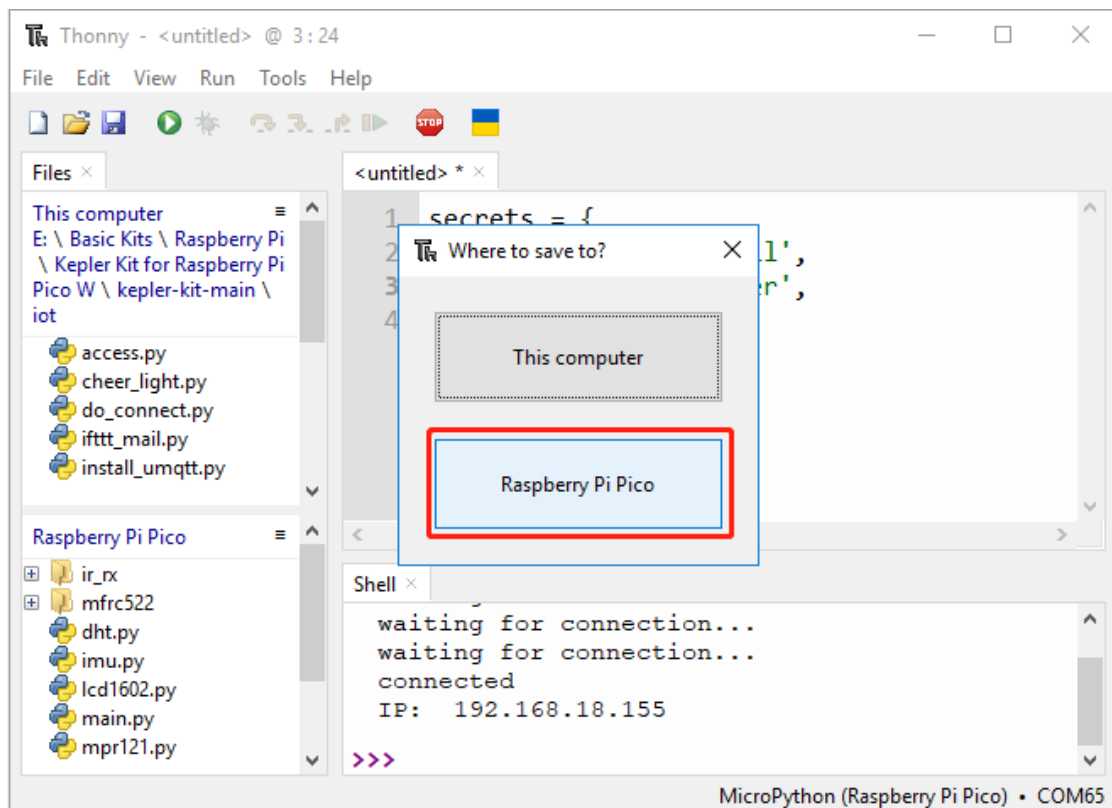
### 5.1.3 3. Speichern privater Informationen in secrets.py

Wenn Sie Ihr Pico W-Projekt teilen möchten, wollen Sie sicher nicht, dass andere Ihr WLAN-Passwort oder Ihren API-Schlüssel einsehen können. Aus Sicherheitsgründen legen wir daher eine `secrets.py`-Datei an, um diese sensiblen Informationen zu schützen.

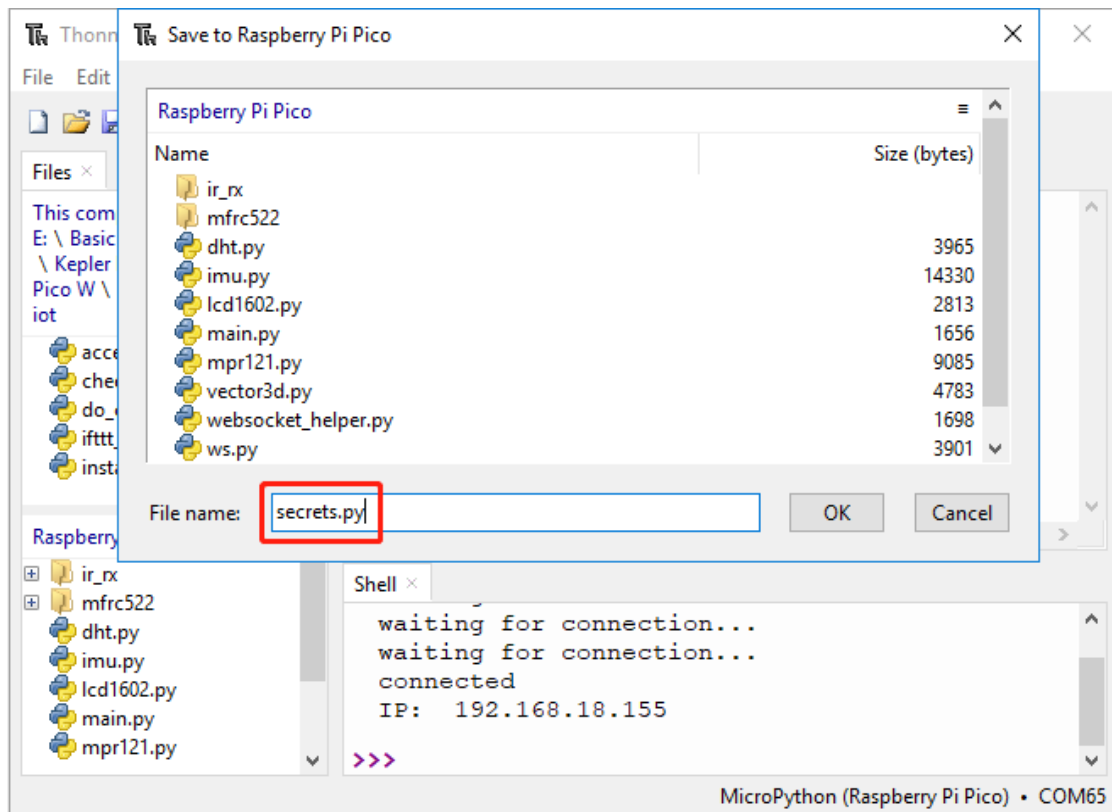
# Fügen Sie den folgenden Code in eine neue Skriptdatei in Thonny ein. Passen Sie SSID und PASSWORD Ihren eigenen Anmeldedaten an.

```
secrets = {
    'ssid': 'SSID',
    'password': 'PASSWORD',
}
```

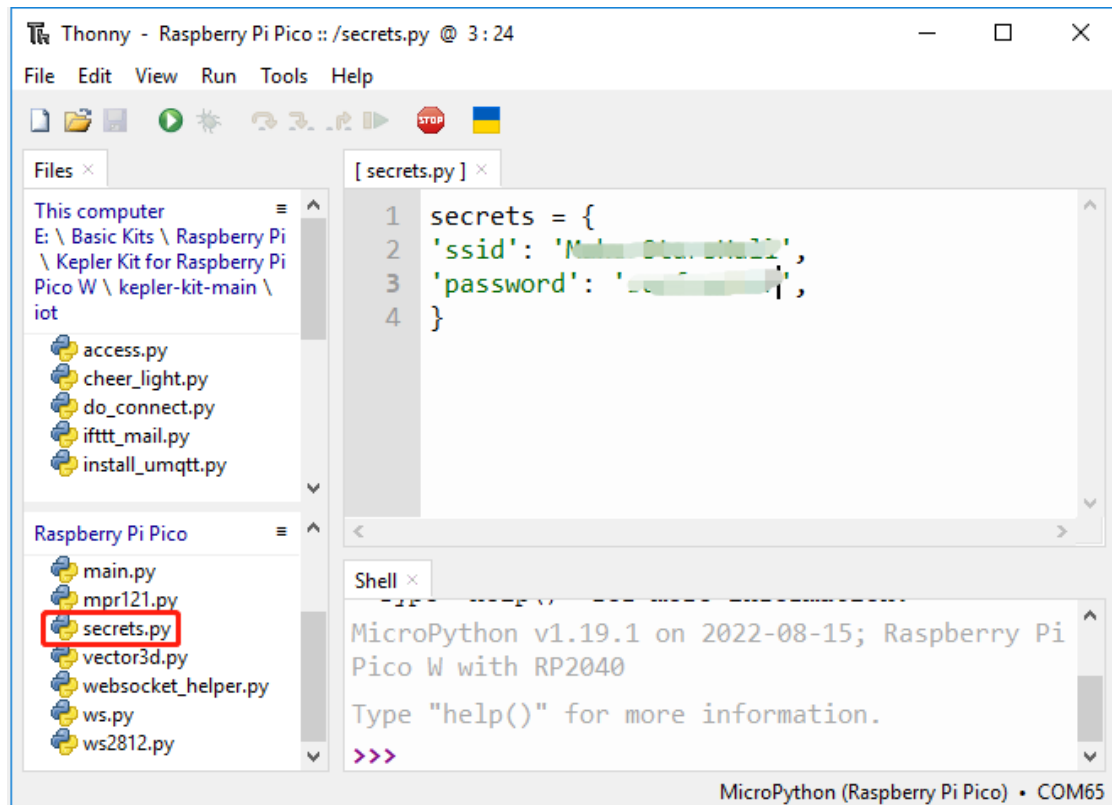
# Wählen Sie Raspberry Pi Pico im Popup-Fenster aus, das erscheint, wenn Sie auf „Speichern“ klicken oder Strg+S drücken.



# Vergeben Sie den Dateinamen `secrets.py`.



# Nun ist das Skript auf Ihrem Raspberry Pi Pico W einsehbar.



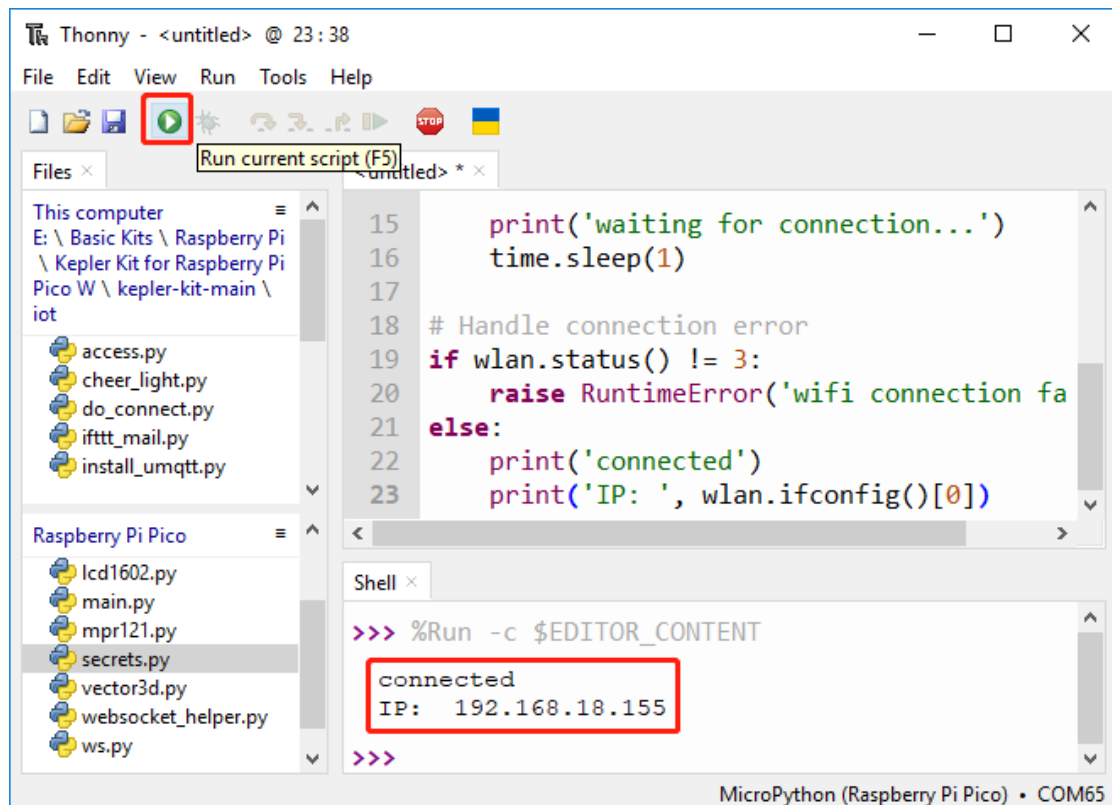
# In anderen Skripten können Sie die Datei folgendermaßen einbinden. Nach der Ausführung sollten Sie eine erfolgreiche WLAN-Verbindung feststellen. Die `secrets.py`-Datei wird als Bibliothek importiert, sodass kein Risiko eines Informationslecks besteht.

```
import network
import time
from secrets import secrets

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(secrets['ssid'], secrets['password'])

# Wait for connect or fail
wait = 10
while wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    wait -= 1
    print('waiting for connection...')
    time.sleep(1)

# Handle connection error
if wlan.status() != 3:
    raise RuntimeError('wifi connection failed')
else:
    print('connected')
    print('IP: ', wlan.ifconfig()[0])
```



### 5.1.4 4. Internetverbindung über do\_connect.py

Da unsere zukünftigen Projekte aller Voraussicht nach eine Netzwerkverbindung benötigen, bietet es sich an, eine separate do\_connect.py-Datei anzulegen und darin die relevanten Funktionen zu speichern. So können wir den Code für komplexere Projekte erheblich vereinfachen.

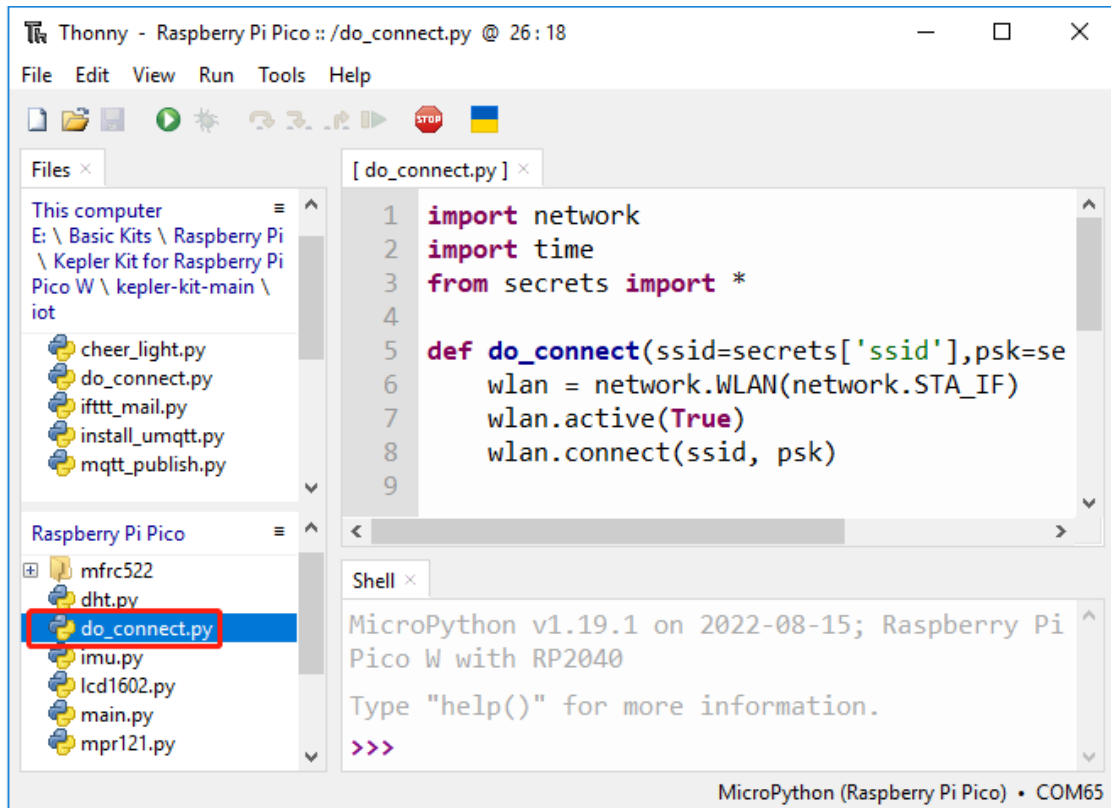
# Kopieren Sie den nachfolgenden Code in eine neue Skriptdatei und speichern Sie diese auf dem Raspberry Pi Pico als do\_connect.py.

```
import network
import time
from secrets import *

def do_connect(ssid=secrets['ssid'],psk=secrets['password']):
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, psk)

    # Wait for connect or fail
    wait = 10
    while wait > 0:
        if wlan.status() < 0 or wlan.status() >= 3:
            break
        wait -= 1
        print('waiting for connection...')
        time.sleep(1)

    # Handle connection error
    if wlan.status() != 3:
        raise RuntimeError('wifi connection failed')
    else:
        print('connected')
        ip=wlan.ifconfig()[0]
        print('network config: ', ip)
        return ip
```



# Ein Aufruf in anderen Skripten nach dem folgenden Schema ermöglicht die Netzwerkverbindung des Raspberry Pi Pico W.

```
from do_connect import *
do_connect()
```

## 5.2 2. Folgen Sie dem @CheerLights

Dies ist ein romantisches Projekt: Werden Sie Teil der LED-Farbwechsel-Community und ermöglichen Sie LEDs weltweit, gleichzeitig die Farbe zu wechseln.

Platzieren Sie es in einer Ecke Ihres Büros, um sich daran zu erinnern, dass Sie nicht alleine sind.

Sie können einfach einen Tweet mit @cheerlights und dem gewünschten Farbnamen absenden. Dies wird die LEDs weltweit in die von Ihnen angegebene Farbe tauchen.

### Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Ein vollständiges Set zu kaufen ist definitiv praktisch, hier ist der Link dazu:

Name	ARTIKEL IM KIT	LINK
Kepler Kit	450+	

Sie können die Teile auch einzeln über die folgenden Links erwerben.

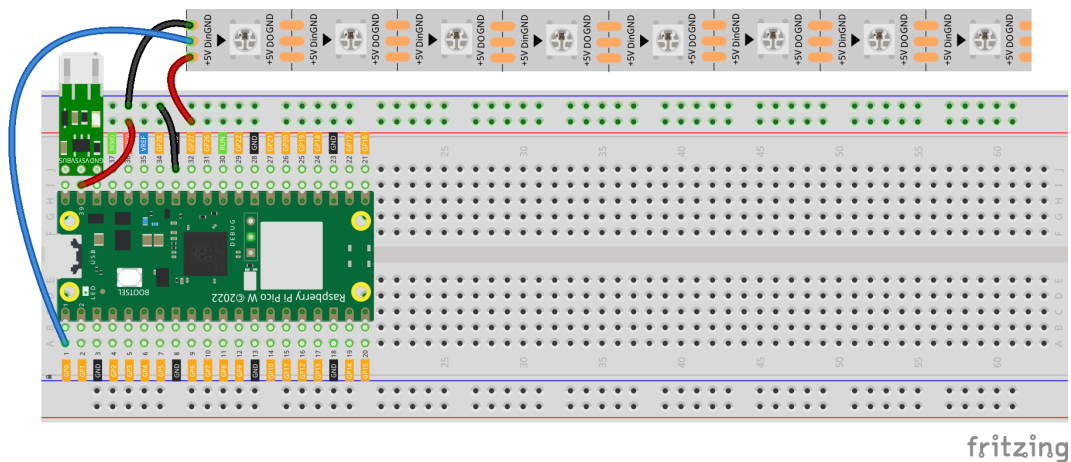
SN	KOMPONENTE	MEN-GE	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>WS2812 RGB 8-LED-Streifen</i>	1	
6	<i>Li-Po-Lademodul</i>	1	
7	18650-Batterie	1	
8	Batteriehalter	1	

## Schritte

### 1. Schaltkreis aufbauen.

Das hier verwendete Li-Po-Lademodul versorgt Ihren Schaltkreis mit Strom, sodass Sie das USB-Kabel abziehen und Ihr Projekt woanders nutzen können!

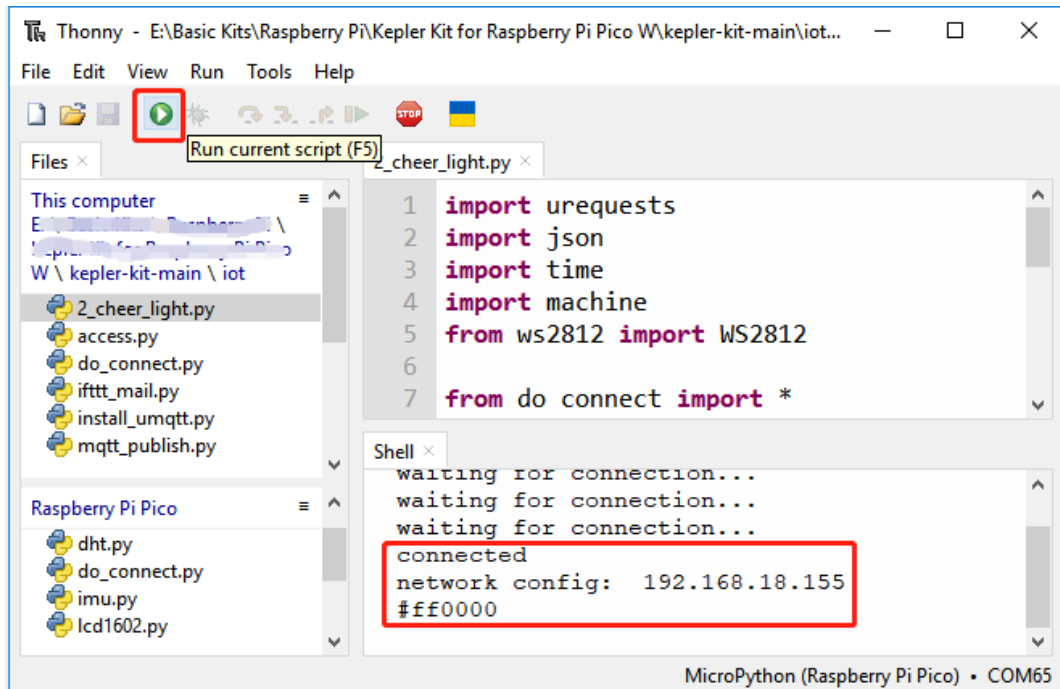
**Warnung:** Stellen Sie sicher, dass Ihr Li-Po-Lademodul so angeschlossen ist, wie im Diagramm dargestellt. Andernfalls könnte ein Kurzschluss Ihren Akku und die Schaltung beschädigen.



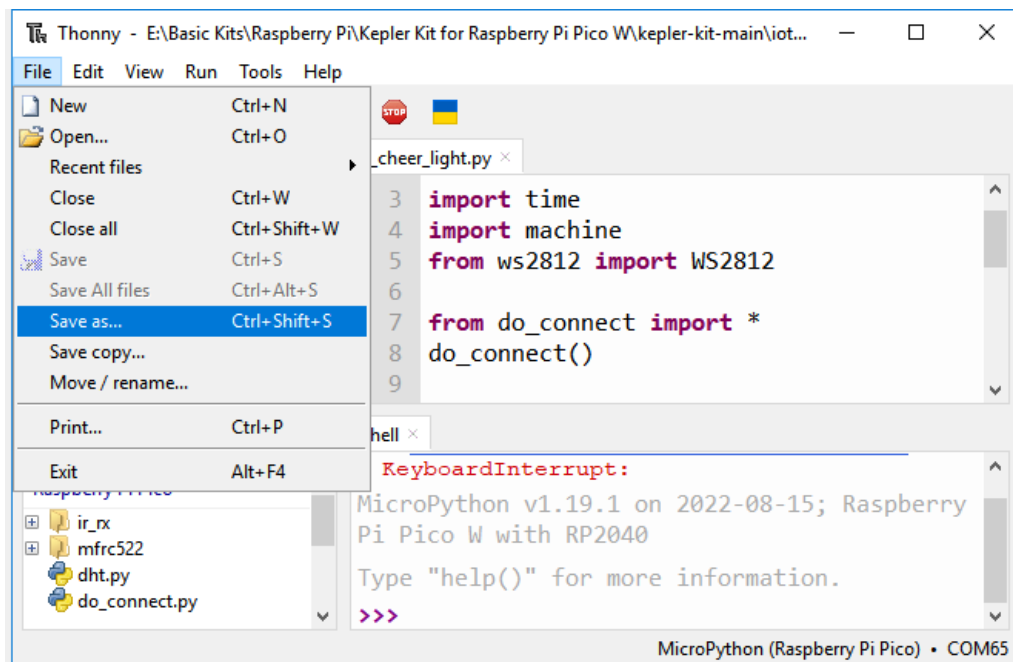
2. Wechseln Sie in das Verzeichnis, in dem Sie zuvor das [Code-Paket](#) heruntergeladen haben, und öffnen Sie die Datei `2_cheer_light.py` im Pfad `kepler-kit-main/iot`.
3. Um das Skript auszuführen, klicken Sie auf die Schaltfläche **Aktuelles Skript ausführen** oder drücken Sie F5. Anschließend sehen Sie die Verbindungsaufforderung, die IP-Adresse und die Farbe (0xff0000 steht für Rot) in der Shell.

**Bemerkung:** Bevor Sie den Code ausführen, stellen Sie sicher, dass Sie die Skripte `do_connect.py` und `secrets.py` auf Ihrem Pico W haben. Falls nicht, beziehen Sie sich auf [1. Zugang zum Netzwerk](#), um diese zu erstellen.

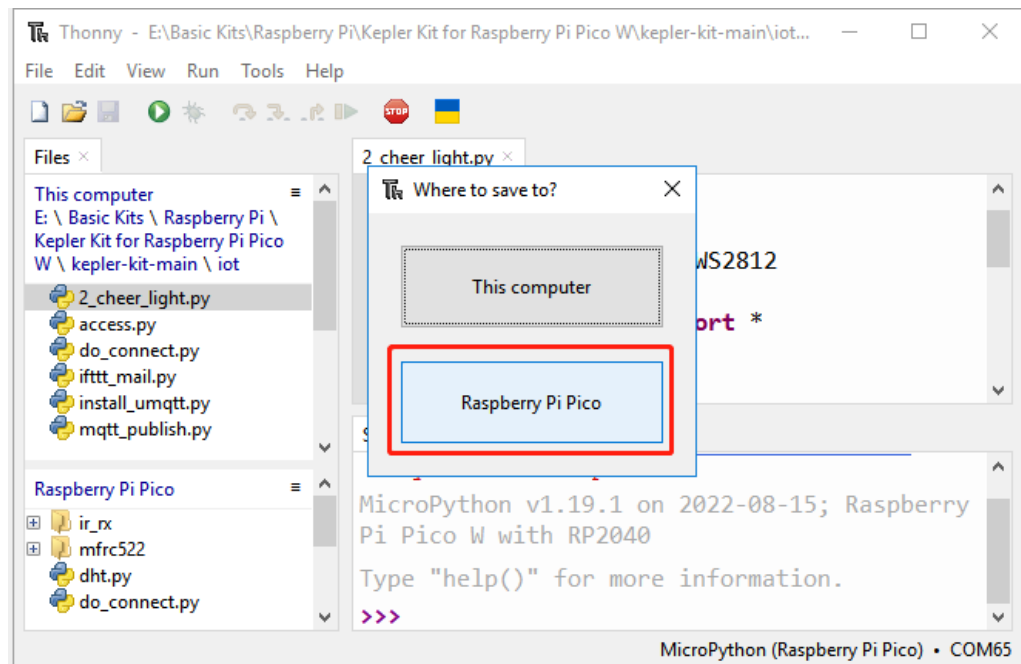




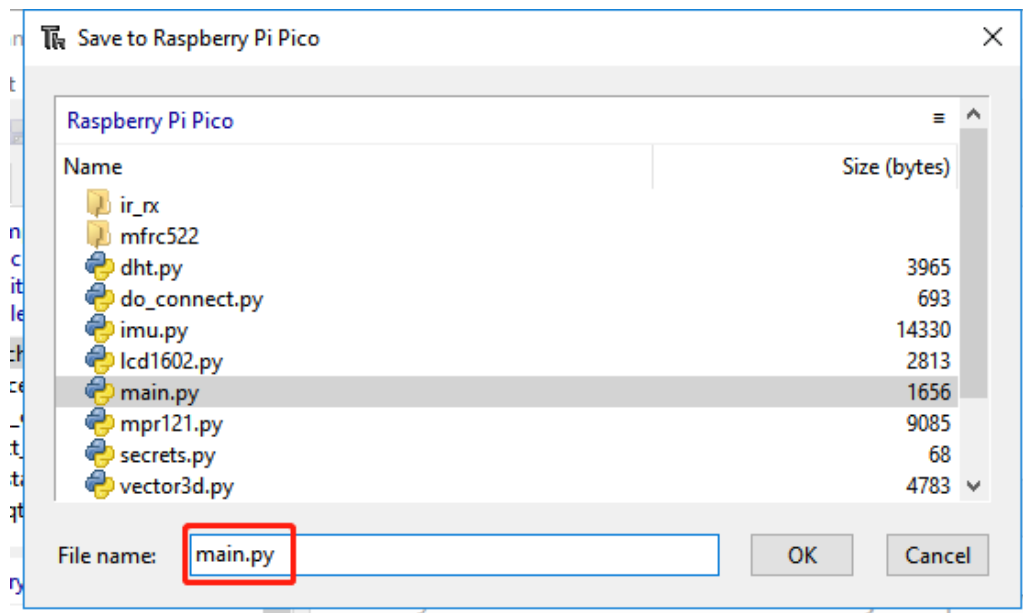
4. Nachdem das Skript ausgeführt wurde, zeigt der WS2812 RGB-Streifen eine Farbe an, die sich manchmal ändern wird.
5. Wenn Sie dieses Skript beim Booten ausführen möchten, müssen Sie es als `main.py` auf dem Raspberry Pi Pico W speichern. Folgendermaßen:
  - Beenden Sie das laufende Skript und klicken Sie auf **Datei -> Speichern unter**.



- Wählen Sie im aufklappenden Fenster **Raspberry Pi Pico** aus.



- Setzen Sie den Dateinamen auf `main.py`. Ein Hinweis erscheint, wenn die Datei bereits auf Ihrem Pico W existiert.



- Jetzt können Sie das USB-Kabel abziehen und das Raspberry Pi Pico W über das Li-Po-Lademodul mit Strom versorgen. Stellen Sie es in eine Ecke, und es wird automatisch funktionieren.

### Wie funktioniert es?

Das Raspberry Pi Pico W muss mit dem Internet verbunden sein, wie in [1. Zugang zum Netzwerk](#) beschrieben. Für dieses Projekt reicht das aus.

```
from do_connect import *
do_connect()
```

WS2812 RGB-Streifen einstellen; weitere Nutzungsdetails finden Sie unter [3.3 RGB LED-Streifen](#).

```
import machine
from ws2812 import WS2812
ws = WS2812(machine.Pin(18), 8)
```

Jetzt brauchen wir eine Methode, um die Farbe von @CheerLights abzurufen. Es gibt ein Backend-System, das die Farbänderungen von Twitter empfängt und sie im JSON-Format an die URL: <http://api.thingspeak.com/channels/1417/field/2/last.json> sendet.

Wenn Sie diese URL direkt in Ihrem Browser öffnen, sehen Sie etwas Ähnliches wie das Folgende. Wir benötigen lediglich die field2 Daten, die einen hexadezimalen Farbcode darstellen.

```
{"created_at": "2022-08-16T06:12:44Z", "entry_id": 870488, "field2": "#ff00ff"}
```

Wir verwenden das urequests Modul, um diese Daten abzurufen und das json Modul, um diesen String in ein Python-Wörterbuch zu konvertieren. Der folgende Code holt die neueste @CheerLights-Farbe von der URL und gibt einen Farbwert zurück, der von WS2812 verwendet werden kann.

```
def get_colour():
    url = "http://api.thingspeak.com/channels/1417/field/2/last.json"
    try:
        r = urequests.get(url)
        if r.status_code > 199 and r.status_code < 300:
            cheerlights = json.loads(r.content.decode('utf-8'))
            print(cheerlights['field2'])
            colour = int('0x'+cheerlights['field2'][1:7]) #Von String zu Integer
            ↪konvertieren
            r.close()
            return colour
        else:
            return None
    except Exception as e:
        print(e)
        return None
```

Abschließend verwenden wir eine Schleife, um den WS2812 einmal pro Minute zu betreiben.

```
while True:
    colour = get_colour()
    if colour is not None:
        ws.write_all(colour)
    time.sleep(60)
```

## 5.3 3. Sicherheitssystem über @IFTTT

Mit diesem Projekt erstellen wir ein Sicherheitsgerät, das mithilfe eines PIR-Sensors erkennt, wenn ein Einbrecher oder ein streunendes Tier in Ihr Zuhause eindringt. In diesem Fall erhalten Sie eine E-Mail-Benachrichtigung.

Webhook wird als grundlegendster Dienst verwendet. Ein POST-Request wird von Raspberry Pi Pico W an den IFTTT-Dienst gesendet. Mit IFTTT erstellen wir ein Applet, das den Webhook abfängt und eine E-Mail sendet.

### 1. Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

Ein Komplet-Set ist definitiv praktisch, hier ist der Link dazu:

Bezeichnung	ARTIKEL IM KIT	LINK
Kepler-Kit	450+	

Sie können die Bauteile auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1(S8050)	
6	<i>Widerstand</i>	2(1K, 10K)	
7	<i>Taster</i>	1	
8	Aktiver <i>Summer</i>	1	
9	<i>Li-Po-Lademodul</i>	1	
10	18650-Batterie	1	
11	Batteriehalter	1	
12	<i>PIR-Bewegungssensormodul</i>	1	

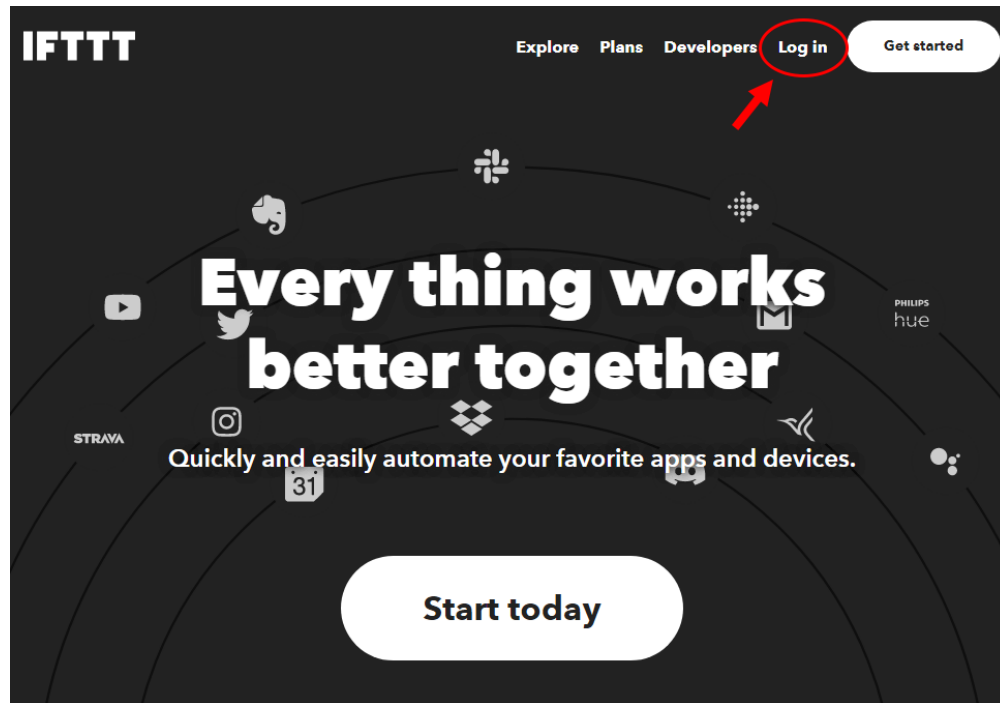
## 2. Schaltkreis aufbauen

**Warnung:** Achten Sie darauf, dass Ihr Li-Po-Ladegerät wie im Schaltplan dargestellt angeschlossen ist. Andernfalls kann ein Kurzschluss sowohl Ihre Batterie als auch Ihre Schaltung beschädigen.

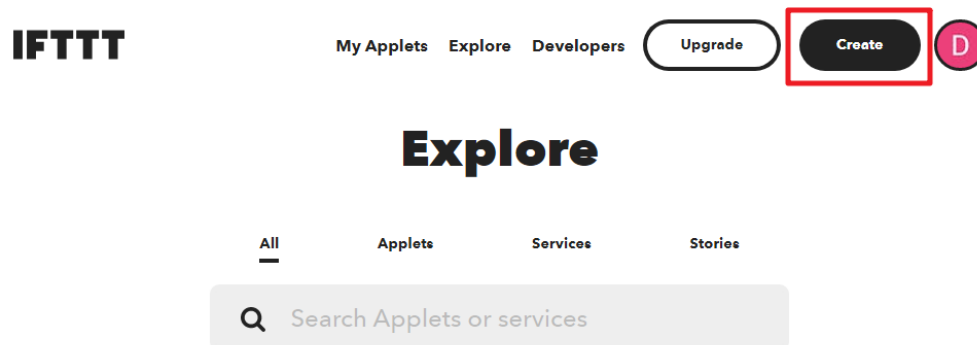
IFTTT ist ein kostenloser Dienst, der vielfältige Möglichkeiten bietet, unterschiedliche Datendienste miteinander zu verknüpfen.

Bitte folgen Sie den unten stehenden Schritten auf IFTTT.

- ### 5.3. 3. Sicherheitssystem über @IFTTT



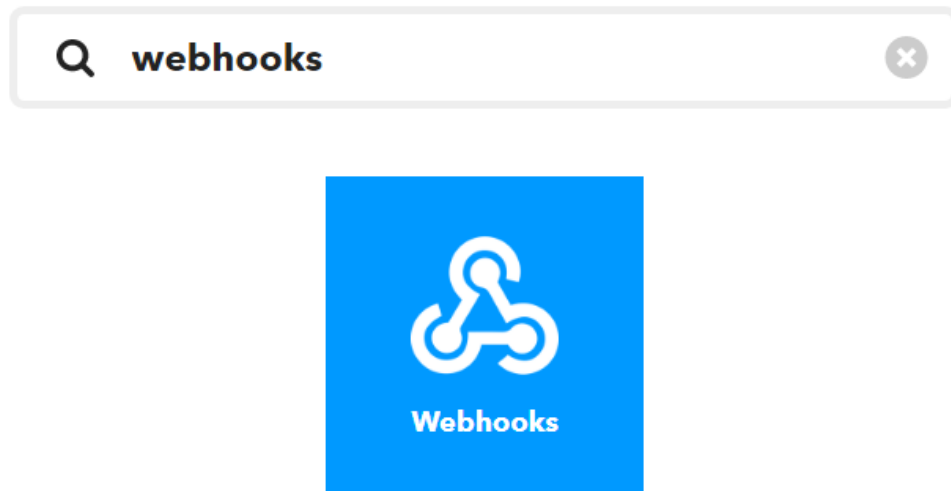
2. Klicken Sie auf **Erstellen**.



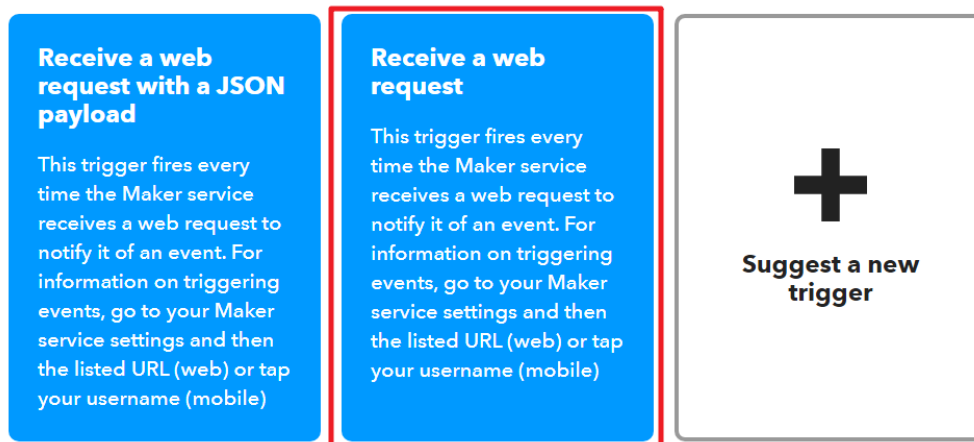
3. Fügen Sie ein **Wenn das** Ereignis hinzu.



4. Suchen Sie nach **Webhooks**.



5. Wählen Sie **Web-Anforderung empfangen** aus.



6. Tragen Sie den Ereignisnamen ein (z.B. Sicherheitswarnung) und klicken Sie auf **Trigger erstellen**.

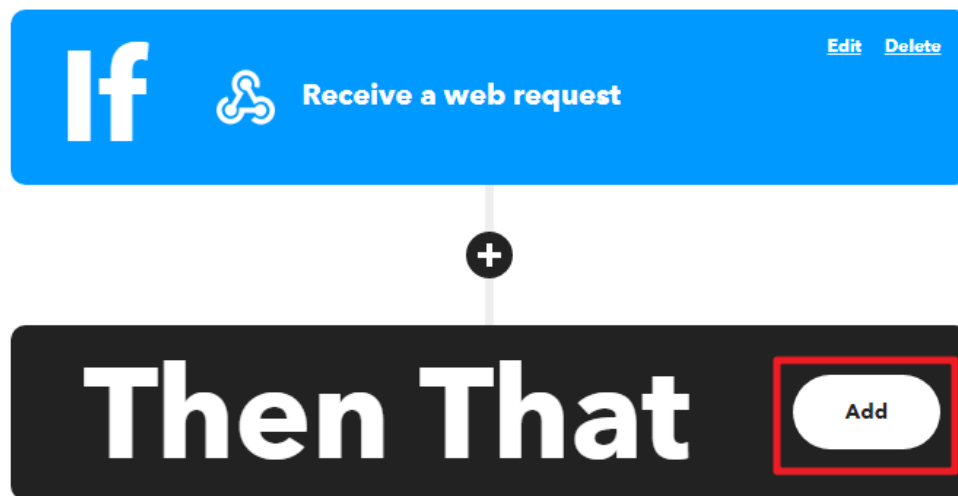
**Event Name**

**SecurityWarning**

The name of the event, like "button\_pressed" or "front\_door\_opened". Use only letters, numbers, and underscores

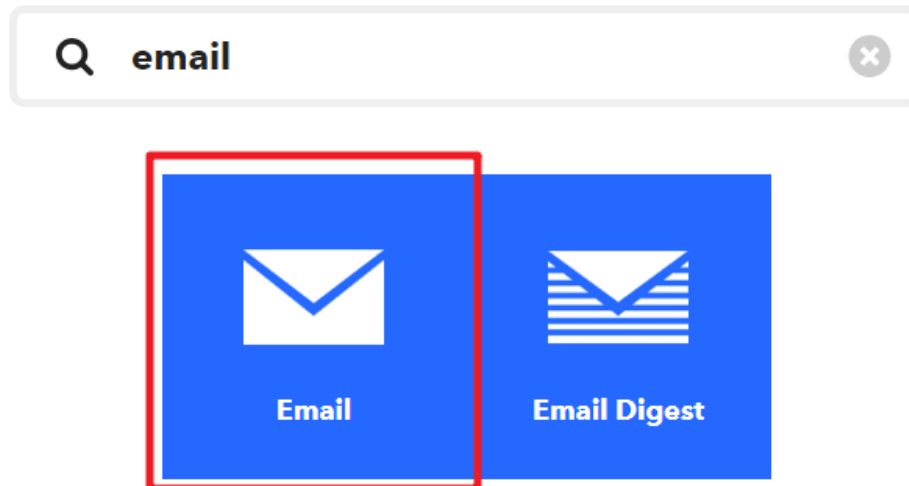
**Create trigger**

7. Fügen Sie ein **Dann das** Ereignis hinzu.

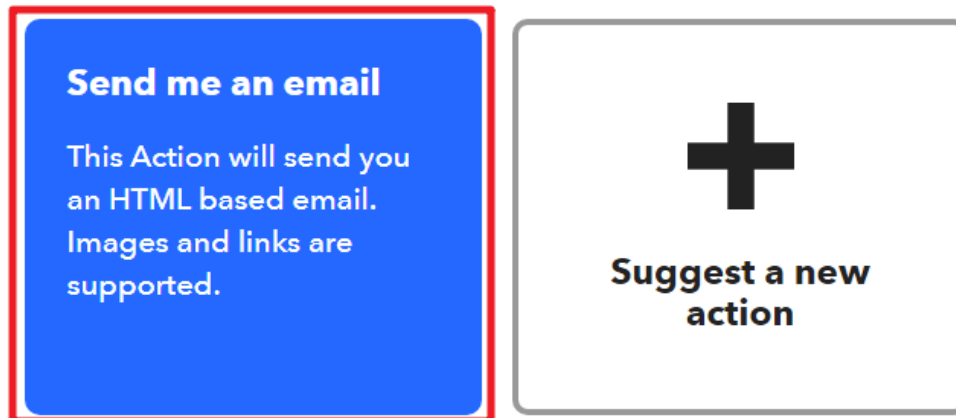


8. Suchen Sie nach E-Mail.






9. Klicken Sie auf **Mir eine E-Mail senden**.



10. Füllen Sie **Betreff** und **Inhalt** aus und klicken Sie dann auf **Aktion erstellen**.



# Send me an email

This Action will send you an HTML based email. Images and links are supported.

**Subject**

SecurityWarning

Add ingredient

**Body**

What: **EventName** <br>  
When: **OccurredAt** <br>

Add ingredient

Create action



11. Klicken Sie auf **Weiter**, um die Konfiguration abzuschließen.



12. Ändern Sie den Titelnamen und schon sind Sie fertig.

The diagram shows the IFTTT applet configuration interface. It consists of a blue box with a trigger icon and an action icon. Below the icons is the text 'Applet Title'. Below that is a white input field containing the text 'PIR Sensor & Send Email'. Below the input field is the text 'by kyuinochi' and '23/140'. Below the blue box is a large black rounded rectangle with the text 'Finish'.

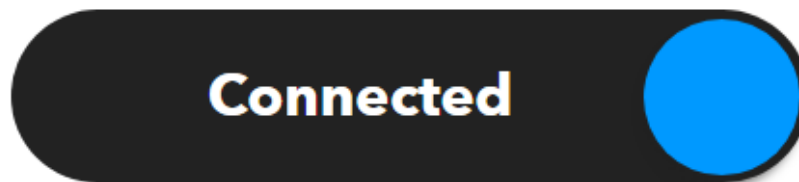
13. Sie werden automatisch zur Detailseite des Applets weitergeleitet, wo Sie sehen können, dass das Applet derzeit verbunden ist. Hier können Sie den Schalter umlegen, um es zu starten bzw. zu stoppen.



# PIR Sensor & Send Email

[Edit title](#)

by **kyuuinochi**



**Connected Aug 19, 2022**

**Never run**

Check the log of your  
Applet runs

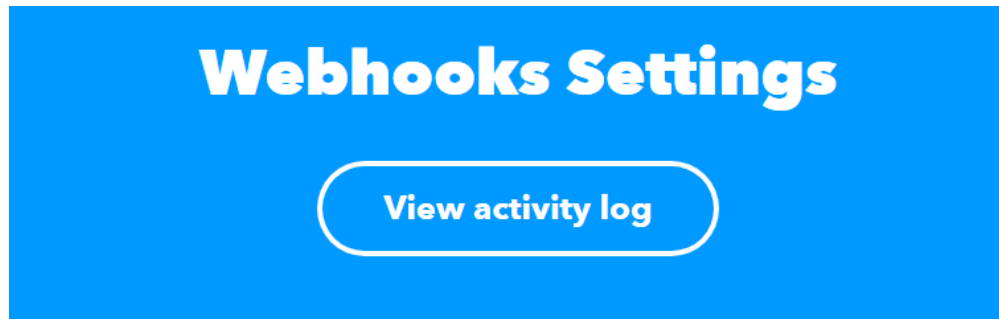
**View activity**

Realtime Applets usually  
run within 10 seconds

**Check now**

#### 4. Ausführung des Skripts

1. Jetzt, wo wir das IFTTT-Applet erstellt haben, benötigen wir noch den API-Schlüssel, den Sie von beziehen können, um den Zugang des Pico W zu IFTTT zu ermöglichen.



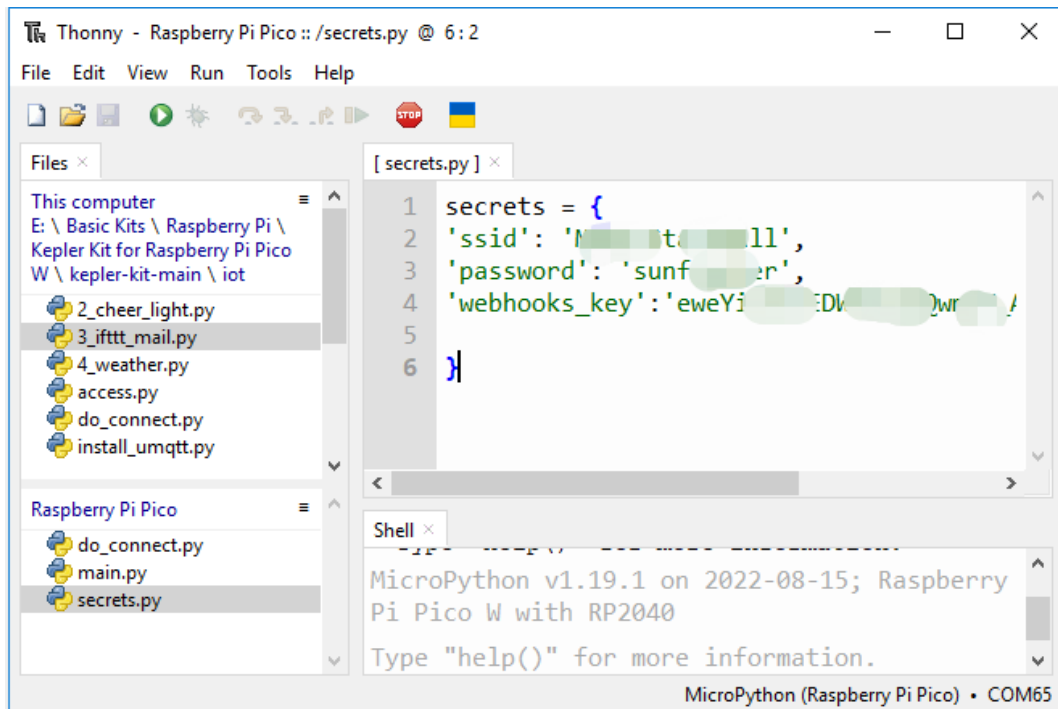
## Details



Status  
Active

URL  
https://maker.ifttt.com/uses/...xp

2. Kopieren Sie diesen in das secrets.py Skript auf Ihrem Raspberry Pi Pico W.



**Bemerkung:** Falls die Skripte do\_connect.py und secrets.py noch nicht auf Ihrem Pico W vorhanden sind, entnehmen Sie bitte *1. Zugang zum Netzwerk*, wie diese erstellt werden können.

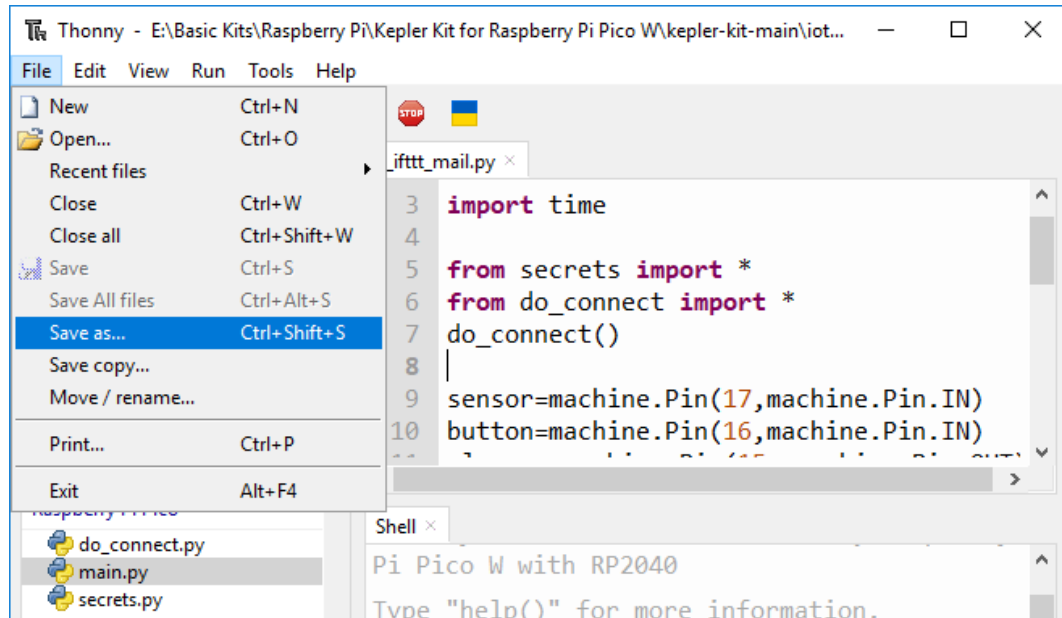
```
secrets = {
```

(Fortsetzung auf der nächsten Seite)

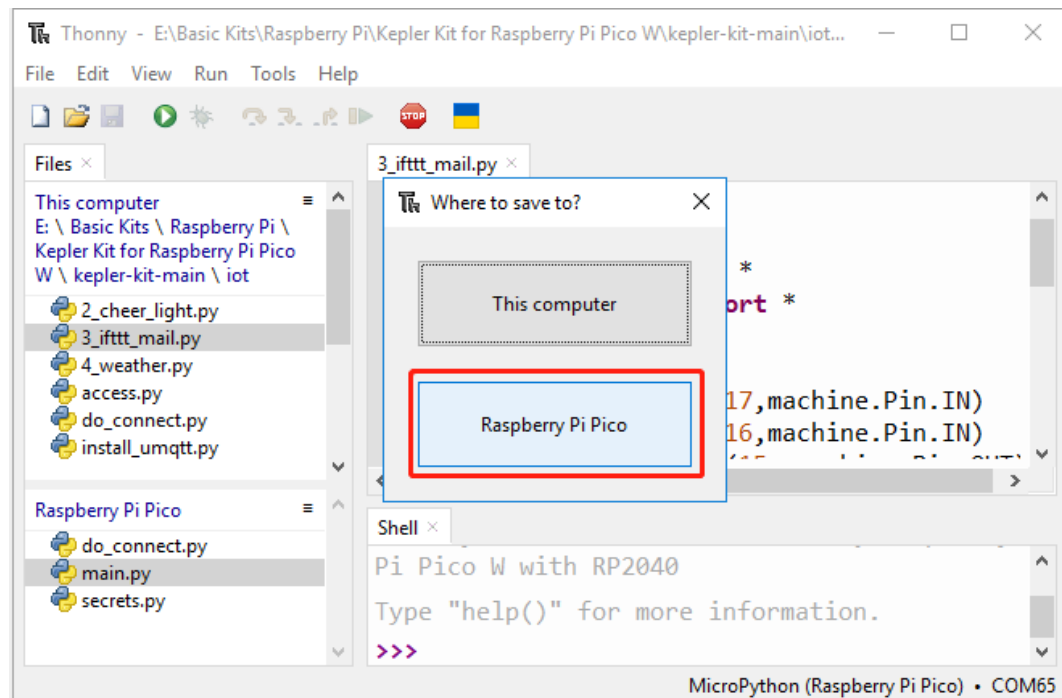
(Fortsetzung der vorherigen Seite)

```
'ssid': 'SSID',
'password': 'PASSWORT',
'webhooks_key': 'WEBHOOKS_API_KEY'
}
```

3. Öffnen Sie die Datei `3_ifttt_mail.py` im Verzeichnis `kepler-kit-main/iot` und wählen Sie dann **Datei** -> **Speichern unter** oder drücken Sie `Strg+Umschalt+S`.

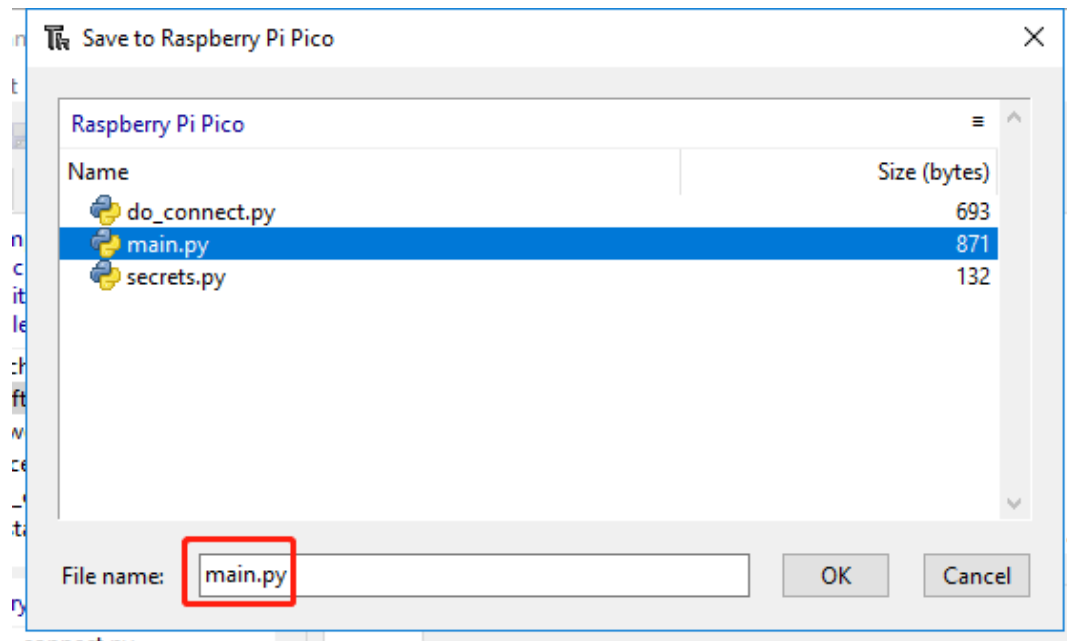


4. Wählen Sie im erscheinenden Popup-Fenster **Raspberry Pi Pico** aus.



5. Benennen Sie die Datei als `main.py` um. Eine Aufforderung erscheint, falls bereits eine gleichnamige Datei auf

Ihrem Pico W existiert.



- Nun können Sie das USB-Kabel entfernen und den Raspberry Pi Pico W über das Li-Po-Ladegerät mit Strom versorgen. Sobald das Skript läuft, ertönt ein akustisches Signal. Der Signalton setzt sich fort, wenn das PIR-Modul eine vorbeigehende Person oder ein Tier erkennt, und eine E-Mail-Warnung wird an Sie gesendet. Durch Betätigen der Taste kann das Skript neu gestartet werden.

### So funktioniert es

Der Raspberry Pi Pico W muss, wie in *1. Zugang zum Netzwerk* beschrieben, mit dem Internet verbunden sein. Für dieses Projekt reicht das aus.

```
from do_connect import *
do_connect()
```

Liest Daten vom PIR-Modul und ruft die Funktion `motion_detected()` auf, wenn es jemanden vorbeigehen bemerkt (Daten von 0 bis 1).

```
import machine

sensor=machine.Pin(17,machine.Pin.IN)

sensor.irq(trigger=machine.Pin.IRQ_RISING, handler=motion_detected)
```

Als Nächstes sendet der Pico W Daten an IFTTT. Wie Sie sehen können, ist die `message`, die Sie an IFTTT senden, eine URL-Zeichenfolge. IFTTT identifiziert den Absender über `secrets['webhooks_key']`, das ausgelöste Ereignis wird durch `event` identifiziert. Stellen Sie also sicher, dass diese korrekt sind.

```
import urequests
from secrets import *

event='SecurityWarning'
message=f"https://maker.ifttt.com/trigger/{event}/with/key/{secrets['webhooks_key']}"

def motion_detected(pin):
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

urequests.post(message)
print(message)
global warn_flag
warn_flag=True
sensor.irq(handler=None)

```

Wenn `motion_detected()` aufgerufen wird, wird die Variable `warn_flag` auf `True` gesetzt, was dazu führt, dass der Signalton weitergeht.

```

while True:
    if warn_flag==True:
        alarm.toggle()
        time.sleep_ms(50)

```

Der Button dient hier zum Neustart des Skripts.

```

button=machine.Pin(16,machine.Pin.IN)

def reset_device(pin):
    machine.reset()

button.irq(trigger=machine.Pin.IRQ_RISING, handler=reset_device)

```

## 5.4 4. Echtzeit-Wetterdaten von @OpenWeatherMap

Dieses Projekt zielt darauf ab, eine intelligente Uhr zu bauen, die neben der aktuellen Uhrzeit auch das Wetter in Ihrer Stadt auf dem LCD-Display anzeigt.

### 1. Erforderliche Bauteile

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist sicherlich praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	INHALT DES KITS	LINK
Kepler-Kit	450+	

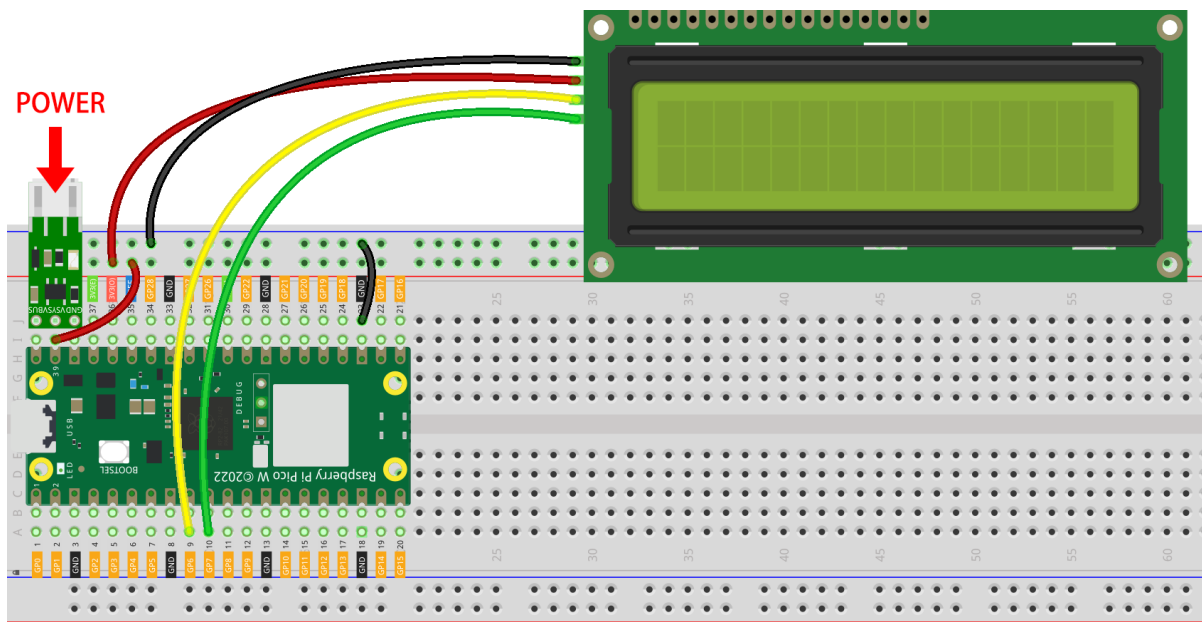
Sie können die Bauteile aber auch separat über die untenstehenden Links erwerben.



SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>I2C LCD1602</i>	1	
6	<i>Li-Po-Lademodul</i>	1	
7	18650-Batterie	1	
8	Batteriehalter	1	

## 2. Schaltungsaufbau

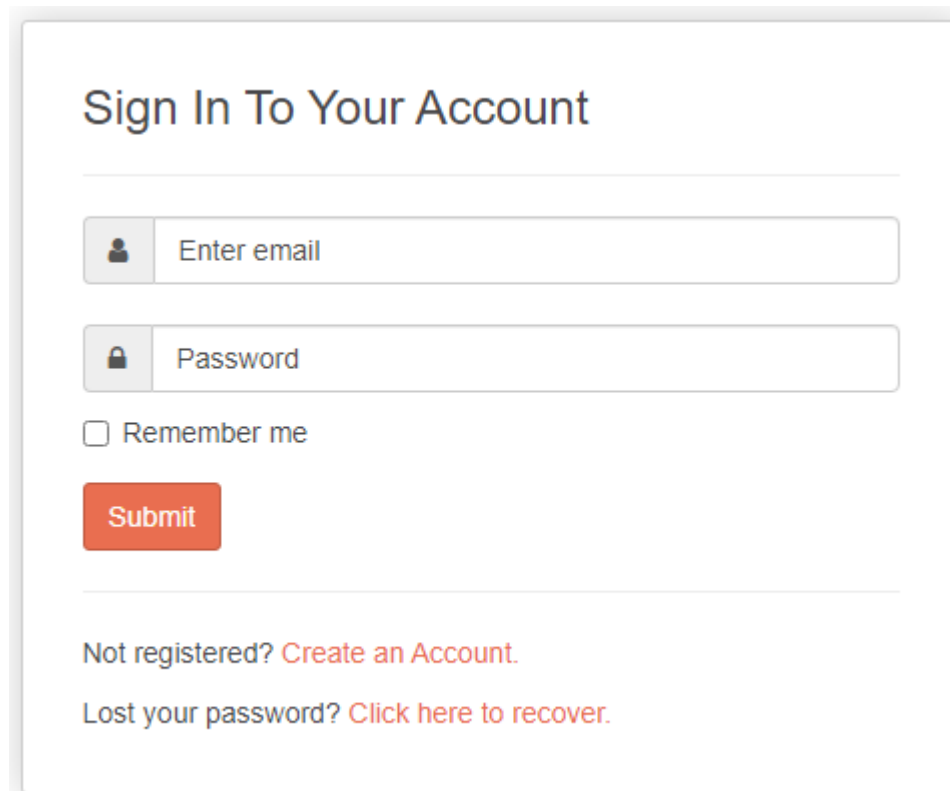
**Warnung:** Achten Sie darauf, das Li-Po-Ladegerät gemäß dem Schaltplan anzuschließen. Andernfalls könnten sowohl der Akku als auch die Schaltung Schaden nehmen.



## 3. OpenWeather-API-Schlüssel erhalten

ist ein Online-Dienst von OpenWeather Ltd, der globale Wetterdaten via API liefert. Dies umfasst aktuelle Wetterdaten, Prognosen, Kurzzeitvorhersagen und historische Wetterdaten für beliebige geografische Standorte.

1. Loggen Sie sich bei ein oder erstellen Sie ein neues Konto.



A sign-in form titled "Sign In To Your Account". It features two input fields: "Enter email" with a person icon and "Password" with a lock icon. Below these is a checkbox labeled "Remember me". A red "Submit" button is positioned below the checkbox. At the bottom, there are two links: "Not registered? [Create an Account.](#)" and "Lost your password? [Click here to recover.](#)".

2. Navigieren Sie zur API-Seite über die Navigationsleiste.



3. Wählen Sie **Aktuelle Wetterdaten** und klicken Sie auf Abonnieren.

## Current Weather Data



- Access current weather data for any location including over 200,000 cities
- We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations
- JSON, XML, and HTML formats
- Included in both free and paid subscriptions

4. Im Bereich **Aktuelle Wetterdaten und Prognosen** wählen Sie den passenden Dienst aus. Für unser Projekt reicht die kostenlose Version aus.

### Current weather and forecasts collection

Free	Startup	Developer	Professional	Enterprise
<div>Get API key</div>	<div>40 USD/ month</div> <div>Subscribe</div>	<div>180 USD/ month</div> <div>Subscribe</div>	<div>470 USD/ month</div> <div>Subscribe</div>	<div>from 2000 USD/ month</div> <div>Subscribe</div>
<div>60 calls/minute</div> <div>1,000,000 calls/month</div>	<div>600 calls/minute</div> <div>10,000,000 calls/month</div>	<div>3,000 calls/minute</div> <div>100,000,000 calls/month</div>	<div>30,000 calls/minute</div> <div>1,000,000,000 calls/month</div>	<div>200,000 calls/minute</div> <div>5,000,000,000 calls/month</div>
<div>Current Weather</div> <div>3-hour Forecast 5 days</div> <div>Hourly Forecast 4 days</div> <div>Daily Forecast 16 days</div> <div>Climatic Forecast 30 days</div>	<div>Current Weather</div> <div>3-hour Forecast 5 days</div> <div>Hourly Forecast 4 days</div> <div>Daily Forecast 16 days</div> <div>Climatic Forecast 30 days</div>	<div>Current Weather</div> <div>3-hour Forecast 5 days</div> <div>Hourly Forecast 4 days</div> <div>Daily Forecast 16 days</div> <div>Climatic Forecast 30 days</div>	<div>Current Weather</div> <div>3-hour Forecast 5 days</div> <div>Hourly Forecast 4 days</div> <div>Daily Forecast 16 days</div> <div>Climatic Forecast 30 days</div>	<div>Current Weather</div> <div>3-hour Forecast 5 days</div> <div>Hourly Forecast 4 days</div> <div>Daily Forecast 16 days</div> <div>Climatic Forecast 30 days</div>

5. Kopieren Sie den Schlüssel von der Seite **API-Schlüssel**.

[New Products](#)
[Services](#)
[API keys](#)
[Billing plans](#)
[Payments](#)
[Block logs](#)
[My orders](#)

[My profile](#)
[Ask a question](#)

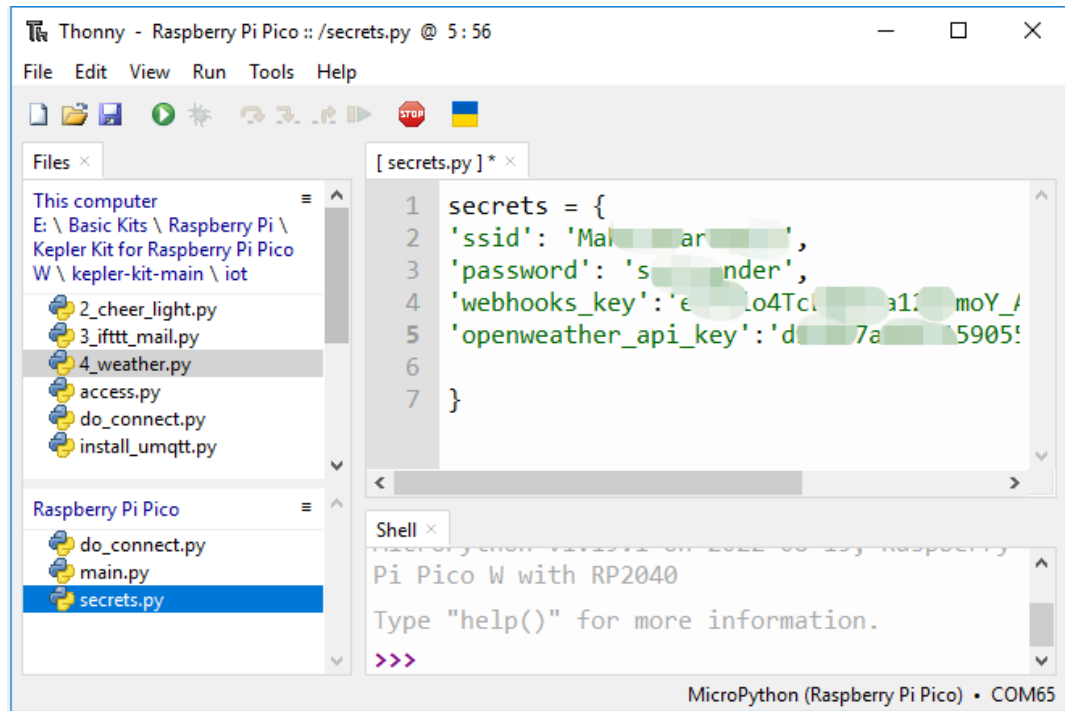
You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

Key	Name	Status	Actions
67ae390557f0d37c	Default	Active	<div></div> <div></div>

Create key

Generate

6. Fügen Sie diesen in das `secrets.py`-Skript auf Ihrem Raspberry Pi Pico W ein.



**Bemerkung:** Falls Sie die Skripte `do_connect.py` und `secrets.py` noch nicht auf Ihrem Pico W haben, entnehmen Sie bitte *1. Zugang zum Netzwerk*, wie diese erstellt werden können.

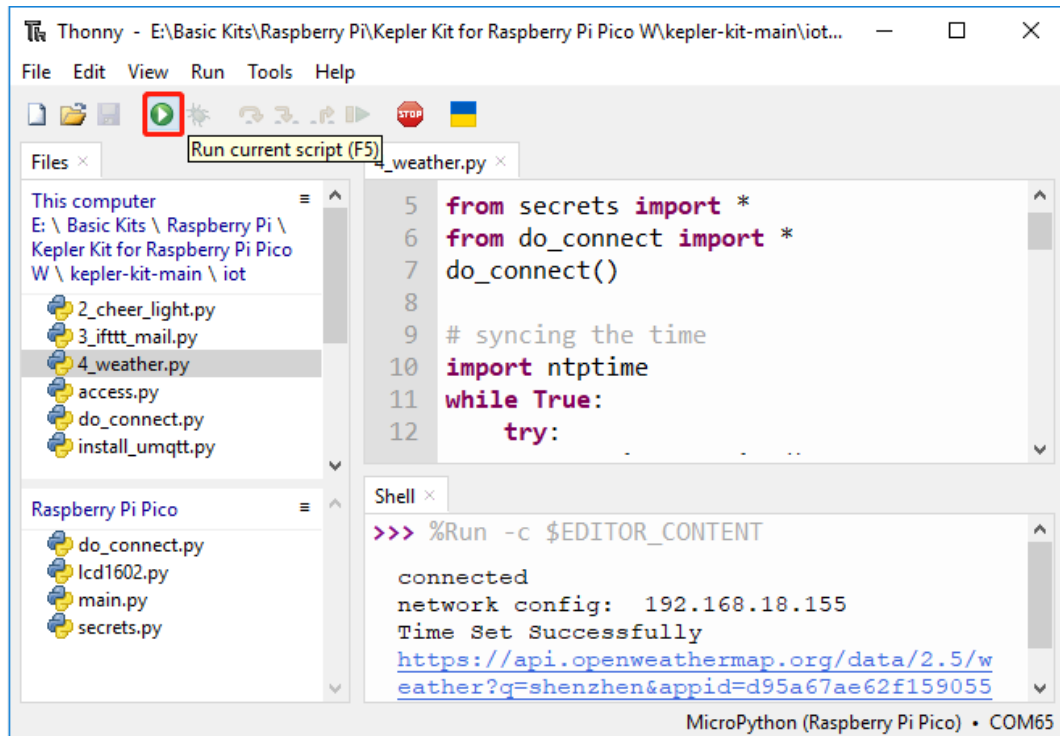
```

secrets = {
'ssid': 'SSID',
'password': 'PASSWORT',
'webhooks_key': 'WEBHOOKS_API_KEY',
'openweather_api_key': 'OPENWEATHERMAP_API_KEY'
}

```

#### 4. Skript ausführen

1. Öffnen Sie die Datei `4_weather.py` im Verzeichnis `kepler-kit-main/iot` und klicken Sie auf **Aktuelles Skript ausführen** oder drücken Sie F5.



- Nachdem das Skript ausgeführt wurde, werden auf dem I2C LCD1602 die Uhrzeit sowie die Wetterinformationen Ihrer Region angezeigt.

**Bemerkung:** Falls das Display leer bleibt, können Sie den Kontrast durch Drehen des Potentiometers auf der Rückseite des Moduls erhöhen.

- Um das Skript automatisch beim Start auszuführen, können Sie es als `main.py` auf dem Raspberry Pi Pico W speichern.

### Wie funktioniert es?

Der Raspberry Pi Pico W muss gemäß *1. Zugang zum Netzwerk* mit dem Internet verbunden sein. Für dieses Projekt verwenden wir es einfach so.

```

from do_connect import *
do_connect()

```

Nachdem die Verbindung zum Internet hergestellt wurde, sorgt dieser Code-Abschnitt für die Synchronisation der Pico W-Uhrzeit mit der Greenwich Mean Time.

```

import ntptime
while True:
    try:
        ntptime.settime()
        print('Time Set Successfully')
        break
    except OSError:
        print('Time Setting...')
        continue

```

Für die Initialisierung Ihres LCD verweisen wir auf *3.4 Flüssigkristallanzeige*.

```
from lcd1602 import LCD
lcd = LCD()
lcd.clear()
string = 'Lade...'
lcd.message(string)
```

Bevor wir die Wetterdaten abrufen, müssen wir die Einheit für bestimmte Wetterdaten (z. B. Temperatur, Windgeschwindigkeit) auswählen. In diesem Fall ist die Einheit `metric`.

```
# Open Weather
TEMPERATURE_UNITS = {
    "standard": "K",
    "metric": "°C",
    "imperial": "°F",
}

SPEED_UNITS = {
    "standard": "m/s",
    "metric": "m/s",
    "imperial": "mph",
}

units = "metric"
```

Die folgende Funktion ruft die Wetterdaten von `openweathermap.org` ab. Es wird eine URL-Nachricht mit Ihrer Stadt, API-Schlüsseln und einer festgelegten Einheit an die Website gesendet. Als Antwort erhalten Sie eine JSON-Datei mit den Wetterdaten.

```
def get_weather(city, api_key, units='metric', lang='en'):
    """
    Get weather data from openweathermap.org
    city: City name, state code and country code divided by comma, Please, refer to
    ↪ ISO 3166 for the state codes or country codes. https://www.iso.org/obp/ui/#search
    api_key: Your unique API key (you can always find it on your openweather account
    ↪ page under the "API key" tab https://home.openweathermap.org/api_keys)
    unit: Units of measurement. standard, metric and imperial units are available. If
    ↪ you do not use the units parameter, standard units will be applied by default. More:
    ↪ https://openweathermap.org/current#data
    lang: You can use this parameter to get the output in your language. More: https://
    ↪ openweathermap.org/current#multi
    """
    url = f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}&
    ↪ units={units}&lang={lang}"
    print(url)
    res = urequests.post(url)
    return res.json()
```

Die Rohdaten könnten beispielsweise folgendermaßen aussehen:

```
weather data example:
{
    'timezone': 28800,
    'sys': {
```

(Fortsetzung auf der nächsten Seite)

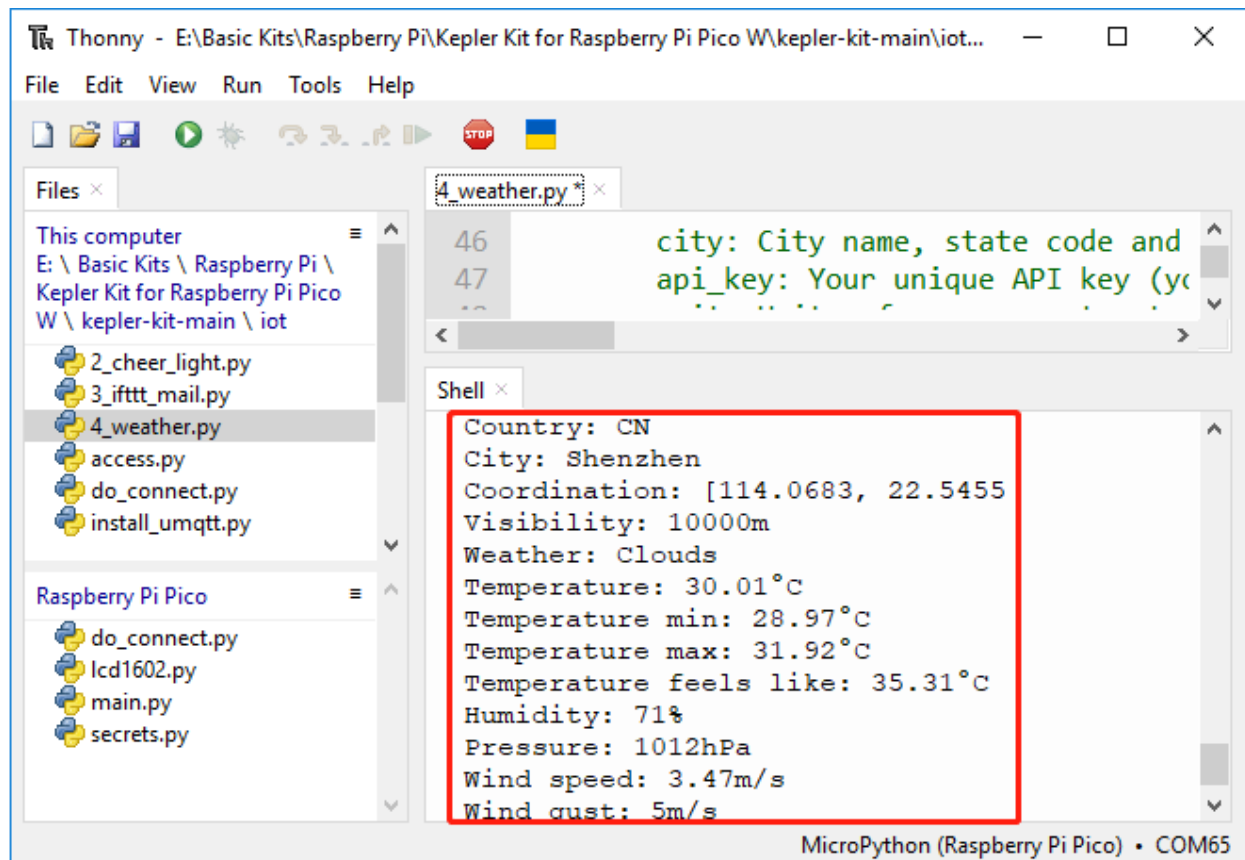
(Fortsetzung der vorherigen Seite)

```

        'type': 2,
        'sunrise': 1659650200,
        'country': 'CN',
        'id': 2031340,
        'sunset': 1659697371
    },
    'base': 'stations',
    'main': {
        'pressure': 1008,
        'feels_like': 304.73,
        'temp_max': 301.01,
        'temp': 300.4,
        'temp_min': 299.38,
        'humidity': 91,
        'sea_level': 1008,
        'grnd_level': 1006
    },
    'visibility': 10000,
    'id': 1795565,
    'clouds': {
        'all': 96
    },
    'coord': {
        'lon': 114.0683,
        'lat': 22.5455
    },
    'name': 'Shenzhen',
    'cod': 200,
    'weather': [{
        'id': 804,
        'icon': '04d',
        'main': 'Clouds',
        'description': 'overcast clouds'
    }],
    'dt': 1659663579,
    'wind': {
        'gust': 7.06,
        'speed': 3.69,
        'deg': 146
    }
}

```

Mit der Funktion `print_weather(weather_data)` werden diese Rohdaten in ein leicht verständliches Format umgewandelt und ausgegeben. Die Funktion ist jedoch nicht aufgerufen, und Sie können diese Zeile in der `while True`-Schleife bei Bedarf einkommentieren.



```
# Ausgabe in der Shell
print_weather(weather_data)
```

In der `while True`-Schleife wird die Funktion `get_weather()` zuerst aufgerufen, um die für dieses Projekt benötigten Wetter-, Temperatur- und Feuchtigkeitinformationen abzurufen.

```
weather_data = get_weather('shenzhen', secrets['openweather_api_key'], units=units)
weather = weather_data["weather"][0]["main"]
t = weather_data["main"]["temp"]
rh = weather_data["main"]["humidity"]
```

Die Ortszeit wird ermittelt. Hierzu wird die Funktion `time.localtime()` aufgerufen, die ein Tupel zurückgibt. Daraus extrahieren wir die Stunden und Minuten.

Beachten Sie, dass wir Pico W bereits auf die Greenwich Mean Time synchronisiert haben. Daher müssen wir die Zeitzone Ihres Standorts hinzufügen.

```
# get time (+24 allows for western hemisphere)
# if negative, add 24
# hours = time.localtime()[3] + int(weather_data["timezone"] / 3600) + 24 #only for west hemisphere
hours=time.localtime()[3]+int(weather_data["timezone"] / 3600)
mins=time.localtime()[4]
```

Schließlich werden die Wetterinformationen und die Zeit auf dem LCD1602 angezeigt.



```

lcd.clear()
time.sleep_ms(200)
string = f'{hours:02d}:{mins:02d} {weather}\n'
lcd.message(string)
string = f'{t}{TEMPERATURE_UNITS[units]} {rh}%rh'
lcd.message(string)

```

Ihr LCD1602 wird zu einer Uhr, die alle 30 Sekunden aktualisiert wird, wenn die Hauptschleife alle 30 Sekunden ausgeführt wird.

## 5.5 5. Cloud-Rufsystem mit @MQTT

Das Message Queuing Telemetry Transport (MQTT) ist ein einfaches Nachrichtenprotokoll. Es ist zudem das gängigste Protokoll für das Internet der Dinge (IoT).

MQTT-Protokolle legen die Art und Weise fest, wie IoT-Geräte Daten übertragen. Sie sind ereignisgesteuert und verwenden das Pub/Sub-Modell für die Vernetzung. Sender (Publisher) und Empfänger (Subscriber) kommunizieren über Topics. Ein Gerät veröffentlicht eine Nachricht zu einem spezifischen Topic, und alle Geräte, die dieses Topic abonniert haben, erhalten die Nachricht.

In diesem Abschnitt wird ein Service-Klingelsystem mit Pico W, HiveMQ (einem kostenlosen öffentlichen MQTT-Broker-Dienst) und vier Tasten vorgestellt. Die vier Tasten stehen für vier Tische im Restaurant. Über HiveMQ können Sie sehen, an welchem Tisch Gäste Bedienung benötigen, wenn ein Kunde die Taste drückt.

### 1. Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten:

Ein Komplettsset zu kaufen, ist definitiv praktisch. Hier ist der Link:

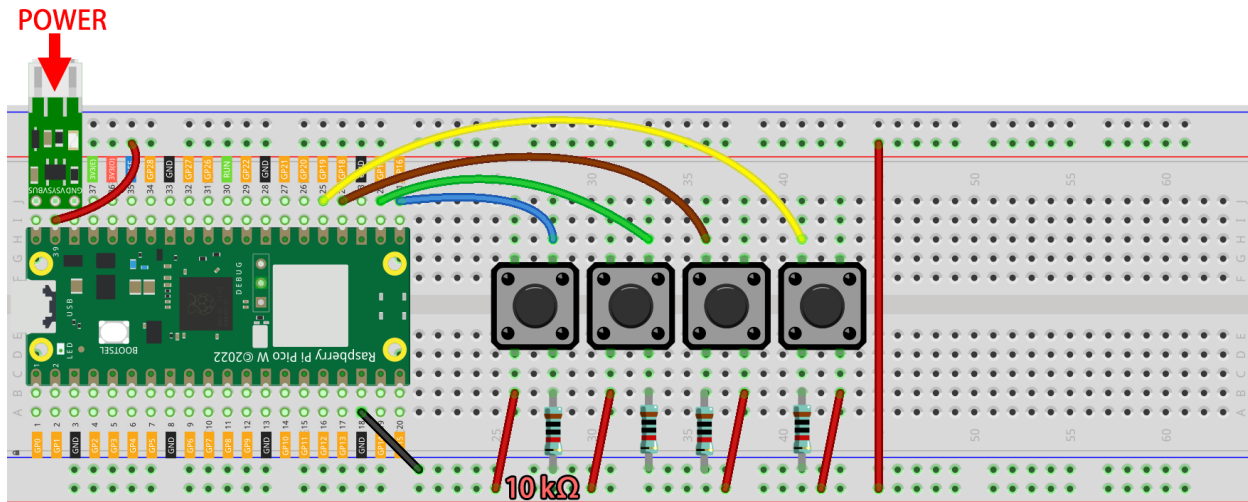
Bezeichnung	ELEMENTE IM SET	LINK
Kepler-Kit	450+	

Die Einzelteile können auch separat über die untenstehenden Links erworben werden.

SN	KOMPONENTE	MEN-GE	LINK
1	<a href="#">Raspberry Pi Pico W</a>	1	
2	Micro-USB-Kabel	1	
3	<a href="#">Steckbrett</a>	1	
4	<a href="#">Jumperkabel</a>	Mehrere	
5	<a href="#">Widerstand</a>	4(10K)	
6	<a href="#">Taster</a>	4	
7	<a href="#">Li-Po-Lademodul</a>	1	
8	18650 Akku	1	
9	Batteriehalter	1	

## 2. Schaltkreis aufbauen

**Warnung:** Achten Sie darauf, dass Ihr Li-Po-Ladegerät wie im Schaltplan dargestellt angeschlossen ist. Andernfalls könnte ein Kurzschluss Ihre Batterie und den Schaltkreis beschädigen.



## 3. HiveMQ besuchen

HiveMQ ist eine MQTT-Broker- und Client-basierte Nachrichtenplattform, die eine schnelle, effiziente und zuverlässige Datenübertragung zu IoT-Geräten ermöglicht.

1. Öffnen Sie in Ihrem Browser.
2. Verbinden Sie den Client mit dem öffentlichen Standard-Proxy.

**Connection** ● disconnected

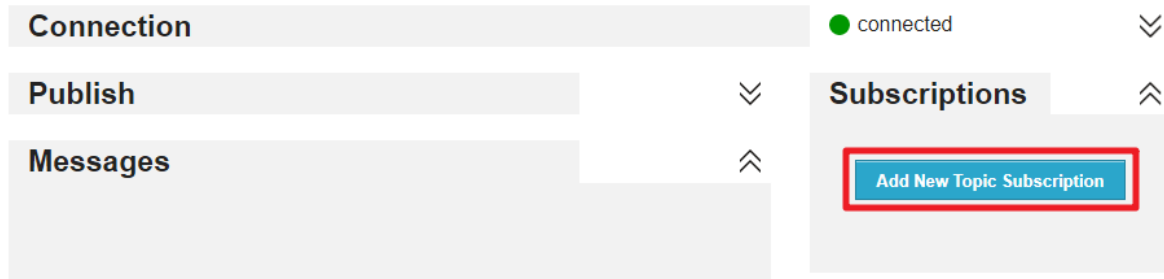
Host:  Port:  ClientID:  **Connect**

Username:  Password:  Keep Alive:  SSL: ☐ Clean Session: ☒

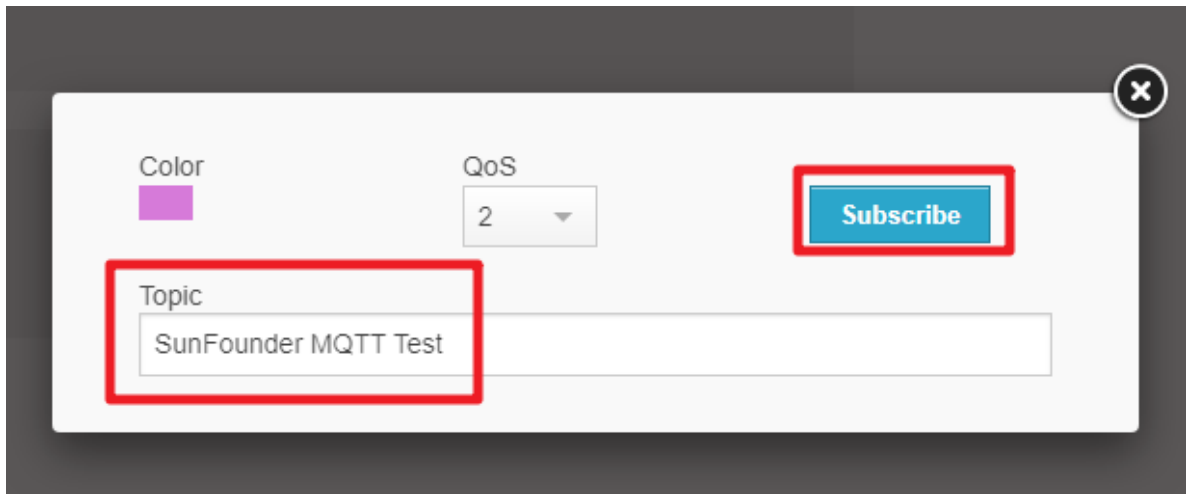
Last-Will Topic:  Last-Will QoS:  Last-Will Retain: ☐

Last-Will Message:

3. Klicken Sie auf **Neues Topic-Abonnement hinzufügen**.



4. Tragen Sie die Topics ein, die Sie verfolgen möchten, und klicken Sie auf **Abonnieren**. Wählen Sie persönliche Topics, um Nachrichten von anderen Benutzern zu vermeiden, und achten Sie auf die Groß- und Kleinschreibung.



#### 4. MQTT-Modul installieren

Bevor wir mit dem Projekt starten können, müssen wir das MQTT-Modul für Pico W installieren.

1. Verbinden Sie sich mit dem Netzwerk, indem Sie `do_connect()` in der Shell ausführen, wie zuvor beschrieben.

##### Bemerkung:

- Geben Sie die folgenden Befehle in die Shell ein und drücken Sie **Enter**, um sie auszuführen.
- Falls Sie die Skripte `do_connect.py` und `secrets.py` noch nicht auf Ihrem Pico W haben, beziehen Sie sich bitte auf [1. Zugang zum Netzwerk](#), um sie zu erstellen.

```
from do_connect import *
do_connect()
```

2. Nach einer erfolgreichen Netzwerkverbindung importieren Sie das `mip`-Modul in der Shell und verwenden `mip` zur Installation des `umqtt.simple`-Moduls, einem vereinfachten MQTT-Client für MicroPython.

```
import mip
mip.install('umqtt.simple')
```

3. Nach Abschluss der Installation sehen Sie, dass das `umqtt`-Modul im `/lib/`-Pfad des Pico W installiert ist.

```

Shell x
do_connect()

connected
network config: 192.168.18.130
'192.168.18.130'

>>> import mip
>>> mip.install('umqtt.simple')

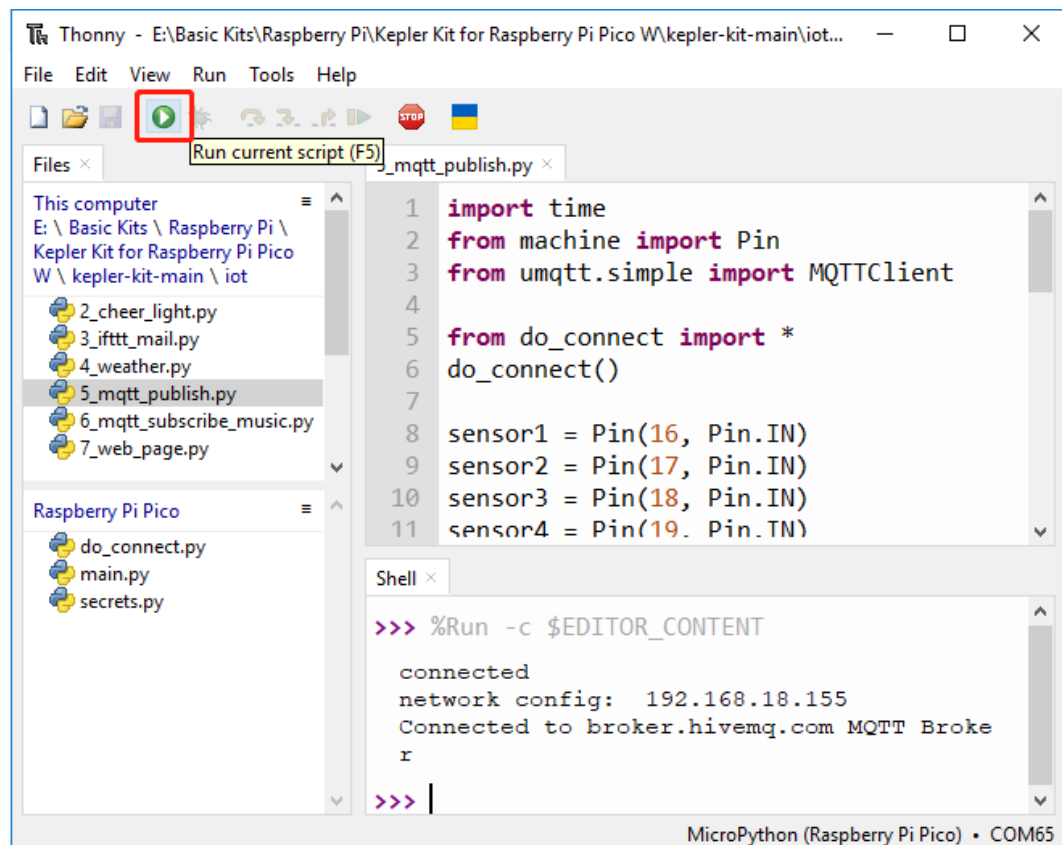
Installing umqtt.simple (latest) from https://micropython.org/pi/v2 to /lib
Exists: /lib/umqtt/simple.mpy
Done

>>> |

```

## 5. Skript ausführen

1. Öffnen Sie die Datei 5\_mqtt\_publish.py im Verzeichnis kepler-kit-main/iot.
2. Klicken Sie auf die Schaltfläche **Aktuelles Skript ausführen** oder drücken Sie F5, um es zu starten.



```

Thonny - E:\Basic Kits\Raspberry Pi\Kepler Kit for Raspberry Pi Pico W\kepler-kit-main\iot...
File Edit View Run Tools Help

Run current script (F5)

Files x
This computer
E:\Basic Kits\Raspberry Pi\Kepler Kit for Raspberry Pi Pico W\kepler-kit-main\iot
2_cheer_light.py
3_ifttt_mail.py
4_weather.py
5_mqtt_publish.py
6_mqtt_subscribe_music.py
7_web_page.py

Raspberry Pi Pico
do_connect.py
main.py
secrets.py

1 import time
2 from machine import Pin
3 from umqtt.simple import MQTTClient
4
5 from do_connect import *
6 do_connect()
7
8 sensor1 = Pin(16, Pin.IN)
9 sensor2 = Pin(17, Pin.IN)
10 sensor3 = Pin(18, Pin.IN)
11 sensor4 = Pin(19, Pin.IN)

Shell x
>>> %Run -c $EDITOR_CONTENT

connected
network config: 192.168.18.155
Connected to broker.hivemq.com MQTT Broker
>>> |

```

3. Kehren Sie zurück und sobald Sie eine der Tasten auf dem Steckbrett drücken, werden Sie die Nachrichten im HiveMQ erscheinen sehen.



4. Falls Sie möchten, dass dieses Skript beim Hochfahren ausgeführt wird, speichern Sie es auf dem Raspberry Pi Pico W als `main.py`.

### Wie funktioniert es?

Das Raspberry Pi Pico W muss mit dem Internet verbunden sein, wie in *1. Zugang zum Netzwerk* beschrieben. Für dieses Projekt reicht das aus.

```
from do_connect import *
do_connect()
```

Initialisieren Sie 4 Tasten-Pins.

```
sensor1 = Pin(16, Pin.IN)
sensor2 = Pin(17, Pin.IN)
sensor3 = Pin(18, Pin.IN)
sensor4 = Pin(19, Pin.IN)
```

Erstellen Sie zwei Variablen, um die URL und die Client-ID des MQTT-Brokers zu speichern, mit dem wir uns verbinden werden. Da wir einen öffentlichen Broker verwenden, wird unsere Client-ID nicht verwendet, selbst wenn eine erforderlich ist.

```
mqtt_server = 'broker.hivemq.com'
client_id = 'Jimmy'
```

Verbinden Sie sich mit dem MQTT-Agenten und halten Sie die Verbindung für eine Stunde. Bei einem Fehlschlag wird der Pico W zurückgesetzt.

```
try:
    client = MQTTClient(client_id, mqtt_server, keepalive=3600)
    client.connect()
    print('Connected to %s MQTT Broker'%(mqtt_server))
except OSError as e:
    print('Failed to connect to the MQTT Broker. Reconnecting...')
    time.sleep(5)
    machine.reset()
```

Erstellen Sie eine Variable `topic`, die das Thema angibt, dem der Abonnent folgen muss. Es sollte dasselbe wie das im **Schritt 4** unter **2. HiveMQ besuchen** ausgefüllte Thema sein. Übrigens konvertiert `b` hier den String in Bytes, da MQTT ein binärbasiertes Protokoll ist, bei dem die Steuerelemente Binärbytes und keine Textstrings sind.

```
topic = b'SunFounder MQTT Test'
```

Setzen Sie Unterbrechungen für jede Taste. Wenn eine Taste gedrückt wird, wird eine Nachricht unter `topic` veröffentlicht.

```
def press1(pin):
    message = b'button 1 is pressed'
    client.publish(topic, message)
    print(message)

sensor1.irq(trigger=machine.Pin.IRQ_RISING, handler=press1)
```

- [UMQTT Client API](#)

## 5.6 6. Cloud-Player mit @MQTT

Es wird empfohlen, zunächst das Projekt *5. Cloud-Rufsystem mit @MQTT* abzuschließen, um einige Module zu installieren und die Konfiguration der HiveMQ-Plattform vorzunehmen.

In diesem Projekt wird das Pico W als Abonnent fungieren und den Songtitel unter dem entsprechenden Topic empfangen. Falls der Songtitel bereits im Code hinterlegt ist, wird Pico W den Summer den Song spielen lassen.

### 1. Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Ein vollständiges Kit zu kaufen ist definitiv praktisch, hier ist der Link:

Bezeichnung	ARTIKEL IM KIT	LINK
Kepler-Kit	450+	

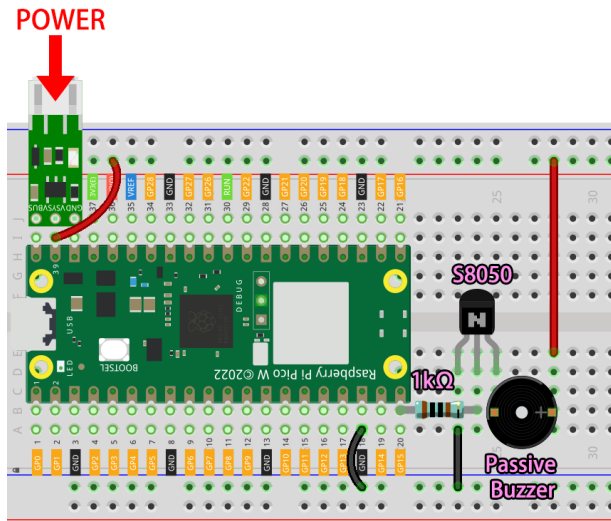
Alternativ können Sie die Teile auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1(S8050)	
6	<i>Widerstand</i>	1(1K)	
7	Passiver <i>Summer</i>	1	
8	<i>Li-Po-Lademodul</i>	1	
9	18650-Batterie	1	
10	Batteriehalter	1	

### 2. Schaltkreis aufbauen

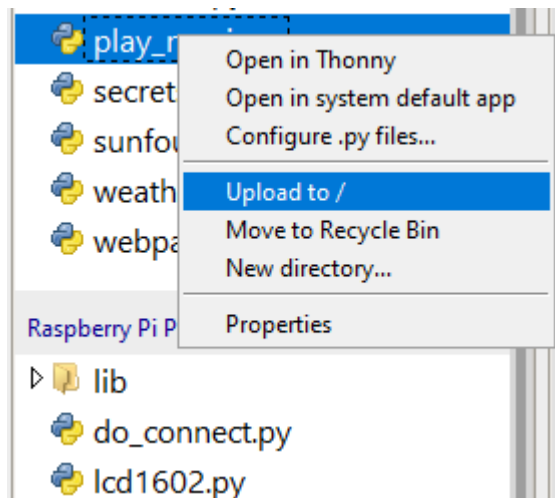
Im Kit sind zwei Summer enthalten, wir verwenden einen passiven Summer (einen mit freiliegender Leiterplatte auf der Rückseite). Für den Betrieb des Summers ist ein Transistor erforderlich, hier verwenden wir S8050.

**Warnung:** Stellen Sie sicher, dass Ihr Li-Po-Ladegerät wie im Diagramm gezeigt angeschlossen ist. Andernfalls könnte ein Kurzschluss sowohl Ihre Batterie als auch die Schaltung beschädigen.

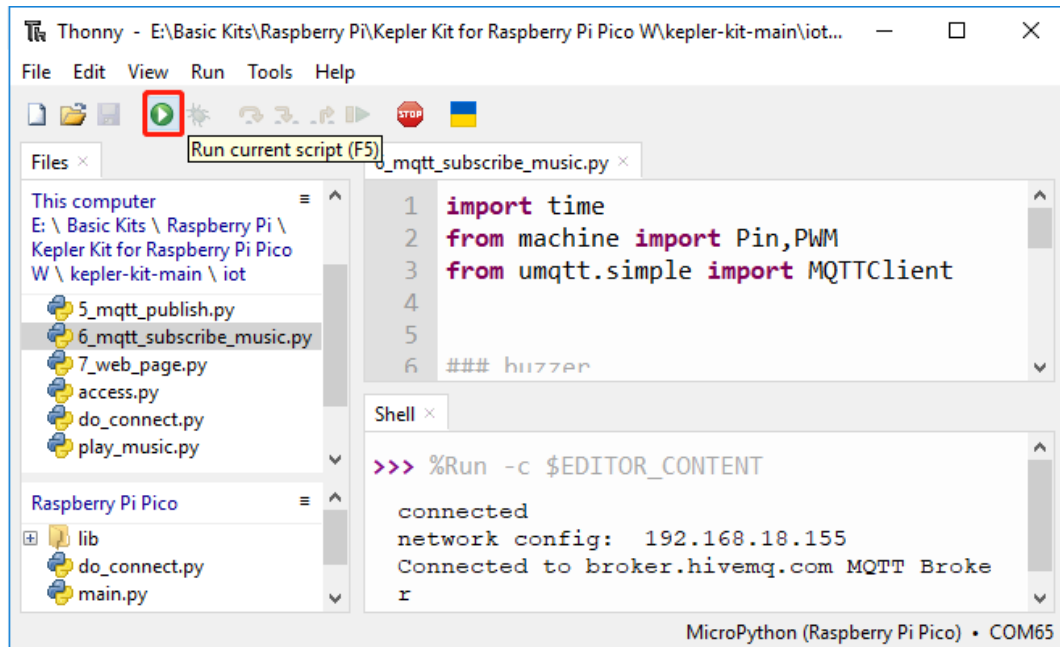


### 3. Code ausführen

1. Laden Sie die Datei `play_music.py` aus dem Pfad `kepler-kit-main/iot` auf das Raspberry Pi Pico W hoch.



2. Öffnen Sie die Datei `6_mqtt_subscribe_music.py` im Pfad `kepler-kit-main/iot` und klicken Sie auf die Schaltfläche **Aktuelles Skript ausführen** oder drücken Sie F5, um es auszuführen.



**Bemerkung:** Bevor Sie den Code ausführen, stellen Sie sicher, dass Sie die Skripte `do_connect.py` und `secrets.py` auf Ihrem Pico W haben. Wenn nicht, beziehen Sie sich bitte auf [1. Zugang zum Netzwerk](#), um sie zu erstellen.

- Öffnen Sie in Ihrem Browser, geben Sie das Topic als **SunFounder MQTT Music** ein und den Songtitel als **Nachricht**. Nach dem Klicken auf die **Veröffentlichen**-Schaltfläche wird der am Pico W angeschlossene Summer den entsprechenden Song abspielen.

**Bemerkung:** In `play_music.py` sind `nokia`, `starwars`, `nevergonnagiveyouup`, `gameofthrones`, `songofstorms`, `zeldatheme`, `harrypotter` enthalten.

## Publish

Topic

QoS

0

Retain

☐

Message

- Wenn Sie möchten, dass dieses Skript beim Hochfahren ausgeführt wird, können Sie es auf dem Raspberry Pi Pico W als `main.py` speichern.

### Wie funktioniert es?

Um es einfacher zu gestalten, haben wir den MQTT-Code vom Rest des Codes separiert. Daraus resultiert der folgende Code, der die grundlegendste Funktionalität der MQTT-Abonnements an drei Stellen implementiert.



```

import time
from umqtt.simple import MQTTClient

from do_connect import *
do_connect()

mqtt_server = 'broker.hivemq.com'
client_id = 'Jimmy'

# to subscribe the message
topic = b'SunFounder MQTT Music'

def callback(topic, message):
    print("New message on topic {}".format(topic.decode('utf-8')))
    message = message.decode('utf-8')
    print(message)

try:
    client = MQTTClient(client_id, mqtt_server, keepalive=60)
    client.set_callback(callback)
    client.connect()
    print('Connected to %s MQTT Broker'%(mqtt_server))
except OSError as e:
    print('Failed to connect to MQTT Broker. Reconnecting...')
    time.sleep(5)
    machine.reset()

while True:
    client.subscribe(topic)
    time.sleep(1)

```

Beim Verbindungsaufbau mit dem MQTT-Broker rufen wir die Funktion `client.set_callback(callback)` auf, die als Rückruffunktion für die empfangenen Abonnement-Nachrichten dient.

```

try:
    client = MQTTClient(client_id, mqtt_server, keepalive=60)
    client.set_callback(callback)
    client.connect()
    print('Connected to %s MQTT Broker'%(mqtt_server))
except OSError as e:
    print('Failed to connect to MQTT Broker. Reconnecting...')
    time.sleep(5)
    machine.reset()

```

Als nächstes kommt die Rückruffunktion, die die Nachricht aus dem abonnierten Thema ausgibt. MQTT ist ein binär-basiertes Protokoll, bei dem die Steuerelemente binäre Bytes und keine Textzeichenfolgen sind. Daher müssen diese Nachrichten mit `message.decode('utf-8')` dekodiert werden.

```

def callback(topic, message):
    print("New message on topic {}".format(topic.decode('utf-8')))
    message = message.decode('utf-8')
    print(message)

```

Verwenden Sie eine `While True`-Schleife, um regelmäßig Nachrichten unter diesem Thema zu erhalten.

```
while True:
    client.subscribe(topic)
    time.sleep(1)
```

Als nächstes wird Musik abgespielt. Diese Funktion befindet sich im Skript `play_music.py`, das aus drei Hauptteilen besteht.

- **Tone:** Simuliert einen spezifischen Ton basierend auf der Grundfrequenz, der zum Abspielen verwendet wird.

```
NOTE_B0 = 31
NOTE_C1 = 33
...
NOTE_DS8 = 4978
REST = 0
```

- **Score:** Bearbeiten Sie die Musik in ein für das Programm nutzbares Format. Diese Partituren stammen von Robson Coutos kostenloser Weitergabe, Sie können auch Ihre Lieblingsmusik im folgenden Format hinzufügen.

```
# notes of the melody followed by the duration.
# a 4 means a quarter note, 8 an eighth, 16 sixteenth, so on
# !!negative numbers are used to represent dotted notes,
# so -4 means a dotted quarter note, that is, a quarter plus an
↪eighth!!
song = {
    "nokia": [NOTE_E5, 8, NOTE_D5, 8, NOTE_FS4, 4, NOTE_GS4, 4, NOTE_
↪CS5, 8, NOTE_B4, 8, NOTE_D4, 4,
                NOTE_E4, 4, NOTE_B4, 8, NOTE_A4, 8, NOTE_CS4, 4, NOTE_
↪E4, 4, NOTE_A4, 2],
    "starwars": [,,,],
    "nevergonnagiveyouup": [,,,],
    "gameofthrone": [,,,],
    "songofstorms": [,,,],
    "zeldatheme": [,,,],
    "harrypotter": [,,,],
}
```

- **Play:** Dieser Teil ist im Grunde genommen das gleiche wie *3.2 Eigener Ton*, aber leicht optimiert, um zur obigen Partitur zu passen.

```
import time
import machine

# change this to make the song slower or faster
tempo = 220

# this calculates the duration of a whole note in ms
wholenote = (600000 * 4) / tempo

def tone(pin, frequency, duration):
    if frequency is 0:
        pass
    else:
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

        pin.freq(frequency)
        pin.duty_u16(30000)
        time.sleep_ms(duration)
        pin.duty_u16(0)

def noTone(pin):
    tone(pin,0,100)

def play(pin,melody):

    # iterate over the notes of the melody.
    # Remember, the array is twice the number of notes (notes + durations)
    for thisNote in range(0,len(melody),2):
        # calculates the duration of each note
        divider = melody[thisNote+1]
        if divider > 0:
            noteDuration = wholenote/divider
        elif divider < 0:
            noteDuration = wholenote/- (divider)
            noteDuration *= 1.5

        # we only play the note for 90% of the duration, leaving 10% as a pause
        tone(pin,melody[thisNote],int(noteDuration*0.9))

        # Wait for the specif duration before playing the next note.
        time.sleep_ms(int(noteDuration))

        # stop the waveform generation before the next note.
        noTone(pin)

```

Zurück zur Hauptfunktion und lassen Sie MQTT die Musikwiedergabe auslösen. In der Rückruffunktion überprüfen Sie, ob die gesendete Nachricht der Name eines enthaltenen Liedes ist. Falls ja, weisen Sie den Liednamen der Variable melody zu und setzen play\_flag auf True.

```

def callback(topic, message):
    print("New message on topic {}".format(topic.decode('utf-8')))
    message = message.decode('utf-8')
    print(message)
    if message in song.keys():
        global melody,play_flag
        melody = song[message]
        play_flag = True

```

In der Hauptschleife wird, wenn play\_flag auf True gesetzt ist, melody abgespielt.

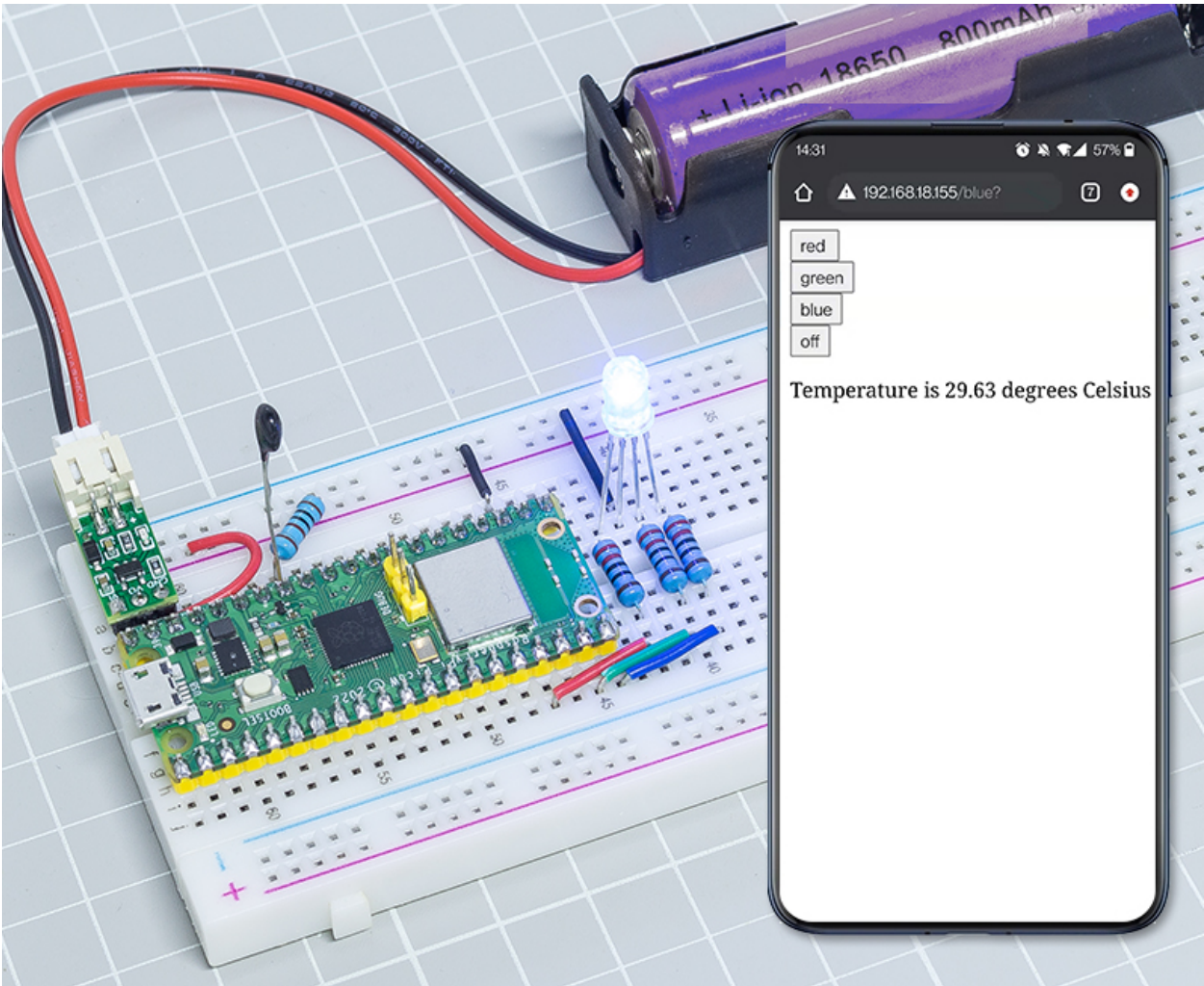
```

while True:
    client.subscribe(topic)
    time.sleep(1)
    if play_flag is True:
        play(buzzer,melody)
        play_flag = False

```

## 5.7 7. Ein Webserver einrichten

In diesem Artikel erfahren Sie, wie Sie den Pico W in einen Webserver verwandeln, über den Sie Schaltkreise steuern und Sensordaten direkt aus Ihrem Browser abrufen können.



### 1. Benötigte Komponenten

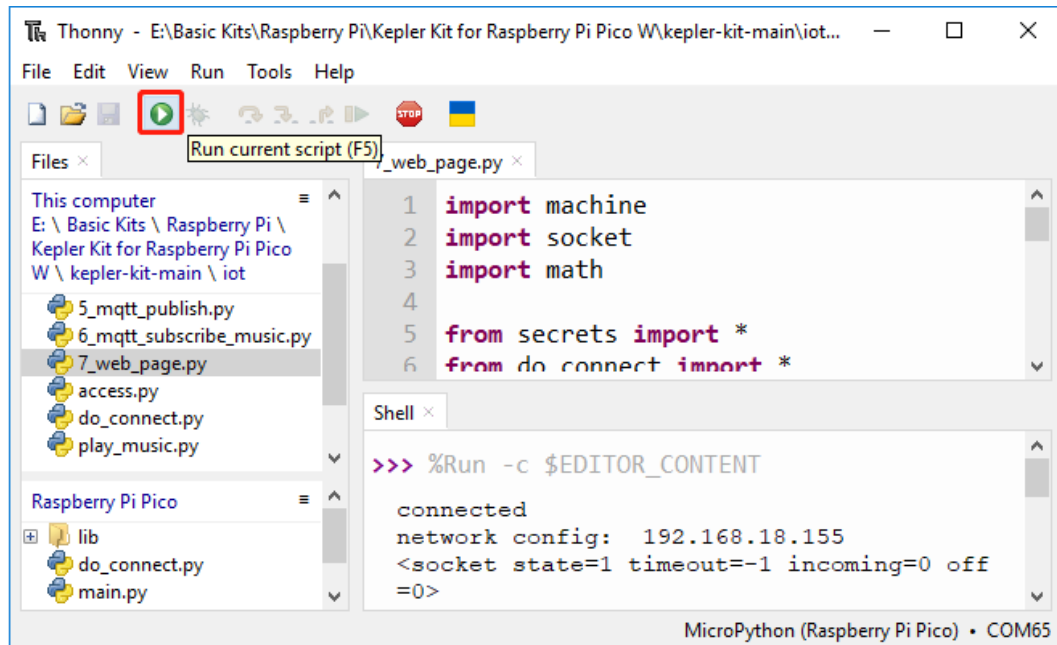
Für dieses Projekt benötigen Sie die folgenden Bauteile.

Ein komplettes Bausatz ist natürlich praktisch. Hier ist der Link dazu:

Bezeichnung	BESTANDTEILE IN DIESEM KIT	LINK
Kepler Kit	Über 450	

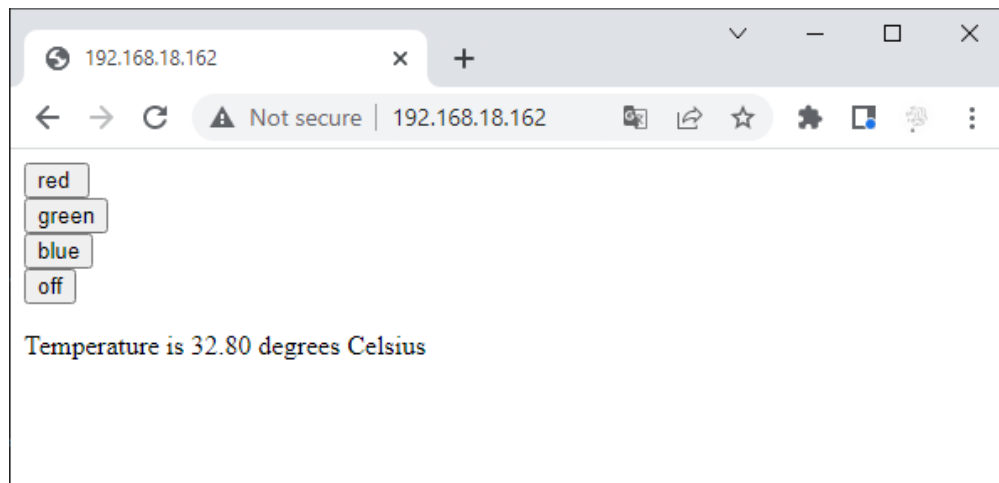
Die einzelnen Komponenten können Sie auch über die folgenden Links erwerben.

**Warnung:** Achten Sie darauf, dass Ihr Li-Po-Ladegerät genau wie im Schaltplan angeschlossen ist. Andernfalls könnte ein Kurzschluss Ihre Batterie und die Schaltung beschädigen.



**Bemerkung:** Bevor Sie den Code ausführen, stellen Sie sicher, dass die Skripte `do_connect.py` und `secrets.py` auf Ihrem Pico W vorhanden sind. Wenn nicht, folgen Sie den Anweisungen unter *1. Zugang zum Netzwerk*, um sie zu erstellen.

3. Geben Sie die IP-Adresse des Pico W in Ihren Browser ein, um die für dieses Projekt erstellte Webseite aufzurufen. Klicken Sie auf einen beliebigen Button, um die Farbe der RGB-LEDs zu ändern und die Temperatur sowie die Luftfeuchtigkeit zu aktualisieren.



4. Wenn Sie möchten, dass dieses Skript beim Start ausgeführt wird, speichern Sie es als `main.py` auf dem Raspberry Pi Pico W.

### Wie funktioniert es?

Die Webseite, die Sie besuchen, wird tatsächlich auf einem Server gehostet. Der Socket auf dem Server sendet uns die Webseite, sobald wir sie aufrufen. Ein Socket ist die Methode, mit der ein Server auf einen Client hören kann, der eine Verbindung herstellen möchte.

In diesem Projekt fungiert Pico W als Ihr Server, und Ihr Computer greift über einen Browser auf die auf Pico W

gehostete Webseite zu.

Zuerst erstellen wir einen Socket, der eine IP-Adresse und einen benötigt. Details zur Netzwerkverbindung und zur Ermittlung der IP-Adresse finden Sie unter [1. Zugang zum Netzwerk](#). Als Port verwenden wir 80. Nachdem der Socket eingerichtet ist, geben wir ihn zurück und verwenden ihn für den nächsten Schritt.

Socket-Bibliothek - Python Docs

```
import socket

def open_socket(ip):
    # Einen Socket öffnen
    address = (ip, 80)
    connection = socket.socket()
    connection.bind(address)
    connection.listen(1)
    print(connection)
    return connection
```

Anschließend richten Sie Ihren Webdienst ein, bei dem der zuvor eingerichtete Socket zum Einsatz kommt. Der folgende Code ermöglicht es Ihrem Pico W, Zugriffsanfragen von Ihrem Browser entgegenzunehmen.

```
def serve(connection):
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        client.close()
```

Als Nächstes benötigen Sie eine HTML-Seite, die Sie dem Besucher senden können. Dieses Beispiel speichert eine einfache HTML-Seite in Form von Zeichen in der Variablen `html`.

---

**Bemerkung:** Wenn Sie in der Lage sein möchten, Ihr eigenes HTML zu schreiben, können Sie Hilfe unter finden.

---

```
def webpage(value):
    html = f"""
        <!DOCTYPE html>
        <html>
        <body>
        <form action="./red">
        <input type="submit" value="Rot" />
        </form>
        <form action="./green">
        <input type="submit" value="Grün" />
        </form>
        <form action="./blue">
        <input type="submit" value="Blau" />
        </form>
        <form action="./off">
        <input type="submit" value="Aus" />
        </form>
        <p>Die Temperatur beträgt {value} Grad Celsius</p>
        </body>
        </html>
```

(Fortsetzung auf der nächsten Seite)



(Fortsetzung der vorherigen Seite)

```

        """
    return html

```

HTML-Seite an den Besucher senden.

```

def serve(connection):
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        html=webpage(0)
        client.send(html)
        client.close()

```

Die Seite kann über Ihren Browser aufgerufen werden, wenn Sie die oben genannten Teile kombinieren. Um die Wirkung zu sehen, führen Sie den unten stehenden Code mit Thonny aus.

```

import machine
import socket

from secrets import *
from do_connect import *

def webpage(value):
    html = f"""
        <!DOCTYPE html>
        <html>
        <body>
        <form action="/red">
        <input type="submit" value="red " />
        </form>
        <form action="/green">
        <input type="submit" value="green" />
        </form>
        <form action="/blue">
        <input type="submit" value="blue" />
        </form>
        <form action="/off">
        <input type="submit" value="off" />
        </form>
        <p>Temperature is {value} degrees Celsius</p>
        </body>
        </html>
        """
    return html

def open_socket(ip):
    # Open a socket
    address = (ip, 80)
    connection = socket.socket()
    connection.bind(address)
    connection.listen(1)
    print(connection)

```

(Fortsetzung auf der nächsten Seite)



(Fortsetzung der vorherigen Seite)

```

    return(connection)

def serve(connection):
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        html=webpage(0)
        client.send(html)
        client.close()

try:
    ip=do_connect()
    if ip is not None:
        connection=open_socket(ip)
        serve(connection)
except KeyboardInterrupt:
    machine.reset()

```

Wenn Sie den obigen Code ausführen, werden Sie feststellen, dass lediglich eine Webseite angezeigt wird; eine Steuerung der RGB-LEDs oder die Anzeige von Sensorwerten ist nicht möglich. Der Webdienst muss weiter verfeinert werden.

Zunächst müssen wir wissen, welche Informationen der Server erhält, wenn der Browser auf die Webseite zugreift. Ändern Sie deshalb die Funktion `serve()` geringfügig, um `request` auszugeben.

```

def serve(connection):
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        print(request)
        html=webpage(0)
        client.send(html)
        client.close()

```

Führen Sie das Skript erneut aus, und die Shell wird die folgende Nachricht ausgeben, wenn wir eine Taste auf der Webseite drücken.

```

b'GET /red? HTTP/1.1\r\nHost: 192.168.18.162\r\nConnection: keep-alive.....q=0.5\r\n\r\n'
↪n'
b'GET /favicon.ico HTTP/1.1\r\nHost: 192.168.18.162\r\nConnection: keep-alive.....q=0.
↪5\r\n\r\n'
b'GET /blue? HTTP/1.1\r\nHost: 192.168.18.162\r\nConnection: keep-alive.....q=0.5\r\n\r\n'
↪r'n'
b'GET /favicon.ico HTTP/1.1\r\nHost: 192.168.18.162\r\nConnection: keep-alive.....q=0.
↪5\r\n\r\n'

```

Das ist zu viel zum Lesen!

Was wir wirklich benötigen, ist jedoch nur der kleine Informationsbrocken vor `/red?`, `/blue?`. Er zeigt uns an, welcher Knopf gedrückt wurde. Deshalb haben wir `serve()` leicht modifiziert, um die Tasteninformation zu extrahieren.

```

def serve(connection):
    while True:

```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
client = connection.accept()[0]
request = client.recv(1024)
request = str(request)
try:
    request = request.split()[1]
except IndexError:
    pass
print(request)
html=webpage(0)
client.send(html)
client.close()
```

Führen Sie das Programm erneut aus, und die Shell wird die folgende Nachricht ausgeben, wenn wir eine Taste auf der Webseite drücken.

```
/red?
/favicon.ico
/blue?
/favicon.ico
/off?
/favicon.ico
```

Anschließend müssen wir nur noch die Farbe der RGB-LED entsprechend dem Wert von request ändern.

```
def serve(connection):
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        try:
            request = request.split()[1]
        except IndexError:
            pass

        print(request)

        if request == '/off?':
            red.low()
            green.low()
            blue.low()
        elif request == '/red?':
            red.high()
            green.low()
            blue.low()
        elif request == '/green?':
            red.low()
            green.high()
            blue.low()
        elif request == '/blue?':
            red.low()
            green.low()
            blue.high()
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
html=webpage(0)
client.send(html)
client.close()
```

Zuletzt soll der Thermistorwert auf der Webseite angezeigt werden (siehe [2.13 Thermometer](#) für Details zur Verwendung des Thermistors). Dies wird tatsächlich durch Ändern des Texts im HTML erreicht. Wir setzen die Parameter in der Funktion `webpage(value)` und ändern einfach die eingehenden Parameter, um die auf der Webseite angezeigte Zahl zu ändern.

```
def serve(connection):
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        try:
            request = request.split()[1]
        except IndexError:
            pass

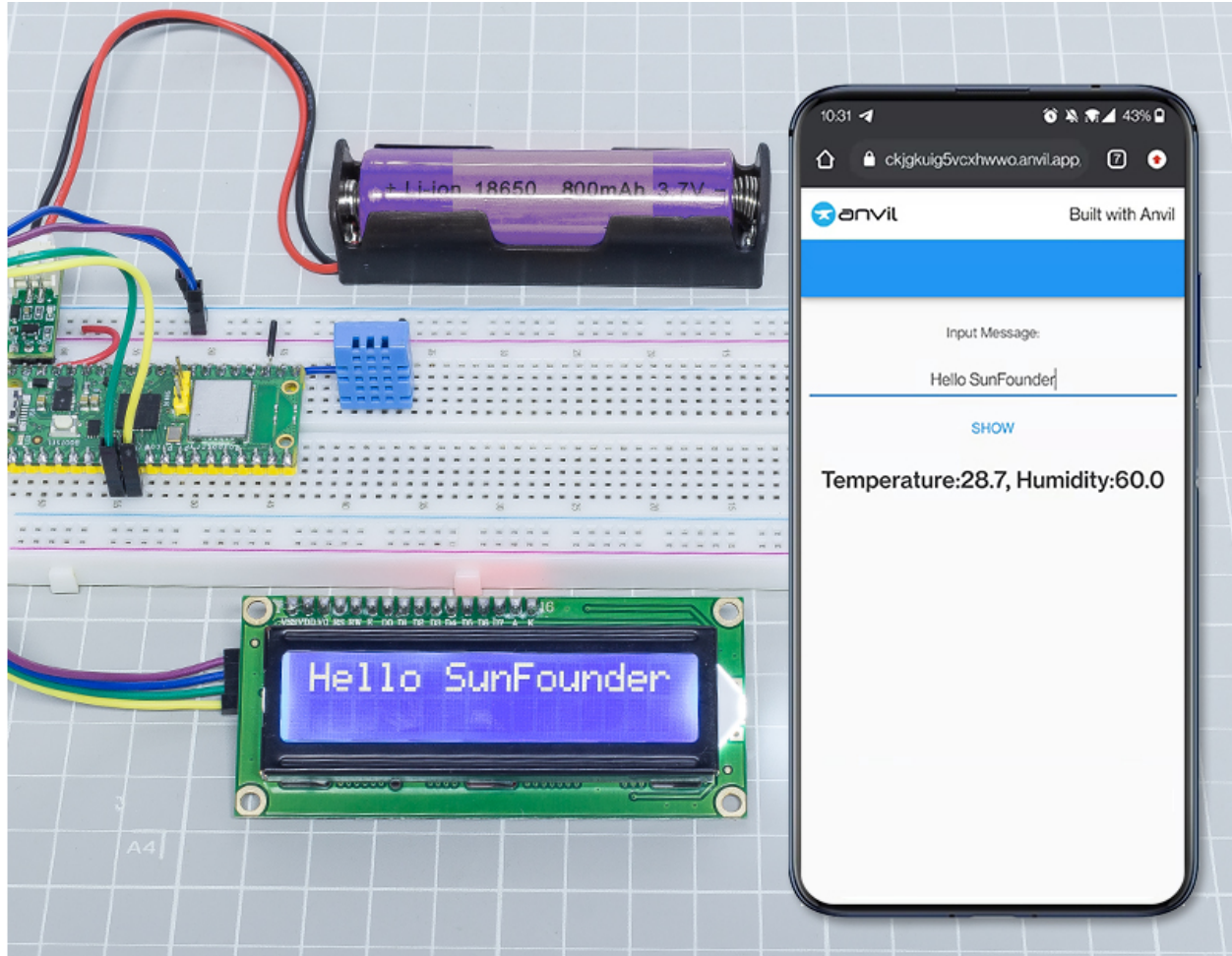
        #print(request)

        if request == '/off?':
            red.low()
            green.low()
            blue.low()
        elif request == '/red?':
            red.high()
            green.low()
            blue.low()
        elif request == '/green?':
            red.low()
            green.high()
            blue.low()
        elif request == '/blue?':
            red.low()
            green.low()
            blue.high()

        value='%.2f'%temperature()
        html=webpage(value)
        client.send(html)
        client.close()
```

## 5.8 8. Web-App mit @Anvil erstellen

In diesem Projekt ermöglichen wir eine bidirektionale Kommunikation zwischen dem Raspberry Pi Pico W und den Anvil-Servern. Die vom Pico W gesendeten Temperatur- und Feuchtigkeitsdaten werden in Echtzeit in Anvil angezeigt. Darüber hinaus können Sie Nachrichten in Anvil eingeben, die auf dem I2C LCD1602 des Pico W dargestellt werden.



### 1. Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Bauteile.

Ein vollständiges Kit ist definitiv praktisch. Hier ist der entsprechende Link:

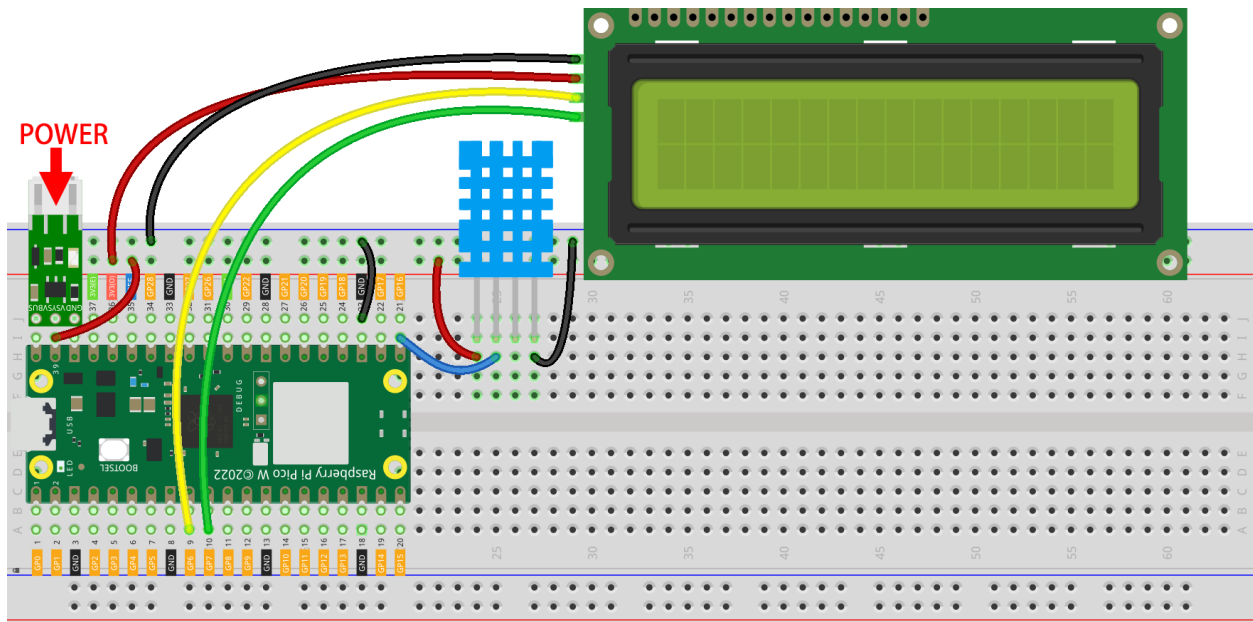
Bezeichnung	INHALT DES KITS	LINK
Kepler Kit	450+	

Sie können die Komponenten natürlich auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>I2C LCD1602</i>	1	
6	<i>DHT11 Temperatur- und Feuchtigkeitssensor</i>	1	
7	<i>Li-Po-Lademodul</i>	1	
8	18650 Batterie	1	
9	Batteriehalter	1	

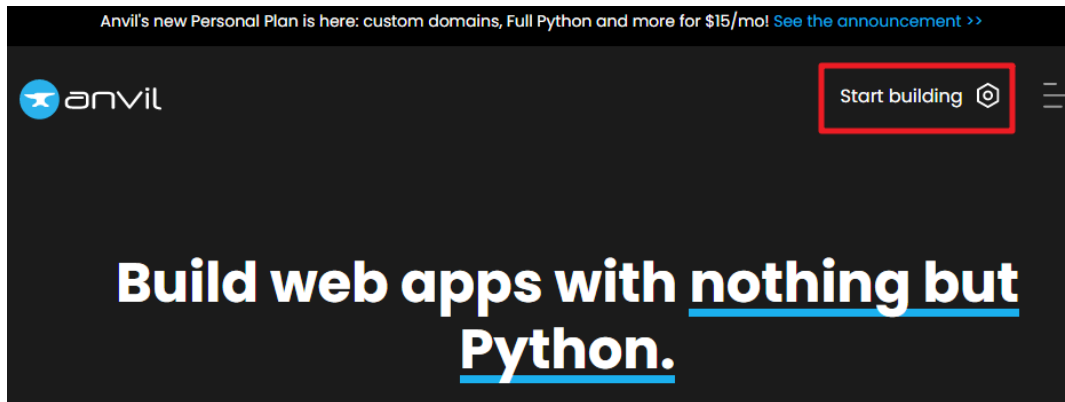
## 2. Schaltung aufbauen

**Warnung:** Achten Sie darauf, dass Ihr Li-Po-Lademodul gemäß dem Schaltplan angeschlossen ist. Andernfalls besteht die Gefahr eines Kurzschlusses, der Ihre Batterie und die Schaltung beschädigen könnte.



## 3. Eine Anvil-App erstellen

1. Besuchen Sie und klicken Sie auf **Jetzt entwickeln**.



2. Melden Sie sich an oder registrieren Sie sich.

**Sign In**

**New user?**

Sign up for free -->

---

**Returning user?**


Email address

Password

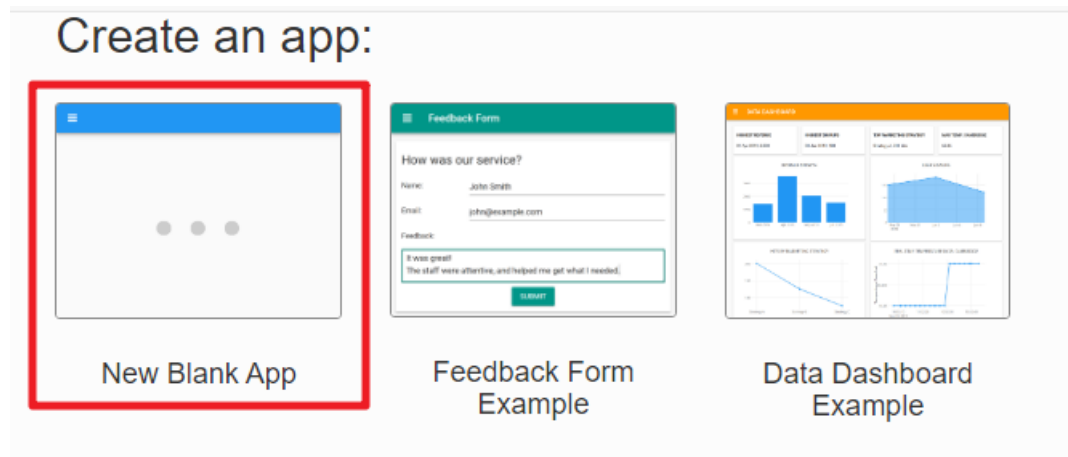
[Forgot password?](#) [Sign in -->](#)

---

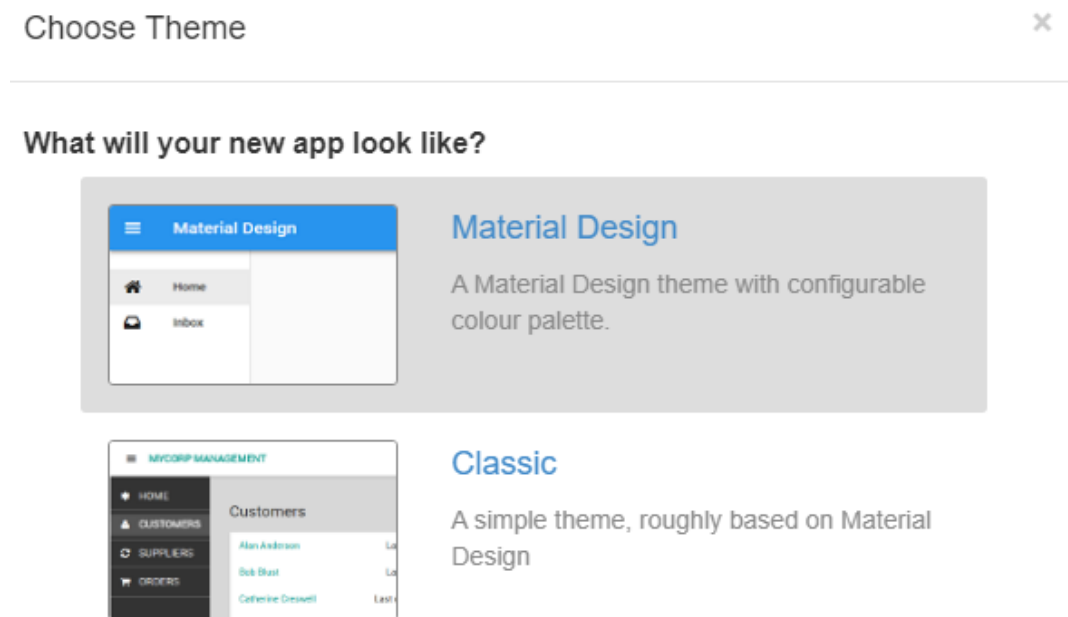
Sign in another way

 Sign in with Google

3. Erstellen Sie eine **Neue leere App**.

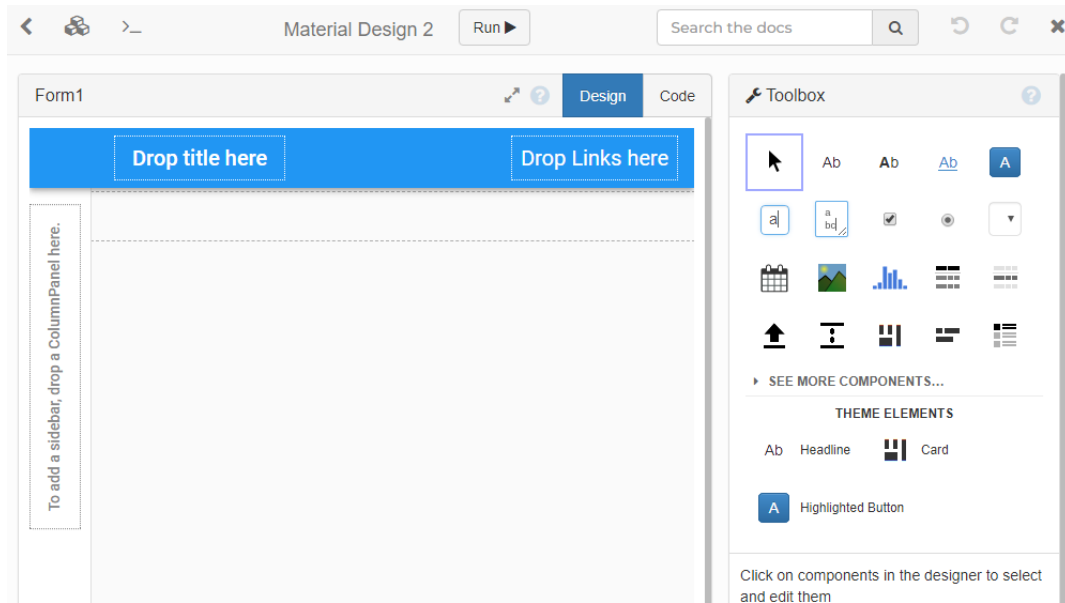


4. Wählen Sie das **Material Design Theme** aus.

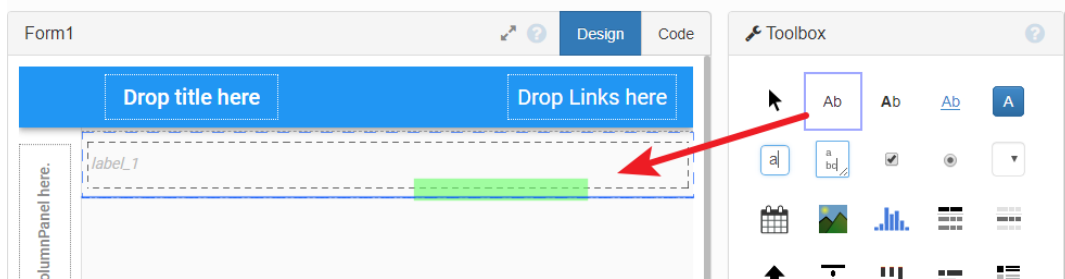


5. Nun befinden Sie sich auf der App-Bearbeitungsseite.

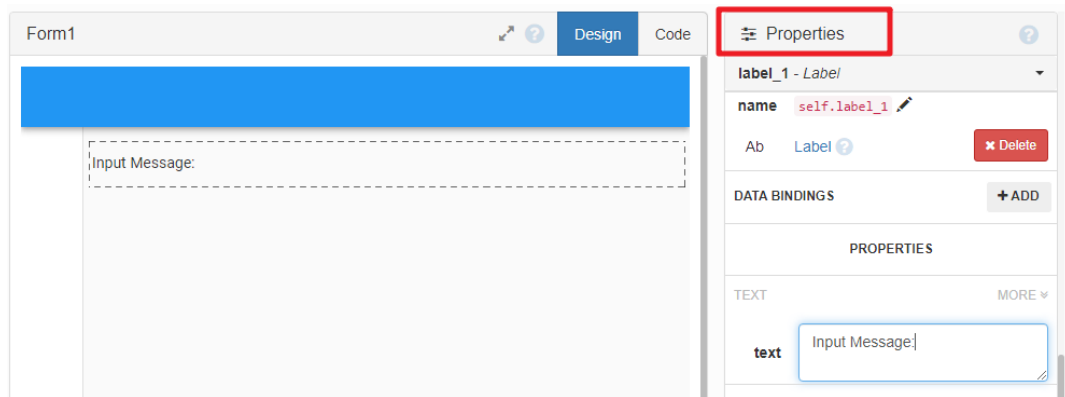




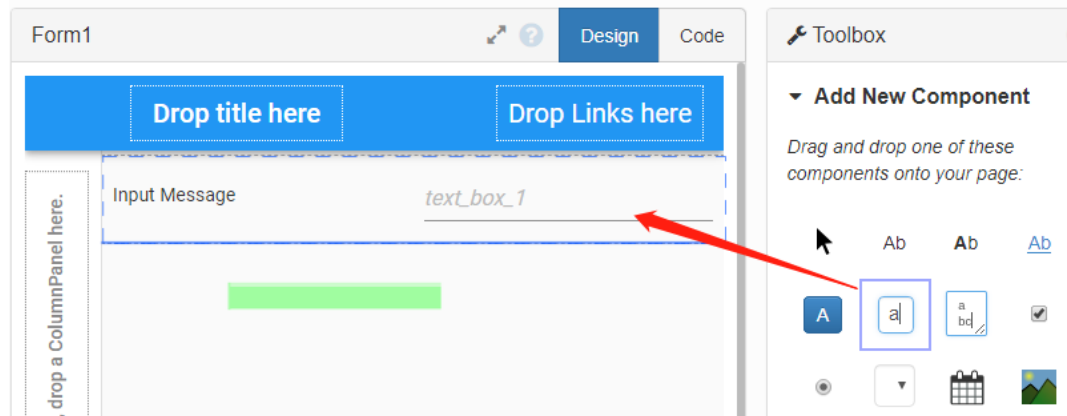
6. Ziehen Sie ein **Label**-Werkzeug aus der Toolbox und platzieren Sie es auf **Titel hier ablegen**.



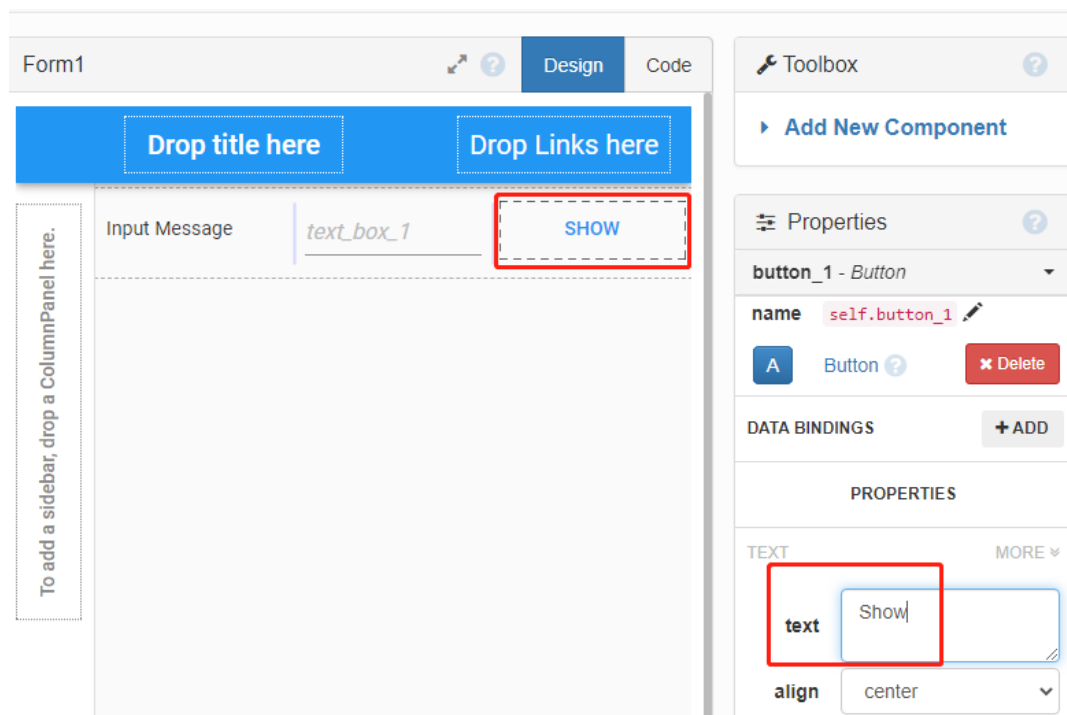
7. Den Label-Text können Sie im **Text**-Feld unter dem Menüpunkt **Eigenschaften** eingeben.



8. Ebenso ziehen Sie ein **Textfeld** nach rechts.



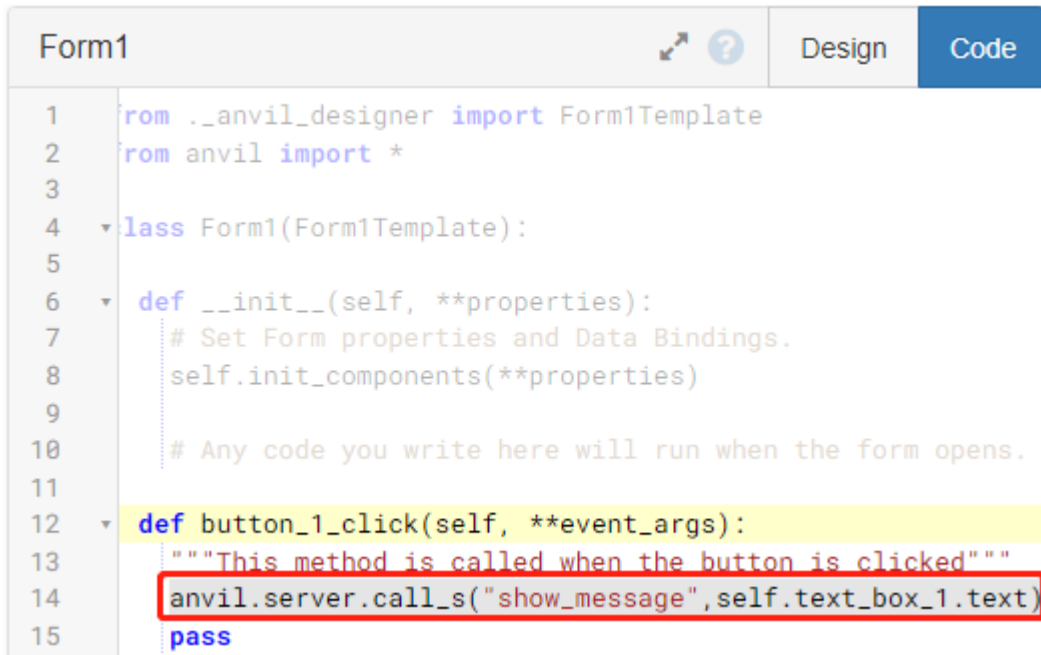
9. Ziehen Sie einen **Button** ganz nach rechts und passen Sie das **Text**-Feld an. Dieser Button wird zum „Senden“ einer Nachricht an den Raspberry Pi Pico W verwendet.



10. Nach einem Doppelklick auf den **SHOW**-Button wechselt das Formular von der Design-Seite zur Code-Seite und hebt den Code für den **Button** hervor. Fügen Sie den folgenden Code ein, um eine Funktion auf dem Server (in diesem Fall Pico W) aufzurufen.

```
anvil.server.call_s("show_message",self.text_box_1.text)
```

- `show_message` ist die Funktion, die programmiert wird, wenn der Pico W programmiert wird.
- `self.text_box_1.text` ist die Nachricht, die Sie im Textfeld eingeben und die an `show_message()` weitergeleitet wird.

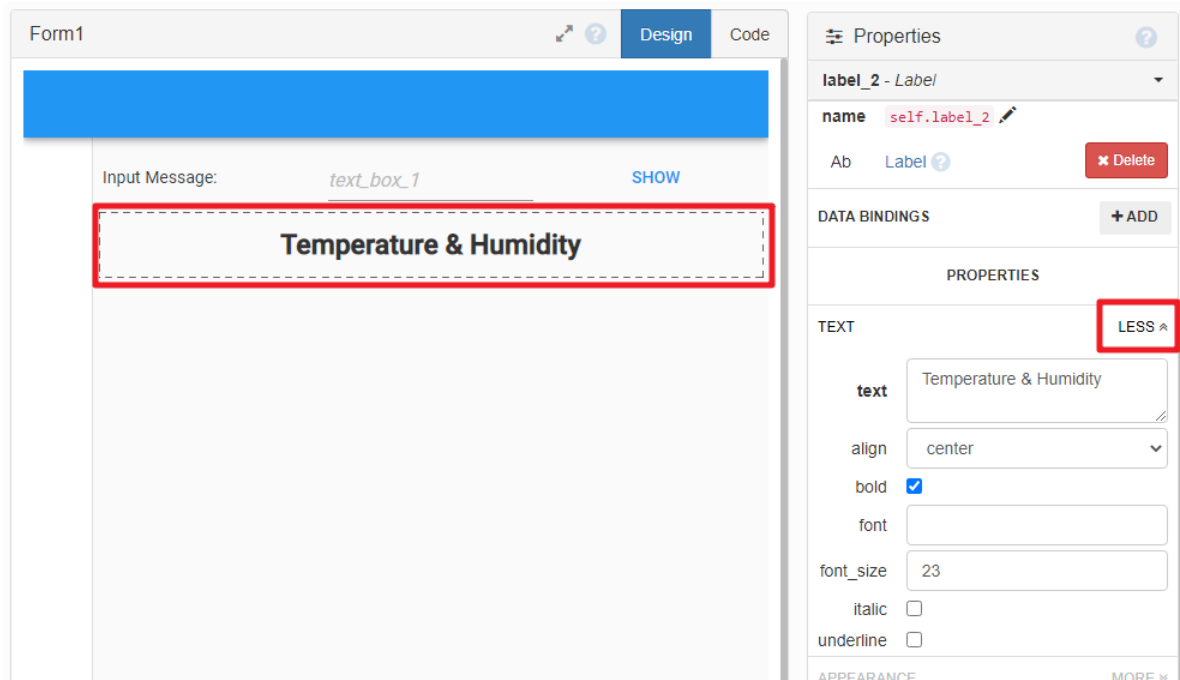


```

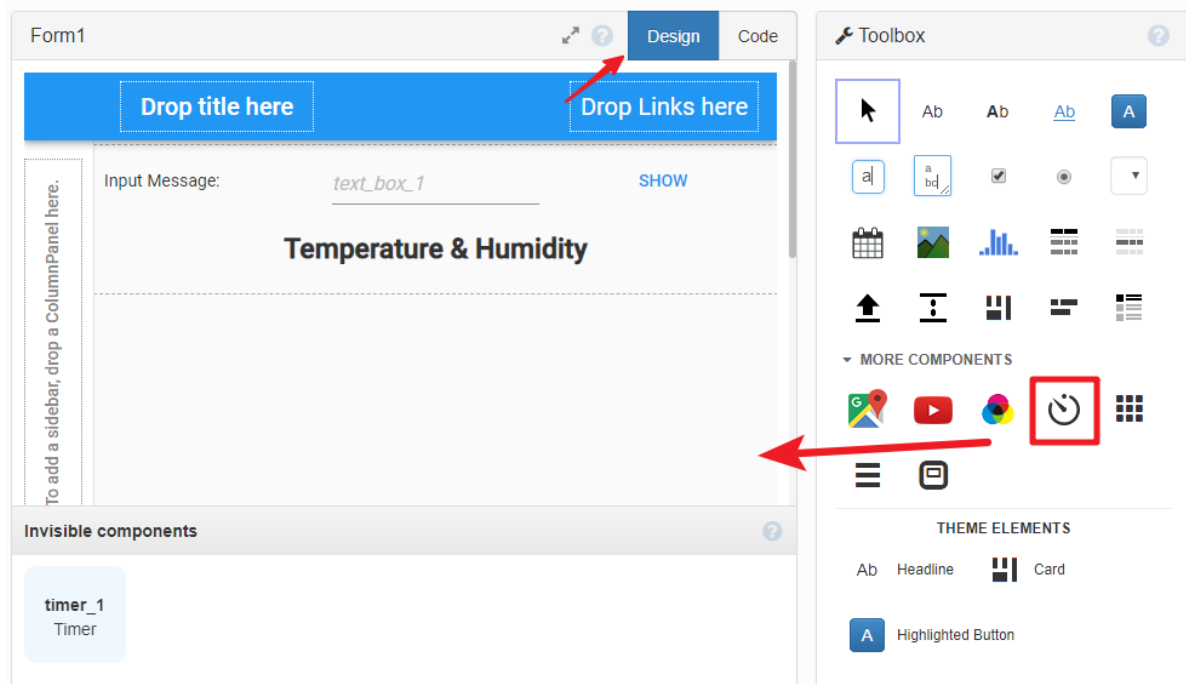
1 from ._anvil_designer import Form1Template
2 from anvil import *
3
4 class Form1(Form1Template):
5
6     def __init__(self, **properties):
7         # Set Form properties and Data Bindings.
8         self.init_components(**properties)
9
10        # Any code you write here will run when the form opens.
11
12    def button_1_click(self, **event_args):
13        """This method is called when the button is clicked"""
14        anvil.server.call_s("show_message", self.text_box_1.text)
15    pass

```

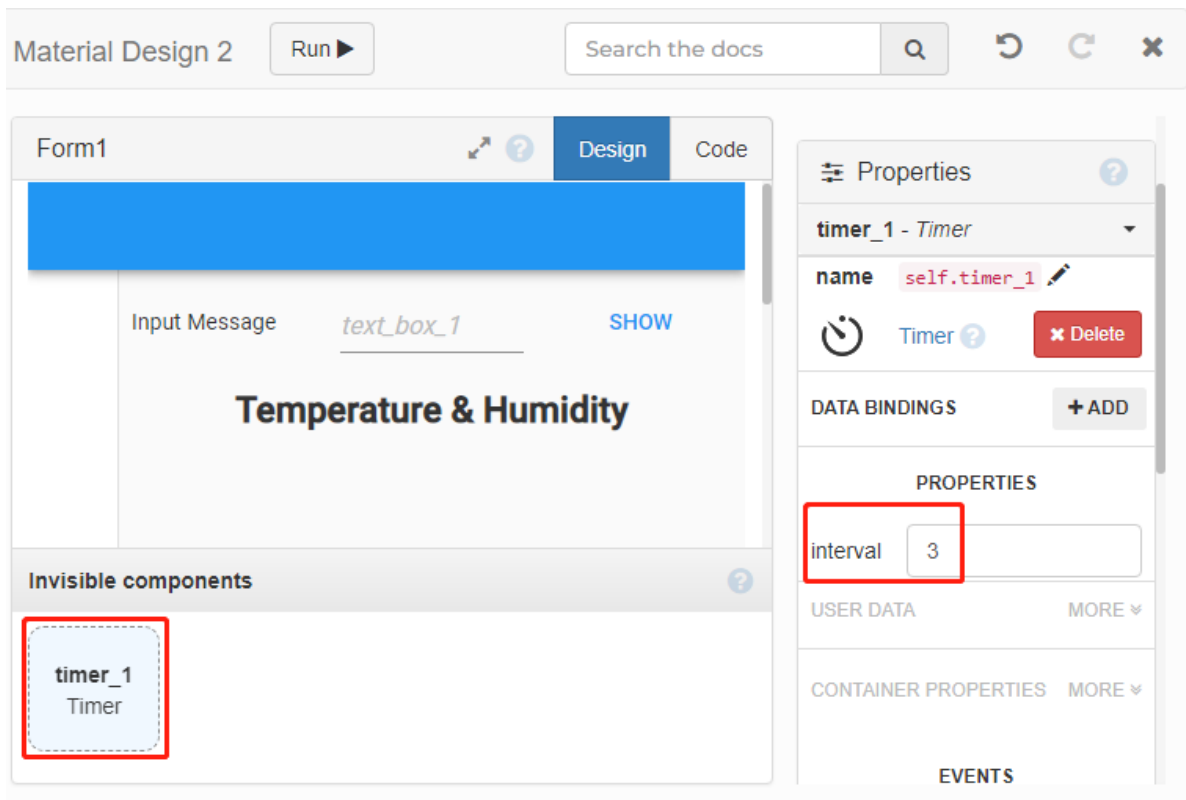
11. Wechseln Sie zurück zur Design-Seite, ziehen Sie ein weiteres Label und platzieren Sie es unter den vorherigen Elementen. Dieses Label wird die DHT11-Sensordaten vom Pico W anzeigen.



12. Klicken Sie in der **Toolbox** auf **Weitere Komponenten** und ziehen Sie den **Timer** in das Formular.

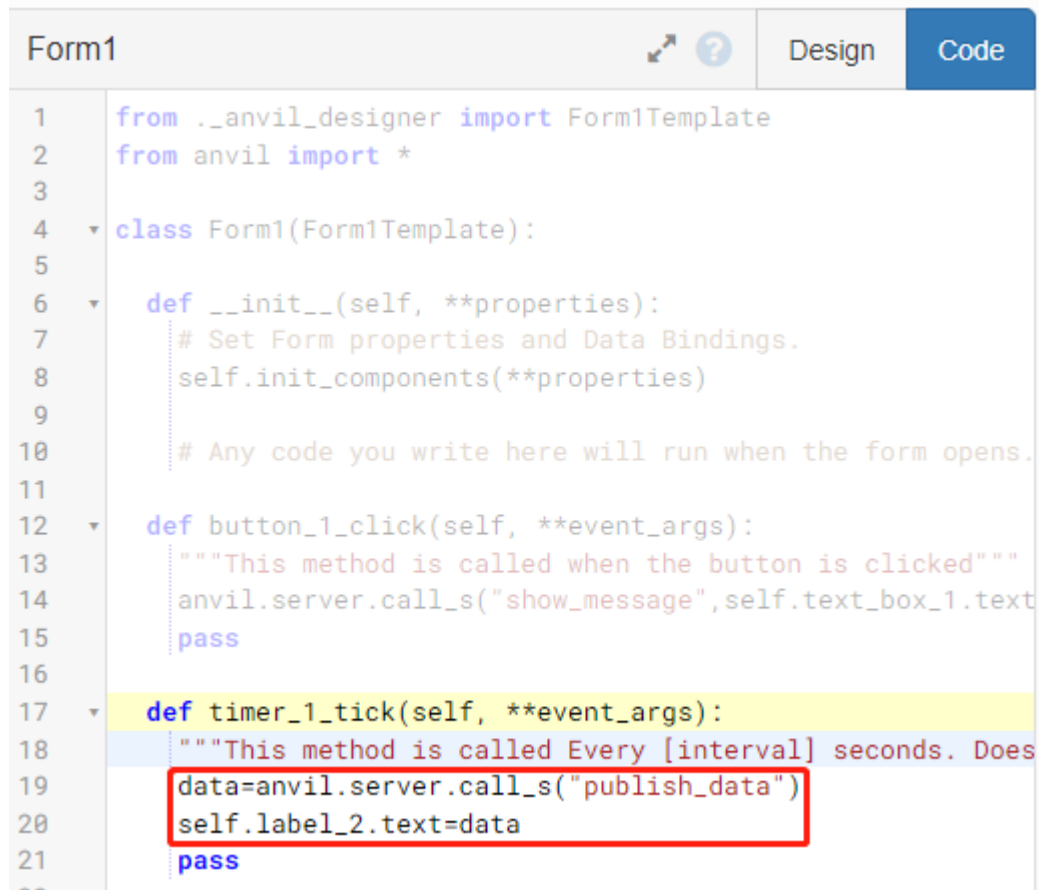


13. Setzen Sie den Timer im Menü **Eigenschaften** auf ein Intervall von 3 Sekunden. Diese Zeit wird verwendet, um den Bildschirm für unsere Sensordaten zu aktualisieren.



14. Doppelklicken Sie auf das **Timer**-Werkzeug, um es zu programmieren. Verwenden Sie die Funktion `anvil.server.call_s()` um die Funktion `publish_data()` vom Server aufzurufen und die anzuzeigende Nachricht in der Anvil-App abzurufen. Weisen Sie diese der Variablen `self.label_2.text` zu und Sie sind fertig.

```
data=anvil.server.call_s("publish_data")
self.label_2.text=data
```



```
Form1
Design Code
1 from ._anvil_designer import Form1Template
2 from anvil import *
3
4 class Form1(Form1Template):
5
6     def __init__(self, **properties):
7         # Set Form properties and Data Bindings.
8         self.init_components(**properties)
9
10        # Any code you write here will run when the form opens.
11
12    def button_1_click(self, **event_args):
13        """This method is called when the button is clicked"""
14        anvil.server.call_s("show_message",self.text_box_1.text)
15        pass
16
17    def timer_1_tick(self, **event_args):
18        """This method is called Every [interval] seconds. Does
19        data=anvil.server.call_s("publish_data")
20        self.label_2.text=data
21        pass
```

15. Damit ist der Anvil-Programmteil abgeschlossen. Weitere Details zur Verwendung von Anvil finden Sie unter .

#### 4. Pico W einrichten

Um die Verbindung des Raspberry Pi Pico W zu den Anvil-Diensten zu vereinfachen, verwendet Anvil ein spezielles Firmware-Image. Die Firmware des Pico W ist in MicroPython geschrieben und erscheint als USB-Laufwerk mit zwei Dateien (boot.py und main.py). Vor dem Schreiben des Codes muss der Pico W mit der angepassten Firmware geflasht und mit unserem WLAN verbunden werden.

1. Laden Sie die spezielle Firmware für den Raspberry Pi Pico W herunter. Es wird empfohlen, die vollständige Version herunterzuladen.

## Anvil Pico W Firmware v0.1.2 Latest

This is the Anvil firmware for the Raspberry Pi Pico W. See <https://anvil.works/pico> for more information.

This release fixes a couple of memory leaks in `anvil.pico.call`, which were causing the Pico to run out of memory eventually.

**Which firmware file should I download?**

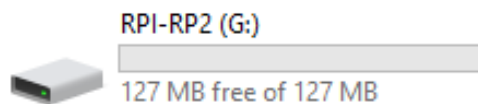
If you are starting from scratch, you should use the `pico-w-anvil-v0.1.2-complete.uf2` firmware. This will overwrite the entire flash memory, and includes template `boot.py` and `main.py` files on the USB mass storage filesystem.

If you already have files in the filesystem on your Pico W, you should download and flash the `pico-w-anvil-v0.1.2-firmware-only.uf2` firmware. This will **not** overwrite files created in the filesystem of previous Anvil firmware releases.

▼ Assets 4

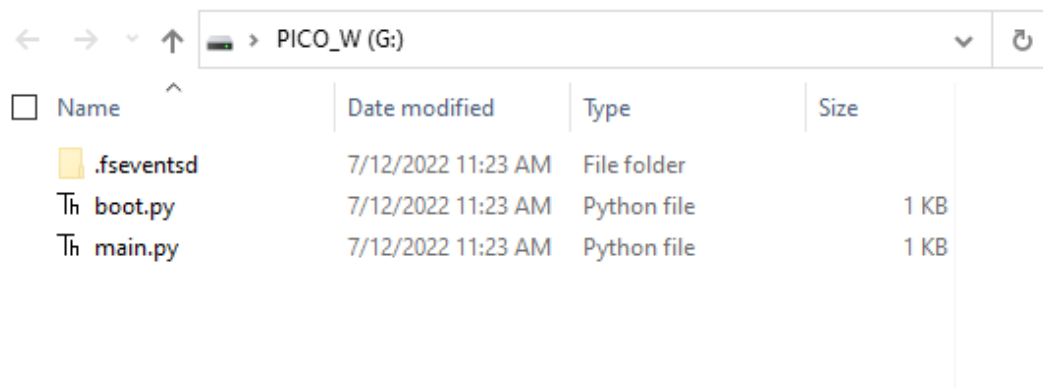
 <code>pico-w-anvil-v0.1.2-complete.uf2</code>	4 MB	Jul 12, 2022
 <code>pico-w-anvil-v0.1.2-firmware-only.uf2</code>	1.93 MB	Jul 12, 2022
 Source code (zip)		Jul 08, 2022
 Source code (tar.gz)		Jul 08, 2022

- Halten Sie die **BOOTSEL**-Taste am Pico W gedrückt und schließen Sie das Gerät über ein Micro-USB-Kabel an Ihren Computer an. Lassen Sie die BOOTSEL-Taste los, sobald das Laufwerk RPI-RP2 auf Ihrem Computer erscheint.

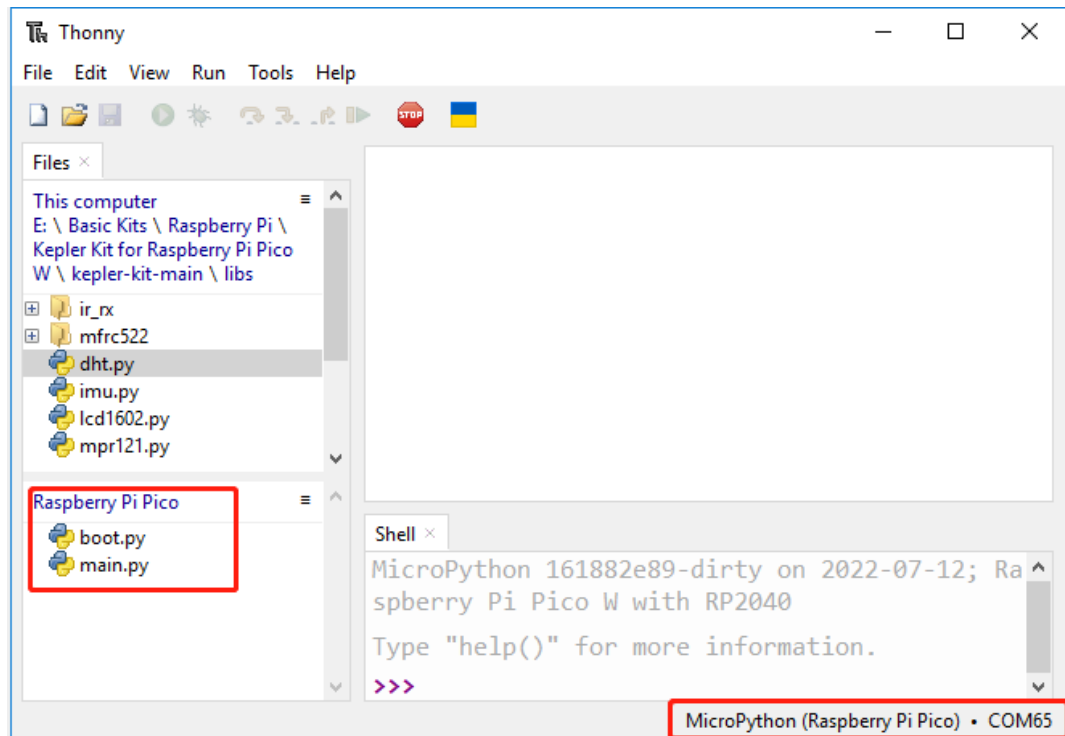


- Ziehen Sie die gerade heruntergeladene `.uf2`-Datei hinein. Der Pico W wird nun die Firmware installieren. Sobald der Vorgang abgeschlossen ist, wird das Laufwerk neu geladen und Sie sehen die Dateien `main.py` und `boot.py`.

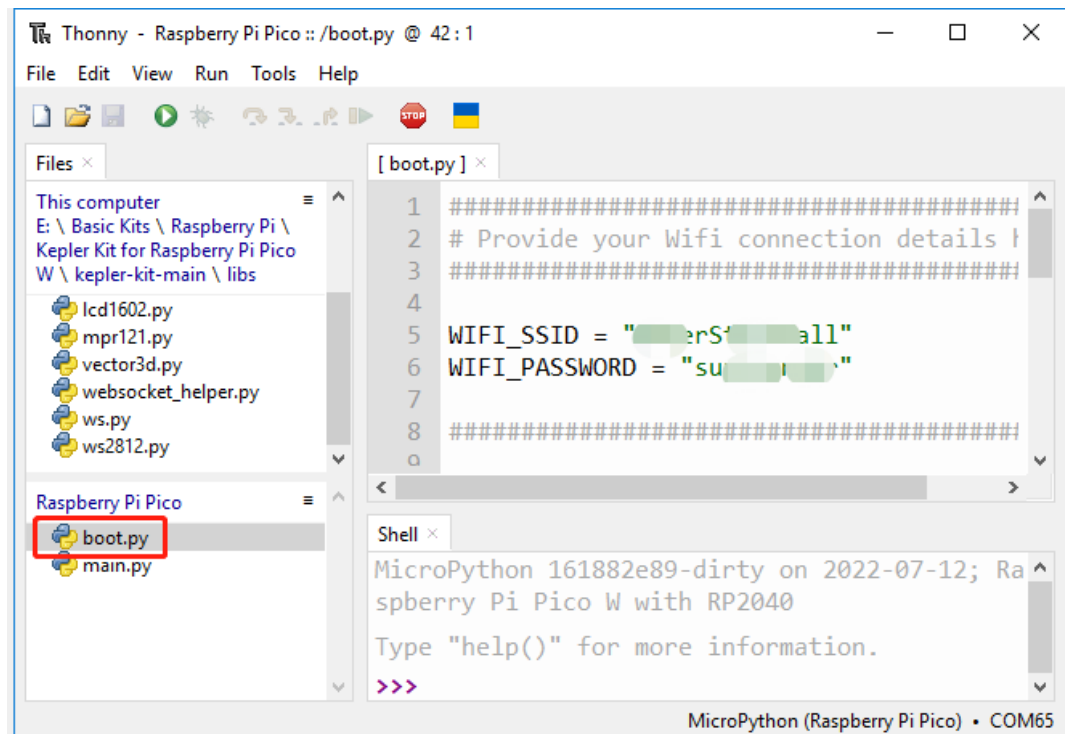
**Bemerkung:** Erstellen Sie vor dem erneuten Flashen der Firmware eine Sicherung aller wichtigen Dateien auf dem Pico W.



4. Wählen Sie in der Thonny IDE „MicroPython(Raspberry Pi Pico).COMXX“ als Interpreter aus. Nachdem Sie auf **Ansicht** -> **Dateien** geklickt haben, sehen Sie das lokale Laufwerk und das Laufwerk des Raspberry Pi Pico.



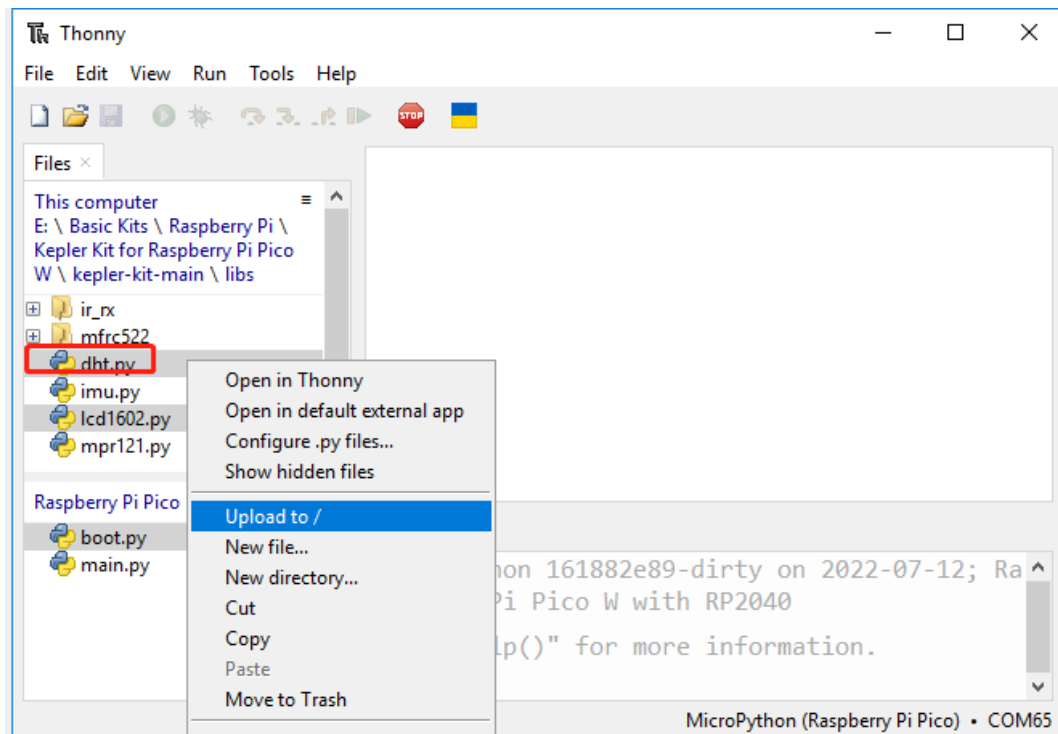
5. Doppelklicken Sie auf das `boot.py`-Skript und tragen Sie die SSID und das PASSWORT Ihres WLANs ein.



## 5. Code vervollständigen

1. Laden Sie `dht.py` und `lcd1602.py` aus dem Verzeichnis `kepler-kit-main/libs` auf den Raspberry Pi Pico

W hoch.



- Öffnen Sie `main.py` und ersetzen Sie den vorhandenen Code durch den folgenden Code.

```
import anvil.pico
import uasyncio as a
from machine import Pin, I2C

from lcd1602 import LCD
lcd = LCD()

from dht import DHT11
sensor = DHT11(Pin(16, Pin.OUT, Pin.PULL_DOWN))

UPLINK_KEY = "<uplink_key_goes_here>"

@anvil.pico.callable(is_async=True)
async def show_message(text):
    print(f"show anvil's input message: {text}")
    lcd.clear()
    lcd.message(text)
    return

@anvil.pico.callable(is_async=True)
async def publish_data():
    sensor.measure()
    return "Temperature: {}, Humidity: {}".format(sensor.temperature,
    sensor.humidity)

# Connect the Anvil Uplink. In MicroPython, this call will block forever.
```

(Fortsetzung auf der nächsten Seite)

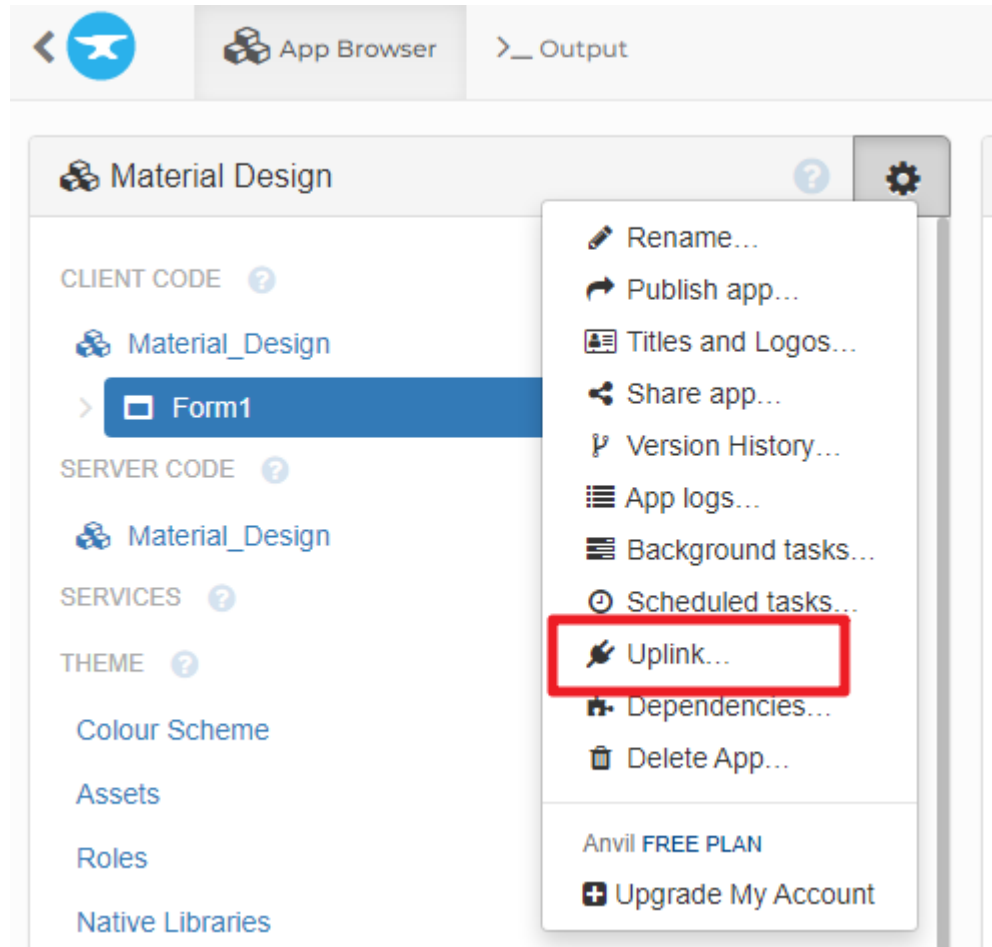


(Fortsetzung der vorherigen Seite)

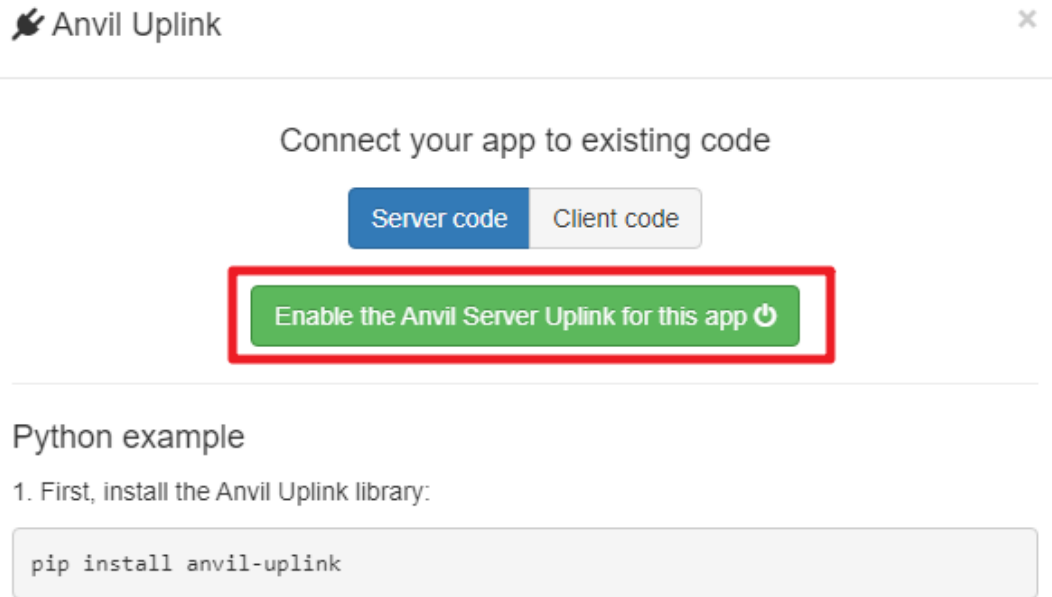
```
anvil.pico.connect(UPLINK_KEY)

# There's lots more you can do with Anvil on your Pico W.
#
# See https://anvil.works/pico for more information
```

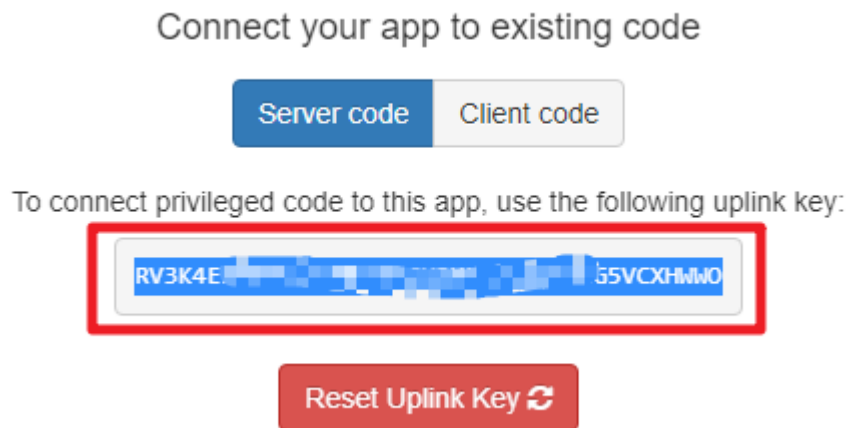
3. Kehren Sie zur Anvil-Oberfläche zurück und wählen Sie die Uplink-Option in den App-Browser-Einstellungen.



4. Klicken Sie auf **Den Anvil Server Uplink für diese App aktivieren**, um den Uplink-Schlüssel zu erhalten.

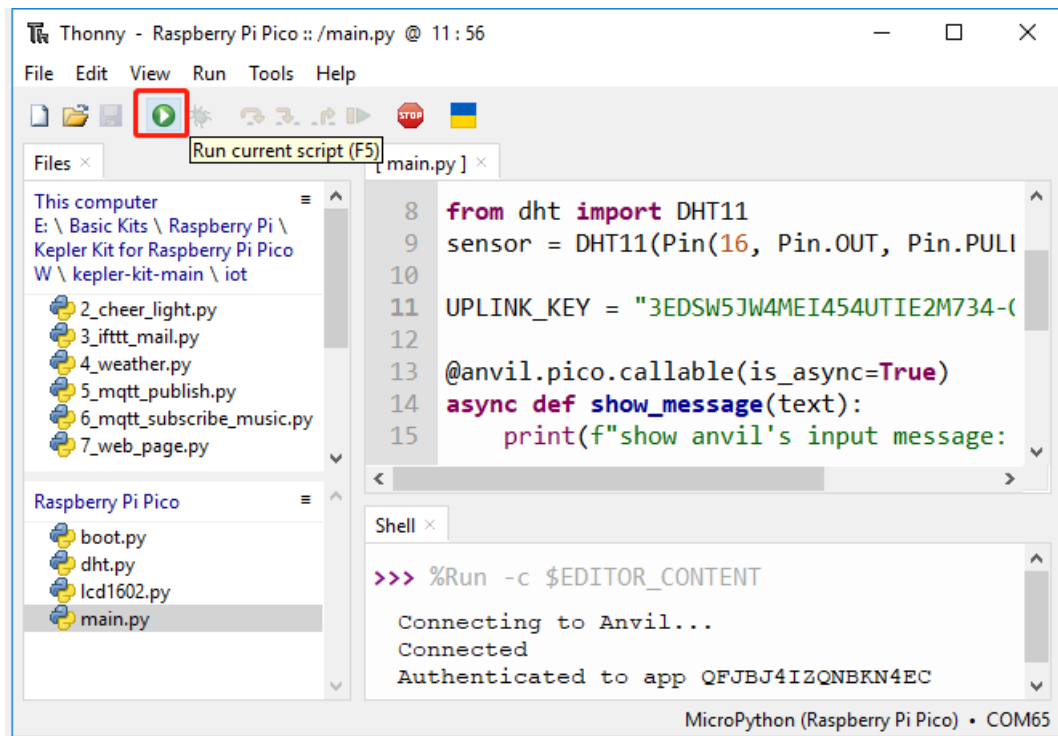


5. Kopieren Sie diesen und ersetzen Sie damit <uplink\_key\_goes\_here> in `main.py`, damit Ihr Pico W sich mit der von Ihnen erstellten Anvil-App verbinden kann.



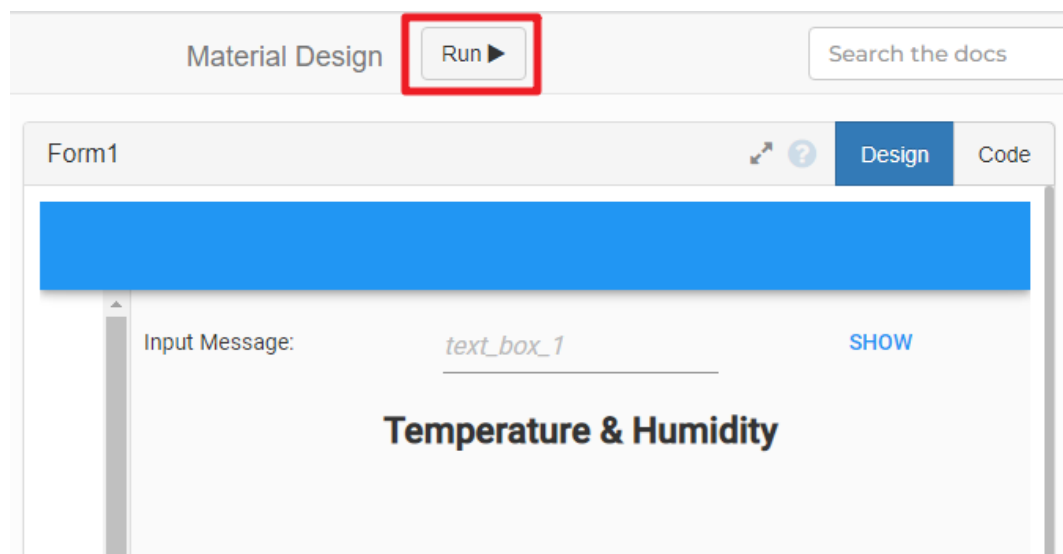
## 6. Das Projekt ausführen

1. Klicken Sie auf die Schaltfläche **Aktuelles Skript ausführen** oder drücken Sie F5. Nach erfolgreicher Verbindung sehen Sie in der Shell eine Meldung, die den erfolgreichen Verbindungsaufbau bestätigt.



2. Starten Sie Anvil. Nun können Sie die Temperatur und Luftfeuchtigkeit über die Anvil-App ablesen. Wenn Sie eine Nachricht in das Textfeld eingeben und dann auf die Schaltfläche **ANZEIGEN** klicken, wird die eingegebene Nachricht auf dem I2C LCD1602 angezeigt.

**Bemerkung:** Falls die eingegebenen Zeichen nicht auf dem I2C LCD1602 angezeigt werden, können Sie das Potentiometer auf der Rückseite des Moduls drehen, um den Kontrast zu erhöhen.

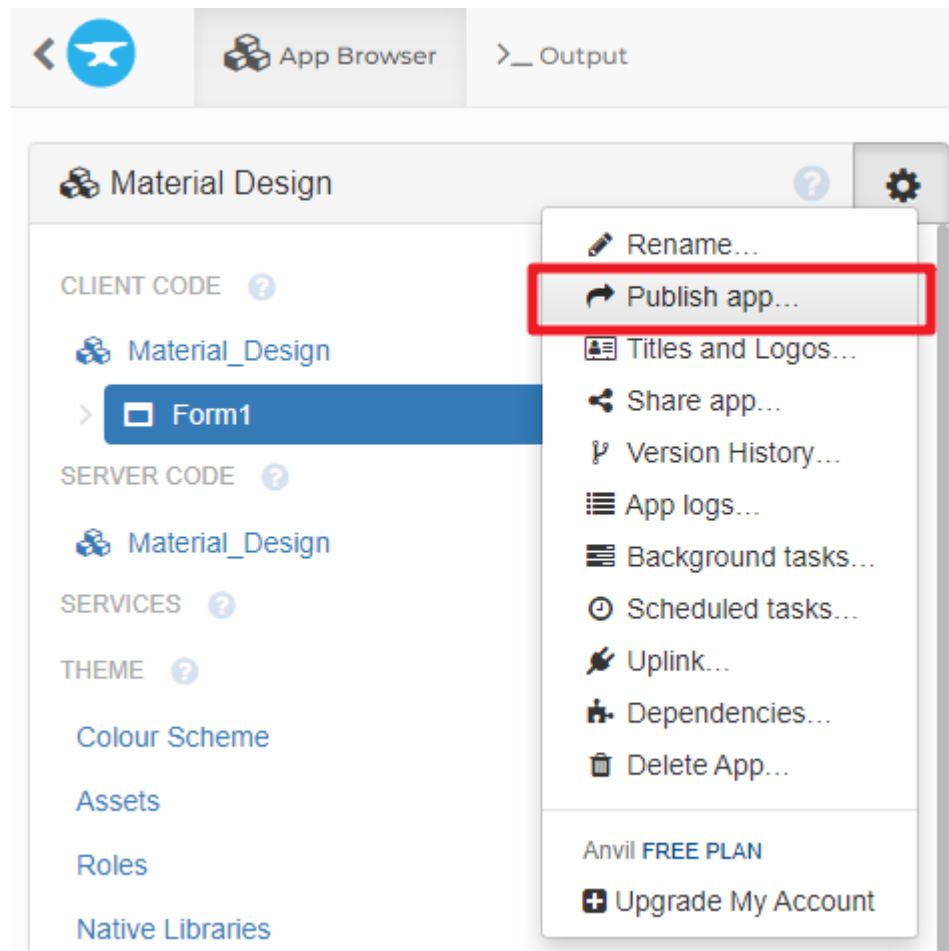


## 7. App veröffentlichen

Wenn Sie Ihre selbst erstellte App teilen möchten, können Sie einen Freigabelink wie folgt generieren.

1. Kehren Sie zur **Anvil**-Seite zurück und klicken Sie auf die Option **App veröffentlichen** in den **App-Browser**-

## Einstellungen.



2. Im Tab **Über privaten Link teilen** sehen Sie eine Liste von Links. Über diesen Link kann jeder auf Ihre App zugreifen.

➔ Publish app ✕

---

### Share via private link

Only those with a private link can use your app

Anyone with this private link can see your app:

<https://CKJGKUIG5VCXHWWO.anvil.app/CHZUMU32FS7N3GT7GA...>
COPY

Copy and paste it into an email, or share it on Facebook or Twitter:

f Share

---

To generate a new private link, click "Reset Link".

Links you have previously shared will no longer work.

Reset Link ↻

### Share via public link

Choose an alias or custom domain for your app

☐ Embed my app in a web page


---

➤ Allow someone to copy my app

---

Close

3. Greifen Sie auf den Link zu und Ihre App ist direkt einsatzbereit.

Built with  **anvil**
Build web apps for free with Anvil

Input Message:
 
SHOW

**Temperature: 28.7, Humidity: 60.0**

4. Sie können Ihre App auch über einen öffentlichen Link teilen. Geben Sie dazu Ihren personalisierten Domainnamen ein und klicken Sie unten auf **Anwenden**, um die Änderung wirksam zu machen.

Publish app

Share via private link
Only those with a private link can use your app

Share via public link
Choose an alias or custom domain for your app

Your app will be available at the following link:

https://

sunfounder-test

.anvil.app

This alias is available!

Apply your changes to view the link

Add custom domain

☐ Embed my app in a web page

> Allow someone to copy my app

Cancel

Apply

OK

### Wie funktioniert es?

Hier ist das Grundgerüst von `main.py`, welches die Basis für die Kommunikation zwischen Pico W und der Anvil-App bildet.

```
import anvil.pico
import uasyncio as a

UPLINK_KEY = "<uplink_key_goes_here>"

# Connect the Anvil Uplink. In MicroPython, this call will block forever.
anvil.pico.connect(UPLINK_KEY)

# There's lots more you can do with Anvil on your Pico W.
#
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
# See https://anvil.works/pico for more information
```

Konfigurieren Sie dht11 und lcd1602. Details zur Verwendung dieser beiden Komponenten finden Sie unter [6.2 Temperatur - Feuchtigkeit](#) und [3.4 Flüssigkristallanzeige](#).

```
from machine import Pin, I2C

from lcd1602 import LCD
lcd = LCD()

from dht import DHT11
sensor = DHT11(Pin(16, Pin.OUT, Pin.PULL_DOWN))
```

Im Anvil-Code haben wir zwei interne Funktionen des Servers (Pico W) aufgerufen.

Die erste ist `show_message()`, deren Aufgabe es ist, die von Anvil eingegebene Nachricht auf dem LCD anzuzeigen. Der Dekorator `@anvil.pico.callable(is_async=True)` macht diese Funktion für Anvil aufrufbar.

```
@anvil.pico.callable(is_async=True)
async def show_message(text):
    print(f"show anvil's input message: {text}")
    lcd.clear()
    lcd.message(text)
    return
```

Als Nächstes kommt `publish_data()`, die dazu dient, den Wert des DHT11 zu ermitteln und die Temperatur und Luftfeuchtigkeit an Anvil zurückzugeben. Auch hier wird der Dekorator `@anvil.pico.callable(is_async=True)` verwendet, um die Funktion für Anvil aufrufbar zu machen.

```
@anvil.pico.callable(is_async=True)
async def publish_data():
    sensor.measure()
    return "Temperature: {}, Humidity: {}".format(sensor.temperature, sensor.humidity)
```

## 5.9 9. Spiel mit dem @SunFounder Controller

In diesem Projekt lernen Sie, wie Sie ein Fernsteuerungsprojekt mit der Sunfounder Controller-App realisieren können. Innerhalb eines lokalen Netzwerks können Sie die Pico W-Schaltung mit Ihrem Smartphone oder Tablet steuern. Diese App ist besonders nützlich, wenn Sie einen einfachen Roboter mit Pico W bauen möchten.

Wir werden die Schiebeleiste in der App verwenden, um den Servowinkel zu steuern, und die Anzeige in der App, um die vom Ultraschallsensor erfasste Entfernung anzuzeigen.

### 1. Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

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

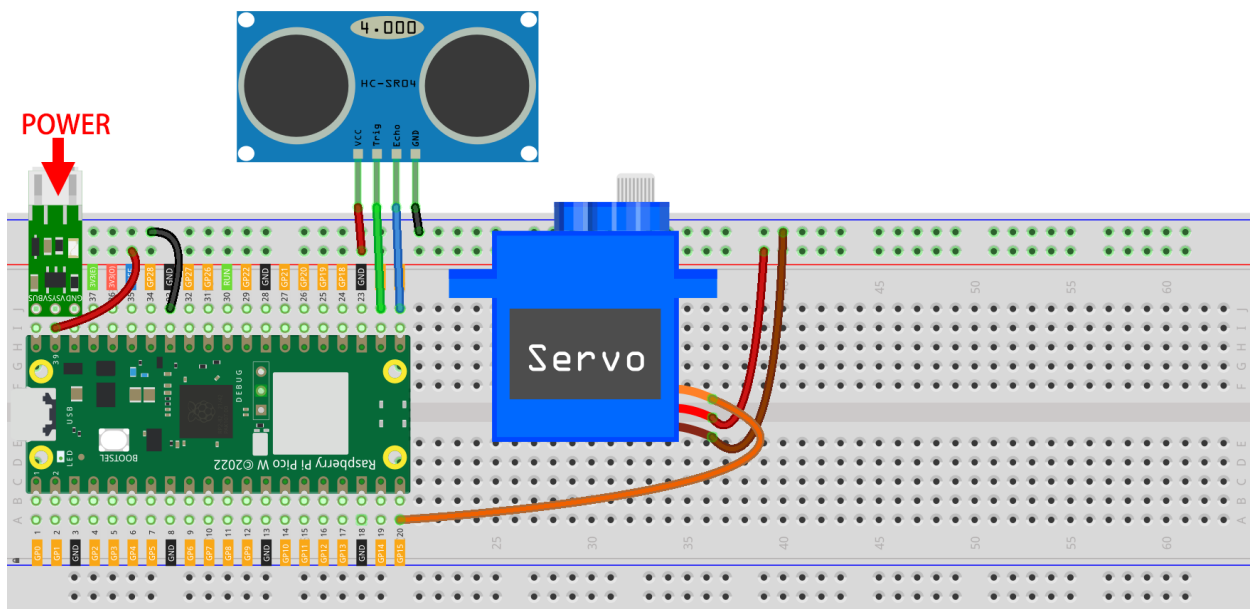
Name	ARTIKEL IM KIT	LINK
Kepler-Kit	450+	

Sie können die Teile auch einzeln über die folgenden Links kaufen.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Servo</i>	1	
6	<i>Ultraschallmodul</i>	1	
7	<i>Li-Po-Lademodul</i>	1	
8	18650 Akku	1	
9	Batteriehalter	1	

## 2. Schaltung aufbauen

**Warnung:** Stellen Sie sicher, dass Ihr Li-Po-Ladegerät gemäß dem Schaltplan angeschlossen ist. Andernfalls besteht die Gefahr eines Kurzschlusses, der Ihren Akku und die Schaltung beschädigen könnte.



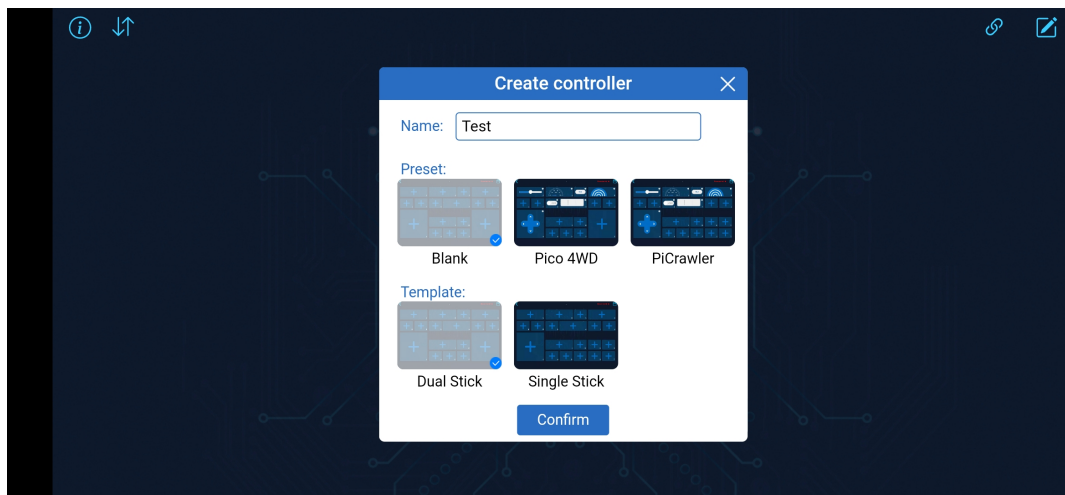
## 3. SunFounder Controller einrichten

1. Laden Sie die [SunFounder Controller APP](#) aus dem **APP Store(iOS)** oder **Google Play(Android)** herunter.
2. Öffnen Sie die App und klicken Sie auf die **+**-Schaltfläche auf der Startseite, um einen Controller zu erstellen.

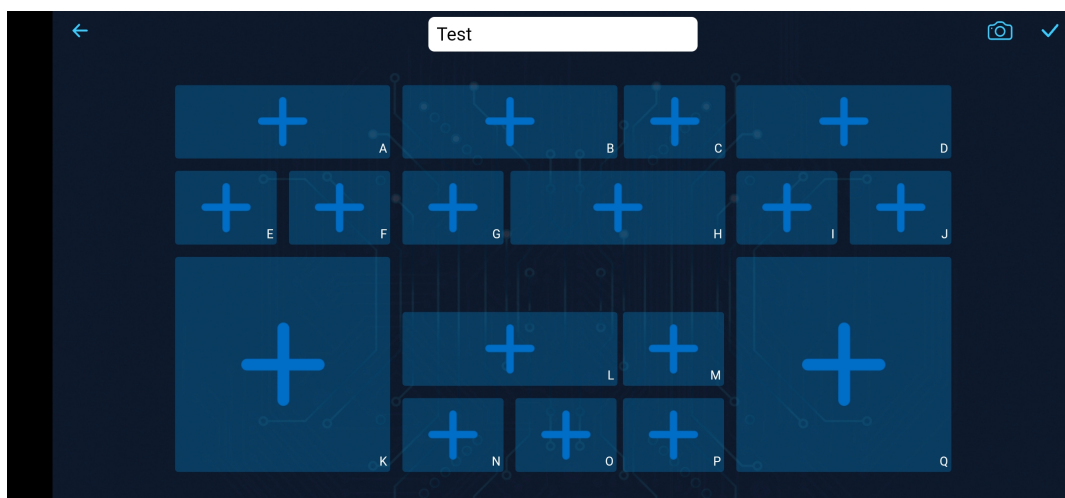




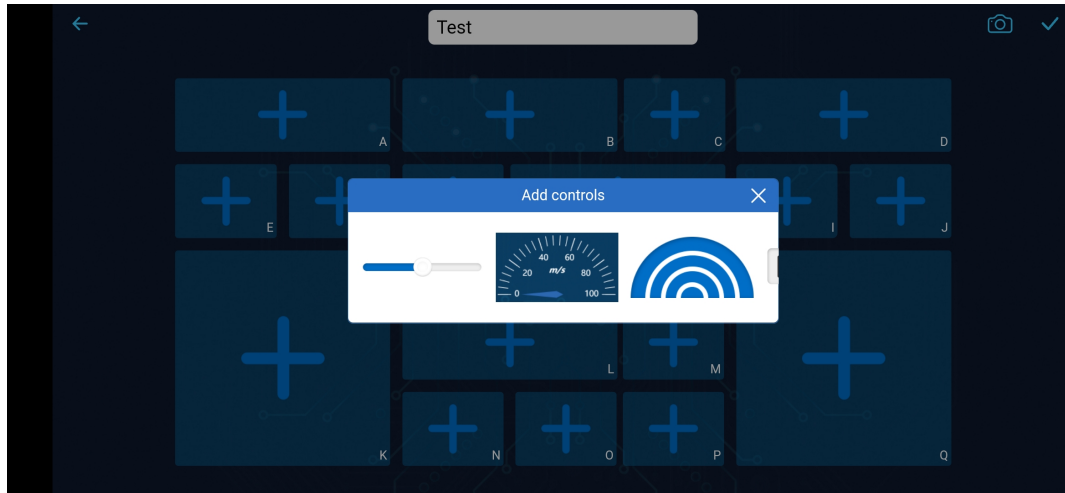
3. Hier wählen wir **Leer** und **Zwei-Stick-Steuerung** aus.



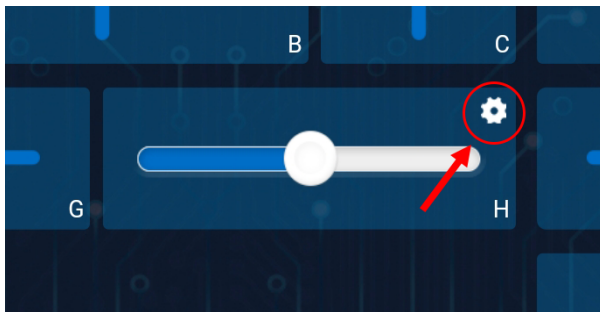
4. Nun erhalten wir einen leeren Controller.



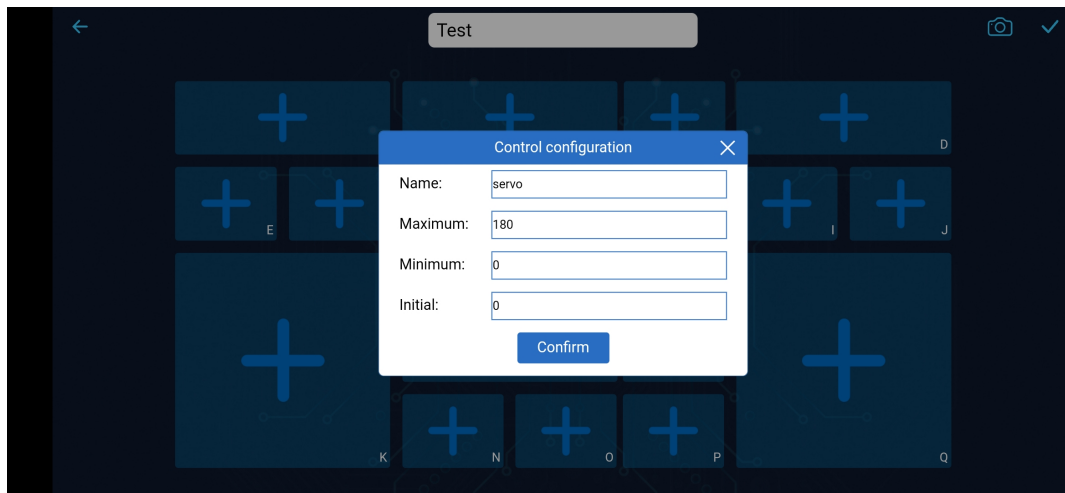
5. Klicken Sie im Bereich **H** und fügen Sie ein **Schieberegler**-Widget hinzu.



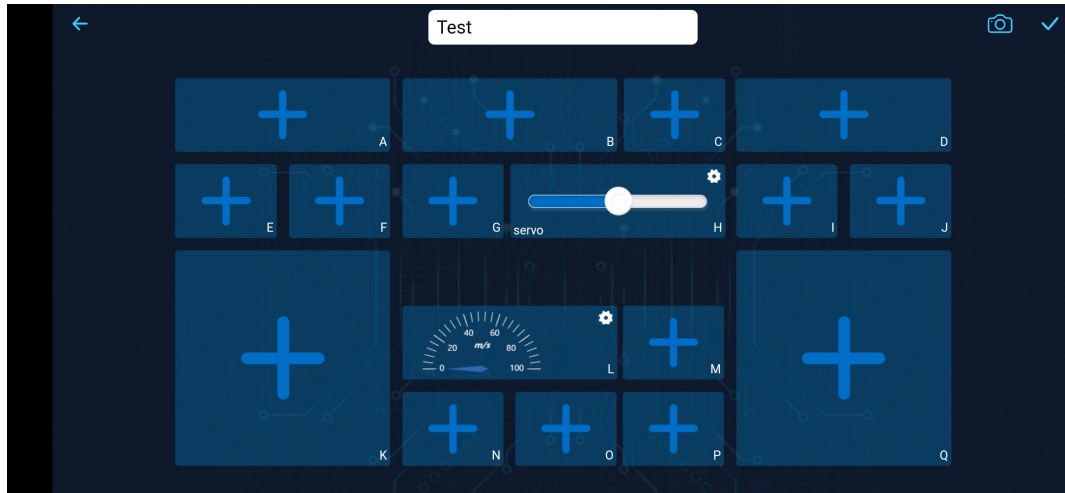
6. Klicken Sie auf das Zahnrad des Steuerelements, um das Einstellungsfenster zu öffnen.



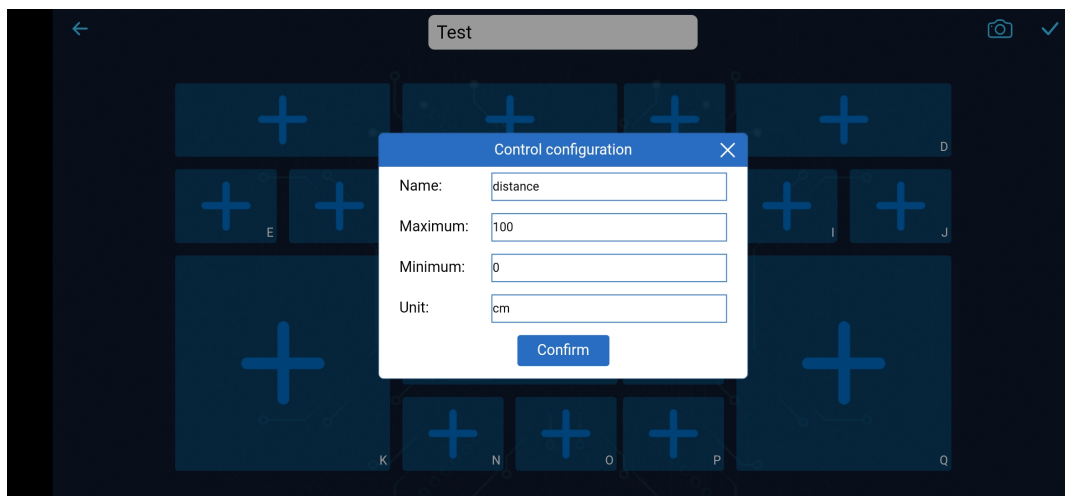
7. Setzen Sie Maximum auf 180 und Minimum auf 0, dann klicken Sie auf **Bestätigen**.



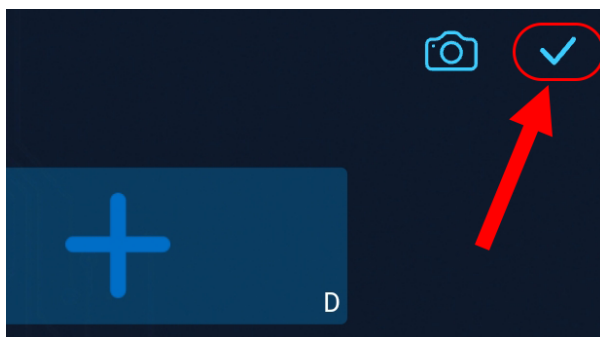
8. Klicken Sie im Bereich L und fügen Sie ein **Anzeige**-Widget hinzu.



9. Klicken Sie auf das Zahnrad der Anzeige, öffnen Sie das Einstellungsfenster, setzen Sie das Maximum auf 100, das Minimum auf 0 und die Einheit auf cm.



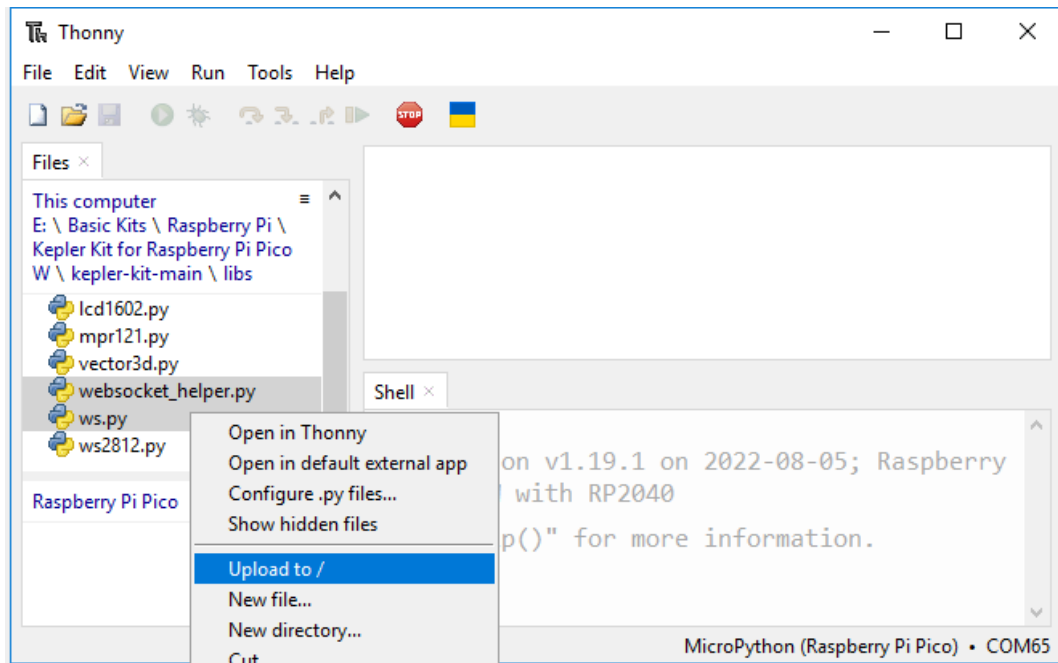
10. Nach Abschluss der Widget-Einstellungen klicken Sie auf Speichern.



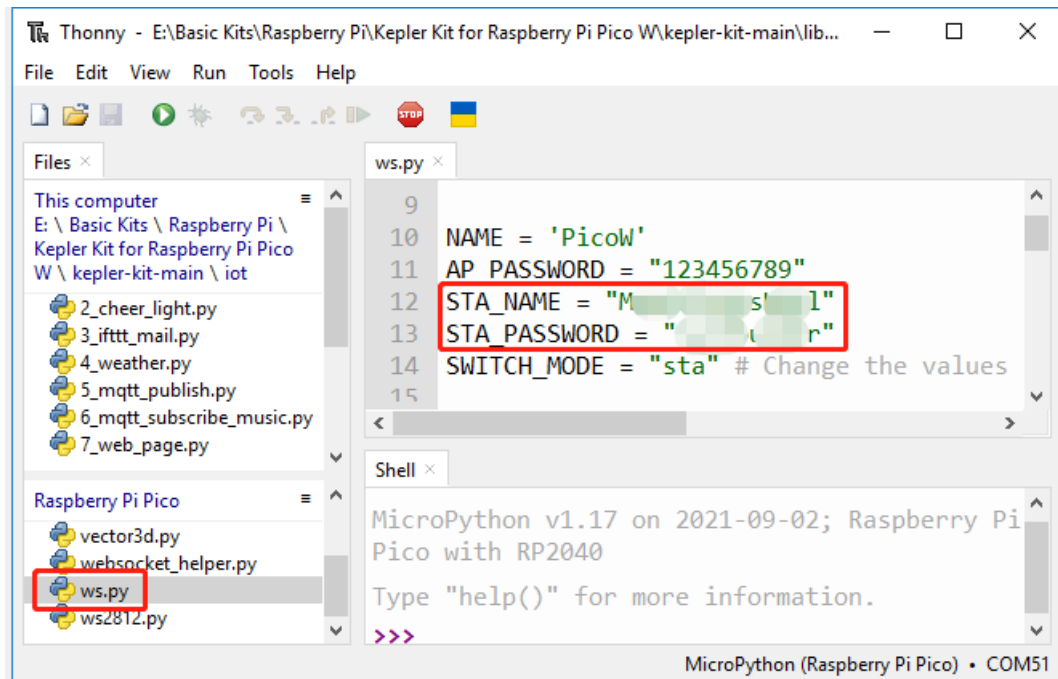
#### 4. Code ausführen

**Bemerkung:** Falls Ihr Pico W momentan die Anvil-Firmware verwendet, müssen Sie [1.3 Installation von MicroPython auf Ihrem Pico](#).

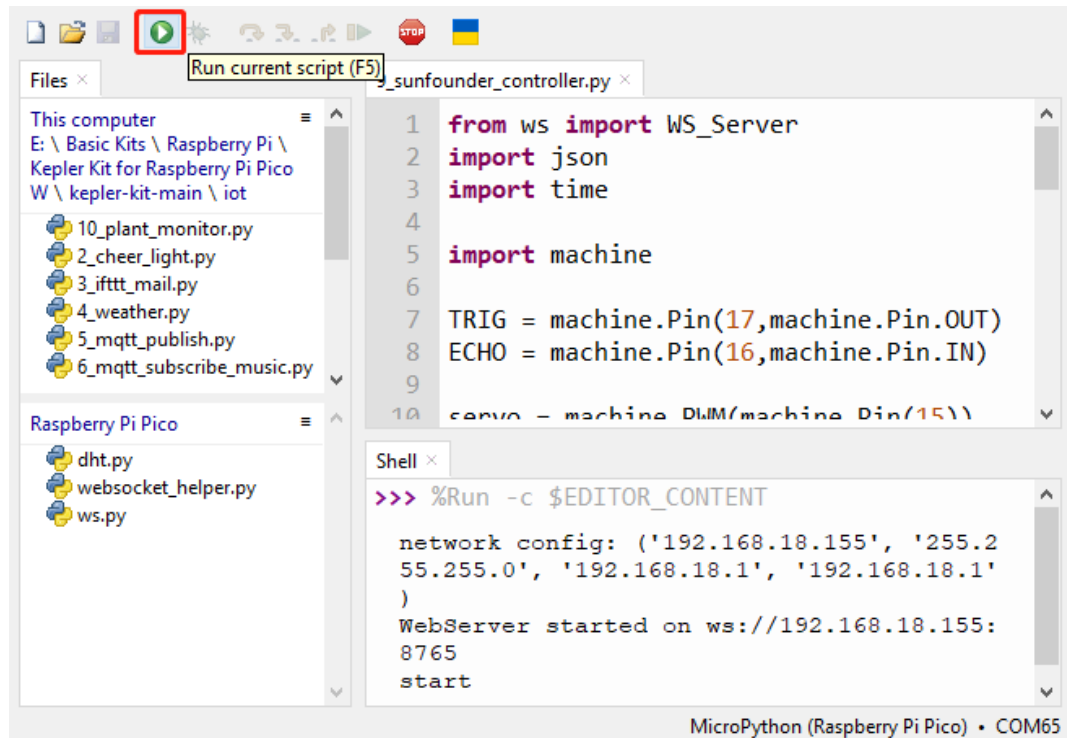
1. Laden Sie `ws.py` und `websocket_helper.py` aus dem Verzeichnis `kepler-kit-main/libs` auf den Raspberry Pi Pico W hoch.



2. Doppelklicken Sie auf das Skript `ws.py` und geben Sie den SSID und das PASSWORT Ihres WLANs ein.

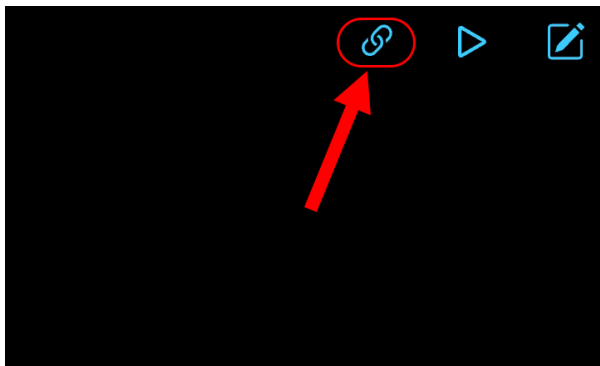


3. Öffnen Sie die Datei `9_sunfounder_controller.py` im Verzeichnis `kepler-kit-main/iot`. Klicken Sie auf die Schaltfläche **Dieses Skript ausführen** oder drücken Sie F5. Nach erfolgreicher Verbindung wird die IP-Adresse des Pico W angezeigt.

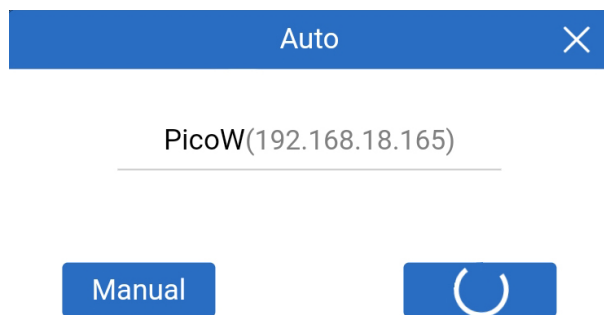


**Bemerkung:** Falls Sie dieses Skript beim Hochfahren ausführen lassen möchten, können Sie es als `main.py` auf dem Raspberry Pi Pico W speichern.

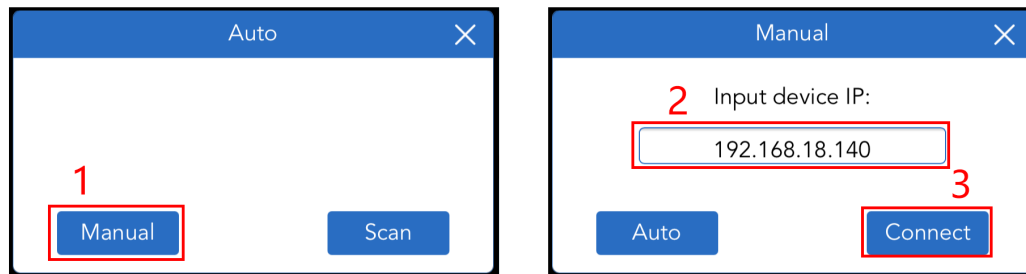
4. Kehren Sie zur SunFounder Controller APP zurück und klicken Sie auf die Schaltfläche **Verbinden**.



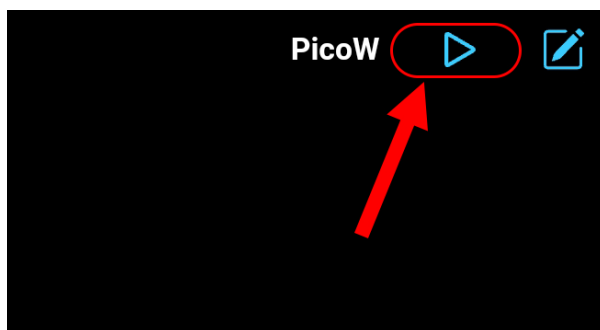
5. Wenn der PicoW erkannt wird, tippen Sie direkt darauf, um die Verbindung herzustellen.



6. Falls keine automatische Suche erfolgt, können Sie auch manuell die IP-Adresse eingeben, um eine Verbindung herzustellen.



7. Wenn Sie den Schieberegler im H-Bereich bewegen, nachdem Sie auf die Schaltfläche **Ausführen** geklickt haben, wird der Servo seinen Winkel einstellen. Die Anzeige im L-Bereich zeigt die Entfernung an, falls sich Ihre Hand innerhalb von 100 cm vor dem Ultraschallsensor befindet.



### Wie funktioniert das?

Die Klasse `WS_Server` in der Bibliothek `ws.py` implementiert die Kommunikation mit der APP. Im Folgenden ist das Grundgerüst für die Implementierung der wesentlichen Funktionalitäten dargestellt.

```
from ws import WS_Server
import json
import time

ws = WS_Server(8765) # init websocket

def main():
    ws.start()
    while True:
        status,result=ws.transfer()
        time.sleep_ms(100)

try:
    main()
finally:
    ws.stop()
```

Zuerst müssen wir ein `WS_Server`-Objekt erstellen.

```
ws = WS_Server(8765)
```

Starten Sie es.

```
ws.start()
```

Anschließend wird eine `while True`-Schleife verwendet, um den Datenaustausch zwischen dem Pico W und der SunFounder Controller APP durchzuführen.

```
while True:
    # websocket transfer data
    status,result = ws.transfer()

    # the status of transfer data
    print(status)

    # the data you recv
    print(result)

    # the data you send
    print(ws.send_dict)

    time.sleep_ms(100)
```

`status` ist `False`, wenn es fehlschlägt, Daten von der SunFounder Controller APP zu erhalten.

Und `result` sind die Daten, die der Pico W von der SunFounder Controller APP abgerufen hat. Drucken Sie diese aus, und es wird etwas Ähnliches wie das Folgende erscheinen. Dies sind die Werte aller Widget-Bereiche.

```
{'C': None, 'B': None, 'M': None,,,,, 'A': None, 'R': None}
```

In diesem Fall drucken wir die Werte des H-Bereichs separat aus und verwenden sie zur Steuerung des Schaltkreises.

```
status,result=ws.transfer()
#print(result)
if status == True:
    print(result['H'])
```

Und das `ws.send_dict` Wörterbuch enthält die Daten, die der Pico W an die SunFounder Controller APP sendet. Dieses Wörterbuch wird in der `WS_Server` Klasse erstellt und versendet, wenn die Methode `ws.transfer()` aufgerufen wird.

Die entsprechende Nachricht wird unten dargestellt.

```
{'Check': 'SunFounder Controller', 'Name': 'PicoW', 'Type': 'Blank'}
```

Dies ist eine leere Nachricht. Um sie ins Widget der SunFounder Controller APP zu kopieren, müssen wir den jeweiligen Bereichen im Wörterbuch Werte zuweisen. Beispielsweise weisen wir dem L-Bereich den Wert 50 zu.

```
ws.send_dict['L'] = 50
```

Die unten dargestellten Daten sehen dann wie folgt aus:

```
{'L': 50, 'Type': 'Blank', 'Name': 'PicoW', 'Check': 'SunFounder Controller'}
```

Für weitere Informationen zur Verwendung des SunFounder Controllers, siehe bitte [SunFounder Controller APP](#).



## 5.10 10. Pflanzenüberwachung mit dem @SunFounder Controller

In diesem Projekt lernen Sie, wie Sie mit der Sunfounder Controller-App ein automatisches Bewässerungssystem für Pflanzen aufbauen können.

In der App können Sie die aktuelle Temperatur und Luftfeuchtigkeit der Umgebung sowie den Wasserstand in den Blumentöpfen überprüfen. Zum Gießen der Pflanzen können Sie ebenfalls den Button in der App drücken.

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

Es ist natürlich bequem, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ELEMENTE IN DIESEM SET	LINK
Kepler-Kit	450+	

Sie können die Einzelteile auch über die folgenden Links separat erwerben.

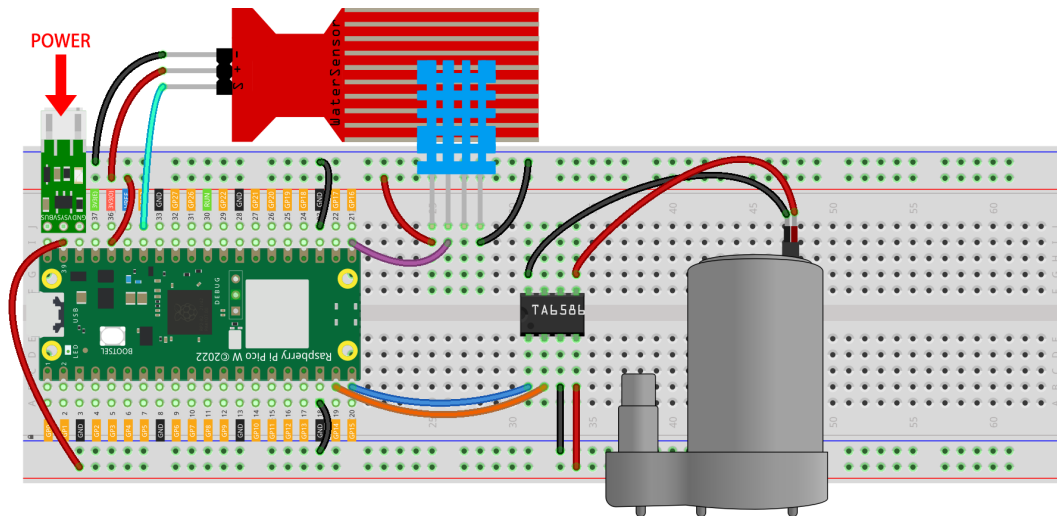
SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>DHT11 Temperatur- und Feuchtigkeitssensor</i>	1	
6	<i>Wasserspiegelsensor-Modul</i>	1	
7	<i>TA6586 - Motorsteuerungs-Chip</i>	1	
8	<i>Li-Po-Lademodul</i>	1	
9	18650 Batterie	1	
10	Batteriehalter	1	
11	<i>DC-Wasserpumpe</i>	1	

### Ablauf

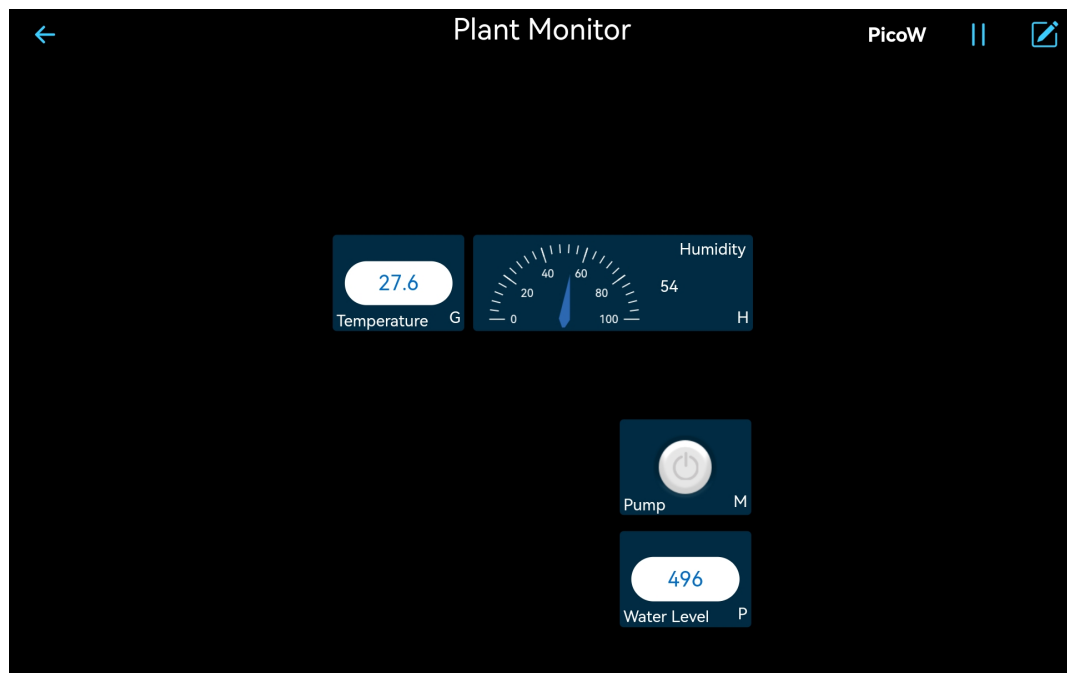
**Bemerkung:** Es wird empfohlen, das vorherige Projekt 9. *Spiel mit dem @SunFounder Controller* abzuschließen, da es Ihnen den grundlegenden Umgang mit dem SunFounder Controller näherbringt.

1. Verdrahten Sie den Schaltkreis.

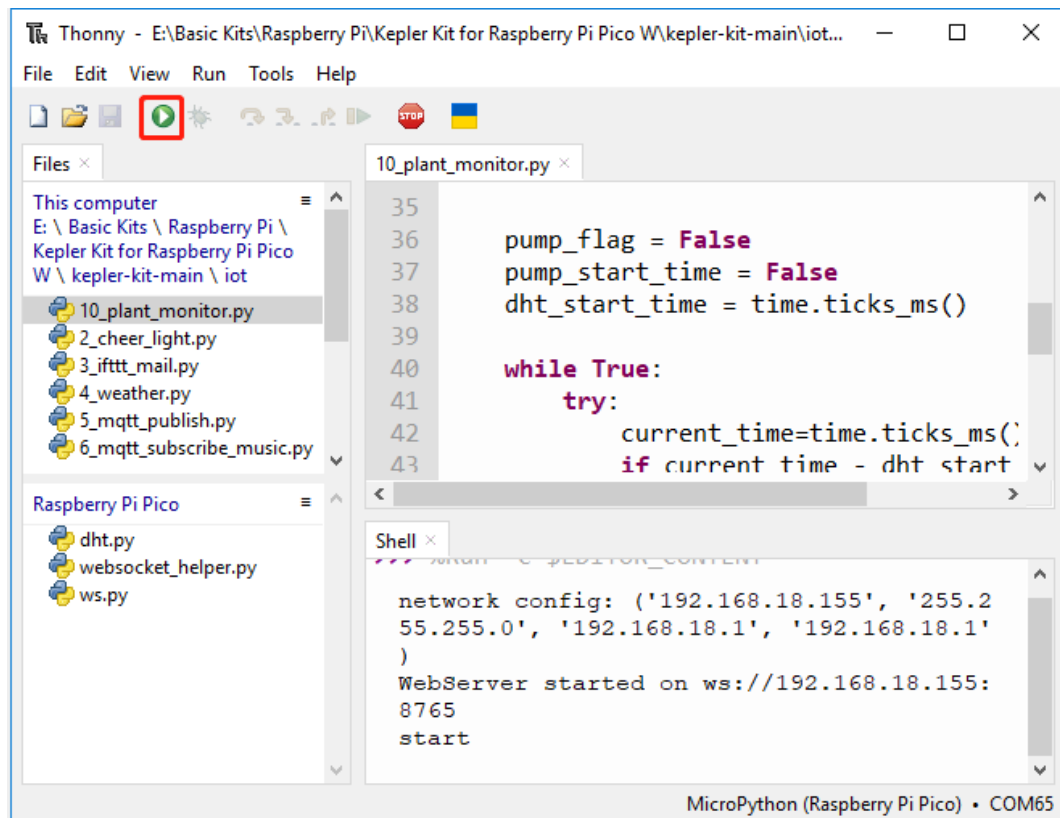




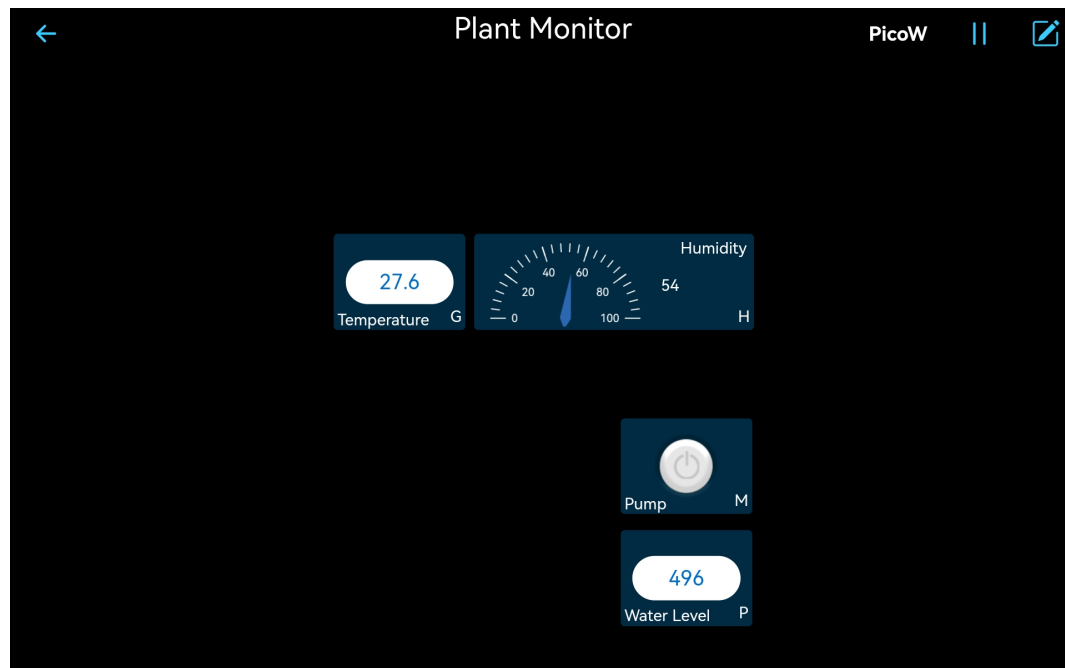
- Erstellen Sie einen neuen Controller, fügen Sie die folgenden Widgets hinzu und ändern Sie deren Namen.



- Öffnen Sie die Datei `10_plant_monitor.py` im Pfad `kepler-kit-main/iot`. Klicken Sie auf die Schaltfläche **Aktuelles Skript ausführen** oder drücken Sie F5, um es zu starten. Nach erfolgreicher Verbindung wird die IP von Pico W angezeigt.



4. Kehren Sie zur SunFounder-App zurück und klicken Sie auf „Start“, nachdem Sie sich mit PicoW verbunden haben. In der App können Sie die Temperatur und Luftfeuchtigkeit der Umgebung sowie den Wasserstand der Topfpflanze sehen. Wenn Sie der Meinung sind, dass die Pflanze mehr Wasser benötigt, können Sie den Button drücken, um sie für fünf Sekunden zu bewässern.



5. Wenn Sie möchten, dass dieses Skript automatisch startet, können Sie es als `main.py` auf dem Raspberry Pi Pico W speichern.

### Wie funktioniert das Ganze?

Dieses Projekt funktioniert im Grunde genommen genauso wie *9. Spiel mit dem @SunFounder Controller*.

Zusätzlich verwendet das Projekt auch den DHT11, eine Pumpe und ein Wasserstandmodul. Details zur Verwendung dieser Komponenten finden Sie unter *6.2 Temperatur - Feuchtigkeit*, *3.6 Pumpensteuerung*, *2.14 Wasserstand erfüllen*.



Dieses Kapitel beinhaltet die Installation der Arduino-IDE, das Hochladen von Code auf den Raspberry Pi mit der Arduino-IDE und eine Vielzahl von interessanten und praxisnahen Projekten, die Ihnen einen schnellen Einstieg in die Arduino-Programmierung ermöglichen.

Wir empfehlen, die Kapitel in der angegebenen Reihenfolge durchzuarbeiten.

### 1. Erste Schritte

## 6.1 1.1 Arduino IDE installieren (Wichtig)

Die Arduino-IDE, bekannt als Arduino Integrated Development Environment, bietet die gesamte notwendige Softwareunterstützung für die Fertigstellung eines Arduino-Projekts. Es handelt sich um eine speziell für Arduino entwickelte Programmiersoftware, die vom Arduino-Team bereitgestellt wird und das Schreiben von Programmen sowie deren Upload auf das Arduino-Board ermöglicht.

Die Arduino IDE 2.0 ist ein Open-Source-Projekt und stellt einen großen Fortschritt gegenüber ihrem robusten Vorgänger, der Arduino IDE 1.x, dar. Sie kommt mit einer überarbeiteten Benutzeroberfläche, einem verbesserten Board- und Bibliotheksmanager, einem Debugger, einer Autocomplete-Funktion 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.

### 6.1.1 Voraussetzungen

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

## 6.1.2 Arduino IDE 2.0 herunterladen

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



 **Arduino IDE 2.0.0**

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

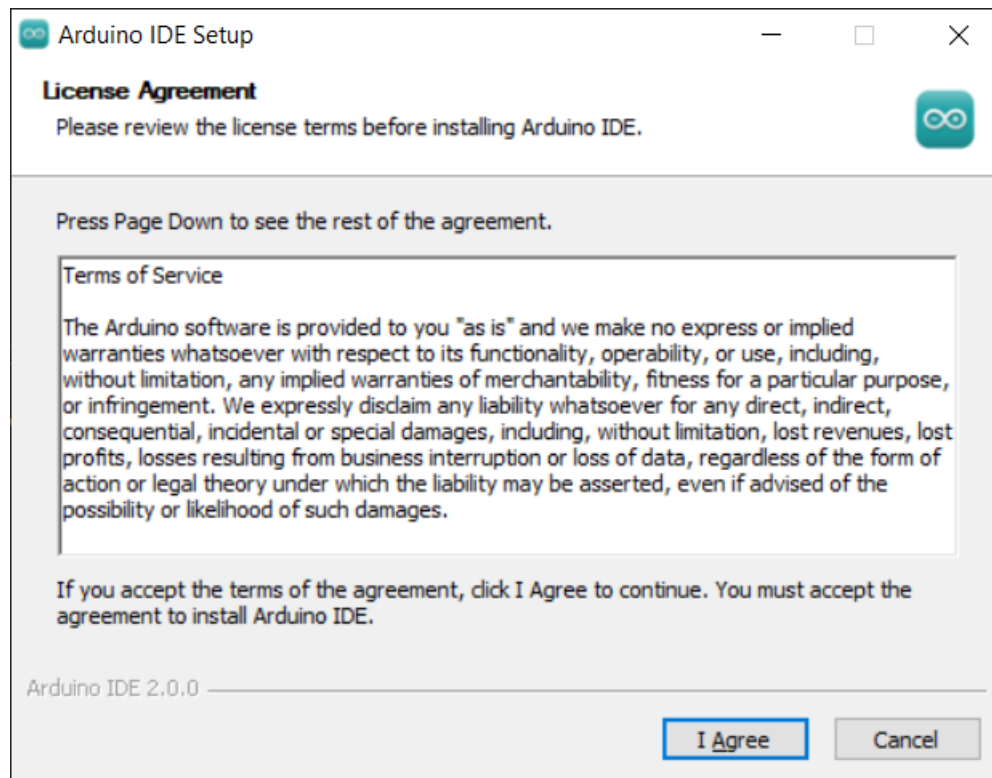
**DOWNLOAD OPTIONS**

- Windows** Win 10 and newer, 64 bits
- Windows** MSI installer
- Windows** ZIP file
- Linux** AppImage 64 bits (X86-64)
- Linux** ZIP file 64 bits (X86-64)
- macOS** 10.14: "Mojave" or newer, 64 bits

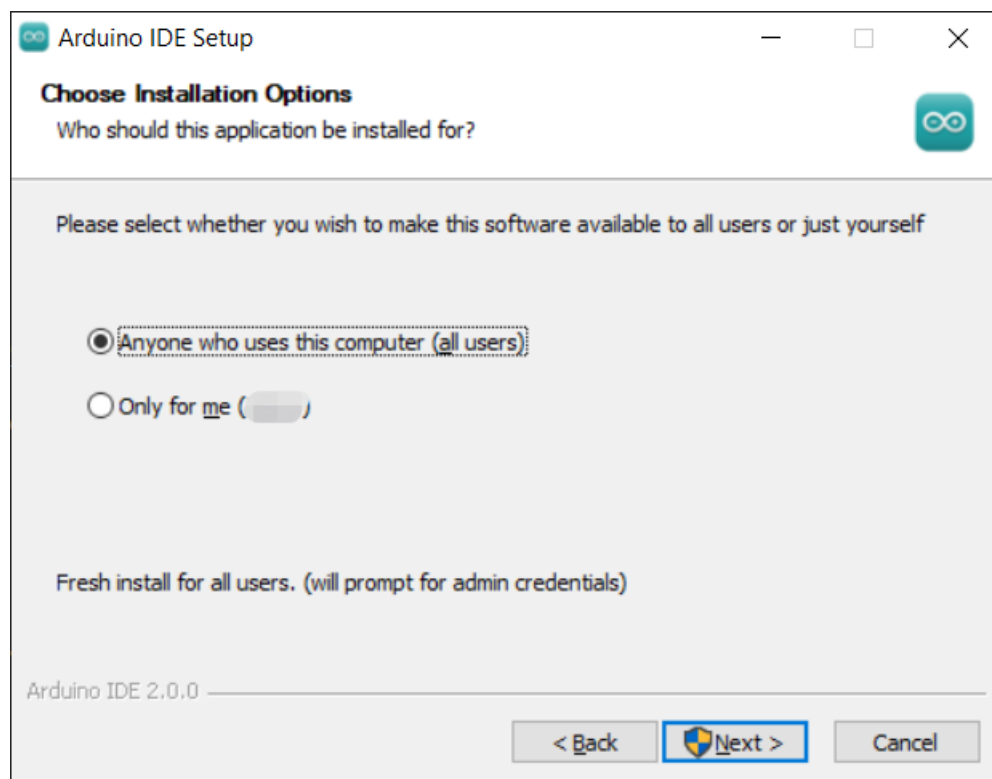
## 6.1.3 Installation

### Windows

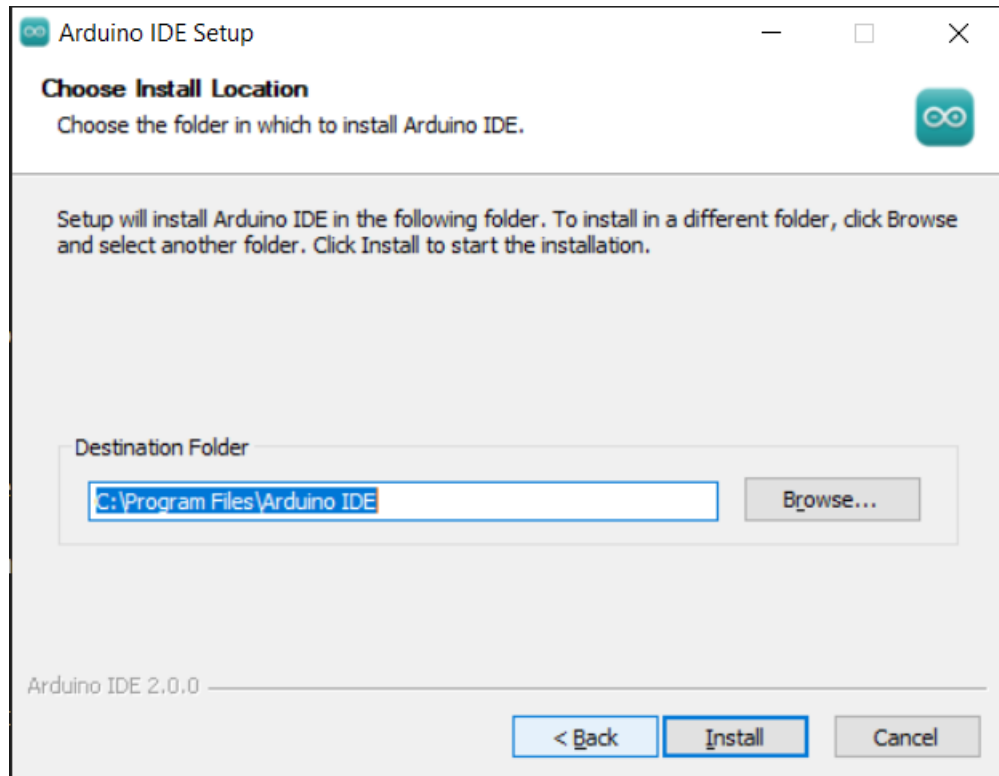
1. Doppelklicken Sie auf die Datei `arduino-ide_xxxx.exe`, um die heruntergeladene Datei auszuführen.
2. Lesen Sie die Lizenzvereinbarung und stimmen Sie dieser zu.



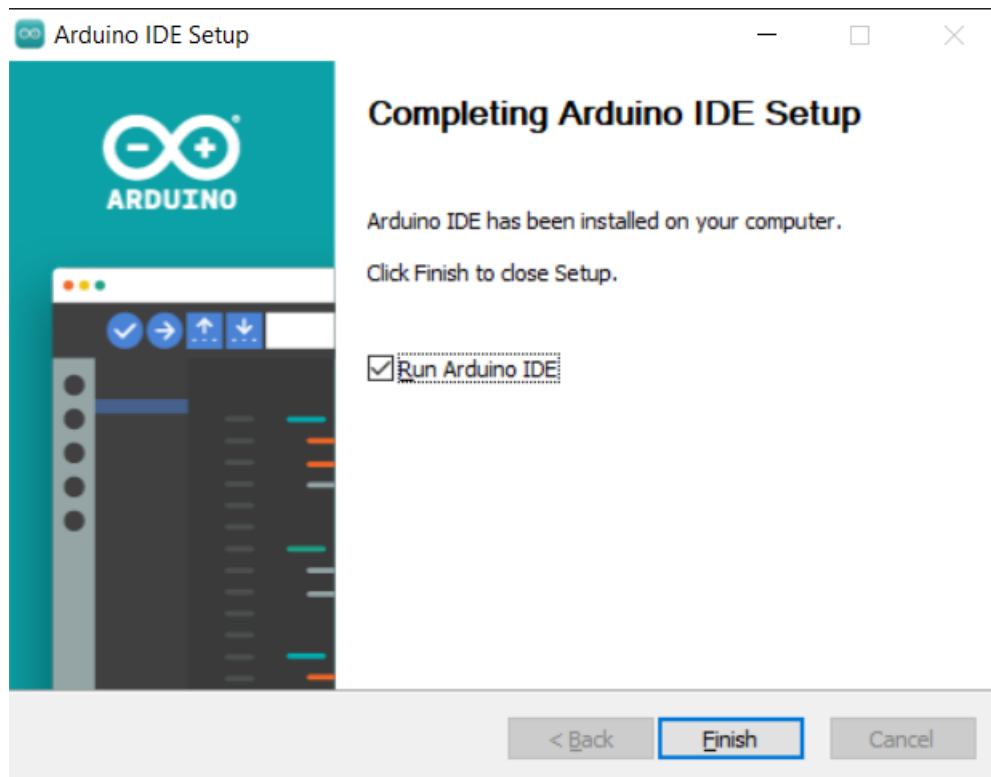
3. Wählen Sie die Installationsoptionen aus.



4. Wählen Sie den Installationsort. Es wird empfohlen, die Software auf einem anderen Laufwerk als dem Systemlaufwerk zu installieren.



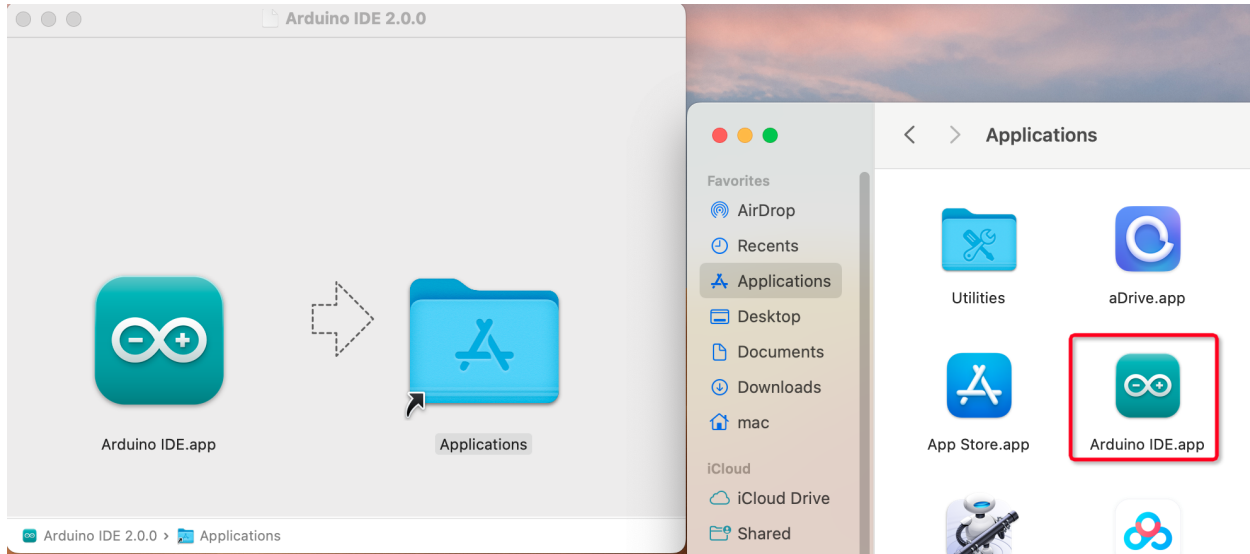
5. Abschließend klicken Sie auf „Fertigstellen“.





## 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 sollte die Arduino IDE erfolgreich installiert sein.

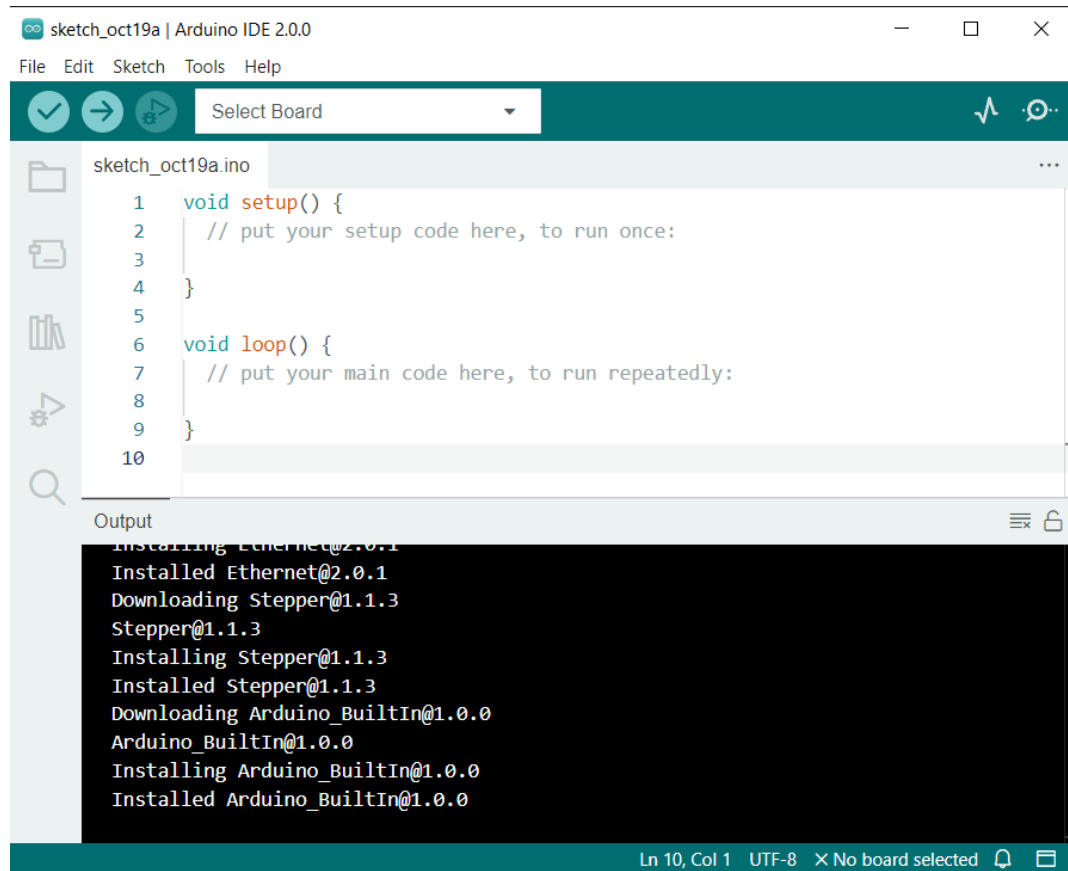


## Linux

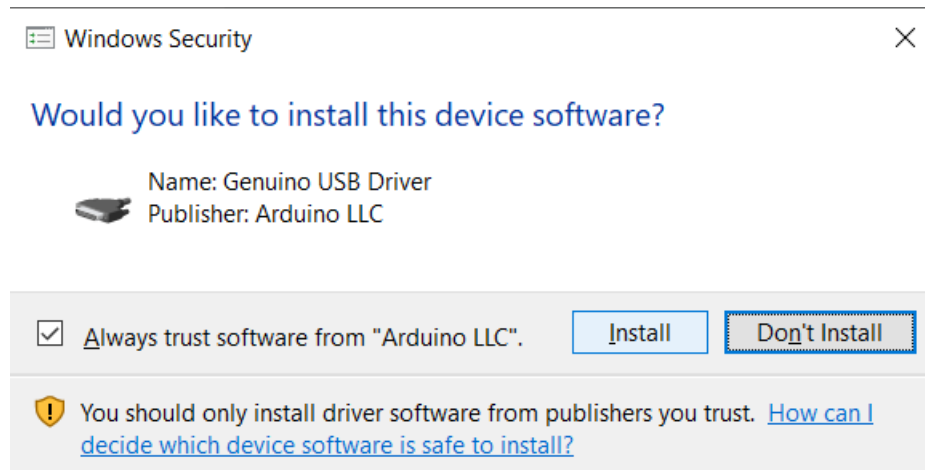
Für das Tutorial zur Installation der Arduino IDE 2.0 unter Linux verweisen wir auf: <https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing#linux>

### 6.1.4 IDE öffnen

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



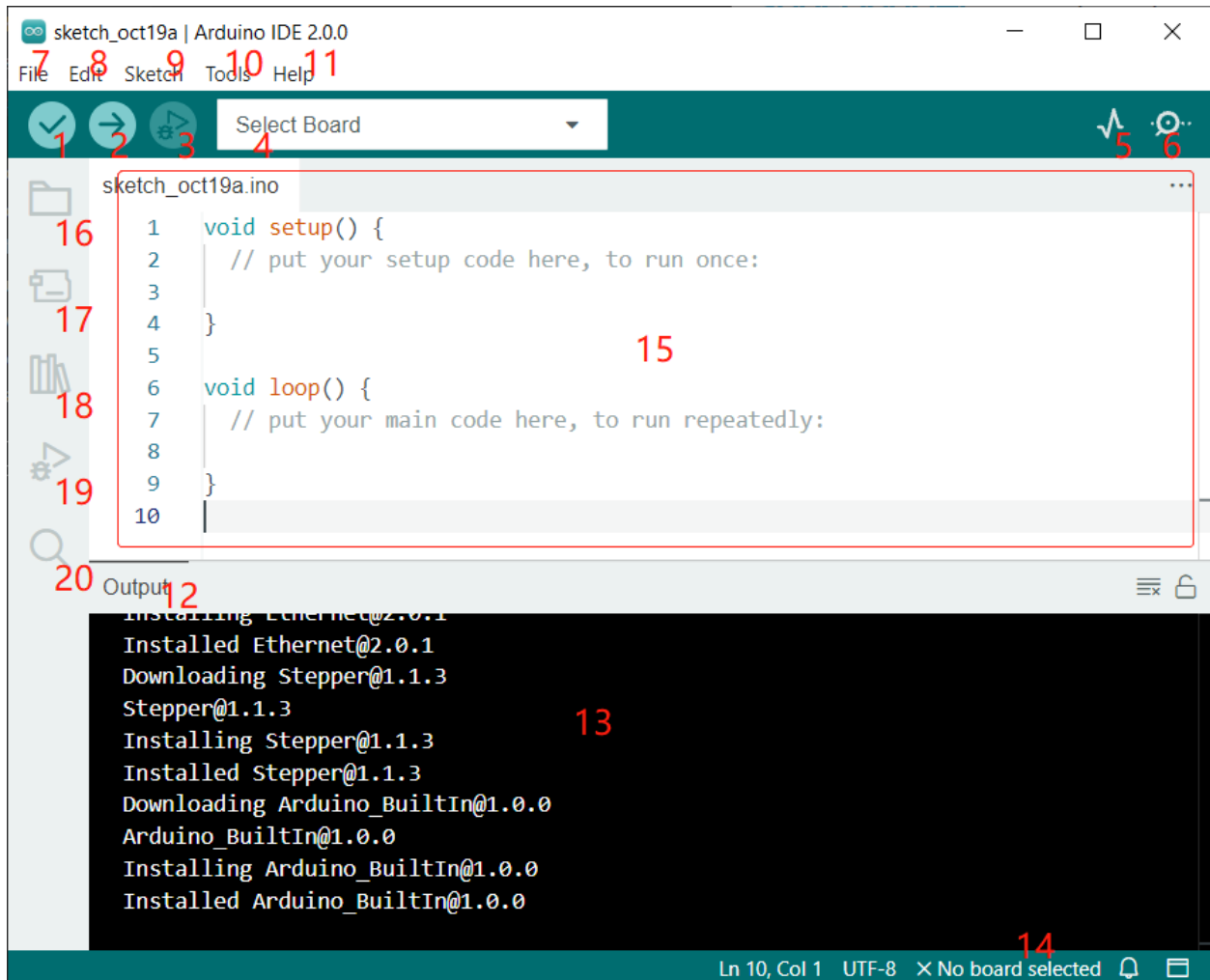
2. Zudem könnte Ihre Firewall oder Ihr Sicherheitszentrum einige Male nachfragen, ob Sie bestimmte Gerätetreiber installieren möchten. Bitte installieren Sie alle davon.



3. Nun ist Ihre Arduino IDE einsatzbereit!

**Bemerkung:** Falls einige Installationen aufgrund von Netzwerkproblemen oder aus anderen Gründen nicht funktioniert haben sollten, können Sie die Arduino IDE erneut öffnen, und der Rest der Installation wird abgeschlossen. Das Ausgabefenster wird erst dann automatisch geöffnet, wenn alle Installationen abgeschlossen sind und Sie auf „Überprüfen“ oder „Hochladen“ klicken.

## 6.2 1.2 Vorstellung der Arduino IDE



1. **Überprüfen (Verify):** Kompiliert Ihren Code. Jegliche Syntaxfehler werden durch Fehlermeldungen hervorgehoben.
2. **Hochladen (Upload):** Lädt den Code auf Ihr Board. Wenn Sie auf den Button klicken, flackern die RX- und TX-LEDs auf dem Board schnell und hören erst auf, wenn der Upload abgeschlossen ist.
3. **Debuggen (Debug):** Dient der zeilenweisen Fehlerüberprüfung.
4. **Board auswählen (Select Board):** Schnelleinrichtung von Board und Port.
5. **Serielle Darstellung (Serial Plotter):** Überprüfung der Veränderungen der ausgelesenen Werte.
6. **Serieller Monitor (Serial Monitor):** Ein Fenster öffnet sich nach dem Klicken auf den Button. Es empfängt die Daten, die von Ihrer Steuerplatine gesendet werden. Sehr nützlich für die Fehlersuche.
7. **Datei (File):** Ein Dropdown-Menü erscheint nach dem Klicken, einschließlich Optionen für das Erstellen, Öffnen, Speichern, Schließen von Dateien und weitere Konfigurationsparameter.
8. **Bearbeiten (Edit):** Beim Klicken erscheint ein Dropdown-Menü mit verschiedenen Bearbeitungsoptionen wie **Ausschneiden**, **Kopieren**, **Einfügen**, **Suchen**, und dergleichen, jeweils mit den zugehörigen Tastenkombinationen.

9. **Skizze (Sketch):** Beinhaltet Aktionen wie **Überprüfen**, **Hochladen**, **Dateien hinzufügen**, usw. Eine besonders wichtige Funktion ist **Bibliothek einbinden (Include Library)** - hier können Sie Bibliotheken hinzufügen.
10. **Werkzeuge (Tool):** Enthält verschiedene Tools - am häufigsten genutzt sind Board (das verwendete Board) und Port (der Port, an dem sich Ihr Board befindet). Bevor Sie den Code hochladen, müssen Sie diese auswählen oder überprüfen.
11. **Hilfe (Help):** Falls Sie Anfänger sind, können Sie hier unter den Menüoptionen die benötigte Hilfe finden, einschließlich Bedienung in der IDE, grundlegende Informationen, Fehlerbehebung, Code-Erläuterungen, usw.
12. **Ausgabebereich (Output Bar):** Hier können Sie den Ausgabe-Tab wechseln.
13. **Ausgabefenster (Output Window):** Anzeige von Informationen.
14. **Board und Port:** Hier sehen Sie eine Vorschau des für den Code-Upload ausgewählten Boards und Ports. Bei Fehlern können Sie diese erneut unter **Werkzeuge (Tools)** -> **Board / Port** auswählen.
15. Der Bearbeitungsbereich der IDE. Hier können Sie Ihren Code schreiben.
16. **Skizzenbuch (Sketchbook):** Dient der Verwaltung Ihrer Skizzen-Dateien.
17. **Board-Verwaltung (Board Manager):** Zur Verwaltung der Board-Treiber.
18. **Bibliothek-Verwaltung (Library Manager):** Zum Verwalten Ihrer Bibliotheksdateien.
19. **Debuggen (Debug):** Unterstützt das Debugging des Codes.
20. **Suchen (Search):** Ermöglicht die Codesuche in Ihren Skizzen.

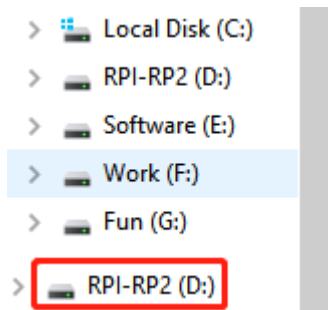
## 6.3 1.3 Raspberry Pi Pico W einrichten (Wichtig)

### 6.3.1 1. UF2-Firmware installieren

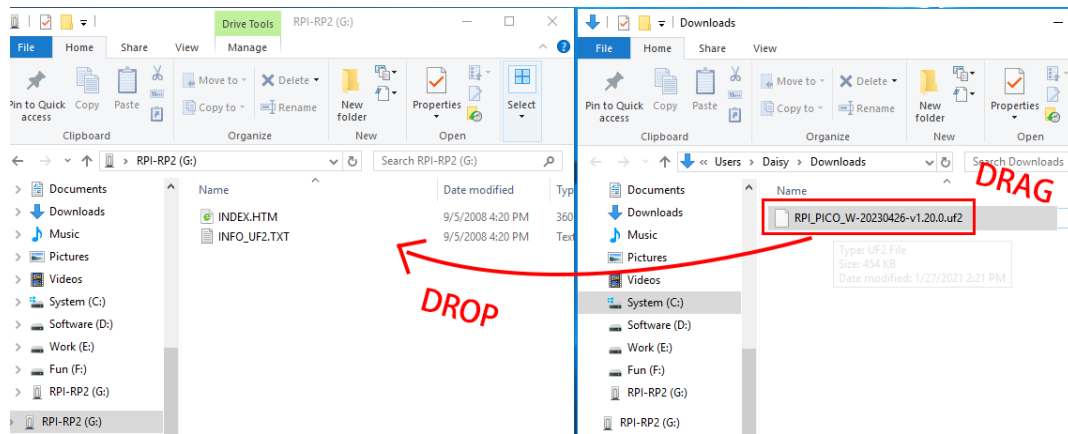
Wenn Sie den Raspberry Pi Pico W zum ersten Mal anschließen oder währenddessen die BOOTSEL-Taste gedrückt halten, erscheint das Gerät als Laufwerk, dem jedoch kein COM-Port zugewiesen ist. Dies verhindert das Hochladen von Code.

Um dieses Problem zu beheben, müssen Sie die UF2-Firmware installieren. Diese Firmware ist sowohl mit MicroPython als auch mit der Arduino IDE kompatibel.

1. Laden Sie die UF2-Firmware über den folgenden Link herunter.
  - Raspberry Pi Pico W UF2-Firmware
2. Verbinden Sie Ihren Raspberry Pi Pico W über ein Micro-USB-Kabel mit Ihrem Computer. Ihr Pico W wird als Massenspeichergerät mit dem Namen **RPI-RP2** eingebunden.



3. Ziehen Sie die heruntergeladene UF2-Firmware in das Laufwerk **RPI-RP2**.

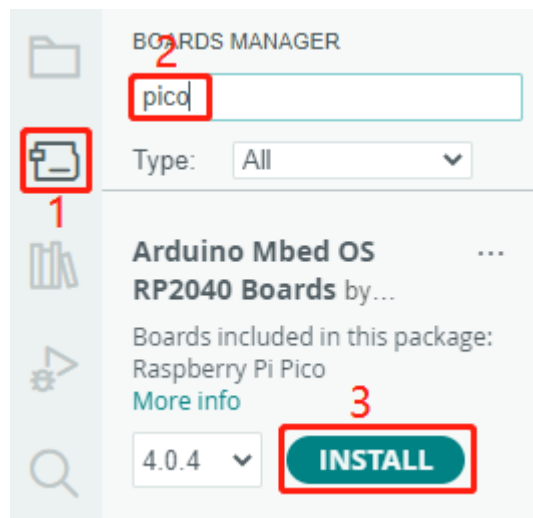


4. Nach diesem Vorgang verschwindet das Laufwerk **RPI-RP2**, und Sie können mit den nächsten Schritten fortfahren.

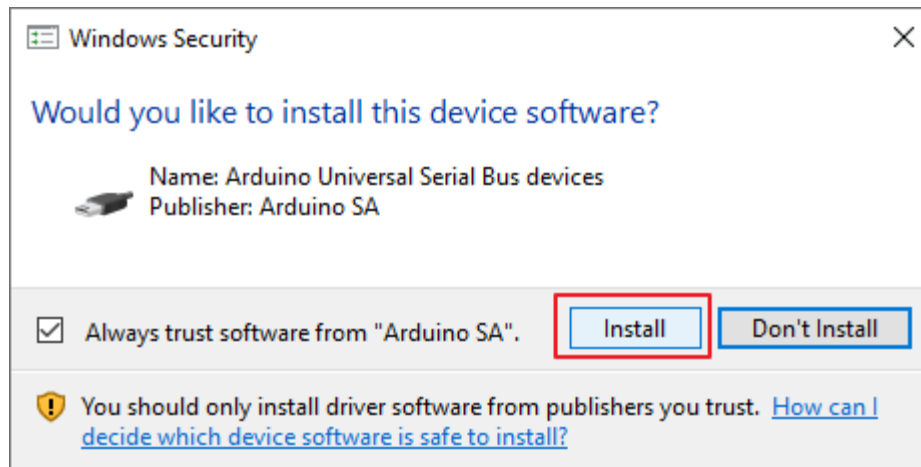
## 6.3.2 2. Das Board-Paket installieren

Um den Raspberry Pi Pico W zu programmieren, müssen Sie das entsprechende Paket in der Arduino IDE installieren. Hier ist eine Schritt-für-Schritt-Anleitung:

1. Im **Boards Manager**-Fenster suchen Sie nach **pico**. Klicken Sie auf die Schaltfläche **Installieren**, um die Installation zu starten. Damit installieren Sie das **Arduino Mbed OS RP2040 Boards**-Paket, das die Unterstützung für den Raspberry Pi Pico W enthält.



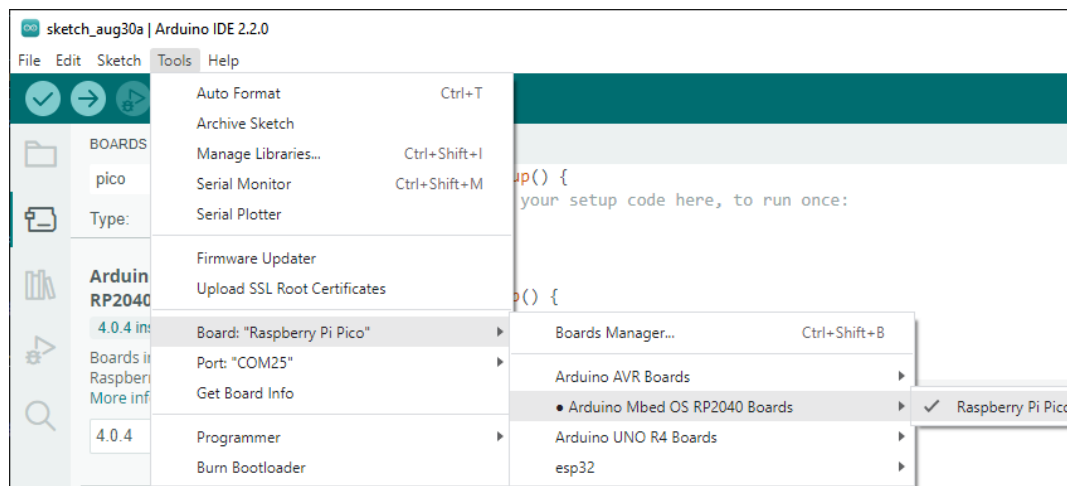
2. Während des Vorgangs erscheinen einige Popup-Aufforderungen zur Installation spezifischer Gerätetreiber. Wählen Sie „**Installieren**“.



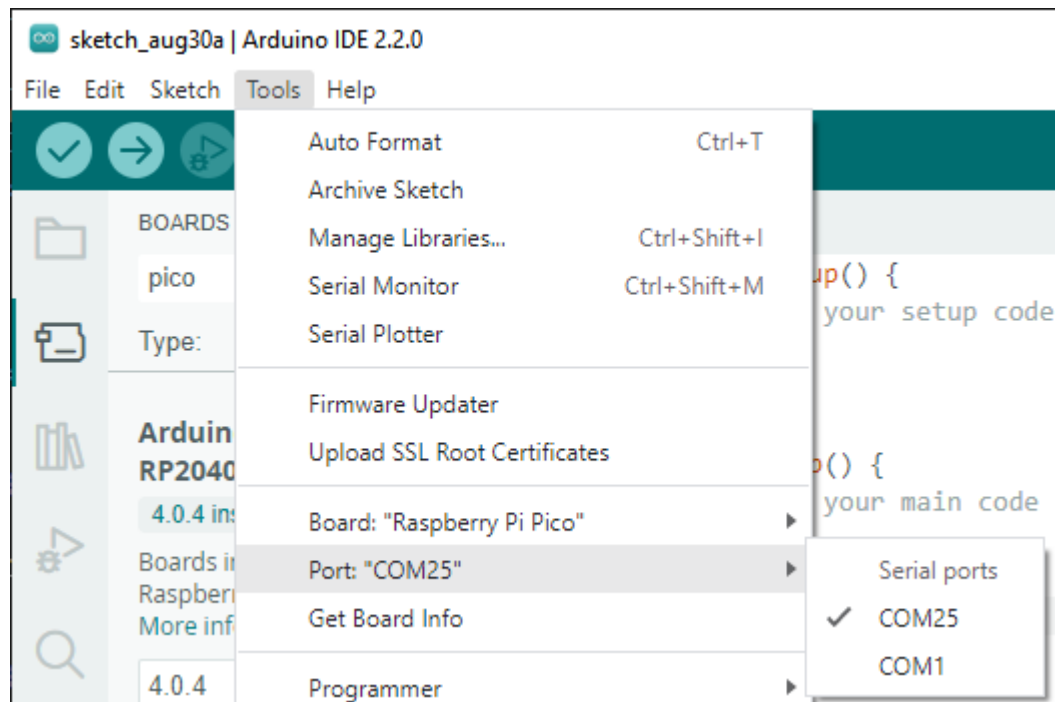
3. Anschließend erscheint eine Benachrichtigung, die den erfolgreichen Abschluss der Installation bestätigt.

### 6.3.3 3. Auswahl von Board und Port

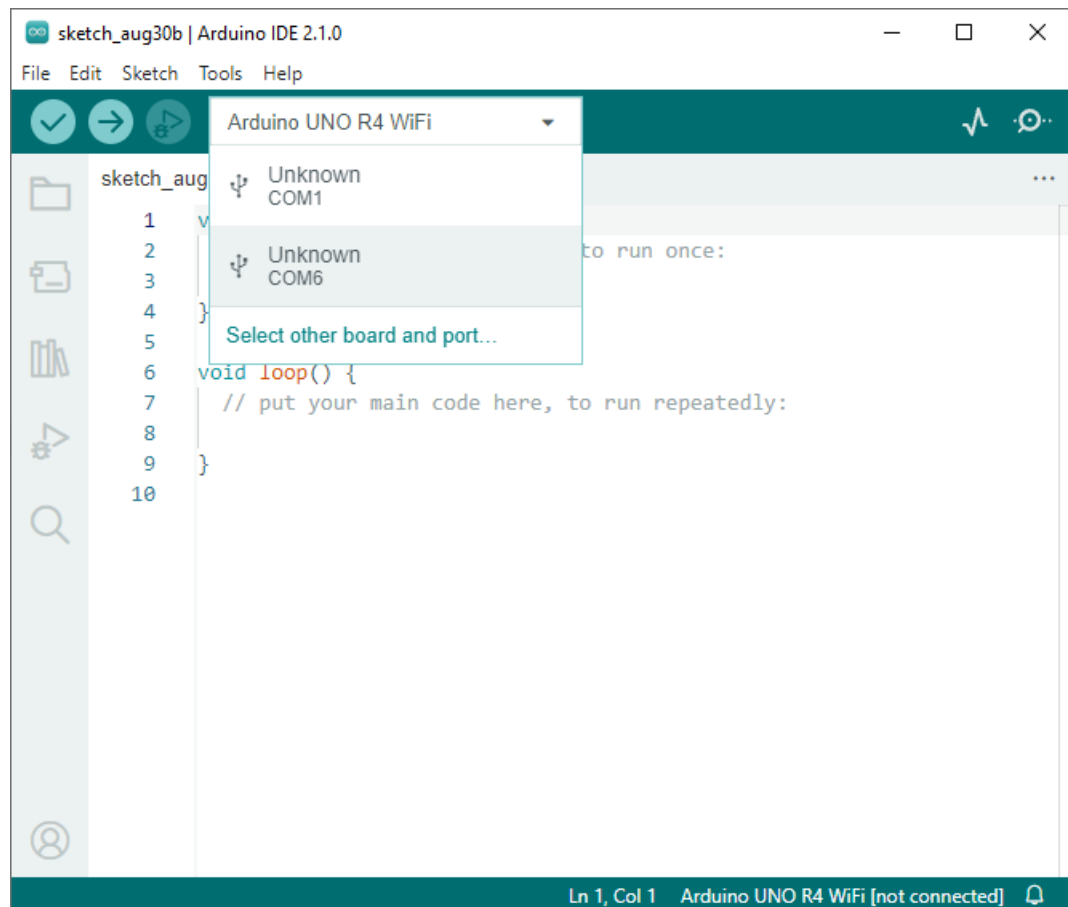
1. Um das passende Board auszuwählen, navigieren Sie zu **Werkzeuge -> Board -> Arduino Mbed OS RP2040 Boards -> Raspberry Pi Pico**.



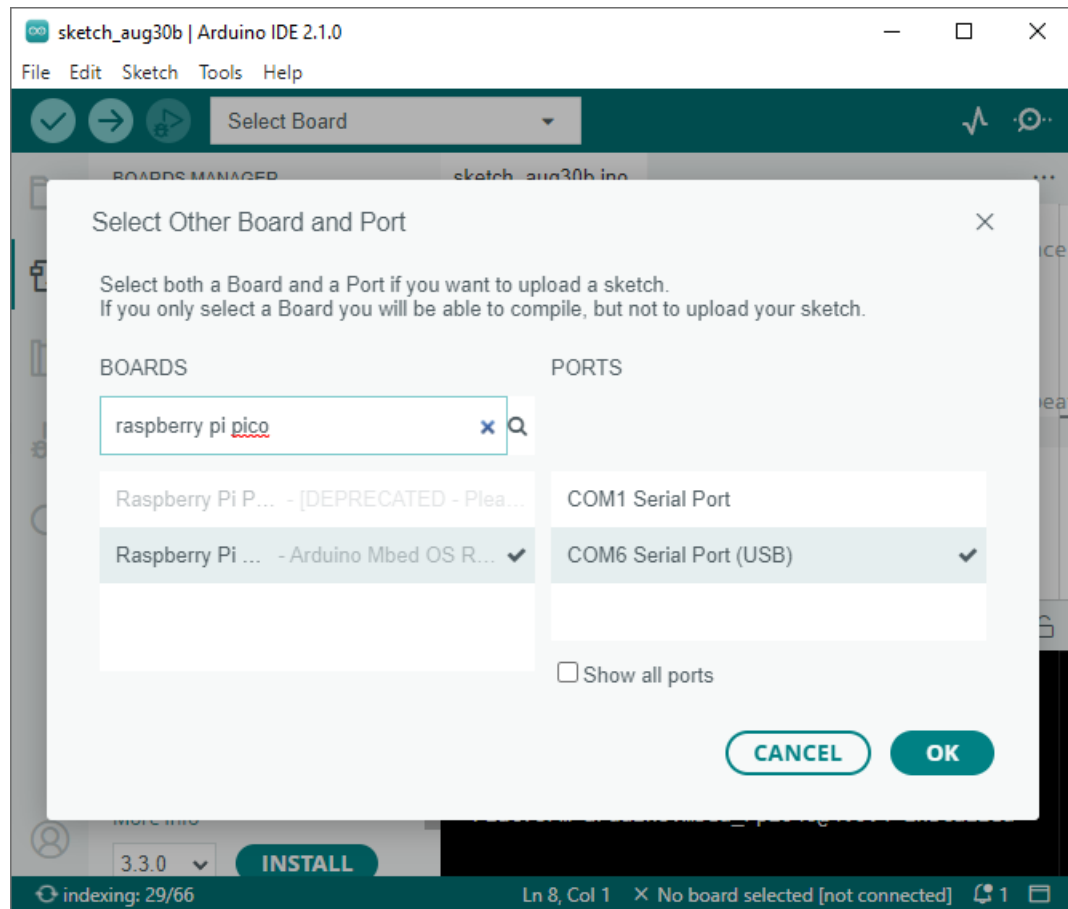
2. Falls Ihr Raspberry Pi Pico W am Computer angeschlossen ist, stellen Sie den richtigen Port ein, indem Sie zu **Werkzeuge -> Port** navigieren.



3. Arduino 2.0 bietet eine neue Schnellauswahl-Funktion. Für den Raspberry Pi Pico W, der normalerweise nicht automatisch erkannt wird, klicken Sie auf **Andere Boards und Ports auswählen**.



4. Geben Sie **Raspberry Pi Pico** in die Suchleiste ein, wählen Sie es aus, wenn es erscheint, wählen Sie den entsprechenden Port und klicken Sie auf **OK**.



5. Später können Sie es über dieses Schnellzugriffsfenster einfach erneut auswählen.



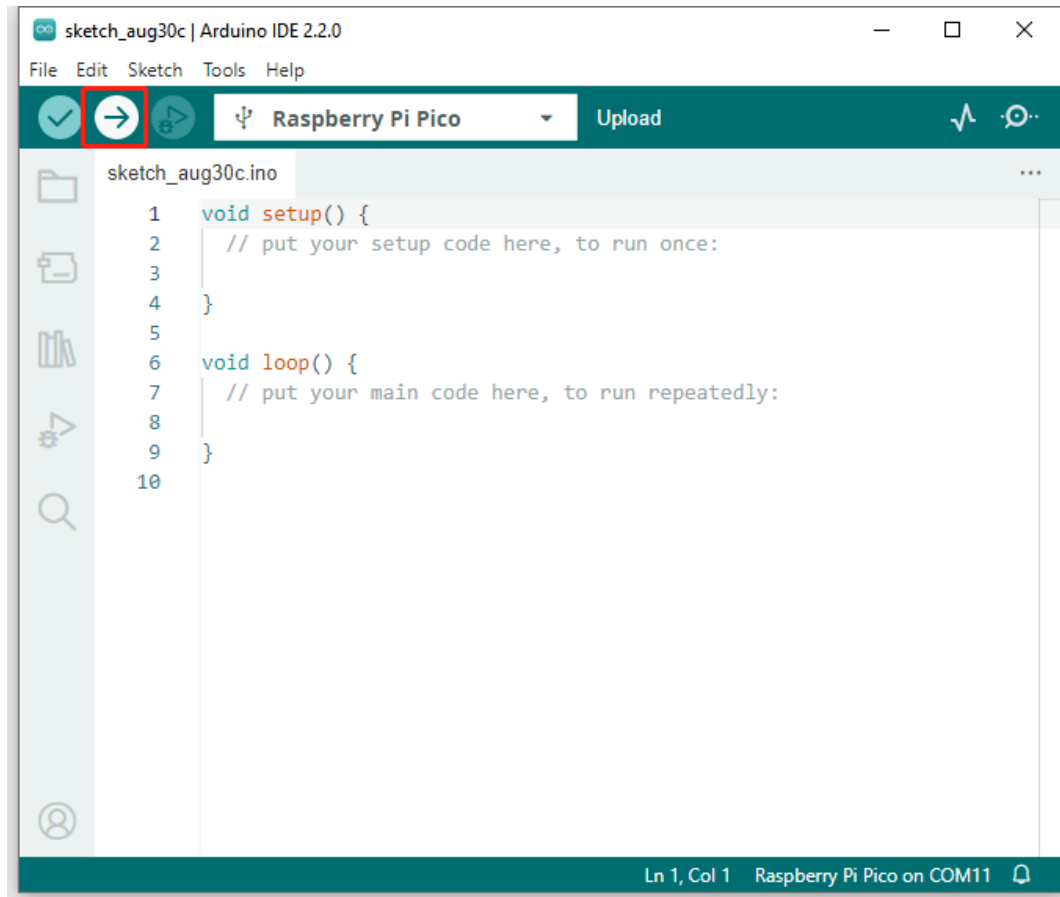
6. Mit einer dieser Methoden können Sie das korrekte Board und den Port einstellen. Nun können Sie Code auf den Raspberry Pi Pico W hochladen.



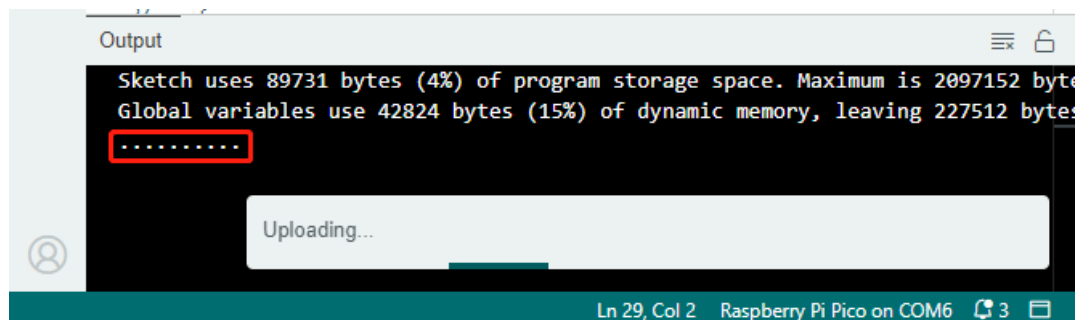
### 6.3.4 4. Code hochladen

Jetzt gehen wir darauf ein, wie Sie Code auf Ihren Raspberry Pi Pico W hochladen können.

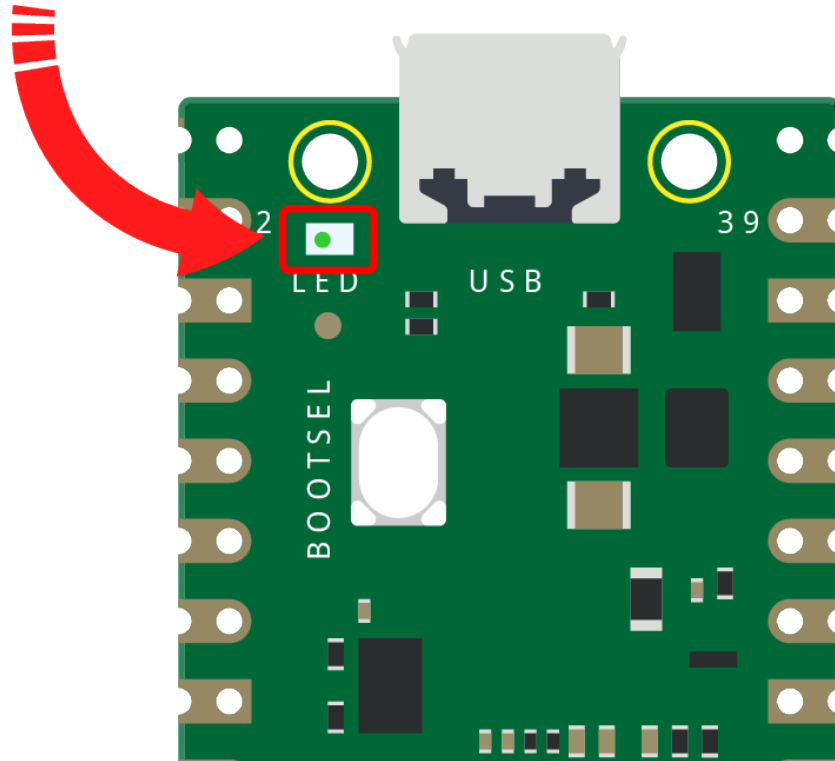
1. Öffnen Sie eine beliebige .ino-Datei oder verwenden Sie den aktuell angezeigten leeren Sketch. Klicken Sie dann auf die Schaltfläche **Hochladen**.



2. Warten Sie, bis die Hochlademeldung erscheint, wie unten gezeigt.



3. Halten Sie die **BOOTSEL**-Taste gedrückt, ziehen Sie Ihren Raspberry Pi Pico W kurz ab und stecken Sie ihn wieder ein.

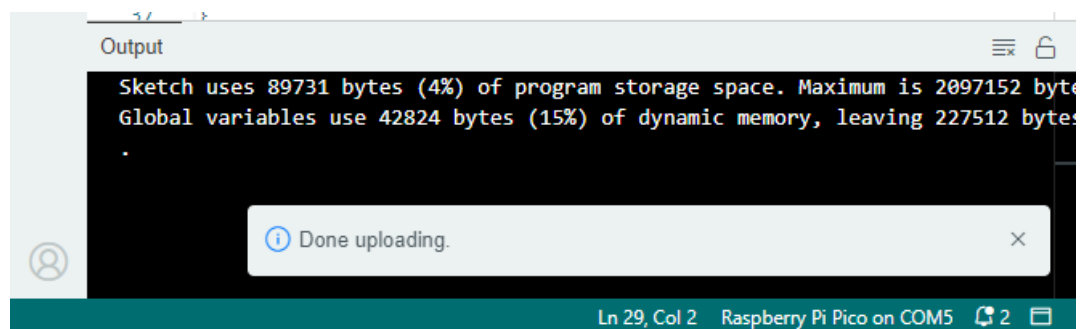


---

**Bemerkung:**

- Dieser Schritt ist entscheidend, insbesondere für Erstnutzer der Arduino IDE. Wenn Sie diesen Schritt überspringen, wird das Hochladen fehlschlagen.
  - Sobald der Code erfolgreich hochgeladen wurde, wird Ihr Pico W vom Computer erkannt. Für die zukünftige Nutzung stecken Sie ihn einfach an den Computer.
- 

4. Ein Hinweis auf den erfolgreichen Upload wird angezeigt.



## 6.4 1.4 Bibliotheken installieren (Wichtig)

### Code herunterladen

Laden Sie den relevanten Code über den untenstehenden Link herunter.

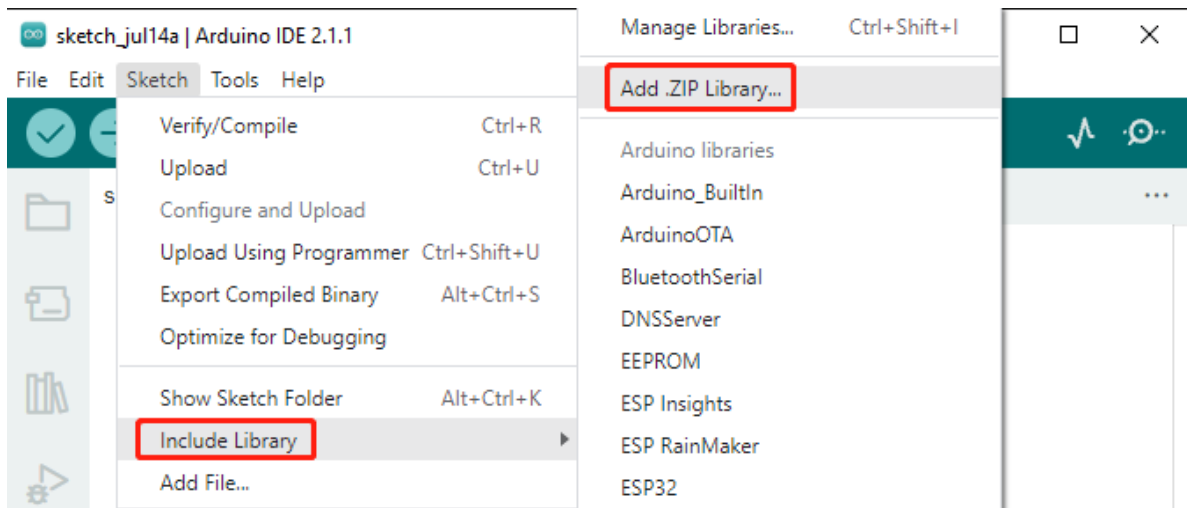
- SunFounder Kepler Kit Beispiel
- Oder schauen Sie sich den Code auf [Kepler Kit - GitHub](#) an.

### 6.4.1 Bibliotheken hinzufügen

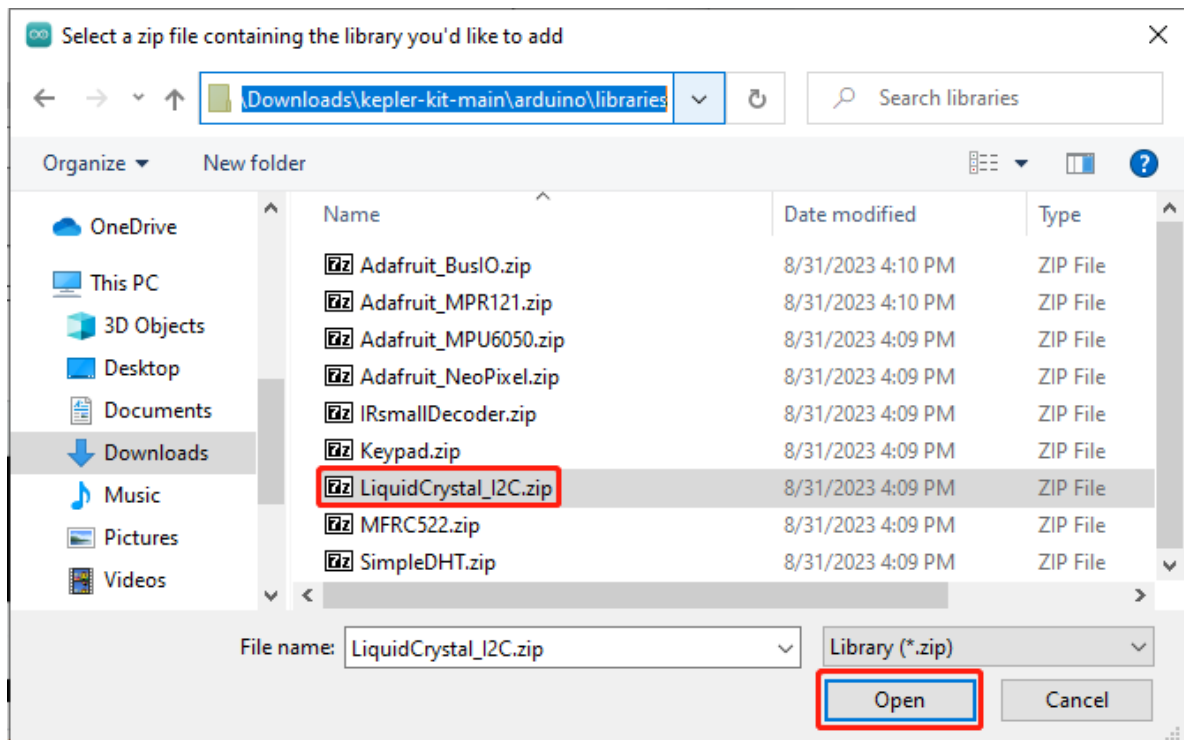
Eine Bibliothek, die einige Funktionsdefinitionen und Header-Dateien zusammenfasst, enthält normalerweise zwei Dateien: .h (Header-Datei, inklusive Funktionsdeklaration, Makrodefinition, Konstruktordefinition usw.) und .cpp (Ausführungsdatei, mit Funktionsimplementierung, Variablendefinition etc.). Wenn Sie eine Funktion aus einer solchen Bibliothek nutzen möchten, müssen Sie lediglich die entsprechende Header-Datei einfügen (z.B. `#include <dht.h>`) und dann die Funktion aufrufen. Dies macht Ihren Code kompakter. Falls Sie die Bibliothek nicht verwenden möchten, können Sie die Funktionsdefinition auch direkt im Code hinterlegen. Dies führt jedoch zu einem längeren und weniger übersichtlichen Code.

Einige Bibliotheken sind bereits in der Arduino IDE integriert, während andere erst hinzugefügt werden müssen. Sehen wir uns nun an, wie das funktioniert.

1. Öffnen Sie die Arduino IDE und navigieren Sie zu **Skizze -> Bibliothek einbinden -> .ZIP-Bibliothek hinzufügen**.



2. Navigieren Sie zu dem Verzeichnis, in dem sich die Bibliotheksdateien befinden, etwa dem Ordner `kepler-kit-main\arduino\libraries`, und wählen Sie die gewünschte Bibliotheksdatei aus, wie z.B. `LiquidCrystal_I2C.zip`. Klicken Sie dann auf **Öffnen**.



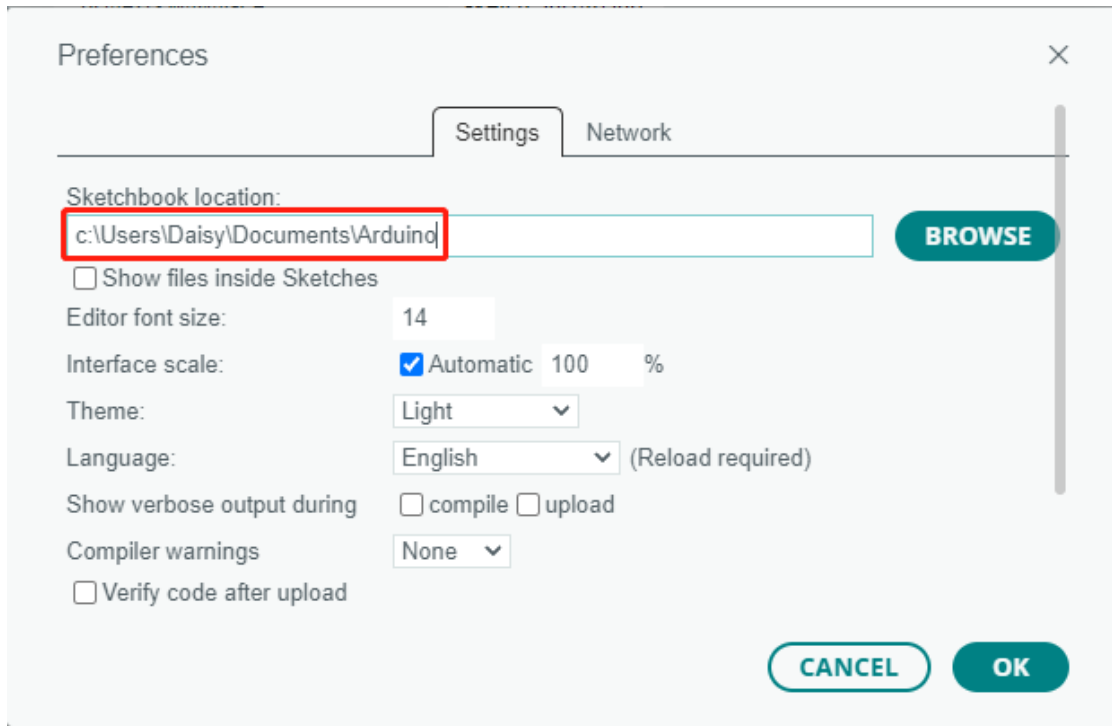
3. Nach kurzer Zeit erhalten Sie eine Benachrichtigung, die eine erfolgreiche Installation bestätigt.



4. Wiederholen Sie diesen Vorgang, um weitere Bibliotheken hinzuzufügen.

**Bemerkung:** Die installierten Bibliotheken finden Sie im Standardbibliotheksverzeichnis der Arduino IDE, das normalerweise unter C:\Users\xxx\Documents\Arduino\libraries zu finden ist.

Falls Ihr Bibliotheksverzeichnis abweicht, können Sie dieses überprüfen, indem Sie zu **Datei -> Einstellungen** navigieren.



## 2. Ausgabe & Eingabe

### 6.5 2.1 - Hallo, LED!

Genauso wie der Ausdruck „Hallo Welt!“ der erste Schritt beim Programmierenlernen ist, stellt das Ansteuern einer LED mittels Programm die klassische Einführung in die physische Programmierung dar.

- *LED*

#### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

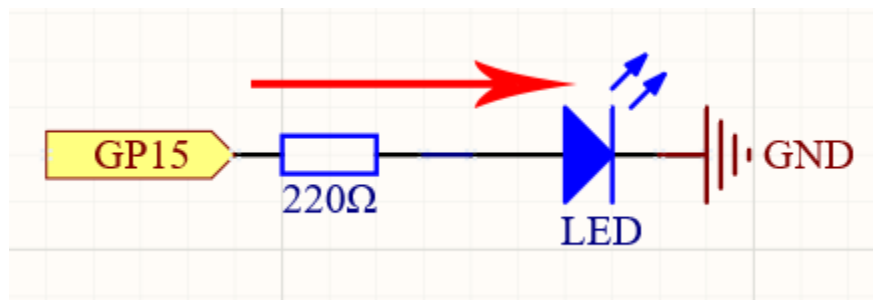
Ein Komplettsset ist definitiv praktisch, hier ist der Link dazu:

Name	ARTIKEL IN DIESEM SET	KAUF-LINK
Kepler-Set	450+	

Sie können die Bauteile auch einzeln über die folgenden Links erwerben.

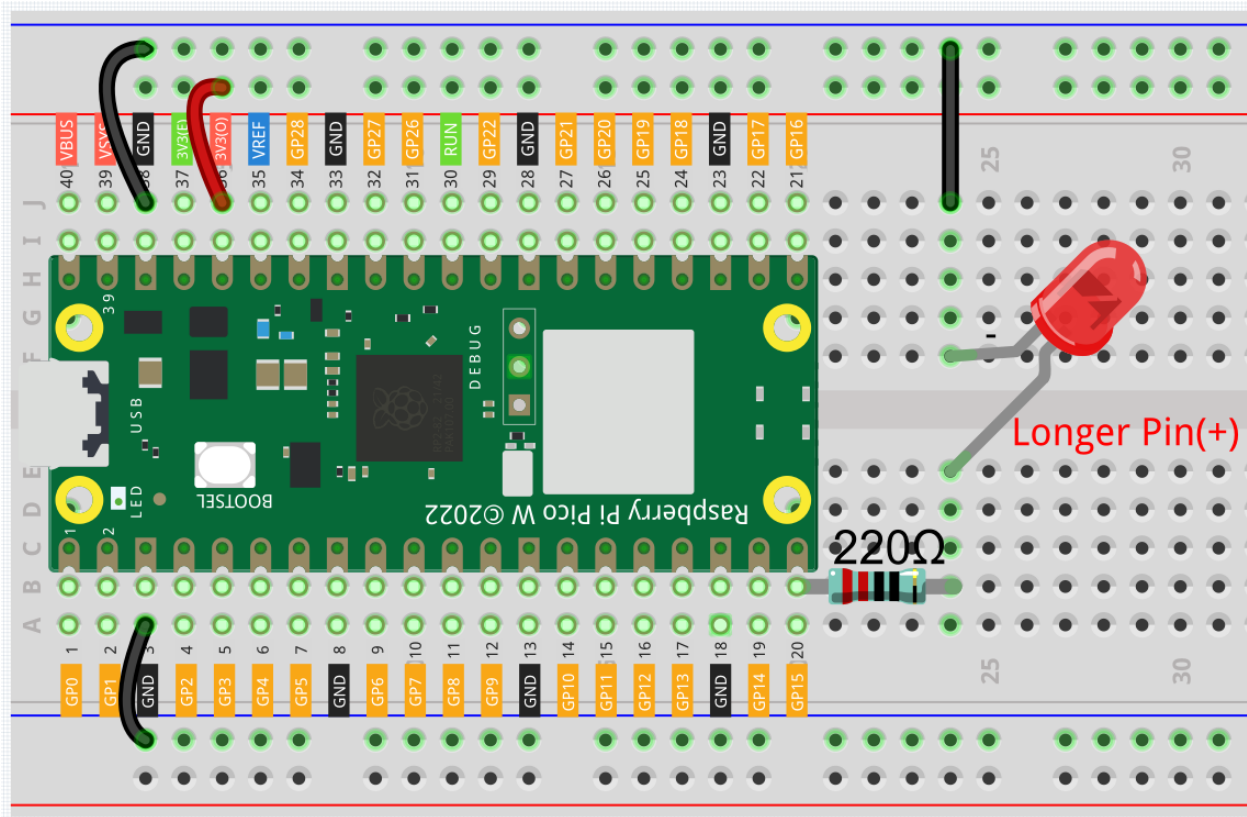
SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1 (220)	
6	<i>LED</i>	1	

### Schaltplan



Das Prinzip dieser Schaltung ist simpel und die Stromrichtung ist in der Abbildung dargestellt. Wenn GP15 ein hohes Signal (3,3 V) ausgibt, leuchtet die LED nach dem 220-Ohm-Vorwiderstand auf. Bei einem niedrigen Signal (0 V) geht die LED aus.

### Verdrahtung



Lassen Sie uns den Stromfluss folgen und die Schaltung aufbauen!

1. Wir verwenden das elektrische Signal vom GP15-Pin der Pico W-Platine, um die LED zum Leuchten zu bringen; hier beginnt die Schaltung.
2. Der Strom muss durch einen 220-Ohm-Widerstand fließen (zum Schutz der LED). Stecken Sie ein Ende des Widerstands (beliebiges Ende) in die gleiche Reihe wie den GP15-Pin des Pico W (Reihe 20 in meiner Schaltung) und das andere Ende in eine freie Reihe des Steckbretts (Reihe 24 in meiner Schaltung).
3. Nehmen Sie die LED; ein Bein ist länger als das andere. Stecken Sie das längere Bein in dieselbe Reihe wie das Ende des Widerstands und das kürzere Bein über die mittlere Lücke des Steckbretts in die gleiche Reihe.
4. Stecken Sie das Stecker-zu-Stecker-Kabel (M2M) in dieselbe Reihe wie das kurze Bein der LED und verbinden Sie es mit der negativen Stromschiene des Steckbretts.
5. Verbinden Sie die negative Stromschiene mit dem GND-Pin des Pico W.

## Code

### Bemerkung:

- Sie können die Datei `2.1_hello_led.ino` im Pfad `kepler-kit-main/arduino/2.1_hello_led` öffnen.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, die Platine (Raspberry Pi Pico) und den richtigen Anschluss auszuwählen, bevor Sie auf **Hochladen** klicken.

Nachdem der Code ausgeführt wurde, wird die LED blinken.

**Wie funktioniert es?**

Hier schließen wir die LED an den digitalen Pin 15 an, daher müssen wir zu Beginn des Programms eine int-Variable namens ledPin deklarieren und den Wert 15 zuweisen.

```
const int ledPin = 15;
```

Jetzt initialisieren Sie den Pin in der setup()-Funktion, wo Sie den Pin auf den OUTPUT-Modus setzen müssen.

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

In der loop()-Funktion wird digitalWrite() verwendet, um ein 3,3-V-Hochpegelsignal für ledPin bereitzustellen, was eine Spannungsdifferenz zwischen den LED-Pins erzeugt und die LED zum Leuchten bringt.

```
digitalWrite(ledPin, HIGH);
```

Wenn das Pegelsignal auf LOW geändert wird, wird das Signal von ledPin auf 0 V zurückgesetzt, um die LED auszuschalten.

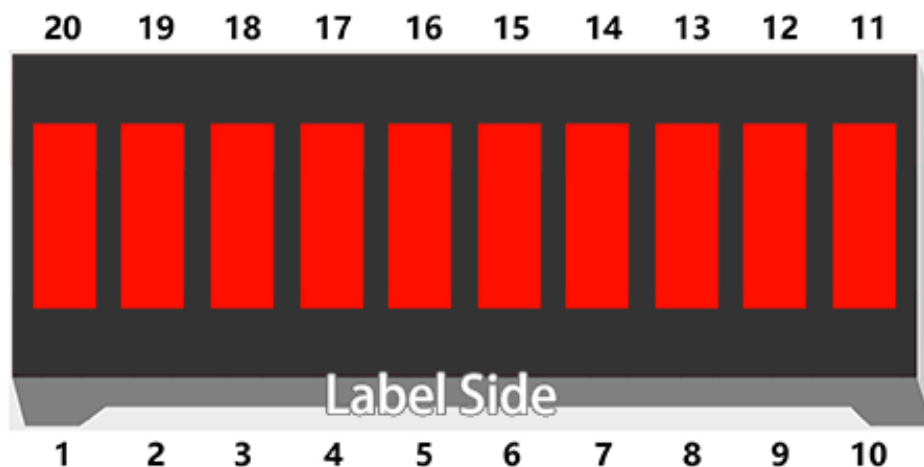
```
digitalWrite(ledPin, LOW);
```

Für einen sichtbaren Wechsel zwischen Ein- und Ausschalten ist eine Verzögerung notwendig, daher verwenden wir den Befehl delay(1000), um den Controller für 1000 ms inaktiv zu halten.

```
delay(1000);
```

## 6.6 2.2 - Pegelanzeige

Nachdem das erste Projekt lediglich das Blinken einer einzelnen LED zum Ziel hatte, wollen wir uns nun dem LED-Balkendiagramm zuwenden. Dieses besteht aus einer Serie von 10 LEDs in einem Kunststoffgehäuse und dient in der Regel zur Darstellung von Leistungs- oder Lautstärkepegeln.



- *LED-Balkendiagramm*

### Erforderliche Bauteile

Für dieses Projekt benötigen wir die folgenden Komponenten.

Es ist sicherlich bequem, gleich ein komplettes Kit zu kaufen. Hier der entsprechende Link:

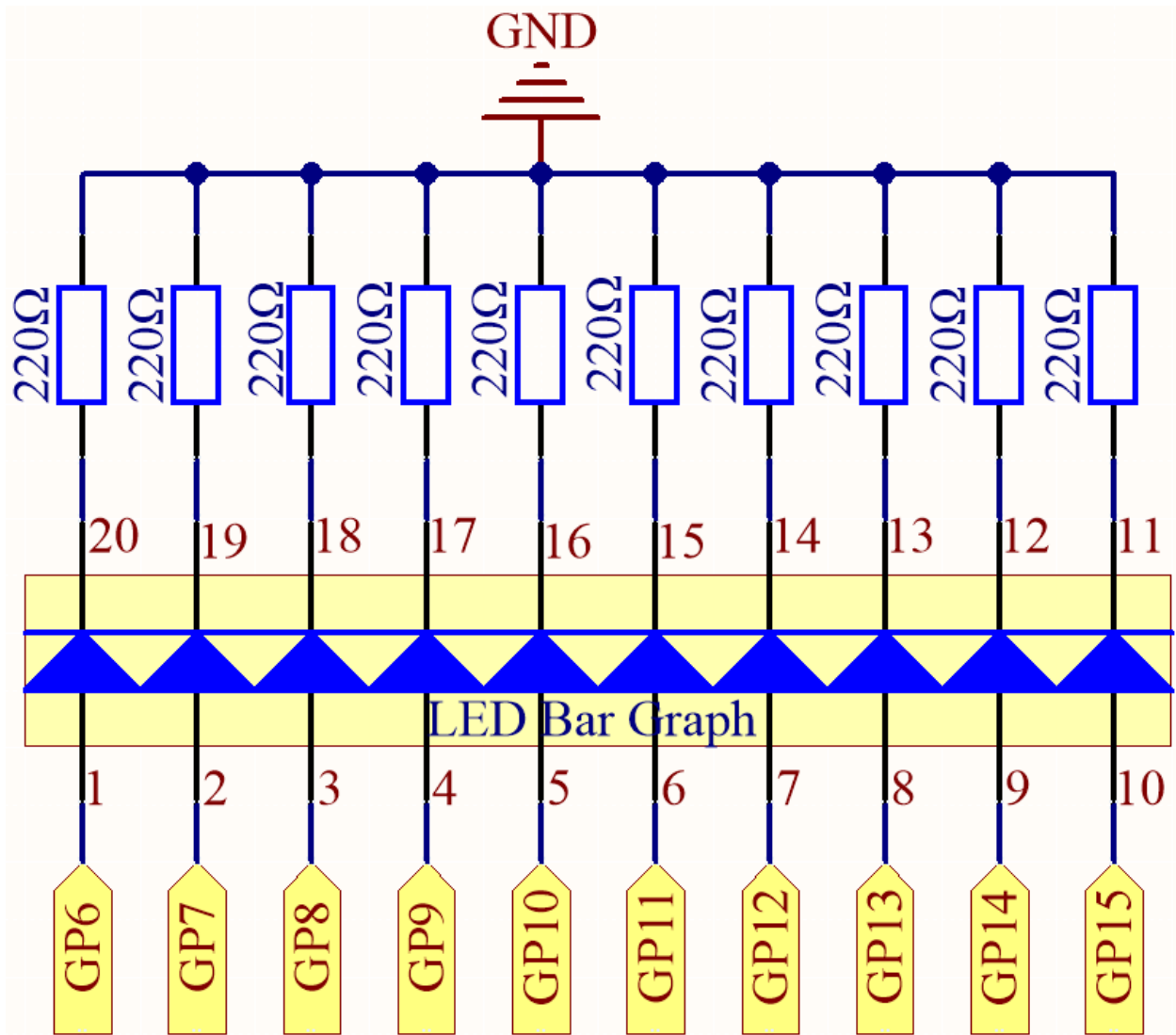


Bezeichnung	ARTIKEL IM SET	KAUF-LINK
Kepler Kit	450+	

Die Bauteile können jedoch auch einzeln über die unten aufgeführten Links erworben werden.

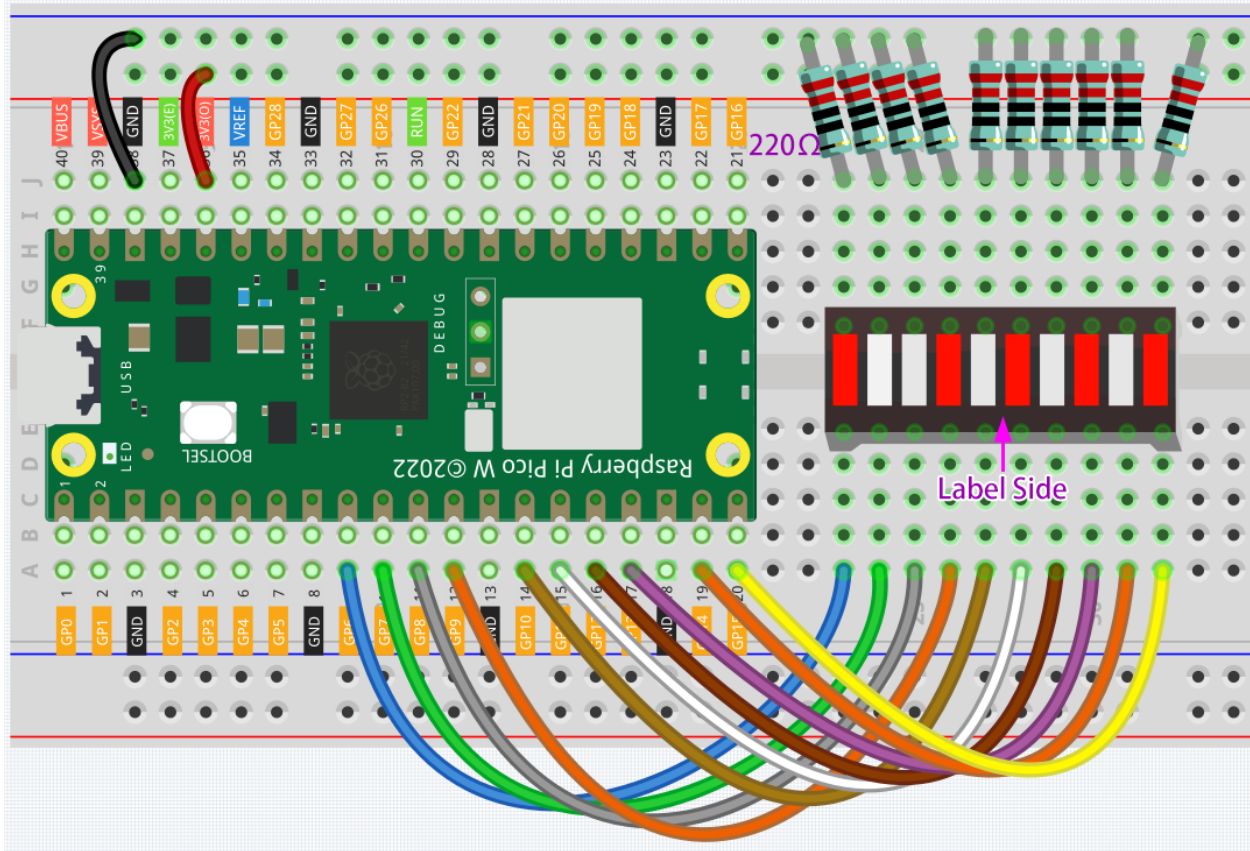
Nr.	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	10 (220)	
6	<i>LED-Balkendiagramm</i>	1	

### Schaltplan



Das LED-Balkendiagramm besteht aus 10 einzeln ansteuerbaren LEDs. Dabei ist die Anode jeder dieser LEDs an die Pins GP6 bis GP15 angeschlossen. Die Kathode ist jeweils über einen 220-Ohm-Widerstand mit GND verbunden.

#### Verkabelung



## Programmcode

### Bemerkung:

- Sie können die Datei 2.2\_display\_the\_level.ino im Verzeichnis kepler-kit-main/arduino/2.2\_display\_the\_level öffnen.
- Oder Sie kopieren den Code in die **Arduino IDE**.
- Vergessen Sie nicht, vor dem Hochladen das richtige Board (Raspberry Pi Pico) und den passenden Port auszuwählen.

Sobald das Programm läuft, werden Sie feststellen, dass die LEDs im LED-Balkendiagramm nacheinander aufleuchten und erlöschen.

### Funktionsweise

Jede der zehn LEDs im LED-Balkendiagramm wird durch einen eigenen Pin gesteuert. Das bedeutet, dass wir diese zehn Pins zuerst definieren müssen.

Im Abschnitt `setup()` wird eine For-Schleife verwendet, um die Pins 6 bis 15 nacheinander als Ausgang (OUTPUT) zu initialisieren.

```
for(int i=6;i<=15;i++)
{
    pinMode(i,OUTPUT);
}
```

In der `loop()`-Funktion wird ebenfalls eine For-Schleife verwendet, um die LEDs sequenziell blinken zu lassen (0,5 Sekunden ein, dann 0,5 Sekunden aus).

```
for(int i=6;i<=15;i++)
{
    digitalWrite(i,HIGH);
    delay(500);
    digitalWrite(i,LOW);
    delay(500);
}
```

### 6.7 2.3 - Verblässende LED

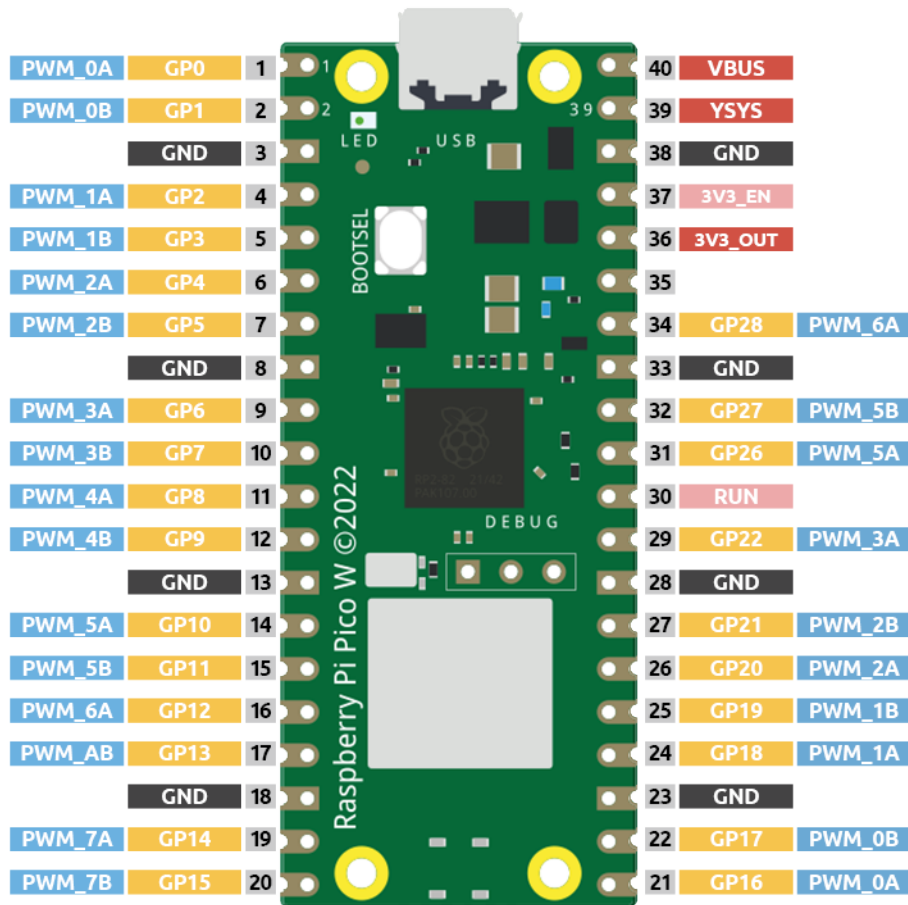
Bislang haben wir lediglich zwei Ausgangssignale verwendet: hohes und niedriges Signal (oder auch als 1 & 0, EIN & AUS bezeichnet), was als digitale Ausgabe bezeichnet wird. In der Praxis funktionieren jedoch viele Geräte nicht einfach durch Ein- oder Ausschalten, etwa bei der Geschwindigkeitsregelung eines Motors oder der Helligkeitsregulierung einer Schreibtischlampe. Früher wurde ein regelbarer Widerstand genutzt, um diese Ziele zu erreichen, was jedoch stets unzuverlässig und ineffizient war. Deshalb hat sich die Pulsweitenmodulation (PWM) als praktikable Lösung für solche komplexen Probleme etabliert.

Ein digitales Ausgangssignal, das aus einem hohen und einem niedrigen Signal besteht, wird als Puls bezeichnet. Die Pulsdauer dieser Pins kann durch Änderung der Ein-/Ausschaltgeschwindigkeit angepasst werden.

Kurz gesagt, wenn wir in einer kurzen Zeitspanne (wie etwa 20 ms, die durchschnittliche visuelle Verweilzeit der meisten Menschen) die LED einschalten, ausschalten und wieder einschalten, werden wir nicht bemerken, dass sie ausgeschaltet wurde, jedoch wird die Helligkeit etwas geringer sein. Je länger die LED in diesem Zeitraum eingeschaltet ist, desto heller wird sie sein. Mit anderen Worten, je breiter der Puls im Zyklus ist, desto größer ist die „elektrische Signalstärke“, die vom Mikrocontroller ausgegeben wird. So steuert PWM die Helligkeit der LED (oder die Geschwindigkeit des Motors).

- [Pulsweitenmodulation – Wikipedia](#)

Beim Einsatz von PWM mit dem Pico W gibt es einige Punkte zu beachten. Werfen wir einen Blick auf dieses Bild.



Jeder GPIO-Pin des Pico W unterstützt PWM, tatsächlich stehen jedoch insgesamt nur 16 unabhängige PWM-Ausgänge zur Verfügung (statt 30), die zwischen GP0 bis GP15 auf der linken Seite verteilt sind, und der PWM-Ausgang der rechten GPIO ist dem der linken Seite gleichwertig.

Worauf wir achten müssen, ist, denselben PWM-Kanal während der Programmierung nicht für verschiedene Zwecke einzusetzen. (Zum Beispiel sind GP0 und GP16 beide PWM\_0A)

Nachdem wir dieses Wissen erworben haben, versuchen wir nun, den Effekt einer verblassenden LED zu erzielen.

- *LED*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

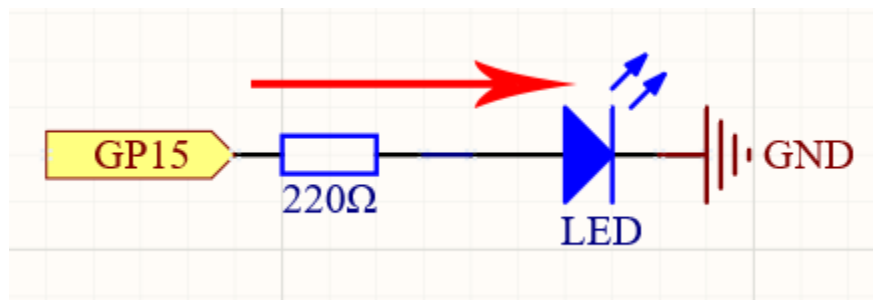
Es ist sicherlich praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IM SET	KAUF-LINK
Kepler Kit	450+	

Sie können die Komponenten auch einzeln über die folgenden Links kaufen.

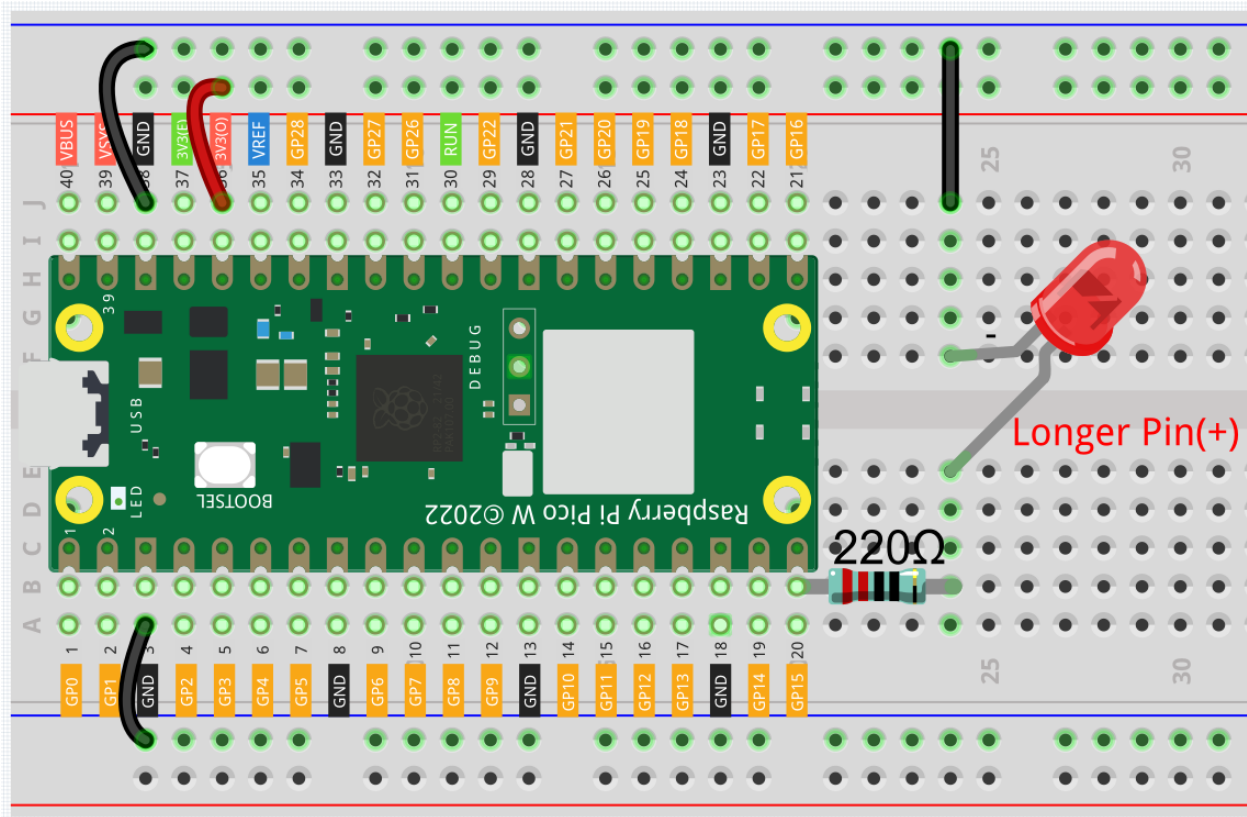
SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1 (220)	
6	<i>LED</i>	1	

### Schaltplan



Dieses Projekt nutzt denselben Schaltkreis wie das erste Projekt [2.1 - Hallo, LED!](#), allerdings mit einem unterschiedlichen Signaltyp. Im ersten Projekt wurden digitale Hoch- und Niedrigpegel (0&1) direkt von GP15 ausgegeben, um die LEDs ein- oder auszuschalten. In diesem Projekt wird ein PWM-Signal von GP15 ausgegeben, um die Helligkeit der LED zu steuern.

### Verdrahtung



## Code

### Bemerkung:

- Die Datei `2.3_fading_led.ino` können Sie im Pfad `kepler-kit-main/arduino/2.3_fading_led` finden.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf den **Upload**-Button klicken.

Mit der Ausführung des Programms wird die LED allmählich heller.

### Funktionsweise

Pin 15 wird als `ledPin` deklariert.

```
const int ledPin = 15;
```

`analogWrite()` weist im `loop()` dem `ledPin` einen analogen Wert (PWM-Welle) zwischen 0 und 255 zu, um die Helligkeit der LED zu ändern.

```
analogWrite(ledPin, value);
```

Mithilfe einer `for`-Schleife kann der Wert von `analogWrite()` schrittweise zwischen dem Minimalwert (0) und dem Maximalwert (255) geändert werden.

```
for (int value = 0 ; value <= 255; value += 5) {  
    analogWrite(ledPin, value);  
}
```

Um das experimentelle Phänomen deutlich zu sehen, muss der for-Schleife ein `delay(30)` hinzugefügt werden, um die Zeit der Helligkeitsänderung zu steuern.

```
for (int value = 0 ; value <= 255; value += 5) {  
    analogWrite(ledPin, value);  
    delay(30);  
}
```

## 6.8 2.4 - Farbenfrohes Licht

Wie wir wissen, kann Licht überlagert werden. Zum Beispiel ergibt die Mischung aus blauem und grünem Licht Zyan-Licht, rotes und grünes Licht ergibt gelbes Licht. Dies wird als „additive Farbmischung“ bezeichnet.

- [Additive Farbmischung - Wikipedia](#)

Basierend auf dieser Methode können wir mit den drei Grundfarben Licht in jeder sichtbaren Farbe erzeugen, je nach spezifischem Mischungsverhältnis. Zum Beispiel lässt sich Orange durch mehr Rot und weniger Grün erzeugen.

In diesem Kapitel werden wir die Geheimnisse der additiven Farbmischung mit einer RGB-LED ergründen!

Eine RGB-LED ist im Grunde eine Kapselung einer roten, einer grünen und einer blauen LED unter einer Lampenkappe; alle drei LEDs teilen sich einen gemeinsamen Kathodenpin. Da jedem Anodenpin ein elektrisches Signal zugeführt wird, kann das Licht der entsprechenden Farbe angezeigt werden. Durch Veränderung der elektrischen Signalstärke an jedem Anodenpin können diverse Farben erzeugt werden.

- [RGB-LED](#)

### Benötigte Komponenten

Für dieses Projekt werden die folgenden Komponenten benötigt.

Ein Komplettsset ist natürlich praktisch, hier ist der Link dazu:

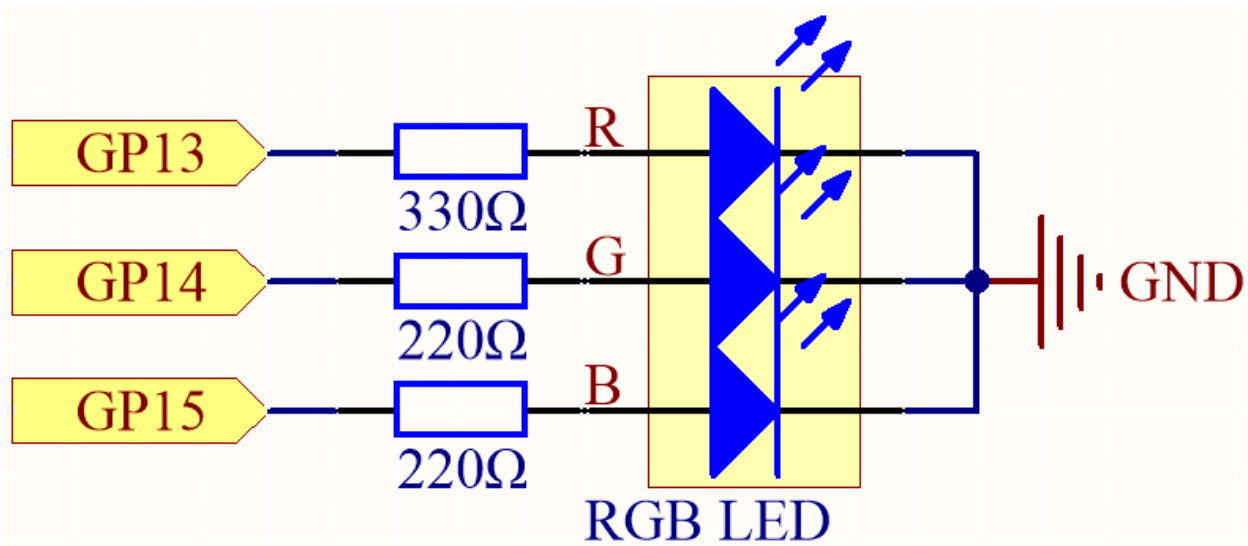
Name	ARTIKEL IM SET	KAUF-LINK
Kepler Kit	450+	

Sie können die Teile auch einzeln über die folgenden Links kaufen.



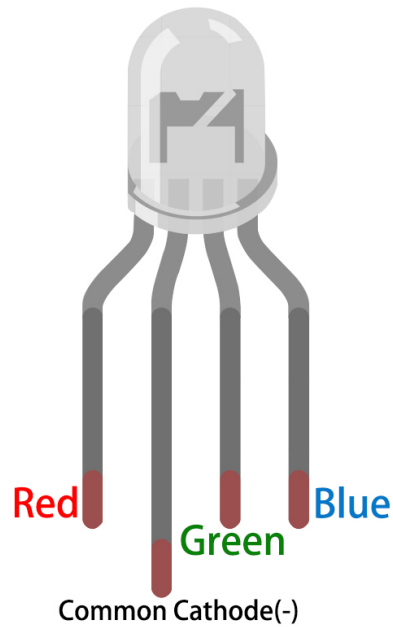
SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	3(1-330, 2-220)	
6	<i>RGB-LED</i>	1	

### Schaltplan

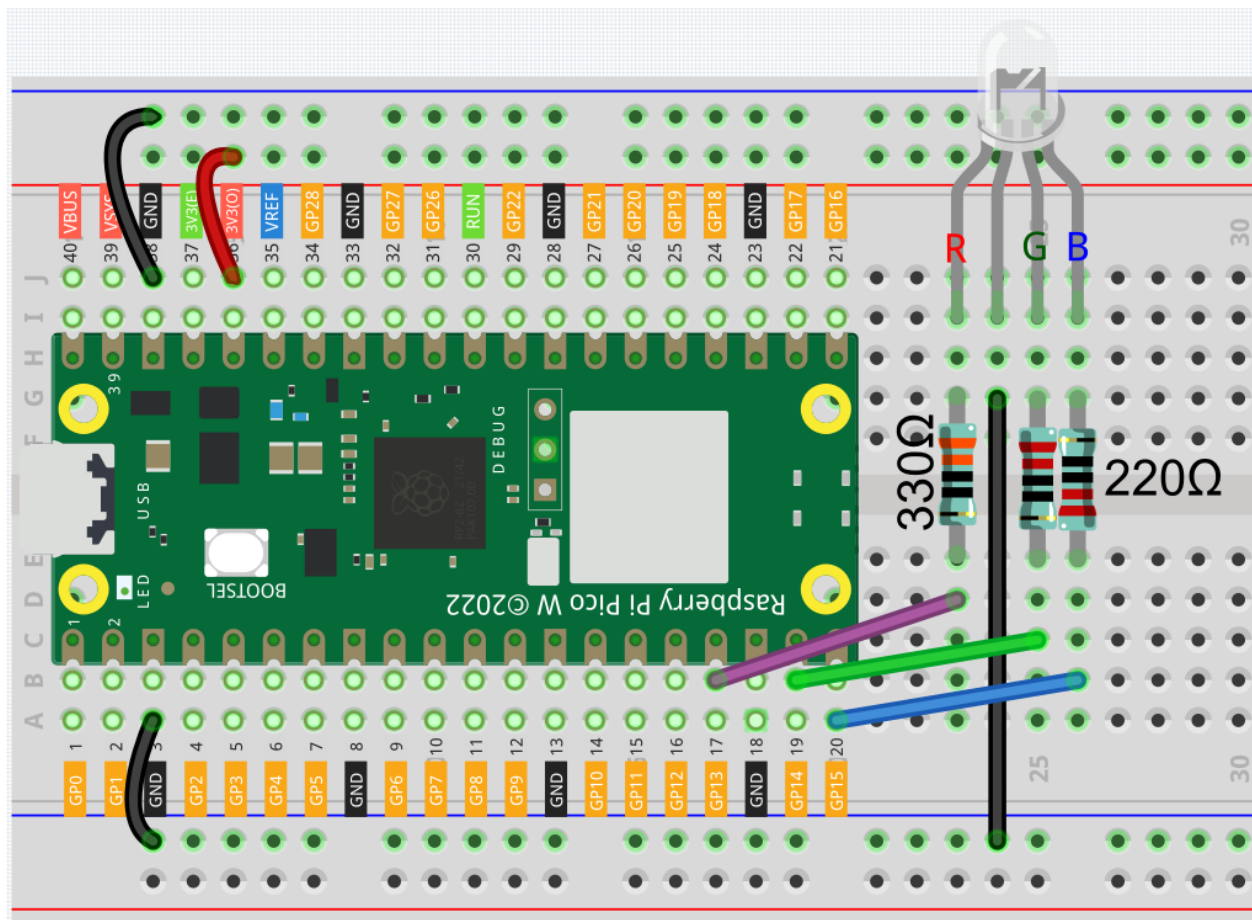


Die PWM-Pins GP13, GP14 und GP15 steuern die roten, grünen und blauen Pins der RGB-LED. Der gemeinsame Kathodenpin ist mit GND verbunden. So kann die RGB-LED durch Überlagerung des Lichts an diesen Pins mit unterschiedlichen PWM-Werten eine spezifische Farbe anzeigen.

### Verkabelung



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

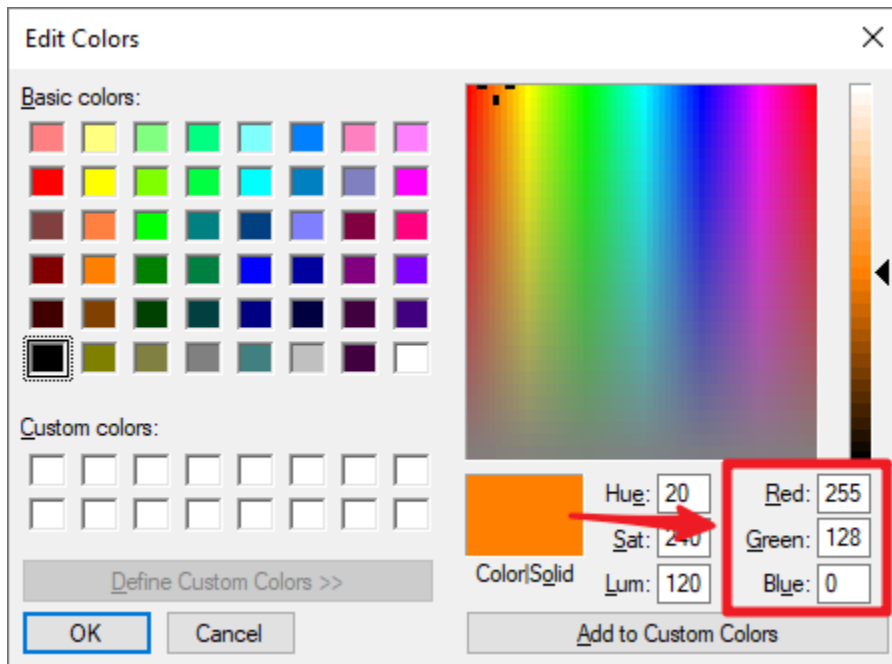


Code

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

**Bemerkung:**

- Sie können die Datei `2.4_colorful_light.ino` im Verzeichnis `kepler-kit-main/arduino/2.4_colorful_light` öffnen.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf die **Upload**-Schaltfläche klicken.



Tragen Sie den RGB-Wert in `color_set()` ein, dann wird die RGB-LED die gewünschten Farben leuchten.

**Funktionsweise**

In diesem Beispiel ist die Funktion zum Zuweisen von Werten an die drei Pins der RGB-LED in einer eigenständigen Unterfunktion `color()` verpackt.

```
void color (unsigned char red, unsigned char green, unsigned char blue)
{
    analogWrite(redPin, red);
    analogWrite(greenPin, green);
    analogWrite(bluePin, blue);
}
```

In `loop()`, dient der RGB-Wert als Eingabeargument, um die Funktion `color()` aufzurufen und damit die RGB-LED in verschiedenen Farben leuchten zu lassen.

```
void loop()
{
    color(255, 0, 0); // Rot
    delay(1000);
}
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

color(0, 255, 0); // Grün
delay(1000);
color(0, 0, 255); // Blau
delay(1000);
}

```

## 6.9 2.5 - Tastenwert auslesen

Anhand der Bezeichnung GPIO (General-purpose input/output) lässt sich erkennen, dass diese Pins sowohl Eingabe- als auch Ausgabefunktionen haben. In den vorherigen Lektionen haben wir die Ausgabefunktion verwendet, in diesem Kapitel werden wir die Eingabefunktion nutzen, um den Wert der Taste auszulesen.

- *Taster*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

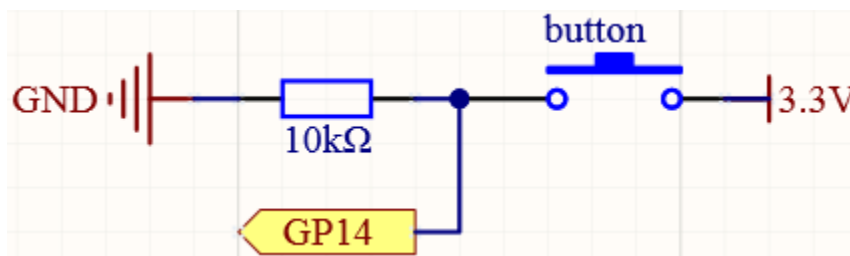
Ein komplettes Kit zu kaufen ist definitiv praktisch, hier ist der Link:

Bezeichnung	ELEMENTE IN DIESEM KIT	KAUF-LINK
Kepler-Kit	450+	

Die Teile können auch einzeln über die folgenden Links erworben werden.

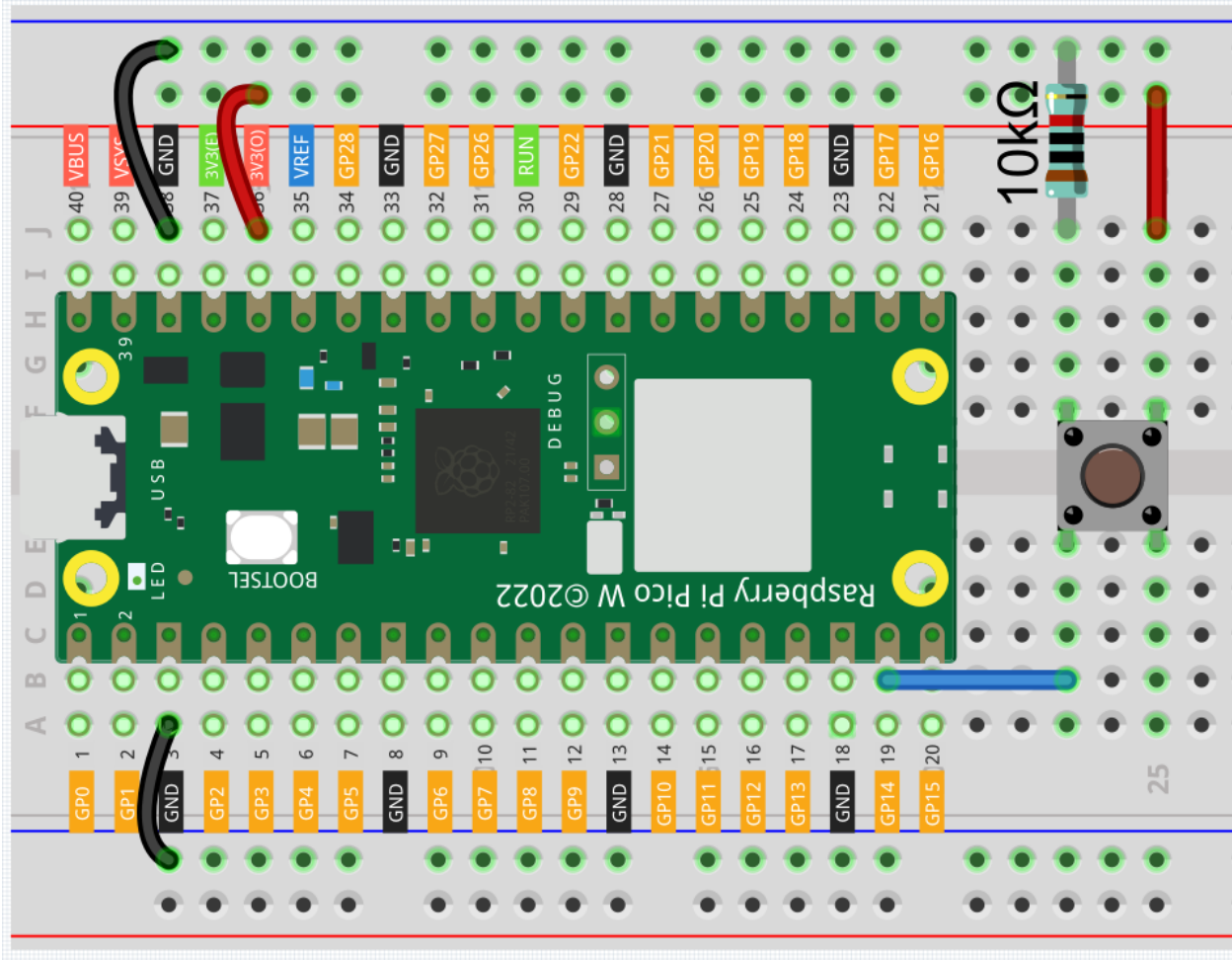
SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Taster</i>	1	

### Schaltplan



Ein Anschluss des Tastenpins ist mit 3,3V verbunden, und der andere Anschluss ist mit GP14 verbunden. Wenn die Taste gedrückt wird, wird GP14 auf „High“ gesetzt. Ist die Taste jedoch nicht gedrückt, befindet sich GP14 in einem schwebenden Zustand und könnte sowohl „High“ als auch „Low“ sein. Um einen stabilen „Low“-Zustand zu erhalten, wenn die Taste nicht gedrückt ist, muss GP14 über einen 10K-Pull-down-Widerstand erneut mit GND verbunden werden.

### Verkabelung



**Bemerkung:** Man kann den vierbeinigen Taster als H-förmigen Taster betrachten. Seine linken (rechten) beiden Beine sind miteinander verbunden, was bedeutet, dass er nach Überqueren der mittleren Trennlinie die beiden halben Reihen derselben Reihenummer verbindet. (Beispielsweise sind in meiner Schaltung E23 und F23 verbunden, ebenso wie E25 und F25).

Bevor die Taste gedrückt wird, sind die linke und rechte Seite voneinander unabhängig, und der Strom kann nicht von einer Seite zur anderen fließen.

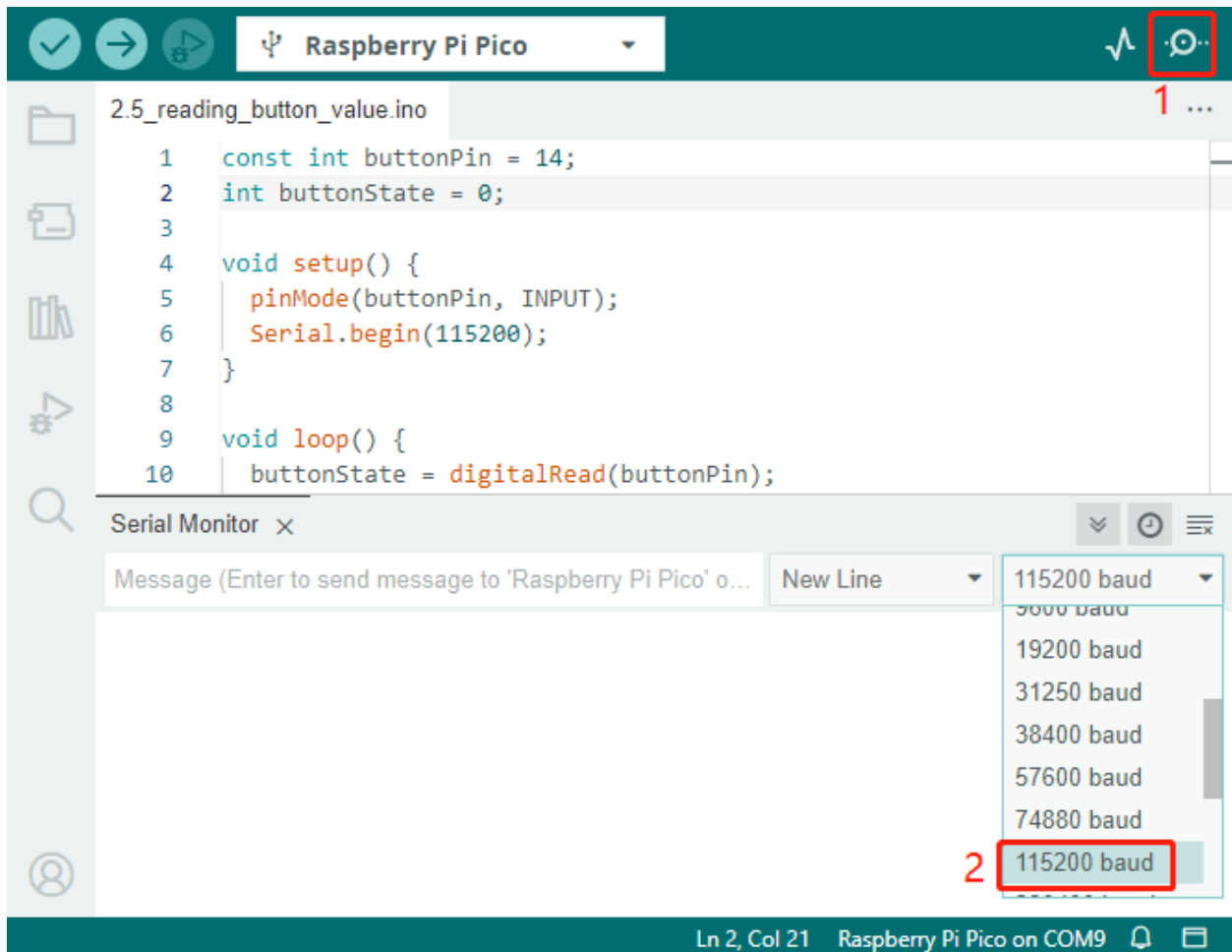
### Code

#### Bemerkung:

- Sie können die Datei `2.5_reading_button_value.ino` unter dem Pfad `kepler-kit-main/arduino/2.5_reading_button_value` öffnen.

- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, die Platine (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.

Nachdem der Code ausgeführt wurde, klicken Sie auf das Lupensymbol in der oberen rechten Ecke der Arduino IDE (Serial Monitor).



Jetzt, wenn Sie den Knopf drücken, wird im Serial Monitor „Sie haben den Knopf gedrückt!“ angezeigt.

### Wie funktioniert das?

Um den Serial Monitor zu aktivieren, müssen Sie die serielle Kommunikation in `setup()` starten und die Datenrate auf 9600 einstellen.

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

- `Serial`

Für den Knopf müssen wir ihren Modus auf `INPUT` setzen, um ihre Werte abrufen zu können.

```
pinMode(buttonPin, INPUT);
```

Lesen Sie den Status von `buttonPin` in `loop()` und weisen Sie ihn der Variablen `buttonState` zu.

```
buttonState = digitalRead(buttonPin);
```

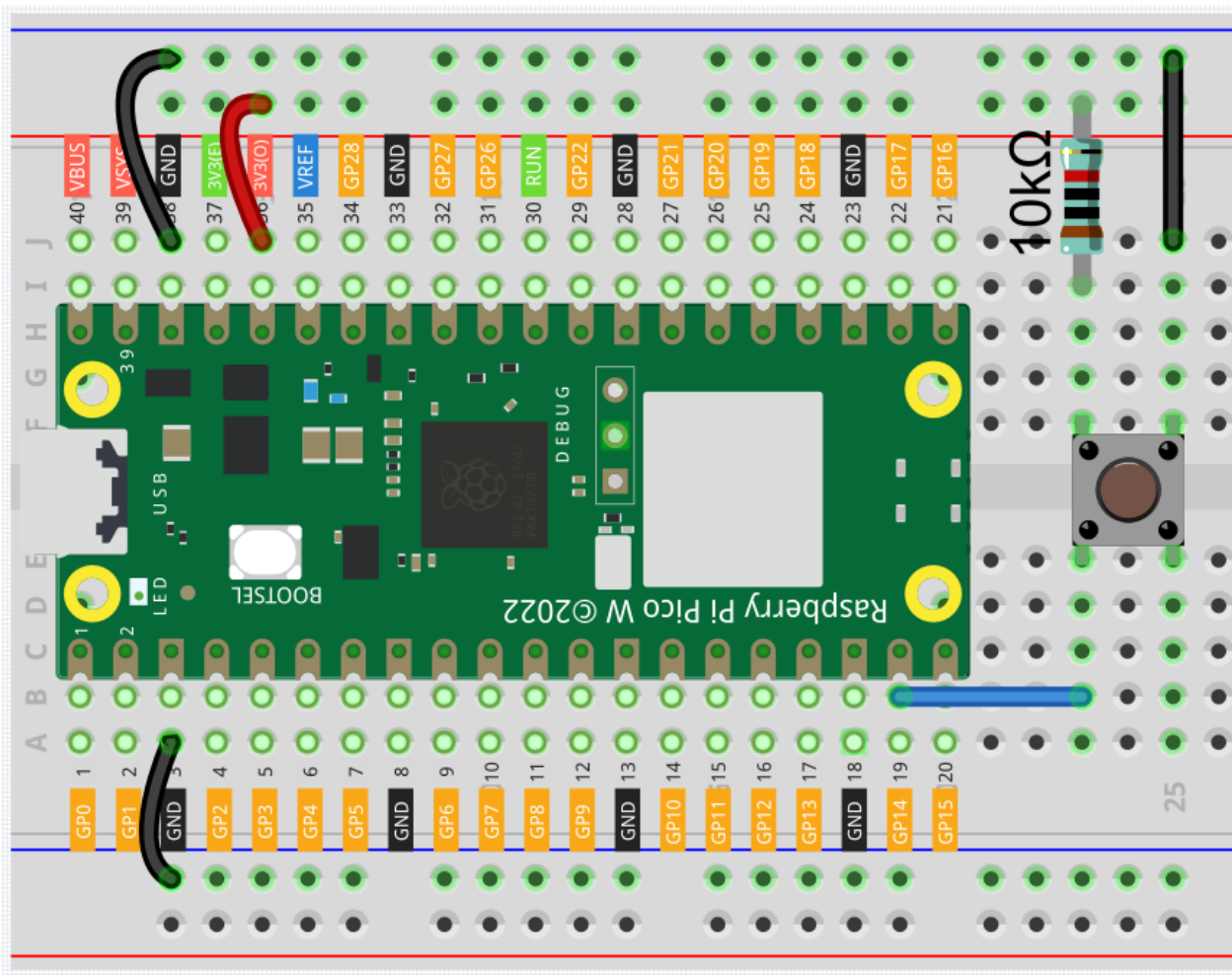
- `digitalRead()`

Wenn der `buttonState` HIGH ist, wird die LED blinken. Im Serial Monitor wird „You pressed the button!“ angezeigt.

```
if (buttonState == HIGH) {
    Serial.println("You pressed the button!");
}
```

### Pull-up Arbeitsmodus

Als nächstes folgt die Verdrahtung und der Code, wenn der Knopf im Pull-up-Arbeitsmodus ist, probieren Sie es bitte aus.



Der einzige Unterschied, den Sie im Vergleich zum Pull-down-Modus sehen werden, ist, dass der 10K-Widerstand mit 3,3V verbunden ist und der Knopf mit GND verbunden ist. Wenn der Knopf also gedrückt wird, erhält GP14 ein niedriges Signal, was das Gegenteil des im Pull-down-Modus erhaltenen Wertes ist. Ändern Sie diesen Code also zu `if (buttonState == LOW)`.

## 6.10 2.6 - Kipp es!



Der Kippschalter ist ein 2-poliges Bauelement mit einer Metallkugel im Inneren. In aufrechter Position sind die beiden Anschlüsse verbunden; neigt man den Schalter, werden die Anschlüsse getrennt.

### Erforderliche Bauteile

Für dieses Projekt werden die folgenden Komponenten benötigt.

Ein Komplettsset ist definitiv praktisch, hier der Link dazu:

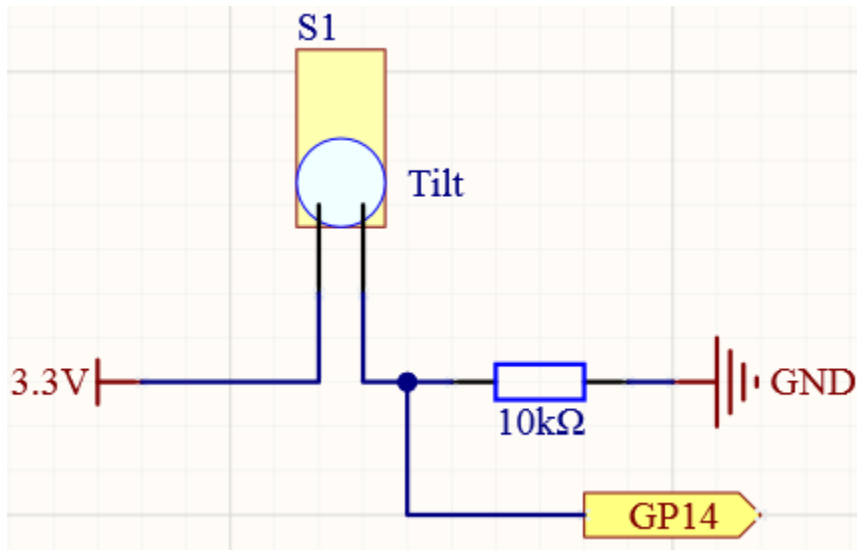
Bezeichnung	INHALT DES KITS	KAUF-LINK
Kepler Kit	450+	

Die Komponenten können auch einzeln über die untenstehenden Links gekauft werden.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<a href="#">Raspberry Pi Pico W</a>	1	
2	Micro-USB-Kabel	1	
3	<a href="#">Steckbrett</a>	1	
4	<a href="#">Jumperkabel</a>	Mehrere	
5	<a href="#">Widerstand</a>	1 (10K)	
6	<a href="#">Neigungsschalter</a>	1	

### Schaltplan



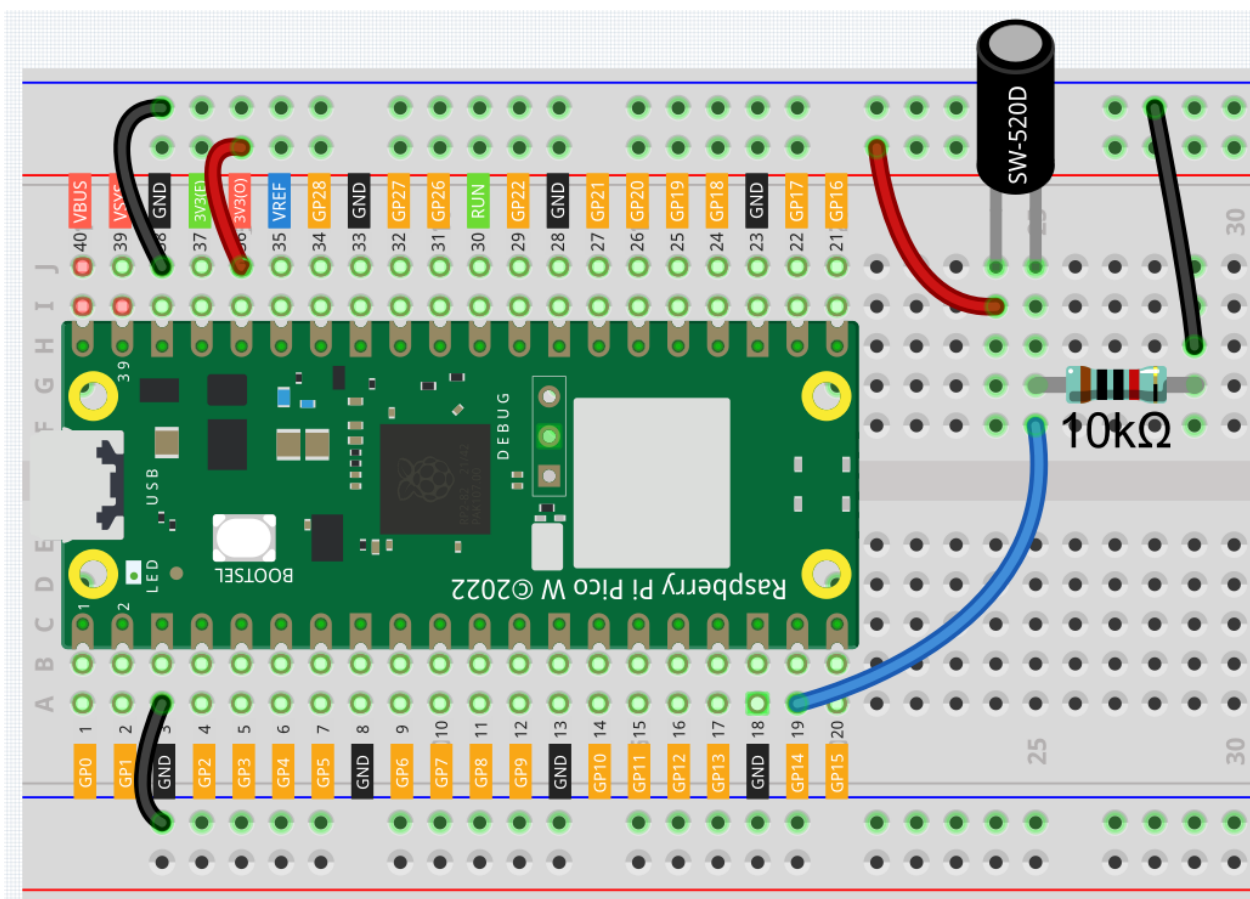


In aufrechter Position wird GP14 auf High gesetzt; kippt man den Schalter, wechselt GP14 auf Low.

Der 10K-Widerstand dient dazu, GP14 im gekippten Zustand stabil auf Low zu halten.

- *Neigungsschalter*

#### Verkabelung



#### Code

### Bemerkung:

- Die Datei `2.6_tilt_it.ino` finden Sie unter dem Pfad `kepler-kit-main/arduino/2.4_colorful_light`.
- Alternativ können Sie diesen Code in die **Arduino IDE** kopieren.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf **Hochladen** klicken.

Nach dem Hochladen des Programms erscheint in der Konsole die Meldung „Der Schalter funktioniert!“, wenn Sie das Breadboard (Kippschalter) kippen.

## 6.11 2.7 - Links und Rechts Umschalten



Der Schiebeschalter ist ein 3-poliges Bauteil. Der mittlere Pin (Pin 2) dient als gemeinsamer Anschluss. Wenn der Schalter nach links geschoben wird, werden die beiden linken Pins miteinander verbunden. Bei Verschiebung nach rechts werden die beiden rechten Pins verbunden.

### Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

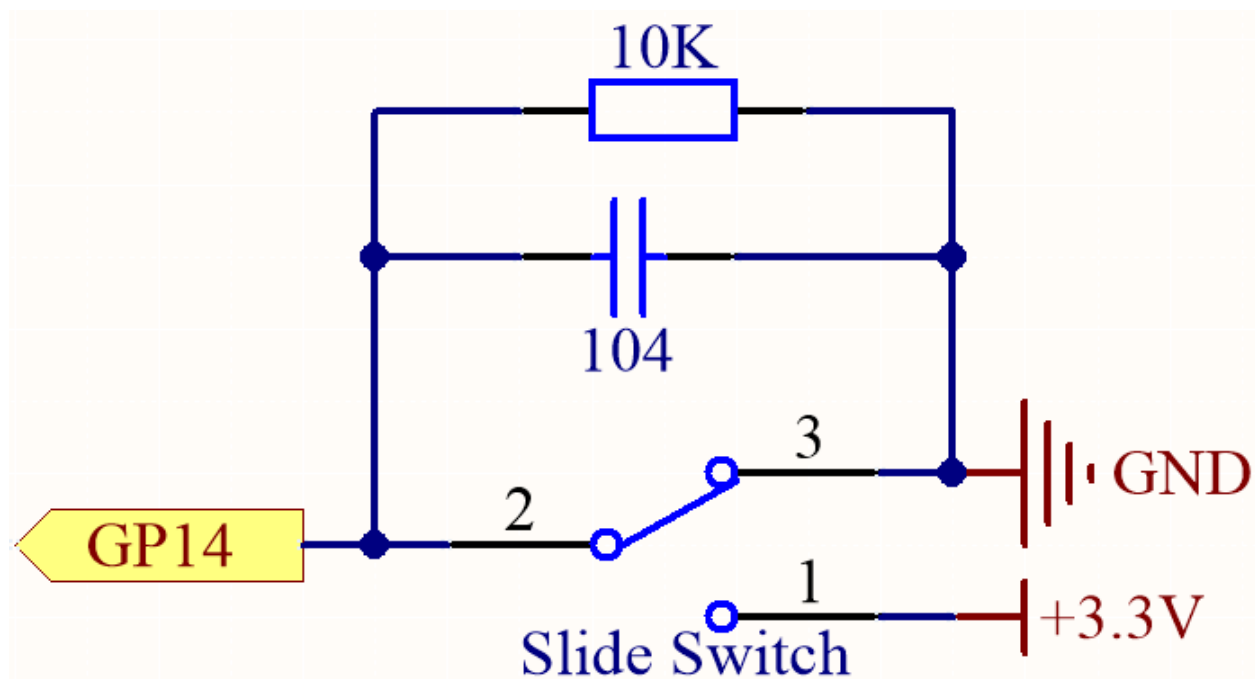
Ein Gesamtpaket zu kaufen ist definitiv praktisch, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	KAUF-LINK
Kepler Kit	450+	

Sie können die Komponenten auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Kondensator</i>	1(104)	
7	<i>Schiebeschalter</i>	1	

### Schaltplan



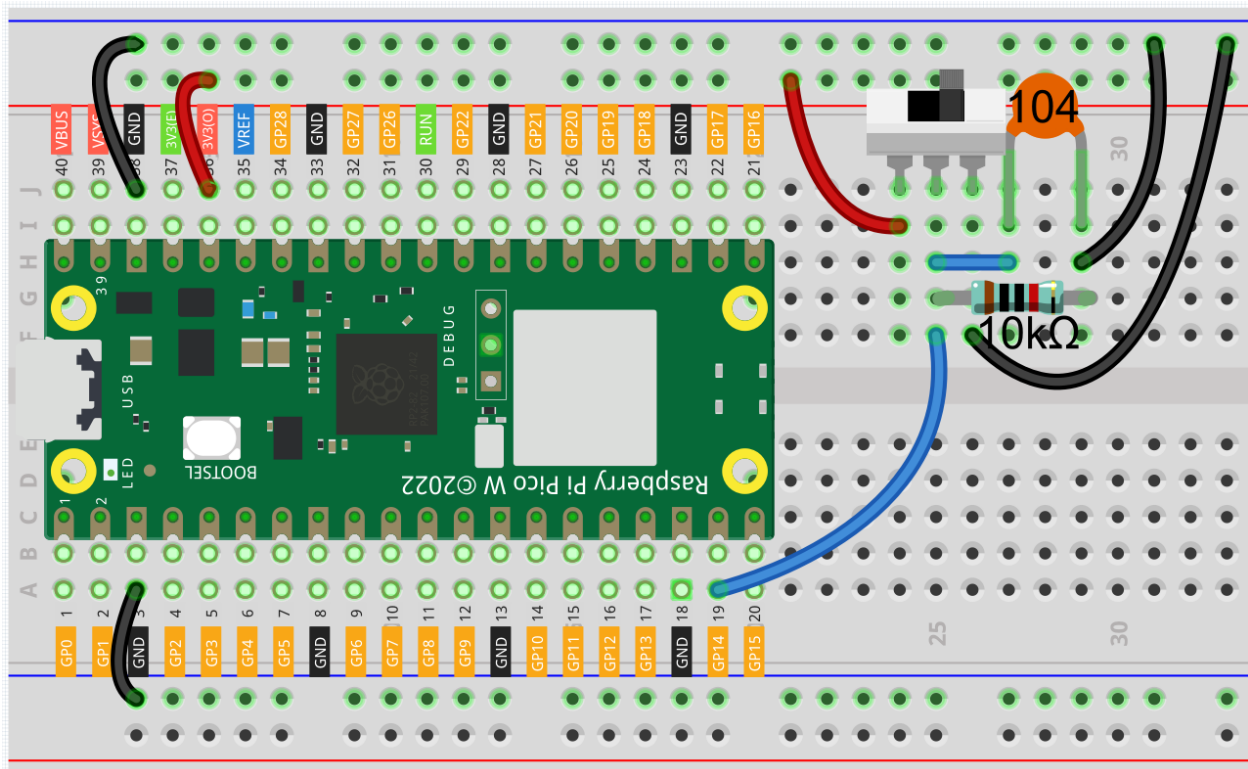
GP14 erhält ein unterschiedliches Signalniveau, je nachdem, ob der Schiebeschalter nach rechts oder links verschoben wird.

Der Zweck des 10K-Widerstands besteht darin, GP14 während des Umschaltens auf einem niedrigen Pegel zu halten (nicht ganz links und nicht ganz rechts).

Der 104-Keramikkondensator dient hier zur Eliminierung von Störungen.

- *Schiebeschalter*
- *Kondensator*

### Verdrahtung



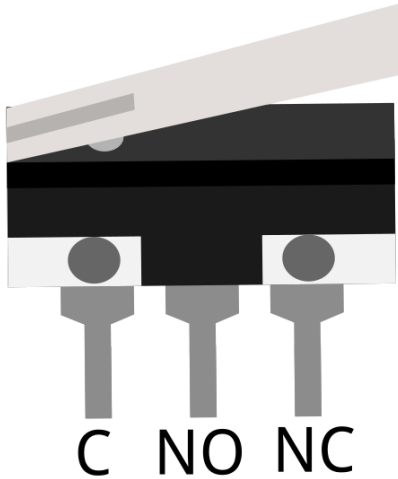
## Code

### Bemerkung:

- Die Datei 2.7\_toggle\_left\_right.ino finden Sie unter dem Pfad kepler-kit-main/arduino/2.7\_toggle\_left\_right.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.

Während das Programm läuft, wird im seriellen Monitor „EIN“ oder „AUS“ angezeigt, je nachdem, in welche Richtung Sie den Schalter schieben.

## 6.12 2.8 - Sanft Drücken



Ein Mikroschalter ist ebenfalls ein 3-poliges Gerät, die Reihenfolge der drei Pins sind C (Common Pin), NO (Normalerweise offen) und NC (Normalerweise geschlossen).

Wenn der Mikroschalter nicht gedrückt ist, sind 1 (C) und 3 (NC) miteinander verbunden. Wird er gedrückt, sind 1 (C) und 2 (NO) miteinander verbunden.

- *Mikroschalter*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

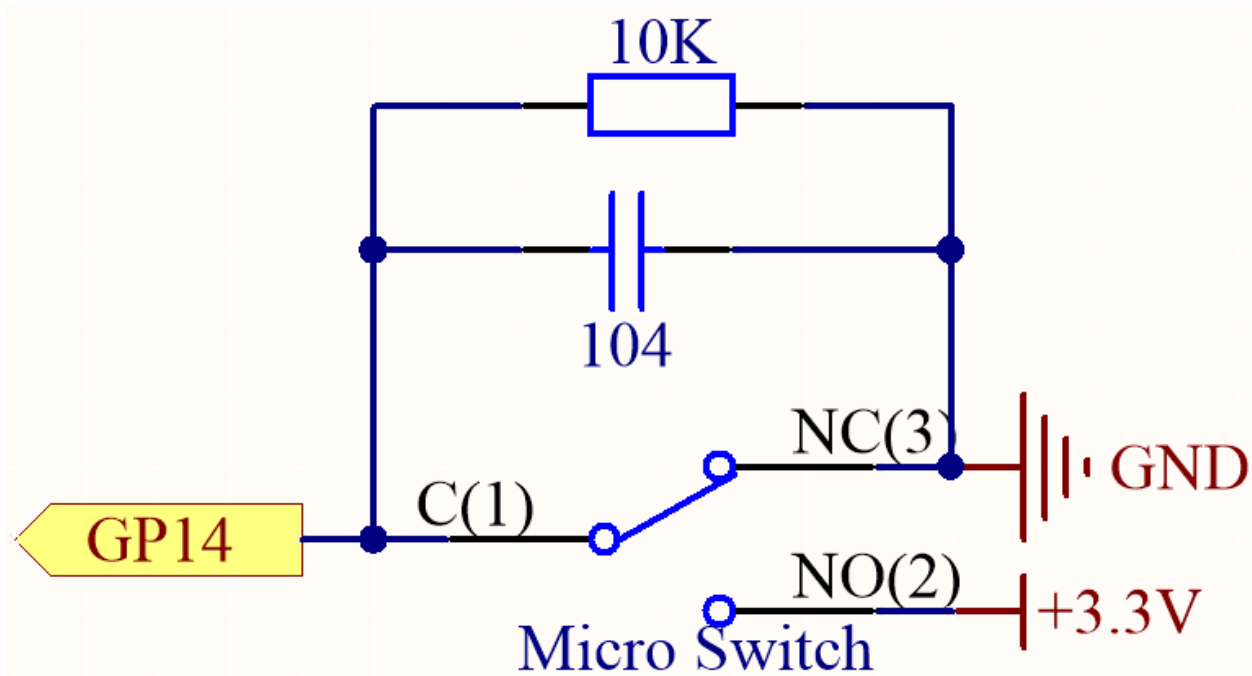
Es ist definitiv praktisch, ein komplettes Kit zu kaufen, hier ist der Link:

Bezeichnung	ARTIKEL IN DIESEM KIT	KAUF-LINK
Kepler-Kit	450+	

Sie können die Teile auch einzeln über die untenstehenden Links kaufen.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1 (10K)	
6	<i>Kondensator</i>	1 (104)	
7	<i>Mikroschalter</i>	1	

### Schaltplan

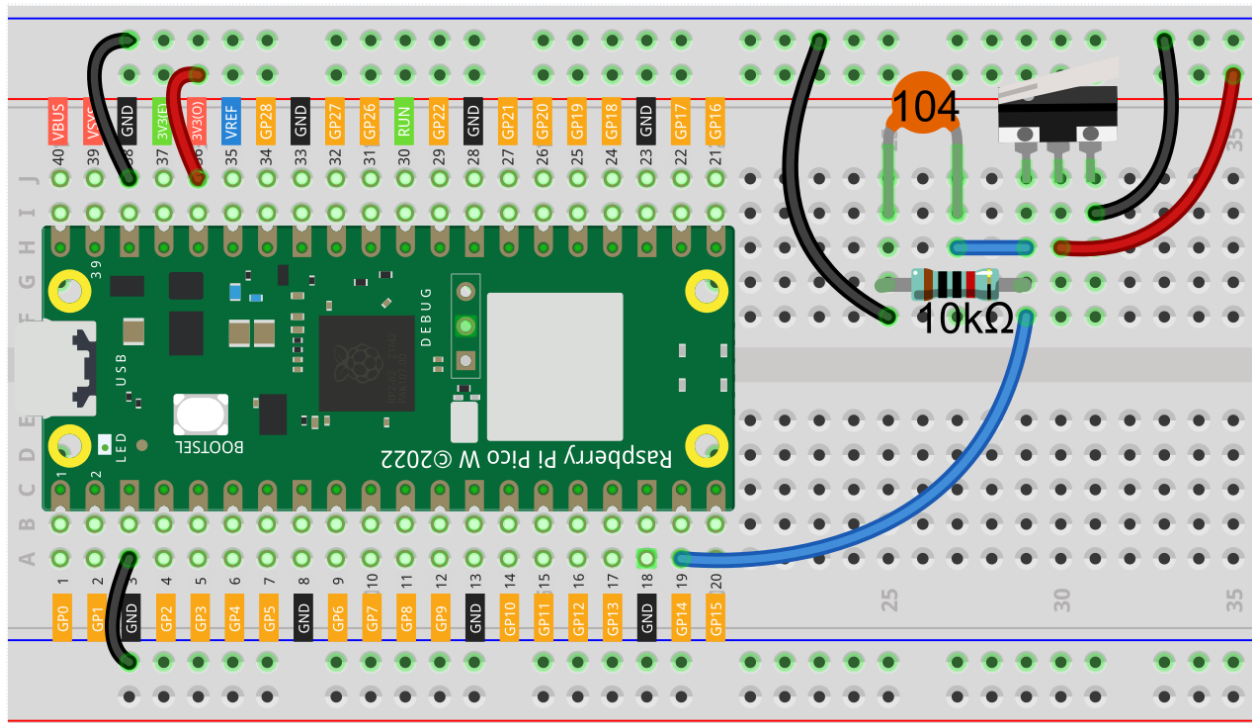


Standardmäßig ist GP14 niedrig und wird beim Drücken hoch.

Der Zweck des 10K-Widerstands ist es, GP14 während des Drückens niedrig zu halten.

Der 104-Keramikkondensator wird hier verwendet, um Rauschen zu eliminieren.

### Verkabelung



### Code

**Bemerkung:**

- Sie können die Datei `2.8_press_gently.ino` im Pfad `kepler-kit-main/arduino/2.8_press_gently` öffnen.
- Oder kopieren Sie diesen Code in die **Arduino-IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.

Nach dem Start des Programms erscheint „The switch works!“ im seriellen Monitor, wenn Sie den Schiebeschalter nach rechts bewegen.

## 6.13 2.9 - Magnetismus spüren

Der am häufigsten verwendete Reed-Schalter enthält ein Paar magnetisierbarer, flexibler Metallzungen, deren Enden bei geöffnetem Schalter durch eine kleine Lücke getrennt sind.

Ein Magnetfeld eines Elektromagneten oder eines Permanentmagneten führt dazu, dass die Metallzungen sich gegenseitig anziehen und somit einen elektrischen Stromkreis schließen. Die Federkraft der Zungen lässt sie sich wieder trennen und den Kreislauf öffnen, sobald das Magnetfeld aufhört.

Ein geläufiges Anwendungsbeispiel für Reed-Schalter ist die Überwachung des Öffnens von Türen oder Fenstern in einem Sicherheitssystem.

- *Reedschalter*

**Erforderliche Komponenten**

Für dieses Projekt benötigen wir die folgenden Komponenten.

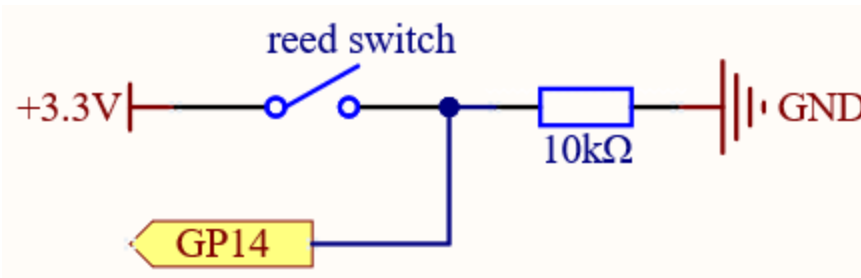
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	KAUF-LINK
Kepler Kit	450+	

Sie können diese auch separat über die folgenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	MEN-GE	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1 (10K)	
6	<i>Reedschalter</i>	1	

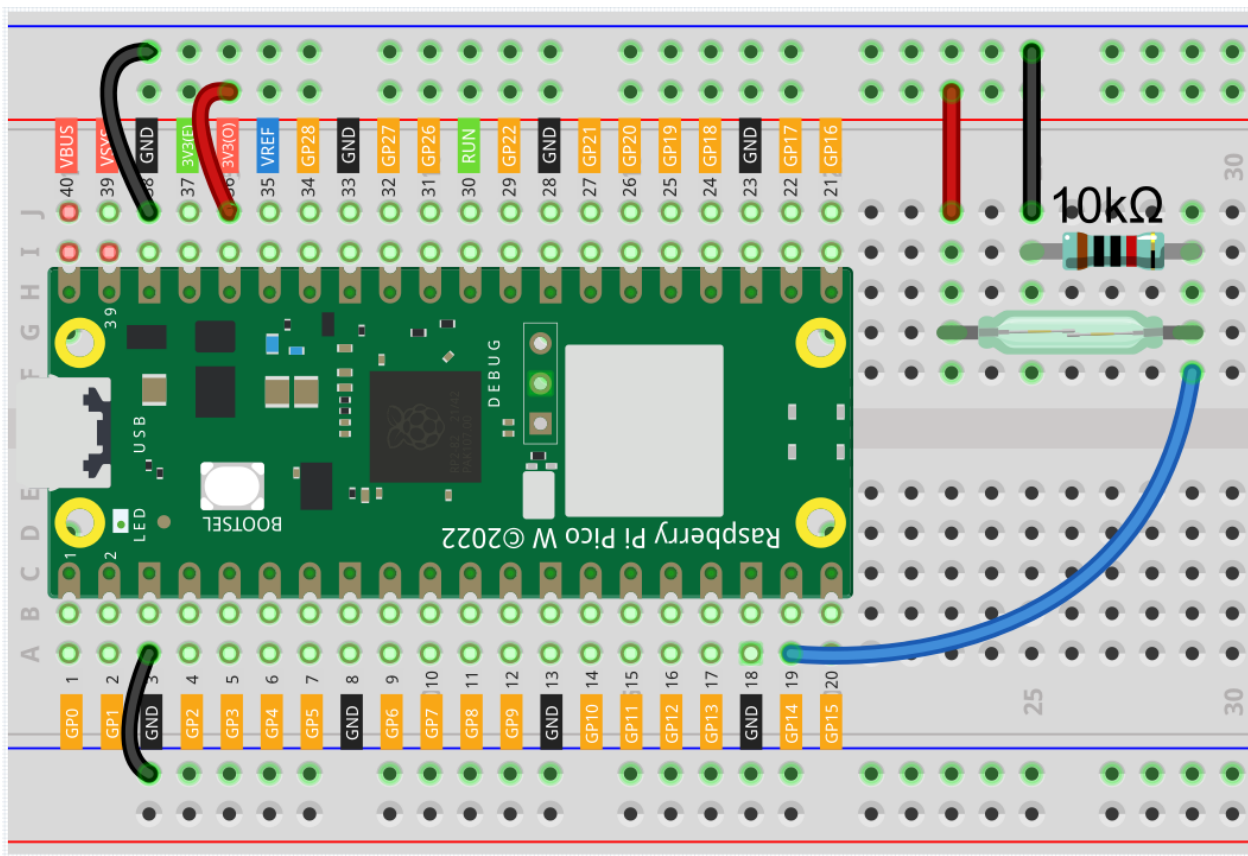
**Schaltplan**



Standardmäßig ist GP14 niedrig; er wird hoch, wenn der Magnet in der Nähe des Reed-Schalters ist.

Der 10K-Widerstand dient dazu, den GP14 auf einem konstant niedrigen Niveau zu halten, wenn kein Magnet in der Nähe ist.

## Verdrahtung



## Code

**Bemerkung:**

- Sie können die Datei `2.9_feel_the_magnetism.ino` im Pfad `kepler-kit-main/arduino/2.9_feel_the_magnetism` öffnen.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den korrekten Anschluss auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.



Wenn sich ein Magnet nähert, schließt sich der Stromkreis. Genau wie der Knopf im Kapitel 2.5 - *Tastenwert auslesen*.

## 6.14 2.10 - Menschliche Bewegung erfassen

Der passive Infrarotsensor (PIR-Sensor) ist ein gängiger Sensor, der infrarotes (IR) Licht messen kann, das von Objekten in seinem Sichtfeld abgestrahlt wird. Einfach ausgedrückt, erfasst er die von Körpern abgestrahlte Infrarotstrahlung und kann dadurch die Bewegung von Menschen und anderen Lebewesen erkennen. Konkret informiert er die Hauptsteuerung darüber, dass jemand den Raum betreten hat.

### *PIR-Bewegungssensormodul*

#### **Erforderliche Komponenten**

Für dieses Projekt werden die folgenden Komponenten benötigt.

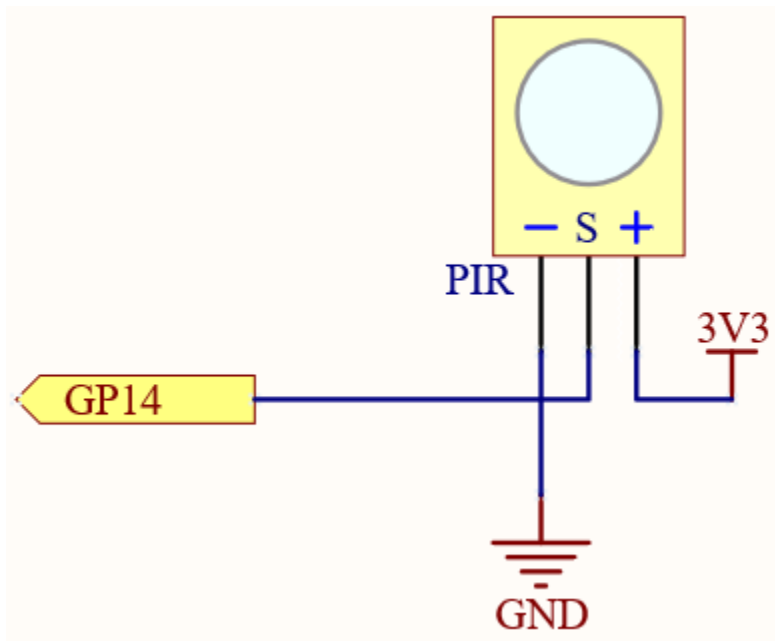
Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Bezeichnung	ELEMENTE IN DIESEM KIT	KAUF-LINK
Kepler Kit	450+	

Die Teile können auch einzeln über die folgenden Links gekauft werden.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>PIR-Bewegungssensormodul</i>	1	

#### **Schaltplan**



Wenn das PIR-Modul eine vorbeigehende Person erkennt, wird GP14 auf „High“ gesetzt, ansonsten bleibt es auf „Low“.

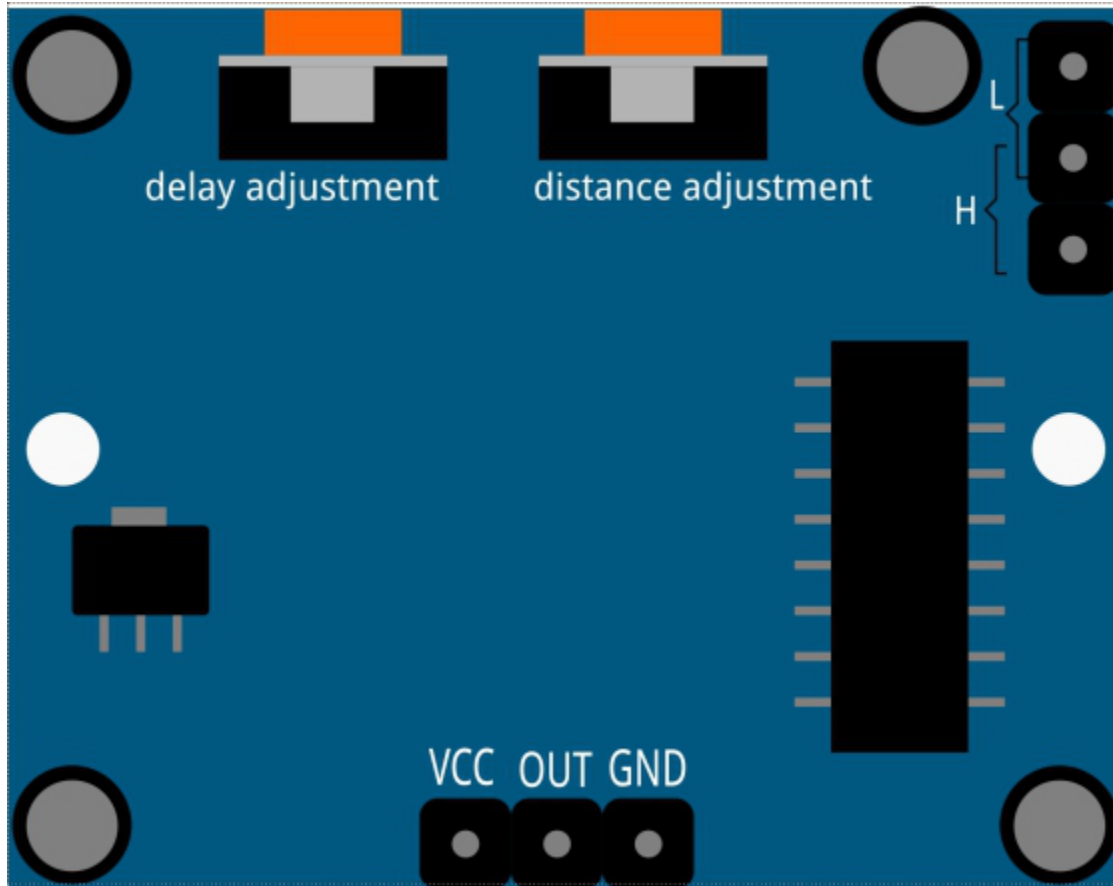
#### Verdrahtung

[illegible]

-

### Mehr erfahren

Der PIR ist ein sehr empfindlicher Sensor. Um ihn an die Einsatzumgebung anzupassen, muss er justiert werden. Richten Sie die Seite mit den beiden Potentiometern zu sich aus und drehen Sie beide Potentiometer gegen den Uhrzeigersinn ganz nach links. Setzen Sie dann die Jumperkappe auf den Pin mit L und den mittleren Pin.



#### 1. Auslösemodus

Der Jumper in der Ecke ermöglicht dem PIR, in den wiederholbaren oder nicht-wiederholbaren Auslösemodus zu wechseln.

Derzeit ist der Jumper so gesetzt, dass der PIR im nicht-wiederholbaren Modus arbeitet. In diesem Modus sendet der PIR bei erkannter Bewegung für etwa 2,8 Sekunden ein High-Signal an die Hauptsteuerung. .. In den ausgegebenen Daten sehen wir, dass die Arbeitsdauer stets rund 2800 ms beträgt.

Als nächstes ändern wir die Position der Jumperkappe und verbinden den mittleren Pin mit dem H-Pin, um den PIR in den wiederholbaren Auslösemodus zu versetzen. In diesem Modus sendet der PIR, solange sich ein Lebewesen im Erfassungsbereich bewegt, kontinuierlich ein High-Signal an die Hauptsteuerung. .. In den ausgegebenen Daten sehen wir, dass die Arbeitsdauer variabel ist.

#### 2. Verzögerungseinstellung

Das linke Potentiometer dient zur Einstellung des Intervalls zwischen zwei Arbeitszyklen.

Aktuell ist es ganz nach links gedreht, sodass der PIR nach Beendigung des High-Signal-Zyklus eine Ruhephase von etwa 5 Sekunden einlegt. In dieser Zeit werden keine Infrarotstrahlen im Zielbereich erfasst. .. In den ausgegebenen Daten sehen wir, dass die Ruhezeit immer mindestens 5000 ms beträgt.

Wenn wir das Potentiometer im Uhrzeigersinn drehen, verlängert sich auch die Ruhezeit. Wenn es

ganz im Uhrzeigersinn gedreht ist, beträgt die Ruhezeit bis zu 300 Sekunden.

### 3. Reichweiteinstellung

Das mittlere Potentiometer dient zur Einstellung des Erfassungsbereichs des PIR.

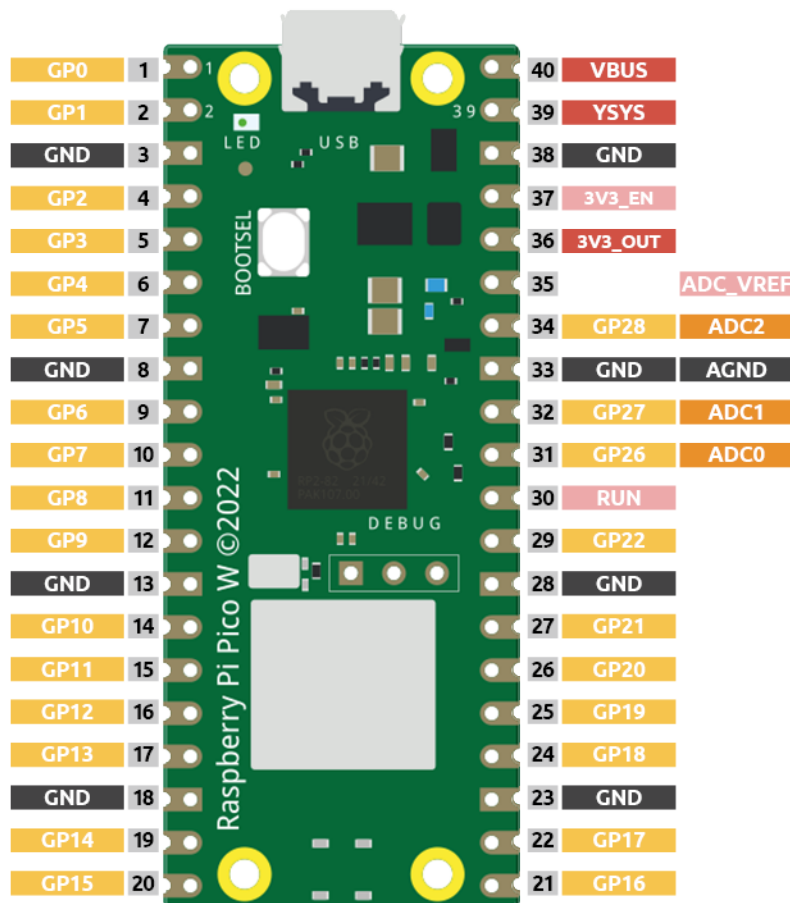
Drehen Sie den Knopf des Potentiometers **im Uhrzeigersinn**, um den Erfassungsbereich zu erhöhen. Der maximale Erfassungsbereich beträgt etwa 0-7 Meter. Dreht man es **gegen den Uhrzeigersinn**, verringert sich der Erfassungsbereich. Der minimale Erfassungsbereich beträgt dann etwa 0-3 Meter.

## 6.15 2.11 - Drehen Sie den Knopf

In vorherigen Projekten haben wir den digitalen Eingang am Pico W verwendet. Ein Taster kann beispielsweise den Pin von einem niedrigen (aus) auf einen hohen Pegel (ein) umschalten. Dies ist ein binärer Arbeitszustand.

Der Pico W kann jedoch auch eine andere Art von Eingangssignal empfangen: den analogen Eingang. Dieser kann in einem beliebigen Zustand von vollständig geschlossen bis vollständig geöffnet sein und verfügt über eine Reihe möglicher Werte. Der analoge Eingang ermöglicht es dem Mikrocontroller, die Lichtintensität, Schallintensität, Temperatur, Feuchtigkeit usw. der physischen Welt zu erfassen.

Normalerweise benötigt ein Mikrocontroller eine zusätzliche Hardware, um den analogen Eingang umzusetzen - den Analog-Digital-Wandler (ADC). Aber der Pico W hat bereits einen integrierten ADC, den wir direkt nutzen können.



Der Pico W hat drei GPIO-Pins, die analogen Eingang nutzen können: GP26, GP27, GP28, also die analogen Kanäle 0, 1 und 2. Zusätzlich gibt es einen vierten analogen Kanal, der mit dem eingebauten Temperatursensor verbunden ist und hier nicht vorgestellt wird.

In diesem Projekt versuchen wir, den Analogwert eines Potentiometers auszulesen.

- *Potentiometer*

### Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

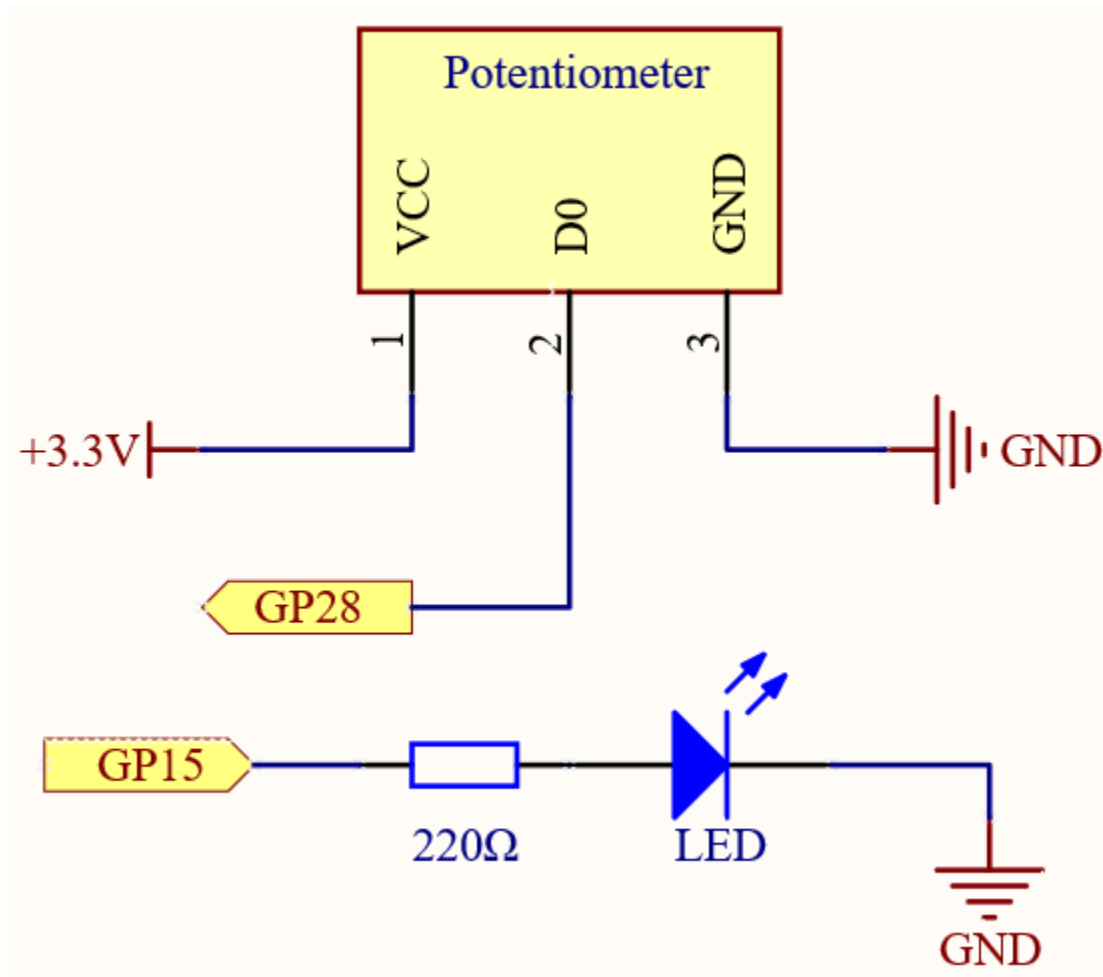
Es ist definitiv praktisch, ein ganzes Set zu kaufen. Hier ist der Link:

Name	ELEMENTE IN DIESEM KIT	KAUF-LINK
Kepler Kit	450+	

Die Teile können auch einzeln über die folgenden Links gekauft werden.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(220)	
6	<i>LED</i>	1	
7	<i>Potentiometer</i>	1	

### Schaltplan



Das Potentiometer ist ein analoges Bauelement und kann in zwei verschiedene Richtungen gedreht werden.

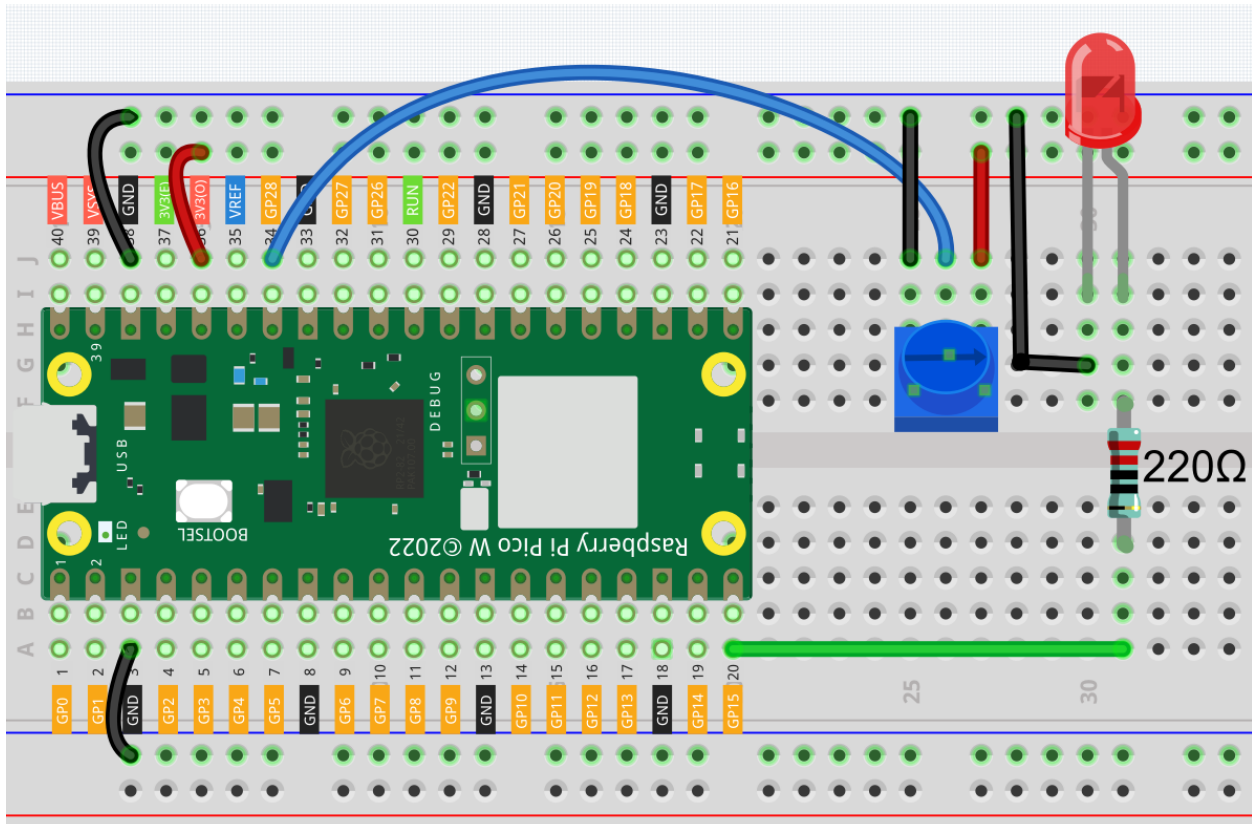
Verbinden Sie den mittleren Pin des Potentiometers mit dem analogen Pin GP28. Der Raspberry Pi Pico W enthält einen mehrkanaligen, 16-Bit-Analog-Digital-Wandler. Das bedeutet, dass er die Eingangsspannung zwischen 0 und der Betriebsspannung (3,3V) auf einen Ganzzahlwert zwischen 0 und 65535 abbildet, sodass der Wert von GP28 zwischen 0 und 65535 liegt.

Die Berechnungsformel lautet wie folgt:

$$(V_p/3.3V) \times 65535 = A_p$$

Programmieren Sie anschließend den Wert von GP28 (Potentiometer) als PWM-Wert von GP15 (LED). Auf diese Weise werden Sie feststellen, dass sich die Helligkeit der LED beim Drehen des Potentiometers gleichzeitig verändert.

### Verkabelung



## Code

### Bemerkung:

- Sie können die Datei `2.11_turn_the_knob.ino` im Pfad `kepler-kit-main/arduino/2.11_turn_the_knob` öffnen.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, die Platine (Raspberry Pi Pico) und den korrekten Anschluss auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.

Wenn das Programm läuft, können wir den aktuell von Pin GP28 gelesenen Analogwert im seriellen Monitor sehen. Drehen Sie den Knopf, und der Wert wird sich von 0 bis 1023 ändern. Gleichzeitig wird die Helligkeit der LED zunehmen, je höher der Analogwert ist.

### Wie funktioniert es?

Um den seriellen Monitor zu aktivieren, müssen Sie die serielle Kommunikation in `setup()` starten und die Datenrate auf 9600 einstellen.

```
void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}
```

- `Serial`

In der Loop-Funktion wird der Wert des Potentiometers gelesen, dann wird dieser Wert von 0-1023 auf 0-255 abgebildet, und schließlich wird der abgebildete Wert verwendet, um die Helligkeit der LED zu steuern.



```
void loop() {
  int sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);
  int brightness = map(sensorValue, 0, 1023, 0, 255);
  analogWrite(ledPin, brightness);
}
```

- `analogRead()` wird verwendet, um den Wert des `sensorPin` (Potentiometer) zu lesen und ihn der Variable `sensorValue` zuzuweisen.

```
int sensorValue = analogRead(sensorPin);
```

- Der Wert von `SensorValue` wird im seriellen Monitor ausgegeben.

```
Serial.println(sensorValue);
```

- Hier wird die Funktion `map(value, fromLow, fromHigh, toLow, toHigh)` benötigt, da der gelesene Potentiometerwert im Bereich 0-1023 liegt und der Wert eines PWM-Pins im Bereich 0-255 liegt. Sie wird verwendet, um eine Zahl von einem Bereich in einen anderen umzumappen.

```
int brightness = map(sensorValue, 0, 1023, 0, 255);
```

- Nun können wir diesen Wert verwenden, um die Helligkeit der LED zu steuern.

```
analogWrite(ledPin, brightness);
```

## 6.16 2.12 - Das Licht erfassen

Der Fotowiderstand ist ein typisches Bauelement für analoge Eingänge und wird ähnlich wie ein Potentiometer verwendet. Sein Widerstandswert hängt von der Lichtintensität ab: Je stärker das einfallende Licht, desto geringer der Widerstandswert; umgekehrt nimmt er zu.

- *Fotowiderstand*

### Erforderliche Bauteile

Für dieses Projekt benötigen wir die folgenden Bauteile.

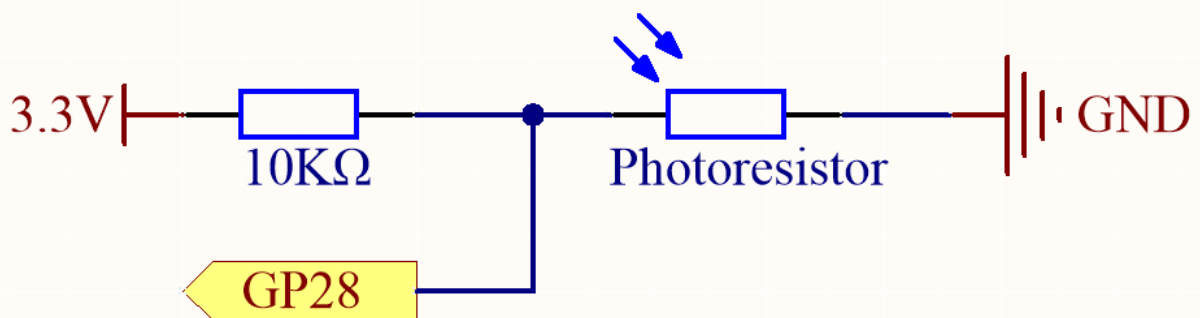
Ein komplettes Set zu kaufen, ist definitiv praktisch. Hier ist der Link:

Bezeichnung	ELEMENTE IN DIESEM KIT	KAUF-LINK
Kepler Kit	450+	

Alternativ können Sie die Teile auch einzeln über die folgenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Fotowiderstand</i>	1	

### Schaltbild



In dieser Schaltung sind der 10K-Widerstand und der Fotowiderstand in Reihe geschaltet. Der durch sie fließende Strom ist identisch. Der 10K-Widerstand dient als Schutz, und GP28 liest den Wert nach der Spannungsumwandlung des Fotowiderstands.

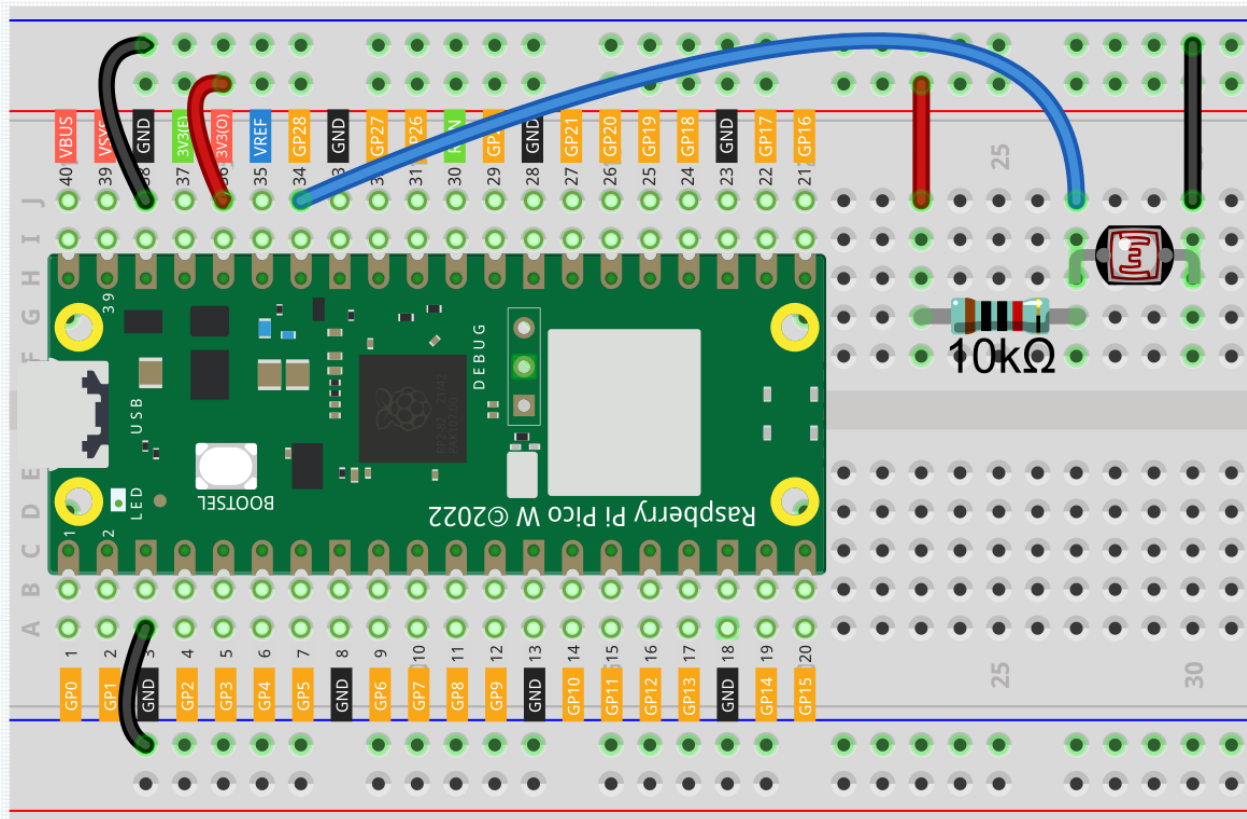
Wenn das Licht intensiver wird, verringert sich der Widerstand des Fotowiderstands und damit auch seine Spannung. Daraufhin sinkt der Wert von GP28. Ist das Licht stark genug, nähert sich der Widerstand des Fotowiderstands dem Wert 0, und der Wert von GP28 wird ebenfalls nahezu 0 sein. In diesem Fall spielt der 10K-Widerstand eine schützende Rolle, sodass 3,3V und GND nicht direkt miteinander verbunden und somit ein Kurzschluss vermieden wird.

In einer dunklen Umgebung steigt der Wert von GP28. Ist es dunkel genug, wird der Widerstand des Fotowiderstands unendlich groß, seine Spannung nähert sich 3,3V an (der 10K-Widerstand ist vernachlässigbar), und der Wert von GP28 erreicht nahezu den Maximalwert von 65535.

Die Berechnungsformel lautet wie folgt:

$$(V_p/3,3V) \times 65535 = A_p$$

### Verkabelung



## Programmcode

**Bemerkung:**

- Die Datei `2.12_feel_the_light.ino` befindet sich im Verzeichnis `kepler-kit-main/arduino/2.12_feel_the_light`.
- Alternativ können Sie den Code auch direkt in die **Arduino IDE** kopieren.
- Denken Sie daran, vor dem Hochladen des Programms die richtige Platine (Raspberry Pi Pico) und den entsprechenden Port auszuwählen.

Nach dem Start des Programms gibt der serielle Monitor die Werte des Fotowiderstands aus. Sie können die Werte verändern, indem Sie eine Taschenlampe darauf richten oder ihn mit der Hand abdecken.

## 6.17 2.13 - Thermometer

Ein Thermometer ist ein Gerät, das die Temperatur oder ein Temperaturgefälle (den Grad der Wärme oder Kälte eines Objekts) misst. Ein Thermometer besteht aus zwei wichtigen Elementen: (1) einem Temperatursensor (z.B. der Glühbirne eines Quecksilberthermometers oder dem pyrometrischen Sensor in einem Infrarotthermometer), bei dem eine Veränderung mit einer Temperaturänderung eintritt; und (2) einer Methode zur Umwandlung dieser Veränderung in einen Zahlenwert (z.B. die sichtbare Skala, die auf einem Quecksilberthermometer markiert ist, oder die digitale Anzeige bei einem Infrarotmodell). Thermometer finden in Technologie und Industrie zur Prozessüberwachung, in der Meteorologie, in der Medizin und in der wissenschaftlichen Forschung breite Anwendung.

Ein Thermistor ist eine Art von Temperatursensor, dessen Widerstand stark temperaturabhängig ist. Es gibt zwei Typen: Negativer Temperaturkoeffizient (NTC) und Positiver Temperaturkoeffizient (PTC), auch bekannt als NTC und PTC.

Der Widerstand von PTC-Thermistoren steigt mit der Temperatur, während der Zustand von NTC dem entgegengesetzt ist.

In diesem Experiment verwenden wir einen **NTC-Thermistor**, um ein Thermometer zu bauen.

- *Thermistor*

### Benötigte Bauteile

Für dieses Projekt benötigen wir die folgenden Bauteile.

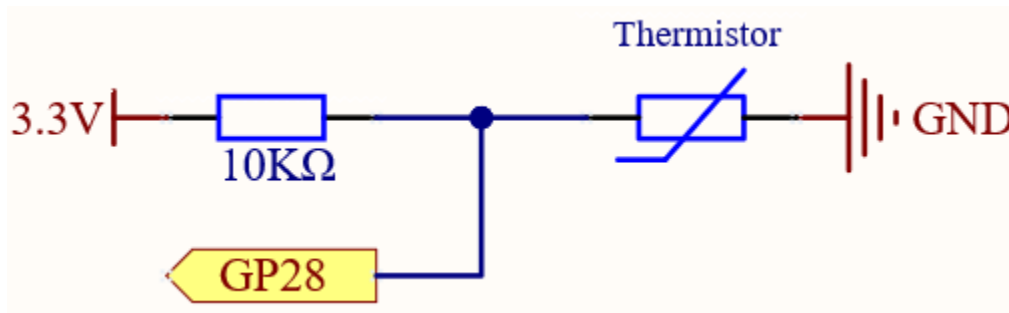
Ein komplettes Set zu kaufen ist definitiv praktisch, hier ist der Link:

Name	ARTIKEL IN DIESEM SET	KAUF-LINK
Kepler Kit	450+	

Sie können sie auch einzeln über die untenstehenden Links kaufen.

SN	BAUTEILBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1 (10K)	
6	<i>Thermistor</i>	1	

### Schaltplan



In diesem Schaltkreis sind der 10K-Widerstand und der Thermistor in Reihe geschaltet, und der durch sie fließende Strom ist derselbe. Der 10K-Widerstand dient als Schutz, und GP28 liest den Wert nach der Spannungsumwandlung des Thermistors.

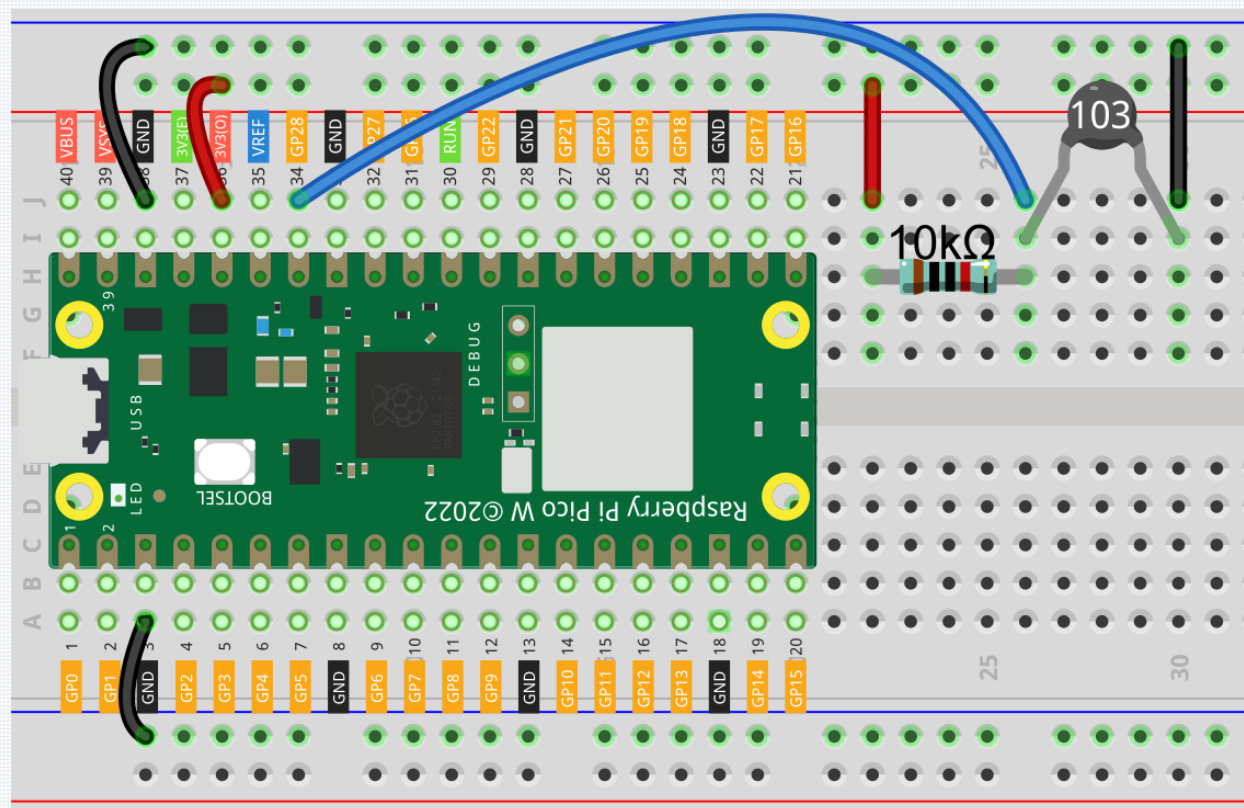
Wenn die Temperatur steigt, sinkt der Widerstandswert des NTC-Thermistors, dann sinkt seine Spannung, so dass der Wert von GP28 sinkt; Wenn die Temperatur hoch genug ist, wird der Widerstand des Thermistors nahezu 0 sein, und der Wert von GP28 wird nahezu 0 sein. In diesem Fall spielt der 10K-Widerstand eine schützende Rolle, so dass 3,3V und GND nicht direkt miteinander verbunden sind, was zu einem Kurzschluss führen würde.

Wenn die Temperatur fällt, wird der Wert von GP28 steigen. Wenn die Temperatur niedrig genug ist, wird der Widerstand des Thermistors unendlich sein, und seine Spannung wird nahe an 3,3V liegen (der 10K-Widerstand ist vernachlässigbar), und der Wert von GP28 wird nahe am Maximalwert von 65535 liegen.

Die Berechnungsformel ist unten dargestellt.

$$(V_p/3,3V) \times 65535 = A_p$$

### Verdrahtung



### Bemerkung:

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

### Code

### Bemerkung:

- Sie können die Datei `2.13_thermometer.ino` unter dem Pfad `kepler-kit-main/arduino/2.13_thermometer` öffnen.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.

Nachdem das Programm ausgeführt wurde, wird der Serielle Monitor die Temperaturen in Celsius und Fahrenheit ausgeben.

### Wie funktioniert es?

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

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

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

```
long a = analogRead(analogPin);
```

Diese Zeile dient zum Auslesen des Werts des Thermistors.

```
float tempC = beta / (log((1025.0 * 10 / a - 10) / 10) + beta / 298.0) - 273.0;  
float tempF = 1.8 * tempC + 32.0;
```

Diese Berechnungen wandeln die Werte des Thermistors in Grad Celsius und Fahrenheit um.

---

**Bemerkung:** Hier ist der Zusammenhang zwischen Widerstand und Temperatur:

$$RT = RN \exp(B(1/TK - 1/TN))$$

- RT ist der Widerstand des NTC-Thermistors, wenn die Temperatur TK beträgt.
- RN ist der Widerstand des NTC-Thermistors bei der Nenntemperatur TN. Hier beträgt der Zahlenwert von RN 10k.
- TK ist eine Kelvin-Temperatur und die Einheit ist K. Hier beträgt der Zahlenwert von TK 273,15 + Grad Celsius.
- TN ist eine Nenntemperatur in Kelvin; die Einheit ist ebenfalls K. Hier beträgt der Zahlenwert von TN 273,15+25.
- Und B (Beta), die Materialkonstante des NTC-Thermistors, wird auch als Wärmeempfindlichkeitsindex bezeichnet und hat einen Zahlenwert von 3950.
- exp ist die Abkürzung für Exponential, und die Basiszahl e ist eine natürliche Zahl und beträgt ungefähr 2,7.

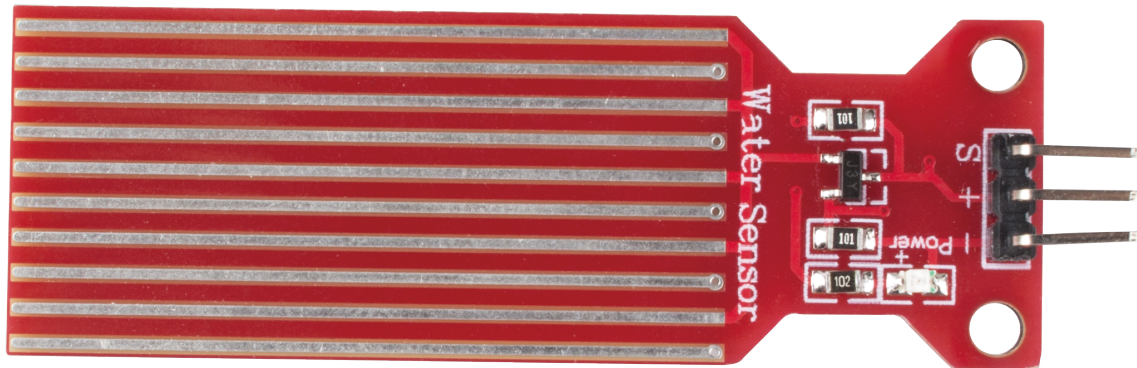
Wandeln Sie diese Formel  $TK=1/(\ln(RT/RN)/B+1/TN)$  um, um die Kelvin-Temperatur zu erhalten, die minus 273,15 gleich Grad Celsius ist.

Dieser Zusammenhang ist eine empirische Formel. Sie ist nur dann genau, wenn die Temperatur und der Widerstand im wirksamen Bereich liegen.

---

Dieser Code bezieht sich darauf, Rt in die Formel  $TK=1/(\ln(RT/RN)/B+1/TN)$  einzusetzen, um die Kelvin-Temperatur zu erhalten.

## 6.18 2.14 - Den Wasserstand erfühlen



Der Wassersensor ist für die Wassererkennung konzipiert und kann vielseitig zur Erfassung von Niederschlägen, Wasserständen und sogar Flüssigkeitsaustritten eingesetzt werden.

Der Sensor misst den Wasserstand durch eine Reihe von freiliegenden parallelen Drahtspuren, um die Größe der Wassertropfen/das Volumen zu messen. Das Wasservolumen lässt sich leicht in ein analoges Signal umwandeln, und der ausgegebene analoge Wert kann direkt vom Hauptsteuerbrett abgelesen werden, um den Wasserstandsalarm zu aktivieren.

**Warnung:** Der Sensor darf nicht vollständig ins Wasser getaucht werden; bitte lassen Sie nur den Teil, an dem sich die zehn Spuren befinden, mit dem Wasser in Kontakt kommen. Das Einschalten des Sensors in einer feuchten Umgebung beschleunigt die Korrosion der Sonde und verkürzt die Lebensdauer des Sensors. Es wird daher empfohlen, den Sensor nur dann mit Strom zu versorgen, wenn Messungen durchgeführt werden.

- *Wasserspiegelsensor-Modul*

### Benötigte Bauteile

Für dieses Projekt benötigen wir die folgenden Komponenten.

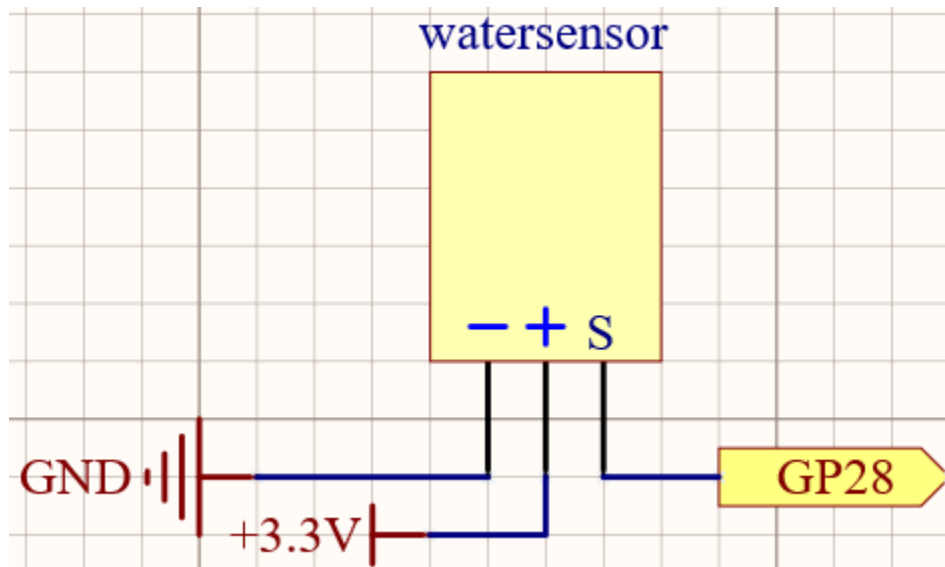
Ein Komplettsset ist natürlich praktisch, hier ist der Link:

Bezeichnung	INHALT DES KITS	KAUF-LINK
Kepler Kit	450+	

Die Bauteile können auch einzeln über die untenstehenden Links erworben werden.

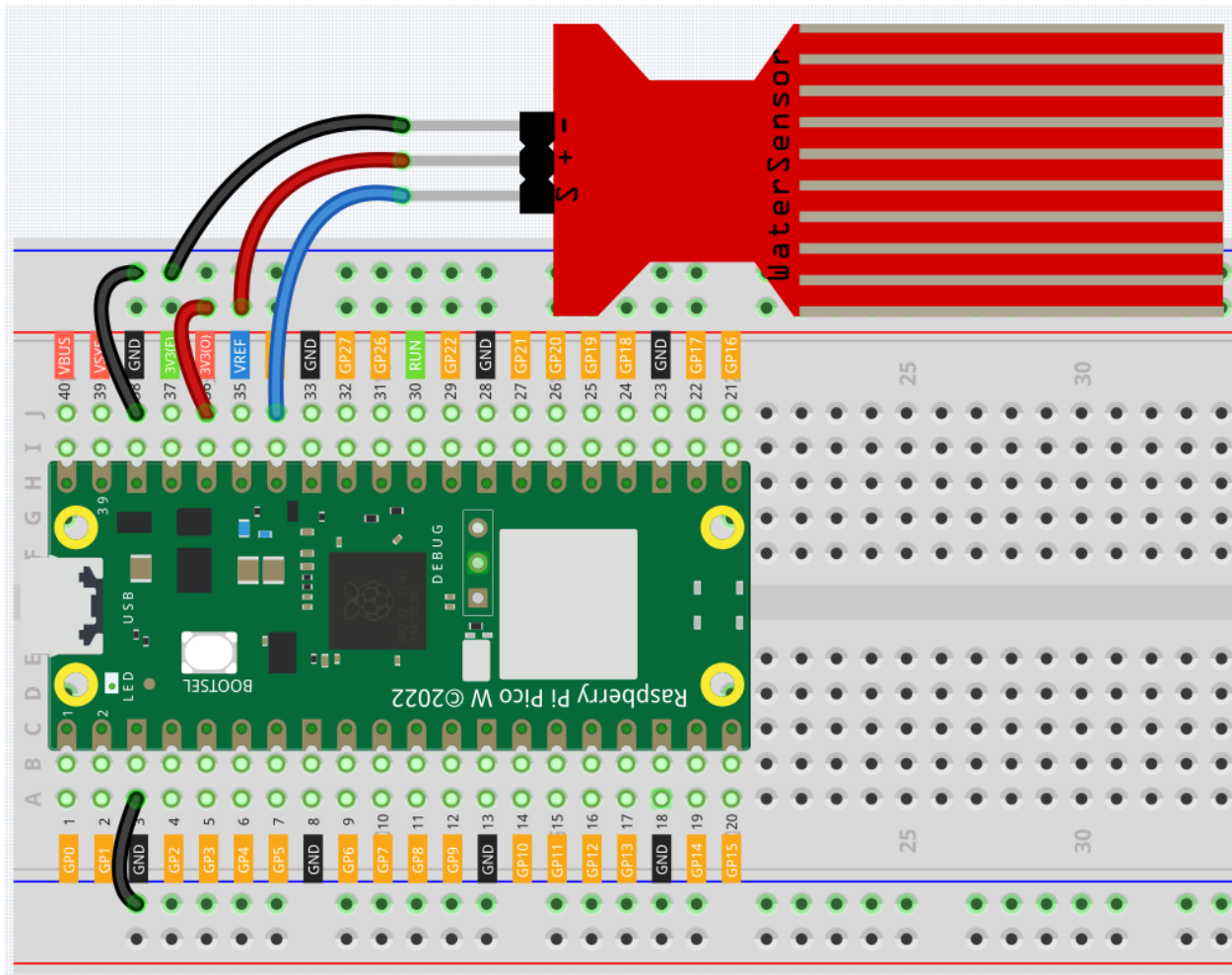
SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Wasserspiegelsensor-Modul</i>	1	

### Schaltplan



### Verkabelung





## Code

### Bemerkung:

- Die Datei `2.14_feel_the_water_level.ino` finden Sie unter dem Pfad `kepler-kit-main/arduino/2.14_feel_the_water_level`.
- Alternativ können Sie den Code in die **Arduino IDE** kopieren.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf **Hochladen** klicken.

Nachdem das Programm gestartet ist, tauchen Sie das Wassersensormodul langsam ins Wasser. Mit zunehmender Tiefe wird die Shell einen größeren Wert ausgeben.

### Mehr erfahren

Es gibt eine Möglichkeit, das Analogeingabemodul als digitales Modul zu verwenden.

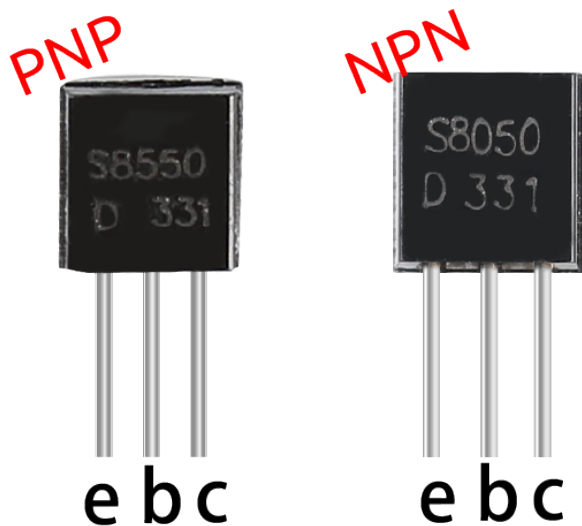
Zuerst messen Sie den Wert des Wassersensors in einer trockenen Umgebung und verwenden diesen als Schwellenwert. Anschließend führen Sie die Programmierung durch und lesen den Wert des Wassersensors erneut. Weicht der Wert des Wassersensors erheblich von dem in einer trockenen Umgebung ab, wurde er einer Flüssigkeit ausgesetzt. Das heißt, dieses Gerät kann neben einem Wasserrohr platziert werden, um festzustellen, ob das Rohr undicht ist.

**Bemerkung:**

- Die Datei `2.14_water_level_threshold.ino` finden Sie unter dem Pfad `kepler-kit-main/arduino/2.14_water_level_threshold`.
- Alternativ können Sie den Code in die **Arduino IDE** kopieren.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf **Hochladen** klicken.

## 6.19 2.15 - Zwei Arten von Transistoren

Dieses Kit enthält zwei Typen von Transistoren, S8550 und S8050. Ersterer ist ein PNP-Transistor und der Letztere ein NPN-Transistor. Beide sehen sehr ähnlich aus, daher ist es wichtig, ihre Beschriftungen genau zu prüfen. Während ein NPN-Transistor durch ein High-Level-Signal aktiviert wird, benötigt ein PNP-Transistor ein Low-Level-Signal. Beide Transistortypen finden häufig Anwendung in berührungslosen Schaltern, wie in diesem Experiment.



Verwenden wir eine LED und einen Taster, um den Umgang mit Transistoren zu verstehen!

*Transistor*

**Benötigte Bauteile**

Für dieses Projekt sind folgende Komponenten erforderlich.

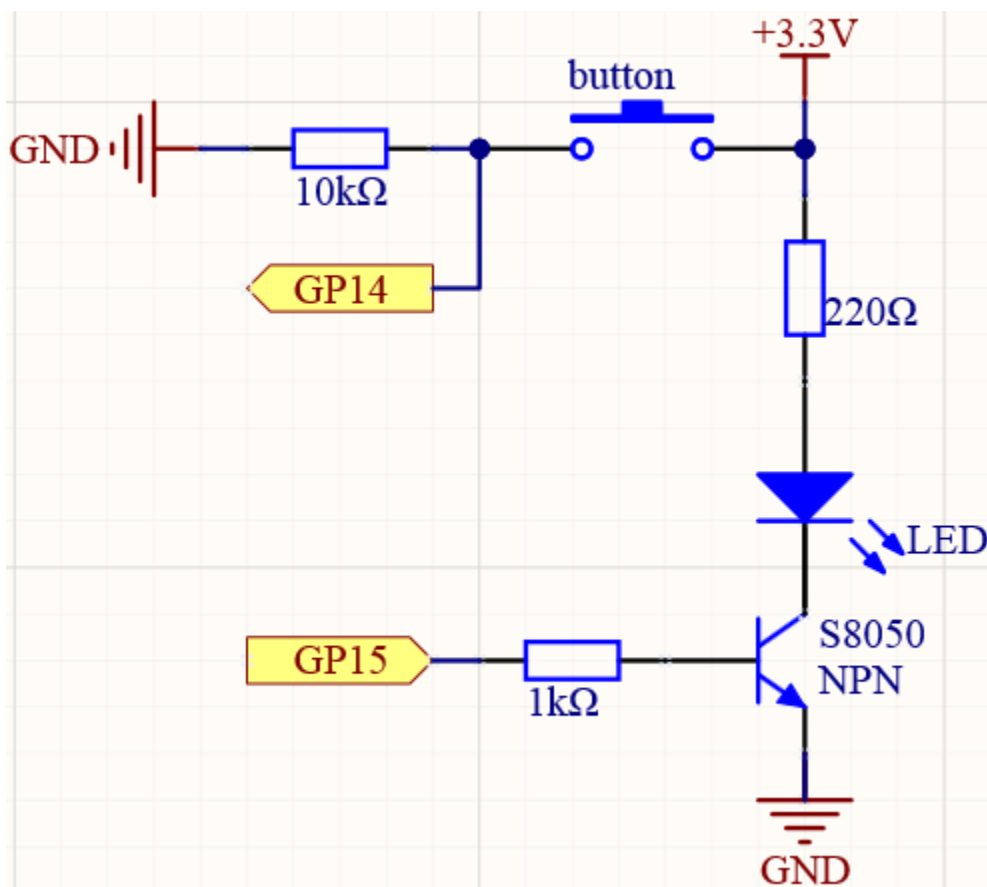
Ein Komplettsset ist durchaus praktisch, hier der Link dazu:

Bezeichnung	INHALT DES KITS	KAUF-LINK
Kepler Kit	450+	

Die Komponenten können auch einzeln über die untenstehenden Links erworben werden.

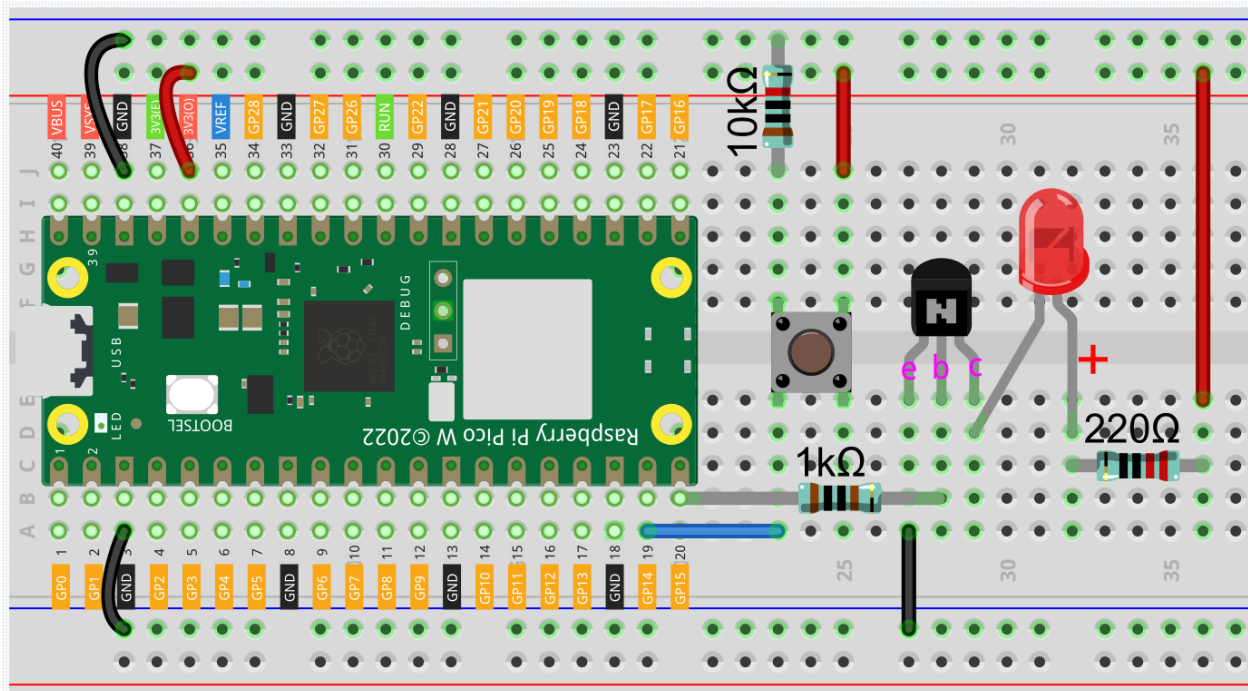
SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	3(220, 1K, 10K)	
6	<i>LED</i>	1	
7	<i>Taster</i>	1	
8	<i>Transistor</i>	1(S8050/S8550)	

#### Anschluss des NPN (S8050) Transistors

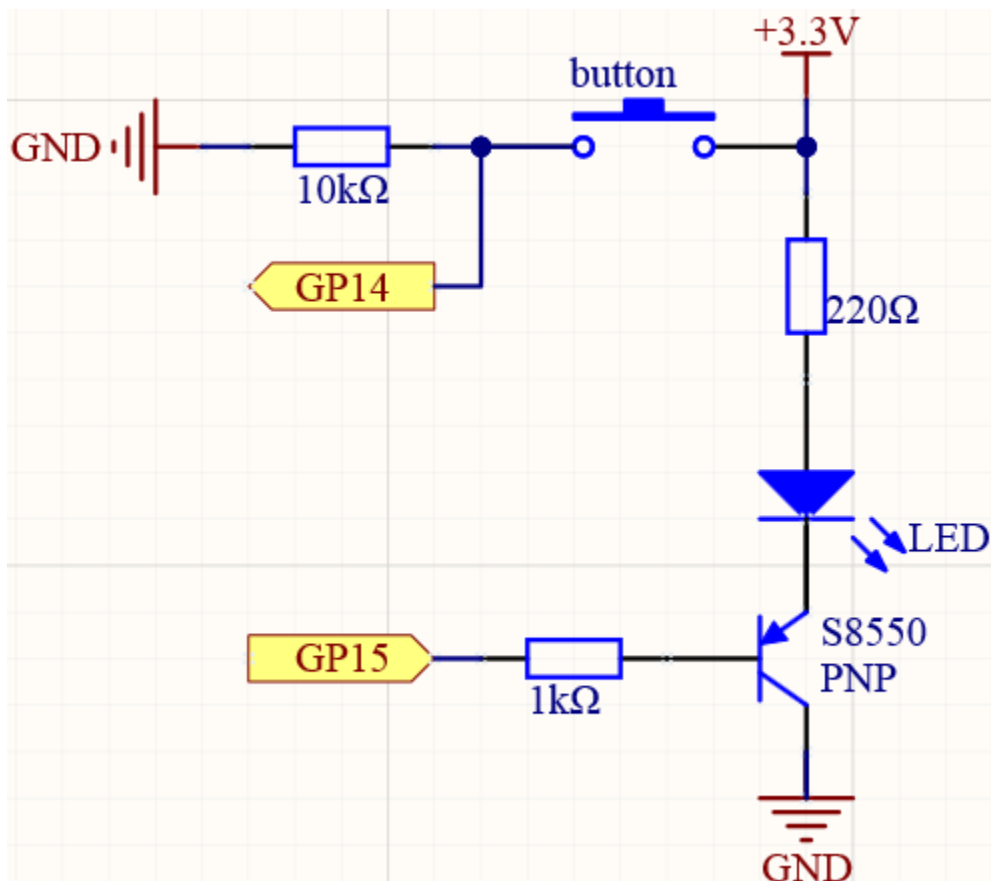


In dieser Schaltung ist GP14 high, wenn der Taster gedrückt wird.

Durch Programmierung von GP15 auf High und nach einem 1k-Strombegrenzungswiderstand (zum Schutz des Transistors) wird der S8050 (NPN-Transistor) zum Leiten gebracht, sodass die LED aufleuchtet.



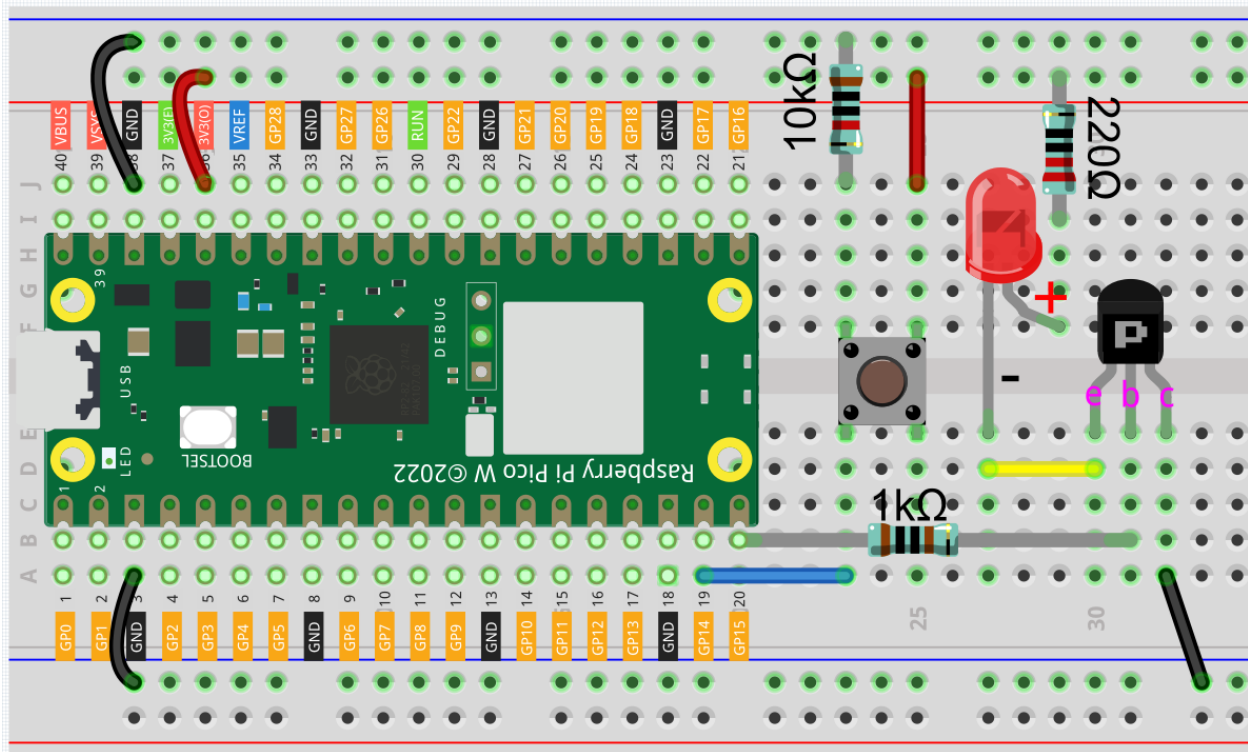
Anschluss des PNP (S8550) Transistors



In dieser Schaltung ist GP14 standardmäßig auf Low und wird auf High gesetzt, wenn der Taster gedrückt wird.

Durch Programmierung von GP15 auf **Low** und nach einem 1k-Strombegrenzungswiderstand wird der S8550 (PNP-Transistor) zum Leiten gebracht, sodass die LED leuchtet.

Der einzige Unterschied, den Sie zwischen dieser und der vorherigen Schaltung feststellen werden, ist, dass in der vorherigen Schaltung die Kathode der LED mit dem **Kollektor** des **S8050 (NPN-Transistor)** verbunden ist, während sie hier mit dem **Emitter** des **S8550 (PNP-Transistor)** verbunden ist.



## Code

### Bemerkung:

- Die Datei `2.15_transistor.ino` finden Sie unter dem Pfad `kepler-kit-main/arduino/2.15_transistor`.
- Alternativ können Sie den Code in die **Arduino IDE** kopieren.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf **Hochladen** klicken.

Beide Transistortypen können mit demselben Code gesteuert werden. Wenn wir den Taster drücken, sendet der Pico W ein High-Level-Signal an den Transistor; lassen wir ihn los, sendet er ein Low-Level-Signal. Man wird feststellen, dass in den beiden Schaltungen diametral entgegengesetzte Phänomene auftreten.

- Die Schaltung mit dem S8050 (NPN-Transistor) leuchtet auf, wenn der Taster gedrückt wird, was bedeutet, dass sie ein High-Level-Leitungssignal erhält;
- Die Schaltung mit dem S8550 (PNP-Transistor) leuchtet auf, wenn sie losgelassen wird, was bedeutet, dass sie ein Low-Level-Leitungssignal erhält.

## 6.20 2.16 - Steuern eines weiteren Stromkreises

Im Alltag können wir einen Schalter betätigen, um eine Lampe ein- oder auszuschalten. Aber was, wenn Sie die Lampe mit einem Pico W so steuern möchten, dass sie automatisch nach zehn Minuten ausgeht?

Ein Relais kann Ihnen dabei helfen.

Ein Relais ist im Grunde ein spezieller Schalter, der von einer Seite des Stromkreises (in der Regel einem Niederspannungsstromkreis) gesteuert wird und dazu dient, die andere Seite des Stromkreises (meist ein Hochspannungsstromkreis) zu steuern. Dies macht es praktisch, unsere Haushaltsgeräte umzurüsten, damit sie programmgesteuert, zu intelligenten Geräten oder sogar internetfähig werden.

**Warnung:** Das Modifizieren von Elektrogeräten ist sehr gefährlich. Versuchen Sie es nicht leichtfertig und bitte nur unter professioneller Anleitung.

- *Relais*

In diesem Beispiel verwenden wir einen einfachen, mit einem Breadboard-Strommodul betriebenen Stromkreis, um zu zeigen, wie man ihn mit einem Relais steuert.

- `cpn_power_module`

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

Ein Komplettsset ist definitiv praktisch, hier ist der Link:

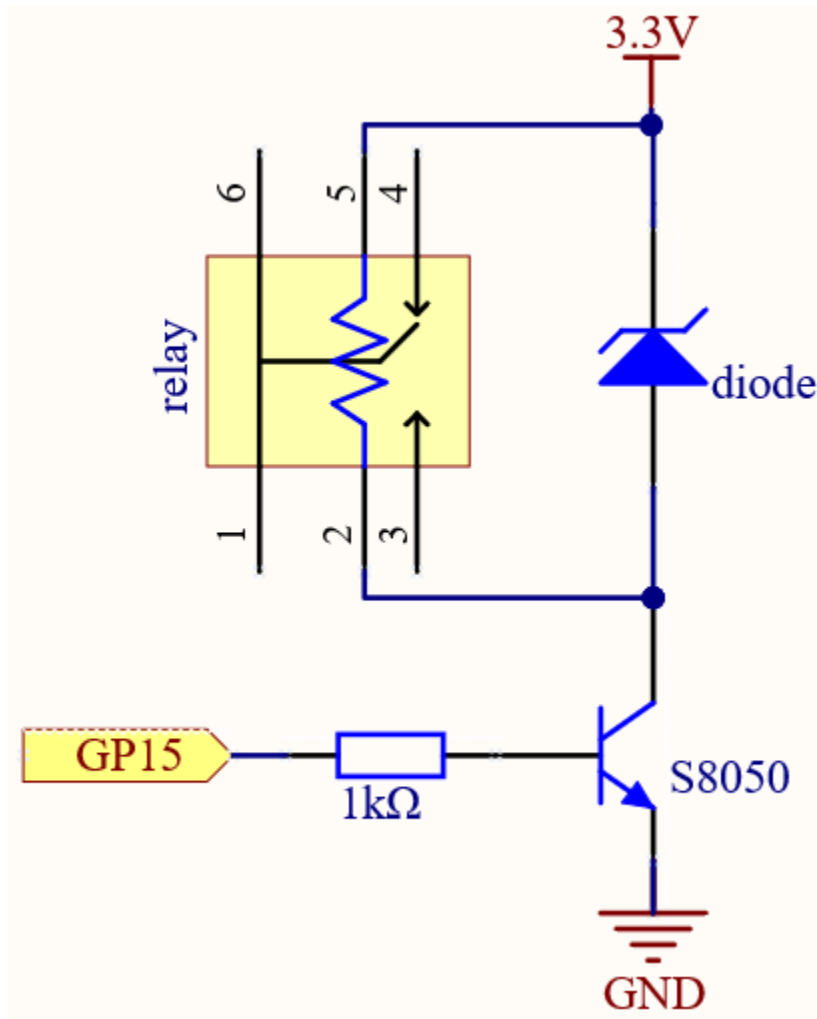
Name	ARTIKEL IN DIESEM KIT	KAUF-LINK
Kepler Kit	450+	

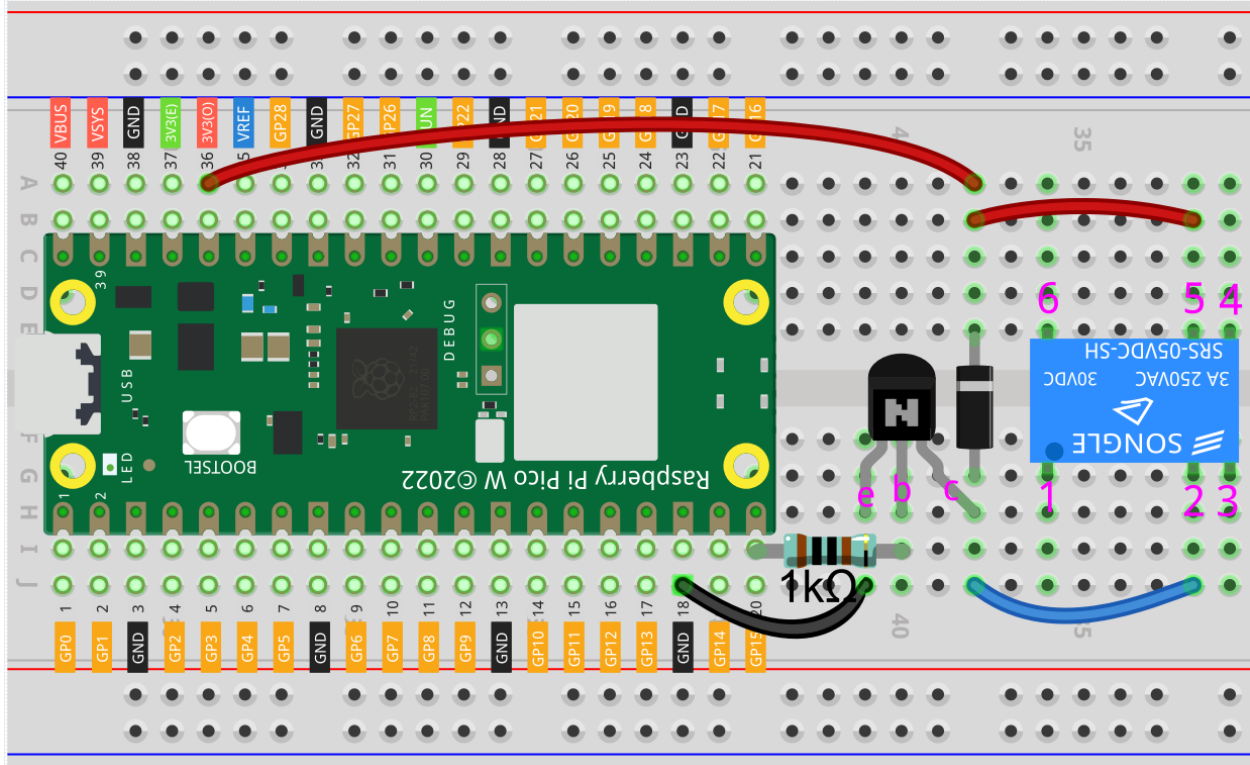
Sie können die Teile auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1 (S8050)	
6	<i>Diode</i>	1	
7	<i>Relais</i>	1	

### Verdrahtung

Bauen Sie zunächst einen Niederspannungsstromkreis, um ein Relais zu steuern. Für das Schalten des Relais ist ein hoher Strom erforderlich, weshalb ein Transistor erforderlich ist. Hier verwenden wir den S8050.





Eine Diode (Freilaufdiode) dient hier zum Schutz des Stromkreises. Die Kathode ist das Ende mit dem Silberstreifen, das mit der Stromquelle verbunden ist, die Anode ist mit dem Transistor verbunden.

Wenn die Eingangsspannung von Hoch (5V) auf Niedrig (0V) wechselt, ändert der Transistor seinen Zustand von Sättigung zu Sperrzustand und der Strom kann plötzlich nicht mehr durch die Spule fließen.

Wird diese Freilaufdiode nicht eingebaut, kann die Spule eine selbstinduzierte elektrische Spannung erzeugen, die mehrere Male höher ist als die Versorgungsspannung. Diese Spannung könnte den Transistor zerstören.

Durch das Hinzufügen der Diode wird ein neuer Stromkreis zwischen Spule und Diode gebildet, der durch die in der Spule gespeicherte Energie entladen wird. Dadurch wird übermäßige Spannung vermieden, die Bauteile wie Transistoren im Stromkreis beschädigen könnte.

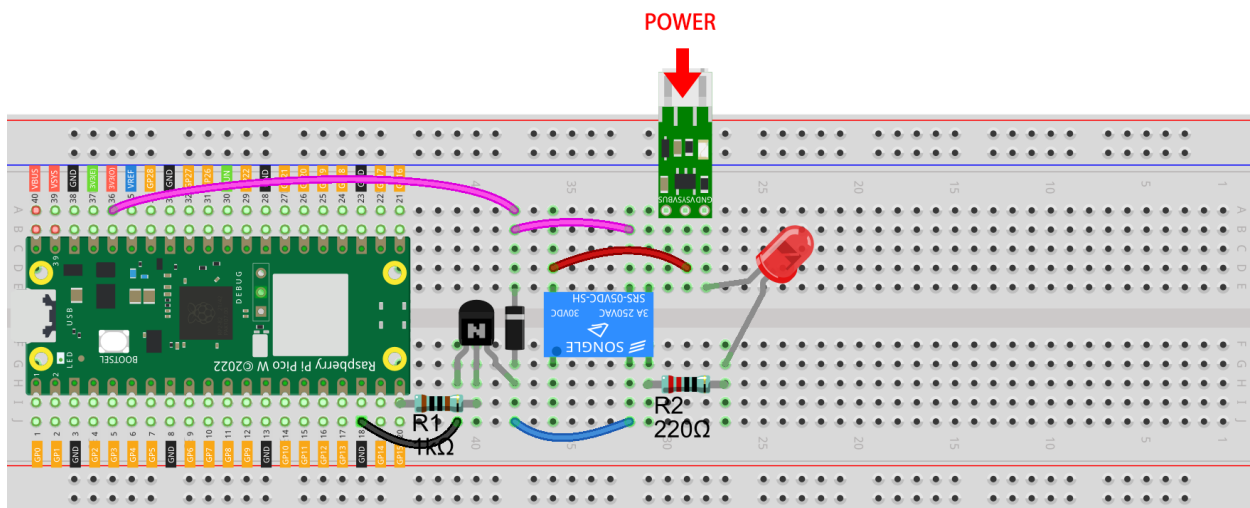
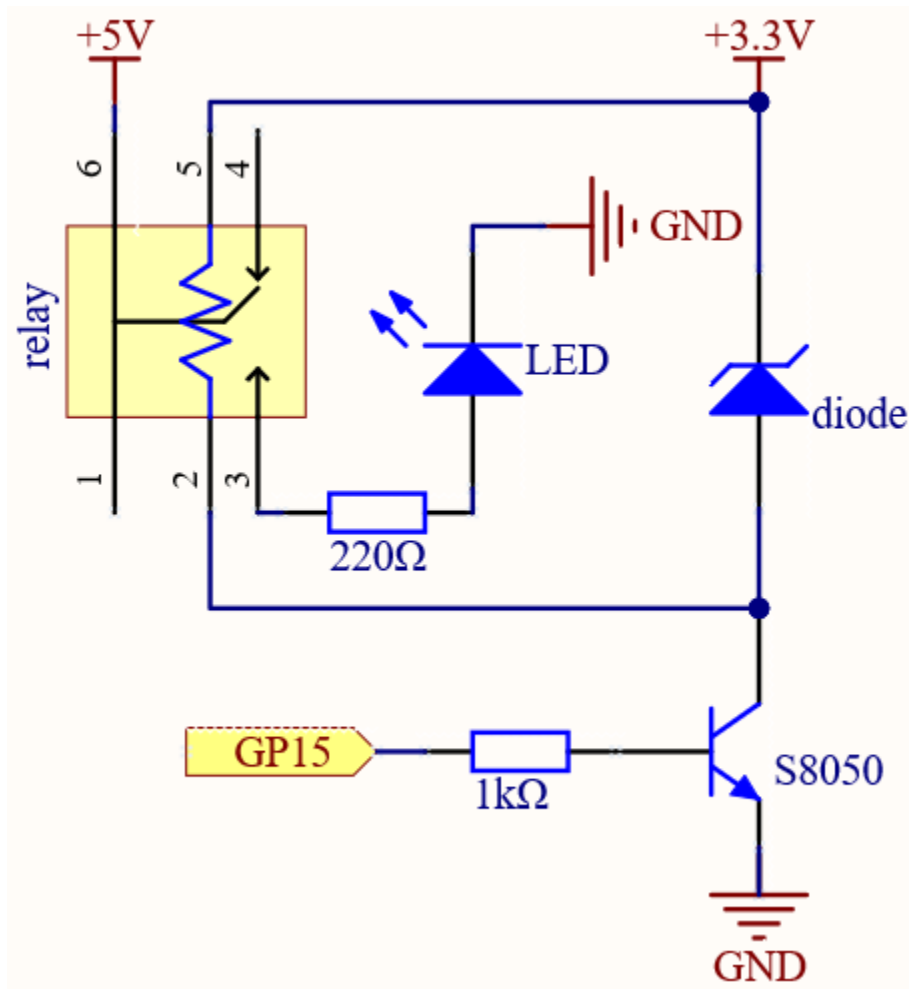
- *Diode*
- [Flyback Diode - Wikipedia](#)

Nach dem Hochladen des Programms hören Sie ein „Klick-Klack“-Geräusch, das vom Kontaktor im Inneren des Relais stammt.

Anschließend verbinden wir die beiden Enden des Laststromkreises mit den Pins 3 und 6 des Relais.

..(Nehmen Sie den einfachen, mit dem Breadboard-Strommodul betriebenen Stromkreis aus dem vorherigen Artikel als Beispiel.)





Jetzt kann das Relais den Laststromkreis ein- und ausschalten.

Code

**Bemerkung:**

- Sie können die Datei `2.16_relay.ino` im Verzeichnis `kepler-kit-main/arduino/2.16_relay` öffnen.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, die korrekte Platine (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf **Hochladen** klicken.

Nach dem Ausführen des Codes wird das Relais den Betriebszustand des gesteuerten Stromkreises alle zwei Sekunden ändern. Sie können eine der Zeilen manuell auskommentieren, um die Zuordnung zwischen Relaisschaltung und Laststromkreis genauer darzustellen.

### Mehr erfahren

Pin 3 des Relais ist normalerweise offen und wird nur dann geschlossen, wenn der Kontaktor aktiv ist; Pin 4 ist normalerweise geschlossen und öffnet sich, wenn der Kontaktor aktiviert wird. Pin 1 ist mit Pin 6 verbunden und stellt den gemeinsamen Anschluss des Laststromkreises dar.

Durch das Umschalten eines Endes des Laststromkreises von Pin 3 auf Pin 4 erhalten Sie genau den entgegengesetzten Betriebszustand.

### 3. Ton & Anzeige & Bewegung

## 6.21 3.1 - Piepton

Der aktive Summer ist ein typisches digitales Ausgabegerät, das genauso einfach zu bedienen ist wie eine LED zum Leuchten zu bringen!

- *Summer*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

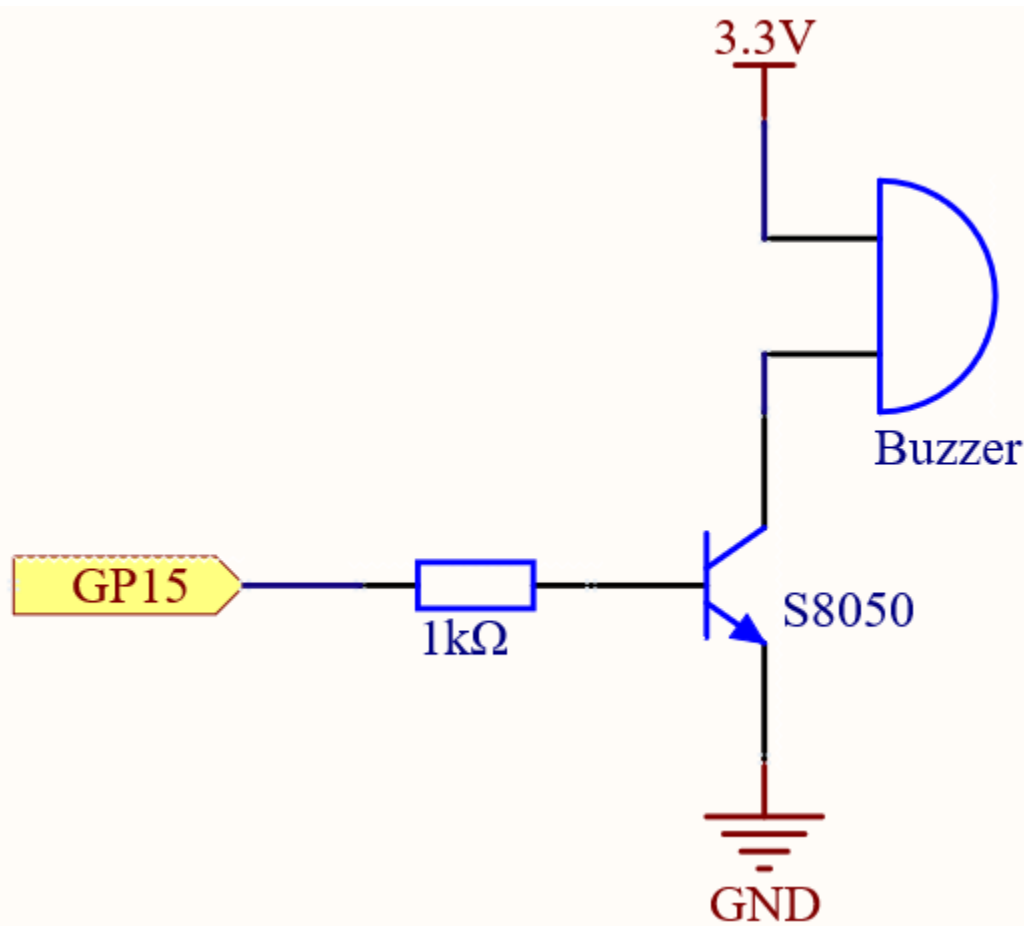
Es ist durchaus praktisch, ein gesamtes Set zu kaufen, hier der Link:

Bezeichnung	ELEMENTE IN DIESEM KIT	KAUF-LINK
Kepler-Kit	450+	

Sie können die Teile auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1(S8050)	
6	<i>Widerstand</i>	1(1K)	
7	Aktiver <i>Summer</i>	1	

## Schaltplan



Wenn der GP15-Ausgang auf „High“ geschaltet ist, lässt der 1K-Strombegrenzungswiderstand (zum Schutz des Transistors) den S8050 (NPN-Transistor) durchschalten, sodass der Summer ertönt.

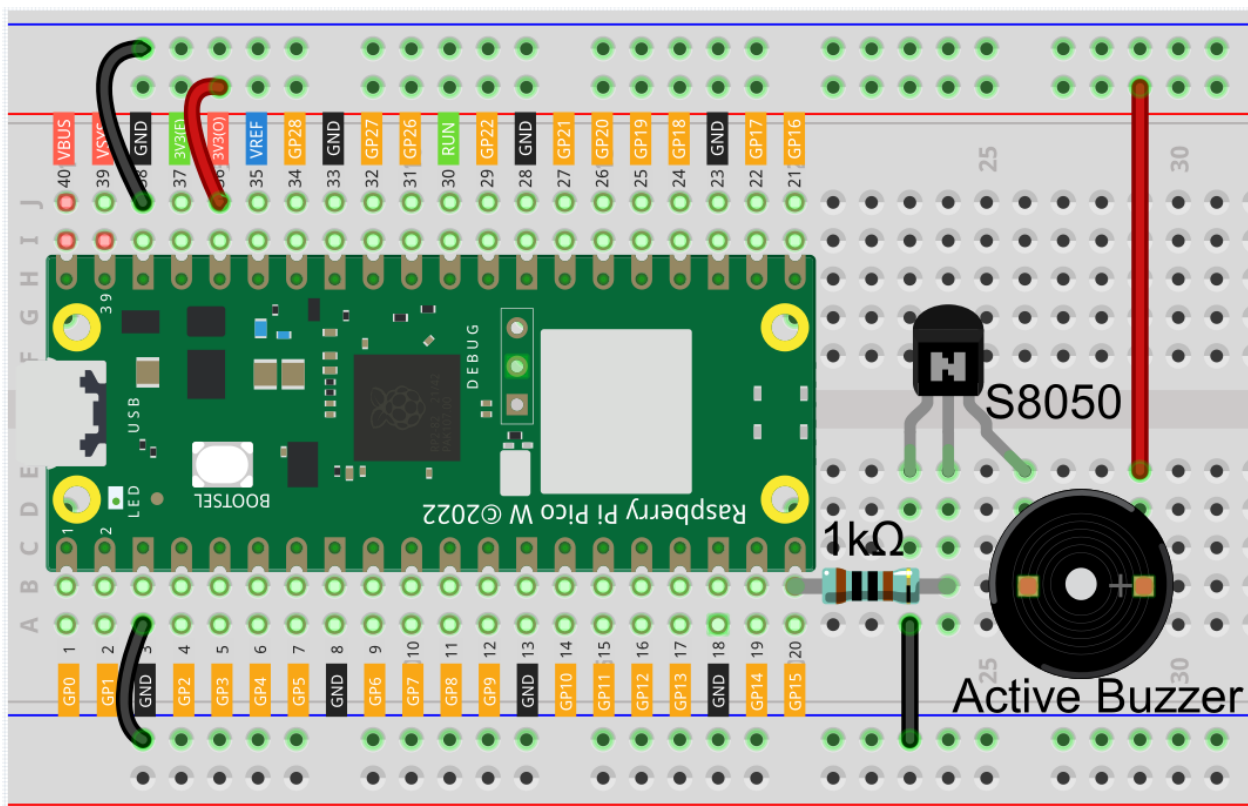
Die Aufgabe des S8050 (NPN-Transistor) ist es, den Strom zu verstärken und den Summer lauter klingen zu lassen. Tatsächlich könnten Sie den Summer auch direkt an GP15 anschließen, würden jedoch feststellen, dass der Ton dann leiser ist.

### Verkabelung

Im Kit sind zwei verschiedene Summertypen enthalten. Wir benötigen den aktiven Summer. Drehen Sie sie um, der versiegelte Rücken (nicht die freiliegende PCB) ist der, den wir verwenden möchten.



Für den Betrieb des Summers ist ein Transistor erforderlich, hier verwenden wir den S8050 (NPN-Transistor).



### Code

**Bemerkung:**

- Die Datei `3.1_beep.ino` finden Sie im Pfad `kepler-kit-main/arduino/3.1_beep`.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, vor dem Klicken auf die Schaltfläche **Hochladen** das Board (Raspberry Pi Pico) und den korrekten Port auszuwählen.

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

## 6.22 3.2 - Individueller Ton

Im vorherigen Projekt haben wir einen aktiven Summer verwendet. Diesmal greifen wir auf einen passiven Summer zurück.

Ähnlich wie der aktive Summer funktioniert auch der passive Summer auf Grundlage der elektromagnetischen Induktion. Der Unterschied besteht darin, dass ein passiver Summer keine eigene Schwingungsquelle hat. Deshalb gibt er keinen Ton ab, wenn Gleichstromsignale verwendet werden. Dies ermöglicht es jedoch dem passiven Summer, seine eigene Schwingungsfrequenz anzupassen und unterschiedliche Töne wie „Do, Re, Mi, Fa, Sol, La, Si“ auszugeben.

Lassen Sie den passiven Summer eine Melodie spielen!

- *Summer*

### Erforderliche Bauteile

Für dieses Projekt benötigen wir die folgenden Bauteile.

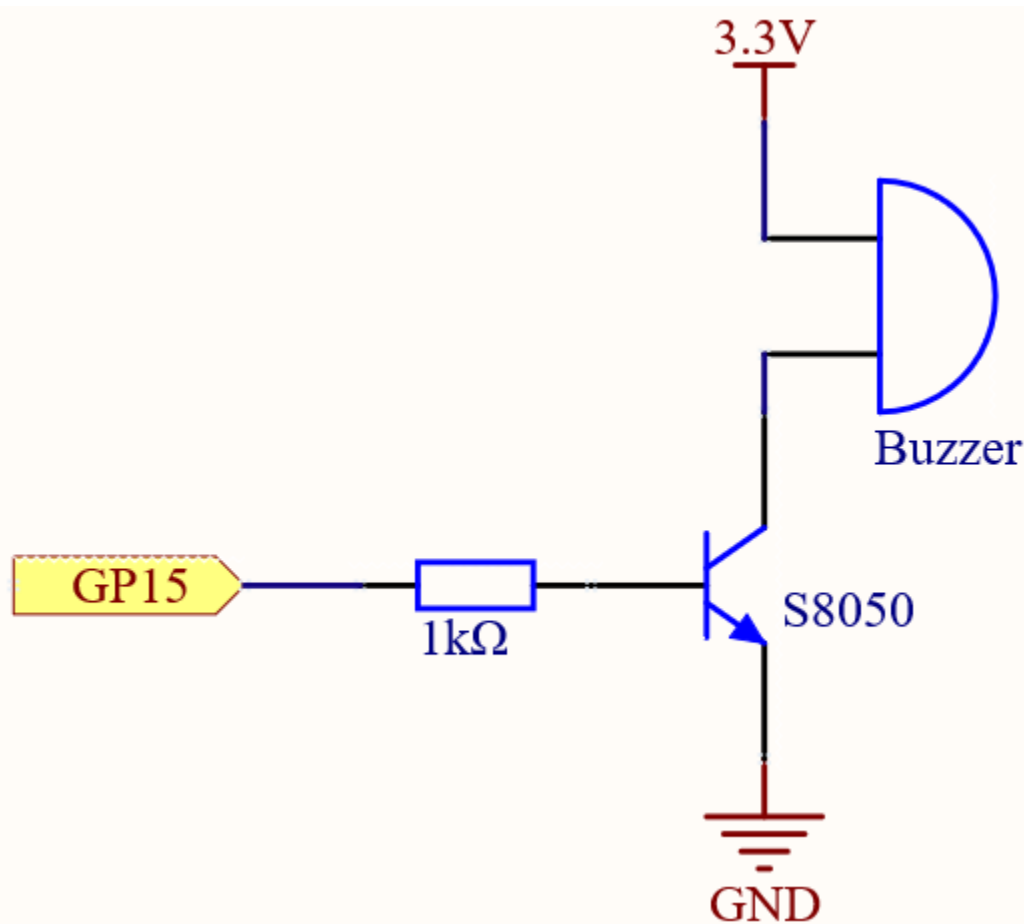
Ein komplettes Set zu kaufen ist definitiv praktisch, hier ist der Link:

Bezeichnung	ELEMENTE IN DIESEM KIT	KAUF-LINK
Kepler Kit	450+	

Alternativ können Sie die Teile auch einzeln über die folgenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1 (S8050)	
6	<i>Widerstand</i>	1 (1K)	
7	Passive <i>Summer</i>	1	

### Schaltplan



Wenn der GP15-Ausgang hoch ist, leitet der S8050 (NPN-Transistor) nach dem 1K-Strombegrenzungswiderstand (zum Schutz des Transistors) den Strom, sodass der Summer ertönt.

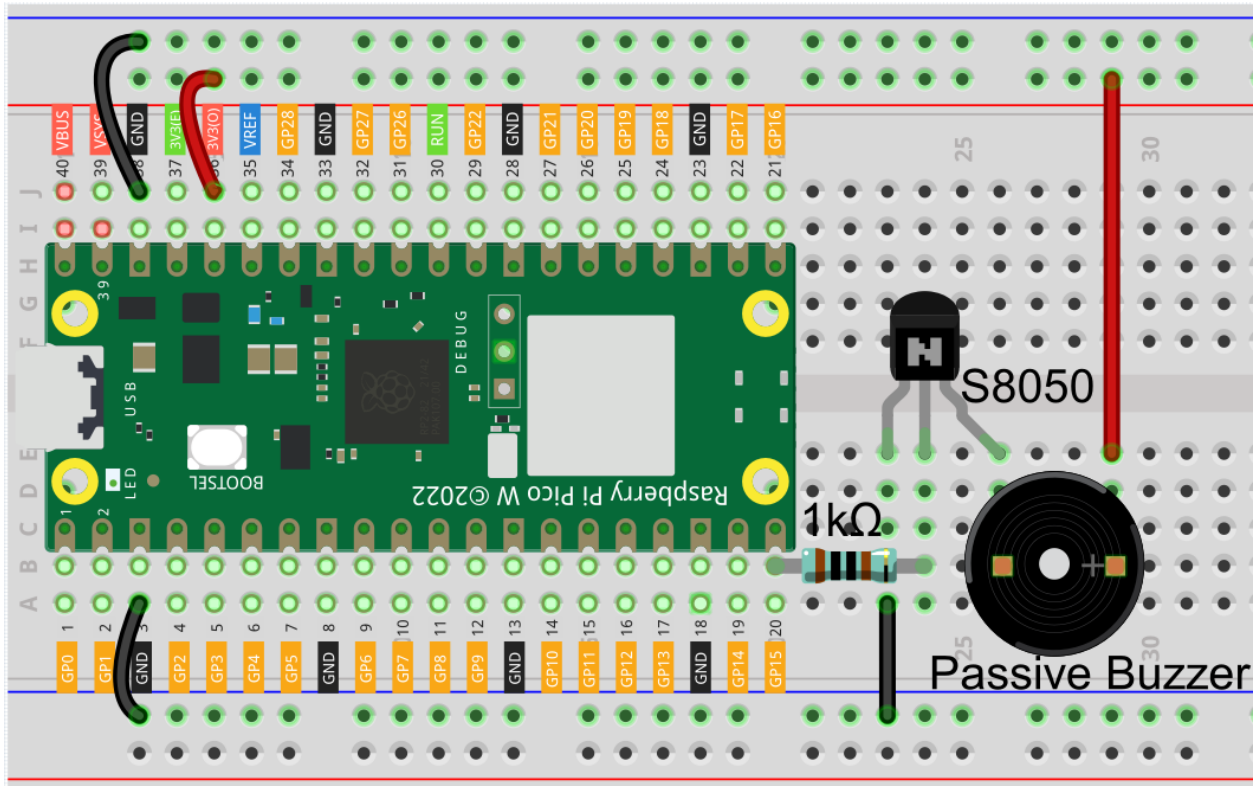
Die Aufgabe des S8050 (NPN-Transistor) besteht darin, den Strom zu verstärken und den Klang des Summers lauter zu machen. Tatsächlich könnten Sie den Summer auch direkt an GP15 anschließen, würden jedoch feststellen, dass der Ton leiser ist.

### Verkabelung



Im Kit sind zwei Summer enthalten; wir verwenden einen passiven Summer (einen mit freiliegender Leiterplatte auf der Rückseite).

Für die Funktion des Summers ist ein Transistor erforderlich; hier verwenden wir den S8050.



## Code

### Bemerkung:

- Die Datei `3.2_custom_tone.ino` finden Sie unter dem Pfad `kepler-kit-main/arduino/3.2_custom_tone`.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, die Platine (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.

### Wie funktioniert es?

Wenn dem passiven Summer ein digitales Signal gegeben wird, kann er nur die Membran bewegen, ohne einen Ton zu erzeugen.

Daher verwenden wir die Funktion `tone()` um das PWM-Signal zu erzeugen, das den passiven Summer zum Klingen bringt.

Diese Funktion hat drei Parameter:

- **pin**, der GPIO-Pin, der den Summer steuert.
- **frequency**, die Tonhöhe des Summers wird durch die Frequenz bestimmt; je höher die Frequenz, desto höher die Tonhöhe.
- **Duration**, die Dauer des Tons.
- `tone`

### Mehr erfahren

Wir können den spezifischen Ton gemäß der Grundfrequenz des Klaviers simulieren, um ein vollständiges Musikstück zu spielen.

- [Piano key frequencies - Wikipedia](#)

---

### Bemerkung:

- Die Datei `3.2_custom_tone_2.ino` finden Sie unter dem Pfad `kepler-kit-main/arduino/3.2_custom_tone_2`.
  - Oder kopieren Sie diesen Code in die **Arduino IDE**.
  - Vergessen Sie nicht, die Platine (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.
- 

## 6.23 3.3 WS2812 RGB-Strip

WS2812 ist eine intelligente LED-Lichtquelle, bei der die Steuerschaltung und der RGB-Chip in einem 5050-Komponentenpaket integriert sind. Sie enthält einen intelligenten digitalen Port-Datenlatch und eine Signalformungsverstärkungs-Schaltung. Darüber hinaus verfügt sie über einen präzisen internen Oszillator und einen programmierbaren konstanten Stromsteuerungsteil, der effektiv die hohe Konsistenz der Lichtfarbe jedes Pixels sicherstellt.

Das Datenübertragungsprotokoll verwendet den einzelnen NZR-Kommunikationsmodus. Nach dem Einschalten des Pixels erhält der DIN-Port Daten vom Controller, das erste Pixel sammelt die anfänglichen 24-Bit-Daten und sendet sie an den internen Datenlatch. Die restlichen Daten werden von der internen Signalformungsverstärkungs-Schaltung umgeformt und über den DO-Port an das nächste kaskadierte Pixel gesendet. Nach der Übertragung für jedes Pixel reduziert sich das Signal um 24 Bit. Das Pixel verwendet die Auto-Reshaping-Transmit-Technologie, sodass die Anzahl der kaskadierten Pixel nicht durch die Signalübertragung begrenzt ist, sondern nur von der Geschwindigkeit der Signalübertragung abhängt.

- [WS2812 RGB 8-LED-Streifen](#)

### Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

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

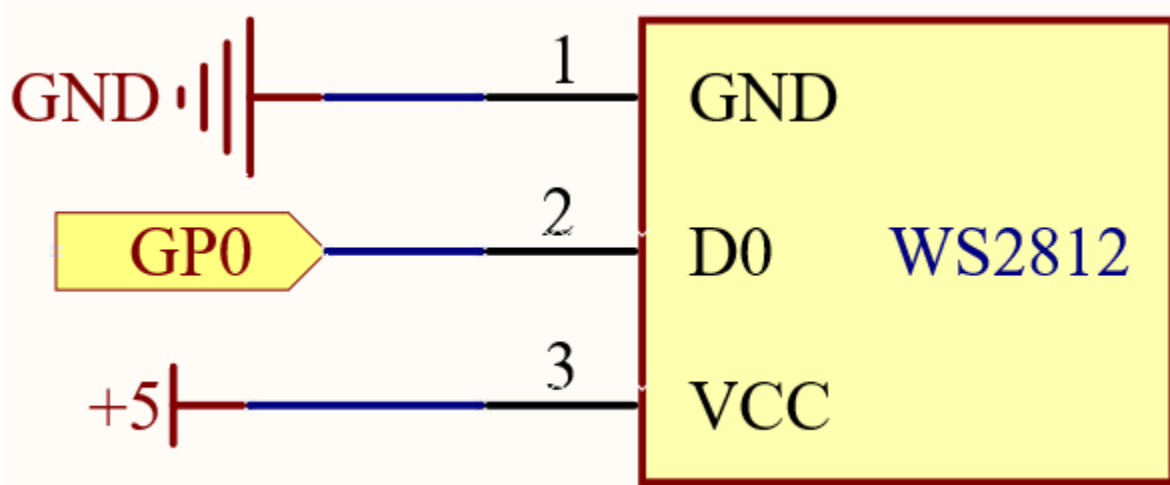
Bezeichnung	ARTIKEL IN DIESEM KIT	KAUF-LINK
Kepler-Kit	450+	

Sie können die Komponenten auch einzeln über die folgenden Links kaufen.

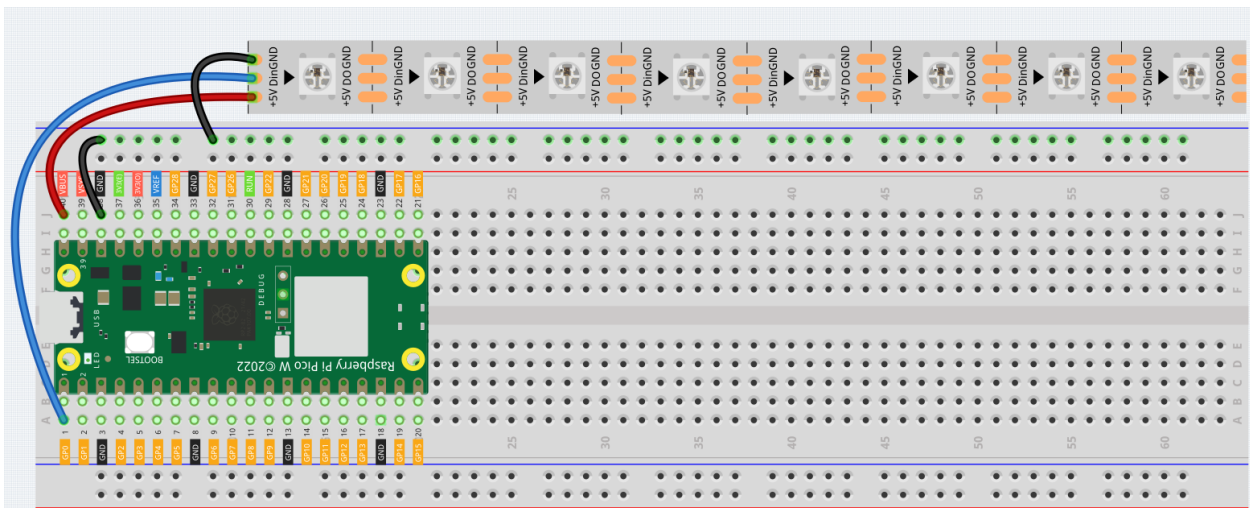


SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>WS2812 RGB 8-LED-Streifen</i>	1	

### Schaltplan



### Verdrahtung



**Warnung:** Achten Sie besonders auf den Strombedarf.

Obwohl der LED-Strip mit einer beliebigen Anzahl von LEDs im Pico W verwendet werden kann, ist die Leistung

seines VBUS-Pins begrenzt. Hier werden wir acht LEDs verwenden, die sicher sind. Wenn Sie jedoch mehr LEDs verwenden möchten, benötigen Sie eine separate Stromversorgung.

### Code

---

#### Bemerkung:

- Sie können die Datei `3.3_rgb_led_strip.ino` im Verzeichnis `kepler-kit-main/arduino/3.3_rgb_led_strip` öffnen.
  - Oder kopieren Sie diesen Code in die **Arduino IDE**.
  - Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.
  - Die Bibliothek `Adafruit_NeoPixel` wird hier verwendet. Bitte beziehen Sie sich auf [Bibliotheken hinzufügen](#) für das Hinzufügen in die Arduino IDE.
- 

Wählen Sie einige Ihrer Lieblingsfarben aus und zeigen Sie sie auf dem RGB-LED-Strip an!

#### Wie funktioniert es?

Ein Objekt vom Typ `Adafruit_NeoPixel` wird deklariert, welches an `PIXEL_PIN` angeschlossen ist und auf dem Streifen befinden sich `PIXEL_COUNT` RGB-LEDs.

```
#define PIXEL_PIN    0
#define PIXEL_COUNT 8

// Declare our NeoPixel strip object:
Adafruit_NeoPixel strip(PIXEL_COUNT, PIXEL_PIN, NEO_GRB + NEO_KHZ800);
// Argument 1 = Number of pixels in NeoPixel strip
// Argument 2 = Arduino pin number (most are valid)
// Argument 3 = Pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
//   NEO_RGBW    Pixels are wired for RGBW bitstream (NeoPixel RGBW products)
```

Streifenobjekt initialisieren und alle Pixel auf ‚aus‘ setzen.

#### Funktionen

- `strip.begin()` : NeoPixel-Streifenobjekt initialisieren (ERFORDERLICH).
- `strip.setPixelColor(index, color)` : Pixel-Farbe (im RAM) setzen, die `color` muss ein einzelner ‚gepackter‘ 32-Bit-Wert sein.
- `strip.Color(red, green, blue)` : Farbe als einzelner ‚gepackter‘ 32-Bit-Wert.
- `strip.show()` : Streifen mit neuem Inhalt aktualisieren.

#### Mehr erfahren

Wir können zufällige Farben generieren und ein farbenfrohes, fließendes Licht erzeugen.

---

#### Bemerkung:

- Sie können die Datei `3.3_rgb_led_strip_flowring.ino` im Pfad `kepler-kit-main/arduino/3.3_rgb_led_strip_flowring` öffnen.
  - Oder diesen Code in die **Arduino IDE** kopieren.
  - Vergessen Sie nicht, die Platine (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf die **Hochladen**-Schaltfläche klicken.
- 

Oder lassen Sie diesen WS2812 LED-Streifen in einem Regenbogenzyklus um das Farbrad (Bereich 65535) rotieren.

---

**Bemerkung:**

- Sie können die Datei `3.3_rgb_led_strip_rainbow.ino` im Pfad `kepler-kit-main/arduino/3.3_rgb_led_strip_rainbow` öffnen.
  - Oder diesen Code in die **Arduino IDE** kopieren.
  - Vergessen Sie nicht, die Platine (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf die **Hochladen**-Schaltfläche klicken.
- 

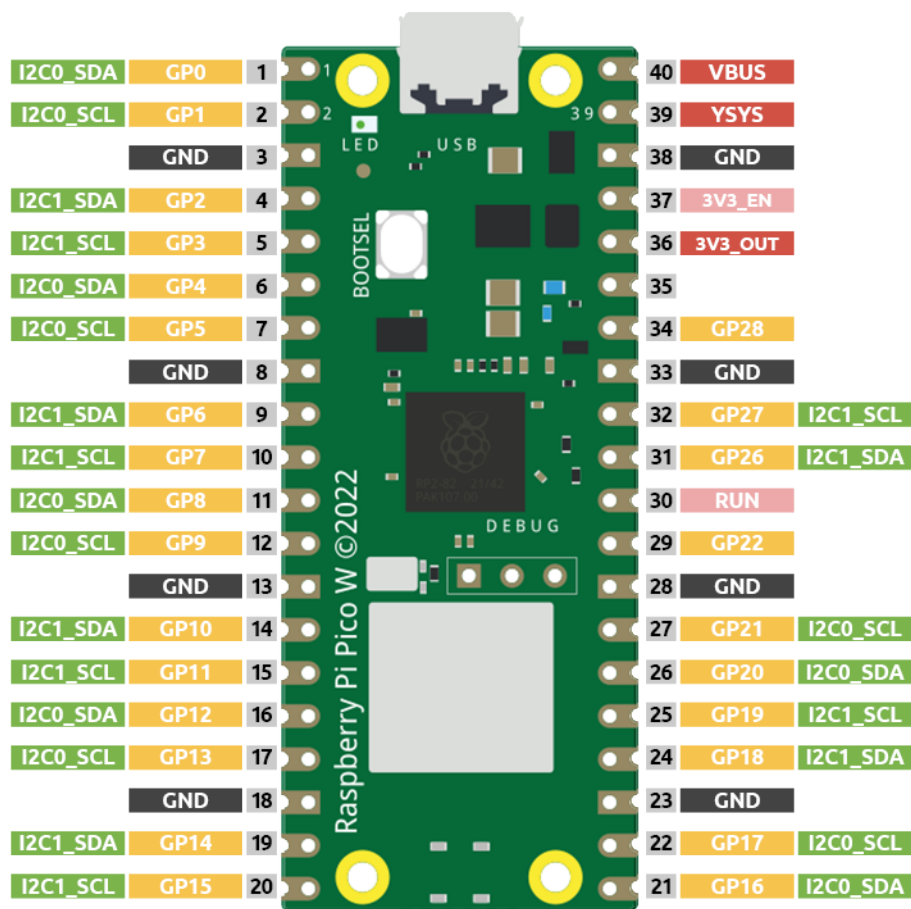
- `strip.getPixelColor(index)` : Die Farbe eines zuvor eingestellten Pixels abfragen.
- `strip.ColorHSV(pixelHue)` : Farbton, Sättigung und Wert in eine ‚gepackte‘ 32-Bit-RGB-Farbe umwandeln, die an `setPixelColor()` oder andere RGB-kompatible Funktionen übergeben werden kann.
- `strip.gamma32()` : Ermöglicht eine „echtere“ Farbwiedergabe, bevor sie jedem Pixel zugewiesen wird.

## 6.24 3.4 - Flüssigkristallanzeige

Das LCD1602 ist ein Zeichen-Typ-Flüssigkristallanzeige, auf dem gleichzeitig 32 (16\*2) Zeichen dargestellt werden können.

Wie allgemein bekannt ist, haben LCDs und andere Displays, obwohl sie die Mensch-Maschine-Interaktion erheblich bereichern, einen gemeinsamen Nachteil. Wenn sie an einen Controller angeschlossen sind, belegen sie mehrere I/O-Ports, was besonders problematisch ist, wenn der Controller nicht viele externe Ports hat. Dies schränkt auch andere Funktionen des Controllers ein. Um dieses Problem zu lösen, wurde das LCD1602 mit einem I2C-Bus entwickelt.

- [\*I2C LCD1602\*](#)
- [Inter-Integrated Circuit - Wikipedia](#)



In diesem Projekt verwenden wir die I2C0-Schnittstelle, um das LCD1602 zu steuern und Text anzuzeigen.

**Benötigte Komponenten**

Für dieses Projekt benötigen wir die folgenden Komponenten.

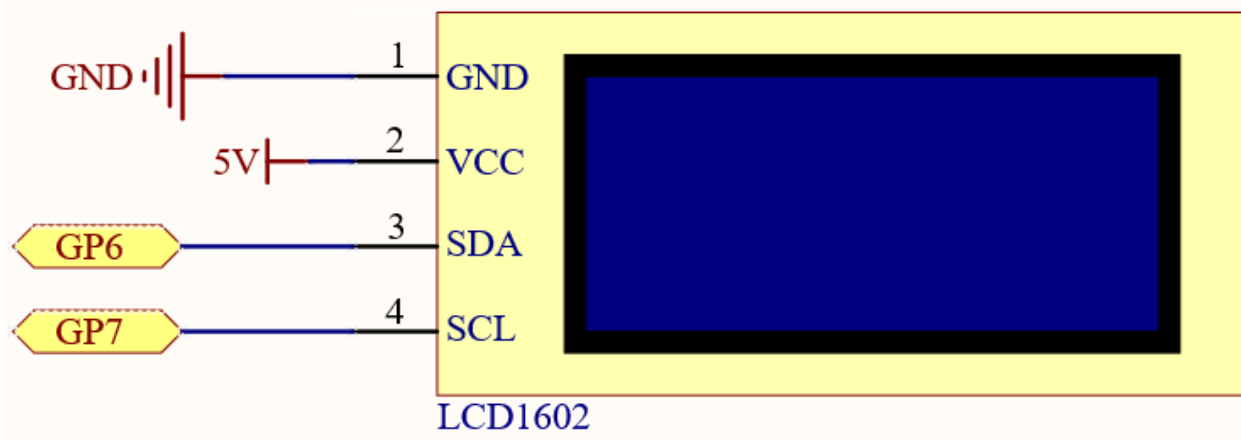
Es ist definitiv praktisch, ein ganzes Kit zu kaufen. Hier ist der Link:

Name	TEILE IN DIESEM KIT	KAUF-LINK
Kepler Kit	450+	

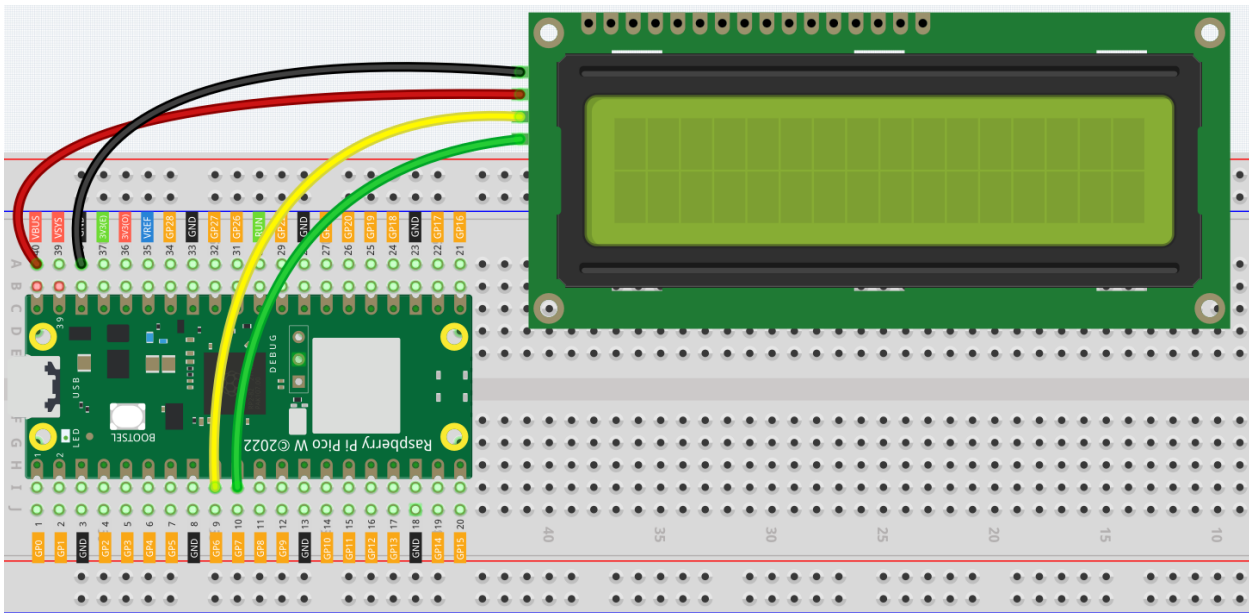
Sie können die Teile auch separat über die folgenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>I2C LCD1602</i>	1	

## Schaltplan



## Verkabelung



## Code

## Bemerkung:

- Sie können die Datei `3.4_liquid_crystal_display.ino` im Pfad `kepler-kit-main/arduino/3.4_liquid_crystal_display` öffnen.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf die **Upload**-Taste klicken.
- Die Bibliothek `LiquidCrystal_I2C` wird hier verwendet. Bitte lesen Sie [Bibliotheken hinzufügen](#), um sie der Arduino IDE hinzuzufügen.

Nach dem Start des Programms werden zwei Textzeilen nacheinander auf dem LCD angezeigt und dann wieder verschwinden.

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

---

### Wie funktioniert das?

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

```
#include "LiquidCrystal_I2C.h"
```

### Bibliotheksfunktionen

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

Erstellt eine neue Instanz der Klasse `LiquidCrystal_I2C`, die ein bestimmtes, an Ihr Arduino-Board angeschlossenes LCD repräsentiert.

**lcd\_Addr:** Die Standardadresse des LCD beträgt 0x27. **lcd\_cols:** Das LCD1602 hat 16 Spalten. **lcd\_rows:** Das LCD1602 hat 2 Reihen.

```
void init()
```

Initialisiert das LCD.

```
void backlight()
```

Schaltet die (optionale) Hintergrundbeleuchtung ein.

```
void nobacklight()
```

Schaltet die (optionale) Hintergrundbeleuchtung aus.

```
void display()
```

Schaltet die LCD-Anzeige ein.

```
void nodisplay()
```

Schaltet die LCD-Anzeige schnell aus.

```
void clear()
```

Löscht die Anzeige und setzt die Cursorposition zurück.

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

Setzt den Cursor auf die Position col,row.

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

Gibt den Text auf dem LCD aus.

**data:** Die auszugebende Daten (char, byte, int, long oder String).

**BASE (optional):** Die Basis, in der Zahlen ausgegeben werden: BIN für Binär (Basis 2), DEC für Dezimal (Basis 10), OCT für Oktal (Basis 8), HEX für Hexadezimal (Basis 16).

### Weitere Informationen

Laden Sie den Code auf das Pico W. Die im seriellen Monitor eingegebenen Inhalte werden auf dem LCD angezeigt.

#### Bemerkung:

- Sie finden die Datei `3.4_liquid_crystal_display_2.ino` im Verzeichnis `kepler-kit-main/arduino/3.4_liquid_crystal_display_2`.
- Oder kopieren Sie diesen Code direkt in die **Arduino IDE**.
- Vergessen Sie nicht, das richtige Board (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf **Hochladen** klicken.

Zusätzlich zur Datenerfassung von elektronischen Komponenten kann das Pico W auch Daten aus dem seriellen Monitor lesen. Dazu können Sie `Serial.read()` als Steuerelement des Schaltungsexperiments verwenden.

Starten Sie die serielle Kommunikation in `setup()` und setzen Sie die Datenrate auf 9600.

```
Serial.begin(9600);
```

Der Zustand des seriellen Monitors wird in `loop()` überprüft. Die Datenverarbeitung erfolgt nur, wenn Daten empfangen werden.

```
if (Serial.available() > 0){}
```

Leeren Sie den Bildschirm.

```
lcd.clear();
```

Liest den Eingabewert im seriellen Monitor und speichert ihn in der Variable `incomingByte`.

```
char incomingByte = Serial.read();
```

Zeigt jeden eingegebenen Buchstaben auf dem LCD an und überspringt das Zeilenumbruchzeichen.

```
while (Serial.available() > 0) {
  char incomingByte=Serial.read();
  if(incomingByte==10){break;}// skip the line-feed character
  lcd.print(incomingByte);// display each character to the LCD
}
```

- [Serial Read](#)

## 6.25 3.5 - Kleiner Ventilator

Nun nutzen wir den TA6586, um den Gleichstrommotor in beide Richtungen drehen zu lassen. Da der Gleichstrommotor einen vergleichsweise hohen Strombedarf hat, verwenden wir aus Sicherheitsgründen ein Spannungsmodul zur Stromversorgung des Motors.

- *Gleichstrommotor*
- *TA6586 - Motorsteuerungs-Chip*
- `cpn_power_module`

#### Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Bauteile:

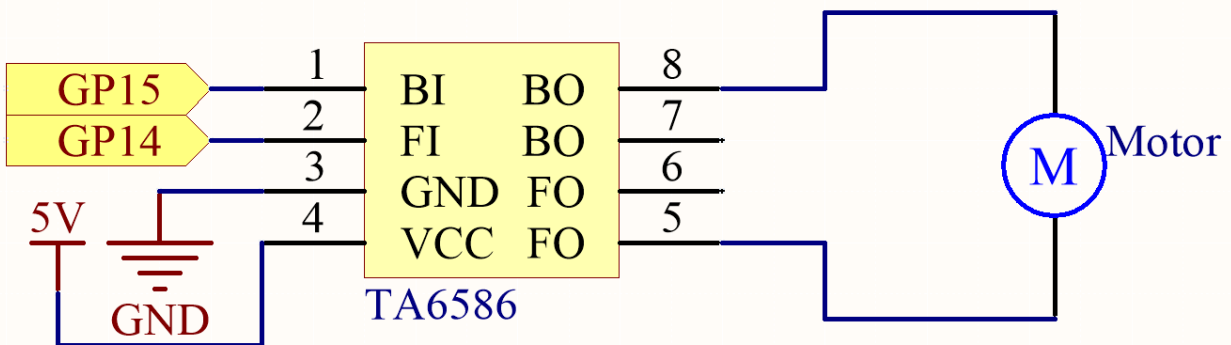
Ein komplettes Set zu kaufen, ist definitiv praktisch. Hier ist der Link dazu:

Bezeichnung	ARTIKEL IN DIESEM KIT	KAUF-LINK
Kepler-Kit	450+	

Alternativ können Sie die einzelnen Komponenten auch über die unten aufgeführten Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>TA6586 - Motorsteuerungs-Chip</i>	1	
6	<i>Gleichstrommotor</i>	1	
7	<i>Li-Po-Lademodul</i>	1	
8	18650 Batterie	1	
9	Batteriehalter	1	

### Schaltplan

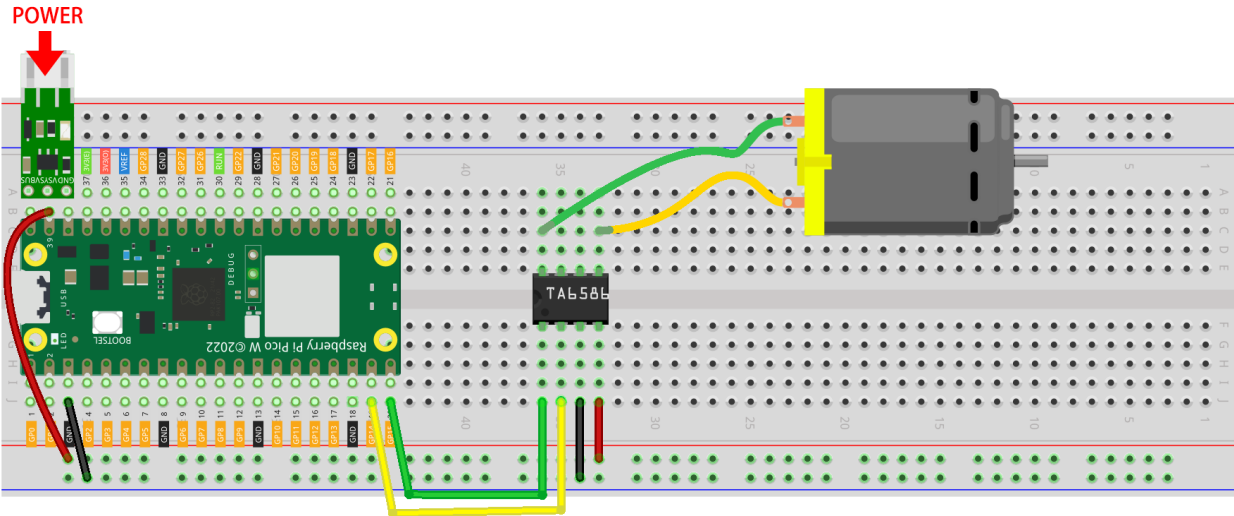


### Verkabelung

#### Bemerkung:

- Da Gleichstrommotoren einen hohen Strombedarf haben, verwenden wir hier aus Sicherheitsgründen ein Li-Po-Ladegerät-Modul zur Stromversorgung des Motors.
- Achten Sie darauf, dass Ihr Li-Po-Ladegerät-Modul gemäß dem Schaltplan verbunden ist. Andernfalls könnten Kurzschlüsse sowohl Ihre Batterie als auch die Schaltung beschädigen.





## Code

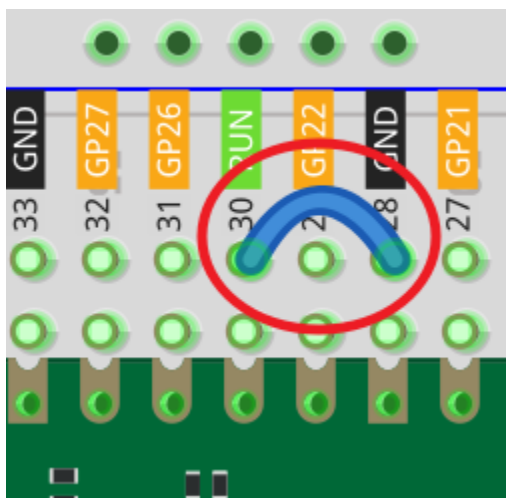
### Bemerkung:

- Die Datei `3.5_small_fan.ino` finden Sie im Verzeichnis `kepler-kit-main/arduino/3.5_small_fan`.
- Alternativ können Sie den Code auch in die **Arduino-IDE** kopieren.
- Vergewissern Sie sich, dass Sie das richtige Board (Raspberry Pi Pico) und den korrekten Port ausgewählt haben, bevor Sie auf **Hochladen** klicken.

Sobald das Programm läuft, wird der Motor in einem regelmäßigen Muster hin und her drehen.

### Bemerkung:

- Falls Sie den Code nicht erneut hochladen können, müssen Sie den **RUN**-Pin am Pico W mit einem Draht auf GND legen, um ihn zurückzusetzen. Danach entfernen Sie den Draht, um den Code erneut auszuführen.
- Dies liegt daran, dass der Motor mit zu hohem Strom arbeitet, was dazu führen kann, dass der Pico W die Verbindung zum Computer verliert.



## 6.26 3.6 - Pumpensteuerung

Kleine Kreislumpen eignen sich hervorragend für Projekte zur automatischen Pflanzenbewässerung. Sie können ebenfalls für den Bau kleiner, intelligenter Wasserspiele verwendet werden.

Als Antriebskomponente dient ein Elektromotor, der genau wie ein herkömmlicher Motor betrieben wird.

- *DC-Wasserpumpe*
- *Gleichstrommotor*
- *TA6586 - Motorsteuerungs-Chip*
- `cpn_power_module`

### Bemerkung:

1. Schließen Sie den Schlauch an den Motoranschluss an, tauchen Sie die Pumpe ins Wasser und schalten Sie sie ein.
2. Achten Sie darauf, dass der Wasserstand stets höher als der des Motors ist. Leerlauf kann den Motor durch Hitzegenerierung beschädigen und zusätzlich Lärm verursachen.
3. Falls Sie Pflanzen bewässern, sollte vermieden werden, dass Erde angesaugt wird, da dies die Pumpe verstopfen könnte.
4. Wenn kein Wasser aus dem Schlauch kommt, könnte Restwasser im Schlauch den Luftstrom blockieren und muss zuerst abgelassen werden.

### Erforderliche Komponenten

Für dieses Projekt werden die folgenden Bauteile benötigt.

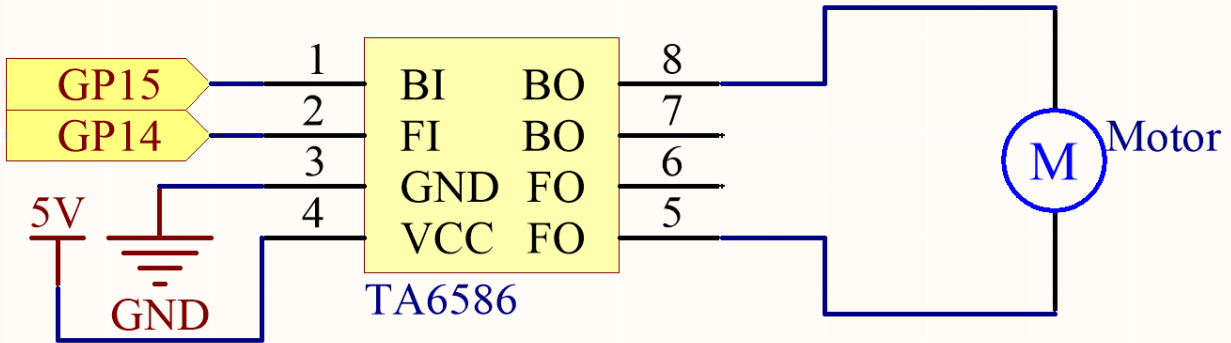
Ein Komplettsatz zu kaufen ist definitiv praktisch, hier ist der Link:

Bezeichnung	KOMPONENTEN IM SET	KAUF-LINK
Kepler-Kit	450+	

Sie können die Komponenten auch einzeln über die folgenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>TA6586 - Motorsteuerungs-Chip</i>	1	
6	<i>Li-Po-Lademodul</i>	1	
7	18650-Akku	1	
8	Batteriehalter	1	
9	<i>DC-Wasserpumpe</i>	1	

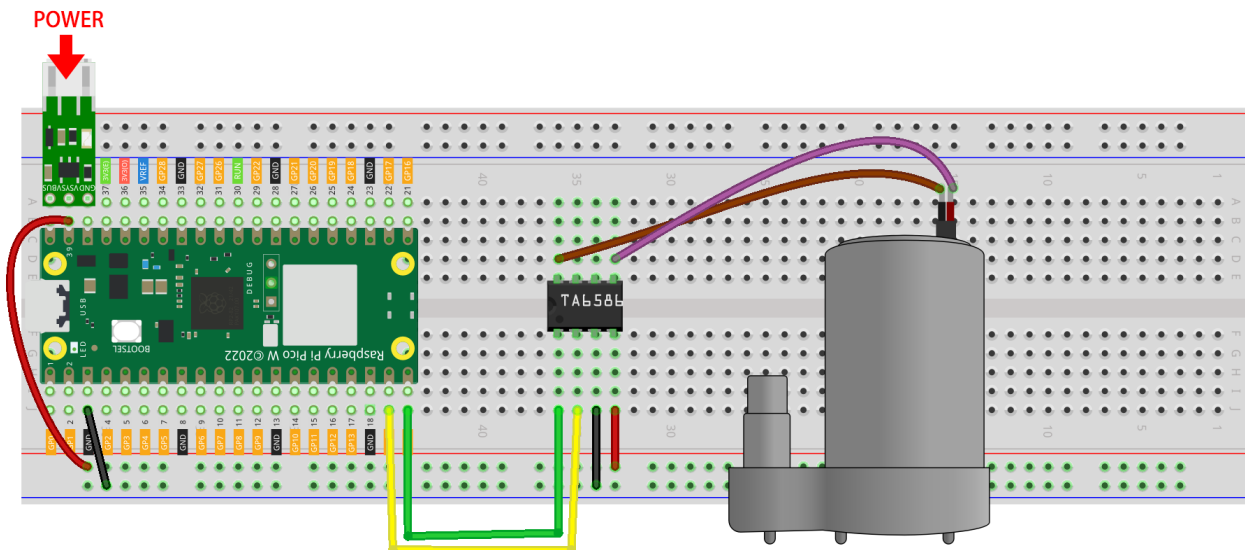
### Schaltplan



### Verkabelung

#### Bemerkung:

- Da Pumpen einen hohen Strombedarf haben, nutzen wir hier aus Sicherheitsgründen ein Li-Po-Ladegerät-Modul zur Stromversorgung des Motors.
- Achten Sie darauf, dass Ihr Li-Po-Ladegerät-Modul wie im Diagramm dargestellt angeschlossen ist. Andernfalls könnte ein Kurzschluss Ihren Akku und die Schaltung beschädigen.



### Code

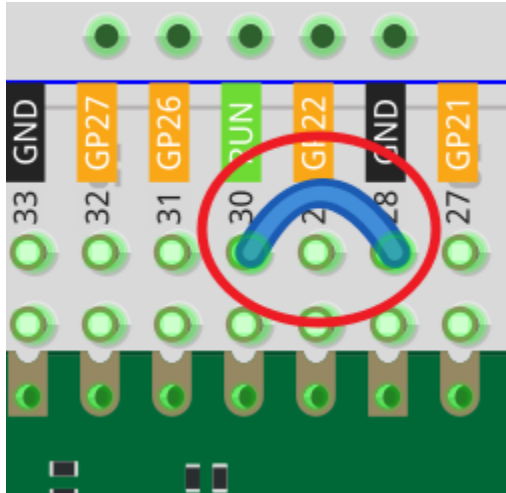
#### Bemerkung:

- Sie können die Datei `3.6_pumping.ino` im Pfad `kepler-kit-main/arduino/3.6_pumping` öffnen.
- Oder kopieren Sie den Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf die **Upload**-Schaltfläche klicken.

Nachdem der Code ausgeführt wurde, beginnt die Pumpe zu arbeiten, und Sie werden gleichzeitig sehen, wie das Wasser aus dem Schlauch fließt.

**Bemerkung:**

- Wenn ein erneutes Hochladen des Codes nicht möglich ist, verbinden Sie den **RUN**-Pin am Pico W mit einem Draht mit GND, um ihn zurückzusetzen. Dann entfernen Sie den Draht, um den Code erneut auszuführen.
- Dies liegt daran, dass der Motor mit zu hohem Strom betrieben wird, was dazu führen kann, dass der Pico W die Verbindung zum Computer verliert.



## 6.27 3.7 - Schwingender Servo

In diesem Set gibt es neben LED und passivem Summer auch ein Gerät, das durch ein PWM-Signal gesteuert wird: der Servo.

Ein Servo ist ein Positionsservo-Gerät, das für Steuerungssysteme geeignet ist, die ständige Winkeländerungen erfordern und aufrechterhalten können. Es wird häufig in hochwertigen ferngesteuerten Spielzeugen eingesetzt, wie Flugzeugen, U-Boot-Modellen und ferngesteuerten Robotern.

Jetzt versuchen Sie, den Servo schwingen zu lassen!

- *Servo*

### Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten:

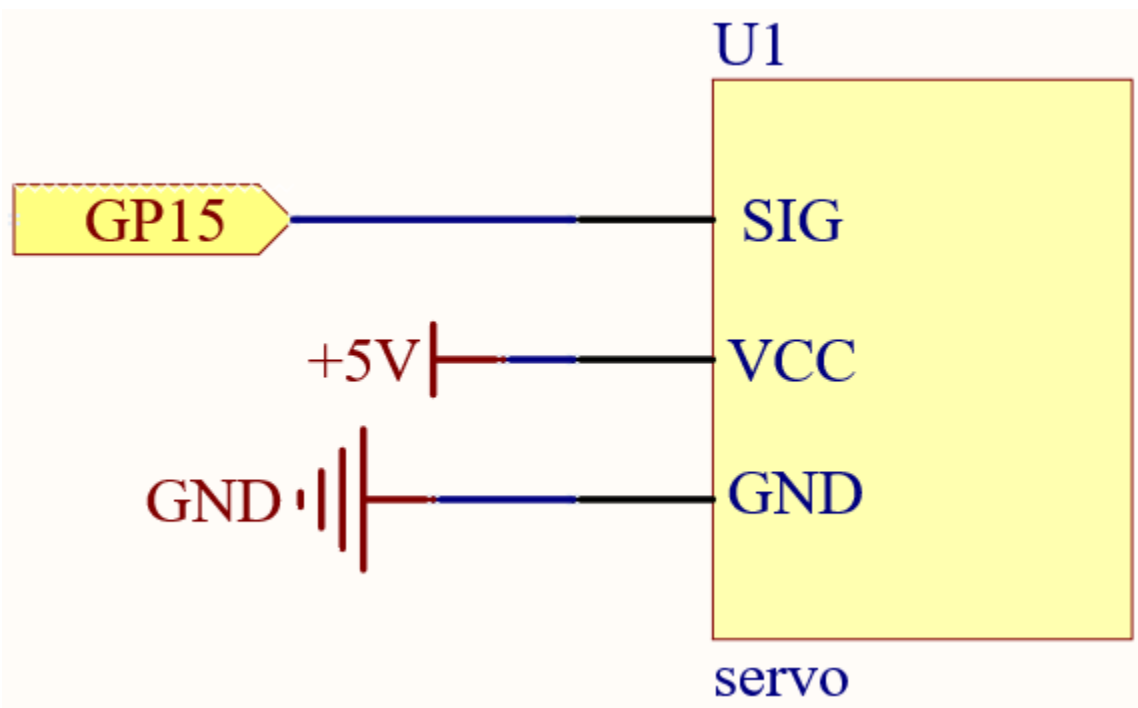
Es ist sicherlich praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM SET	KAUF-LINK
Kepler Kit	450+	

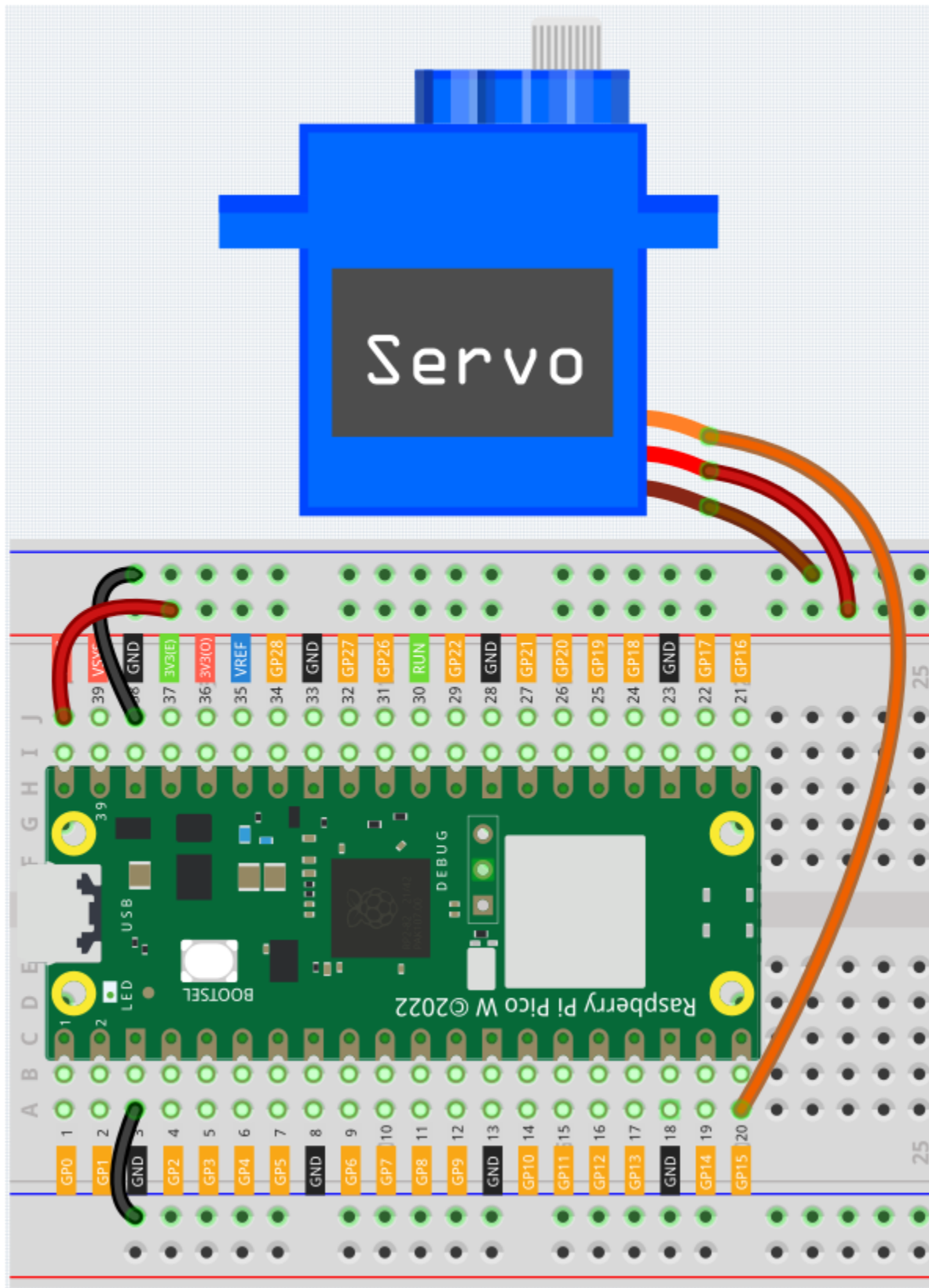
Sie können diese auch separat über die untenstehenden Links kaufen.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Servo</i>	1	

### Schaltplan



### Verkabelung



- Das orangefarbene Kabel ist das Signal und wird an GP15 angeschlossen.

- Das rote Kabel ist VCC und wird an VBUS(5V) angeschlossen.
- Das braune Kabel ist GND und wird an GND angeschlossen.

## Code

---

### Bemerkung:

- Sie können die Datei `3.7_swinging_servo.ino` im Pfad `kepler-kit-main/arduino/3.7_swinging_servo` öffnen.
  - Oder kopieren Sie diesen Code in die **Arduino IDE**.
  - Vergessen Sie nicht, die Platine (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.
- 

Wenn das Programm läuft, sehen wir, wie der Servoarm sich von 0° bis 180° hin und her bewegt.

### Wie funktioniert es?

Mit Hilfe der Bibliothek `Servo.h` können Sie den Servo leicht steuern.

```
#include <Servo.h>
```

### Bibliotheksfunktionen

```
Servo
```

Erstellen Sie ein **Servo**-Objekt, um einen Servo zu steuern.

```
uint8_t attach(int pin);
```

Verwandeln Sie einen Pin in einen Servo-Treiber. Ruft `pinMode` auf. Gibt 0 bei Fehler zurück.

```
void detach();
```

Gibt einen Pin vom Servo-Treiber frei.

```
void write(int value);
```

Setzt den Winkel des Servos in Grad, von 0 bis 180.

```
int read();
```

Gibt den mit dem letzten `write()`-Befehl eingestellten Wert zurück.

```
bool attached();
```

Gibt 1 zurück, wenn der Servo aktuell angeschlossen ist.

## 4. Controller

## 6.28 4.1 - Den Joystick bedienen

Wenn du häufig Videospiele spielst, dürfte dir der Joystick bestens bekannt sein. Er wird normalerweise verwendet, um die Spielfigur zu bewegen, den Bildschirm zu drehen usw.

Das Prinzip hinter der Fähigkeit des Joysticks, dem Computer unsere Bewegungen mitzuteilen, ist recht simpel. Man kann sich den Joystick als Kombination aus zwei senkrecht zueinander stehenden Potentiometern vorstellen. Diese beiden Potentiometer messen den analogen Wert des Joysticks in vertikaler und horizontaler Richtung, was in einem Wert (x,y) in einem rechtwinkligen Koordinatensystem resultiert.

Der Joystick dieses Sets verfügt auch über einen digitalen Eingang, der aktiviert wird, wenn der Joystick gedrückt wird.

- *Joystick-Modul*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

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

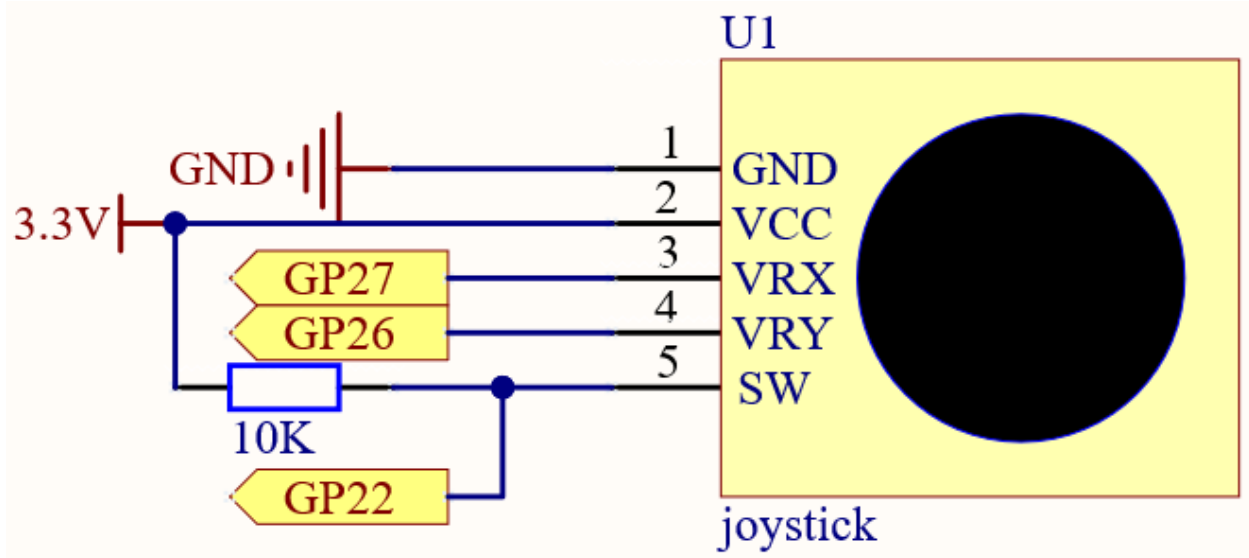
Name	ARTIKEL IN DIESEM SET	KAUF-LINK
Kepler Kit	450+	

Die Bauteile können auch einzeln über die folgenden Links erworben werden.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Joystick-Modul</i>	1	

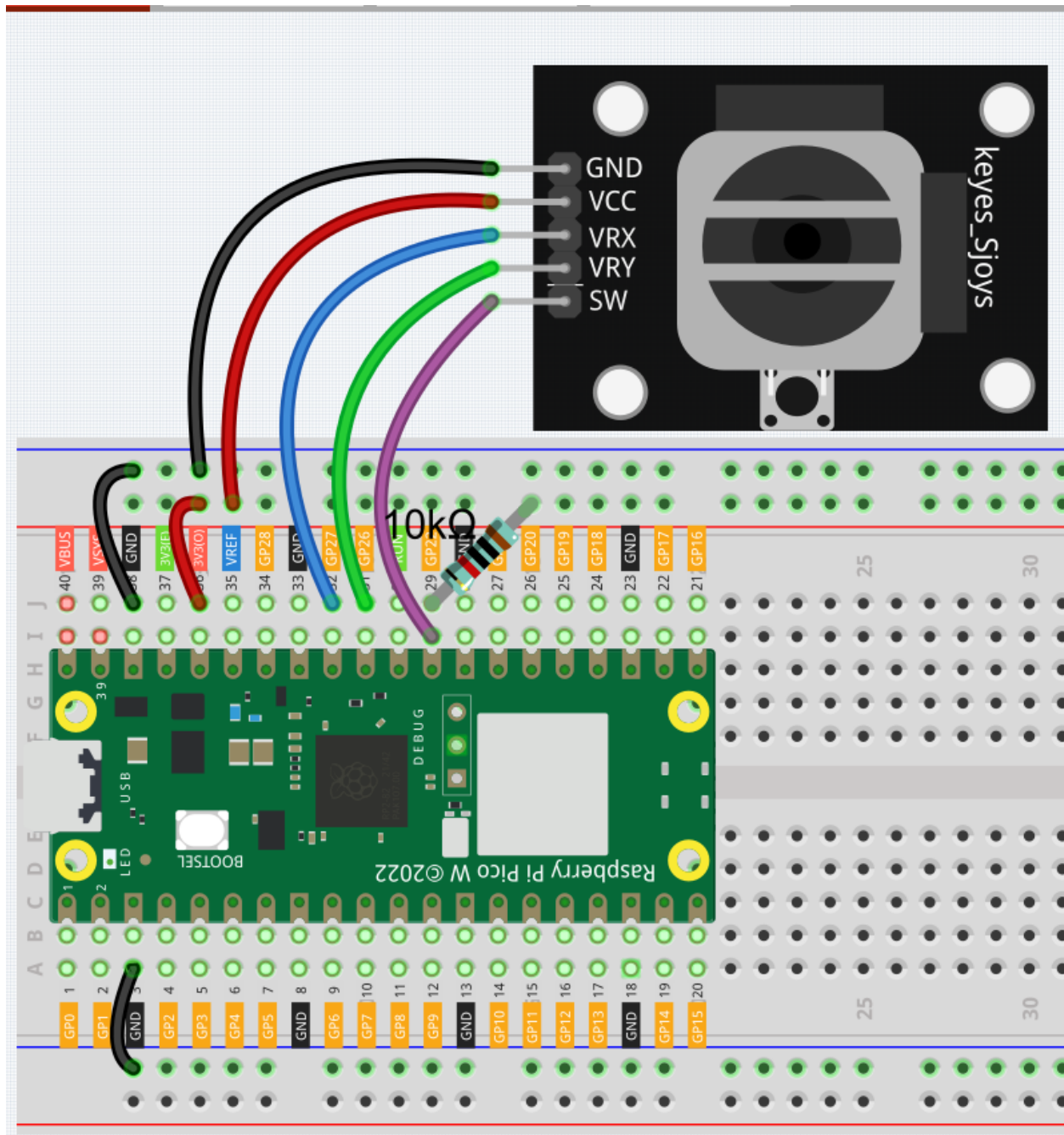
### Schaltplan





Der SW-Pin ist über einen 10K Pull-up-Widerstand angeschlossen. Der Grund dafür ist, ein stabiles hohes Signal am SW-Pin (Z-Achse) zu erhalten, wenn der Joystick nicht gedrückt wird; andernfalls befindet sich der SW in einem unbestimmten Zustand und der Ausgabewert kann zwischen 0/1 variieren.

#### Verdrahtung



## Code

### Bemerkung:

- Die Datei `4.1_toggle_the_joyostick.ino` finden Sie im Pfad `kepler-kit-main/arduino/4.1_toggle_the_joyostick`.
- Oder kopieren Sie den Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf den **Hochladen**-Button klicken.

Nach dem Ausführen des Programms gibt die Shell die x,y,z-Werte des Joysticks aus.

- Die Werte der x- und y-Achse sind analoge Werte, die zwischen 0 und 65535 variieren.
- Die Z-Achse hat einen digitalen Wert mit einem Status von 1 oder 0.

## 6.29 4.2 - 4x4 Tastenfeld

Das 4x4-Tastenfeld, auch als Matrix-Tastenfeld bekannt, besteht aus einer Matrix von 16 Tasten, die in einer einzigen Bedienoberfläche integriert sind.

Solche Tastenfelder findet man vor allem bei Geräten, die digitale Eingaben erfordern, wie zum Beispiel Taschenrechner, Fernbedienungen, Tastentelefone, Verkaufsautomaten, Geldautomaten, Zahlenschlösser und elektronische Türschlösser.

In diesem Projekt lernen wir, wie man ermittelt, welche Taste gedrückt wurde und den entsprechenden Tastenwert erhält.

- [4x4 Tastenfeld](#)
- [E.161 - Wikipedia](#)

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

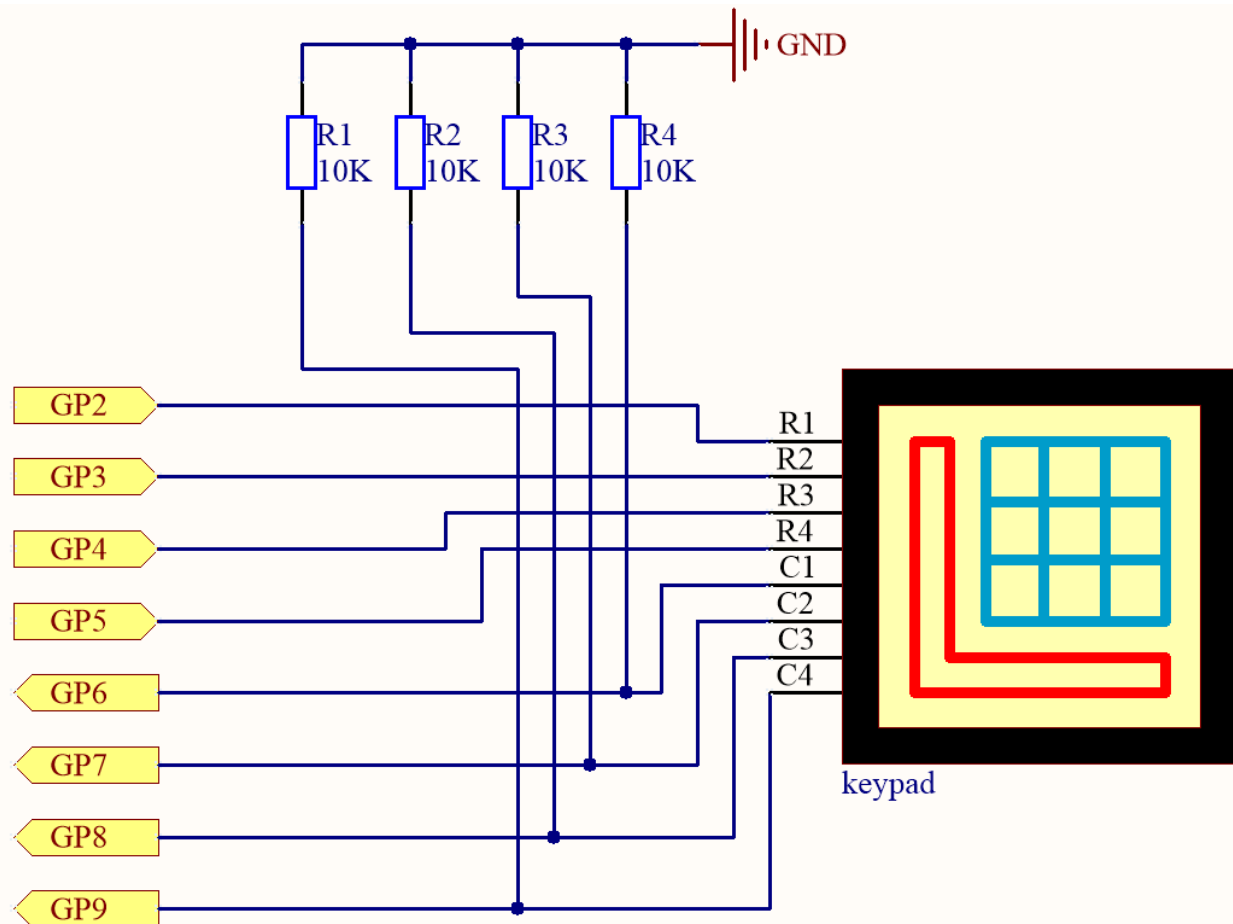
Es ist natürlich praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM SET	KAUF-LINK
Kepler Kit	450+	

Die Teile können aber auch einzeln über die untenstehenden Links gekauft werden.

SN	KOMPONENTENÜBERSICHT	AN-ZAHL	KAUF-LINK
1	<a href="#">Raspberry Pi Pico W</a>	1	
2	Micro-USB-Kabel	1	
3	<a href="#">Steckbrett</a>	1	
4	<a href="#">Jumperkabel</a>	Mehrere	
5	<a href="#">Widerstand</a>	4(10K)	
6	<a href="#">4x4 Tastenfeld</a>	1	

### Schaltplan

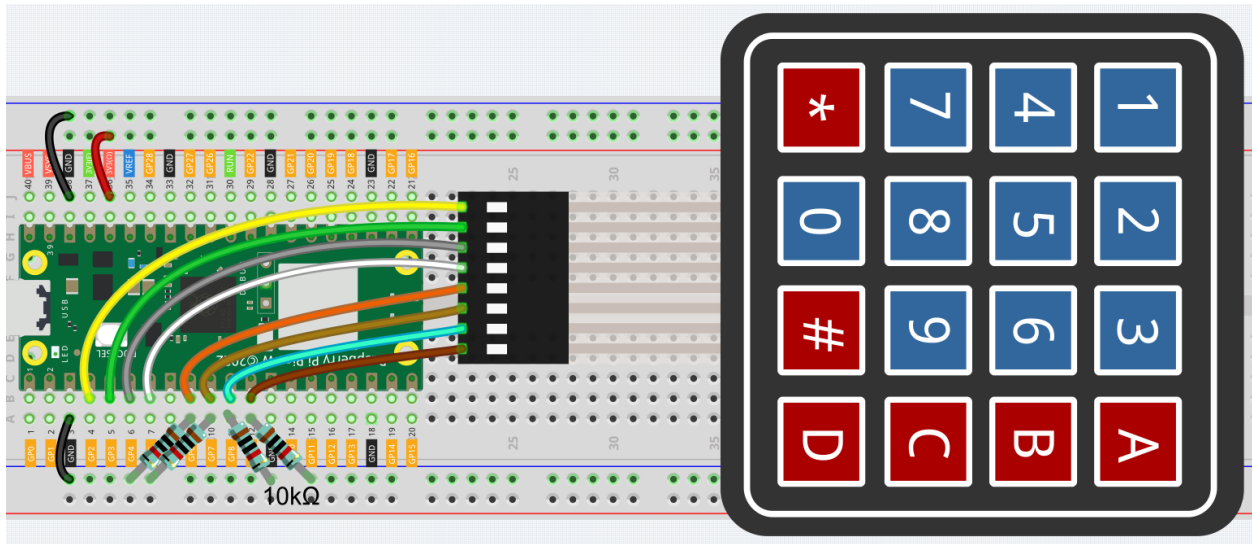


Vier Pull-down-Widerstände sind mit den jeweiligen Spalten des Matrix-Tastenfelds verbunden, damit G6 ~ G9 einen stabilen niedrigen Pegel erhalten, wenn keine Taste gedrückt ist.

Die Reihen des Tastenfelds (G2 ~ G5) sind so programmiert, dass sie einen hohen Pegel haben. Wird einer der Anschlüsse G6 ~ G9 als hoch gelesen, wissen wir, welche Taste gedrückt wurde.

Zum Beispiel wird bei einem hohen Signal auf G6 die Taste mit der Nummer 1 gedrückt; dies ist darauf zurückzuführen, dass die Steuerpins dieser Taste G2 und G6 sind. Wenn die Taste gedrückt wird, werden G2 und G6 miteinander verbunden und G6 ist ebenfalls hoch.

### Verdrahtung



Um die Verdrahtung zu vereinfachen, sind im obigen Schema die Spalte und die Reihe des Matrix-Tastenfelds sowie die 10K-Widerstände gleichzeitig in die Löcher eingesteckt, in denen sich G6 ~ G9 befinden.

### Code

#### Bemerkung:

- Die Datei `4.2_4x4_keypad.ino` finden Sie im Verzeichnis `kepler-kit-main/arduino/4.2_4x4_keypad`.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.
- Die Bibliothek Keypad wird hier verwendet. Bitte beachten Sie [Bibliotheken hinzufügen](#) für weitere Informationen zur Integration in die Arduino IDE.

Nach dem Ausführen des Programms wird die Shell die Tasten ausgeben, die Sie auf dem Tastenfeld gedrückt haben.

#### Funktionsweise

Mithilfe der Bibliothek `Keypad.h` können Sie das Tastenfeld einfach nutzen.

```
#include <Keypad.h>
```

Bibliotheksfunktionen:

```
Keypad(char *userKeymap, byte *row, byte *col, byte numRows, byte numCols)
```

Initialisiert die interne Tastenbelegung entsprechend `userKeymap`.

`userKeymap`: Die Symbole auf den Tasten des Tastenfelds.

`row, col`: Pin-Konfiguration.

`numRows, numCols`: Größe des Tastenfelds.

```
char getKey()
```

Gibt die gedrückte Taste zurück, falls vorhanden. Diese Funktion ist nicht blockierend.

## 6.30 4.3 - Elektroden-Tastatur

Der MPR121 ist eine gute Wahl, wenn Sie Ihrem Projekt eine Vielzahl von Berührungsschaltern hinzufügen möchten. Das Modul besitzt Elektroden, die mit Leitern erweitert werden können. Verbinden Sie die Elektroden beispielsweise mit einer Banane, verwandeln Sie diese in einen Berührungsschalter.

- *MPR121 Modul*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten:

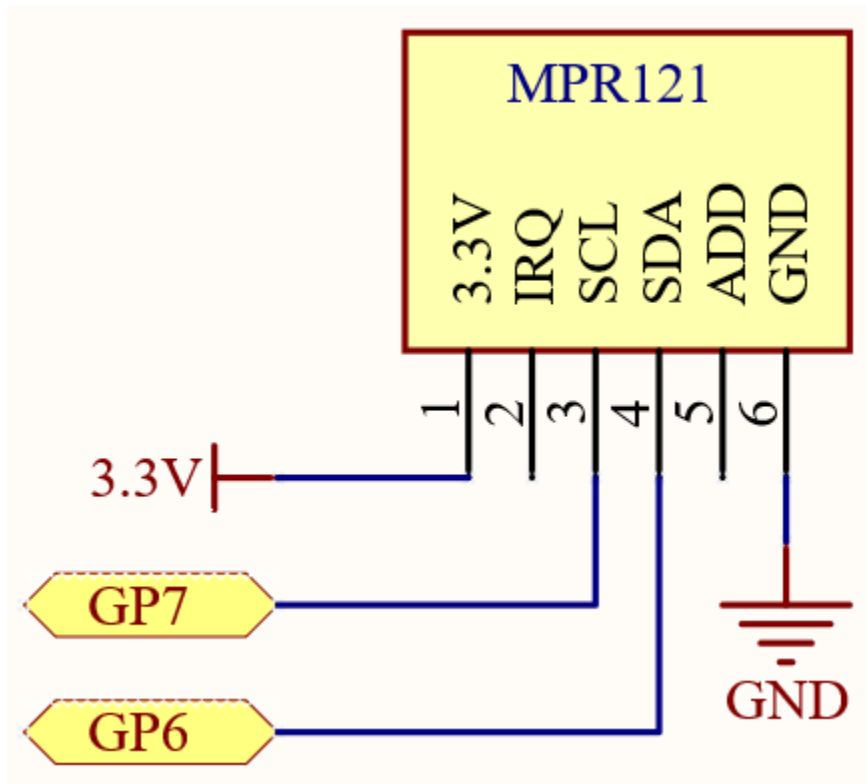
Einen Komplettausatz zu kaufen ist sicherlich praktisch, hier ist der Link dazu:

Name	ARTIKEL IN DIESEM KIT	KAUF-LINK
Kepler-Kit	450+	

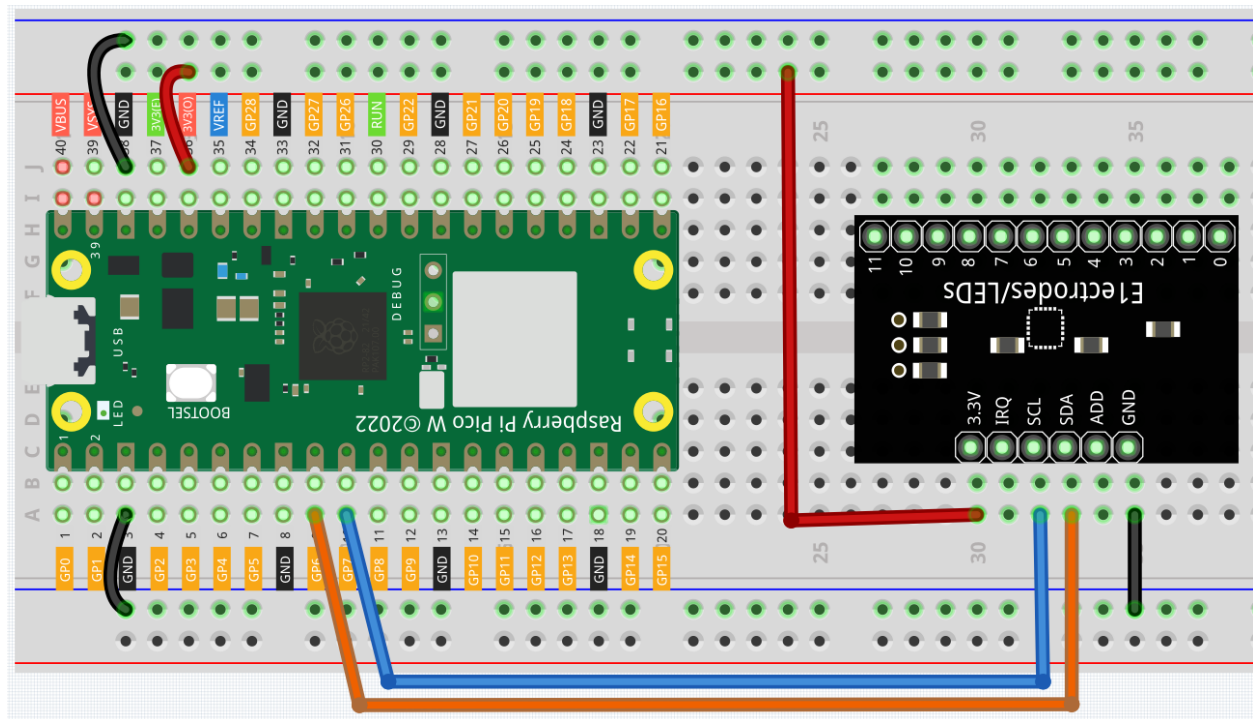
Alternativ können Sie die Teile auch einzeln über die folgenden Links beziehen:

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>MPR121 Modul</i>	1	

### Schaltplan



### Verdrahtung



### Code

### Bemerkung:

- Die Datei `4.3_electrode_keyboard.ino` finden Sie im Verzeichnis `kepler-kit-main/arduino/4.3_electrode_keyboard`.
  - Alternativ können Sie den Code in die **Arduino IDE** kopieren.
  - Vergessen Sie nicht, das richtige Board (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf **Hochladen** klicken.
  - Die Bibliotheken `Adafruit_MPR121` und `Adafruit_BusIO` werden hier verwendet. Bitte lesen Sie [Bibliotheken hinzufügen](#), um zu erfahren, wie sie zur Arduino IDE hinzugefügt werden können.
- 

Sobald das Programm läuft, können Sie die zwölf Elektroden auf dem MPR121-Modul berühren, und der Berührungstatus wird in einem 12-Bit-Booleschen Array gespeichert und im seriellen Monitor angezeigt. Wenn die erste und die elfte Elektrode berührt werden, wird `1000000000010` ausgegeben.

Sie können die Elektroden durch Anschluss anderer Leiter wie Früchte, Draht, Folie usw. erweitern. Dadurch eröffnen sich Ihnen weitere Möglichkeiten, diese Elektroden zu betätigen.

### Wie funktioniert es?

Initialisieren Sie das MPR121-Objekt. Ab diesem Zeitpunkt werden die Zustände der Modul-Elektroden als Ausgangswerte gespeichert. Wenn Sie die Elektroden erweitern, müssen Sie das Beispiel neu starten, um die Ausgangswerte zurückzusetzen.

```
#include "Adafruit_MPR121.h"

Adafruit_MPR121 cap = Adafruit_MPR121();

void setup() {
  Serial.begin(9600);
  int check = cap.begin(0x5A);
  if (!check) {
    Serial.println("MPR121 not found, check wiring?");
    while (1);
  }
  Serial.println("MPR121 found!");
}
```

Erhält den Wert der aktuellen Elektrode, es wird ein 12-Bit-Binärwert erhalten. Wenn Sie die erste und die elfte Elektrode berühren, erhält sie „1000000000010“.

```
// Get the currently touched pads
currouched = cap.touched();
```

Determine if the electrode state has changed.

```
void loop() {
  currouched = cap.touched();
  if (currouched != lasttouched) {}

  // reset our state
  lasttouched = currouched;
}
```

Wenn eine Änderung des Elektrodenzustands erkannt wird, werden die Werte von `currouched` bitweise im Array `touchStates[12]` gespeichert. Schließlich wird das Array ausgegeben.



```

if (curr_touched != last_touched) {
    for (int i = 0; i < 12; i++) {
        if (curr_touched & (1 << i)) touchStates[i] = 1;
        else touchStates[i] = 0;
    }
    for (int i = 0; i < 12; i++){
        Serial.print(touchStates[i]);
    }
    Serial.println();
}

```

## 5. Mikrochip

### 6.31 5.1 Mikrochip - 74HC595

Ein integrierter Schaltkreis (englisch: integrated circuit), kurz IC, ist eine Art Miniatur-Elektronikbauteil.

Mithilfe eines bestimmten Prozesses werden die für einen Schaltkreis benötigten Transistoren, Widerstände, Kondensatoren, Induktoren und weitere Komponenten sowie Verdrahtungen auf einem oder mehreren kleinen Halbleiter-Wafern oder dielektrischen Substraten angeordnet und dann in einem Gehäuse verpackt. Dadurch entsteht eine mikrostrukturierte Einheit mit den erforderlichen Schaltkreisfunktionen; alle Komponenten sind als eine Einheit strukturiert. Dies markiert einen bedeutenden Schritt in Richtung Mikrominiaturisierung, geringem Energieverbrauch, Intelligenz und hoher Zuverlässigkeit der Elektronikkomponenten.

Die Erfinder der integrierten Schaltkreise sind Jack Kilby (integrierte Schaltungen auf Germaniumbasis (Ge)) und Robert Norton Noyce (integrierte Schaltungen auf Siliziumbasis (Si)).

Das Set enthält einen IC, den 74HC595, der den Gebrauch von GPIO-Pins erheblich reduziert. Genauer gesagt können durch das Schreiben einer 8-Bit-Binärzahl 8 Pins für den digitalen Signalausgang ersetzt werden.

- [Binärzahl - Wikipedia](#)
- [74HC595](#)

#### Erforderliche Komponenten

Für dieses Projekt werden die folgenden Komponenten benötigt.

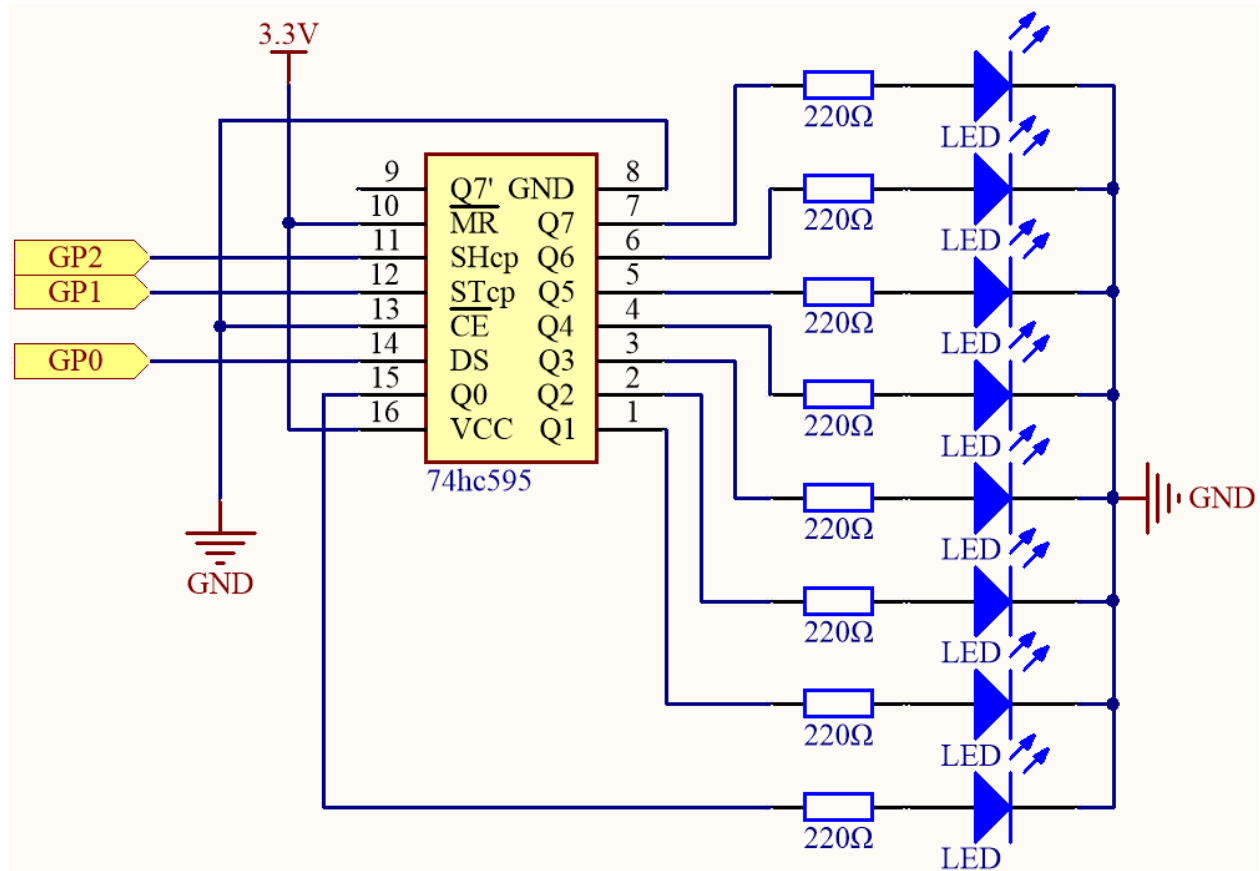
Ein Komplettsset ist natürlich praktisch, hier der Link:

Bezeichnung	ELEMENTE IN DIESEM SET	KAUF-LINK
Kepler Set	450+	

Sie können die Einzelteile auch über die unten stehenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	8(220)	
6	<i>LED</i>	8	
7	<i>74HC595</i>	1	

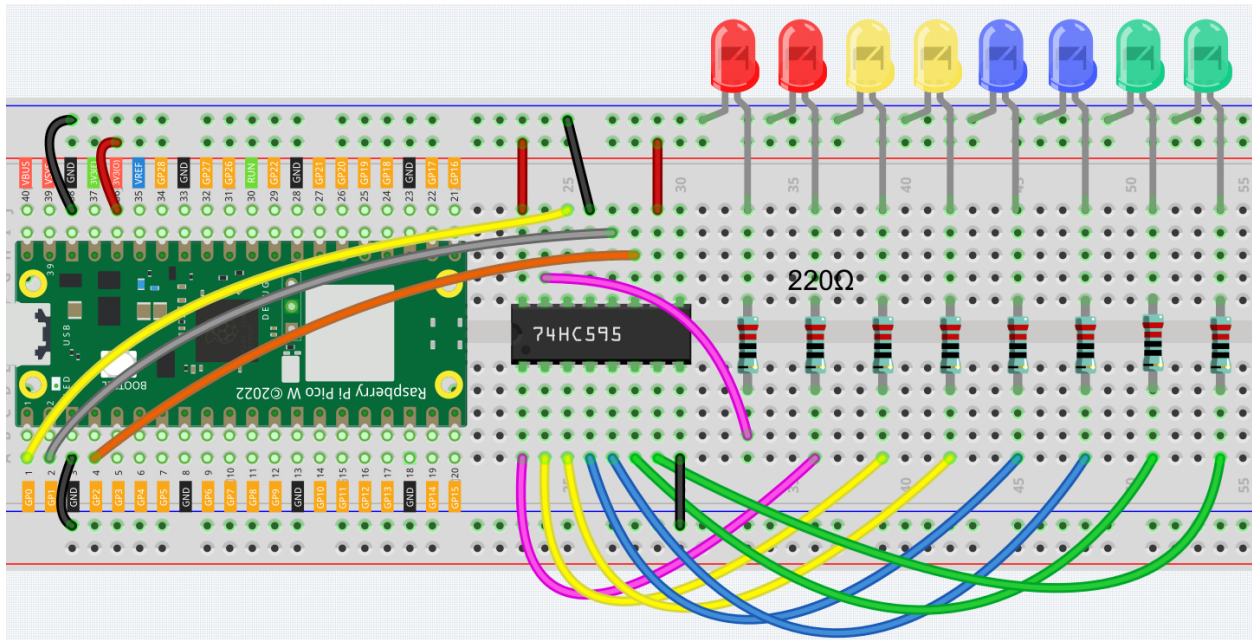
### Schaltplan



- Wenn MR (Pin 10) auf hohem Pegel und OE (Pin 13) auf niedrigem Pegel ist, wird die Daten beim ansteigenden Flanken von SHcp eingelesen und über die ansteigende Flanke von SHcp ins Speicherregister übertragen.
- Sind die beiden Taktgeber miteinander verbunden, ist das Schieberegister immer einen Impuls vor dem Speicherregister.
- Im Speicherregister gibt es einen seriellen Schieberegisterpin (Ds), einen seriellen Ausgangspin (Q) und eine asynchrone Reset-Taste (niedriges Pegel).

- Das Speicherregister gibt einen parallelen 8-Bit-Bus in drei Zuständen aus.
- Ist OE aktiviert (niedriges Pegel), werden die Daten im Speicherregister auf den Bus (Q0 ~ Q7) ausgegeben.

### Verdrahtung



### Code

#### Bemerkung:

- Die Datei 5.1\_microchip\_74hc595.ino finden Sie im Verzeichnis kepler-kit-main/arduino/5.1\_microchip\_74hc595.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den korrekten Port auszuwählen, bevor Sie auf die Schaltfläche **Hochladen** klicken.

Während das Programm läuft, können Sie sehen, wie die LEDs nacheinander aufleuchten.

#### Wie funktioniert das?

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

```
int dataArray[] = {0b00000000, 0b00000001, 0b00000011, 0b00000111, 0b00001111, 0b00011111,
↪ 0b00111111, 0b01111111, 0b11111111};
```

Setzen Sie zuerst STcp auf niedriges Pegel und dann auf hohes Pegel. Dadurch wird ein ansteigender Impuls von STcp erzeugt.

```
digitalWrite(STcp, LOW);
```

shiftOut() wird verwendet, um ein Byte Daten bitweise zu verschieben. Das heißt, es verschiebt ein Byte Daten in dataArray[num] zum Schieberegister über den DS-Pin. MSBFIRST bedeutet, dass von den höheren Bits aus bewegt wird.

```
shiftOut(DS, SHcp, MSBFIRST, dataArray[num]);
```

Nach Ausführung von `digitalWrite(STcp, HIGH)`, ist STcp an der ansteigenden Flanke. Zu diesem Zeitpunkt werden die Daten im Schieberegister ins Speicherregister verschoben.

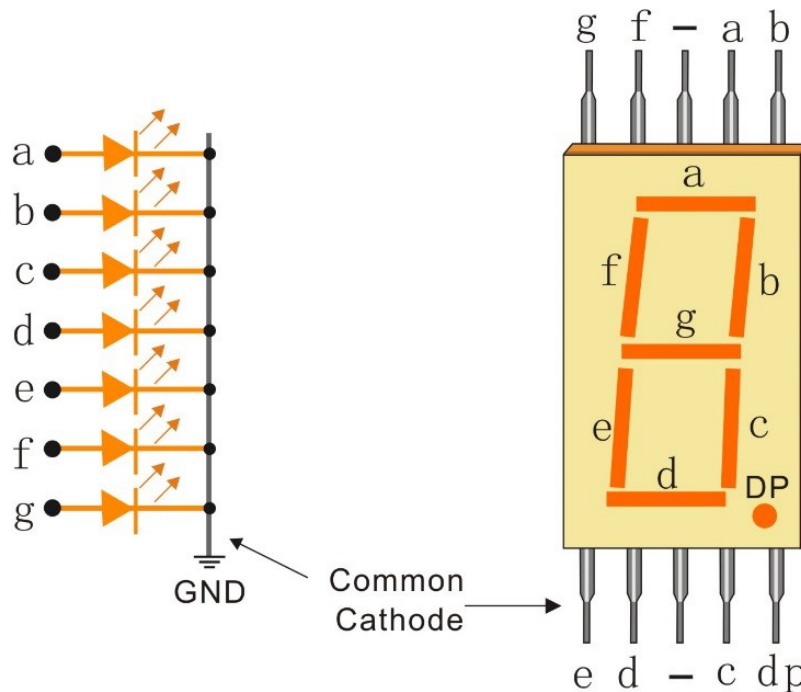
```
digitalWrite(STcp, HIGH);
```

Nach 8 Durchläufen wird ein Byte Daten ins Speicherregister übertragen. Dann werden die Daten des Speicherregisters auf den Bus (Q0-Q7) ausgegeben. Zum Beispiel wird durch `shiftOut B00000001` die von Q0 gesteuerte LED eingeschaltet und die von Q1~Q7 gesteuerten LEDs ausgeschaltet.

## 6.32 5.2 - Zahlenanzeige

LED-Segmentanzeigen sind überall im Alltag zu finden. Zum Beispiel kann sie auf einer Klimaanlage zur Temperaturanzeige verwendet werden oder an einer Verkehrsanzeige, um einen Timer anzuzeigen.

Die LED-Segmentanzeige besteht im Wesentlichen aus einem Gerät, das mit 8 LEDs verpackt ist, von denen 7 streifenförmige LEDs eine „8“-Form bilden und eine etwas kleinere gepunktete LED als Dezimalpunkt dient. Diese LEDs sind als a, b, c, d, e, f, g und dp gekennzeichnet. Sie haben eigene Anodenpins und teilen sich Kathoden. Ihre Pin-Positionen sind im untenstehenden Bild dargestellt.



Das bedeutet, dass sie von 8 digitalen Signalen gleichzeitig gesteuert werden muss, um vollständig zu funktionieren, und der 74HC595 kann dies leisten.

- *7-Segment-Anzeige*

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

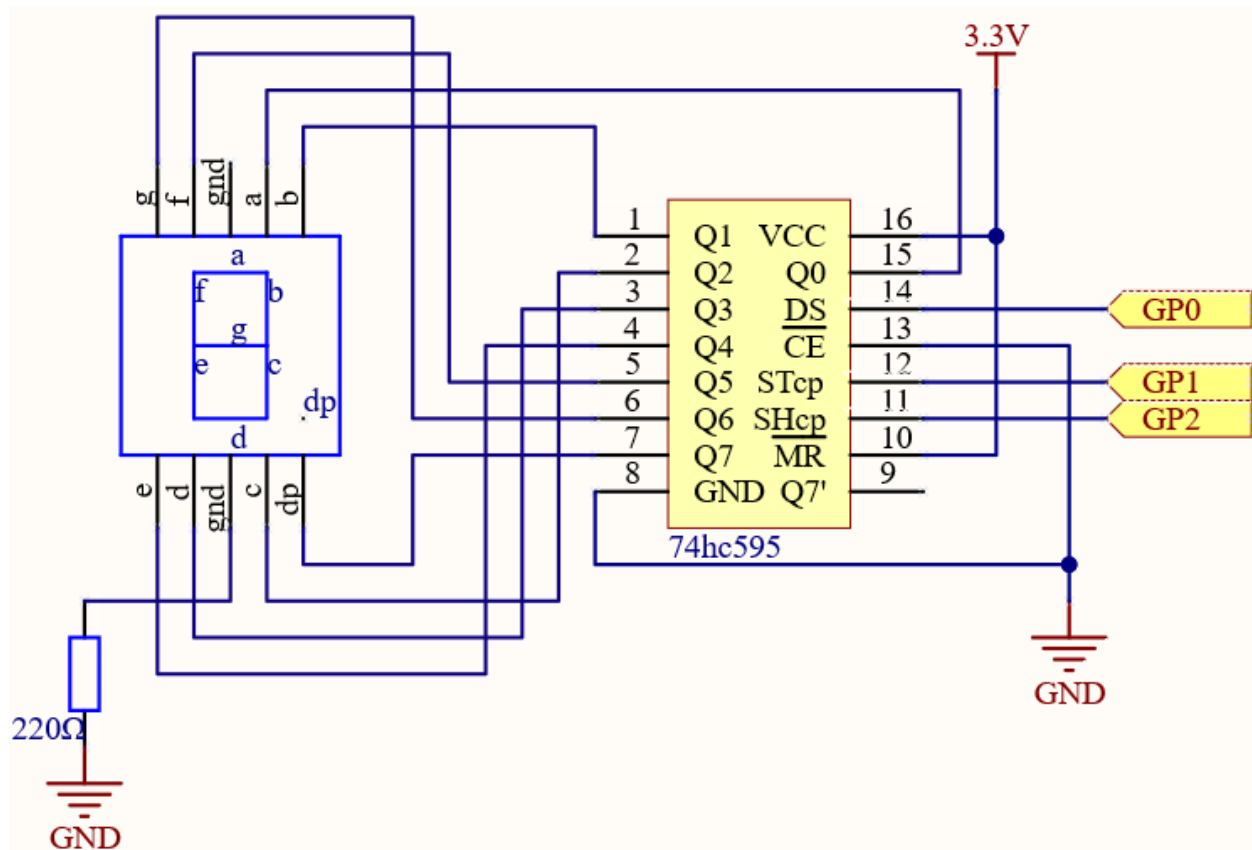
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	KAUF-LINK
Kepler Kit	450+	

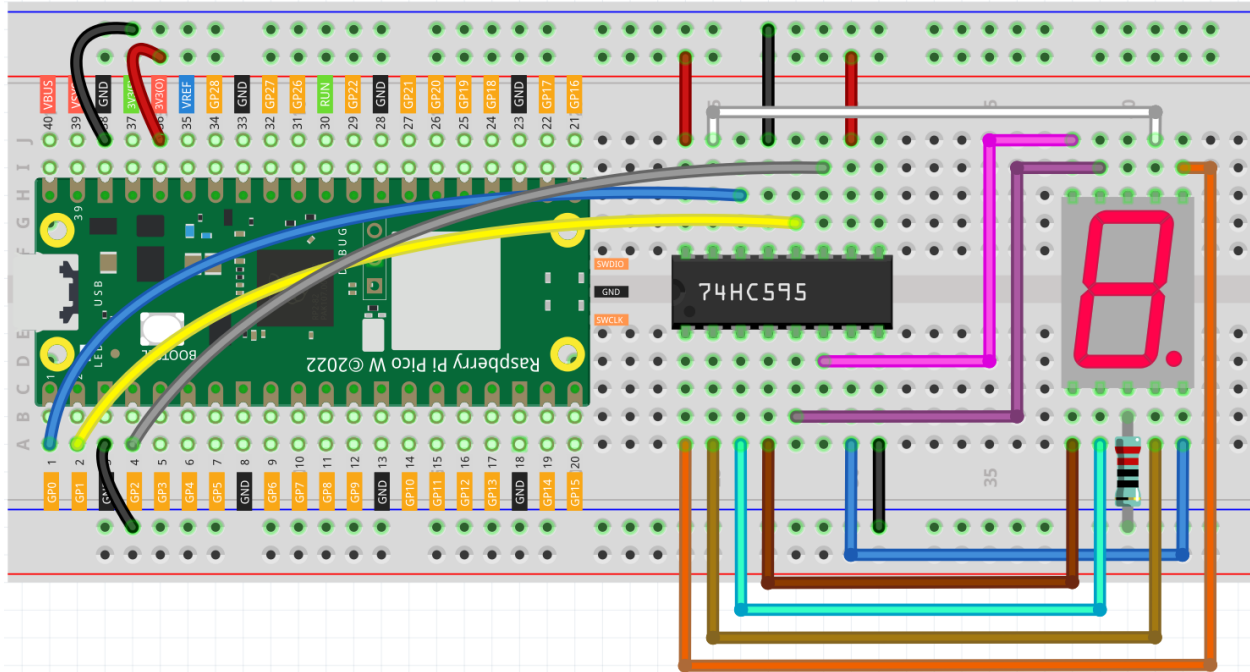
Sie können sie auch einzeln über die untenstehenden Links kaufen.

SN	KOMPONENTENBESCHREIBUNG	MEN-GE	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro USB Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(220)	
6	<i>7-Segment-Anzeige</i>	1	
7	<i>74HC595</i>	1	

## Schaltplan



## Verdrahtung



Tab. 1: Verdrahtung

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

## Code

### Bemerkung:

- Sie können die Datei 5.2\_number\_display.ino im Pfad kepler-kit-main/arduino/5.2\_number\_display öffnen.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf die Schaltfläche **Upload** klicken.

Wenn das Programm läuft, können Sie sehen, dass die LED-Segmentanzeige die Zahlen 0~9 nacheinander anzeigt.

### Wie funktioniert es?

shiftOut() lässt den 74HC595 8 digitale Signale ausgeben. Es gibt das letzte Bit der Binärzahl an Q0 aus und den Ausgang des ersten Bits an Q7. Das bedeutet, dass beim Schreiben der Binärzahl „00000001“ Q0 ein hohes Signal ausgibt und Q1~Q7 ein niedriges Signal.

Nehmen wir an, das 7-Segment-Display zeigt die Zahl „1“ an, wir müssen ein hohes Signal für b, c schreiben und ein niedriges Signal für a, d, e, f, g und dg. Das heißt, die Binärzahl „00000110“ muss geschrieben werden. Aus Gründen

der Lesbarkeit verwenden wir die Hexadezimalnotation als „0x06“.

- [Hexadezimal](#)
- [BinaryHex Konverter](#)

Ebenso können wir das LED-Segmentdisplay auf die gleiche Weise zur Anzeige anderer Zahlen verwenden. Die folgende Tabelle zeigt die entsprechenden Codes für diese Zahlen.

Tab. 2: Glyphen-Code

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

Fügen Sie diese Codes in die Funktion `shiftOut()` ein, um die jeweiligen Zahlen auf dem LED-Segmentdisplay darzustellen.

## 6.33 5.3 - Zeitmesser

Ein 4-stelliges 7-Segment-Display besteht aus vier miteinander verknüpften 7-Segment-Anzeigen.

Das 4-stellige 7-Segment-Display arbeitet eigenständig. Es nutzt das Prinzip der visuellen Persistenz des menschlichen Auges, um die Zeichen jedes 7-Segment-Displays in einer Schleife schnell anzuzeigen und so fortlaufende Zeichenfolgen zu bilden.

Zum Beispiel, wenn „1234“ angezeigt wird, erscheint die „1“ im ersten 7-Segment-Display, während „234“ nicht angezeigt werden. Nach einer kurzen Zeit zeigt das zweite 7-Segment-Display „2“, während die ersten, dritten und vierten 7-Segmente aus bleiben. Und so weiter, alle vier Ziffern werden nacheinander angezeigt. Dieser Prozess ist sehr kurz (typischerweise 5ms), und durch den optischen Nachleuchteffekt sowie das Prinzip der visuellen Persistenz sehen wir alle vier Zeichen gleichzeitig.

### Benötigte Komponenten

Für dieses Projekt werden die folgenden Komponenten benötigt.

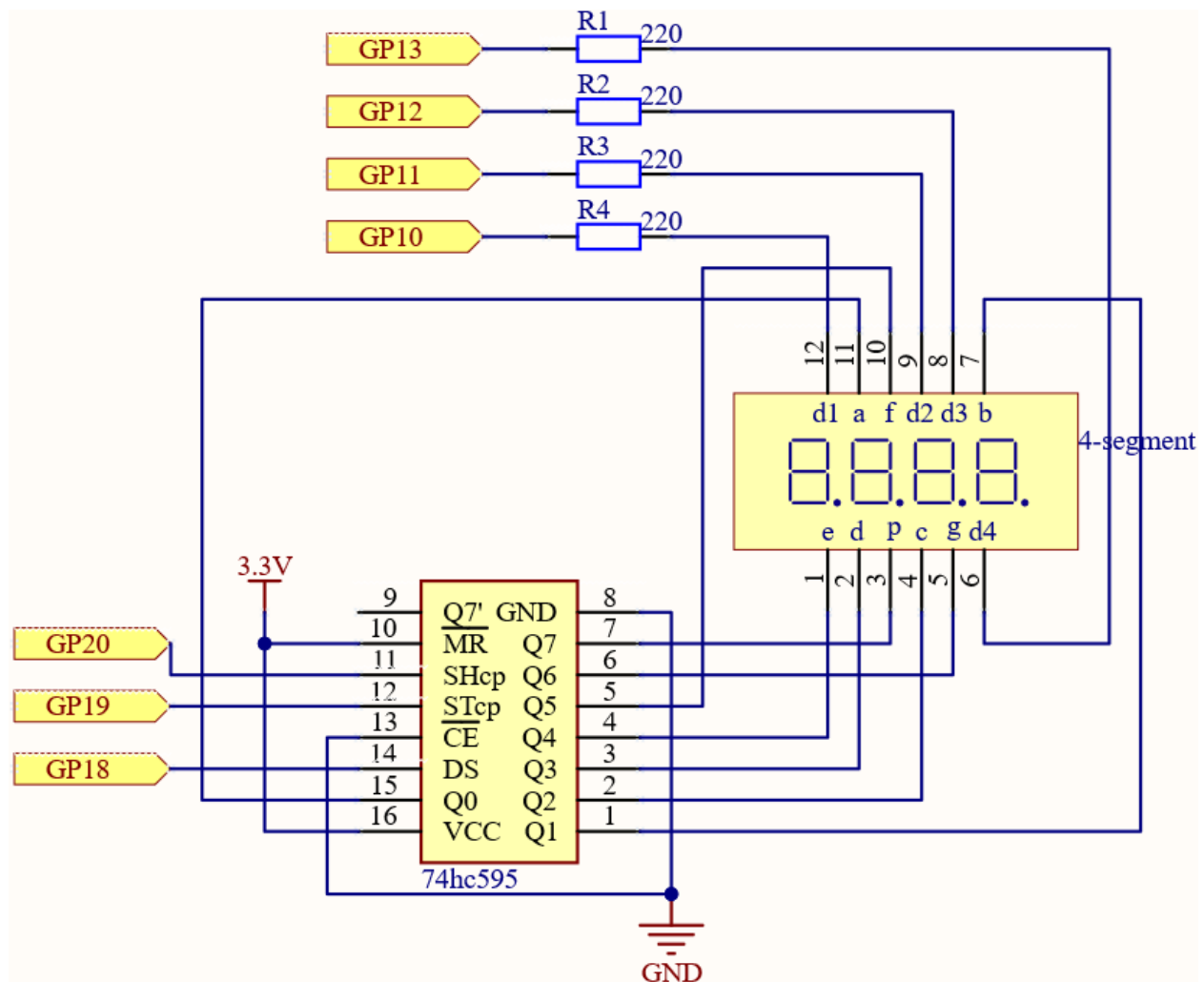
Ein Komplettsset ist definitiv praktisch, hier ist der Link:

Name	ARTIKEL IN DIESEM SET	KAUF-LINK
Kepler-Set	450+	

Alternativ können Sie die Komponenten auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	4(220)	
6	<i>4-stellige 7-Segment-Anzeige</i>	1	
7	<i>74HC595</i>	1	

### Schaltplan

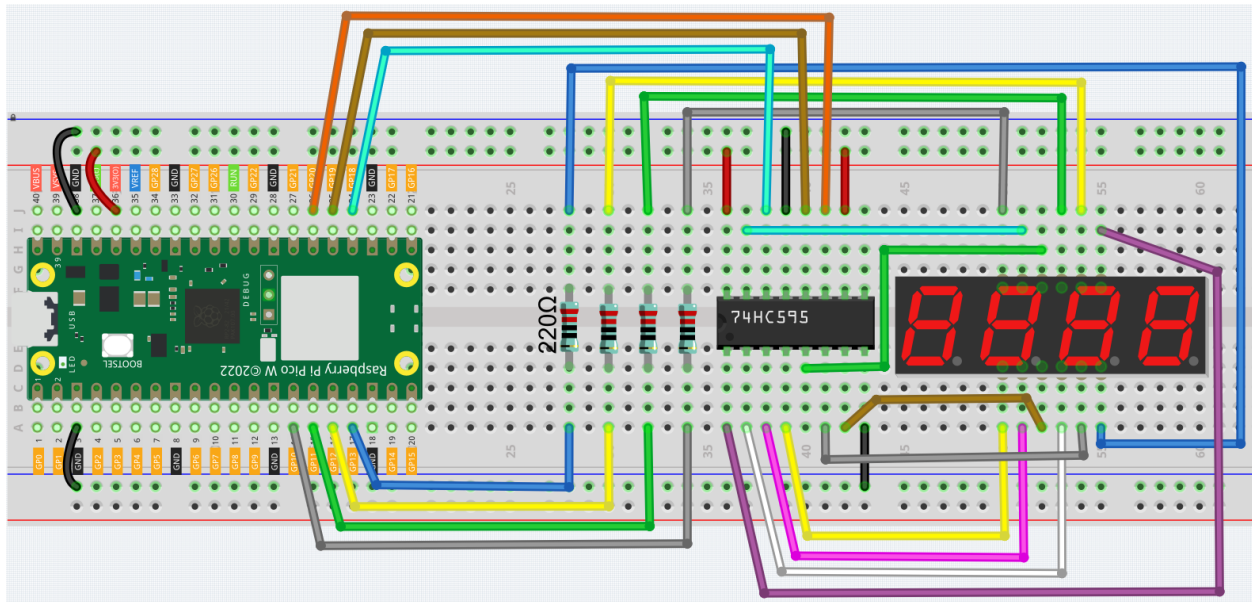


Die Verdrahtungsprinzipien sind im Grunde die gleichen wie bei 5.1 Mikrochip - 74HC595, der einzige Unterschied besteht darin, dass Q0-Q7 an die a ~ g Pins des 4-stelligen 7-Segment-Displays angeschlossen sind.

Dann werden G10 ~ G13 verwendet, um auszuwählen, welches 7-Segment-Display aktiv sein soll.



## Verdrahtung



## Code

### Bemerkung:

- Die Datei `5.3_time_counter.ino` finden Sie im Verzeichnis `kepler-kit-main/arduino/5.3_time_counter`.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den korrekten Port vor dem Klicken auf die **Hochladen**-Taste auszuwählen.

Nach dem Ausführen des Programms wird das 4-stellige 7-Segment-Display zu einem Zähler, und die Zahl erhöht sich jede Sekunde um 1.

### Wie funktioniert es?

Das Senden von Signalen an jedes 7-Segment-Display erfolgt auf die gleiche Weise wie bei [5.2 - Zahlenanzeige](#), mit der Funktion `hc595_shift()`. Der Kernpunkt beim 4-stelligen 7-Segment-Display ist die selektive Aktivierung jedes 7-Segment-Displays. Der damit verbundene Code ist wie folgt.

```
const int placePin[4] = {13,12,11,10};

void setup ()
{
    for (int i = 0; i<4;i++){
        pinMode(placePin[i],OUTPUT);
    }
}

void loop()
{
    pickDigit(0);
    hc595_shift(count%10/1);
}
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```

    pickDigit(1);
    hc595_shift(count%100/10);

    pickDigit(2);
    hc595_shift(count%1000/100);

    pickDigit(3);
    hc595_shift(count%10000/1000);
}

void pickDigit(int digit){
    for(int i = 0; i < 4; i++){
        digitalWrite(placePin[i],HIGH);
    }
    digitalWrite(placePin[digit],LOW);
}

```

Hier werden vier Pins (GP10, GP11, GP12, GP13) verwendet, um jede Stelle des 4-stelligen 7-Segment-Displays einzeln zu steuern. Wenn der Status dieser Pins LOW ist, ist das entsprechende 7-Segment-Display aktiv; wenn der Status HIGH ist, arbeitet es nicht.

Die Funktion `pickDigit(digit)` wird verwendet, um alle 7-Segment-Displays zu deaktivieren und dann eine bestimmte Ziffer individuell zu aktivieren. Danach wird `hc595_shift()` verwendet, um den entsprechenden 8-Bit-Code für das 7-Segment-Display zu schreiben.

Das 4-stellige 7-Segment-Display muss kontinuierlich nacheinander aktiviert werden, damit alle vier Ziffern sichtbar sind. Das bedeutet, dass man im Hauptprogramm nicht einfach Code hinzufügen kann, der das Timing beeinflusst.

Allerdings ist es notwendig, diesem Beispiel eine Timing-Funktion hinzuzufügen. Wenn wir ein `delay(1000)` einfügen, wird offensichtlich, dass nur ein 7-Segment-Display jeweils aktiv ist und die Illusion entlarvt wird, dass alle vier 7-Segment-Displays gleichzeitig arbeiten.

Eine ausgezeichnete Methode, dies zu erreichen, ist die Verwendung der `millis()`-Funktion.

```

void setup()
{
    timerStart = millis();
}

void loop()
{
    unsigned int count = (millis() - timerStart) / 1000;
}

```

Die `millis()`-Funktion gibt die Anzahl der Millisekunden zurück, die seit dem Start des aktuellen Programms vergangen sind. Der erste Zeitwert wird als `timerStart` gespeichert;

wenn die Zeit erneut abgerufen werden muss, rufen wir die `millis()`-Funktion wieder auf und subtrahieren `timerStart`, um die bisherige Laufzeit des Programms zu ermitteln.

Abschließend wird dieser Zeitwert umgewandelt, um ihn auf dem 4-stelligen 7-Segment-Display darzustellen.

- `millis()`

## 6.34 5.4 - 8x8 Pixelgrafik

Eine ED-Matrix ist ein Display mit geringer Auflösung, das auf einer Dot-Matrix-Technologie basiert. Sie nutzt ein Array aus lichtemittierenden Dioden als Pixel für strukturierte Darstellungen.

Diese Anzeigen sind hell genug, um bei Tageslicht im Freien sichtbar zu sein, und finden sich zum Beispiel an Geschäften, Werbetafeln, Schildern und variablen Anzeigesystemen (wie sie in öffentlichen Verkehrsmitteln verwendet werden).

In diesem Bausatz wird eine 8x8-Dot-Matrix mit 16 Pins verwendet. Die Anoden sind in Reihen und die Kathoden in Spalten (auf Schaltungsebene) verbunden, die gemeinsam diese 64 LEDs steuern.

Um die erste LED zu beleuchten, muss für Row1 ein hoher Pegel und für Col1 ein niedriger Pegel bereitgestellt werden. Für die zweite LED gilt: hoher Pegel für Row1, niedriger Pegel für Col2, und so weiter. Durch die Steuerung des Stromflusses zwischen den einzelnen Reihen und Spalten kann jede LED individuell angesteuert werden, um Zeichen oder Bilder darzustellen.

- [LED-Punktmatrix](#)
- [74HC595](#)

### Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Bauteile.

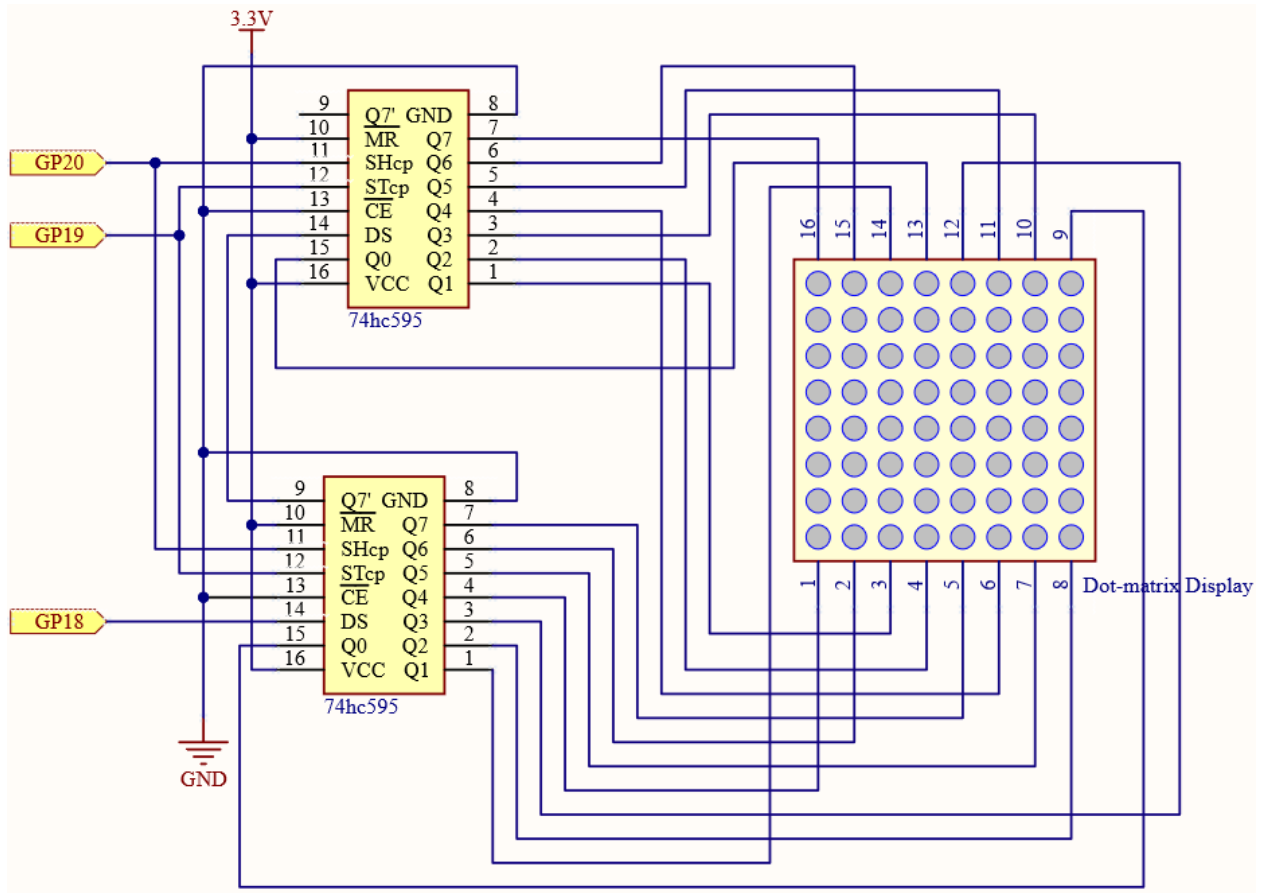
Ein Komplettbausatz ist definitiv praktisch, hier ist der Link:

Bezeichnung	TEILE IM KIT	KAUF-LINK
Kepler-Kit	450+	

Sie können die Komponenten auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	MEN-GE	KAUF-LINK
1	<a href="#">Raspberry Pi Pico W</a>	1	
2	Micro-USB-Kabel	1	
3	<a href="#">Steckbrett</a>	1	
4	<a href="#">Jumperkabel</a>	Mehrere	
5	<a href="#">LED-Punktmatrix</a>	1	
6	<a href="#">74HC595</a>	2	

### Schaltplan



Die 8x8 Dot-Matrix wird von zwei 74HC595-Chips gesteuert, wobei einer die Reihen und der andere die Spalten kontrolliert. Beide Chips teilen sich die Ports G18~G20, was die Anzahl der benötigten I/O-Ports des Pico W Boards deutlich reduziert.

Der Pico W muss jeweils eine 16-Bit-Binärzahl ausgeben, wobei die ersten 8 Bits an den 74HC595 gehen, der die Reihen steuert, und die letzten 8 Bits an den 74HC595, der die Spalten steuert. Auf diese Weise kann die Dot-Matrix ein spezifisches Muster anzeigen.

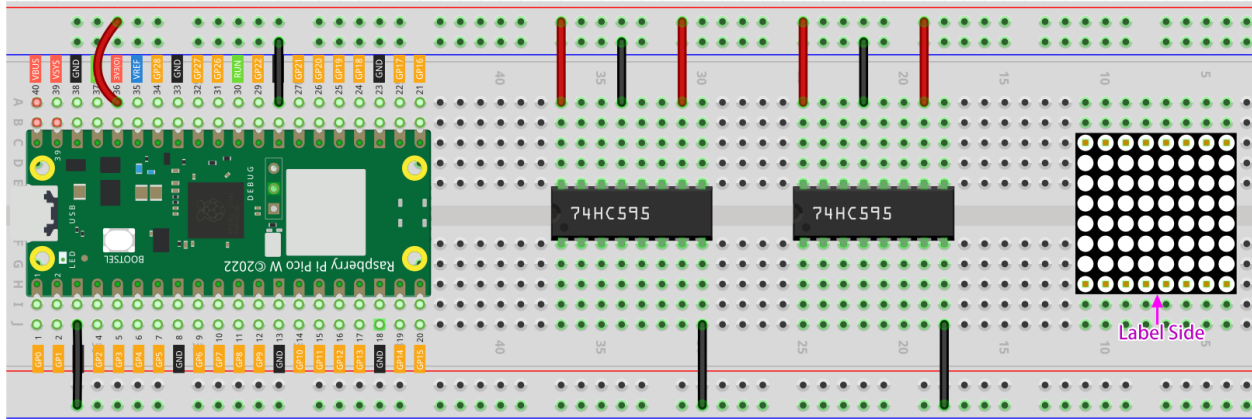
Q7': Serieller Ausgangspin, verbunden mit dem DS eines weiteren 74HC595, um mehrere 74HC595s in Reihe zu schalten.

### Verkabelung

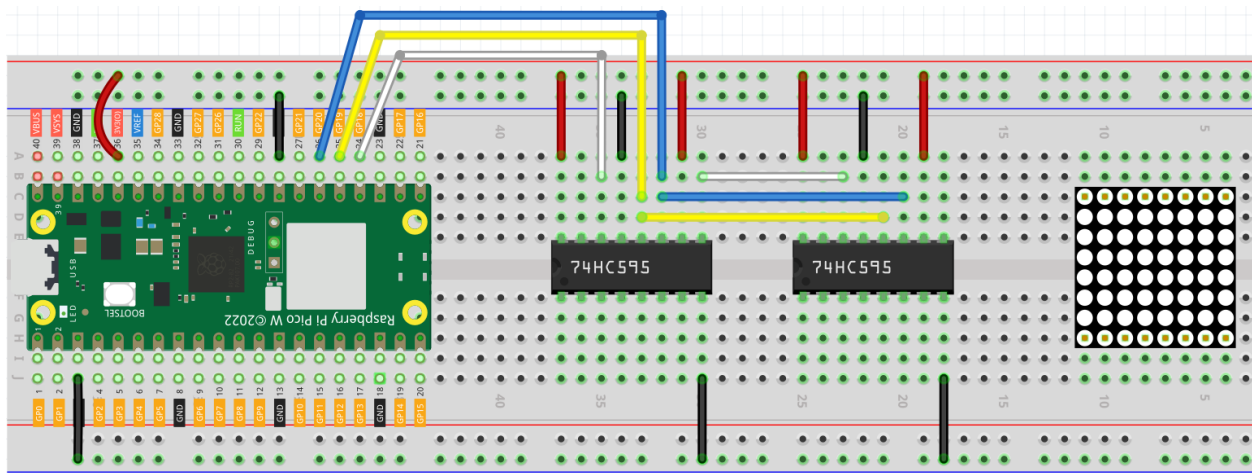
Bauen wir den Schaltkreis auf. Die Verkabelung ist etwas komplex, daher gehen wir am besten schrittweise vor.

**Schritt 1:** Zunächst setzen Sie den Pico W, die LED-Punktmatrix und zwei 74HC595-Chips ins Steckbrett ein. Schließen Sie die 3,3V und GND-Anschlüsse des Pico W an die äußeren Steckbuchsen der Platine an. Verbinden Sie dann Pin 16 und 10 der beiden 74HC595-Chips mit VCC und Pin 13 und Pin 8 mit GND.

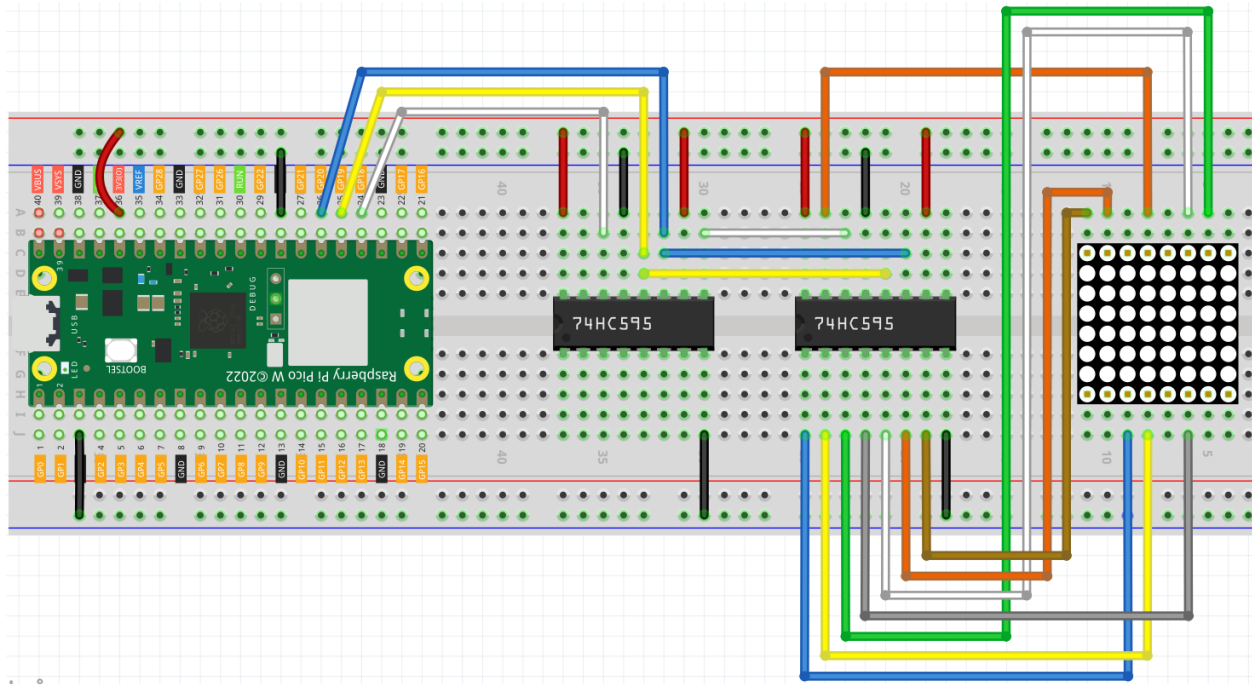
**Bemerkung:** In der oberen Fritzing-Grafik ist die Seite mit dem Label unten abgebildet.



**Schritt 2:** Verknüpfen Sie den Pin 11 beider 74HC595-Chips miteinander und dann mit GP20. Verfahren Sie genauso mit Pin 12 und GP19. Anschließend verbinden Sie Pin 14 des linken 74HC595 mit GP18 und Pin 9 mit Pin 14 des rechten 74HC595.

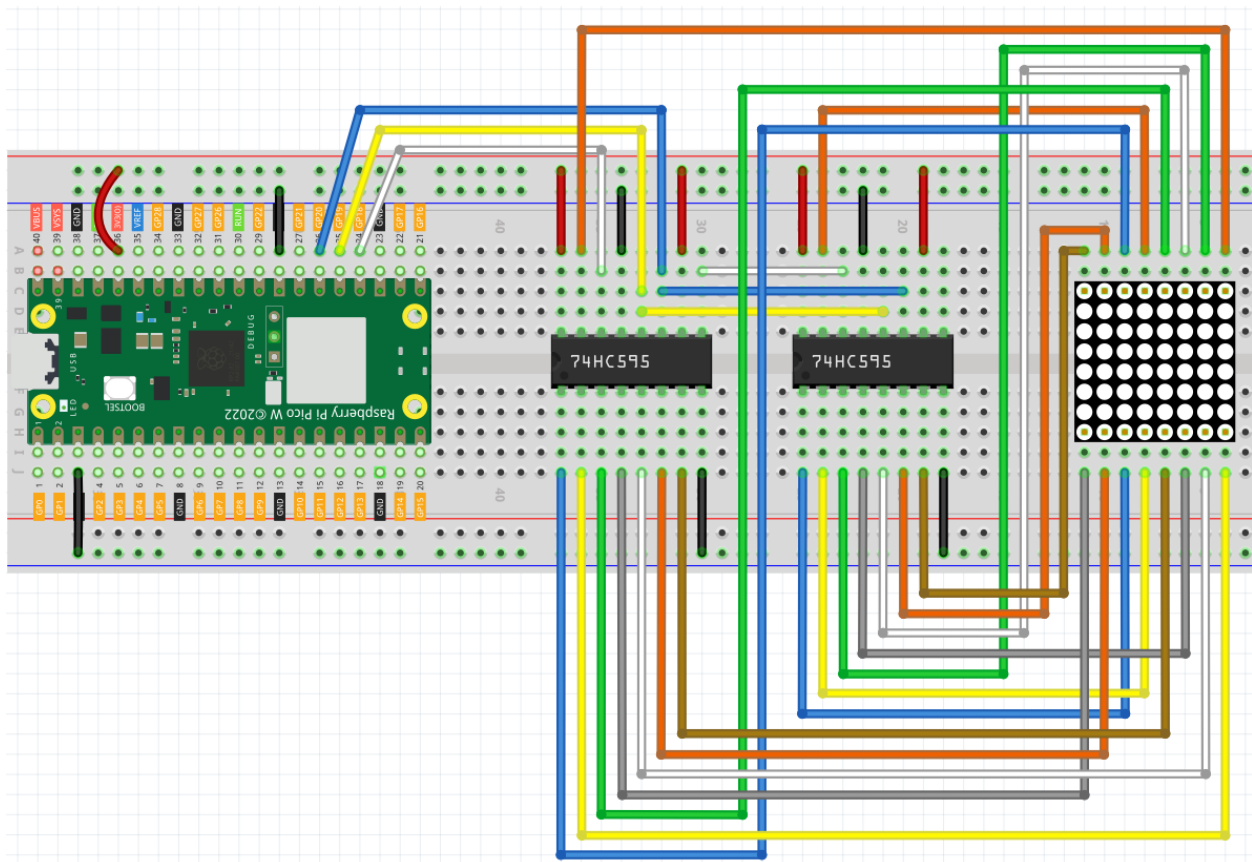


**Schritt 3:** Der rechte 74HC595 ist für die Steuerung der Spalten der LED-Punktmatrix zuständig. Die Zuordnung finden Sie in der untenstehenden Tabelle. Somit korrespondieren die Pins Q0-Q7 des 74HC595 mit den Pins 13, 3, 4, 10, 6, 11, 15 und 16 der LED-Matrix.



**Schritt 4:** Jetzt geht es an die Reihen der LED-Punktmatrix. Der linke 74HC595 steuert diese. Auch hier finden Sie die Zuordnung in der untenstehenden Tabelle. Die Pins Q0-Q7 dieses Chips sind mit den Pins 9, 14, 8, 12, 1, 7, 2 und 5 der LED-Matrix verknüpft.

74HC595	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
LED Dot Matrix	9	14	8	12	1	7	2	5



## Code

### Bemerkung:

- Öffnen Sie die Datei `5.4_8x8_pixel_graphics.ino` im Verzeichnis `kepler-kit-main/arduino/5.4_8x8_pixel_graphics`.
- Alternativ können Sie den Code auch in die **Arduino IDE** kopieren.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf die Schaltfläche **Upload** klicken.

Sobald das Programm läuft, wird ein **X**-Symbol auf der 8x8-Punktmatrix dargestellt.

### Wie funktioniert es?

Wir verwenden zwei 74HC595-Chips, um Signale für die Zeilen und Spalten der Punktmatrix bereitzustellen. Die Signalübertragung funktioniert ähnlich wie bei `shiftOut()` in den vorherigen Kapiteln, allerdings schreiben wir hier eine 16-Bit-Binärzahl auf einmal.

Die Hauptfunktion ruft `shiftOut()` zweimal auf, schreibt zwei 8-Bit-Binärzahlen und sendet sie dann an den Bus, sodass ein Muster angezeigt werden kann.

Allerdings verursacht die gemeinsame Polung der LEDs in der Punktmatrix bei gleichzeitiger Steuerung mehrerer Reihen oder Spalten Interferenzen (z.B. wenn (1,1) und (2,2) gleichzeitig leuchten, werden (1,2) und (2,1) unweigerlich ebenfalls leuchten). Daher ist es notwendig, jeweils nur eine Spalte (oder eine Reihe) zu aktivieren, das Ganze 8-mal zu wiederholen und nach dem Prinzip des Nachbildes das menschliche Auge die 8 Muster zusammenführen zu lassen, um ein Gesamtbild aus 8x8 Informationspunkten zu erhalten.



```

for(int num = 0; num <=8; num++)
{
    digitalWrite(STcp,LOW); //ground ST_CP and hold low for as long as you are
    ↪transmitting
    shiftOut(DS,SHcp,MSBFIRST,dataArray[num]);
    shiftOut(DS,SHcp,MSBFIRST,0x80>>num);
    //return the latch pin high to signal chip that it
    //no longer needs to listen for information
    digitalWrite(STcp,HIGH); //pull the ST_CPST_CP to save the data
}

```

In diesem Beispiel nutzt die Hauptfunktion eine verschachtelte for-Schleife. Bei einem Wert von i gleich 1 wird nur die erste Zeile aktiviert (der Chip der Steuerzeile erhält den Wert 0x80), und das Muster der ersten Zeile wird geschrieben. Bei i gleich 2 wird die zweite Zeile aktiviert (der Chip der Steuerzeile erhält den Wert 0x40), und das Muster der zweiten Zeile wird geschrieben. Und so weiter, bis alle 8 Ausgaben vollzogen sind.

Ähnlich wie bei der 4-stelligen 7-Segment-Anzeige muss die Aktualisierungsrate hochgehalten werden, um ein Flackern des menschlichen Auges zu vermeiden. Daher sollten zusätzliche sleep()-Aufrufe in der Hauptfunktion möglichst vermieden werden.

### Mehr erfahren

Ersetzen Sie dataArray durch eines der folgenden Arrays und schauen Sie, welche Muster erscheinen!

```

int dataArray1[] = {0xFF,0xEF,0xC7,0xAB,0xEF,0xEF,0xEF,0xFF};
int dataArray2[] = {0xFF,0xEF,0xEF,0xEF,0xAB,0xC7,0xEF,0xFF};
int dataArray3[] = {0xFF,0xEF,0xDF,0x81,0xDF,0xEF,0xFF,0xFF};
int dataArray4[] = {0xFF,0xF7,0xFB,0x81,0xFB,0xF7,0xFF,0xFF};
int dataArray5[] = {0xFF,0xBB,0xD7,0xEF,0xD7,0xBB,0xFF,0xFF};
int dataArray6[] = {0xFF,0xFF,0xF7,0xEB,0xDF,0xBF,0xFF,0xFF};

```

Oder versuchen Sie, eigene Grafiken zu entwerfen.

## 6. Fortgeschritten

### 6.35 6.1 - Abstandsmessung

Das Ultraschall-Sensormodul arbeitet nach dem Prinzip von Sonar- und Radarsystemen, um den Abstand zu einem Objekt zu ermitteln.

- [Ultraschallmodul](#)

#### Benötigte Bauteile

Für dieses Projekt benötigen wir die folgenden Komponenten.

Ein Komplettsset ist definitiv praktisch, hier der Link dazu:

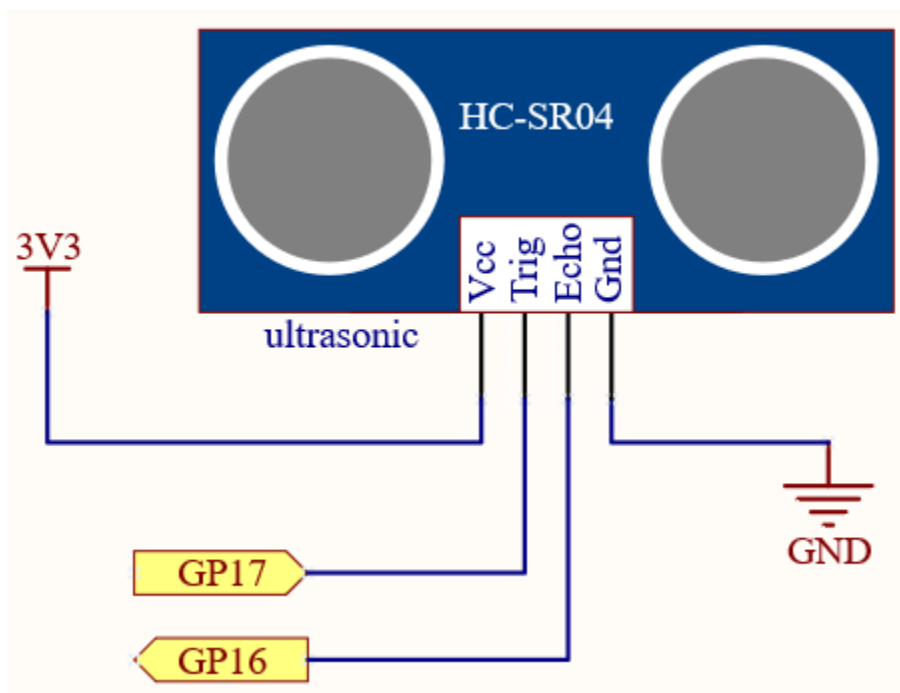
Bezeichnung	INHALT DES KITS	KAUF-LINK
Kepler Kit	450+	

Die Komponenten können auch einzeln über die untenstehenden Links erworben werden.

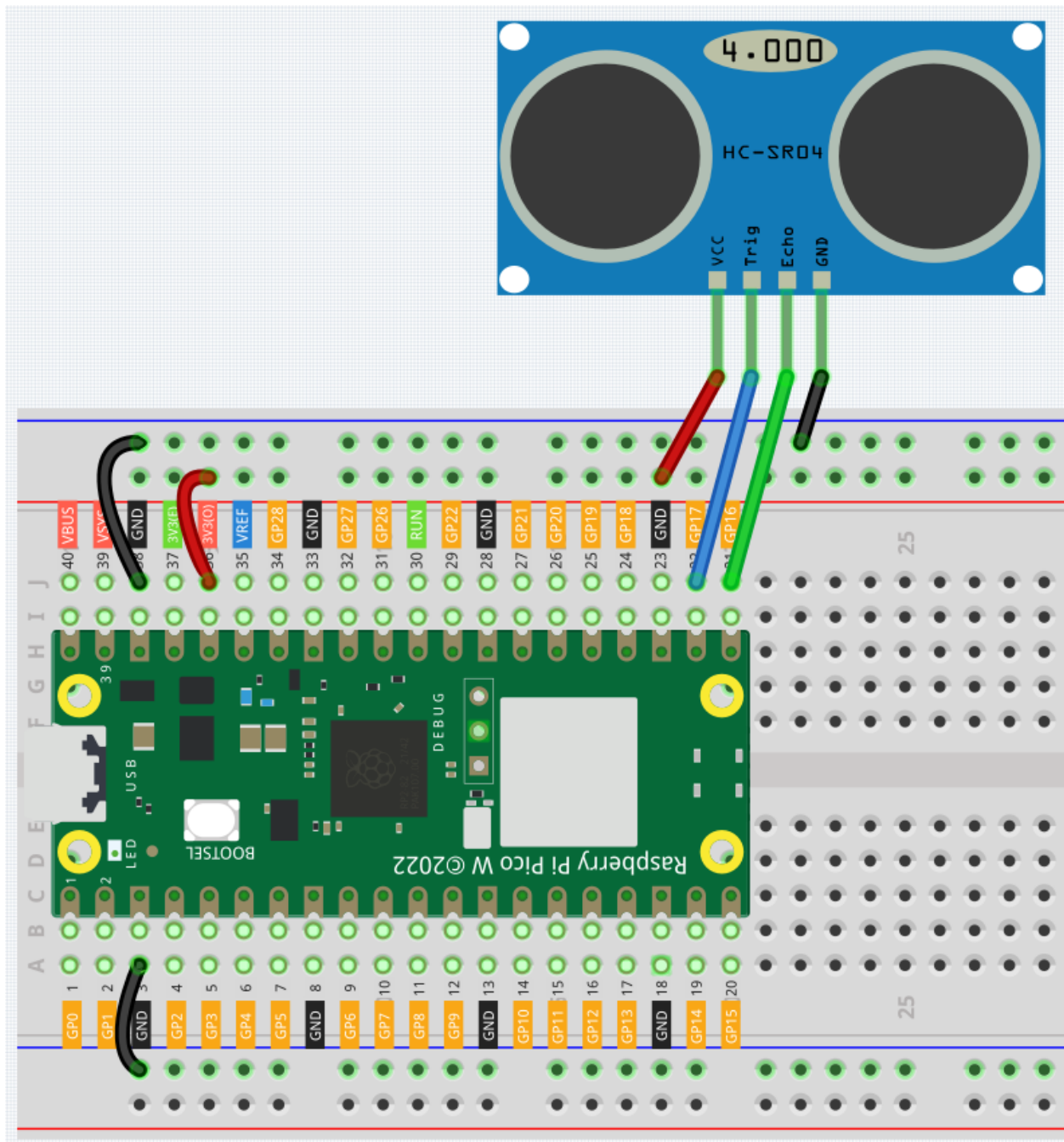


SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Ultraschallmodul</i>	1	

### Schaltplan



### Verkabelung



## Code

### Bemerkung:

- Die Datei `6.1_ultrasonic.ino` finden Sie unter dem Pfad `kepler-kit-main/arduino/6.1_ultrasonic`.
- Alternativ können Sie den Code in die **Arduino IDE** kopieren.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf **Hochladen** klicken.

Sobald das Programm läuft, wird die serielle Monitoranzeige den Abstand des Ultraschallsensors zum vorausliegenden

Hindernis ausgeben.

### Wie funktioniert es?

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

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

Ein PING wird durch einen HIGH-Puls von 2 oder mehr Mikrosekunden ausgelöst. (Geben Sie vorher einen kurzen LOW-Puls aus, um einen sauberen HIGH-Puls zu gewährleisten.)

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

Der Echo-Pin wird verwendet, um das Signal von PING zu lesen, ein HIGH-Puls, dessen Dauer die Zeit (in Mikrosekunden) von der Aussendung des Pings bis zum Empfang des Echos des Objekts ist.

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

Die Schallgeschwindigkeit beträgt 340 m/s oder 29 Mikrosekunden pro Zentimeter.

Dies gibt die vom Ping zurückgelegte Strecke an, hin und zurück, also teilen wir durch 2, um den Abstand des Hindernisses zu erhalten.

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

Beachten Sie, dass der Ultraschallsensor das Programm pausiert, während er arbeitet, was bei komplexen Projekten zu Verzögerungen führen kann.

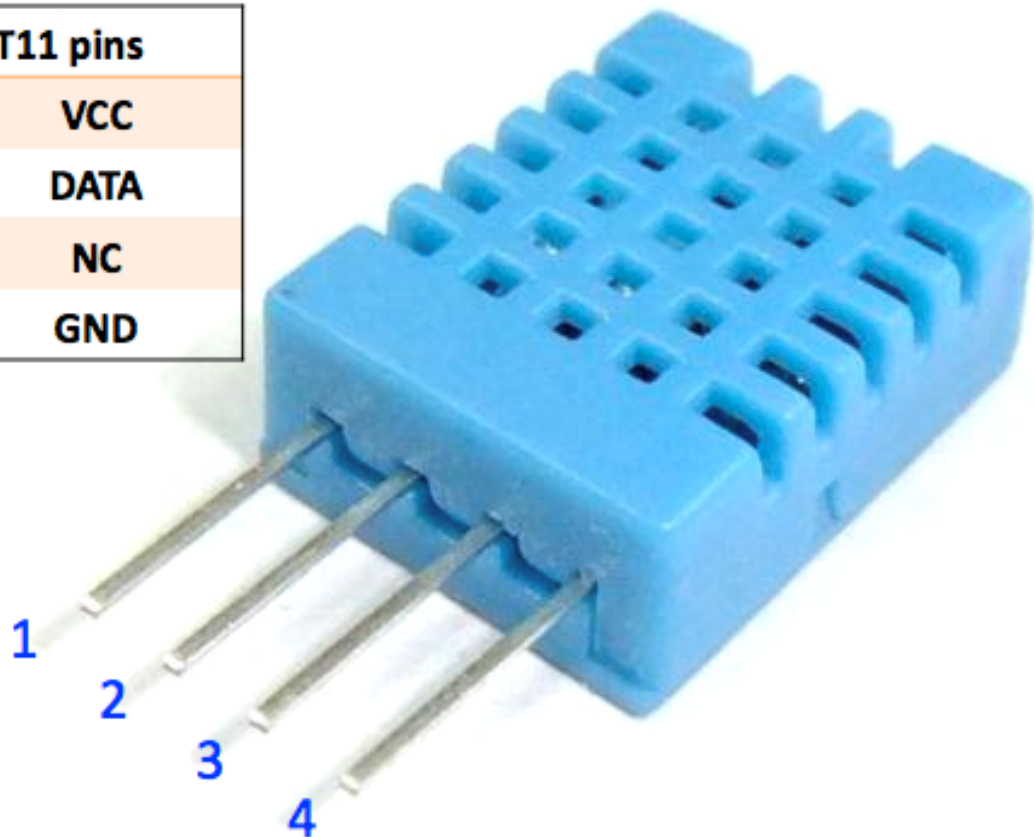
## 6.36 6.2 - Temperatur - Feuchtigkeit

Temperatur und Feuchtigkeit sind sowohl in Bezug auf die physikalische Größe selbst als auch auf das alltägliche Leben der Menschen eng miteinander verbunden. Die Temperatur und Feuchtigkeit der menschlichen Umgebung haben direkten Einfluss auf die thermoregulatorische Funktion und den Wärmeübertragungseffekt des menschlichen Körpers. Dies wirkt sich weiter auf die Denkaktivität und den mentalen Zustand aus und beeinflusst damit die Effizienz unseres Lernens und Arbeitens.

Die Temperatur ist eine der sieben grundlegenden physikalischen Größen im Internationalen Einheitensystem und dient zur Messung des Wärmezustands eines Objekts. Das Grad Celsius ist eine der weltweit am häufigsten verwendeten Temperaturskalen und wird durch das Symbol „°C“ ausgedrückt.

Feuchtigkeit ist die Konzentration von Wasserdampf in der Luft. Die relative Luftfeuchtigkeit wird im Alltag häufig verwendet und in %RH angegeben. Sie steht in engem Zusammenhang mit der Temperatur. Für ein bestimmtes Volumen eingeschlossenen Gases gilt: Je höher die Temperatur, desto niedriger die relative Feuchtigkeit und umgekehrt.

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



In diesem Kit ist ein grundlegender digitaler Temperatur- und Feuchtigkeitssensor, der **DHT11**, enthalten. Er verwendet einen kapazitiven Feuchtigkeitssensor und einen Thermistor, um die umgebende Luft zu messen und gibt ein digitales Signal an den Datenpins aus (keine analogen Eingangspins erforderlich).

- *DHT11 Temperatur- und Feuchtigkeitssensor*

### Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

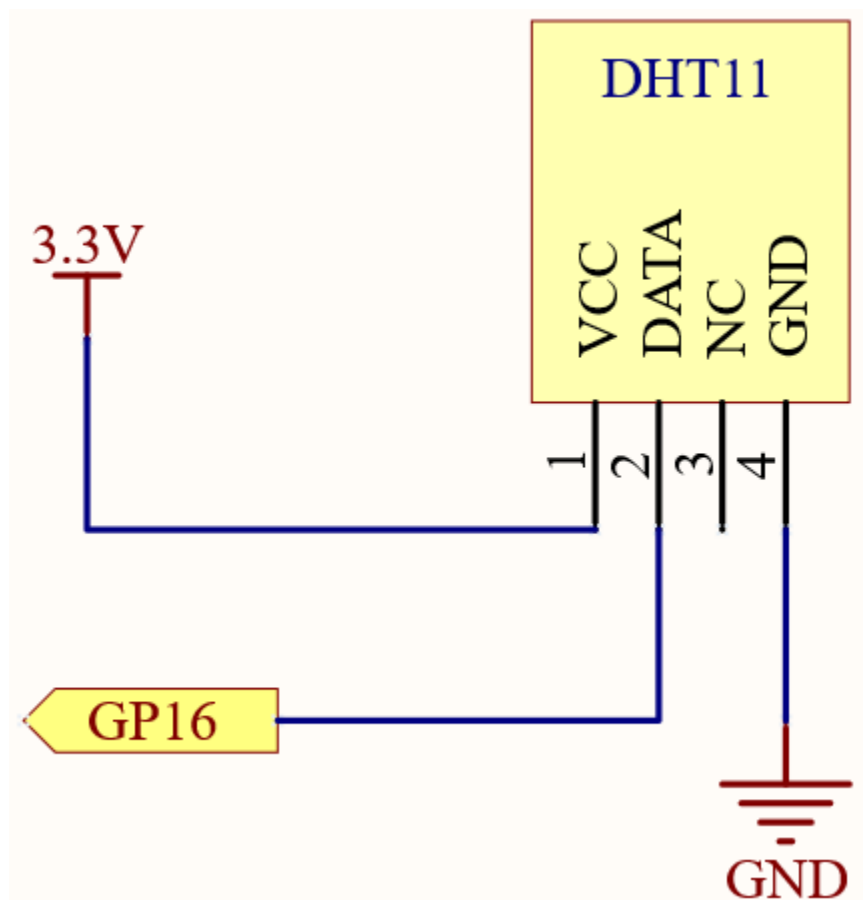
Es ist definitiv praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	KAUF-LINK
Kepler-Kit	450+	

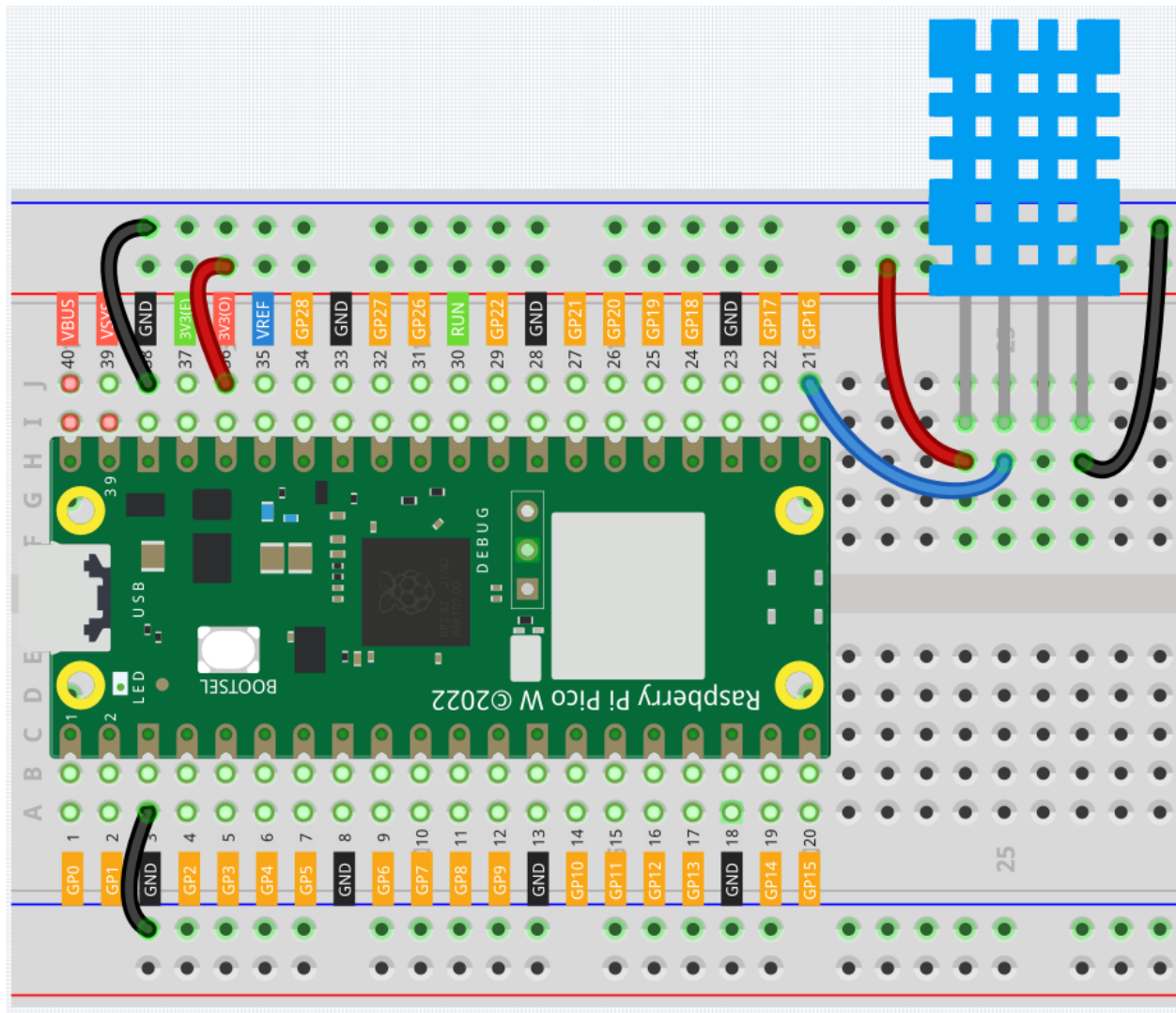
Sie können die einzelnen Komponenten auch über die untenstehenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>DHT11 Temperatur- und Feuchtigkeitssensor</i>	1	

### Schaltplan



### Verkabelung



## Code

**Bemerkung:**

- Die Datei `6.2_dht11.ino` finden Sie im Pfad `kepler-kit-main/arduino/6.2_dht11`.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, die Platine (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf den **Hochladen**-Button klicken.
- Hier wird die Bibliothek `SimpleDHT` verwendet. Wie Sie diese zur Arduino IDE hinzufügen, erfahren Sie unter *[Bibliotheken hinzufügen](#)*.

Nach dem Ausführen des Codes werden Sie sehen, dass der Serieller Monitor kontinuierlich die Temperatur und die Feuchtigkeit ausgibt. Im Laufe der stabilen Programmausführung werden diese beiden Werte immer präziser.

## Wie funktioniert es?

Initialisierung des DHT11-Objekts. Für dieses Gerät ist lediglich ein digitaler Eingang erforderlich.

```
int pinDHT11 = 16;
SimpleDHT11 dht11(pinDHT11);
```

Auslesen der aktuellen Temperatur und Feuchtigkeit, die in den Variablen `temperature` und `humidity` gespeichert werden. `err` dient zur Überprüfung der Gültigkeit der Daten.

```
byte temperature = 0;
byte humidity = 0;
int err = dht11.read(&temperature, &humidity, NULL);
```

Filtern ungültiger Daten.

```
if (err != SimpleDHTErrSuccess) {
    Serial.print("Read DHT11 failed, err=");
    Serial.print(SimpleDHTErrCode(err));
    Serial.print(", ");
    Serial.println(SimpleDHTErrDuration(err));
    delay(1000);
    return;
}
```

Ausgabe der Temperatur und Feuchtigkeit.

```
Serial.print((int)temperature);
Serial.print(" °C, ");
Serial.print((int)humidity);
Serial.println(" %RH");
```

Abschließend ist die Abtastrate des DHT11 1 HZ, daher ist eine `delay(1500)` in der Schleife erforderlich.

```
delay(1500);
```

## 6.37 6.3 - 6-Achsen-Bewegungsverfolgung

Der MPU-6050 ist ein 6-Achsen-Bewegungsverfolgungsgerät, das einen 3-Achsen-Gyroskop mit einem 3-Achsen-Beschleunigungsmesser kombiniert.

Ein Beschleunigungsmesser ist ein Instrument, das die Eigengeschwindigkeitsänderung misst. Ein in Ruhe auf der Erdoberfläche liegender Beschleunigungsmesser würde beispielsweise eine Beschleunigung durch die Erdanziehungskraft von etwa  $g \approx 9,81 \text{ m/s}^2$  in Richtung der Erdoberfläche messen.

Beschleunigungsmesser haben zahlreiche Anwendungen in Industrie und Wissenschaft. Beispiele hierfür sind Trägheitsnavigationssysteme für Flugzeuge und Raketen, Systeme zur Ausrichtung von Bildern auf Tablets und Digitalkameras und so weiter.

Gyroskope werden verwendet, um die Ausrichtung und Winkelgeschwindigkeit eines Geräts oder Systems zu messen. Einsatzgebiete für Gyroskope sind unter anderem Anti-Roll-Systeme und Airbags in Automobilen, Bewegungserfassungssysteme für Smart-Geräte, Lageregelungssysteme für Drohnen und mehr.

- *MPU6050 Modul*

### Erforderliche Komponenten

Für dieses Projekt benötigen wir folgende Komponenten.

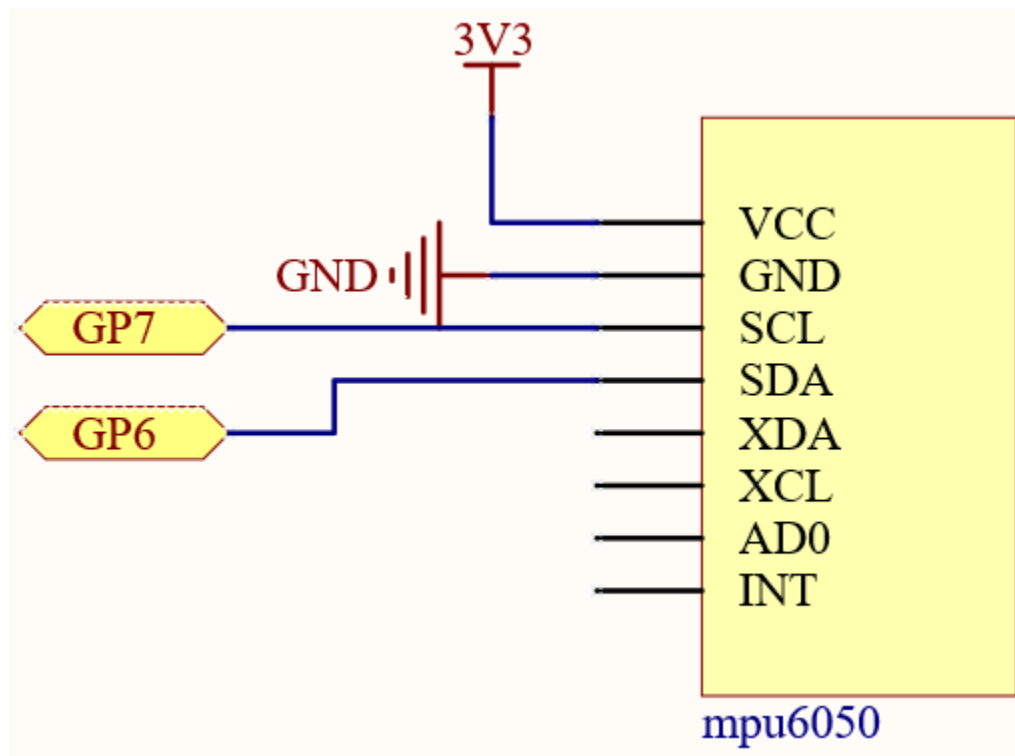
Es ist definitiv praktisch, ein komplettes Kit zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM KIT	KAUF-LINK
Kepler-Kit	450+	

Sie können die Komponenten auch einzeln über die untenstehenden Links erwerben.

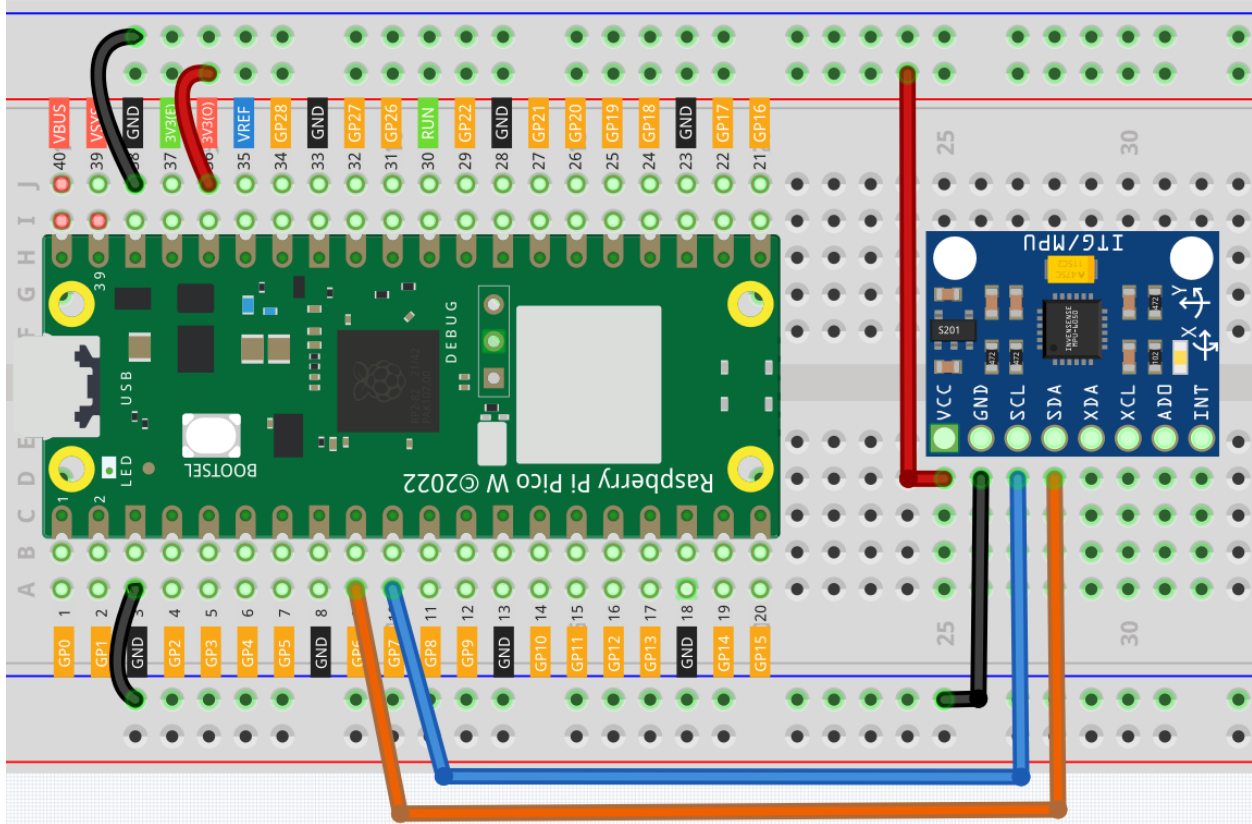
SN	KOMPONENTENBESCHREIBUNG	MEN-GE	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>MPU6050 Modul</i>	1	

### Schaltplan



### Verkabelung





## Code

### Bemerkung:

- Sie können die Datei `6.3_6axis_motion_tracking.ino` im Verzeichnis `kepler-kit-main/arduino/6.3_6axis_motion_tracking` öffnen.
- Oder kopieren Sie diesen Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den entsprechenden Port auszuwählen, bevor Sie auf die **Hochladen**-Taste klicken.
- Die Bibliothek `Adafruit_MPU6050` wird hier verwendet. Bitte beziehen Sie sich auf [Bibliotheken hinzufügen](#), um sie zur Arduino IDE hinzuzufügen.

Nach dem Ausführen des Programms können Sie die Werte des 3-Achsen-Beschleunigungsmessers und des 3-Achsen-Gyroskops in der Ausgabe sehen. Wenn Sie den MPU6050 zufällig drehen, werden diese Werte entsprechend variieren. Um die Änderungen besser verfolgen zu können, können Sie eine der Ausgabezeilen auskommentieren und sich auf einen anderen Datensatz konzentrieren.

### Wie funktioniert es?

Erzeugen Sie ein MPU6050-Objekt.

```
#include <Adafruit_MPU6050.h>
#include <Wire.h>
```

```
Adafruit_MPU6050 mpu;
```

Initialisieren Sie den MPU6050 und konfigurieren Sie seine Genauigkeit.

```

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

  // Set range
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

  Serial.println("");
  delay(100);
}

```

Erfassen Sie neue Sensorevents mit den dazugehörigen Messwerten.

```

sensors_event_t a, g, temp;
mpu.getEvent(&a, &g, &temp);

```

Im Anschluss können Sie Echtzeit-Werte für Beschleunigung und Winkelgeschwindigkeit aus den Daten a.acceleration.x, a.acceleration.y, a.acceleration.z, g.gyro.x, g.gyro.y, g.gyro.z ablesen.

```

Serial.print("Acceleration X: ");
Serial.print(a.acceleration.x);
Serial.print(", Y: ");
Serial.print(a.acceleration.y);
Serial.print(", Z: ");
Serial.print(a.acceleration.z);
Serial.println(" m/s^2");

Serial.print("Rotation X: ");
Serial.print(g.gyro.x);
Serial.print(", Y: ");
Serial.print(g.gyro.y);
Serial.print(", Z: ");
Serial.print(g.gyro.z);
Serial.println(" rad/s");

```

## 6.38 6.4 - IR-Fernbedienung

Im Bereich der Unterhaltungselektronik dienen Fernbedienungen zur Steuerung von Geräten wie Fernsehern und DVD-Playern. In einigen Fällen ermöglichen sie die Bedienung von Geräten, die außer Reichweite sind, etwa Zentral-Klimaanlagen.

Ein IR-Empfänger ist ein Bauteil mit einer Fotozelle, die auf Infrarotlicht abgestimmt ist. Er wird fast immer zur Fernbedienungserkennung eingesetzt - jeder Fernseher und DVD-Player hat einen solchen an der Vorderseite, um das IR-Signal vom Bediengerät zu empfangen. In der Fernbedienung selbst ist eine passende IR-LED, die IR-Impulse sendet, um den Fernseher ein- oder auszuschalten oder den Kanal zu wechseln.

- *Infrarotempfänger*

### Benötigte Komponenten

Für dieses Projekt werden die folgenden Komponenten benötigt.

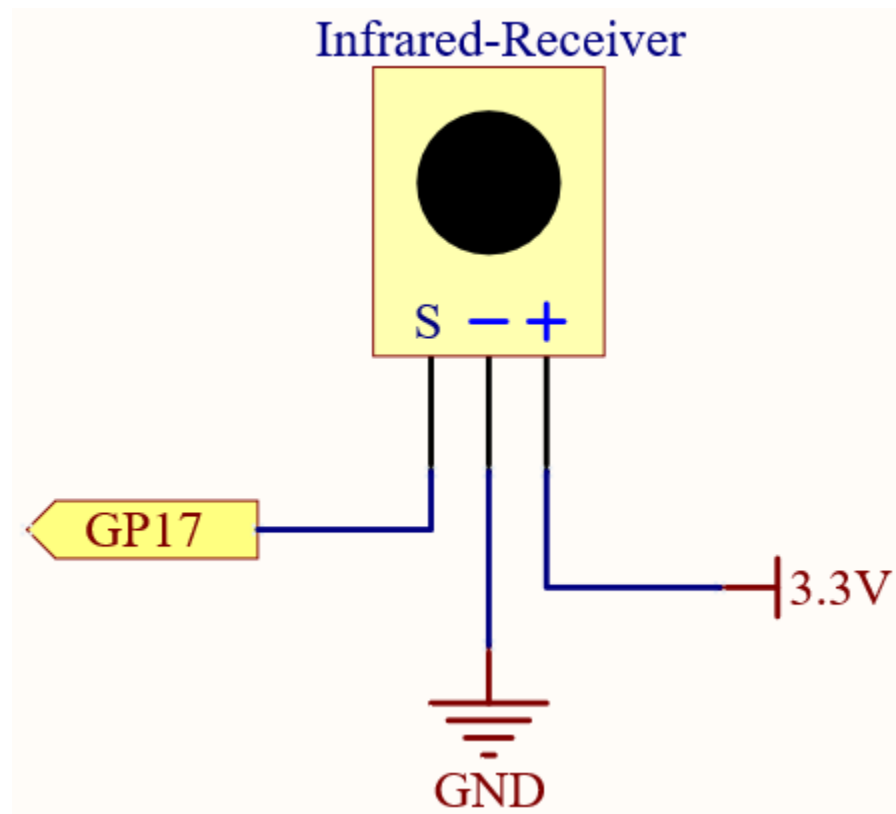
Es ist durchaus praktisch, ein komplettes Set zu kaufen, hier ist der Link:

Name	ARTIKEL IN DIESEM SET	KAUF-LINK
Kepler Kit	450+	

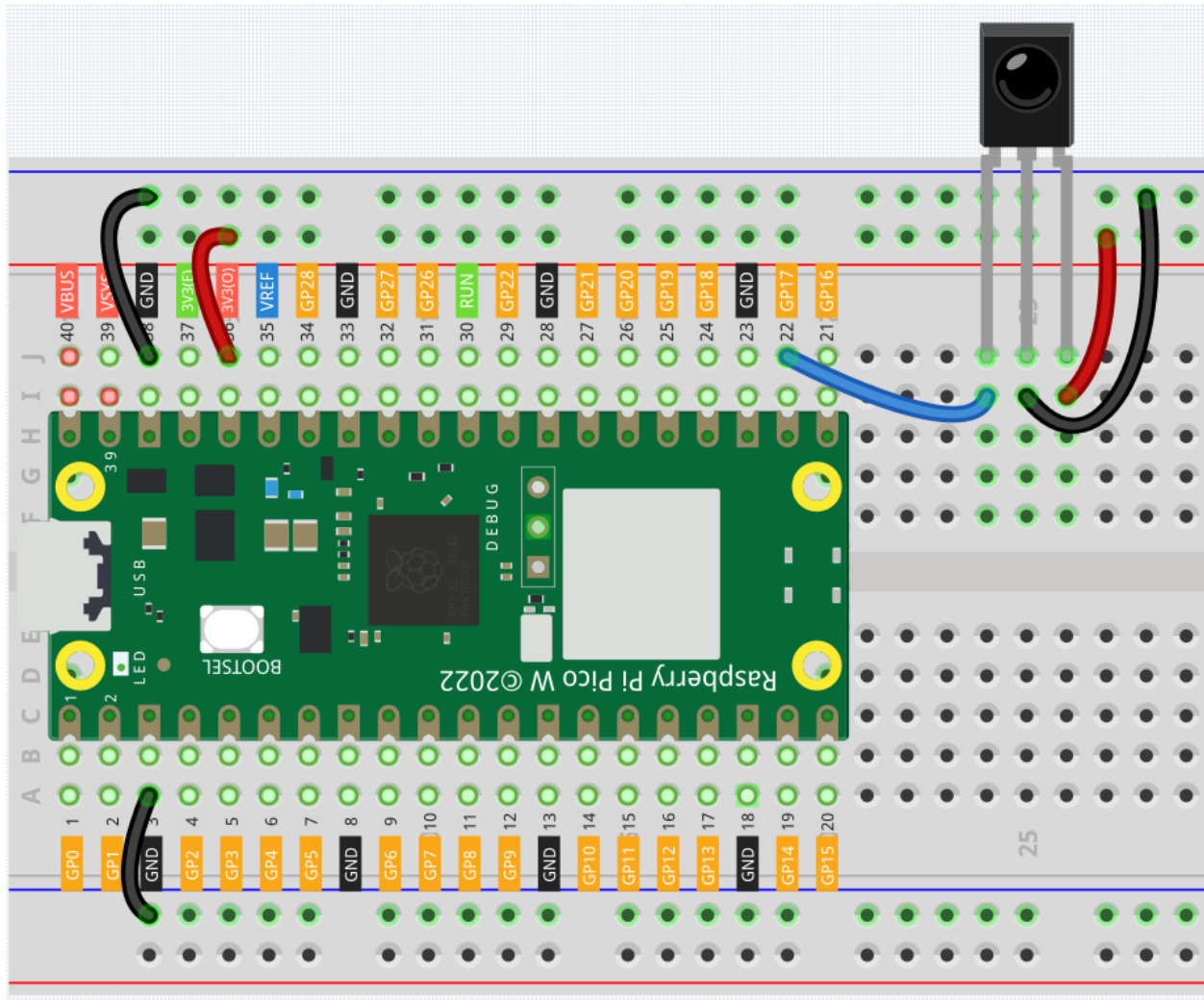
Sie können die Komponenten auch einzeln über die folgenden Links erwerben.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Infrarotempfänger</i>	1	

### Schaltplan



Verdrahtung



## Code

### Bemerkung:

- Die Datei `6.4_ir_remote_control.ino` finden Sie unter dem Pfad `kepler-kit-main/arduino/6.4_ir_remote_control`.
- Oder kopieren Sie den Code in die **Arduino IDE**.
- Vergessen Sie nicht, das Board (Raspberry Pi Pico) und den richtigen Port auszuwählen, bevor Sie auf den **Hochladen**-Button klicken.
- Die Bibliothek `IRsmallDecoder` wird hier verwendet. Bitte beziehen Sie sich auf [Bibliotheken hinzufügen](#), um sie in die Arduino IDE hinzuzufügen.

Die neue Fernbedienung enthält ein Plastikstück am Ende, das die Batterie isoliert. Dieses Plastikstück muss entfernt werden, um die Fernbedienung in Betrieb zu nehmen. Sobald das Programm läuft und Sie die Fernbedienung drücken, wird der Serial Monitor den gedrückten Knopf ausgeben.

## 6.39 6.5 - Funkfrequenz-Identifikation

Funkfrequenz-Identifikation (RFID) bezeichnet Technologien, die drahtlose Kommunikation zwischen einem Objekt (oder Tag) und einem Abfragegerät (oder Lesegerät) nutzen, um solche Objekte automatisch zu verfolgen und zu identifizieren. Die Reichweite der Tag-Übertragung ist auf einige Meter vom Lesegerät begrenzt. Eine direkte Sichtlinie zwischen dem Lesegerät und dem Tag ist nicht zwingend erforderlich.

Die meisten Tags enthalten mindestens einen integrierten Schaltkreis (IC) und eine Antenne. Der Mikrochip speichert Informationen und ist für die Verwaltung der Funkfrequenzkommunikation (RF) mit dem Lesegerät verantwortlich. Passive Tags verfügen nicht über eine unabhängige Energiequelle und sind auf ein externes elektromagnetisches Signal angewiesen, das vom Lesegerät bereitgestellt wird, um ihren Betrieb zu ermöglichen. Aktive Tags verfügen über eine eigenständige Energiequelle, etwa eine Batterie. Dadurch können sie erweiterte Verarbeitungs-, Übertragungsfähigkeiten und Reichweite haben.

- *MFRC522 Modul*

### Erforderliche Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

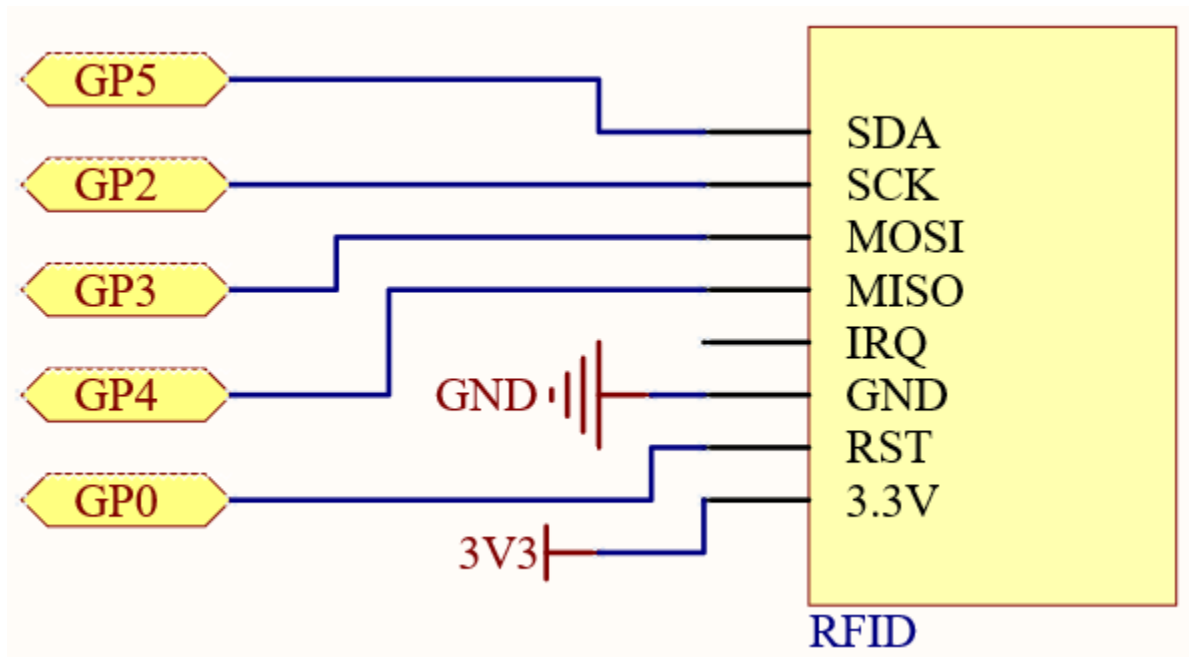
Es ist durchaus praktisch, ein ganzes Kit zu kaufen, hier ist der Link:

Bezeichnung	ARTIKEL IN DIESEM KIT	KAUF-LINK
Kepler-Kit	450+	

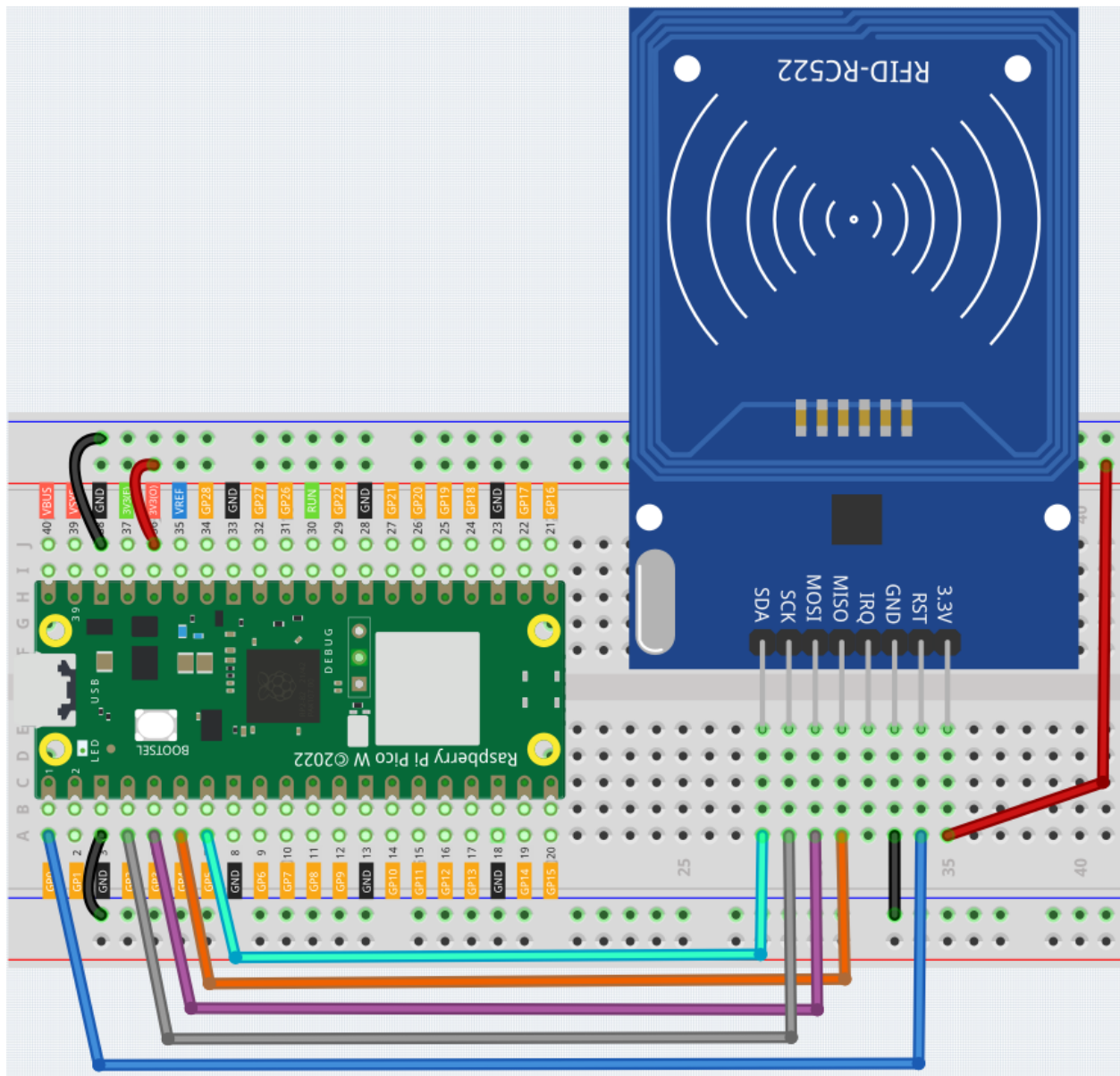
Sie können die einzelnen Komponenten auch separat über die folgenden Links kaufen.

SN	KOMPONENTENBESCHREIBUNG	AN-ZAHL	KAUF-LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>MFRC522 Modul</i>	1	

### Schaltplan



Verdrahtung



## Code

### Bemerkung:

- Die Datei `6.5_rfid_write.ino` finden Sie im Pfad `kepler-kit-main/arduino/6.5_rfid_write`.
- Alternativ können Sie diesen Code in die **Arduino IDE** kopieren.
- Vergessen Sie nicht, vor dem Klicken auf den **Hochladen**-Button das Board (Raspberry Pi Pico) und den korrekten Port auszuwählen.
- Die Bibliothek `MFRC522` wird hier verwendet. Weitere Informationen zum Hinzufügen in die Arduino IDE finden Sie unter [Bibliotheken hinzufügen](#).

Die Hauptfunktion ist in zwei Teile gegliedert:

- `6.5_rfid_write.ino`: Dient zum Schreiben von Informationen auf die Karte (oder den Schlüssel).



- `6.5_rfid_read.ino`: Dient zum Lesen der Informationen auf der Karte (oder dem Schlüssel).

**Bemerkung:**

- Die Datei `6.5_rfid_write.ino` finden Sie im Pfad `kepler-kit-main/arduino/6.5_rfid_write`.
- Alternativ können Sie diesen Code in die **Arduino IDE** kopieren.
- Vergessen Sie nicht, vor dem Klicken auf den **Hochladen**-Button das Board (Raspberry Pi Pico) und den korrekten Port auszuwählen.

Nach dem Ausführen können Sie eine Nachricht im seriellen Monitor eingeben, die mit # endet. Anschließend schreiben Sie die Nachricht auf die Karte, indem Sie die Karte (oder den Schlüssel) nahe am MFRC522-Modul platzieren.

**Bemerkung:**

- Die Datei `6.5_rfid_read.ino` finden Sie im Pfad `kepler-kit-main/arduino/6.5_rfid_read`.
- Alternativ können Sie diesen Code in die **Arduino IDE** kopieren.
- Vergessen Sie nicht, vor dem Klicken auf den **Hochladen**-Button das Board (Raspberry Pi Pico) und den korrekten Port auszuwählen.

Nach dem Ausführen können Sie die auf der Karte (oder dem Schlüssel) gespeicherte Nachricht lesen.

**Funktionsweise?**

```
#include <MFRC522.h>

#define RST_PIN      0
#define SS_PIN       5

MFRC522 mfrc522(SS_PIN, RST_PIN);
```

Zunächst wird die Klasse `MFRC522()` instanziiert.

Für eine einfachere Handhabung wird die MFRC522-Bibliothek durch die folgenden Funktionen weiter abstrahiert.

- `void simple_mfrc522_init()`: Startet die SPI-Kommunikation und initialisiert das MFRC522-Modul.
- `void simple_mfrc522_get_card()`: Hält das Programm an, bis die Karte (oder der Schlüssel) erkannt wird, und gibt die UID der Karte sowie den PICC-Typ aus.
- `void simple_mfrc522_write(String text)`: Schreibt einen Text auf die Karte (oder den Schlüssel).
- `void simple_mfrc522_write(byte* buffer)`: Schreibt Informationen auf die Karte (oder den Schlüssel), die üblicherweise vom seriellen Port stammen.
- `void simple_mfrc522_write(byte section, String text)`: Schreibt einen Text in einen bestimmten Sektor. Bei `section` auf 0 werden die Sektoren 1-2 beschrieben; bei `section` auf 1 die Sektoren 3-4.
- `void simple_mfrc522_write(byte section, byte* buffer)`: Schreibt Informationen in einen bestimmten Sektor, die üblicherweise vom seriellen Port stammen. Bei `section` auf 0 werden die Sektoren 1-2 beschrieben; bei `section` auf 1 die Sektoren 3-4.
- `String simple_mfrc522_read()`: Liest die Informationen auf der Karte (oder dem Schlüssel) und gibt einen String zurück.

- `String simple_mfrc522_read(byte section)`: Liest die Informationen in einem bestimmten Sektor und gibt einen String zurück. Bei `section` auf 0 werden die Sektoren 1-2 beschrieben; bei `section` auf 1 die Sektoren 3-4.

Im Beispiel `6.5_rfid_write.ino` wird die Funktion `Serial.readBytesUntil()` verwendet, eine gängige Methode für serielle Eingaben.

- `Serial.readBytesUntil`

Dieses Kapitel enthält eine Einführung in Piper Make, Anleitungen zur Verbindung und Programmierung von Pico W mit Piper Make sowie mehrere interessante Projekte, die Ihnen einen schnellen Einstieg in Piper ermöglichen.

Wir empfehlen, dieses Kapitel in der angegebenen Reihenfolge zu lesen.

Piper Make ist eine kinderleichte und unterhaltsame Möglichkeit, Projekte mit dem Raspberry Pi Pico W zu realisieren. Es verwendet Blöcke ähnlich wie Scratch, sodass keinerlei Programmierkenntnisse erforderlich sind. Die zugrunde liegende Arbeitsweise basiert auf CircuitPython mit ergänzenden Bibliotheken.

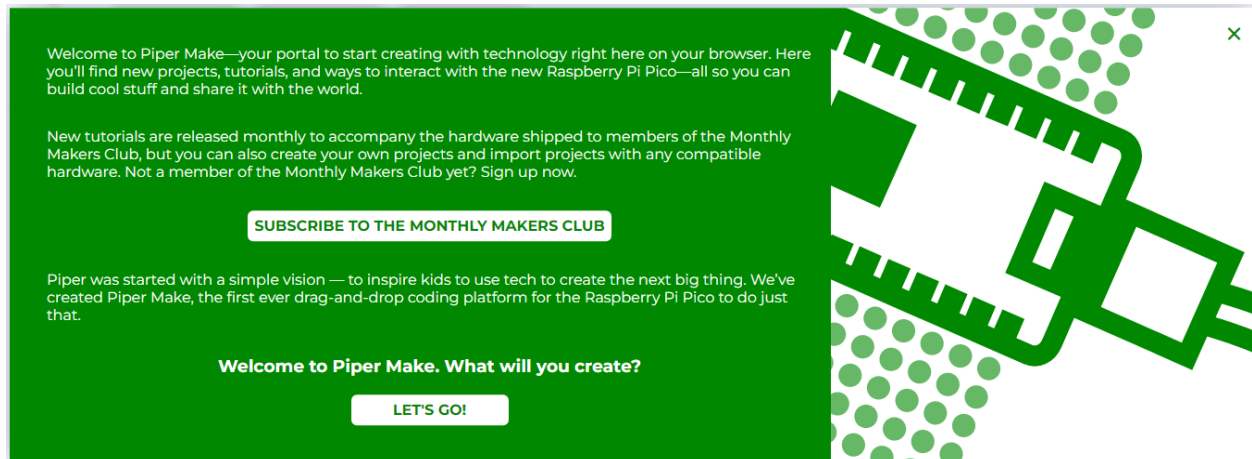
### 1. Einstieg

## 7.1 1.1 Einrichtung des Pico

Zuerst besuchen Sie Piper Make über den folgenden Link:

<https://make.playpiper.com/>

In dem aufpoppenden Fenster können Sie, falls Sie keine weiteren Tutorials abonnieren möchten, einfach auf **Los geht's!** oder den **x**-Button klicken.

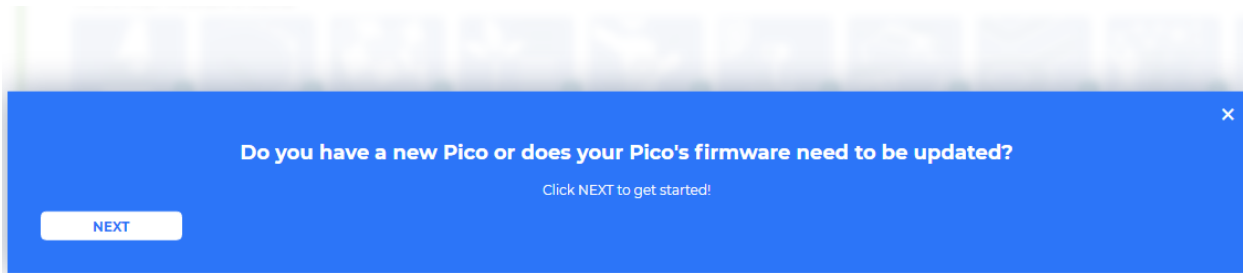


**Hinweis:** Wenn Sie ein anderes Pop-up-Fenster sehen, wird Ihre Browserversion nicht unterstützt. Bitte aktualisieren Sie Ihren Browser und versuchen Sie es erneut.

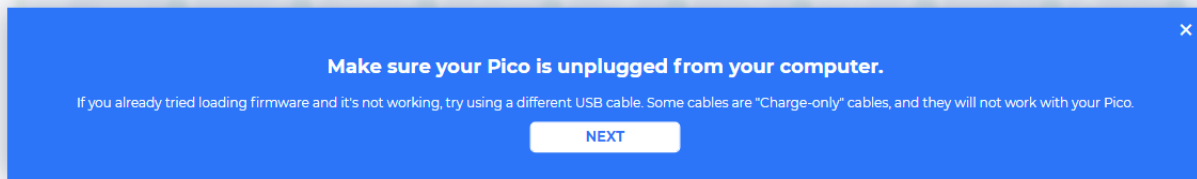
Finden Sie die Schaltfläche **SETUP MY PICO** und klicken Sie darauf. Folgen Sie den Anweisungen zur Konfiguration.



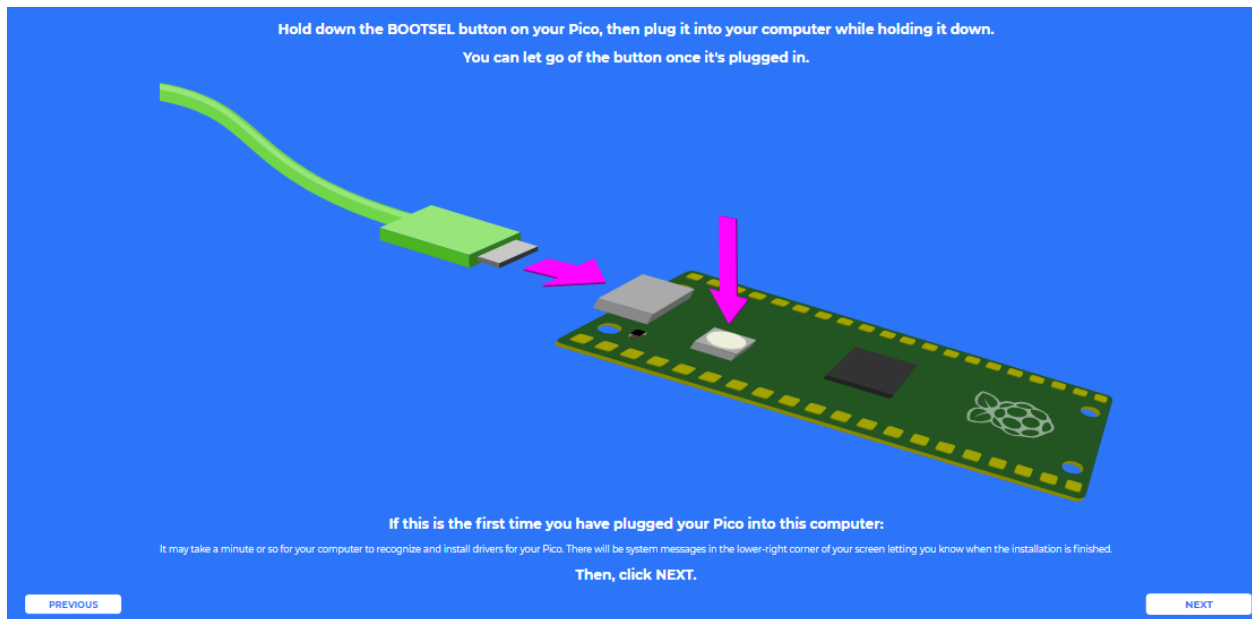
Klicken Sie auf **Weiter**, um mit der Konfiguration Ihres Pico W zu beginnen, selbst wenn Sie diesen bereits zuvor eingerichtet haben. Dies sind die gleichen Schritte, die Sie zur Aktualisierung Ihrer Pico W-Firmware verwenden werden.



In diesem Schritt müssen Sie sicherstellen, dass Ihr Pico W nicht mit Ihrem Computer verbunden ist, da er im nächsten Schritt auf eine bestimmte Weise angeschlossen werden muss. Achten Sie darauf, dass Ihr Kabel sowohl Strom als auch Daten übertragen kann, da viele Micro-USB-Kabel nur Strom liefern.



Drücken Sie nun die RST-Taste (weiß) am Pico W und stecken Sie den Pico W in Ihren Computer. Sobald er angeschlossen ist, können Sie die Taste loslassen.

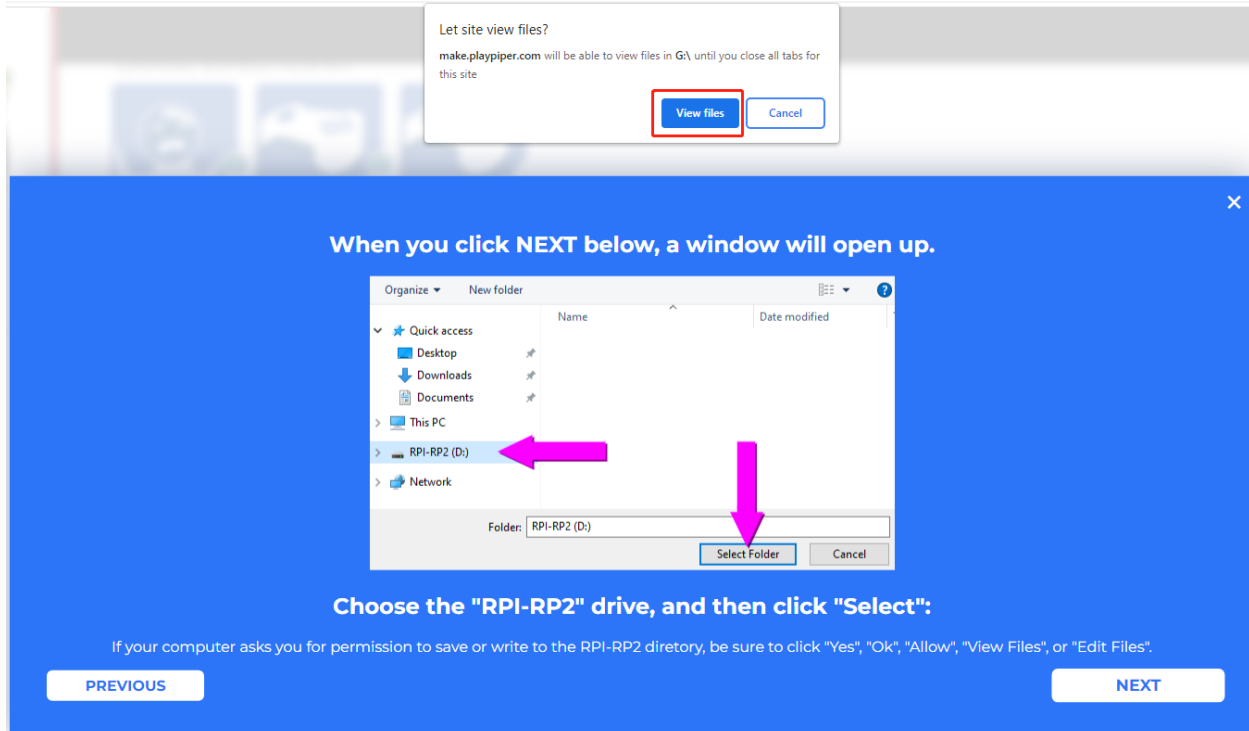


Ihr Pico W wird als USB-Laufwerk angezeigt. Klicken Sie danach auf **Weiter** und wählen Sie das Laufwerk **RPI-RP2** aus.

---

**Hinweis:** Nach der Auswahl des **RPI-RP2**-Laufwerks erscheint oben ein Pop-up-Fenster, in dem Sie der Webseite erlauben müssen, Dateien anzuzeigen.

---



Let site view files?  
make.playpiper.com will be able to view files in G:\ until you close all tabs for this site

**View files** Cancel

When you click **NEXT** below, a window will open up.

Organize New folder

Quick access  
Desktop  
Downloads  
Documents  
This PC  
RPI-RP2 (D:) ←  
Network

Name Date modified

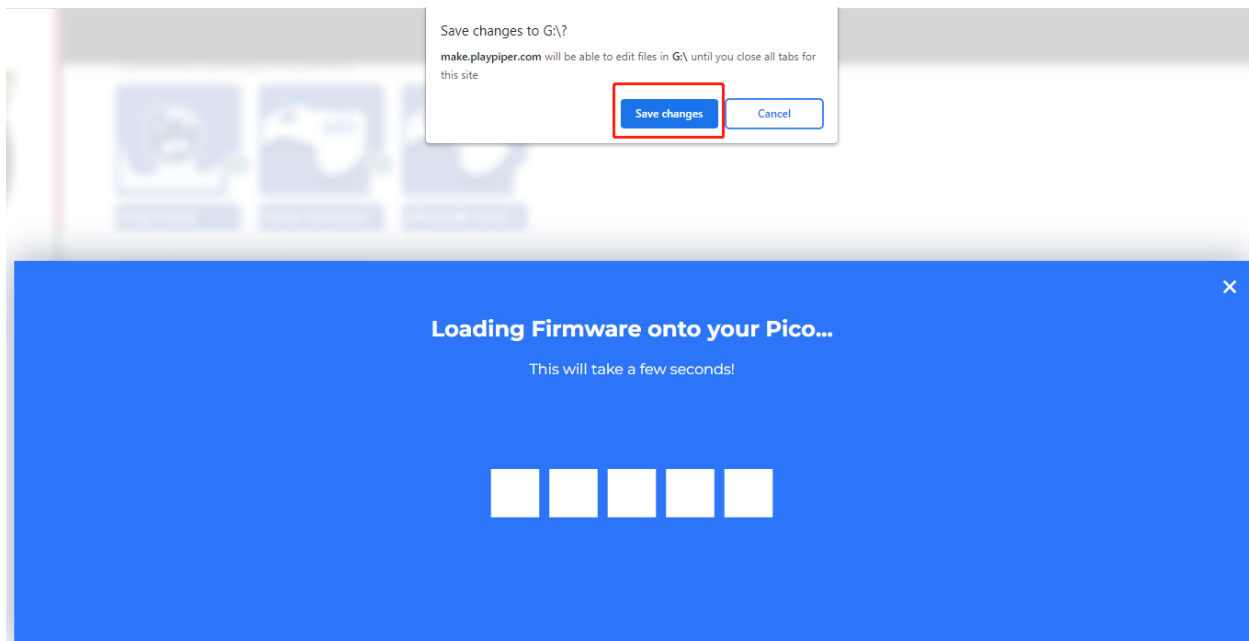
Folder: RPI-RP2 (D:) **Select Folder** Cancel

**Choose the "RPI-RP2" drive, and then click "Select":**

If your computer asks you for permission to save or write to the RPI-RP2 directory, be sure to click "Yes", "Ok", "Allow", "View Files", or "Edit Files".

**PREVIOUS** **NEXT**

Nun wird Piper Make die Firmware auf Ihren Pico W laden. Sie müssen erneut Änderungen auf der Festplatte zulassen, auf der sich der Pico W befindet.



Save changes to G:\?  
make.playpiper.com will be able to edit files in G:\ until you close all tabs for this site

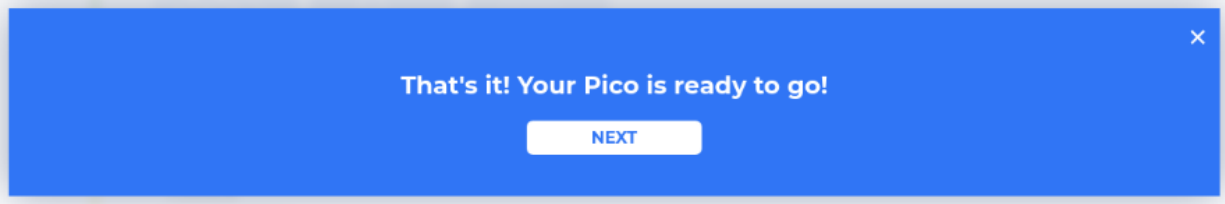
**Save changes** Cancel

**Loading Firmware onto your Pico...**

This will take a few seconds!

Progress bar: 5 white squares

Wenn diese Aufforderung erscheint, bedeutet das, dass Ihr Pico W eingerichtet ist und Sie ihn verwenden können.

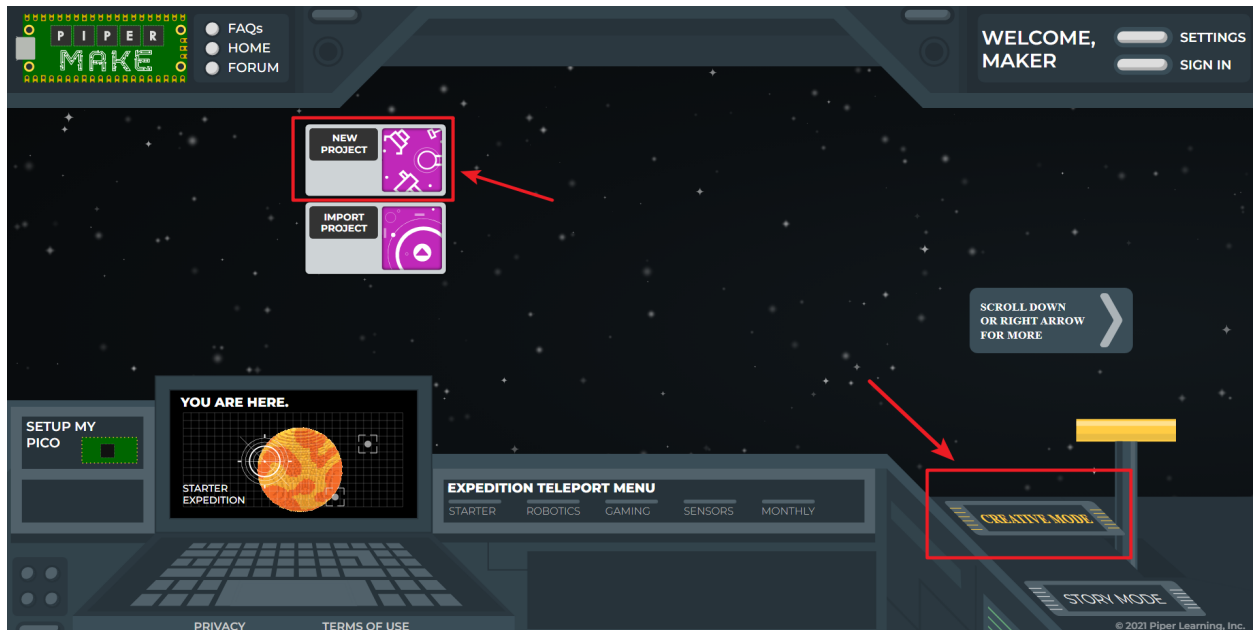


## 7.2 1.2 Schnellstartanleitung für Piper Make

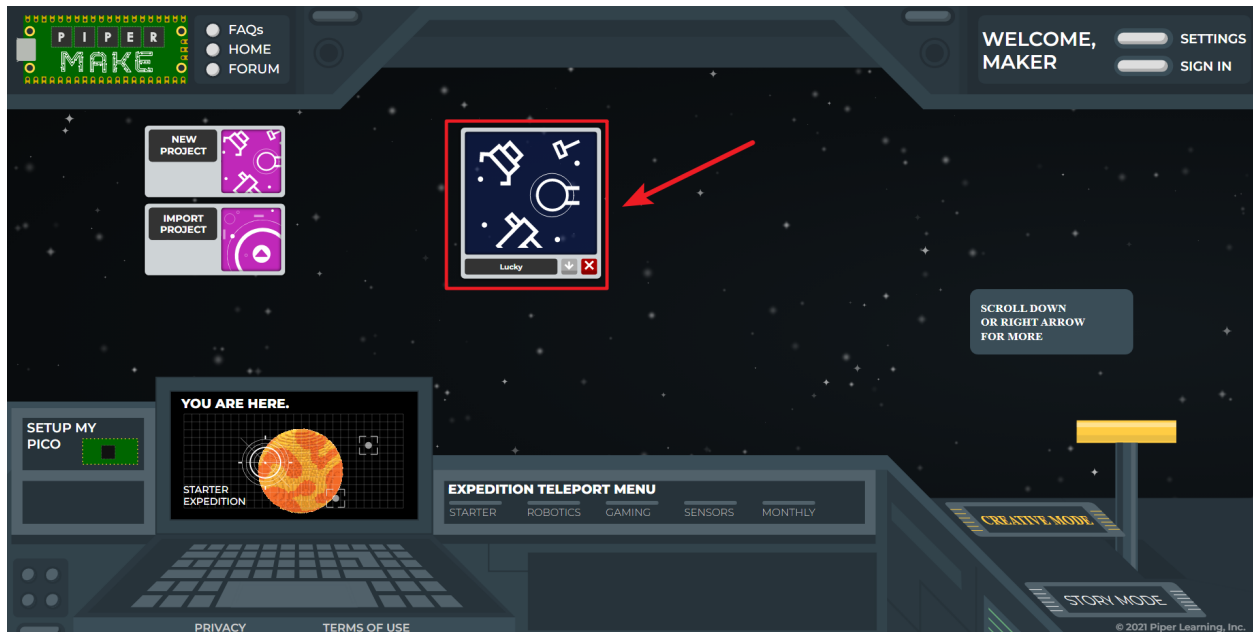
### 7.2.1 1. Neues Projekt erstellen

Nachdem Sie Pico W eingerichtet haben, ist es an der Zeit, die Programmierung zu erlernen. Lassen Sie uns nun die integrierte LED einschalten.

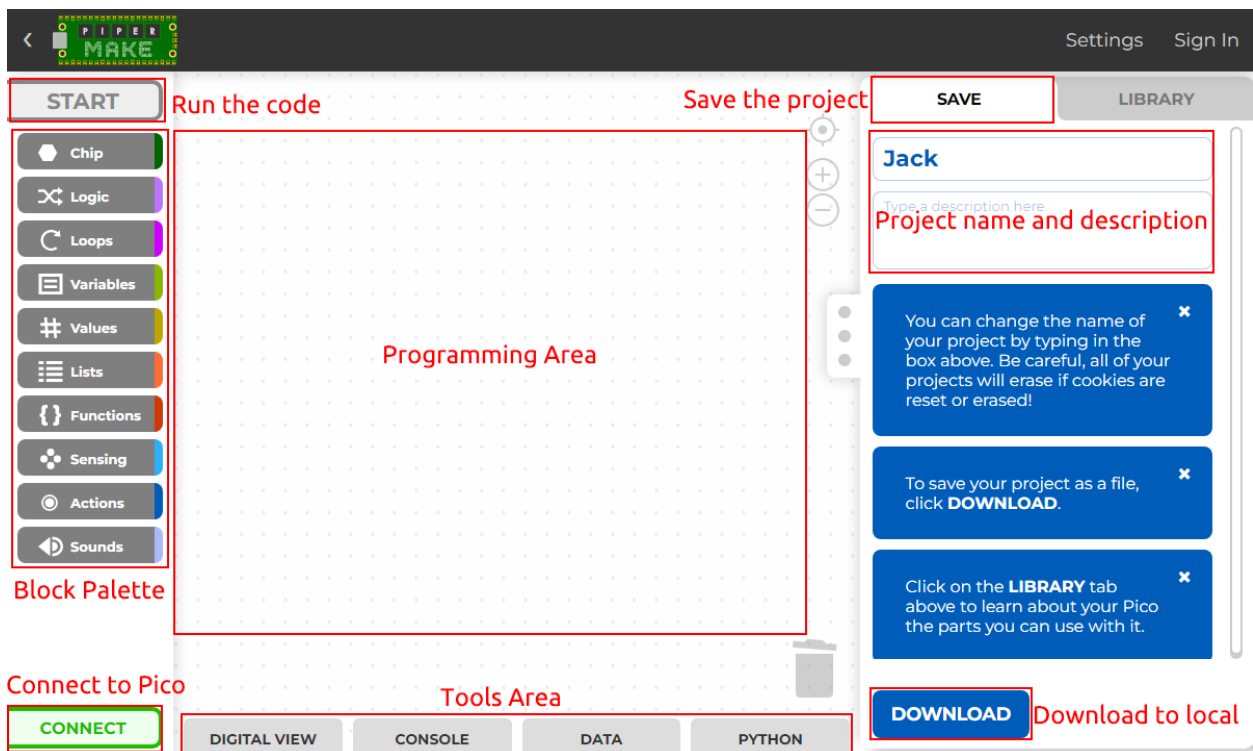
Wechseln Sie in den KREATIVMODUS und klicken Sie auf die Schaltfläche **Neues Projekt**. Ein neues Projekt wird im Abschnitt **MEINE PROJEKTE** angezeigt und erhält einen zufälligen Namen, den Sie von der Programmierseite aus ändern können.



Öffnen Sie dann das neu erstellte Projekt.



Navigieren Sie zur Programmierseite von Piper Make.



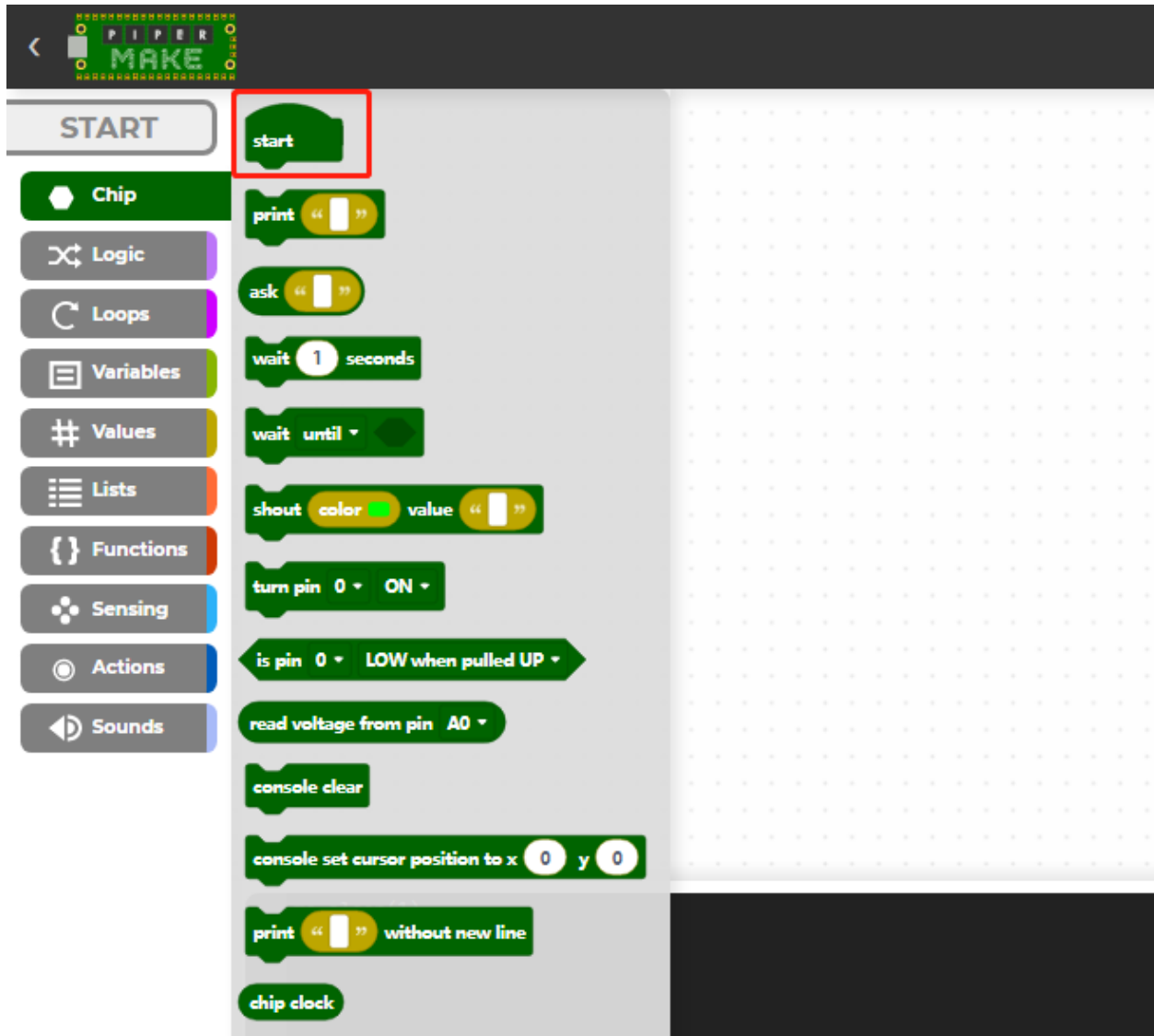
- **START:** Dient zum Ausführen des Codes. Ist es grau, besteht keine Verbindung zu Pico W.
- **Blockpalette:** Enthält verschiedene Arten von Blöcken.
- **VERBINDEN:** Dient zum Herstellen einer Verbindung zu Pico W. Wenn nicht verbunden, ist es grün; nach der Verbindung wird es zu **TRENNEN (rot)**.
- **Programmierbereich:** Ziehen Sie die Blöcke hierher, um die Programmierung abzuschließen.
- **Werkzeugbereich:** Klicken Sie auf **DIGITALANSICHT**, um die Pin-Belegung von Pico W zu sehen; Sie kön-



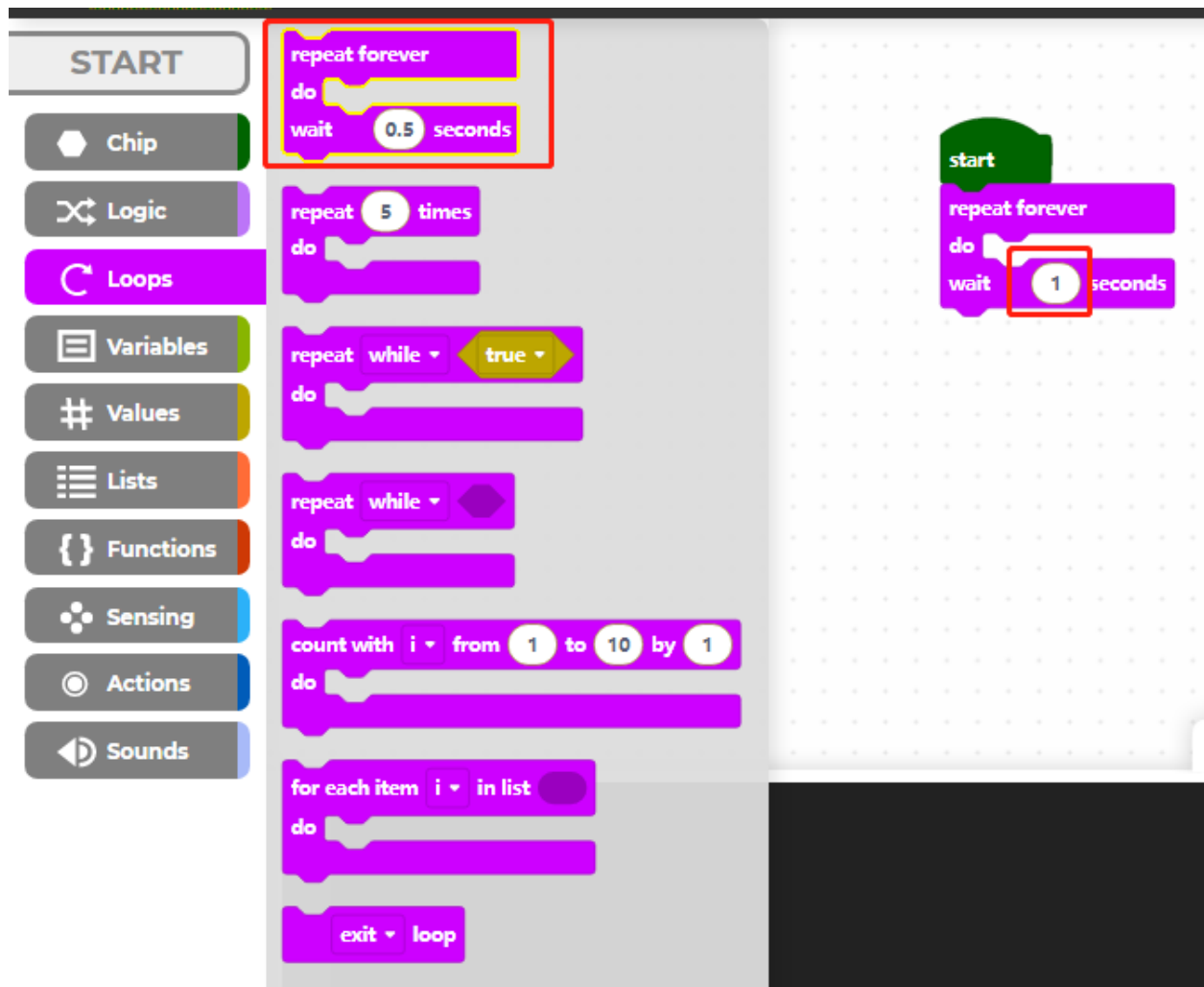
nen die Druckinformationen im **KONSOLE** einsehen; Daten können aus **DATEN** gelesen werden, und Sie können auf **Python** klicken, um den Python-Quellcode anzuzeigen.

- **Projektname und -beschreibung:** Sie können den Projektnamen und die Beschreibung ändern.
- **HERUNTERLADEN:** Sie können auf die Schaltfläche **HERUNTERLADEN** klicken, um sie lokal zu speichern, normalerweise im |-Format. Das nächste Mal können Sie es über die Schaltfläche **Projekt importieren** auf der Startseite importieren.

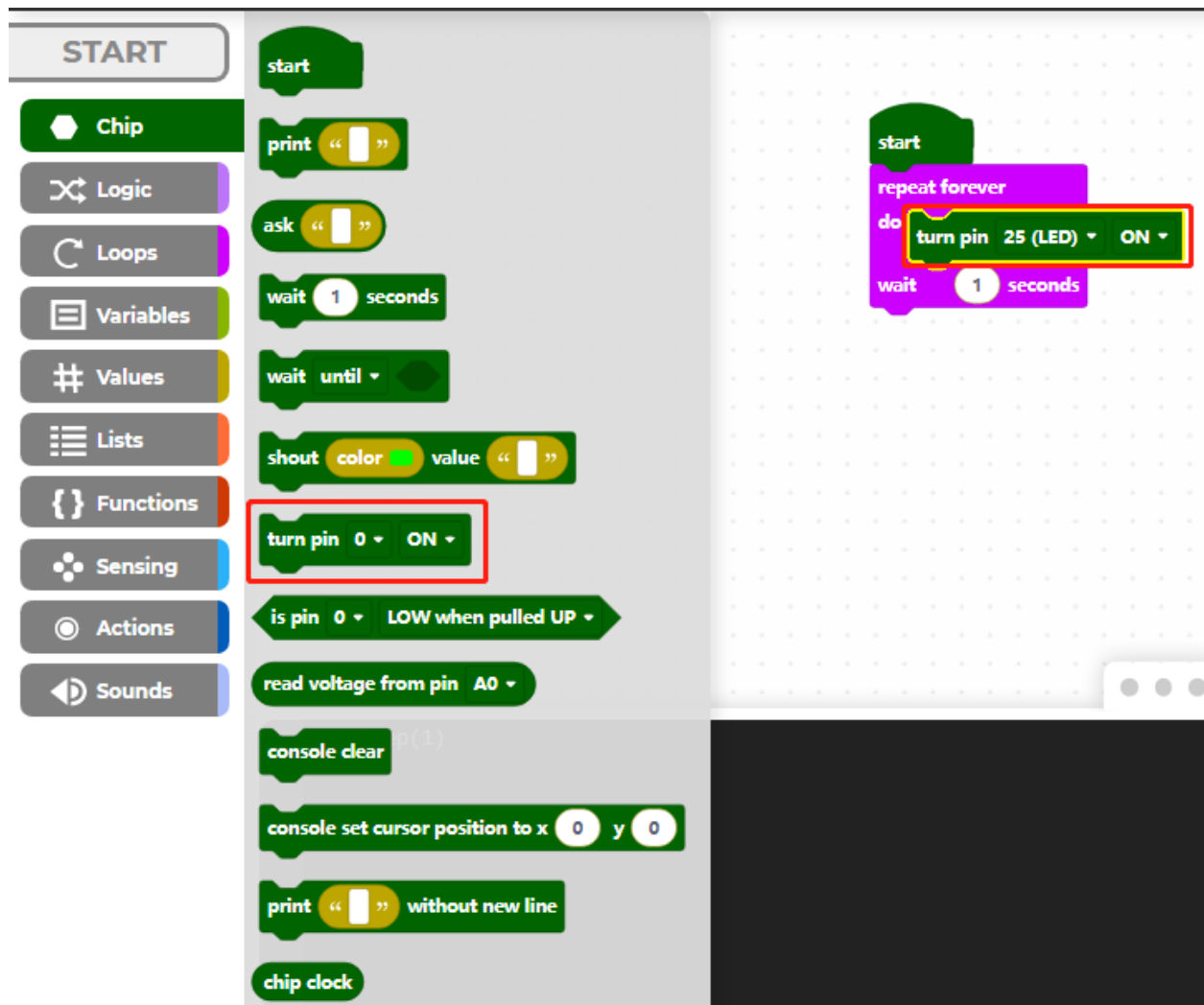
Klicken Sie auf die **Chip**-Palette und ziehen Sie den [Start]-Block in den **Programmierbereich**.



Ziehen Sie dann den [Schleife]-Block aus der **Schleifen**-Palette an das untere Ende des [Start]-Blocks und stellen Sie das Schleifenintervall auf 1 Sekunde ein.

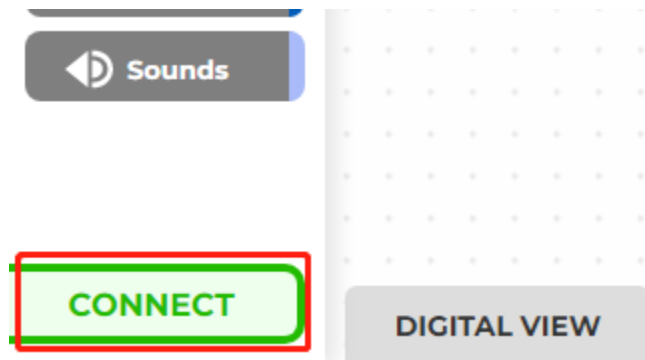


Die integrierte LED des Raspberry Pi Pico befindet sich am Pin25. Daher verwenden wir den Block [Pin () EIN/AUS] aus der **Chip**-Palette zur Steuerung.

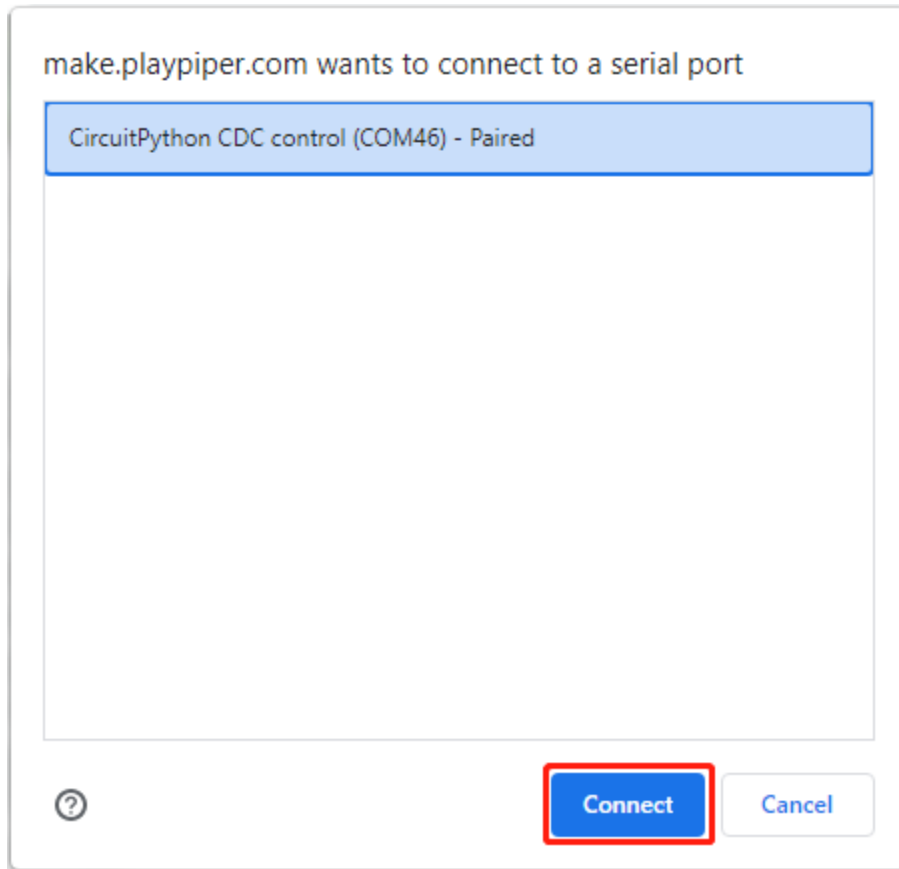


## 7.2.2 2. Verbindung zu Pico W herstellen

Klicken Sie nun auf die Schaltfläche **VERBINDEN**, um eine Verbindung zu Pico W herzustellen. Nach dem Klick erscheint ein neues Popup-Fenster.



Wählen Sie den erkannten **CircuitPython CDC-Steueranschluss (COMXX)** aus und klicken Sie dann auf **Verbinden**.

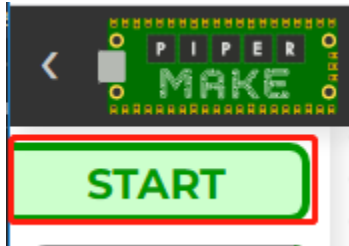


Bei erfolgreicher Verbindung ändert sich das grüne **VERBINDEN** in der unteren linken Ecke in ein rotes **TRENNEN**.

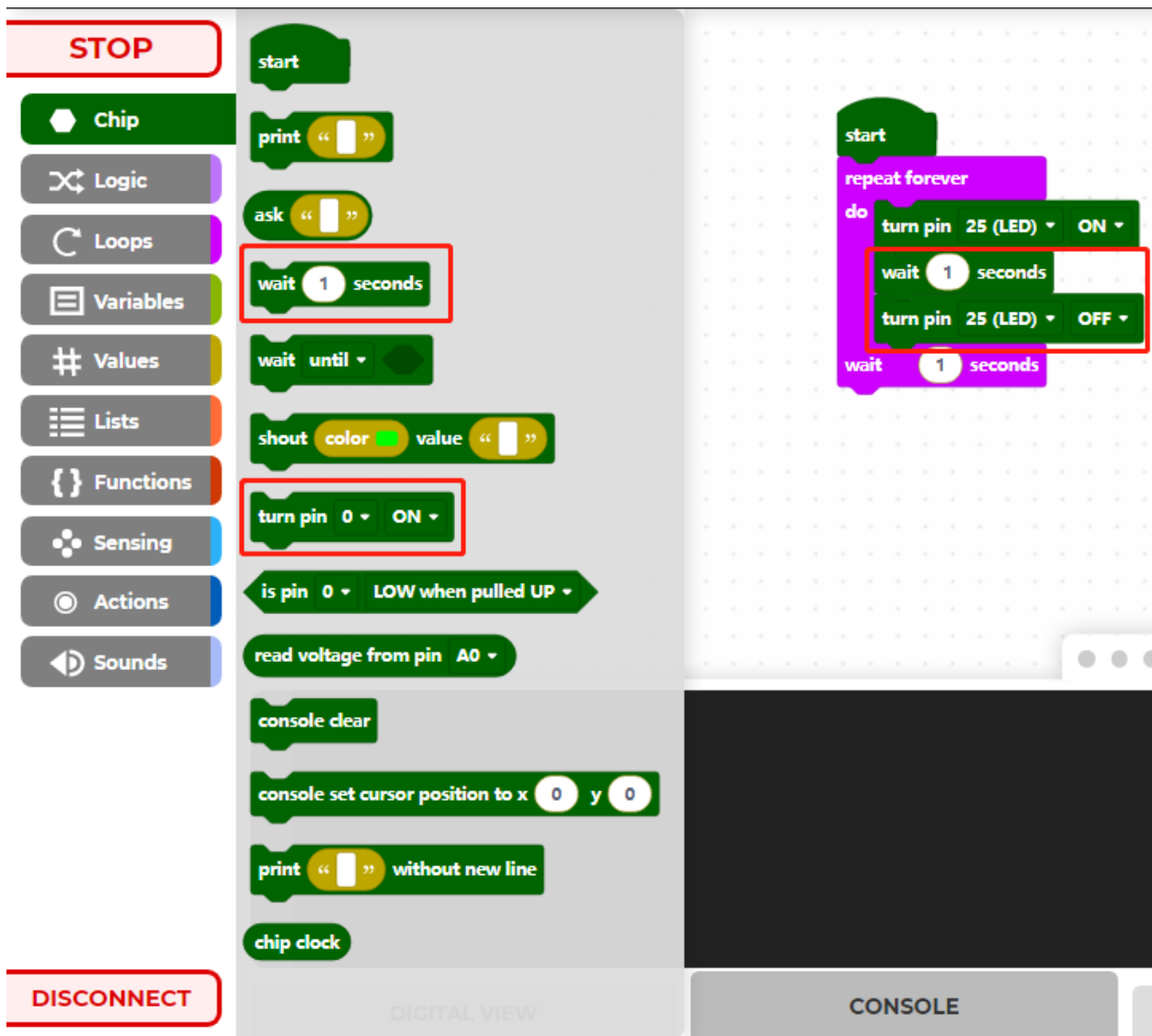


### 7.2.3 3. Code ausführen

Klicken Sie jetzt auf die Schaltfläche **START**, um diesen Code auszuführen, und Sie werden sehen, dass die LED am Pico W leuchtet. Ist die Schaltfläche bei Ihnen grau, bedeutet dies, dass keine Verbindung zu Pico W besteht. Bitte stellen Sie die Verbindung erneut her.



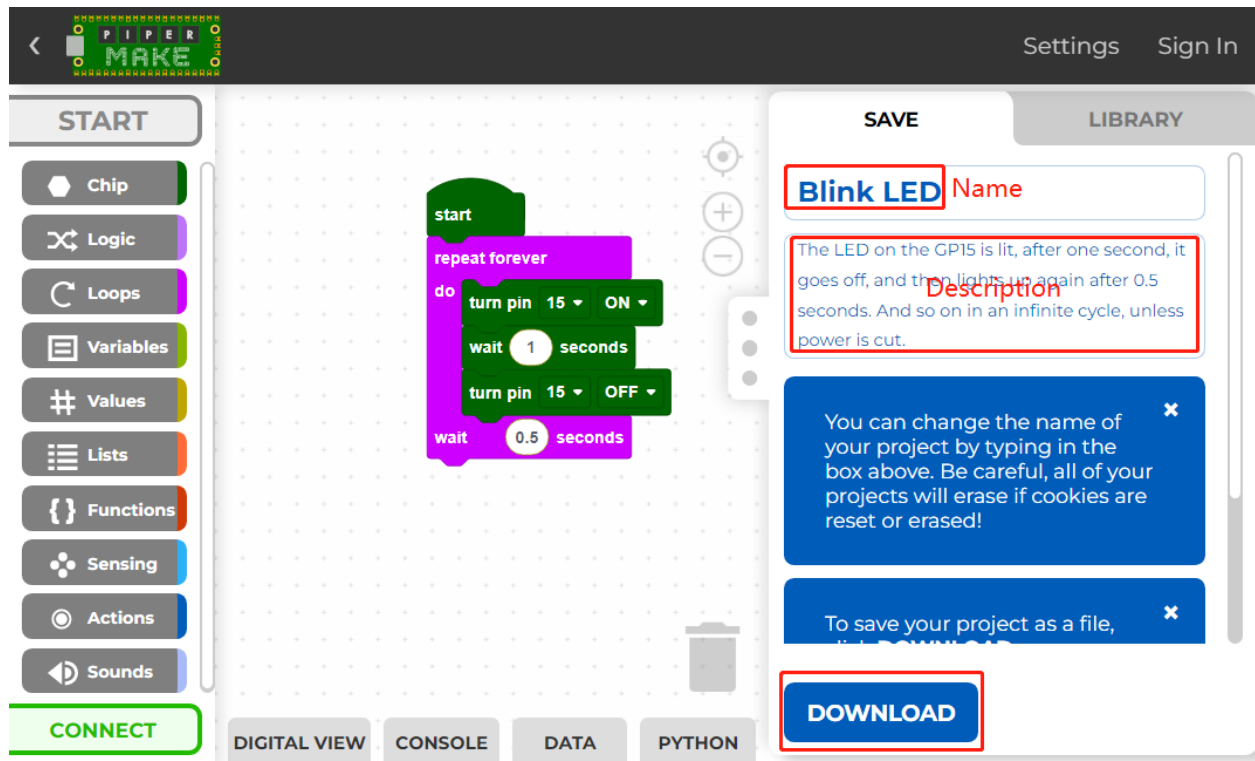
Schalten Sie dann den Pin25 jede Sekunde in der Schleife aus und klicken Sie erneut oben links auf **START**, damit Sie sehen können, wie die integrierte LED blinkt.



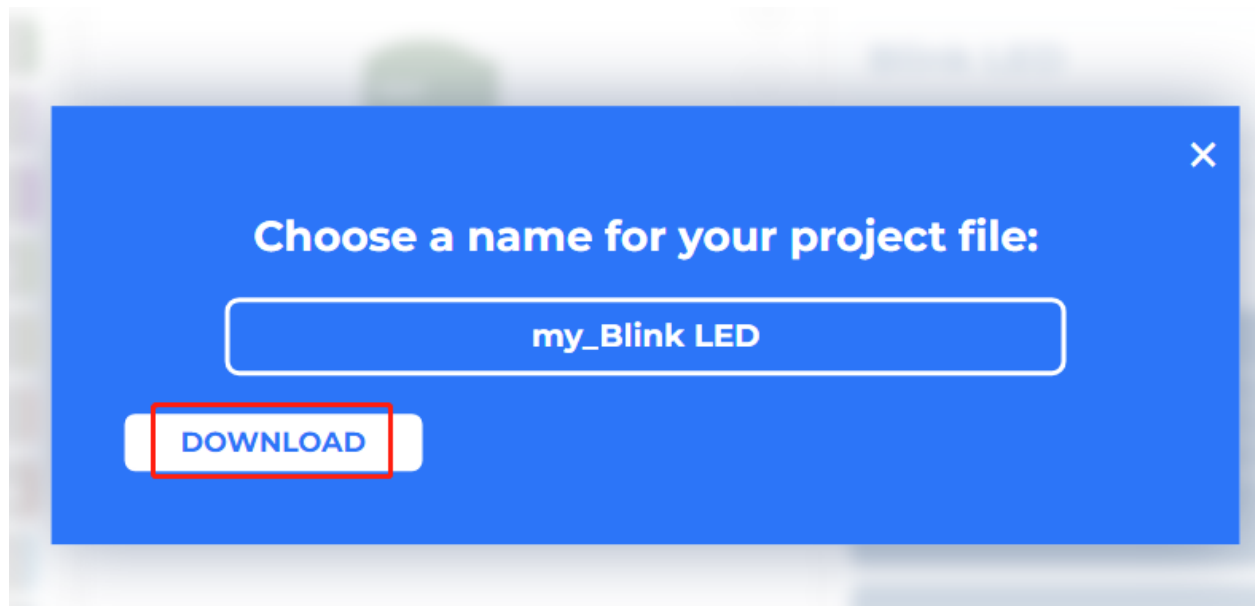
## 7.3 1.3 Wie speichert oder importiert man Code?

### 7.3.1 Code speichern

Nachdem Sie den Code geschrieben haben, können Sie den Codenamen und die Beschreibung ändern und dann auf die Schaltfläche **Herunterladen** klicken, um den Code lokal zu speichern oder ihn mit anderen zu teilen.



Geben Sie anschließend den Dateinamen ein und klicken Sie erneut auf die Schaltfläche **Herunterladen**, um den Code als .png-Datei zu speichern.

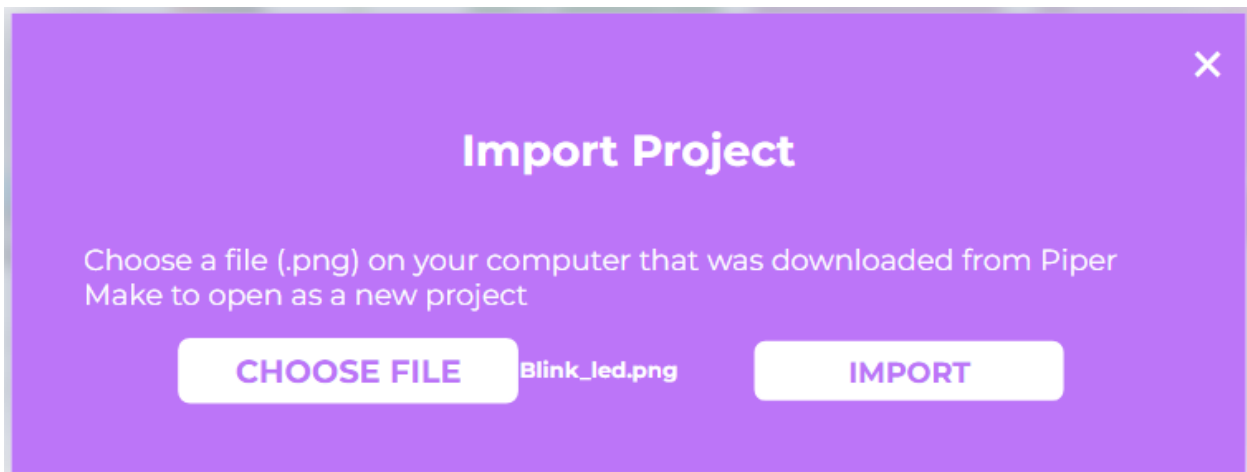


### 7.3.2 Code importieren

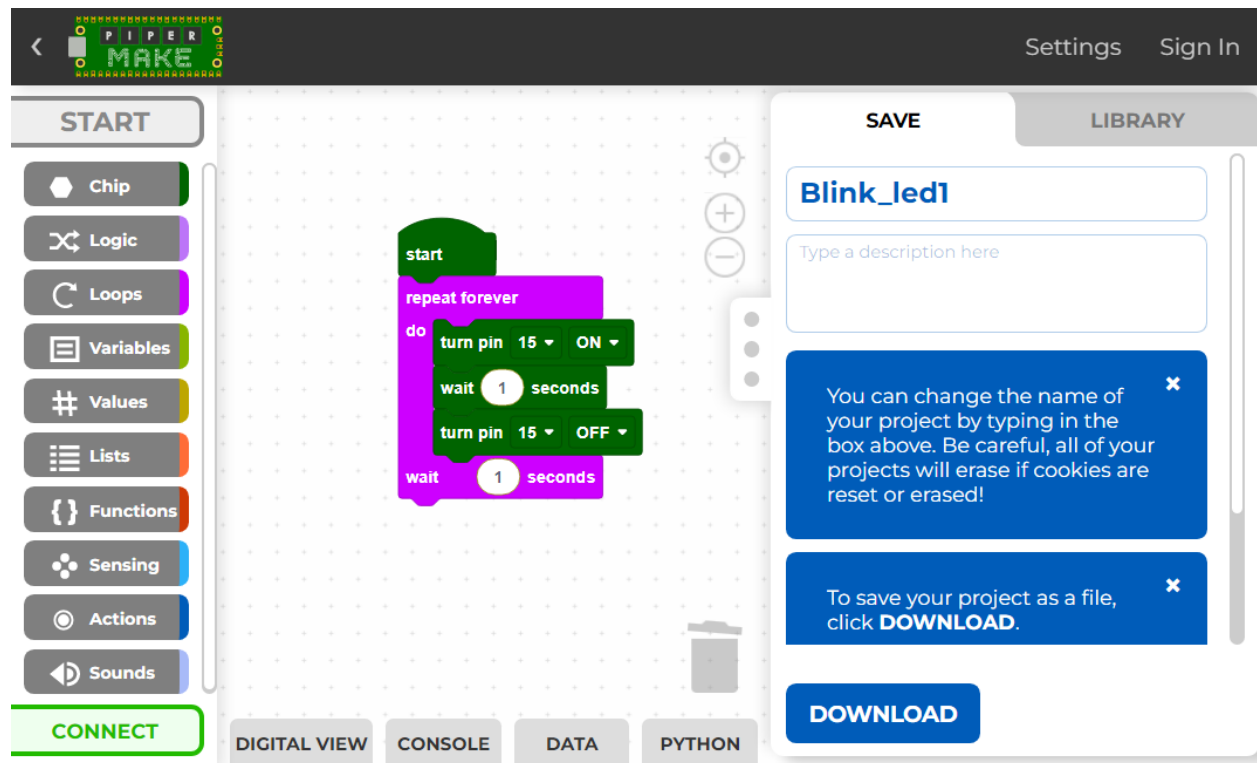
Auf der [Startseite](#) von Piper Make klicken Sie auf **Projekt importieren**.



Wählen Sie die .png-Datei im Pfad `kepler-kit-main\piper` aus und klicken Sie auf **Importieren**. Beachten Sie, dass Sie zuerst das [SunFounder Kepler Kit](#) Paket herunterladen müssen. Oder sehen Sie sich den Code unter [Kepler Kit - GitHub](#) an.



Jetzt können Sie die Datei sehen, die Sie importiert haben.



## 2. Projekte

### 7.4 2.1 LED Blinken Lassen

Für dieses Projekt wollen wir eine externe LED zum Blinken bringen. Ein steckbares Breadboard ist dabei der beste Partner für Anfänger, wenn es um den Einsatz erweiterter elektronischer Bauteile geht.

Das Breadboard ist eine rechteckige Kunststoffplatte mit einer Reihe kleiner Löcher. Diese Löcher ermöglichen uns das einfache Einstecken von elektronischen Bauteilen und das Aufbauen von Schaltkreisen. Die Bauteile werden dabei nicht permanent fixiert, was die Fehlerkorrektur und das Neustarten des Projekts erleichtert.

#### Benötigte Bauteile

Für dieses Projekt benötigen wir folgende Komponenten.

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

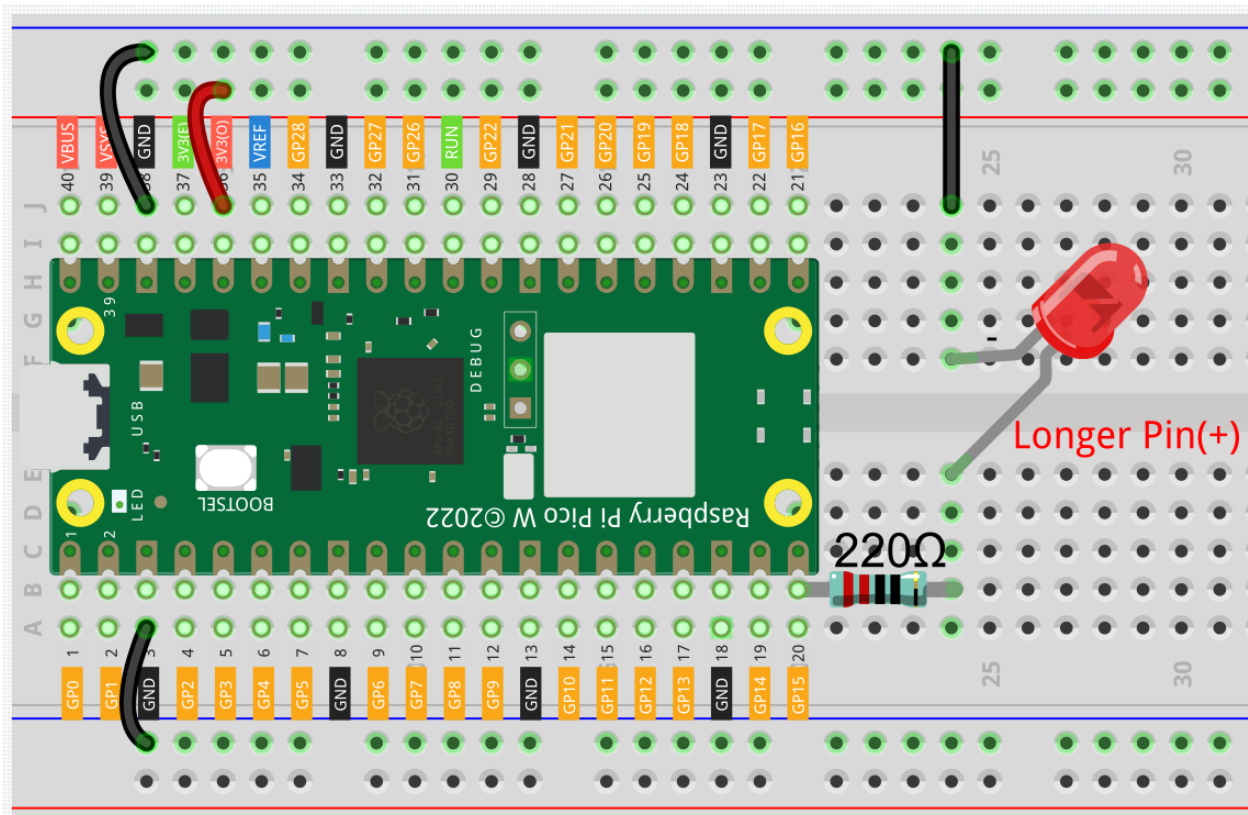
Bezeichnung	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Die Bauteile können auch einzeln über die unten stehenden Links erworben werden.



SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1 (220)	
6	<i>LED</i>	1	

### Verdrahtung

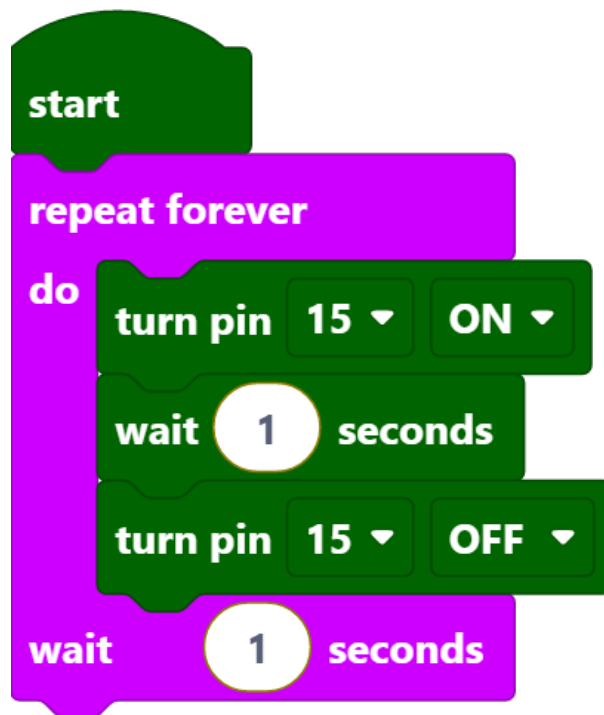


- Der Farbring des 220-Ohm-Widerstands ist rot, rot, schwarz, schwarz und braun.
- Das längere Bein der LED ist als Anode (+) bekannt, das kürzere als Kathode (-).

### Code

#### Hinweis:

- Um den Code zu schreiben, können Sie sich an dem unten stehenden Bild orientieren.
- Importieren Sie `2.1_blink_led.png` aus dem Verzeichnis `kepler-kit-main\piper`. Für detaillierte Anleitungen siehe [Code importieren](#).



Nachdem der Pico W angeschlossen wurde, klicken Sie auf die **Start**-Schaltfläche, und die LED wird anfangen zu blinken. Weitere Details finden Sie unter [1.2 Schnellstartanleitung für Piper Make](#).

#### Funktionsweise

Dies ist der Hauptteil der Schleife: Schalten Sie den Pin15 ein, um die LED leuchten zu lassen, warten Sie eine Sekunde und schalten Sie den Pin15 aus, damit die LED erlischt. Warten Sie eine weitere Sekunde und wiederholen Sie den vorherigen Zyklus, sodass die LED zwischen den Zuständen Ein und Aus wechselt.

- [start]: Dieser Block stellt das Grundgerüst des Programms dar und markiert dessen Beginn.
- [repeat forever do() wait()seconds]: Besagt, dass die darin enthaltenen Blöcke wiederholt ausgeführt werden, wobei das Ausführungsintervall selbst definiert wird.
- [turn pin () ON/OFF]: Legt fest, dass ein bestimmter Pin auf hohem (ON) oder niedrigem (OFF) Niveau liegt.
- [wait () seconds]: Setzt das Ausführungsintervall zwischen den Blöcken fest.

## 7.5 2.2 Taster

In diesem Projekt werden wir lernen, wie man eine LED mit Hilfe eines Tasters ein- oder ausschaltet.

#### Benötigte Bauteile

Für dieses Projekt benötigen wir folgende Komponenten.

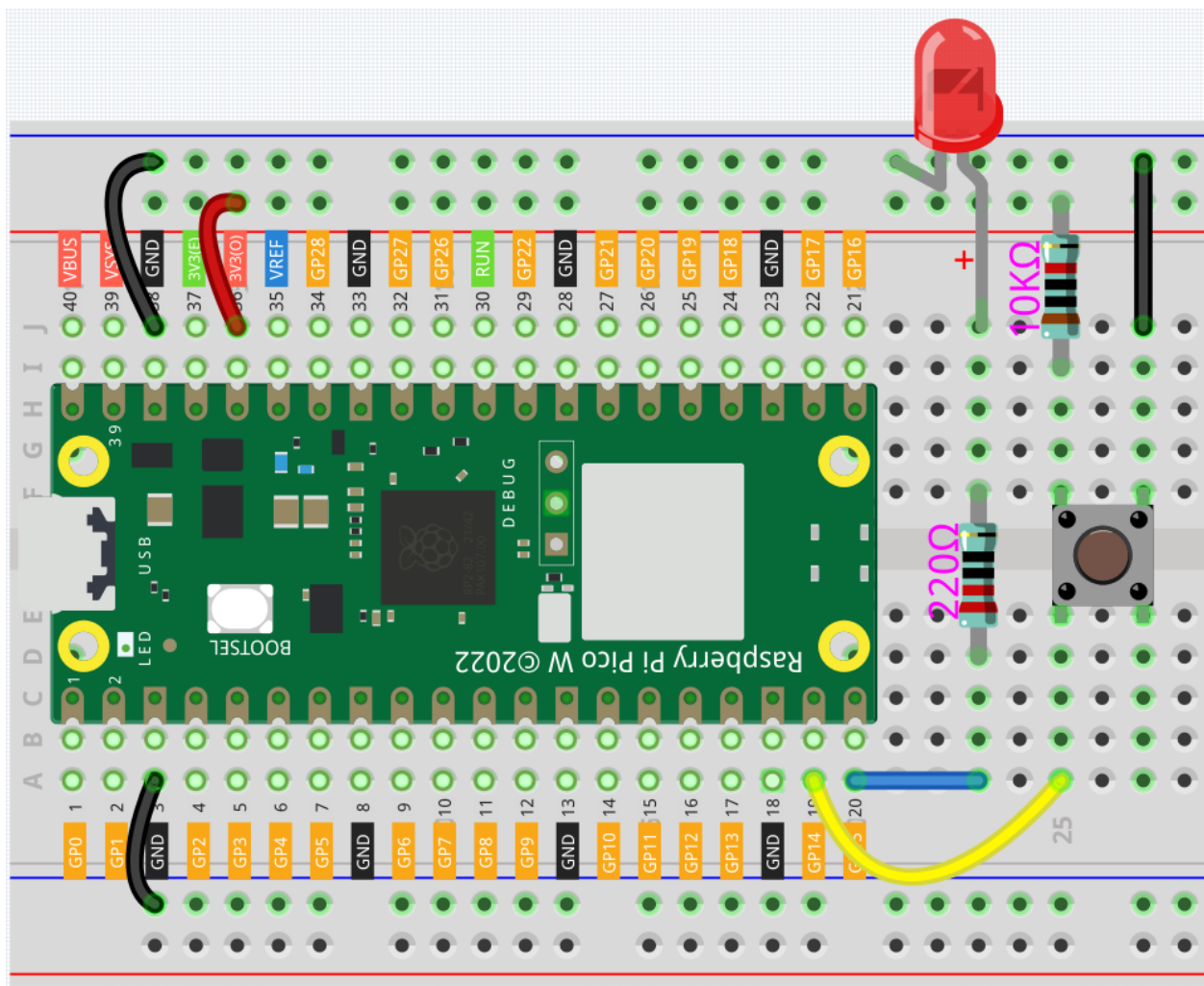
Es ist natürlich praktisch, ein ganzes Kit zu kaufen, hier der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Die Bauteile können auch einzeln über die unten stehenden Links erworben werden.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	2 (220, 10K)	
6	<i>LED</i>	1	
7	<i>Taster</i>	1	

## Verdrahtung



- Ein Pin des Tasters ist mit 3,3V verbunden, der gegenüberliegende Pin mit GP14. Wenn der Taster gedrückt wird,

liegt an GP14 eine hohe Spannung an.

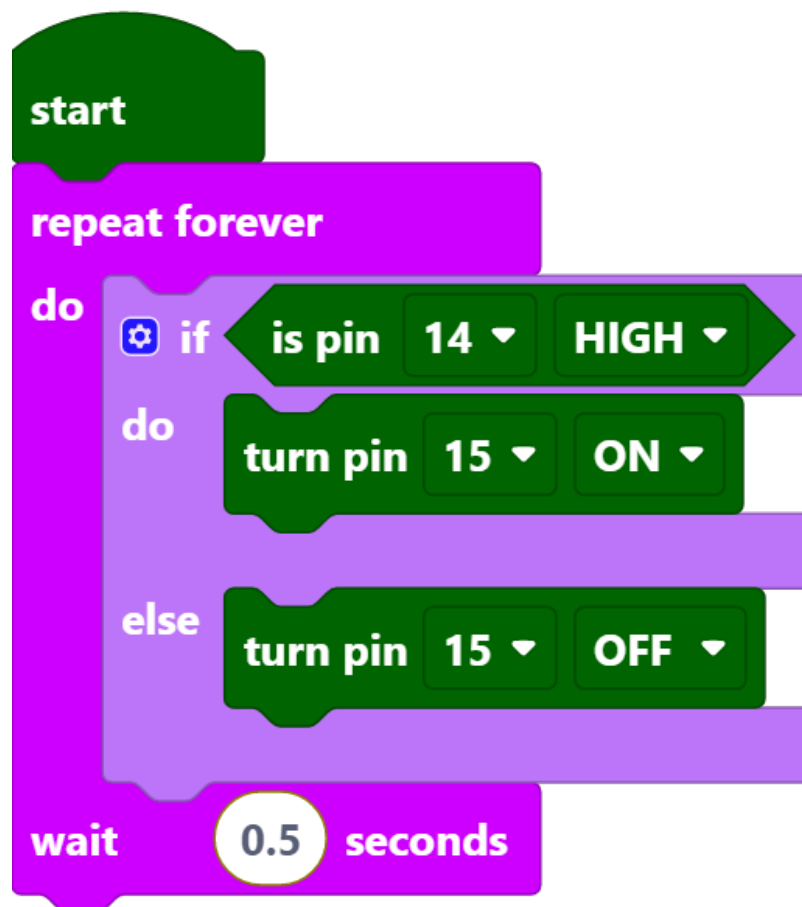
- Wenn der Taster nicht gedrückt ist, befindet sich GP14 in einem unbestimmten Zustand und kann hoch oder niedrig sein. Um einen stabilen niedrigen Pegel zu erhalten, wenn der Taster nicht gedrückt ist, muss GP14 über einen 10K-Pull-down-Widerstand erneut mit GND verbunden werden.

### Code

---

#### Hinweis:

- Sie können den Code durch Ziehen und Ablegen gemäß dem unten stehenden Bild schreiben.
  - Importieren Sie `2.2_button.png` aus dem Pfad `kepler-kit-main\piper`. Für detaillierte Anleitungen siehe [Code importieren](#).
- 



Nachdem der Pico W angeschlossen wurde, klicken Sie auf die **Start**-Schaltfläche und der Code beginnt auszuführen. Wenn der Taster gedrückt wird, leuchtet die LED auf. Wird der Taster losgelassen, erlischt die LED.

#### Funktionsweise

Wenn der Taster gedrückt ist, ist Pin14 hoch. Wenn also Pin14 hoch gelesen wird, schalten Sie Pin15 ein (LED leuchtet); andernfalls schalten Sie Pin15 aus (LED ist aus).

- `[if () do () else ()]`: Dies ist ein Bedingungsblock. Je nach Zustand nach dem `[if]`-Block wird entschieden, ob die Blöcke im `[do]`-Block oder die Blöcke im `[else]`-Block ausgeführt werden.
- `[is pin () HIGH]`: Dient zum Auslesen des Pegels eines bestimmten Pins. Wenn der gelesene Pegel mit dem eingestellten HIGH/LOW übereinstimmt, werden die Blöcke im `[do]`-Block ausgeführt, andernfalls die Blöcke

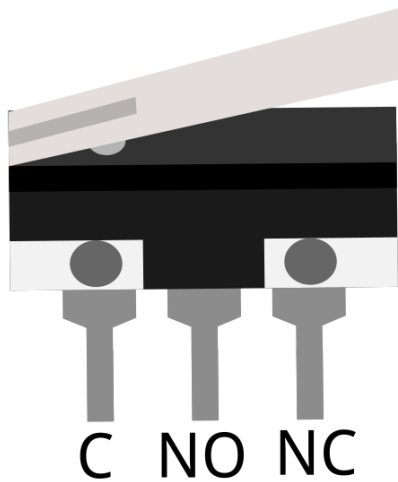
im [else]-Block.

## 7.6 2.3 Serviceklingel

Für dieses Projekt haben wir einen Mikroschalter und einen aktiven Summer verwendet, um eine Serviceklingel zu erstellen. Betätigen Sie den Schalter, und der Summer gibt einen Ton aus.

Der Mikroschalter ist ebenfalls ein 3-poliges Bauteil, die Reihenfolge der 3 Pins lautet C (gemeinsamer Pin), NO (normalerweise offen) und NC (normalerweise geschlossen).

Wenn der Mikroschalter nicht gedrückt ist, sind 1 (C) und 3 (NC) miteinander verbunden, wenn gedrückt, sind 1 (C) und 2 (NO) miteinander verbunden.



### Benötigte Bauteile

Für dieses Projekt benötigen wir folgende Bauteile.

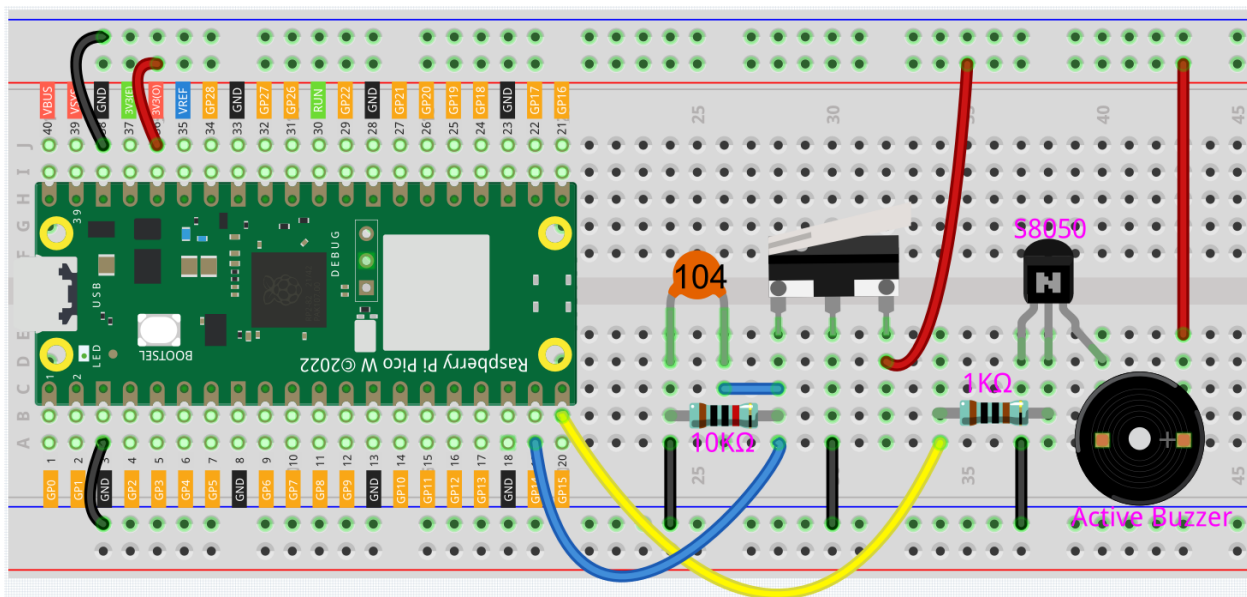
Es ist sicherlich praktisch, ein komplettes Kit zu kaufen, hier der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die Bauteile auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	2 (1K, 10K)	
6	<i>Kondensator</i>	1 (104)	
7	<i>Mikroschalter</i>	1	
8	<i>Transistor</i>	1 (S8050)	
9	Aktiver <i>Summer</i>	1	

## Verdrahtung

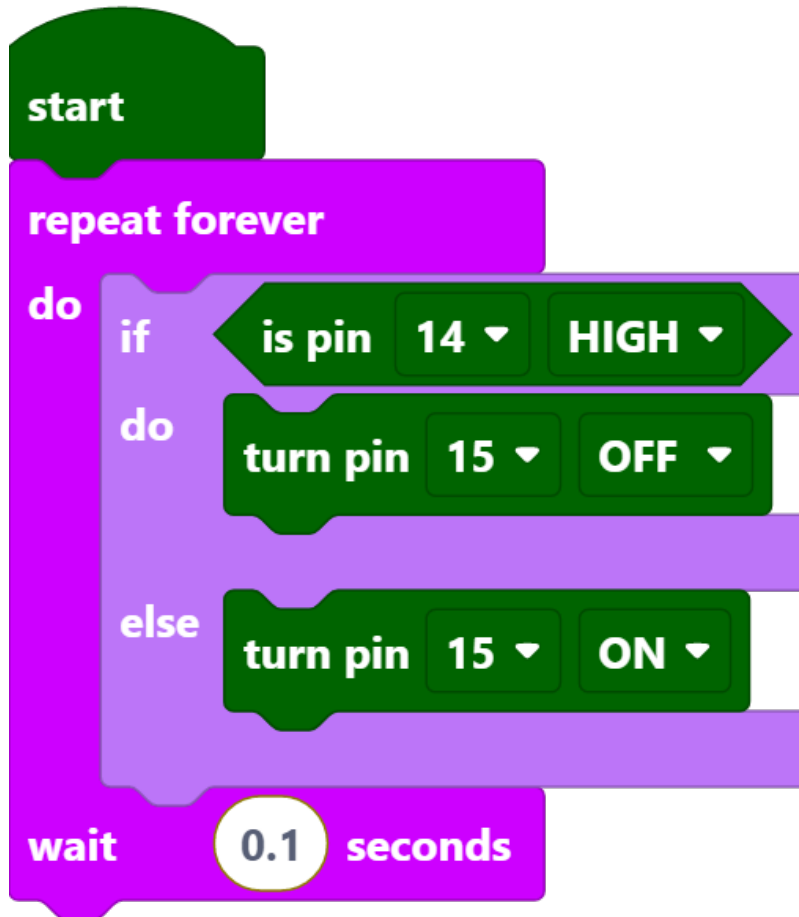


- Standardmäßig sind die Pins 1 und 3 des Mikroschalters miteinander verbunden, und GP14 ist niedrig. Wenn der Mikroschalter gedrückt wird, ist GP14 hoch.
- GP15 gibt ein Hochpegelsignal aus, um den Summer ertönen zu lassen.

### Code

### Hinweis:

- Sie können den unten stehenden Bildern folgen, um den Code per Drag-and-Drop zu schreiben.
- Importieren Sie `2.3_service_bell.png` aus dem Verzeichnis `kepler-kit-main\piper`. Für detaillierte Anleitungen siehe *Code importieren*.



Nachdem der Pico W angeschlossen wurde, klicken Sie auf die **Start**-Schaltfläche und der Code beginnt auszuführen. Tippen Sie auf den Schalter, und der Summer gibt einen Ton aus.

**Hinweis:** Der Code dieses Projekts ist genau derselbe wie im vorherigen Projekt [2.2 Taster](#).

## 7.7 2.4 Regenbogenlicht

In diesem Projekt werden wir RGB-LEDs verwenden, um ein Farbspektrum wie einen Regenbogen darzustellen.

Eine RGB-LED ist im Grunde genommen eine Kapselung einer roten, einer grünen und einer blauen LED unter einer Lampenkappe, wobei alle drei LEDs einen gemeinsamen Kathodenpin teilen. Da jedem Anodenpin ein elektrisches Signal zugeführt wird, kann das Licht der entsprechenden Farbe angezeigt werden. Durch Änderung der Signalstärke an jeder Anode können verschiedene Farben erzeugt werden.

### Benötigte Bauteile

Für dieses Projekt benötigen wir folgende Bauteile.

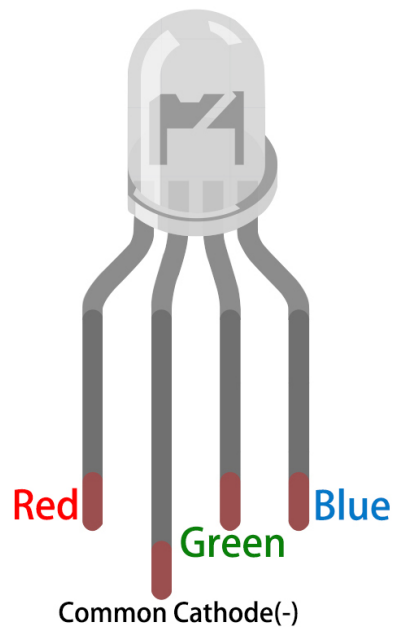
Es ist sicherlich praktisch, ein komplettes Kit zu kaufen, hier der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die Bauteile auch einzeln über die untenstehenden Links erwerben.

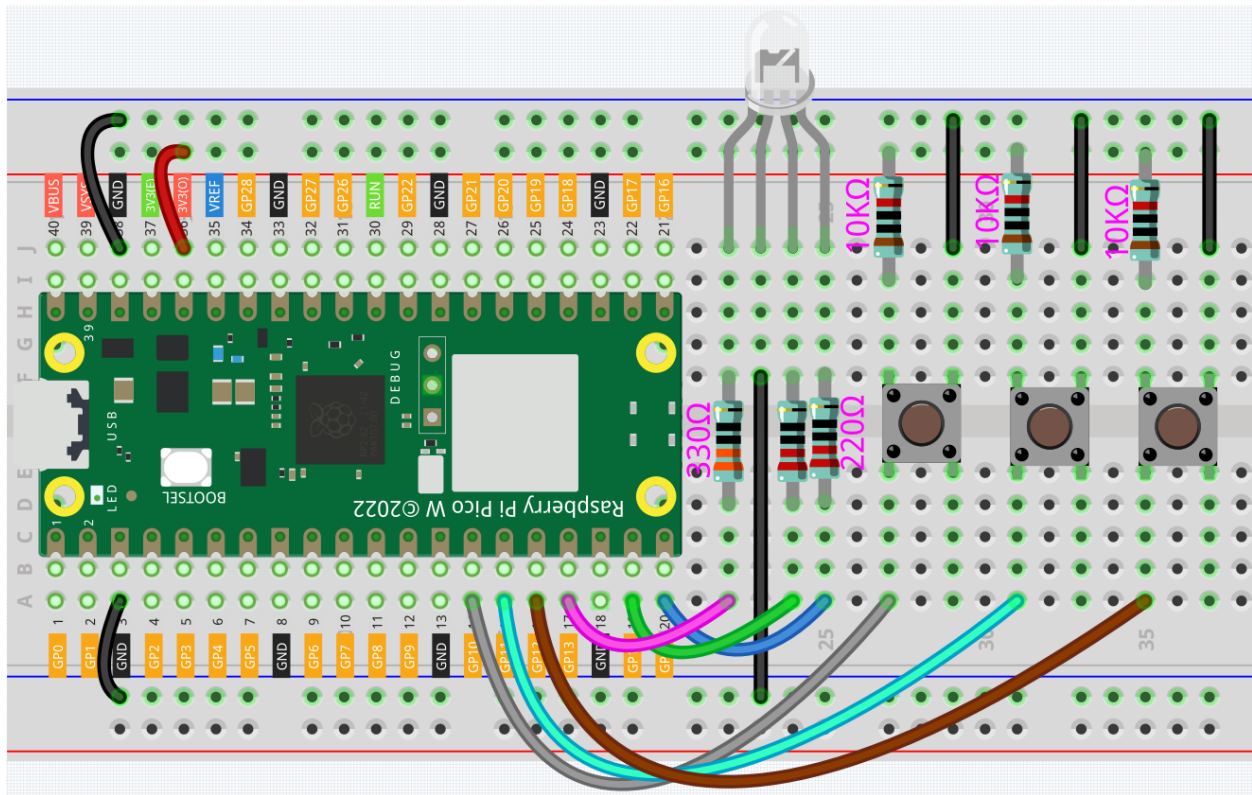
SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	6(1-330, 2-220, 3-10K)	
6	<i>Taster</i>	3	
7	<i>RGB-LED</i>	1	

### Verdrahtung



Eine RGB-LED hat 4 Pins: Der längste Pin ist der gemeinsame Kathodenpin, der normalerweise mit GND verbunden ist. Der Pin links neben dem längsten Pin ist Rot, und die beiden rechten Pins sind Grün und Blau.





- Bei gleicher Stromstärke leuchtet die rote LED heller als die anderen beiden. Daher muss ein etwas größerer Widerstand (330) verwendet werden, um ihre Helligkeit zu reduzieren.
- Die 3 Tasten dienen zur Steuerung der Beleuchtung der roten, grünen und blauen LEDs.

### Code

#### Bemerkung:

- Sie können dem unten stehenden Bild folgen, um den Code per Drag-and-Drop zu schreiben.
- Importieren Sie `2.4_rainbow_light.png` aus dem Verzeichnis `kepler-kit-main\piper`. Für detaillierte Anleitungen siehe [Code importieren](#).



Nachdem der Pico W angeschlossen wurde, klicken Sie auf die **Start**-Schaltfläche und der Code beginnt auszuführen. Ein Druck auf diese Tasten löst jeweils eine einzelne Farbe aus, aber wenn zwei oder alle drei Tasten gleichzeitig gedrückt werden, geben die RGB-LEDs eine Vielzahl von verschiedenen Farben aus, maximal bis zu 7.

**Bemerkung:** Tatsächlich können RGB-LEDs bis zu 16 Millionen Farben darstellen, aber da Piper Make keinen Block zur Ausgabe eines PWM-Signals hat, verwenden wir hier nur den [turn pin() (ON/OFF)]-Block, um 7 Farben darzu-

stellen.

### Funktionsweise

Sie können sich dieses Projekt als die Verwendung von drei Tasten zur Steuerung der RGB-LED vorstellen, wobei drei „if“-Bedingungen festgelegt sind, um zu bestimmen, ob die Tasten gedrückt sind oder nicht. Beim Drücken der Tasten werden die Pegel der entsprechenden Pins hochgezogen, wodurch die RGB-LED verschiedene Farben anzeigt.

## 7.8 2.5 Schlagzeug-Set

In diesem Projekt wollen wir ein Schlagzeug-Set mit drei Tasten und einem Schiebeschalter bauen. Jetzt können Sie Ihr eigenes Schlagzeug spielen.

### Benötigte Bauteile

Für dieses Projekt benötigen wir folgende Bauteile.

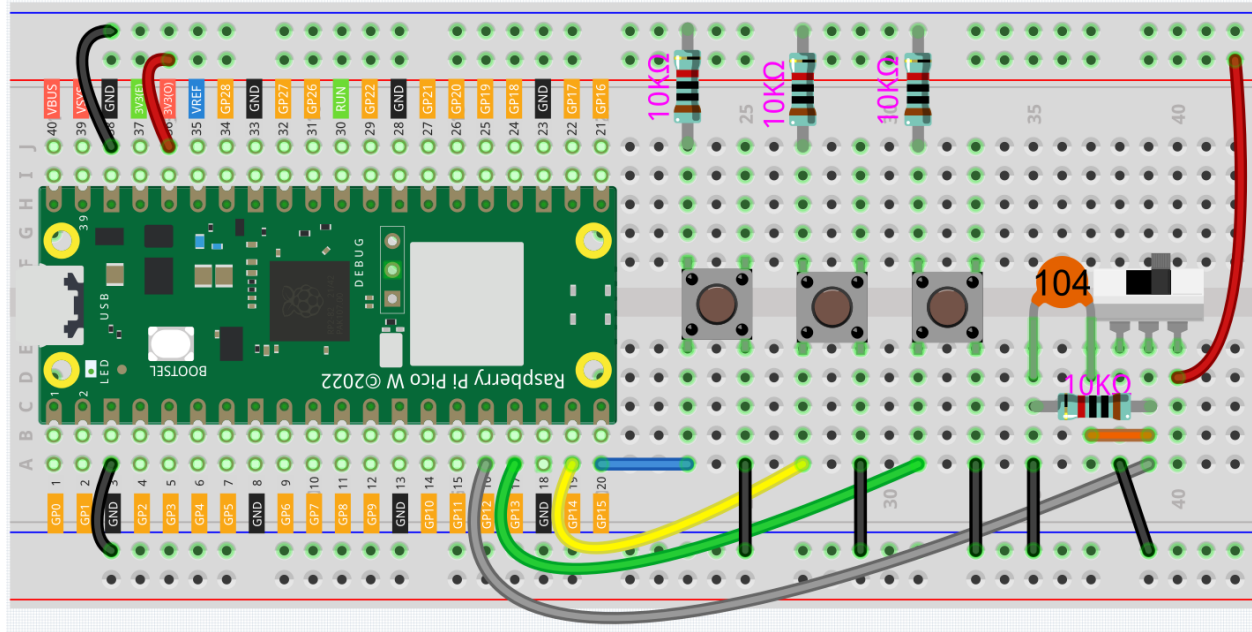
Es ist definitiv praktisch, ein ganzes Kit zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die Bauteile auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<a href="#">Raspberry Pi Pico W</a>	1	
2	Micro-USB-Kabel	1	
3	<a href="#">Steckbrett</a>	1	
4	<a href="#">Jumperkabel</a>	Mehrere	
5	<a href="#">Widerstand</a>	4(10K)	
6	<a href="#">Taster</a>	3	
7	<a href="#">Kondensator</a>	1(104)	
8	<a href="#">Mikroschalter</a>	1	

### Verdrahtung



- Wenn der Schiebeschalter nach rechts verschoben wird, ist GP12 hoch; wenn er nach links verschoben wird, ist GP12 niedrig.
- Jeder der drei Tasten ist mit einem Pull-down-Widerstand verbunden. Standardmäßig sind GP13~GP15 niedrig; wenn die Taste gedrückt wird, werden GP13~GP15 hoch.

## Code

### Bemerkung:

- Sie können dem unten stehenden Bild folgen, um den Code per Drag-and-Drop zu schreiben.
- Importieren Sie `2.5_drum_kit.png` aus dem Verzeichnis `kepler-kit-main\piper`. Für detaillierte Anleitungen siehe [Code importieren](#).



Nachdem der Pico W angeschlossen wurde, klicken Sie auf die **Start**-Schaltfläche und der Code beginnt auszuführen. Durch Drücken verschiedener Tasten oder Verschieben des Schiebeschalters werden unterschiedliche Schlagzeugklänge erzeugt, ganz wie bei einem echten Schlagzeug-Set.

**Bemerkung:** Wenn Sie einen Computer verwenden, müssen Sie Kopfhörer oder Lautsprecher an Ihren Computer anschließen, um den erzeugten Klang zu hören.

## 7.9 2.6 Intelligenter Wassertank

In diesem Projekt verwenden wir einen Wasserstandssensor und ein Servo, um einen intelligenten Wassertank zu simulieren. Der Wasserstandssensor ist im Tank angebracht, um den Wasserstand zu messen. Sobald dieser unter einen bestimmten Schwellenwert fällt, öffnet das vom Servo gesteuerte Ventil, um Wasser nachzufüllen.

### Benötigte Komponenten

Für dieses Projekt werden die folgenden Komponenten benötigt.

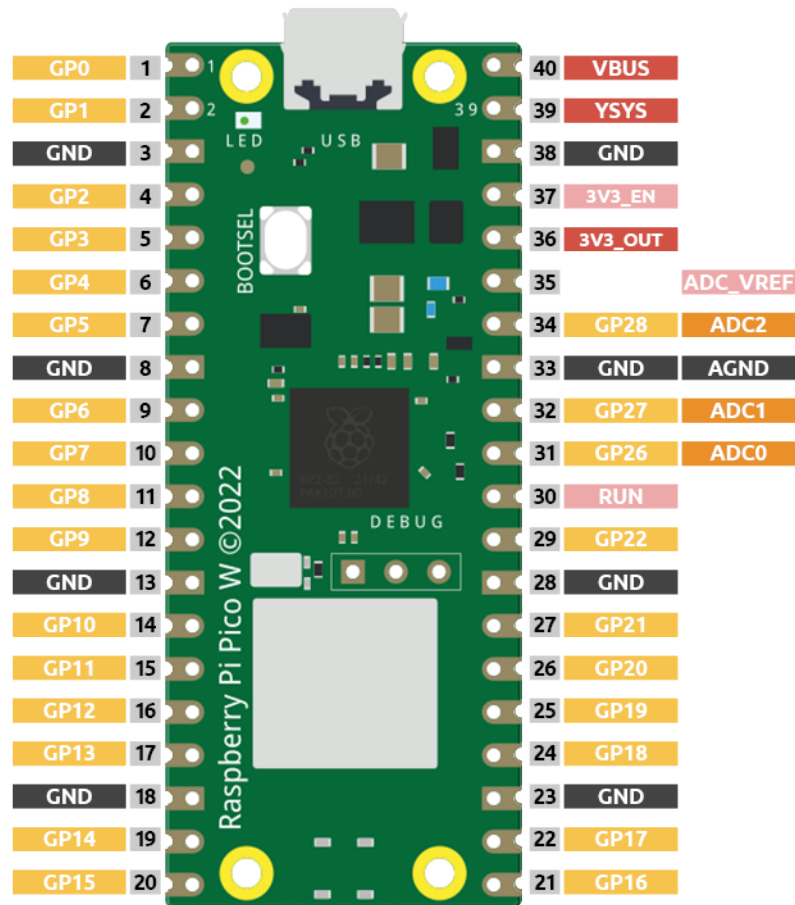
Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	KOMPONENTEN IN DIESEM SET	LINK
Kepler Kit	450+	

Alternativ können die Komponenten auch einzeln über die untenstehenden Links erworben werden.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<a href="#">Raspberry Pi Pico W</a>	1	
2	Micro-USB-Kabel	1	
3	<a href="#">Steckbrett</a>	1	
4	<a href="#">Jumperkabel</a>	Mehrere	
5	<a href="#">Servo</a>	1	
6	<a href="#">Wasserspiegelsensor-Modul</a>	1	

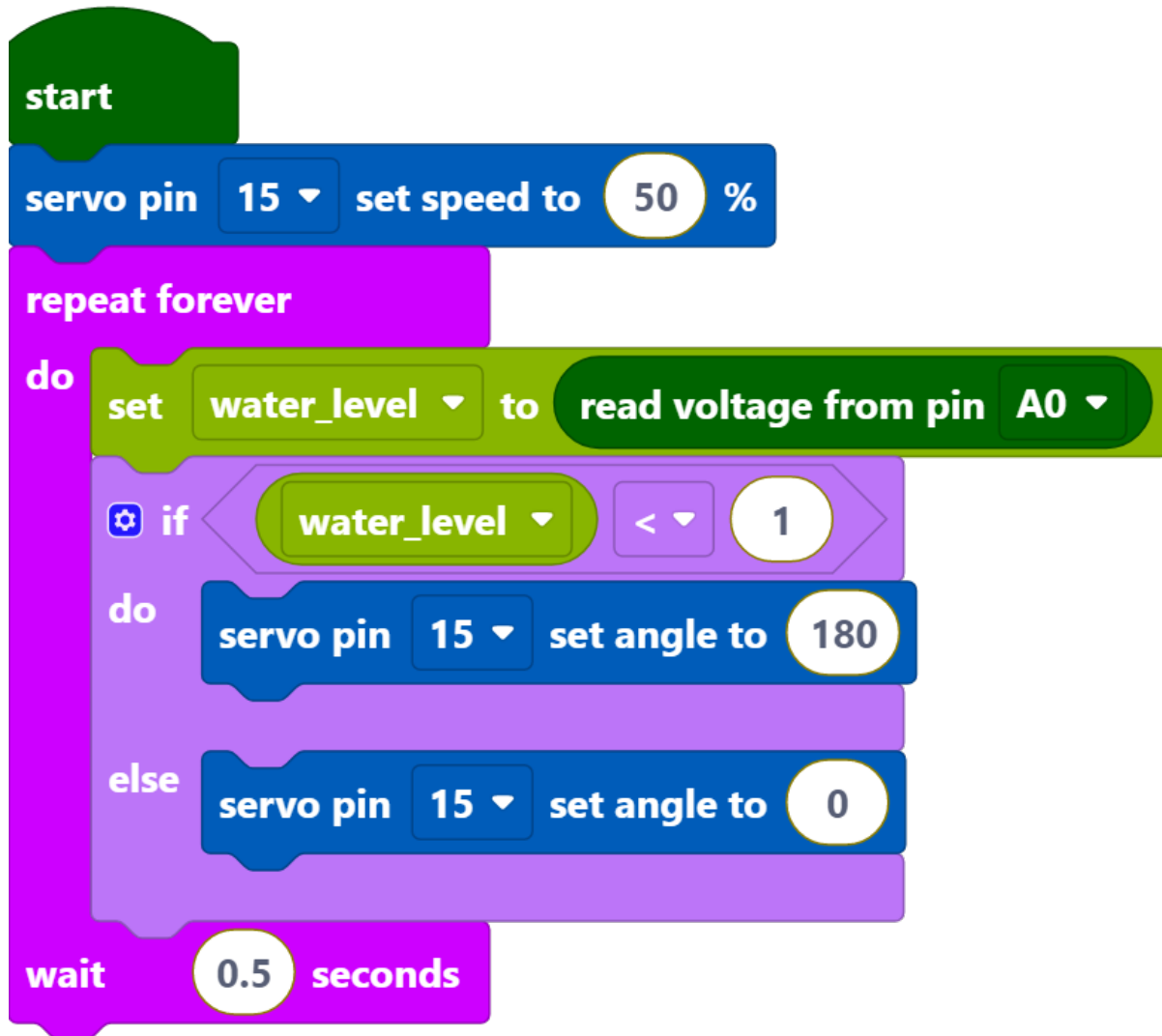
### Verkabelung



Der Pico W verfügt über drei GPIO-Pins, die analoge Eingaben nutzen können: GP26, GP27, GP28, also die analogen Kanäle 0, 1 und 2. Darüber hinaus gibt es einen vierten analogen Kanal, der mit dem eingebauten Temperatursensor verbunden ist und hier nicht weiter behandelt wird.

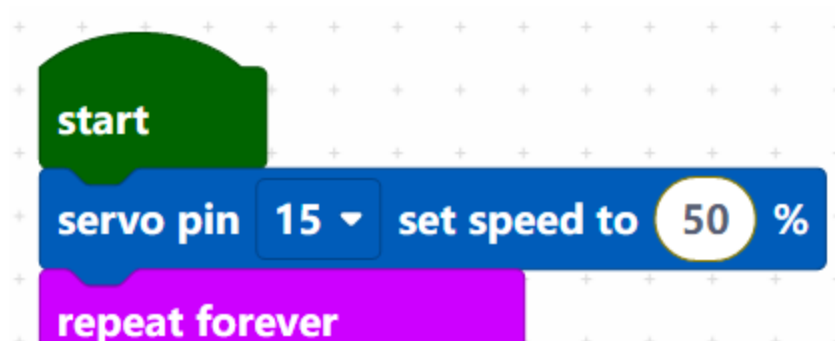






Nach dem Anschluss von Pico W klicken Sie auf die **Start**-Taste und der Code wird ausgeführt. Wenn der Wasserstand unter ein Drittel des Sensors fällt, dreht sich das Servo auf 180 Grad, um den Einlass zu öffnen; ist der Wasserstand höher als ein Drittel des Sensors, dreht sich das Servo auf 0 Grad, um den Einlass zu schließen.

#### Funktionsweise



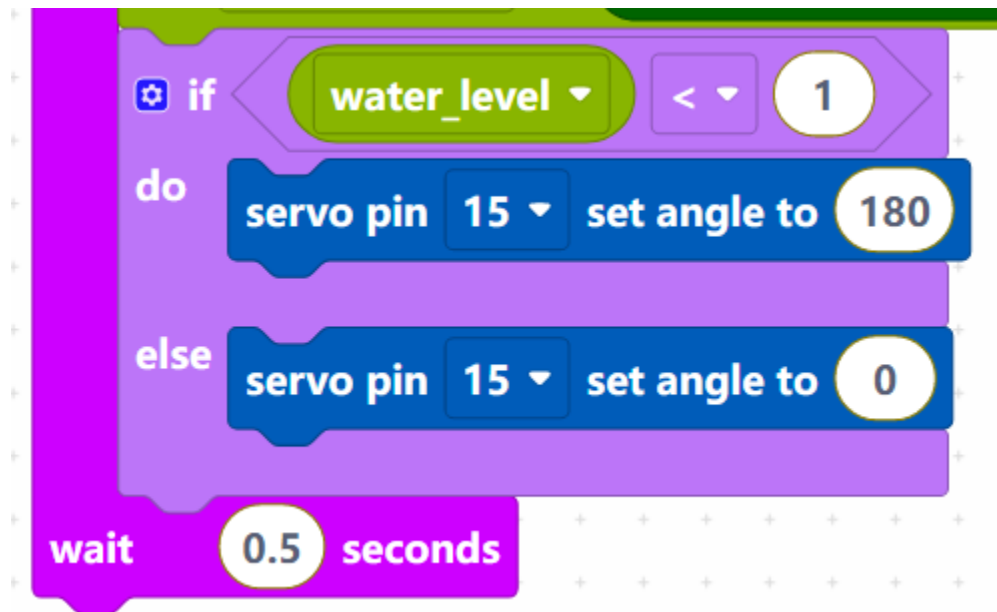
Stellen Sie die Rotationsgeschwindigkeit von Pin 15 (Servo) auf 15% ein.

- [servo pin() set speed to ()%]: Dient zur Einstellung der Rotationsgeschwindigkeit des Servo-Pins. Der Bereich liegt zwischen 0% und 100%.



Liest den Wert von Pin A0 und speichert ihn in der Variable [water\_level].

- [set (water\_level) to]: Dient zur Einstellung des Variablenwerts. Die Variable muss aus der **Variables**-Palette erstellt werden.
- [read voltage from pin ()]: Dient zum Ablesen der Spannung der analogen Pins (A0~A2). Der Bereich liegt zwischen 0 und 3,3 V.



Legen Sie den Spannungsschwellenwert auf 1 fest. Wenn die Spannung des Wasserstandssensors unter 1 liegt, soll sich das Servo auf die 180°-Position drehen, andernfalls auf die 0°-Position.

- [servo pin () set angle to ()]: Stellt den Winkel des Servo-Pins ein. Der Bereich liegt zwischen 0 und 180 Grad.

## 7.10 2.7 Schwenk-Servo

In diesem Projekt verwenden wir ein Servo und ein Potentiometer, um ein Lenkrad zu simulieren. Das Drehen des Potentiometers bewirkt, dass das Servo gleichzeitig schwenkt.

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

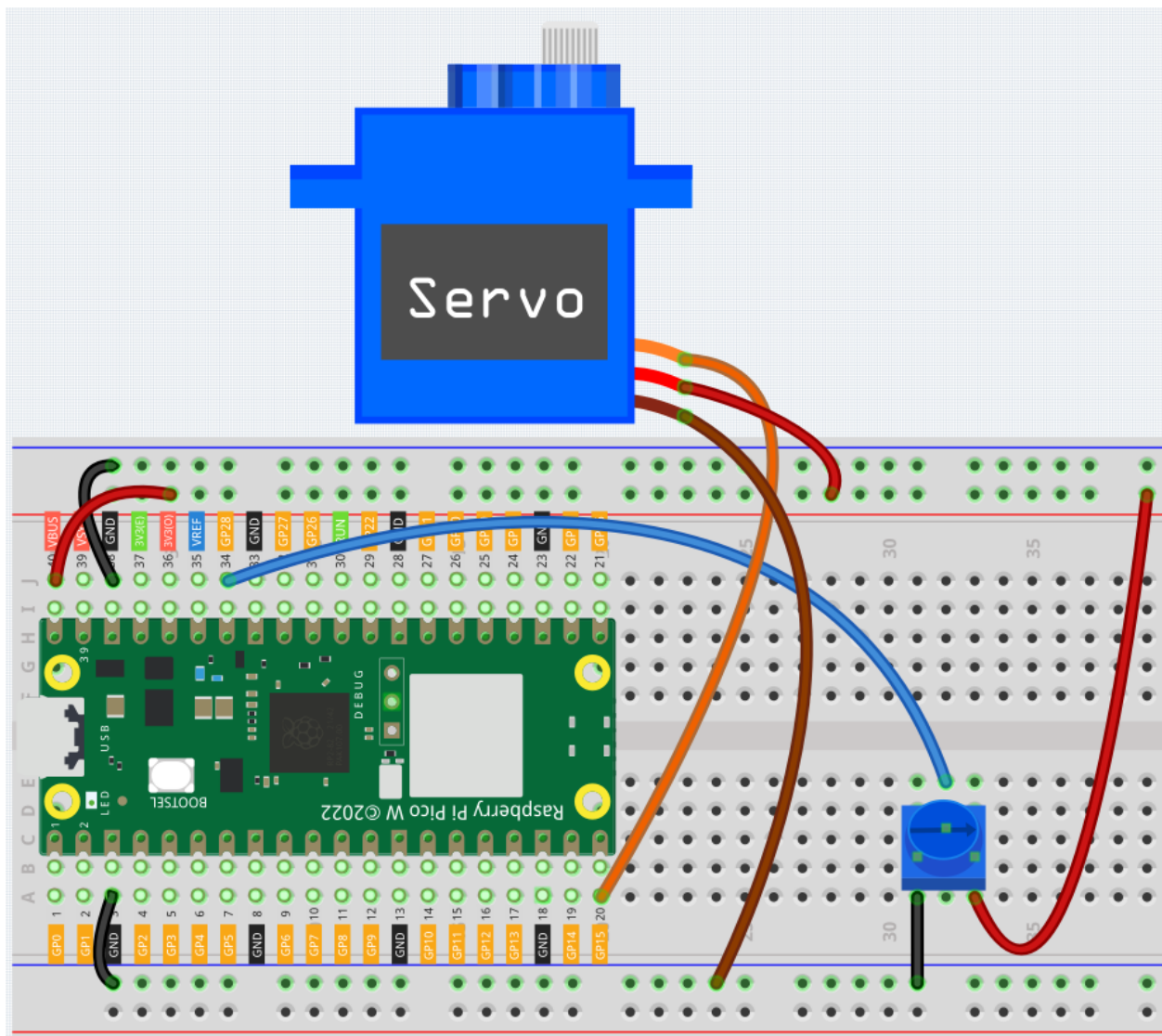
Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	KOMPONENTEN IN DIESEM SET	LINK
Kepler Kit	450+	

Alternativ können Sie die Komponenten auch einzeln über die untenstehenden Links kaufen.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Servo</i>	1	
6	<i>Potentiometer</i>	1	

## Verkabelung

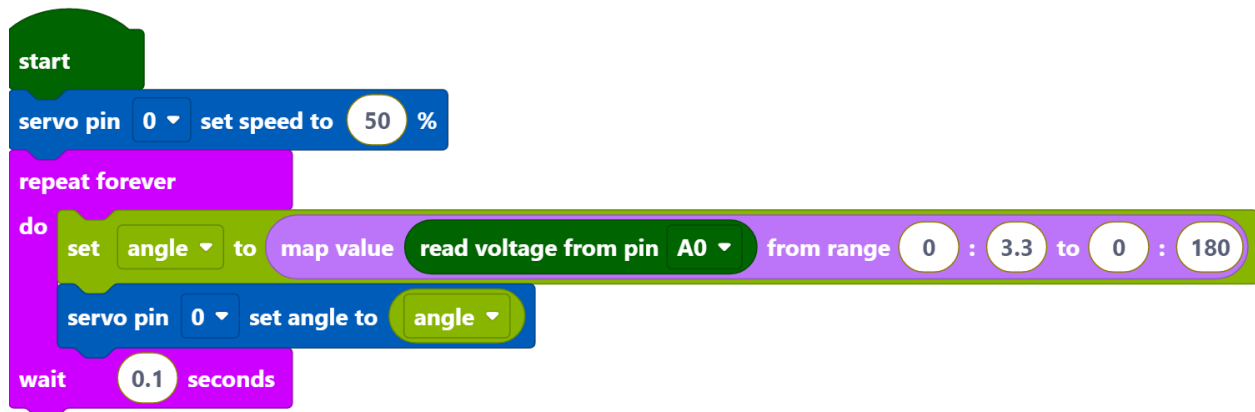


- Das orangefarbene Kabel (Signal) des Servos ist mit GP15 verbunden, das rote Kabel (Strom) mit VBUS und das braune Kabel (Masse) mit GND.
- Das Potentiometer ist ein Widerstandselement mit 3 Anschlüssen: Die beiden äußeren Pins sind mit 5V und GND verbunden, der mittlere Pin mit GP26(A0).

### Code

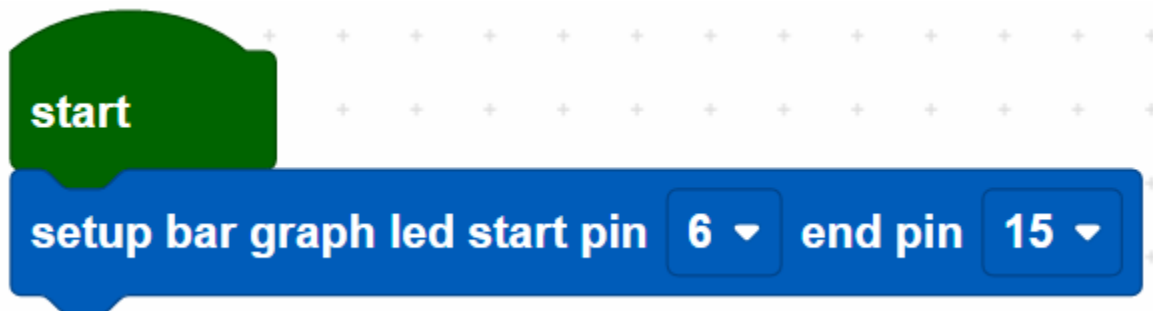
#### Bemerkung:

- Sie können sich an der untenstehenden Abbildung orientieren, um den Code per Drag-and-Drop zu erstellen.
- Importieren Sie `2.7_swing_servo.png` aus dem Pfad `kepler-kit-main\piper`. Detaillierte Anleitungen finden Sie unter [Code importieren](#).



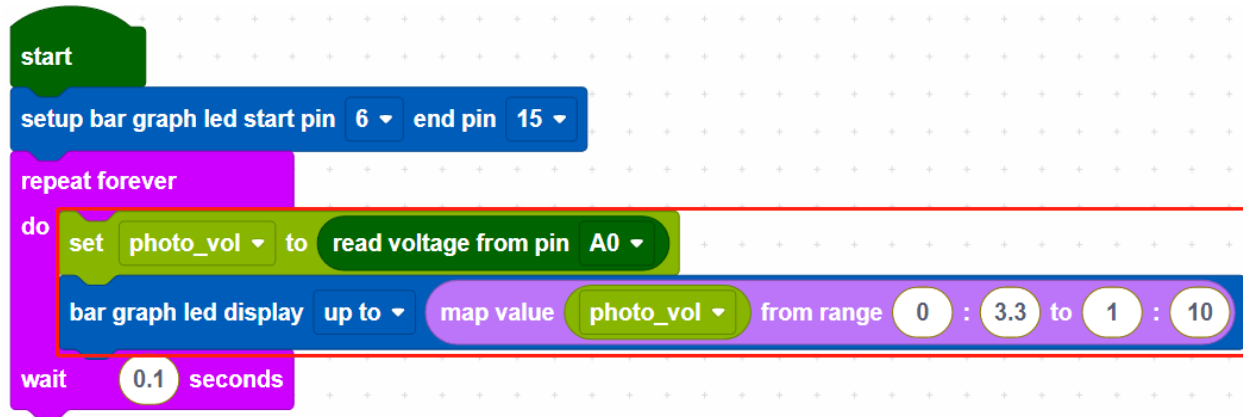
- Nach dem Anschluss des Pico W klicken Sie auf die **Start**-Taste und der Code wird ausgeführt.
- Drehen Sie das Potentiometer und das Servo folgt. Um es deutlich zu sehen, können Sie einen Steuerknüppel in die Servo-Welle einsetzen.

#### Funktionsweise



Stellen Sie die Rotationsgeschwindigkeit von Pin 15 (Servo) auf 15% ein.

- `[servo pin() set speed to ()%]`: Dient zur Einstellung der Rotationsgeschwindigkeit des Servo-Pins. Der Bereich liegt zwischen 0% und 100%.



Erstellen Sie eine Variable [Winkel], lesen Sie dann die Spannung von A0. Verwenden Sie den Block [Wert () von () bis () umrechnen], um die Spannung von A0 im Bereich von 0 bis 3,3V auf einen Winkelbereich von 0 bis 180° umzurechnen. Nutzen Sie den umgerechneten Winkel als Rotationswinkel des Servos.

- [Wert () von () bis () umrechnen]: Ein Wert wird von einem Bereich in einen anderen umgerechnet.

**Bemerkung:** Die Spannung von A0~A2 liegt im Bereich von 0~3,3V, selbst wenn Ihre Stromversorgung an VBUS (5V) angeschlossen ist.

## 7.11 2.8 Lichtintensitätsanzeige

In diesem Projekt verwenden wir einen Fotowiderstand und die LED-Balkenanzeige, um eine Lichtintensitätsanzeige zu erstellen. Je stärker das Licht, desto mehr LEDs der Balkenanzeige leuchten auf.

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

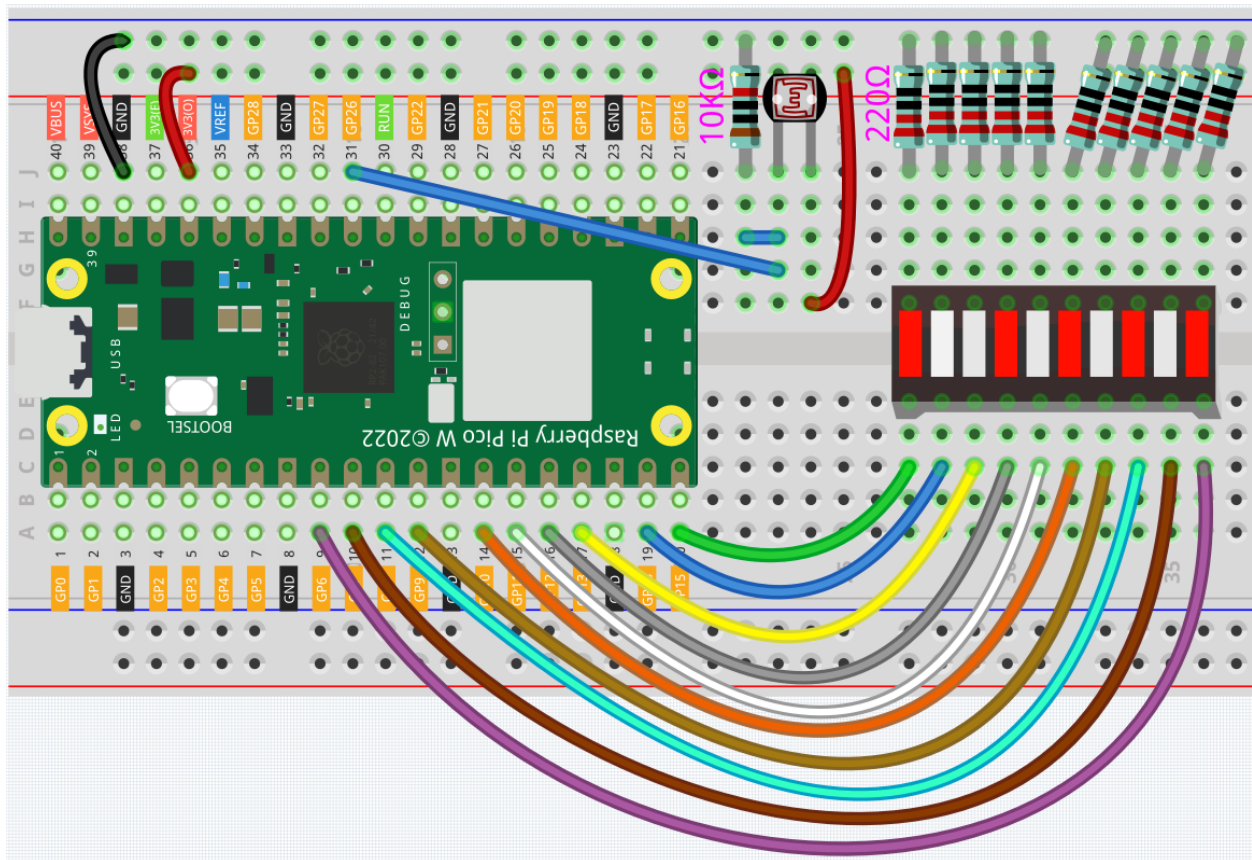
Ein komplettes Set zu kaufen, ist definitiv praktisch. Hier ist der Link:

Name	KOMPONENTEN IN DIESEM SET	LINK
Kepler Kit	450+	

Alternativ können Sie die Komponenten auch einzeln über die untenstehenden Links kaufen.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	11(10-220, 1-10K)	
6	<i>LED-Balkendiagramm</i>	1	
7	<i>Fotowiderstand</i>	1	

## Verkabelung

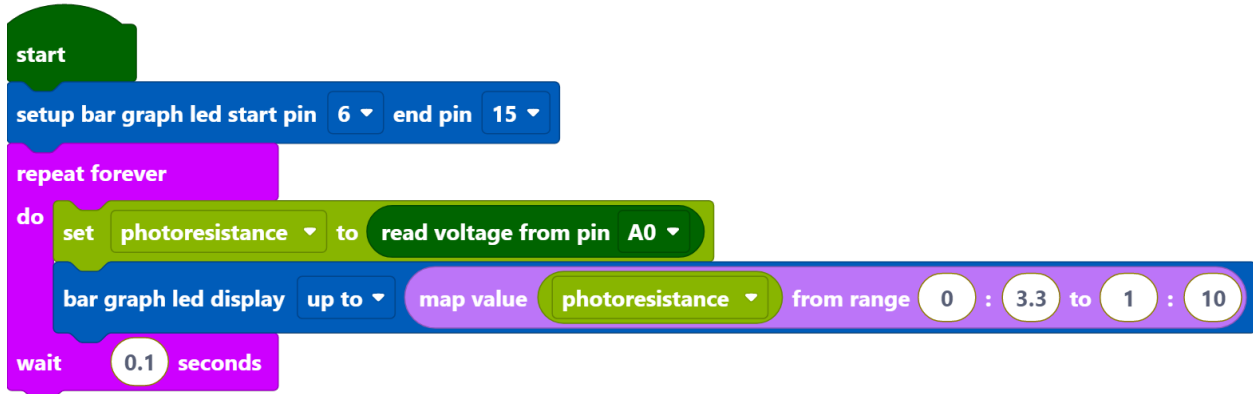


- Die LED-Balkenanzeige besteht aus 10 LEDs. Die Seite mit dem Etikett ist die Anode und die gegenüberliegende Seite die Kathode.
- Die Anoden der LED-Balkenanzeige sind mit GP6~GP15 verbunden. Die Kathoden sind über einen 220-Ohm-Widerstand mit GND verbunden.
- Verbinden Sie ein Ende des Fotowiderstands mit 3,3V und das andere Ende mit GP26 (A0). Gleichzeitig muss GP26 über einen weiteren 10K-Widerstand mit GND verbunden sein. So sinkt der Widerstand des Fotowiderstands bei stärkerem Licht, und die Spannung an A0 steigt.

## Code

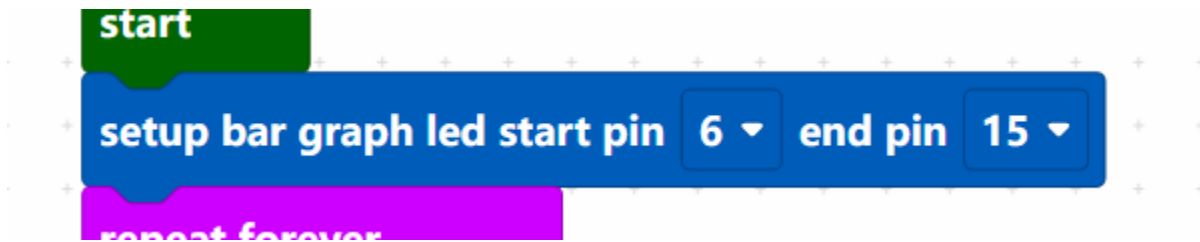
## Bemerkung:

- Sie können sich an der untenstehenden Abbildung orientieren, um den Code per Drag-and-Drop zu erstellen.
- Importieren Sie `2.8_light_intensity_display.png` aus dem Pfad `kepler-kit-main\piper`. Detaillierte Anleitungen finden Sie unter [Code importieren](#).

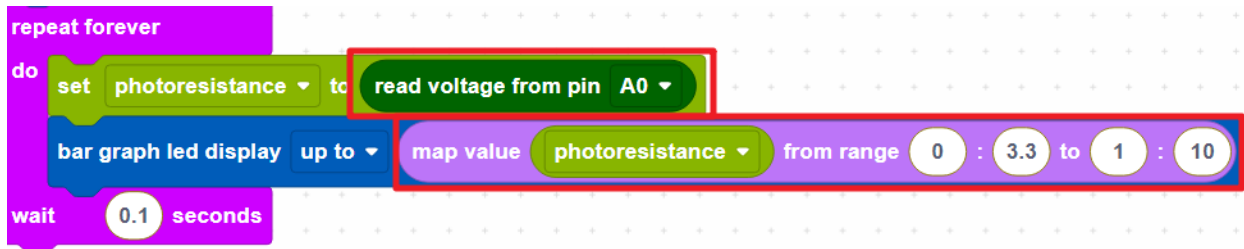


- Nach dem Anschluss des Pico W klicken Sie auf die **Start**-Taste und der Code wird ausgeführt.
- Bei stärkerem Licht leuchten mehr LEDs auf der LED-Balkenanzeige auf.
- Sollte die LED-Balkenanzeige nach dem Ausführen des Codes nicht richtig leuchten, können Sie versuchen, sie umzudrehen.

## Funktionsweise



Die Pins der LED-Balkenanzeige, die mit GP6 ~ GP15 verbunden sind, werden festgelegt.



Speichern Sie den Spannungswert von A0 (GP26) in der Variable [photo\_vol]. Verwenden Sie den Block [Wert () von () bis () umrechnen], um die Variable [photo\_vol] im Bereich von 0 bis 3,3V auf 0 bis 10 (die Anzahl der LEDs auf der LED-Balkenanzeige) abzubilden.

- [Wert () von () bis () umrechnen]: Ein Wert wird von einem Bereich in einen anderen umgerechnet.

## 7.12 2.9 Glückskatze

In diesem Projekt verwenden wir ein PIR-Modul und ein Servo, um eine Glückskatze zu bauen. Das PIR-Modul dient zur Erkennung von Besuchern, während das Servo die winkende Bewegung der Glückskatze nachahmt.

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

Ein komplettes Set zu kaufen ist definitiv praktisch. Hier ist der Link:

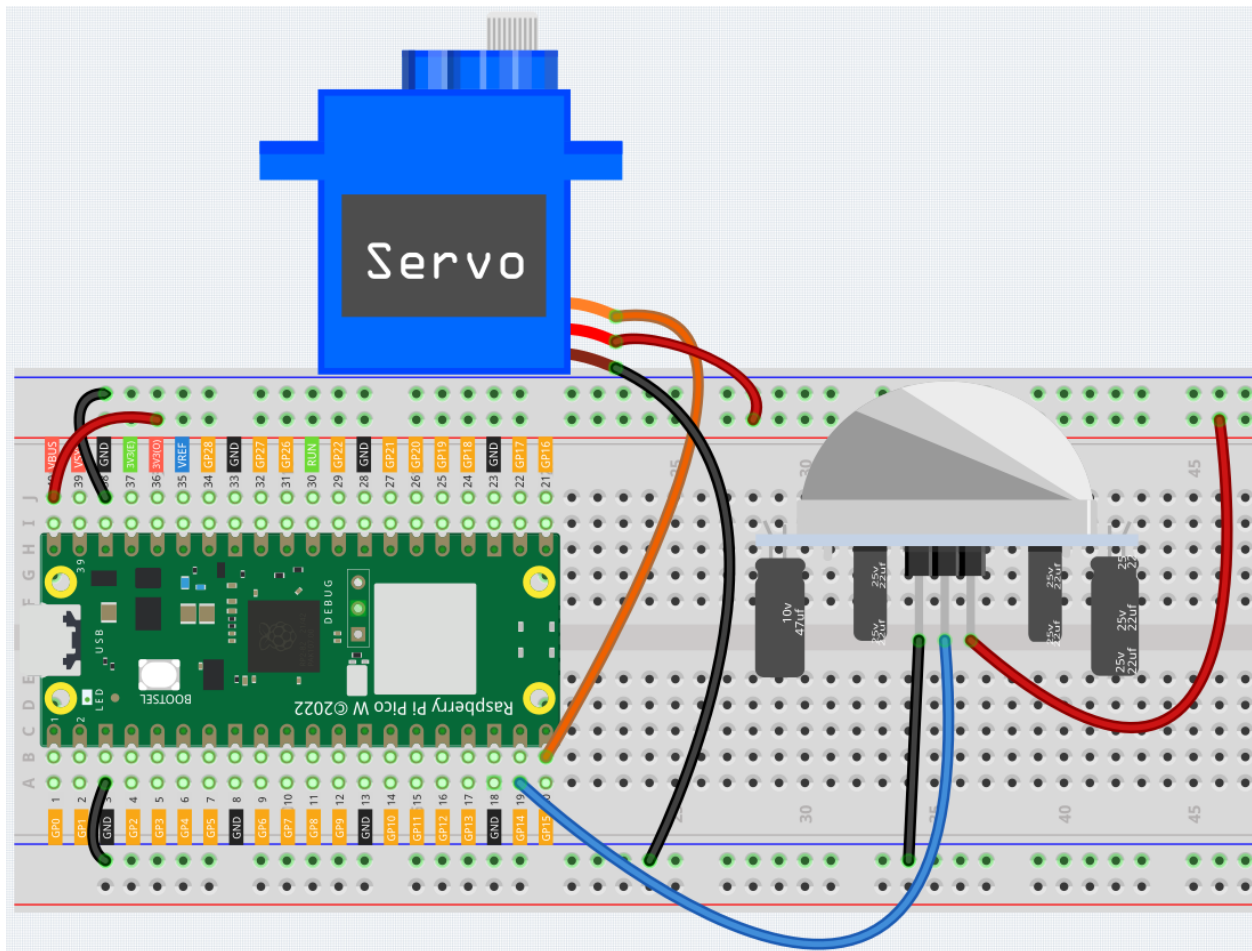
Name	KOMPONENTEN IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die Komponenten auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Servo</i>	1	
6	<i>PIR-Bewegungssensormodul</i>	1	

### Verkabelung



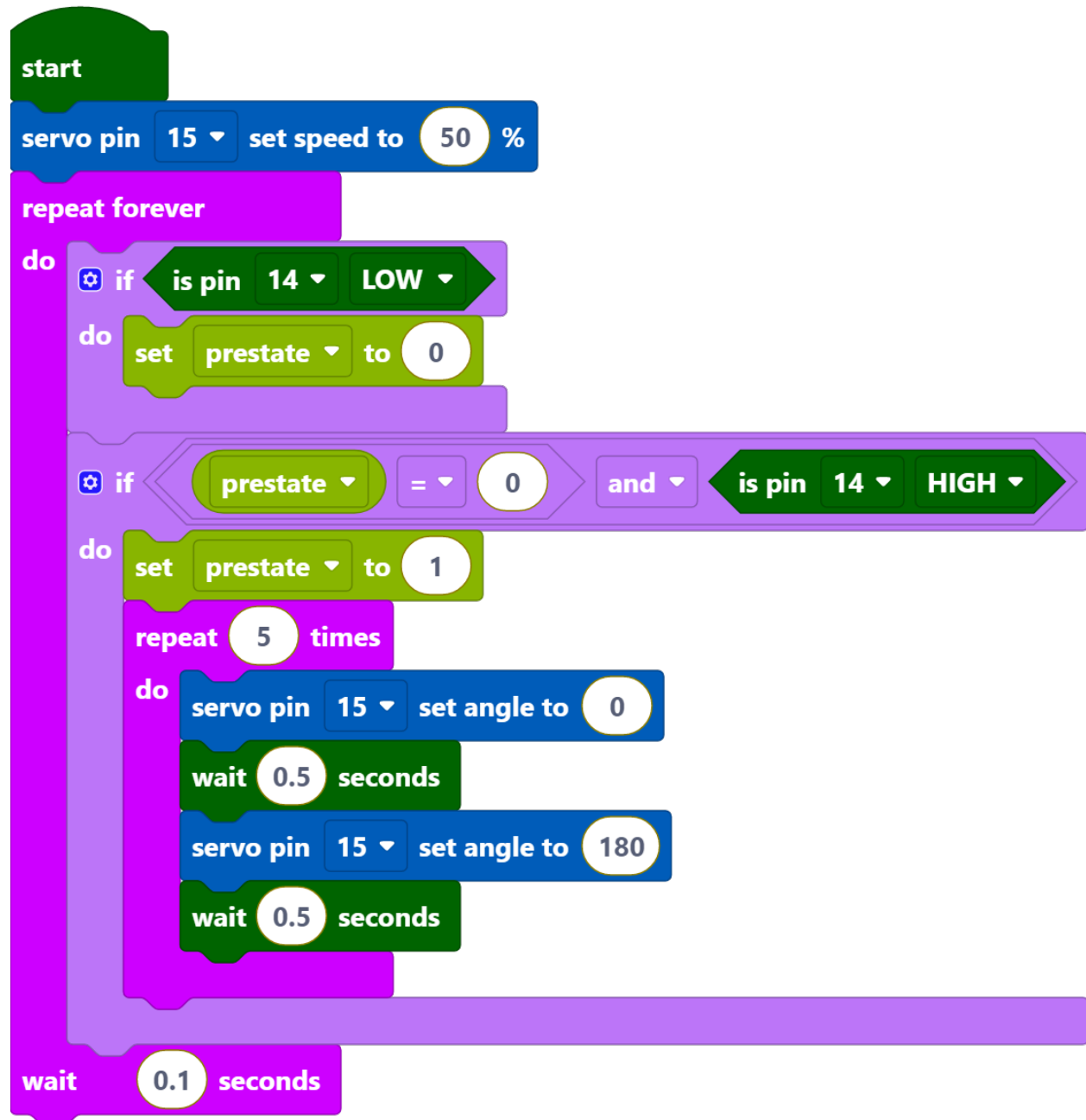


- Das orangefarbene Kabel (Signal) des Servos ist mit GP15 verbunden, das rote Kabel (Stromversorgung) mit VBUS und das braune Kabel (Masse) mit GND.
- Der mittlere Pin des PIR-Moduls ist mit GP3 verbunden.

### Code

#### Bemerkung:

- Sie können sich an der Abbildung unten orientieren, um den Code per Drag-and-Drop zu schreiben.
- Importieren Sie `2.9_lucky_cat.png` aus dem Verzeichnis `kepler-kit-main\piper`. Detaillierte Anleitungen finden Sie unter [Code importieren](#).

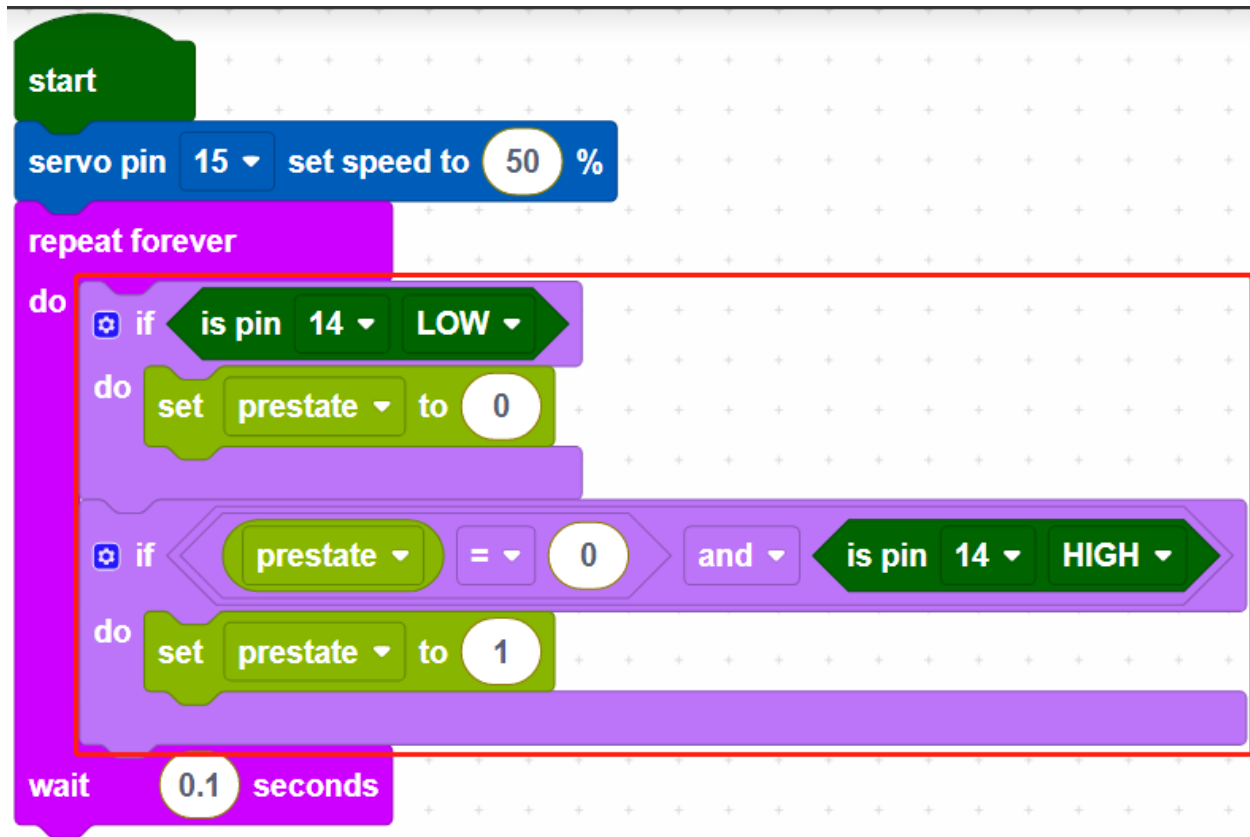


- Nach dem Anschluss des Pico W klicken Sie auf die **Start**-Taste, und der Code wird ausgeführt.
- Wenn das PIR-Modul einen Kunden erkennt, schwingt das Servo fünfmal hin und her und bleibt dann stehen.

So funktioniert es

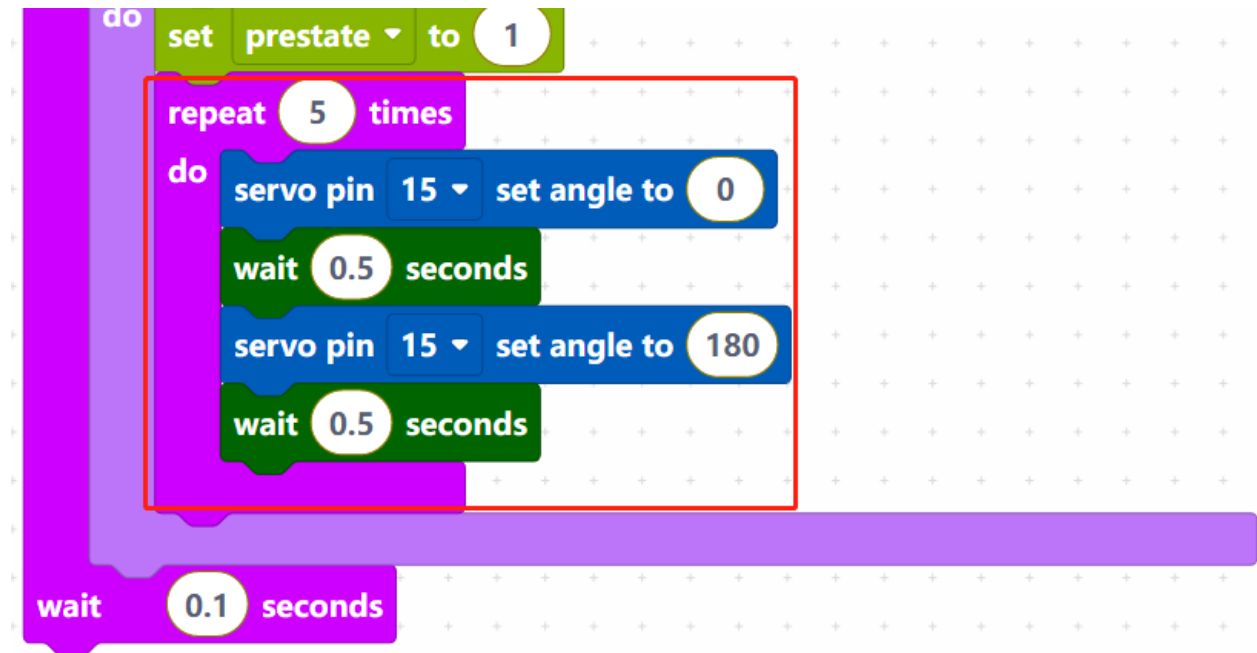


Stellen Sie die Drehgeschwindigkeit des Pins15 (Servo) auf 15% ein.



Wenn GP14 niedrig ist, setzen Sie die Variable [prestate] auf 0. Wenn die Variable [prestate] 0 ist und GP14 hoch ist (Mensch erkannt), setzen Sie die Variable [prestate] auf 1.

Ziel ist es, den Hauptcode nur auszuführen, wenn GP14 von niedrig auf hoch wechselt, und nur einmal zu reagieren, wenn das PIR-Modul weiterhin Menschen erkennt.



Lassen Sie das Servo 5-mal zwischen 0 und 180 Grad rotieren.

- [repeat () times do]: Führen Sie den Code im Do-Block eine bestimmte Anzahl von Malen aus.

## 7.13 2.10 Fließende LEDs

Das Kit ist mit einem WS2812 RGB LED-Streifen ausgestattet, der bunte Farben anzeigen kann, und jede LED kann unabhängig gesteuert werden.

In diesem Projekt verwenden wir den Neigungsschalter, um die Fließrichtung der LEDs auf dem WS2812 RGB LED-Streifen zu steuern.

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

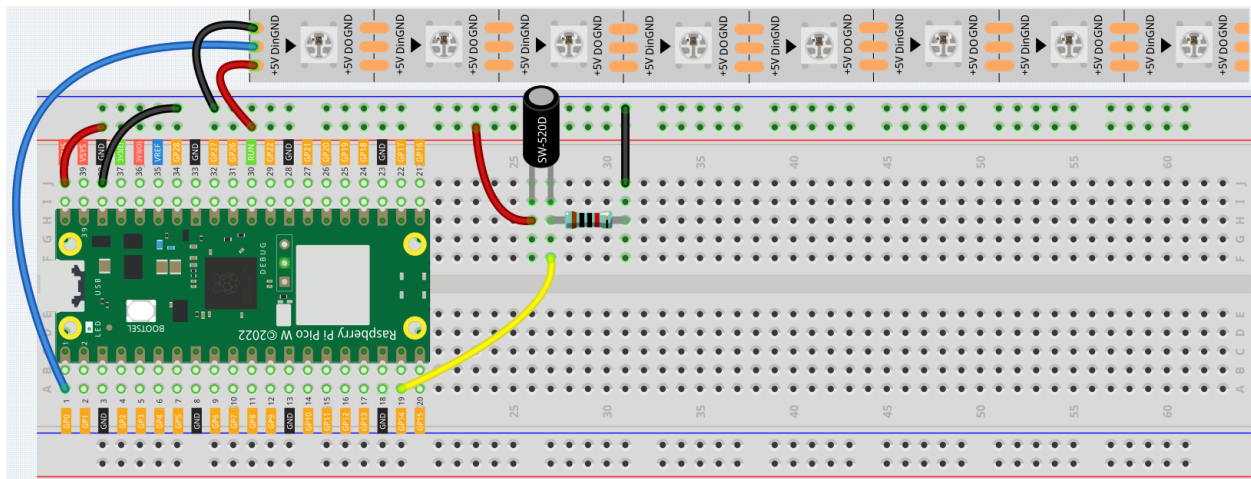
Ein komplettes Set zu kaufen ist definitiv praktisch. Hier ist der Link:

Name	KOMPONENTEN IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die Komponenten auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Widerstand</i>	1(10K)	
6	<i>Neigungsschalter</i>	1	
7	<i>WS2812 RGB 8-LED-Streifen</i>	1	

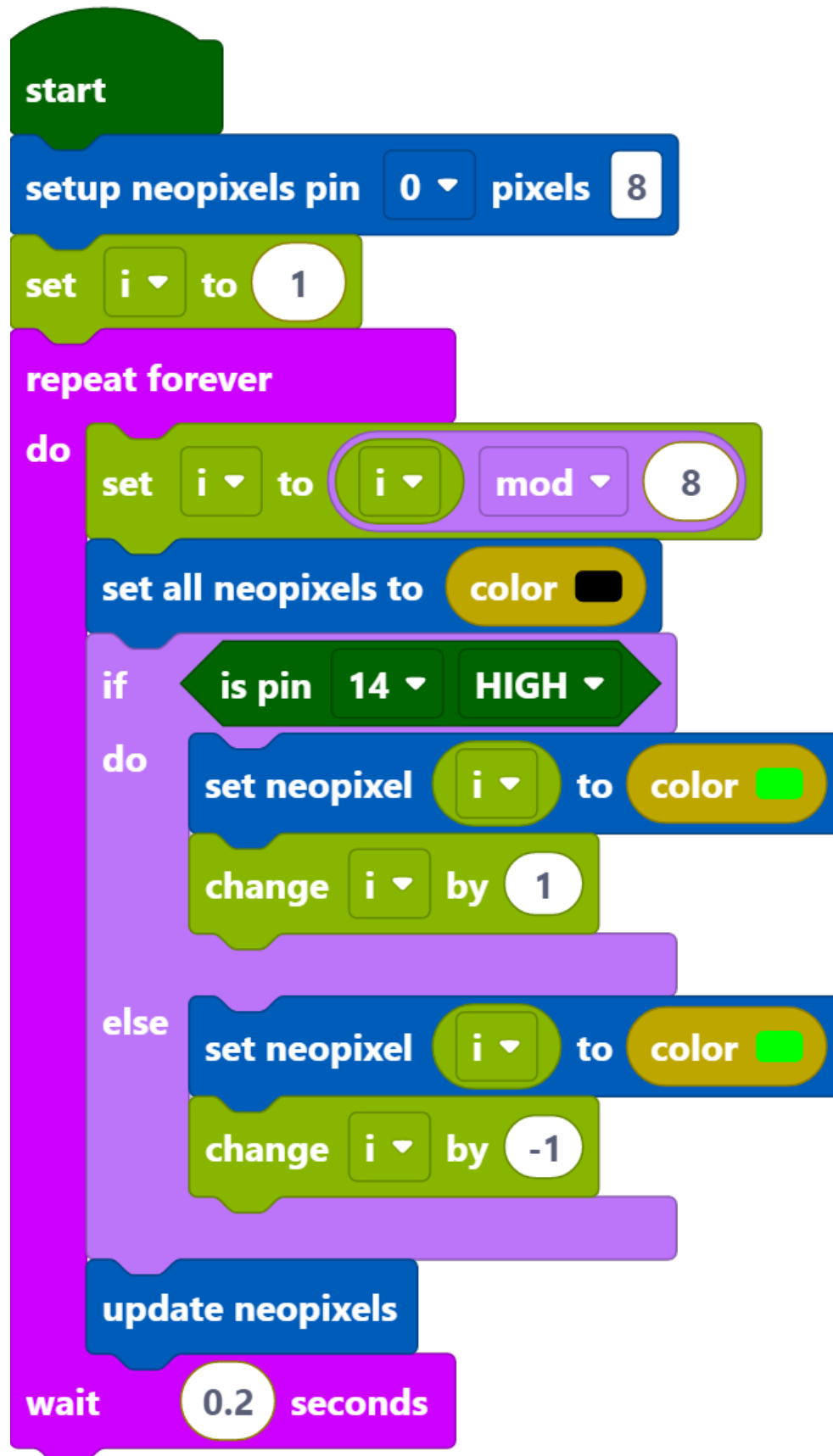
### Verkabelung



### Code

#### Bemerkung:

- Sie können sich an der Abbildung unten orientieren, um den Code per Drag-and-Drop zu schreiben.
- Importieren Sie `2.10_floating_led.png` aus dem Verzeichnis `kepler-kit-main\piper`. Detaillierte Anleitungen finden Sie unter [Code importieren](#).

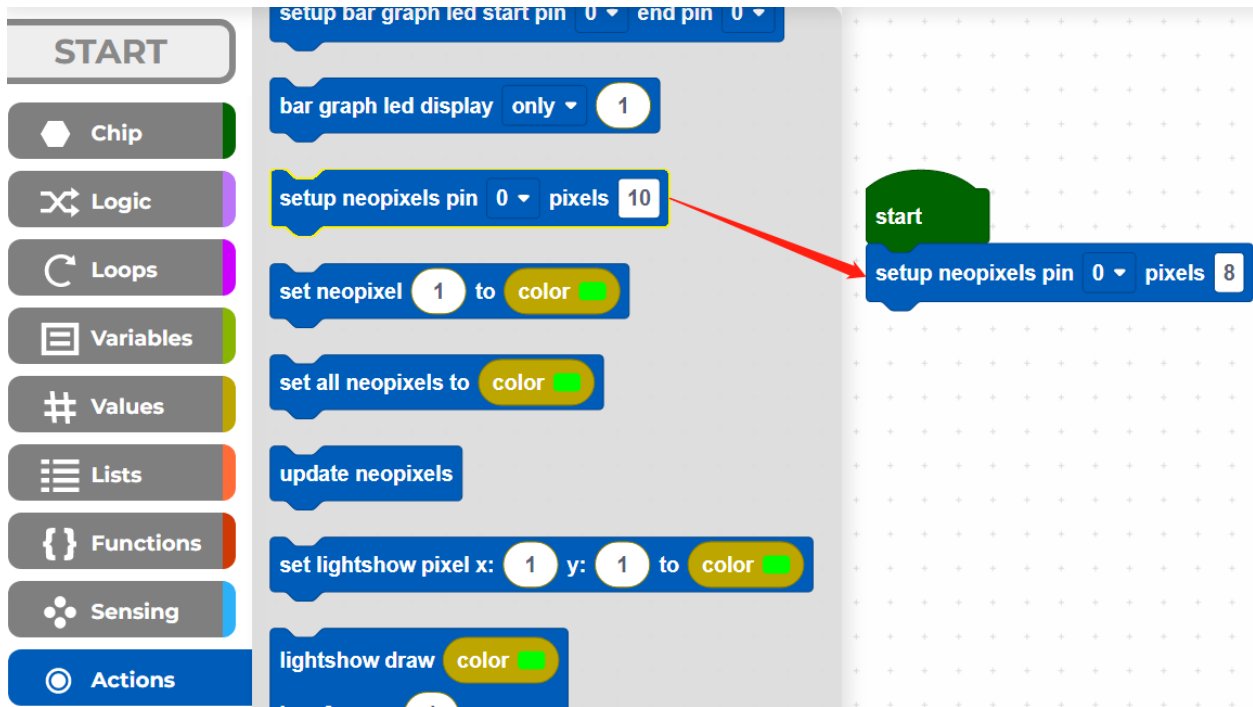


Nach dem Anschluss des Pico W klicken Sie auf die **Start**-Taste, und der Code wird ausgeführt.

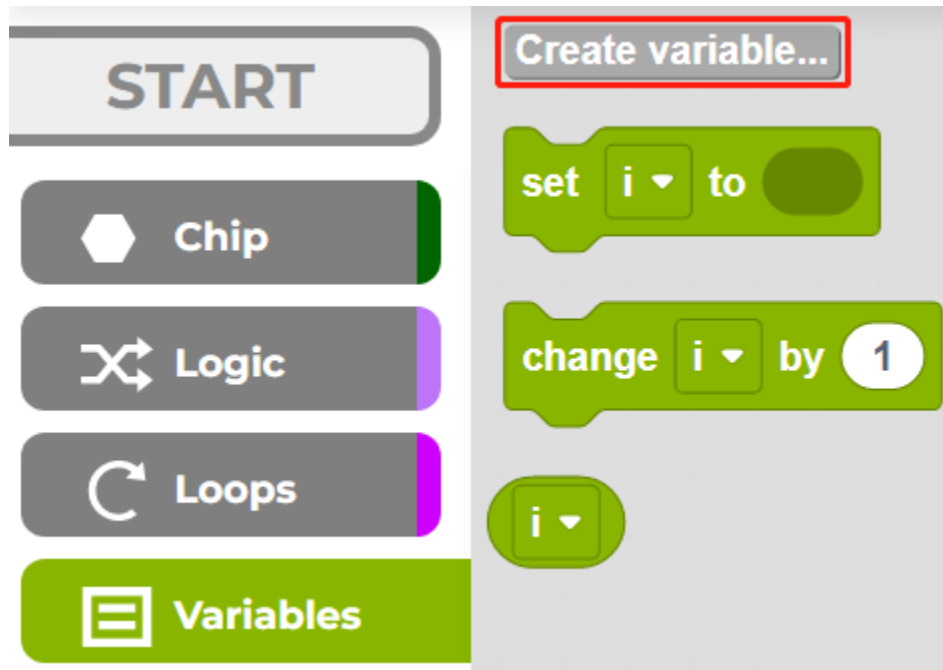
Wenn der Neigungsschalter vertikal positioniert ist, leuchten die LEDs auf dem WS2812 RGB LED-Streifen nacheinander in Grün auf. Wenn der Neigungsschalter horizontal positioniert ist, leuchten die LEDs in umgekehrter Richtung in Grün auf.

### Programmierung

**Schritt 1:** Verwenden Sie den [setup neopixel pin() pixels()] Block aus der **Actions**-Palette, um den WS2812 RGB LED-Streifen zu initialisieren. **0** bedeutet, dass der angeschlossene Pin GP0 ist und **8** bedeutet, dass 8 RGB-LEDs auf dem WS2812 RGB LED-Streifen vorhanden sind.

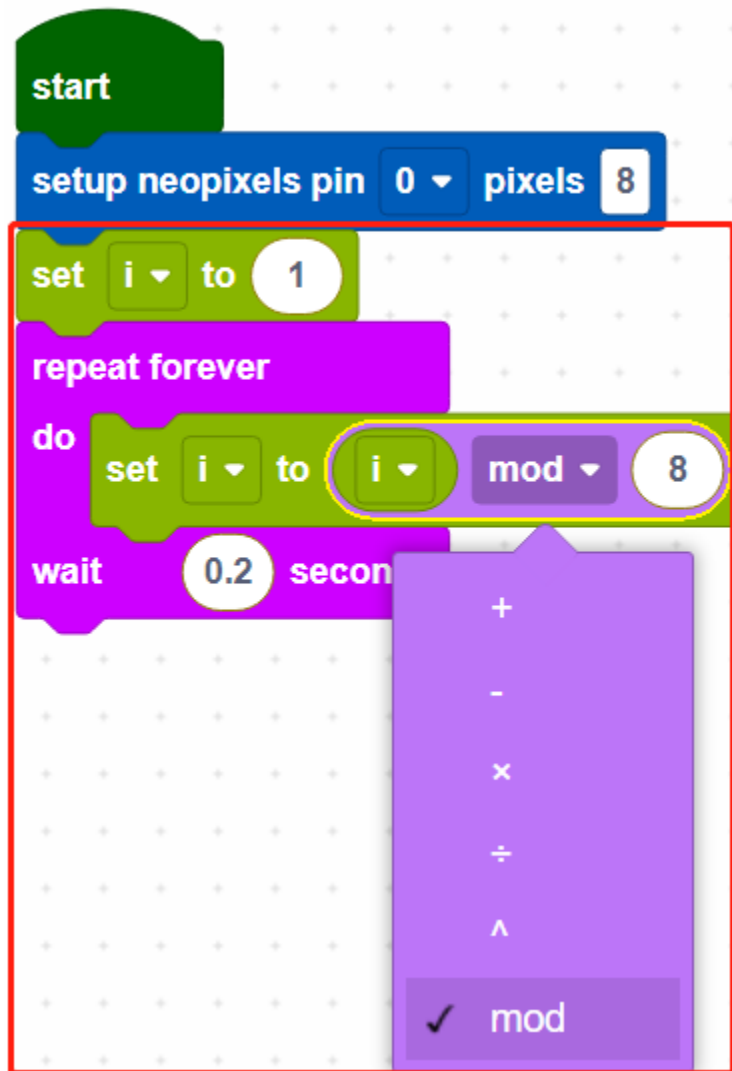


**Schritt 2:** Klicken Sie in der **Variables**-Palette auf den **Create variable**-Button, um eine Variable namens **i** zu erstellen, die die LEDs auf dem WS2812 RGB LED-Streifen repräsentiert.



**Schritt 3:** Setzen Sie den Anfangswert der Variable **i** auf 1 (die LED in der Nähe der Kabel), verwenden Sie dann im [repeat forever]-Block  $[\text{() mod } ()]$ , um den Wert von **i** von 0 bis 7 festzulegen.

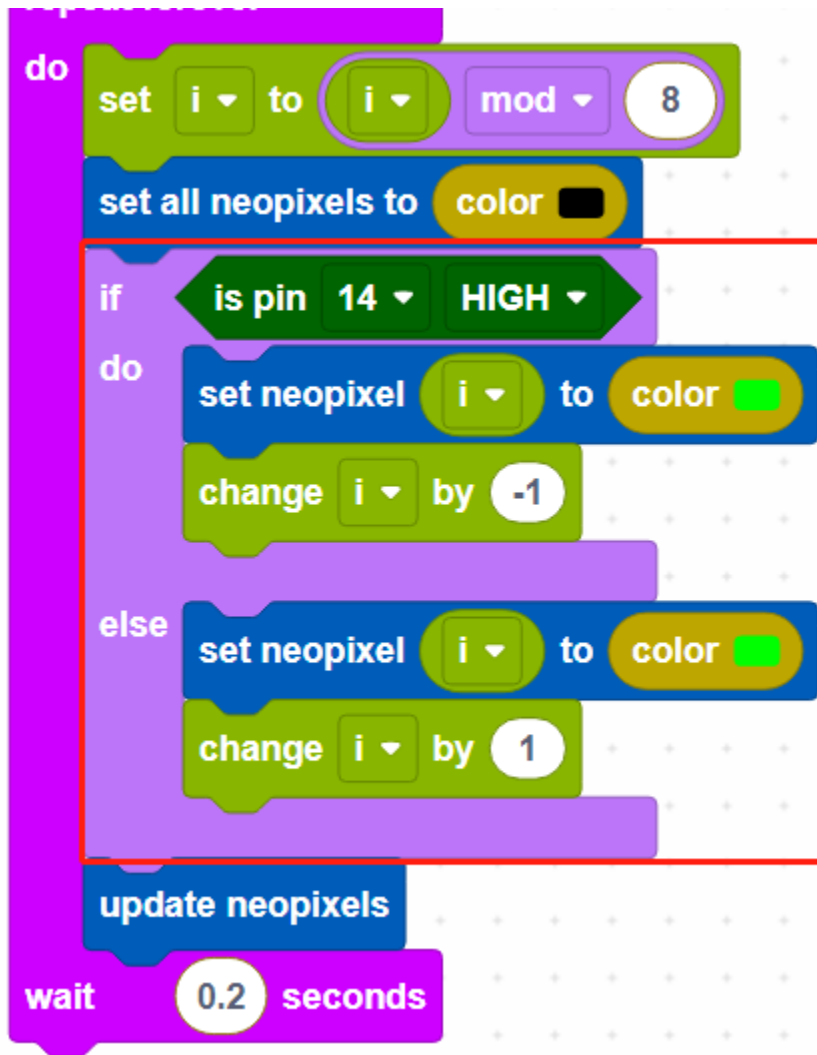




**Schritt 4:** Setzen Sie alle Neopixel auf Schwarz, um alle LEDs auszuschalten, und verwenden Sie dann [updates neopixels], um diesen Effekt auf den WS2812 RGB LED-Streifen zu übertragen.



**Schritt 5:** Wenn Pin14 hoch gelesen wird, lassen Sie die LEDs auf dem WS2812 RGB LED-Streifen nacheinander in Grün aufleuchten, ansonsten leuchten sie in umgekehrter Reihenfolge in Grün auf.



- [change () by ()]: Wird verwendet, um den Wert einer Variable um einen bestimmten Schritt zu erhöhen (positiv) oder zu verringern (negativ).

## 7.14 2.11 Rückfahrssystem

Für dieses Projekt haben wir ein Ultraschallmodul und einen aktiven Summer verwendet, um ein Rückfahralarmsystem zu erstellen. Das Ultraschallmodul dient zur Entfernungsmessung, und der Summer gibt je nach Entfernung unterschiedliche Alarmfrequenzen aus.

### Benötigte Komponenten

Für dieses Projekt benötigen wir die folgenden Komponenten.

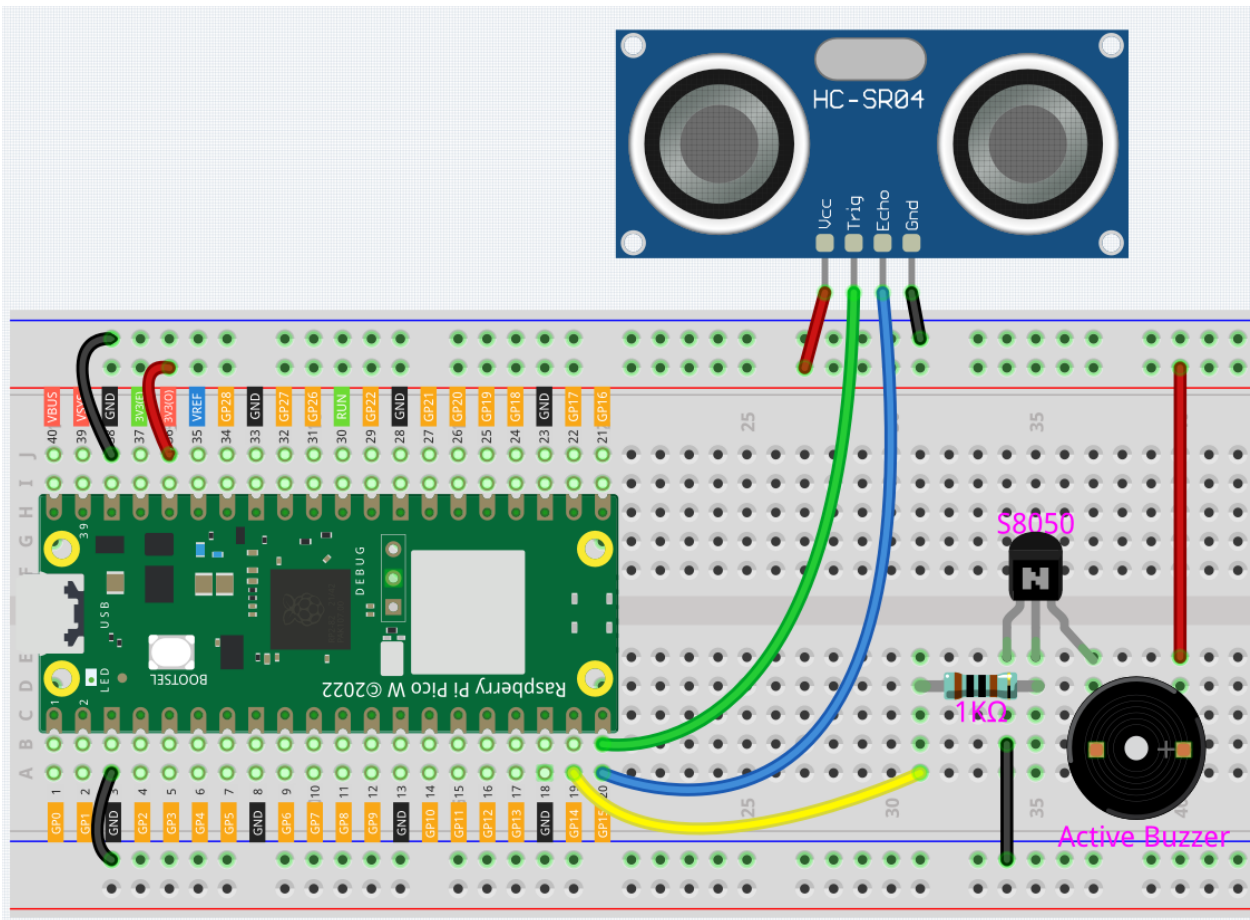
Ein komplettes Set zu kaufen ist definitiv praktisch. Hier ist der Link:

Name	KOMPONENTEN IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die Komponenten auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>Transistor</i>	1(S8050)	
6	<i>Widerstand</i>	1(1K)	
7	Aktiver <i>Summer</i>	1	
8	<i>Ultraschallmodul</i>	1	

Verkabelung



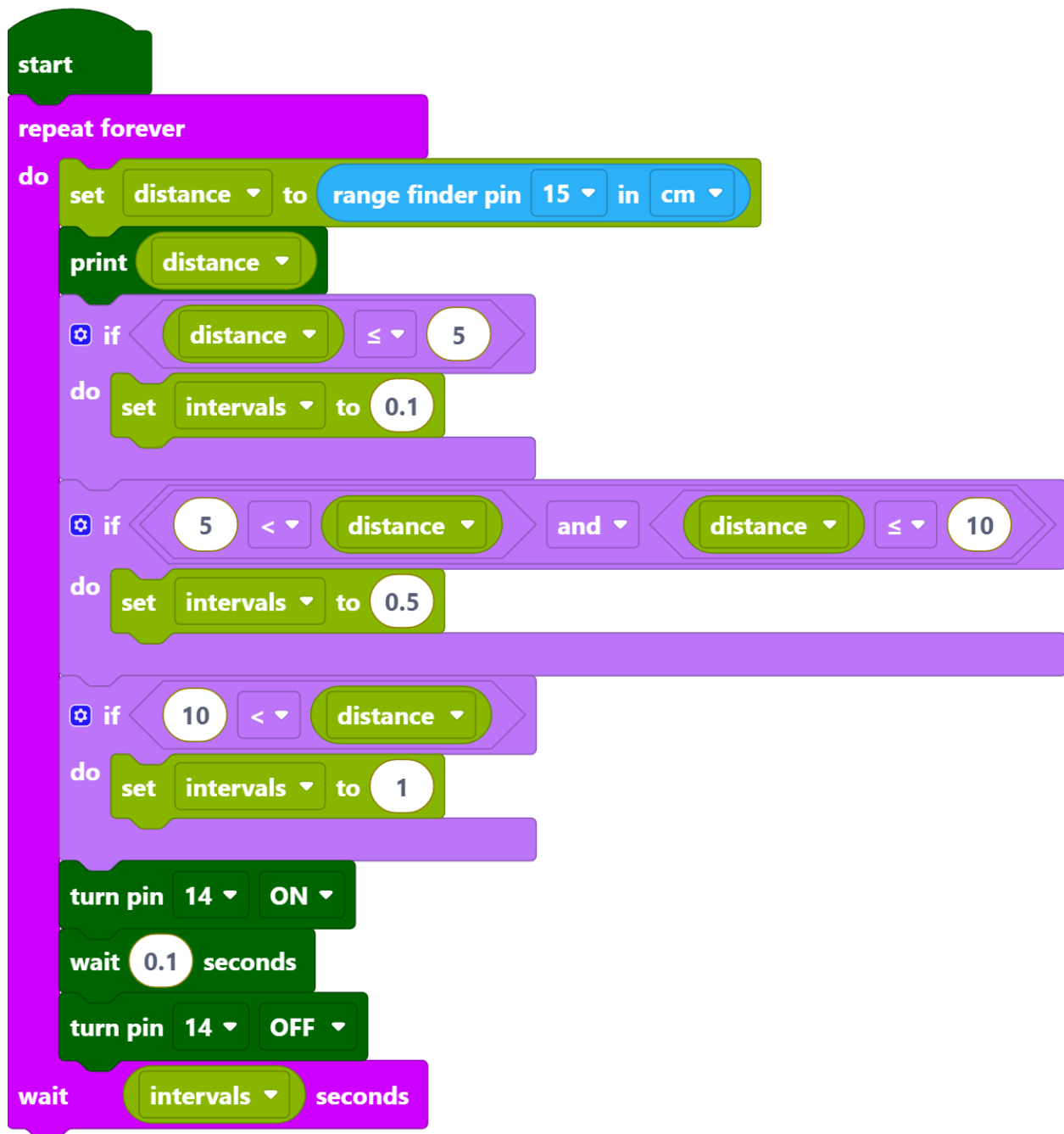
- Die Echo- und Trig-Pins des Ultraschallmoduls sind gleichzeitig mit GP15 verbunden, damit das Ultraschallmodul Signale von GP15 sendet und empfängt.
- Der mittlere Pin des Transistors, der mit dem Summer verbunden ist, ist über einen 1k-Widerstand mit GP14

verbunden.

## Code

### Bemerkung:

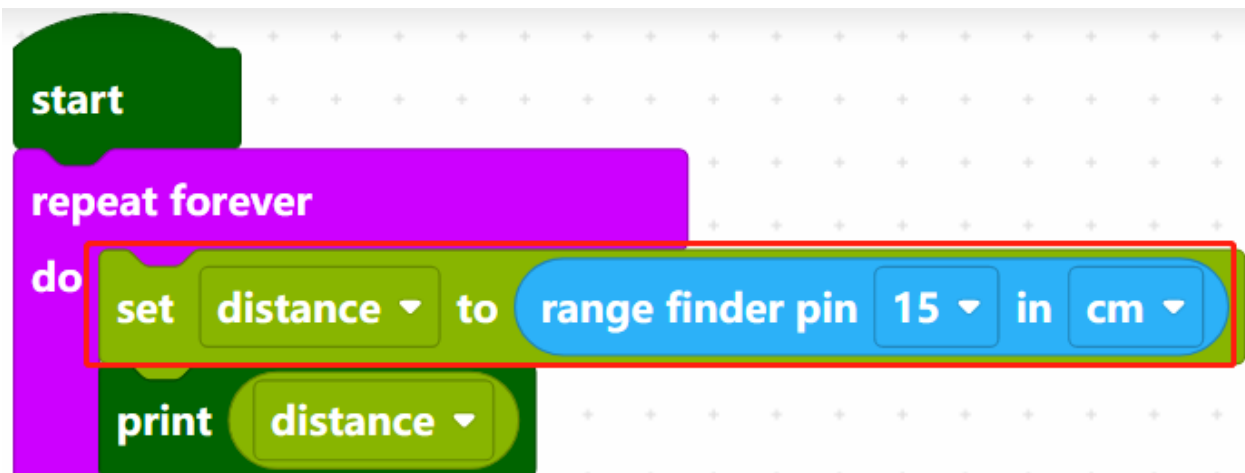
- Sie können sich an der Abbildung unten orientieren, um den Code per Drag-and-Drop zu schreiben.
- Importieren Sie `2.11_reversing_system.png` aus dem Verzeichnis `kepler-kit-main\piper`. Detaillierte Anleitungen finden Sie unter [Code importieren](#).



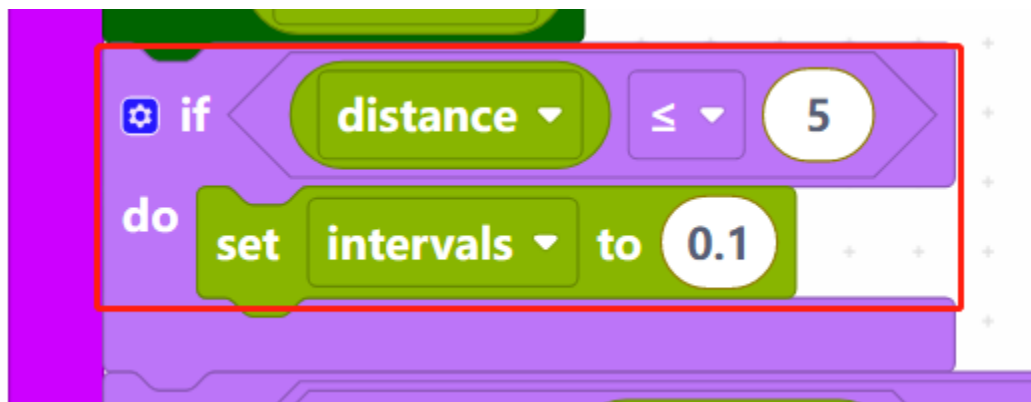
- Nach dem Anschluss des Pico W klicken Sie auf die **Start**-Taste, und der Code wird ausgeführt.

- Wenn die Ultraschallerkennung eine Entfernung von weniger als 5 cm misst, gibt der Summer einen scharfen Ton (0,1 s) aus.
- Wenn die erkannte Entfernung zwischen 5~10 cm liegt, gibt der Summer einen etwas langsameren Ton (0,5 s) aus.
- Wenn die erkannte Entfernung größer als 10 cm ist, erfolgt alle 1 Sekunde ein Tonsignal.

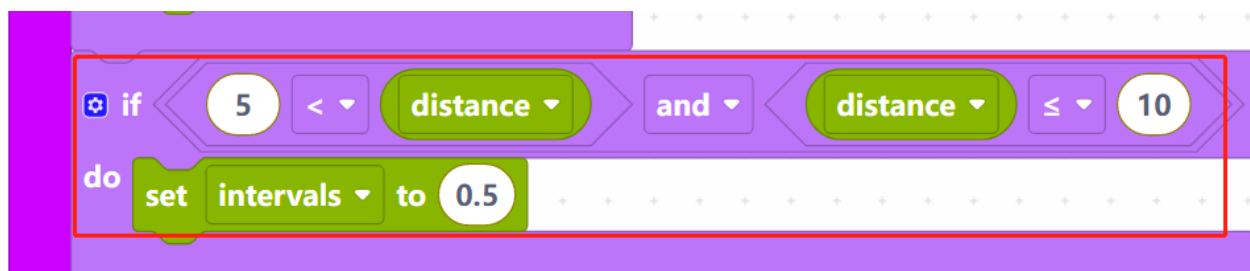
#### Funktionsweise



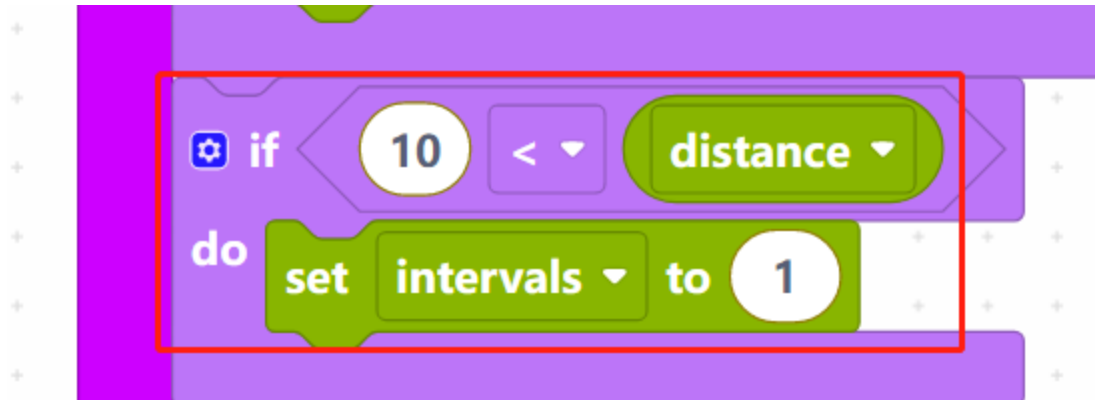
Lesen Sie die Entfernung (in cm) der Ultraschallerkennung und speichern Sie sie in der Variable [distance].



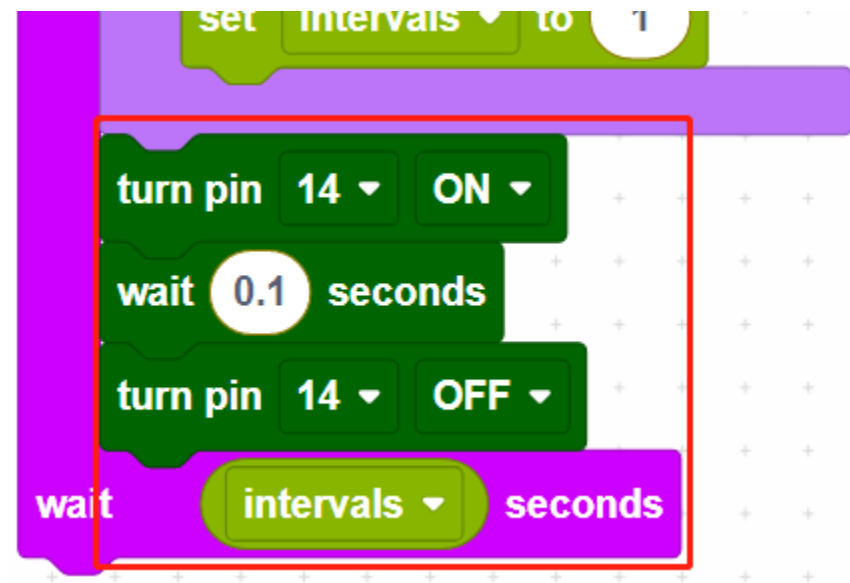
Wenn die Entfernung weniger als oder gleich 5 ist, setzen Sie die Variable [intervals] auf 0,1 s. Die Variable [intervals] ist das Intervall zwischen den Summerklängen.



Wenn die Entfernung größer als 5 und weniger als oder gleich 10 ist, setzen Sie [intervals] auf 0,5 s.



Wenn die Entfernung größer als 10 ist, setzen Sie die [intervals]-Zeit auf 1 s.



Lassen Sie den Summer schließlich alle [intervals] Sekunden ertönen.

## 7.15 2.12 Intelligenter Ventilator

In diesem Projekt erstellen wir einen temperaturgesteuerten intelligenten Ventilator mit Thermistor, TA6586, Motor und Spannungsmodul. Der Ventilator dreht sich automatisch, sobald die eingestellte Temperatur erreicht ist.

Für dieses Projekt benötigen wir die folgenden Komponenten.

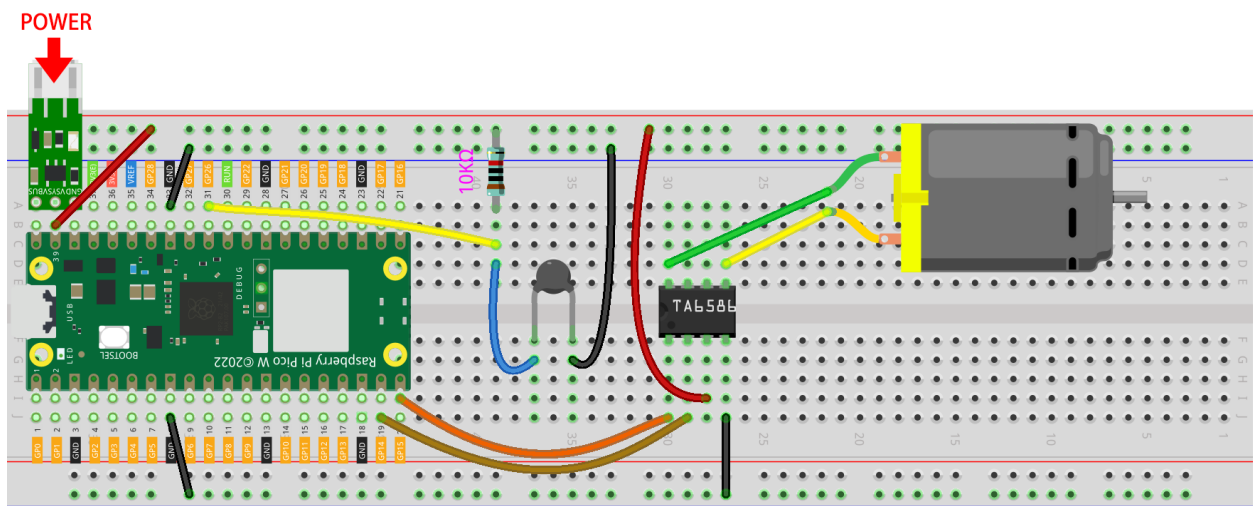
Es ist definitiv praktisch, ein komplettes Set zu kaufen. Hier ist der Link:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die Komponenten auch einzeln über die untenstehenden Links erwerben.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<i>Raspberry Pi Pico W</i>	1	
2	Micro-USB-Kabel	1	
3	<i>Steckbrett</i>	1	
4	<i>Jumperkabel</i>	Mehrere	
5	<i>TA6586 - Motorsteuerungs-Chip</i>	1	
6	<i>Gleichstrommotor</i>	1	
7	<i>Li-Po-Lademodul</i>	1	
8	18650 Batterie	1	
9	Batteriehalter	1	
10	<i>Widerstand</i>	1(10K)	
11	<i>Fotowiderstand</i>	1	

## Verkabelung

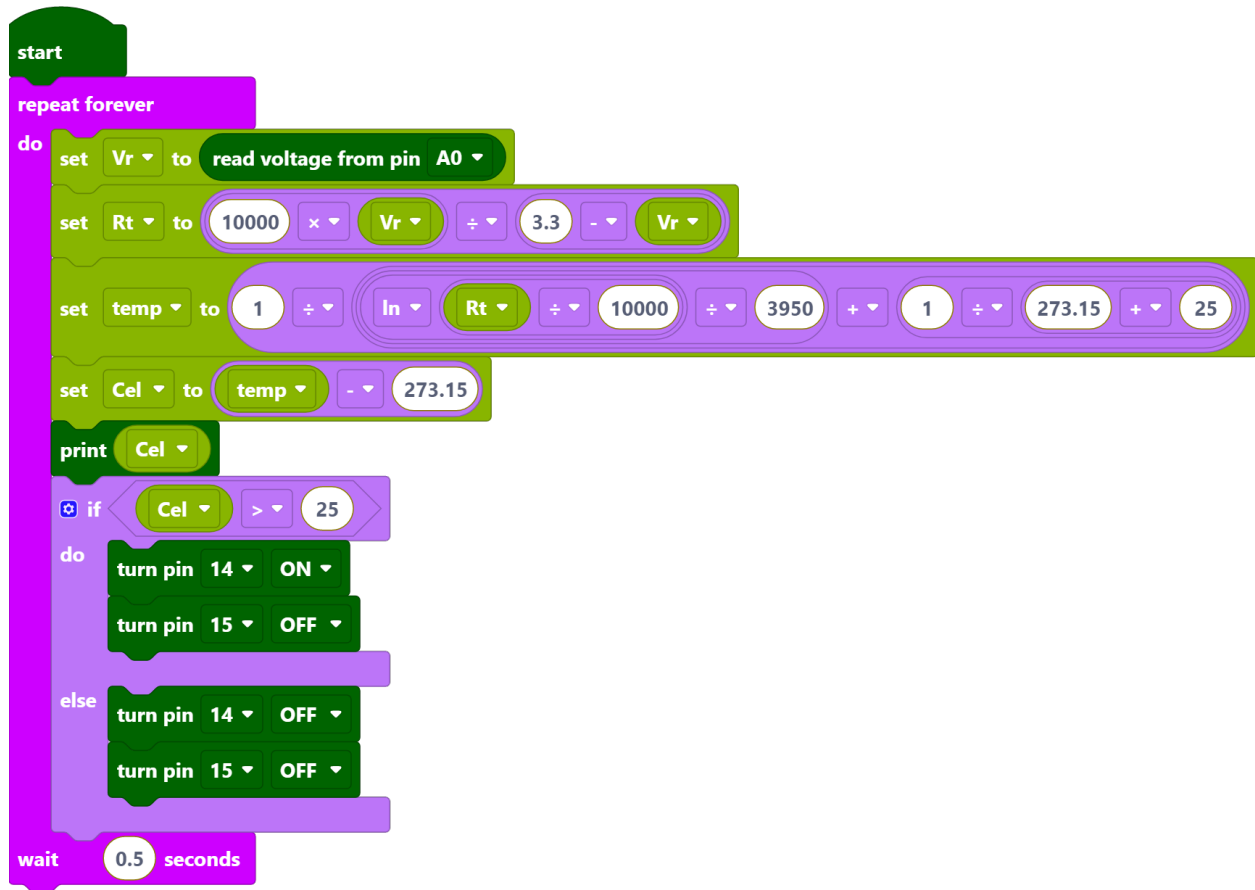


## Code

### Bemerkung:

- Sie können sich an der Abbildung unten orientieren, um den Code per Drag-and-Drop zu schreiben.
- Importieren Sie `2.12_smart_fan.png.png` aus dem Verzeichnis `kepler-kit-main\piper`. Detaillierte Anleitungen finden Sie unter [Code importieren](#).

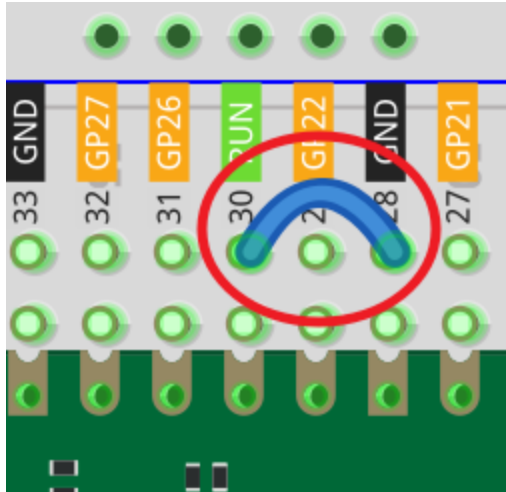




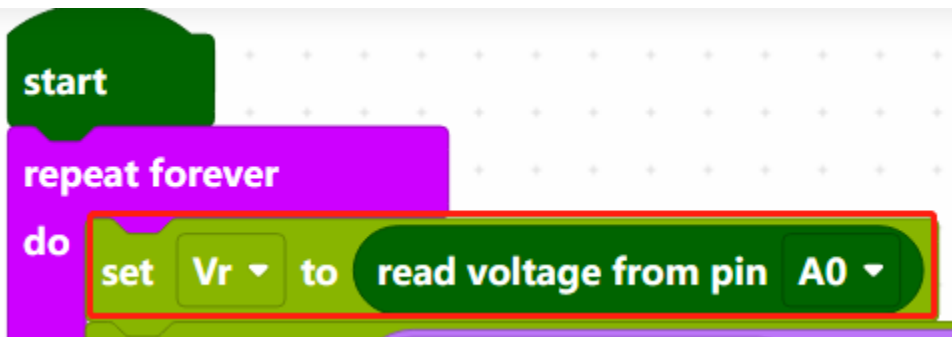
- Nach dem Anschluss des Pico W klicken Sie auf die **Start**-Taste, und der Code wird ausgeführt.
- Klicken Sie auf CONSLE, um die aktuelle Temperatur in Grad Celsius zu sehen.
- Der Ventilator beginnt sich zu drehen, wenn die Temperatur über 25 Grad steigt, und stoppt, wenn sie unter 25 Grad fällt.

#### Bemerkung:

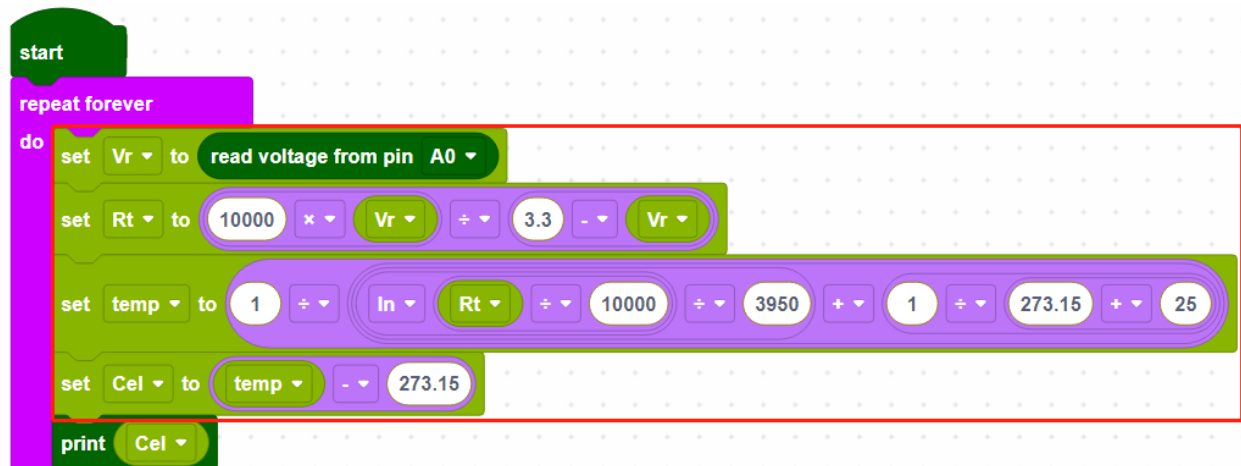
- Wenn der Motor sich nach dem Klicken auf die Stopptaste weiter dreht, müssen Sie den Run-Pin am Pico W zu diesem Zeitpunkt mit einem Draht auf GND zurücksetzen und dann den Draht wieder abziehen, um den Code erneut auszuführen.
- Dies liegt daran, dass der Motor mit zu hohem Strom arbeitet, was dazu führen kann, dass der Pico W die Verbindung zum Computer verliert.



### Funktionsweise



Die Spannung von A0 (GP26) wird gelesen und der Variablen [Vr] zugewiesen.



Diese Berechnungen wandeln die Werte des Thermistors in Grad Celsius um.

**Bemerkung:** Hier ist die Beziehung zwischen dem Widerstand und der Temperatur:

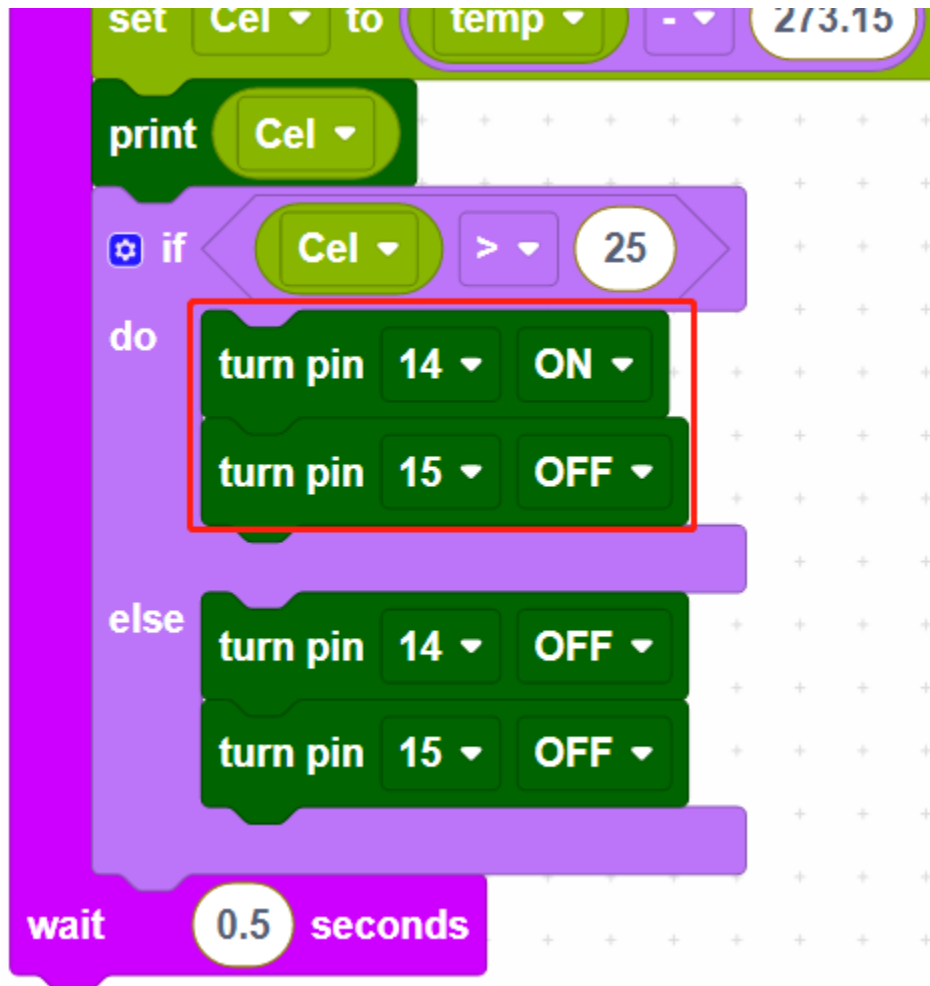
$$RT = RN \exp(B(1/TK - 1/TN))$$

- RT ist der Widerstand des NTC-Thermistors bei der Temperatur TK.

- RN ist der Widerstand des NTC-Thermistors bei der Nenntemperatur TN. Hier beträgt der numerische Wert von RN 10k.
- TK ist eine Kelvin-Temperatur und die Einheit ist K. Hier beträgt der numerische Wert von TK 273,15 + Grad Celsius.
- TN ist eine Nenntemperatur in Kelvin; die Einheit ist ebenfalls K. Hier beträgt der numerische Wert von TN 273,15 + 25.
- Und B (Beta), die Materialkonstante des NTC-Thermistors, wird auch als Wärmeempfindlichkeitsindex bezeichnet und hat den numerischen Wert 3950.
- exp steht für Exponentialfunktion, und die Basiszahl e ist eine natürliche Zahl und beträgt ungefähr 2,7.

Verwenden Sie diese Formel  $TK=1/(\ln(RT/RN)/B+1/TN)$ , um die Kelvin-Temperatur zu erhalten, von der 273,15 abgezogen Grad Celsius entspricht.

Diese Beziehung ist eine empirische Formel. Sie ist nur dann genau, wenn die Temperatur und der Widerstand im effektiven Bereich liegen.



Wenn die Temperatur höher als 25 Grad Celsius ist, setzen Sie GP14 auf EIN und GP15 auf AUS, um den Motor rotieren zu lassen. Alternativ können Sie deren Zustände auch umkehren. Wenn die Temperatur niedriger als 25 Grad Celsius ist, setzen Sie sowohl GP14 als auch GP15 auf NIEDRIG, um den Motor anzuhalten.

## 7.16 2.13 Reaktionsspiel

In diesem Projekt verwenden wir mehrere Tasten, einen Summer und LEDs, um ein Reaktionsspiel zu erstellen. Drücken Sie die Schiedsrichtertaste, um das Spiel zu starten; der Summer gibt dabei kontinuierlich Töne aus, um das Fortsetzen des Spiels anzuzeigen. Betätigen Sie schnell die beiden Spielertasten; wenn die Schiedsrichtertaste erneut gedrückt wird, ist das Spiel beendet und der Summer hört auf zu summen. Überprüfen Sie dann auf Piper Make in der Konsole, welcher Spieler schneller war.

### Benötigte Komponenten

Für dieses Projekt benötigen wir folgende Komponenten:

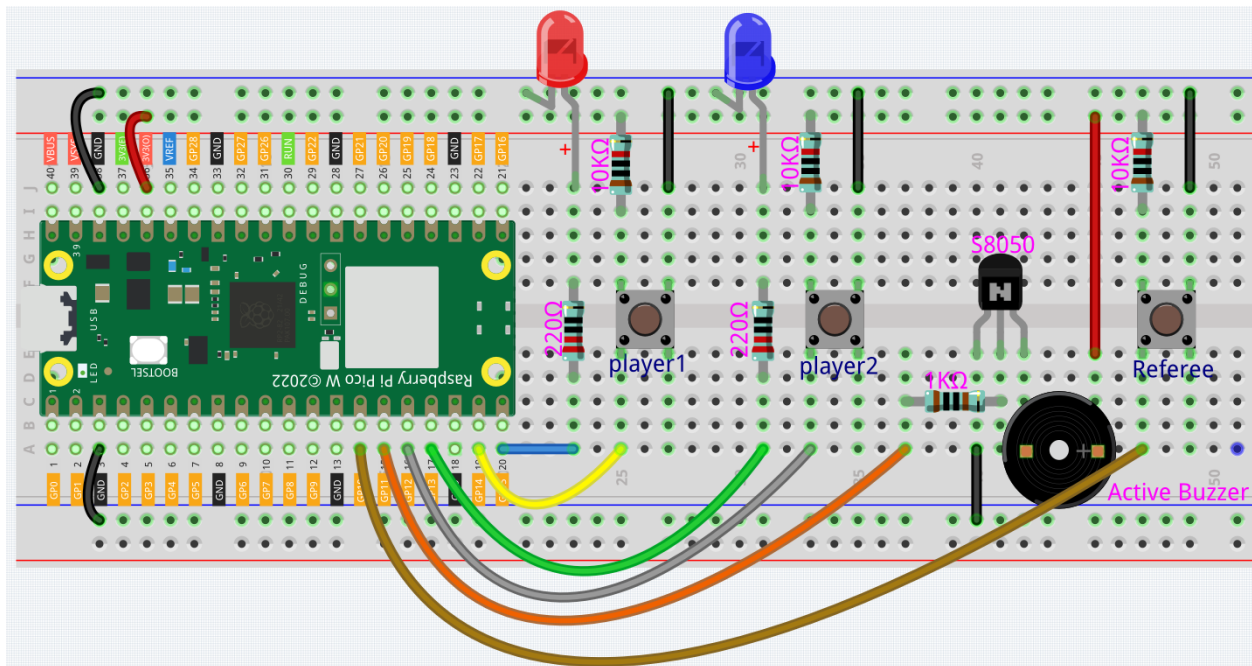
Ein komplettes Kit ist natürlich praktisch. Hier ist der Link dazu:

Name	ARTIKEL IN DIESEM SET	LINK
Kepler Kit	450+	

Sie können die einzelnen Komponenten auch über die untenstehenden Links kaufen.

SN	KOMPONENTE	AN-ZAHL	LINK
1	<a href="#">Raspberry Pi Pico W</a>	1	
2	Micro USB Kabel	1	
3	<a href="#">Steckbrett</a>	1	
4	<a href="#">Jumperkabel</a>	Mehrere	
5	<a href="#">Transistor</a>	1(S8050)	
6	<a href="#">Widerstand</a>	6(2-220, 1-1K, 3-10K)	
7	Aktiver <a href="#">Summer</a>	1	
8	<a href="#">Taster</a>	3	
9	<a href="#">LED</a>	2	

### Verkabelung

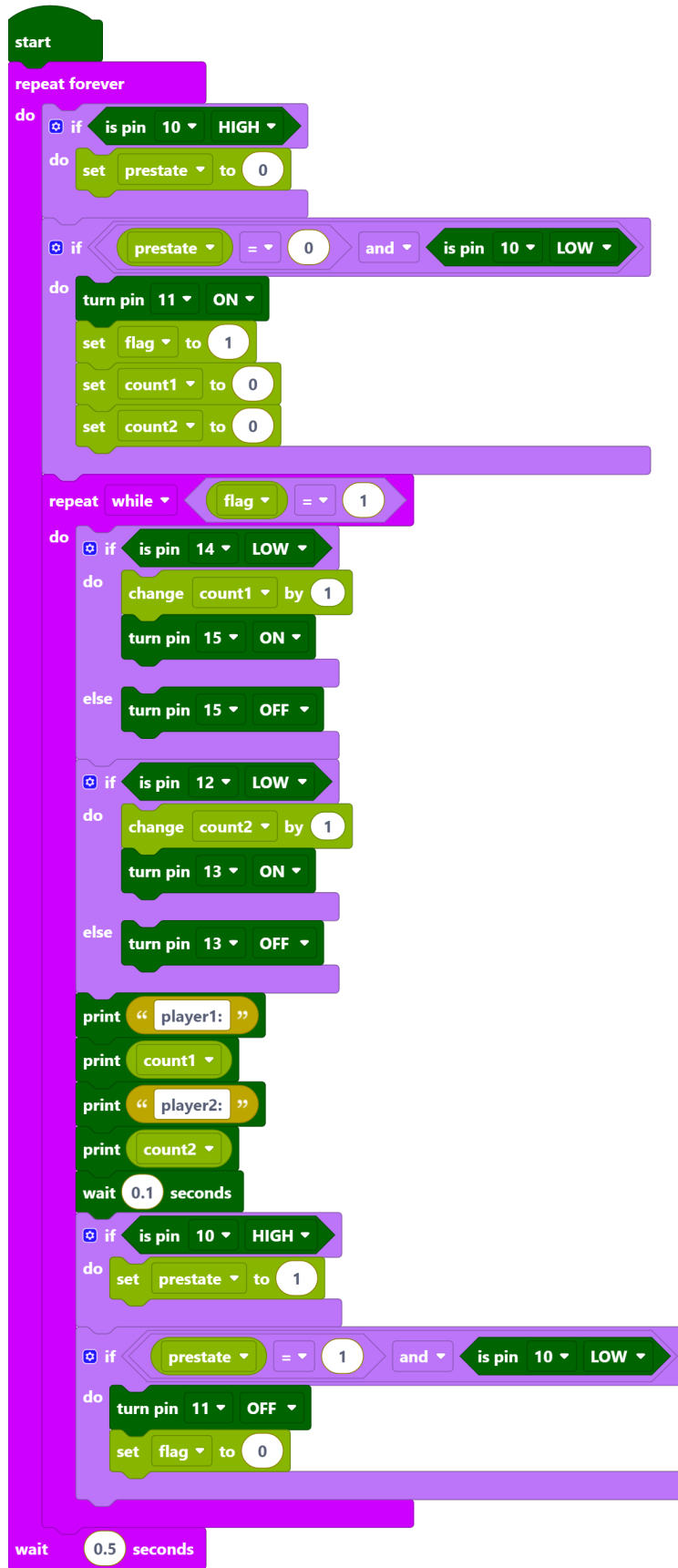


- Definieren Sie zwei Tasten als Spieler1 (GP14) und Spieler2 (GP12), beide sind mit einem Pull-up-Widerstand verbunden. Bei Betätigung der Tasten gehen GP14 und GP12 jeweils auf niedrig.
- Die zugehörigen Indikatoren sind mit GP15 und GP13 verbunden und leuchten auf, wenn diese Pins auf hoch gesetzt sind.
- Definieren Sie eine Schiedsrichtertaste, die mit GP10 verbunden ist. Bei Betätigung geht GP10 auf niedrig.
- Der aktive Summer ist mit GP11 verbunden. Wenn GP11 auf hoch gesetzt ist, gibt der Summer einen Ton aus.

#### Code

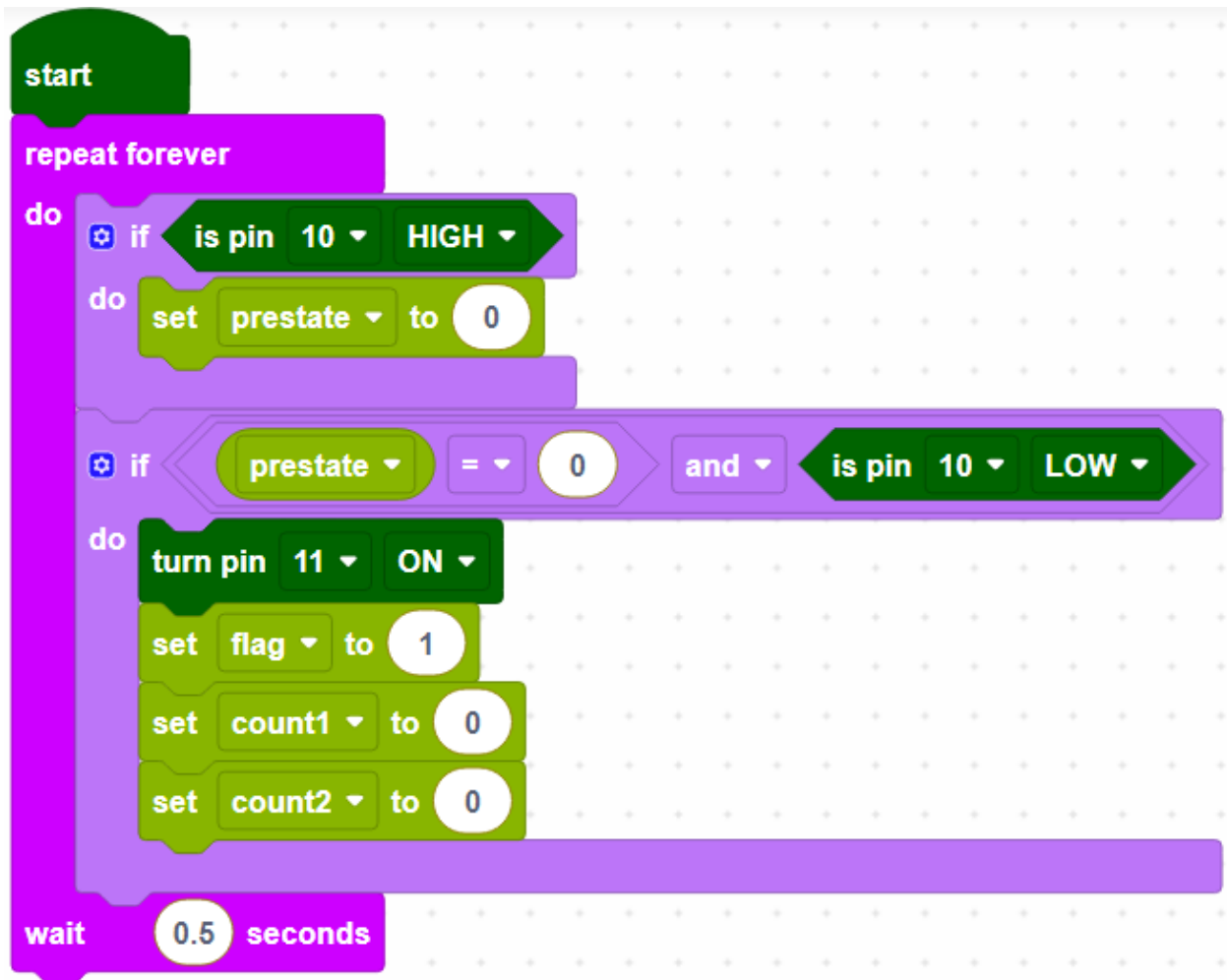
#### Bemerkung:

- Die Abbildung unten können Sie als Orientierung für das Schreiben des Codes per Drag-and-Drop verwenden.
- Importieren Sie `2.13_reaction_game.png` aus dem Verzeichnis `kepler-kit-main\piper`. Detaillierte Anleitungen finden Sie unter [Code importieren](#).

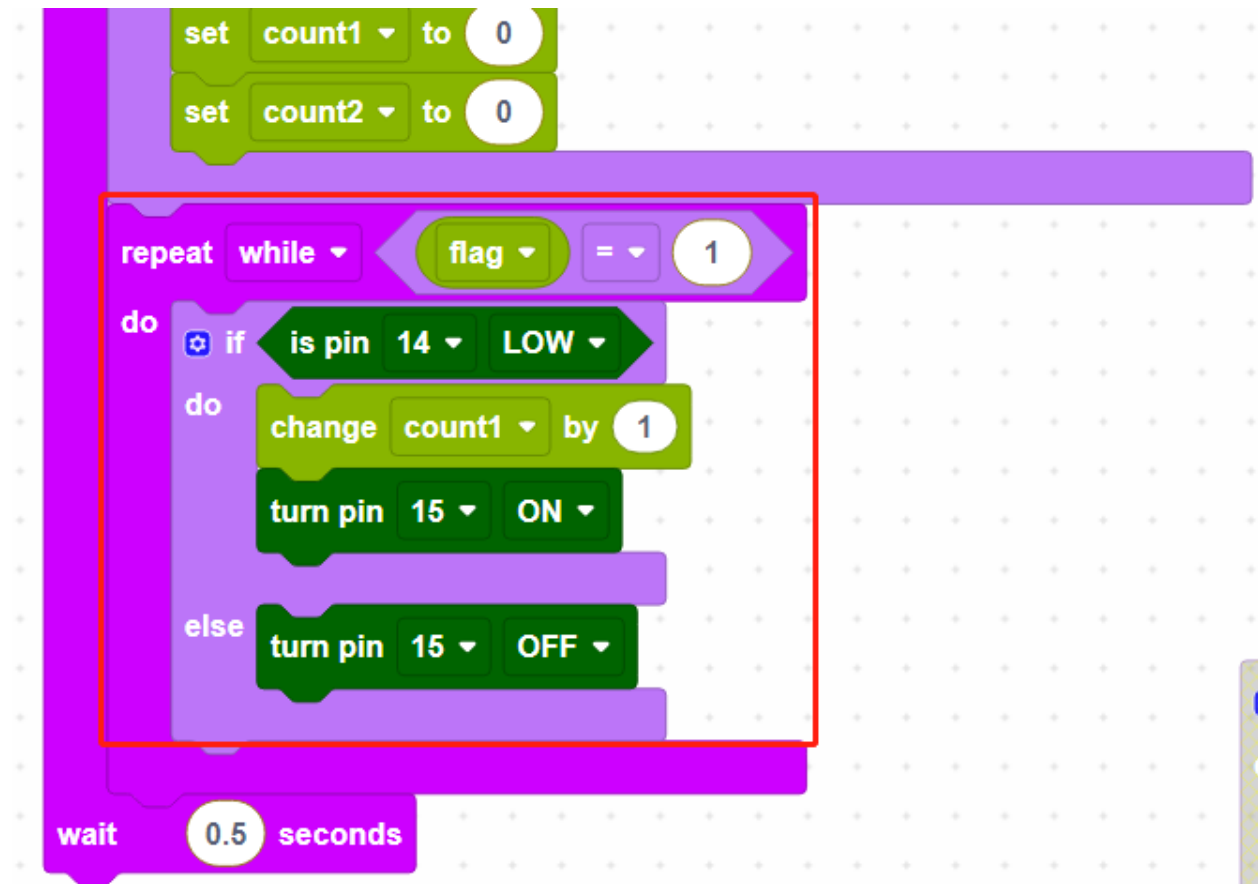


- Nach dem Anschluss von Pico W klicken Sie auf die **Start**-Taste, und der Code wird ausgeführt.
- Betätigen Sie die Schiedsrichtertaste und der Summer gibt einen kontinuierlichen Ton aus, was den Spielstart signalisiert.
- Drücken Sie nun die **Spielertasten** einzeln und schnell. Die zugehörigen LEDs werden aufleuchten.
- Wird die **Schiedsrichtertaste** erneut gedrückt, stoppt der Summer, was das Spielende bedeutet.
- Klicken Sie auf diesem Punkt auf die Konsole, um zu sehen, welcher Spieler mehr Betätigungen hatte.

#### Funktionsweise

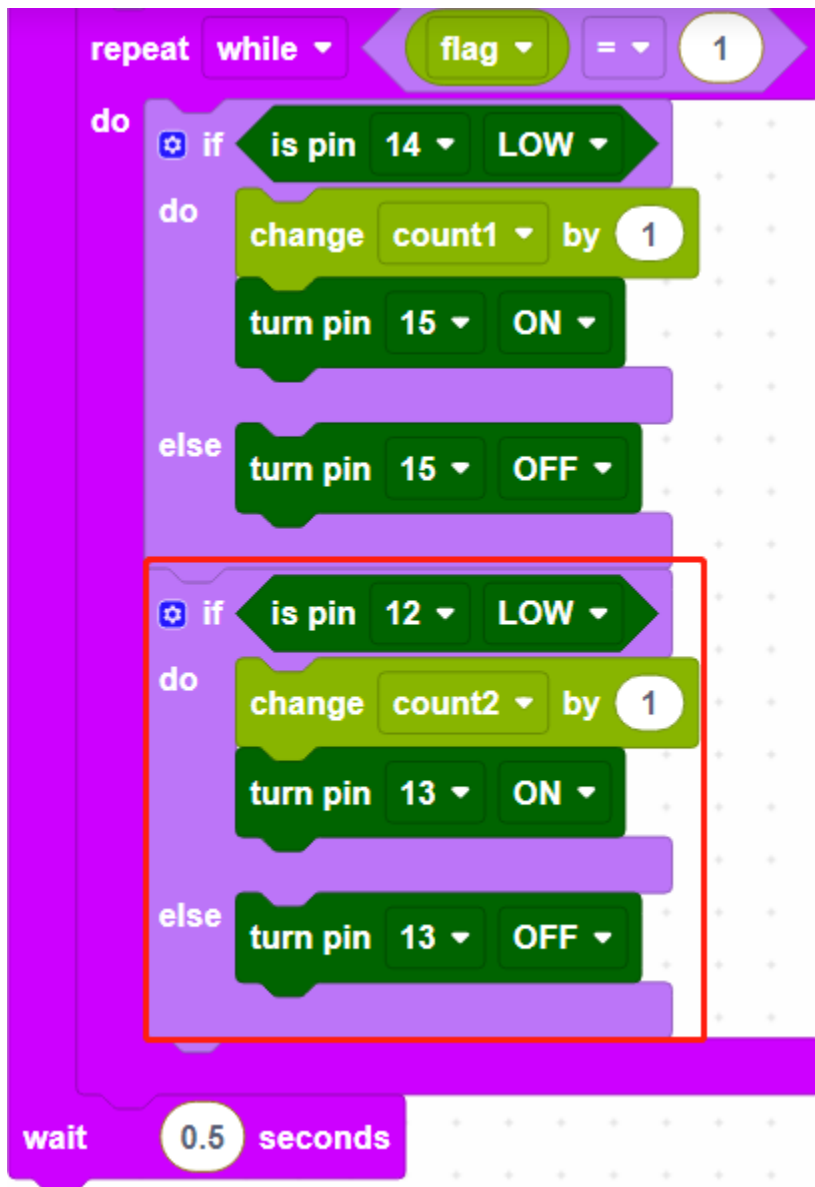


GP10 ist die Schiedsrichtertaste. Wenn diese nicht gedrückt ist (hoch), hat das Spiel noch nicht begonnen. Wenn GP10 niedrig ist (Schiedsrichtertaste gedrückt), beginnt das Spiel; setzen Sie GP11 auf hoch (Summer), erstellen Sie Variablen und setzen Sie die Anfangswerte.

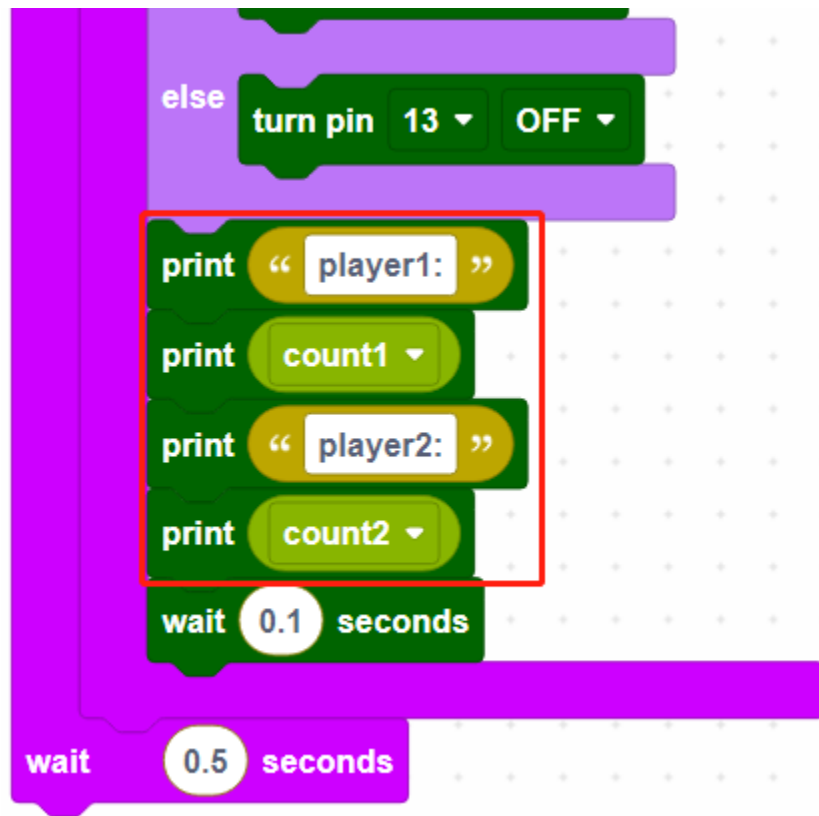


Wenn die Flagge 1 für den Spielstart ist, dann lesen Sie den Wert von GP14 (Spieler1); wenn die Spieler1-Taste gedrückt wird, speichern Sie die Anzahl der Betätigungen in der Variablen [count1] und lassen Sie den Indikator von GP15 aufleuchten.

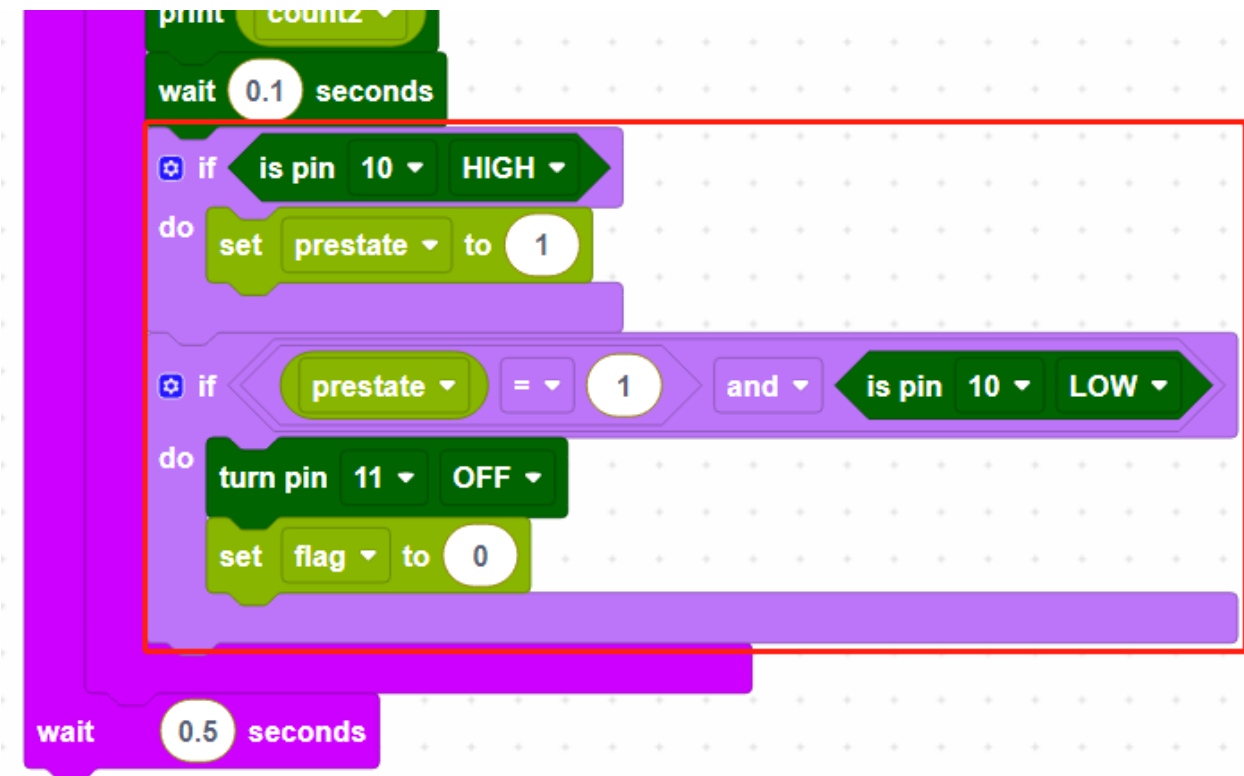




Lesen Sie die Anzahl der Betätigungen von GP12 (Spieler2) auf die gleiche Weise.

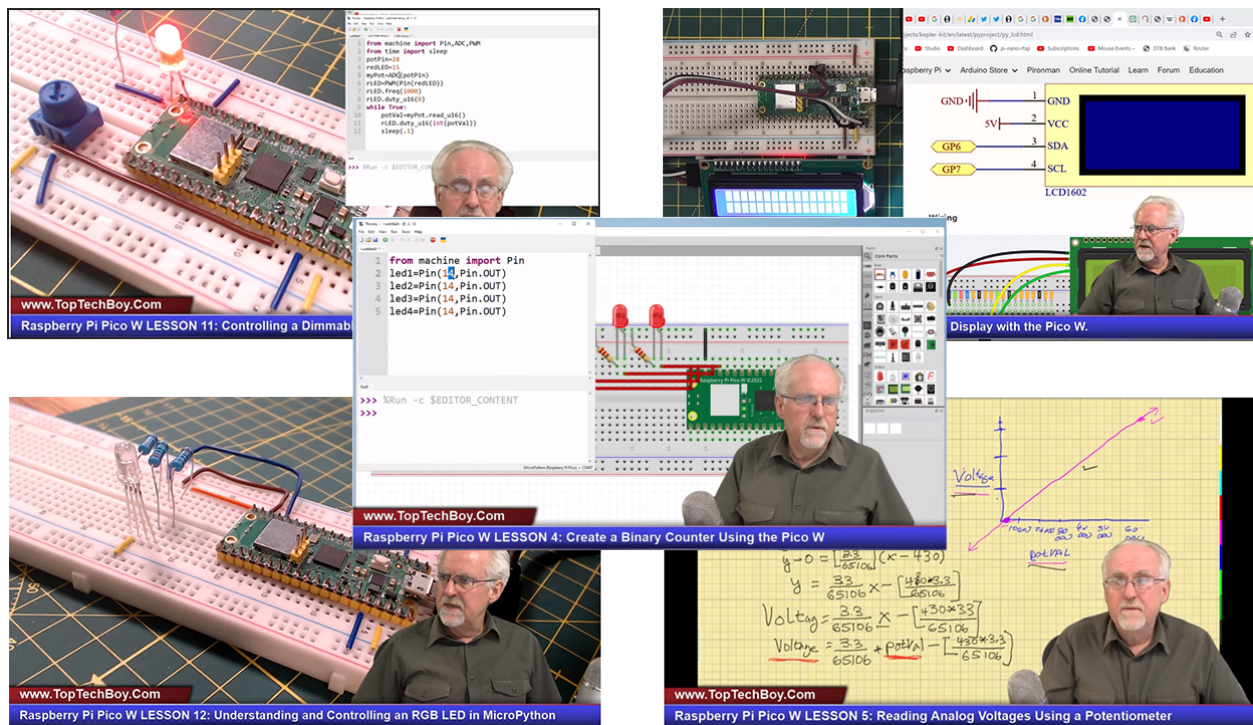


Drucken Sie die Anzahl der Betätigungen für Spieler1 und Spieler2 jeweils aus.



Wenn die Schiedsrichtertaste erneut gedrückt wird, hört der Summer auf zu arbeiten und das Spiel endet.

Falls Sie den Inhalt der Online-Dokumentation etwas herausfordernd finden, keine Sorge. Ein schrittweiser Videokurs ist hier, um Ihren Lernprozess angenehmer und ansprechender zu gestalten.



Den Videokurs können Sie unter folgendem Link abrufen: .

Diese Videos konzentrieren sich auf den Raspberry Pi Pico W und führen Sie durch das Erlernen von MicroPython, die Funktionsweisen verschiedener Komponenten sowie das dazugehörige Schaltungswissen. Jedes Video ist so konzipiert, dass es sowohl unterhaltsam als auch lehrreich ist und Konzepte auf eine ansprechende und interaktive Weise präsentiert. Tauchen Sie in diesen Videokurs ein und beteiligen Sie sich an den praktischen Übungen. So verwandelt sich Ihre Lernreise mit dem Raspberry Pi Pico W in ein bereicherndes und vergnügliches Erlebnis.



## 9.1 Arduino

### 1. Code-Upload in der Arduino IDE fehlgeschlagen?

- Überprüfen Sie, ob Ihr Pico im Arduino IDE korrekt erkannt wird. Der Port sollte COMXX (Raspberry Pi Pico) sein. Für Anweisungen verweisen wir auf [1.3 Raspberry Pi Pico W einrichten \(Wichtig\)](#).
- Überprüfen Sie, ob das Board (Raspberry Pi Pico) und der Port (COMXX (Raspberry Pi Pico)) korrekt ausgewählt sind.
- Wenn Ihr Code in Ordnung ist und Sie das richtige Board und den richtigen Port ausgewählt haben, der Upload jedoch immer noch nicht erfolgreich ist, klicken Sie erneut auf das **Upload**-Symbol. Wenn der Fortschrittsbalken „Upload...“ anzeigt, trennen Sie das USB-Kabel und halten Sie die **BOOTSEL**-Taste gedrückt, während Sie es wieder einstecken. Der Code wird dann erfolgreich hochgeladen.

## 9.2 MicroPython

### 1. Wie öffne und führe ich den Code aus?

Für detaillierte Anleitungen verweisen wir auf [Code direkt öffnen und ausführen](#).

### 2. Wie lade ich eine Bibliothek auf den Raspberry Pi Pico W hoch?

Für detaillierte Anleitungen verweisen wir auf [1.4 Bibliotheken auf den Pico hochladen](#).

### 3. Keine MicroPython (Raspberry Pi Pico W) Interpreter-Option in der Thonny IDE?

- Stellen Sie sicher, dass Ihr Pico W über ein USB-Kabel an Ihren Computer angeschlossen ist.
- Überprüfen Sie, ob Sie MicroPython für Pico W installiert haben ([1.3 Installation von MicroPython auf Ihrem Pico](#)).
- Der Raspberry Pi Pico W Interpreter ist nur in der Version 3.3.3 oder höher der Thonny IDE verfügbar. Wenn Sie eine ältere Version verwenden, aktualisieren Sie bitte ([1.2 Installation der Thonny IDE](#)).

- Wenn das Li-Po-Lademodul zu diesem Zeitpunkt auf dem Steckbrett eingesteckt ist, ziehen Sie es zuerst ab und stecken Sie den Pico W wieder in den Computer.

#### 4. Kann den Pico W Code nicht öffnen oder Code über die Thonny IDE auf Pico W speichern?

- Überprüfen Sie, ob Ihr Pico W über ein USB-Kabel an Ihren Computer angeschlossen ist.
- Stellen Sie sicher, dass Sie den Interpreter als **MicroPython (Raspberry Pi Pico)** ausgewählt haben.

#### 5. Kann der Raspberry Pi Pico W gleichzeitig in Thonny und Arduino verwendet werden?

NEIN, dazu sind unterschiedliche Vorgänge erforderlich.

- Wenn Sie es zuerst in der Arduino IDE verwendet haben und es nun in der Thonny IDE verwenden möchten, müssen Sie MicroPython darauf installieren (*1.3 Installation von MicroPython auf Ihrem Pico*).
- Wenn Sie es zuerst in der Thonny IDE verwendet haben und es nun in der Arduino IDE verwenden möchten, folgen Sie den Anweisungen unter *1.3 Raspberry Pi Pico W einrichten (Wichtig)*.

#### 6. Ihr Computer ist Win7 und Pico W wird nicht erkannt.

- Laden Sie den USB-CDC-Treiber von <http://aem-origin.microchip.com/en-us/mindi-sw-library?swsearch=Atmel%2520USB%2520CDC%2520Virtual%2520COM%2520Driver> herunter.
- Entpacken Sie die Datei `amtel_devices_cdc.inf` in einen Ordner namens `pico-serial`.
- Ändern Sie den Namen der Datei `amtel_devices_cdc.inf` in `pico-serial.inf`.
- Öffnen/Bearbeiten Sie die `pico-serial.inf` in einem einfachen Editor wie dem Notepad.
- Entfernen und ersetzen Sie die Zeilen unter den folgenden Überschriften:

```
[DeviceList]
%PI_CDC_PICO%=DriverInstall, USB\VID_2E8A&PID_0005&MI_00

[DeviceList.NTAMD64]
%PI_CDC_PICO%=DriverInstall, USB\VID_2E8A&PID_0005&MI_00

[DeviceList.NTIA64]
%PI_CDC_PICO%=DriverInstall, USB\VID_2E8A&PID_0005&MI_00

[DeviceList.NT]
%PI_CDC_PICO%=DriverInstall, USB\VID_2E8A&PID_0005&MI_00

[Strings]
Manufacturer = "ATMEL, Inc."
PI_CDC_PICO = "Pi Pico Serial Port"
Serial.SvcDesc = "Pi Pico Serial Driver"
```

1. Schließen und speichern Sie die Datei, und behalten Sie den Namen `pico-serial.inf` bei.
2. Gehen Sie zur Geräteliste Ihres PCs, finden Sie den Pico unter Ports, der beispielsweise als CDC-Gerät bezeichnet wird. Ein gelbes Ausrufezeichen weist darauf hin.
3. Klicken Sie mit der rechten Maustaste auf das CDC-Gerät und aktualisieren oder installieren Sie den Treiber, indem Sie die von Ihnen erstellte Datei aus dem Speicherort auswählen, an dem Sie sie gespeichert haben.

## 9.3 Piper Make

1. **Wie richte ich den Pico W in Piper Make ein?**

Für detaillierte Anleitungen verweisen wir auf *1.1 Einrichtung des Pico*.

2. **Wie lade ich Code herunter oder importiere ihn?**

Für detaillierte Anleitungen verweisen wir auf *1.3 Wie speichert oder importiert man Code?*.

3. **Wie stelle ich eine Verbindung zu Pico W her?**

Für detaillierte Anleitungen verweisen wir auf *2. Verbindung zu Pico W herstellen*.





## KAPITEL 10

---

### Urheberrechtshinweis

---

Sämtliche Inhalte dieses Handbuchs, einschließlich, jedoch nicht beschränkt auf Texte, Bilder und Code, sind Eigentum der SunFounder Company. Sie dürfen diese nur für persönliche Studien, Recherchen, Vergnügen oder andere nichtkommerzielle oder gemeinnützige Zwecke unter Einhaltung der jeweiligen Vorschriften und Urheberrechtsgesetze verwenden, ohne die rechtlichen Ansprüche des Autors und der betreffenden Rechteinhaber zu verletzen. Bei kommerzieller Nutzung ohne Genehmigung behält sich die Firma das Recht vor, rechtliche Schritte einzuleiten.