SmartHome4SENIORS
Anleitung für den Baukasten

Co-funded by
the European Union

Die Unterstützung der Europäischen Kommission für die Erstellung dieser Veröffentlichung stellt keine Billigung der Inhalte dar. Diese spiegeln ausschließlich die Meinung der Autoren wider, und die Kommission kann nicht für die Verwendung der darin enthaltenen Informationen verantwortlich gemacht werden.

Hinweis

Bitte beachten Sie folgende Hinweise:

1. Dieses Produkt enthält Kleinteile. Das Verschlucken oder falsche Verwenden dieser kann eine ernsthafte medizinische Gefahr darstellen. Rufen Sie in diesen Fällen oder bei anderweitigen Unfällen sofort medizinische Hilfe.
2. Die Verwendung dieses Produkts und seiner Teile in der Nähe von Steckdosen oder anderen Stromkreisen ist wegen der Gefahr eines Stromschlags untersagt.
3. Halten Sie leitende Materialien von diesem Produkt fern.
4. Es ist kleinen Kindern untersagt, dieses Produkt ohne Aufsicht von Erwachsenen zu benutzen. Dieses Produkt muss außerhalb der Reichweite von Kindern und Haustieren aufbewahrt werden.
5. Lagern oder verwenden Sie dieses Produkt nicht bei extremen Umgebungsbedingungen wie extremer Hitze oder Kälte, hoher Luftfeuchtigkeit, direktem Sonnenlicht usw.
6. Denken Sie daran, den Strom abzuschalten, wenn das Produkt nicht genutzt wird.
7. Einige Teile dieses Produkts können sich bei der Verwendung warm anfühlen. Das ist normal.
8. Unsachgemäßer Gebrauch kann zu Überhitzung führen.
9. Die Verwendung von Bauteilen, die nicht den Empfehlungen entsprechen oder zum Produkt gehören, können zu Schäden an diesem führen.

Einleitung

Der SmartHome4SENIORS Baukasten basiert auf der Verwendung des Raspberry Pi Pico (RPi Pico) Mikrocontrollers. Der Baukasten wurde im Rahmen des gleichnamigen Erasmus+ kofinanzierten Projekts mit der Projektnummer 2021-1-DE02-KA220-ADU-000033587 entwickelt.

Smart-Home-Automatisierungen liegen im Trend und helfen Tausenden von Menschen, Prozesse in ihren Haushalten zu vereinfachen. Der SmartHome4SENIORS-Baukasten wurde vor dem Hintergrund entwickelt, dass viele Senior:innen Smart-Home-Technologien ablehnen, und soll demonstrieren, wie diese Produkte im Alter unterstützen können. Er soll dabei helfen, dass diese Zielgruppe Smart-Home-Technologielösungen besser versteht und die Verwendung dieser Produkte (angepasst an die Bedürfnisse von älteren Menschen) in Betracht zieht.

Ziel ist die praxisorientierte Demonstration der Möglichkeiten von Smart Home Technologien für Senior:innen und wie man mit Hilfe dieser ein gesundheitsförderliches und sicheres Leben Zuhause führen kann. Der Baukasten bietet den Nutzer:innen eine Gelegenheit, in die Welt der intelligenten Heimautomatisierungslösungen einzutauchen.

Der Bausatz enthält die gesamte erforderliche Hardware (Mikrocontroller, Elektronik, Sensoren, Peripheriegeräte usw.) und beinhaltet anwendungsorientierte Computer- und Programmierkonzepte.

Der Zweck dieses Handbuchs ist es:

- Sie über die Komponenten des Bausatzes und die Verwendung der wichtigsten elektronischen Elemente zu informieren.

- Sie Schritt für Schritt darin anzuleiten den Bausatz zusammenzubauen, unter Berücksichtigung der entsprechenden Vorsichtsmaßnahmen.
- Eine Anleitung für die Verwendung der Komponenten und deren Verbindung mit dem RPi Pico Mikrocontroller zu sein.
- Anleitungen zu den Funktionen zu geben und die jeweils benötigten Bausatz-Komponenten aufzulisten.

Viel Spaß beim Lesen, bei den praktischen Übungen und beim Experimentieren mit dem SmartHome4SENIORS Baukasten.

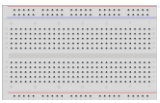
Inhaltsverzeichnis

Enthalten im SmartHome4SENIORS Baukasten sind

1 x Raspberry Pi Pico Microcontroller

1 x MB-102 Stromversorgungsmodul

1 x weiße Lochrasterplastine 830 pcs



1 x Druckknopff & 1 x Knopfkappe

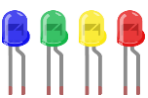
1 x Klingel

5 x Widerstände mit 220 Ohm & 2 x 1k Ohm

1 x LDR-Fotowiderstand

1 x 100uF-Kondensator

4 x LED 3mm (blau, grün, rot, gelb)



1 x RGB LED

1 x Ampel-LED-Modul

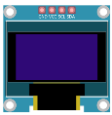
1 x Lüfter (DC-Motor)

1 x TIP-120 Steuermodul

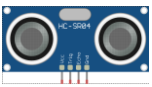
1 x Diode 1N4007

1 x SG90 Servomotor

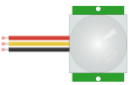
1 x OLED SSD1306 I2C Anzeige



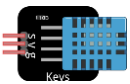
1 x HC-SR04 Ultraschallsensor



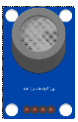
1 x PIR-Bewegungsmelder-Sensor HC-SR501



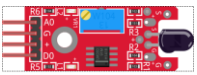
1 x DHT11 Digitaler Temperatur- und Luftfeuchtigkeitssensor



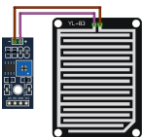
1 x MQ-135 Luftqualitätssensor



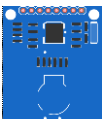
1 x Flammendetektionssensor



1 x Regensensor



1 x RFID-Lesegerät RC522



1 x RFID Tag 3.56MHz

1 x Dreh-Potentiometer Linear B1k Ohm

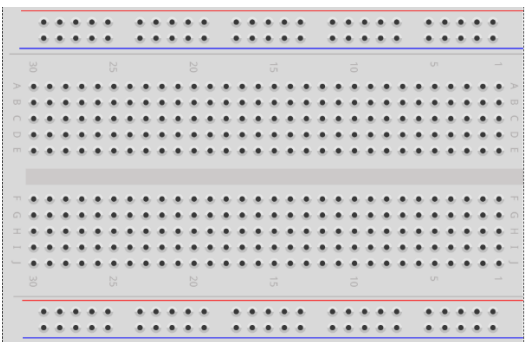
1 x USB zu micro-USB Kabel

6 x AA 1,5V Batterien & 1 x Batteriehalterung

9 x Überbrückungskabel	19 x Metallschrauben 2mm & 19 x Metallmuttern 2mm	4 x Metall-Schrauben 2mm
		

Beschreibung der Bestandteile

1. Breadboard (Steckbrett)



Ein Breadboard (auch Steckbrett oder Lochrasterplatine) ist eine Kunststoffplatte mit winzigen Löchern, in die elektronische Bauteile (Transistoren, Widerstände, Chips usw.) leicht eingesetzt werden können, um einen Prototyp einer elektronischen Schaltung zu bauen und zu testen. Das Innere besteht aus Reihen von winzigen Metallklammern, die die anzuschließenden Leitungen halten.

Die meisten Breadboards sind mit Reihen von Zahlen, Buchstaben sowie Plus- und Minuszeichen beschriftet. Die Beschriftungen sollen Ihnen helfen, bestimmte Löcher auf dem Steckbrett zu finden, damit Sie beim Aufbau einer Schaltung den Anweisungen folgen können.

Die langen Streifen an den beiden Seiten des Breadboards sind in der Regel mit rot und blau oder rot und schwarz sowie mit Plus- (+) und Minuszeichen (-) gekennzeichnet. Diese Reihen werden als "Busse" oder "Schienen" bezeichnet und dienen der Stromversorgung der Schaltung.

Der positive "Bus" ist rot markiert, hat das Pluszeichen (+) und liefert den Strom.

Der negative "Bus" ist blau oder schwarz markiert, hat ein Minuszeichen (-) und stellt die Masse dar.

Vorteile der Verwendung eines Breadboards:

- erleichtert die schnelle Überprüfung einfacher und komplexer Schaltungen und erleichtert die Überprüfung von Schaltungen im Ausgangsstadium
- leicht zu verstellen
- flexibel
- kein Bohren von Löchern nötig
- kein Löten notwendig
- einfaches Austesten von Schaltungen und Programmen

2. Was ist ein Widerstand?

Ein Widerstand kann als eine Komponente in einen Stromkreis eingebaut werden. Sobald dies geschieht, wird der Strom um einen bestimmten Betrag reduziert. Um die Stärke eines Widerstands zu bestimmen, gibt es ein Muster aus farbigen Streifen.

Farbe	Erstes Band	Zweites Band	3tes Band (nur 5tes Band)	Multiplikatoren (3tes oder 4tes Band)	Toleranz
schwarz	0	0	0	1	
braun	1	1	1	10	± 1%
rot	2	2	2	100	± 2%
orange	3	3	3	1000	
gelb	4	4	4	10000	
grün	5	5	5	100000	± 0,5%
blau	6	6	6	1000000	± 0,25%
lila	7	7	7	10000000	± 0,1%
grau	8	8	8		± 0,05%
weiß	9	9	9		
gold				0,1	± 5%
silber				0,01	± 10%
keine					± 1%

(Bild nach Future Owns) verfügbar unter https://www.tomshardware.com/how-to/resistor-color-codes

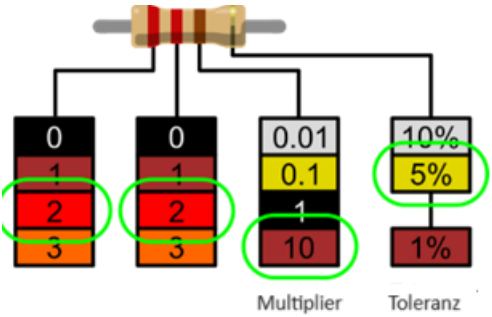
Die Codes der Widerstände und ihre Verwendung:

Widerstands-arten	4-Band Farbcodes	5-Band Farbcodes	Verwendung
220Ohm	Rot-rot-braun-gold	Rot-rot-schwarz-gold	LED Lichtschutz
1K Ohm (1Kiloohm)	Brown-schwarz-rot-gold	Braun-schwarz-schwarz-braun-gold	LED Lichtschutz, Spannungsteiler

Widerstände haben keine Polarität und können daher in jeder beliebigen Ausrichtung in einer Schaltung verwendet werden. Um die richtigen Werte für den Farbcode des Widerstands zu ermitteln, müssen die farbigen Streifen auf dem Widerstand verstanden werden.

Auf einem typischen Vierband-Widerstand für den Hobbybereich gibt es drei Farben in jeweils einer Gruppe. Diese bestehen aus jeweils zwei Ziffern und dem Multiplikator. Das letzte Band ist die Toleranz des Widerstands, eine Art Fehlerspanne. Für die meisten Bastler:innen ist eine Toleranz von 5 % (Gold) perfekt und üblich.

220 Ohm Resistor (4er-Band)

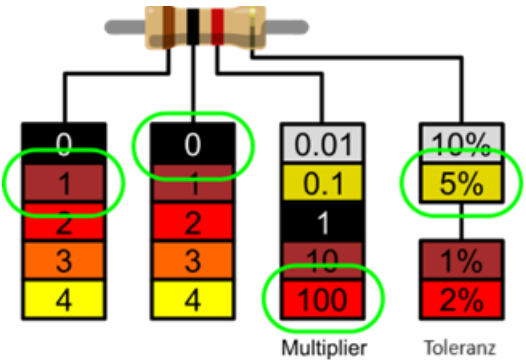


(Quelle: Future) verfügbar unter https://www.tomshardware.com/how-to/resistor-color-codes (übersetzt aus dem Englischen9

1. Die erste Ziffer ist rot und mit Hilfe des Decoders lässt sich ablesen, dass rot den Wert 2 hat.
2. Die zweite Zahl ist ebenfalls rot, was in Kombination 22 ergibt.
3. Der Multiplikator ist braun und wird als 10 dekodiert. Multipliziert mit den ersten beiden Zahlen ergibt das 220.
4. Das letzte Band, die Toleranz, ist gold. Gold steht für 5%, was bedeutet, dass ein Widerstand mit einer Fehlermarge von 5 % akzeptiert werden kann.

Für Hersteller, die eine höhere Präzision benötigen, gibt es auch Fünf-Band-Widerstände, die eine dritte signifikante Zahl haben. Die zusätzliche Zahl sorgt für Klarheit, die in empfindlichen Schaltkreisen wichtig sein kann, z. B. in wissenschaftlichen und technischen Konstruktionen.

1K Ohm Widerstand (4er-Band)



(Quelle: Tom's Hardware) verfügbar unter https://www.tomshardware.com/how-to/resistor-color-codes (übersetzt)

1. Die 1. Zeile ist braun und mit Hilfe des Decoders lässt sich der Wert 1 ermitteln.
2. Die 2. Zeile ist schwarz, also 10.
3. Der Multiplikator ist rot und wird zu 100 dekodiert. 10 multipliziert mit 100 ergibt 1000.
4. Der letzte Bereich, die Toleranz, ist gold. Gold ist 5 % und bedeutet, dass der Widerstand mit einer Fehlermarge von 5 % akzeptiert werden kann.

3. Der Kondensator

Ein Kondensator ist ein Gerät, das elektrische Energie in einem elektrischen Feld speichert. Er ist ein passives elektronisches

Bauteil mit zwei Anschlüssen. Er besteht aus zwei elektrischen Leitern, die durch einen Abstand voneinander getrennt sind. Der Raum zwischen den Leitern kann mit einem Vakuum oder mit einem isolierenden Material, dem Dielektrikum, gefüllt sein. (Wikipedia)

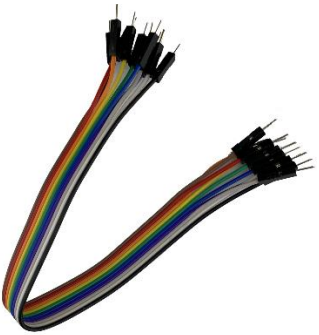
Der 100uF-Kondensator ist ein elektrolytischer Entkopplungskondensator. Diese Kondensatoren eignen sich hervorragend zur Unterdrückung von Spannungsspitzen. Die Verwendung eines solchen Kondensators zwischen der Stromversorgung und der Masse des Schaltkreises gewährleistet eine reibungslose Stromversorgung.

4. Die Diode

Eine Diode ist ein elektronisches Bauteil mit zwei Anschlüssen, das den Strom hauptsächlich in eine Richtung leitet (asymmetrischer Leiter). Sie hat einen geringen Widerstand (im Idealfall gleich Null) in einer Richtung und einen hohen (im Idealfall unbegrenzten) Widerstand in der anderen.

Die häufigste Funktion einer Diode besteht darin, einen elektrischen Strom in eine Richtung fließen zu lassen (Vorwärtsrichtung der Diode), während sie ihn in die entgegengesetzte Richtung sperrt (Rückwärtsrichtung). Die Diode kann somit als elektronische Version eines Rückschlagventils betrachtet werden. Dieses unidirektionale Verhalten wird als Gleichrichtung bezeichnet und dient der Umwandlung von Wechselstrom (ac) in Gleichstrom (dc). Als Gleichrichter können Dioden unter anderem dazu verwendet werden, Modulationen aus Funksignalen in Funkempfängern zu extrahieren. (Wikipedia https://en.wikipedia.org/wiki/Diode)

5. Das Überbrückungskabel



Überbrückungskabel sind einfache Drähte, die an beiden Enden mit Steckerstiften versehen sind, so dass sie zwei Punkte ohne Löten miteinander verbinden können. Überbrückungskabel werden in der Regel mit Lochrasterplatinen und anderen Prototyping-Tools verwendet, um eine Schaltung bei Bedarf einfach ändern zu können. Die Farbvariation der Drähte kann als Vorteil genutzt werden, um zwischen verschiedenen Arten von Verbindungen zu unterscheiden, z. B. Erdung oder Strom.

Überbrückungsdrähte gibt es in der Regel in drei Ausführungen: Stecker-zu-Stecker, Stecker-zu-Buchse und Buchse-zu-Buchse. Der Unterschied zwischen ihnen liegt im Endpunkt des Kabels. Männliche Enden haben einen hervorstehenden Stift und können in Dinge eingesteckt werden, während weibliche Enden dies nicht tun und zum Einstecken in Dinge verwendet werden. Überbrückungsdrähte von männlich zu männlich sind am gebräuchlichsten. Wenn man zwei Anschlüsse auf einem Breadboard verbindet, ist meist ein Stecker-zu-Stecker-Kabel erforderlich. (https://blog.sparkfuneducation.com/what-is-jumper-wire)

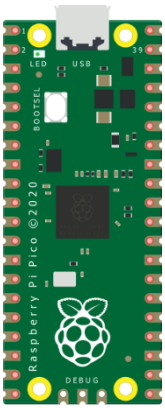
Vorbereitungen für das Projekt

1. Hinweis vor Beginn

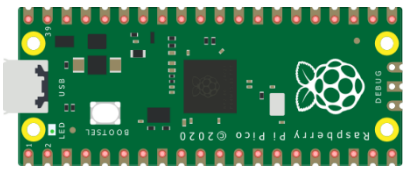
HINWEIS: Da es sich bei den Experimenten um Schaltungsexperimente handelt, kann ein falscher Anschluss oder Kurzschluss Ihr RPi Pico Entwicklungsboard beschädigen. Bitte überprüfen Sie die Schaltung immer noch einmal, bevor Sie die Stromversorgung anschließen.

2. Raspberry Pi Pico Microcontroller

Das ist der Raspberry Pi Pico:

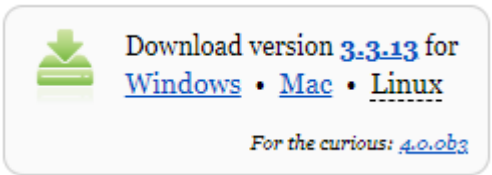


Stecken Sie das Micro-USB-Kabel in den Anschluss an der linken Seite der Karte.

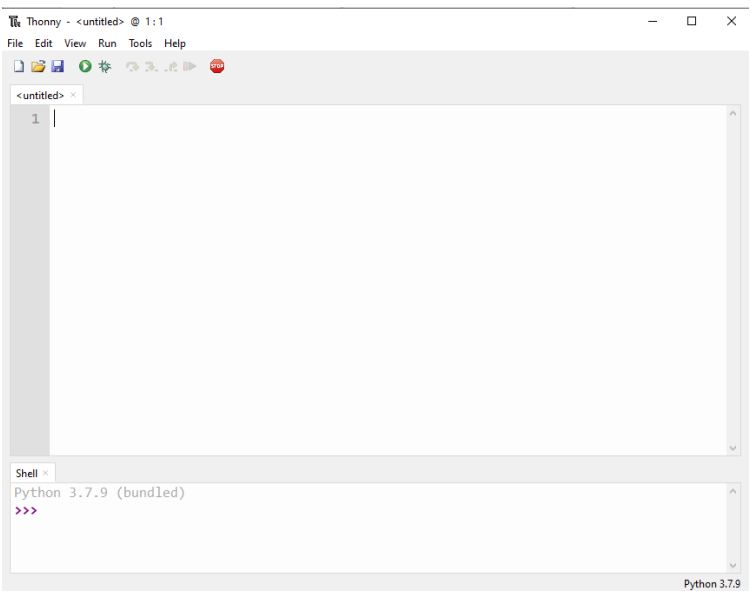


3. Installieren Sie Thonny IDE

Besuchen Sie https://thonny.org und wählen Sie das entsprechende Betriebssystem. Folgen Sie den Anweisungen, um die Installation abzuschließen.



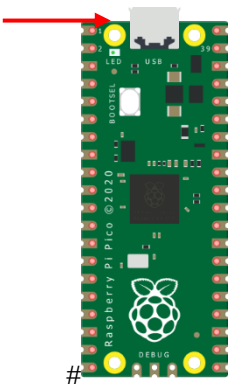
In diesem Handbuch werden alle Tutorials in Windows 10 programmiert, unter Verwendung eines RPi Pico Mikrocontrollers und der entsprechenden Firmware. Nachdem die Installation abgeschlossen ist, öffnen Sie Thonny auf Ihrem Computer.



4. Installation der Firmware

Der RPi Pico kann mit einer Python-Variante, genannt MicroPython, programmiert werden. Um MicroPython auf dem RPi Pico zu verwenden, müssen Sie zuerst die Firmware installieren.

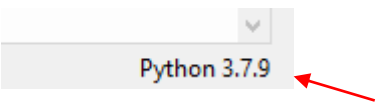
Schritt 1: Suchen Sie die BOOTSEL-Taste auf Ihrem Raspberry Pi Pico.



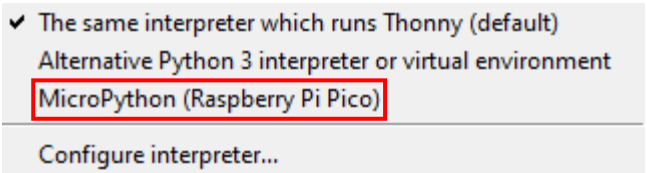
#

Schritt 2: Drücken Sie die BOOTSEL-Taste und halten Sie diese gedrückt, während Sie das andere Ende des Micro-USB-Kabels mit Ihrem Computer verbinden.

Schritt 3: In der unteren rechten Ecke von Thonny sehen Sie die Version von Python, die Sie derzeit verwenden.

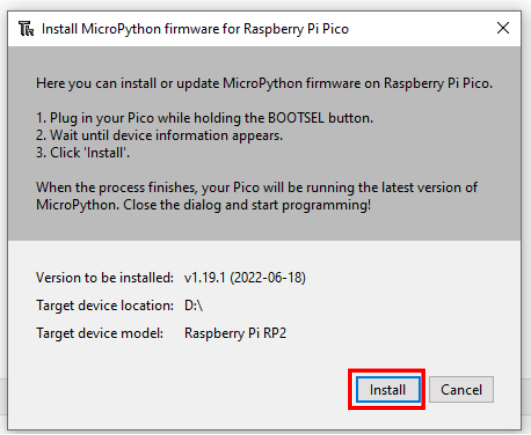


Klicken Sie auf die Python-Version und wählen Sie MicroPython (Raspberry Pi Pico).

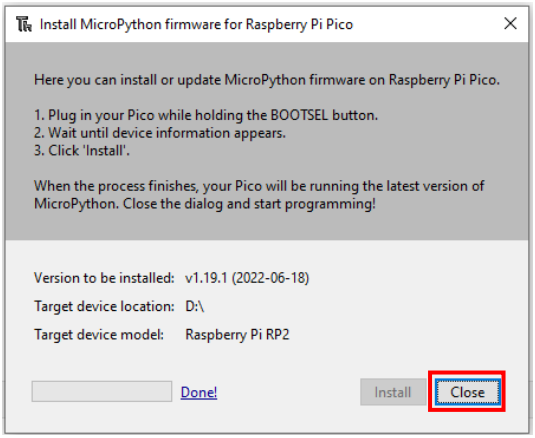


Wenn diese Option nicht angezeigt wird, vergewissern Sie sich, dass das Kabel am Pico und/oder an Ihrem Computer richtig angeschlossen ist.

Schritt 4: Ein Dialogfeld wird angezeigt, in dem Sie aufgefordert werden, die neueste Firmware-Version auf Ihrem Pico zu installieren. Klicken Sie auf die Schaltfläche „Installieren“, um die Firmware auf Ihren Pico zu kopieren.



Schritt 5: Warten Sie, bis die Installation abgeschlossen ist, und klicken Sie auf „Schließen“**.**



Beim nächsten Gebrauch müssen Sie den Raspberry Pi Pico nur noch an Ihren Computer anschließen und schon sind Sie startklar.

4. Einführung in das MicroPython Programmieren

Sie werden nun in der Entwicklungsumgebung Thonny IDE einen einfachen Python-Code ausführen und sich so mit der Thonny Shell und MicroPython vertraut machen.

Stellen Sie zunächst sicher, dass Ihr Raspberry Pi Pico an Ihren Computer angeschlossen ist und Sie den MicroPython-Interpreter ausgewählt haben.

Das Shell-Panel am unteren Rand des Thonny-Editors sollte wie folgt aussehen:

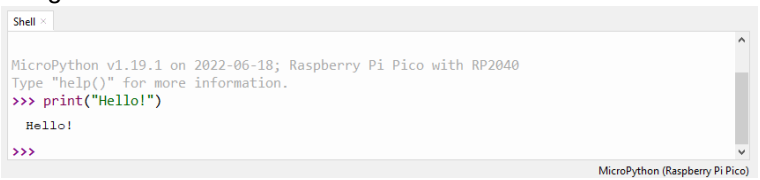
Thonny ist bereit, mit Ihrem Pico zu kommunizieren, indem es das REPL-Framework (read-eval-print loop) verwendet. Dieses

ermöglicht Ihnen, den Code direkt in der Shell zu schreiben und die Ausgabe zu erhalten.

Geben Sie den folgenden Befehl ein:

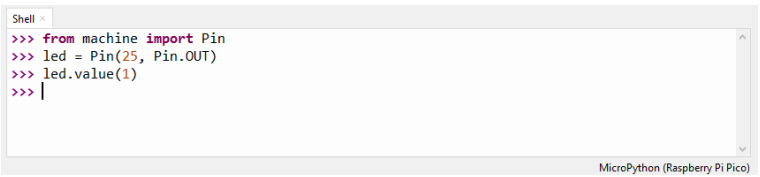
```
print(“Hello!”)
```

Drücken Sie dann die Eingabetaste. Es erscheint folgende Ausgabe:



Mit MicroPython können Sie hardwarespezifische Module hinzufügen, die Sie zur Programmierung Ihres Pico verwenden können. Im folgenden Beispiel wird das Maschinenmodul verwendet, um die integrierte LED des Pico einzuschalten. Schreiben Sie den folgenden Code in Thonny's Shell:

```
from machine import Pin
led = Pin(25, Pin.OUT)
led.value(1)
```



Drücken Sie die Eingabetaste. Sofort leuchtet die integrierte LED des Pico auf.

Um die LED auszuschalten, schreiben Sie den folgenden Code:

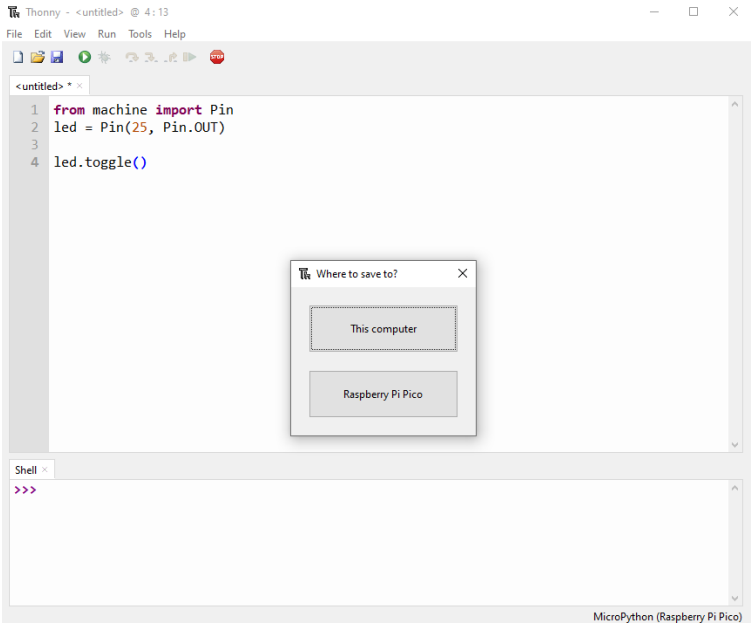
```
led.value(0)
```

```
Shell
>>> from machine import Pin
>>> led = Pin(25, Pin.OUT)
>>> led.value(1)
>>> led.value(0)
>>>
MicroPython (Raspberry Pi Pico)
```

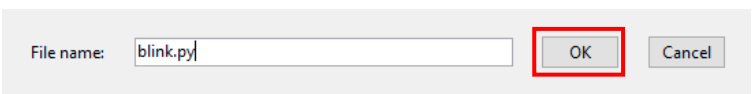
Im weiteren Verlauf dieses Abschnitts werden Sie Ihr erstes "richtiges" Programm schreiben. Das wird die Onboard-LED jedes Mal zum Blinken bringen, wenn Sie Ihr Programm ausführen.
Thonny Shell ist nützlich, um schnelle Befehle auszuprobieren und sicherzustellen, dass alles richtig funktioniert. Längere Programme sollten jedoch in einer .py-Datei gespeichert werden. Mit Thonny können Sie Programme direkt auf dem Raspberry Pi Pico speichern und sie dann ausführen.
Öffnen Sie Thonny Python und geben Sie im Hauptfenster des Editors den folgenden Code ein:

```
from machine import Pin
led = Pin(25, Pin.OUT)
led.toggle()
```

Speichern Sie nun Ihr Programm, indem Sie auf das Symbol „Speichern“ oder Strg+S auf Ihrer Tastatur drücken.



Thonny fragt nun, wo Sie Ihr Programm speichern möchten. Wählen Sie den **Raspberry Pi Pico**. Speichern Sie die Datei unter dem Namen blink.py und klicken Sie auf **OK**. Sie müssen immer die Erweiterung .py hinzufügen, damit Thonny die Datei als Python-Datei erkennen kann.



Jedes Mal, wenn Sie nun auf das Play-Symbol klicken, wird sich die Onboard LED ein und ausschalten.

Wenn Sie Ihren Code noch einen Schritt weiter ausführen, können Sie die Onboard-LED in einem bestimmten Rhythmus blinken lassen.

Schreiben Sie den folgenden Code und speichern Sie Ihr Programm unter demselben Namen wie oben.

```
from machine import Pin
from time import sleep
led = Pin(25, Pin.OUT)
while True:
      led.toggle()
      sleep(1)
```

Wenn Sie nun das Programm starten, blinkt die integrierte LED jede Sekunde solange bis Sie das Programm anhalten. Um das Programm zu stoppen, können Sie auf das STOPP-Symbol klicken oder Strg+C auf Ihrer Tastatur drücken.

In den weiteren Tutorials erfahren Sie, wie Sie weitere Elektronik und Sensoren hinzufügen und steuern und Programme erstellen, die miteinander kommunizieren können.

SmartHome Baukasten Aufbau

Für den Zusammenbau werden die folgenden Teile benötigt:

- 6 x Sperrholzstücke
- 10 x Metallbolzen
- 10 x Metallmuttern

Für den Aufbau sind sieben Schritte erforderlich. Alle Sperrholzteile sind an der rechten Ecke mit der folgenden Bezeichnung versehen:

BP: Basisteil

B: Rückseitenteil

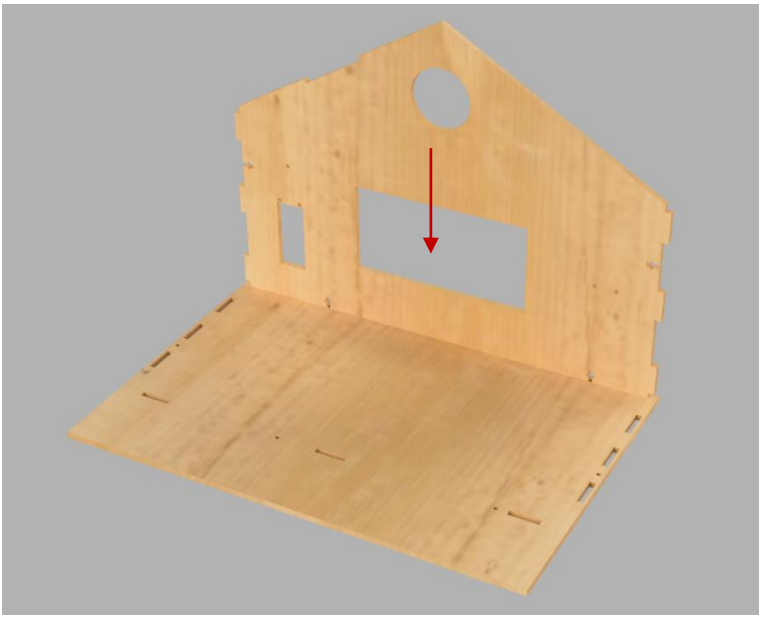
F: Vorderes Seitenteil

L: Linkes Seitenteil

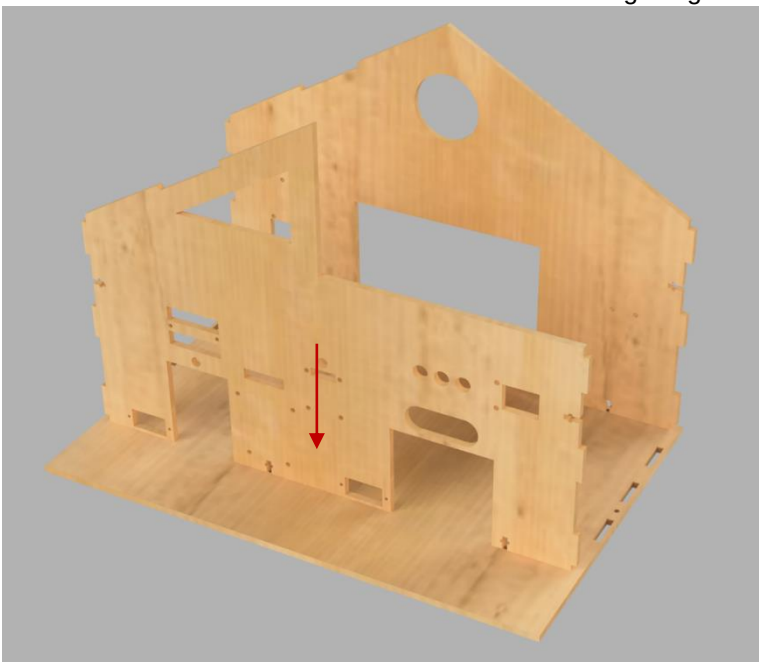
R: Rechtes Seitenteil

Schritt 1: Legen Sie das Sperrholzunterteil auf eine horizontale Fläche.

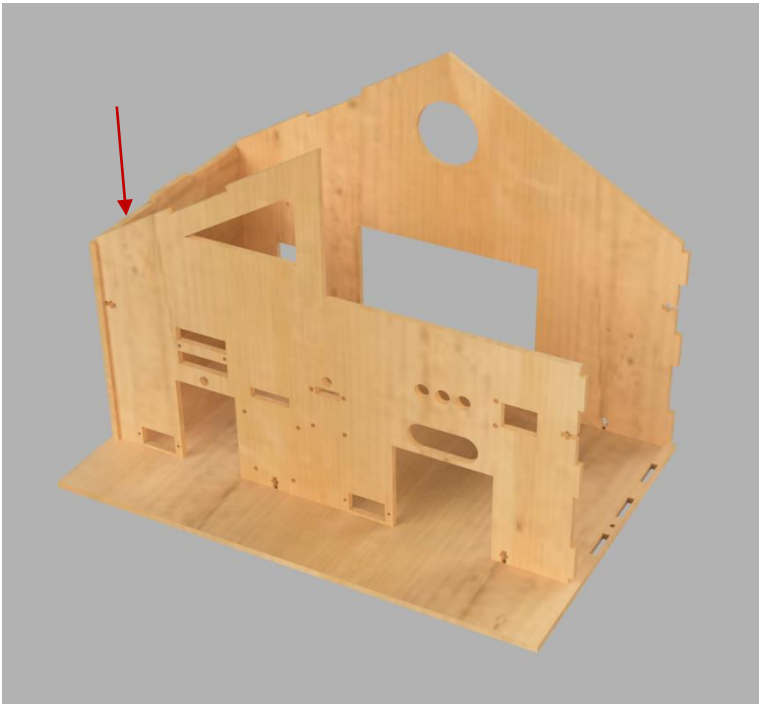
Schritt 2: Setzen Sie das Rückseitenteil wie auf dem Bild gezeigt ein.



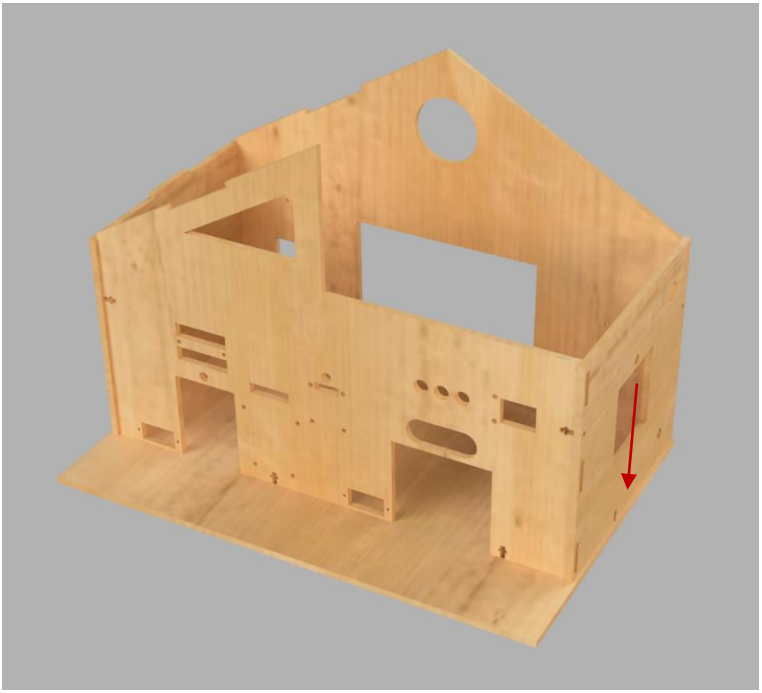
Schritt 3: Setzen Sie das Vorderteil wie auf dem Bild gezeigt ein.



Schritt 4: Setzen Sie das Sperrholzteil auf der linken Seite ein, wie auf dem Bild gezeigt.

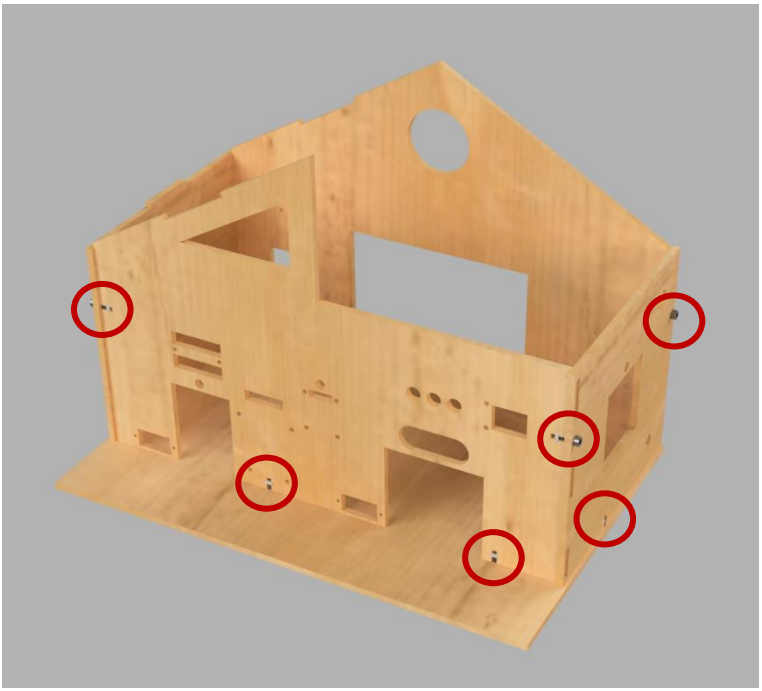


Schritt 5: Legen Sie das Sperrholzstück auf der rechten Seite ein, wie auf dem Bild gezeigt.

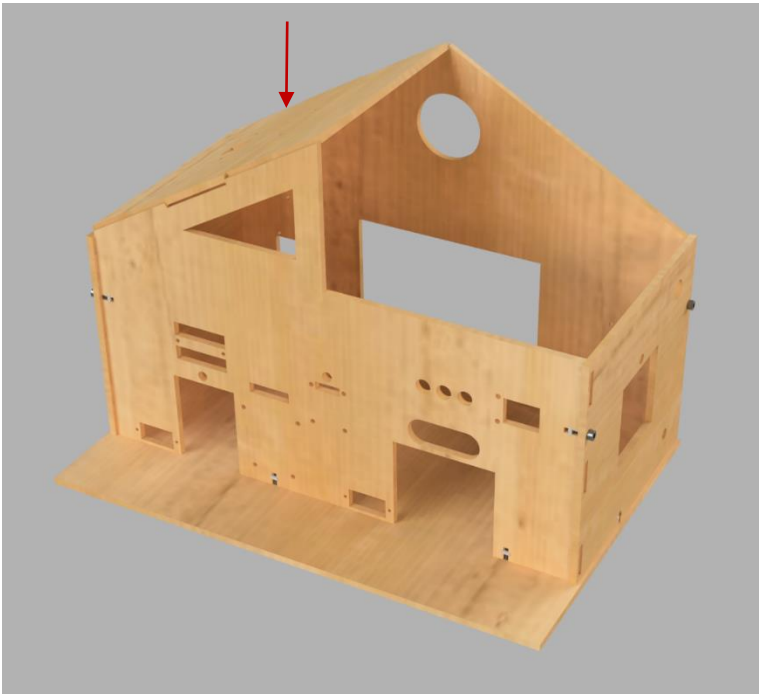


Schritt 6: Ziehen Sie die Sperrholzteile mit 10 Schrauben und 10 Muttern zusammen.

- 6 Schrauben und Muttern werden unter dem Bodenteil angebracht und sind für das Zusammenhalten der Wände verantwortlich.
- 4 Schrauben und Muttern werden an der linken und rechten Wand angebracht und sind für die Stabilität der Konstruktion verantwortlich.

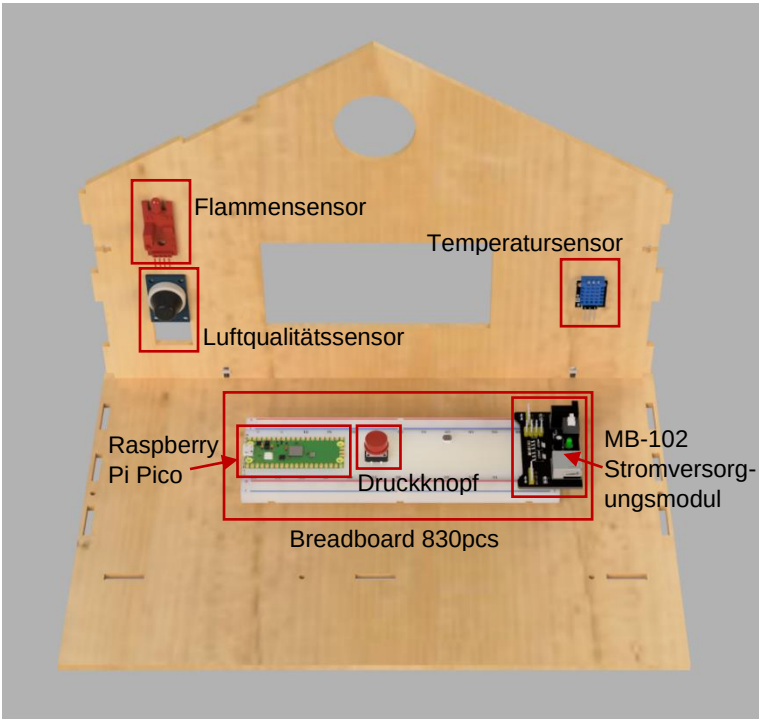


Schritt 7: Legen Sie das Sperrholzdach auf das Hausmodell, wie auf dem Bild gezeigt:



SmartHome Baukasten - Anbringen der Elektronik

Es gibt verschiedene elektronische Geräte, die in das SmartHome Hausmodell eingebaut werden. Die folgenden Bilder zeigen im Detail, welche Elektronik zu montieren ist und wie sie an den vorgesehenen Steckplätzen angebracht wird. Sie benötigen dafür einen Kreuzschlitzschraubendreher und eine Zange.



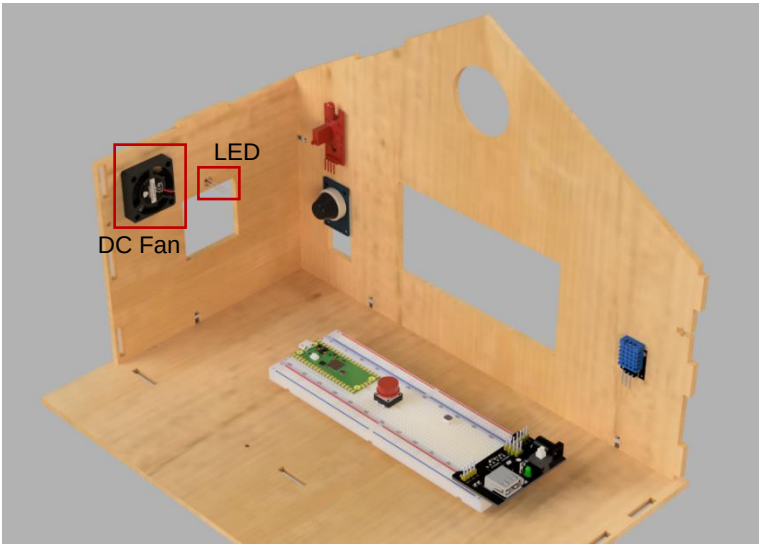
Legen Sie den Raspberry Pi Pico zusammen mit dem Stromversorgungsmodul (MB-102) und dem Druckknopf auf das Breadboard. Das Breadboard hat eine Klebefläche auf der

Unterseite. Ziehen Sie die Schutzschicht ab und legen Sie es auf das Basisteil des Hauses.

Es gibt einige elektronische Bauteile, die an der Rückwand montiert werden müssen. Dies sind der Flammensensor, der Luftqualitätssensor und der Temperatur- und Feuchtigkeitssensor DHT11. Dafür benötigen Sie folgende Elemente:

- **Flammensensor:** 1x Schraube und 1x Mutter. Montieren Sie ihn durch den mittleren Teil des Sensors und achten Sie darauf, dass der Sensor (schwarzer Teil) nach oben zeigt.
- **Luftqualitätssensor:** 2x Schraube und 2x Mutter. Montieren Sie ihn durch die Befestigungslöcher oben links und oben rechts.
- **DHT11:** 2x Schraube und 2x Mutter. Sie müssen den Sensor (blaues Teil) nach vorne biegen, um an die Befestigungslöcher zu gelangen. Montieren Sie den Sensor dann durch die Befestigungslöcher oben links und oben rechts.

Auf dem linken Seitenteil müssen Sie einen Gleichstromlüfter und eine LED-Leuchte montieren. Für den Gleichstromlüfter verwenden Sie die mitgelieferten Schrauben und Muttern (4x Schrauben und 4x Muttern). Setzen Sie die LED in das Montageloch ein.

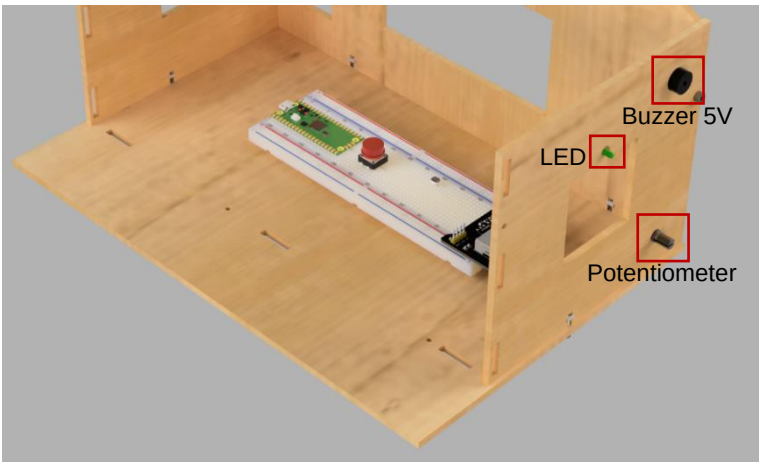


Auf dem rechten Teil müssen Sie einen 5-V-Summer, ein Potentiometer und eine LED anbringen.

Setzen Sie den **Summer (Buzzer)** in den Schlitz in der oberen rechten Ecke ein (achten Sie darauf, dass seine Stifte auf der Innenseite des Hausmodells liegen).

Das **Potentiometer** hat bereits einen Mutterring, den Sie abschrauben müssen. Legen Sie das Potentiometer von der Innenseite des Hauses nach außen, so dass der Hebel wie auf dem Bild auf der nächsten Seite aussieht. Dann schrauben Sie den Überwurfring zurück, bis er fest genug ist, um den Hebel bei Drehbewegungen in Position zu halten.

Schließlich muss eine LED installiert werden. Stecken Sie die **LED** in das Montageloch.

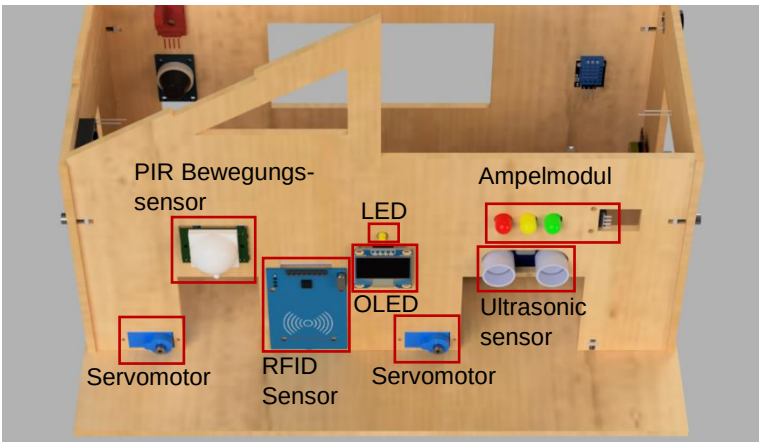


An der Vorderseite des Hausmodells müssen mehrere Sensoren und Elektronik angebracht werden.

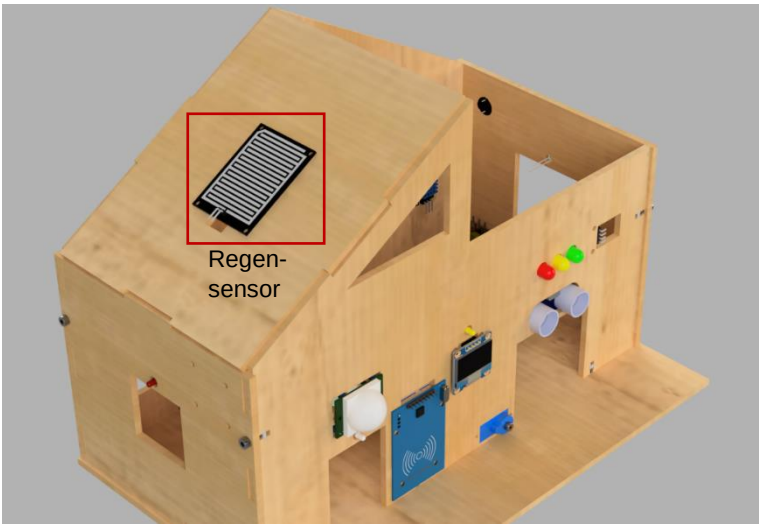
- **Servo-Motoren**: 4x Schrauben. Platzieren Sie die beiden Servo-Motoren von der Innenseite des Hauses nach außen und befestigen Sie diese mit den mitgelieferten Schrauben.
- **PIR-Bewegungsmelder:** 2x Schrauben und 2x Muttern. Montieren Sie ihn durch die Befestigungslöcher oben links und unten rechts.
- **RFID-Sensor:** 2x Bolzen und 2x Muttern. Montieren Sie ihn durch die Befestigungslöcher oben links und unten rechts.
- **OLED-Display**: 2x Schrauben und 2x Muttern. Montieren Sie es durch die Befestigungslöcher oben links und unten rechts.
- **Ampelmodul:** 2x Bolzen und 2x Muttern. Setzen Sie das Modul von innen nach außen ein. Montieren Sie es dann

durch die beiden Befestigungslöcher auf der rechten Seite.

- **Ultrasonic-Sensor:** Setzen Sie ihn von der Innenseite des Hauses nach außen ein. Die Klemmwirkung hält ihn an seinem Platz.
- **LED:** Stecken Sie die LED in das Montageloch; die Spannung hält sie in Position.



Zum Schluss sollte der **Regensensor** auf dem Dach installiert werden. Verwenden Sie 2 Schrauben und 2 Muttern und montieren Sie ihn durch die Befestigungslöcher oben links und unten rechts.



Nachdem alle elektronischen Komponenten und Sensoren an ihrem Platz montiert sind, müssen Sie diese mit den im Paket enthaltenen Jumperkabeln verkabeln. Dieses Verfahren wird im Abschnitt Tutorials erklärt, in dem die verschiedenen GPIO-Pins des Raspberry Pi Pico und die Pins der einzelnen elektronischen Komponenten oder Sensoren erläutert werden.

In der Verpackung finden Sie auch 6 AA-Batterien und einen Batteriehalter. Legen Sie die Batterien in den Batteriehalter und legen Sie ihn vorerst beiseite. In einem späteren Tutorial erfahren Sie, wo Sie ihn platzieren sollten und wie er mit dem MB-102 Power Modul verbunden wird.

Basis Tutorial

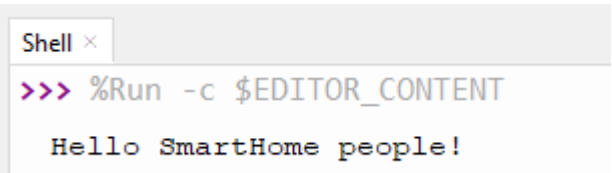
0. “Hallo an alle SmartHome-Fans!”

In diesem grundlegenden Tutorial lernen Sie, wie sich eine einfache Nachricht in Thonny mit Hilfe der Python-Programmierung drucken lässt.

Die Druck() Funktion

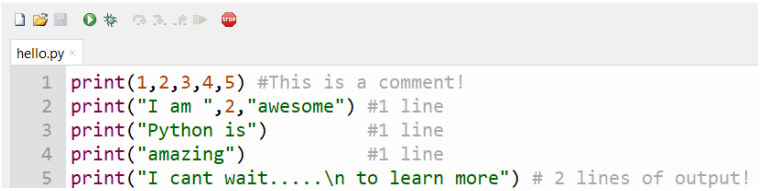
1. Rufen Sie den Thonny-Editor auf, schreiben Sie den folgenden Code und drücken Sie auf das Play-Symbol. Thonny wird Sie bitten, Ihr Programm zuerst zu speichern. Speichern Sie es unter dem Namen hello.py.

2. Überprüfen Sie das Shell-Fenster.

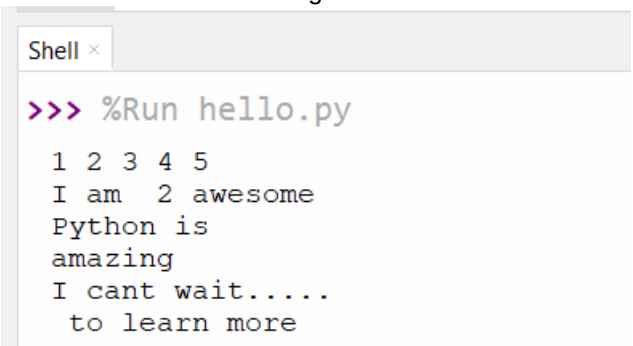


3. Gut gemacht! Sie haben gerade Ihr erstes Python-Programm erstellt.

Die Funktion print ist eine eingebaute Python-Funktion, mit der sich Text in der Shell drucken lässt. Sie kann auch Parameter annehmen. Erstellen Sie ein neues Programm und kopieren Sie den folgenden Code. Drücken Sie dann auf play. Der Text erscheint nun in der Shell.



Wie Sie in dem Shell-Fenster sehen, druckt jede Druckfunktion den Text in eine eigene Zeile. Wenn Sie jedoch das "\n" (Newline-Zeichen) verwenden, können Sie die Zeile in derselben Druckanweisung wechseln.



Übung

Benutzen Sie die Funktion print, um in 3 separaten Zeilen "Goodbye SmartHome-Fans!" mit nur einer print-Anweisung zu drucken.

1. LED Lichter kontrollieren

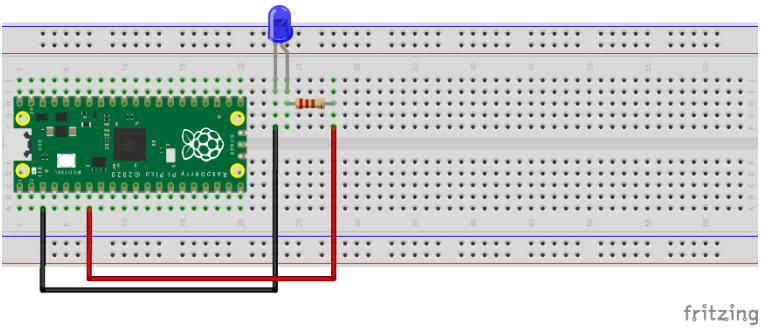
Einleitung

In diesem Tutorial lernen Sie, wie Sie eine LED-Leuchte anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen led.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 2 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x LED (in einer beliebigen Farbe)
- 1 x 220 Ohm-Widerstand

Verdrahtungsplan



- Schließen Sie das längere Ende (+) der LED an einen 220-Ohm-Widerstand an
- Verbinden Sie den Widerstand mit GPIO5 (rotes Kabel)
- Verbinden Sie das kürzere Ende (-) der LED mit einem GND-Pin

Code

MicroPython-Code für das Tutorial:

Thonny - Raspberry Pi Pico :: /led.py @ 9 : 1

File Edit View Run Tools Help

[led.py]

```
1  from machine import Pin #import  Pin from library machine
2  from time import sleep  #import sleep from library time
3
4  led = Pin(5, Pin.OUT)
5
6  while True:
7      led.toggle()
8      sleep(1)
9
```

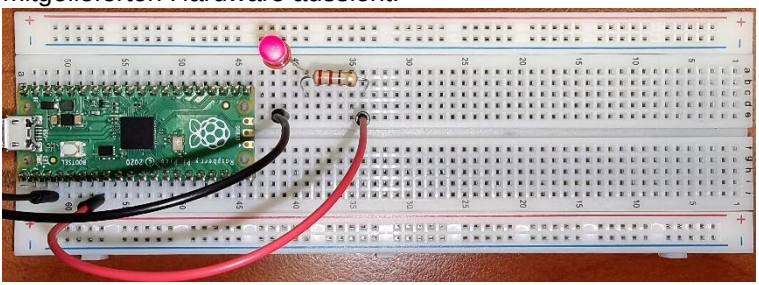
Shell

```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

MicroPython (Raspberry Pi Pico)

Beispielbild:

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:



2. Drucktaste

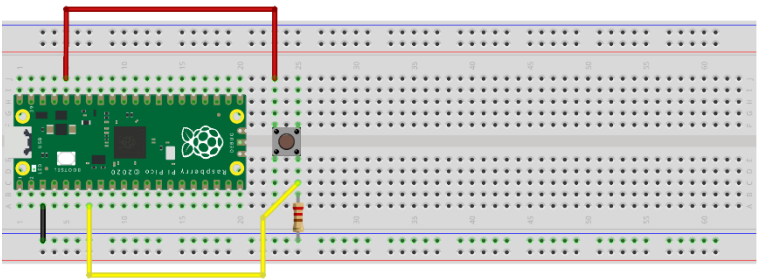
Einleitung

In diesem Tutorial lernen Sie, wie Sie einen Taster anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen button.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 1 x Druckknopf (jede Farbe)
- 3 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x 220 Ohm-Widerstand
- 1 x Tastenkappe

Verdrahtungsplan



- Verbinden Sie die obere linke Tastenseite mit dem 3v3-Pin (rotes Kabel)
- Verbinden Sie die untere rechte Tastenseite mit GPIO5 (gelbes Kabel)
- Verbinden Sie einen GND-Pin mit der (-)-Schiene (schwarzes Kabel)

- den 220-Ohm-Widerstand an die (-)-Schiene und die untere rechte Tastenseite anschließen

Code

MicroPython-Code für das Tutorial:

Thonny - Raspberry Pi Pico :: /button.py @ 1 : 36

File Edit View Run Tools Help

[button.py] *

```
1  from machine import Pin #import Pin from library machine
2  from time import sleep  #import sleep from library time
3
4  button = Pin(5, Pin.IN, Pin.PULL_DOWN)
5
6  while True:
7      if button.value():
8          print("Button is pressed!")
9          sleep(1)
10
```

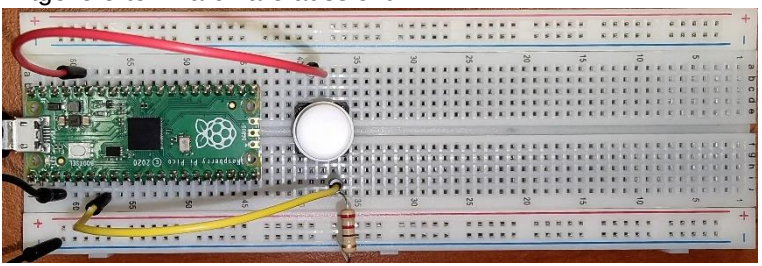
Shell

```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  Button is pressed!
```

MicroPython (Raspberry Pi Pico)

Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:



3. Klingel

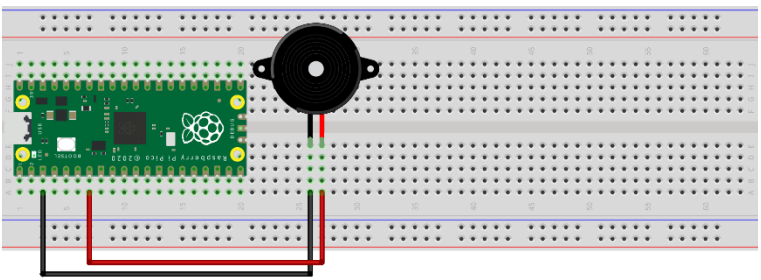
Einleitung

In diesem Tutorial lernen Sie, wie Sie einen Summer anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen buzzer.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 2 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x Klingel

Verdrahtungsplan



- Verbinden Sie das längere Ende (+) des Summers mit dem GPIO5-Pin
- Verbinden Sie das kürzere Ende (-) des Summers mit einem GND-Pin

Code

MicroPython-Code für das Tutorial:

Thonny - Raspberry Pi Pico :: /buzzer.py @ 16 : 1

File Edit View Run Tools Help

[buzzer.py]

```
1  from machine import Pin #import Pin from library machine
2  from time import sleep  #import sleep from library time
3
4  buzzer = Pin(5, Pin.OUT)
5
6  buzzer.value(0)
7  sleep(0.5)
8  buzzer.value(1)
9  sleep(0.5)
10 buzzer.value(0)
11 sleep(0.5)
12 buzzer.value(1)
13 sleep(0.5)
14 buzzer.value(0)
15 sleep(0.5)
16
```

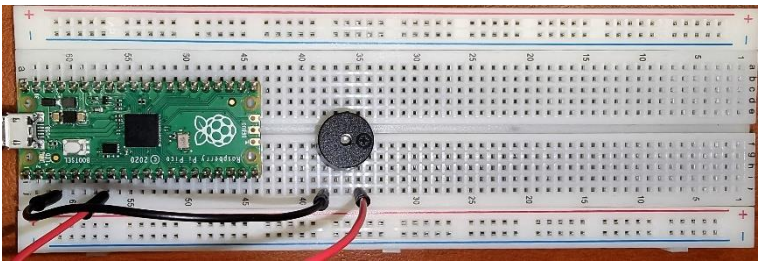
Shell

```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

MicroPython (Raspberry Pi Pico)

Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:



4. Potentiometer

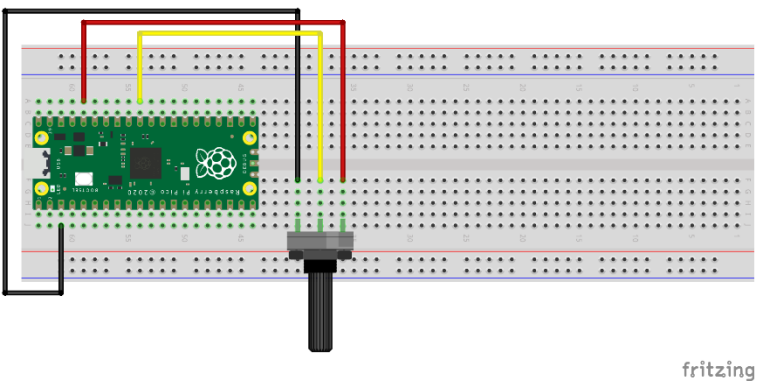
Einleitung

In diesem Tutorial lernen Sie, wie Sie ein Potentiometer anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen pot.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 3 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x Potentiometer

Verdrahtungsplan

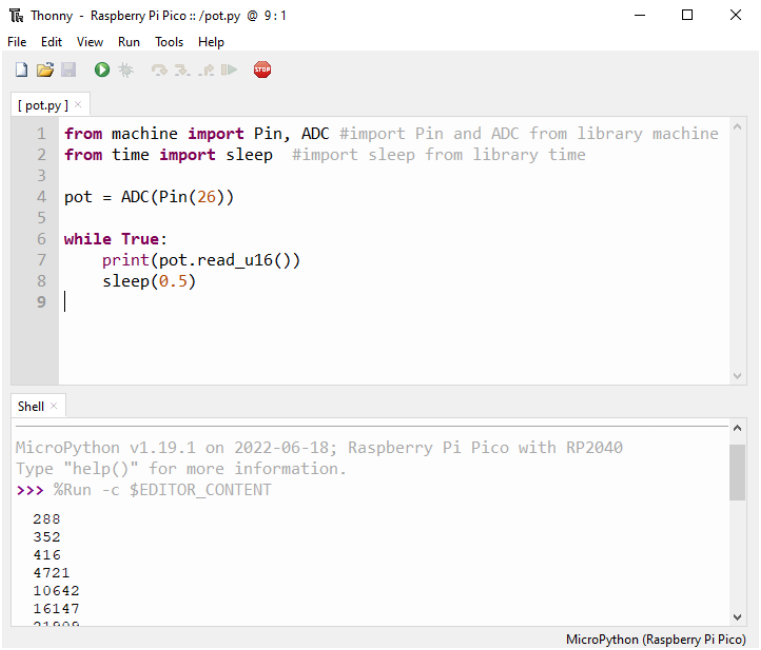


- das schwarze Kabel soll mit dem GND-Pin (Pin 3) verbunden werden
- das gelbe Kabel sollte an GPIO26 ADC Pin angeschlossen werden

- das rote Kabel sollte an den 3V3 Power Pin angeschlossen werden
- das Potentiometer nach links drehen, so dass es ausgeschaltet ist

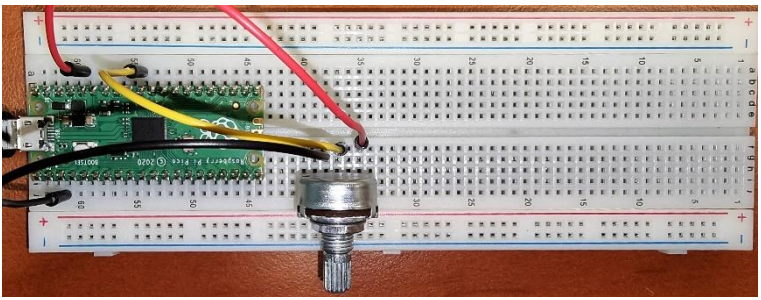
Code

MicroPython-Code für das Tutorial:



Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:



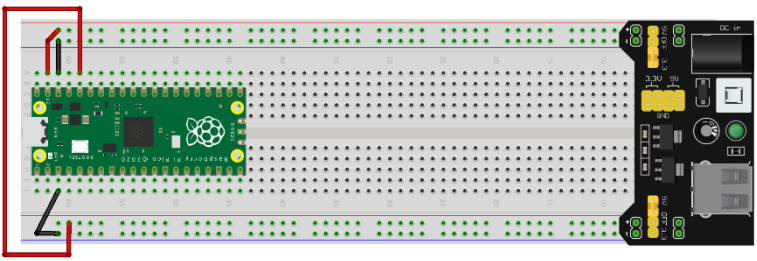
Tutorials für Fortgeschrittene

Für den fortgeschrittenen Teil müssen Änderungen am Breadboard vorgenommen werden. Dies beinhaltet das Hinzufügen weiterer Materialien und deren Anschlussmöglichkeiten.

Benötigte Materialien

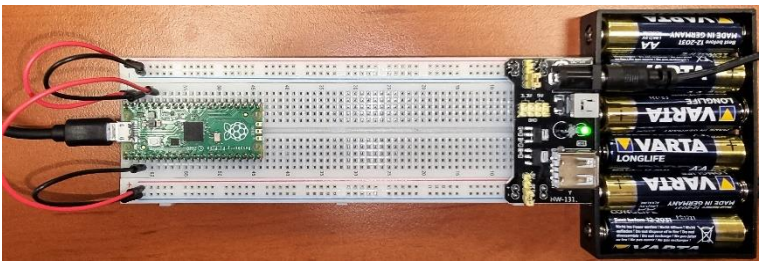
- 1 x 6-AA Batterien
- 1 x MB-102 Stromversorgungsmodul
- 4 x Überbrückungsdrähte von Stecker zu Stecker

Folgen Sie bitte dem Schema unten um die neuen Komponenten anzuschließen:



- dieser Aufbau wird für die weiteren Tutorials verwendet
- Anschlüsse auf der Oberseite: VSYS 5V ((+) rot) und GND ((-) schwarz)
- Anschlüsse auf der Unterseite: 3V3 ((+) rot) und GND ((-) schwarz)

Beispielbild



5. LED Ampel-Modul

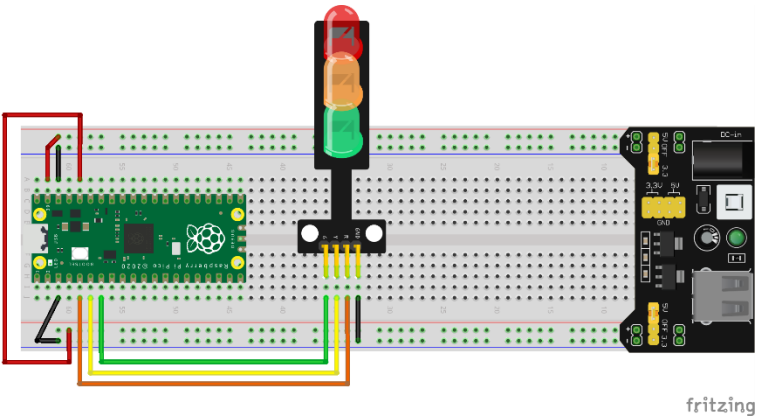
Beschreibung

In diesem Tutorial erfahren Sie, wie Sie das LED-Ampelmodul anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen traffic.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 4 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x LED Ampel-Modul

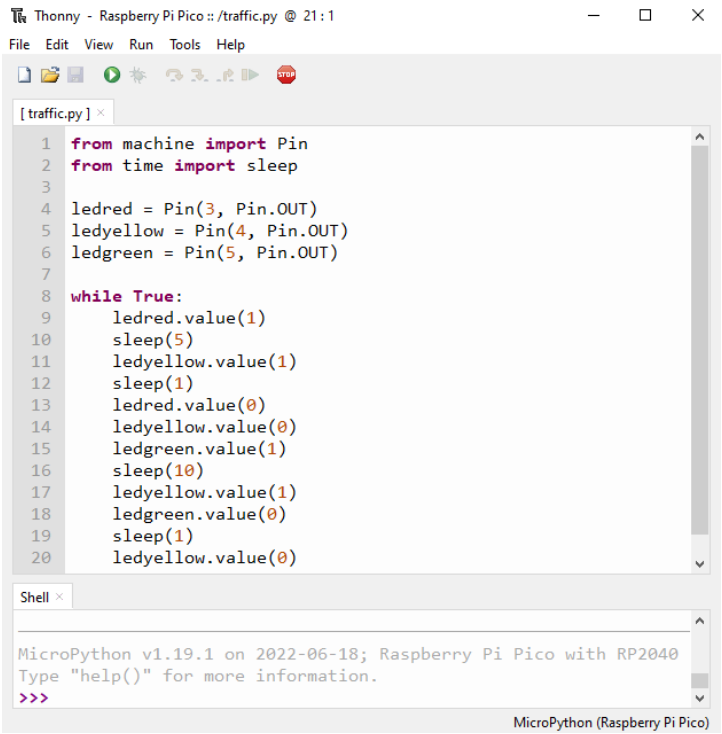
Verdrahtungsplan



- das orangefarbene Kabel (R) ist mit dem GPIO3-Pin verbunden
- das gelbe Kabel (Y) ist mit dem GPIO4-Pin verbunden
- grünes Kabel (G) ist mit dem GPIO5-Pin verbunden
- das schwarze Kabel (GND) ist mit der GND-Schiene ((-) schwarz) verbunden

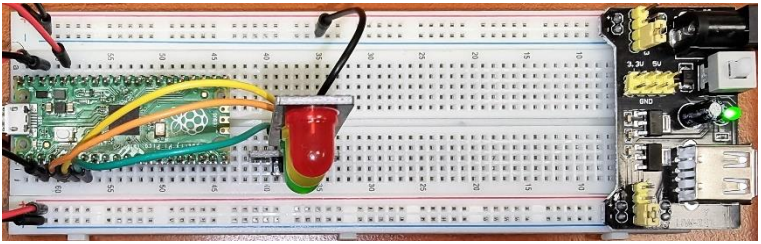
Code

MicroPython-Code für das Tutorial:



Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:



6. LDR Fotoresistor

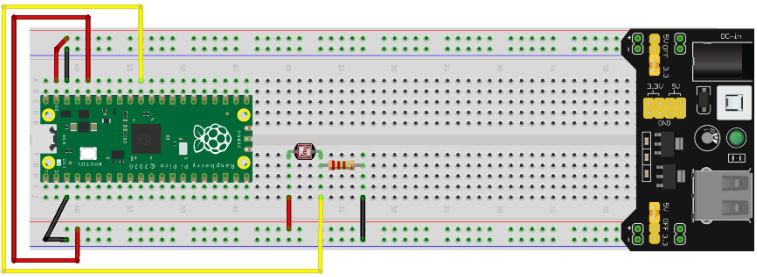
Beschreibung

In diesem Tutorial lernen Sie, wie Sie einen LDR-Fotoresistor anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen ldr.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 3 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x LDR Fotoresistor
- 1 x 220 Ohmresistor

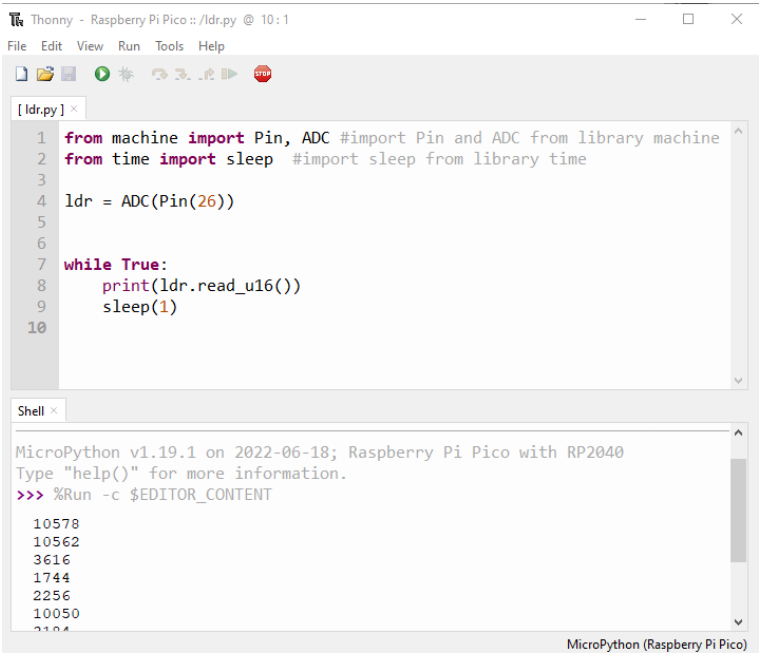
Verdrahtungsplan



- die linke Seite des LDR ist mit der 3V-Schiene ((+) rot) verbunden
- die rechte Seite des LDR ist mit einem 220-Ohm-Widerstand und dem GPIO26 ADC-Pin (gelbes Kabel) verbunden
- die rechte Seite des Widerstands ist mit der GND-Schiene ((-) schwarz) verbunden

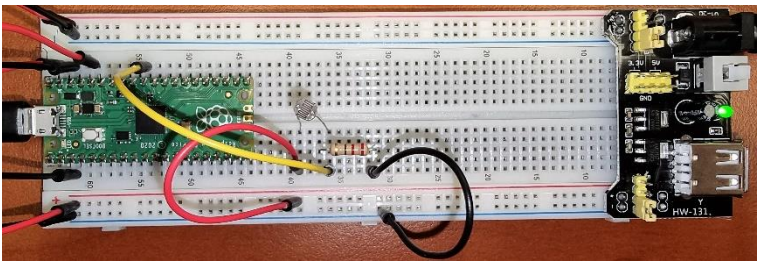
Code

MicroPython-Code für das Tutorial:



Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:



7. DC Motor (Kleiner Ventilator)

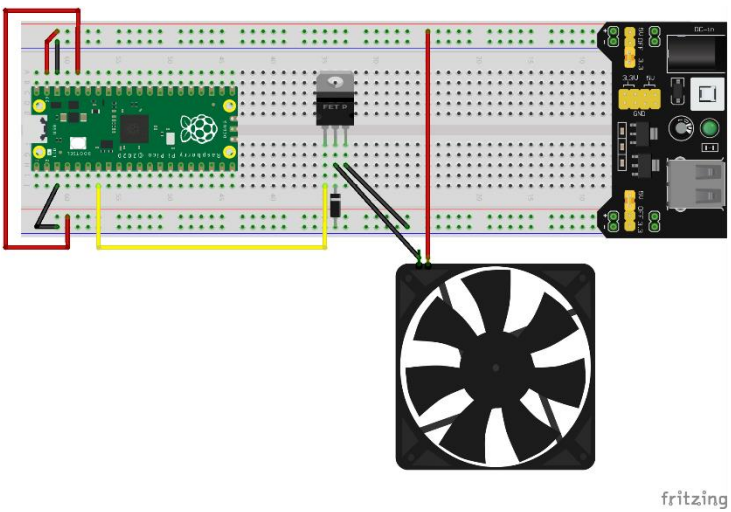
Beschreibung

In diesem Tutorial lernen Sie, wie Sie einen kleinen Gleichstrommotor anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen fan.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

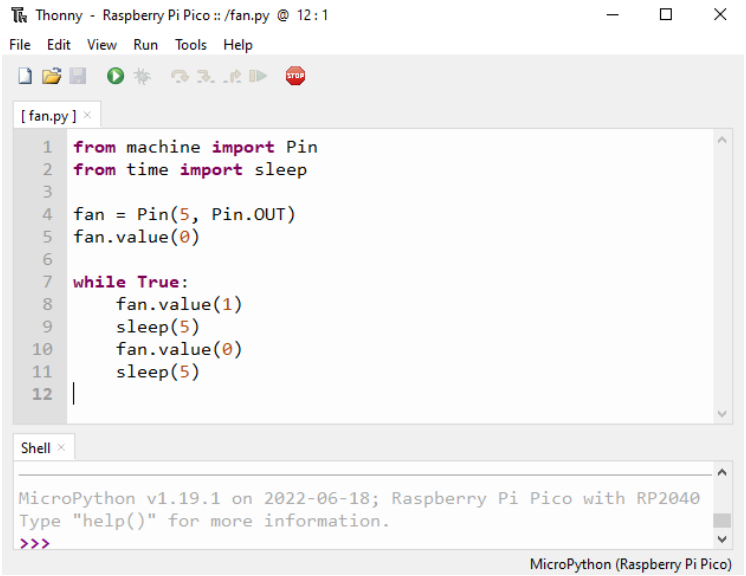
- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 1 x Diode
- 3 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x DC-Motor (Kleiner Ventilator)
- 1 x TIP-120 Transistor

Verdrahtungsplan



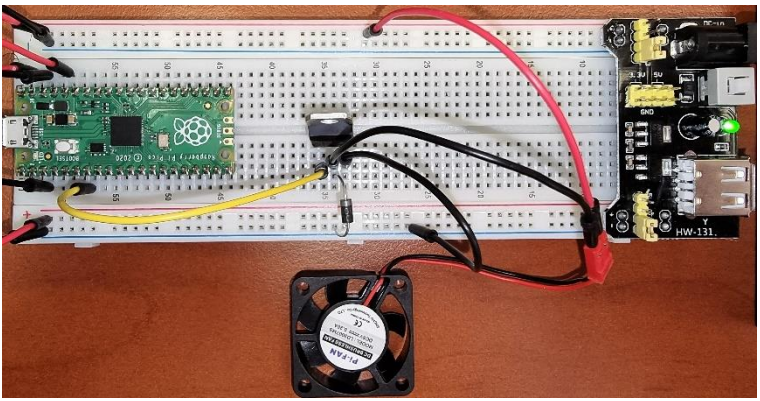
Code

MicroPython-Code für das Tutorial:



Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:


HW-131.

8. SG-90 Servo Motor

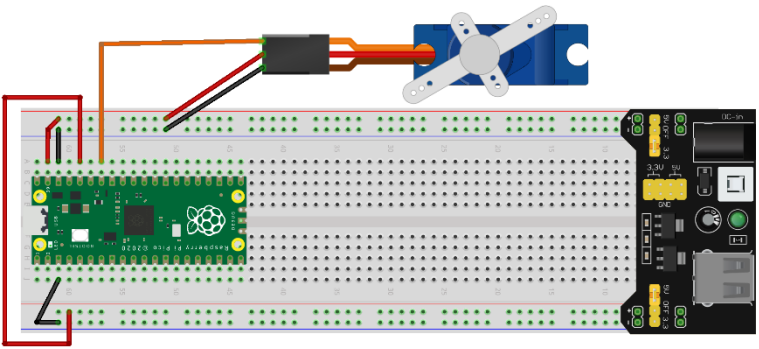
Beschreibung

In diesem Tutorial lernen Sie, wie Sie einen Servomotor anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen servo.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 3 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x SG-90 Servo Motor

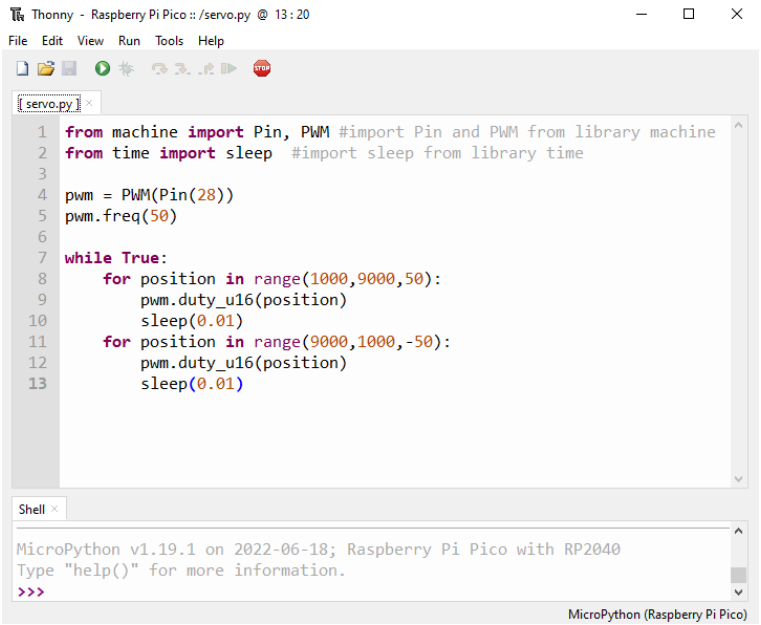
Verdrahtungsplan



- das rote Kabel wird mit der 5V-Schiene verbunden (+)
- das schwarz/braune Kabel wird mit der GND-Schiene (-) verbunden
- das orange Kabel wird mit GPIO28 ADC Pin verbunden

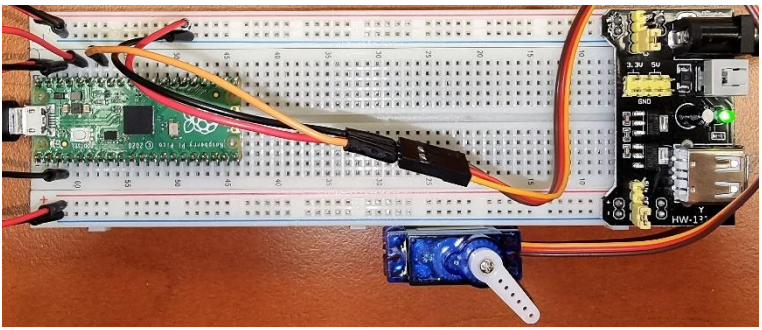
Code

MicroPython-Code für das Tutorial:



Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:



9. OLED I2C SSD1306 Display

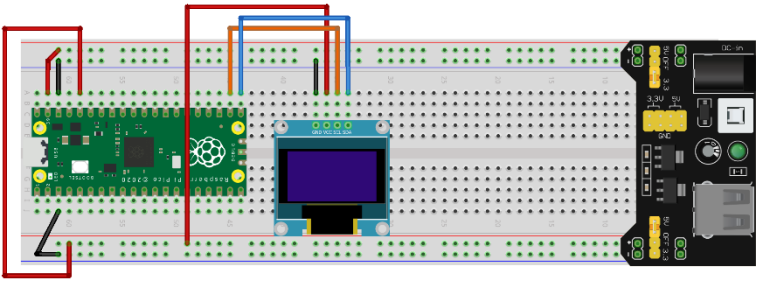
Beschreibung

In diesem Tutorial lernen Sie, wie Sie das I2C ICC OLED-Display anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen oled.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 4 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x OLED I2C SSD1306 Display

Verdrahtungsplan

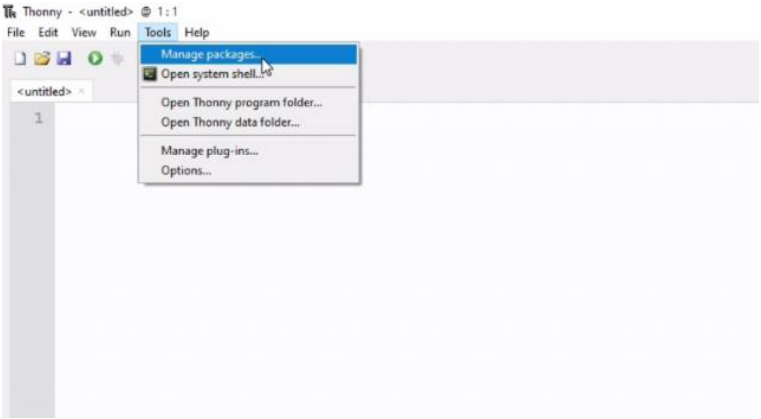


- das rote Kabel wird mit der 3v3-Schiene verbunden (+)
- das schwarze Kabel wird mit der GND-Schiene verbunden (-)
- das orange Kabel wird mit GPIO17 I2C0 SCL Pin verbunden
- das blaue Kabel wird mit dem GPIO26 I2C0 SDA-Pin verbunden

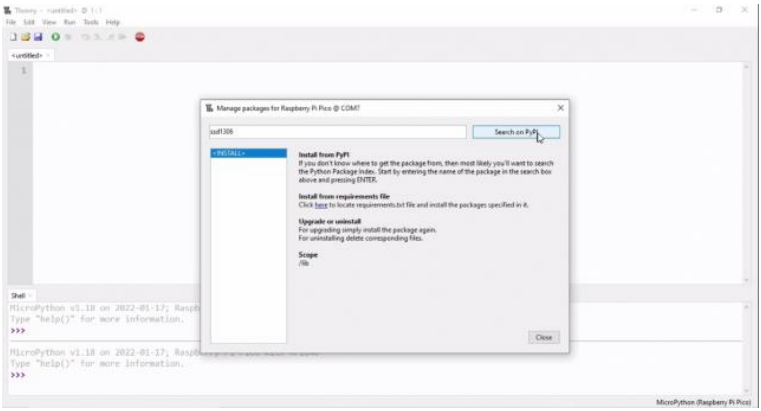
Code

Bevor Sie mit der Programmierung des OLED-Displays beginnen können, müssen Sie zunächst das SSD1306-Paket zu Ihrem RPi Pico hinzufügen. Um das zu tun, folgen Sie bitte den nächsten Schritten:

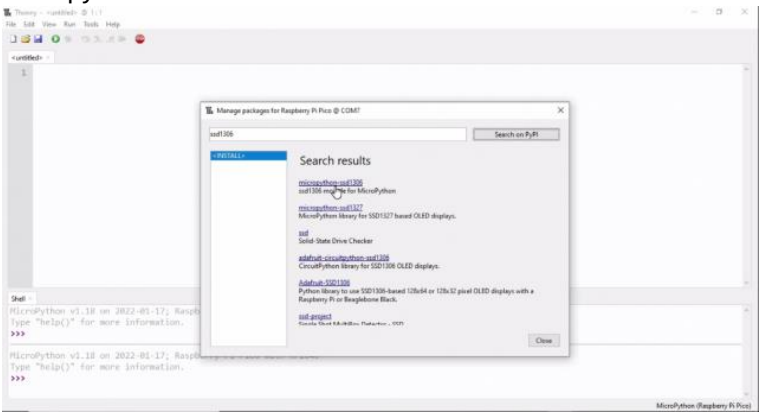
1. Öffnen Sie Thonny und gehen Sie zu **Tools → Manage packages…**



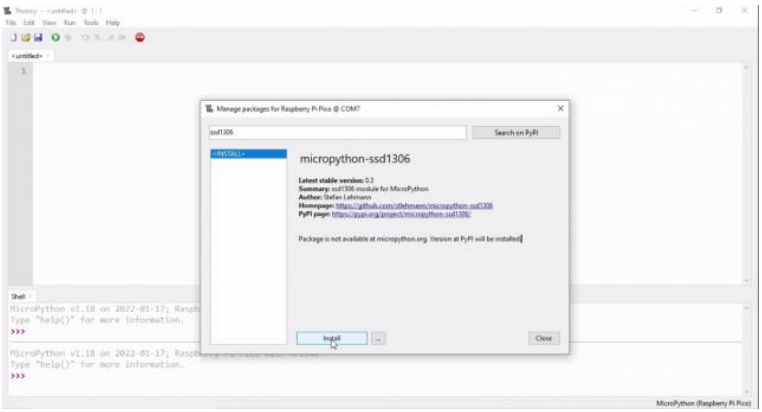
1. Geben Sie nun „SSD1306“ ein und klicken Sie auf *Search.*



2. Sobald die Suche beendet ist, klicken Sie auf micropython-ssd1306.



2. Klicken Sie nun auf “Install”.



3. Warten Sie auf die Installation des Programms und klicken Sie dann auf Schließen.

Jetzt können Sie mit der Programmierung des OLED-Displays fortfahren.

MicroPython-Code für das Tutorial:

Thonny - Raspberry Pi Pico :: /oled.py @ 20 : 1

File Edit View Run Tools Help

[oled.py]

```
1  from machine import Pin, I2C
2  import ssd1306
3
4  WIDTH =128
5  HEIGHT = 64
6
7  PIN_SCL = 17
8  PIN_SDA = 16
9
10 i2c = I2C(0,scl=Pin(PIN_SCL),sda=Pin(PIN_SDA))
11 oled = ssd1306.SSD1306_I2C(WIDTH,HEIGHT,i2c)
12 oled.fill(0)
13
14 oled.text("OLED", 0, 0)
15 oled.text("Display", 0, 10)
16 oled.text("Hello SmartHome", 0, 30)
17 oled.text("  People!", 0, 40)
18
19 oled.show()
20
```

Shell

```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

MicroPython (Raspberry Pi Pico)

Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:

10. RFID Reader RC522

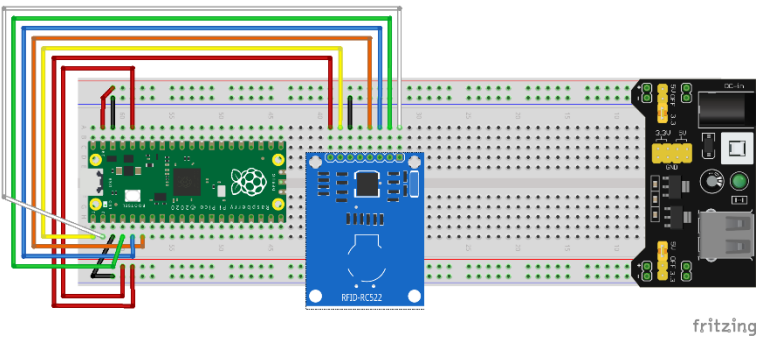
Beschreibung

In diesem Tutorial lernen Sie, wie Sie ein RFID-Lesemodul anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen rfid.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen..

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 7 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x RFID RC522 module
- 1 x RFID Tag

Verdrahtungsplan



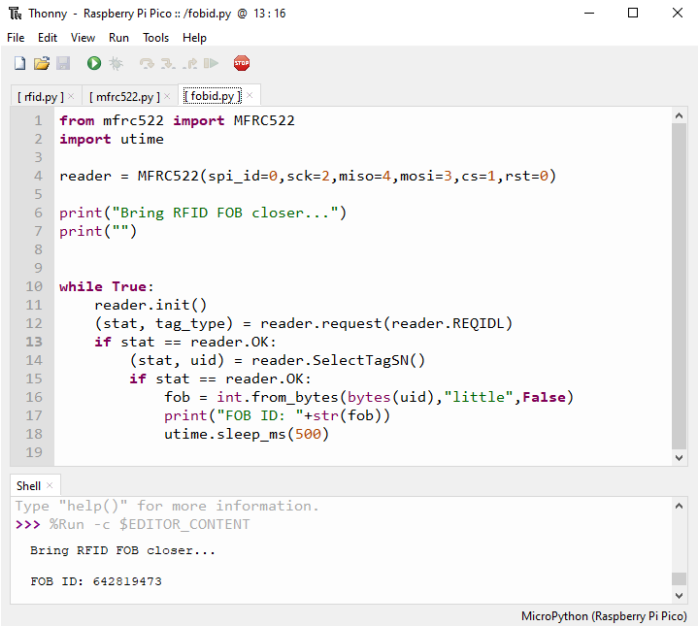
- 3v3 (rotes Kabel) wird mit der 3v3-Schiene (+) verbunden
- GND (schwarzes Kabel) wird mit der GND-Schiene (-) verbunden
- RST (gelbes Kabel) wird mit dem GPIO0-Pin verbunden
- SDA (weißes Kabel) wird mit dem GPIO1-Pin verbunden

- SCK (grünes Kabel) wird mit dem GPIO2-Pin verbunden
- MOSI (blaues Kabel) wird mit dem GPIO3-Pin verbunden
- MISO (orangefarbenes Kabel) wird mit dem GPIO4-Pin verbunden

Code

Ähnlich wie bei der OLED-Anzeige wird eine zusätzliche Bibliothek benötigt, die das RFID-Modul steuert, nämlich die MFRC522-Bibliothek. Sie können diese von https://github.com/pimylifeup/MFRC522-python/blob/master/mfrc522/MFRC522.py herunterladen. Laden Sie die Datei herunter und öffnen Sie sie in Thonny Python. Klicken Sie dann auf Datei → Speichern unter... wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen mfrc522.py.

Identifizieren Sie nun die ID des Anhängers, damit das RFID-Lesegerät funktioniert. Dazu müssen Sie ein Programm erstellen, das den RFID-Anhänger liest und seine ID angibt. Sehen Sie sich das Programm unten an:



Klicken Sie auf Play und scannen Sie den Anhänger. Dadurch erhalten Sie seine ID. Zurück zum Programm rfid.py. Ändern Sie die Zahl in Zeile 20, damit sie mit der ID Ihres Anhängers übereinstimmt. Klicken Sie dann auf Speichern und führen Sie Ihr Programm aus.MicroPython-Code für das Tutorial:

Thonny - Raspberry Pi Pico :: /rfid.py @ 11 : 1

File Edit View Run Tools Help

[rfid.py] [mfrc522.py] [fobid.py]

```
from machine import Pin
from mfrc522 import MFRC522
import utime
import time

reader = MFRC522(spi_id=0,sck=2,miso=4,mosi=3,cs=1,rst=0)

print("Bring RFID FOB Closer...")
print("")

while True:
    reader.init()
    (stat, tag_type) = reader.request(reader.REQIDL)
    if stat == reader.OK:
        (stat, uid) = reader.SelectTagSN()
        if stat == reader.OK:
            fob = int.from_bytes(bytes(uid),"little",False)

            if fob == 642819473:
                print("Fob ID: "+ str(fob))
                print("Fob is accepted")
                time.sleep(1)
            else:
                print("Fob is not accepted")
```

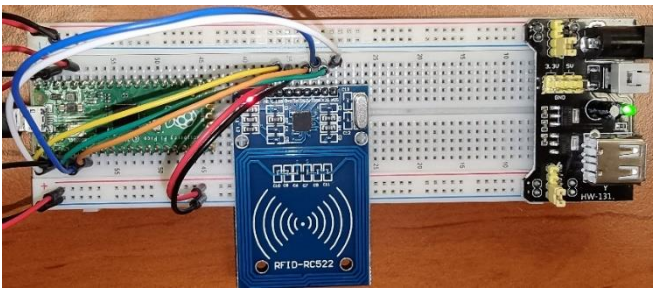
Shell

```
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  Bring RFID TAG Closer...

  Fob ID: 642819473
  Fob is accepted
```

MicroPython (Raspberry Pi Pico)

Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:



Tutorials mit Sensoren

11. Regensensor

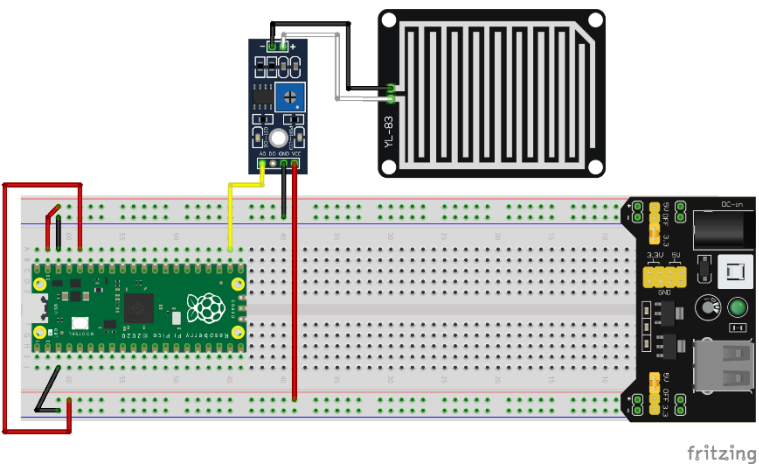
Beschreibung

In diesem Tutorial lernen Sie, wie Sie den Regensensor anschließen und steuern. Öffnen Sie Thonny Python, gehen Sie dann auf Datei -> Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen raindrop.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 3 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x Raindrop sensor

Verdrahtungsplan



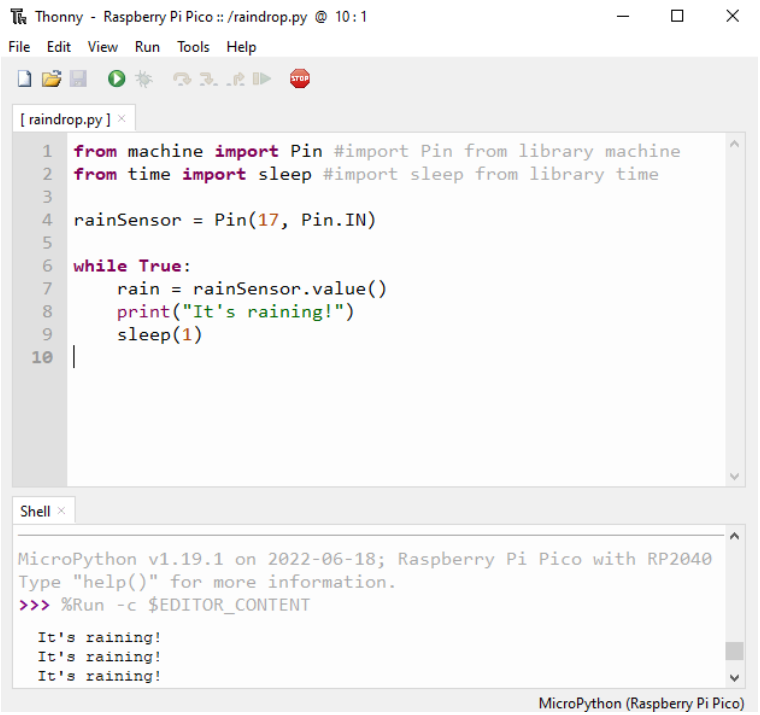
- 3v3V (rotes Kabel) ist mit der 3v3V-Schiene (+) verbunden

- GND (schwarzes Kabel) ist mit der GND-Schiene (-) verbunden

- DO (orangefarbenes Kabel) ist mit dem GPIO17-Pin verbunden

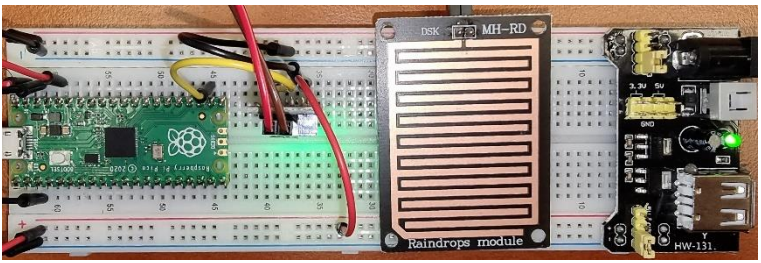
Code

MicroPython-Code für das Tutorial:



Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:


DSK
MH-RD
Raindrops module
HW-131

12. HC-SR04 Ultrasonic Sensor

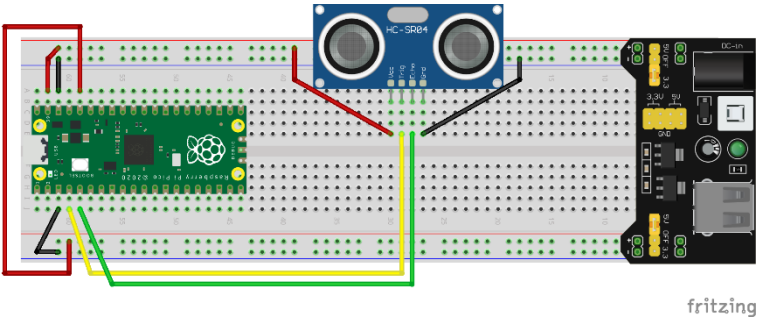
Beschreibung

In diesem Tutorial lernen Sie, wie Sie den HC-SR04 Ultraschallsensor anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei -> Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen ultrasonic.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 4 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x HC-SR04 Ultrasonic sensor

Verdrahtungsplan

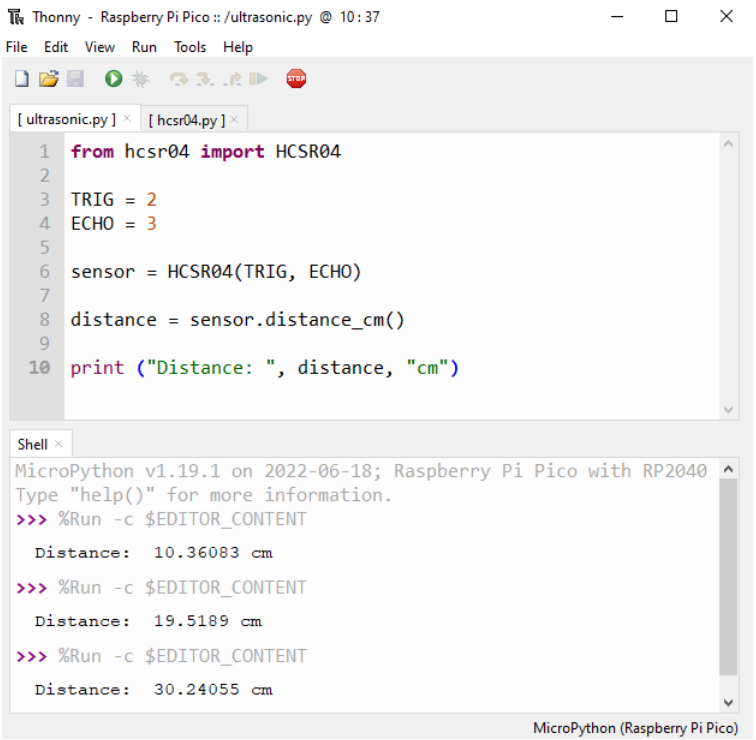


- VCC (rotes Kabel) ist mit der 5V-Schiene verbunden (+)
- GND (schwarzes Kabel) ist mit der GND-Schiene verbunden (-)
- TRIG (orangefarbenes Kabel) ist mit dem GPIO2-Pin verbunden
- ECHO (grünes Kabel) ist mit dem GPIO3-Pin verbunden

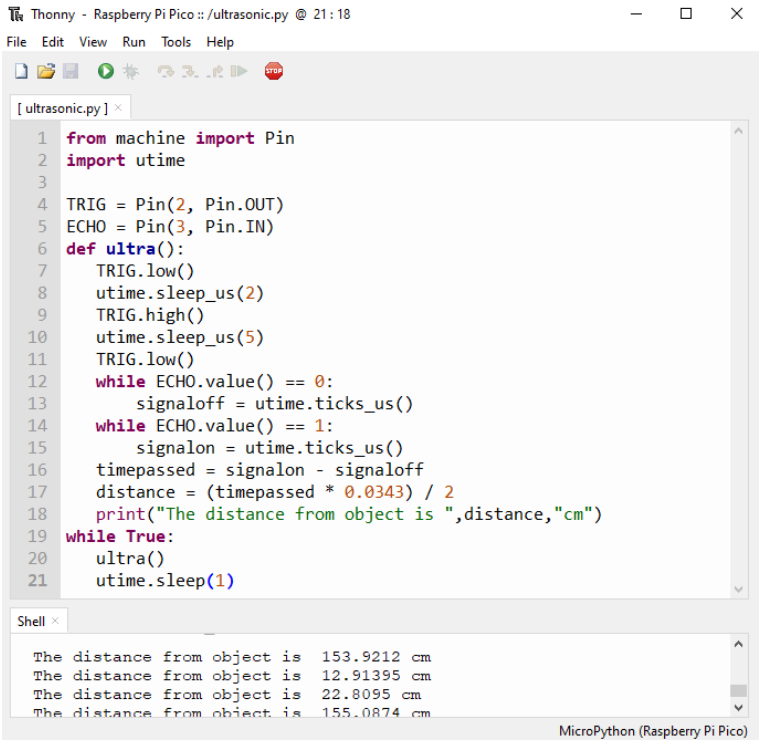
Code

Um den HC-SR04-Ultrasonic Sensor zu verwenden, können wir ein eigenes Programm entwickeln oder eine der online verfügbaren Bibliotheken verwenden, z. B. auf Github. Wenn Sie sich für den Download der Bibliothek hcsr04.py entscheiden, sollten Sie diese in Ihrem Pico unter demselben Namen speichern.

MicroPython-Code unter Verwendung einer vorhandenen Bibliothek:

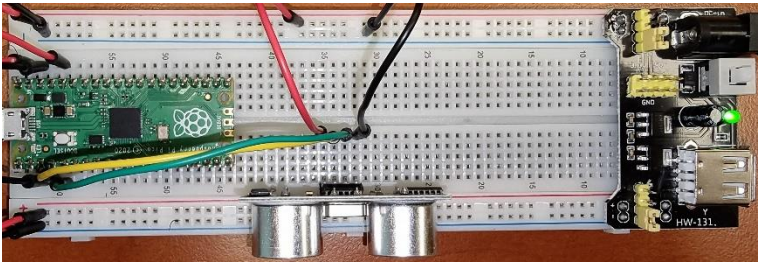


MicroPython-Code ohne vorhandene Bibliothek:



Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:


GND
HW-131

13. PIR Bewegungssensor

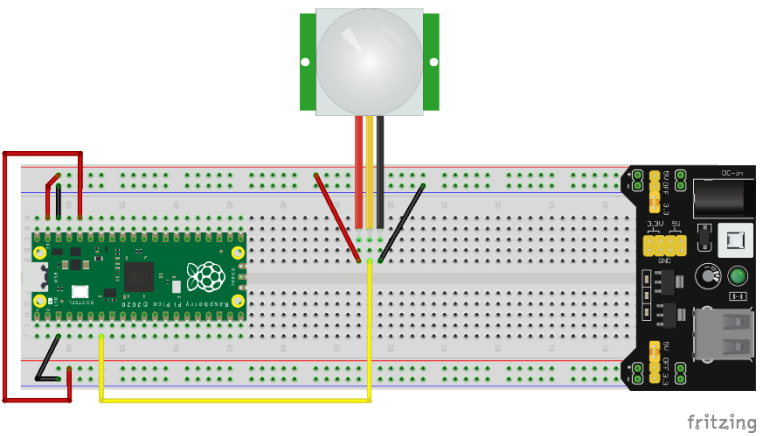
Beschreibung

In diesem Tutorial erfahren Sie, wie Sie den PIR-Bewegungssensor anschließen und steuern können. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen motion.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 3 x Überbrückungsdrähte (männlich/weiblich)
- 1 x PIR Bewegunssensor

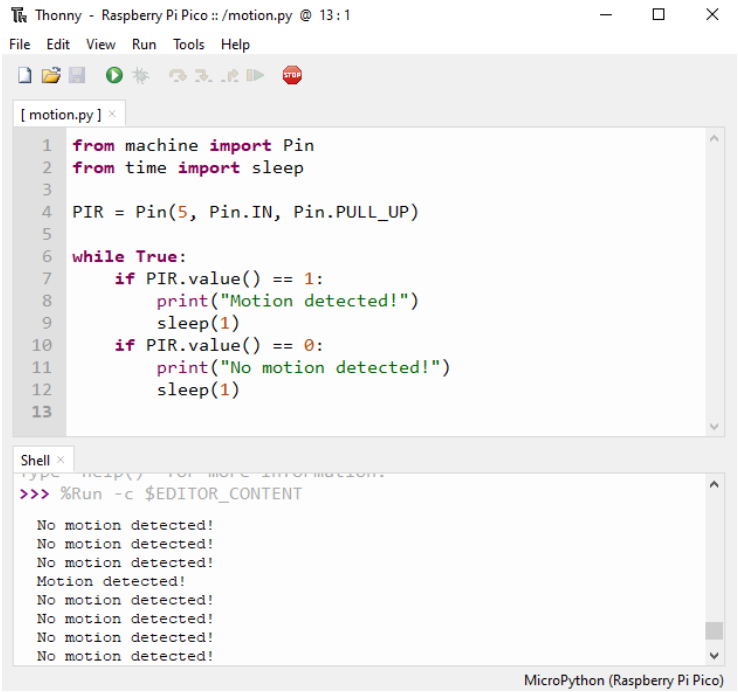
Verdrahtungsplan



- VCC (rotes Kabel) ist mit der 5V-Schiene verbunden (+)
- GND (schwarzes Kabel) ist mit der GND-Schiene verbunden (-)
- OUT (oranges Kabel) ist mit dem GPIO5-Pin verbunden

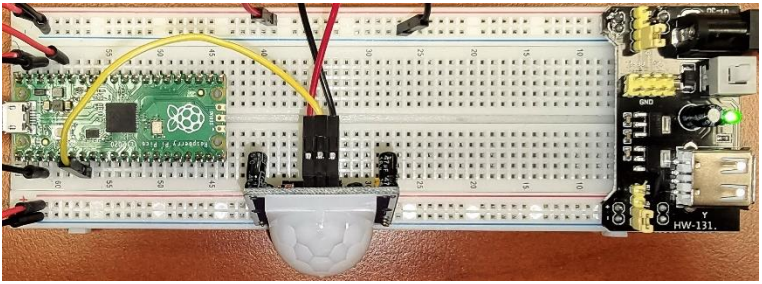
Code

MicroPython-Code für das Tutorial:



Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:


GND
HW-131

14. DHT11 Sensor

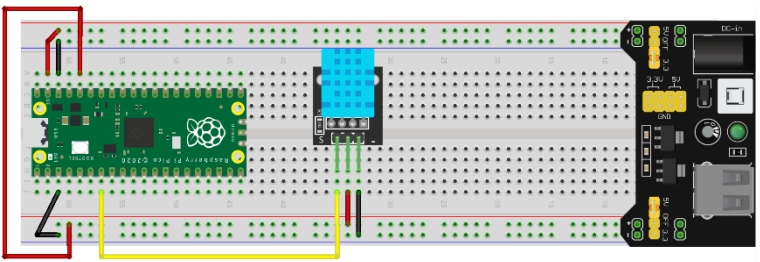
Beschreibung

In diesem Tutorial lernen Sie, wie Sie den Temperatur- und Feuchtigkeitssensor DHT11 anschließen und steuern. Öffnen Sie Thonny Python, gehen Sie dann auf Datei → Speichern unter..., wählen Sie Raspberry Pi Pico und speichern Sie Ihre Datei unter dem Namen dht11.py. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 3 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x DHT11 temperature sensor

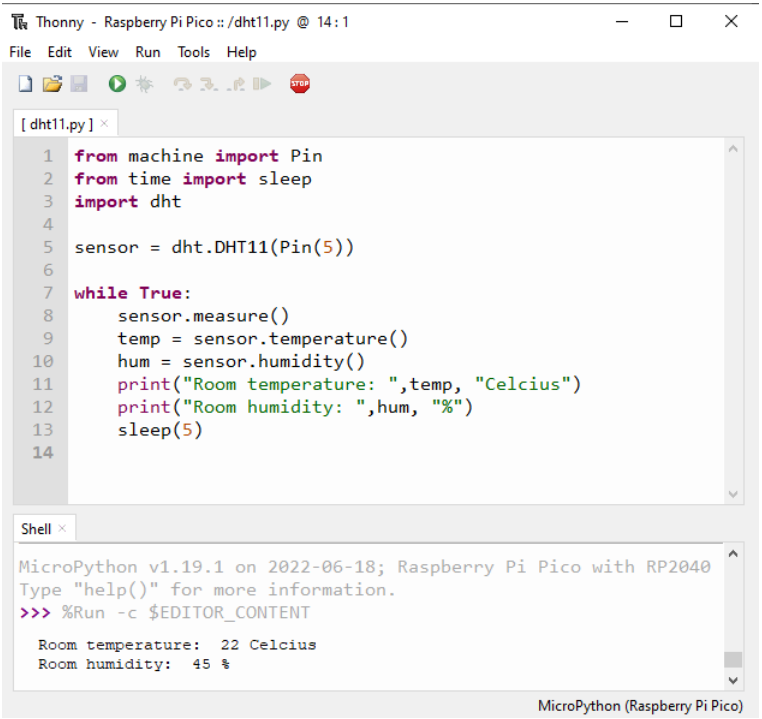
Verdrahtungsplan



- VCC (rotes Kabel) ist mit der 3v3-Schiene verbunden (+)
- GND (schwarzes Kabel) ist mit der GND-Schiene verbunden (-)
- S (grünes Kabel) ist mit dem GPIO5-Pin verbunden

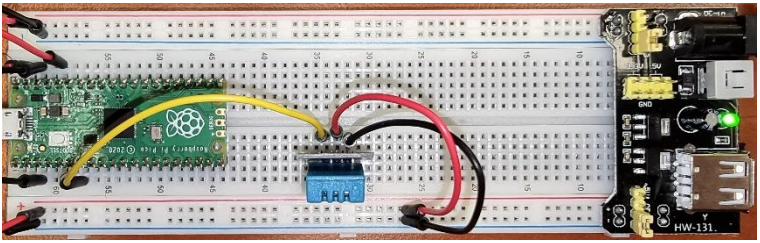
Code

MicroPython-Code für das Tutorial:



Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:


GND
HW-131

15. Flammensensor

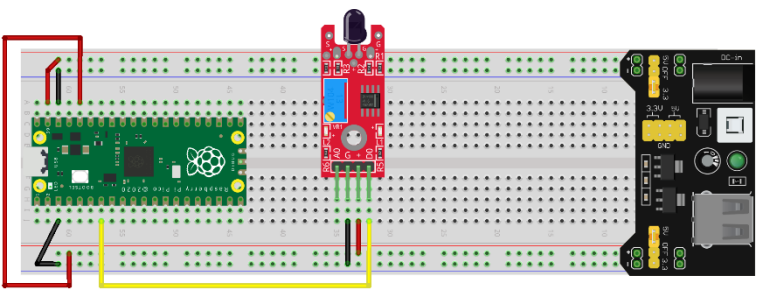
Beschreibung

In diesem Tutorial erfahren Sie, wie Sie den KY-026 Flammensensor anschließen und kontrollieren. Öffnen Sie Thonny Python, gehen Sie zu Datei → Speichern als…, wählen Sie Raspberry Pi Pico und speichern Sie die Datei unter dem Namen `flame.py`. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 3 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x KY-026 flame sensor

Verdrahtungsplan



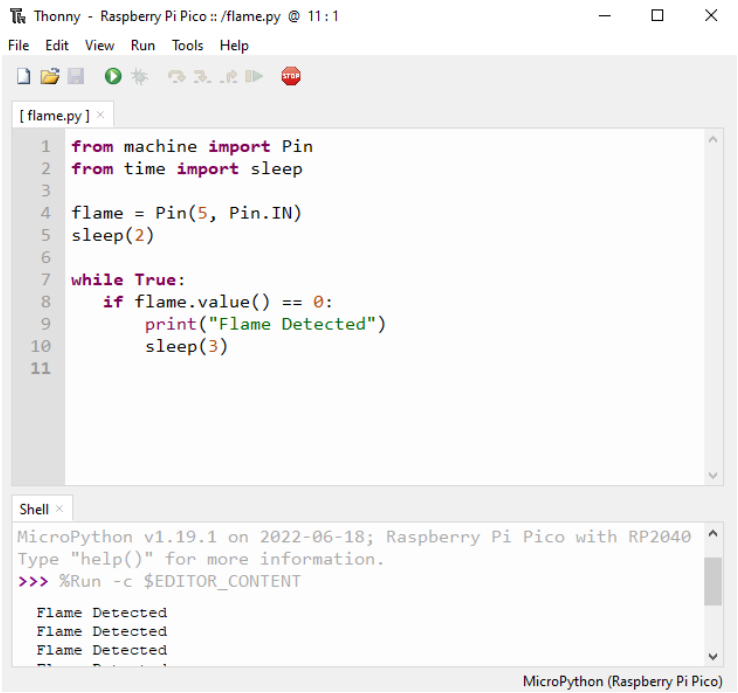
– VCC (rotes Kabel) wird mit der 5V Schiene (+) verbunden

– GND (schwarzes Kabel) wird mit der GND Schiene (-) verbunden

– DO (grünes Kabel) wird mit dem GPIO5 pin verbunden

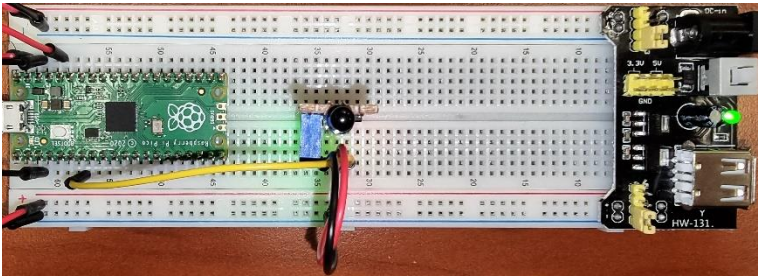
Code

MicroPython-Code für das Tutorial:



Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:


3.3V
5V
GND
HW-131

16. MQ-135 Gaserkennungssensor

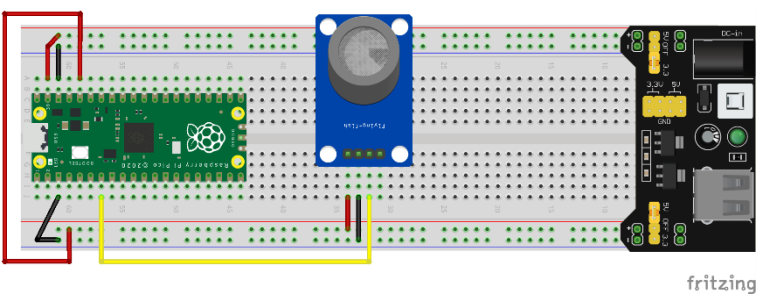
Beschreibung

In diesem Tutorial erfahren Sie, wie Sie einen MQ-135 Gasmessungssensor anschließen und programmieren können. Öffnen Sie Thonny Python, gehen Sie zu Datei → Speichern als…, wählen Sie Raspberry Pi Pico und speicern Sie die Datei unter `gas.py`. Schließen Sie nun die Elektronik an und schreiben Sie Ihr Programm. Folgen Sie dazu den Anweisungen.

Benötigte Materialien

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Micro-USB-Kabel
- 3 x Überbrückungsdrähte von Stecker zu Stecker
- 1 x MQ-135 Gaserkennungssensor

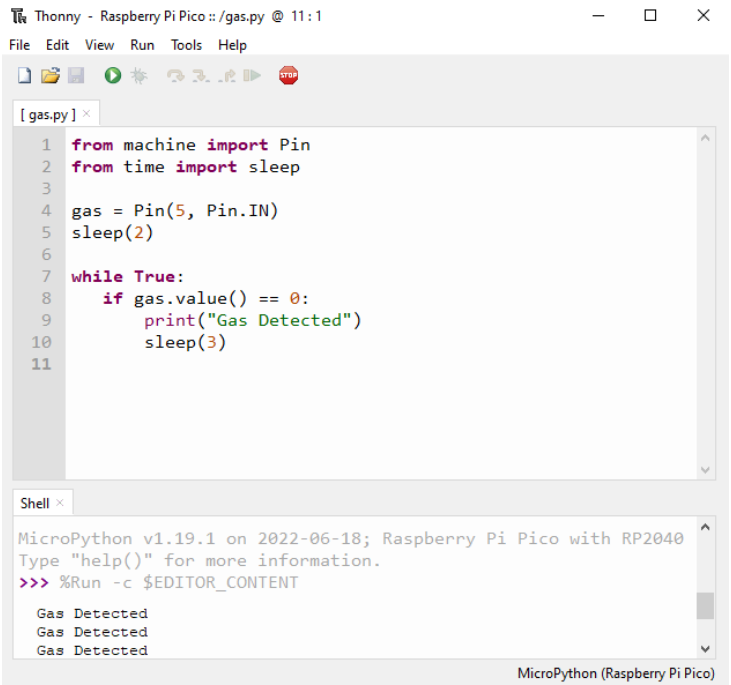
Verdrahtungsplan



- VCC (rotes Kabel) ist mit der 3v3-Schiene verbunden (+)
- GND (schwarzes Kabel) ist mit der GND-Schiene verbunden (-)
- DO/OUT (gelbes Kabel) ist mit dem GPIO5-Pin verbunden

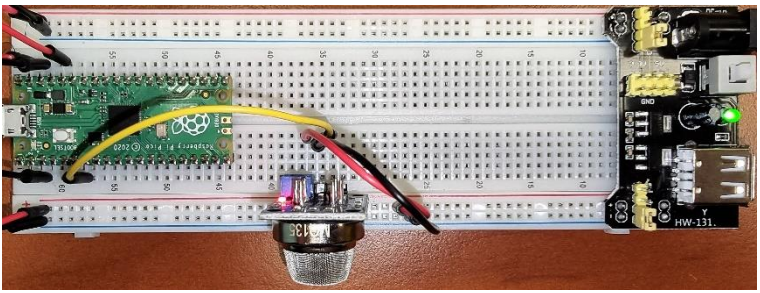
Code

MicroPython-Code für das Tutorial:



Beispielbild

Das Bild zeigt, wie das Tutorial bei Verwendung der mitgelieferten Hardware aussieht:


HW-131

Anhang: MicroPython-Zusammenfassung

Digitaler Output		
Abrufen der Pin-Klasse	`from machine import Pin`	
Initialisierung des digitalen Ausgangsobjekts	`led = Pin(pin_value, Pin.OUT)`	`pin_value` von 0 bis 40
Digitalen Ausgang öffnen (3,3V-Ausgang)	`led.value(1)`	ON (AN)
Digitalen Ausgang schließen (0V-Ausgang)	`led.value(0)`	OFF (AUS)

Digitaler Input		
Abrufen der Pin-Klasse	`from machine import Pin`	
Initialisierung des digitalen Ausgangsobjekts	`button = Pin(pin_value, Pin.IN)`	`pin_value` von 0 bis 40
	`button = Pin(pin_value), Pin.IN, Pin.PULL_UP)`	Activation of PULL UP resistance
	`button = Pin(pin_value), Pin.IN, Pin.PULL_DOWN)`	Activation of PULL DOWN resistance
Lesen des Eingangs	`value = button.value(1)`	Return value could be 0 if pin is at 0V, or 1 if pin is at 3.3V

Analoger Output (Pulse Width Modulation - PWM)		
Abrufen der PWM-Klasse	`from machine import PWM`	
Initialisierung des analogen Ausgansobjekts	`led = PWM(Pin(pin_value), frequency)`	`pin_value` from 0 to 40 `frequency` in HZ, from 0 to 78125
Lesen des Eingangs	`led.duty(duty_cycle)`	`duty_cycle` from 0 to 1023 (0V output to 3.3V output respectively)

Analoger Input		
Abrufen der ADC-Klasse	`from machine import ADC`	
Initialisierung des analogen Eingangs	`pot = ADC(Pin(pin_value))`	`pin_value` can be GPIO26, GPIO27 and GPIO28
Angabe, bei welcher Spannung der Eingang seinen maximalen Wert abgibt (bei ESP32 normalerweise 3.3V)	`pot.atten(ADC.ATTN_11DB)`	**ADC.ATTN_0DB:** full range voltage: 1.2 V **ADC.ATTN_2_5DB:** full range voltage: 1.5 V **ADC.ATTN_6DB**: full range voltage: 2.0 V **ADC.ATTN_11DB**: full range voltage: 3.3 V
Erklärung des Eingangswertebereichs (Standard 12bit)	`pot.width(ADC.WIDTH_10BIT)`	**ADC.WIDTH_9BIT**: range 0 to 511 **ADC.WIDTH_10BIT**: range 0 to 1023 **ADC.WIDTH_11BIT**: range 0 to 2047 **ADC.WIDTH_12BIT**: range 0 to 4095
Lesen des Eingangs	`value = pot.readl()`	`value` is an integer from 0 to the maximum of the range specified by the ADC.WIDTH_#BIT statement (see previous)

Die Zeitbibliothek		
Abrufen der Schlaffunktion	`from machine import sleep`	
Nutzung der Schlaffunktion	`sleep(sec)`	`sec` is the number of seconds for which the program will be delayed
Aufrufen der Zeitklasse	`from machine import time`	
Verwendung der Zeitfunktion	`current_time = time()`	The `current_time` variable will take a numeric value, equal to the number of seconds since the last reset on the board.

Struktur der If-Anweisungen

`if <expr1>:`
`  <statement1>`
`elif <expr2>:`
`  <statement2>`
`elif <expr3>:`
`  <statement3>`
`  (...)`
`else:`
`  <statementn>`
	<expr#>: the control condition that must return True or False
	
	<statement#>: set of commands to be executed when the adjacent condition is satisfied
	
	<expr#> (e.g. the set <statement2> is executed when <expr2> is satisfied)
	
	<statementn>: set of instructions executed when none of the <expr#> conditions are satisfied

While-Schleifenstruktur	
`while <expr>:` `  <statement(s)>`	<expr>: the control condition that must return True or False <statement#>: set of commands to be executed as long as <expr> condition is satisfied

Struktur der For-Schleife	
`for <var> in <iterable>:` `  <statement(s)>`	**<iterable>**: a collection of objects, for example a list containing numbers, alphanumerics, etc. **<var>**: a variable to which the value of the next item in the collection <iterable> is assigned. **<stametement(s)>**: a set of instructions executed at each iteration
`for <var> in range(<start>, <end>, <step>):` `  <statement(s)>`	**range(<start>, <end>, <step>)**: function that returns a sequence of numbers from <start> to <end>-1, with a <step> difference between two consecutive numbers (<start> and <step> parameters are optional). **<var>**: variable to which the value of the next element of the sequence produced by range is assigned.

		<statement(s)>: a set of instructions executed in each iteration
Verschiedenes		
DHT11	import dht	Importieren der DHT Bibliothek
	sensor = dht.DHT11(Pin(pin_number))	Initialisierung der Sensorvariable mit der zugehörigen pin_number.
	sensor.measure()	Aktualisieren von Sensordaten
	temp = sensor.temperature()	Speichern des aktuellen Temperaturwerts
	hum = sensor.humidity()	Speichern des aktuellen Luftfeuchtigkeitswerts
OLED DISPLAY	from machine import I2C	Importieren der I2C-Bibliothek
	import ssd1306	Importieren der ssd1306-Bibliothek
	i2c = I2C(-1, scl=Pin(1), sda=Pin(0))	Initialisierung der i2c-Variablen an den SCL- und SDA-Pins des Pico
	oled_width = 128 oled_height = 64 oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)	Initialisierung des Bildschirms
	oled.text('Hello, World 1!', 0, 0) oled.text('Hello, World 2!', 0, 10) oled.text('Hello, World 3!', 0, 20)	Speichern von Meldungen im Bildschirmpuffer
	oled.show()	Anzeigen von Meldungen (notwendig, um im Bildschirmpuffer gespeicherte Meldungen anzuzeigen)
	display.pixel(3, 4, 1)	Setzen des Pixels an der Position (x,y) auf dem Bildschirm, mit x=3 & y=4, auf den Zustand 1 (d.h. Anzeige)