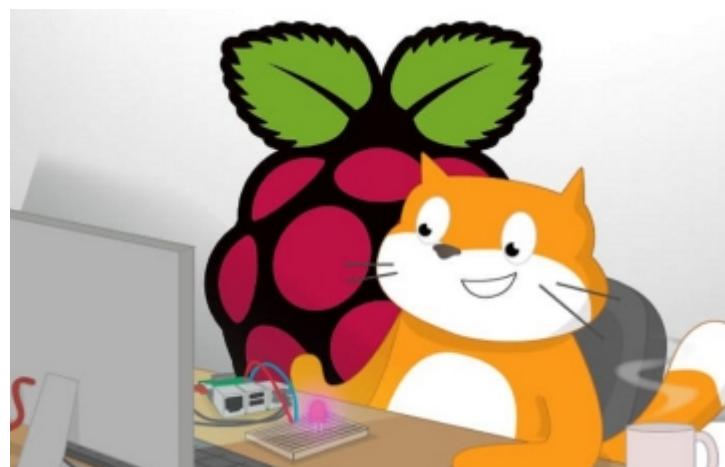




## **Einsteigerkurs**

**Python Fortgeschritten**

**Praxis**





## Ausgabe 1 – LED blinken

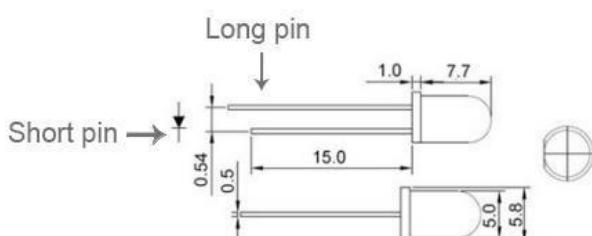


### Einleitung

In den ersten Aufgaben wird gezeigt, was LED ist und wie sie funktioniert, wie man es richtig mit dem Raspberry-Pi mit GPIO-Ports verbindet und dann verwenden wir Beispielskript, um einen Code mit unserer Hardware auszuführen und zu testen. Es wird eine 5mm LED verwendet, welche schnell ein- und ausgeschaltet wird.

### Spezifikation

Unten abgebildet ist die LED-Spezifikation, der lange Stift zeigt die positive Seite an, während der kurze Stift die negative Seite anzeigt.



Hinweis: Es ist wichtig, nicht versehentlich den langen Stift mit dem kurzen Stift zu verwechseln, das dazu führt, dass die Schaltung nicht richtig funktioniert.



## Benötigte Hardware

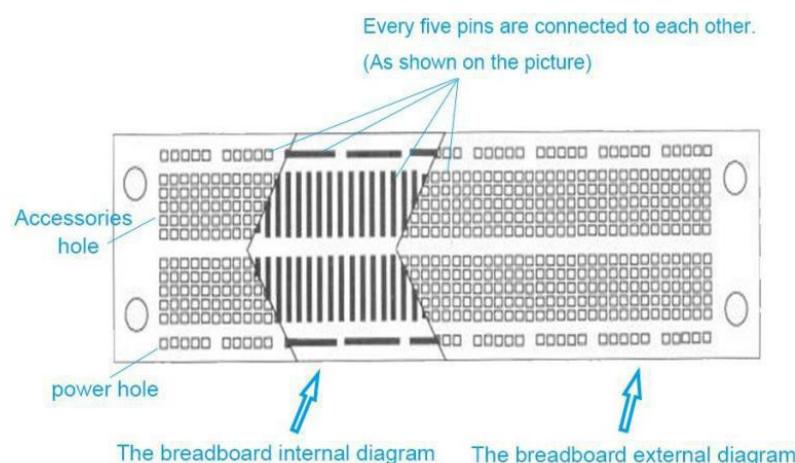
Im Folgenden finden Sie die Liste aller Komponenten, die für diese Lektion erforderlich sind. Es muss sichergestellt, dass die richtige Hardware verwendet wird. Der Widerstand kann entweder 220 oder 300 Ohm sein, beide funktionieren.

Materialdiagramm	Materialname	Anzahl (Betrag)
	Led	1
	220/3000 Widerstand	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



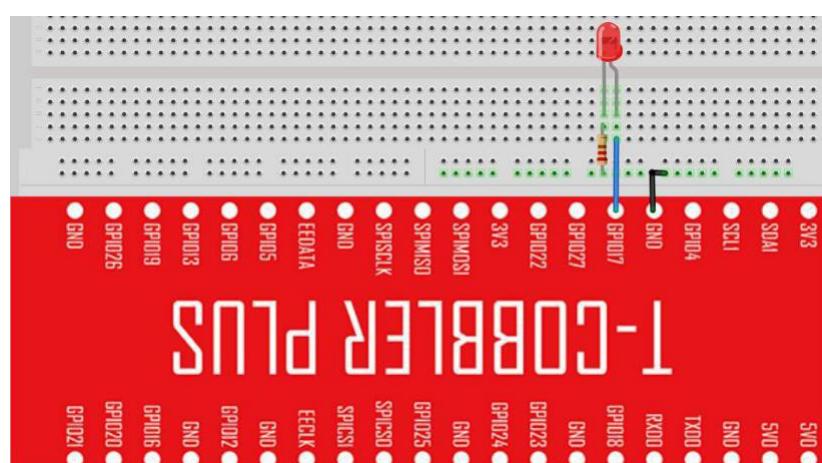
## Breadboard-Schema

Unten ist das Breadboard-Schema zu sehen. Es ist wichtig zu verstehen, wie das Breadbaord funktioniert, damit mit den Komponenten richtig gearbeitet werden kann.



## Verbindungsdiagramm

Wie im Diagramm gezeigt, verbinden wir den negativen Pin der LED mit dem Widerstand und der positive Pin direkt an den Raspberry Pi GPIO.



## Verbindung

Led	Raspberry Pi
Langer Pin (+)	GPIO17
Kurzer Pin (-)	GND



## Codeübersicht

Wir definieren GPIO 17 als blinkenden LED-Pin und richten dann den Pin als GPIO-Ausgang (OUT) ein. Danach läuft das Programm in einer While-Schlaufe, bis es mit STRG-C oder STRG-Z beendet wird. Die LED wird eingeschalten, indem Sie GPIO auf HIGH gesetzt wird. Danach wird für 0,2 Sekunden (200 Millisekunden) gewartet, bevor die LED wieder ausgeschaltet wird (GPIO auf LOW). Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 blink.py
```

---

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  # http://elecrow.com/
4
5  import time
6  import RPi.GPIO as GPIO
7
8  # define LED pin
9  led_pin = 17
10
11 # set GPIO mode to GPIO.BCM
12 GPIO.setmode(GPIO.BCM)
13 # set puin as input
14 GPIO.setup(led_pin, GPIO.OUT)
15
16 try:
17     while True:
18         # turn on LED
19         GPIO.output(led_pin, GPIO.HIGH)
20         # Wait half a second
21         time.sleep(0.2)
22         # turn off LED
23         GPIO.output(led_pin, GPIO.LOW)
24         # Wait half a second
25         time.sleep(0.2)
26 except KeyboardInterrupt:
27     # CTRL+C detected, cleaning and quitting the script
28     GPIO.cleanup()
```

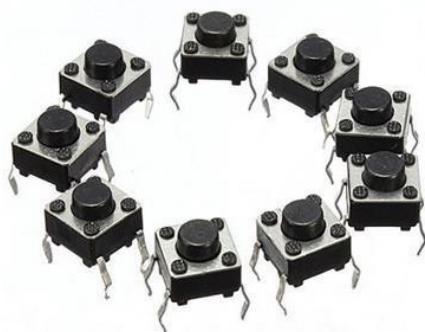
---

## Anwendungseffekt

Wenn Sie das Programm ausführen, wird eine LED für 0,2 Sekunde eingeschaltet, dann für 0,2 Sekunden ausgeschaltet, danach wiederholt sich der Vorgang.



## Aufgabe 2 – Button verwenden

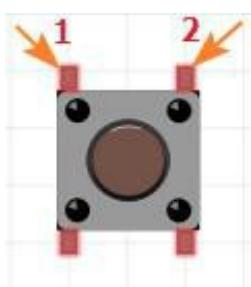


### Einleitung

In dieser Aufgabe wird gezeigt, wie ein Taster verwendet wird. Die Taste ist sehr nützlich in vielen Anwendungen zum Beispiel Licht einschalten durch Drücken oder Musik spielen. In unserem Beispiel verwenden wir die Schaltfläche, um anzusehen, ob sie gedrückt oder freigegeben wurde.

### Spezifikation

Unten ist die Schaltfläche gezeigt. Diese hat 2 Pins, welche durch einen Tastendruck gleichzeitig geschlossen oder geöffnet werden.





## Benötigte Hardware

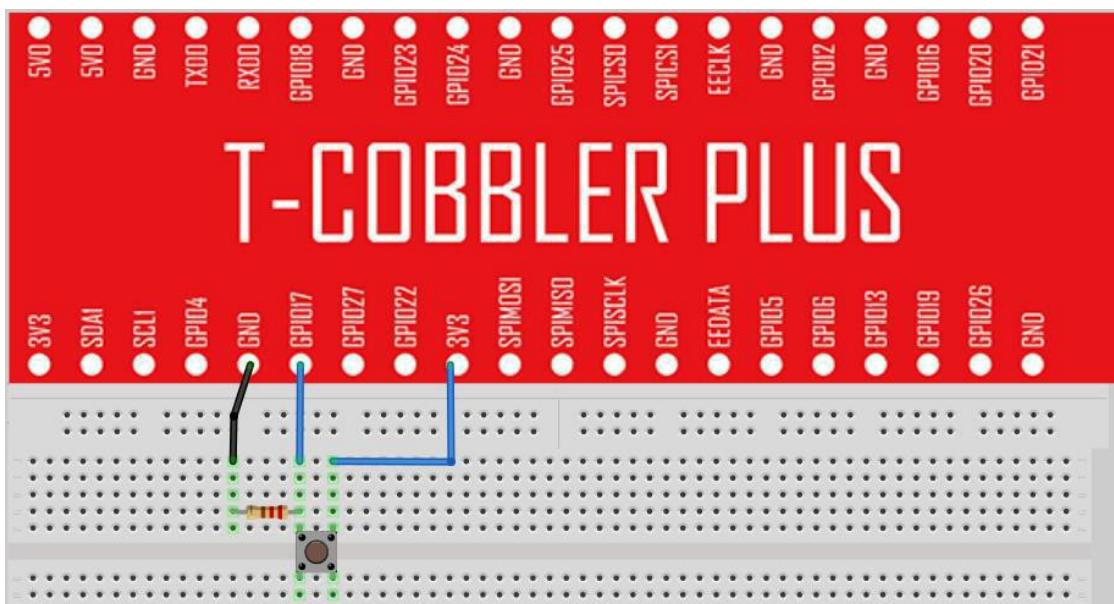
Im Folgenden finden Sie die Liste aller Komponenten, die für diese Aufgabe erforderlich sind. Achten Sie darauf, dass der Widerstand 10K (Kilo) Ohm und nicht 10 Ohm ist.

Materialdiagramm	Materialname	Anzahl (Betrag)
	Schaltfläche	1
	10K-Widerstand	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Draht	Mehrere



## Verbindungsdiagramm

Wie wir im Diagramm sehen können, verbinden wir die Taste mit dem 10K Ohm Widerstand auf der negativen Seite sowie mit dem GPIO-Port bei GPIO17. Der andere Pin der Taste geht an 3V Pin auf dem Raspberry-Pi. Bitte beachten Sie: Wenn Sie die Taste in die falsche Richtung anschliessen, kann der Raspberry Pi das Gegenteil anzeigen, GPIO HIGH, wenn die Taste losgelassen wird, und GPIO LOW, wenn die Taste gedrückt wird, daher funktioniert das Skript möglicherweise nicht wie erwartet.



## Verbindung

Schaltfläche	Raspberry Pi
Pin 1	GPIO17
Pin 2	3V3



## Codeübersicht

Werfen wir einen Blick durch unseren Code: Wir setzen den Button-Pin als GPIO 17 und dann den Modus als GPIO (BCM) und als Eingang (IN), da wir Eingaben von der Taste erhalten wollen, wenn sie gedrückt wurde oder nicht. Wenn der Eingang auf TRUE gesetzt wurde, dann wurde die Taste gedrückt und es soll "Taste gedrückt" ausgegeben werden. Wenn die Taste nicht gedrückt wurde, dann soll "Taste freigegeben" angezeigt werden. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 button.py
```

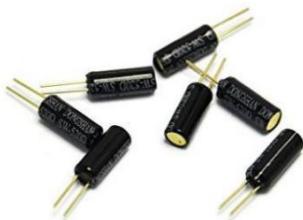
```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  # http://elecrow.com/
4
5  import RPi.GPIO as GPIO
6  import time
7
8  # configure button pin
9  button_pin = 17
10
11 # set board mode to GPIO.BCM
12 GPIO.setmode(GPIO.BCM)
13
14 # setup button pin as input
15 GPIO.setup(button_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
16
17 try:
18     while True:
19         # check if button pressed
20         if(GPIO.input(button_pin)):
21             # Button is pressed
22             print("Button Pressed")
23         else:
24             # it's not pressed
25             print("Button Released")
26         time.sleep(0.1)
27     except KeyboardInterrupt:
28         GPIO.cleanup()
```

## Anwendungseffekt

Wenn Sie das Programm ausführen, wird "Taste gedrückt" angezeigt, wenn die Taste gedrückt und "Taste losgelassen" angezeigt, wenn die Taste losgelassen wird.



## Aufgabe 3 - Kugelschalter

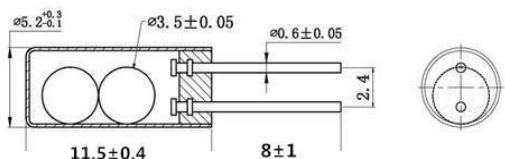


### Einleitung

In dieser Aufgabe erfahren wir mehr über den Kugelschalter, der auch als "Neigesensor" bezeichnet wird, sobald der Schalter zur Seite geneigt ist, er könnte entweder geöffnet oder geschlossen sein. Der Ball berührt die Seite des Schalters, wodurch er einen Schaltkreis schliesst. Wenn er auf die andere Seite kippt, dann öffnet sich der Schaltkreis (aufgrund des Balles, der die Seite nicht berührt).

### Spezifikation

Unten ist die Kugelschalter Spezifikation, wie wir innerhalb des Schalters dort kleine Kugeln sehen können, sobald sie die linke Seite berühren, wird es den Kreis zwischen den beiden "Sticks" (Pins) schliessen, so dass der Strom fliessen kann.





## Benötigte Hardware

Unten ist die Hardware-Liste für unsere Aufgabe, stellen Sie bitte sicher, dass Sie den 10K Ohm Widerstand und nicht einen 10 Ohm Widerstand für dieses Beispiel verwenden.

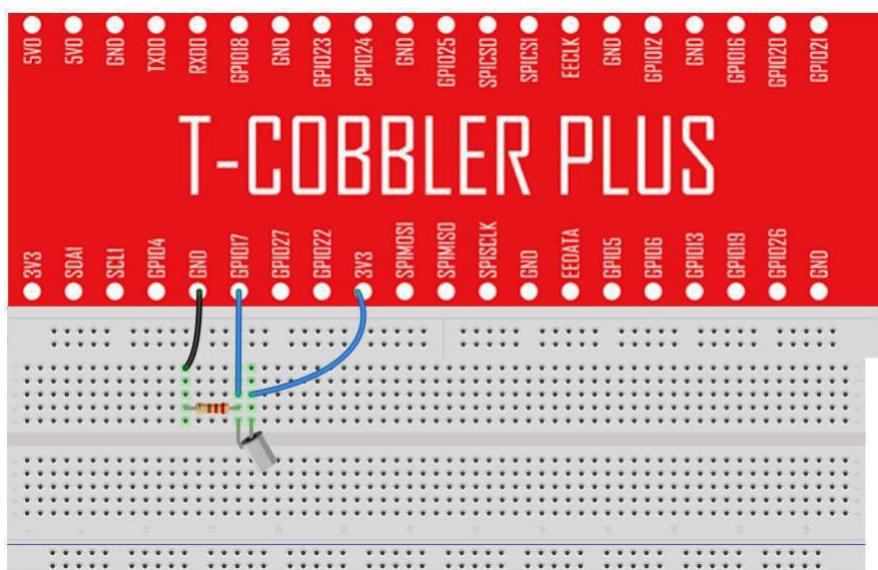
Materialdiagramm	Materialname	Anzahl (Betrag)
	Kugelschalter	1
	10K-Widerstand	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Wie im Diagramm zu sehen ist, verbinden wir den Negativstift mit dem GND im Raspberry Pi sowie an den 10K Ohm Widerstand und an einen Pin des Kugelschalters.

Hinweis: Es muss ausprobiert werden, wie der Kugelschalter angeschlossen werden muss. GPIO kann HIGH oder LOW sein, das ist bei Ausführung des Skriptes zu testen. Wenn das Skript nicht wie in diesem Tutorial beschrieben funktioniert, versuchen Sie, die Pins des Kugelschalters zu ändern.



## Verbindung

Wie bereits erwähnt, hat der Kugelschalter keine spezifischen Pins, versuchen Sie beides und sehen Sie die Ergebnisse.

Komponente	Raspberry Pi
Pin 1	GPIO17
Pin 2	3V3



## Codeübersicht

Lassen Sie uns durch unseren Code gehen: Wir richten den Switch-Pin auf GPIO17 als GPIO ein. IN, weil wir Eingang vom Kugelschalter erhalten, wenn er die Schaltung durch Kippen schliesst oder nicht. Dann gehen wir für, während Schleife (für immer) und drucken "BALL ist HIGH", wenn es die Schaltung schliesst oder "BALL ist LOW", wenn nicht. Sie können den Kugelschalter umkippen, um zu sehen, wie sich die Daten ändern. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 ball_switch.py
```

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  # http://elecrow.com/
4
5  import RPi.GPIO as GPIO
6  import time
7
8  # configure button pin
9  ball_switch_pin = 17
10
11 # set board mode to GPIO.BCM
12 GPIO.setmode(GPIO.BCM)
13
14 # setup ball pin input
15 GPIO.setup(ball_switch_pin, GPIO.IN)
16
17 try:
18     while True:
19         # check ball state
20         if(GPIO.input(ball_switch_pin)):
21             # Ball is high means it's closing the circuit by touching
22             print("Ball is HIGH")
23         else:
24             # Ball is low means it's not closing the circuit
25             print("Ball is LOW")
26             time.sleep(0.1)
27 except KeyboardInterrupt:
28     GPIO.cleanup()
```

## Anwendungseffekt

Das Skript druckt "BALL is HIGH", wenn es die Schaltung schliesst oder "BALL ist LOW", wenn nicht. Sie können den Kugelschalter umkippen, um zu sehen, wie sich die Daten ändern.



## Aufgabe 4 – Aktiver Buzzer



### Einleitung

Active buzzer ist ein sehr nützliches Modul, das in vielen Anwendungen verwendet werden kann, zum Beispiel: Wecker, Bewegungserkennung oder als Warnung an eine USV, um Sie wissen zu lassen, dass der Akku leer ist. Egal, worum es bei Ihrem Projekt geht, der Buzzer kann wahrscheinlich ein sehr nützliches Bauteil sein.

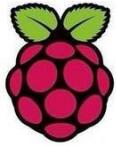
Im folgenden Tutorial erfahren wir, wie der Summer funktioniert und wie man ihn mit dem Raspberry Pi GPIO steuert, wie man ihn mit dem Breadboard und den Jumpern verdrahtet und mit dem Python-Skriptbeispiel verwendet, das auf unserer GitHub-Seite angehängt ist.

**Achten Sie darauf, den aktiven Summer nicht mit dem passiven Summer im Kit zu verwechseln, der Unterschied ist, dass der aktive Summer einen Aufkleber oben hat (ein Siegel) der passive Summer hat keinen.**



## Benötigte Hardware

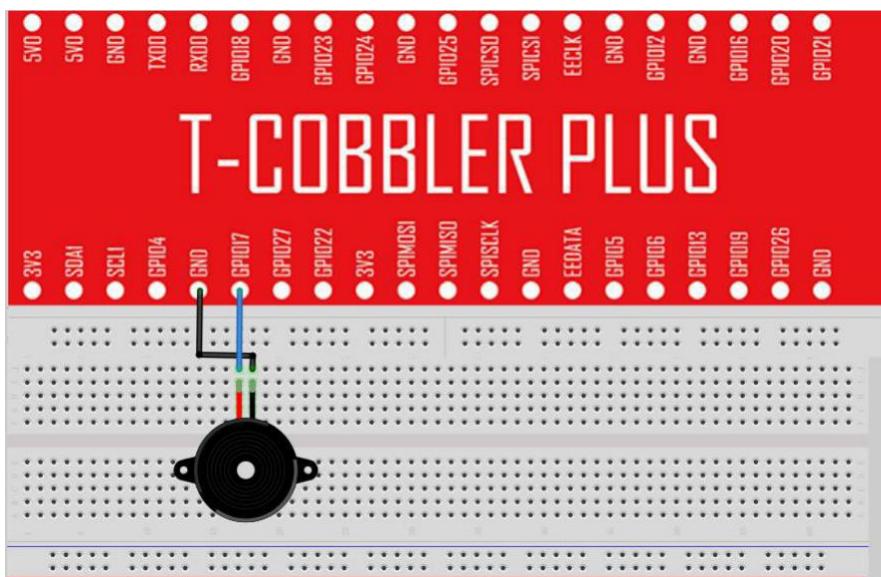
Unten ist die Liste der Hardware, stellen Sie sicher, nicht den aktiven Summer und passive Summer im Kit zu verwechseln. Beide haben ein Tutorial, das durchgeht und erklärt, wie sie funktionieren, aber im Moment konzentrieren wir uns nur auf den aktiven Summer.

Materialdiagramm	Materialname	Anzahl (Betrag)
 A small circular icon of a speaker with a label that reads "HYDRO REACTIVE AFTER WASHING".	Aktiver Buzzer	1
 A stylized illustration of a Raspberry Pi Model B+ board with its characteristic red and white color scheme and four USB ports.	Raspberry Pi Board	1
 A red T-shaped breadboard with a central vertical column and two horizontal rows of connection points.	T-Cubbler Plus	1
 A bundle of 40-pin ribbon cables, each with a grey plastic connector on one end and a metal pin on the other.	40P GPIO Kabel	1
 A photograph of a standard grey rectangular breadboard with a grid of holes for component placement.	Steckbrett	1
 A bundle of jumper wires with various colored insulation, including red, blue, yellow, and orange.	Jumper-Kabel	Mehrere



## Verbindungsdiagramm

Wie wir im Diagramm sehen können, ist die Verbindung ziemlich einfach. Wir verbinden den positiven Pin des Summers mit GPIO17 auf dem Raspberry Pi und den negativen Pin mit dem Ground.



## Verbindung

Bitte beachten Sie: Der lange Stift des Summers ist der positive Anschluss, während der kurze Stift des Summers der negative Anschluss ist.

Aktiver Summer	Raspberry Pi
Langer Pin (+)	GPIO17
Kurzer Pin (-)	GND



## Codeübersicht

Der Code ist ziemlich einfach. Wir verwenden keine Schleife in diesem Code. Wir richten den Summer als GPIO17 als GPIO OUT ein. Der Pin wird dann auf HIGH oder LOW gesetzt, um den Buzzer zu steuern

Wir verwenden den Befehl GPIO.cleanup(), um den GPIO17-Pin freizugeben. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 active_buzzer.py
```

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 # http://elecrow.com/
4
5 import RPi.GPIO as GPIO
6 import time
7
8 buzzer_pin = 17
9
10 GPIO.setmode(GPIO.BCM)
11 GPIO.setup(buzzer_pin, GPIO.OUT)
12
13 # Make buzzer sound
14 GPIO.output(buzzer_pin, GPIO.HIGH)
15 time.sleep(0.5)
16 # Stop buzzer sound
17 GPIO.output(buzzer_pin, GPIO.LOW)
18
19 GPIO.cleanup()
```

## Anwendungseffekt

Wenn Sie das Programm ausführen, wird der Summer 0,5 Sekunden lang eingeschaltet und dann ausgeschaltet.



## Ausgabe 5 – Passiver Summer



### Einleitung

In dieser Aufgabe lernen wir mehr über den passiven Summer.

Der passive Summer ist dem aktiven Summer sehr ähnlich. Allerdings muss er mit einem wechselnden Signal angesteuert (der aktive Summer wird durch Strom aktiviert, der durch ihn läuft). In dieser Aufgabe lernen wir, wie man den passiven Summer verwendet.

### Spezifikation:

Arbeitsspannung: 3V/5V Widerstand: 16Ohm Resonanzfrequenz: 2 kHz



## Benötigte Hardware

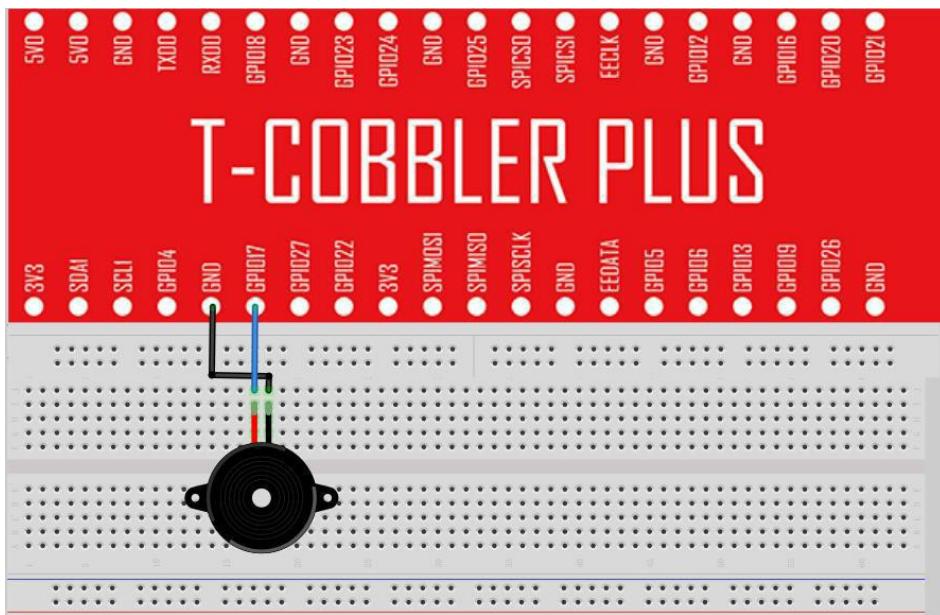
Im Folgenden finden Sie die Hardware, die wir für diese Aufgabe benötigen. Stellen Sie sicher, die Hardware vorzubereiten und vor allem nicht den passiven Summer mit dem aktiven Summer im Kit zu verwechseln.

Materialdiagramm	Materialname	Anzahl (Betrag)
	Passiver Buzzer	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Kabel	Mehrere



## Verbindungsdiagramm

Wie wir im Diagramm sehen können, ist das Design genau das gleiche wie das vorherige Beispiel mit dem aktiven Summer. Wenn Sie gerade die aktive Summer-Aufgabe abgeschlossen haben, können Sie einfach das Summer-Modul wechseln, ohne den Rest der Pins herauszunehmen.



## Verbindung

Achten Sie darauf, den langen Anschluss (die positive Seite) nicht mit dem kurzen Anschluss (der negativen Seite) zu verwechseln.

Passiver Summer	Raspberry Pi
Langer Pin (+)	GPIO17
Kurzer Pin (-)	GND



## Codeübersicht

Der passive Summer ist etwas komplizierter als der aktive Summer. Der Code ist lang und jede Zeile wird kommentiert, um genau zu erklären, was passiert. Der passive Summer wird mit einem Signal einer bestimmten Frequenz angesteuert. Durch Veränderung die Pitches kann der Ton geändert werden. In diesem Beispiel soll die Tonleiter gespielt werden. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 passive_buzzer.py
```

```
1 import RPi.GPIO as GPIO #import the GPIO library
2 import time #import the time library
3
4 class Buzzer(object):
5
6     def __init__(self):
7
8         GPIO.setmode(GPIO.BCM)
9         self.buzzer_pin = 17 #set to GPIO pin 17
10        GPIO.setup(self.buzzer_pin, GPIO.OUT)
11        print("buzzer ready")
12
13    def buzz(self,pitch, duration): #create the function "buzz" and feed it the pitch and duration)
14
15        if(pitch==0):
16            time.sleep(duration)
17            return
18
19        period = 1.0 / pitch #in physics, the period (sec/cyc) is the inverse of the frequency (cyc/sec)
20        delay = period / 2 #calcuate the time for half of the wave
21        cycles = int(duration * pitch) #the number of waves to produce is the duration times the frequency
22
23        for i in range(cycles): #start a loop from 0 to the variable "cycles" calculated above
24            GPIO.output(self.buzzer_pin, True) #set pin 18 to high
25            time.sleep(delay) #wait with pin 18 high
26            GPIO.output(self.buzzer_pin, False) #set pin 18 to low
27            time.sleep(delay) #wait with pin 18 low
28
29    def play(self): 28 von 114
```

## Anwendungseffekt

Der passive Summer wird die Tonleiter vorwärts und rückwärts spielen.



## Aufgabe 6 – RGB-LED



### Einleitung

Bevor wir in unseren vorherigen Aufgaben verschiedene Arten von LEDs verwendet haben (in unserem Kit haben wir 5mm LEDs in verschiedenen Farben) ist die RGB-LED ein wenig anders. Der RGB steht für Rot, Grün und Blau. Es ermöglicht uns, 3 Farben zu steuern und mit diesen Farben, erzeugen viele Farben! Wir lernen, wie man jede der Farben separat aktiviert und später kombiniert und unterschiedliche Farbpaletten erstellt.

Die RGB-LED ist sehr nützlich für alle Anwendungen, wenn Sie LED zum Beispiel für Indikationszwecke benötigen, gibt es keine Notwendigkeit, rote LED für ein Szenario und blau für ein anderes zu verwenden. Wir können RGB-LED verwenden und die Farben je nach unseren Bedürfnissen ändern!

### Spezifikation

Emitting Light Color: Blau, Rot, Grün  
Grösse (ca.): 5 x 35mm / 0,2" x 1,37"  
(D \* L)  
Vorwärtsspannung: 3,0-3,4V  
Lichtstärke: 12000-14000mcd  
Pin Definition





## Benötigte Hardware

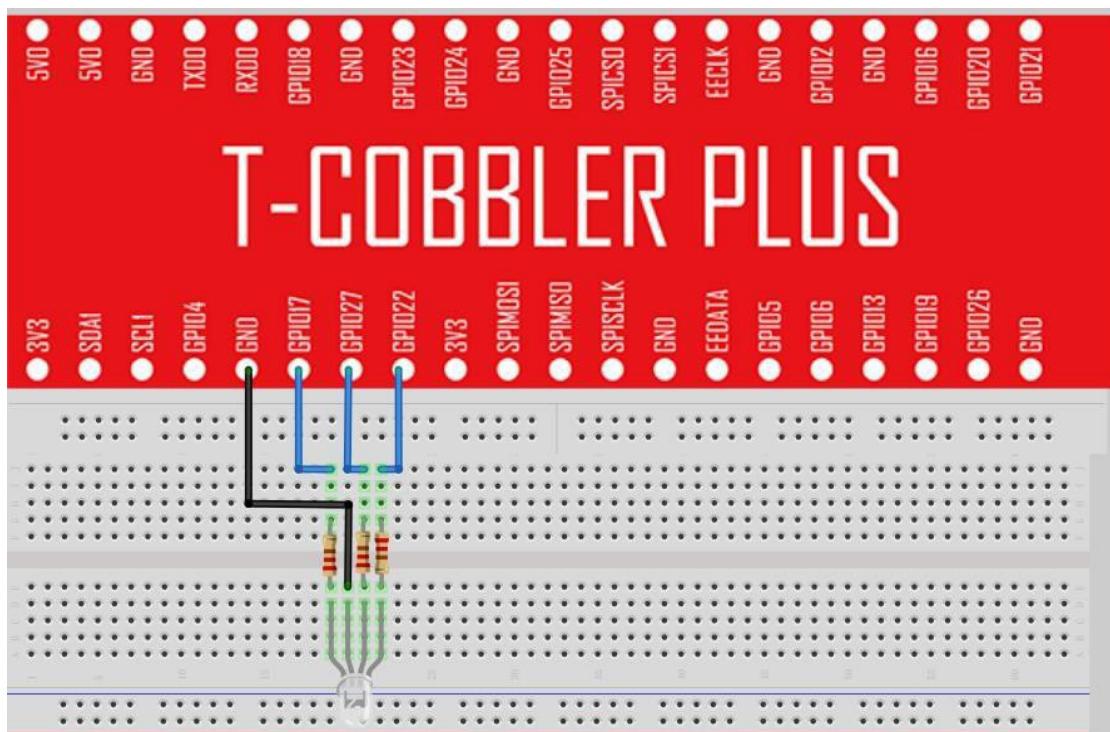
Im Folgenden finden Sie die Hardware, die wir benötigen, um diese Aufgabe abzuschliessen. Bitte beachten Sie den 220/330ohm Widerstand, wir benötigen 3 davon. Entweder 220 oder 330 funktionieren, wir können auch wenige 220 und wenige 330ohm Widerstände kombinieren, es gibt kein Problem damit.

Materialdiagramm	Materialname	Anzahl (Betrag)
	RGB-LED	1
	220/330O Widerstand	3
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Wie wir im Diagramm sehen können, müssen wir jeden Pin der RGB-LED (mit Ausnahme des Ground) mit einem Widerstand an die GPIO verbinden, damit wir ihn separat steuern können.



## Verbindung

Stellen Sie sicher, dass Sie die Pins richtig anschliessen. Denken Sie daran, jede Farbe durch verschiedenen GPIO-Pin gesteuert, wenn Sie versehentlich den falschen Stift auf die falsche Farbe setzen, wird dies dazu führen, dass das Skript andere Farben zeigen, als wir es programmiert haben.

RGB-LED	Raspberry Pi
R	GPIO17
G	GPIO27
B	GPIO22
GND	GND



## Codeübersicht

Der Code ist lang, aber es ist alles kommentiert, um zu erklären, was vor sich geht. Wir werden hauptsächlich durch den Hauptteil des Codes gehen. Wir definieren das Objekt RGB, das wir zuvor im Code eine Klasse erstellt haben, dann schalten wir die RGB-LED "rot" für 0,3 Sekunden ein und schalten sie aus. Das Gleiche tun wir danach für die grüne und die blaue LED. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 RGB.py
```

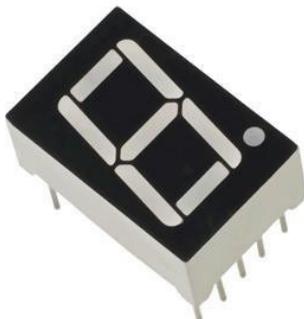
```
67  def main():
68      # Initialize RGB Class
69      RGB_LED = RGB()
70
71      # Blink Red
72      RGB_LED.turn_on("red")
73      time.sleep(0.3)
74      RGB_LED.turn_off("red")
75      time.sleep(0.3)
76
77      # Blink Green
78      RGB_LED.turn_on("green")
79      time.sleep(0.3)
80      RGB_LED.turn_off("green")
81      time.sleep(0.3)
82
83      # Blink Blue
84      RGB_LED.turn_on("blue")
85      time.sleep(0.3)
86      RGB_LED.turn_off("blue")
87
88  main()
```

## Anwendungseffekt

Das Skript wird rote LED, grüne LED und dann blaue LED mit einem Abstand von 0,3 Sekunden Verzögerung einschalten.



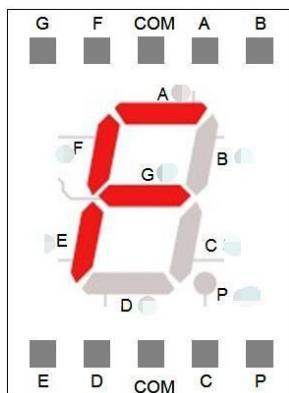
## Aufgabe 8 – einfache 7-Segment-Anzeige



### Einleitung

In dieser Aufgabe erfahren Sie mehr über die Segmentanzeige. In unserem Kit haben wir 2 Arten von Segmentanzeige eine ist ein-(1-stellig) Segmentanzeige und eine ist 4-stellig. Beide Displays sind gleich, der einzige Unterschied ist die Anzahl der Ziffern, die wir steuern. In unserer ersten Aufgabe lernen wir, wie man ein Segment steuert und eine Zahl zeigt. Ein Segment kann eine Zahl zwischen 0 und 9 anzeigen.

### Pin-Definition



Wie wir sehen können, ist die Segment-LED mit Buchstaben in dieser Pin-Definition markiert, das reale Segment im Kit hat keine Buchstaben darauf. Bitte beachten Sie diese Definition beim Verdrahten der Segment-LED, um sicherzustellen, dass wir sie richtig verdrahten.



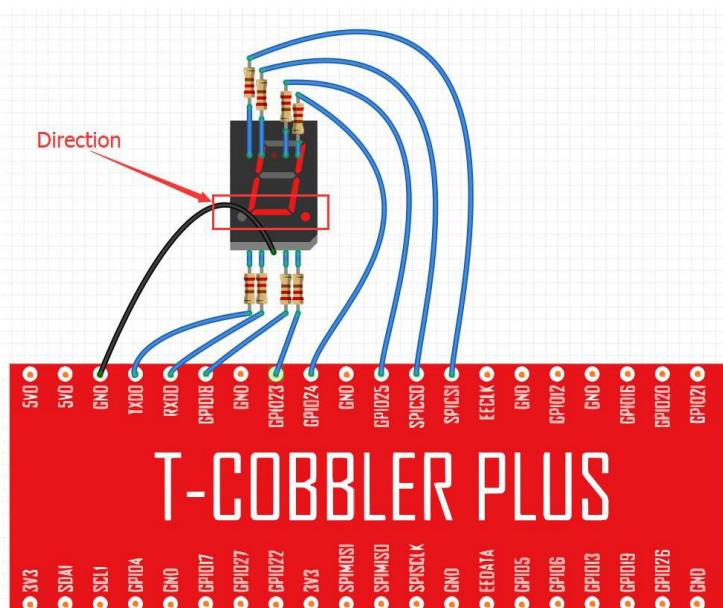
## Benötigte Hardware

Unten ist die Hardware, die wir für diese Aufgabe benötigen, 1 Ziffer (nicht mit 4-stelligen) Segmentanzeige und 220 oder 330 Ohm Widerstände verwechselt, wir benötigen 8 von ihnen.

Materialdiagramm	Materialname	Anzahl (Betrag)
	1 Digit 7 Segmentanzeige	1
	220/330O Widerstände	8
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere

## Verbindungsdiagramm

Wie wir sehen können, ist das Verdrahtungsdiagramm umfangreich. Nehmen Sie sich Zeit, um die Pin-Definition, wie zuvor gezeigt richtig zu verbinden. Bitte beachten Sie den kleinen Punkt auf der Segment-LED, der Ihnen mitteilt, in welche Richtung die Segment-LED gerichtet werden soll. Der Punkt sollte nach unten gerichtet werden. Vergessen Sie nicht, alle Widerstände auf die Pins zu legen, während Sie sie an die GPIO-Ports anschliessen. Alle Stifte mit Ausnahme des GND-Pins (Boden) sollten mit einem Widerstand versehen sein.



# Verbindung

Befolgen Sie die Verbindung sorgfältig, um sicherzustellen, dass sie versehentlich kein Segment falsch verdrahten, was zu einer Fehleransicht der Zahlen führt, wenn wir das Testskript am Ende der Aufgabe ausführen.

Segment	Raspberry Pi
Eine	GPIO25
B	GPIO20
C	GPIO24
D	GPIO21
Und	GPIO16
F	GPIO18
G	GPIO12
H	GPIO23
GND	GND



## Codeübersicht

Der Code ist lang, aber keine Angst! Dies ist aufgrund des Klassencodes, der Hauptcode ist eigentlich sehr einfach. Was wir getan haben, ist, dass wir jede Led auf dem Segment als Brief von A nach H geführt und dann drehen wir sie auf der Grundlage der Farben, die sie sein sollten. In unserem Beispiel werden wir von 0 nach 9 schleifen und die Zahlen anzeigen, die sich derzeit in der Schleife befinden, sodass, wenn sich die Schleife in der ersten Runde (0) befindet, die Zahl Null bis zur letzten Zahl angezeigt wird, die 9 sein wird. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 1digit_segment.py
```

```
1 import RPi.GPIO as GPIO ## Import GPIO library
2 import time ## Import 'time' library. Allows us to use 'sleep'
3
4 # Segment 1 Digit Numbers Illustration
5 # H Is the dot on the right side of the segment
6 #       A
7 #   -----
8 #   |       |
9 #   B |       | C
10 #    |   D   |
11 #   -----
12 #    |       |
13 #   E |       | F
14 #    |       |
15 #   ----- (self.H)
16 #       G
17
18 class Segment():
19
20     def __init__(self):
21
22         # set the GPIO mode as GPIO.BCM
23         GPIO.setmode(GPIO.BCM)
24
25         # set pin names for each of the segment pins
26         self.A = 25
27         self.B = 20
28         self.C = 24
29         self.D = 21
30         self.E = 16
31         self.F = 18
32         self.G = 12
33         self.H = 23
34
35         # set the segment pins as GPIO.OUT
36         GPIO.setup(self.A, GPIO.OUT)
37         GPIO.setup(self.B, GPIO.OUT)
38         GPIO.setup(self.C, GPIO.OUT)
```

## Anwendung Wirkung

Es werden die Zahlen von 0 bis 9 auf dem LED-Segment mit einem Abstand von 1 Sekunde zwischen jeder Zahl angezeigt.



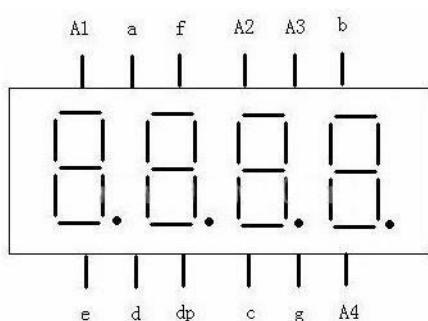
## Aufgabe 9 – vierstellige 7-Segment-Anzeige



### Einleitung

In der vorherigen Aufgabe, die wir über eine einstellige Segmentanzeige gelernt haben, ist die 4-stellige 7-Segment-Anzeige nicht anders. In dieser Aufgabe erfahren wir mehr über das 4-stellige Segment, wie Sie es aktivieren und mit einem einfachen Python-Skript steuern können. Das Skript erhält die aktuelle Uhrzeit vom Raspberry-Pi und zeigt sie auf der Segment-LED an.

### Pin-Definition



Die Pin-Definition des 4-stelligen Segments ist der 1-stelligen Anzeige sehr ähnlich. Bitte beachten Sie die Buchstaben auf jedem Stift und schliessen Sie ihn korrekt an.



## Benötigte Hardware

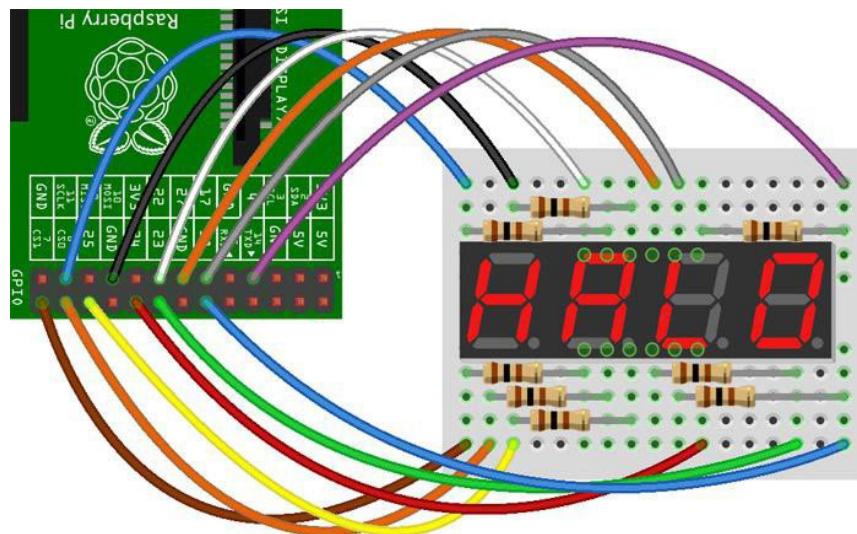
Für dieses Beispiel benötigen wir einen Widerstand von 220/330Ohm (8 davon, 1 für jeden Pin). Bitte beachten Sie, dass Sie entweder 220 oder 330 verwenden oder beide kombinieren können, alle Optionen funktionieren perfekt.

Materialdiagramm	Materialname	Anzahl (Betrag)
A small digital display module with seven segments and a common cathode connection.	Segmentanzeige	1
A cylindrical resistor component with two leads.	220/330O Widerstände	8
A Raspberry Pi Model B+ single-board computer with a Broadcom SoC, RAM, and various connectors.	Raspberry Pi Board	1
A red T-shaped breadboard with a central component holder and multiple horizontal and vertical connection rails.	T-Cubbler Plus	1
A flat ribbon cable with 40 pins, used for connecting the Raspberry Pi's GPIO pins to the breadboard.	40P GPIO Kabel	1
A standard breadboard with a grid of holes for component placement and a central power rail.	Steckbrett	1
A bundle of multi-colored jumper wires used for connecting components on the breadboard.	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Bitte beachten Sie die vorherige Pin-Definition, um sicherzustellen, dass die richtigen Pins an die richtige Stelle angeschlossen werden. Wir müssen Widerstand fast zu jedem Pin hinzufügen. Die Richtung der Verbindung ist mit dem Punkt an der Unterseite.



## Verbindung

Seg / digit	7Seg Pin	Resistor	BCM GPIO #
bot left	1	YES	7
bot centre	2	YES	8
decimal pt	3	YES	25
bot right	4	YES	23
centre centre	5	YES	18
digit 4	6	NO	24
top right	7	YES	4
digit 3	8	NO	17
digit 2	9	NO	27
top left	10	YES	10
top centre	11	YES	11
digit 1	12	NO	22



## Codeübersicht

In diesem Beispiel zeigen wir die aktuelle Zeit unseres Raspberry-Pi-Systems auf der 4-Segment-LED-Anzeige. Bitte beachten Sie, dass Sie die richtige Zeitzone über die "sudo raspi-config"-Einstellungen konfigurieren müssen, da die Zeit, die Sie sehen werden, sich möglicherweise von der tatsächlichen Zeitzone unterscheidet, in der Sie sich gerade befinden. Lassen Sie uns durch unseren Code gehen und versuchen, es zu verstehen: Wir zeigen die Stunden auf den ersten beiden Spalten der Segment-LED und die Minuten auf den zweiten beiden Spalten. Wir starten zuerst eine While-Schleife (für immer) wir erhalten die aktuelle Zeit aus der Zeitbibliothek, dann gehen wir für jede Ziffer (wir haben insgesamt 4 Ziffern) und Schleife durch insgesamt 7 Zahlen. Und dann geben wir HIGH für die richtige Zahl aus, die sich derzeit auf der Uhr befindet (basierend auf der aktuellen Zeit) und LOW für die Zeit, die falsch ist (nicht die aktuelle Zeit). Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 4digit_segment.py
```

```
38  try:
39      while True:
40          n = time.ctime()[11:13]+time.ctime()[14:16]
41          s = str(n).rjust(4)
42          for digit in range(4):
43              for loop in range(0,7):
44                  GPIO.output(segments[loop], num[s[digit]][loop])
45                  if (int(time.ctime()[18:19])%2 == 0) and (digit == 1):
46                      GPIO.output(25, 1)
47                  else:
48                      GPIO.output(25, 0)
49                  GPIO.output(digits[digit], 0)
50                  time.sleep(0.001)
51                  GPIO.output(digits[digit], 1)
52  finally:
53      GPIO.cleanup()
```

## Anwendungseffekt

Wenn wir das Programm ausführen, wird uns die aktuelle Uhrzeit auf dem Segment angezeigt.



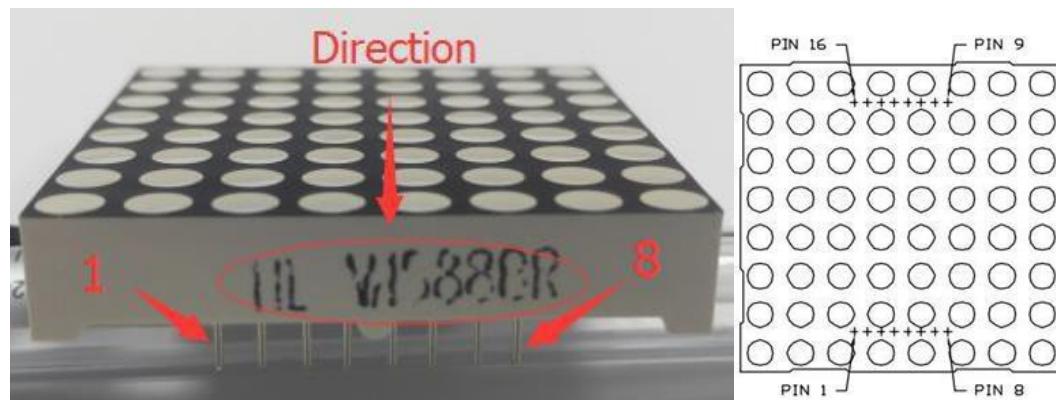
## Ausgabe 10 – LED-Matrix



### Einleitung

In dieser Aufgabe erfahren Sie, wie Sie die Matrixanzeige verwenden. Die Matrix-Anzeige ist sehr ähnlich der Segment-LED nur statt nur wenige LEDs. Die LEDs sind in Zeilen und Spalten angeordnet. Es können ganze Reihen, Spalten oder einzelne LED auf der Matrix eingeschaltet werden. In unserem Beispiel verwenden wir die Matrix-LED, um eine herzförmige Animation zu erstellen.

### Pin-Definition



Bitte beachten Sie die Richtung der Matrix-LED: Der schwarze Text sollte sich vorne befinden.



## Benötigte Hardware

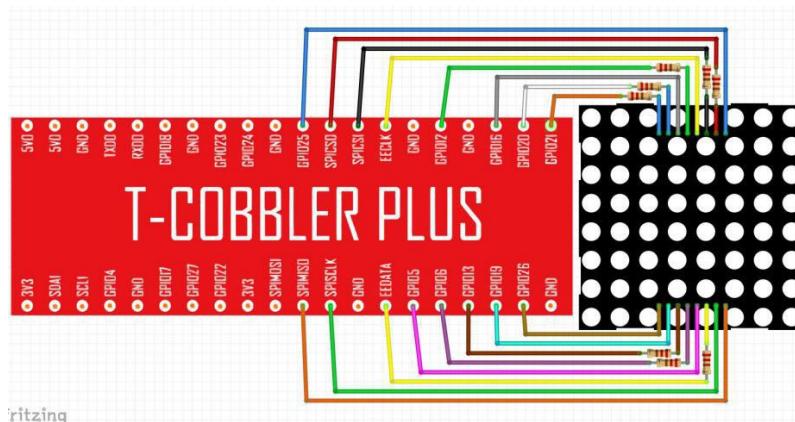
Für dieses Beispiel, genau wie das vorherige, benötigen wir 8x 220 oder 330Ohm Widerstände. Bitte beachten Sie die richtige Verbindung der Widerstände, damit dieses Beispiel richtig funktioniert.

Materialdiagramm	Materialname	Anzahl (Betrag)
	LED-Matrix	1
	220/330O Widerstände	8
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Es gibt viele Pins, die wir verbinden müssen und zu einigen von ihnen müssen wir Widerstände befestigen, bitte schauen Sie sorgfältig und stellen Sie sicher, dass Sie dem Diagramm folgen.



## Verbindung

LED-MATRIX	Raspberry Pi
pin1	GPIO26
Pin2	GPIO19
pin3	GPIO13
pin4	GPIO6
pin5	GPIO5
pin6	EE-DATA
pin7	SPI-SCLK
pin8	SPI-MISO
pin9	GPIO25
pin10	SPI-CSO
pin11	SPI-CSI
pin12	EE-CLK
pin13	GPIO12
pin14	GPIO16
pin15	GPIO20
pin16	GPIO21



## Codeübersicht

In diesem Beispiel werden wir die Matrix-LED in Herzform aktivieren, wir erstellen ein Array von LED's und entscheiden dann, welche LED eingeschaltet ist und welche eine Herzform durch die Zahlen 1 und 0 erzeugt. Dann gehen wir für jeden Platz in der Reihe von Zahlen und senden Einschalten auf die LED, die 1 (ein) und lassen Sie die LED aus, die 0 (aus). Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 matrix_heart.py
```

```
11  ### Initialize an SPI connection using BCM-mode pins 21, 20, and 16
12  max7219 = spi.SPI(clk=21, cs=20, mosi=16, miso=None)
13
14  ### Zero out all registers
15  for cmd in range(16):
16      packet = cmd << 8
17      max7219.put(packet,12)
18
19  ### Enable all columns via the scan limit register
20  max7219.put(int("101100000111",2),12)
21
22  ### Disable shutdown mode
23  max7219.put(int("110000000001",2),12)
24
25  ### Define the values for each column that gives us a heart shape
26  heart_shape = np.array([
27      [ 0, 0, 0, 0, 0, 0, 0, 0 ],
28      [ 0, 1, 1, 0, 0, 1, 1, 0 ],
29      [ 1, 1, 1, 1, 1, 1, 1, 1 ],
30      [ 1, 1, 1, 1, 1, 1, 1, 1 ],
31      [ 0, 1, 1, 1, 1, 1, 1, 0 ],
32      [ 0, 0, 1, 1, 1, 1, 0, 0 ],
33      [ 0, 0, 0, 1, 1, 0, 0, 0 ],
34      [ 0, 0, 0, 0, 0, 0, 0, 0 ],
35  ] )
36
37  ### Set each column according to our heart shape matrix
38  for col in range(8):
39      cmd = (col+1) << 8
40      values = 0
41      ### Without flipud, the heart would be upside-down since the LED matrix
42      ### defines 0,0 as the bottom left corner while numpy puts it at the top
43      ### left.
44      for i in np.flipud(heart_shape[:,col]):
45          values <=> 1
```

## Anwendungseffekt

Es werden bestimmte LEDs ein- und bestimmte LEDs ausgeschaltet, sodass die Matrix ein Herz anzeigt.



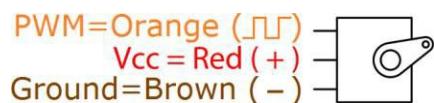
## Aufgabe 11 – Servo



### Einleitung

In dieser Aufgabe erfahren wir mehr über das 9G-Servo. Das 9G-Servo ist ein sehr nützliches Modul, das es uns ermöglicht, ein Servo in einem gewissen Bereich zubewegen. So ist es möglich, bestimmte Bewegungen oder Aktionen auszuführen. Wir werden lernen, wie man den Servo bewegt, indem wir ihm die Position (in Grad) vorgeben, zu der er sich bewegen soll.

### Pin-Definition



Die Pin-Definition des Servos ist sehr einfach. Wir haben insgesamt 3 Drähte: rot, orange und braun (oder schwarz). Rot und Braun zeigen positiv und negativ, während die Orange der Datenstift (PWN) ist, der es uns ermöglicht, das Servo selbst zu bewegen.



## Benötigte Hardware

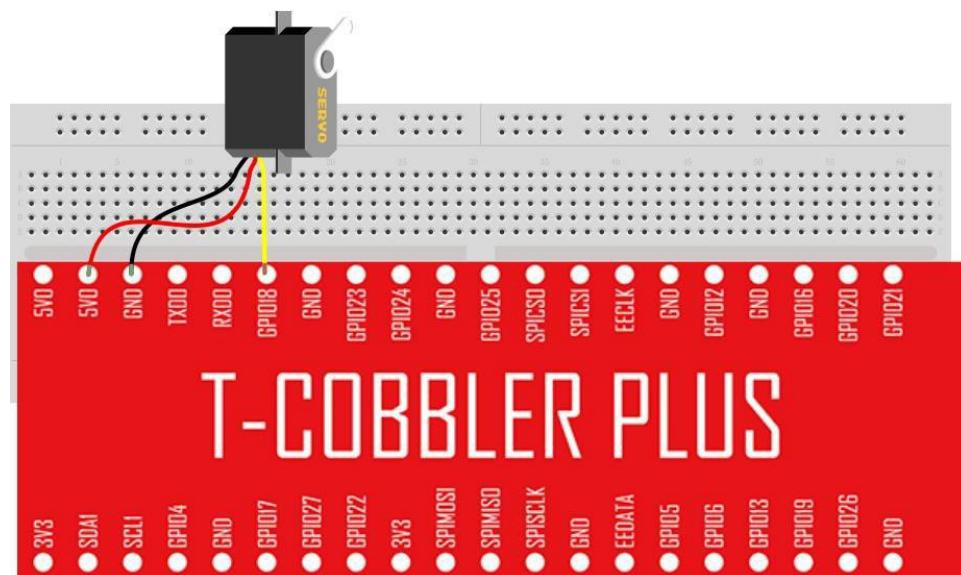
Die Hardware ist geradlinig, wir brauchen keine Widerstände oder spezielle Komponenten auf diesem. Wir verwenden das Servo und schliessen es mit dem Breadboard mit den Jumpers, folgen Sie dem Diagramm, um sicherzustellen, dass das Servo mit dem richtigen GPIO-Pin verbunden.

Materialdiagramm	Materialname	Anzahl (Betrag)
	Servomotor	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Roter Draht für 5V positives Schwarz ist für das Negative, der orange Draht, den wir für PWN-Befehle verwenden, wird direkt an GPIO18 angeschlossen.



## Verbindung

Diener	Raspberry Pi
Gelber Draht	GPIO18
Roter Draht	5v
Schwarzer Draht	GND



## Codeübersicht

Was wir mit unserem 9G Servo machen werden, ist ganz einfach. Wir werden zuerst die Klasse "sg90" initialisieren, die bereits alles vorcodiert und für uns mit dem Parameter "0" als Ausgangsrichtung vorbereitet hat. Dann wird es links drehen, indem Sie die Richtung auf 100 und Geschwindigkeit von 80 (von 100), um es in die andere Richtung nach rechts zu drehen, werden wir es auf -100 (gegenüber 100) mit der gleichen Geschwindigkeit von 80 drehen. Es wird weiterhin in einer Schleife gehen, bis gedrückt CTRL + C oder STRG + Z, um die Software zu beenden. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 servo.py
```

```
46  def main():
47
48      s = sg90(0)
49
50      try:
51          while True:
52              print("Turn left ...")
53              s.setdirection( 100, 80 )
54              time.sleep(0.5)
55              print("Turn right ...")
56              s.setdirection( -100, 80 )
57              time.sleep(0.5)
58      except KeyboardInterrupt:
59          s.cleanup()
60
61  if __name__ == "__main__":
62      main()
```

## Anwendungseffekt

Das Servo wird mit einer Geschwindigkeit von 80% der Maximalgeschwindigkeit für nach links und rechts gedreht, bis der Benutzer STRG+C oder STRG+Z drückt, um das Skript zu beenden.



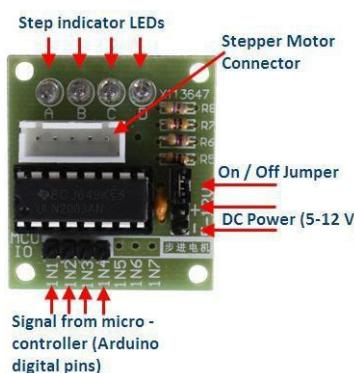
## Aufgabe 12 - Schrittmotor



### Einleitung

In dieser Aufgabe lernen wir über den Schrittmotor. Schrittmotor ähnlich dem Servo ist eine andere Art von "Motor", aber im Gegensatz zum Servo kann es bis zu 360 Grad drehen und halten. Sie drehen sich in beide Richtungen. Der Schrittmotor wird von einem speziellen "Fahrer" gesteuert, ohne diesen Fahrer sind wir nicht in der Lage, den Schrittmotor direkt vom Raspberry Pi aus zu steuern. Der Schrittmotor ist besonders für präzise Einsatzanwendungen wie Robotik, 3D-Drucker und so weiter sehr nützlich. Er wird dort eingesetzt, wo eine präzise Bewegung gebraucht wird, die auf der Anzahl der "Schritte" basiert. In unserem Beispiel lernen wir, wie man den Schrittmotor steuert, indem wir ihm die Anzahl der Schritte geben, die er sich umdrehen soll.

### Pin-Definition





## Benötigte Hardware

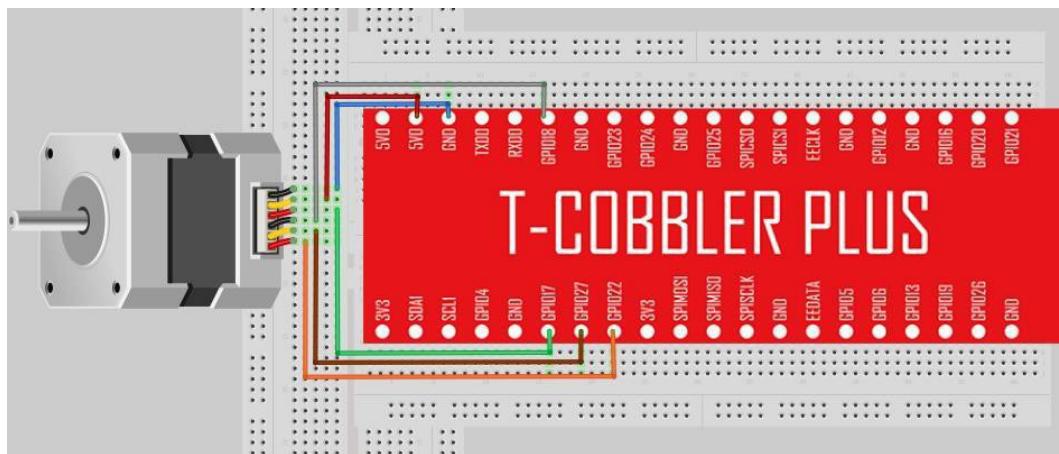
Für dieses Beispiel benötigen wir 2 spezielle Komponenten, die erste ist natürlich der Schrittmotor, aber die zweite ist die ULN2003 Schrittmotorplatine. Wie bereits erwähnt, werden wir ohne diese Fahrerplatine den Schrittmotor nicht direkt steuern können.

Materialdiagramm	Materialname	Anzahl (Betrag)
	Schrittmotor	1
	ULN2003 Schritt Motortreiberplatine	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Bitte beachten Sie, dass das Diagramm nicht 100% korrekt ist. Wir müssen zuerst den Schrittmotor mit dem Treiber verbinden und dann dem Diagramm folgen, um den Treiber mit dem Raspberry-Pi oder T-Cobbler plus zu verbinden.



## Verbindung

Wir haben insgesamt 4 OUTPUT GPIO Pins (IN1,2,3 und 4), die wir verwenden werden, um die Schritte des Schrittmotors zu steuern.

Schrittmotor	Raspberry Pi
IN1	GPIO12
IN2	GPIO16
IN3	GPIO20
IN4	GPIO21
5v	5v
GND	GND



## Codeübersicht

Werfen wir einen Blick auf unseren Code, der Hauptcode initialisiert das Schrittmotorobjekt und dann führen wir eine Demo des anderen Typs aus: Zuerst wird die Demo 1 Schritt drehen, dann 20 Schritte, dann eine Vierteldrehung um 90 Grad und danach stoppt der Motor. Wie wir sehen können, wir 2 Arten von Funktionen:

1. turnSteps, die den Motor um die Anzahl der Stopps drehen
2. turnDegrees, die wir Grad wie 90,180,360 und es wird es umdrehen.

Wir können beide Funktionen verwenden, um das gleiche Ergebnis auf unterschiedliche Weise zu archivieren. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 stepper.py
```

```
121 def main():
122
123     print("moving started")
124     motor = Stepmotor()
125     print("One Step")
126     motor.turnSteps(1)
127     time.sleep(0.5)
128     print("20 Steps")
129     motor.turnSteps(20)
130     time.sleep(0.5)
131     print("quarter turn")
132     motor.turnDegrees(90)
133     print("moving stopped")
134     motor.close()
135
136 if __name__ == "__main__":
137     main()
```

## Anwendungseffekt

Das Programm wird dazu führen, dass der Schrittmotor verschiedene Bewegungen ausführt. Sie können das Intervall des Schrittmotors ändern, um ihn langsamer oder schneller zu bewegen.



## Aufgabe 13 - Ultraschall



### Einleitung

In dieser Aufgabe erfahren wir mehr über den Ultraschall-Ranging-Sensor. Der Ultraschall ist wahrscheinlich der günstigste Sensor auf dem Markt, er eröffnet viele Möglichkeiten, Anwendungen in vielen verschiedenen Bereichen zu entwickeln, von Robotik über IoT bis hin zu Smart City! Der Ultraschall-Bereichssensor ist in der Lage, den Bereich von Punkt 0 (die Sensorposition) bis zum vor ihm stehenden Objekt zu erkennen, er kann bis zu 5 Meter voraus detektieren. In dieser Aufgabe lernen wir, wie Sie den Ultraschallsensor steuern und den Abstand zum Objekt vor unserem Sensor messen.



## Benötigte Hardware

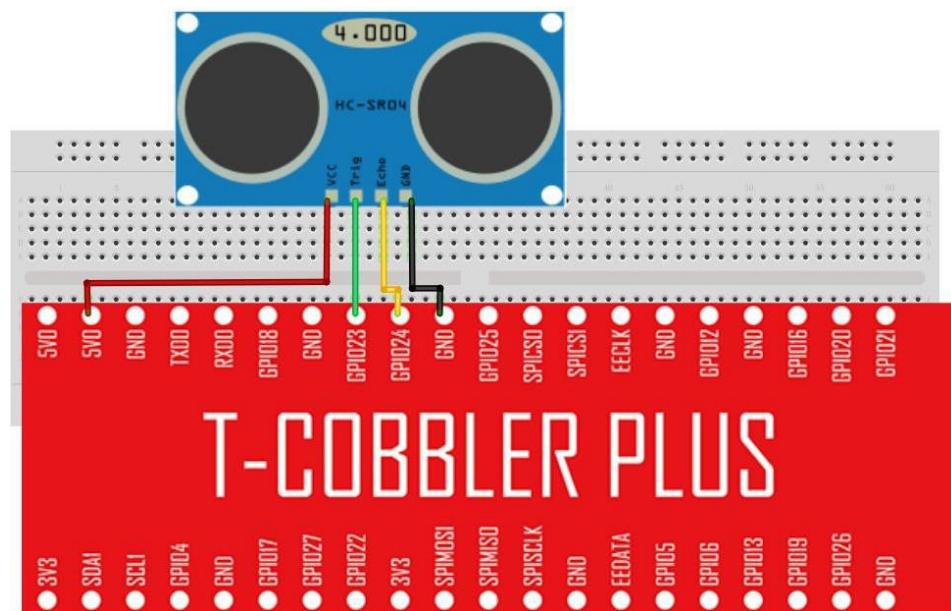
Die Hauptkomponente, die wir benötigen, ist der HCSR04 Ultraschallsensor, es ist schwierig, ihn nicht zu erkennen, da er wie 2 Roboteraugen aussieht. Der Sensor hat 2 verschiedene Arten von Verbindungen, die wir auf der nächsten Seite erklären werden, bitte achten Sie darauf.

Materialdiagramm	Materialname	Anzahl (Betrag)
	HCSR04 Ultraschallsensor	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere

## Verbindungsdiagramm

Im Verbindungsdiagramm haben wir die 2 Basisstifte, die eine ist die negative und die andere ist die positive (schwarze und rote Drähte). Die anderen Drähte sind die Steuerdrähte namens TRIG und ECHO. Wir verwenden sie, um einen Impuls zu senden und die Antwort zu empfangen, so dass wir die Entfernung durch die Zeit, die es dauert, bis der Impuls zurück zum Sensor zurück kommt, gemessen werden kann.

Achten Sie darauf, sie richtig zu verbinden, TRIG geht zu GPIO23 und ECHO geht zu GPIO24.



## Verbindung

Komponente	Raspberry Pi
Vcc	5v
Trig	GPIO23
Echo	GPIO24
GND	GND



## Codeübersicht

Der Abstandssensorcode ist sehr geradlinig. Wir messen den Puls und wie lange es dauert, um zurückzukommen, damit wir den Abstand vom Sensor zum Objekt vor ihm kennen. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 distance.py
```

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 # Author : www.modmypi.com
4 # Link: https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi
5
6 import RPi.GPIO as GPIO
7 import time
8
9 # we set GPIO mode as GPIO.BCM
10 GPIO.setmode(GPIO.BCM)
11
12 # we set the pins for the TRIG and ECHO
13 TRIG = 23
14 ECHO = 24
15
16 # measuring distance
17 print("Distance Measurement In Progress")
18 GPIO.setup(TRIG,GPIO.OUT)
19 GPIO.setup(ECHO,GPIO.IN)
20
21 GPIO.output(TRIG, False)
22 print("Waiting For Sensor To Settle")
23 time.sleep(2)
24
25 GPIO.output(TRIG, True)
26 time.sleep(0.00001)
27 GPIO.output(TRIG, False)
28
29 while GPIO.input(ECHO)==0:
30     pulse_start = time.time()
31
32 while GPIO.input(ECHO)==1:
33     pulse_end = time.time()
34
35 pulse_duration = pulse_end - pulse_start
36
37 distance = pulse_duration * 17150
38
```

## Anwendungseffekt

Der Abstandssensor misst den Abstand und gibt ihn anschliessend auf der Konsole in cm aus.



## Aufgabe 14 – Touch-Sensor



### Einleitung

In dieser Aufgabe erfahren wir mehr über den Touch-Sensor. Der Touch-Sensor ist ähnlich der Taste, die wir in den vorherigen Aufgaben verwendet haben. Der Unterschied ist, dass anstatt einer physischen Taste eine kapazitive Touch-Oberfläche verwendet wird. Die meisten Produkte verwenden heute kapazitive Berührung anstelle der tatsächlichen Taste, so dass die Verwendung des Touch-Sensors absolut sinnvoll für die nächsten Projekte ist. Der Touch-Sensor an sich ist ein wenig langweilig, so dass wir versuchen, etwas Spass mit ihm zu haben. Wir verbinden unsere RGB-LED mit dem Touch-Sensor und sobald eine Berührung erkannt wird, schalten wir die RGB-LED ein.



## Benötigte Hardware

Die wichtigsten Hardwareteile, die wir für diese Aufgabe benötigen, sind der Touch-Sensor und die RGB-LED, die wir zuvor in der vorherigen Aufgabe verwendet haben.

Der Touch-Sensor ist sehr leicht zu unterscheiden, Sie werden sehen, dass es eine Art "Fingerdruck"-Malerei darauf gibt, der Fingerabdruck steht für den Finger, der den Sensor berührt, um die Berührung zu erkennen.

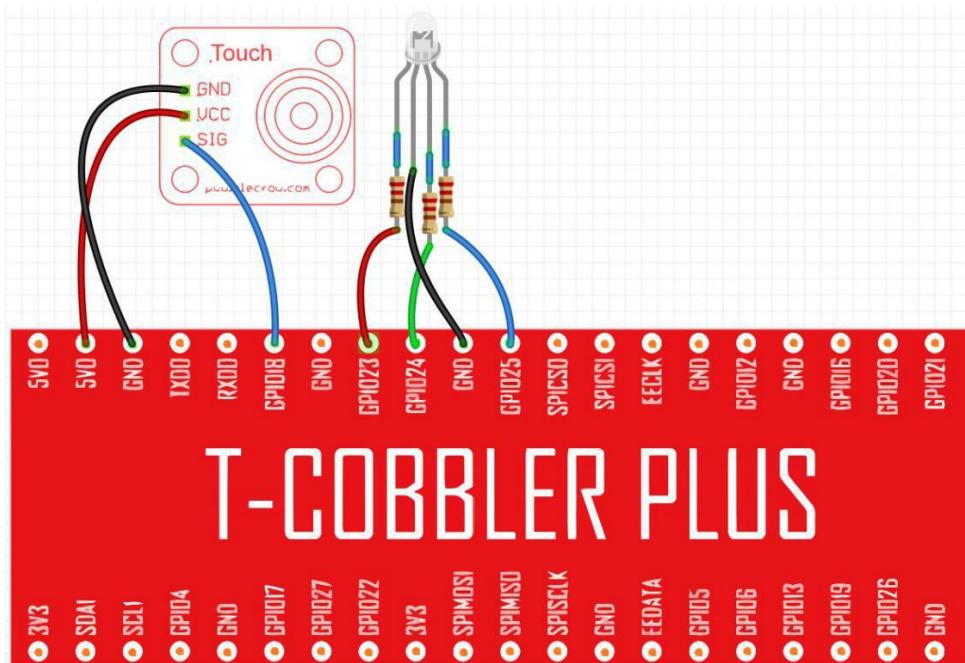
Wir benötigen auch 3 220/330ohm Widerstände, um mit der RGB-LED zu gehen (nicht für den Touch-Sensor) Sie können entweder 220ohm oder 330ohm verwenden oder kombinieren sie beide, alles wird gut funktionieren.

Materialdiagramm	Materialname	Anzahl (Betrag)
	Touch-Sensor	1
	RGB-LED	1
	220/330O Widerstand	3
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrer e



## Verbindungsdiagramm

Im Verbindungsdiagramm können wir sehen, dass wir die RGB-LED wie in früheren Aufgaben an GPIO17,27 und 22 anschliessen werden. Der Touch-Sensor hat 3 verschiedene Pins, einer ist der positive (VCC) ein anderer ist negativ (GND) und GPIO Pin, den wir als GPIO konfigurieren werden. IN, um einen Fingertouch über den Sensor zu erhalten.



## Verbindung

RGB-LED	Raspberry Pi
R	GPIO17
G	GPIO27
B	GPIO22
GND	GND

Touch-Sensor	Raspberry Pi
GND	GND
Vcc	Vcc
Sig	GPIO18



## Codeübersicht

In unserem Beispiel verwenden wir sowohl RGB-LED als auch Touch-Sensor, beim Drücken des Touch-Sensors wählen wir nach dem Zufallsprinzip eine Farbe: rotgrün oder blau. Dann drehen wir die LED der zufällig gewählten Farbe. Wenn wir aufhören, den Touch-Sensor zu berühren, schaltet sich die LED aus. Komplizieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 touch.py
```

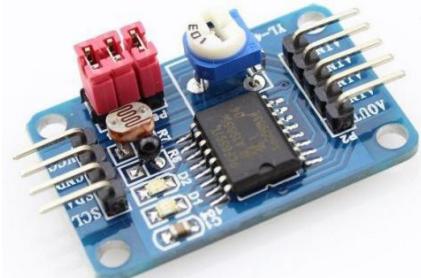
```
67  def main():
68      # Initialize RGB Class
69      RGB_LED = RGB()
70
71      # define touch pin
72      touch_pin = 18
73
74      # set GPIO pin to INPUT
75      GPIO.setup(touch_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
76
77      colors = ["red","green","blue"]
78
79  try:
80      # select random color to turn on
81      random_color = random.choice(colors)
82      while True:
83          # check if touch detected
84          if(GPIO.input(touch_pin)):
85              # select random color to turn on
86              RGB_LED.turn_on(random_color)
87          else:
88              RGB_LED.turn_off(random_color)
89      random_color = random.choice(colors)
```

## Anwendungseffekt

Beim Berühren des Berührungssensors wird eine Farbe nach dem Zufallsprinzip vom Skript ausgewählt, entweder rot, grün oder blau. Dann schaltet sich die LED ein. Wenn der Touch-Sensor losgelassen wird, schaltet sich die LED aus.



## Aufgabe 15 – PCF8591 MODULE



### Einleitung

Das PCF8591 Modul ist ein 8-Bit A/D-D/A Converter mit vier analogen Eingängen, einem analogen Ausgang und einer seriellen I2C-Bus-Schnittstelle. Das PCF8591-Modul verfügt über einen I2C-Pin-Header auf der einen Seite und einen I2C-Anschluss auf der gegenüberliegenden Seite. Die Platine unterstützt auch I2C-Kaskadierung, so dass die Verwendung von mehreren Modulen mit dem I2C-Bus zur gleichen Zeit durch den Anschluss der Stift-Header und Stecker möglich ist.

Bitte beachten Sie, dass wir das PCF8591-Modul für viele unsere Aufgaben ab diesem Zeitpunkt verwenden werden. Wir verbinden Module wie lichtdurchleuchtende Thermistor, um ihren Wert zu erhalten, der sich je nach Umgebung ändert.

Die erste Aufgabe in Bezug auf die PCF8591 ist nur zu zeigen, wie es zu verwenden und richtig anschliessen.



## Benötigte Hardware

Für diese Aufgabe benötigen wir das PCF8591-Modul.

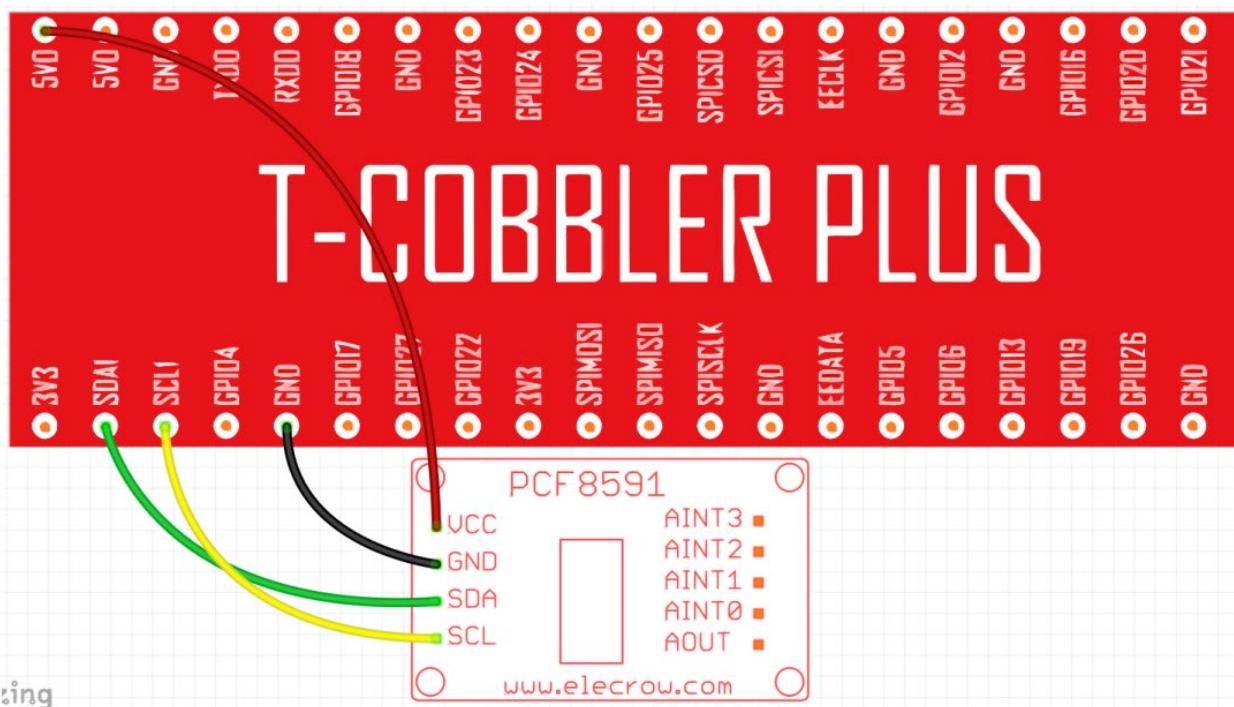
Es sind keine Widerstände oder andere Teile erforderlich. Aber wie wir bereits erwähnt haben, werden wir das PCF8591-Modul später mit anderen Komponenten kombinieren, um deren Daten und sich ändernde Werte zu erhalten.

Materialdiagramm	Materialname	Anzahl (Betrag)
	PCF8591	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Das Verbindungsdiagramm ist ziemlich gerade, die PCF8591 haben seitenseitig und rechts, wir verwenden die linke Seite, um es mit unserem Raspberry Pi und der rechten Seite später zu verbinden, um mit Komponenten und Sensor zu verbinden. Die linke Seite kann als Ausgang und die rechte Seite als Eingabe behandelt werden.



## Verbindung

PCF8591	Raspberry Pi
Vcc	5v
GND	GND
Sda	SDA1
Scl	SCL1



## Codeübersicht

Der Code ist einfach und wir werden ihn nicht für die nächsten Aufgaben ändern, derselbe Code kann auch angewendet werden, wenn andere Komponenten verwendet werden. Mit der smbus-Bibliothek verbinden wir uns mit der Port-A0-Adresse, die wir in den nächsten Aufgaben verwenden werden, dann bekommen wir den Wert und geben ihn aus. Wenn keine Komponenten angeschlossen sind, ändert sich der Wert nur gering, aber wenn die Komponenten angehängt werden, wird der Wert basierend auf der Komponente geändert. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 pcf8591.py
```

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  # www.elecrow.com
4  import smbus
5  import time
6
7
8  # NOTE: the PCF8591 script can be used both with photoresistance example and flame sensor example
9  # please follow up the instructions on the PCF8591 schematic to connect the right sensors in the right way.
10
11 address = 0x48
12
13 A0 = 0x40
14 A1 = 0x41
15 A2 = 0x42
16 A3 = 0x43
17
18 bus = smbus.SMBus(1)
19
20 while True:
21     bus.write_byte(address,A0)
22     value = bus.read_byte(address)
23     print("Value: %s" % value)           67 von 114
24     time.sleep(0.1)
25
26
```

## Anwendungseffekt

Das Ausführen des Programms führt zu einer Wertänderung basierend auf dem PCF8591 selbst. Es ist noch keine Komponente angeschlossen.



## Ausgabe 16 – Flammen-Sensor



### Einführung

Der Flammensensor kann verwendet werden, um Feuer oder Licht anderer Wellenlängen von 760 nm bis 1100 nm Licht zu erkennen. Im Feuerlöschroboterspiel spielt die Flamme eine wichtige Rolle in der Sonde, die als Augen des Roboters verwendet werden kann, um Feuerquelle oder Fussball zu finden.

Die Betriebstemperatur des Flammensensors beträgt -25 Grad Celsius bis 85 Grad Celsius. Dies wird der erste Sensor sein, den wir an das PCF8591-Modul anschliessen werden, um die Daten zu erhalten, der Flammensensor kann nicht allein so verbunden werden, wir werden die PCF8591 als Hilfe verwenden, um seine Signale auszulesen.



## Benötigte Hardware

Hauptbestandteil sind der PCF8591 und der Flammensensor. Beim Anschliessen des Flammensors müssen wir auch einen 10K Ohm-Widerstand verwenden, stellen Sie bitte sicher, dass es 10K Ohm und nicht 10 Ohm ist.

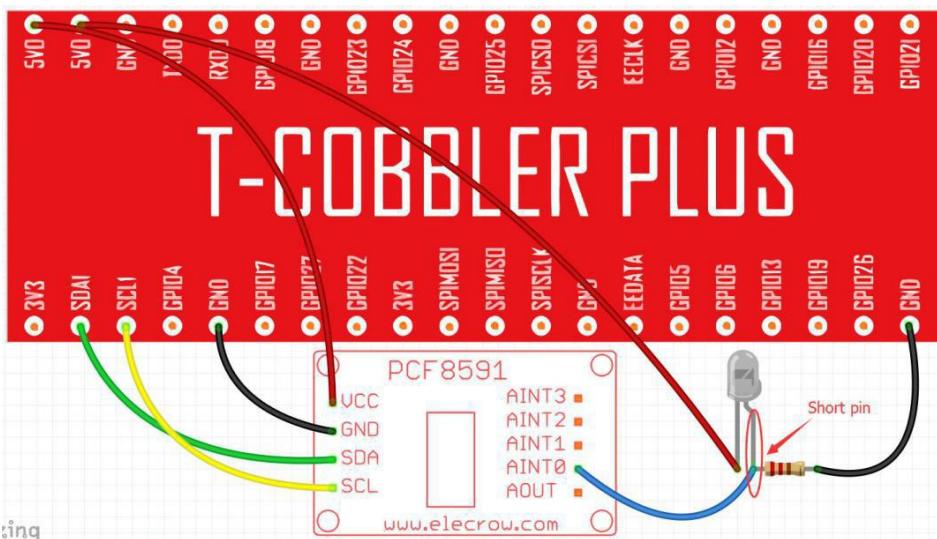
Der Widerstand geht direkt an den Flammensensor und dann an den Eingangsstift des PCF8591-Moduls, wie auf der nächsten Seite gezeigt wird.

Materialdiagramm	Materialname	Anzahl (Betrag)
	Flammensensor	1
	10K-Widerstand	1
	PCF8591	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Die Verbindung ist einfach, wir müssen den kurzen Pin des Flammensensors (den negativen Pin) an den Widerstand und an den GND-Pin sowie an den AINT0-Pin am PCF8591-Modul anschliessen. Der lange Stift des Flammensensors geht direkt auf 5V auf dem Raspberry Pi (es gibt 2 5V-Pins, so dass Sie einen für den PCF8591 und den anderen für den Flammensensor verwenden könnten). Wenn Sie vergessen haben, wie Sie das PCF8591-Modul an den Raspberry Pi anschliessen, lesen Sie die vorherige Aufgabe.



## Verbindung

Flammensensor	Raspberry Pi
Kurzer Pin	5v
Langer Pin	AINTO

PCF8591	Raspberry Pi
Vcc	5v
GND	GND
Sda	SDA1
Scl	SCL1



## Codeübersicht

Wir verwenden dasselbe Skript, das wir zuvor verwendet haben, indem wir den Flammensensor anbringen, indem wir Flamme entweder durch Feuerzeug oder Kerze einleiten, ändert sich der Wert. Wir können diesen Wert verwenden, um zu erkennen, ob es derzeit Flamme oder gar keine Flamme gibt.

Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 pcf8591.py
```

```
1  #!/usr/bin/python
2  # -*- coding:utf-8 -*-
3  # www.elecrow.com
4  import smbus
5  import time
6
7
8  # NOTE: the PCF8591 script can be used both with photoresistance example and flame sensor example
9  # please follow up the instructions on the PCF8591 schematic to connect the right sensors in the right way.
10
11 address = 0x48
12
13 A0 = 0x40
14 A1 = 0x41
15 A2 = 0x42
16 A3 = 0x43
17
18 bus = smbus.SMBus(1)
19
20 while True:
21     bus.write_byte(address,A0)
22     value = bus.read_byte(address)
23     print("Value: %s" % value)
24     time.sleep(0.1)
25
26
```

## Anwendungseffekt

Das Programm zeigt einen Wert an, der sich basierend auf dem Flammensensor ändert.



## Aufgabe 17 – Fotowiderstand



### Einleitung

Da der Widerstand des Sensors je nach Lichtmenge variiert, der er ausgesetzt ist, ändert sich die Ausgangsspannung mit der Lichtintensität. Es kann verwendet werden, um andere Module auszulösen. Dies wird unsere dritte Komponente sein, die wir mit dem PCF8591 Modul verbinden werden.



## Benötigte Hardware

Die Haupthardware-Komponente wird der Fotowiderstandssensor sein. Wir benötigen 10K Ohm Widerstand sowie, um den Fotowiderstandssensor mit dem PCF8591 Modul zu verbinden, stellen Sie sicher, dass Sie die 10K Ohm und nicht die 10 Ohm Widerstand versehentlich nehmen.

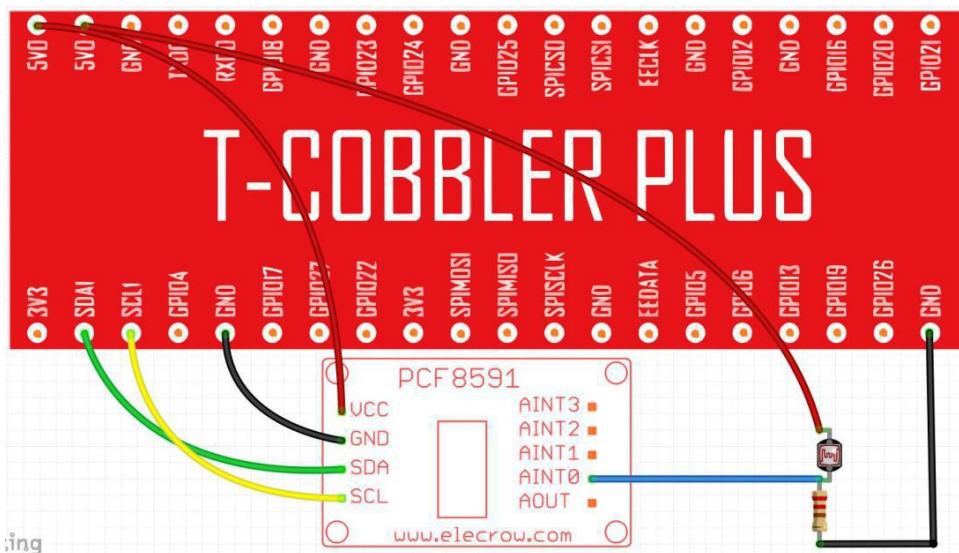
Wir verwenden die gleiche Verbindung für das PCF8591-Modul, wie wir sie in unseren vorherigen Aufgaben verwendet haben.

Materialdiagramm	Materialname	Anzahl (Betrag)
A photograph of a phototransistor component, showing its physical appearance with a metal lead and a plastic housing.	Fotowiderstandssensor	1
A photograph of a standard resistor component, showing its cylindrical shape and color coding.	10K-Widerstand	1
A photograph of the PCF8591 I2C interface module, which is a blue printed circuit board with various electronic components and pins.	PCF8591	1
A photograph of a Raspberry Pi Model B+ Board, showing the Broadcom SoC, RAM, and other onboard components.	Raspberry Pi Board	1
A photograph of the T-Cubbler Plus breadboard, which is a red and silver breadboard designed for prototyping.	T-Cubbler Plus	1
A photograph of a 40-pin ribbon cable, specifically a GPIO cable used for connecting the Raspberry Pi to external hardware.	40P GPIO Kabel	1
A photograph of a standard white solderless breadboard, used for temporary circuit prototyping.	Steckbrett	1
A photograph of several jumper wires, which are short lengths of wire used to connect different parts of a circuit.	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Das Diagramm ist dem vorherigen sehr ähnlich. Werfen wir einen Blick über die Fotowiderstandsstifte, es hat keine kurze oder lange. Wir müssen einen Pin an den 10K Ohm-Widerstand und dann an den GND sowie an den AINT0-Pin am PCF8591-Modul anschliessen. Der andere Pin, die positive Seite, die wir wählen, geht direkt auf 5V. Die PCF8591 Modulanschlüsse bleiben exakt gleich.



## Verbindung

Fotowiderstandssensor	Raspberry Pi
Pin 1	5v
Pin 2	AINTO

PCF8591	Raspberry Pi
Vcc	5v
GND	GND
Sda	SDA1
Scl	SCL1



## Codeübersicht

Dieses Mal nach dem Anbringen des Lichtsensors am PCF-Modul ändert sich der Wert basierend auf der Lichtmenge im Raum, versucht, das Licht zu schalten oder einzuschalten oder verschiedene Lichtquellen einzuführen, wie z. B. das Blitzlicht Ihres Mobiltelefons, um den Wert zu ändern.

Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 pcf8591.py
```

```
1  #!/usr/bin/python
2  # -*- coding:utf-8 -*-
3  # www.elecrow.com
4  import smbus
5  import time
6
7
8  # NOTE: the PCF8591 script can be used both with photoresistance example and flame sensor example
9  # please follow up the instructions on the PCF8591 schematic to connect the right sensors in the right way.
10
11 address = 0x48
12
13 A0 = 0x40
14 A1 = 0x41
15 A2 = 0x42
16 A3 = 0x43
17
18 bus = smbus.SMBus(1)
19
20 while True:
21     bus.write_byte(address,A0)
22     value = bus.read_byte(address)
23     print("Value: %s" % value)
24     time.sleep(0.1)
25
26
```

## Anwendungseffekt

Der gemessene Wert ändert sich basierend auf dem eingefallenen Licht. Je höher der Wert, desto höher ist die Lichtmenge, je geringer die Lichtmenge , desto niedriger wird der Wert.



## Ausgabe 18 – Thermistor



### Einleitung

Der Widerstand eines Thermistors steigt, wenn die Umgebungstemperatur sinkt, so dass der RPI die Spannung erkennen und so die aktuelle Temperatur berechnen kann. Der Erfassungsbereich dieses Sensors liegt zwischen -40 und 125 Grad Celsius bei einer Genauigkeit von  $\pm 1,5^{\circ}\text{C}$ . Wie in unseren vorherigen Aufgaben verwenden wir das PCF8591-Modul, um den Thermistor mit dem Raspberry-Pi zu verbinden und die Temperatur zu berechnen.

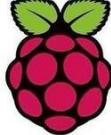


## Benötigte Hardware

Die Hauptkomponente in dieser Aufgabe ist der Thermistor, er sieht aus wie eine LED mit einem kleinen schwarzen Stift am Ende. Wir benötigen auch den 10K Ohm-Widerstand aus den vorherigen Aufgaben, um eine Verbindung zum Thermistor und GND-Pin herzustellen.

Bitte stellen Sie sicher, dass Sie 10K Ohm Widerstand nicht mit 10 Ohm Widerstand verwechseln.

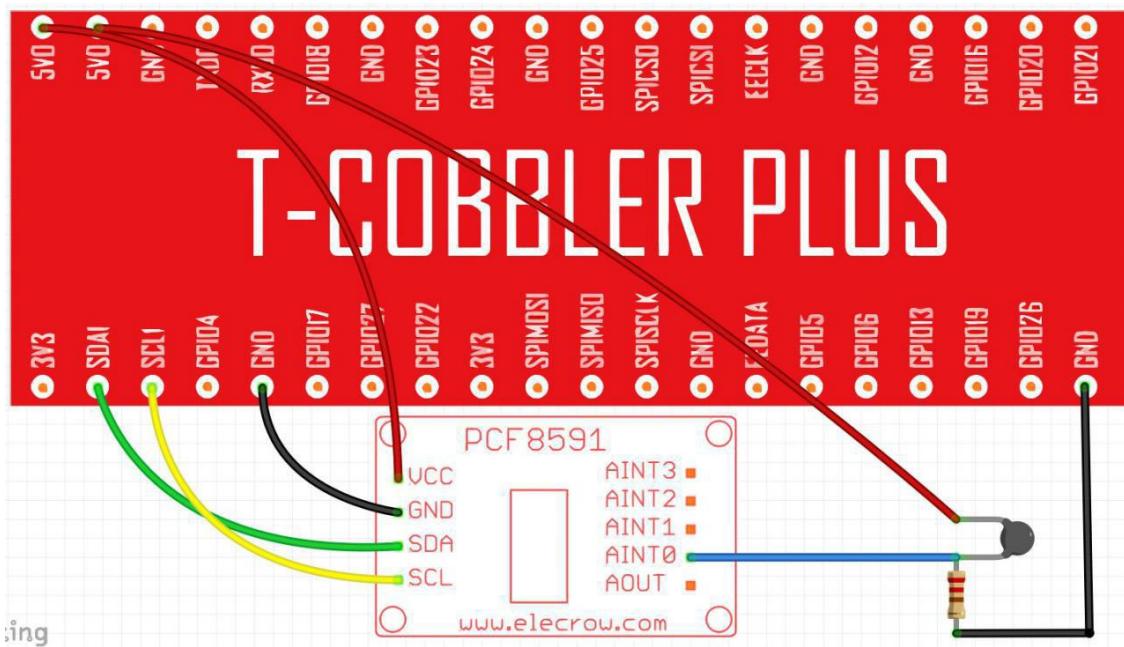
Und natürlich das Modul PCF8591, die wir verwenden, um den Wert von der Thermistor-Sensor zu erhalten.

Materialdiagramm	Materialname	Anzahl (Betrag)
	Thermistorsensor	1
	10K-Widerstand	1
	PCF8591	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Wie im Diagramm gezeigt, verwenden wir die PCF8591 wie zuvor. Der Thermistor hat keine spezifischen Pins, so dass ein Pin wir mit dem GND über den 10K Ohm Widerstand sowie an AINT0 am PCF8591 Modul verbinden. Der andere Pin verbinden wir direkt an 5V-Pin, es gibt 2 5V-Pins, so dass es nicht notwendig ist, sie an der gleichen Stelle zusammenzuschlieben.



## Verbindung

Thermistorsensor		Raspberry Pi
Pin 1		5v
Pin 2		AINT0

PCF8591	Raspberry Pi
Vcc	5v
GND	GND
Sda	SDA1
Scl	SCL1



## Codeübersicht

Wie in unseren bisherigen Beispielen ändert das PCF8591-Modul den Wert basierend auf dem Thermistor. Komplizieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 pcf8591.py
```

```
1  #!/usr/bin/python
2  # -*- coding:utf-8 -*-
3  # www.elecrow.com
4  import smbus
5  import time
6
7
8  # NOTE: the PCF8591 script can be used both with photoresistance example and flame sensor example
9  # please follow up the instructions on the PCF8591 schematic to connect the right sensors in the right way.
10
11 address = 0x48
12
13 A0 = 0x40
14 A1 = 0x41
15 A2 = 0x42
16 A3 = 0x43
17
18 bus = smbus.SMBus(1)
19
20 while True:
21     bus.write_byte(address,A0)
22     value = bus.read_byte(address)
23     print("Value: %s" % value)
24     time.sleep(0.1)
25
26
```

## Anwendungseffekt

Das laufende Programm zeigt je nach Temperatur des Thermistor einen unterschiedlichen Wert an, je höher die Temperatur, desto höher der Wert.



## Aufgabe 19 – Potentiometer



### Einleitung

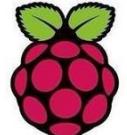
In diesem Beispiel verwenden wir ein Potentiometer, wir lesen seinen Wert mit einem analogen Eingang einer RPI-Platine und ändern die Blinkrate der eingebauten LED entsprechend. Der Analogwert des Widerstands wird als Spannung gelesen, da die analogen Eingänge so funktionieren. Für diese Aufgabe wie die vorherigen Aufgaben verwenden wir das PCF8591-Modul, um die Werte aus dem Potentiometer zu lesen.



## Benötigte Hardware

Die Haupthardware, die wir verwenden werden, ist das Potentiometer, im Gegensatz zu den vorherigen Beispielen für das Potentiometer werden wir nicht in der Notwendigkeit bei der Verwendung von 10K Ohm Widerstände aufgrund der Beständigkeit des Potentiometers selbst sein.

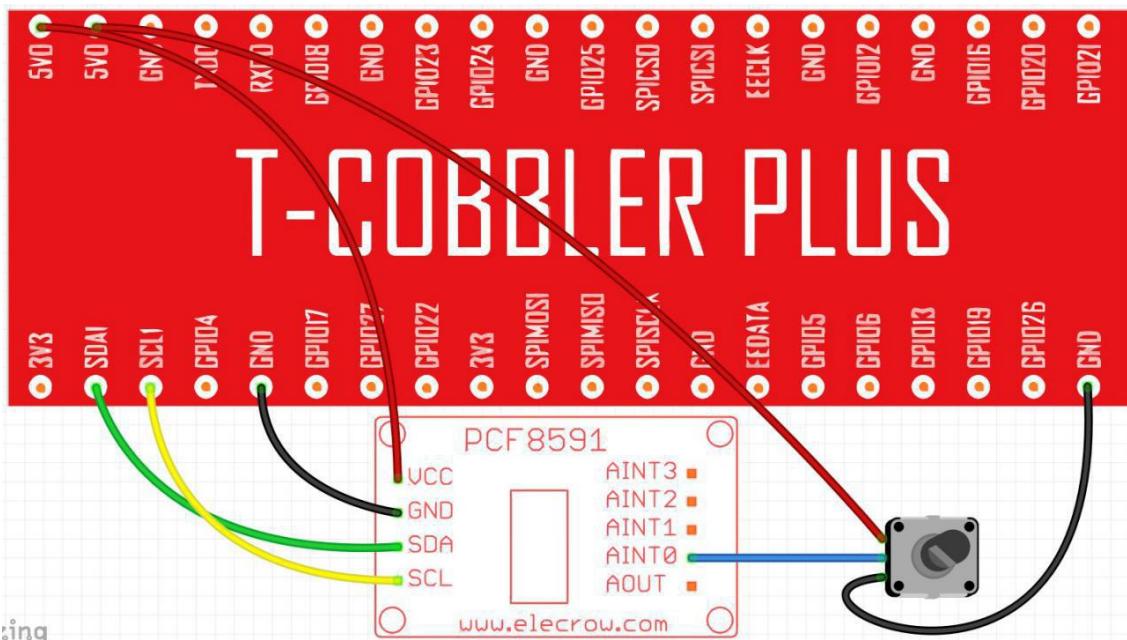
Und natürlich müssen wir auch das PCF8591-Modul verwenden, um den Wert aus dem Potentiometer abzulesen.

Materialdiagramm	Materialname	Anzahl (Betrag)
	Potentiometer	1
	PCF8591	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Das Diagramm bleibt das gleiche wie bei den vorherigen Aufgaben, aber wir verbinden den Widerstand jetzt nicht, da wir keinen Nutzen für das Potentiometer haben. Das Potentiometer hat 3 Pins, GND, VCC und AINT0 stellen Sie sicher, dass die richtigen Pins an den richtigen Stellen angeschlossen werden.



## Verbindung

Komponente	Raspberry Pi
Up Pin	5v
Mid Pin	AINTO

PCF8591	Raspberry Pi
Vcc	5v
GND	GND
Sda	SDA1
Scl	SCL1



## Codeübersicht

Dieses Mal, da wir das Potentiometer verwenden, wird es erforderlich sein, dass wir einige manuelle Aktionen durchführen, also das Potentiometer entweder rechts oder links zu. Auf diese Weise können wir mehrere Dinge wie zum Beispiel die Helligkeit von LEDs steuern. Versuchen Sie, es umzudrehen und zu sehen, wie sich der Wert ändert.

Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 pcf8591.py
```

```
1  #!/usr/bin/python
2  # -*- coding:utf-8 -*-
3  # www.elecrow.com
4  import smbus
5  import time
6
7
8  # NOTE: the PCF8591 script can be used both with photoresistance example and flame sensor example
9  # please follow up the instructions on the PCF8591 schematic to connect the right sensors in the right way.
10
11 address = 0x48
12
13 A0 = 0x40
14 A1 = 0x41
15 A2 = 0x42
16 A3 = 0x43
17
18 bus = smbus.SMBus(1)
19
20 while True:
21     bus.write_byte(address,A0)
22     value = bus.read_byte(address)
23     print("Value: %s" % value)
24     time.sleep(0.1)
25
26
```

## Anwendungseffekt

Durch das Ausführen des Programms wird es den Wert des Potentiometers gezeigt. Um den Wert zu ändern, muss das Potentiometer physisch nach links oder rechts gedreht werden.



## Ausgabe 20 – Füllstandsüberwachung



### Einleitung

Dies ist ein Füllstandmessung, es ist relativ einfach zu realisieren. Wir müssen nur den Wert des Analogports (A0 oder andere) lesen und dann in einen Prozentsatz konvertieren. Wir verwenden das PCF8591-Modul auch für das Beispiel zur Wasserstandsüberwachung.

### Spezifikation

Betriebsspannung: DC3-5V

Betriebsstrom: weniger als 20mA Sensortyp: Analog

Produktionsprozess: FR4 doppelseitige HASL Luftfeuchtigkeit: 10% -90% nicht kondensierende  
Erfassungsfläche: 40mmx16mm Produktabmessungen: 62mmx20mmx8mm



## Benötigte Hardware

Wir benötigen den Wassersensor als Haupthardwarekomponente und schliessen ihn an den PCF8591 an, um den Wert zu erhalten, sobald das Wasser Kontakt aufgenommen hat.

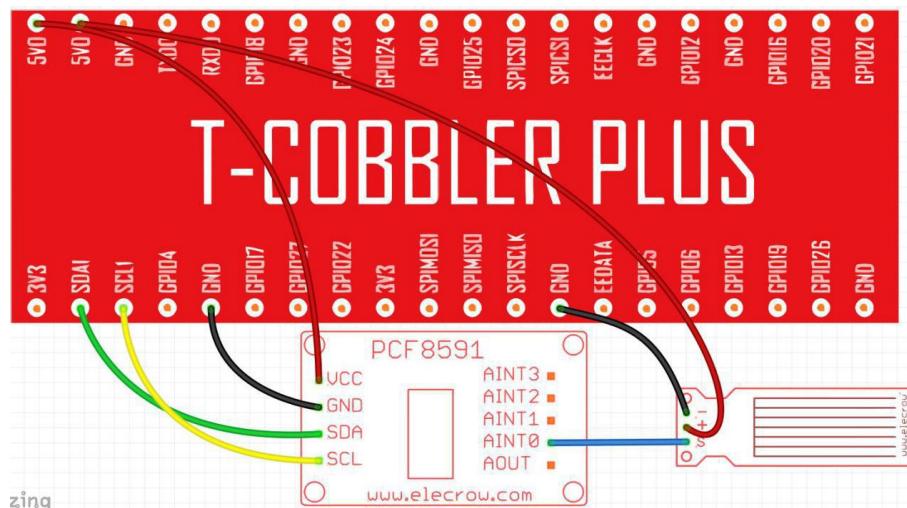
Wie unsere bisherigen Beispiele haben wir für den Wassersensor keinen Nutzen in 10K Ohm Widerständen.

Materialdiagramm	Materialname	Anzahl (Betrag)
	Wassersensor	1
	PCF8591	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Der Wassersensor hat 3 Pins, wir verbinden das Negativ mit GND, das Positive mit 5V und den S-Pin, den wir für den GPIO bis AINT0 verwenden. Sobald der Stromkreis durch Bewässerung die Oberfläche des Wassersensors berührt, sendet er ein Signal an den AINT0-Pin und wir werden in der Lage sein, die Wassermenge über den Sensor zu kennen.



## Verbindung

Wassersensor	Raspberry Pi
Negativ (-)	GND
Positiv (+)	5v
S	AINTO

PCF8591	Raspberry Pi
Vcc	5v
GND	GND
Sda	SDA1
Scl	SCL1



## Codeübersicht

Für dieses Beispiel, wenn wir den Wassersensor verwenden, ändert sich der Wert, indem wir Wassertropfen in den Sensor einführen. Je mehr Wasser, desto höher sind die Werte. Aber bitte beachten Sie: Der Wassersensor ist nicht wasserdicht und sollte nicht vollständig unter Wasser gesetzt werden. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 pcf8591.py
```

```
1  #!/usr/bin/python
2  # -*- coding:utf-8 -*-
3  # www.elecrow.com
4  import smbus
5  import time
6
7
8  # NOTE: the PCF8591 script can be used both with photoresistance example and flame sensor example
9  # please follow up the instructions on the PCF8591 schematic to connect the right sensors in the right way.
10
11 address = 0x48
12
13 A0 = 0x40
14 A1 = 0x41
15 A2 = 0x42
16 A3 = 0x43
17
18 bus = smbus.SMBus(1)
19
20 while True:
21     bus.write_byte(address,A0)
22     value = bus.read_byte(address)
23     print("Value: %s" % value)
24     time.sleep(0.1)
25
26
```

## Anwendungseffekt

Durch Ausführen des Skripts ändert sich der Wert basierend auf der Menge an Wassertropfen auf dem Sensor, je höher die Wassermenge, desto höher ist der Wert.



## Ausgabe 21 – Joystick



### Einleitung

Dieses Experiment soll lernen, wie man den Joystick des analogen und digitalen Outputs verwendet. Das Joystick-Experiment wird unsere letzte Aufgabe mit dem PCF8591 Modul sein. Wie wir in unseren vorherigen Aufgaben gelernt haben, ist das PCF8591-Modul für viele Szenarien sehr nützlich. Das Joystick-Beispiel ist ein sehr gutes, ein sehr beliebter Anwendungsfall für Robotik und Anwendungen, die Steuern oder Bewegung erfordern. Durch den Abschluss der PCF8591 Aufgaben, hoffen wir, dass Sie in der Lage sein werden, jede analoge Schaltung für jedes Projekt zu erstellen.



## Benötigte Hardware

Für die Aufgabe benötigen wir das Joystick-Modul, das sehr leicht zu erkennen ist.

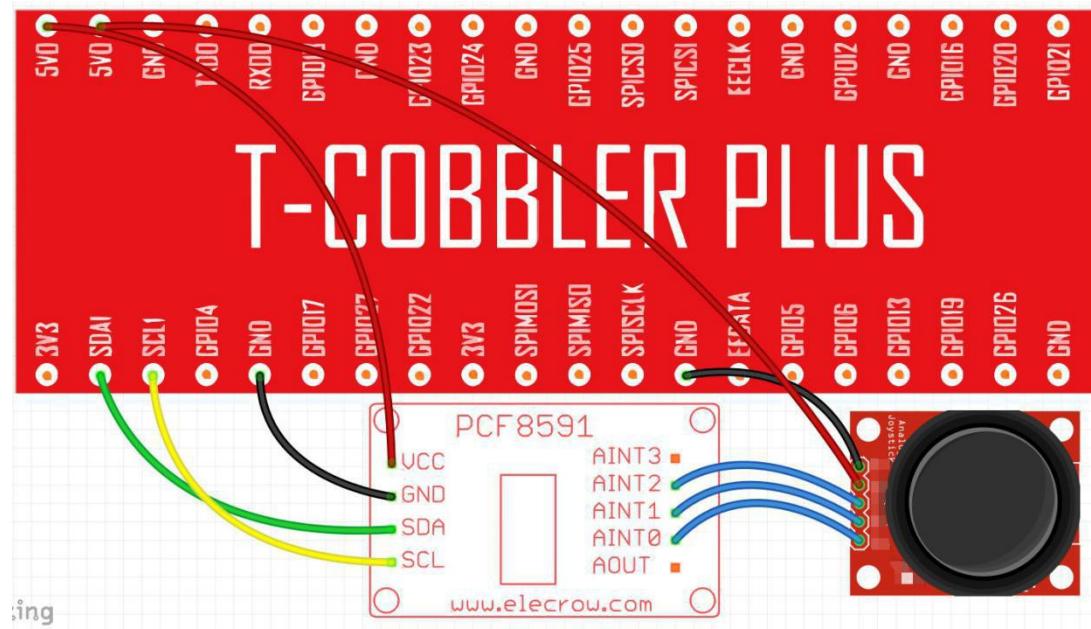
Wir benötigen auch das PCF8591-Modul, das mit der Schaltung verbunden ist, wie die vorherigen Aufgaben. Auch für diese Aufgabe brauchen wir keine Widerstände.

Materialdiagramm	Materialname	Anzahl (Betrag)
A black and silver joystick module with two pins visible at the bottom.	Joystick-Modul	1
A blue printed circuit board with various components and a yellow ribbon cable attached.	PCF8591	1
A pink Raspberry Pi Model B+ board with a blackberry logo graphic.	Raspberry Pi Board	1
A red T-shaped breadboard with a metal clip at the top.	T-Cubbler Plus	1
A brown ribbon cable with 40 pins, used for connecting the Raspberry Pi to the breadboard.	40P GPIO Kabel	1
A standard white breadboard with a grid of holes for component placement.	Steckbrett	1
A bundle of various colored jumper wires.	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Der Joystick hat 5 Pins, 2 davon sind die GND und der VCC, der an 5V-Pin geht. Der Rest sind VRX, VRY und SW. Stellen Sie sicher, dass diese Pins an der richtigen Stelle auf dem PCF8591 Modul angeschlossen werden. SW geht an AINT0, VRY geht an AINT1 und VRX geht an AINT2.



## Verbindung

Joystick	Raspberry Pi
GND	GND
5v	5v
VRX	AINT2
kostenlos	AINT1
Sw	AINT0

PCF8591	Raspberry Pi
Vcc	5v
GND	GND
Sda	SDA1
Scl	SCL1



## Codeübersicht

Dieses Mal beachten Sie, dass wir im Gegensatz zu den vorherigen Beispielen eine Verbindung zu 3 Pins des PCF8591-Moduls anstelle eines herstellen müssen. Wir müssen eine Verbindung zu 0,1 und 2 herstellen. (eine für jede Richtung, in die sich der Joystick bewegt). Danach sollte es gut funktionieren mit einem neuen Skript, das wir speziell für den **pcf8591** namens **pcf8591\_joystick.py**

Komplizieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 pcf8591_joystick.py
```

```
1 #!/usr/bin/python
2 # -*- coding:utf-8 -*-
3 # www.elecrow.com
4 import smbus
5 import time
6
7
8 # NOTE: the PCF8591 script can be used both with photoresistance example and flame sensor example
9 # please follow up the instructions on the PCF8591 schematic to connect the right sensors in the right way.
10
11 address = 0x48
12
13 A0 = 0x40
14 A1 = 0x41
15 A2 = 0x42
16 A3 = 0x43
17
18 bus = smbus.SMBus(1)
19
20 while True:
21     bus.write_byte(address,A0)
22     value = bus.read_byte(address)
23     print("Value: %s" % value)
```

## Anwendungseffekt

Das Skript zeigt Werte an, die sich basierend auf der Joystickrichtung ändern. Versuchen Sie, es in verschiedene Richtungen zu bewegen, um zu sehen, wie sich der Wert ändert.



## Ausgabe 22 – IR-Fernbedienung

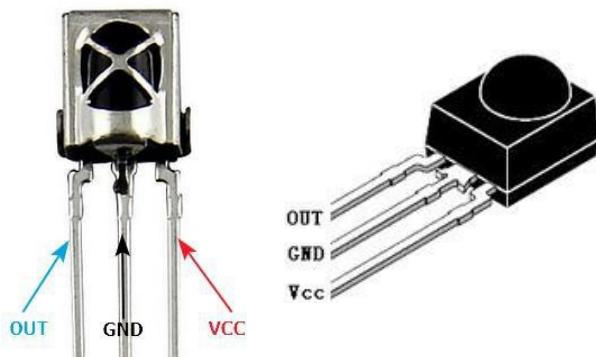


### Einleitung

In dieser Aufgabe verwenden wir die **lirc-Bibliothek**, um Infrarotsignale zu lesen, die durch Tasten der Fernbedienung zurückgegeben werden, und sie in Schaltflächenwerte zu übersetzen. Wenn eine Taste gedrückt wird, sendet der IR-Sender in der Fernbedienung die entsprechenden IR-Codierungssignale aus. Auf der anderen Seite, wenn der IR-Empfänger bestimmte Codierungssignale empfängt, wird er sie dekodieren, um zu identifizieren, welche Taste gedrückt wird.

Bevor Sie diese Aufgabe durchgehen, stellen Sie sicher, dass Sie die Installationsanforderungen befolgt haben, die wir am Anfang dieses Dokuments eingeführt haben.

### IR-Pin-Definition





## Benötigte Hardware

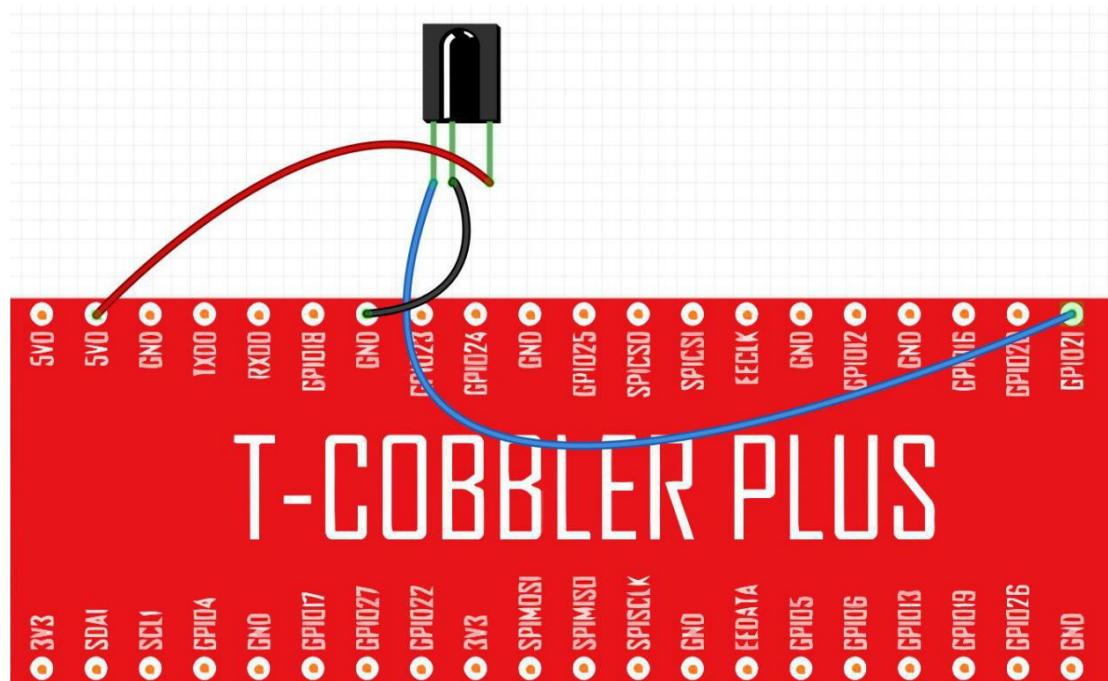
Für diese Aufgabe müssen wir den IR-Empfänger verwenden, der wie eine kleine LED mit schwarzem Stift am Ende aussieht (verwechseln Sie ihn nicht mit dem Flammensensor, den wir zuvor verwendet haben, es hat 2 Pins) sowie die Fernbedienung, um die Signale an den IR-Empfänger zu senden, um sie zu akzeptieren.

Materialdiagramm	Materialname	Anzahl (Betrag)
	IR-Fernbedienung	1
	IR-Empfänger	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Damit die Fernbedienung das Signal sendet, brauchen wir nur die Fernbedienung selbst. Wir müssen den IR-Empfänger an die GPIO-Pins anschliessen. Der IR-Empfänger hat 3 Pins, GND und VCC (VCC geht auf 5V) und OUT Pin den wir auf GPIO21 verbinden werden, um die Befehle von der IR-Fernbedienung zu empfangen.



## Verbindung

Und	Raspberry Pi
OUT	GPIO21
GND	GND
Vcc	5v



## Codeübersicht

Bevor wir beginnen, stellen Sie sicher, dass Sie die Treiber lirc und python-lirc zu Beginn unserer Aufgaben installiert haben. Ohne die Treiber funktioniert die Aufgabe nicht richtig und Sie können Ihre IR-LED nicht verwenden. Wir erhalten Daten vom die IR-LED und dann konvertieren wir die Daten. Nach der Konvertierung sind wir in der Lage zu sagen, welche Taste genau auf der IR-Fernbedienung gedrückt wurde. Dann drucken wir es mit dem Befehl print(Out[0])

Komplizieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python2 IR.py
```

```
6  GPIO.setmode(11)
7  GPIO.setup(17, 0)
8  GPIO.setup(18, 0)
9  PORT = 42001
10 HOST = "localhost"
11 Socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12
13 Lirc = lirc.init("keys")
14 #lirc.set_blocking(False, Lirc)           # Un-Comment to stop nextcode() from waiting for a signal ( will return empty array when
15
16 def handler(signal, frame):
17     Socket.close()
18     GPIO.cleanup()
19     exit(0)
20
21 signal.signal(signal.SIGTSTP, handler)
22
23 def sendCmd(cmd):
24     n = len(cmd)
25     a = array('c')
26     a.append(chr((n >> 24) & 0xFF))
27     a.append(chr((n >> 16) & 0xFF))
28     a.append(chr((n >> 8) & 0xFF))
29     a.append(chr(n & 0xFF))
30     Socket.send(a.tostring() + cmd)
31
32 while True:
33
34     Out = lirc.nextcode()
35     print(Out[0])
```

## Anwendungseffekt

Das Ausführen des Programms ermöglicht es uns, den IR-Code durch die IR-Fernbedienung zu lesen, drücken Sie die IR-Fernbedienungstasten, um sie auf dem Terminalbildschirm zu sehen, nachdem das Python-Skript sie mit dem IR-Empfänger erfasst hat.



## Aufgabe 23 – IR-Fernbedienung mit LED



### Einleitung

In der vorherigen Aufgabe haben wir gelernt, wie man die Befehle von der IR-Fernbedienung akzeptiert und sie über unseren Python-Code auf dem Raspberry-Pi empfängt.

Aber was ist das Interessante daran?

Während dieser Aufgabe werden wir das IR-Fernbedienungsprojekt ein wenig weiterbringen und wir werden die LED mit der IR-Fernbedienung steuern, wir werden lernen, wie man eine bestimmte Taste so konfiguriert, dass eine bestimmte Funktion zum Drehen unserer LED verwendet wird, sobald sie gedrückt ist.



## Benötigte Hardware

Für die Hardware benötigen wir die IR-Fernbedienung und den IR-Empfänger.

Für die LED benötigen wir eine kleine LED (Sie können jede gewünschte Farbe wählen) und 220 oder 330Ohm Widerstand, um die LED beim Einstecken nicht auszubrennen.

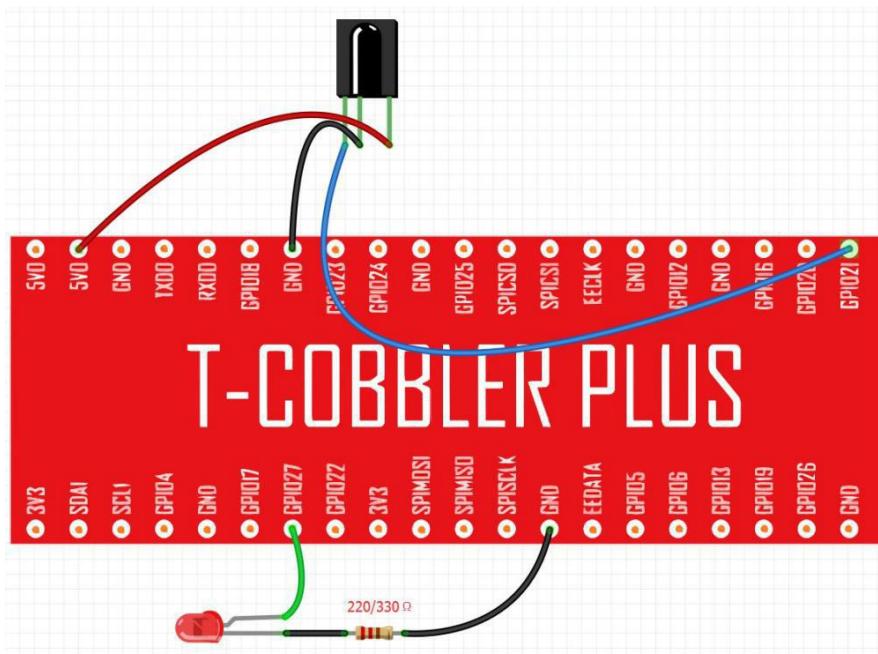
Achten Sie darauf, 220 oder 330ohm Widerstand zu verwenden, nicht K Widerstand.

Materialdiagramm	Materialname	Anzahl (Betrag)
	IR-Fernbedienung	1
	IR-Empfänger	1
	Led	1
	220/330ohm Widerstand	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Wir halten die Verbindung aus unserer vorherigen Aufgabe in Bezug auf den IR-Empfänger, aber dieses Mal schliessen wir auch die LED an. Die LED hat 2 Stifte, langen Stift und kurzen Stift. Der lange Pin ist das Positive, dass wir uns direkt mit GPIO27 verbinden und der kurze Pin geht durch den Widerstand zur GND-Verbindung am Raspberry Pi.



## Verbindung

Led	Raspberry Pi
Langer Pin (+)	GPIO27
Kurzer Pin (-)	GND

Und	Raspberry Pi
OUT	GPIO21
GND	GND
Vcc	5v



## Codeübersicht

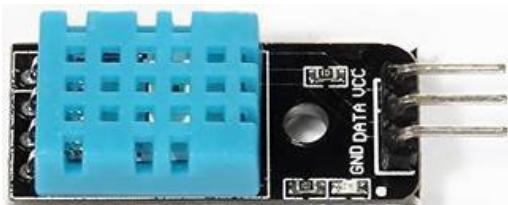
Tipps: Siehe die vorherige Demo (Schritt 4 bis Schritt 7).

## Anwendungseffekt

Läuft das Programm, schaltet eine LED eine Sekunde lang ein.



## Aufgabe 24 – DHT 11



### Einleitung

In dieser Aufgabe erfahren Sie, wie Sie das DHT11-Modul verwenden, das einfach und einfach zu bedienen ist. Das DH11-Modul ist ein kombiniertes Modul für Temperatur und Luftfeuchtigkeit, es ist erschwinglich und relativ genau. Es ist sehr nützlich in mehreren IoT-Projekten wie Smart Home oder Umgebungsüberwachung. Es kann sowohl im Innen- als auch im Außenbereich verwendet werden.



## Benötigte Hardware

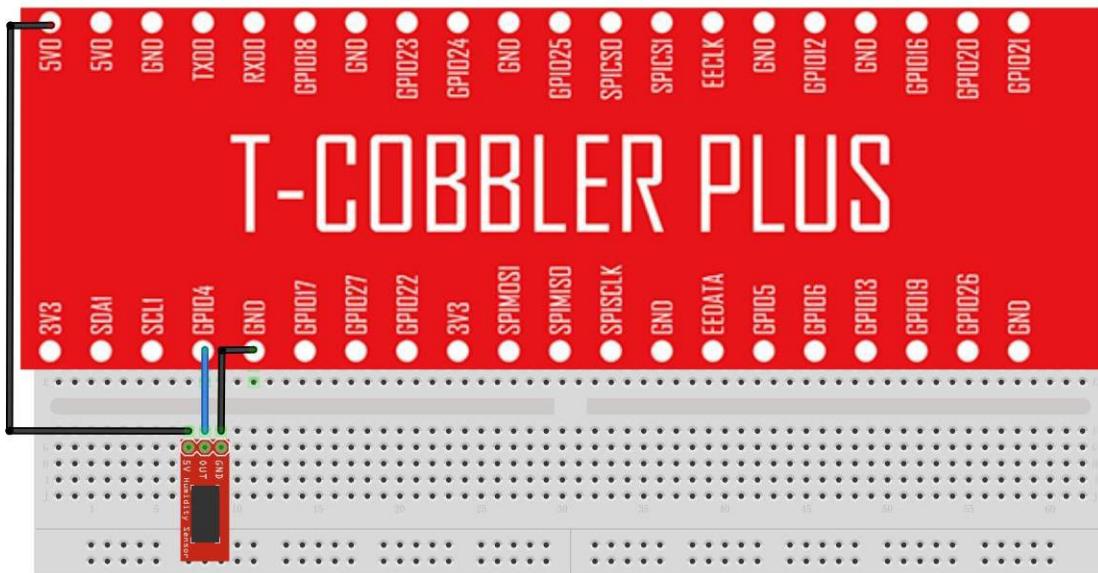
Die Hauptkomponente, die wir verwenden werden, ist die DHT11, es kann leicht als seine blaue Deckschicht erkennbar sein.

Materialdiagramm	Materialname	Anzahl (Betrag)
	DHT11 Modul	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Die Verbindung ist geradeaus, unser DHT11 Sensor hat 3 Pins: GND, VCC (5V) und Data OUT. Mit dem Daten-Out-Pin können wir die Werte der Temperatur und der Luftfeuchtigkeit abrufen. Der Daten-Pin geht direkt an GPIO4 Pin, während der VCC zu 5V und GND an GND geht.



## Verbindung

DHT11	Raspberry Pi
GND	GND
DATEN AUS	GPIO4
Vcc	5v



## Codeübersicht

Innerhalb des DH11-Skripts verwenden wir die adafruit DHT-Bibliothek, die es uns ermöglicht, die Daten vom Sensor zu erhalten. Wir werden spezifisch, dass wir Sensor 11 verwenden, da die DHT-Familie mehrere Sensoren hat und der, den wir verwenden, ist der DHT-11.

In einer Codezeile (Zeile 33) werden wir sowohl Temperatur als auch Luftfeuchtigkeit erhalten und ausdrucken, wenn sie nicht leer sind. Wenn sie leer zurückgeben, kann etwas mit Ihren Verbindungen falsch sein, stellen Sie sicher, dass Sie den Sensor an die richtigen Pins anschliessen!

Komplizieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 dh11.py
```

```
23 import sys
24 import Adafruit_DHT
25
26 # set type of the sensor
27 sensor = 11
28 # set pin number
29 pin = 4
30
31 # Try to grab a sensor reading. Use the read_retry method which will retry up
32 # to 15 times to get a sensor reading (waiting 2 seconds between each retry).
33 humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
34
35 # Un-comment the line below to convert the temperature to Fahrenheit.
36 # temperature = temperature * 9/5.0 + 32
37
38 # Note that sometimes you won't get a reading and
39 # the results will be null (because Linux can't
40 # guarantee the timing of calls to read the sensor).
41 # If this happens try again!
42 if humidity is not None and temperature is not None:
43     print('Temp={0:0.1f}* Humidity={1:0.1f}%'.format(temperature, humidity))
44 else:
45     print('Failed to get reading. Try again!')
```

## Anwendungseffekt

Das Skript erhält die Temperatur und die Luftfeuchtigkeit aus dem DH11-Sensor und druckt sie auf die Konsole.



## Aufgabe 25 – LCD1602 MIT I2C



### Einleitung

Dies ist ein Experiment zur Verwendung von LCD1602 mit I2C, die nächste Aufgabe wird ein Temperatur- und Feuchtigkeitsüberwachung realisiert.

In dieser Aufgabe zeigen wir einfach "Hallo Welt"-Text auf unserem LCD-Bildschirm, um ein gutes Beispiel zu zeigen, wie wir die LCD-Anzeige verwenden können, um bestimmten Text anzuzeigen.



## Benötigte Hardware

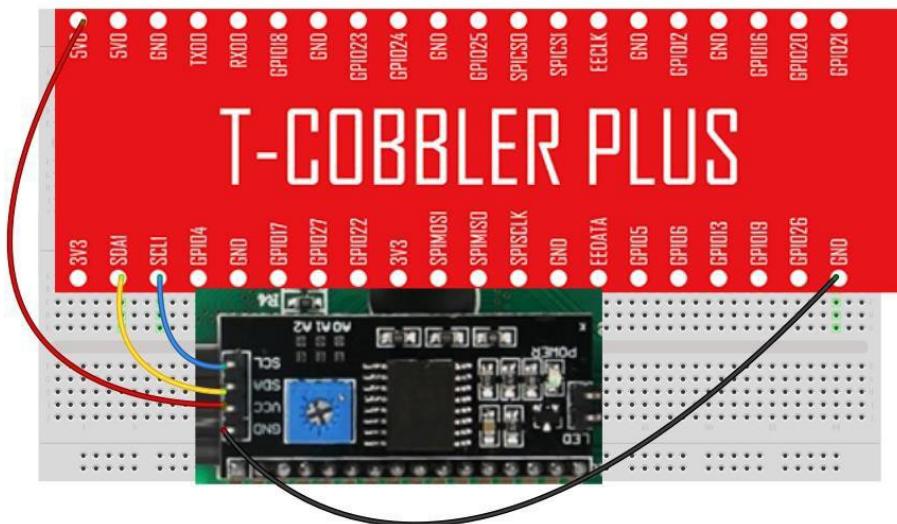
Die Hauptkomponente, die wir verwenden werden, ist die LCD1602, wir werden hier in keinem Widerstand notiert sein.

Materialdiagramm	Materialname	Anzahl (Betrag)
	LCD1602 mit IIC	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm

Die Verbindungen sind ganz einfach, stellen Sie sicher, dass Sie den SDA- und SCL-Pin an den richtigen Stellen setzen oder Sie werden keine Daten auf dem LCD-Display sehen können. Wir brauchen hier keine Widerstände, so dass Sie es problemlos direkt mit dem Raspberry Pi verbinden können.



## Verbindung

LCD1602	Raspberry Pi
GND	GND
Vcc	5v
Sda	SDA1
Scl	SCL1



## Codeübersicht

Der Code ist sehr einfach. Wir werden die LCD-Anzeige mit der Adafruit char lcd Bibliothek initialisieren, wir werden die Hintergrundbeleuchtung drehen, so dass wir sehen können, was wir auf dem Bildschirm drucken, wir werden das Skript nur für den Fall, dass etwas Text von vor links und dann zeigen wir eine Nachricht von "Hallo Welt" wir warten 5 Sekunden und wiederholen Sie alles wieder, sie könnten den Bildschirm flackern sehen, das ist völlig normal.

Komplizieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 lcd.py
```

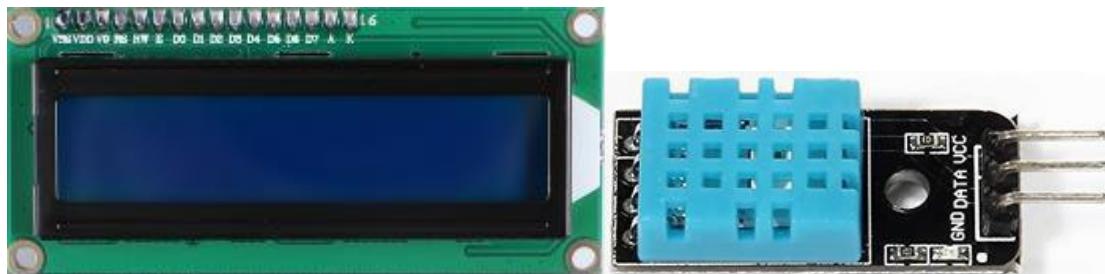
```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  import sys
4  import time
5  import Adafruit_CharLCD as LCD
6
7  # Define LCD column and row size for 16x2 LCD.
8  lcd_columns = 16
9  lcd_rows    = 2
10
11 # Initialize the LCD using the pins
12 lcd = LCD.Adafruit_CharLCDBackpack(address=0x21)
13
14 try:
15     while True:
16         # Turn backlight on
17         lcd.set_backlight(0)
18         # clean the LCD screen
19         lcd.clear()
20         lcd.message('Hello world')
21         time.sleep(5)
22     except KeyboardInterrupt:
23         # Turn the screen off
24         lcd.clear()
25         lcd.set_backlight(1)
```

## Anwendungseffekt

Die LCD-Anzeige zeigt "Hallo Welt" Nachricht, können Sie das Skript durch Drücken von STRG + C oder STRG + Z unterbrechen



## Ausgabe 26 – Temperatur und Feuchtigkeit Monitoring



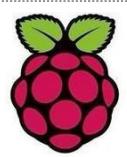
### Einleitung

Dies ist ein komplexerer Aufbau, es kann die Innentemperatur und Luftfeuchtigkeit gemessen und auf dem LCD angezeigt werden. Da wir in unserer vorherigen Aufgabe "Hallo Welt" gezeigt haben, werden wir diese Aufgabe noch einen Schritt weiterführen und Echtzeitdaten vom DHT11-Sensor anzeigen, nach Abschluss dieser Aufgabe werden Sie in der Lage sein zu verstehen, wie Sie jede Art von Nachricht auf dem LCD-Monitor anzeigen und so viele andere Projekte im Zusammenhang mit demselben Modul durchführen können.



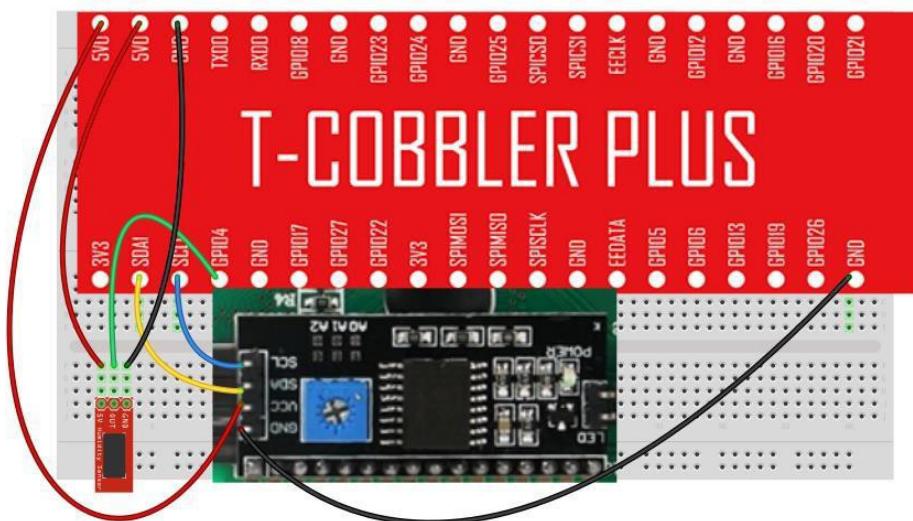
## Benötigte Hardware

Zu diesem Beispiel haben wir auch das DHT11-Modul hinzugefügt, wir benötigen sowohl das DHT11 als auch das LCD-Modul, um diese Aufgabe zu erfüllen, stellen Sie sicher, dass Sie beide vorbereiten.

Materialdiagramm	Materialname	Anzahl (Betrag)
	DHT11 Modul	1
	LCD1602 mit IIC	1
	Raspberry Pi Board	1
	T-Cubbler Plus	1
	40P GPIO Kabel	1
	Steckbrett	1
	Jumper-Drähte	Mehrere



## Verbindungsdiagramm



## Verbindung

DHT11	Raspberry Pi
GND	GND
DATEN AUS	GPIO4
Vcc	5v

LCD1602	Raspberry Pi
GND	GND
Vcc	5v
Sda	SDA1
Scl	SCL1



## Codeübersicht

Unser DHT-LCD-Beispiel ist dem vorherigen LCD-Beispiel sehr ähnlich, ausser dieses Mal zeigen wir nicht nur die "Hallo Welt"-Meldung an, wir erhalten die aktuellen Temperatur- und Feuchtigkeitswerte von unserem DH11-Sensor. Wenn Sie sie in Fahrenheit anzeigen möchten, müssen Sie die Zeile in Zeile 28 entkommentieren, wenn Sie sie in F anzeigen möchten. Kompilieren Sie das Skript, indem Sie in das Beispielverzeichnis gehen und folgenden Befehl ausführen:

```
python3 lcd_dht.py
```

```
20  try:
21      while True:
22          # Turn backlight on
23          lcd.set_backlight(0)
24          # get temperature and humidity
25          humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
26          # clean the LCD screen
27          lcd.clear()
28          # Un-comment the line below to convert the temperature to Fahrenheit.
29          # temperature = temperature * 9/5.0 + 32
30          if humidity is not None and temperature is not None:
31              lcd.message('Temp={0:0.1f}*  Humidity={1:0.1f}%'.format(temperature, humidity))
32          else:
33              print('Failed to get reading, Retrying in 5 seconds!')
34          # wait 5 seconds for the next try
35          time.sleep(5)
36      except KeyboardInterrupt:
37          # Turn the screen off
38          lcd.clear()
39          lcd.set_backlight(1)
```

## Anwendungseffekt

Laufend schaltet das Programm die LCD-Anzeige ein und zeigt die aktuelle Raumtemperatur und -feuchtigkeit des DHT11-Sensors an.