

Vorbemerkung

Über Unsere Firma

WayinTop, Your Top Way to Inspiration, ist ein professioneller Hersteller von über 2.000 Open Source-Motherboards, -Modulen und -Komponenten. WayinTop hat sich zum Ziel gesetzt, die wunderbare Welt der eingebetteten Elektronik zu erforschen und zu entmystifizieren, einschließlich, aber nicht beschränkt auf Arduino und Raspberry Pi. Wir sind bestrebt, die am besten gestalteten Produkte für Hersteller aller Altersgruppen und Könnensstufen herzustellen. Unabhängig von Ihrer Vision oder Ihrem Kenntnisstand sind unsere Produkte und Ressourcen darauf ausgelegt, die Elektronik besser zugänglich zu machen. WayinTop wurde 2013 gegründet und ist mittlerweile auf über 100 Mitarbeiter und eine über 50.000 Quadratmeter große Fabrik in China angewachsen. Mit unseren unermüdlichen Bemühungen haben wir auch das Angebot um Werkzeuge, Ausrüstungen, Verbindungssätze und verschiedene DIY-Produkte erweitert, die wir sorgfältig ausgewählt und getestet haben.

US Amazon Store Homepage:

<https://www.amazon.com/shops/A22PZZC3JNHS9L>

CA Amazon Store Homepage:

<https://www.amazon.ca/shops/A22PZZC3JNHS9L>

UK Amazon Store Homepage:

<https://www.amazon.co.uk/shops/A3F8F97TMOROPI>

DE Amazon Store Homepage:

<https://www.amazon.de/shops/A3F8F97TMOROPI>

FR Amazon Store Homepage:

<https://www.amazon.fr/shops/A3F8F97TMOROPI>

IT Amazon Store Homepage:

<https://www.amazon.it/shops/A3F8F97TMOROPI>

ES Amazon Store Homepage:

<https://www.amazon.es/shops/A3F8F97TMOROPI>

JP Amazon Store Homepage:

<https://www.amazon.co.jp/shops/A1F5OUAXY2TP0K>

Inhalt

Lektion 0 Installieren und bedienen Sie das Raspberry Pi System.....	4
Lektion 1 LED.....	14
Lektion 2 Button &LED.....	27
Lektion 3 Fließendes Wasserlicht.....	35
Lektion 4 Analog & PWM.....	42
Lektion 5 RGBLED.....	50
Lektion 6 Aktiver Summer.....	61
Lektion 7 Passiver Summer.....	68
Lektion 8 AD/DA Wandler.....	77
Lektion 9 Potentiometer & LED.....	86
Lektion 10 Potentiometer & RGBLED.....	94
Lektion 11 Fotowiderstand & LED.....	102
Lektion 12 Thermistor.....	110
Lektion 13 Joystick.....	118
Lektion 14 Relais & Motor.....	125
Lektion 15 Servo.....	133
Lektion 16 Schrittmotor.....	143
Lektion 17 74HC595 & LEDBar Graph.....	155
Lektion 18 74HC595 & 7-Segment-Anzeige.....	167
Lesson 19 74HC595&4-Digit 7-Segment Display.....	176
Lektion 20 74HC595 & LED Matrix.....	190
Lektion 21 LCD1602.....	204
Lektion 22 DHT11.....	218
Lesson 23 Matrix Keypad.....	229
Lektion 24 Ultraschall.....	237
Lektion 25 Infrarot Sensor.....	246

Lektion26 MPU6050.....	254
Lektion 27 Sound Sensor.....	263
Lektion 28 RFID RC522.....	270
Lektion 29 4N35.....	295
Lektion 30 NE555.....	304
Lektion 31 Infrarot Fernbedienung.....	313
Lektion 32 Summer Schweißen.....	327
Lektion 33 Fließendes Wasserlicht Schweißen.....	331

Lektion 0 Installieren und bedienen Sie das Raspberry Pi System

Überblick

In dieser Lektion erfahren Sie, wie Sie das Raspberry Pi-System installieren und bedienen.

Erforderliche Teile:

1 x Raspberry Pi

1 x LCD Anzeige

1 x HDMI Kabel

1 x SD Karte

1 x SD Karte Reader

1 x Maus und Tastatur

Sie können die spezifischen Installationsschritte auf den folgenden Websites abrufen:

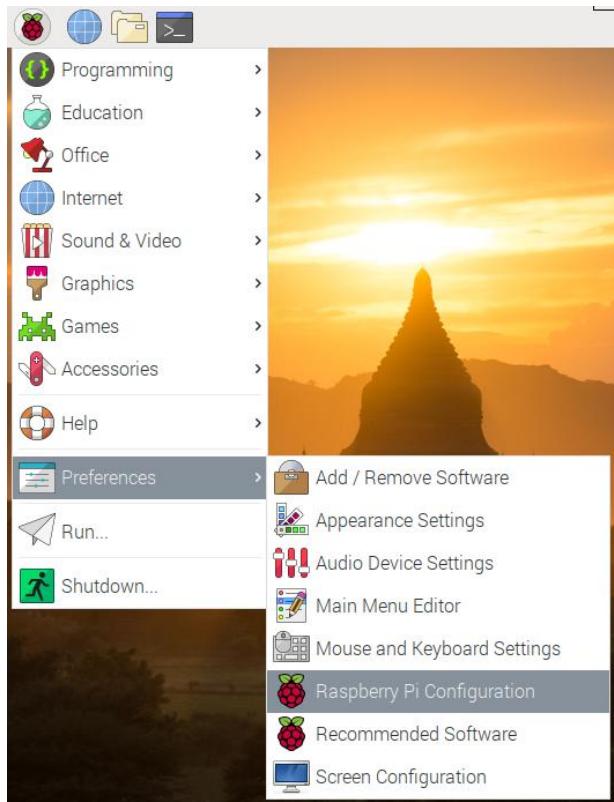
<https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up>

Nach der Installation des Systems erhalten Sie die grundlegenden Anweisungen auf den folgenden Websites:

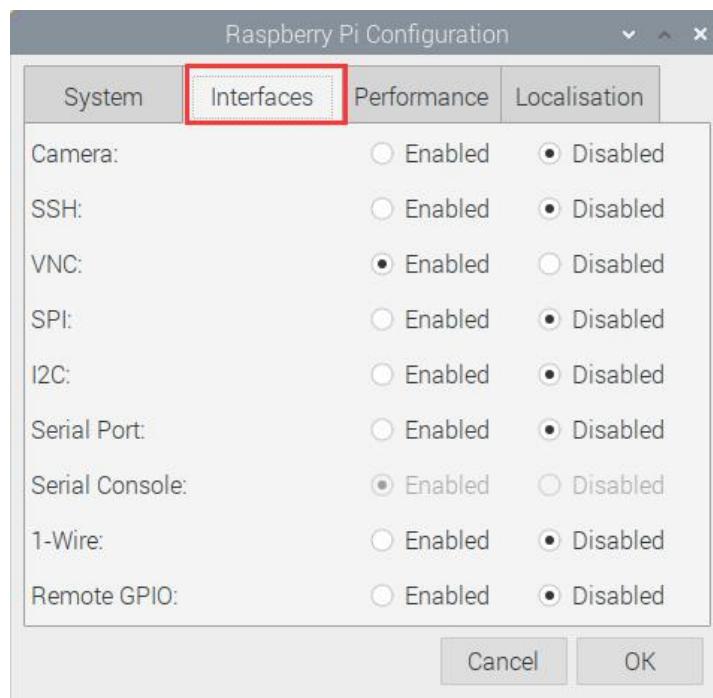
<https://projects.raspberrypi.org/en/projects/raspberry-pi-using>

Nachdem wir die oben genannten grundlegenden Vorgänge beherrschen, lernen wir, den Raspberry Pi mithilfe eines Computers zu steuern, um eine Verbindung zu Wi-Fi herzustellen:

Öffnen Sie die Raspberry Pi Oberfläche und klicken Sie auf ‘Preferences’->‘Raspberry Pi Configuration’ wie nachfolgend dargestellt.

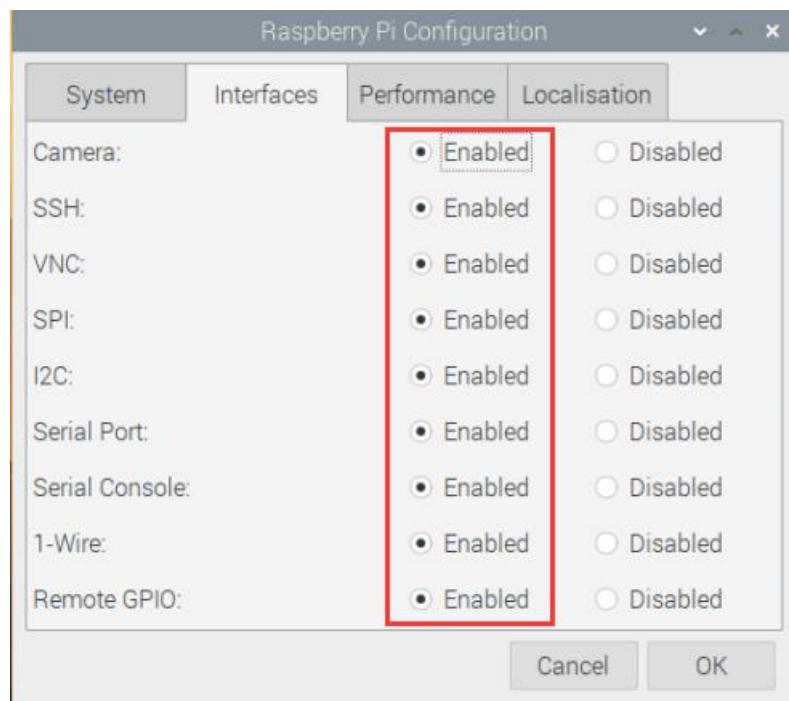


Wählen Sie im Popup-Fenster die Option ‘Interfaces’ aus, wie unten gezeigt:





Aktivieren Sie alle Schnittstellen wie unten gezeigt:



Klicken Sie auf ‘OK’ und starten Sie den Raspberry Pi neu.

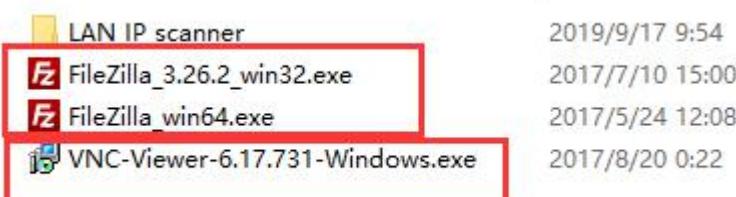
Die Raspberry Pi Konfiguration ist abgeschlossen und verwendet dann den Computer, um eine Verbindung zum Raspberry PiviaWi-Fi herzustellen:

Schritte

- 1) Öffnen Sie die Datei ‘Supporting software’ in der Begleitdatei wie folgt:



- 2) Installieren Sie die Software ‘FileZilla’ und ‘VNC’ software in der Datei:



3) After installing the two software, open the [Advanced IP Scanner](#) file in the [LAN IP scanner](#) file as shown below:Nach der Installation der beiden Software, öffnen Sie die Datei [Advanced IP Scanner](#) in der Datei [LAN IP scanner](#) wie folgt:



4) Wählen Sie die Software '[advanced_ip_scanner.exe](#)' wie unten gezeigt:

 platforms	2019/9/17 9:54
 printsupport	2019/9/17 9:54
 advanced_ip_scanner.exe	2014/11/28 7:06
 advanced_ip_scanner_console.exe	2014/11/28 7:06
 advanced_ip_scanner_MAC.bin	2019/9/16 10:58
 advanced_ip_scanner_zh_cn.qm	2014/11/28 7:06
 details_panel_zh_cn.tpl	2014/11/28 7:06
 libeay32.dll	2014/11/28 7:06
 mac_interval_tree.txt	2014/11/28 7:06
 msvcp120.dll	2014/11/28 7:06
 msvcr120.dll	2014/11/28 7:06
 pcre.dll	2014/11/28 7:06
 Qt5Core.dll	2014/11/28 7:06
 Qt5Gui.dll	2014/11/28 7:06

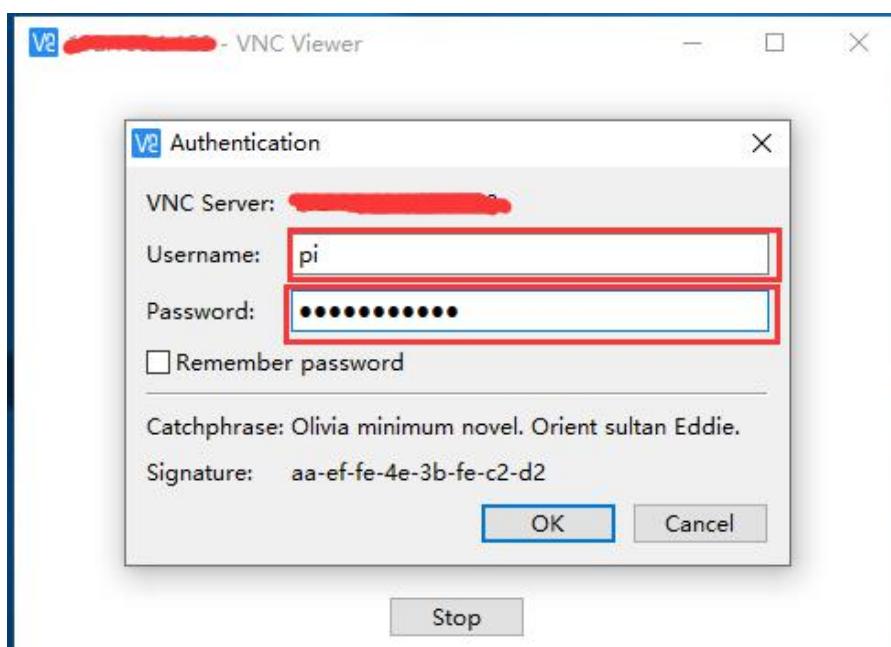
5) Klicken Sie auf '[Scan](#)', um die Raspberry Pi IP zu scannen (In der oberen rechten Ecke des Desktops im Raspberry Pi System können Sie die IP Adresse auch anzeigen, indem Sie die Maus auf das WLAN Symbol setzen), wie unten gezeigt:



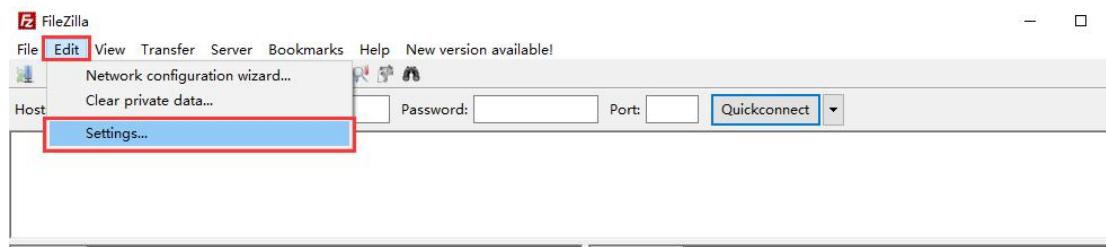
6.) Notieren Sie die IP-Adresse nach dem Scannen wie folgt:

DESKTOP...			50:7B:9D:A6:00:1F
001-PC			F4:4D:30:1C:3C:41
SKY-201...			1C:1B:0D:A5:DE:C8
192.168...			68:3E:34:E0:05:19
192.168...			DC:A6:32:0A:38:F5
192.168...			7C:03:AB:3F:82:13
192.168...	Raspberry Pi Fou...	B8:27:EB:CA:AF:DC	
192.168...			88:40:3B:A2:5F:CF
192.168...			8C:25:05:94:85:DB
192.168...			14:D0:0D:92:C3:D7
192.168...			8C:16:45:44:63:79
PS2019E...			00:CF:E0:53:8F:A1
192.168...			60:EE:5C:76:C2:AB
PS2019F...			00:CF:E0:53:90:AD
192.168...			70:EF:00:29:54:70
192.168...			5C:C3:07:BF:87:AB
192.168...			DC:A6:32:15:5E:F4

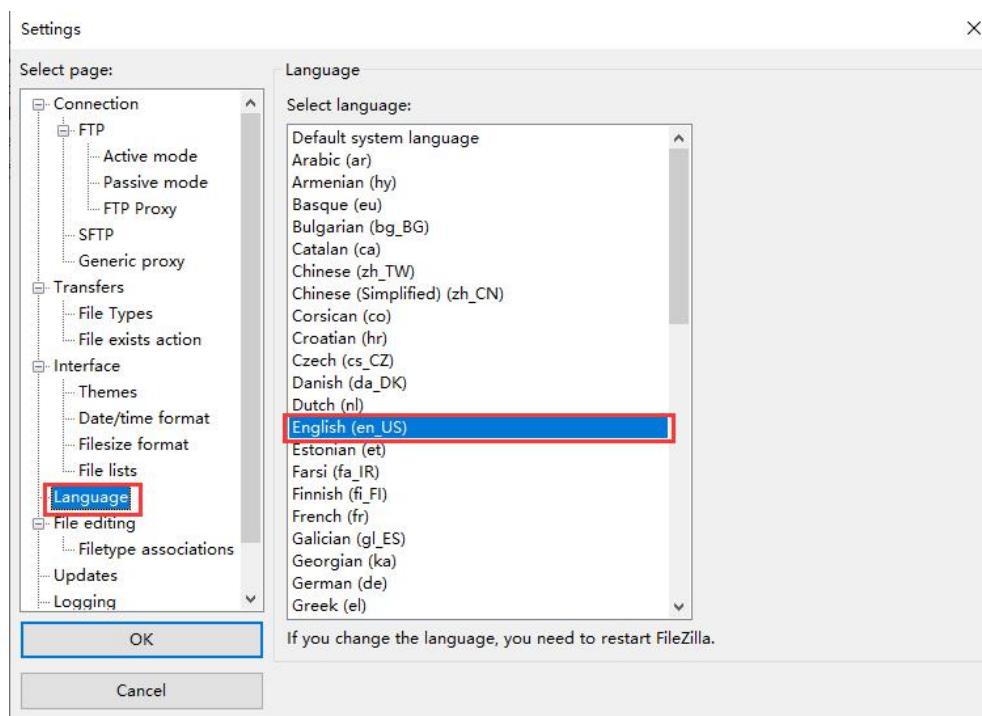
Öffnen Sie die VNC-Software, nachdem Sie die IP-Adresse aufgezeichnet haben. Geben Sie die IP-Adresse in das Eingabefeld ein, drücken Sie die Enter Button, geben Sie den Benutzernamen und das Kennwort in das Popup-Fenster ein und klicken Sie auf "OK", um eine Verbindung zum Raspberry Pi herzustellen. Wie nachfolgend dargestellt:



8) Verwenden Sie die '[FileZilla](#)' Software, um eine drahtlose Übertragung von Dateien zu erreichen. Öffnen Sie die '[FileZilla](#)' Software und stellen Sie die Sprache ein. Klicken Sie wie unten gezeigt auf die zweite Schaltfläche im Menü '[Edit](#)' und wählen Sie dann die Schaltfläche '[Settings](#)' im Dropdown-Feld.



9) In diesem Tutorial wird Englisch als Beispiel verwendet. Klicken Sie auf '[Language](#)', wählen Sie '[English](#)' und dann auf '[OK](#)'. Schließen Sie nach dem Einstellen die Software und öffnen Sie sie erneut, um sie erfolgreich einzustellen.

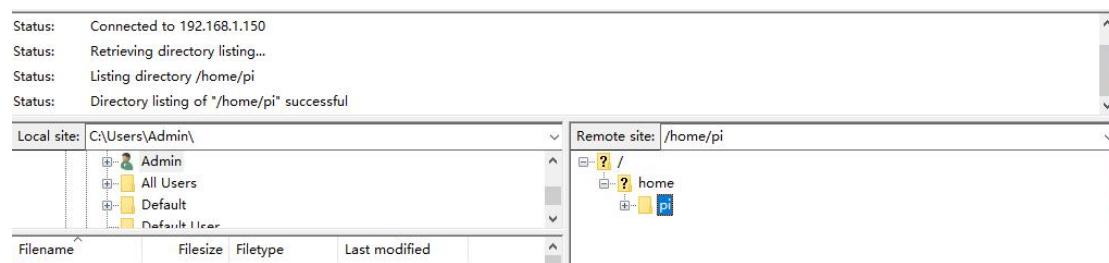


10) Nachdem die Einstellung für die Anzeigesprache abgeschlossen ist, stellen Sie eine Verbindung zum Raspberry Pi her, geben Sie die zuvor aufgezeichnete IP-Adresse in das erste Eingabefeld ein, geben Sie den Benutzernamen in das zweite Eingabefeld ein, geben Sie das Kennwort in das dritte Eingabefeld ein und geben Sie

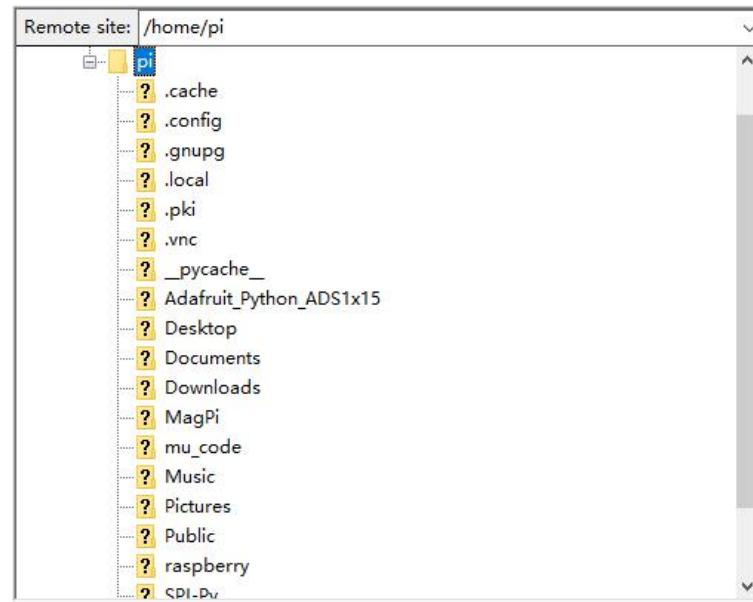
den Port ein Nummernstandard 22 im vierten Eingabefeld, und klicken Sie dann auf die Schaltfläche [‘Quickconnect’](#).



Die Verbindung ist wie in der folgenden Abbildung dargestellt:



Im zukünftigen Tutorial befinden sich alle Dateien im pi-Ordner. Nach erfolgreicher Verbindung können Sie die Dateien auf dem Computer direkt in die pi-Datei kopieren. (Diese Datei ist die pi-Datei im Raspberry Pi. Sie müssen die von uns bereitgestellte Codedatei in diesen Ordner kopieren, bevor Sie dem Tutorial folgen.)



WiringPi GPIO Pins

Es gibt drei GPIO-Verdrahtungsstifte von Raspberry Pi: basierend auf der BCM-Chipnummer, basierend auf der physischen Seriennummer und basierend auf connectionPi. Die Entsprechung zwischen diesen drei GPIO-Nummern ist wie folgt:

wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin
-	-	3.3v	1 2	5v	-	-
8	R1:0/R2:2	SDA0	3 4	5v	-	-
9	R1:1/R2:3	SCL0	5 6	0V	-	-
7	4	GPIO7	7 8	TXD	14	15
-	-	0V	9 10	RXD	15	16
0	17	GPIO0	11 12	GPIO1	18	1
2	R1:21/R2:27	GPIO2	13 14	0V	-	-
3	22	GPIO3	15 16	GPIO4	23	4
-	-	3.3v	17 18	GPIO5	24	5
12	10	MOSI	19 20	0V	-	-
13	9	MISO	21 22	GPIO6	25	6
14	11	SCLK	23 24	CE0	8	10
-	-	0V	25 26	CE1	7	11
30	0	SDA.0	27 28	SCL.0	1	31
21	5	GPIO.21	29 30	0V	-	-
22	6	GPIO.22	31 32	GPIO.26	12	26
23	13	GPIO.23	33 34	0V	-	-
24	19	GPIO.24	35 36	GPIO.27	16	27
25	26	GPIO.25	37 38	GPIO.28	20	28
0V			39 40	GPIO.29	21	29
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin

For RPi B

For RPi B+ / 2 model B

Sie können auch den Befehl ["gpio readall"](#) verwenden, um deren Korrespondenz anzuzeigen. Wie nachfolgend dargestellt:

BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
					Pi	3				
		3.3V			1	2		5v		
2	8	SDA.1	ALTO	1	3	4		5V		
3	9	SCL.1	ALTO	1	5	6		0v		
4	7	GPIO. 7	IN	1	7	8	1	ALT5	TxD	15
		0v			9	10	1	ALT5	RxD	16
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1
27	2	GPIO. 2	IN	0	13	14		0v		18
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	4
		3.3V			17	18	0	IN	GPIO. 5	5
10	12	MOSI	ALTO	0	19	20		0v		24
9	13	MISO	ALTO	0	21	22	0	IN	GPIO. 6	6
11	14	SCLK	ALTO	0	23	24	1	OUT	CE0	10
		0v			25	26	1	OUT	CE1	11
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31
5	21	GPIO.21	IN	1	29	30		0v		1
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26
13	23	GPIO.23	IN	0	33	34		0v		12
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28
		0v			39	40	0	IN	GPIO.29	29
										21
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
					Pi	3				

Wenn Sie Raspberry Pi 4B verwenden, treten Fehler auf, wenn Sie den Befehl ["gpio readall"](#) ausführen. Wie folgt:

```
pi@raspberrypi:~ $ gpio readall
Oops - unable to determine board type... model: 17
```

Dies liegt daran, dass die offizielle Version der Bibliothek, die 4B unterstützt, noch nicht veröffentlicht wurde, was dazu führt, dass einige Befehle nicht ordnungsgemäß verwendet werden können. Das nächste Projekt ist davon jedoch nicht betroffen. Für dieses Problem können Sie es lösen, indem Sie einen Patch installieren. Führen Sie einfach die folgenden Befehle im Terminal aus:

["wget https://project-downloads.drogon.net/wiringpi-latest.deb"](#)

["sudo dpkg -i wiringpi-latest.deb"](#)

(Hinweis: Wenn Sie einen Raspberry Pi 4B verwenden, muss dieser Schritt ausgeführt werden, da er sonst die Signalausgabe des GPIO-Anschlusses beeinträchtigt.)

Führen Sie nach Abschluss der Installation die Befehle "gpio -v" und "gpio readall" erneut aus.

```
pi@raspberrypi:~ $ gpio -v
gpio version: 2.52
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Pi 4B, Revision: 01, Memory: 1024MB, Maker: Sony
  * Device tree is enabled.
  *--> Raspberry Pi 4 Model B Rev 1.1
  * This Raspberry Pi supports user-level GPIO access.
pi@raspberrypi:~ $ gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     | 3.3v |      |   | 1 || 2 |   |      | 5v  |   | |
| 2   | 8   | SDA.1 | ALT0 | 1 | 3 || 4 |   |      | 5v  |   |
| 3   | 9   | SCL.1 | ALT0 | 1 | 5 || 6 |   |      | 0v  |   |
| 4   | 7   | GPIO. 7 | IN   | 1 | 7 || 8 | 1 | IN   | TxD | 15 | 14 |
|     |     | 0v    |      |   | 9  || 10 | 1 | IN   | RxD | 16 | 15 |
| 17  | 0   | GPIO. 0 | IN   | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1 | 18 |
| 27  | 2   | GPIO. 2 | IN   | 0 | 13 || 14 |   |      | 0v  |   |
| 22  | 3   | GPIO. 3 | IN   | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4 | 23 |
|     |     | 3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5 | 24 |
| 10  | 12  | MOSI  | IN   | 0 | 19 || 20 |   |      | 0v  |   |
| 9   | 13  | MISO  | IN   | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6 | 25 |
| 11  | 14  | SCLK  | IN   | 0 | 23 || 24 | 1 | IN   | CE0  | 10 | 8  |
|     |     | 0v    |      |   | 25 || 26 | 1 | IN   | CE1  | 11 | 7  |
| 0   | 30  | SDA.0 | IN   | 1 | 27 || 28 | 1 | IN   | SCL.0 | 31 | 1  |
| 5   | 21  | GPIO.21 | IN  | 1 | 29 || 30 |   |      | 0v  |   |
| 6   | 22  | GPIO.22 | IN  | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26 | 12 |
| 13  | 23  | GPIO.23 | IN  | 0 | 33 || 34 |   |      | 0v  |   |
| 19  | 24  | GPIO.24 | IN  | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27 | 16 |
| 26  | 25  | GPIO.25 | IN  | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28 | 20 |
|     |     | 0v    |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29 | 21 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Weitere Informationen zu connectionPi finden Sie unter: <http://wiringpi.com/>

Lektion 1 LED

Überblick

In dieser Lektion erfahren Sie, wie Sie mit der Stromnetz und dem Widerstand des Raspberry Pi das LED Licht blinken lassen.

Erforderliche Teile:

1 x Raspberry Pi

1 x LED

1 x 220 Ohm Widerstand

2 x Jumper Kabel

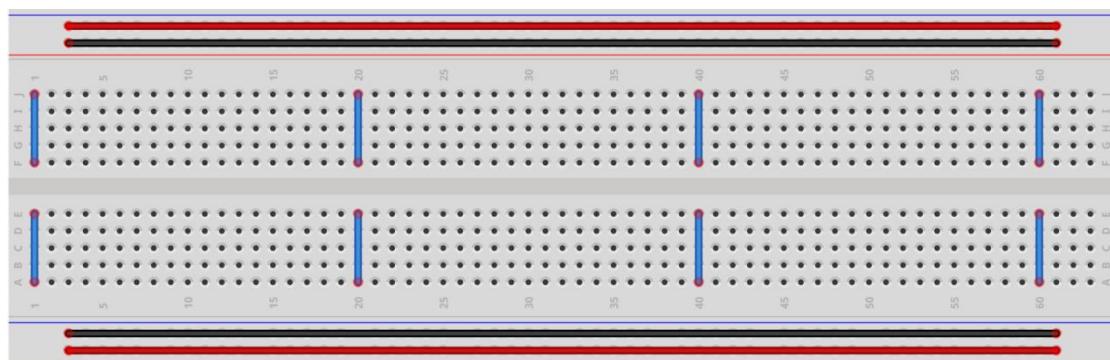
1 x Steckbrett

Produkt Einführung

Steckbrett

Mit einem Steckbrett können Sie Schaltkreise schnell prototypisieren, ohne zu löten.

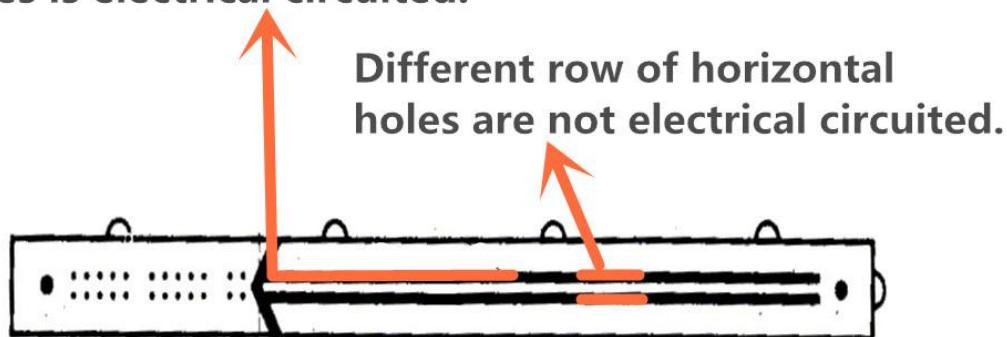
Das spezifische Prinzip lautet wie folgt:



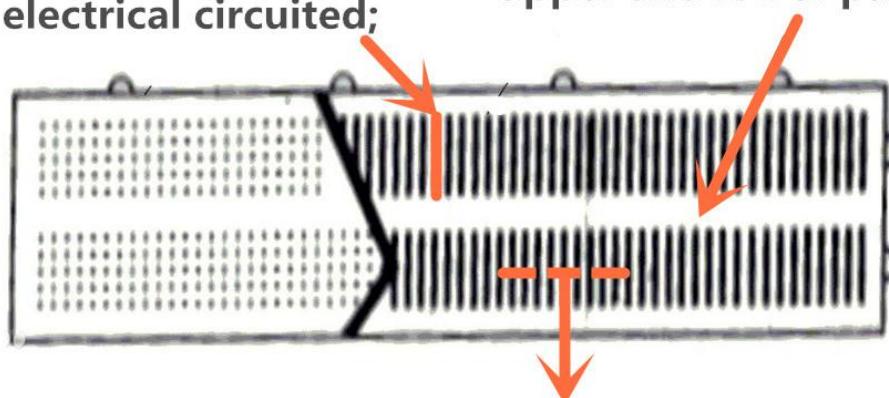


Die Innenseite des Steckbretts besteht aus Metallstreifen. Wenn es in die Löcher der Platine eingeführt wird, kann es mit dem Metallstreifen in Kontakt sein, um den Zweck der Leitung zu erreichen. Normalerweise sind 5 Löcher durch einen Metallstreifen verbunden. Auf beiden Seiten der Steckbrettplatte befinden sich zwei Reihen vertikaler Löcher, die ebenfalls 5 Löcher als Gruppe bilden. Diese beiden Reihen dienen zur Stromversorgung der Komponenten am Steckbrett.

The same row of horizontal holes is electrical circuited.



The Groove to isolate the upper and lower parts.



LED

Die LEDs können großartige Anzeigeleuchten erzeugen. In dieser Lektion verwenden wir die am häufigsten verwendete LED, eine 5 mm lange blaue LED (5 mm bezieht sich auf den Durchmesser der LED, andere gängige Größen sind 3 mm und 10 mm).

Sie können eine LED nicht direkt an eine Batterie oder eine Spannung anschließen, weil:

- 1) Die LED hat eine positive und eine negative Leitung und leuchtet nicht, wenn sie falsch platziert ist;
- 2) Die LED muss mit einem Widerstand versehen sein, um den durch sie fließenden Strom zu begrenzen oder zu drosseln. sonst brennt es aus!



The way to distinguish the positive and negative poles of the LED : Long Pin-positive; Short Pin-negative.

Widerstände

Wie der Name schon sagt, widerstehen Widerstände dem Stromfluss. Je höher der Wert des Widerstands ist, desto widerstandsfähiger ist er und desto weniger elektrischer Strom fließt durch ihn.

Wir werden dies verwenden, um zu steuern, wie viel Strom durch die LED fließt und wie hell sie daher leuchtet.



Die Widerstandseinheit heißt Ohm und wird normalerweise auf Ω abgekürzt. Da ein Ohm ein niedriger Widerstandswert ist (er widersteht nicht viel Strom), bezeichnen wir auch die Werte der Widerstände in $k\Omega$ (1.000 Ω) und $M\Omega$ (1.000.000 Ω). Diese werden Kiloohm und Megaohm genannt.

In dieser Lektion werden drei verschiedene Widerstandswerte verwendet: 220Ω , $1k\Omega$ und $10k\Omega$. Diese Widerstände sehen alle gleich aus, mit der Ausnahme, dass sie

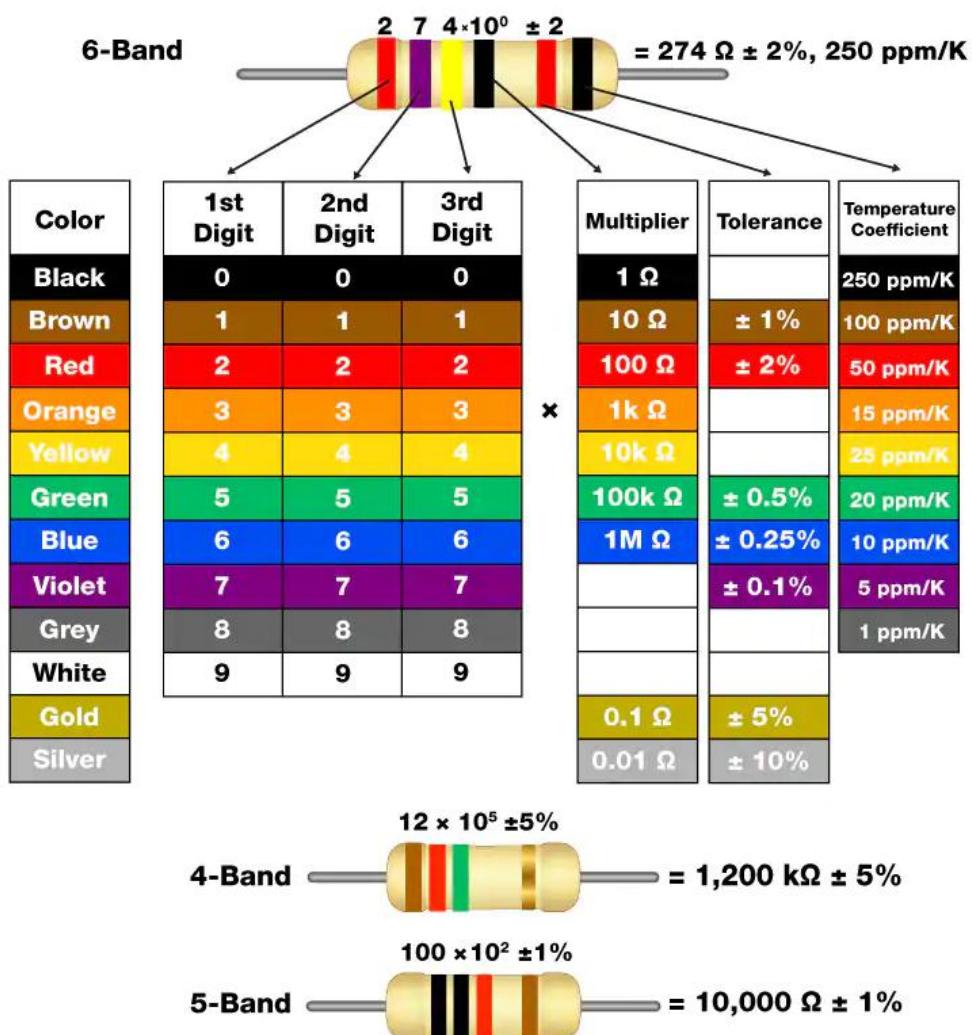


verschiedenfarbige Streifen aufweisen. Diese Streifen zeigen den Wert des Widerstands an.

Widerstände sehen alle gleich aus, außer dass sie verschiedenfarbige Streifen haben. Diese Streifen geben Auskunft über den Wert des Widerstandes.

Das folgende Bild zeigt Ihnen, wie Sie den Widerstandswert unterscheiden.

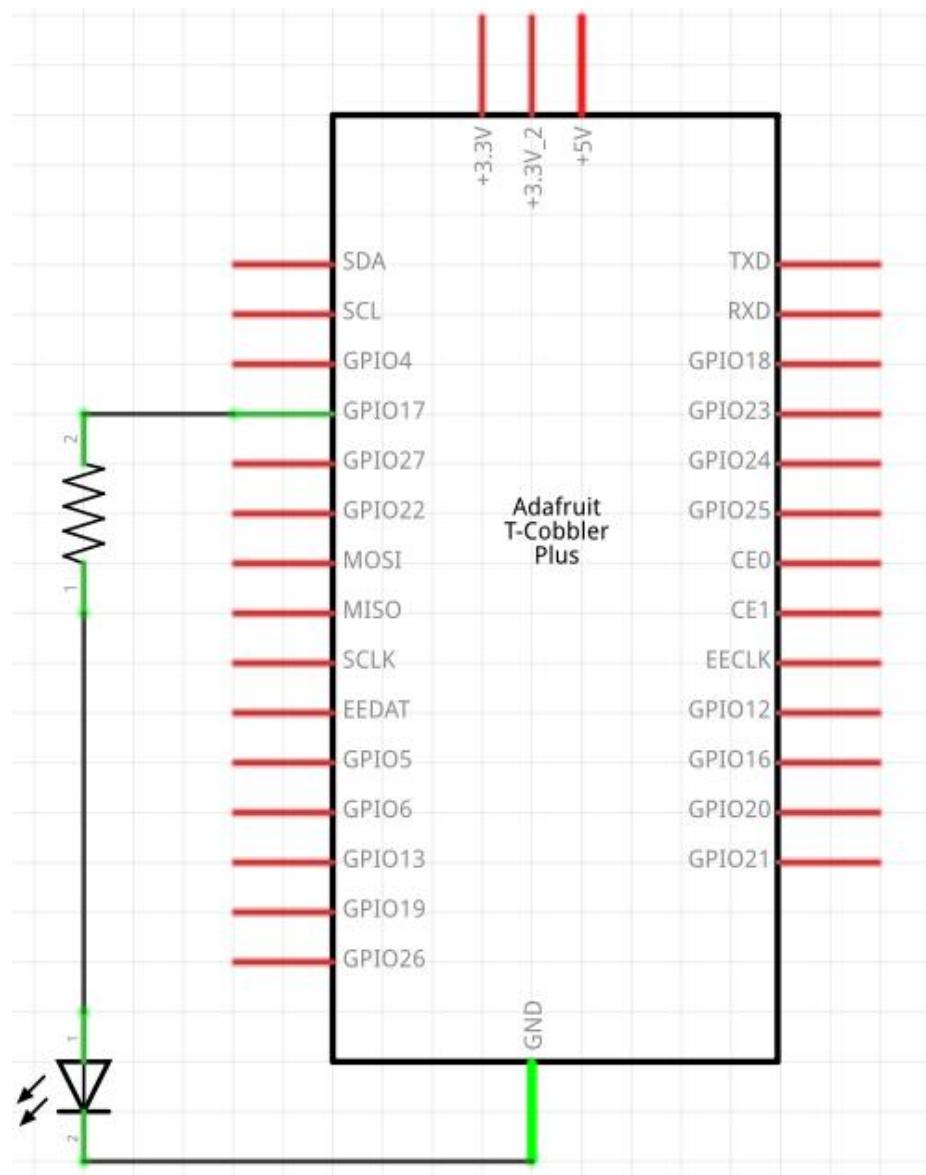
How to Read Resistor Color Codes



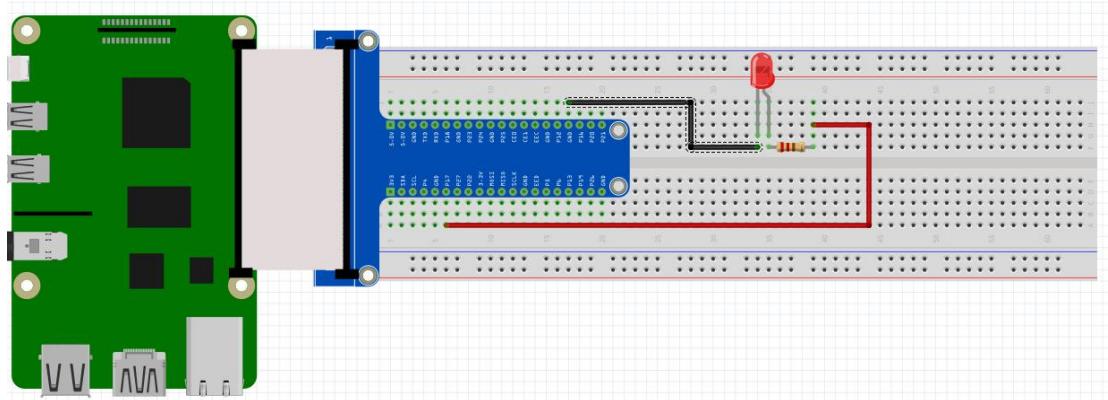
Wenn Sie feststellen, dass diese Unterscheidungsmethode zu kompliziert ist, können Sie auch einen Multimeter verwenden, um den Wert des Widerstands zu messen. Der Widerstand unterscheidet sich von LED, es hat keine positiven und negativen Pole, so

dass es normal funktionieren kann, indem es an den negativen oder positiven Pol der LED angeschlossen wird.

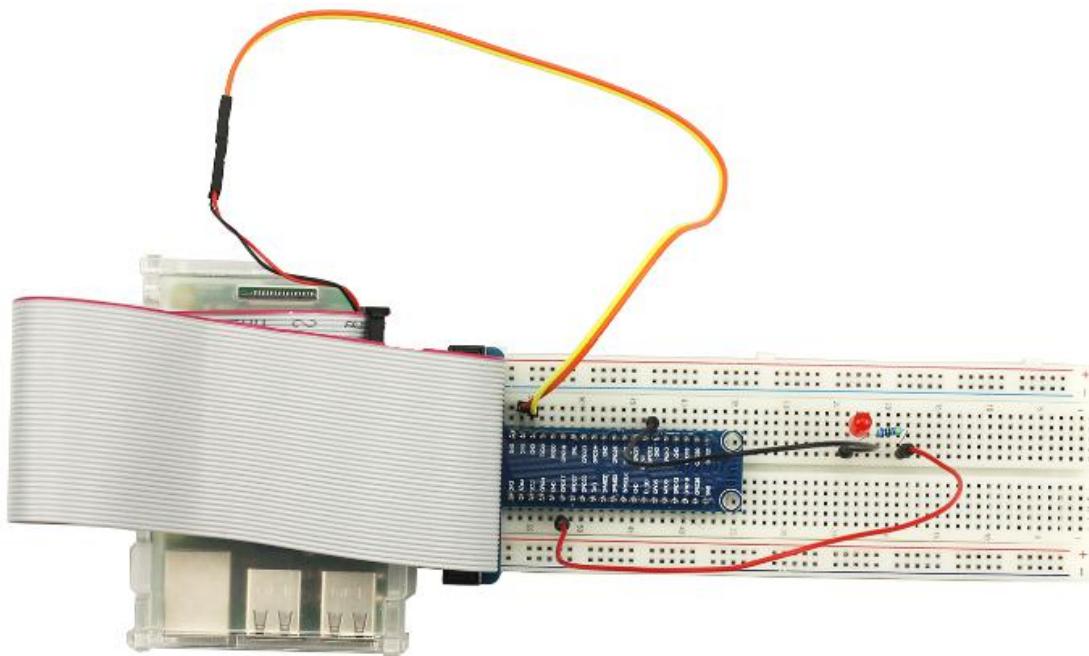
Verbindung Schema



Verdrahtung Diagramm



Beispiel Abbildung



C Code

Hinweis: Bevor Sie den Code ausführen, müssen Sie die von uns bereitgestellte Codedatei in den pi-Ordner verschieben oder eine ausführbare Datei in dem Verzeichnis erstellen, in dem der Befehl ausgeführt wird (z. B. code/C/1.LED/), und dann die schreiben Code in die Datei (Diese Erinnerung wird in den nächsten Lektionen nicht gegeben).

Öffnen Sie das Terminal und geben Sie den Befehl `cd code/C/1.LED/` ein, um das `LED.c` code Verzeichnis aufzurufen.

Geben Sie den Befehl `ls` ein, um die Datei LED.c im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/C/1.LED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/1.LED/
pi@raspberrypi:~/code/C/1.LED $ ls
LED.c
pi@raspberrypi:~/code/C/1.LED $
```

Geben Sie den Befehl `gcc LED.c -o LED -lwiringPi` ein, um die LED für die ausführbare Datei für LED.c zu generieren, und geben Sie den Befehl `ls` ein, um sie nachzusehen.

Geben Sie den Befehl `sudo ./LED` ein, um den Code auszuführen. Das Ergebnis lautet wie folgt:



```
pi@raspberrypi:~/code/C/1.LED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/1.LED/
pi@raspberrypi:~/code/C/1.LED $ ls
LED.c
pi@raspberrypi:~/code/C/1.LED $ gcc LED.c -o LED -lwiringPi
pi@raspberrypi:~/code/C/1.LED $ ls
LED LED.c
pi@raspberrypi:~/code/C/1.LED $ sudo ./LED
wiringPi initialize successfully, GPIO 0(wiringPi pin)
led on...
...led off
led on...
pi@raspberrypi:~/code/C/1.LED $
```

Drücken Sie "`Ctrl + c`", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>

#define ledPin 0

int main(void)
{
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
```

```
}

printf("wiringPi initialize successfully, GPIO %d(wiringPi pin)\n",ledPin);

pinMode(ledPin, OUTPUT);

while(1){
    digitalWrite(ledPin, HIGH);
    printf("led on...\n");
    delay(1000);
    digitalWrite(ledPin, LOW);
    printf("...led off\n");
    delay(1000);

}

return 0;
}
```

Code interpretation

```
#define ledPin 0
```

Dieser Code ist eine GPIO-Makrodefinition (Pin-Benennung). Der mit dem LEDPin in der Schaltung verbundene GPIO ist GPIO17. GPIO17 ist in der connectionPi-Nummer als 0 definiert. "LedPin" sollte also als 0-Pin definiert werden.

```
If(wiringPiSetup() == -1){
Printf("setup wiringPi failed !");
Return 1;
}
```

```
Printf("wiringPi initialize successfully, GPIO %d(wiringPi pin)\n", ledPin);
```

Initialisieren Sie in der Hauptfunktion main () zuerst wiringPi und drucken Sie dann die ersten Ergebnisse aus. Dieser Vorgang wird hauptsächlich anhand des "if" Satzes beurteilt. Wenn die Initialisierung fehlschlägt, beenden Sie das Programm.

```
pinMode(ledPin, OUTPUT);
```

```
While(1){
digitalWrite(ledPin, HIGH);
Printf("led on...\n");
```

```
Delay(1000);
digitalWrite(ledPin, LOW);
Printf("...led off\n");
Delay(1000);
}
```

Nachdem das `connectionPi` erfolgreich initialisiert wurde, wird "ledPin" auf den Ausgabemodus gesetzt. Geben Sie dann die while-Schleife ein, bei der es sich um eine Endlosschleife handelt. Das heißt, das Programm wird immer in dieser Schleife ausgeführt, es sei denn, es endet extern. Verwenden Sie in dieser Schleife "`digitalWrite(ledPin, HIGH)`", um die Ausgabe von "ledPin" hoch zu machen, dann leuchtet die LED. Nach einer Verzögerung von 1 Sekunde, verwenden Sie "`digitalWrite (ledPin, LOW)`", um die Ausgabe von "ledPin" niedrig zu halten. Schalten Sie dann die LED aus und verzögern Sie. Wiederholen Sie den Zyklus und die LED beginnt die ganze Zeit zu blinken.

Die Konfigurationsfunktion von GPIO lautet wie folgt:

`Void pinMode(int pin, int mode);`

Dies setzt den Modus des Pins auf INPUT, OUTPUT, PWM_OUTPUT oder GPIO_CLOCK. Beachten Sie, dass nur der Verdrahtungs Pi Pin 1 (BCM_GPIO 18) den PWM Ausgang unterstützt und nur der Verdrahtungs Pi Pin 7 (BCM_GPIO 4) den CLOCK Ausgangsmodus unterstützt. Diese Funktion ist im Sys Modus nicht verfügbar. Wenn Sie den Pin Modus ändern müssen, können Sie das Programm `gpio` im Skript verwenden, bevor Sie das Programm starten.

`Void digitalWrite (int pin, int value);`

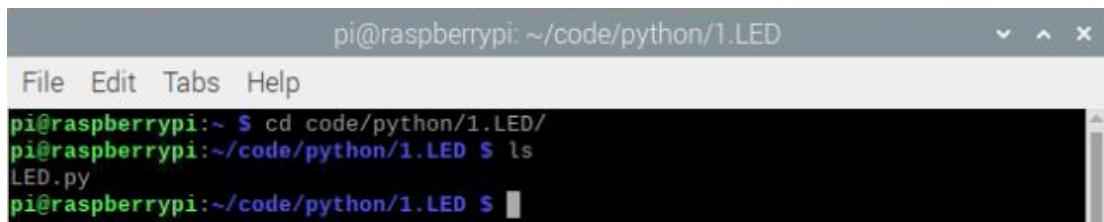
Schreiben Sie den Wert HIGH oder LOW (1 or 0) in den am Ausgang voreingestellten Pin.

Python code

Hinweis: (Da der Raspberry Pi viele Modelle und Systeme hat, unterscheiden sich einige Ausführungsbefehle. Einige der folgenden Ausführungsbefehle verwenden Python3, andere Python. Sie können ihn gemäß Ihrem eigenen Raspberry Pi testen. Manchmal sind beide Befehle in Ordnung.)

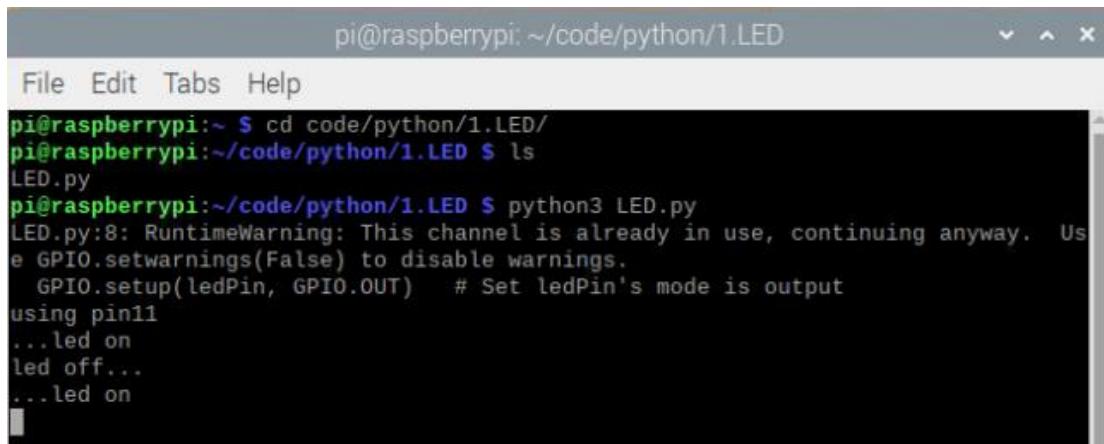
Öffnen Sie das Terminal und geben Sie den Befehl `cd code/python/1.LED/` ein, um das Codeverzeichnis `LED.py` aufzurufen.

Geben Sie den Befehl ls ein, um die Dateien im Verzeichnis LED.py anzuzeigen



```
pi@raspberrypi:~/code/python/1.LED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/1.LED/
pi@raspberrypi:~/code/python/1.LED $ ls
LED.py
pi@raspberrypi:~/code/python/1.LED $
```

Führen Sie den Code aus, indem Sie den Befehl `python3 LED.py` eingeben. Das Ergebnis lautet wie folgt:



```
pi@raspberrypi:~/code/python/1.LED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/1.LED/
pi@raspberrypi:~/code/python/1.LED $ ls
LED.py
pi@raspberrypi:~/code/python/1.LED $ python3 LED.py
LED.py:8: RuntimeWarning: This channel is already in use, continuing anyway.  Use GPIO.setwarnings(False) to disable warnings.
    GPIO.setup(ledPin, GPIO.OUT)    # Set ledPin's mode is output
using pin11
...led on
led off...
...led on
pi@raspberrypi:~/code/python/1.LED $
```

Drücken Sie "`Ctrl + c`", um das Programm zu beenden. Das Folgende ist der Programmcode:

Import RPi.GPIO as GPIO

Import time

ledPin = 11

Def setup():

GPIO.setmode(GPIO.BOARD)

GPIO.setup(ledPin, GPIO.OUT)

GPIO.output(ledPin, GPIO.LOW)

Print ('using pin%d' % ledPin)

Def loop():

While True:

 GPIO.output(ledPin, GPIO.HIGH)

 Print ('...led on')

 Time.sleep(1)

 GPIO.output(ledPin, GPIO.LOW)

 Print ('led off...')

 Time.sleep(1)

Def destroy():

 GPIO.output(ledPin, GPIO.LOW)

 GPIO.cleanup()

If __name__ == '__main__':

 Setup()

 Try:

 Loop()

 Except KeyboardInterrupt:

 Destroy() will be executed.

 Destroy()

Code interpretation

Import RPi.GPIO as GPIO

Import time

ledPin = 11

Importieren Sie das erforderliche RPi.GPIO Modul und das Zeitmodul

GPIO17 verwendet den Pin 11 des Mainboards, also definieren Sie ledPin als 11

Def setup():

GPIO.setmode(GPIO.BOARD)

GPIO.setup(ledPin, GPIO.OUT)

GPIO.output(ledPin, GPIO.LOW)

Print ('using pin%d'%ledPin)

In 'Setup ()' wird 'GPIO.setmode (GPIO.BOARD)' verwendet, um den Modus des GPIO entsprechend der physischen Position des Pins einzustellen. Stellen Sie "ledPin" auf den Ausgabemodus (output low) und drucken Sie dann den 'ledPin' serielle Ausgabe.

Def loop():

While True:

 GPIO.output(ledPin, GPIO.HIGH)

 Print ('...led on')

 Time.sleep(1)

 GPIO.output(ledPin, GPIO.LOW)

 Print ('led off...')

Time.sleep(1)

In deloop () gibt es eine Schleife, die eine Endlosschleife ist. Das heißt, das Programm wird immer in dieser Schleife ausgeführt, es sei denn, es wird durch einen externen Faktor beendet. Stellen Sie in dieser Schleife den LEDPin Ausgang auf einen hohen Pegel ein, und die LED leuchtet auf. Nach einer Verzögerung von 1 Sekunde stellen Sie den LEDPin Ausgang auf einen niedrigen Pegel ein, und die LED wird ausgeschaltet, worauf eine Verzögerung folgt. Wiederholen Sie den Zyklus und die LED beginnt die ganze Zeit zu blinken.

Def destroy():

```
GPIO.output(ledPin, GPIO.LOW)
```

```
GPIO.cleanup()
```

Wenn das Programm beendet ist, wird schließlich die Unterfunktion ausgeführt, die LED wird ausgeschaltet und dann wird der IO Port freigegeben. Wenn Sie das Programmterminal direkt schließen, wird auch das Programm beendet, aber die Funktion destroy () wird nicht ausgeführt. Daher werden GPIO Ressourcen nicht freigegeben. Bei der nächsten Verwendung von GPIO wird möglicherweise eine Warnmeldung angezeigt. Es ist daher keine gute Angewohnheit, das Programmterminal direkt zu schließen.

```
if __name__ == '__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy() will be executed.

        destroy()
```

Der Code beginnt mit "if __name__ == '__main__'::" und wird dann nacheinander ausgeführt. Wenn Sie das Programm Ctl + C drücken, wird das Programm "except KeyboardInterrupt:" beendet.

Lektion 2 Button & LED

Überblick

In dieser Lektion lernen Sie, wie Sie mit dem Raspberry Pi den LED Status über eine Button steuern.

Erforderliche Teile

1 x Raspberry Pi

1 x LED

1 x 220 Ohm Widerstand

4 x DuPont Wires

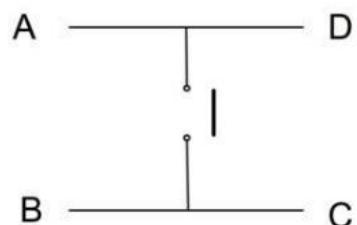
1 x Steckbrett

1 x Taste

Produkt Einführung

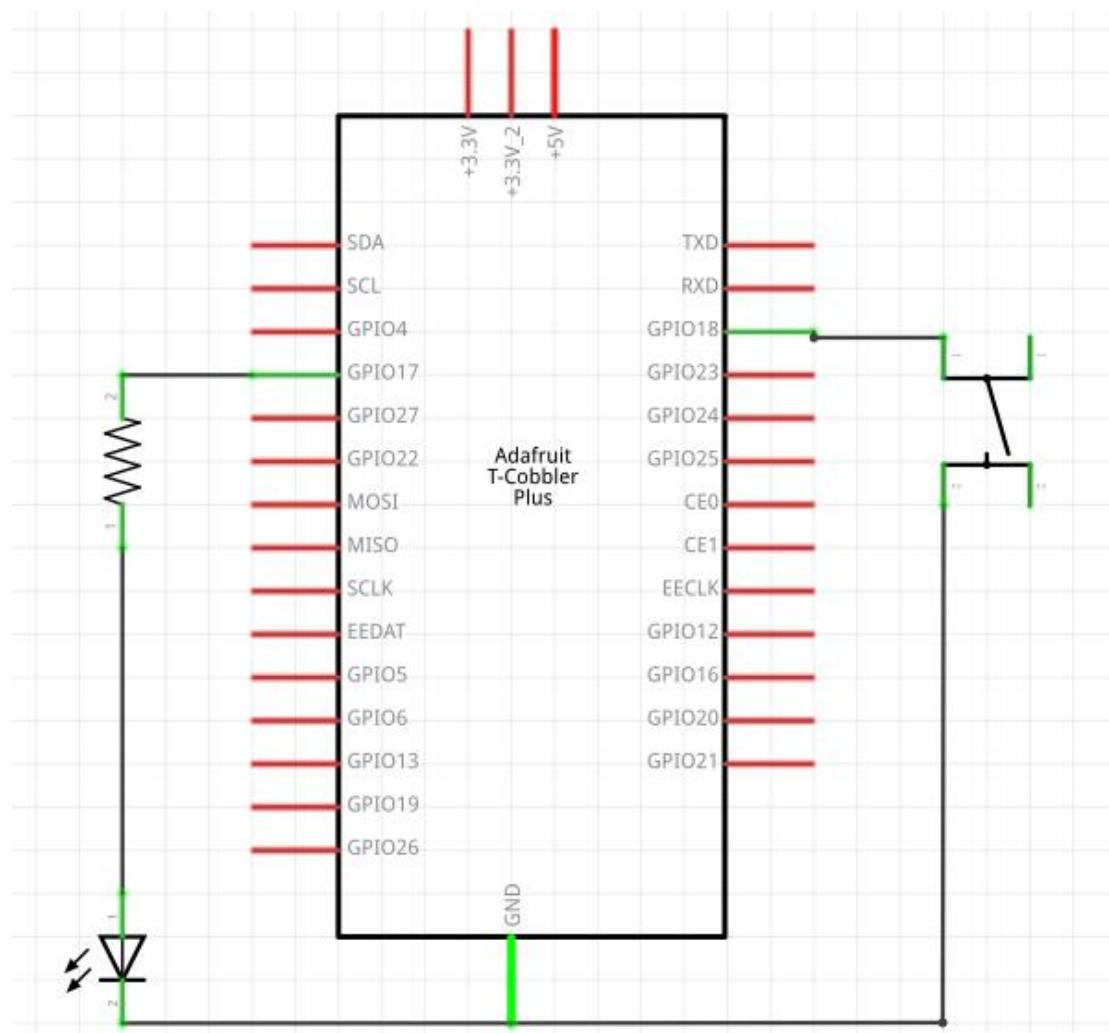
Druckknopf

Der Schalter ist eine sehr einfache Komponente. Wenn Sie die Button drücken, verbinden sie die beiden Kontakte miteinander, sodass Strom durch sie fließen kann. In dieser Lektion wird der Mini-Druckknopf verwendet.

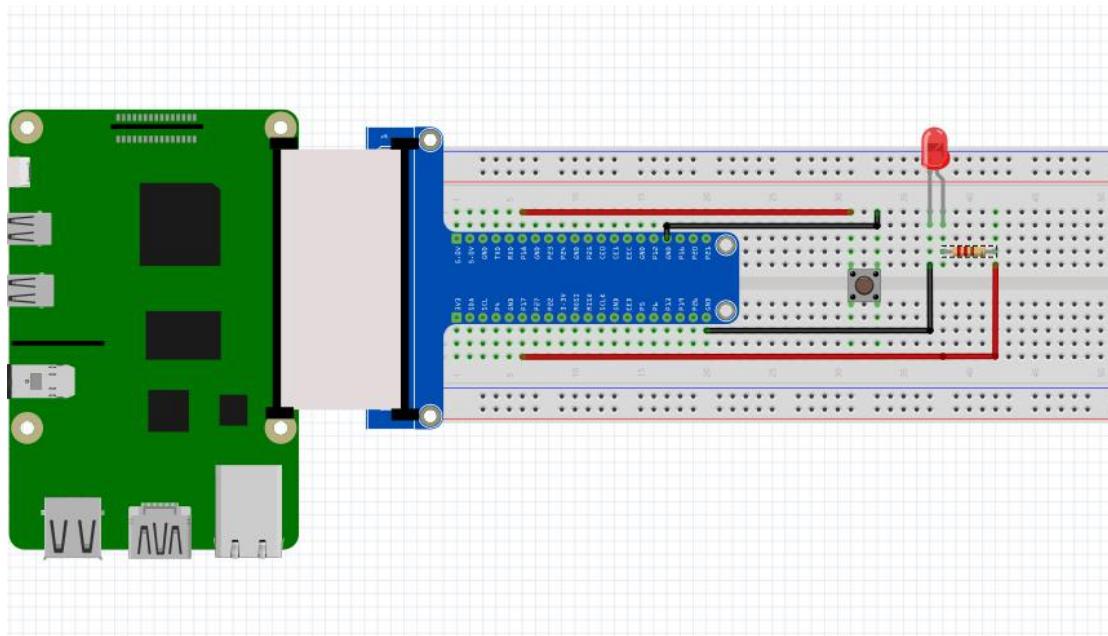


Tatsächlich werden nur zwei Stiftverbindungen verwendet. Innerhalb des Knopfes sind die Stifte B und C miteinander verbunden, ebenso wie A und D.

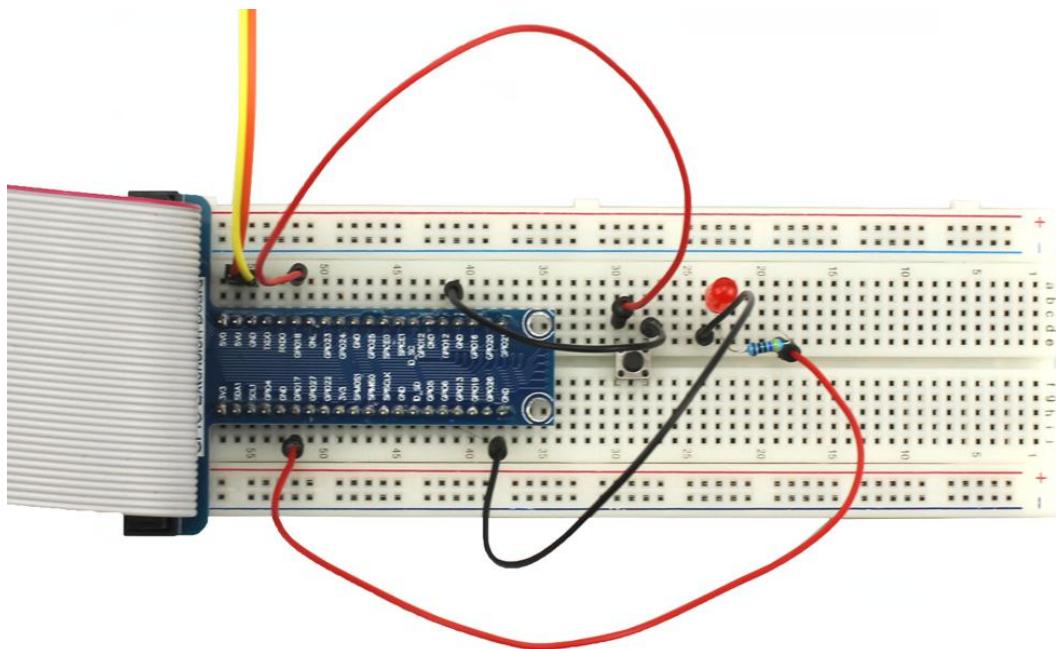
Verbindung Diagramm



Verdrahtung Diagramm



Beispiel Abbildung



C code

Öffnen Sie das Terminal und geben Sie den Befehl `cd code/C/2.Button`/ein, um das `Button.c` aufzurufen.

Geben Sie den Befehl `ls` ein, um die Datei `Button.c` im Verzeichnis anzuzeigen.

```
pi@raspberrypi: ~code/C/2.Button
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/2.Button/
pi@raspberrypi:~/code/C/2.Button $ ls
button.c
pi@raspberrypi:~/code/C/2.Button $
```

Geben Sie den Befehl `gcc Button.c -o Button -lwiringPi` ein, um Button für eine ausführbare Datei in Button.c zu generieren, und geben Sie den Befehl `ls` ein, um ihn anzuzeigen.

Führen Sie den Code aus, indem Sie den Befehl `sudo ./Button` eingeben. Das Ergebnis ist wie folgt:

```
pi@raspberrypi: ~/code/C/2.Button\nFile Edit Tabs Help\n...led off\n...led off\n...led off\n...led off\n...led off\n...led off\n...led off\n...led off\n...led off
```

Sie können "Ctrl+C" drücken, um das Programm zu beenden.

Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>

#define ledPin      0
#define buttonPin 1

int main(void)
{

```

```

if(wiringPiSetup() == -1){
    printf("setup wiringPi failed !");
    return 1;
}

pinMode(ledPin, OUTPUT);
pinMode(buttonPin, INPUT);

pullUpDnControl(buttonPin, PUD_UP);
while(1){

    if(digitalRead(buttonPin) == LOW){
        digitalWrite(ledPin, HIGH);
        printf("led on...\n");
    }
    else {
        digitalWrite(ledPin, LOW);
        printf("...led off\n");
    }
}

return 0;
}

```

Code interpretation

```

#define ledPin 0
#define ButtonPin 1

```

In the circuit connection, the 'LED' and the Button are connected to GPIO 17 and GPIO 18, which correspond to 0 and 1 of 'PI' respectively. Therefore, 'ledPin' and 'ButtonPin' are defined as 0 and 1 respectively.

```

If(digitalRead(ButtonPin) == LOW){
    digitalWrite(ledPin, HIGH);
    Printf("led on...\n");
}
Else {
    digitalWrite(ledPin, LOW);
    Printf("...led off\n");
}

```

About 'digitalRead()':

This function returns the read value on the specified pin. Depending on the logic level

of the pin, it will be assigned a value of "High" or "Low" (1 or 0).

Python code

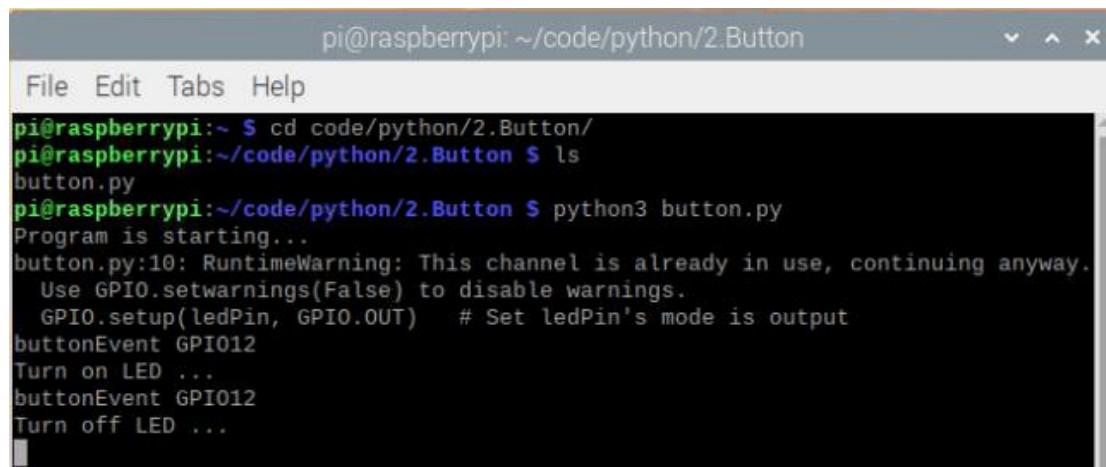
Open the terminal and enter the cd code/python/2.Button/ command to enter the code directory.

Enter the ls command to view the file Button.py in the directory.



```
pi@raspberrypi:~/code/python/2.Button
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/2.Button/
pi@raspberrypi:~/code/python/2.Button $ ls
button.py
pi@raspberrypi:~/code/python/2.Button $
```

Run the code by typing the python3 Button.py command, and the result is as follows:



```
pi@raspberrypi:~/code/python/2.Button
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/2.Button/
pi@raspberrypi:~/code/python/2.Button $ ls
button.py
pi@raspberrypi:~/code/python/2.Button $ python3 button.py
Program is starting...
button.py:10: RuntimeWarning: This channel is already in use, continuing anyway.
  Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(ledPin, GPIO.OUT)  # Set ledPin's mode is output
buttonEvent GPIO12
Turn on LED ...
buttonEvent GPIO12
Turn off LED ...

```

Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
ledPin = 11
buttonPin = 12

def setup():
    print ('Program is starting...')
    GPIO.setmode(GPIO.BOARD)
```

```
GPIO.setup(ledPin, GPIO.OUT)
GPIO.setup(buttonPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

def loop():
    while True:
        if GPIO.input(buttonPin)==GPIO.LOW:
            GPIO.output(ledPin,GPIO.HIGH)
            print ('led on ...')
        else :
            GPIO.output(ledPin,GPIO.LOW)
            print ('led off ...')

def destroy():
    GPIO.output(ledPin, GPIO.LOW)
    GPIO.cleanup()

if __name__ == '__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy() will be executed.
        destroy()
```

Code interpretation

```
import RPi.GPIO as GPIO
ledPin = 11
ButtonPin = 12
```

Importieren Sie das erforderliche RPi.GPIO-Modul

"GPIO17" verwendet Pin 11 des Mainboards, definieren Sie "ledPin" als 11

"GPIO18" verwendet Pin 12 des Mainboards, definieren Sie "buttonPin" als 12

```
def setup():
    print ('Program is starting...')
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(ledPin, GPIO.OUT)
    GPIO.setup(ButtonPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

In 'setup()' wird 'GPIO.setmode(GPIO.BOARD)' verwendet, um den Modus des GPIO entsprechend der physischen Position des Pins einzustellen. Stellen Sie 'ledPin' auf den Ausgabemodus und 'ButtonPin' auf den Eingabemodus mit Pull-up Widerstand.

```
def loop():
```

```
while True:  
    if GPIO.input(ButtonPin)==GPIO.LOW:  
        GPIO.output(ledPin,GPIO.HIGH)  
        print ('led on ...')  
    else :  
        GPIO.output(ledPin,GPIO.LOW)  
        print ('led off ...')
```

Fügen Sie die Schleifenanweisung 'while True' zur Anweisung 'def loop ()' hinzu, um weiterhin festzustellen, ob die Schaltfläche gedrückt wurde. "GPIO.input (ButtonPin)" wird niedrig zurückgegeben, wenn die Taste gedrückt wird. Dann ist das Ergebnis von "if" "true", LEDPin Ausgänge hoch (LED an), andernfalls wird der LEDPin-Ausgang niedrig (LED aus).

Lektion 3 Fließendes Wasserlicht

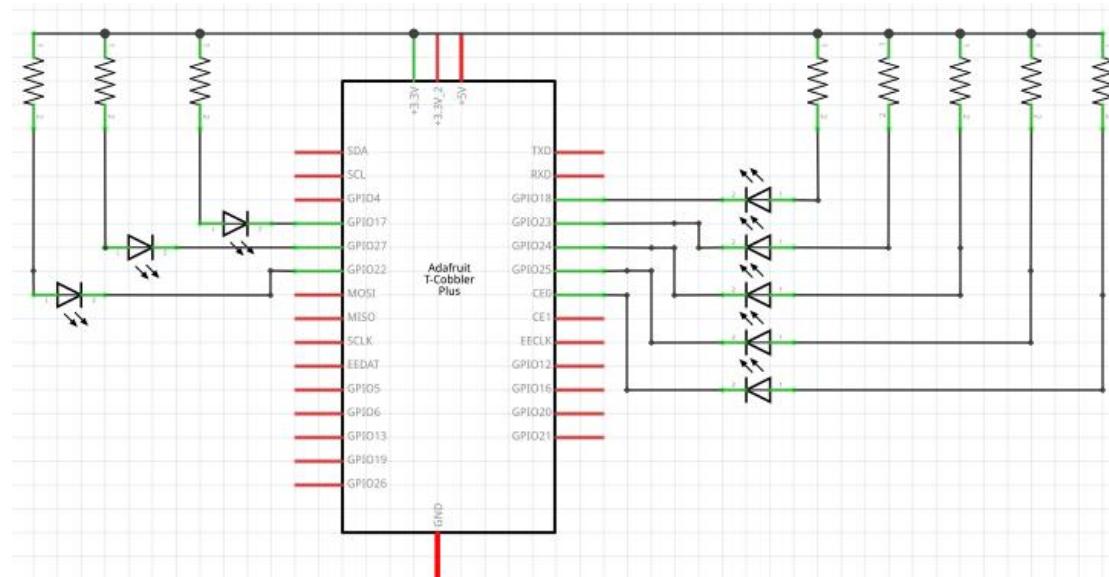
Überblick

In dieser Lektion lernen Sie, wie Sie mit einer Reihe von LEDs fließendes Wasser zum Leuchten bringen.

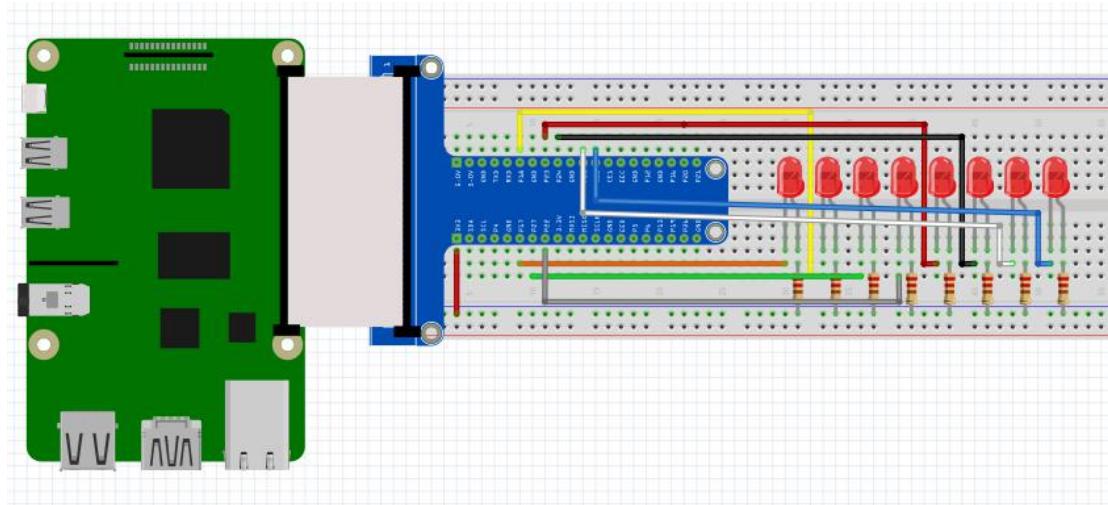
Erforderliche Teile

- 1 x Raspberry Pi
- 8 x LEDs
- 8 x 220 Ohm Widerstands
- 9 x Jumper Kabel
- 1 x Steckbrett

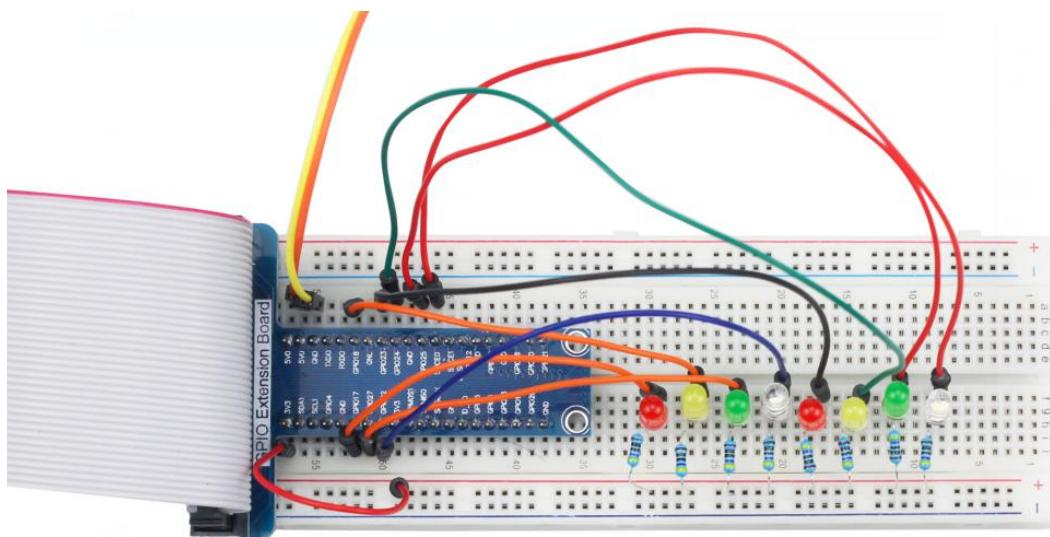
Verbindung Diagramm



Verdrahtung Diagramm



Beispiel Abbildung



C code

Öffnen Sie das Terminal und geben Sie den Befehl `cd code/C/3.Waterlight/` ein, um das Codeverzeichnis Waterlight.c aufzurufen.

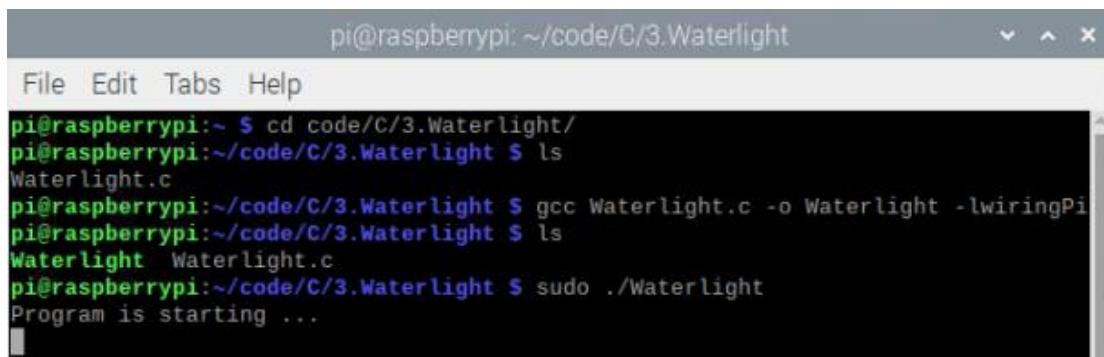
Geben Sie den Befehl `ls` ein, um die Datei im Verzeichnis Waterlight.c anzuzeigen.



```
pi@raspberrypi:~/code/C/3.Waterlight
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/3.Waterlight/
pi@raspberrypi:~/code/C/3.Waterlight $ ls
Waterlight.c
pi@raspberrypi:~/code/C/3.Waterlight $
```

Geben Sie den Befehl `gcc Waterlight.c -o Waterlight -lwiringPi` ein, um die ausführbare Datei `Waterlight.c` zu generieren. Geben Sie den Befehl `ls` ein, um sie anzuzeigen.

Führen Sie den Code aus, indem Sie den Befehl `sudo ./Waterlight` eingeben. Die Ergebnisse lauten wie folgt:



```
pi@raspberrypi:~/code/C/3.Waterlight
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/3.Waterlight/
pi@raspberrypi:~/code/C/3.Waterlight $ ls
Waterlight.c
pi@raspberrypi:~/code/C/3.Waterlight $ gcc Waterlight.c -o Waterlight -lwiringPi
pi@raspberrypi:~/code/C/3.Waterlight $ ls
Waterlight Waterlight.c
pi@raspberrypi:~/code/C/3.Waterlight $ sudo ./Waterlight
Program is starting ...

```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>
#define leds 8
int pins[leds] = {0,1,2,3,4,5,6,10};
void led_on(int n)
{
    digitalWrite(n, LOW);
}

void led_off(int n)
{
    digitalWrite(n, HIGH);
```

```
}

int main(void)
{
    int i;
    printf("Program is starting ... \n");

    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    for(i=0;i<leds;i++){
        pinMode(pins[i], OUTPUT);
    }
    while(1){
        for(i=0;i<leds;i++){
            led_on(pins[i]);
            delay(100);
            led_off(pins[i]);
        }
        for(i=leds-1;i>=0;i--){
            led_on(pins[i]);
            delay(100);
            led_off(pins[i]);
        }
    }
    return 0;
}
```

Code interpretation

```
while(1){
    for(i=0;i<leds;i++){
        led_on(pins[i]);
        delay(100);
        led_off(pins[i]);
    }
    for(i=leds-1;i>=0;i--){
        led_on(pins[i]);
        delay(100);
        led_off(pins[i]);
    }
}
```

```
}
```

```
}
```

Konfigurieren Sie GPIO0-GPIO7 im Programm für den Ausgabemodus. Dann werden in der "while" Schleife der Hauptfunktion zwei "for" Schleifen verwendet, um den Wasserfluss von links nach rechts und von rechts nach links zu bewirken.

Python code

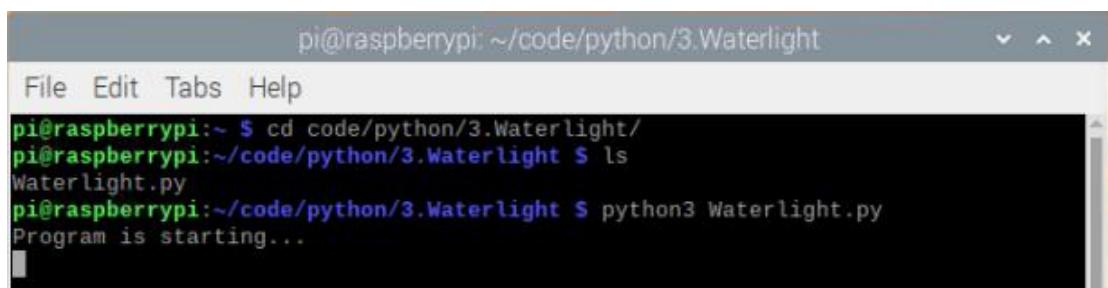
Öffnen Sie das Terminal und geben Sie mit dem Befehl `cd code/python/3.Waterlight/` das Codeverzeichnis ein;

Geben Sie den Befehl `ls` ein, um die Datei `Waterlight.py` im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/python/3.Waterlight
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/3.Waterlight/
pi@raspberrypi:~/code/python/3.Waterlight $ ls
Waterlight.py
pi@raspberrypi:~/code/python/3.Waterlight $
```

Führen Sie den Code aus, indem Sie den Befehl `python3 Waterlight.py` eingeben. Die Ergebnisse lauten wie folgt:



```
pi@raspberrypi:~/code/python/3.Waterlight
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/3.Waterlight/
pi@raspberrypi:~/code/python/3.Waterlight $ ls
Waterlight.py
pi@raspberrypi:~/code/python/3.Waterlight $ python3 Waterlight.py
Program is starting...

```

Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import time
ledPins = [11, 12, 13, 15, 16, 18, 22, 24]
def setup():
    print ('Program is starting...')
    GPIO.setmode(GPIO.BOARD)
    for pin in ledPins:
```

```
GPIO.setup(pin, GPIO.OUT)
GPIO.output(pin, GPIO.HIGH)

def loop():
    while True:
        for pin in ledPins:
            GPIO.output(pin, GPIO.LOW)
            time.sleep(0.1)
            GPIO.output(pin, GPIO.HIGH)

    for pin in ledPins[::-1]:
        GPIO.output(pin, GPIO.LOW)
        time.sleep(0.1)
        GPIO.output(pin, GPIO.HIGH)

def destroy():
    for pin in ledPins:
        GPIO.output(pin, GPIO.HIGH)
    GPIO.cleanup()

if __name__ == '__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy() will be executed.
        destroy()
```

Code interpretation

Import RPi.GPIO as GPIO

Import time

ledPins = [11, 12, 13, 15, 16, 18, 22, 24]

Importieren Sie das erforderliche RPi.GPIO-Zeitmodul

Wir benötigen 8 Pins für den Streamer. Definieren Sie daher einen variablen Array ‘ledPins’ Speicher Pin.

Def setup():

Print ('Program is starting...')

GPIO.setmode(GPIO.BARD)

For pin in ledPins:

GPIO.setup(pin, GPIO.OUT)

GPIO.output(pin, GPIO.HIGH)

Def loop():

While True:

For pin in ledPins:

GPIO.output(pin, GPIO.LOW)

Time.sleep(0.1)

GPIO.output(pin, GPIO.HIGH)

For pin in ledPins[::-1]:

GPIO.output(pin, GPIO.LOW)

Time.sleep(0.1)

GPIO.output(pin, GPIO.HIGH)

In der Funktion 'loop ()' wird zwei "for" Schleifen werden verwendet, um die Durchflusslichter von rechts nach links und von links nach rechts zu implementieren. 'ledPins[::-1]' wird verwendet, um 'ledPins' in umgekehrter Reihenfolge 'Elemente' zu durchlaufen.

Lektion 4 Analog & PWM

Überblick

In dieser Lektion lernen wir, wie Sie die Helligkeit einer LED steuern.

Erforderliche Teile

1 x Raspberry Pi

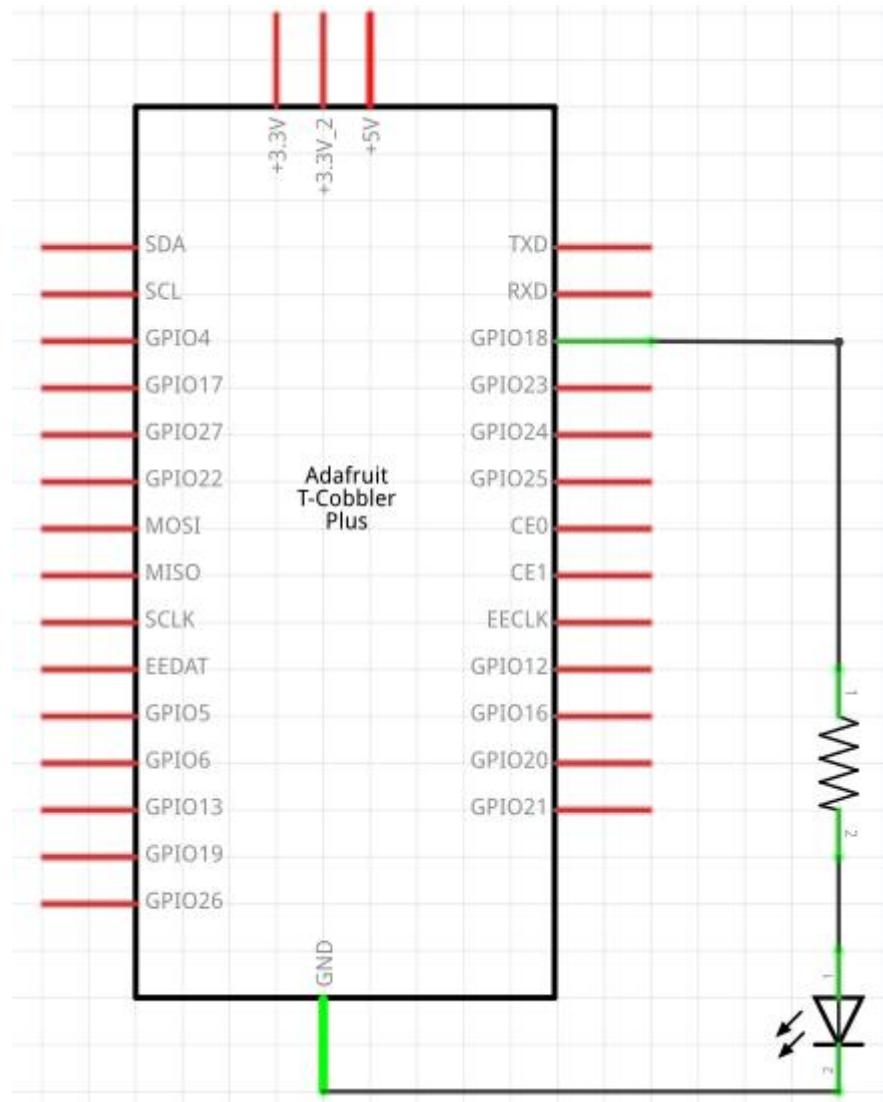
1 x LED

1 x 220 ohm Widerstand

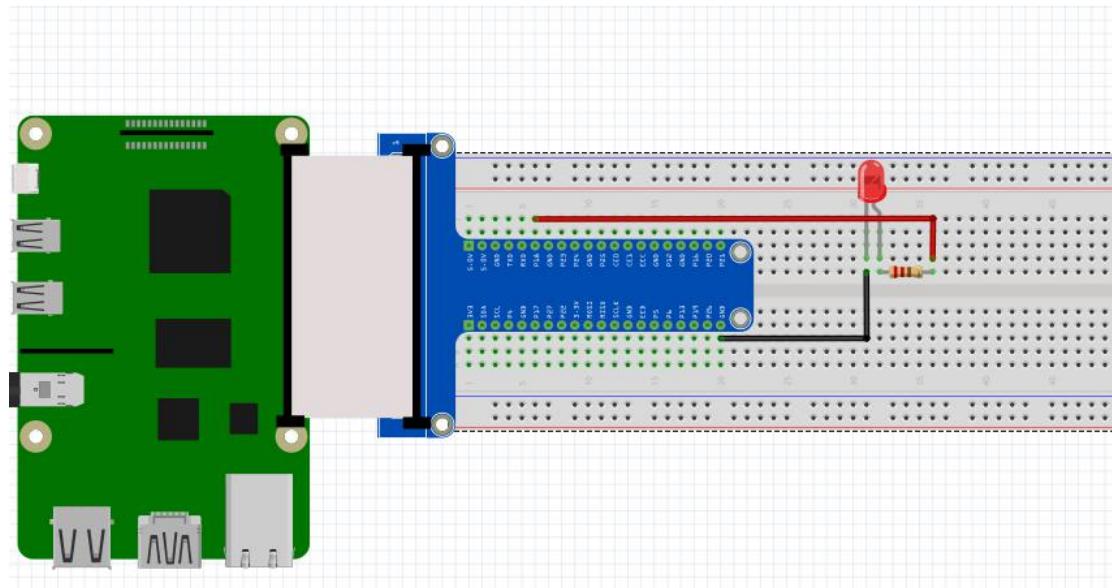
2 x Jumper Kabel

1 x Steckbrett

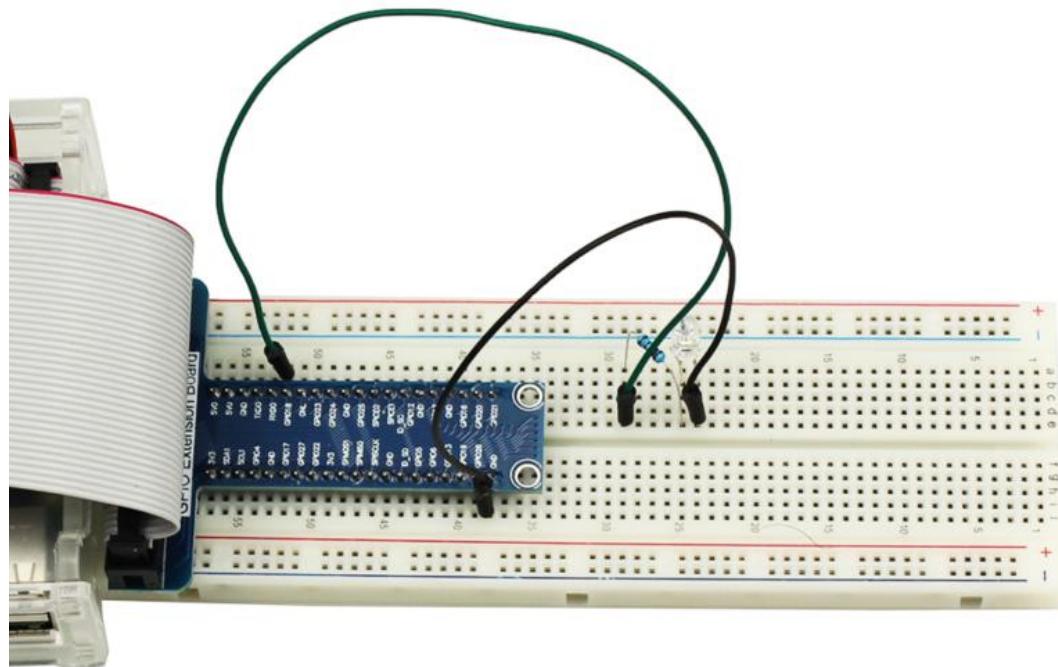
Schaltplan



Verdrahtung Diagramm



Beispiel Abbildung



C code

Öffnen Sie das Terminal und geben Sie den `cd code/C/4.PWMLED/` ein, um das Codeverzeichnis `pwmLED.c` aufzurufen.

Geben Sie den Befehl `ls` ein, um die Datei `pwmLED.c` im Verzeichnis anzuzeigen.



```
pi@raspberrypi: ~/code/C/4.PWMLED
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/4.PWMLED/
pi@raspberrypi:~/code/C/4.PWMLED $ ls
pwmLED.c
pi@raspberrypi:~/code/C/4.PWMLED $
```

Geben Sie den Befehl `gcc Waterlight.c -o Waterlight -lwiringPi` ein, um die ausführbare Datei `Waterlight.c` `Waterlight` zu generieren. Geben Sie den Befehl `ls` ein, um sie anzuzeigen.

Führen Sie den Code aus, indem Sie den Befehl `sudo ./Waterlight` eingeben. Die Ergebnisse lauten wie folgt:



```
pi@raspberrypi: ~/code/C/4.PWMLED
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/4.PWMLED/
pi@raspberrypi:~/code/C/4.PWMLED $ ls
pwmLED.c
pi@raspberrypi:~/code/C/4.PWMLED $ gcc pwmLED.c -o pwm -lwiringPi
pi@raspberrypi:~/code/C/4.PWMLED $ ls
pwm  pwmLED.c
pi@raspberrypi:~/code/C/4.PWMLED $ sudo ./pwm
pi@raspberrypi:~/code/C/4.PWMLED $
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>
#include <softPwm.h>

#define ledPin    1
int main(void)
{
    int i;
```

```

if(wiringPiSetup() == -1){
    printf("setup wiringPi failed !");
    return 1;
}

softPwmCreate(ledPin, 0, 100);

while(1){
    for(i=0;i<100;i++){
        softPwmWrite(ledPin, i);
        delay(20);
    }
    delay(300);
    for(i=100;i>=0;i--){
        softPwmWrite(ledPin, i);
        delay(20);
    }
    delay(300);
}
return 0;
}

```

Code interpretation

```

while(1){
    for(i=0;i<100;i++){
        softPwmWrite(ledPin, i);
        delay(20);
    }
    delay(300);
    for(i=100;i>=0;i--){
        softPwmWrite(ledPin, i);
        delay(20);
    }
    delay(300);
}

```

Es gibt zwei "for" Schleifen in der "while" Schleife. Die erste macht die ledPin Ausgabe PWM von 0% bis 100% und die zweite macht die ledPin Ausgabe PWM von 100% bis 0%.

Sie können auch die Rate einstellen, mit der sich der LED Status ändert, indem Sie die Parameter der Funktion delay () in der "for" Schleife ändern.

```
int softPwmCreate (int pin, int initialValue, int pwmRange) ;
```

Erstellen Sie einen softwaregesteuerten PWM Pin.

```
void softPwmWrite (int pin, int value) ;
```

Aktualisieren Sie den PWM Wert am Pin.

Python code

Öffnen Sie das Terminal und geben Sie mit dem Befehl `cd code/python/4.PWMLED/` das Codeverzeichnis ein.

Geben Sie den Befehl `ls` ein, um die Datei `pwmLED.py` im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/python/4.PWMLED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/4.PWMLED/
pi@raspberrypi:~/code/python/4.PWMLED $ ls
pwmLED.py
pi@raspberrypi:~/code/python/4.PWMLED $
```

Führen Sie den Code mit dem Befehl `python3 pwmLED.py` aus. Die Ergebnisse lauten wie folgt:



```
pi@raspberrypi:~/code/python/4.PWMLED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/4.PWMLED/
pi@raspberrypi:~/code/python/4.PWMLED $ ls
pwmLED.py
pi@raspberrypi:~/code/python/4.PWMLED $ python3 pwmLED.py
```

Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import time
LedPin = 12
```

```
def setup():
    global p
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(LedPin, GPIO.OUT)
    GPIO.output(LedPin, GPIO.LOW)
    p = GPIO.PWM(LedPin, 1000)

    p.start(0)

def loop():
    while True:
        for dc in range(0, 101, 1):
            p.ChangeDutyCycle(dc)
            time.sleep(0.01)
        time.sleep(1)
        for dc in range(100, -1, -1):
            p.ChangeDutyCycle(dc)
            time.sleep(0.01)
        time.sleep(1)

def destroy():
    p.stop()
    GPIO.output(LedPin, GPIO.LOW)
    GPIO.cleanup()

if __name__ == '__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy() will be executed.
        destroy()
```

Code interpretation

Def setup():

Global p

GPIO.setmode(GPIO.BOARD)

GPIO.setup(LedPin, GPIO.OUT)

GPIO.output(LedPin, GPIO.LOW)

p = GPIO.PWM(LedPin, 1000)

P.start(0)

Definieren Sie eine Variable 'p' vom Typ 'global', mit der der Modus des GPIO entsprechend der physischen Position des Pins eingestellt wird. Die LED ist mit einem IO Port namens GPIO18 verbunden. Und 'LedPin' ist als 12 definiert und wird entsprechend dem physischen Pin-Modus auf den Ausgangsmodus eingestellt.

Erstellen Sie dann eine PWM Instanz und stellen Sie die PWM Frequenz auf 1000 Hz mit einem anfänglichen Arbeitszyklus von 0% ein.

Def loop():

While True:

For dc in range(0, 101, 1):

p.ChangeDutyCycle(dc)

Time.sleep(0.01)

Time.sleep(1)

For dc in range(100, -1, -1):

p.ChangeDutyCycle(dc)

Time.sleep(0.01)

Time.sleep(1)

In der "while" Schleife gibt es zwei "for" Schleifen für LED Atmungseffekte. Die erste macht die 'LedPin' Ausgabe PWM von 0% auf 100% und die zweite macht die 'LEDPin' Ausgabe PWM von 100% auf 0%.

Lektion 5 RGBLED

Überblick

In dieser Lektion lernen Sie, wie Sie das Raspberry Pi Netzteil und die Widerstände verwenden, um die RGB LEDs gemeinsam zu beleuchten.

Erforderliche Teile

1 x Himbeer-Pi

1 x RGB LED

3 x 220 Ohm Widerstand

4 x Jumper Kabel

1 x Steckbrett

Produkt Einführung

RGB LED

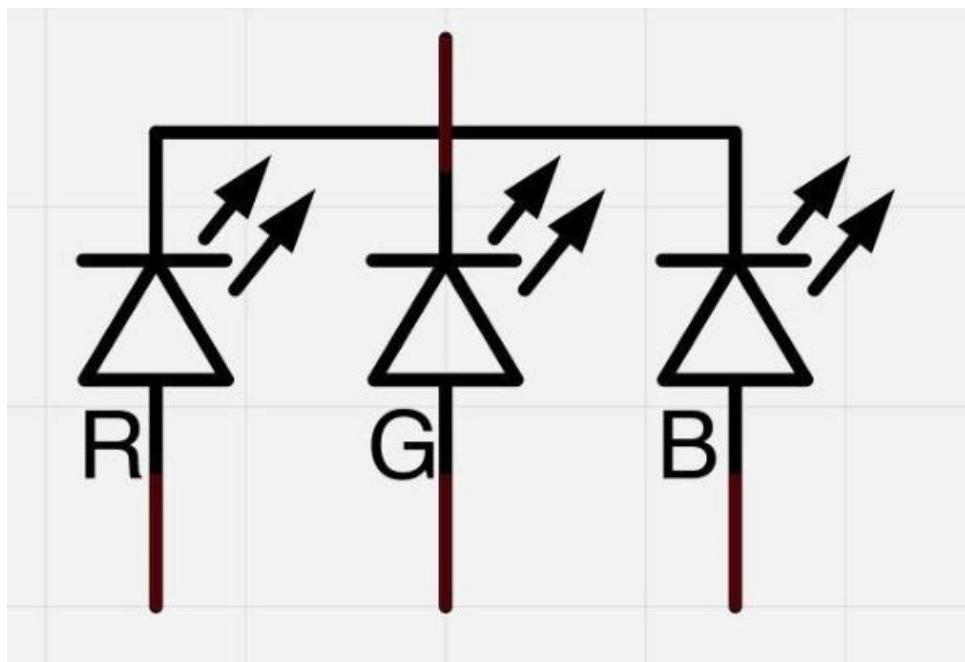
Die RGB-LED sieht von der Oberfläche aus wie eine normale LED aus, außer dass die RGB-LED vier Stifte und die normale LED nur zwei Stifte hat. Tatsächlich ist die RGB-LED intern mit drei LEDs ausgestattet, einer roten, einer grünen und einer blauen.

Sie können jede gewünschte Farbe anzeigen, indem Sie die Helligkeit jeder der drei LEDs anpassen. Die Art und Weise, in der die LED-Farben gemischt werden, entspricht der Art und Weise, in der die Farbe auf der Palette gemischt wird. Es ist sehr schwierig, die Farbanpassung ohne die Hauptsteuerkarte durchzuführen. Glücklicherweise können wir das UNO R3-Entwicklungsboard verwenden, um unsere Arbeit zu vereinfachen. Diese Karte hat die Funktion des analogen Schreibens. Sie können die LED steuern, indem Sie die mit " ~ " gekennzeichneten Pins der Platine verwenden, um variable Leistung auszugeben.

In diesem Tutorial werden wir die gemeinsame Kathoden-LED verwenden. Ein Pin ist mit Masse verbunden, während die anderen drei Pins mit „ ~ “ mit dem digitalen Pin UNO R3 verbunden sind.

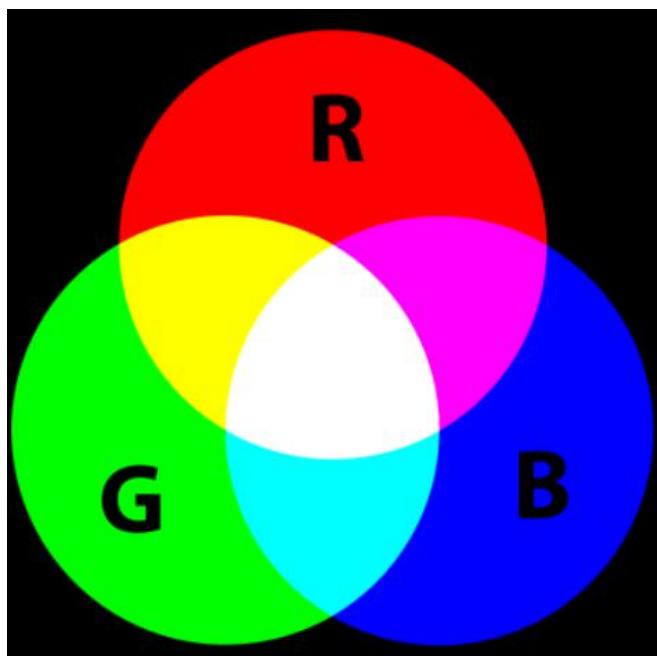


Dieser Pin benötigt keinen Widerstand, um direkt an GND anzuschließen. Die anderen drei Pins erfordern jedoch Widerstände, um zu verhindern, dass übermäßiger Strom fließt. Die anderen drei Stifte werden ebenfalls positive Stifte. Die linke Seite der CATHODE ist der Reihe nach der GRÜNE und der BLAUE Stift, und die rechte Seite ist der ROTE Stift.



Farbe

Wir können jede gewünschte Farbe mischen, indem wir die Beleuchtungsstärke der roten, grünen und blauen LEDs ändern. Theoretisch haben unsere Augen drei Arten von Lichtempfängern (rot, grün, blau). Unsere Augen und unser Gehirn empfangen und verarbeiten die Rot-, Grün- und Blautöne und wandeln sie in eine Spektralfarbe um. Heutzutage, elektronische Produkte wie Farbfernseher.

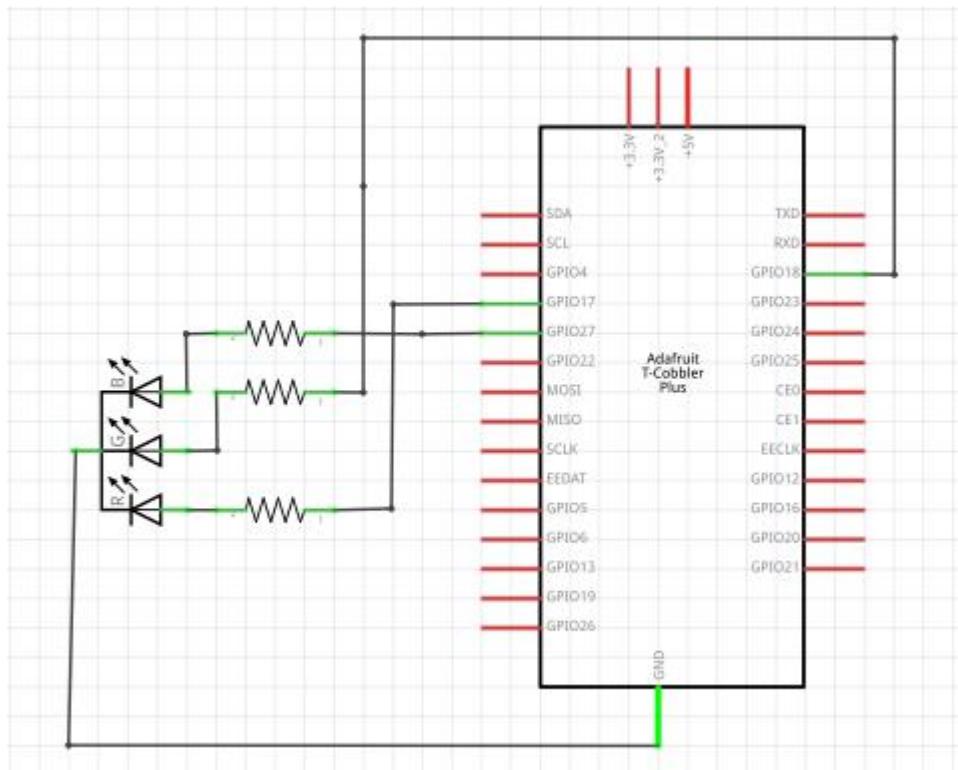


Wenn wir beispielsweise die drei LEDs auf die gleiche Helligkeit einstellen, wird die gesamte Lichtfarbe in Weiß angezeigt. Wenn wir die blaue LED ausschalten, nur die rote und die grüne LED mit der gleichen Helligkeit, wird die LED gelb gefärbt.

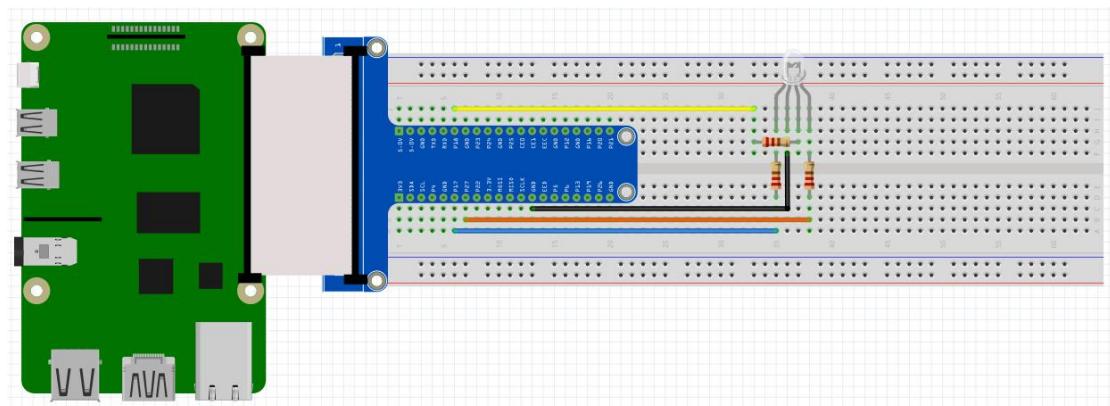
Wir können die Helligkeit von Rot, Grün und Blau jeder RGB-LED leicht steuern, wodurch es einfach ist, verschiedene Farben zu steuern.

Um die Farbe Schwarz anzupassen, müssen wir die Helligkeit der drei Farben auf die niedrigste einstellen. Weil Schwarz eine fast matte Farbe ist.

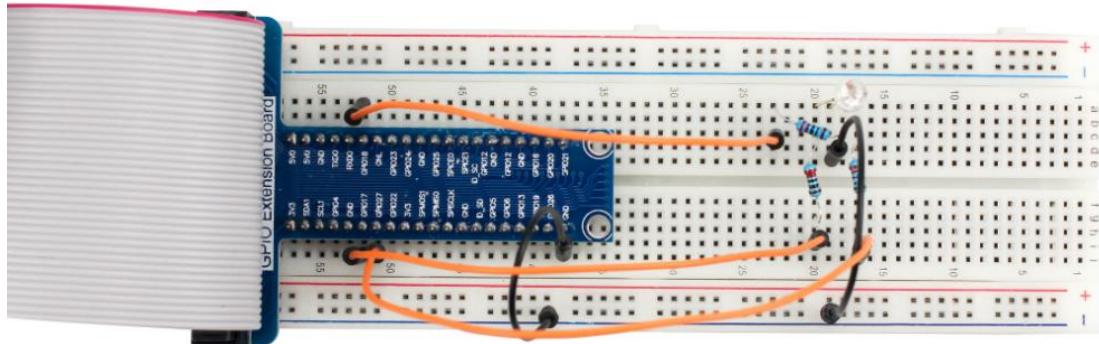
Schaltplan



Verdrahtung Diagramm



Beispiel Abbildung



C code

Öffnen Sie den Befehl zur Eingabe des [cd code/C/5.RGBLED/](#) im Codeverzeichnis RGBLED.c.

Geben Sie den Befehl [ls](#) ein, um die Datei im Verzeichnis RGBLED.c anzuzeigen.

```
pi@raspberrypi:~/code/C/5.RGBLED
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/5.RGBLED/
pi@raspberrypi:~/code/C/5.RGBLED $ ls
RGBLED.c
pi@raspberrypi:~/code/C/5.RGBLED $
```

Geben Sie den Befehl [gcc RGBLED.c -o RGBLED -lwiringPi -lpthread](#) ein, um die ausführbare Datei [RGBLED.c](#) **RGBLED** zu generieren, geben Sie den Befehl [ls](#) zum Anzeigen ein.

Geben Sie den Befehl [sudo ./RGBLED](#) ein, um den Code auszuführen. Die Ergebnisse lauten wie folgt:

```
pi@raspberrypi:~/code/C/5.RGBLED
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/5.RGBLED/
pi@raspberrypi:~/code/C/5.RGBLED $ ls
RGBLED.c
pi@raspberrypi:~/code/C/5.RGBLED $ gcc RGBLED.c -o RGBLED -lwiringPi -lpthread
pi@raspberrypi:~/code/C/5.RGBLED $ ls
RGBLED RGBLED.c
pi@raspberrypi:~/code/C/5.RGBLED $ sudo ./RGBLED
Program is starting ...
r=83, g=86, b=77
r=15, g=93, b=35
r=86, g=92, b=49
r=21, g=62, b=27
r=90, g=59, b=63
r=26, g=40, b=26
r=72, g=36, b=11
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>
#include <stdlib.h>

#define ledPinRed    0
#define ledPinGreen   1
#define ledPinBlue    2

void ledInit(void)
{
    softPwmCreate(ledPinRed, 0, 100);
    softPwmCreate(ledPinGreen, 0, 100);
    softPwmCreate(ledPinBlue, 0, 100);
}

void ledColorSet(int r_val, int g_val, int b_val)
{
    softPwmWrite(ledPinRed, r_val);
    softPwmWrite(ledPinGreen, g_val);
```

```
softPwmWrite(ledPinBlue, b_val);
}

int main(void)
{
    int r,g,b;
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    printf("Program is starting ...\\n");
    ledInit();

    while(1){
        r=random()%100;
        g=random()%100;
        b=random()%100;
        ledColorSet(r,g,b);
        printf("r=%d, g=%d, b=%d \\n",r,g,b);
        delay(300);
    }
    return 0;
}
```

Code interpretation

```
Void ledInit(void)

{

softPwmCreate(ledPinRed, 0, 100);

softPwmCreate(ledPinGreen,0, 100);

softPwmCreate(ledPinBlue, 0, 100);

}
```

Erstellen Sie in der Unterfunktion von 'ledInit ()' einen Software PWM Steuerstift, der die Stifte R, G und B steuert.

```
Void ledColorSet(int r_val, int g_val, int b_val)

{
    softPwmWrite(ledPinRed, r_val);

    softPwmWrite(ledPinGreen, g_val);

    softPwmWrite(ledPinBlue, b_val);

}
```

Erstellen Sie dann eine Unterfunktion und stellen Sie die PWM für die drei Pins ein.

```
While(1){

r=random()%100;

g=random()%100;

b=random()%100;

ledColorSet(r,g,b);

printf("r=%d, g=%d, b=%d \n",r,g,b);

delay(300);

}
```

Schließlich werden im "while" Zyklus der Hauptfunktion drei Zufallszahlen erhalten und als PWM-Arbeitszyklus bezeichnet und den entsprechenden Pins zugewiesen. Daher kann die RGBLED die Farben immer zufällig wechseln.

Die zugehörigen Funktionen der Software PWM können wie folgt beschrieben werden:

Long random();

Diese Funktion gibt eine Zufallszahl zurück.

Python code

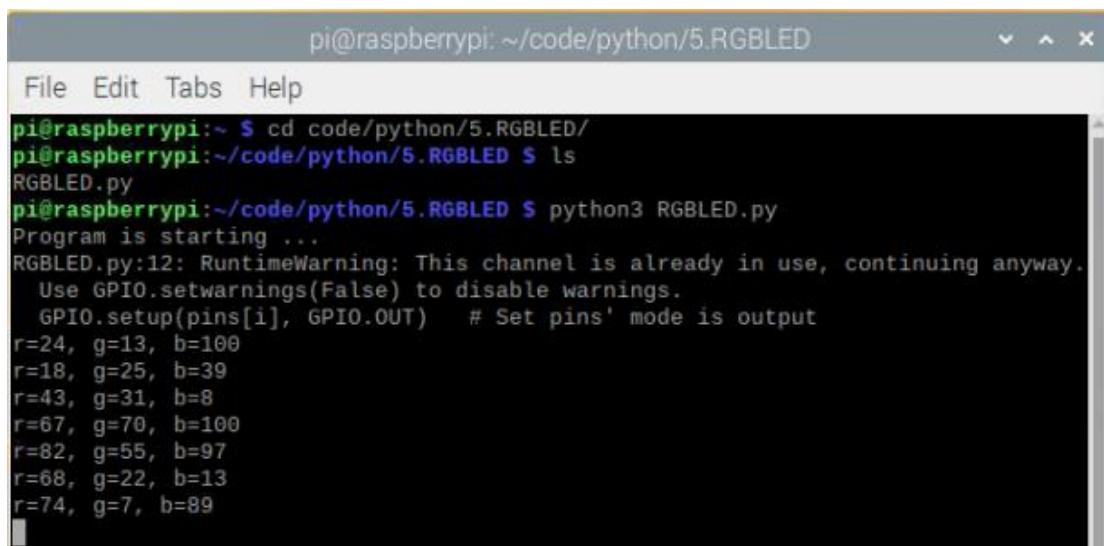
Öffnen Sie das Terminal und geben Sie mit dem Befehl `cd code/python/5.RGBLED/` das Codeverzeichnis ein.

Geben Sie `ls` ein, um die Datei `RGBLED.py` im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/python/5.RGBLED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/5.RGBLED/
pi@raspberrypi:~/code/python/5.RGBLED $ ls
RGBLED.py
pi@raspberrypi:~/code/python/5.RGBLED $
```

Geben Sie `python3 RGBLED.py` ein, um den Code auszuführen. Die Ergebnisse lauten wie folgt:



```
pi@raspberrypi:~/code/python/5.RGBLED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/5.RGBLED/
pi@raspberrypi:~/code/python/5.RGBLED $ ls
RGBLED.py
pi@raspberrypi:~/code/python/5.RGBLED $ python3 RGBLED.py
Program is starting ...
RGBLED.py:12: RuntimeWarning: This channel is already in use, continuing anyway.
  Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(pins[i], GPIO.OUT)  # Set pins' mode is output
r=24, g=13, b=100
r=18, g=25, b=39
r=43, g=31, b=8
r=67, g=70, b=100
r=82, g=55, b=97
r=68, g=22, b=13
r=74, g=7, b=89
|
```

Das Folgende ist der Code:

```
import RPi.GPIO as GPIO
import time
import random
pins = {'pin_R':11, 'pin_G':12, 'pin_B':13}
def setup():
    global p_R,p_G,p_B
    print ('Program is starting ... ')
    GPIO.setmode(GPIO.BOARD)
    for i in pins:
```

```

GPIO.setup(pins[i], GPIO.OUT)

GPIO.output(pins[i], GPIO.HIGH)
p_R = GPIO.PWM(pins['pin_R'], 2000)
p_G = GPIO.PWM(pins['pin_G'], 2000)
p_B = GPIO.PWM(pins['pin_B'], 2000)
p_R.start(0)
p_G.start(0)
p_B.start(0)
def setColor(r_val,g_val,b_val):
    p_R.ChangeDutyCycle(r_val)
    p_G.ChangeDutyCycle(g_val)
    p_B.ChangeDutyCycle(b_val)
def loop():
    while True :
        r=random.randint(0,100)#get a random in (0,100)
        g=random.randint(0,100)
        b=random.randint(0,100)
        setColor(r,g,b)#set random as a duty cycle value
        print ('r=%d, g=%d, b=%d' %(r ,g, b))
        time.sleep(0.3)
def destroy():
    p_R.stop()
    p_G.stop()
    p_B.stop()
    GPIO.cleanup()
if __name__ == '__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy() will be executed.

destroy()

```

Code interpretation

```

def loop():
while  True ::

r= random. randint( (0, ,100) )
g= random. randint( (0, ,100) )

```

```
b= =random. randint( 0, ,100 )  
setColor( (r, ,g, ,b) )  
print ( ('r=%d, g=%d, b=%d'  %(r , ,g, , b ))  
time. sleep( (0.3) )
```

Im vorherigen Kapitel haben wir gelernt, wie man mit der Python-Sprache eine Pin-Out-PWM erstellt. In diesem Projekt haben wir drei Pins Ausgang PWM, die Verwendung ist genau die gleiche wie im vorherigen Kapitel. Im "while" Zyklus der "Schleifen" Funktion erhalten wir zuerst drei Zufallszahlen und geben dann die drei Zufallszahlen als PWM Werte der drei Pins an. Lassen Sie RGBLEDs zufällig zwischen verschiedenen Farben wechseln.

random.randint(a, b)

Diese Funktion kann eine zufällige Ganzzahl innerhalb des angegebenen Bereichs (a, b) zurückgeben.

Lektion 6 Aktiver Summer

Überblick

In dieser Lektion lernen Sie, wie Sie mit dem Raspberry Pi einen Summer ertönen.

Erforderliche Teile

1 x Raspberry Pi

1 x Aktiver Summer

1 x 220 Ohm Widerstand

6 x Männlich zu Männlich Jumper Kabel

1 x Steckbrett

1 x Taste

1 x S8050NPN Triode

Produkt Einführung

Aktiver Summer

Summer werden mit Gleichstrom betrieben und sind mit einer integrierten Schaltung ausgestattet. Sie werden häufig in Computern, Druckern, Fotokopierern, Alarmen, elektronischem Spielzeug, elektronischen Kraftfahrzeugen, Telefonen, Zeitschaltuhren und anderen elektronischen Produkten für Sprachgeräte verwendet.

Summer können in aktive und passive unterteilt werden. Wenn man die Seite der Stifte betrachtet, ist der eine mit einer grünen Leiterplatte ein passiver Summer, während der andere, der mit einem schwarzen Band umschlossen ist, ein aktiver ist.

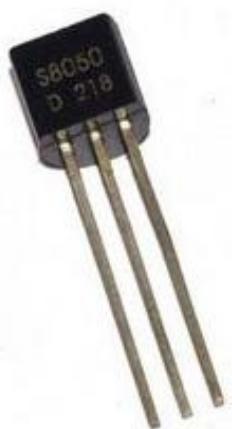
Der Unterschied besteht darin, dass der aktive Summer über integrierte Schwingungen verfügt, sodass er bei Stromversorgung Geräusche erzeugt. Passive Summer haben keine solche Quelle, daher wird bei Verwendung eines Gleichstromsignals kein Ton erzeugt. Stattdessen müssen Sie eine Rechteckwelle mit einer Frequenz zwischen 2k und 5k verwenden, um sie anzutreiben.



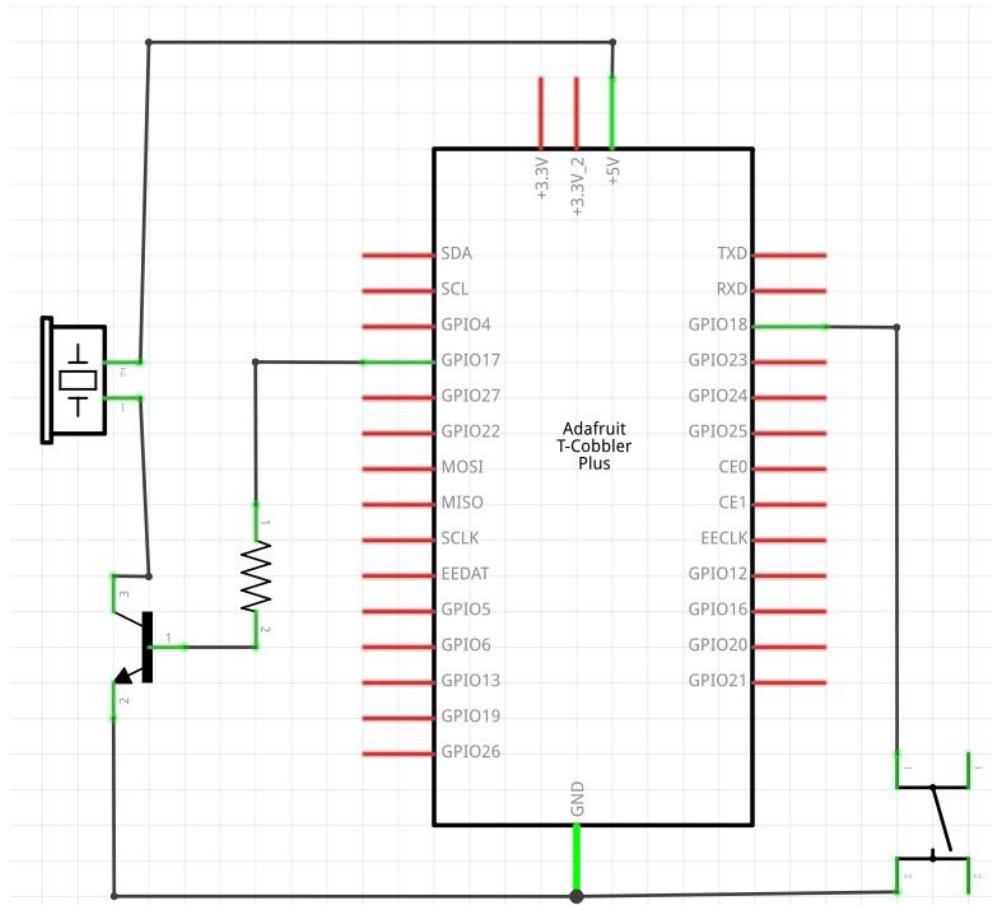
S8050 Triode

Die Triode S8050 ist eine NPN-Siliziumröhre mit geringer Leistung. Die Kollektor Basis Spannung (V_{cbo}) kann bis zu 40 V betragen und der Kollektorstrom beträgt (I_c) 0,5 A. S8050 ist eines der am häufigsten verwendeten Halbleitertriodenmodelle für das Design von Schaltungshardware.

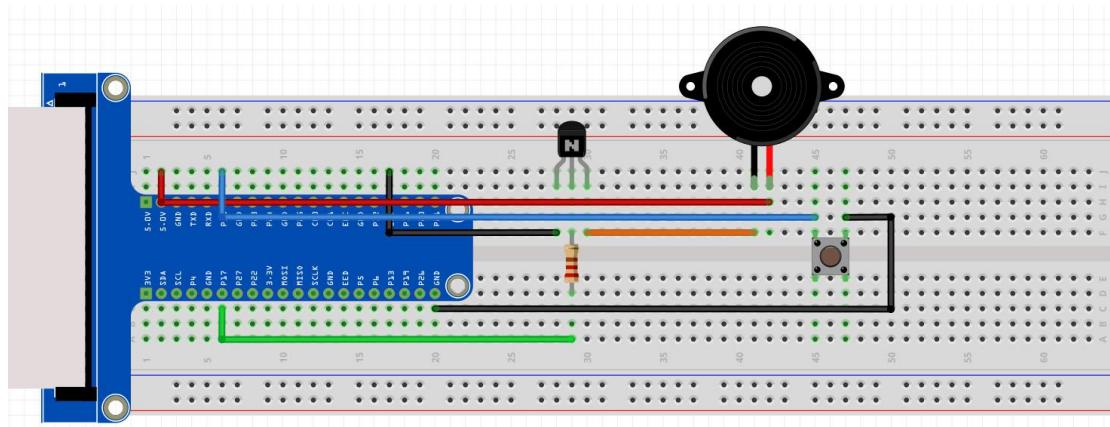
Es wird häufig in verschiedenen Verstärkerschaltungen gesehen und hat ein breites Anwendungsspektrum, hauptsächlich für die Hochfrequenzverstärkung. Kann auch als Schaltkreis verwendet werden. Von links nach rechts sind die Stifte der Emitter, die Basis und der Kollektor.



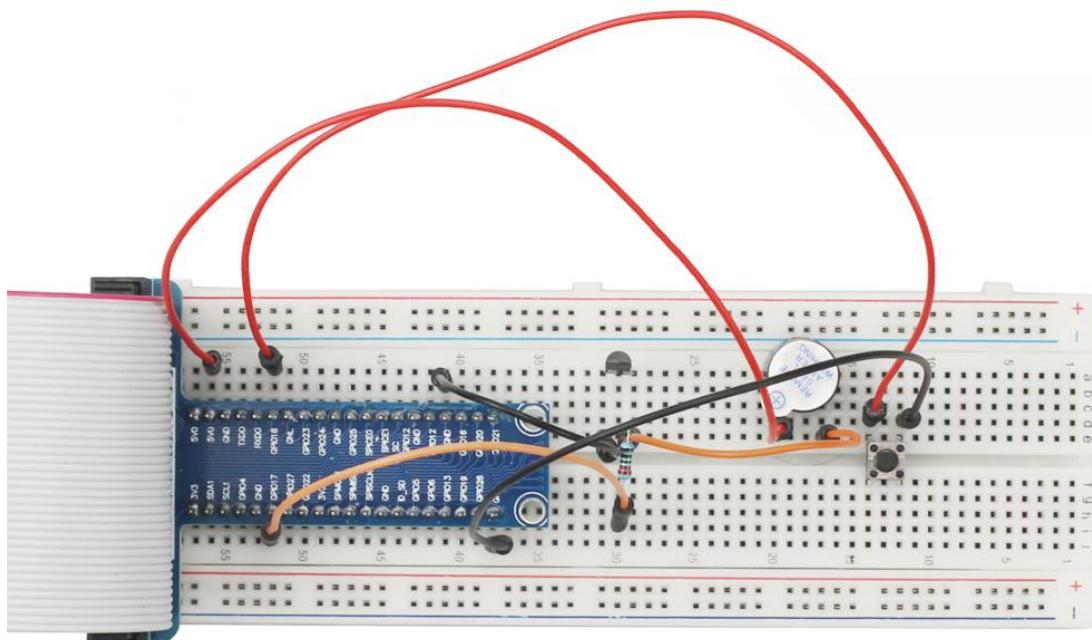
Schaltplan



Verdrahtung Diagramm



Beispielbild



C code

Öffnen Sie das Terminal und geben Sie den [cd code / C / 6.Buzzer](#) / ein, um das Codeverzeichnis Buzzer.c aufzurufen;

Geben Sie den Befehl [ls](#) ein, um die Datei Buzzer.c im Verzeichnis anzuzeigen.

```
pi@raspberrypi:~/code/C/6.Buzzer
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/6.Buzzer/
pi@raspberrypi:~/code/C/6.Buzzer $ ls
Buzzer.c
pi@raspberrypi:~/code/C/6.Buzzer $
```

Geben Sie den Befehl [gcc Buzzer.c -o buzzer -lwiringPi](#) ein, um den ausführbaren Summer Buzzer.c zu generieren, und geben Sie den Befehl [ls](#) ein, um ihn anzuzeigen.

Geben Sie den Befehl `sudo ./buzzer` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
...buzzer off  
...buzzer off
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>  
#include <stdio.h>  
  
#define buzzerPin      0  
#define buttonPin 1  
  
int main(void)  
{  
    if(wiringPiSetup() == -1){  
        printf("setup wiringPi failed !");  
        return 1;  
    }  
  
    pinMode(buzzerPin, OUTPUT);  
    pinMode(buttonPin, INPUT);  
  
    pullUpDnControl(buttonPin, PUD_UP);  
    while(1){  
  
        if(digitalRead(buttonPin) == LOW){  
            digitalWrite(buzzerPin, HIGH);  
            printf("buzzer on...\n");  
        }  
        else {  
            digitalWrite(buzzerPin, LOW);  
        }  
    }  
}
```

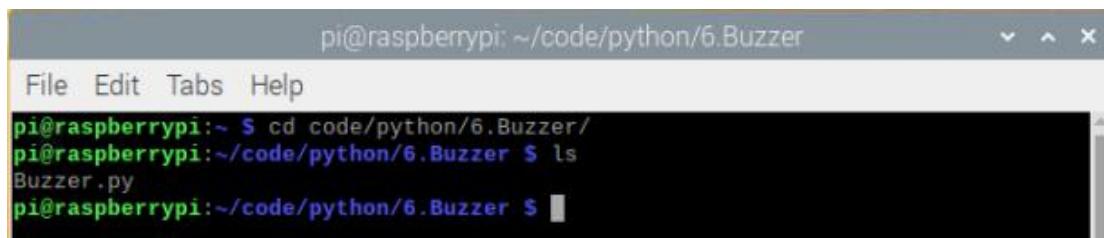
```
        printf("...buzzer off\n");
    }
}

return 0;
}
```

Python code

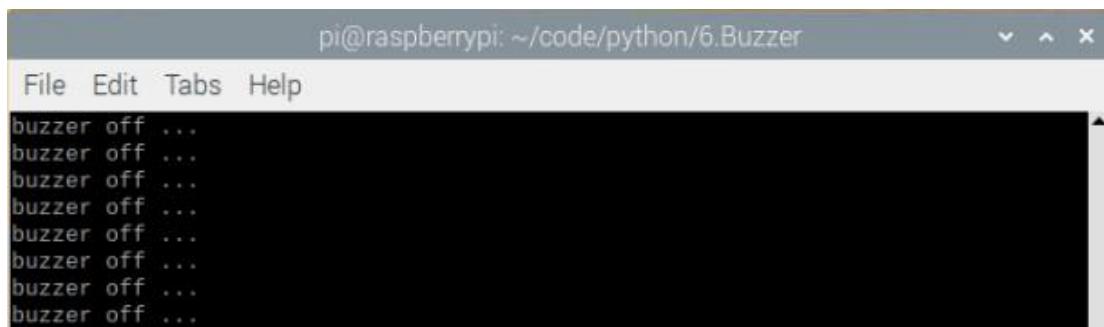
Öffnen Sie das Terminal und geben Sie mit dem Befehl `cd code / python / 6.Buzzer /` das Codeverzeichnis ein;

Geben Sie den Befehl `ls` ein, um die Datei Buzzer.py im Verzeichnis anzuzeigen.



```
pi@raspberrypi: ~/code/python/6.Buzzer
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/6.Buzzer/
pi@raspberrypi:~/code/python/6.Buzzer $ ls
Buzzer.py
pi@raspberrypi:~/code/python/6.Buzzer $
```

Geben Sie den Befehl `python3 Buzzer.py` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi: ~/code/python/6.Buzzer
File Edit Tabs Help
buzzer off ...
```

Das Folgende ist der Code:

```
import RPi.GPIO as GPIO
buzzerPin = 11
buttonPin = 12
def setup():
    print ('Program is starting...')
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(buzzerPin, GPIO.OUT)
```

```

GPIO.setup(buttonPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
def loop():
    while True:
        if GPIO.input(buttonPin)==GPIO.LOW:
            GPIO.output(buzzerPin,GPIO.HIGH)
            print ('buzzer on ...')
        else :
            GPIO.output(buzzerPin,GPIO.LOW)
            print ('buzzer off ...')
def destroy():
    GPIO.output(buzzerPin, GPIO.LOW)
    GPIO.cleanup()

if __name__ == '__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy() will be executed.
        destroy()

```

Code Interpretation

```

def loop():
    while True:
        if GPIO.input(buttonPin)==GPIO.LOW:
            GPIO.output(buzzerPin,GPIO.HIGH)
            print ('buzzer on ...')
        else :
            GPIO.output(buzzerPin,GPIO.LOW)
            print ('buzzer off ...')

```

Implementieren Sie alle Aktionen in der Funktion 'loop ()'. Verwenden Sie die 'if'-Anweisung, um zu bestimmen, wann die Taste gedrückt wird. Wenn diese Taste gedrückt wird, aktivieren Sie den Summer mit der Anweisung 'GPIO.output (buzzerPin, GPIO.HIGH)'. Andernfalls deaktivieren Sie mit der Anweisung 'GPIO.output (buzzerPin, GPIO.LOW)' den Summer.

Lektion 7 Passiver Summer

Überblick

In dieser Lektion lernen Sie, wie Sie mit dem Raspberry Pi einen passiven Summer ertönen.

Erforderliche Teile

1 x Raspberry Pi

1 x Passiver Summer

1 x 1K Widerstand

6 x doppelte Männlich zu Männlich Jumper Kabel

1 x Steckbrett

1 x Taste

1 x S8050NPN Triode

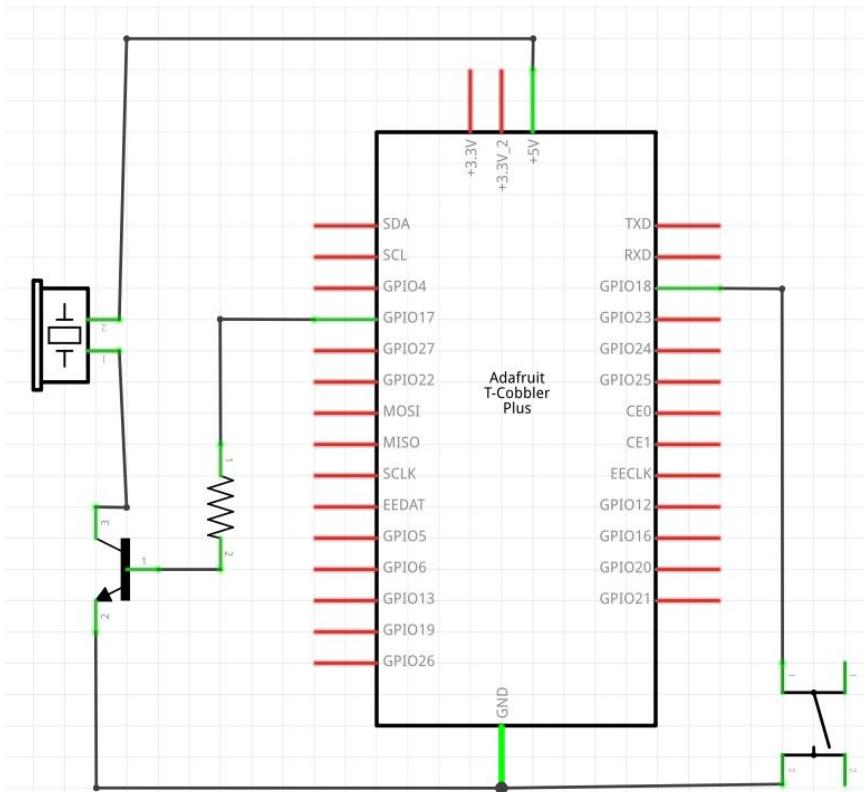
Produkt Einführung

Passiver Summer

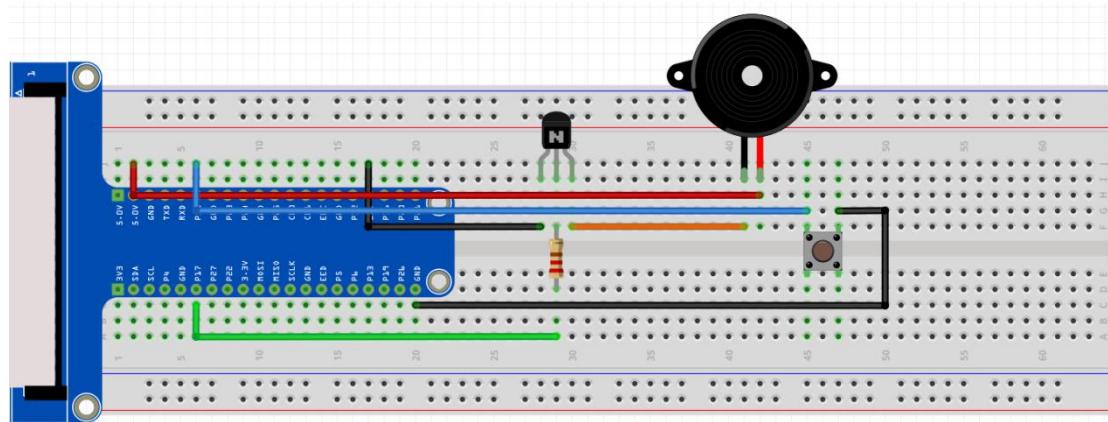
Der passive Summer erzeugt mithilfe von PWM Schall, indem Luftvibrationen erzielt werden. Solange die Schwingungsfrequenz geändert wird, können unterschiedliche Geräusche erzeugt werden. Senden Sie beispielsweise einen 523Hz Impuls, der Do erzeugen kann, einen Impuls von 587 Hz, der IF Re erzeugen kann, 659Hz Impuls einen Impuls von 659 Hz, der Mi erzeugen kann.



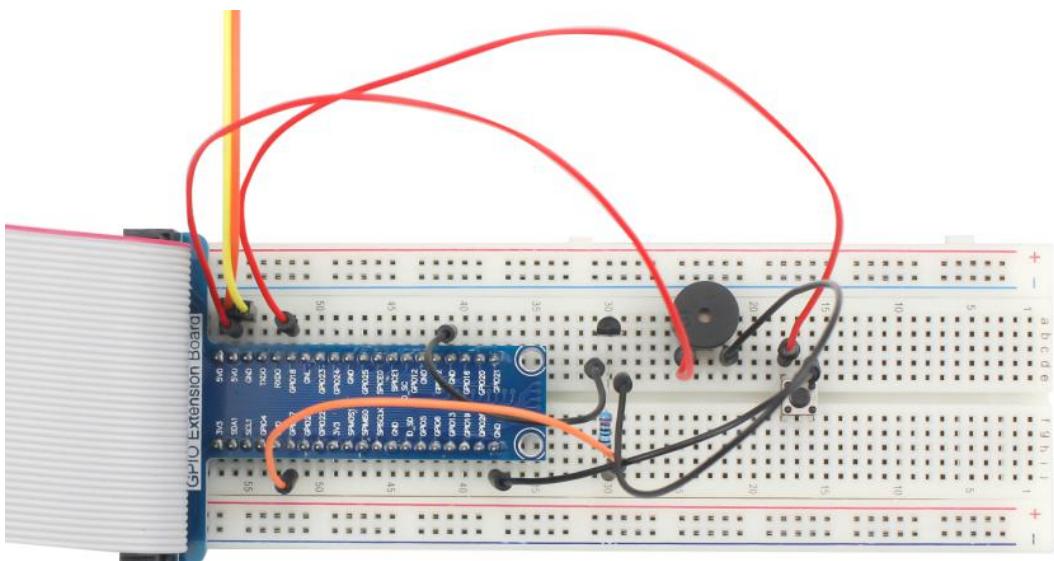
Schaltplan



Verdrahtung Diagramm



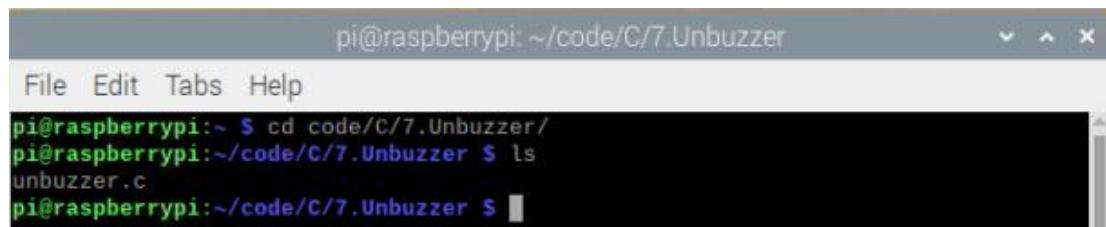
Beispielbild



C code

Öffnen Sie das Terminal und geben Sie den Befehl `cd code / C / 7.Unbuzzer /` ein, um das Codeverzeichnis unbuzzer.c aufzurufen;

Geben Sie den Befehl `ls` ein, um die Datei unbuzzer.c im Verzeichnis anzuzeigen.

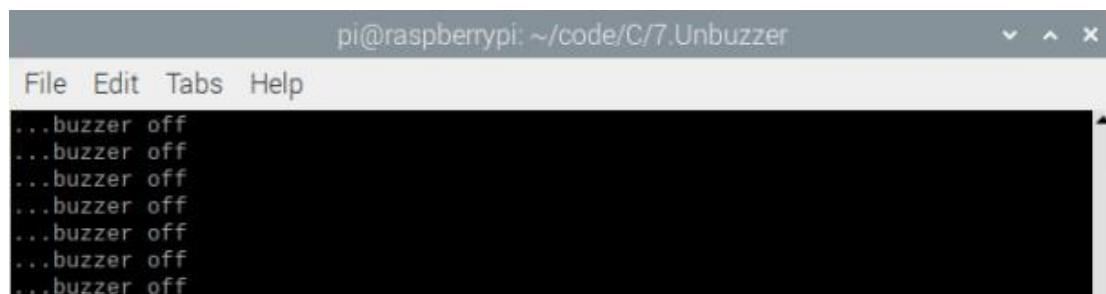


```
pi@raspberrypi: ~/code/C/7.Unbuzzer
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/7.Unbuzzer/
pi@raspberrypi:~/code/C/7.Unbuzzer $ ls
unbuzzer.c
pi@raspberrypi:~/code/C/7.Unbuzzer $
```

A screenshot of a terminal window titled "pi@raspberrypi: ~/code/C/7.Unbuzzer". The window has a standard title bar with File, Edit, Tabs, and Help options. The main area shows a command-line session. The user first types "cd code/C/7.Unbuzzer/" followed by pressing Enter. Then, they type "ls" and press Enter again. The output shows a single file named "unbuzzer.c". The terminal window has a dark background with light-colored text and a scroll bar on the right side.

Geben Sie den Befehl `gcc unbuzzer.c -o unbuzzer -lwiringPi -lm -lpthread` ein, um die ausführbare Datei unbuzzer.c unbuzzer zu generieren, und geben Sie den Befehl `ls` zum Anzeigen ein

Geben Sie den Befehl `sudo ./unbuzzer` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi: ~/code/C/7.Unbuzzer
File Edit Tabs Help
...buzzer off
```

A screenshot of a terminal window titled "pi@raspberrypi: ~/code/C/7.Unbuzzer". The window shows a series of seven lines of text, each consisting of "...buzzer off", indicating the program is running and printing its status to the terminal.

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>
#include <softTone.h>
#include <math.h>
#define buzzPin 0
#define buttonPin 1
void alertor(int pin){
    int x;
    double sinVal, toneVal;
    for(x=0;x<360;x++){
        toneVal = sin((x * PI) / 180);
        if(digitalRead(buttonPin) == 1)
            toneVal = -toneVal;
        softTone(buzzPin, toneVal, 1000);
        delay(1);
    }
}
```

```
sinVal = sin(x * (M_PI / 180));
toneVal = 2000 + sinVal * 500;
softToneWrite(pin,toneVal);
delay(1);
}
}

void stopAlertor(int pin){
    softToneWrite(pin,0);
}

int main(void)
{
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    pinMode(buzzeRPin, OUTPUT);
    pinMode(buttonPin, INPUT);
    softToneCreate(buzzeRPin);
    pullUpDnControl(buttonPin, PUD_UP);
    while(1){
        if(digitalRead(buttonPin) == LOW){
            alertor(buzzeRPin);
            printf("alertor on...\n");
        }
        else {
            stopAlertor(buzzeRPin);
            printf "...buzzer off\n";
        }
    }
    return 0;
}
```

Code Interpretation

softToneCreate(buzzeRPin);

Logischerweise ist der Code der gleiche wie der aktive Summer, aber die Methode zur Steuerung des Summers ist unterschiedlich. Passiver Summer erfordert eine bestimmte Frequenz von PWM zur Steuerung, daher müssen Sie einen Software PWM Pin über 'softToneCreate' (buzzeRPin) erstellen. Hier wird 'softTone' speziell verwendet, um eine Rechteckwelle mit einer festen Frequenz von 50% und einer

variablen Frequenz und einem variablen Arbeitszyklus zu erzeugen, was eine bessere Wahl für die Steuerung des Summers ist.

```
void alertor(int pin){  
    int x;  
    double sinVal, toneVal;  
    for(x=0;x<360;x++){  
        sinVal = sin(x * (M_PI / 180));  
        toneVal = 2000 + sinVal * 500;  
        softToneWrite(pin,toneVal);  
        delay(1);  
    }  
}
```

In der 'while' Schleife der Hauptfunktion wird beim Drücken der Taste die Unterfunktion 'alertor ()' aufgerufen und der Alarm ertönt ein Warnton. Die Frequenzkurve des Alarms basiert auf einer Sinuskurve. Wir müssen den Sinuswert zwischen 0 und 360 Grad berechnen und ihn mit einem Wert (500) plus der Resonanzfrequenz des Summers multiplizieren. Wir können die 'PWM' Frequenz über 'softToneWrite' (**pin**, toneVal). einstellen.

```
void stopAlertor(int pin){  
    softToneWrite(pin,0);  
}
```

Um den Summer auszuschalten, stellen Sie einfach die PWM-Frequenz des Summerpins auf 0.

Die Funktionen von "softTone" sind wie folgt:

```
int softToneCreate (int pin) ;
```

Erstellen Sie einen softwaregesteuerten Tonstift.

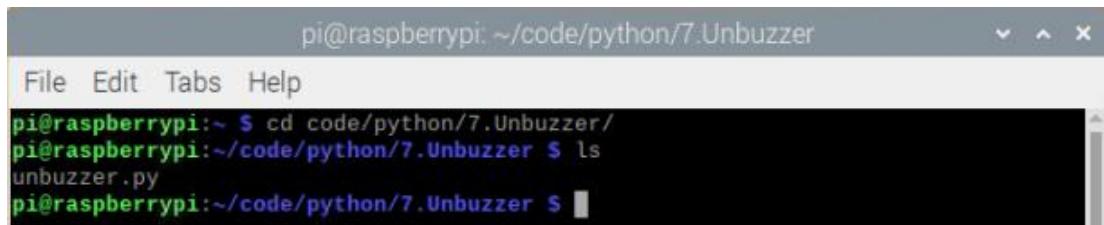
```
void softToneWrite (int pin, int freq) ;
```

Aktualisiert den Audiofrequenzwert an einem bestimmten Pin.Erstellen.

Python code

Öffnen Sie das Terminal und geben Sie mit dem Befehl [cd code / python / 7.Unbuzzer](#) / das Codeverzeichnis ein;

Geben Sie den Befehl `ls` ein, um die Datei Unbuzzer.py im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/python/7.Unbuzzer
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/7.Unbuzzer/
pi@raspberrypi:~/code/python/7.Unbuzzer $ ls
unbuzzer.py
pi@raspberrypi:~/code/python/7.Unbuzzer $
```

Geben Sie den Befehl `python3 Unbuzzer.py` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi:~/code/python/7.Unbuzzer
File Edit Tabs Help
buzzer off ...
```

Das Folgende ist der Code:

```
import RPi.GPIO as GPIO
import time
import math

buzzerPin = 11
buttonPin = 12

def setup():
    global p
    print ('Program is starting...')
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(buzzerPin, GPIO.OUT)
    GPIO.setup(buttonPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    p = GPIO.PWM(buzzerPin, 1)
    p.start(0);

def loop():
    while True:
        if GPIO.input(buttonPin)==GPIO.LOW:
```

```

        alertor()
        print ('buzzer on ...')
else :
    stopAlertor()
    print ('buzzer off ...')

def alertor():
    p.start(50)
for x in range(0,361):
    sinVal = math.sin(x * (math.pi / 180.0))
    toneVal = 2000 + sinVal * 500
    p.ChangeFrequency(toneVal)
    time.sleep(0.001)

def stopAlertor():
    p.stop()

def destroy():
    GPIO.output(buzzerPin, GPIO.LOW)
    GPIO.cleanup()

if __name__ == '__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()

```

Code Interpretation

Der Code ist logisch mit dem aktiven Summer identisch, die Steuerung des Summers ist jedoch unterschiedlich. Passiver Summer erfordert PWM mit einer bestimmten Frequenz zur Steuerung. Sie müssen daher einen Software PWM Pin über softToneCreate (buzzeRPin) erstellen. Die Methode zum Erstellen von PWM wird auch in RGB LED Leuchten eingeführt.

```

def alertor():
    p.start(50)
for x in range(0,361):
    sinVal = math.sin(x * (math.pi / 180.0))
    toneVal = 2000 + sinVal * 500
    p.ChangeFrequency(toneVal)
    time.sleep(0.001)

```

Während des while Zyklus der Hauptfunktion wird beim Drücken der Taste der subfunction alertor() aufgerufen und der Alertor gibt einen Warnton aus. Die Frequenzkurve des Alarms basiert auf der Sinuskurve. Wir müssen den Sinuswert von 0 bis 360 Grad berechnen und einen bestimmten Wert (the value: 500) plus die Resonanzfrequenz des Summers multiplizieren. Wir können die PWM-Frequenz über ‘p.ChangeFrequency (toneVal)’ einstellen.

```
def stopAlertor():
    p.stop()
```

Wenn die Taste losgelassen wird, schaltet sich der Summer aus.

Lektion 8 AD/DA Wandler

Überblick

In dieser Lektion lernen Sie, wie Sie analoge Größen über das AD/DA Modul lesen, in digitale Größen umwandeln und die digitale Größe in analoge Ausgänge umwandeln. Das heißt, ADC und DAC.

Erforderliche Teile

- 1 x Raspberry Pi
- 1 x PCF8591 Modul
- 1 x 220 Ohm Widerstand
- 11 x Jumper Kabel
- 1 x Steckbrett
- 1 x Potentiometer
- 1 x LED

Produkt Einführung

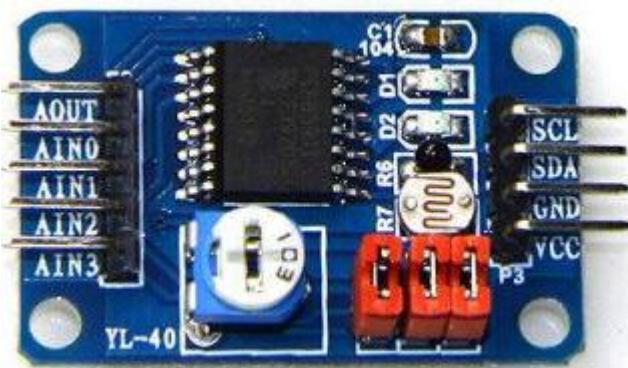
Potentiometer

Das Potentiometer ist ein Widerstand mit drei Pins und der Widerstandswert kann gemäß einer bestimmten Variationsregel eingestellt werden. Potentiometer bestehen üblicherweise aus einem Widerstand und einer beweglichen Bürste. Wenn sich die abnehmbare Bürste entlang des Widerstands bewegt, das Ausgangsende erhält einen gewissen Widerstandswert oder Spannung, die in einem bestimmten Verhältnis zur Verschiebung steht. Das Potentiometer kann als Dreipol verwendet werden oder eine zweipolige Komponente. Letzteres kann als variabler Widerstand angesehen werden. Diese Lektion konzentriert sich auf die Verwendung als variabler Widerstand.

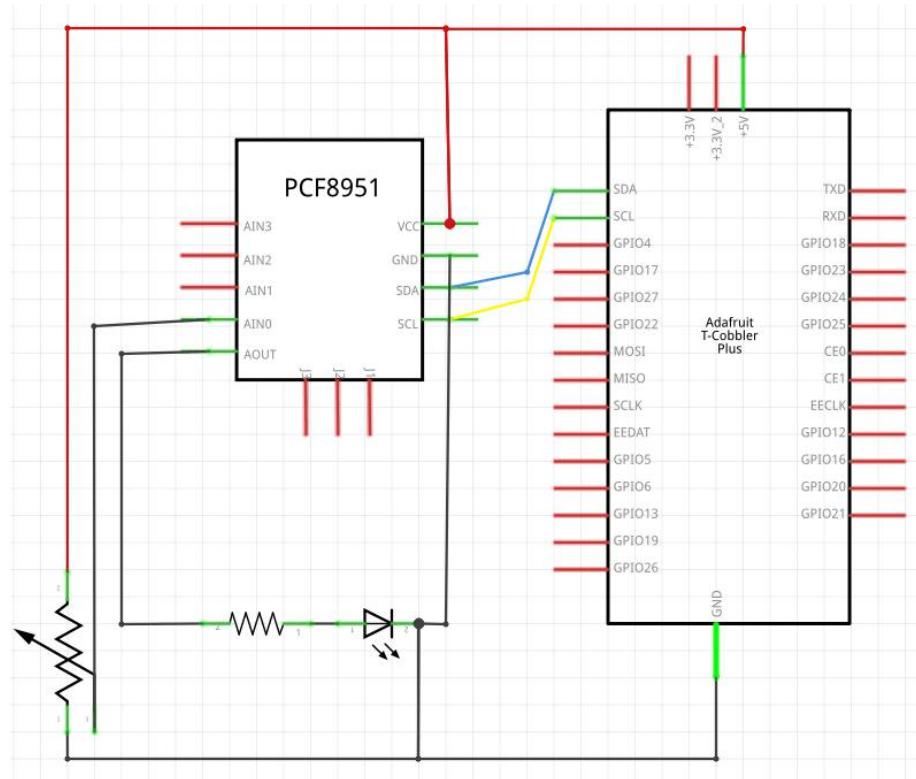


PCF8591 Modul

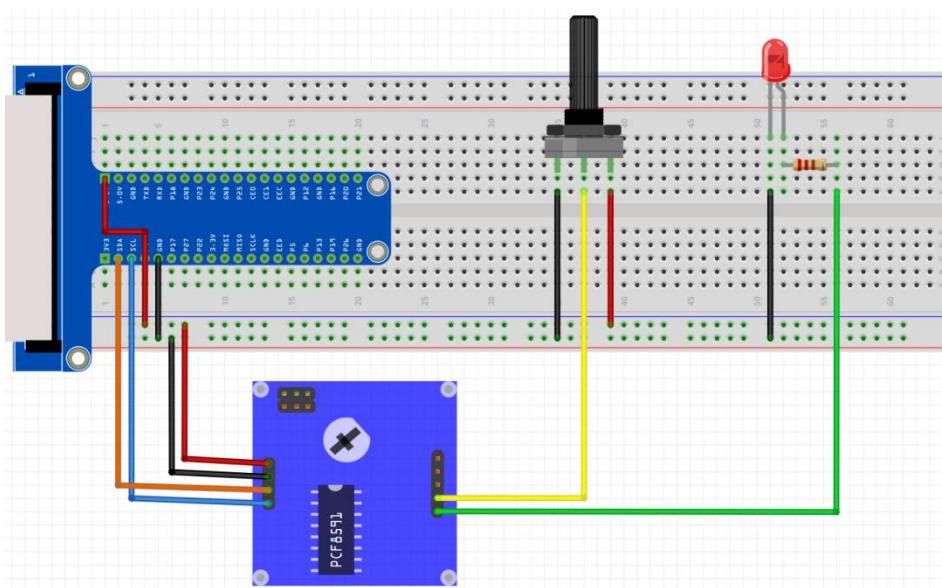
Der PCF8591 ist ein Single-Chip 8-Bit CMOS Daten Erfassungsgerät mit geringem Stromverbrauch und einfacher Versorgung, vier analogen Eingängen, einem analogen Ausgang und einer seriellen I2C-bus Schnittstelle. Die drei Adresspins A0, A1 und A2 des PCF8591 können verwendet werden Hardware Adressprogrammierung, die den Zugriff auf acht PCF8591 Geräte auf demselben I2C-bus ohne zusätzliche Hardware ermöglicht. Die am PCF8591-Gerät eingegebenen Adress-, Steuer- und Datensignale werden seriell über einen bidirektionalen zweizeiligen I2C-Bus übertragen.



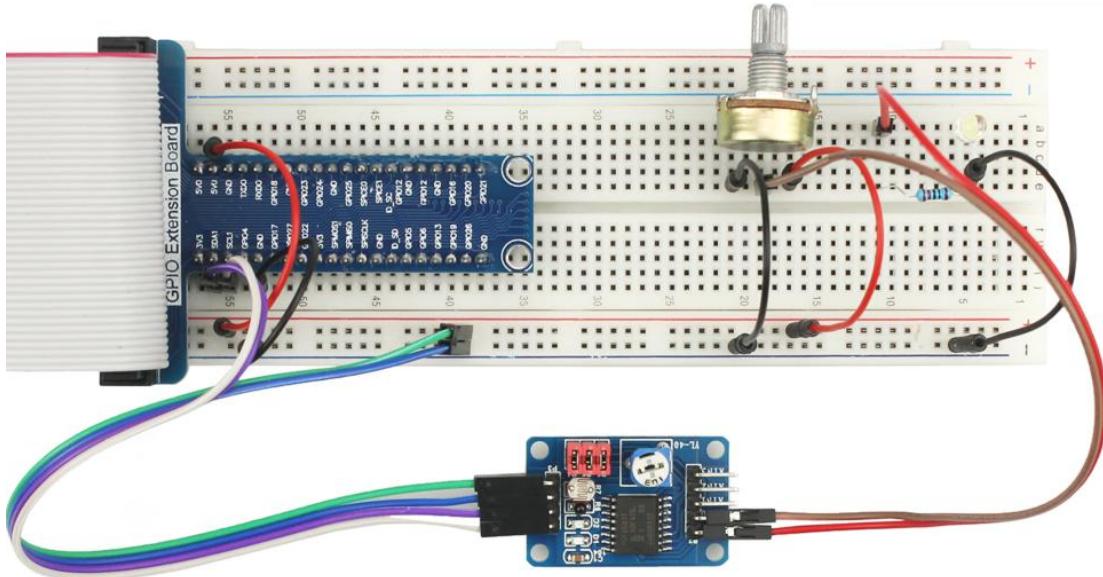
Schaltplan



Verdrahtung Diagramm



Beispiel Abbildung



Fügen Sie die I2C Bibliotheksdatei hinzu:

Geben Sie den Befehl [sudo apt-get install i2c-tools](#) ein und drücken Sie die Eingabetaste (siehe Abbildung unten).

```
pi@raspberrypi:~ $ sudo apt-get install i2c-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
i2c-tools is already the newest version (3.1.2-3).
i2c-tools set to manually installed.
The following packages were automatically installed and are no longer required:
  libgooglepinyin0 libscim8v5 libsunpinyin3v5 scim scim-gtk-immodule
  scim-im-agent scim-modules-socket sunpinyin-data
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 94 not upgraded.
pi@raspberrypi:~ $
```

C code

Öffnen Sie das Terminal und geben Sie den Befehl [cd code / C / 8.ADDA](#) / ein, um das AD.c code everzeichnis aufzurufen.

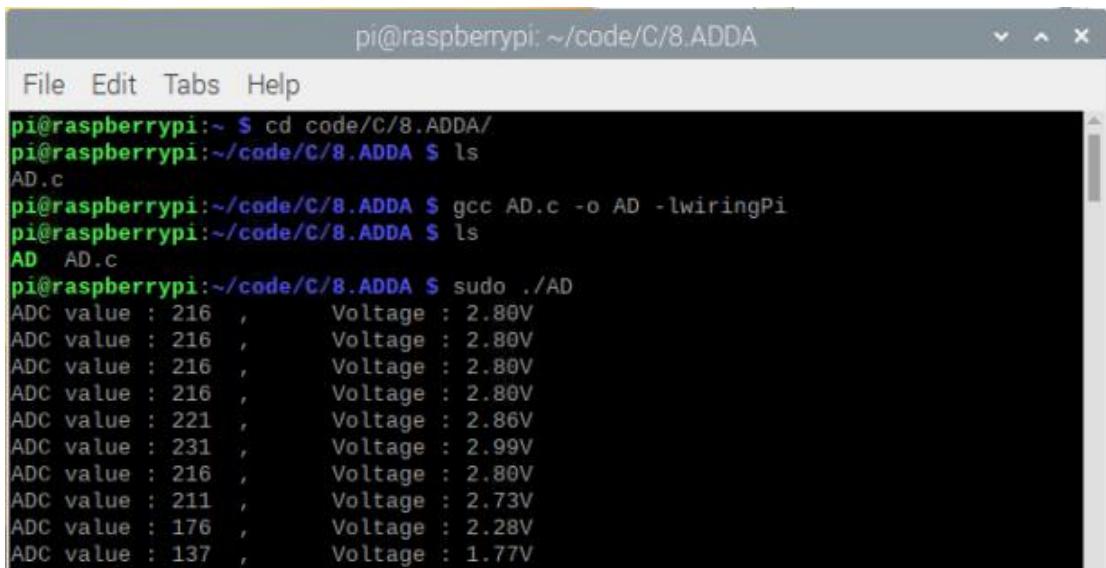
Geben Sie den Befehl `ls` ein, um die Datei AD.c im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/C/8.ADDA
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/8.ADDA/
pi@raspberrypi:~/code/C/8.ADDA $ ls
AD.c
pi@raspberrypi:~/code/C/8.ADDA $
```

Geben Sie den Befehl `gcc AD.c -o AD -lwiringPi` ein, um die ausführbare AD.c-Datei AD zu generieren. Geben Sie den Befehl `ls` zum Anzeigen ein.

Geben Sie den Befehl `sudo ./AD` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi:~/code/C/8.ADDA
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/8.ADDA/
pi@raspberrypi:~/code/C/8.ADDA $ ls
AD.c
pi@raspberrypi:~/code/C/8.ADDA $ gcc AD.c -o AD -lwiringPi
pi@raspberrypi:~/code/C/8.ADDA $ ls
AD AD.c
pi@raspberrypi:~/code/C/8.ADDA $ sudo ./AD
ADC value : 216 , Voltage : 2.80V
ADC value : 221 , Voltage : 2.86V
ADC value : 231 , Voltage : 2.99V
ADC value : 216 , Voltage : 2.80V
ADC value : 211 , Voltage : 2.73V
ADC value : 176 , Voltage : 2.28V
ADC value : 137 , Voltage : 1.77V
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>

#define address 0x48          //pcf8591 default address
#define pinbase 64            //any number above 64
#define A0 pinbase + 0
#define A1 pinbase + 1
#define A2 pinbase + 2
```

```
#define A3 pinbase + 3

int main(void){
    int value;
    float voltage;
    wiringPiSetup();
    pcf8591Setup(pinbase,address);

    while(1){
        value = analogRead(A0); //read A0 pin
        analogWrite(pinbase+0,value);
        voltage = (float)value / 255.0 * 3.3; // calculate voltage
        printf("ADC value : %d ,\tVoltage : %.2fV\n",value,voltage);
        delay(100);
    }
}
```

Code Interpretation

```
#define address 0x48      //pcf8591 default address
#define pinbase 64          //any number above 64
#define nbase + 0
#define A1 pinbase + 1
#define A2 pinbase + 2
#define A3 pinbase + 3
```

Die Standard I2C Adresse von PCF8591 lautet 0x48. Die Pin Basis kann ein beliebiger Wert größer oder gleich 64 sein. Wir definieren die ADC-Eingangskanäle A1, A2, A0, A3 des PCF8591.

`pcf8591Setup(pinbase,address);`

In der Hauptfunktion können ADC und DAC nach der Initialisierung von PCF8591 mit 'pcf8591Setup' (pinbase, address) mit den Funktionen 'analogRead ()' and 'analogWrite ()' betrieben werden.

```
while(1){
    value = analogRead(A0); //read A0 pin
    analogWrite(pinbase+0,value);
    voltage = (float)value / 255.0 * 3.3; // calculate voltage
    printf("ADC value : %d ,\tVoltage : %.2fV\n",value,voltage);
    delay(100);
```

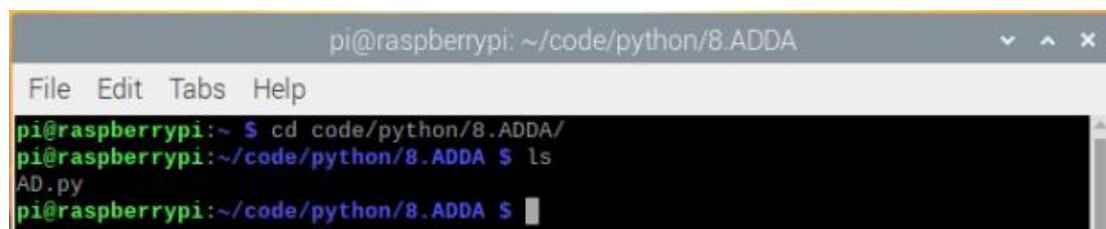
{}

Im "while" -Zyklus wird 'analogRead (A0)' verwendet, um den ADC Wert des A0 Ports (angeschlossenes Potentiometer) zu lesen, und dann wird der gelesene ADC Wert über 'AnalogWrite ()' ausgegeben. Der entsprechende Istspannungswert wird dann berechnet und angezeigt.

Python code

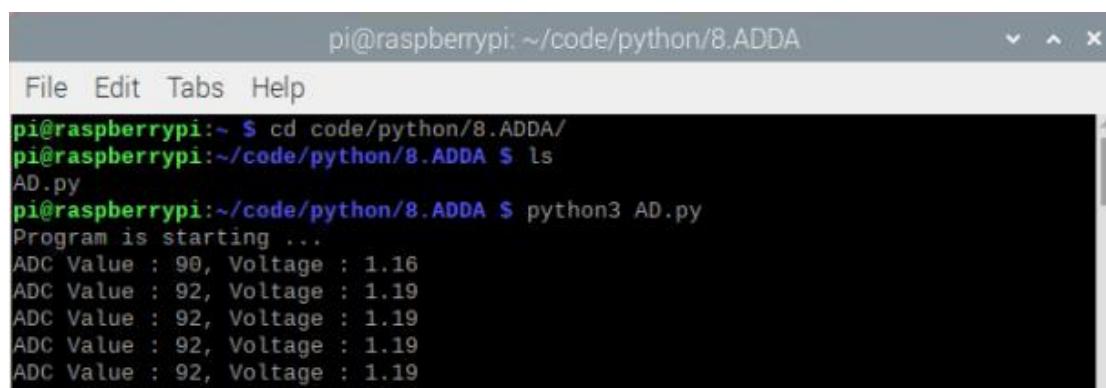
Öffnen Sie das Terminal und geben Sie mit dem Befehl `cd code / python / 8.ADDA /` das Codeverzeichnis ein;

Geben Sie den Befehl `ls` ein, um die Datei ADDA.py im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/python/8.ADDA
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/8.ADDA/
pi@raspberrypi:~/code/python/8.ADDA $ ls
AD.py
pi@raspberrypi:~/code/python/8.ADDA $
```

Geben Sie den Befehl `python3 ADDA.py` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi:~/code/python/8.ADDA
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/8.ADDA/
pi@raspberrypi:~/code/python/8.ADDA $ ls
AD.py
pi@raspberrypi:~/code/python/8.ADDA $ python3 AD.py
Program is starting ...
ADC Value : 90, Voltage : 1.16
ADC Value : 92, Voltage : 1.19
```

Das Folgende ist der Code:

```
import smbus
import time

address = 0x48
bus=smbus.SMBus(1)
```

cmd=0x40

```

def analogRead(chn):
    value = bus.read_byte_data(address,cmd+chn)
    return value

def analogWrite(value):
    bus.write_byte_data(address,cmd,value)

def loop():
    while True:
        value = analogRead(0)
        analogWrite(value)
        voltage = value / 255.0 * 3.3
        print ('ADC Value : %d, Voltage : %.2f%(value,voltage))
        time.sleep(0.01)

def destroy():
    bus.close()

if __name__ == '__main__':
    print ('Program is starting ... ')
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

Code Interpretation

```

import time
address = 0x48
bus=smbus.SMBus(1)
cmd=0x40
```

'address = 0x48' definiert die 'I2C' Adresse und das Steuerbyte von' PCF8591 'und instanziert dann den Objektbus von' SMBus '. cmd = 0x40 'ist ein Befehl für das' bus'-Objekt, der verwendet werden kann Betreiben Sie ADC und DAC des PCF8591 Moduls.

```

def analogRead(chn):
    value = bus.read_byte_data(address,cmd+chn)
    return value
```

Diese Unterfunktion wird zum Lesen von 'ADC' verwendet. Der Parameter "chn" repräsentiert die Eingangskanalnummer: 0,1,2,3. Der Rückgabewert ist der ADC Wert.

```
def analogWrite(value):
    bus.write_byte_data(address,cmd,value)
```

Diese Unterfunktion wird verwendet, um 'DAC' zu schreiben. Sein Parameter "Wert" repräsentiert die Qualität der zu schreibenden Zahl, die zwischen '0-255' liegt.

```
def loop():
    while True:
        value = analogRead(0)
        analogWrite(value)
        voltage = value / 255.0 * 3.3
        print ('ADC Value : %d, Voltage : %.2f%(value,voltage))
        time.sleep(0.01)
```

Lesen Sie im "while" Zyklus zuerst den ADC Wert von Kanal 0, schreiben Sie dann den Wert als digitale DAC-Qualität und geben Sie die entsprechende Spannung am Ausgangspin des PCF8591 aus. Berechnen Sie dann den entsprechenden Spannungswert und drucken Sie ihn aus.

```
def destroy():
    bus.close()
```

Lassen Sie das 'bus' Objekt los.

```
import smbus
```

Das Modul '[System Management Bus. This](#)' definiert einen Host, auf dem der Transaktionsobjekttyp '[SMBus](#)' den Linux-Kernel ausführen kann. Der Host-Kernel muss I2C-Unterstützung, I2C-Geräteschnittstellenunterstützung und einen Busadapertreiber haben. All dies kann in den Kernel eingebaut oder aus dem Modul geladen werden. In Python können Sie die Hilfe '[smbus](#)' verwenden, um verwandte Funktionen und deren Beschreibungen anzuzeigen. '[bus = smbus.SMBus\(1\)](#)': Erstellen Sie ein '[SMBus](#)' Klassenobjekt. '[bus.read_byte_data\(address, cmd + chn\)](#)': Liest ein Datenbyte von der Adresse und kehrt zurück. '[bus.write_byte_data\(address, cmd, value\)](#)': Schreiben Sie ein Datenbyte in eine Adresse.

Lektion 9 Potentiometer & LED

Überblick

In diesem Kurs lernen Sie, wie Sie die Helligkeit von LED über ein Potentiometer steuern.

Erforderliche Teile

1 x Raspberry Pi

1 x PCF8591 Modul

1 x 220 Ohm Widerstand

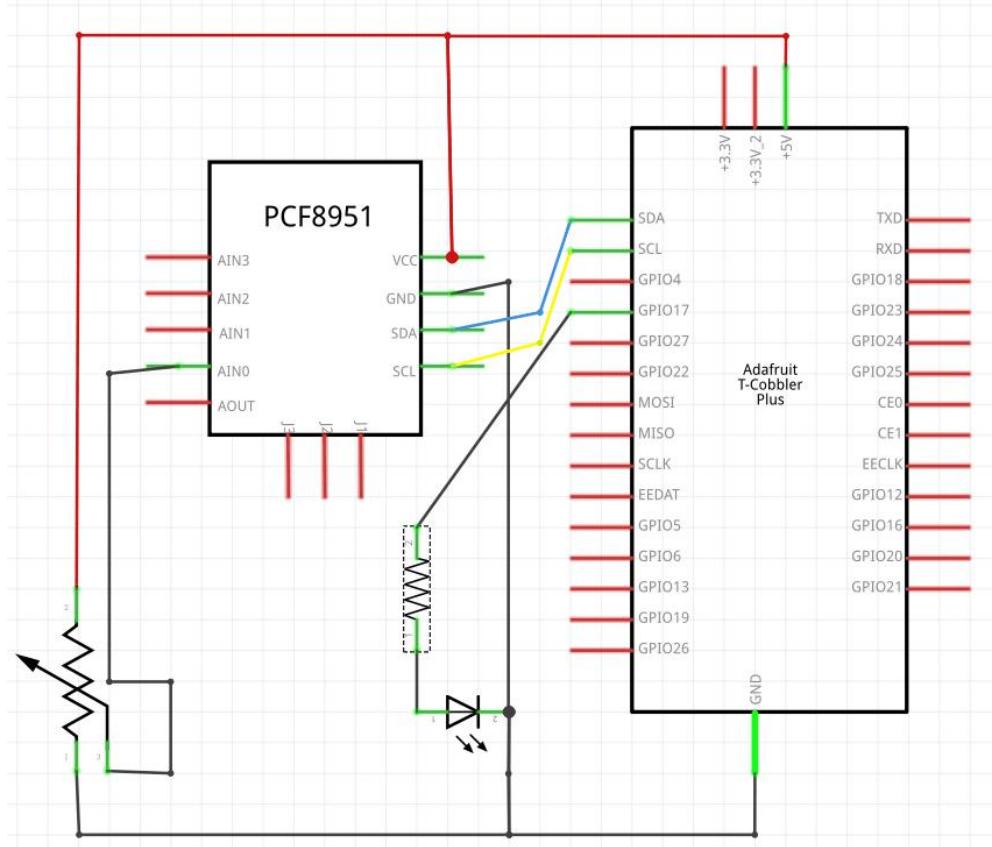
11 x Jumper Kabel

1 x Steckbrett

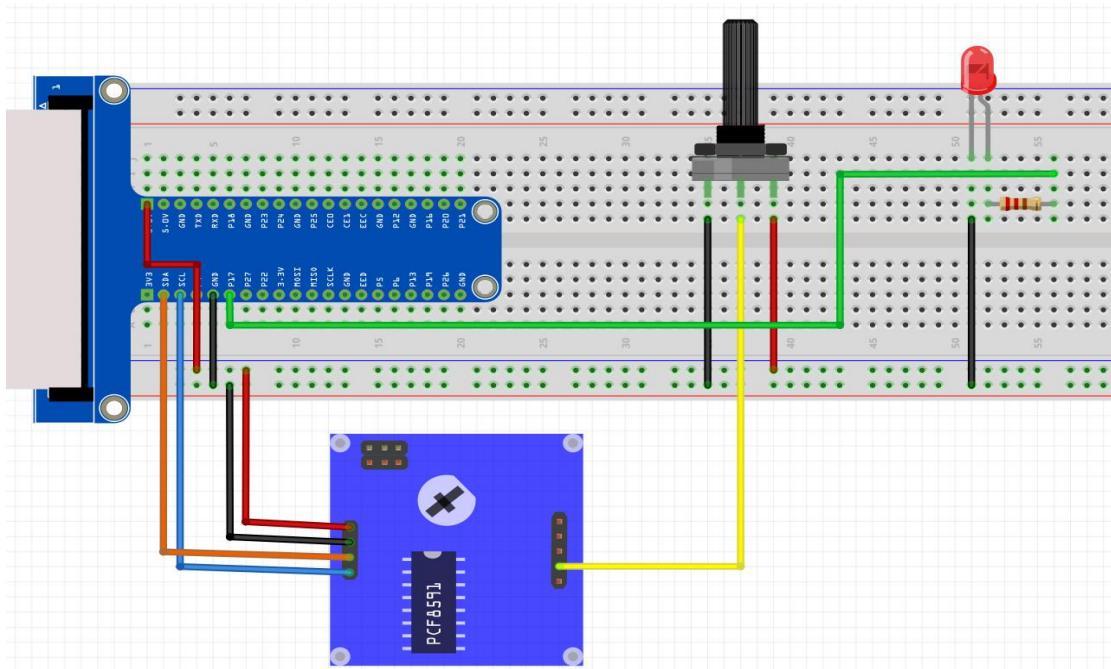
1 x Potentiometer

1 x LED

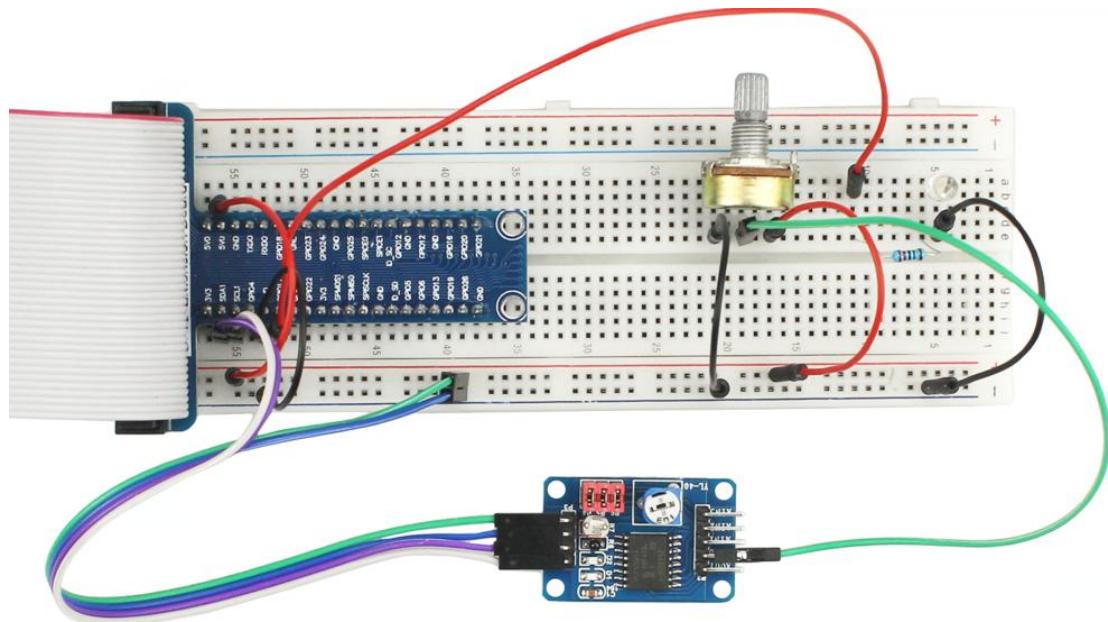
Schaltplan



Verdrahtung Diagramm



Beispielbild



C code

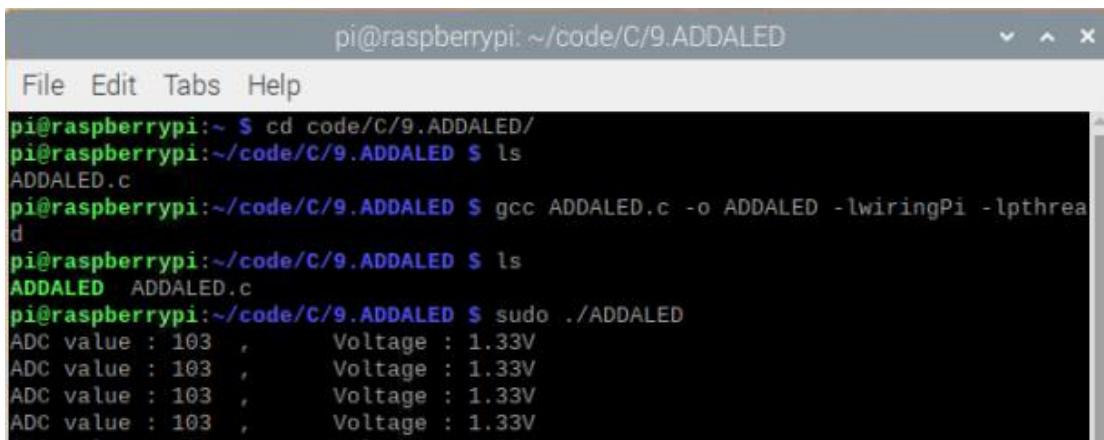
Öffnen Sie das Terminal und geben Sie den Befehl `cd code / C / 9.ADDALED` ein, um das Codeverzeichnis ADDALED.c aufzurufen.

Geben Sie den Befehl `ls` ein, um die Datei ADDALED.c im Verzeichnis anzuzeigen.

```
pi@raspberrypi:~/code/C/9.ADDALED
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/9.ADDALED
pi@raspberrypi:~/code/C/9.ADDALED $ ls
ADDALED.c
pi@raspberrypi:~/code/C/9.ADDALED $
```

Geben Sie den Befehl `gcc ADDALED.c -o ADDALED -lwiringPi -lpthread` ein, um die ausführbare Datei ADDALED.c ADDALED zu generieren. Geben Sie den Befehl `ls` zum Anzeigen ein.

Geben Sie den Befehl **sudo ./ADDALED** ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



The screenshot shows a terminal window titled "pi@raspberrypi: ~ / code / C / 9 . ADDALED". The window contains the following text:

```
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/9.ADDALED/
pi@raspberrypi:~/code/C/9.ADDALED$ ls
ADDALED.c
pi@raspberrypi:~/code/C/9.ADDALED$ gcc ADDALED.c -o ADDALED -lwiringPi -lpthread
pi@raspberrypi:~/code/C/9.ADDALED$ ls
ADDALED ADDALED.c
pi@raspberrypi:~/code/C/9.ADDALED$ sudo ./ADDALED
ADC value : 103 , Voltage : 1.33V
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>
#include <softPwm.h>

#define address 0x48
#define pinbase 64
#define A0 pinbase + 0
#define A1 pinbase + 1
#define A2 pinbase + 2
#define A3 pinbase + 3

#define ledPin 0
int main(void){
    int value;
    float voltage;
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    softPwmCreate(ledPin,0,100);
    pcf8591Setup(pinbase,address);
```

```
while(1){  
    value = analogRead(A0);  
    softPwmWrite(ledPin,value*100/255);  
    voltage = (float)value / 255.0 * 3.3;  
    printf("ADC value : %d \tVoltage : %.2fV\n",value,voltage);  
    delay(100);  
}  
return 0;  
}
```

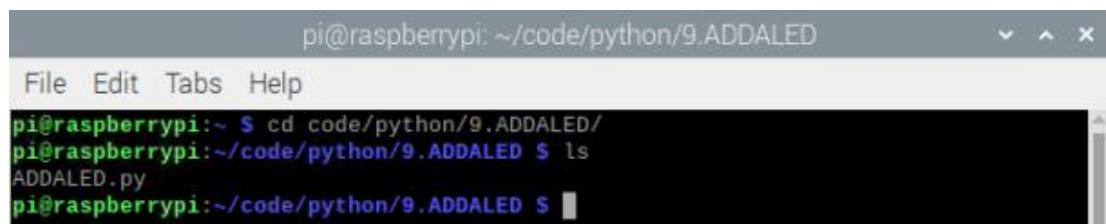
Code Interpretation

Lesen Sie im Code den ADC Wert des Potentiometers und ordnen Sie ihn dem Arbeitszyklus der PWM zu, um die Helligkeit der LED zu steuern. Eine ausführliche Erklärung finden Sie in den Lektionen 8 und 4.

Python code

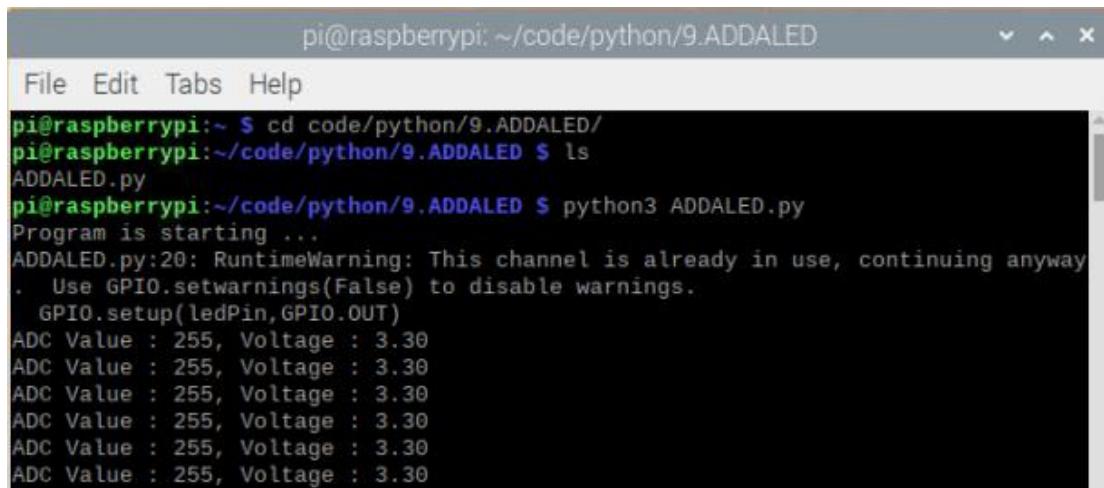
Öffnen Sie das Terminal und geben Sie mit dem Befehl `cd code / python / 9.ADDALED /` das Codeverzeichnis ein;

Geben Sie den Befehl `ls` ein, um die Datei ADDALED.py im Verzeichnis anzuzeigen.



A screenshot of a terminal window titled "pi@raspberrypi: ~ /code/python/9.ADDALED". The window has a menu bar with "File", "Edit", "Tabs", and "Help". The terminal prompt is "pi@raspberrypi:~ \$". The user enters the command "cd code/python/9.ADDALED/" followed by "ls". The output shows a single file named "ADDALED.py". The terminal prompt then changes to "pi@raspberrypi:~/code/python/9.ADDALED \$".

Geben Sie den Befehl `python3 ADDALED.py` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi: ~ /code/python/9.ADDALED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/9.ADDALED/
pi@raspberrypi:~/code/python/9.ADDALED $ ls
ADDALED.py
pi@raspberrypi:~/code/python/9.ADDALED $ python3 ADDALED.py
Program is starting ...
ADDALED.py:20: RuntimeWarning: This channel is already in use, continuing anyway
. Use GPIO.setwarnings(False) to disable warnings.
    GPIO.setup(ledPin,GPIO.OUT)
ADC Value : 255, Voltage : 3.30
```

Das Folgende ist der Code:

```
import RPi.GPIO as GPIO
import smbus
import time
address = 0x48
bus=smbus.SMBus(1)
cmd=0x40
ledPin = 11

def analogRead(chn):
    value = bus.read_byte_data(address,cmd+chn)
    return value
def analogWrite(value):
    bus.write_byte_data(address,cmd,value)
def setup():
    global p
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(ledPin,GPIO.OUT)
    GPIO.output(ledPin,GPIO.LOW)

    p = GPIO.PWM(ledPin,1000)
    p.start(0)
def loop():
    while True:
```

```

value = analogRead(0)
p.ChangeDutyCycle(value*100/255)
voltage = value / 255.0 * 3.3
print ('ADC Value : %d, Voltage : %.2f%(value,voltage))
time.sleep(0.01)

def destroy():
    bus.close()
    GPIO.cleanup()
if __name__ == '__main__':
    print ('Program is starting ... ')
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()

```

Code Interpretation

```

import RPi.GPIO as GPIO
import smbus
import time
address = 0x48
bus=smbus.SMBus(1)
cmd=0x40
ledPin = 11

```

'address = 0x48' definiert die 'I2C' Adresse und das Steuerbyte von' PCF8591 'und instanziiert dann den Objektbus von' SMBus '. cmd = 0x40 'ist ein Befehl für das Busobjekt, mit dem das betrieben werden kann ADC 'und' DAC '.

```

def analogRead(chn):
    value = bus.read_byte_data(address,cmd+chn)
    return value

```

Diese Unterfunktion wird zum Lesen von 'ADC' verwendet. Der Parameter "chn" repräsentiert die Eingangskanalnummer: 0,1,2,3.

Der Rückgabewert ist der ADC-Wert.

```

def analogWrite(value):

```

```
bus.write_byte_data(address,cmd,value)
```

Diese Unterfunktion wird verwendet, um "DAC" zu schreiben. Der Parameter "Wert" repräsentiert die zu schreibende digitale Qualität zwischen "0-255".

```
def setup():
    global p
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(ledPin,GPIO.OUT)
    GPIO.output(ledPin,GPIO.LOW)
    p = GPIO.PWM(ledPin,1000)
    p.start(0)
```

Definieren Sie den Modus des 'ledPin' Pins sowie die Periode und das Tastverhältnis der PWM

```
def loop():
    while True:
        value = analogRead(0)
        p.ChangeDutyCycle(value*100/255)
        voltage = value / 255.0 * 3.3
        print ('ADC Value : %d, Voltage : %.2f%(value,voltage))
        time.sleep(0.01)
```

Lesen Sie im "while" Zyklus zuerst den ADC-Wert von Kanal 0, schreiben Sie dann den Wert als digitale DAC-Qualität und geben Sie die entsprechende Spannung am Ausgangspin von PCF8591 aus. Berechnen Sie dann den entsprechenden PWM-Arbeitszyklus, um die LED Helligkeit und den Spannungswert zu steuern, und drucken Sie ihn aus.

Lektion 10 Potentiometer & RGBLED

Überblick

In dieser Lektion lernen wir, wie Sie mit 3 Potentiometern die Helligkeit von 3 LEDs von RGBLED steuern, damit sie unterschiedliche Farben anzeigen.

Erforderliche Teile

1 x Raspberry Pi

1 x PCF8591 Modul

3 x 220 Ohm Widerstand

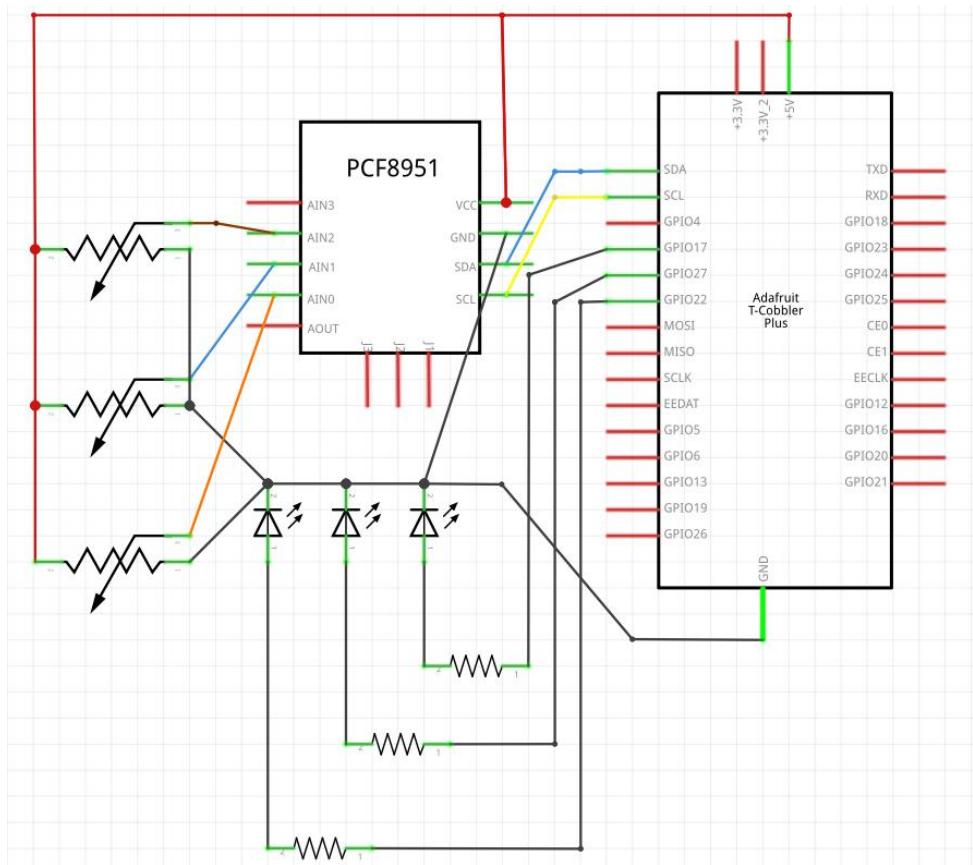
21 x Jumper Kabel

1 x Steckbrett

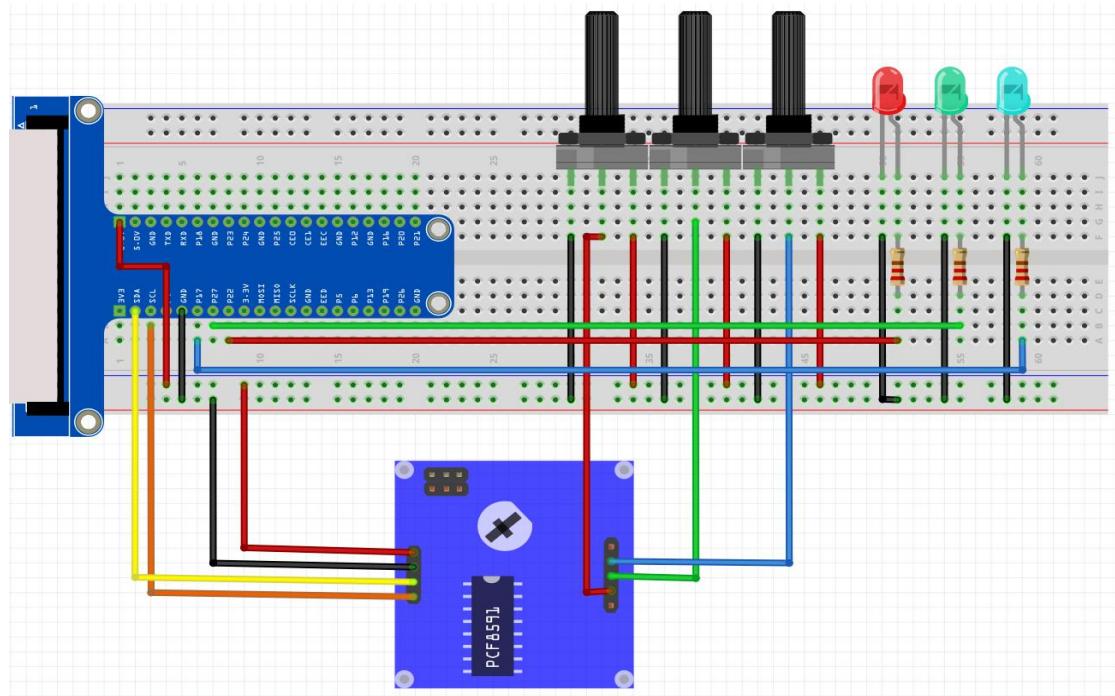
1 x RGBLED

3 x Potentiometer

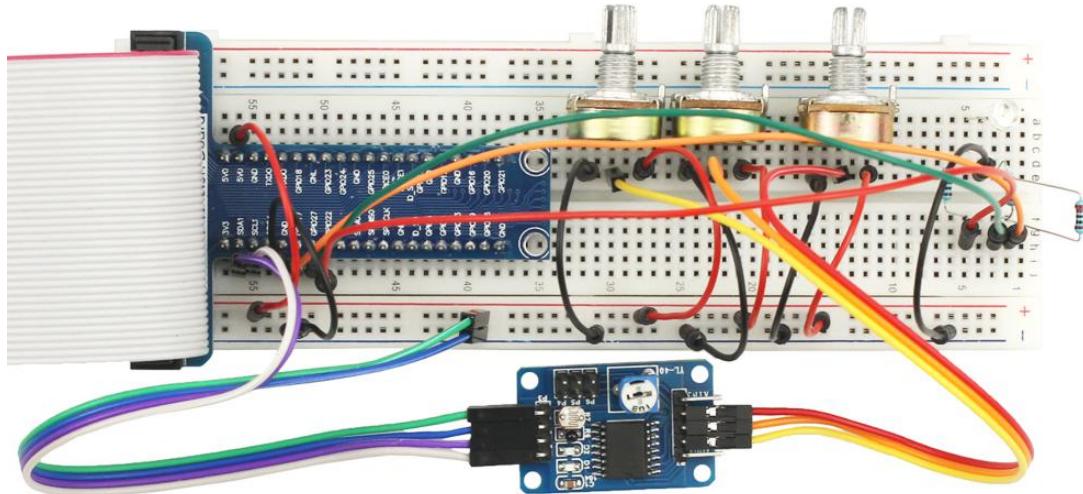
Schaltplan



Verdrahtung Diagramm



Beispielbild



C code

Öffnen Sie das Terminal und geben Sie den Befehl `cd code / C / 10.ADDARGBLED` / ein, um das Codeverzeichnis ADDARGB.c aufzurufen.

Geben Sie den Befehl `ls` ein, um die Datei ADDARGB.c im Verzeichnis anzuzeigen.



```
pi@raspberrypi: ~ / code / C / 10.ADDARGBLED
File Edit Tabs Help
pi@raspberrypi: ~ $ cd code/C/10.ADDARGBLED/
pi@raspberrypi: ~/code/C/10.ADDARGBLED $ ls
ADDARGB.c
pi@raspberrypi: ~/code/C/10.ADDARGBLED $
```

Geben Sie den Befehl `gcc ADDARGB.c -o ADDARGB -lwiringPi -lpthread` ein, um die ausführbare Datei ADDARGB.c ADDARGB.c zu generieren.

Geben Sie den Befehl `ls` zum Anzeigen ein.

```
pi@raspberrypi:~/code/C/10.ADDARGBLED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/10.ADDARGBLED/
pi@raspberrypi:~/code/C/10.ADDARGBLED $ ls
ADDRGB.c
pi@raspberrypi:~/code/C/10.ADDARGBLED $ gcc ADDRGB.c -o ADDRGB -lwiringPi -lpthread
pi@raspberrypi:~/code/C/10.ADDARGBLED $ ls
ADDRGB ADDRGB.c
pi@raspberrypi:~/code/C/10.ADDARGBLED $ sudo ./ADDRGB
ADC value val_Red: 111 , val_Green: 176 , val_Blue: 255
ADC value val_Red: 111 , val_Green: 176 , val_Blue: 255
ADC value val_Red: 111 , val_Green: 176 , val_Blue: 255
ADC value val_Red: 111 , val_Green: 176 , val_Blue: 255
ADC value val_Red: 111 , val_Green: 176 , val_Blue: 255
ADC value val_Red: 111 , val_Green: 176 , val_Blue: 255
```

Geben Sie den Befehl [sudo ./ADDRGB](#) ein, um den Code auszuführen. Das Ergebnis ist wie folgt:

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>
#include <softPwm.h>

#define address 0x48
#define pinbase 64
#define A0 pinbase + 0
#define A1 pinbase + 1
#define A2 pinbase + 2
#define A3 pinbase + 3

#define ledRedPin 3
#define ledGreenPin 2
#define ledBluePin 0
int main(void){

    int val_Red,val_Green,val_Blue;
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
```

```

    }
    softPwmCreate(ledRedPin,0,100);
    softPwmCreate(ledGreenPin,0,100);
    softPwmCreate(ledBluePin,0,100);
    pcf8591Setup(pinbase,address);

    while(1){
        val_Red = analogRead(A0);
        val_Green = analogRead(A1);
        val_Blue = analogRead(A2);
        softPwmWrite(ledRedPin,val_Red*100/255);
        softPwmWrite(ledGreenPin,val_Green*100/255);
        softPwmWrite(ledBluePin,val_Blue*100/255);
        printf("ADC value val_Red: %d \tval_Green: %d \tval_Blue: %d
\n",val_Red,val_Green,val_Blue);
        delay(100);
    }
    return 0;
}

```

Code Interpretation

Im Code werden die ADC Werte der 3 Potentiometer gelesen und auf den PWM Arbeitszyklus abgebildet, um die 3 LEDs mit RGB LEDs unterschiedlicher Farbe zu steuern. Eine ausführliche Interpretation finden Sie in den Lektionen 8 und 5.

Python code

Öffnen Sie das Terminal und geben Sie mit dem Befehl **cd code / python / 10.ADDARGBLED** / das Codeverzeichnis ein.

Geben Sie den Befehl **ls** ein, um die Datei **ADDARGB.py** im Verzeichnis anzuzeigen.



```

pi@raspberrypi: ~code/python/10.ADDARGBLED
File Edit Tabs Help
pi@raspberrypi: ~$ cd code/python/10.ADDARGBLED/
pi@raspberrypi: ~/code/python/10.ADDARGBLED $ ls
ADDARGB.py
pi@raspberrypi: ~/code/python/10.ADDARGBLED $ 

```

Geben Sie den Befehl `python3 ADDARGB.py` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:

```
pi@raspberrypi: ~/code/python/10.ADDARGBLED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/10.ADDARGBLED/
pi@raspberrypi:~/code/python/10.ADDARGBLED $ ls
ADDARGB.py
pi@raspberrypi:~/code/python/10.ADDARGBLED $ python3 ADDARGB.py
Program is starting ...
ADDARGB.py:26: RuntimeWarning: This channel is already in use, continuing anyway
. Use GPIO.setwarnings(False) to disable warnings.
    GPIO.setup(ledRedPin,GPIO.OUT)
ADDARGB.py:27: RuntimeWarning: This channel is already in use, continuing anyway
. Use GPIO.setwarnings(False) to disable warnings.
    GPIO.setup(ledGreenPin,GPIO.OUT)
ADDARGB.py:28: RuntimeWarning: This channel is already in use, continuing anyway
. Use GPIO.setwarnings(False) to disable warnings.
    GPIO.setup(ledBluePin,GPIO.OUT)
ADC Value value_Red: 113 ,      vluue_Green: 177 ,      value_Blue: 255
ADC Value value_Red: 113 ,      vluue_Green: 177 ,      value_Blue: 255
ADC Value value_Red: 113 ,      vluue_Green: 177 ,      value_Blue: 255
ADC Value value_Red: 113 ,      vluue_Green: 177 ,      value_Blue: 255
ADC Value value_Red: 112 ,      vluue_Green: 176 ,      value_Blue: 255
ADC Value value_Red: 113 ,      vluue_Green: 175 ,      value_Blue: 255
```

Das Folgende ist der Code:

```
import RPi.GPIO as GPIO
import smbus
import time

address = 0x48
bus=smbus.SMBus(1)
cmd=0x40

ledRedPin = 15
ledGreenPin = 13
ledBluePin = 11

def analogRead(chn):
    bus.write_byte(address,cmd+chn)
    value = bus.read_byte(address)
    value = bus.read_byte(address)
    return value

def analogWrite(value):
```

```
bus.write_byte_data(address,cmd,value)

def setup():
    global p_Red,p_Green,p_Blue
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(ledRedPin,GPIO.OUT)
    GPIO.setup(ledGreenPin,GPIO.OUT)
    GPIO.setup(ledBluePin,GPIO.OUT)

    p_Red = GPIO.PWM(ledRedPin,1000)
    p_Red.start(0)
    p_Green = GPIO.PWM(ledGreenPin,1000)
    p_Green.start(0)
    p_Blue = GPIO.PWM(ledBluePin,1000)
    p_Blue.start(0)

def loop():
    while True:
        value_Red = analogRead(0)
        value_Green = analogRead(1)
        value_Blue = analogRead(2)
        p_Red.ChangeDutyCycle(value_Red*100/255)
        p_Green.ChangeDutyCycle(value_Green*100/255)
        p_Blue.ChangeDutyCycle(value_Blue*100/255)
        #print read ADC value
        print ('ADC Value
value_Red: %d ,\tvluue_Green: %d ,\tvalue_Blue: %d%(value_Red,value_Green,value_Blue))
        time.sleep(0.01)

def destroy():
    bus.close()
    GPIO.cleanup()

if __name__ == '__main__':
    print ('Program is starting ... ')
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

Code Interpretation

```
def analogRead(chn):
    bus.write_byte(address,cmd+chn)
    value = bus.read_byte(address)
    value = bus.read_byte(address)
    return value
```

Diese Funktion gibt immer die zuletzt gelesenen Daten zurück. Wenn Sie die aktuellen Daten zurückgeben möchten, müssen Sie dies zweimal tun.

```
def loop():
    while True:
        value_Red = analogRead(0)
        value_Green = analogRead(1)
        value_Blue = analogRead(2)
        p_Red.ChangeDutyCycle(value_Red*100/255)
        p_Green.ChangeDutyCycle(value_Green*100/255)
        p_Blue.ChangeDutyCycle(value_Blue*100/255)
        Print('ADC Value
value_Red: %d ,\tvalue_Green: %d ,\tvalue_Blue: %d'%(value_Red,value_Green,value_Blue))
        time.sleep(0.01)
```

Diese Funktion liest die ADC Werte der 3 Potentiometer und ordnet sie dem PWM-Arbeitszyklus zu, um die 3 LED mit jeweils unterschiedlicher RGBLED Farbe zu steuern.

Lektion 11 Fotowiderstand & LED

Überblick

In dieser Lektion lernen Sie, wie Sie einen Fotowiderstand verwenden. Der Fotowiderstand reagiert sehr empfindlich auf die Beleuchtungsstärke. Mit dieser Funktion können wir eine Nachtlampe herstellen. Wenn das Umgebungslicht dunkler wird, wird die LED automatisch heller, und wenn das Umgebungslicht heller wird, wird die LED automatisch dunkler.

Erforderliche Teile

1 x Raspberry Pi

1 x PCF8591 Modul

1 x 220 Ohm Widerstand

1 x 10k Widerstand

9 x Jumper Kabel

1 x Steckbrett

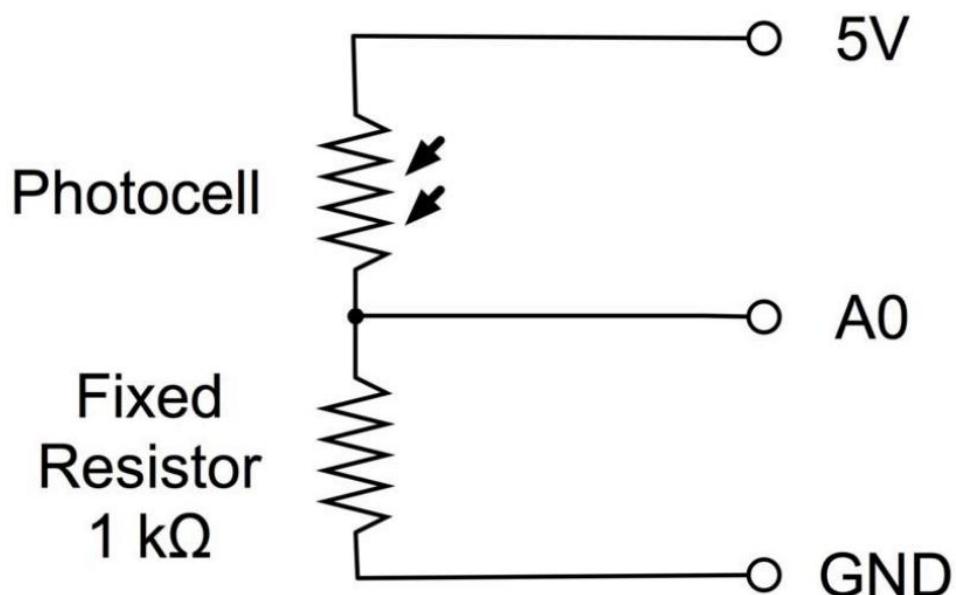
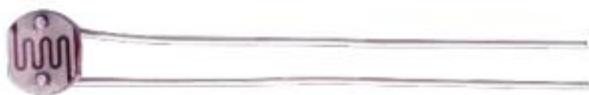
1 x LED

1 x Photoresistor

Produkt Einführung

Lichtempfindlicher Widerstand

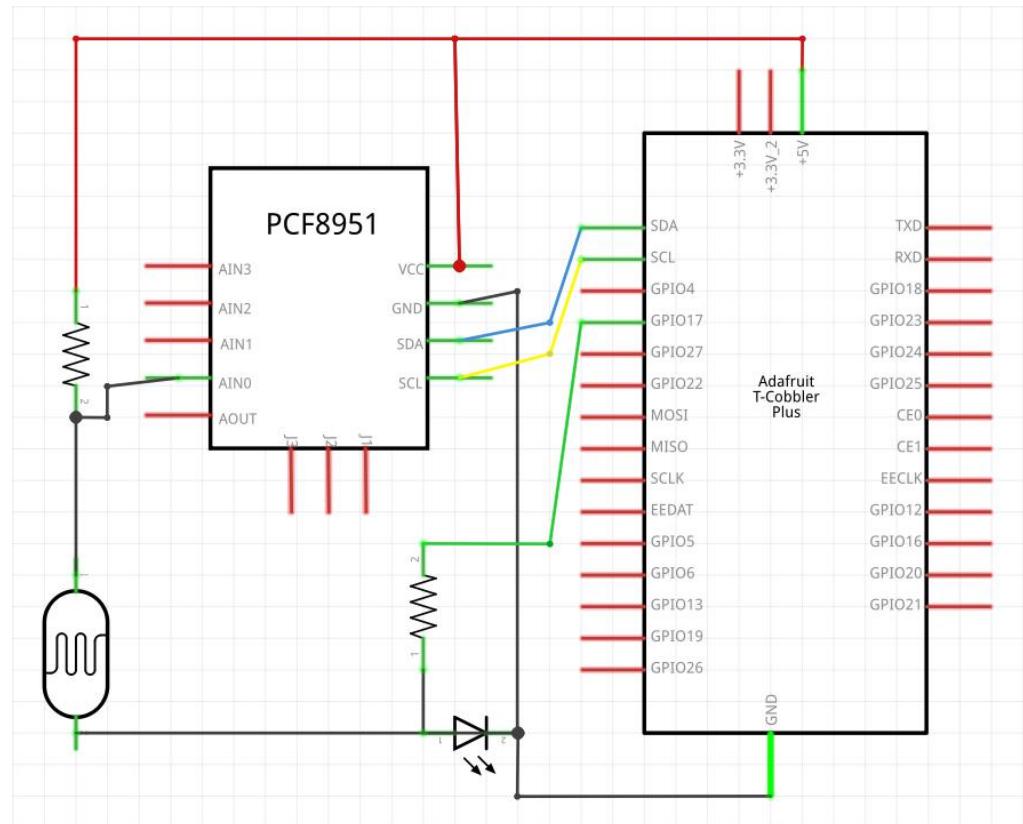
Lichtempfindliche Widerstände, auch als LDRs bekannt. Fotowiderstände sind wie gewöhnliche Widerstände, mit der Ausnahme, dass sich der Widerstandswert ändert, wenn das Licht auf sie fällt. Dies hat einen Widerstand von etwa $50\text{ k}\Omega$ bei Dunkelheit und $50\text{k}\Omega$ bei starkem Licht. Um diesen geänderten Widerstandswert in einen Wert umzuwandeln, den wir am Analogeingang der UNO R3 Board messen können; es muss in Spannung umgewandelt werden. Am einfachsten ist es, ihn mit einem festen Widerstand zu kombinieren.



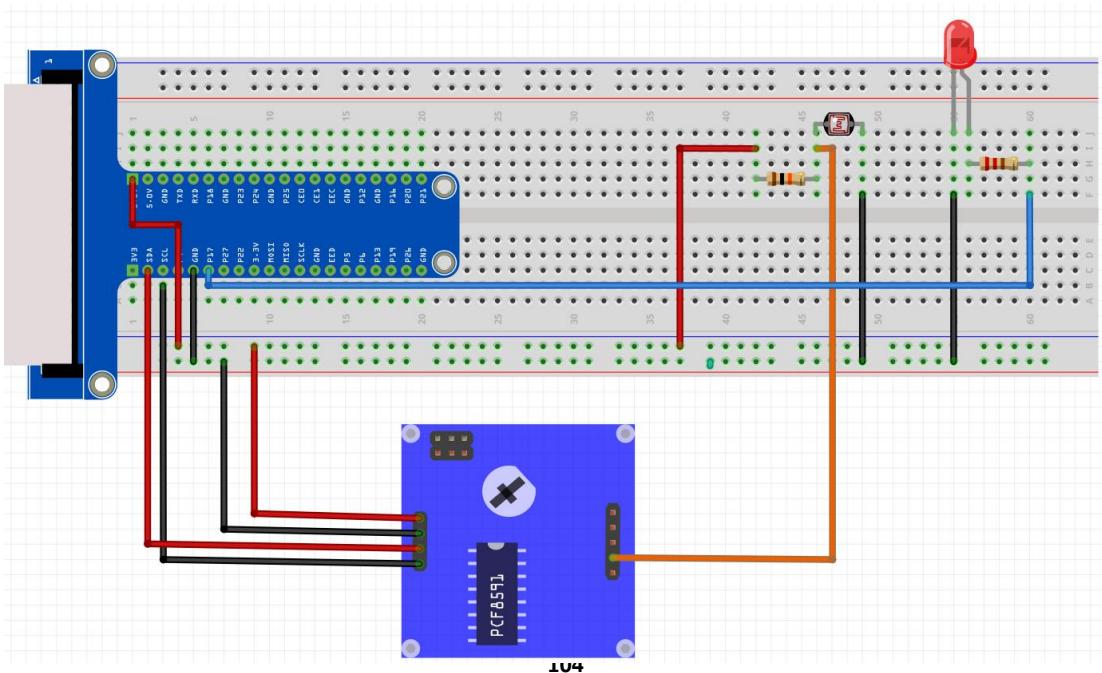
Die Werte des Widerstands und des Fotowiderstands werden so zusammengesetzt, dass sie sich wie einzelne Daten verhalten. Wenn das Licht sehr hell ist, ist der Widerstand des Fotowiderstandes im Vergleich zu einem Widerstand mit festem Wert sehr niedrig, so als ob das Potentiometer auf Maximum eingestellt wäre. Wenn der Fotowiderstand schwach beleuchtet ist, wird der Widerstand größer als ein fester 1 k Ω -Widerstand, als ob das Potentiometer auf GND zeigt. Laden Sie den in diesem Abschnitt angegebenen Code, bedecken Sie den lichtempfindlichen Widerstand mit

Ihrem Finger und platzieren Sie ihn in der Nähe der Lichtquelle. Sie können die LED-Änderungen sehen.

Schaltplan

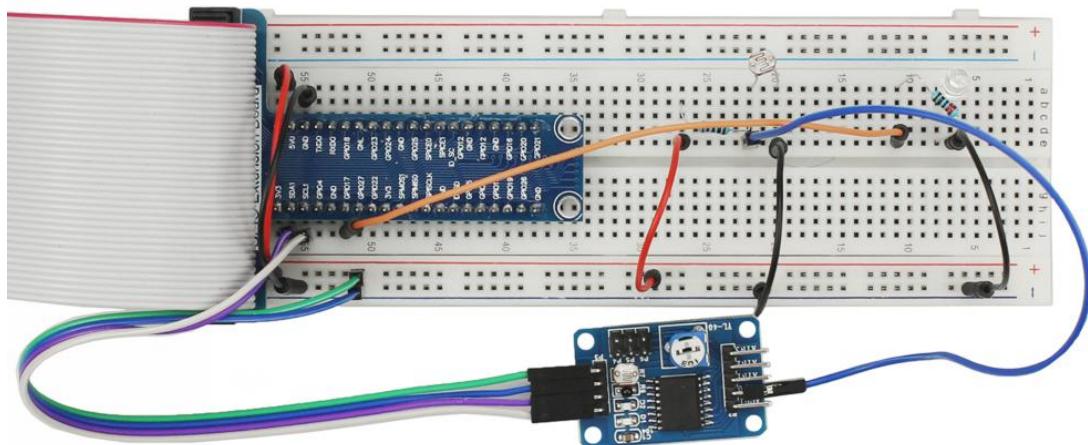


Verdrahtung Diagramm





Beispiel Abbildung



C code

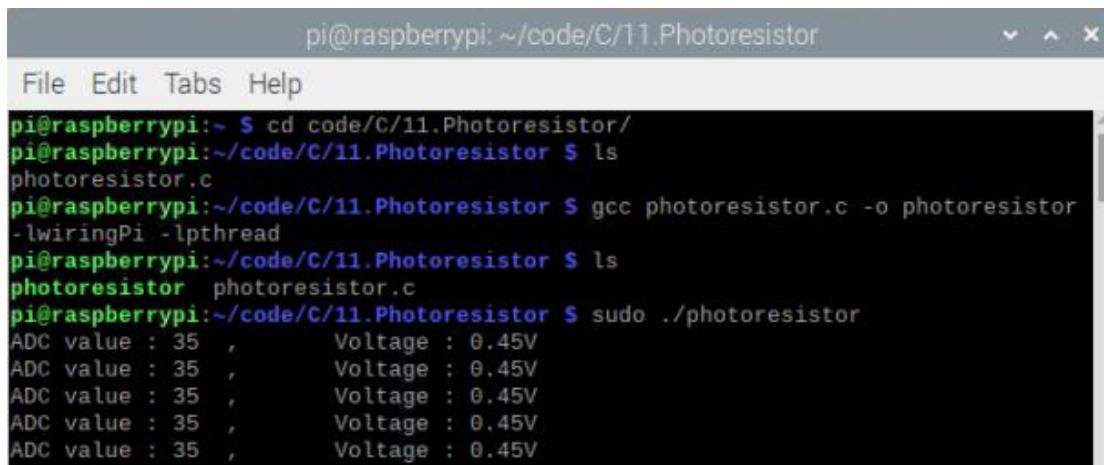
Öffnen Sie das Terminal und geben Sie den Befehl `cd code / C / 11.Photoresistor /` ein, um das Codeverzeichnis photoresistor.c aufzurufen.

Geben Sie den Befehl `ls` ein, um die Datei photoresistor.c im Verzeichnis anzuzeigen.

```
pi@raspberrypi: ~/code/C/11.Photoresistor
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/11.Photoresistor/
pi@raspberrypi:~/code/C/11.Photoresistor $ ls
photoresistor.c
pi@raspberrypi:~/code/C/11.Photoresistor $
```

Geben Sie `gcc photoresistor.c -o photoresistor -lwiringPi -lpthread` ein, um die ausführbare Datei photoresistor zu erzeugen.

Geben Sie den Befehl `sudo ./photoresistor` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi: ~/code/C/11.Photoresistor
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/11.Photoresistor/
pi@raspberrypi:~/code/C/11.Photoresistor $ ls
photoresistor.c
pi@raspberrypi:~/code/C/11.Photoresistor $ gcc photoresistor.c -o photoresistor
-lwiringPi -lpthread
pi@raspberrypi:~/code/C/11.Photoresistor $ ls
photoresistor photoresistor.c
pi@raspberrypi:~/code/C/11.Photoresistor $ sudo ./photoresistor
ADC value : 35 , Voltage : 0.45V
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>
#include <softPwm.h>

#define address 0x48
#define pinbase 64
#define A0 pinbase + 0
#define A1 pinbase + 1
#define A2 pinbase + 2
#define A3 pinbase + 3

#define ledPin 0
int main(void){
    int value;
    float voltage;
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    softPwmCreate(ledPin,0,100);
    pcf8591Setup(pinbase,address);
```

```
while(1){  
    value = analogRead(A0);  
    softPwmWrite(ledPin,value*100/255);  
    voltage = (float)value / 255.0 * 3.3;  
    printf("ADC value : %d \tVoltage : %.2fV\n",value,voltage);  
    delay(100);  
}  
return 0;  
}
```

Code Interpretation

Lesen Sie im Code den ADC Wert des Fotowiderstands und ordnen Sie ihn dem PWM Arbeitszyklus zu, um die Helligkeit der LED zu steuern. Wenn der Fotowiderstand abgedeckt ist oder die Taschenlampe in Richtung Fotowiderstand leuchtet, nimmt die Helligkeit der LED zu oder ab. Eine ausführliche Erklärung finden Sie in den Lektionen 8 und 4.

Python code

Öffnen Sie das Terminal und geben Sie mit dem Befehl `cd code / python / 11.Photoresistor /` das Codeverzeichnis ein;

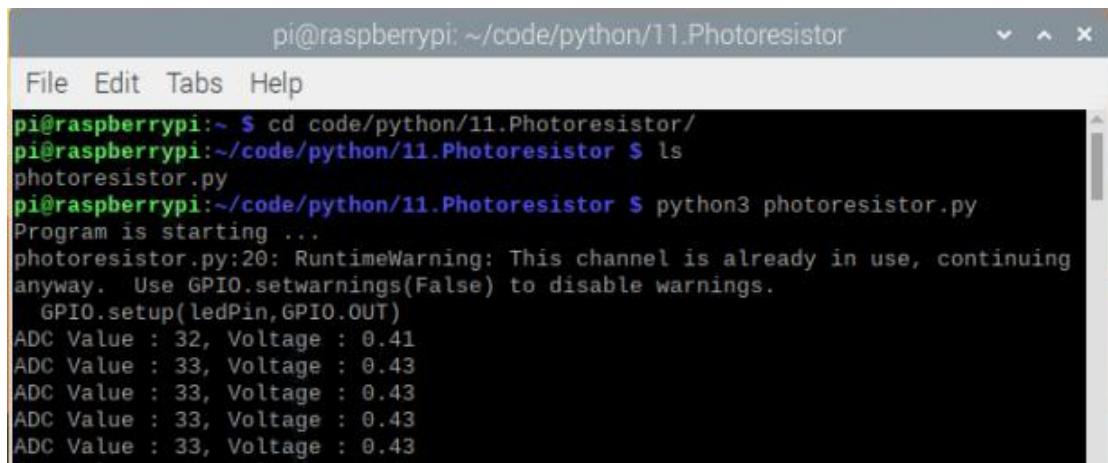
Geben Sie den Befehl `ls` ein, um die Datei photoresistor.py im Verzeichnis anzuzeigen.



The screenshot shows a terminal window titled "pi@raspberrypi: ~ /code/python/11.Photoresistor". The window has a standard Linux terminal interface with a dark background and light-colored text. At the top, there's a menu bar with "File", "Edit", "Tabs", and "Help". Below the menu is a command-line interface where the user has run the following commands:

```
pi@raspberrypi:~ $ cd code/python/11.Photoresistor/  
pi@raspberrypi:~/code/python/11.Photoresistor $ ls  
photoresistor.py  
pi@raspberrypi:~/code/python/11.Photoresistor $
```

Geben Sie den Befehl `python3 photoresistor.py` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi: ~ /code/python/11.Photoresistor
File Edit Tabs Help
pi@raspberrypi: ~ $ cd code/python/11.Photoresistor/
pi@raspberrypi: ~/code/python/11.Photoresistor $ ls
photoresistor.py
pi@raspberrypi: ~/code/python/11.Photoresistor $ python3 photoresistor.py
Program is starting ...
photoresistor.py:20: RuntimeWarning: This channel is already in use, continuing
anyway. Use GPIO.setwarnings(False) to disable warnings.
    GPIO.setup(ledPin,GPIO.OUT)
ADC Value : 32, Voltage : 0.41
ADC Value : 33, Voltage : 0.43
```

Das Folgende ist der Code:

```
import RPi.GPIO as GPIO
import smbus
import time
address = 0x48
bus=smbus.SMBus(1)
cmd=0x40
ledPin = 11
def analogRead(chn):
    value = bus.read_byte_data(address,cmd+chn)
    return value

def analogWrite(value):
    bus.write_byte_data(address,cmd,value)
def setup():
    global p
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(ledPin,GPIO.OUT)
    GPIO.output(ledPin,GPIO.LOW)
    p = GPIO.PWM(ledPin,1000)
    p.start(0)
def loop():
    while True:
        value = analogRead(0)
        p.ChangeDutyCycle(value*100/255)
```

```
voltage = value / 255.0 * 3.3
print ('ADC Value : %d, Voltage : %.2f%(value,voltage))
time.sleep(0.01)

def destroy():
    bus.close()
    GPIO.cleanup()

if __name__ == '__main__':
    print ('Program is starting ... ')
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

Code Interpretation

Dieses Programm entspricht der Potentiometer und LED Lektion. Eine ausführliche Erklärung finden Sie im Kurs Potentiometer und LED.

Lektion 12 Thermistor

Überblick

In dieser Lektion lernen Sie, wie Sie die Temperatur mit einem Raspberry Pi gesteuerten Thermistor ablesen.

Erforderliche Teile

1 x Raspberry Pi

1 x PCF8591 Modul

1 x 10K Widerstand

9 x Jumper Kabel

1 x Steckbrett

1 x Thermistor

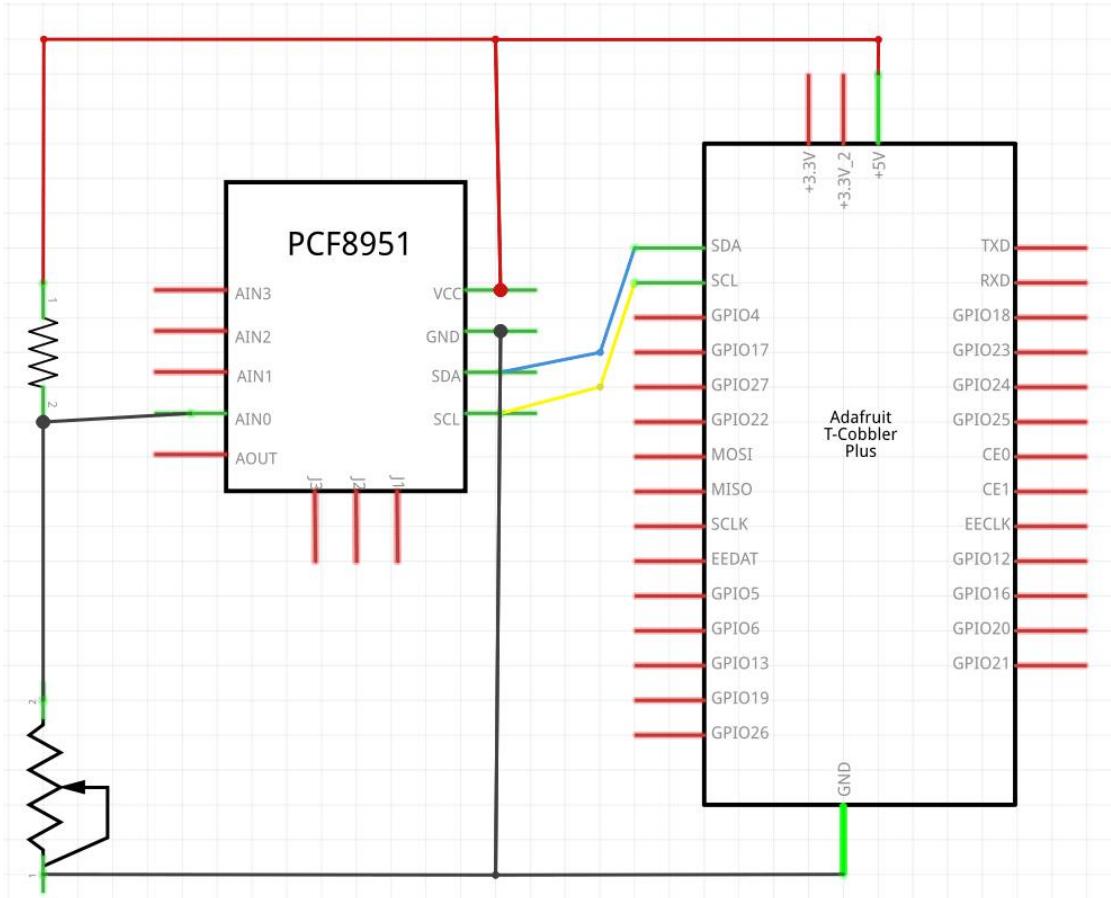
Produkt Einführung

Thermistor

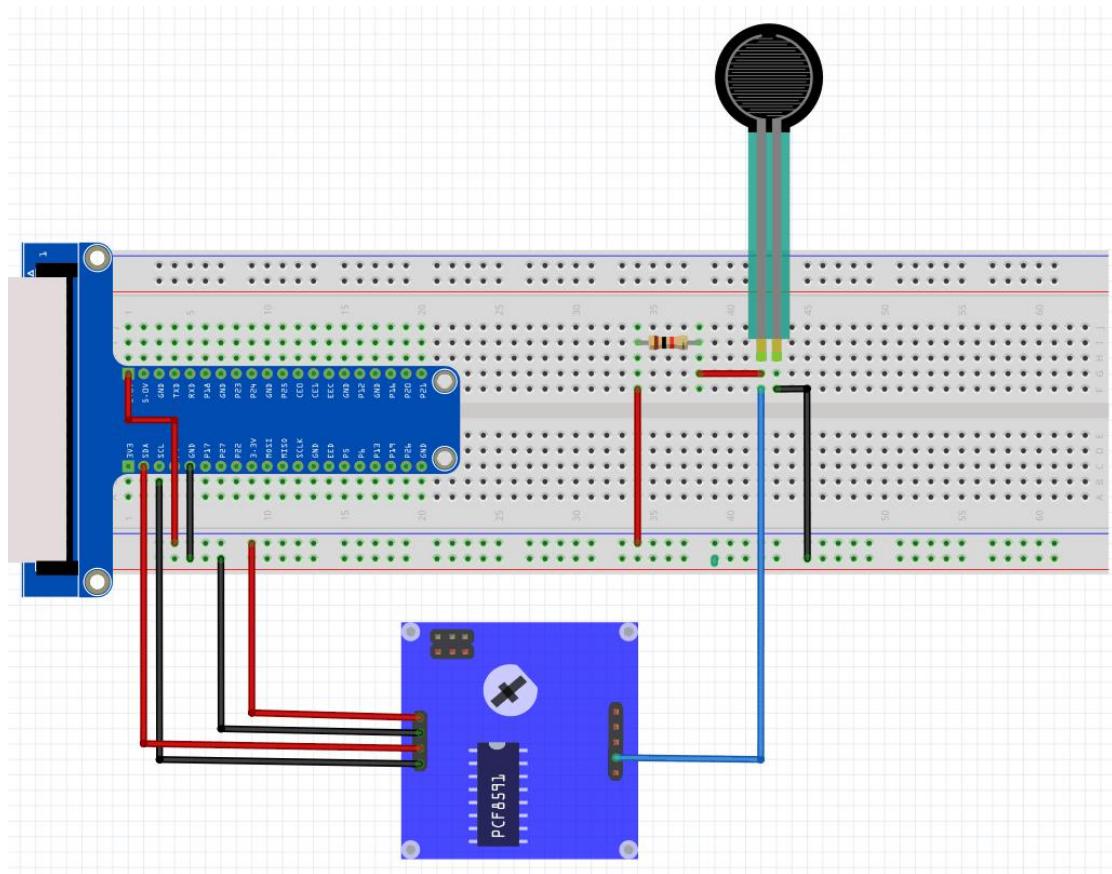
Thermistoren sind eine Art empfindlicher Komponenten. Sie sind nach unterschiedlichen Temperaturkoeffizienten in Thermistoren mit positivem Temperaturkoeffizienten (PTC) und Thermistoren mit negativem Temperaturkoeffizienten (NTC) unterteilt. Das typische Merkmal eines Thermistors ist, dass er temperaturempfindlich ist und bei verschiedenen Temperaturen unterschiedliche Widerstandswerte aufweist. Thermistoren mit positivem Temperaturkoeffizienten (PTC) haben bei höheren Temperaturen höhere Widerstandswerte, und Thermistoren mit negativem Temperaturkoeffizienten (NTC) haben bei höheren Temperaturen niedrigere Widerstandswerte. Sie gehören beide zu Halbleiterbauelementen.



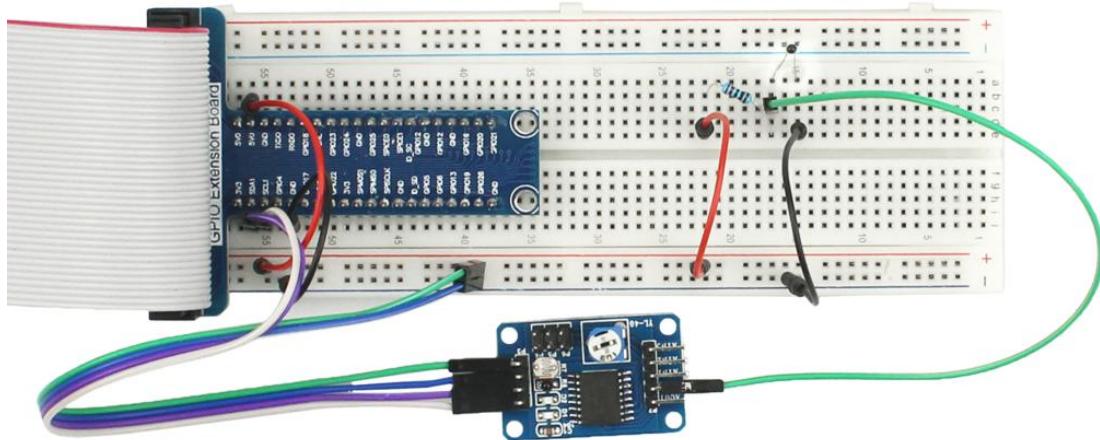
Schaltplan



Verdrahtung Diagramm



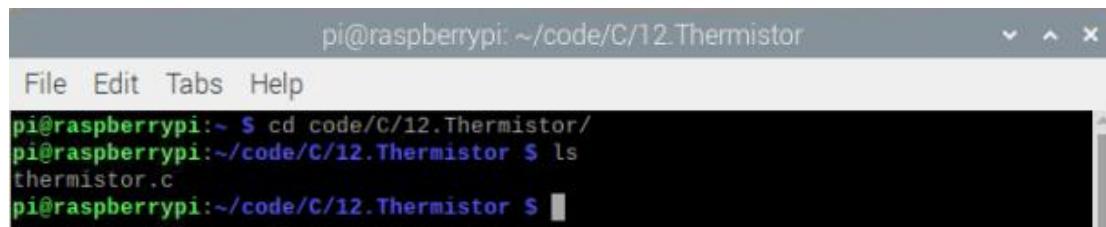
Beispiel Abbildung



C code

Öffnen Sie das Terminal und geben Sie den Befehl `cd code / C / 12.Thermistor /` ein, um das Verzeichnis `thermistor.c` aufzurufen;

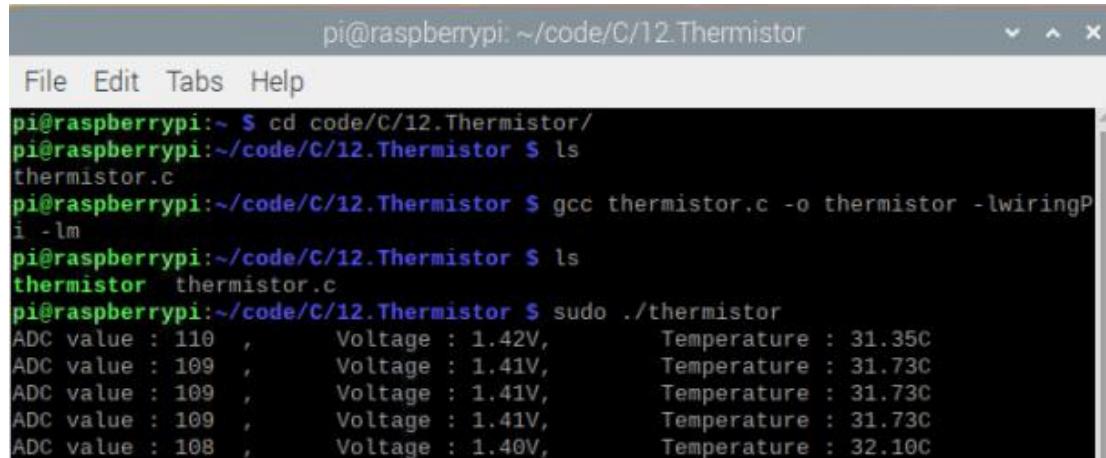
Geben Sie den Befehl `ls` ein, um die Datei `thermistor.c` im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/C/12.Thermistor
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/12.Thermistor/
pi@raspberrypi:~/code/C/12.Thermistor $ ls
thermistor.c
pi@raspberrypi:~/code/C/12.Thermistor $
```

Geben Sie den Befehl `gcc thermistor.c -o thermistor -lwiringPi -lm` ein, um die ausführbare Datei `thermostor` `thermostor.c` zu generieren, und geben Sie den Befehl `ls` ein, um sie anzuzeigen.

Geben Sie den Befehl `sudo ./thermistor` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi:~/code/C/12.Thermistor
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/12.Thermistor/
pi@raspberrypi:~/code/C/12.Thermistor $ ls
thermistor.c
pi@raspberrypi:~/code/C/12.Thermistor $ gcc thermistor.c -o thermistor -lwiringPi -lm
pi@raspberrypi:~/code/C/12.Thermistor $ ls
thermistor  thermistor.c
pi@raspberrypi:~/code/C/12.Thermistor $ sudo ./thermistor
ADC value : 110 ,      Voltage : 1.42V,      Temperature : 31.35C
ADC value : 109 ,      Voltage : 1.41V,      Temperature : 31.73C
ADC value : 109 ,      Voltage : 1.41V,      Temperature : 31.73C
ADC value : 109 ,      Voltage : 1.41V,      Temperature : 31.73C
ADC value : 108 ,      Voltage : 1.40V,      Temperature : 32.10C
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>
#include <math.h>

#define address 0x48
```

```

#define pinbase 64
#define A0 pinbase + 0
#define A1 pinbase + 1
#define A2 pinbase + 2
#define A3 pinbase + 3

int main(void){
    int adcValue;
    float tempK,tempC;
    float voltage,Rt;
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    pcf8591Setup(pinbase,address);
    while(1){
        adcValue = analogRead(A0);
        voltage = (float)adcValue / 255.0 * 3.3;
        Rt = 10 * voltage / (3.3 - voltage);
        tempK = 1/(1/(273.15 + 25) + log(Rt/10)/3950.0);
        tempC = tempK -273.15;
        printf("ADC value : %d ,\tVoltage : %.2fV,
\tTemperature : %.2fC\n",adcValue,voltage,tempC);
        delay(100);
    }
    return 0;
}

```

Code Interpretation

```

while(1){
    adcValue = analogRead(A0);
    voltage = (float)adcValue / 255.0 * 3.3;
    Rt = 10 * voltage / (3.3 - voltage);
    tempK = 1/(1/(273.15 + 25) + log(Rt/10)/3950.0);
    tempC = tempK -273.15;
    printf("ADC value : %d ,\tVoltage : %.2fV,
\tTemperature : %.2fC\n",adcValue,voltage,tempC);
    delay(100);
}

```

Lesen Sie im Code den ADC Wert des PCF8591 A0 Anschlusses und berechnen Sie dann die Spannung und den Widerstand.

Thermistor nach Ohmschem Gesetz. Berechnen Sie abschließend die aktuelle Temperatur. Nach der vorherigen Formel.

Python code

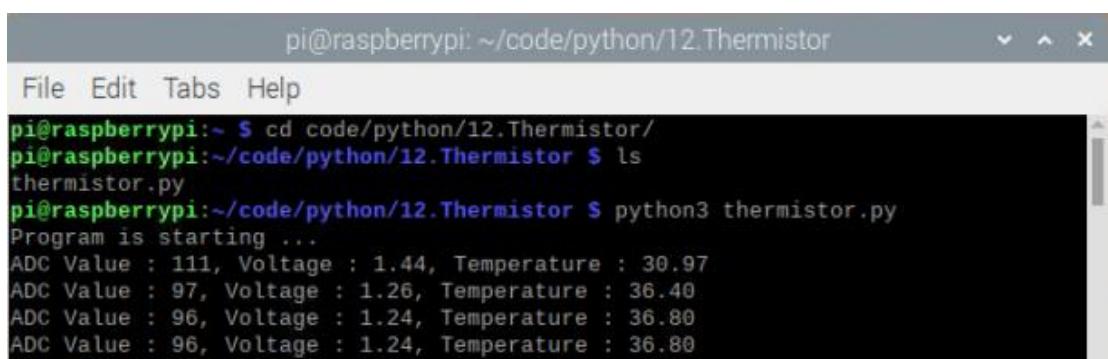
Öffnen Sie das Terminal und geben Sie mit dem Befehl `cd code / python / 12.Thermistor /` das Codeverzeichnis ein;

Geben Sie den Befehl `ls` ein, um die Datei `thermistoe.py` im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/python/12.Thermistor
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/12.Thermistor/
pi@raspberrypi:~/code/python/12.Thermistor $ ls
thermistor.py
pi@raspberrypi:~/code/python/12.Thermistor $
```

Geben Sie den Befehl `python3 thermistoe.py` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi:~/code/python/12.Thermistor
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/12.Thermistor/
pi@raspberrypi:~/code/python/12.Thermistor $ ls
thermistor.py
pi@raspberrypi:~/code/python/12.Thermistor $ python3 thermistor.py
Program is starting ...
ADC Value : 111, Voltage : 1.44, Temperature : 30.97
ADC Value : 97, Voltage : 1.26, Temperature : 36.40
ADC Value : 96, Voltage : 1.24, Temperature : 36.80
ADC Value : 96, Voltage : 1.24, Temperature : 36.80
```

Das Folgende ist der Code:

```
import RPi.GPIO as GPIO
import smbus
import time
import math
address = 0x48
bus=smbus.SMBus(1)
cmd=0x40
```

```

def analogRead(chn):
    value = bus.read_byte_data(address,cmd+chn)
    return value
def analogWrite(value):
    bus.write_byte_data(address,cmd,value)
def setup():
    GPIO.setmode(GPIO.BOARD)
def loop():
    while True:
        value = analogRead(0)
        voltage = value / 255.0 * 3.3
        Rt = 10 * voltage / (3.3 - voltage)
        tempK = 1/(1/(273.15 + 25) + math.log(Rt/10)/3950.0)
        tempC = tempK -273.15
        print('ADC Value : %d, Voltage : %.2f,
Temperature : %.2f%(value,voltage,tempC))
        time.sleep(0.01)
def destroy():
    GPIO.cleanup()
if __name__ == '__main__':
    print ('Program is starting ... ')
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()

```

Code Interpretation

Dieser Code ähnelt dem Potentiometer zur Steuerung von LED-Leuchten, der Algorithmus zur Berechnung der Temperatur ist jedoch unterschiedlich. Das folgende Programm ist der Algorithmus zur Berechnung der Temperatur.

```

def loop():
    while True:
        value = analogRead(0)
        voltage = value / 255.0 * 3.3
        Rt = 10 * voltage / (3.3 - voltage)
        tempK = 1/(1/(273.15 + 25) + math.log(Rt/10)/3950.0)
        tempC = tempK -273.15
        print('ADC Value : %d, Voltage : %.2f,
Temperature : %.2f%(value,voltage,tempC))

```

time.sleep(0.01)

Lesen Sie zuerst den ADC Wert des Anschlusses PCF8591 A0 ab, berechnen Sie dann die Spannung und den Widerstand des Thermistors gemäß dem Ohmschen Gesetz und wandeln Sie ihn dann in einen Temperaturwert gemäß der Formel um.

Lektion 13 Joystick

Überblick

In dieser Lektion lernen Sie, wie Sie mit einem Raspberry Pi einen Joystick Sensor steuern.

Erforderliche Teile

- 1 x Raspberry Pi
- 1 x PCF8591 Modul
- 11 x Jumper Kabel
- 1 x Steckbrett
- 1 x Joystick

Produkt Einführung

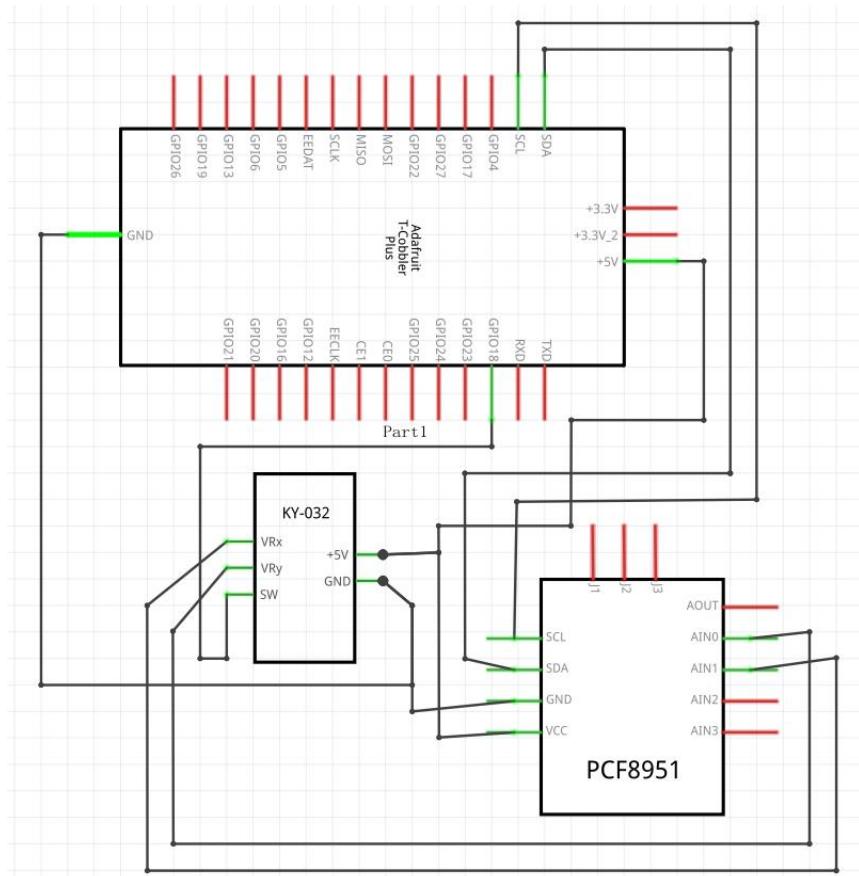
Joystick

Joystick ist eine Art Sensor, der mit Ihren Fingern verwendet wird und in Gamepads und Fernbedienungen weit verbreitet ist. Es kann sich gleichzeitig in Richtung Y oder Richtung X verschieben. Und es kann auch in Richtung Z gedrückt werden.

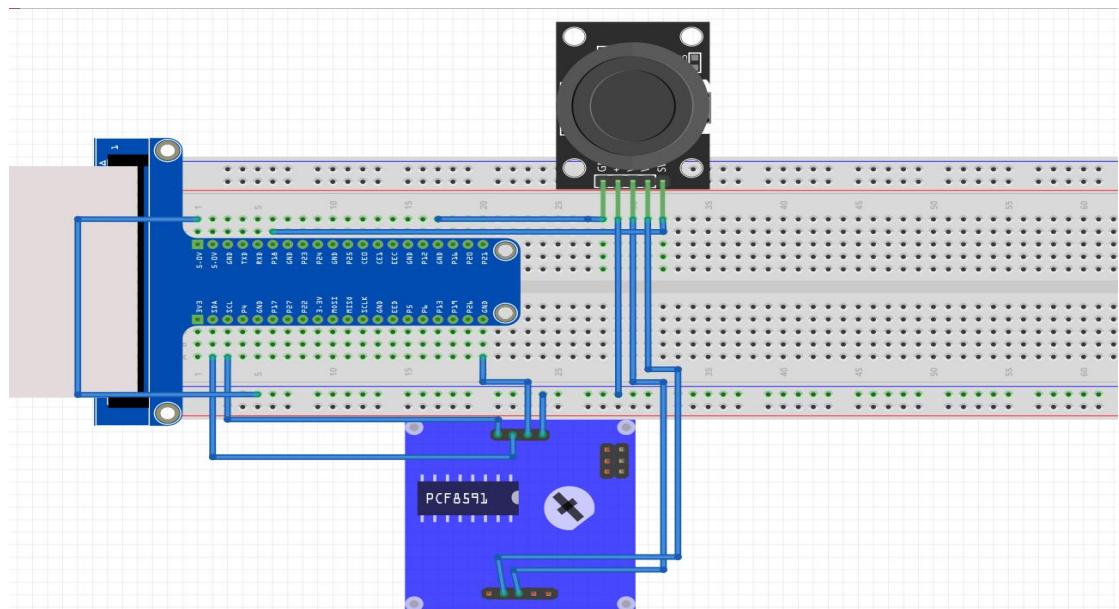




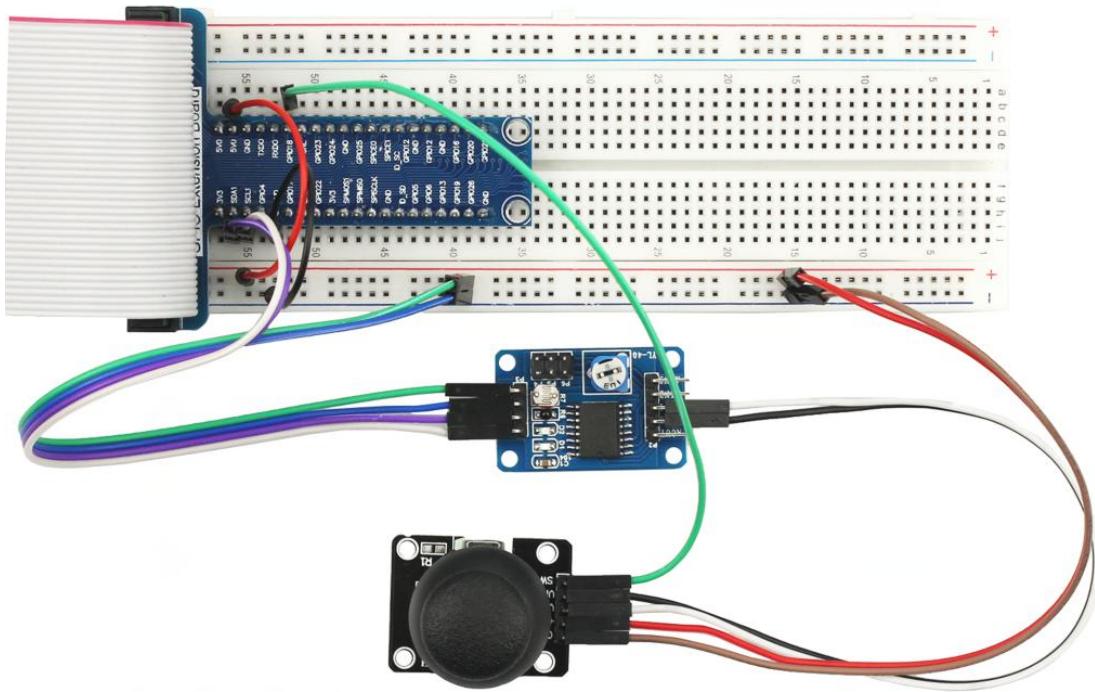
Schaltplan



Verdrahtung Diagramm



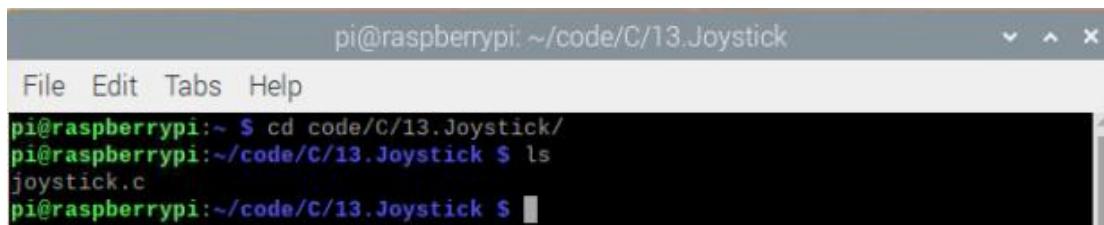
Beispielbild



C code

Öffnen Sie das Terminal und geben Sie den Befehl `cd code / C / 13.Joystick` ein, um das Codeverzeichnis `joystick.c` aufzurufen.

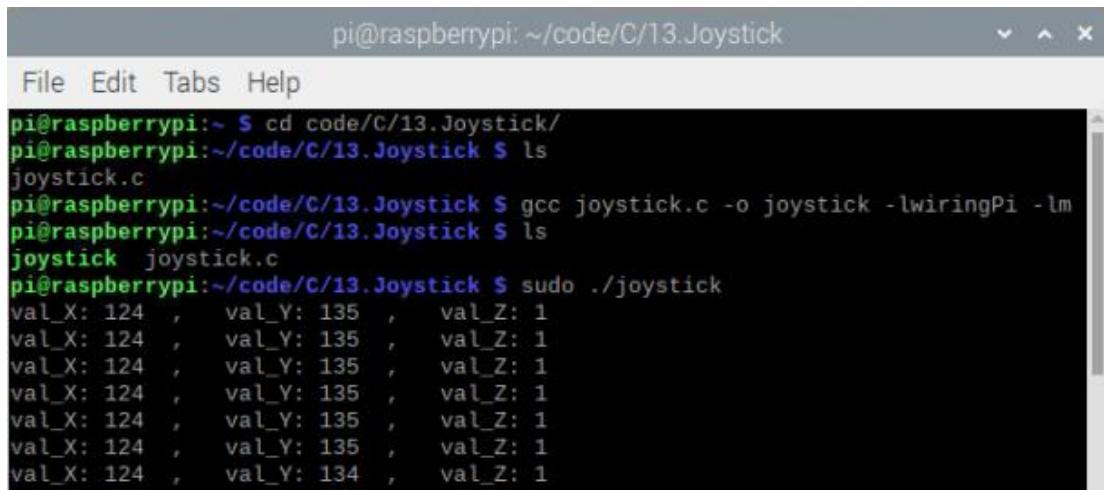
Geben Sie den Befehl `ls` ein, um die Datei `joystick.c` im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/C/13.Joystick
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/13.Joystick/
pi@raspberrypi:~/code/C/13.Joystick $ ls
joystick.c
pi@raspberrypi:~/code/C/13.Joystick $
```

Geben Sie den Befehl `gcc joystick.c -o joystick -lwiringPi -lm` ein, um `joystick.c` als Joystick für ausführbare Dateien zu generieren, und geben Sie den Befehl `ls` zum Anzeigen ein.

Geben Sie den Befehl `sudo ./joystick` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi:~/code/C/13.Joystick
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/13.Joystick/
pi@raspberrypi:~/code/C/13.Joystick $ ls
joystick.c
pi@raspberrypi:~/code/C/13.Joystick $ gcc joystick.c -o joystick -lwiringPi -lm
pi@raspberrypi:~/code/C/13.Joystick $ ls
joystick joystick.c
pi@raspberrypi:~/code/C/13.Joystick $ sudo ./joystick
val_X: 124 , val_Y: 135 , val_Z: 1
val_X: 124 , val_Y: 134 , val_Z: 1
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>
#include <softPwm.h>

#define address 0x48
#define pinbase 64
#define A0 pinbase + 0
#define A1 pinbase + 1
#define A2 pinbase + 2
#define A3 pinbase + 3

#define Z_Pin 1

int main(void){
    int val_X,val_Y,val_Z;
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    pinMode(Z_Pin,INPUT);
```

```
pullUpDnControl(Z_Pin,PUD_UP);
pcf8591Setup(pinbase,address);

while(1){
    val_Z = digitalRead(Z_Pin);
    val_Y = analogRead(A0);
    val_X = analogRead(A1);
    printf("val_X: %d \tval_Y: %d \tval_Z: %d
\n",val_X,val_Y,val_Z);
    delay(100);
}
return 0;
}
```

Code Interpretation

```
while(1){
    val_Z = digitalRead(Z_Pin);
    val_Y = analogRead(A0);
    val_X = analogRead(A1);
    printf("val_X: %d \tval_Y: %d \tval_Z: %d
\n",val_X,val_Y,val_Z);
    delay(100);
}
```

Konfigurieren Sie Z_Pin im Code als Pull-up-Eingabemodus. Verwenden Sie in der while-Schleife der Hauptfunktion AnalogRead (), um die Werte der X- und Y-Achse zu lesen, und digitalRead (), um die Werte der Z-Achse zu lesen, und drucken Sie sie dann aus.

Python code

Öffnen Sie das Terminal und geben Sie mit dem Befehl `cd code / python / 13.Joystick` / das Codeverzeichnis ein

Geben Sie den Befehl `ls` ein, um die Datei joystick.py im Verzeichnis anzuzeigen.

```
pi@raspberrypi: ~/code/python/13.Joystick
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/13.Joystick/
pi@raspberrypi:~/code/python/13.Joystick $ ls
joystick.py
pi@raspberrypi:~/code/python/13.Joystick $
```

Geben Sie den Befehl `python3 joystick.py` ein, um den Code auszuführen. Das Ergebnis ist wie folgt:

```
pi@raspberrypi: ~/code/python/13.Joystick
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/13.Joystick/
pi@raspberrypi:~/code/python/13.Joystick $ ls
joystick.py
pi@raspberrypi:~/code/python/13.Joystick $ python3 joystick.py
Program is starting ...
value_X: 121 ,  value_Y: 131 ,  value_Z: 1
```

Das Folgende ist der Code:

```
import RPi.GPIO as GPIO
import smbus
import time

address = 0x48
bus=smbus.SMBus(1)
cmd=0x40
Z_Pin = 12
def analogRead(chn):
    bus.write_byte(address,cmd+chn)
    value = bus.read_byte(address)
    value = bus.read_byte(address)
    return value
```

```

def analogWrite(value):
    bus.write_byte_data(address,cmd,value)

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(Z_Pin,GPIO.IN,GPIO.PUD_UP)
def loop():
    while True:
        val_Z = GPIO.input(Z_Pin)
        val_Y = analogRead(0)
        val_X = analogRead(1)
        print
        ('value_X: %d ,\tvalue_Y: %d ,\tvalue_Z: %d' %(val_X,val_Y,val_Z))
        time.sleep(0.01)

def destroy():
    bus.close()
    GPIO.cleanup()

if __name__ == '__main__':
    print ('Program is starting ... ')
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()

```

Code Interpretation

```

def loop():
    while True:
        val_Z = GPIO.input(Z_Pin)
        val_Y = analogRead(0)
        val_X = analogRead(1)
        print
        ('value_X: %d ,\tvalue_Y: %d ,\tvalue_Z: %d' %(val_X,val_Y,val_Z))
        time.sleep(0.01)

```

Verwenden Sie in der while Schleife 'analogRead ()', um die Werte der X- und Y-Achse zu lesen, und verwenden Sie 'GPIO.input ()', um die Werte der Z-Achse zu

lesen, und verwenden Sie dann 'print (' value_X:% d, \ tvalue_Y:% d, \ tvalue_Z:% d
'% (val_X, val_Y, val_Z))' Ausgabe.

Lektion 14 Relais & Motor

Überblick

In dieser Lektion lernen Sie eine Art spezielles Schaltermodul, das Relaismodul. Wir verwenden einen Druckknopf, um ein Relais zu steuern und den Motor anzutreiben.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-Erweiterungskarte

1 x Steckbrett

1 x 1k Ω Widerstand

1 x 220 Ω Widerstand

1 x LED

1 x Taste

1 x Relais

1 x Motor

1 x NPN Transistor

1 x Diode

Produkt Einführung

Relais

Das Relais ist ein sicherer Schalter, der einen Stromkreis mit geringer Leistung zur Steuerung des Stromkreises mit hoher Leistung verwenden kann. Es besteht aus Elektromagneten und Kontakten. Der Elektromagnet wird durch einen Stromkreis mit geringer Leistung gesteuert und Kontakte werden in einem Stromkreis mit hoher Leistung verwendet. Wenn der Elektromagnet erregt wird, zieht er Kontakte an.



Motor

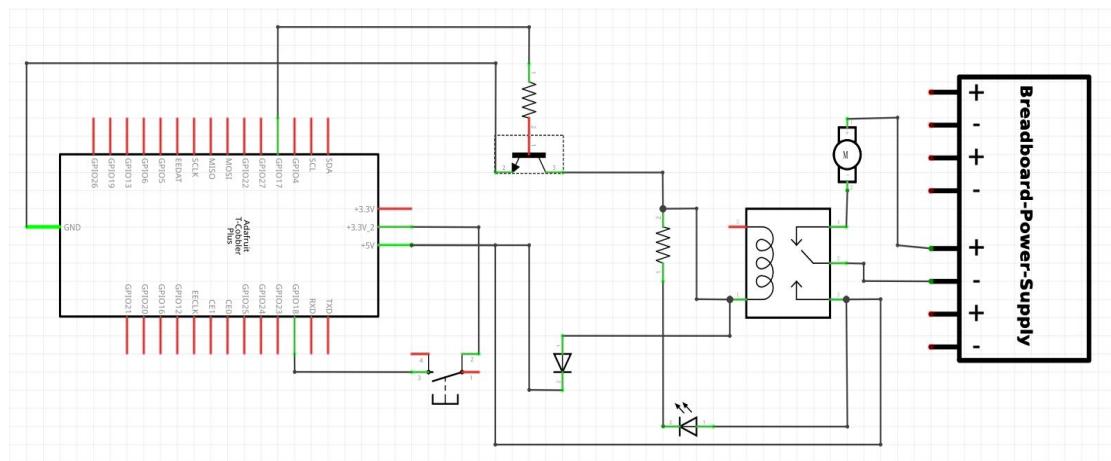
Motor ist ein Gerät, das elektrische Energie in mechanische Energie umwandelt. Der Motor besteht aus zwei Teilen: Stator und Rotor. Wenn der Motor arbeitet, ist der stationäre Teil Stator und der rotierende Teil ist Rotor. Der Stator ist normalerweise das äußere Gehäuse des Motors und verfügt über Klemmen zum Anschließen an die Stromversorgung. Der Rotor ist normalerweise die Welle des Motors und kann andere mechanische Geräte zum Laufen bringen. Das folgende Diagramm zeigt einen kleinen Gleichstrommotor mit zwei Stiften. Wenn der Motor an die Stromversorgung angeschlossen wird, dreht er sich in eine Richtung. Die Polarität der Stromversorgung umkehren, dann dreht sich der Motor in die entgegengesetzte Richtung.



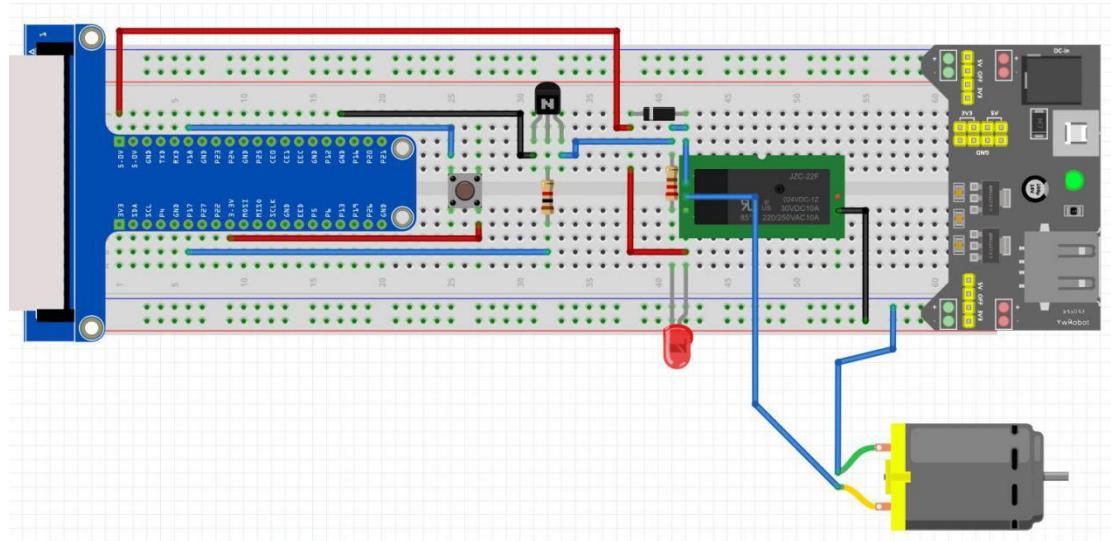
Circuit

Beachten Sie beim Anschließen des Stromkreises, dass der Motor eine Hochleistungskomponente ist. Verwenden Sie nicht die vom RPi bereitgestellte Leistung, da dies zu Schäden am RPi führen kann.

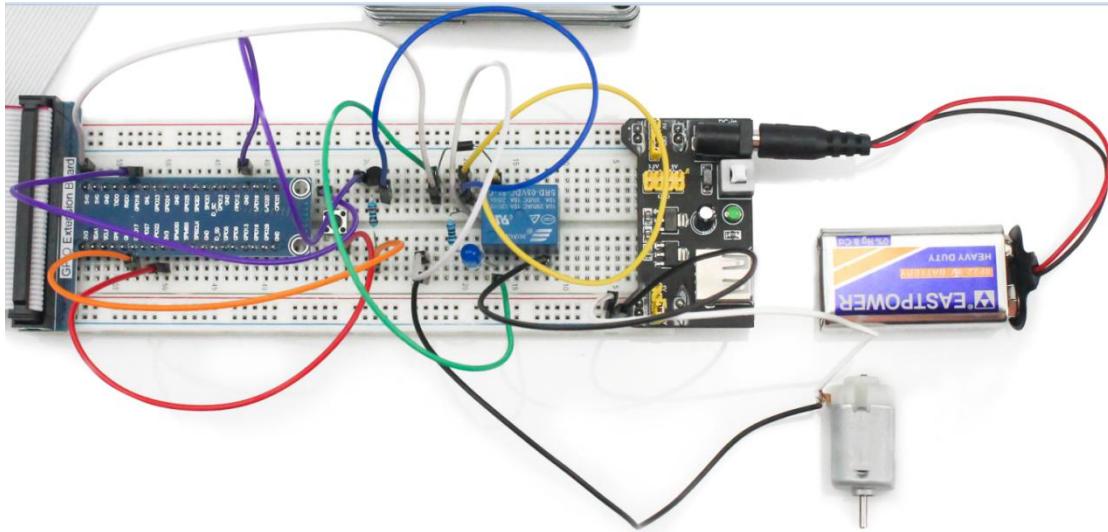
Schaltplan



Verdrahtung Diagramm



Beispielbild



C code

Öffnen Sie das Terminal und geben Sie den Befehl "[cd code / C / 14.Relay /](#)" ein, um das Codeverzeichnis "Relay" aufzurufen.

Geben Sie den Befehl "[ls](#)" ein, um die Datei "Relay.c" im Verzeichnis anzuzeigen.

```
pi@raspberrypi:~/code/C/14.Relay
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/14.Relay/
pi@raspberrypi:~/code/C/14.Relay$ ls
Relay.c
pi@raspberrypi:~/code/C/14.Relay$
```

Geben Sie den Befehl "[gcc Relay.c -o Relay -lwiringPi](#)" ein, um die ausführbare Datei "Relay" von "Relay.c" zu generieren. Geben Sie den Befehl "ls" ein, um sie anzuzeigen.

Geben Sie den Befehl "[sudo ./Relay](#)" ein, um den Code auszuführen. Das Ergebnis ist wie folgt:

```
pi@raspberrypi:~/code/C/14.Relay
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/14.Relay/
pi@raspberrypi:~/code/C/14.Relay$ ls
Relay.c
pi@raspberrypi:~/code/C/14.Relay$ gcc Relay.c -o Relay -lwiringPi
pi@raspberrypi:~/code/C/14.Relay$ ls
Relay Relay.c
pi@raspberrypi:~/code/C/14.Relay$ sudo ./Relay
starting...
pi@raspberrypi:~/code/C/14.Relay$
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>

#define relayPin    0 //define the relayPin
#define buttonPin 1 //define the buttonPin
int main(void)
{
    if(wiringPiSetup() == -1){ //when initialize wiring fairelay,print message to screen
        printf("fairelay !");
        return 1;
    }
    printf("starting...\n");
    pinMode(relayPin, OUTPUT);
    pinMode(buttonPin, INPUT);
    pullUpDnControl(buttonPin, PUD_UP); //pull up to high level
    int i=0;
    int g=0;
    while(1){
        if(digitalRead(buttonPin)==HIGH && g==0)
        {
            delay(20);
            if(digitalRead(buttonPin)==HIGH)
            {
                i=i+1;
                i=i%2;
                g=1;
                if(i==0)
                {
                    digitalWrite(relayPin,LOW);
                    printf("relayPin.....off\n");
                }
                if(i==1)
                {
                    digitalWrite(relayPin,HIGH);
                    printf("relayPin.....on\n");
                }
            }
        }
        else if(digitalRead(buttonPin)==LOW)
```



```
    {
        g=0;
    }

}

return 0;
}
```

Code Interpretation

Definieren Sie in der Hauptfunktion "main ()" zunächst zwei Flags "i = 0" und "g = 0". Mit "i" wird bestimmt, ob die vorherige Taste gedrückt wurde. "G" ist ein Flag-Bit wird verwendet, um zu beurteilen, ob die Taste noch gedrückt ist. "if (digitalRead (buttonPin) == HIGH && g == 0)" ist die Beurteilungsaussage über den Zustand der Taste. "delay (20)" ist eine Verzögerungsfunktion Verzögerung Die Funktion der Zeitfunktion besteht darin, den Jitter der Taste zu beseitigen.

```
int i=0;
int g=0;
while(1){
    if(digitalRead(buttonPin)==HIGH && g==0)
    {
        delay(20);
        if(digitalRead(buttonPin)==HIGH)
```

Python code

1. Verwenden Sie den Befehl "[cd code / python / 14.Relay /](#)", um das Verzeichnis des Relay-Codes einzugeben.

2. Verwenden Sie den Befehl "[python Relay.py](#)", um den Code "Relay.py" auszuführen.

The screenshot shows a terminal window titled 'pi@raspberrypi:~/code/python/14.Relay'. The window contains the following text:

```
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/14.Relay/
pi@raspberrypi:~/code/python/14.Relay $ python Relay.py
Initialization
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import time
```

```
relayPin = 11      # define the relayPin
buttonPin = 12     # define the buttonPin

def setup():
    print ('Initialization')
    GPIO.setmode(GPIO.BOARD)          # Numbers GPIOs by physical location
    GPIO.setup(relayPin, GPIO.OUT)     # Set relayPin's mode is output
    GPIO.setup(buttonPin, GPIO.IN)    # Set buttonpin mode input

def loop():
    i=0
    g=0
    while True:
        reading = GPIO.input(buttonPin)
        if reading == GPIO.HIGH & g==0:
            time.sleep(0.2)
            reading = GPIO.input(buttonPin)
            if reading == GPIO.HIGH:
                i=i+1
                i=i%2
                g=1
                if i==0:
                    GPIO.output(relayPin,GPIO.LOW)
                    print("relayPin.....off")
                if i==1:
                    GPIO.output(relayPin,GPIO.HIGH)
                    print("relayPin.....on")
            else:
                g=0

def destroy():
    GPIO.output(relayPin, GPIO.LOW)      # relay off
    GPIO.cleanup()                      # Release resource

if __name__ == '__main__':
    setup()                           # Program start from here
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

Code Interpretation

Als erstes müssen wir zwei Symbole "i = 0" und "g = 0" in der Funktion 'loop ()' definieren. "i" wird verwendet, um zu bestimmen, ob die vorherige Taste gedrückt wurde. "g" wird verwendet, um zu bestimmen, ob die Taste kontinuierlich gedrückt wird.

i=0

g=0

Verwenden Sie die Anweisung 'reading = GPIO.input (buttonPin)', um den Status zu überprüfen. Der Code 'if Reading == GPIO.HIGH & g == 0' wird verwendet, um zu bestimmen, ob die Taste gedrückt wird. Wenn die Taste gedrückt wird, geben Sie ihr eine Verzögerungsfunktion 'time.sleep (0.2)' zum Entprellen, um das zu verhindern. Wenn nach einer Verzögerung die Taste noch gedrückt wird, wird beurteilt, dass der aktuelle Status der Taste gedrückt ist.

```
reading = GPIO.input(buttonPin)
if reading == GPIO.HIGH & g==0:
    time.sleep(0.2)
    reading = GPIO.input(buttonPin)
    if reading == GPIO.HIGH:
```

Lektion 15 Servo

Überblick

In dieser Lektion lernen wir, wie man die Drehung des SG90-Servos mit Raspberry pi steuert und wie man den Drehwinkel des Servos steuert. Das Servo ist eine Art Getriebemotor, der sich nur um 180 Grad drehen kann. Es wird durch den von Raspberry pi gesendeten elektrischen Impuls gesteuert. Es hat drei Drähte. Der braune Draht ist der Erdungsdraht, der mit dem GND Pin des Raspberry Pi verbunden ist, der rote Draht ist der Pluspol der Stromversorgung, der mit dem 5V Pin des Raspberry Pi verbunden ist, und der orangefarbene Draht ist die Signalleitung, die mit dem GPIO18 Pin des Raspberry pi verbunden ist.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-Erweiterungskarte und Kabel

1 x Steckbrett

1 x SG90 Servo

3 x Jumper Kabel

Produkt Einführung

SG90

Servo ist ein automatisches Steuersystem, das aus Gleichstrommotor, Untersteigungsgtriebe, Sensor und Steuerkreis besteht. Normalerweise kann es sich im Bereich von 180 Grad drehen. Servo kann ein größeres Drehmoment abgeben und wird häufig in Modellflugzeugen, Robotern usw. verwendet. Es hat drei Leitungen, darunter zwei für positive (2-VCC, rot), negative (3-GND, braun) und die Signalleitung (1-Signal, orange).

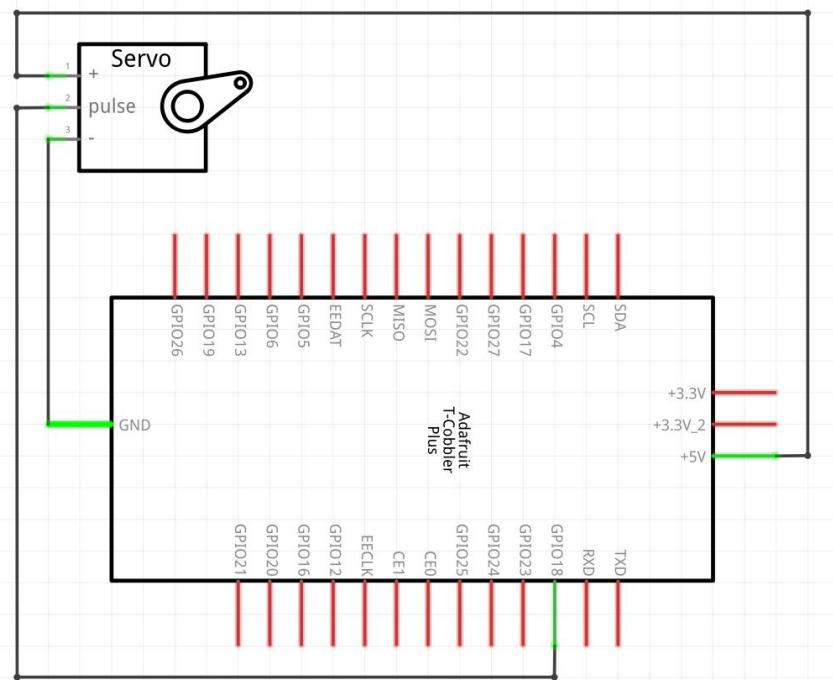


Wir verwenden ein 50 Hz PWM-Signal mit einem Arbeitszyklus in einem bestimmten Bereich, um das Servo anzutreiben. Die Dauer von 0,5 ms bis 2,5 ms des PWM Einzelzyklus Hochpegels entspricht dem Servowinkel von 0 Grad bis 180 Grad linear. Ein Teil der entsprechenden Werte lautet wie folgt:

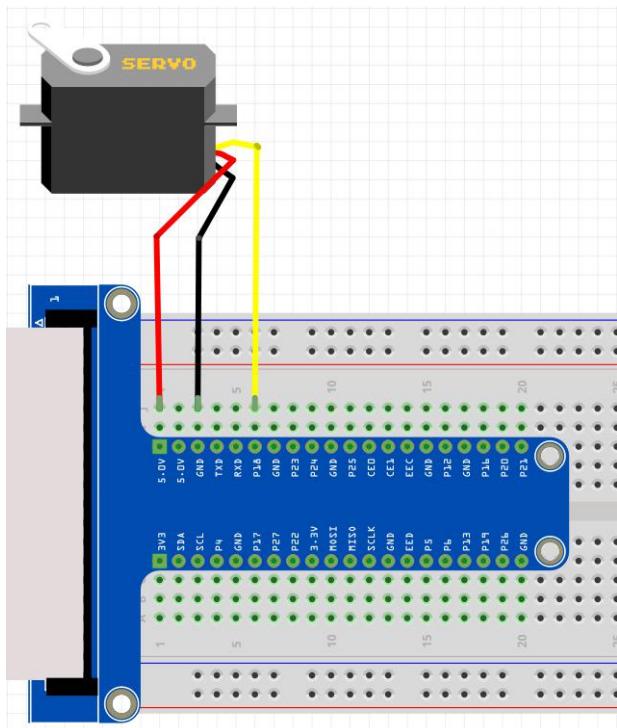
High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	90 degree
2ms	135 degree
2.5ms	180 degree

Wenn Sie das Servosignal ändern, dreht sich das Servo in die angegebene Position.

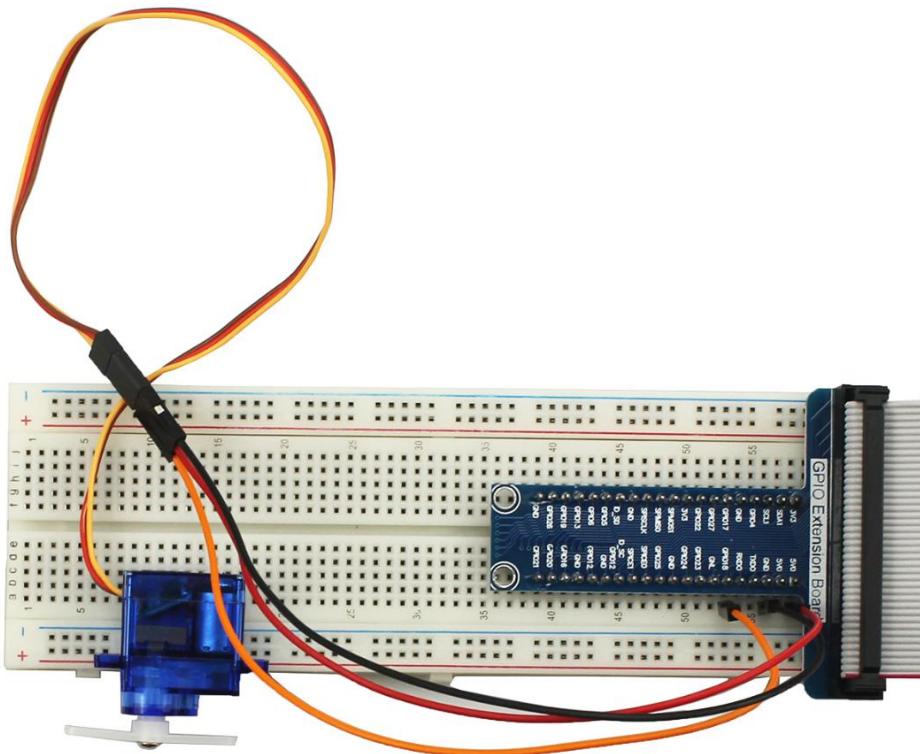
Schaltplan



Verdrahtung Diagramm



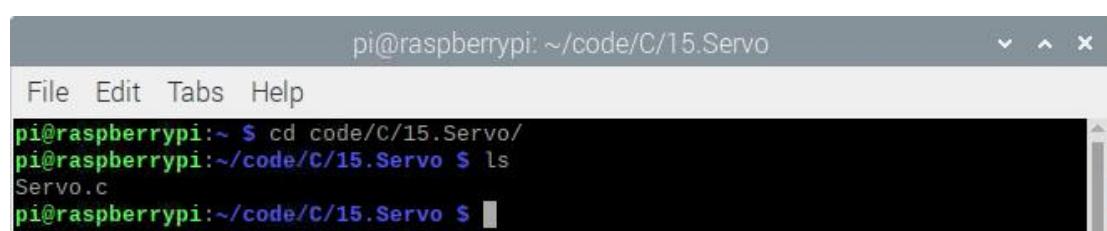
Beispiel Abbildung



C code

Öffnen Sie das Terminal und geben Sie den Befehl "[cd code / C / 15.Servo /](#)" ein, um das Codeverzeichnis "Servo" aufzurufen.

Geben Sie den Befehl "[ls](#)" ein, um die Datei "Servo.c" im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/C/15.Servo
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/15.Servo/
pi@raspberrypi:~/code/C/15.Servo $ ls
Servo.c
pi@raspberrypi:~/code/C/15.Servo $
```

Geben Sie den Befehl "[gcc Servo.c -o Servo -lwiringPi](#)" ein, um die ausführbare Datei "Servo.c" "Servo" zu generieren, und geben Sie den Befehl "[ls](#)" ein, um sie anzuzeigen.

Geben Sie den Befehl "`sudo ./Servo`" ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi: ~/code/C/15.Servo
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/15.Servo/
pi@raspberrypi:~/code/C/15.Servo $ ls
Servo.c
pi@raspberrypi:~/code/C/15.Servo $ gcc Servo.c -o Servo -lwiringPi
pi@raspberrypi:~/code/C/15.Servo $ ls
Servo  Servo.c
pi@raspberrypi:~/code/C/15.Servo $ sudo ./Servo
starting ...
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>
#define OFFSET_MS 3      //Define the unit of servo pulse offset: 0.1ms
#define SERVO_MIN_MS 5+OFFSET_MS          //define the pulse duration for
minimum angle of servo
#define SERVO_MAX_MS 25+OFFSET_MS        //define the pulse duration for
maximum angle of servo

#define servoPin     1      //define the GPIO number connected to servo
long map(long value,long fromLow,long fromHigh,long toLow,long toHigh){
    return (toHigh-toLow)*(value-fromLow) / (fromHigh-fromLow) + toLow;
}
void servoInit(int pin){           //initialization function for servo PMW pin
    softPwmCreate(pin, 0, 200);
}
void servoWrite(int pin, int angle){ //Specif a certain rotation angle (0-180) for
the servo
    if(angle > 180)
        angle = 180;
    if(angle < 0)
        angle = 0;
    softPwmWrite(pin,map(angle,0,180,SERVO_MIN_MS,SERVO_MAX_MS));
}
void servoWriteMS(int pin, int ms){ //specific the unit for pulse(5-25ms) with
specific duration output by servo pin: 0.1ms
```

```

if(ms > SERVO_MAX_MS)
    ms = SERVO_MAX_MS;
if(ms < SERVO_MIN_MS)
    ms = SERVO_MIN_MS;
softPwmWrite(pin,ms);
}

int main(void)
{
    int i;
    if(wiringPiSetup() == -1){ //when initialize wiring faiservo,print message to
screen
        printf("setup wiringPi faiservo !");
        return 1;
    }
    printf("starting ...\n");
    servoInit(servoPin);           //initialize PMW pin of servo
    while(1){
        for(i=SEROV_MIN_MS;i<SERVO_MAX_MS;i++){ //make servo rotate
from minimum angle to maximum angle
            servoWriteMS(servoPin,i);
            delay(10);
        }
        delay(1000);
        for(i=SEROV_MAX_MS;i>SERVO_MIN_MS;i--){ //make servo rotate
from maximum angle to minimum angle
            servoWriteMS(servoPin,i);
            delay(10);
        }
        delay(1000);
    }
    return 0;
}

```

Code Interpretation

Zur Steuerung des Servos ist ein 50 Hz Impuls erforderlich, nämlich ein Zyklus von 20 ms. In der Funktion "softPwmCreate (int pin, int initialValue, int pwmRange)" ist die Einheit des dritten Parameters "pwmRange" 100US, nämlich 0,1 ms. Um die PWM mit einem Zyklus von 20 ms zu erhalten, sollte der pwmRange auf 200 gesetzt werden. In der Unterfunktion von servoInit () erstellen wir einen PWM Pin mit pwmRange 200.

```
void servoInit(int pin){
    softPwmCreate(pin, 0, 200);
}
```

Da 0-180 Grad Servo einer PWM-Impulsbreite von 0,5-2,5 ms entsprechen, mit "pwmRange" 200 und einer Einheit von 0,1 ms. In der Funktion "softPwmWrite (int pin, int value)" entspricht der Bereich 5-25 des Parameterwerts 0-180 Grad Servo. Darüber hinaus sollte die in der Unterfunktion servoWriteMS () geschriebene Zahl im Bereich von 5 bis 25 liegen. In der Praxis weist die Impulsbreite jedoch aufgrund des Herstellungsfehlers jedes Servos auch eine Abweichung auf. Wir definieren also eine minimale und eine maximale Impulsbreite sowie einen Fehlerversatz.

```
void servoWriteMS(int pin, int ms){
    if(ms > SERVO_MAX_MS)
        ms = SERVO_MAX_MS;
    if(ms < SERVO_MIN_MS)
        ms = SERVO_MIN_MS;
    softPwmWrite(pin,ms);
}
```

Geben Sie in der Unterfunktion "servoWrite (int pin, int angle)" direkt den Winkel 0-180 Grad ein und ordnen Sie den Winkel der Impulsbreite zu und geben Sie ihn dann aus.

```
void servoWrite(int pin, int angle){           if(angle > 180)
    angle = 180;
if(angle < 0)
    angle = 0;

softPwmWrite(pin,map(angle,0,180,SERVO_MIN_MS,SERVO_MAX_MS));
}
```

Verwenden Sie schließlich im "while" Zyklus der Hauptfunktion zwei "for" Zyklen, um das Servo von 0 Grad auf 180 Grad und dann von 180 Grad auf 0 Grad zu drehen.

```
while(1){
    for(i=SEROV_MIN_MS;i<SERVO_MAX_MS;i++){
        servoWriteMS(servoPin,i);
        delay(10);
    }
    delay(1000);
    for(i=SEROV_MAX_MS;i>SERVO_MIN_MS;i--){

```

```
    servoWriteMS(servoPin,i);
    delay(10);
}
delay(1000);
}
```

Python code

1. Verwenden Sie den Befehl "[cd code / python / 15.Servo /](#)", um zum Codeverzeichnis "Servo" zu wechseln.
2. Verwenden Sie den Befehl "[python Servo.py](#)", um den Code "Servo.py" auszuführen.



The screenshot shows a terminal window titled "pi@raspberrypi: ~ /code/python/15.Servo". The window has a standard Linux-style title bar with icons for minimize, maximize, and close. Below the title bar is a menu bar with "File", "Edit", "Tabs", and "Help". The main terminal area contains the following text:
pi@raspberrypi:~ \$ cd code/python/15.Servo/
pi@raspberrypi:~/code/python/15.Servo \$ python Servo.py
starting...
[REDACTED]

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import time

DUTY = 0.5
MIN_DUTY=2.5+DUTY
MAX_DUTY=12.5+DUTY
servo = 12

def setup():
    global P
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(servo,GPIO.OUT)
    GPIO.output(servo,GPIO.LOW)

P = GPIO.PWM(servo,50)
```

```
P.start(0)

def map( value, fromLow, fromHigh, toLow, toHigh):
    return (toHigh-toLow)*(value-fromLow) / (fromHigh-fromLow) + toLow

def servoWrite(a):  # make the servo rotate to specific angle (0-180 degrees)
    if(a<0):
        a=0
    elif(a>180):
        a=180
    P.ChangeDutyCycle(map(a,0,180,MIN_DUTY,MAX_DUTY))  #map the
angle to duty cycle and output it

def loop():
    while True:
        for i in range(0,181,1):
            servoWrite(i)
            time.sleep(0.01)
        time.sleep(0.5)
        for i in range (180,-1,-1):
            servoWrite(i)
            time.sleep(0.01)
        time.sleep(0.5)

def destroy():
    p.stop()
    GPIO.cleanup()

if __name__ == '__main__':
    #Program start from here
    print("starting...")
    setup()
    try:
        loop()
    except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program
destroy() will be executed.
        destroy()
```

Code Interpretation

Zur Steuerung des Servos ist ein 50 Hz Impuls erforderlich, nämlich ein Zyklus von 20 ms. Wir müssen also den Code "P = GPIO.PWM (Servo, 50)" verwenden, um die PWM-Frequenz von "Servo" auf 50 Hz einzustellen.

P = GPIO.PWM(servo,50)

Da 0-180 Grad Servo einer PWM-Impulsbreite von 0,5-2,5 ms innerhalb des Zyklus 20 ms und einem Arbeitszyklus von 2,5% bis 12,5% entsprechen. Schreiben Sie den Winkel in die Unterfunktion ‘def servoWrite(a)’ , ordnen Sie den Winkel dem Arbeitszyklus für die Ausgabe von PWM zu und geben Sie dann den Winkel aus, unter dem sich das Servo dreht. Tatsächlich variiert jedoch aufgrund des Herstellungsfehlers jedes Lenkgetriebes auch die Impulsbreite. Daher definieren wir die Funktion “‘def map(value, fromLow, fromHigh, toLow, toHigh)’ , um die minimale Impulsbreite und maximale Impulsbreite sowie den Fehlerversatz zu berechnen.

```
def servoWrite(a): # make the servo rotate to specific angle (0-180 degrees)
    if(a<0):
        a=0
    elif(a>180):
        a=180
    P.ChangeDutyCycle(map(a,0,180,MIN_DUTY,MAX_DUTY)) #map the
angle to duty cycle and output it
```

Finally, in the "while" cycle of main function, use two "for" cycle to make servo rotate from 0 degrees to 180 degrees, and then from 180 degrees to 0 degrees.

```
def loop():
    while True:
        for i in range(0,181,1):
            servoWrite(i)
            time.sleep(0.01)
            time.sleep(0.5)
        for i in range (180,-1,-1):
            servoWrite(i)
            time.sleep(0.01)
            time.sleep(0.5)
```

Lektion 16 Schrittmotor

Überblick

In dieser Lektion lernen Sie, wie Sie die Drehung eines Schrittmotors mit Raspberry pi steuern und wie Sie den Drehwinkel eines Schrittmotors steuern. Wir haben zuvor Gleichstrommotor und Servo gelernt: Der Gleichstrommotor kann sich ständig drehen, aber wir können nicht es dreht sich zu einem bestimmten Winkel. Im Gegenteil, das gewöhnliche Servo kann sich bis zu einem bestimmten Winkel drehen, aber nicht ständig. In diesem Kapitel lernen wir einen Motor kennen, der sich nicht nur konstant, sondern auch in einem bestimmten Winkel drehen kann. Die Verwendung eines Schrittmotors kann leicht zu einer höheren Genauigkeit der mechanischen Bewegung führen.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-Erweiterungskarte

1 x Steckbrett

1 x Schrittmotor

1 x ULN2003 Schrittmotor

Produkt Einführung

Schrittmotor

Der Schrittmotor ist ein Steuergerät mit offenem Regelkreis, das das elektrische Impulssignal in eine Winkelverschiebung oder eine lineare Verschiebung umwandelt. Im nicht überlasteten Zustand hängen die Drehzahl des Motors und die Position des Stopps nur von der Impulssignalfrequenz und der Impulszahl ab und werden von den Laständerungen nicht beeinflusst. Das äußere Teil ist der Stator und das innere ist der Rotor des Motor. Es gibt eine bestimmte Anzahl von Spulen, normalerweise ein ganzzahliges Vielfaches der Phasennummer, im Stator, und beim Einschalten wird ein Elektromagnet gebildet, um einen konkaven Teil (normalerweise Eisen oder Permanentmagnet) des Rotors anzuziehen. Daher kann der Elektromotor angetrieben

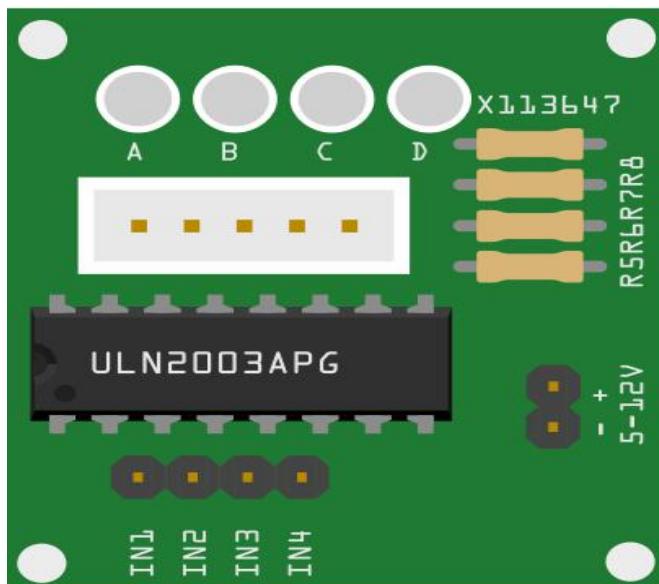
werden, indem die Spulen ordnungsgemäß auf den Stator geleitet werden. Durch Steuern der Anzahl der Drehschritte können Sie den Drehwinkel des Schrittmotors steuern. Durch Steuern der Zeit zwischen zwei Schritten können Sie die Drehzahl des Schrittmotors steuern. Bei Drehung im Uhrzeigersinn lautet die Reihenfolge der eingeschalteten Spule: A B C D A Und der Rotor dreht sich in der Reihenfolge, Schritt für Schritt nach unten, vier Schritte, vier Streicheleinheiten. Wenn die Spulen in umgekehrter Reihenfolge eingeschaltet werden, D C B A D..., dreht sich der Rotor gegen den Uhrzeigersinn. Der Schrittmotor verfügt über andere Steuermethoden, z. B. die Phase A verbinden und dann die Phase AB anschließen. Der Stator befindet sich in der Mitte des AB, nur einen halben Schritt entfernt. Auf diese Weise kann die Stabilität des Schrittmotors verbessert und das Geräusch reduziert werden. Die Reihenfolge der eingeschalteten Spulen ist:

A AB B BC C CD D DA A..., der Motor dreht sich gemäß der Reihenfolge, ein halber Schritt für einen halben Schritt, genannt vier Schritt acht Pat. Wenn die Spule in umgekehrter Reihenfolge eingeschaltet wird, dreht sich der Schrittmotor in umgekehrter Reihenfolge. Der Stator des von uns verwendeten Schrittmotors hat 32 Magnetpole, sodass ein Kreis 32 Schritte benötigt. Die Abtriebswelle des Schrittmotors ist mit einem Untersetzungsgetriebesatz verbunden, und das Untersetzungsverhältnis beträgt 1/64. Die endgültige Abtriebswelle dreht also einen Kreis, der einen Schritt von $32 * 64 = 2048$ erfordert.



ULN2003 Schrittmotor Treiber

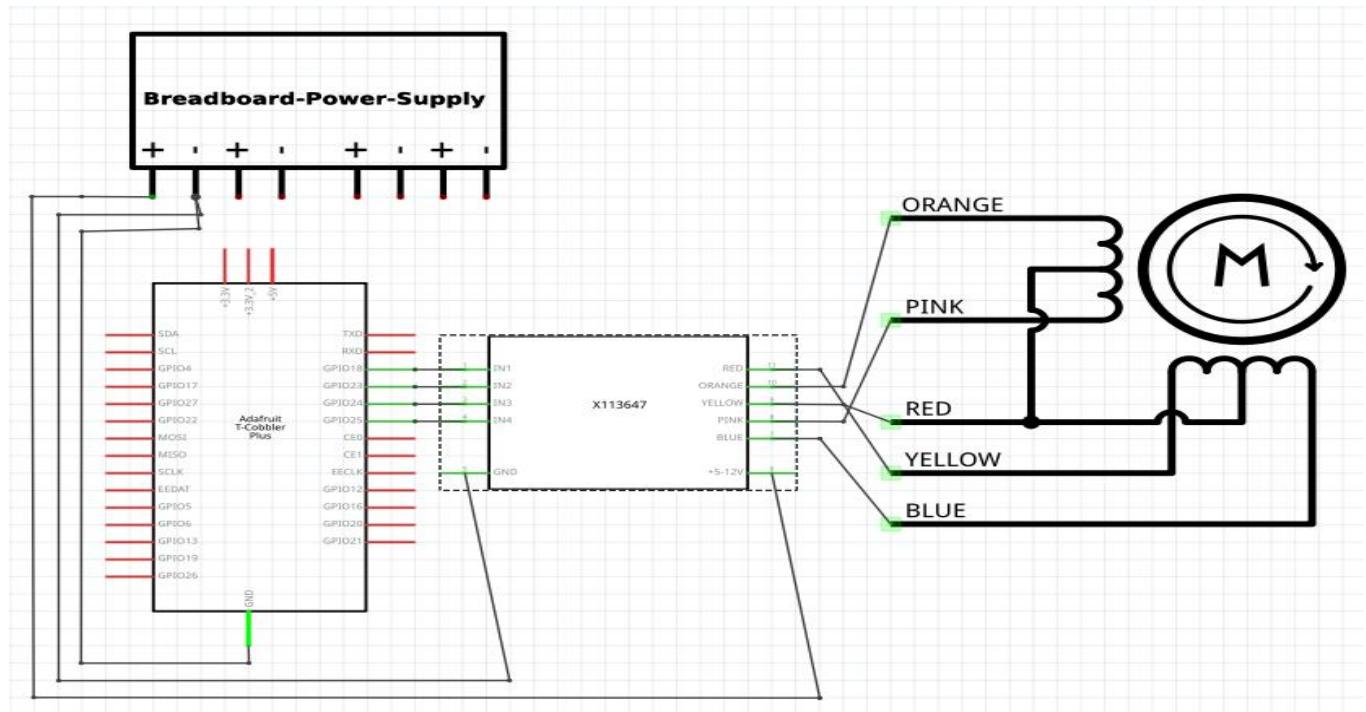
Der Schrittmotortreiber ULN2003 wird verwendet, um das schwache Signal in ein starkes Steuersignal umzuwandeln, um den Schrittmotor anzutreiben. Das Eingangssignal IN1-IN4 entspricht dem Ausgangssignal A-D, und die LED 4 ist in die Karte integriert, um den Zustand der Signale anzuzeigen. Die PWR-Schnittstelle kann als Stromversorgung für Schrittmotor verwendet werden. Standardmäßig sind PWR und VCC durch einen Kurzschluss verbunden.



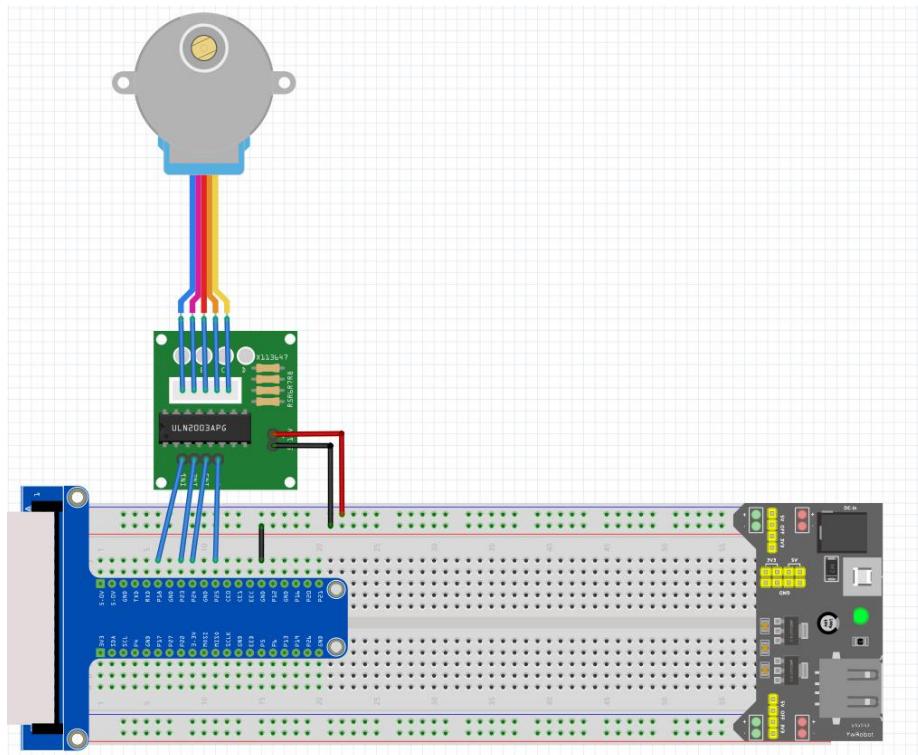
Circuit

Beim Aufbau der Schaltung die Nennspannung des Schrittmotors 5V und die Steckbrettstromversorgung unabhängig verwenden und nicht die RPi-Stromversorgung verwenden. Darüber hinaus muss das Steckbrett-Netzteil Ground mit RPi teilen.

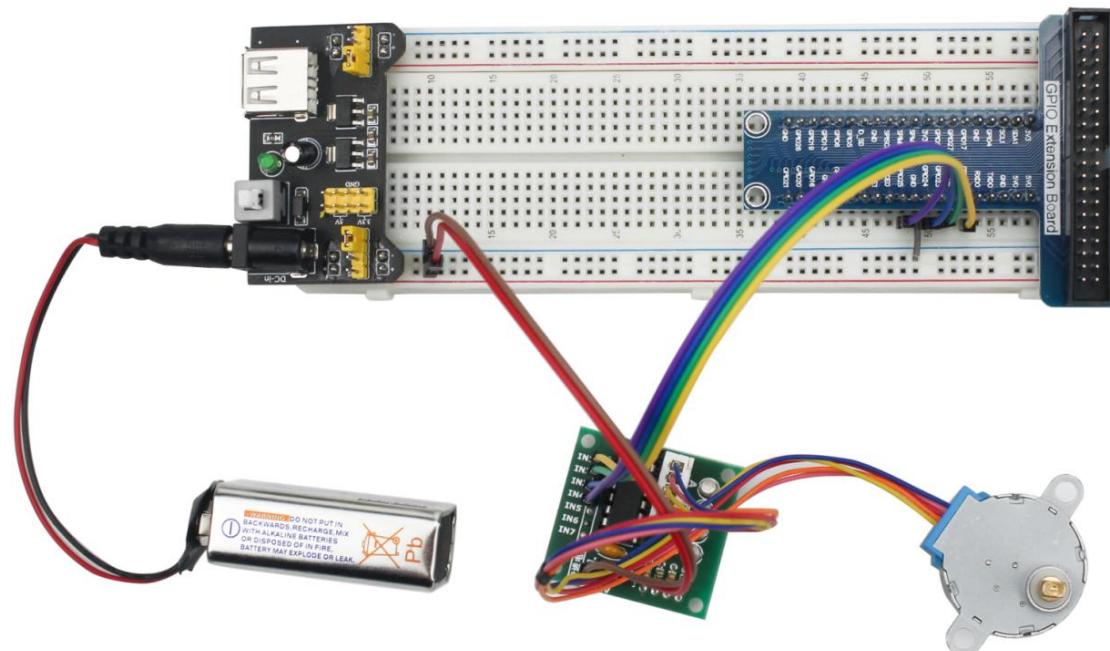
Schaltplan



Verdrahtung Diagramm



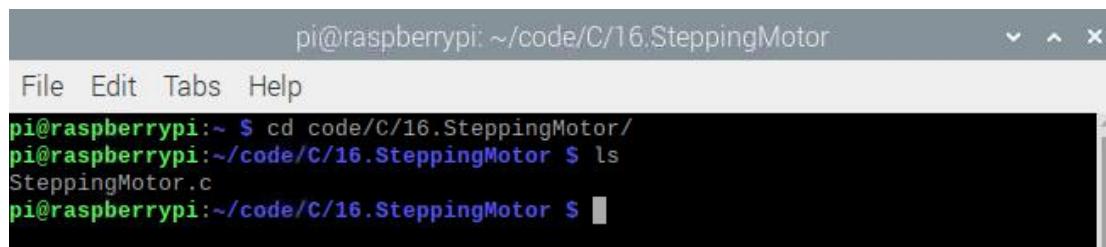
Beispiel Abbildung



C code

Öffnen Sie das Terminal und geben Sie den Befehl "[cd code / C / 16.SteppingMotor /](#)" ein, um das Codeverzeichnis "SteppingMotor" aufzurufen.

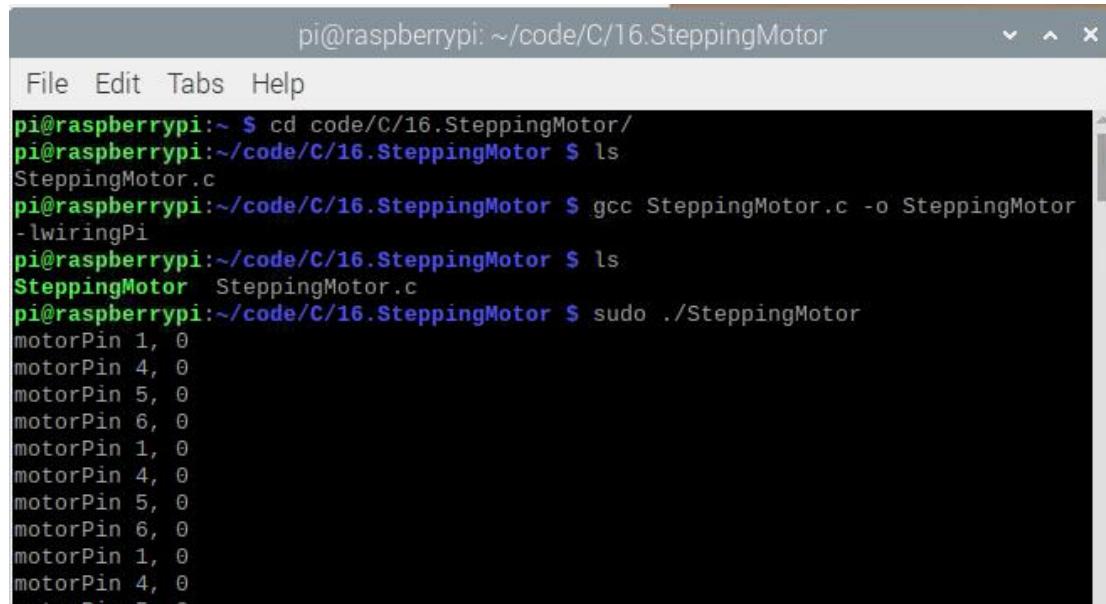
Geben Sie den Befehl "[ls](#)" ein, um die Datei "SteppingMotor.c" im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/C/16.SteppingMotor
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/16.SteppingMotor/
pi@raspberrypi:~/code/C/16.SteppingMotor $ ls
SteppingMotor.c
pi@raspberrypi:~/code/C/16.SteppingMotor $
```

Geben Sie den Befehl "[gcc SteppingMotor.c -o SteppingMotor -lwiringPi](#)" ein, um die ausführbare Datei "SteppingMotor.c" "SteppingMotor" zu generieren. Geben Sie den Befehl "[ls](#)" ein, um sie anzuzeigen.

Geben Sie "[sudo ./SteppingMotor](#)" ein, um den Code auszuführen. Die Ergebnisse werden unten angezeigt:



```
pi@raspberrypi:~/code/C/16.SteppingMotor
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/16.SteppingMotor/
pi@raspberrypi:~/code/C/16.SteppingMotor $ ls
SteppingMotor.c
pi@raspberrypi:~/code/C/16.SteppingMotor $ gcc SteppingMotor.c -o SteppingMotor
-lwiringPi
pi@raspberrypi:~/code/C/16.SteppingMotor $ ls
SteppingMotor SteppingMotor.c
pi@raspberrypi:~/code/C/16.SteppingMotor $ sudo ./SteppingMotor
motorPin 1, 0
motorPin 4, 0
motorPin 5, 0
motorPin 6, 0
motorPin 1, 0
motorPin 4, 0
motorPin 5, 0
motorPin 6, 0
motorPin 1, 0
motorPin 4, 0
motorPin 5, 0
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <stdio.h>
```

```
#include <wiringPi.h>

const int motorPins[]={1,4,5,6};      //define pins connected to four phase ABCD of
stepper motor
const int CCWStep[]={0x01,0x02,0x04,0x08}; //define power supply order for coil
for rotating anticlockwise
const int CWStep[]={0x08,0x04,0x02,0x01}; //define power supply order for coil
for rotating clockwise
//as for four phase stepping motor, four steps is a cycle. the function is used to drive
the stepping motor clockwise or anticlockwise to take four steps
void moveOnePeriod(int dir,int ms){
    int i=0,j=0;
    for (j=0;j<4;j++){ //cycle according to power supply order
        for (i=0;i<4;i++){ //assign to each pin, a total of 4 pins
            if(dir == 1) //power supply order clockwise
                digitalWrite(motorPins[i],(CCWStep[j] == (1<<i)) ? HIGH :
LOW);
            else          //power supply order anticlockwise
                digitalWrite(motorPins[i],(CWStep[j] == (1<<i)) ? HIGH :
LOW);
            printf("motorPin %d, %d \n",motorPins[i],digitalRead(motorPins[i]));
        }
        if(ms<3)           //the delay can not be less than 3ms, otherwise it will
exceed speed limit of the motor
            ms=3;
        delay(ms);
    }
}
//continuous rotation function, the parameter steps specifies the rotation cycles, every
four steps is a cycle
void moveSteps(int dir, int ms, int steps){
    int i;
    for(i=0;i<steps;i++){
        moveOnePeriod(dir,ms);
    }
}
int main(void){
    int i;

    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("setup wiringPi failed !");
    }
}
```

```

        return 1;
    }
    for(i=0;i<4;i++){
        pinMode(motorPins[i],OUTPUT);
    }

    while(1){
        moveSteps(1,3,512);      //rotating 360° clockwise, a total of 2048
steps in a circle, namely, 512 cycles.
        delay(1000);
        moveSteps(0,3,512);      //rotating 360° anticlockwise
        delay(1000);
    }
    return 0;
}

```

Code Interpretation

Definieren Sie im Code zunächst die vier Stifte des Schrittmotors und die Spulenleistungssequenz des Vier Schritt Rotationsmodus.

```

const int motorPins[]={1,4,5,6};
const int CCWStep[]={0x01,0x02,0x04,0x08};
const int CWStep[]={0x08,0x04,0x02,0x01};

```

Die Unterfunktion "moveOnePeriod (int dir, int ms)" treibt den Schrittmotor vier Schritte im Uhrzeigersinn oder im Uhrzeigersinn, vier Schritte als Zyklus an. Wenn der Parameter "dir" die Drehrichtung angibt, dreht sich das Servo vorwärts, wenn "dir" 1 ist. Andernfalls dreht es sich in die umgekehrte Richtung. Der Parameter "ms" gibt die Zeit zwischen zwei Schritten an. Die "ms" des in diesem Projekt verwendeten Schrittmotors beträgt 3 ms (die kürzeste Zeit), wenn es weniger als 3 ms beträgt, wird die Geschwindigkeitsbegrenzung des Schrittmotors überschritten und der Motor kann sich nicht drehen.

```

void moveOnePeriod(int dir,int ms){
    int i=0,j=0;
    for (j=0;j<4;j++){
        for (i=0;i<4;i++){
            if(dir == 1)
                digitalWrite(motorPins[i],(CCWStep[j] == (1<<i)) ? HIGH :
LOW);
            else

```

```

        digitalWrite(motorPins[i],(CWStep[j] == (1<<i)) ? HIGH :
LOW);
        printf("motorPin %d, %d \n",motorPins[i],digitalRead(motorPins[i]));
    }
    if(ms<3)
        ms=3;
    delay(ms);
}
}

```

Die Unterfunktion "moveSteps (int dir, int ms, int schritte)" dienen zum Drehen des Schrittmotors.

```

void moveSteps(int dir, int ms, int steps){
    int i;
    for(i=0;i<steps;i++){
        moveOnePeriod(dir,ms);
    }
}

```

Drehen Sie schließlich im while-Zyklus der Hauptfunktion einen Kreis im Uhrzeigersinn und dann einen Kreis gegen den Uhrzeigersinn. Nach dem bisherigen Wissen über den Schrittmotor kann bekannt sein, dass die Schrittmotordrehung für einen Kreis 2048 Schritte erfordert, dh $2048/4 = 512$ Zyklen.

```

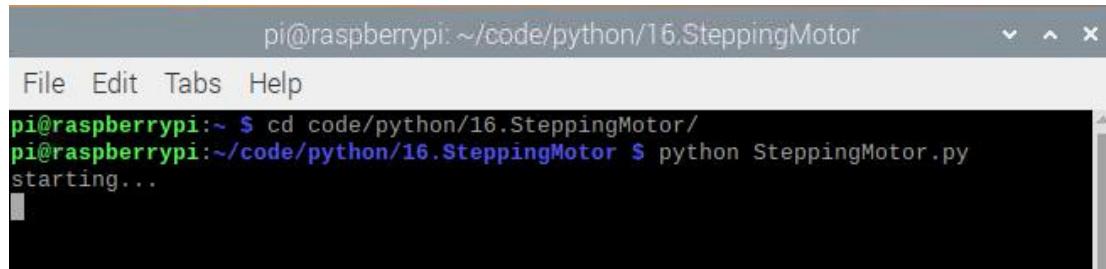
while(1){
    moveSteps(1,3,512);
    delay(1000);
    moveSteps(0,3,512);
    delay(1000);
}

```

Python code

1. Verwenden Sie den Befehl "`cd code / python / 16.SteppingMotor /`", um das Verzeichnis des SteppingMotor-Codes einzugeben.

2. Verwenden Sie den Befehl "[python SteppingMotor.py](#)", um den Code "SteppingMotor.py" auszuführen.



A screenshot of a terminal window titled "pi@raspberrypi: ~ /code/python/16.StepMotor". The window shows the command "cd code/python/16.StepMotor/" followed by "python SteppingMotor.py" and the output "starting...". The terminal has a standard blue header and a black body with white text.

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import time

motorpins = (12,16,18,22)      #define pins connected to four phase ABCD of stepper
motor
Reverse = (0x01,0x02,0x04,0x08) #define power supply order for coil for rotating
anticlockwise
Forward = (0x08,0x04,0x02,0x01) #define power supply order for coil for rotating
clockwise

def setup():
    print("starting...")
    GPIO.setmode(GPIO.BOARD)
    for pin in motorpins:
        GPIO.setup(pin,GPIO.OUT)

def motorang(dir,ms):
    for j in range(0,4,1):
        for i in range(0,4,1):
            if (dir == 1):
                GPIO.output(motorpins[i],((Reverse[j] == 1<<i) and GPIO.HIGH
or GPIO.LOW))
            else:
                GPIO.output(motorpins[i],((Forward[j] == 1<<i) and
GPIO.HIGH or GPIO.LOW))
    if(ms<3):
        ms = 3
```

```

time.sleep(ms*0.001)

def motordirang(dir,ms,steps):
    for i in range(steps):
        motorang(dir,ms)

def loop():
    while True:
        motordirang(1,3,512)
        time.sleep(0.5)
        motordirang(0,3,512)
        time.sleep(0.5)

def destroy():
    GPIO.cleanup()

if __name__ == '__main__':
    # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt: # When 'Ctrl+C' is pressed, the child program
destroy() will be executed.
    destroy()

```

Code Interpretation

Definieren Sie im Code vier Pins des Schrittmotors und die Reihenfolge der Spulenstromversorgung im vierstufigen Rotationsmodus.

```

motorpins = (12,16,18,22)      #define pins connected to four phase ABCD of stepper
motor
Reverse = (0x01,0x02,0x04,0x08) #define power supply order for coil for rotating
anticlockwise
Forward = (0x08,0x04,0x02,0x01) #define power supply order for coil for rotating
clockwise

```

Die Unterfunktion "motorang(dir,ms)" treibt den Schrittmotor vier Schritte im Uhrzeigersinn oder im Uhrzeigersinn, vier Schritte als Zyklus an. Wenn der Parameter "dir" die Drehrichtung angibt, dreht sich das Servo vorwärts, wenn "dir" 1 ist. Andernfalls dreht es sich in die umgekehrte Richtung. Der Parameter "ms" gibt die Zeit zwischen zwei Schritten an. Die "ms" des in diesem Projekt verwendeten Schrittmotors beträgt 3 ms (die kürzeste Zeit), wenn es weniger als 3 ms beträgt, wird

die Geschwindigkeitsbegrenzung des Schrittmotors überschritten und der Motor kann sich nicht drehen.

```
def motorang(dir,ms):
    for j in range(0,4,1):
        for i in range(0,4,1):
            if (dir == 1):
                GPIO.output(motorpins[i],((Reverse[j] == 1<<i) and GPIO.HIGH
or GPIO.LOW))
            else:
                GPIO.output(motorpins[i],((Forward[j] == 1<<i) and
GPIO.HIGH or GPIO.LOW))
            if(ms<3):
                ms = 3
            time.sleep(ms*0.001)
```

Die Unterfunktion motordirang (dir, ms, Schritte) wird für die Häufigkeit des Schrittmotors verwendet und dient zur Steuerung des Drehwinkels des Schrittmotors.

```
def motordirang(dir,ms,steps):
    for i in range(steps):
        motorang(dir,ms)
```

Rufen Sie in der 'while'-Schleife der Hauptfunktion' loop () 'zuerst die Funktion' motordirang (1,3,512) 'auf, um sie im Uhrzeigersinn zu drehen, rufen Sie die function'time.sleep (0,5)' auf, um 0,5 m zu stoppen, und rufen Sie dann die Funktion 'auf. Motordirang (0,3,512) 'einmal gegen den Uhrzeigersinn drehen und dann die Funktion'time.sleep (0.5)' aufrufen, um 0,5 m anzuhalten. Nach dem Verständnis von Schrittmotoren kann bekannt sein, dass eine Umdrehung von Schrittmotoren 2048 Schritte erfordert, dh 2048/4 = 512 Zyklen.

```
def loop():
    while True:
        motordirang(1,3,512)
        time.sleep(0.5)
        motordirang(0,3,512)
        time.sleep(0.5)
```

Lektion 17 74HC595 & LEDBar Graph

Überblick

In dieser Lektion lernen wir, wie Sie mit dem 74HC595 Chip die Raspberry pi Pins erweitern, um acht LED-Leuchten zu beleuchten und sie in Form von Lauflichtern zu präsentieren.

Obwohl wir acht LED direkt an die Raspberry Pi Entwicklungsplatine anschließen können, werden die GPIO Port Ressourcen der Entwicklungsplatine schnell verbraucht. Mehr GPIO-Ports bedeuten, dass mehr Peripheriegeräte an den Raspberry Pi angeschlossen werden können, sodass GPIO Ressourcen sehr wertvoll sind.

Verwenden Sie einen 74HC595 Seriell Parallel Wandlerchip, der viele Pin Ressourcen spart. Der Chip verfügt über acht Ausgänge und drei Eingänge, mit denen Sie Daten einzeln eingeben können.

Der Chip macht das Fahren von LED etwas langsamer, etwa 500.000 Mal pro Sekunde, aber er ist immer noch schneller als Menschen sehen können, sodass sich der Effekt nach dem Gebrauch nicht wesentlich ändert.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-type T-Typ Erweiterungskarte und Jumper Kabel

1 x Steckbrett

1 x 74HC595

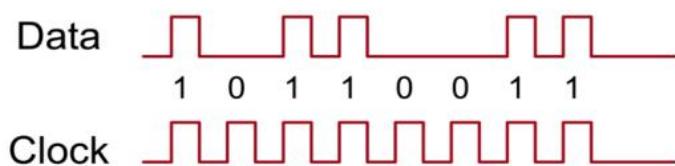
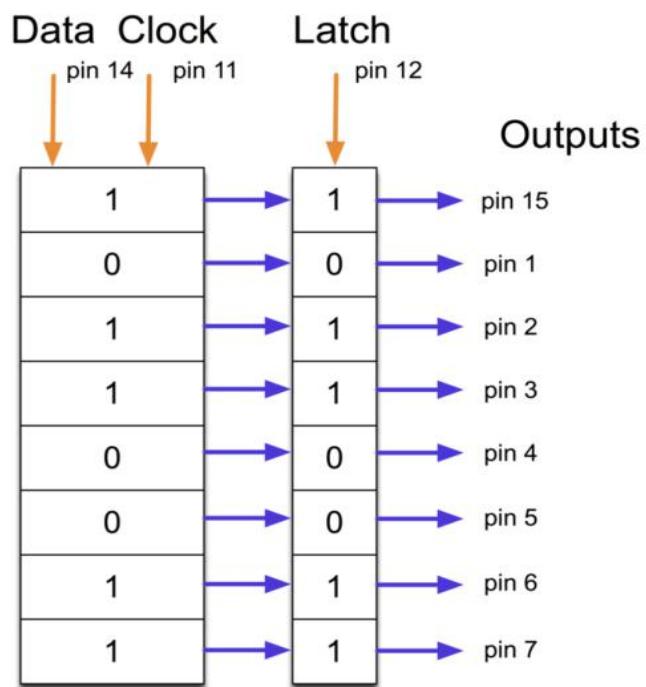
8 x 220Ω Widerstand

8 x LED

Produkt Einführung

74HC595

Das Schieberegister ist ein Chip, der als 8 Speicherzellen betrachtet werden kann, von denen jede 1 oder 0 sein kann. Um diese Werte ein- oder auszuschalten, verwenden wir den 'Data'- und den 'Clock' Pin des Eingangsdatenchips.



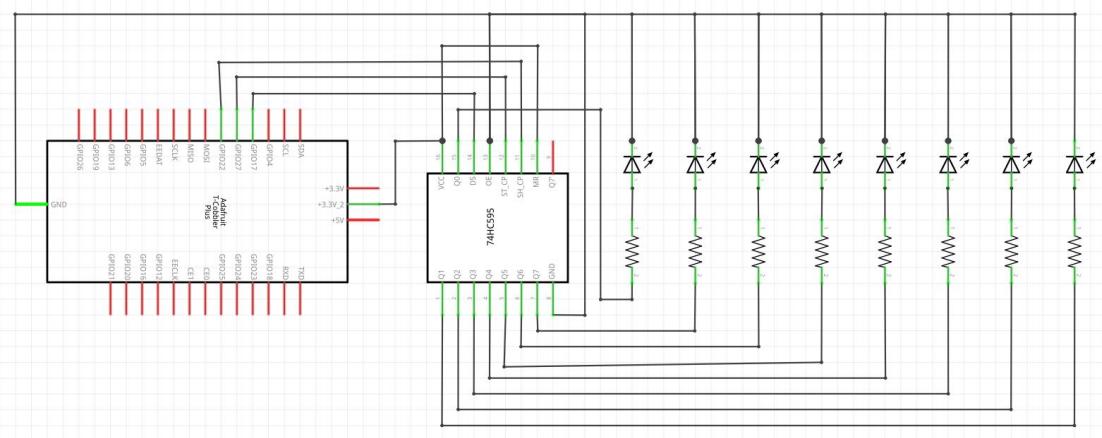
Der Clock-Pin muss 8 Impulse empfangen. Wenn der Datenpin in jedem Impuls hoch ist, wird 1 in das Schieberegister geschoben; sonst ist es 0. Wenn alle 8 Impulse empfangen wurden, werden diese 8 Werte durch Aktivieren des "Latch" -Pins in das Latch-Register kopiert. Das ist notwendig. Andernfalls blinkt die LED nicht richtig, wenn die Daten in das Schieberegister geladen werden.

Der Chip hat auch einen Output Enable (OE) -Pin, der alle Ausgänge aktiviert oder deaktiviert. Sie können es an den UNO R3-Pin anschließen, das unterstützt PWM und verwenden Sie "analogWrite", um die Helligkeit der LED zu steuern. Dieser Pin ist aktiv, also verbinden wir ihn mit GND.

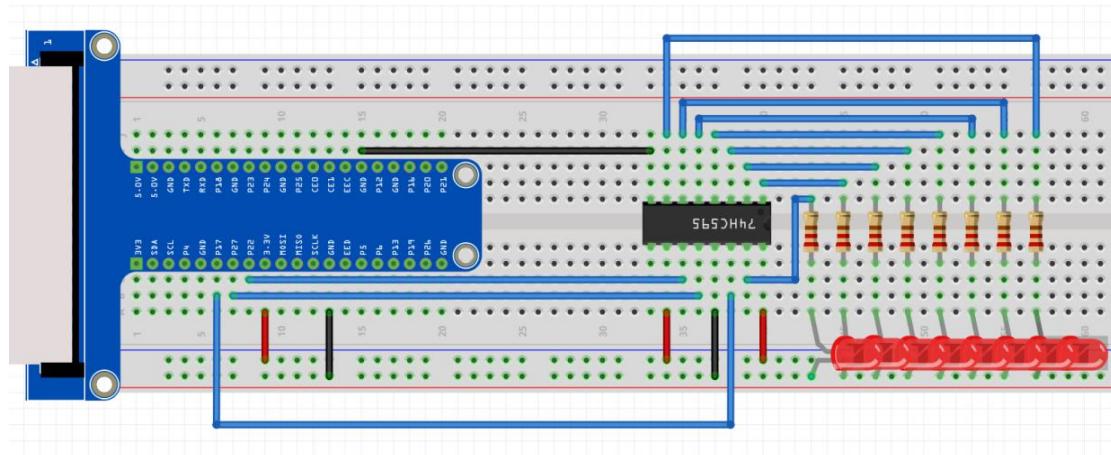
Die Ports des 74HC595 werden wie folgt beschrieben:

Pin	Pin No.	Beschreibung
Q0-Q7	15,1-7	Parallele Datenausgabe.
VCC	16	Die positive Elektrode der Stromversorgung, die Spannung beträgt 2 ~ 6V.
GND	8	Die negative Elektrode der Stromversorgung.
DS	14	Serielle Dateneingabe.
OE	13	Ausgang aktivieren, Wenn sich dieser Pin auf einem hohen Pegel befindet, befindet sich Q0-Q7 im Zustand mit hohem Widerstand. Wenn sich dieser Pin auf einem niedrigen Pegel befindet, befindet sich Q0-Q7 im Ausgangsmodus.
ST_CP	12	Parallele Aktualisierungsausgabe: Wenn der Pegel ansteigt, wird die parallele Datenausgabe aktualisiert.
SH_CP	11	Serieller Schiebetakt: Wenn der elektrische Pegel ansteigt, führt das serielle Dateneingangsregister eine Verschiebung durch.
MR	10	Schieberegister entfernen: Wenn sich dieser Pin auf einem niedrigen Pegel befindet, wird der Inhalt im Schieberegister.
Q7'	9	Serielle Datenausgabe: Kann angeschlossen.

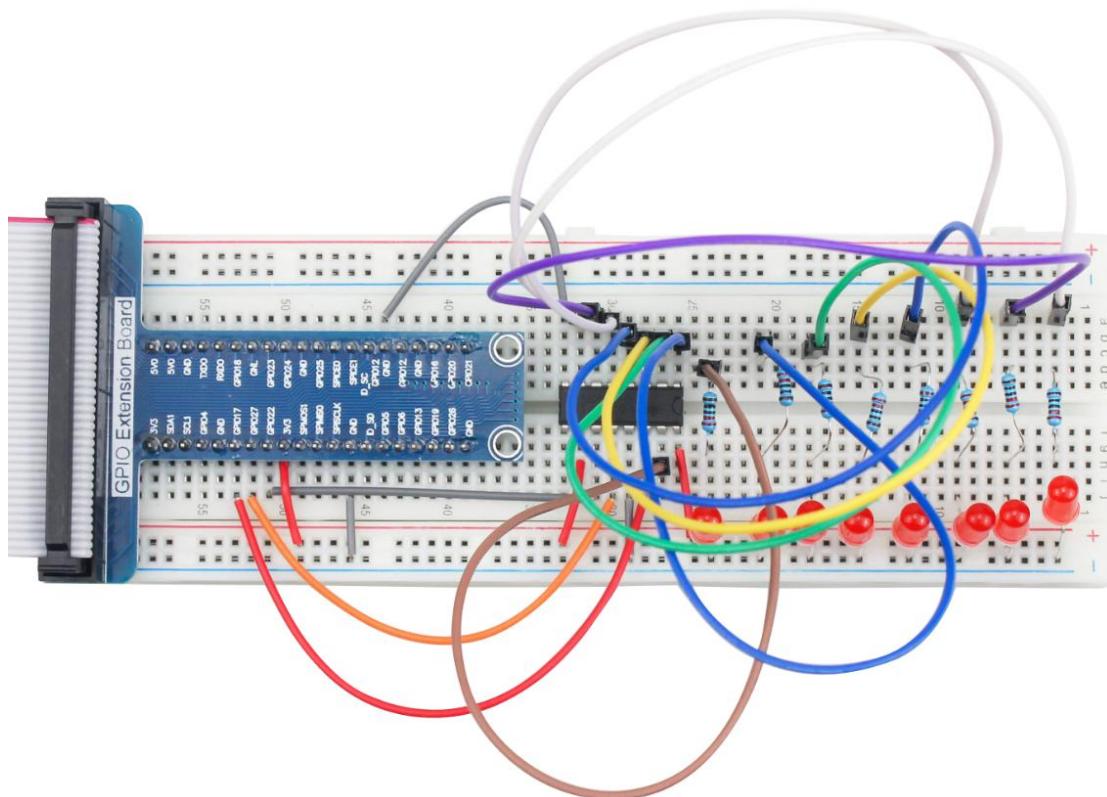
Schaltplan



Verdrahtung Diagramm



Beispiel Abbildung



C code

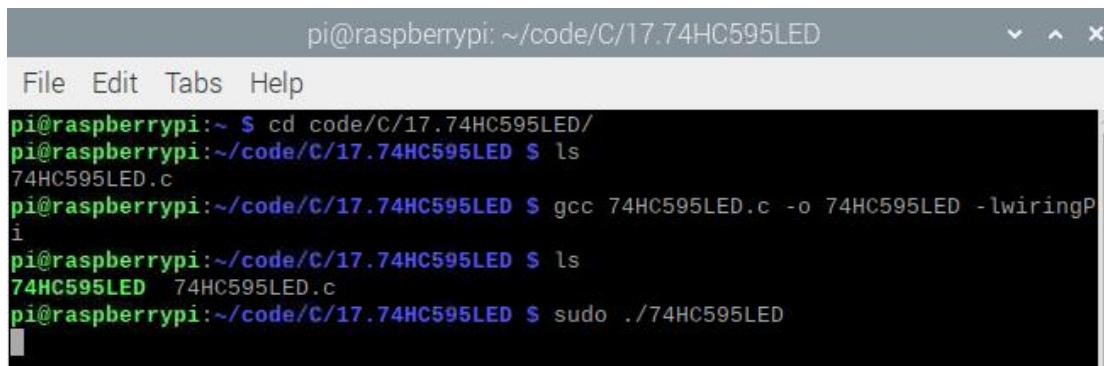
Öffnen Sie das Terminal und geben Sie den Befehl "cd code / C / 17.74HC595LED /" ein, um das Codeverzeichnis "74HC595LED" aufzurufen.

Geben Sie den Befehl "ls" ein, um die Datei "74HC595LED.c" im Verzeichnis anzuzeigen.

```
pi@raspberrypi: ~/code/C/17.74HC595LED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/17.74HC595LED/
pi@raspberrypi:~/code/C/17.74HC595LED $ ls
74HC595LED.c
pi@raspberrypi:~/code/C/17.74HC595LED $ █
```

Geben Sie den Befehl "gcc 74HC595LED.c -o 74HC595LED -lwiringPi" ein, um die ausführbare Datei "74HC595LED.c" "74HC595LED" zu generieren, und geben Sie

den Befehl "ls" ein, um ihn anzuzeigen. Geben Sie den Befehl "sudo ./74HC595LED" ein, um den Code auszuführen. Die Ergebnisse werden angezeigt:



```
pi@raspberrypi:~/code/C/17.74HC595LED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/17.74HC595LED/
pi@raspberrypi:~/code/C/17.74HC595LED $ ls
74HC595LED.c
pi@raspberrypi:~/code/C/17.74HC595LED $ gcc 74HC595LED.c -o 74HC595LED -lwiringPi
pi@raspberrypi:~/code/C/17.74HC595LED $ ls
74HC595LED  74HC595LED.c
pi@raspberrypi:~/code/C/17.74HC595LED $ sudo ./74HC595LED
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>
#include <wiringShift.h>

#define    dataPin    0    //DS Pin of 74HC595(Pin14)
#define    latchPin   2    //ST_CP Pin of 74HC595(Pin12)
#define    clockPin  3    //CH_CP Pin of 74HC595(Pin11)

void sendOut(int dPin,int cPin,int order,int val){
    int i;
    for(i = 0; i < 8; i++){
        digitalWrite(cPin,LOW);
        if(order == LSBFIRST){
            digitalWrite(dPin,((0x01&(val>>i)) == 0x01) ? HIGH : LOW);
            delayMicroseconds(10);
        }
        else {
            digitalWrite(dPin,((0x80&(val<<i)) == 0x80) ? HIGH : LOW);
            delayMicroseconds(10);
        }
        digitalWrite(cPin,HIGH);
        delayMicroseconds(10);
    }
}

int main(void)
{
```

```

int i;
unsigned char x;
if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
    printf("setup wiringPi failed !");
    return 1;
}
pinMode(dataPin,OUTPUT);
pinMode(latchPin,OUTPUT);
pinMode(clockPin,OUTPUT);
while(1){
    x=0x01;
    for(i=0;i<8;i++){
        digitalWrite(latchPin,LOW);      // Output low level to latchPin
        sendOut(dataPin,clockPin,LSBFIRST,x);// Send serial data to 74HC595
        digitalWrite(latchPin,HIGH); // Output high level to latchPin, and
74HC595 will update the data to the parallel output port.
        x<<=1; // make the variable move one bit to left once, then the bright
LED move one step to the left once.
        delay(100);
    }
    x=0x80;
    delay(500);
    for(i=0;i<8;i++){
        digitalWrite(latchPin,LOW);
        sendOut(dataPin,clockPin,LSBFIRST,x);
        digitalWrite(latchPin,HIGH);
        x>>=1;
        delay(100);
    }
    delay(500);
}
return 0;
}

```

Code Interpretation

Im Code definieren wir die Funktion "sendout (dPin, cPin, order, val)", mit der Werte in der angegebenen Reihenfolge ausgegeben werden. "dPin" repräsentiert den

Daten-Pin und "cPin" repräsentiert den Takt in der Reihenfolge von hoch nach niedrig oder niedrig nach hoch. Diese Funktion entspricht der Betriebsart 74HC595.

```
void sendOut(int dPin,int cPin,int order,int val){
    int i;
    for(i = 0; i < 8; i++){
        digitalWrite(cPin,LOW);
        if(order == LSBFIRST){
            digitalWrite(dPin,((0x01&(val>>i)) == 0x01) ? HIGH : LOW);
            delayMicroseconds(10);
        }
        else {
            digitalWrite(dPin,((0x80&(val<<i)) == 0x80) ? HIGH : LOW);
            delayMicroseconds(10);
        }
        digitalWrite(cPin,HIGH);
        delayMicroseconds(10);
    }
}
```

Im Code konfigurieren wir drei Pins zur Steuerung des 74HC595. Definieren Sie eine Ein-Byte-Variable, um den Status von 8 LEDs über die 8 Bits der Variablen zu steuern. Die LED leuchtet, wenn das entsprechende Bit 1 ist. Wenn die Variable 0x01 zugewiesen ist, dh 00000001 in Binärform, leuchtet nur eine LED.

x=0x01;

Verwenden Sie im "while" Zyklus der Hauptfunktion den "for" Zyklus, um x an den 74HC595-Ausgangspin zu senden und die LED zu steuern. Im "for" Zyklus wird x in einem Zyklus um ein Bit nach links verschoben. In der nächsten Runde, wenn Daten von x an 74HC595 gesendet werden, bewegt sich die eingeschaltete LED einmal um ein Bit nach links.

```
while(1){
    x=0x01;
    for(i=0;i<8;i++){
        digitalWrite(latchPin,LOW);
        sendOut(dataPin,clockPin,LSBFIRST,x);
        digitalWrite(latchPin,HIGH);
        x<<=1;
        delay(100);
    }
    x=0x80;
```

```

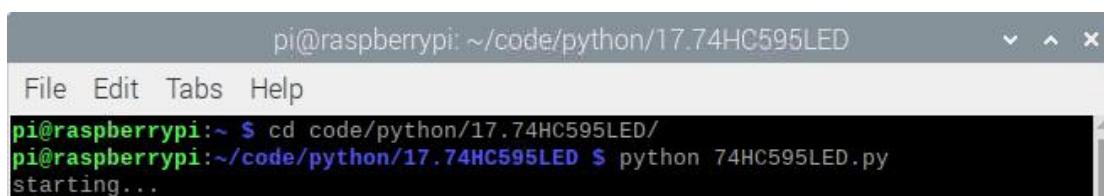
delay(500);
for(i=0;i<8;i++){
    digitalWrite(latchPin,LOW);
    sendOut(dataPin,clockPin,LSBFIRST,x);
    digitalWrite(latchPin,HIGH);
    x>>=1;
    delay(100);
}
delay(500);

}

```

Python code

1. Verwenden Sie den Befehl "[cd code / python / 17.74HC595LED /](#)", um das Verzeichnis "74HC595LED" einzugeben.
2. Verwenden Sie den Befehl "[python 74HC95LED.py](#)", um den Code "[python 74HC95LED.py](#)" auszuführen.



The screenshot shows a terminal window titled "pi@raspberrypi: ~ /code/python/17.74HC595LED". The menu bar includes "File", "Edit", "Tabs", and "Help". The terminal command history shows:

```

pi@raspberrypi:~ $ cd code/python/17.74HC595LED/
pi@raspberrypi:~/code/python/17.74HC595LED $ python 74HC595LED.py
starting...

```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```

import RPi.GPIO as GPIO
import time

LWAY = 1
MWAY = 2

dataPin = 11          #DS Pin of 74HC595(Pin14)
latchPin = 13         #ST_CP Pin of 74HC595(Pin12)
clockPin = 15         #CH_CP Pin of 74HC595(Pin11)

def setup():

```

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(dataPin,GPIO.OUT)
GPIO.setup(latchPin,GPIO.OUT)
GPIO.setup(clockPin,GPIO.OUT)

def sendout(dPin,cPin,way,val):
    for i in range(0,8):
        GPIO.output(cPin,GPIO.LOW)
        if(way == LWAY):
            GPIO.output(dPin,(0x01 & (val>>i)==0x01) and GPIO.HIGH or
GPIO.LOW)
        elif(way == MWAY):
            GPIO.output(dPin,(0x80 & (val<<i)==0x80) and GPIO.HIGH or
GPIO.LOW)
        GPIO.output(cPin,GPIO.HIGH)

def loop():
    while True:
        x=0x01
        for i in range(0,8):
            GPIO.output(latchPin,GPIO.LOW)      #Output low level to
latchPin
            sendout(dataPin,clockPin,LWAY,x)  #Send serial data to 74HC595
            GPIO.output(latchPin,GPIO.HIGH)    #Output high level to
latchPin, and 74HC595 will update the data to the parallel output port.
            x<<=1                         # make the variable move one bit to left
once, then the bright LED move one step to the left once.
            time.sleep(0.1)
        x=0x80
        time.sleep(0.5)
        for i in range(0,8):
            GPIO.output(latchPin,GPIO.LOW)      #Output low level to
latchPin
            sendout(dataPin,clockPin,LWAY,x)  #Send serial data to 74HC595
            GPIO.output(latchPin,GPIO.HIGH)    #Output high level to
latchPin, and 74HC595 will update the data to the parallel output port.
            x>>=1                         # make the variable move one bit to left
once, then the bright LED move one step to the left once.
            time.sleep(0.1)
        time.sleep(0.5)
```

```

def destroy():      # When 'Ctrl+C' is pressed, the function is executed.
    GPIO.cleanup()

if __name__ == '__main__':  # Program starting from here
    print("starting...")
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()

```

Code Interpretation

Im Code definieren wir die sendout(dPin,cPin,way,val), mit der Werte nacheinander ausgegeben werden. "DPin" bedeutet Daten-Pin, "cPin" bedeutet Takt, die Reihenfolge ist von hoch nach niedrig oder von niedrig nach hoch. Diese Funktion entspricht der Betriebsart 74HC595. "LWAY" und "MWAY" sind zwei verschiedene Richtungen.

```

def sendout(dPin,cPin,way,val):
    for i in range(0,8):
        GPIO.output(cPin,GPIO.LOW)
        if(way == LWAY):
            GPIO.output(dPin,(0x01 & (val>>i)==0x01) and GPIO.HIGH or
GPIO.LOW)
        elif(way == MWAY):
            GPIO.output(dPin,(0x80 & (val<<i)==0x80) and GPIO.HIGH or
GPIO.LOW)
        GPIO.output(cPin,GPIO.HIGH)

```

In der Funktion loop () verwenden wir zwei "for" Zyklen, um das Ziel zu erreichen. Definieren Sie zunächst eine Variable "x = 0x01", binary00000001. Wenn es an den Ausgangsport des 74HC595 übertragen wird, gibt das niedrige Bit einen hohen Pegel aus, und eine LED leuchtet auf. Als nächstes wird x um ein Bit verschoben. Wenn x erneut an den Ausgangsport des 74HC595 übertragen wird, wird die eingeschaltete LED verschoben. Wiederholen Sie den Vorgang, der Effekt des fließenden Wasserlichts wird gebildet. Wenn die Richtung der Verschiebungsoperation für x unterschiedlich ist, ist die Strömungsrichtung unterschiedlich.

```
def loop():
```

```
while True:  
    x=0x01  
    for i in range(0,8):  
        GPIO.output(latchPin,GPIO.LOW)      #Output low level to  
latchPin  
        sendout(dataPin,clockPin,LWAY,x)    #Send serial data to 74HC595  
        GPIO.output(latchPin,GPIO.HIGH)      #Output high level to  
latchPin, and 74HC595 will update the data to the parallel output port.  
        x<<=1                          # make the variable move one bit to left  
once, then the bright LED move one step to the left once.  
        time.sleep(0.1)  
    x=0x80  
    time.sleep(0.5)  
    for i in range(0,8):  
        GPIO.output(latchPin,GPIO.LOW)      #Output low level to  
latchPin  
        sendout(dataPin,clockPin,LWAY,x)    #Send serial data to 74HC595  
        GPIO.output(latchPin,GPIO.HIGH)      #Output high level to  
latchPin, and 74HC595 will update the data to the parallel output port.  
        x>>=1                          # make the variable move one bit to left  
once, then the bright LED move one step to the left once.  
        time.sleep(0.1)  
    time.sleep(0.5)
```

Lektion 18 74HC595 & 7-Segment-Anzeige

Überblick

In diesem Kurs lernen Sie, wie Sie mit 74HC595 die 7-Segment-Anzeige steuern. und lassen Sie es sechzehn Dezimalzeichen "0-F" anzeigen.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-Erweiterungskarte und Kabel

1 x Steckbrett

1 x 74HC595

1 x 220Ω Widerstand

1 x 7 Segment Anzeige

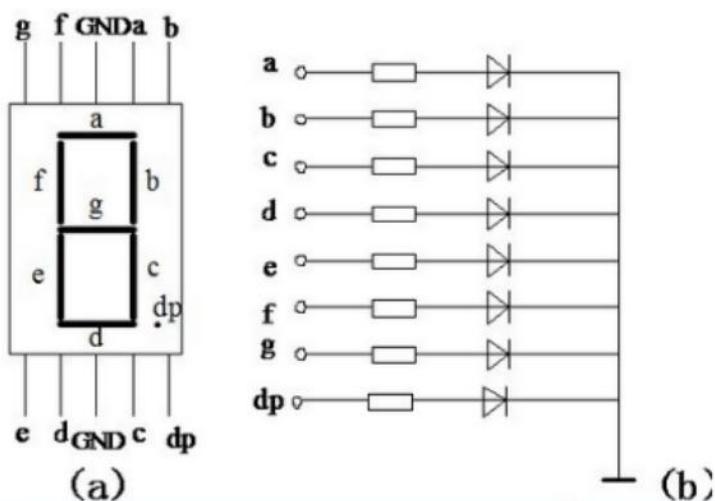
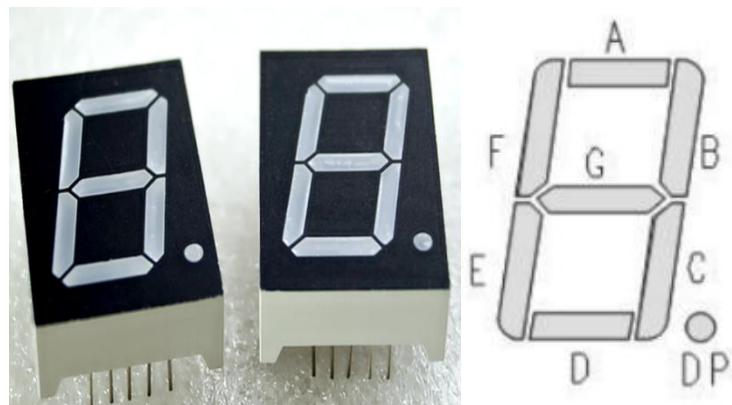
Produkt Einführung

7 Segment Anzeige

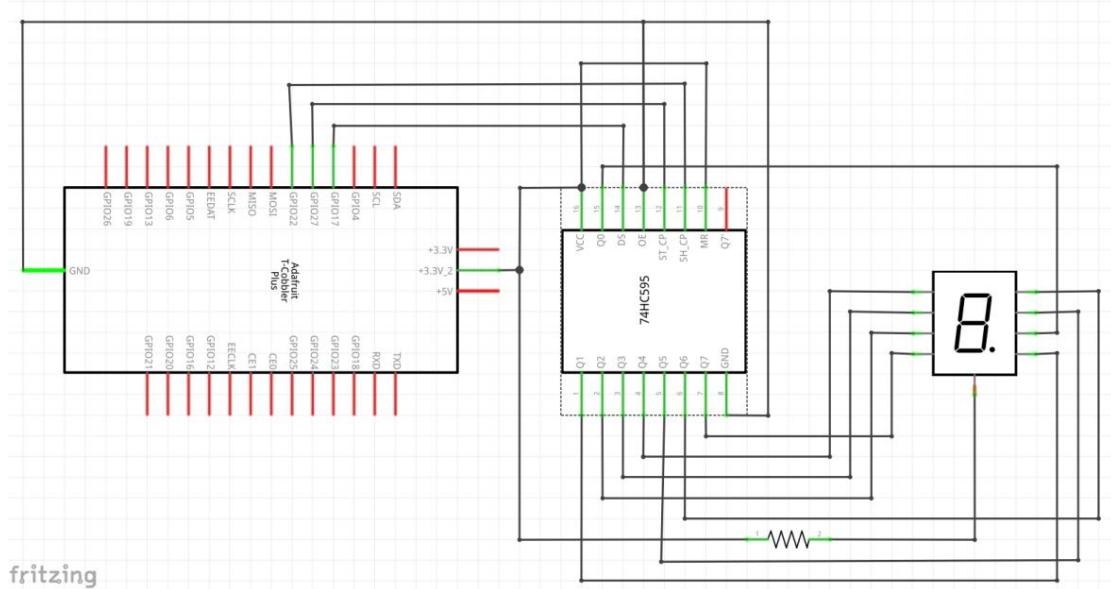
Die digitale Segment LED-Anzeige Tube ist durch eine Vielzahl von LEDs zusammengepackt, um eine "8" -förmige Komponente zu bilden. Diese Segmente werden durch die Buchstaben a, b, c, d, e, f, g bzw. dp dargestellt. Wenn an ein bestimmtes Segment der digitalen Röhre eine Spannung angelegt wird, leuchten diese bestimmten Segmente auf, um das zu bilden, was wir in unseren Augen sehen. Wenn Sie beispielsweise eine Zahl "2" anzeigen, sollten Sie hell a, b, g, e, d beleuchten und nicht hell f, c, dp.

LED-Anzeige Tube sind im Allgemeinen hell und superhell und haben auch unterschiedliche Größen wie 0,5 Zoll und 1 Zoll. Die Anzeige einer kleinen digitalen Röhre besteht normalerweise aus einer Leuchtdiode, und die große digitale Röhre besteht aus zwei oder mehr Leuchtdioden. Im Allgemeinen beträgt die Spannung einer einzelnen LED etwa 1,8 V und der Strom überschreitet 30 mA nicht.

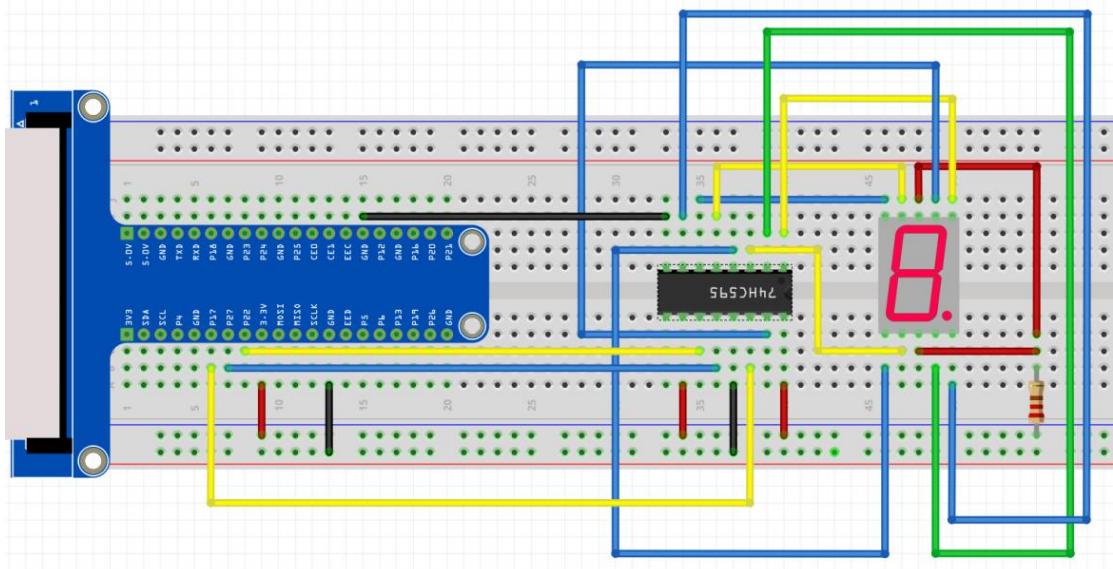
Die Anoden aller Leuchtdioden werden miteinander verbunden und dann mit dem positiven Pol der Stromversorgung verbunden, der sogenannten Digitalröhre mit gemeinsamer Anode. Die Kathoden aller Leuchtdioden sind miteinander verbunden und werden dann als gemeinsame Kathodendigitalröhre mit dem Minuspol der Stromquelle verbunden. Die von der häufig verwendeten digitalen LED-Röhre angezeigten Zahlen und Zeichen sind 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. In dieser Lektion verwenden wir eine einstellige gemeinsame Anode digitale Segment-LED-Anzeige Tube.



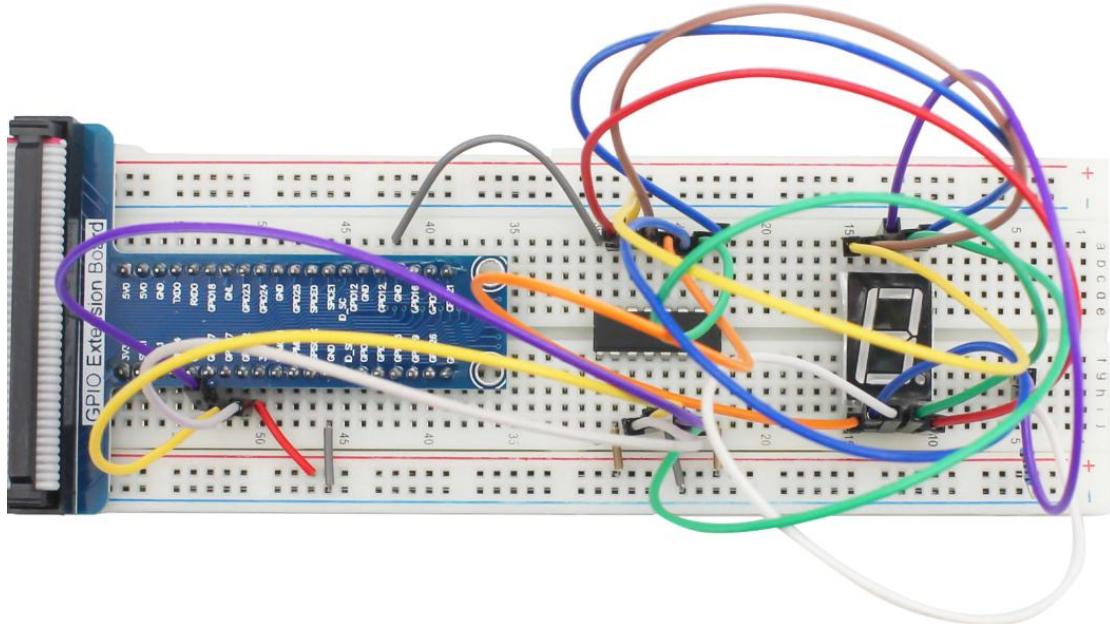
Schaltplan



Verdrahtung Diagramm



Beispielbild



C code

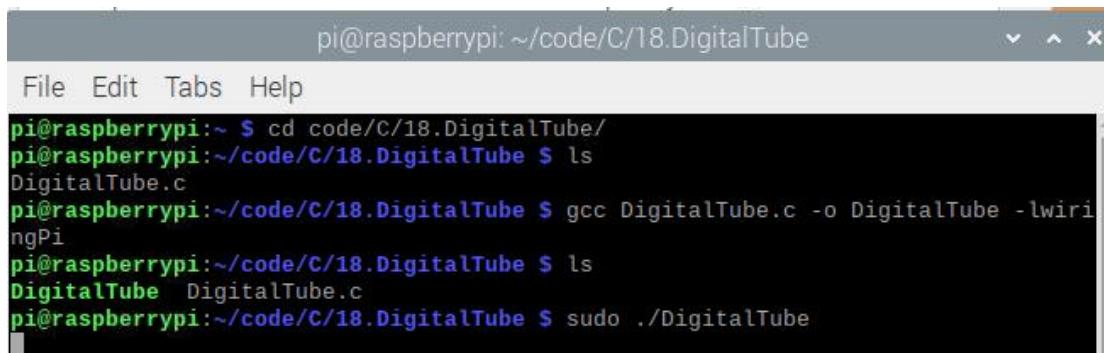
Öffnen Sie das Terminal und geben Sie den Befehl "[cd code / C / 18.DigitalTube /](#)" ein, um das Codeverzeichnis "DigitalTube" aufzurufen;

Geben Sie den Befehl "[ls](#)" ein, um die Datei "DigitalTube.c" im Verzeichnis anzuzeigen;

```
pi@raspberrypi: ~/code/C/18.DigitalTube
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/18.DigitalTube/
pi@raspberrypi:~/code/C/18.DigitalTube $ ls
DigitalTube.c
pi@raspberrypi:~/code/C/18.DigitalTube $
```

Geben Sie den Befehl "[gcc DigitalTube.c -o DigitalTube -lwiringPi](#)" ein, um die ausführbare Datei "DigitalTube" "DigitalTube.c" zu generieren, und geben Sie den Befehl "[ls](#)" ein, um ihn anzuzeigen. Geben Sie den Befehl "[sudo ./DigitalTube](#)" ein, um den Code auszuführen.

Die Ergebnisse werden unten angezeigt:



```
pi@raspberrypi:~/code/C/18.DigitalTube
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/18.DigitalTube/
pi@raspberrypi:~/code/C/18.DigitalTube $ ls
DigitalTube.c
pi@raspberrypi:~/code/C/18.DigitalTube $ gcc DigitalTube.c -o DigitalTube -lwiringPi
pi@raspberrypi:~/code/C/18.DigitalTube $ ls
DigitalTube  DigitalTube.c
pi@raspberrypi:~/code/C/18.DigitalTube $ sudo ./DigitalTube
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>
#include <wiringShift.h>

#define    dataPin    0    //DS Pin of 74HC595(Pin14)
#define    latchPin   2    //ST_CP Pin of 74HC595(Pin12)
#define    clockPin  3    //CH_CP Pin of 74HC595(Pin11)
//encoding for character 0-F of common anode SevenSegmentDisplay.
unsigned char
num[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,0x83,0xc6,0xa1,
0x86,0x8e};

void sendOut(int dPin,int cPin,int order,int val){
    int i;
    for(i = 0; i < 8; i++){
        digitalWrite(cPin,LOW);
        if(order == LSBFIRST){
            digitalWrite(dPin,((0x01&(val>>i)) == 0x01) ? HIGH : LOW);
            delayMicroseconds(10);
        }
        else {
            digitalWrite(dPin,((0x80&(val<<i)) == 0x80) ? HIGH : LOW);
            delayMicroseconds(10);
        }
        digitalWrite(cPin,HIGH);
        delayMicroseconds(10);}
```

```

        }
    }

int main(void)
{
    int i;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("failed !");
        return 1;
    }
    pinMode(dataPin,OUTPUT);
    pinMode(latchPin,OUTPUT);
    pinMode(clockPin,OUTPUT);
    while(1){
        for(i=0;i<sizeof(num);i++){
            digitalWrite(latchPin,LOW);
            sendOut(dataPin,clockPin,MSBFIRST,num[i]);//Output the figures and
the highest level is transferred preferentially.
            digitalWrite(latchPin,HIGH);
            delay(500);
        }
        for(i=0;i<sizeof(num);i++){
            digitalWrite(latchPin,LOW);
            sendOut(dataPin,clockPin,MSBFIRST,num[i] & 0x7f);//Use the
"&0x7f" to display the decimal point.
            digitalWrite(latchPin,HIGH);
            delay(500);
        }
    }
    return 0;
}

```

Code Interpretation

Fügen Sie zuerst die Codierung von "0" - "F" in das Array "num" ein.

Unsigned char

```
num[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,0x83,0xc6,0xa1,
0x86,0x8e};
```

Im "for" Zyklus der Funktion loop () wird der Inhalt des Arrays "num" der Reihe nach ausgegeben. Die digitale Röhre kann die entsprechenden Zeichen korrekt anzeigen.

Beachten Sie, dass in der Funktion "sendOut (int dPin, int cPin, int order, int val)" wird das höchste Bit des Flagbits bevorzugt übertragen.

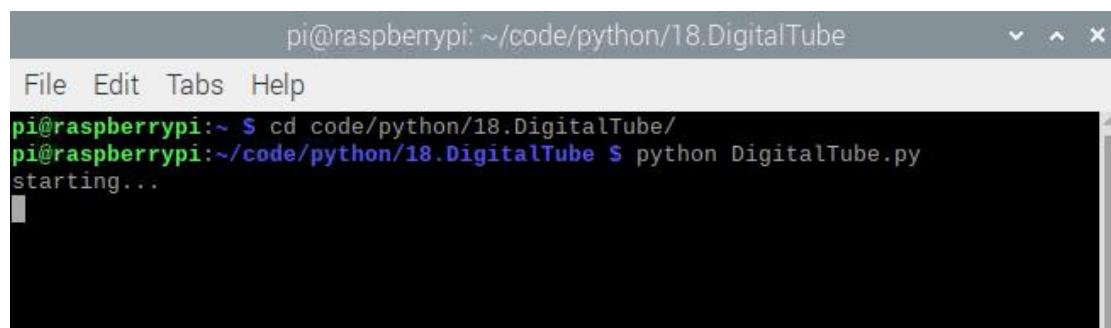
```
for(i=0;i<sizeof(num);i++){
    digitalWrite(latchPin,LOW);
    sendOut(dataPin,clockPin,MSBFIRST,num[i]);//Output the figures and
the highest level is transferred preferentially.
    digitalWrite(latchPin,HIGH);
    delay(500);
}
```

Wenn Sie den Dezimalpunkt anzeigen möchten, setzen Sie das höchste Bit jedes Arrays auf 0, was durch `num [i] & 0x7f` implementiert werden kann.

```
sendOut(dataPin,clockPin,MSBFIRST,num[i] & 0x7f);
```

Python code

1. Verwenden Sie den Befehl "`cd code / python / 18.DigitalTube /`", um das Verzeichnis von DigitalTube einzugeben.
2. Verwenden Sie den Befehl "`python DigitalTube.py`", um den Code "DigitalTube.py" auszuführen.



```
pi@raspberrypi: ~/code/python/18.DigitalTube
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/18.DigitalTube/
pi@raspberrypi:~/code/python/18.DigitalTube $ python DigitalTube.py
starting...
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import time
```

```

LWAY = 1
MWAY = 2
#define the pins connect to 74HC595
dataPin    = 11      #DS Pin of 74HC595(Pin14)
latchPin   = 13      #ST_CP Pin of 74HC595(Pin12)
clockPin   = 15      #CH_CP Pin of 74HC595(Pin11)

num =
[0xfc,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x
8e]

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(dataPin,GPIO.OUT)
    GPIO.setup(latchPin,GPIO.OUT)
    GPIO.setup(clockPin,GPIO.OUT)

def sendout(dPin,cPin,way,val):
    for i in range(0,8):
        GPIO.output(cPin,GPIO.LOW)
        if(way == LWAY):
            GPIO.output(dPin,(0x01 & (val>>i)==0x01) and GPIO.HIGH or
GPIO.LOW)
        elif(way == MWAY):
            GPIO.output(dPin,(0x80 & (val<<i)==0x80) and GPIO.HIGH or
GPIO.LOW)
        GPIO.output(cPin,GPIO.HIGH)

def loop():
    while True:
        for i in range(0,len(num)):
            GPIO.output(latchPin,GPIO.LOW)
            sendout(dataPin,clockPin,MWAY,num[i])  #Output the figures and
the highest level is transferred preferentially.
            GPIO.output(latchPin,GPIO.HIGH)
            time.sleep(0.5)
        for i in range(0,len(num)):
            GPIO.output(latchPin,GPIO.LOW)
            sendout(dataPin,clockPin,MWAY,num[i]&0x7f)  #Use "&0x7f"to
display the decimal point.
        GPIO.output(latchPin,GPIO.HIGH)

```

```

GPIO.output(latchPin,GPIO.HIGH)
time.sleep(0.5)

def destroy():      # When 'Ctrl+C' is pressed, the function is executed.
    GPIO.cleanup()

if __name__ == '__main__':
    print("starting...")
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()

```

Code Interpretation

Fügen Sie zuerst die Codierung von "0" - "F" in das Array "num" ein.

```

num =
[0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x
8e]

```

Im "for" Zyklus der Funktion loop () wird der Inhalt des Arrays "num" der Reihe nach ausgegeben. Die digitale Röhre kann die entsprechenden Zeichen korrekt anzeigen.

Beachten Sie, dass in der Funktion "sendOut(int dPin,int cPin,int order,int val)" wird das höchste Bit des Flagbits bevorzugt übertragen.

```

for i in range(0,len(num)):
    GPIO.output(latchPin,GPIO.LOW)
    sendout(dataPin,clockPin,MWAY,num[i])  #Output the figures and
the highest level is transferred preferentially.
    GPIO.output(latchPin,GPIO.HIGH)
    time.sleep(0.5)

```

Wenn Sie den Dezimalpunkt anzeigen möchten, setzen Sie das höchste Bit jedes Arrays auf 0, was durch num [i] & 0x7f implementiert werden kann.

```

sendout(dataPin,clockPin,MWAY,num[i]&0x7f)  #Use "&0x7f"to display the
decimal point.

```

Lesson 19 74HC595&4-Digit 7-Segment Display

Überblick

In dieser Lektion lernen Sie, wie Sie eine 4-bits 7 Segment Anzeige verwenden. Basierend auf der vorherigen Lektion werden wir die 1-bit 7 Segment Anzeige in eine 4-bits 7 Segment Anzeige ändern.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-Erweiterungskarte und Kabel

1 x Steckbrett

1 x 74HC595

1 x 220Ω Widerstand

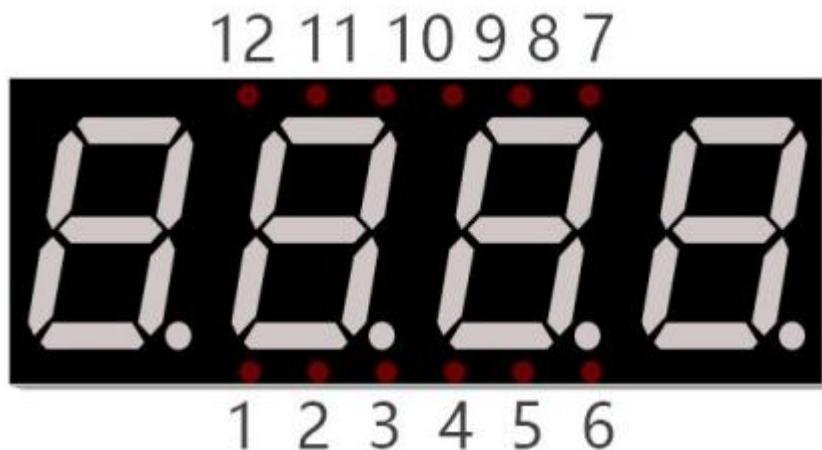
1 x 4-stellige 7-Segment-Anzeige

Produkt Einführung

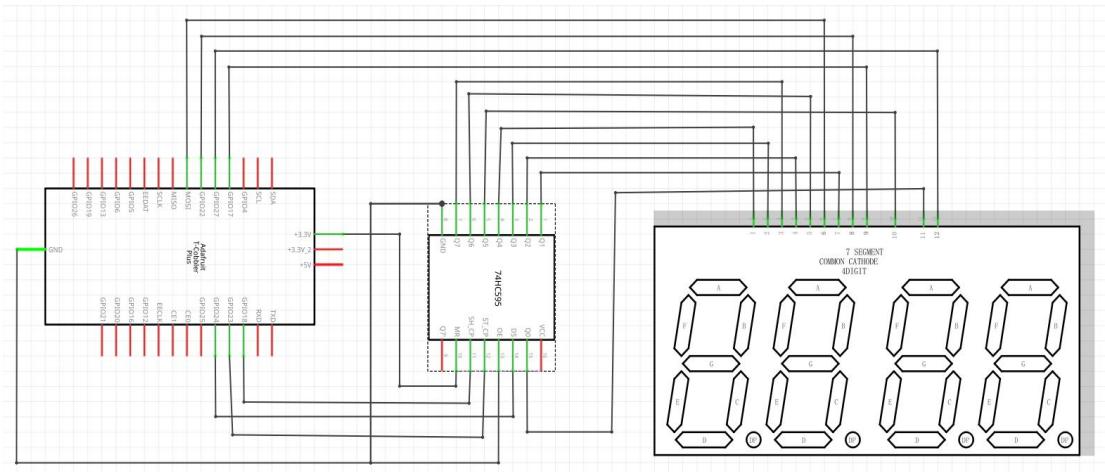
4-stellige 7-Segment-Anzeige

Die 4-stellige 7-Segment-Anzeige integriert vier 7-Segment-Anzeigen, sodass mehr Zahlen angezeigt werden können. Die Anzeigemethode der 4-stelligen 7-Segment-Anzeige ähnelt der 1-stelligen 7-Segment-Anzeige. Der Unterschied zwischen ihnen besteht darin, dass die 4-stellige Anzeige nacheinander nicht zusammen angezeigt wird. Senden Sie zuerst einen hohen Pegel an das gemeinsame Ende der ersten Röhre und dann einen niedrigen Pegel an den Rest der drei gemeinsamen Enden und senden Sie dann den Inhalt an 8 LED Kathodenstifte der ersten Röhre. Zu diesem Zeitpunkt zeigt die erste 7-Segment-Anzeige den Inhalt und die restlichen drei im geschlossenen Zustand an. Ähnlich zeigt das zweite, dritte und vierte 7-Segment den Inhalt der Reihe nach an, nämlich die Scan Anzeige. Obwohl die vier Zahlen der Reihe nach separat angezeigt werden, ist dieser Vorgang sehr schnell und aufgrund des optischen Nachglüheffekts und des Effekts der Sehpersistenz können alle vier Zahlen gleichzeitig angezeigt werden. Im Gegenteil, wenn jede Zahl für eine lange Zeit angezeigt wird, können Sie sehen, dass die Zahlen

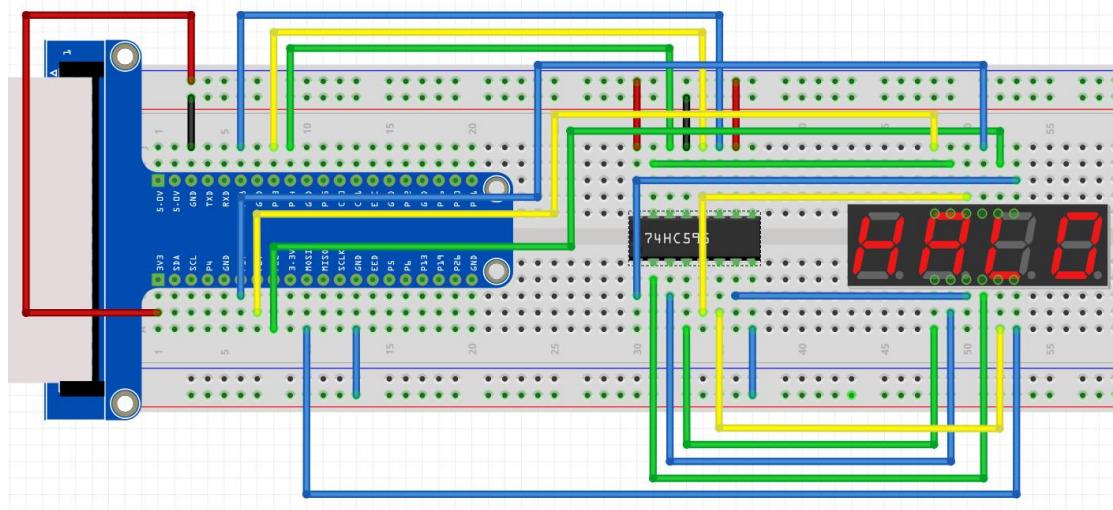
separat angezeigt werden. Entsprechend dem Unterschied zwischen gemeinsamer Kathode und Anode. Die interne Struktur und das Pin-Diagramm sind unten dargestellt:



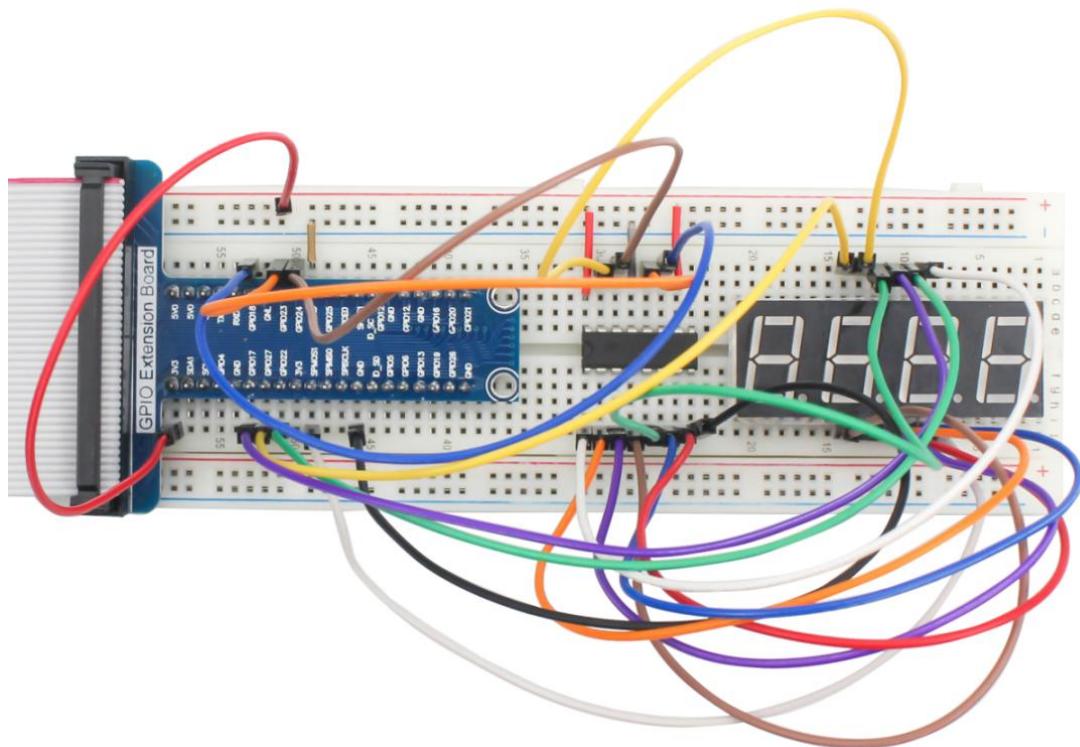
Schaltplan



Verdrahtung Diagramm



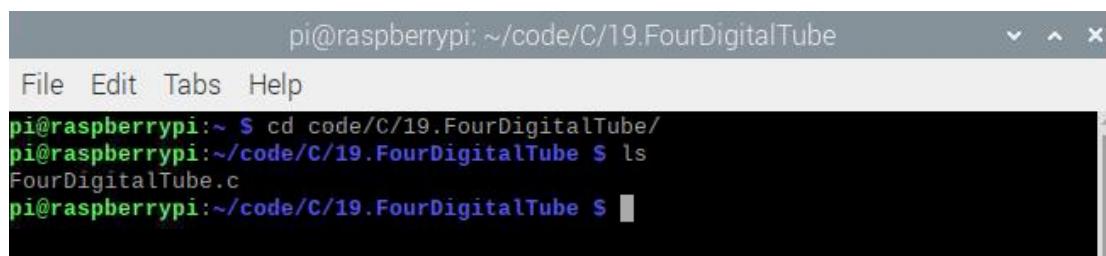
Beispiel Abbildung



C code

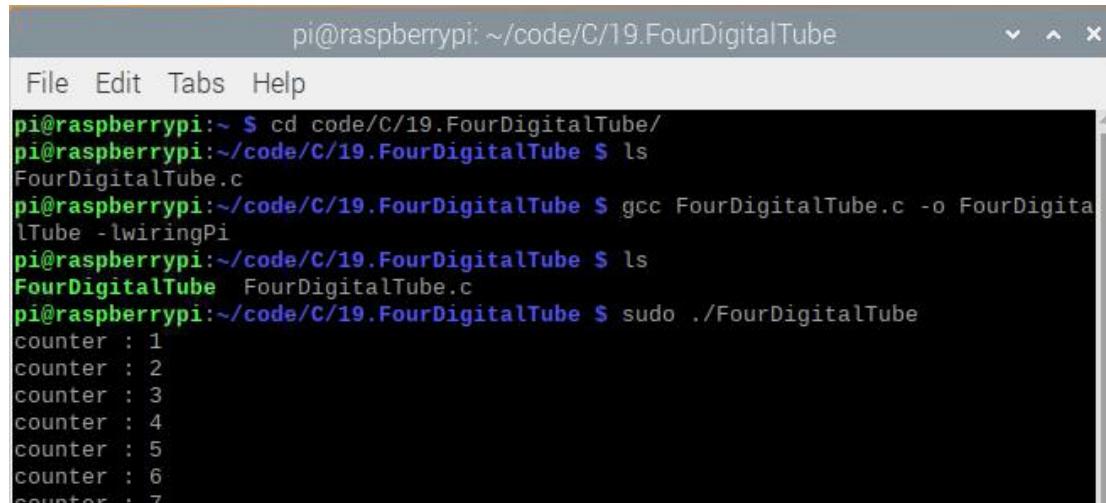
Öffnen Sie das Terminal und geben Sie den Befehl "[cd code / C / 19.FourDigitalTube /](#)" ein, um das Codeverzeichnis "FourDigitalTube" aufzurufen;

Geben Sie den Befehl "[ls](#)" ein, um die Datei "FourDigitalTube.c" im Verzeichnis anzuzeigen;



```
pi@raspberrypi:~/code/C/19.FourDigitalTube
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/19.FourDigitalTube/
pi@raspberrypi:~/code/C/19.FourDigitalTube $ ls
FourDigitalTube.c
pi@raspberrypi:~/code/C/19.FourDigitalTube $
```

Geben Sie den Befehl "[gcc FourDigitalTube.c -o FourDigitalTube -lwiringPi](#)" ein, um die ausführbare Datei "FourDigitalTube.c" "FourDigitalTube" zu generieren. Geben Sie den Befehl "ls" ein, um sie anzuzeigen. Geben Sie "[sudo ./FourDigitalTube](#)" command" ein, um den Code auszuführen. Die Ergebnisse werden unten angezeigt:



```
pi@raspberrypi:~/code/C/19.FourDigitalTube
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/19.FourDigitalTube/
pi@raspberrypi:~/code/C/19.FourDigitalTube $ ls
FourDigitalTube.c
pi@raspberrypi:~/code/C/19.FourDigitalTube $ gcc FourDigitalTube.c -o FourDigitalTube -lwiringPi
pi@raspberrypi:~/code/C/19.FourDigitalTube $ ls
FourDigitalTube FourDigitalTube.c
pi@raspberrypi:~/code/C/19.FourDigitalTube $ sudo ./FourDigitalTube
counter : 1
counter : 2
counter : 3
counter : 4
counter : 5
counter : 6
counter : 7
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>
#include <wiringShift.h>
#include <signal.h>
#include <unistd.h>
#define      dataPin      5      //DS Pin of 74HC595(Pin14)
```

```
#define      latchPin     4    //ST_CP Pin of 74HC595(Pin12)
#define      clockPin     1    //CH_CP Pin of 74HC595(Pin11)
const int digitPin[]={0,2,3,12};           // Define 7-segment display common pin
// character 0-9 code of common anode 7-segment display
unsigned char num[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
int counter = 0;           //variable counter,the number will be displayed by 7-segment
display
//Open one of the 7-segment display and close the remaining three, the parameter
digit is optional for 1,2,4,8
void selectDigit(int digit){
    digitalWrite(digitPin[0],((digit&0x08) != 0x08) ? LOW : HIGH);
    digitalWrite(digitPin[1],((digit&0x04) != 0x04) ? LOW : HIGH);
    digitalWrite(digitPin[2],((digit&0x02) != 0x02) ? LOW : HIGH);
    digitalWrite(digitPin[3],((digit&0x01) != 0x01) ? LOW : HIGH);
}
void sendOut(int dPin,int cPin,int order,int val){
    int i;
    for(i = 0; i < 8; i++){
        digitalWrite(cPin,LOW);
        if(order == LSBFIRST){
            digitalWrite(dPin,((0x01&(val>>i)) == 0x01) ? HIGH : LOW);
            delayMicroseconds(10);
        }
        else {
            digitalWrite(dPin,((0x80&(val<<i)) == 0x80) ? HIGH : LOW);
            delayMicroseconds(10);
        }
        digitalWrite(cPin,HIGH);
        delayMicroseconds(10);
    }
}

void outData(int8_t data){      //function used to output data for 74HC595
    digitalWrite(latchPin,LOW);
    sendOut(dataPin,clockPin,MSBFIRST,data);
    digitalWrite(latchPin,HIGH);
}
void display(int dec){ //display function for 7-segment display
    int delays = 1;
    outData(0xff);
    selectDigit(0x01);      //select the first, and display the single digit
```

```
outData(num[dec%10]);
delay(delays); //display duration

outData(0xff);
selectDigit(0x02); //select the second, and display the tens digit
outData(num[dec%100/10]);
delay(delays);

outData(0xff);
selectDigit(0x04); //select the third, and display the hundreds digit
outData(num[dec%1000/100]);
delay(delays);

outData(0xff);
selectDigit(0x08); //select the fourth, and display the thousands digit
outData(num[dec%10000/1000]);
delay(delays);

}

void timer(int sig){ //Timer function
    if(sig == SIGALRM){ //If the signal is SIGALRM, the value of counter plus
        1, and update the number displayed by 7-segment display
        counter++;
        alarm(1); //set the next timer time
        printf("counter : %d \n",counter);
    }
}

int main(void)
{
    int i;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("failed !");
        return 1;
    }
    pinMode(dataPin,OUTPUT); //set the pin connected to 74HC595 for
output mode
    pinMode(latchPin,OUTPUT);
    pinMode(clockPin,OUTPUT);
//set the pin connected to 7-segment display common end to output mode
    for(i=0;i<4;i++){
        pinMode(digitPin[i],OUTPUT);
        digitalWrite(digitPin[i],HIGH);
```

```

    }
    signal(SIGALRM,timer); //configure the timer
    alarm(1);           //set the time of timer to 1s
    while(1){
        display(counter); //display the number counter
    }
    return 0;
}

```

Code Interpretation

Definieren Sie zunächst den Pin des gemeinsamen Endes der 74HC595 und 7-Segment-Anzeige, die Zeichencodierung und einen variablen "counter", der als Zähler angezeigt werden soll.

```

#define      dataPin      5
#define      latchPin     4
#define      clockPin     1
const int digitPin[]={0,2,3,12};
unsigned char num[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
int counter = 0;

num[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
int counter = 0;

```

Mit der Unterfunktion "selectDigit (int digit)" wird eine der 7-Segment-Digitalröhren geöffnet und die anderen drei 7-Segment-Digitalröhren ausgeschaltet. Der Parameter Digitalwert kann 1, 2, 4, 8 sein.

```

void selectDigit(int digit){
    digitalWrite(digitPin[0],((digit&0x08) != 0x08) ? LOW : HIGH);
    digitalWrite(digitPin[1],((digit&0x04) != 0x04) ? LOW : HIGH);
    digitalWrite(digitPin[2],((digit&0x02) != 0x02) ? LOW : HIGH);
    digitalWrite(digitPin[3],((digit&0x01) != 0x01) ? LOW : HIGH);
}

```

Die Unterfunktion "outData (int8_t data)" wird verwendet, um die 74HC595 Ausgabe zu 8-Bit-Daten zu machen.

```

void outData(int8_t data){
    digitalWrite(latchPin,LOW);
    sendOut(dataPin,clockPin,MSBFIRST,data);
}

```

```
    digitalWrite(latchPin,HIGH);
}
```

Die Unterfunktionsanzeige (int dec) wird verwendet, um die 4-stellige 7-Segment-Anzeige zu einer 4-Bit-Ganzzahl zu machen. Öffnen Sie zuerst das gemeinsame Ende der ersten 7-Segment-Anzeige und schließen Sie es in der Nähe der anderen drei. Zu diesem Zeitpunkt kann es als 1-stellige 7-Segment-Anzeige verwendet werden. Die erste wird zum Anzeigen einer einzelnen Ziffer verwendet, die zweite für die Zehnerstelle, die dritte für die Hunderterstelle und die vierte für die Tausenderstelle. Jede Ziffer wird für einen bestimmten Zeitraum mit delay () angezeigt. Die Zeit in diesem Code ist sehr kurz eingestellt, so dass Sie sehen werden, dass eine andere Ziffer durcheinander ist. Wenn die Zeit lang genug eingestellt ist, sehen Sie, dass jede Ziffer unabhängig angezeigt wird.

```
void display(int dec){
    int delays = 1;
    outData(0xff);
    selectDigit(0x01);
    outData(num[dec%10]);
    delay(delays);
    outData(0xff);
    selectDigit(0x02);
    outData(num[dec%100/10]);
    delay(delays);
    outData(0xff);
    selectDigit(0x04);
    outData(num[dec%1000/100]);
    delay(delays);

    outData(0xff);
    selectDigit(0x08);          outData(num[dec%10000/1000]);
    delay(delays);
}
```

Unterfunktions Timer (int sig) ist die Timer-Funktion, die ein Alarmsignal setzt. Diese Funktion wird einmal in festgelegten Intervallen ausgeführt. Begleitet von der Ausführung wird der variable Zähler 1 hinzugefügt und dann die "Time" des Timers auf 1s zurückgesetzt.

```
void timer(int sig){      //Timer function
    if(sig == SIGALRM){   //If the signal is SIGALRM, the value of counter plus
        1, and update the number displayed by 7-segment display
```

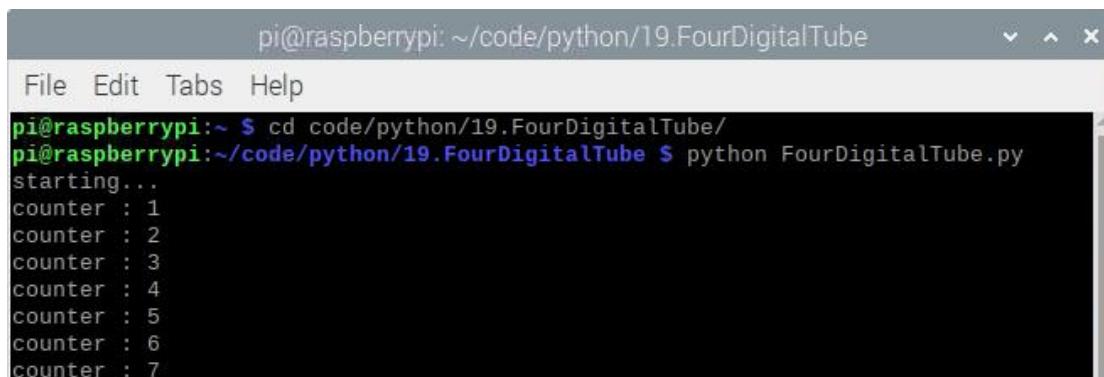
```

        counter++;
        alarm(1);           //set the next timer time
        printf("counter : %d \n",counter);
    }
}

```

Python code

1. Verwenden Sie den Befehl "cd code / python / 19.FourDigitalTube /", um das Verzeichnis "FourDigitalTube" aufzurufen.
2. Verwenden Sie den Befehl "python FourDigitalTube.py", um den Code "FourDigitalTube.py" auszuführen.



```

pi@raspberrypi:~ $ cd code/python/19.FourDigitalTube/
pi@raspberrypi:~/code/python/19.FourDigitalTube $ python FourDigitalTube.py
starting...
counter : 1
counter : 2
counter : 3
counter : 4
counter : 5
counter : 6
counter : 7

```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```

import RPi.GPIO as GPIO
import time
import threading

LWAY = 1
MWAY = 2
#define the pins connect to 74HC595
dataPin  = 18      #DS Pin of 74HC595(Pin14)
latchPin = 16      #ST_CP Pin of 74HC595(Pin12)
clockPin = 12      #CH_CP Pin of 74HC595(Pin11)

counter =0         #Variable counter, the number will be displayed by 7-segment
display
t=0               # define the Timer object

```

```
num = [0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90]
digitPin = (11,13,15,19)      # Define the pin of 7-segment display common end

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(dataPin,GPIO.OUT)
    GPIO.setup(latchPin,GPIO.OUT)
    GPIO.setup(clockPin,GPIO.OUT)
    for Pin in digitPin:
        GPIO.setup(Pin,GPIO.OUT)

def sendout(dPin,cPin,way,val):
    for i in range(0,8):
        GPIO.output(cPin,GPIO.LOW)
        if(way == LWAY):
            GPIO.output(dPin,(0x01 & (val>>i)==0x01) and GPIO.HIGH or
GPIO.LOW)
        elif(way == MWAY):
            GPIO.output(dPin,(0x80 & (val<<i)==0x80) and GPIO.HIGH or
GPIO.LOW)
        GPIO.output(cPin,GPIO.HIGH)

def outData(data):      #function used to output data for 74HC595
    GPIO.output(latchPin,GPIO.LOW)
    sendout(dataPin,clockPin,MWAY,data)
    GPIO.output(latchPin,GPIO.HIGH)

def selectDigit(digit): # Open one of the 7-segment display and close the remaining
three, the parameter digit is optional for 1,2,4,8
    GPIO.output(digitPin[0],GPIO.LOW if ((digit&0x08) != 0x08) else
GPIO.HIGH)
    GPIO.output(digitPin[1],GPIO.LOW if ((digit&0x04) != 0x04) else
GPIO.HIGH)
    GPIO.output(digitPin[2],GPIO.LOW if ((digit&0x02) != 0x02) else
GPIO.HIGH)
    GPIO.output(digitPin[3],GPIO.LOW if ((digit&0x01) != 0x01) else
GPIO.HIGH)

def display(dec):    #display function for 7-segment display
    outData(0xff)    #eliminate residual display
    selectDigit(0x01)  #Select the first, and display the single digit
```

```
outData(num[dec%10])
time.sleep(0.001)    #display duration
outData(0xff)
selectDigit(0x02)    # Select the second, and display the tens digit
outData(num[dec%100//10])
time.sleep(0.001)
outData(0xff)
selectDigit(0x04)    # Select the third, and display the hundreds digit
outData(num[dec%1000//100])
time.sleep(0.001)
outData(0xff)
selectDigit(0x08)    # Select the fourth, and display the thousands digit
outData(num[dec%10000//1000])
time.sleep(0.001)

def timer():          #timer function
    global counter
    global t
    t = threading.Timer(1.0,timer)      #reset time of timer to 1s
    t.start()                         #Start timing
    counter+=1
    print ("counter : %d"%counter)

def loop():
    global t
    global counter
    t = threading.Timer(1.0,timer)      #set the timer
    t.start()                         # Start timing
    while True:
        display(counter)             # display the number counter

def destroy(): # When 'Ctrl+C' is pressed, the function is executed.
    global t
    GPIO.cleanup()
    t.cancel() # cancel the timer

if __name__ == '__main__': # Program starting from here
    print("starting...")
    setup()
    try:
        loop()
    except KeyboardInterrupt:
```

destroy()

Code Interpretation

Definieren Sie zunächst die Pins von 74HC595 und das gemeinsame Ende der 7-Segment-Digitalröhre, den Zeichencode der Zahlen "0-8" und die Variable "counter" für das Timing.

```

dataPin = 18      #DS Pin of 74HC595(Pin14)
latchPin = 16    #ST_CP Pin of 74HC595(Pin12)
clockPin = 12    #CH_CP Pin of 74HC595(Pin11)
counter = 0       #Variable counter, the number will be displayed by 7-segment
display
t=0              # define the Timer object
num = [0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90]
digitPin = (11,13,15,19)  # Define the pin of 7-segment display c

```

Mit der Unterfunktion "selectDigit(digit)" wird eine der 7-Segment-Digitalröhren geöffnet und die anderen drei 7-Segment-Digitalröhren ausgeschaltet. Der Parameter Digitalwert kann 1, 2, 4, 8 sein.

```

def selectDigit(digit): # Open one of the 7-segment display and close the remaining
three, the parameter digit is optional for 1,2,4,8
    GPIO.output(digitPin[0],GPIO.LOW if ((digit&0x08) != 0x08) else
GPIO.HIGH)
    GPIO.output(digitPin[1],GPIO.LOW if ((digit&0x04) != 0x04) else
GPIO.HIGH)
    GPIO.output(digitPin[2],GPIO.LOW if ((digit&0x02) != 0x02) else
GPIO.HIGH)
    GPIO.output(digitPin[3],GPIO.LOW if ((digit&0x01) != 0x01) else GPIO.HIGH)

```

Die Unterfunktion "outData (data)" wird verwendet, um die 74HC595 Ausgabe zu 8-Bit-Daten zu machen.

```

def outData(data):      #function used to output data for 74HC595
    GPIO.output(latchPin,GPIO.LOW)
    sendout(dataPin,clockPin,MWAY,data)
    GPIO.output(latchPin,GPIO.HIGH)

```

Die Unterfunktionsanzeige (int dec) wird verwendet, um die 4-stellige 7-Segment-Anzeige zu einer 4-Bit-Ganzzahl zu machen. Öffnen Sie zuerst das gemeinsame Ende der ersten 7-Segment-Anzeige und schließen Sie es in der Nähe der anderen drei. Zu diesem Zeitpunkt kann es als 1-stellige 7-Segment-Anzeige

verwendet werden. Die erste wird zum Anzeigen einer einzelnen Ziffer verwendet, die zweite für die Zehnerstelle, die dritte für die Hunderterstelle und die vierte für die Tausenderstelle. Jede Ziffer wird für einen bestimmten Zeitraum mit `delay()` angezeigt. Die Zeit in diesem Code ist sehr kurz eingestellt, so dass Sie sehen werden, dass eine andere Ziffer durcheinander ist. Wenn die Zeit lang genug eingestellt ist, sehen Sie, dass jede Ziffer unabhängig angezeigt wird.

```
def display(dec):      #display function for 7-segment display
    outData(0xff)      #eliminate residual display
    selectDigit(0x01)   #Select the first, and display the single digit
    outData(num[dec%10])
    time.sleep(0.001)   #display duration
    outData(0xff)
    selectDigit(0x02)   # Select the second, and display the tens digit
    outData(num[dec%100//10])
    time.sleep(0.001)
    outData(0xff)
    selectDigit(0x04)   # Select the third, and display the hundreds digit
    outData(num[dec%1000//100])
    time.sleep(0.001)
    outData(0xff)
    selectDigit(0x08)   # Select the fourth, and display the thousands digit
    outData(num[dec%10000//1000])
    time.sleep(0.001)
```

Die Unterfunktion „`timer()`“ ist die Timer Rückruf Funktion. Wenn die Zeit abgelaufen ist, wird diese Funktion ausgeführt. Begleitet von der Ausführung wird der variable Zähler 1 hinzugefügt und die Zeit des Timers auf 1s.1s später zurückgesetzt. Die Funktion wird erneut ausgeführt.

```
def timer():          #timer function
    global counter
    global t
    t = threading.Timer(1.0,timer)      #reset time of timer to 1s
    t.start()                         #Start timing
    counter+=1
    print ("counter : %d"%counter)
```

Unterfunktion „`setup()`“ konfigurieren Sie alle Eingangs/ Ausgangsmodi für den verwendeten GPIO Pin. Stellen Sie schließlich in der Schleifenfunktion die Anzeige des Zählers der digitalen Röhre im Zyklus „`while`“ variabel ein. Der Wert ändert sich in der Funktion `timer()`, sodass sich der Inhalt der 7-Segment-Anzeige entsprechend ändert.

```
def loop():
    global t
    global counter
    t = threading.Timer(1.0,timer)      #set the timer
    t.start()                          # Start timing
    while True:
        display(counter)             # display the number
```

Nachdem das Programm ausgeführt wurde, drücken Sie "Strg + c" und führen Sie dann die Unterfunktion destroy (), GPIO-Ressource aus und der Timer wird in der Unterfunktion ausgeschaltet.

```
def destroy(): # When 'Ctrl+C' is pressed, the function is executed.
    global t
    GPIO.cleanup()
    t.cancel() # cancel the timer
```

Lektion 20 74HC595 & LED Matrix

Überblick

In dieser Lektion erfahren Sie, wie Sie den 74HC595 weiterhin zur Steuerung weiterer LEDs verwenden, das ist LEDMatrix. Wir verwenden zwei 74HC595 zur Steuerung einer monochromen LEDMatrix (8 * 8), damit einige Grafiken und Zeichen angezeigt werden.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-Erweiterungskarte

1 x Steckbrett

2 x 74HC595

8 x 220Ω Widerstand

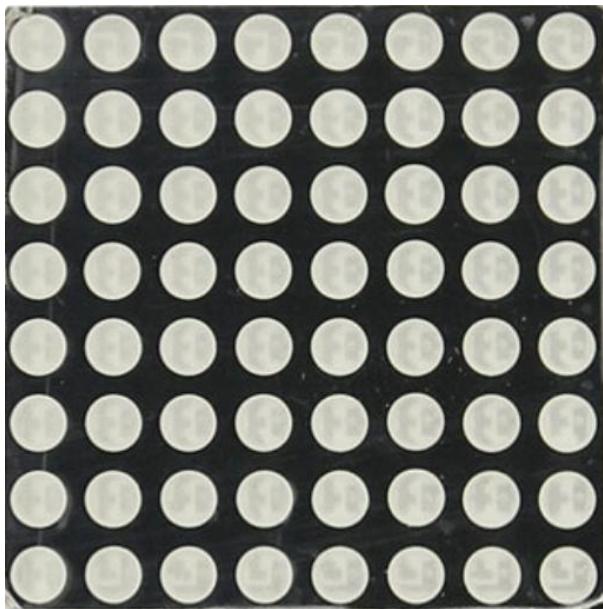
1 x LED Matrix (8 * 8)

Produkt Einführung

LED matrix

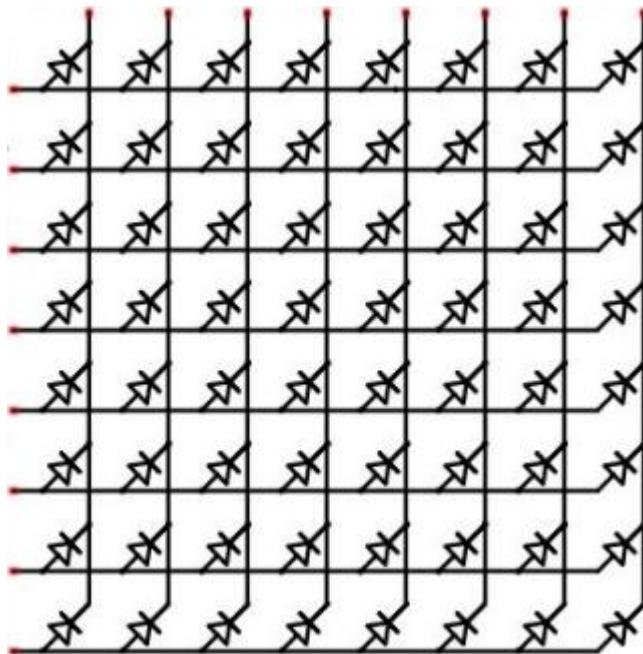
Die LED Matrix ist ein rechteckiges Anzeigemodul, das aus mehreren LEDs besteht.

Das Folgende ist eine 8 * 8 monochrome LED Matrix mit 64 LEDs (8 Zeilen und 8 Spalten).

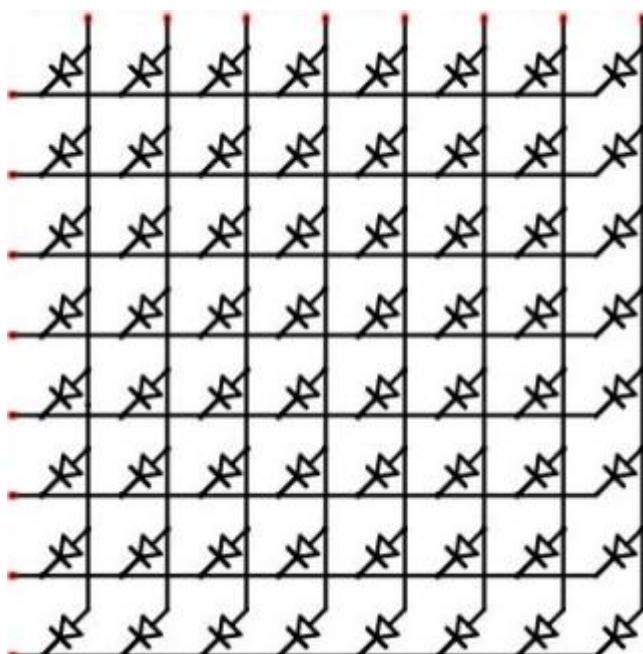


Um den Betrieb zu erleichtern und die Ports zu schonen, sind der positive Pol der LEDs in jeder Reihe und der negative Pol der LEDs in jeder Spalte innerhalb des LED Matrixmoduls miteinander verbunden, das als Common Anode bezeichnet wird. Es gibt eine andere Form. Der negative Pol der LEDs in jeder Reihe und der positive Pol der LEDs in jeder Spalte sind miteinander verbunden, was als gemeinsame Kathode bezeichnet wird. Diejenige, die wir in diesem Projekt verwenden, ist eine gemeinsame Anode LEDMatrix.

Verbindungsmodus der gemeinsamen Anode



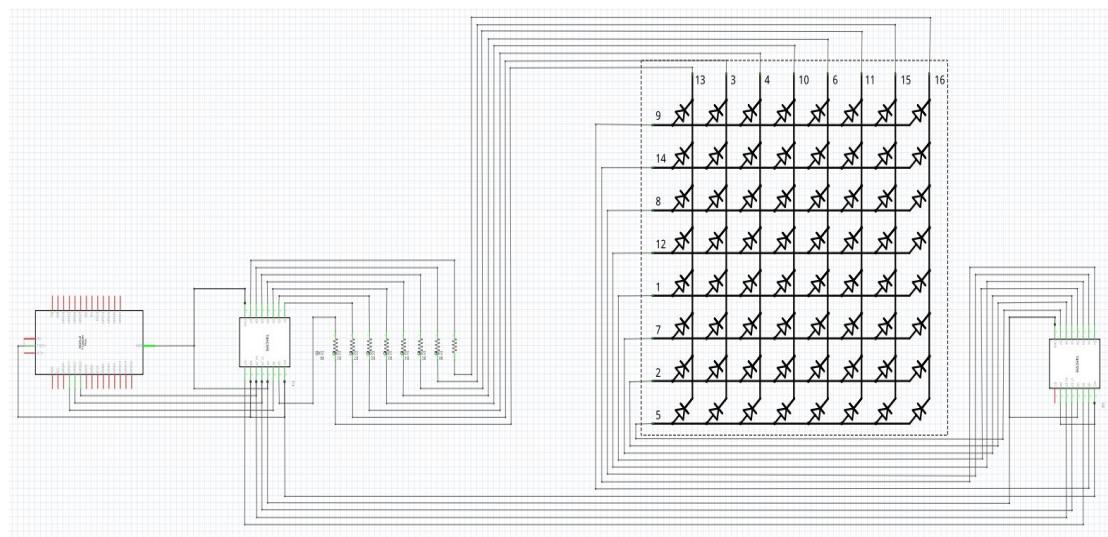
Verbindungsmodus der gemeinsamen Kathode



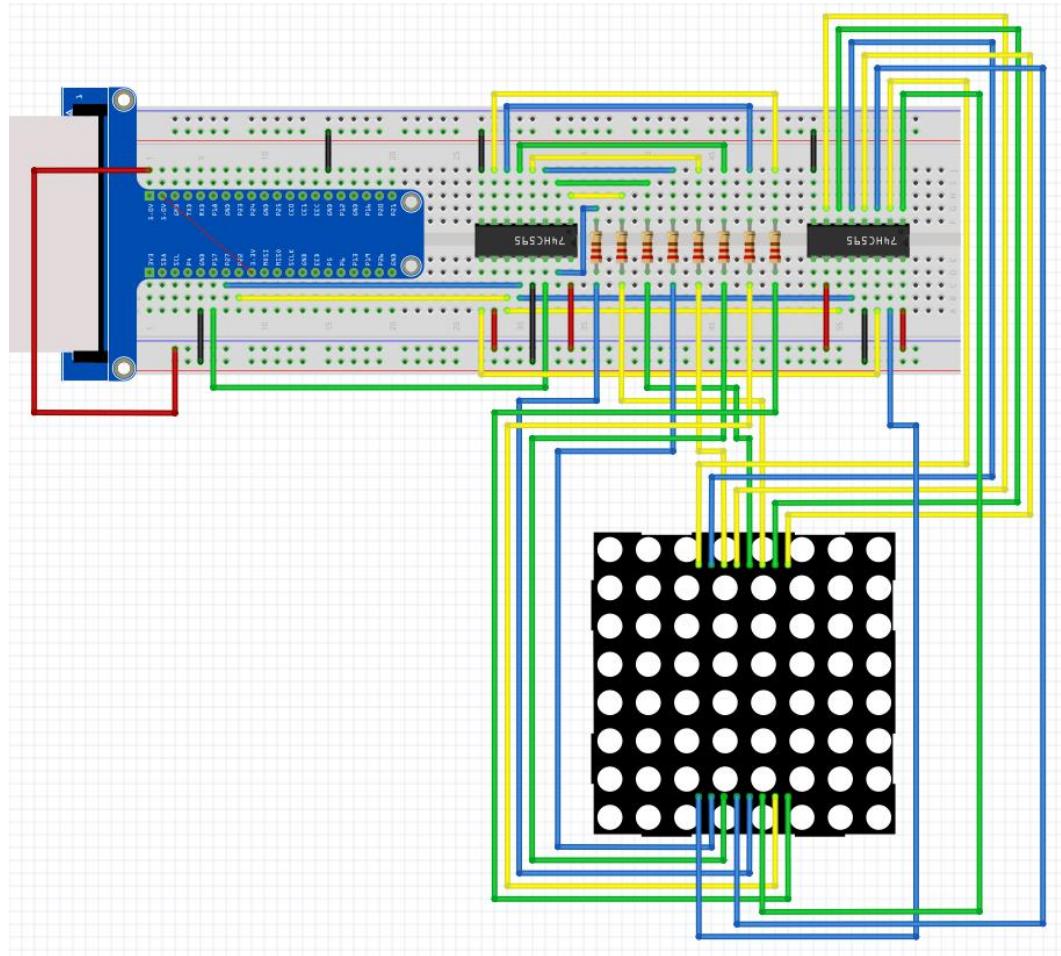
Das Scannen von Zeilen ist eine weitere Art der Punktmatrix. Unabhängig davon, ob eine Zeile oder eine Spalte gescannt wird, sind 16 GPIO erforderlich. Um den GPIO der Steuerkarte zu sparen, werden zwei 74HC595 verwendet. Jedes Stück des 74HC595 verfügt über acht parallele Ausgangsanschlüsse, sodass zwei Teile

insgesamt 16 Anschlüsse haben. Die Steuerleitung und die Datenleitung von zwei 74HC595 sind nicht alle mit dem RPi verbunden, sondern verbinden den Q7 Pin der ersten Stufe 74HC595 mit dem Daten Pin der zweiten Stufe, nämlich zwei 74HC595 sind in Reihe geschaltet. Dies entspricht der Verwendung eines "74HC595" mit 16 parallelen Ausgangsanschlüssen.

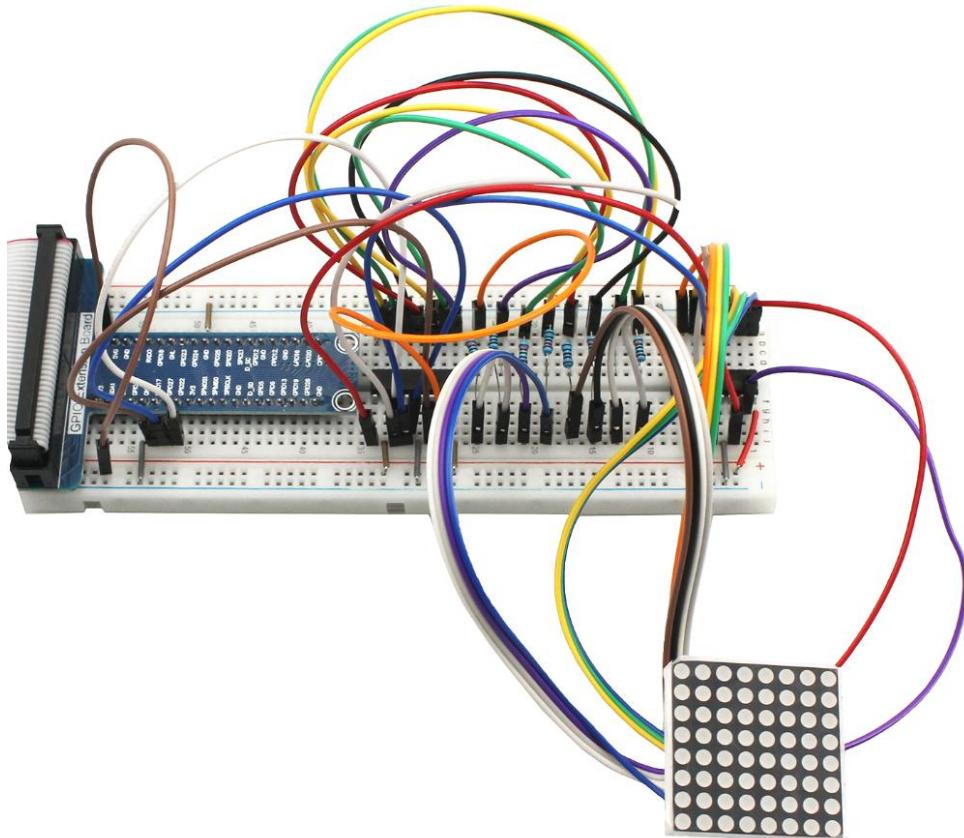
Schaltplan



Verdrahtung Diagramm



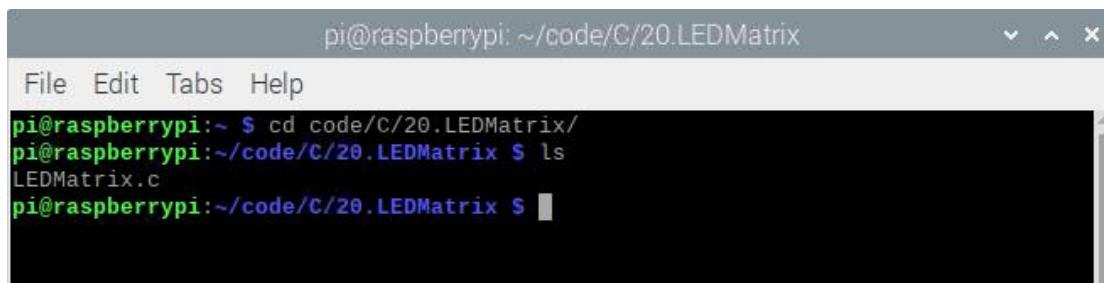
Beispiel Abbildung



C code

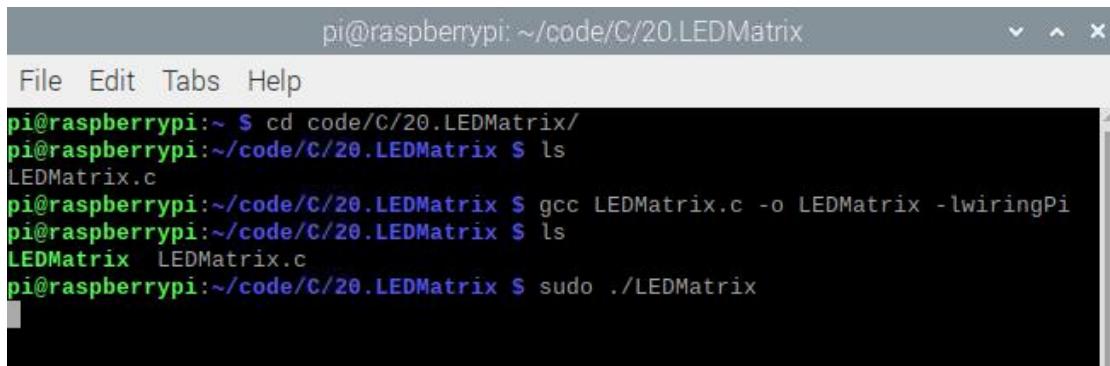
Öffnen Sie das Terminal und geben Sie den Befehl "`cd code / C / 20.LEDMatrix /`" ein, um das Codeverzeichnis "LEDMatrix" aufzurufen;

Geben Sie den Befehl "`ls`" ein, um die Datei "LEDMatrix.c" im Verzeichnis anzuzeigen;



```
pi@raspberrypi:~/code/C/20.LEDMatrix
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/20.LEDMatrix/
pi@raspberrypi:~/code/C/20.LEDMatrix $ ls
LEDMatrix.c
pi@raspberrypi:~/code/C/20.LEDMatrix $
```

Geben Sie den Befehl "[gcc LEDMatrix.c -o LEDMatrix -lwiringPi](#)" ein, um die ausführbare Datei "LEDMatrix" von "LEDMatrix.c" zu generieren. Geben Sie den Befehl "[ls](#)" ein, um sie anzuzeigen. Geben Sie den Befehl "[sudo ./LEDMatrix](#)" ein, um den Code auszuführen. Die Ergebnisse werden unten angezeigt:



```
pi@raspberrypi:~/code/C/20.LEDMatrix
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/20.LEDMatrix/
pi@raspberrypi:~/code/C/20.LEDMatrix $ ls
LEDMatrix.c
pi@raspberrypi:~/code/C/20.LEDMatrix $ gcc LEDMatrix.c -o LEDMatrix -lwiringPi
pi@raspberrypi:~/code/C/20.LEDMatrix $ ls
LEDMatrix LEDMatrix.c
pi@raspberrypi:~/code/C/20.LEDMatrix $ sudo ./LEDMatrix
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>
#include <wiringShift.h>

#define dataPin 0 //DS Pin of 74HC595(Pin14)
#define latchPin 2 //ST_CP Pin of 74HC595(Pin12)
#define clockPin 3 //SH_CP Pin of 74HC595(Pin11)
// data of smiling face
unsigned char pic[]={0x1c,0x22,0x51,0x45,0x45,0x51,0x22,0x1c};
unsigned char data[]={ // data of "0-F"
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ""
    0x00, 0x00, 0x3E, 0x41, 0x41, 0x3E, 0x00, 0x00, // "0"
    0x00, 0x00, 0x21, 0x7F, 0x01, 0x00, 0x00, 0x00, // "1"
    0x00, 0x00, 0x23, 0x45, 0x49, 0x31, 0x00, 0x00, // "2"
    0x00, 0x00, 0x22, 0x49, 0x49, 0x36, 0x00, 0x00, // "3"
    0x00, 0x00, 0x0E, 0x32, 0x7F, 0x02, 0x00, 0x00, // "4"
    0x00, 0x00, 0x79, 0x49, 0x49, 0x46, 0x00, 0x00, // "5"
    0x00, 0x00, 0x3E, 0x49, 0x49, 0x26, 0x00, 0x00, // "6"
    0x00, 0x00, 0x60, 0x47, 0x48, 0x70, 0x00, 0x00, // "7"
    0x00, 0x00, 0x36, 0x49, 0x49, 0x36, 0x00, 0x00, // "8"
    0x00, 0x00, 0x32, 0x49, 0x49, 0x3E, 0x00, 0x00, // "9"
    0x00, 0x00, 0x3F, 0x44, 0x44, 0x3F, 0x00, 0x00, // "A"
```

```
0x00, 0x00, 0x7F, 0x49, 0x49, 0x36, 0x00, 0x00, // "B"
0x00, 0x00, 0x3E, 0x41, 0x41, 0x22, 0x00, 0x00, // "C"
0x00, 0x00, 0x7F, 0x41, 0x41, 0x3E, 0x00, 0x00, // "D"
0x00, 0x00, 0x7F, 0x49, 0x49, 0x41, 0x00, 0x00, // "E"
0x00, 0x00, 0x7F, 0x48, 0x48, 0x40, 0x00, 0x00, // "F"
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ""
};

void sendOut(int dPin,int cPin,int order,int val){
    int i;
    for(i = 0; i < 8; i++){
        digitalWrite(cPin,LOW);
        if(order == LSBFIRST){
            digitalWrite(dPin,((0x01&(val>>i)) == 0x01) ? HIGH : LOW);
            delayMicroseconds(10);
        }
        else {
            digitalWrite(dPin,((0x80&(val<<i)) == 0x80) ? HIGH : LOW);
            delayMicroseconds(10);
        }
        digitalWrite(cPin,HIGH);
        delayMicroseconds(10);
    }
}

int main(void)
{
    int i,j,k;
    unsigned char x;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("failed !");
        return 1;
    }
    pinMode(dataPin,OUTPUT);
    pinMode(latchPin,OUTPUT);
    pinMode(clockPin,OUTPUT);
    while(1){
        for(j=0;j<500;j++)// Repeat enough times to display the smiling face a
period of time
        x=0x80;
        for(i=0;i<8;i++){
            digitalWrite(latchPin,LOW);
```

```
sendOut(dataPin,clockPin,MSBFIRST,pic[i]);// first shift data of
line information to the first stage 74HC959
```

```
sendOut(dataPin,clockPin,MSBFIRST,~x); //then shift data of
column information to the second stage 74HC959
```

```
digitalWrite(latchPin,HIGH); //Output data of two stage 74HC595
at the same time
```

```
x>>=1; // display the next column
delay(1);
}
}
for(k=0;k<sizeof(data)-8;k++){ //sizeof(data) total number of "0-F"
columns
    for(j=0;j<20;j++){ // times of repeated displaying LEDMatrix in every
frame, the bigger the "j", the longer the display time
        x=0x80; // Set the column information to start from the first
column
        for(i=k;i<8+k;i++){
            digitalWrite(latchPin,LOW);
            sendOut(dataPin,clockPin,MSBFIRST,data[i]);
            sendOut(dataPin,clockPin,MSBFIRST,~x);
            digitalWrite(latchPin,HIGH);
            x>>=1;
            delay(1);
        }
    }
}
return 0;
}
```

Code Interpretation

Der erste "for" Zyklus im "while" Zyklus wird verwendet, um ein statisches Lächeln anzusehen. Zeigen Sie Spalteninformationen von links nach rechts an, eine Spalte für eine Spalte, insgesamt 8 Spalten. Wiederholen Sie diesen Vorgang 500 Mal, um die Anzeigezzeit ausreichend zu gewährleisten.

```
for(j=0;j<500;j++){ // Repeat enough times to display the smiling face a period of time
x=0x80;
```

```
for(i=0;i<8;i++){
    digitalWrite(latchPin,LOW);
    sendOut(dataPin,clockPin,MSBFIRST,pic[i]);
    sendOut(dataPin,clockPin,MSBFIRST,~x);

    digitalWrite(latchPin,HIGH);
    x>>=1;
    delay(1);
}
```

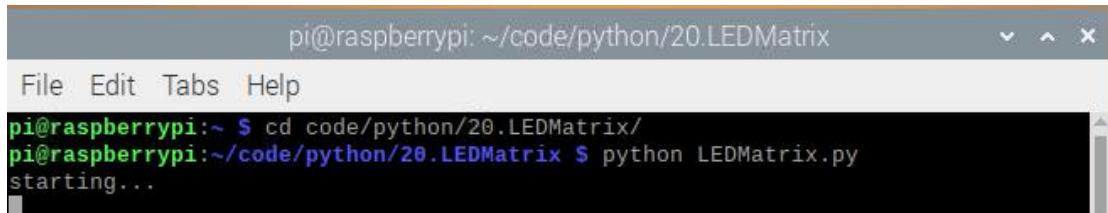
```
}
```

Der zweite "for" Zyklus wird verwendet, um die Bildlaufzeichen "0-F" anzuzeigen, insgesamt $18 * 8 = 144$ Spalten. Zeigen Sie die Spalten 0-8, 1-9, 2-10 und 138-144 an, um einen Bildlaufeffekt zu erzielen. Die Anzeige jedes Frames wird eine bestimmte Anzahl von Malen wiederholt. Je öfter die Anzahl der Wiederholungen, desto länger die Einzelbildanzeige, desto langsamer das Rollen.

```
for(k=0;k<sizeof(data)-8;k++){
    for(j=0;j<20;j++){
        x=0x80;
        for(i=k;i<8+k;i++){
            digitalWrite(latchPin,LOW);
            sendOut(dataPin,clockPin,MSBFIRST,data[i]);
            sendOut(dataPin,clockPin,MSBFIRST,~x);
            digitalWrite(latchPin,HIGH);
            x>>=1;
            delay(1);
        }
    }
}
```

Python code

1. Verwenden Sie den Befehl "[cd code / python / 20.LEDMatrix /](#)", um das Verzeichnis "LEDMatrix" aufzurufen;
2. Verwenden Sie den Befehl "[python LEDMatrix.py](#)", um den Code "LEDMatrix.py" auszuführen.



```
pi@raspberrypi: ~/code/python/20.LEDMatrix
pi@raspberrypi:~/code/python/20.LEDMatrix $ python LEDMatrix.py
starting...
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import time

LWAY = 1
MWAY = 2

dataPin = 11          #DS Pin of 74HC595(Pin14)
latchPin = 13         #ST_CP Pin of 74HC595(Pin12)
clockPin = 15         #CH_CP Pin of 74HC595(Pin11)

pic = [0x1c,0x22,0x51,0x45,0x45,0x51,0x22,0x1c] #data of smiling face
data = [#data of "0-F"
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, # ""
        0x00, 0x00, 0x3E, 0x41, 0x41, 0x3E, 0x00, 0x00, # "0"
        0x00, 0x00, 0x21, 0x7F, 0x01, 0x00, 0x00, 0x00, # "1"
        0x00, 0x00, 0x23, 0x45, 0x49, 0x31, 0x00, 0x00, # "2"
        0x00, 0x00, 0x22, 0x49, 0x49, 0x36, 0x00, 0x00, # "3"
        0x00, 0x00, 0x0E, 0x32, 0x7F, 0x02, 0x00, 0x00, # "4"
        0x00, 0x00, 0x79, 0x49, 0x49, 0x46, 0x00, 0x00, # "5"
        0x00, 0x00, 0x3E, 0x49, 0x49, 0x26, 0x00, 0x00, # "6"
        0x00, 0x00, 0x60, 0x47, 0x48, 0x70, 0x00, 0x00, # "7"
        0x00, 0x00, 0x36, 0x49, 0x49, 0x36, 0x00, 0x00, # "8"
        0x00, 0x00, 0x32, 0x49, 0x49, 0x3E, 0x00, 0x00, # "9"
        0x00, 0x00, 0x3F, 0x44, 0x44, 0x3F, 0x00, 0x00, # "A"
        0x00, 0x00, 0x7F, 0x49, 0x49, 0x36, 0x00, 0x00, # "B"
```

```
0x00, 0x00, 0x3E, 0x41, 0x41, 0x22, 0x00, 0x00, # "C"
0x00, 0x00, 0x7F, 0x41, 0x41, 0x3E, 0x00, 0x00, # "D"
0x00, 0x00, 0x7F, 0x49, 0x49, 0x41, 0x00, 0x00, # "E"
0x00, 0x00, 0x7F, 0x48, 0x48, 0x40, 0x00, 0x00, # "F"
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, # ""
]

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(dataPin,GPIO.OUT)
    GPIO.setup(latchPin,GPIO.OUT)
    GPIO.setup(clockPin,GPIO.OUT)

def sendout(dPin,cPin,way,val):
    for i in range(0,8):
        GPIO.output(cPin,GPIO.LOW)
        if(way == LWAY):
            GPIO.output(dPin,(0x01 & (val>>i)==0x01) and GPIO.HIGH or
GPIO.LOW)
        elif(way == MWAY):
            GPIO.output(dPin,(0x80 & (val<<i)==0x80) and GPIO.HIGH or
GPIO.LOW)
        GPIO.output(cPin,GPIO.HIGH)

def loop():
    while True:
        for j in range(0,500):# Repeat enough times to display the smiling face a
period of time
            x=0x80
            for i in range(0,8):
                GPIO.output(latchPin,GPIO.LOW)
                sendout(dataPin,clockPin,MWAY,pic[i]) #first shift data of line
information to first stage 74HC959

                sendout(dataPin,clockPin,MWAY,~x) #then shift data of column
information to second stage 74HC959
                GPIO.output(latchPin,GPIO.HIGH)# Output data of two stage
74HC955 at the same time
                time.sleep(0.001)# display the next column
```

```

x>>=1
for k in range(0,len(data)-8):#len(data) total number of "0-F" columns
    for j in range(0,20):# times of repeated displaying LEDMatrix in every
frame, the bigger the "j", the longer the display time.
        x=0x80      # Set the column information to start from the first
column
        for i in range(k,k+8):
            GPIO.output(latchPin,GPIO.LOW)
            sendout(dataPin,clockPin,MWAY,data[i])
            sendout(dataPin,clockPin,MWAY,~x)
            GPIO.output(latchPin,GPIO.HIGH)
            time.sleep(0.001)
            x>>=1

def destroy():      # When 'Ctrl+C' is pressed, the function is executed.
    GPIO.cleanup()

if __name__ == '__main__':  # Program starting from here
    print("starting...")
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()

```

Code Interpretation

Der erste "for" Zyklus im "while" Zyklus wird verwendet, um ein statisches Lächeln anzuzeigen. Zeigen Sie Spalteninformationen von links nach rechts an, eine Spalte für eine Spalte, insgesamt 8 Spalten. Wiederholen Sie diesen Vorgang 500 Mal, um die Anzeigezzeit ausreichend zu gewährleisten.

```

for j in range(0,500):# Repeat enough times to display the smiling face a period of
time
    x=0x80
    for i in range(0,8):
        GPIO.output(latchPin,GPIO.LOW)
        sendout(dataPin,clockPin,MWAY,pic[i]) #first shift data of line
information to first stage 74HC959

```

```
sendout(dataPin,clockPin,MWAY,~x) #then shift data of column  
information to second stage 74HC959  
GPIO.output(latchPin,GPIO.HIGH)# Output data of two stage  
74HC595 at the same time  
time.sleep(0.001)# display the next column  
x>>=1
```

Der erste "for" Zyklus im "while" Zyklus wird verwendet, um ein statisches Lächeln anzuzeigen. Zeigen Sie Spalteninformationen von links nach rechts an, eine Spalte für eine Spalte, insgesamt 8 Spalten. Wiederholen Sie diesen Vorgang 500 Mal, um die Anzeigezeit ausreichend zu gewährleisten.

```
for k in range(0,len(data)-8):#len(data) total number of "0-F" columns  
    for j in range(0,20):# times of repeated displaying LEDMatrix in every  
frame, the bigger the "j", the longer the display time.  
        x=0x80      # Set the column information to start from the first  
column  
        for i in range(k,k+8):  
            GPIO.output(latchPin,GPIO.LOW)  
            sendout(dataPin,clockPin,MWAY,data[i])  
            sendout(dataPin,clockPin,MWAY,~x)  
            GPIO.output(latchPin,GPIO.HIGH)  
            time.sleep(0.001)  
            x>>=1
```

Lektion 21 LCD1602

Überblick

In dieser Lektion lernen Sie einen Bildschirm, LCD 1602.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-Erweiterungskarte und Kabel

1 x Steckbrett

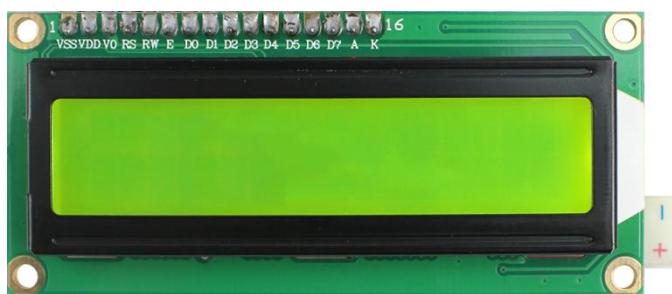
1 x LCD1602

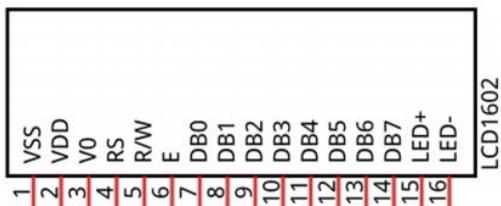
1 x PCF8574

Produkt Einführung

LCD1602

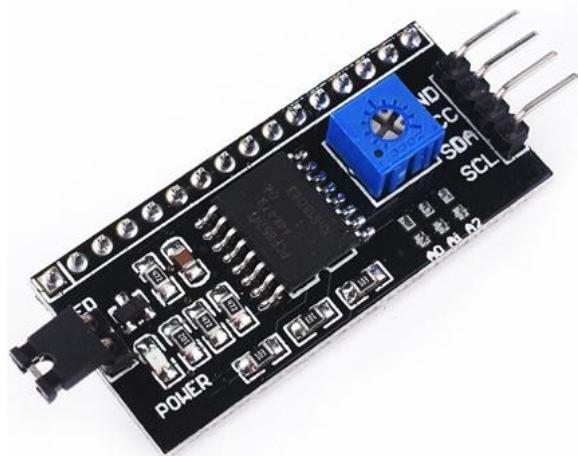
LCD1602 kann 2 Zeichenzeilen in 16 Spalten anzeigen. Es kann Zahlen, Buchstaben, Symbole, ASCII-Code usw. anzeigen. I2C LCD1602 integriert eine I2C-Schnittstelle, die das Modul mit serielllem Eingang und parallelem Ausgang mit LCD1602 verbindet. Wir verwenden nur 4 Zeilen, um den LCD1602 einfach zu bedienen. Wie unten gezeigt, ist ein monochromer LCD1602 Bildschirm, sein Schaltkreis Pin Diagramm lautet:



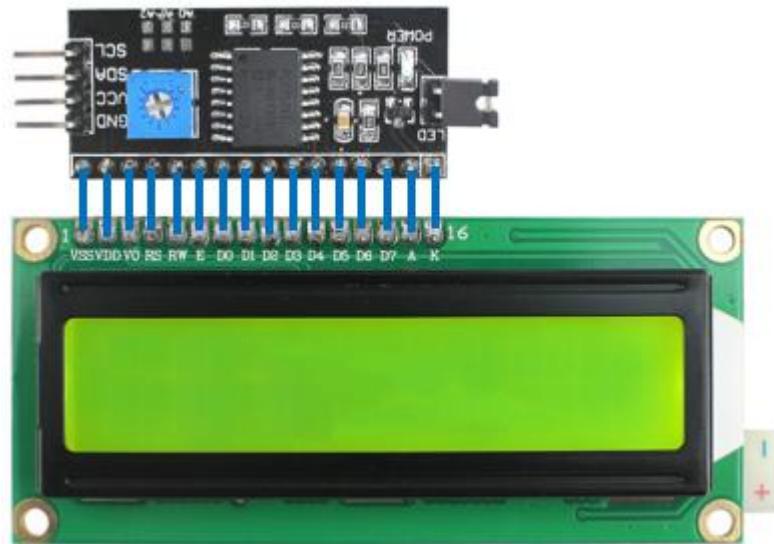


PCF8574

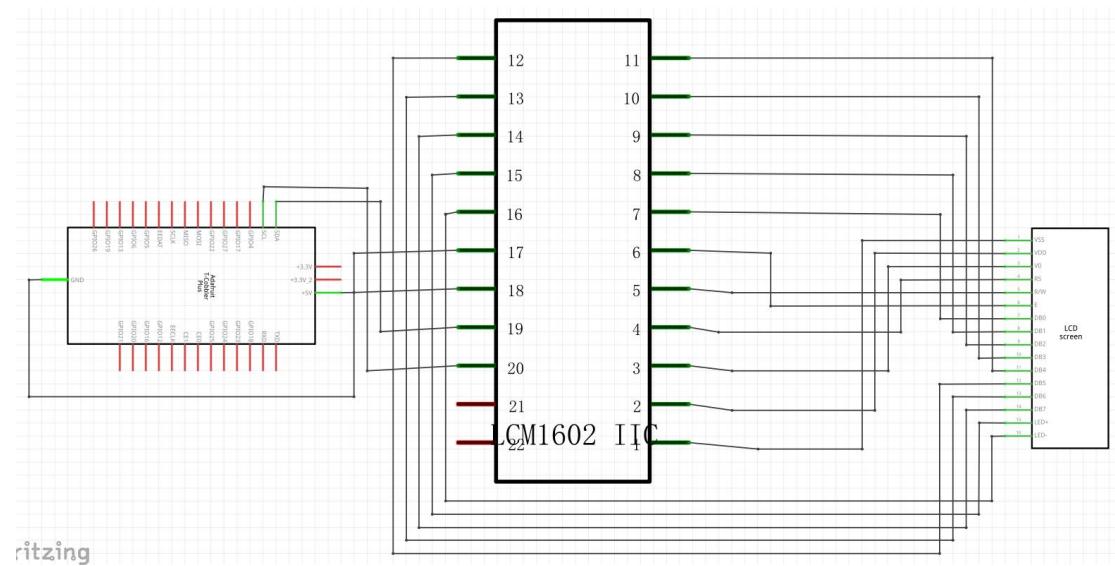
Der in diesem Modul verwendete Seriell Parallel Chip ist PCF8574 (PCF8574A) und seine Standard I2C-Adresse ist 0x27 (0x3F). Sie können den gesamten RPI-Bus auf Ihrer I2C-Geräteadresse über den Befehl "i2cdetect -y 1" bis anzeigen . Der PCF8574 Modul Pin und der LCD1602 Pin entsprechen einander und sind miteinander verbunden: (siehe Abschnitt "Konfiguration I2C" unten) Nachfolgend finden Sie das PCF8574 Pin-Schaltplan und das Block Pin Diagramm:



LCD Anzeige

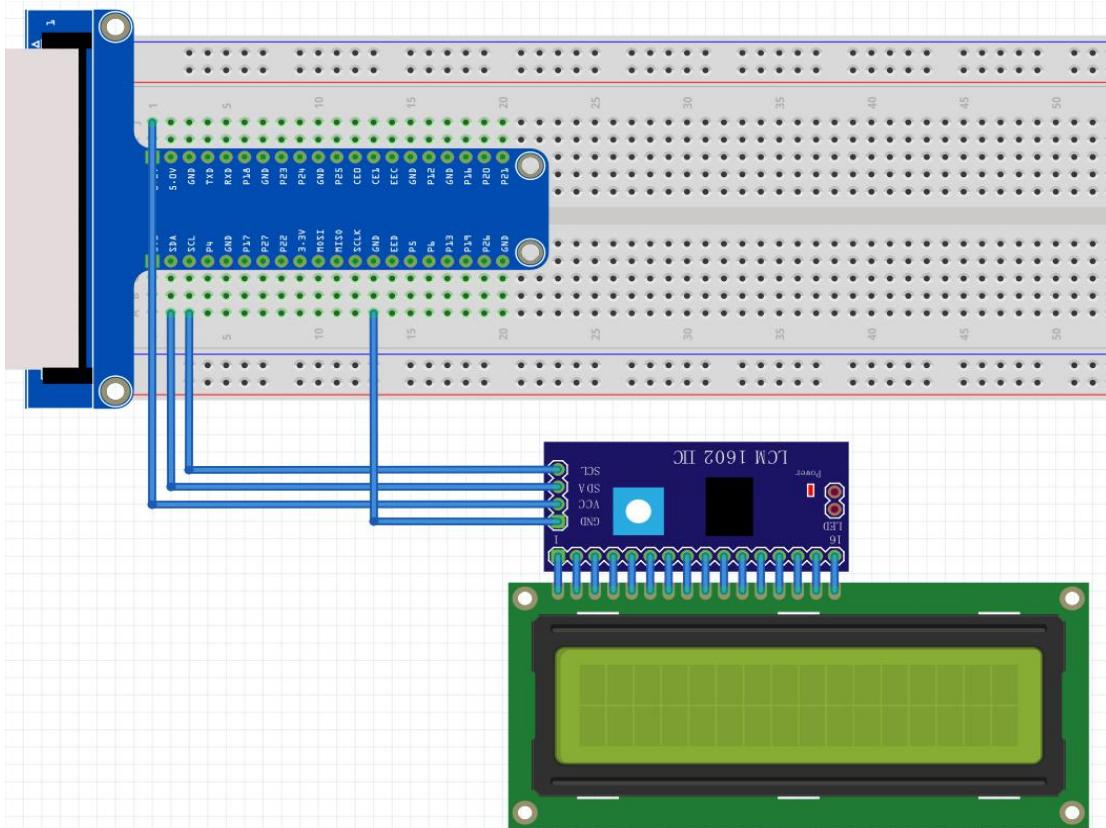


Schaltplan

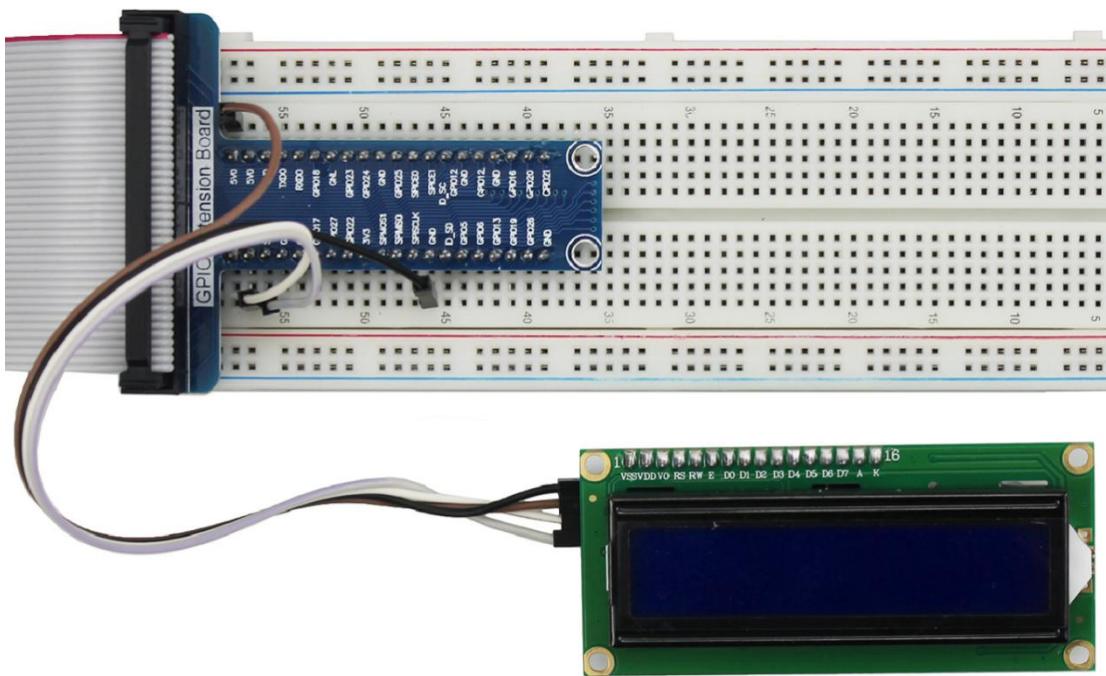




Verdrahtung Diagramm



Beispiel Abbildung



Configure I2C

Die I2C-Schnittstelle Raspberry Pi ist standardmäßig geschlossen. Wenn die VNC Schnittstelle in der Remote Raspberry Pi Schnittstelle vorne geöffnet wird, öffnen wir auch die I2C-Schnittstelle. Wir können einen Befehl eingeben, um zu überprüfen, ob die I2C-Schnittstelle geöffnet ist:

```
lsmod | grep i2c
```

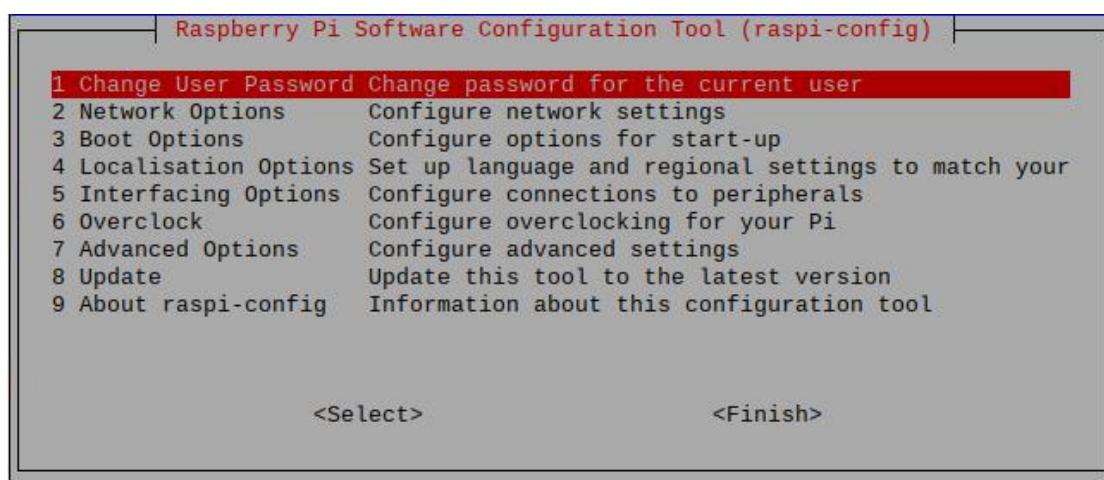
Wenn die I2C-Schnittstelle geöffnet ist, wird Folgendes angezeigt:

```
pi@raspberrypi:~ $ lsmod | grep i2c
i2c_bcm2835      16384  0
i2c_dev          20480  0
pi@raspberrypi:~ $
```

Die folgenden Nummern 16384 und 20480 unterscheiden sich für jeden Raspberry Pi. Solange der angezeigte Inhalt in etwa dem oben genannten Inhalt entspricht, wurde die I2C-Schnittstelle des Raspberry Pi geöffnet. Wenn die I2C-Schnittstelle nicht geöffnet ist, müssen wir sie manuell öffnen. Wir können den Befehl im Terminal eingeben:

```
sudo raspi-config
```

Öffnen Sie dann den folgenden Dialog:



Select "5 Interfacing Options"->"P5 I2C"->"<Yes>"->"<OK>"->"<Finish>", and then restart the Raspberry pi. Now you can check whether the I2C interface Open successfully.

Wählen Sie "**5 Interfacing Options**"->"**P5 I2C**"->"<Yes>"->"<OK>"->"<Finish>" und starten Sie den Raspberry pi neu. Jetzt können Sie überprüfen, ob die I2C-Schnittstelle erfolgreich geöffnet wurde.

Installieren Sie I2C-Tools

Geben Sie den folgenden Befehl ein, um I2C-Tools zu installieren:

```
sudo apt-get install i2c-tools
```

Geben Sie den folgenden Befehl zur Erkennung der I2C-Gerät Adresse ein:

```
i2cdetect -y 1
```

```
pi@raspberrypi:~ $ sudo apt-get install i2c-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
i2c-tools is already the newest version (4.1-1).
i2c-tools set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 131 not upgraded.
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: --
10: --
20: -- - - - - - - 27 - - - - - -
30: --
40: --
50: --
60: --
70: --
pi@raspberrypi:~ $
```

27 ist die I2C-Adresse von PCF8574.

C code

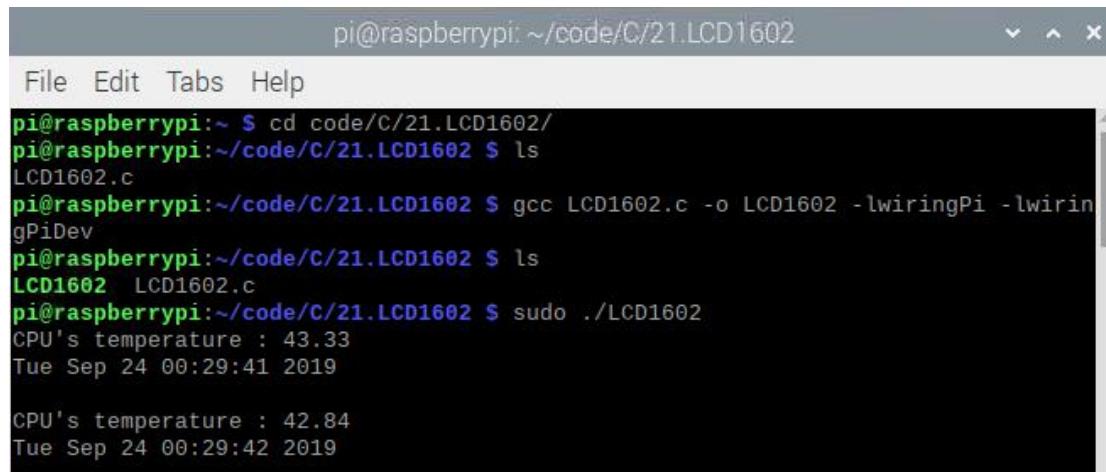
Öffnen Sie das Terminal und geben Sie den Befehl "**cd code / C / 21.LCD1602 /**" ein, um das Codeverzeichnis "LCD1602" aufzurufen;

Geben Sie den Befehl "`ls`" ein, um die Datei "LCD1602.c" im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/C/21.LCD1602
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/21.LCD1602/
pi@raspberrypi:~/code/C/21.LCD1602 $ ls
LCD1602.c
pi@raspberrypi:~/code/C/21.LCD1602 $
```

Geben Sie den Befehl "`gcc LCD1602.c -o LCD1602 -lwiringPi -lwiringPiDev`" ein, um die ausführbare Datei "LCD1602.c" "LCD1602" zu generieren, und geben Sie den Befehl "`ls`" ein, um sie anzuzeigen. Geben Sie den Befehl "`sudo ./LCD1602`" ein, um den Code auszuführen. Die Ergebnisse sind wie folgt:



```
pi@raspberrypi:~/code/C/21.LCD1602
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/21.LCD1602/
pi@raspberrypi:~/code/C/21.LCD1602 $ ls
LCD1602.c
pi@raspberrypi:~/code/C/21.LCD1602 $ gcc LCD1602.c -o LCD1602 -lwiringPi -lwiringPiDev
pi@raspberrypi:~/code/C/21.LCD1602 $ ls
LCD1602 LCD1602.c
pi@raspberrypi:~/code/C/21.LCD1602 $ sudo ./LCD1602
CPU's temperature : 43.33
Tue Sep 24 00:29:41 2019

CPU's temperature : 42.84
Tue Sep 24 00:29:42 2019
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <stdlib.h>
#include <stdio.h>
#include <wiringPi.h>
#include <pcf8574.h>
#include <lcd.h>
#include <time.h>

#define pcf8574_address 0x27      // default I2C address of Pcf8574
//#define pcf8574_address 0x3F      // default I2C address of Pcf8574A
#define BASE 64                  // BASE is not less than 64
```

////// Define the output pins of the PCF8574, which are directly connected to the LCD1602 pin.

```
#define RS      BASE+0
#define RW      BASE+1
#define EN      BASE+2
#define LED      BASE+3
#define D4      BASE+4
#define D5      BASE+5
#define D6      BASE+6
#define D7      BASE+7
```

```
int lcdhd;// used to handle LCD
```

```
void printCPUTemperature()// sub function used to print CPU temperature
```

```
FILE *fp;
char str_temp[15];
float CPU_temp;
// CPU temperature data is stored in this directory.
fp=fopen("/sys/class/thermal/thermal_zone0/temp","r");
fgets(str_temp,15,fp);      // read file temp
CPU_temp = atof(str_temp)/1000.0;    // convert to Celsius degrees
printf("CPU's temperature : %.2f \n",CPU_temp);
lcdPosition(lcdhd,0,0);      // set the LCD cursor position to (0,0)
lcdPrintf(lcdhd,"CPU:%.2fC",CPU_temp);// Display CPU temperature on LCD
fclose(fp);
}
void printDataTime()//used to print system time
{
time_t rawtime;
struct tm *timeinfo;
time(&rawtime);// get system time
timeinfo = localtime(&rawtime);// convert to local time
printf("%s \n",asctime(timeinfo));
lcdPosition(lcdhd,0,1);// set the LCD cursor position to (0,1)

lcdPrintf(lcdhd,"Time:%d:%d:%d",timeinfo->tm_hour,timeinfo->tm_min,timeinfo->
m_sec);
//Display system time on LCD
}
int main(void){
    int i;
```

```

if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
    printf("failed !");
    return 1;
}
pcf8574Setup(BASE,pcf8574_address); // initialize PCF8574
for(i=0;i<8;i++){
    pinMode(BASE+i,OUTPUT);      // set PCF8574 port to output mode
}
digitalWrite(LED,HIGH);        // turn on LCD backlight
digitalWrite(RW,LOW);         // allow writing to LCD
lcdhd = lcdInit(2,16,4,RS,EN,D4,D5,D6,D7,0,0,0,0); // initialize LCD and return
"handle" used to handle LCD
if(lcdhd == -1){
    printf("lcdInit failed !");
    return 1;
}
while(1){
    printCPUtemperature(); // print CPU temperature
    printDataTime();       // print system time
    delay(1000);
}
return 0;
}

```

Code Interpretation

Aus dem Code geht hervor, dass PCF8591 und PCF8574 viele Ähnlichkeiten aufweisen. Sie werden über die I2C-Schnittstelle verwendet, um das GPIO-RPI zu erweitern. Definiert zunächst die I2C-Adresse des PCF8574 und die Erweiterung des GPIO-Pins, der mit dem GPIO-Pin des LCD1602 verbunden ist.

```

#define pcf8574_address 0x27          // default I2C address of Pcf8574
//#define pcf8574_address 0x3F        // default I2C address of Pcf8574A
#define BASE 64                      // BASE is not less than 64
//////// Define the output pins of the PCF8574, which are directly connected to the
LCD1602 pin.
#define RS      BASE+0
#define RW      BASE+1
#define EN      BASE+2
#define LED     BASE+3

```

```
#define D4      BASE+4
#define D5      BASE+5
#define D6      BASE+6
#define D7      BASE+7
```

Initialisieren Sie dann in der Hauptfunktion den PCF8574, setzen Sie alle Pins in den Ausgabemodus und schalten Sie die Hintergrundbeleuchtung des LCD1602 ein.

```
pcf8574Setup(BASE,pcf8574_address);// initialize PCF8574
    for(i=0;i<8;i++){
        pinMode(BASE+i,OUTPUT);      // set PCF8574 port to output mode
    }
    digitalWrite(LED,HIGH);      // turn on LCD backlight
```

Verwenden Sie dann lcdInit (), um LCD1602 zu initialisieren, und setzen Sie den RW Pin von LCD1602 gemäß den Anforderungen dieser Funktion auf 0 (kann geschrieben werden). Der Rückgabewert der Funktion "Handle" wird verwendet, um LCD1602 zu behandeln.

```
lcdhd = lcdInit(2,16,4,RS,EN,D4,D5,D6,D7,0,0,0,0);
```

In der Hauptfunktion "while" werden zwei Unterfunktionen aufgerufen, um die CPU-Temperatur und die Uhrzeit anzuzeigen. Schauen Sie sich zuerst die Unterfunktion "printCPUTemperature ()" an. Die CPU Temperaturdaten werden in der Datei "/ sys / class / thermic / thermische_zone0 / temp" gespeichert. Wir müssen den Inhalt der Datei lesen und in einen Temperaturwert konvertieren, der in der Variablen "CPU_temp" gespeichert ist, und ihn mit "lcdPrintf ()" auf dem LCD anzeigen.

```
void printCPUTemperature(){// sub function used to print CPU temperature
    FILE *fp;
    char str_temp[15];
    float CPU_temp;
    // CPU temperature data is stored in this directory.
    fp=fopen("/sys/class/thermal/thermal_zone0/temp","r");
    fgets(str_temp,15,fp);      // read file temp
    CPU_temp = atof(str_temp)/1000.0;    // convert to Celsius degrees
    printf("CPU's temperature : %.2f\n",CPU_temp);
    lcdPosition(lcdhd,0,0);      // set the LCD cursor position to (0,0)
    lcdPrintf(lcdhd,"CPU:%.2fC",CPU_temp); // Display CPU temperature on LCD
    fclose(fp);
```

{}

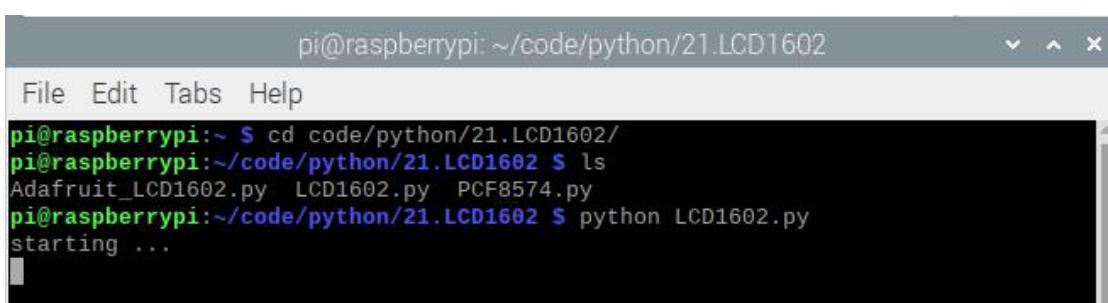
Als nächstes folgt die Unterfunktion "printDataTime ()" zum Drucken der Systemzeit. Holen Sie sich zuerst die Standardzeit und speichern Sie sie in variabler Rohzeit. Konvertieren Sie sie dann in die Ortszeit und zeigen Sie schließlich die Zeitinformationen auf dem LCD1602 an.

```
void printDataTime(){//used to print system time
    time_t rawtime;
    struct tm *timeinfo;
    time(&rawtime);// get system time
    timeinfo = localtime(&rawtime);// convert to local time
    printf("%s \n",asctime(timeinfo));
    lcdPosition(lcdhd,0,1);// set the LCD cursor position to (0,1)

lcdPrintf(lcdhd,"Time:%d:%d:%d",timeinfo->tm_hour,timeinfo->tm_min,timeinfo->t
m_sec);
//Display system time on LCD
}
```

Python code

1. Verwenden Sie den Befehl "[cd code / python / 21.LCD1602 /](#)", um das Verzeichnis von LCD1602 aufzurufen.
2. Verwenden Sie den Befehl "[python LCD1602.py](#)", um den Code "LCD1602.py" auszuführen.



A screenshot of a terminal window titled "pi@raspberrypi: ~ /code/python/21.LCD1602". The window shows the following command-line session:

```
pi@raspberrypi:~ $ cd code/python/21.LCD1602/
pi@raspberrypi:~/code/python/21.LCD1602 $ ls
Adafruit_LCD1602.py  LCD1602.py  PCF8574.py
pi@raspberrypi:~/code/python/21.LCD1602 $ python LCD1602.py
starting ...
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
from PCF8574 import PCF8574_GPIO
from Adafruit_LCD1602 import Adafruit_CharLCD

from time import sleep, strftime
from datetime import datetime

def get_cpu_temp():      # get CPU temperature and store it into file
    "/sys/class/thermal/thermal_zone0/temp"
    tmp = open('/sys/class/thermal/thermal_zone0/temp')
    cpu = tmp.read()
    tmp.close()
    return '{:.2f}'.format( float(cpu)/1000 ) + ' C'

def get_time_now():      # get system time
    return datetime.now().strftime(' %H:%M:%S')

def loop():
    mcp.output(3,1)      # turn on LCD backlight
    lcd.begin(16,2)      # set number of LCD lines and columns
    while(True):
        #lcd.clear()
        lcd.setCursor(0,0)  # set cursor position
        lcd.message( 'CPU: ' + get_cpu_temp()+'\n' )# display CPU temperature
        lcd.message( get_time_now() )    # display the time
        sleep(1)

def destroy():# When 'Ctrl+C' is pressed, the function is executed.
    lcd.clear()

PCF8574_address = 0x27  # I2C address of the PCF8574 chip.
PCF8574A_address = 0x3F  # I2C address of the PCF8574A chip.
# Create PCF8574 GPIO adapter.
try:
    mcp = PCF8574_GPIO(PCF8574_address)
except:
    try:
        mcp = PCF8574_GPIO(PCF8574A_address)
    except:
        print ('I2C Address Error !')
```

```
exit(1)
# Create LCD, passing in MCP GPIO adapter.
lcd = Adafruit_CharLCD(pin_rs=0, pin_e=2, pins_db=[4,5,6,7], GPIO=mcp)

if __name__ == '__main__':
    # Program starting from here
    print('starting ... ')
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

Code Interpretation

Der Code verwendet zwei Dateien, "PCF8574.py" und "Adafruit_LCD1602.py". Diese beiden Dateien und Codedateien werden im selben Verzeichnis gespeichert. Sie werden beide benötigt. Bitte löschen Sie sie nicht. "PCF8574.py" wird verwendet, um I2C-Kommunikationsmodi und Betriebsmethoden für bestimmte Ports für Raspberry Pi und PCF8574-Chips bereitzustellen. "Adafruit_LCD1602.py" wird verwendet, um einige Funktionsoperationsmethoden für LCD1602 bereitzustellen.

‘PCF8574.py’: Dieses Modul bietet zwei Klassen: PCF8574_I2C und PCF8574_GPIO. PCF8574_I2C-Klasse: Bietet Lese- und Schreibmethoden für PCF8574. PCF8574_GPIO-Klasse: Bietet eine Reihe standardisierter GPIO-Funktionen.

‘Adafruit_LCD1602.py’: Dieses Modul bietet die grundlegende Betriebsmethode von LCD1602, einschließlich der Adafruit_CharLCD-Klasse.

Im Code erhält zuerst 'mcp = PCF8574_GPIO(PCF8574_address)' das Objekt, das zum Betreiben des PCF8574-Ports verwendet wird, dann 'lcd = Adafruit_CharLCD (pin_rs = 0, pin_e = 2, pins_db = [4,5,6,7], GPIO = mcp)' Holen Sie sich das Objekt, mit dem der LCD1602 betrieben wird.

```
PCF8574_address = 0x27 # I2C address of the PCF8574 chip.
PCF8574A_address = 0x3F # I2C address of the PCF8574A chip.
# Create PCF8574 GPIO adapter.
try:
    mcp = PCF8574_GPIO(PCF8574_address)
except:
```

```

try:
    mcp = PCF8574_GPIO(PCF8574A_address)
except:
    print ('I2C Address Error !')
    exit(1)
# Create LCD, passing in MCP GPIO adapter.
lcd = Adafruit_CharLCD(pin_rs=0, pin_e=2, pins_db=[4,5,6,7], GPIO=mcp)

```

Entsprechend der Schaltungsverbindung ist Port 3 des PCF8574 mit dem Pluspol der Hintergrundbeleuchtung des LCD1602 verbunden. Verwenden Sie dann in der Funktion loop () mcp.output (3,1), um die Hintergrundbeleuchtung des LCD1602 einzuschalten und die Anzahl der LCD Zeilen und Spalten festzulegen.

```

def loop():
    mcp.output(3,1)      # turn on LCD backlight
    lcd.begin(16,2)      # set number of LCD lines and columns

```

Stellen Sie im nächsten "while" Zyklus die Cursorposition ein und zeigen Sie die CPU Temperatur und Zeit an.

```

while(True):
    #lcd.clear()
    lcd.setCursor(0,0)  # set cursor position
    lcd.message( 'CPU: ' + get_cpu_temp()+'\n')# display CPU temperature
    lcd.message( get_time_now() )   # display the time
    sleep(1)

```

Die CPU Temperatur wird in der Datei “/ sys / class / Thermal / Thermal_Zone0 / Temp” gespeichert. Mit "tmp = open ('/ sys / class / thermic / thermische_zone0 / temp')" wird die Datei geöffnet, und mit "cpu = tmp.read ()" wird der Inhalt der Datei gelesen und anschließend in konvertiert Grad Celsius und zurück.

```

def get_cpu_temp():                      # get CPU temperature and store it into
file
    #"/sys/class/thermal/thermal_zone0/temp"
    tmp = open('/sys/class/thermal/thermal_zone0/temp')
    cpu = tmp.read()
    tmp.close()
    return '{:.2f}'.format( float(cpu)/1000 ) + ' C'

```

‘def get_time_now()’ ist eine Unterfunktion zum Abrufen von Zeit:

```
def get_time_now():      # get system time  
    return datetime.now().strftime('%H:%M:%S')
```

Lektion 22 DHT11

Überblick

In dieser Lektion lernen Sie, wie Sie mit DHT11 Temperatur und Luftfeuchtigkeit messen.

Erforderliche Teile

1 x Raspberry pi

1 x 10K Widerstand

1 x Steckbrett

1 x DHT11 Modul

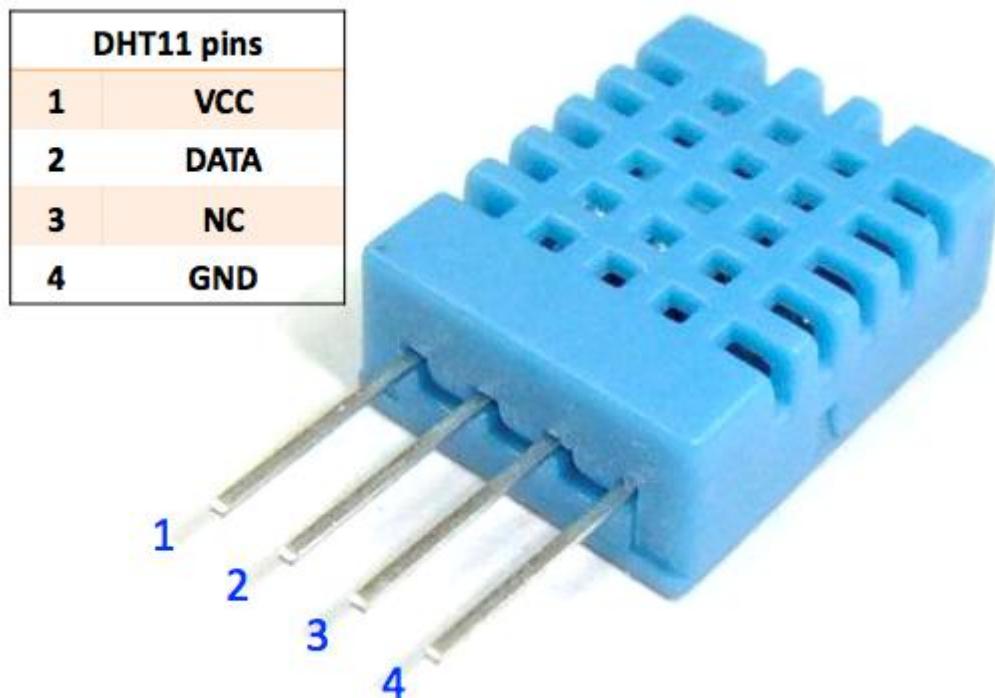
Einige Jumper Kabel

Produkt Einführung

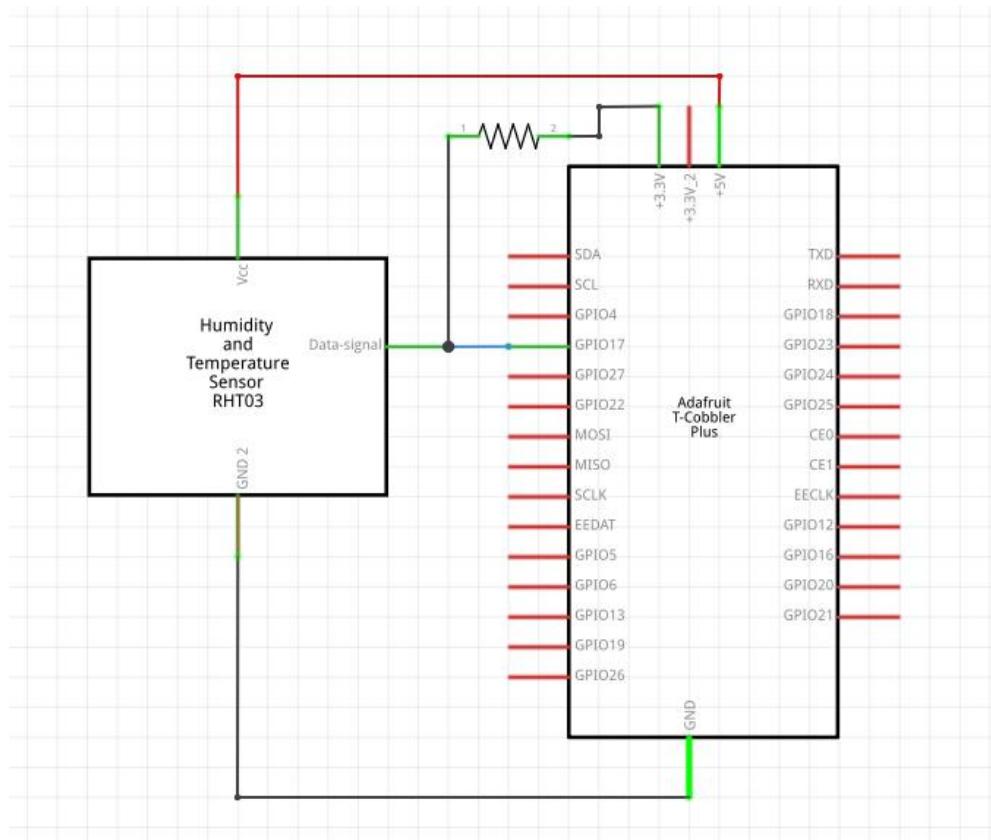
Der digitale Temperatur- und Feuchtigkeitssensor DHT11 ist ein Verbundsensor, der eine kalibrierte digitale Signalausgabe für Temperatur und Luftfeuchtigkeit enthält. Mithilfe der speziellen Technologie zur Erfassung digitaler Module sowie der Technologie zur Erfassung von Temperatur und Feuchtigkeit wird sichergestellt, dass das Produkt eine hohe Zuverlässigkeit und eine hervorragende Langzeitstabilität aufweist. Der Sensor enthält Komponenten für die resistente Luftfeuchtigkeit und

NTC Temperaturmessgeräte und ist an einen Hochleistungs-8-Bit-Mikrocontroller angeschlossen.

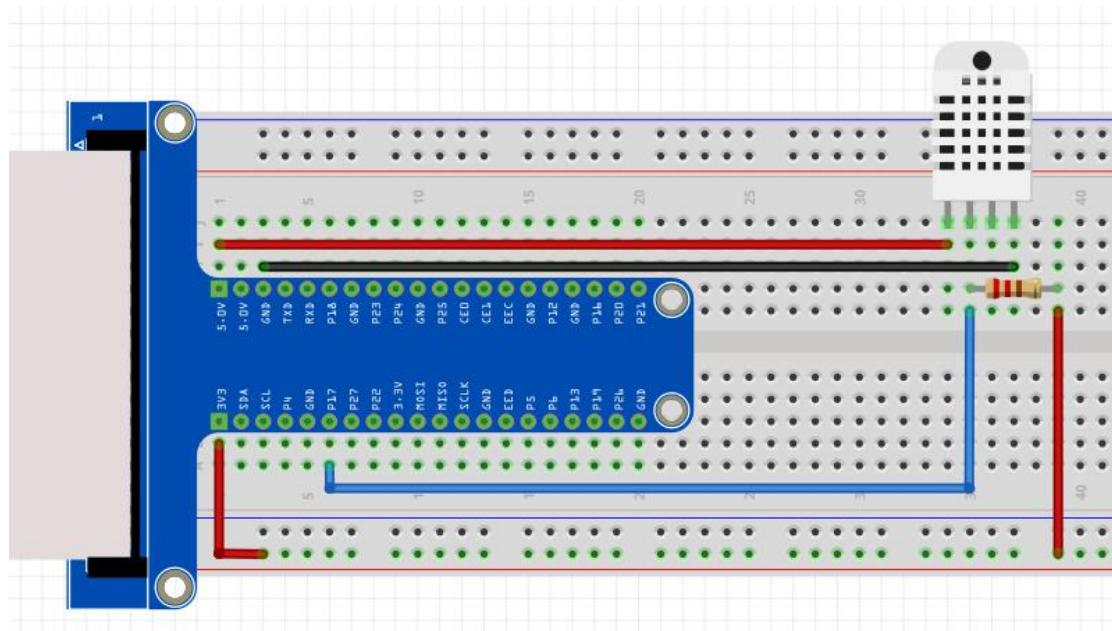
Der DHT11 Sensor verfügt nur über drei Pins: VCC, GND und DATA. Der Kommunikationsprozess beginnt von der DATA-Leitung, die das Startsignal an DHT11 sendet, DHT11 empfängt das Signal und gibt ein Antwortsignal zurück, und dann empfängt der Host das Antwortsignal und beginnt mit dem Empfang von 40-Bit-Temperatur- und Feuchtigkeitsdaten (8-stellige Feuchtigkeits Ganzzahl + 8-stellige Feuchtigkeits-Dezimalzahl + 8-stellige Temperatur Ganzzahl + 8-ZiffernTemperatur dezimal + 8-stellige Prüfsumme).



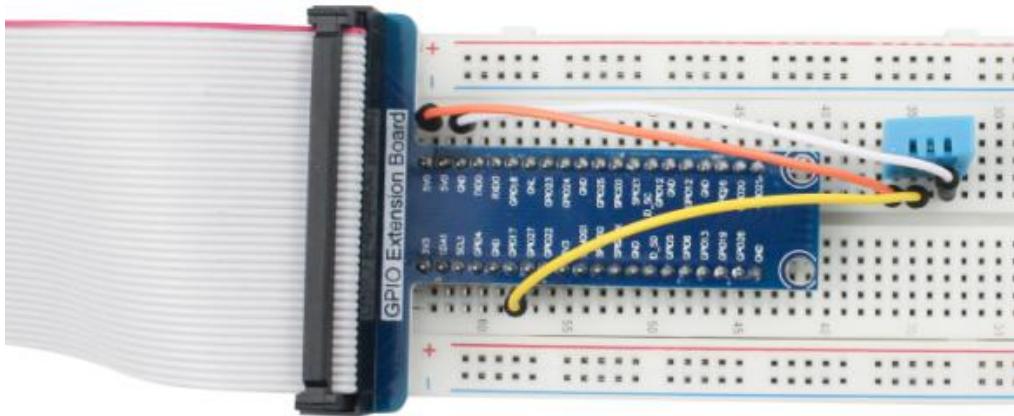
Schaltplan



Verdrahtung Diagramm



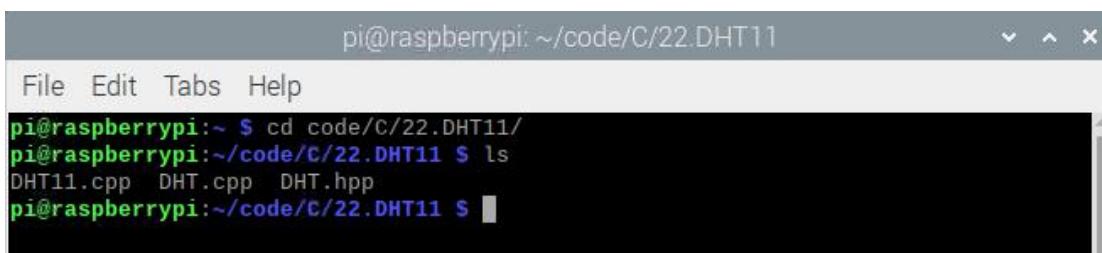
Beispiel Abbildung



C code

Öffnen Sie das Terminal und geben Sie den Befehl "`cd code / C / 22.DHT11 /`" ein, um das Codeverzeichnis "DHT11" aufzurufen;

Geben Sie den Befehl "`ls`" ein, um die Dateien "DHT11.cpp", "DHT.cpp" und "DHT.hpp" im Verzeichnis anzuseigen;



```
pi@raspberrypi:~/code/C/22.DHT11
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/22.DHT11/
pi@raspberrypi:~/code/C/22.DHT11 $ ls
DHT11.cpp  DHT.cpp  DHT.hpp
pi@raspberrypi:~/code/C/22.DHT11 $
```

Geben Sie den Befehl "`gcc DHT11.cpp DHT.cpp -o DHT11 -lwiringPi`" ein, um die ausführbare Datei "DHT11.cpp" und "DHT.cpp" "DHT11" zu generieren. Geben Sie den Befehl "`ls`" ein, um sie anzuseigen. Geben Sie den Befehl "`sudo ./DHT11`" ein, um den Code auszuführen.

Die Ergebnisse sind wie folgt:

```
pi@raspberrypi:~/code/C/22.DHT11
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/22.DHT11/
pi@raspberrypi:~/code/C/22.DHT11 $ ls
DHT11.cpp DHT.cpp DHT.hpp
pi@raspberrypi:~/code/C/22.DHT11 $ gcc DHT11.cpp DHT.cpp -o DHT11 -lwiringPi
pi@raspberrypi:~/code/C/22.DHT11 $ ls
DHT11 DHT11.cpp DHT.cpp DHT.hpp
pi@raspberrypi:~/code/C/22.DHT11 $ sudo ./DHT11
The sumCnt is : 1
DHTLIB_ERROR_TIMEOUT!
Humidity is -999.00 %, Temperature is -999.00 *C

The sumCnt is : 2
DHTLIB_ERROR_TIMEOUT!
Humidity is -999.00 %, Temperature is -999.00 *C

The sumCnt is : 3
DHT11,OK!
Humidity is 45.00 %, Temperature is 28.00 *C

The sumCnt is : 4
DHT11,OK!
Humidity is 44.00 %, Temperature is 27.80 *C
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdint.h>
#include "DHT.hpp"

#define DHT11_Pin 0 //define the pin of sensor

int main(){
    DHT dht; //create a DHT class object
    int chk,sumCnt;//chk:read the return value of sensor; sumCnt:times of reading
    sensor
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("failed !");
        return 1;
    }
    while(1){
        chk = dht.readDHT11(DHT11_Pin); //read DHT11 and get a return value.
        Then determine whether data read is normal according to the return value.
```

```

        sumCnt++;           //counting number of reading times
        printf("The sumCnt is : %d \n",sumCnt);
        switch(chk){
            case DHTLIB_OK:      //if the return value is DHTLIB_OK, the data
is normal.
                printf("DHT11,OK! \n");
                break;
            case DHTLIB_ERROR_CHECKSUM:    //data check has errors
                printf("DHTLIB_ERROR_CHECKSUM! \n");
                break;
            case DHTLIB_ERROR_TIMEOUT:     //reading DHT times out
                printf("DHTLIB_ERROR_TIMEOUT! \n");
                break;
            case DHTLIB_INVALID_VALUE:    //other errors
                printf("DHTLIB_INVALID_VALUE! \n");
                break;
        }
        printf("Humidity is %.2f %%,\t Temperature is %.2f
*C\n\n",dht.humidity,dht.temperature);
        delay(2000);
    }
    return 1;
}

```

Code Interpretation

In diesem Projektcode verwenden wir eine benutzer definierte Bibliotheksdatei "DHT.hpp". Es befindet sich im selben Verzeichnis mit den Programmdateien "DHT11.cpp" und "DHT.cpp". In der Bibliotheksdatei finden Sie Methoden zum Lesen des DHT Sensors. Mit dieser Bibliothek können wir den DHT Sensor leicht ablesen. Erstellen Sie zuerst ein DHT Klassenobjekt im Code.

```

int main(){
    DHT dht;          //create a DHT class object

```

Im Zyklus "while" verwenden Sie "chk = dht.readDHT11(DHT11_Pin)", um den DHT11 zu lesen, und bestimmen Sie, ob die gelesenen Daten gemäß dem Rückgabewert "chk" normal sind. Verwenden Sie dann die Variable "sumCnt", um die Anzahl der Lesezeiten aufzuzeichnen.

```

while(1){

```

```
chk = dht.readDHT11(DHT11_Pin); //read DHT11 and get a return value.  
Then determine whether data read is normal according to the return value.  
    sumCnt++;           //counting number of reading times  
    printf("The sumCnt is : %d \n",sumCnt);  
    switch(chk){  
        case DHTLIB_OK:      //if the return value is DHTLIB_OK, the data  
is normal.  
            printf("DHT11,OK! \n");  
            break;  
        case DHTLIB_ERROR_CHECKSUM:    //data check has errors  
            printf("DHTLIB_ERROR_CHECKSUM! \n");  
            break;  
        case DHTLIB_ERROR_TIMEOUT:     //reading DHT times out  
            printf("DHTLIB_ERROR_TIMEOUT! \n");  
            break;  
        case DHTLIB_INVALID_VALUE:    //other errors  
            printf("DHTLIB_INVALID_VALUE! \n");  
            break;  
    }  
}
```

Drucken Sie abschließend die Ergebnisse aus:

```
printf("Humidity is %.2f %%, \t Temperature is %.2f  
*C\n\n",dht.humidity,dht.temperature);
```

Die Bibliotheksdatei "DHT.hpp" enthält eine DHT Klasse und seine öffentlichen Mitgliedsfunktionen int readDHT11 (int pin) werden verwendet, um den Sensor DHT11 zu lesen und die Temperatur- und Feuchtigkeitsdaten zu speichern, die in den Mitgliedsvariablen doublehumidity und temperatur gelesen wurden. Die Implementierungsmethode der Funktion ist in der Datei "DHT.cpp" enthalten.

```
#include <wiringPi.h>  
#include <stdio.h>  
#include <stdint.h>  
  
///read return flag of sensor  
#define DHTLIB_OK          0  
#define DHTLIB_ERROR_CHECKSUM -1  
#define DHTLIB_ERROR_TIMEOUT -2  
#define DHTLIB_INVALID_VALUE -999
```

```
#define DHTLIB_DHT11_WAKEUP      18
#define DHTLIB_DHT_WAKEUP        1

#define DHTLIB_TIMEOUT           100

class DHT{
public:
    double humidity,temperature; //use to store temperature and humidity
data read
    int readDHT11(int pin); //read DHT11
private:
    uint8_t bits[5]; //Buffer to receiver data
    int readSensor(int pin,int wakeupDelay); //
};

};
```

Python code

1. Geben Sie mit dem Befehl "[cd code / python / 22.DHT11 /](#)" das Verzeichnis des Temperatur- und Feuchtigkeitssensors ein.
2. Verwenden Sie den Befehl "[python DHT11.py](#)", um den Code "DHT11.py" auszuführen.

Nach Ausführung des Programms zeigt das Terminalfenster die aktuelle Lesetemperatur und Luftfeuchtigkeit sowie die Temperatur- und Luftfeuchtigkeitswerte an.

Wie nachfolgend dargestellt:

```
pi@raspberrypi:~/code/python/22.DHT11
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/22.DHT11/
pi@raspberrypi:~/code/python/22.DHT11 $ python DHT11.py
starting ...
The sumCnt is : 1,      chk    : -2
DHTLIB_ERROR_TIMEOUT!
Humidity : -999.00,    Temperature : -999.00

The sumCnt is : 2,      chk    : 0
DHT11,OK!
Humidity : 49.00,      Temperature : 28.30
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import time
import DHT as DHT
DHTPin = 11      #define the pin of DHT11
def loop():
    dht = DHT.DHT(DHTPin)    #create a DHT class object
    sumCnt = 0                #number of reading times
    while(True):
        sumCnt += 1            #counting number of reading times
        chk = dht.readDHT11()   #read DHT11 and get a return value. Then
        determine whether data read is normal according to the return value.
        print ("The sumCnt is : %d, \t chk    : %d"%(sumCnt,chk))
        if(chk is dht.DHTLIB_OK):      #read DHT11 and get a return value.
        Then determine whether data read is normal according to the return value.
            print("DHT11,OK!")
        elif(chk is dht.DHTLIB_ERROR_CHECKSUM): #data check has errors
            print("DHTLIB_ERROR_CHECKSUM!!")
        elif(chk is dht.DHTLIB_ERROR_TIMEOUT):  #reading DHT times out
            print("DHTLIB_ERROR_TIMEOUT!")
        else:                      #other errors
            print("Other error!")

        print("Humidity : %.2f, \t Temperature : %.2f"
```

```
\n"%(dht.humidity,dht.temperature))\n    time.sleep(3)\n\nif __name__ == '__main__':\n    print ('starting ... ')\n    try:\n        loop()\n    except KeyboardInterrupt:\n        GPIO.cleanup()\n        exit()
```

Code Interpretation

In diesem Projektcode verwenden wir ein Modul "DHT.py", das eine Methode zum Lesen des Sensor-DHT bereitstellt. Es befindet sich im selben Verzeichnis wie die Programmdatei "DHT11.py". Mit dieser Bibliothek können wir DHT-Sensoren leicht lesen.

‘DHT.py’ : Dies ist ein Python Modul zum Lesen von Temperatur- und Feuchtigkeitsdaten von DHT-Sensoren. Lokale Funktionen und Variablen werden wie folgt beschrieben:

- Variable Luftfeuchtigkeit: Speichert die vom Sensor gelesenen Luftfeuchtigkeitsdaten.
- Variable Temperatur: Speichert die vom Sensor gelesenen Temperaturdaten.

def readDHT11 (Pin): Lesen Sie die Temperatur und Luftfeuchtigkeit des Sensors DHT11 ab und geben Sie den Wert zurück, mit dem festgestellt wird, ob die Daten normal sind.

Das Modul "DHT.py" enthält DHT-Klassen. Und die Klassenfunktion def readDHT11 (Pin) wird verwendet, um den Sensor DHT11 zu lesen und die gelesenen Temperatur- und Feuchtigkeitsdaten in der variablen Feuchtigkeit und Temperatur des Mitglieds zu speichern.

Erstellen Sie zuerst ein DHT Klassenobjekt im Code.
dht = DHT.DHT(DHTPin) #create a DHT class object

Verwenden Sie dann im Zyklus "while" "chk = dht.readDHT11 (DHT11Pin)", um den DHT11 zu lesen, und bestimmen Sie, ob die gelesenen Daten gemäß dem Rückgabewert "chk" normal sind. Verwenden Sie dann die Variable "sumCnt", um die Anzahl der Lesezeiten aufzuzeichnen.

while(True):

```
    sumCnt += 1          #counting number of reading times
    chk = dht.readDHT11()      #read DHT11 and get a return value. Then
determine whether data read is normal according to the return value.
    print ("The sumCnt is : %d, \t chk      : %d"%(sumCnt,chk))
    if(chk is dht.DHTLIB_OK):      #read DHT11 and get a return value.
Then determine whether data read is normal according to the return value.
        print("DHT11,OK!")
    elif(chk is dht.DHTLIB_ERROR_CHECKSUM): #data check has errors
        print("DHTLIB_ERROR_CHECKSUM!!")
    elif(chk is dht.DHTLIB_ERROR_TIMEOUT):  #reading DHT times out
        print("DHTLIB_ERROR_TIMEOUT!")
    else:                      #other errors
        print("Other error!")
```

Drucken Sie abschließend die Ergebnisse aus:

```
print("Humidity : %.2f, \t Temperature : %.2f \n"%(dht.humidity,dht.temperature))
```

Lesson 23 Matrix Keypad

Überblick

In dieser Lektion lernen Sie, wie Sie mit dem Raspberry Pi die Matrixtastatur benutzen.

Erforderliche Teile

1 x Raspberry Pi

1 x Matrix Tastatur

8 x doppelte männliche Jumper Kabel

1 x Steckbrett

Produkt Einführung

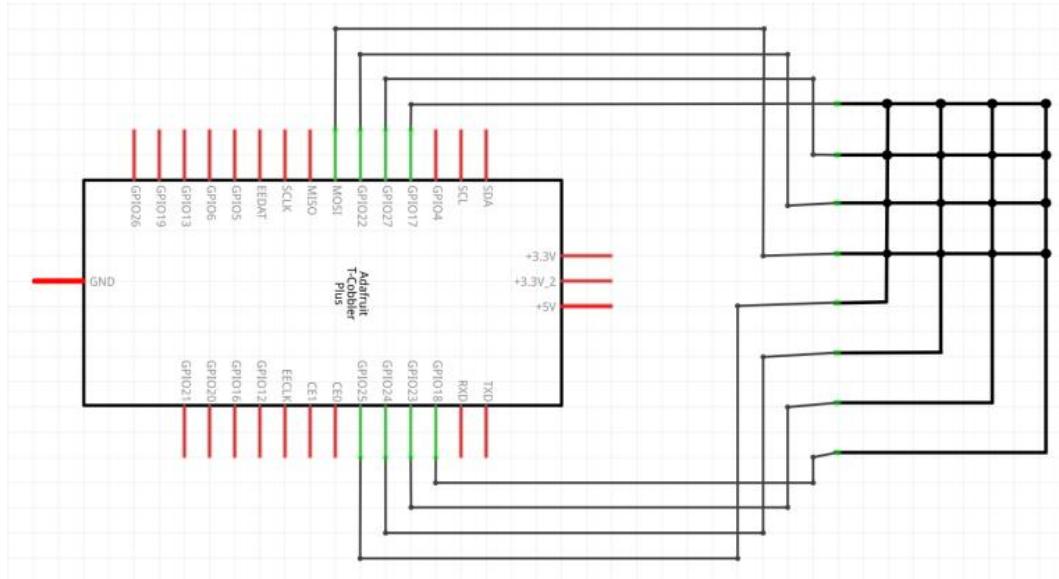
Matrix Keypad

Die Tastatur ist ein Gerät, das eine Reihe von Tasten integriert. Wie bei der Integration der LED Matrix ist bei der 4x4 Tastatur jede Tastenreihe mit einem Pin verbunden und entspricht jeder Spalte. Eine solche Verbindung kann die Belegung des Prozessorports verringern.

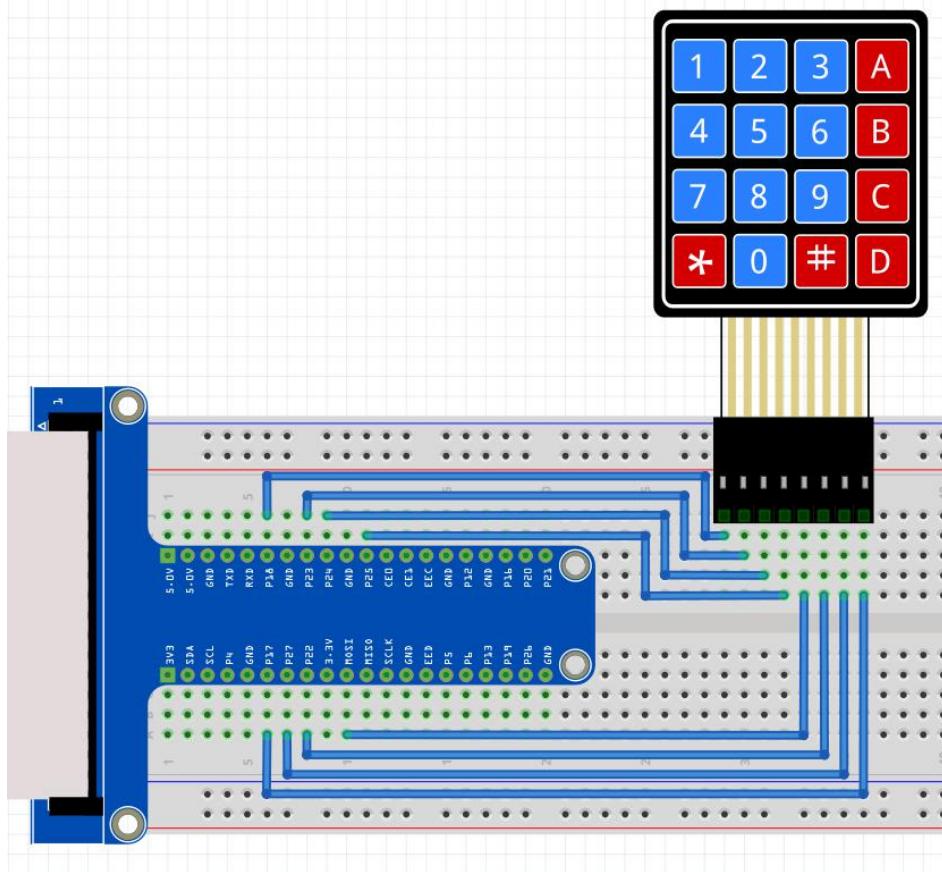




Schaltplan

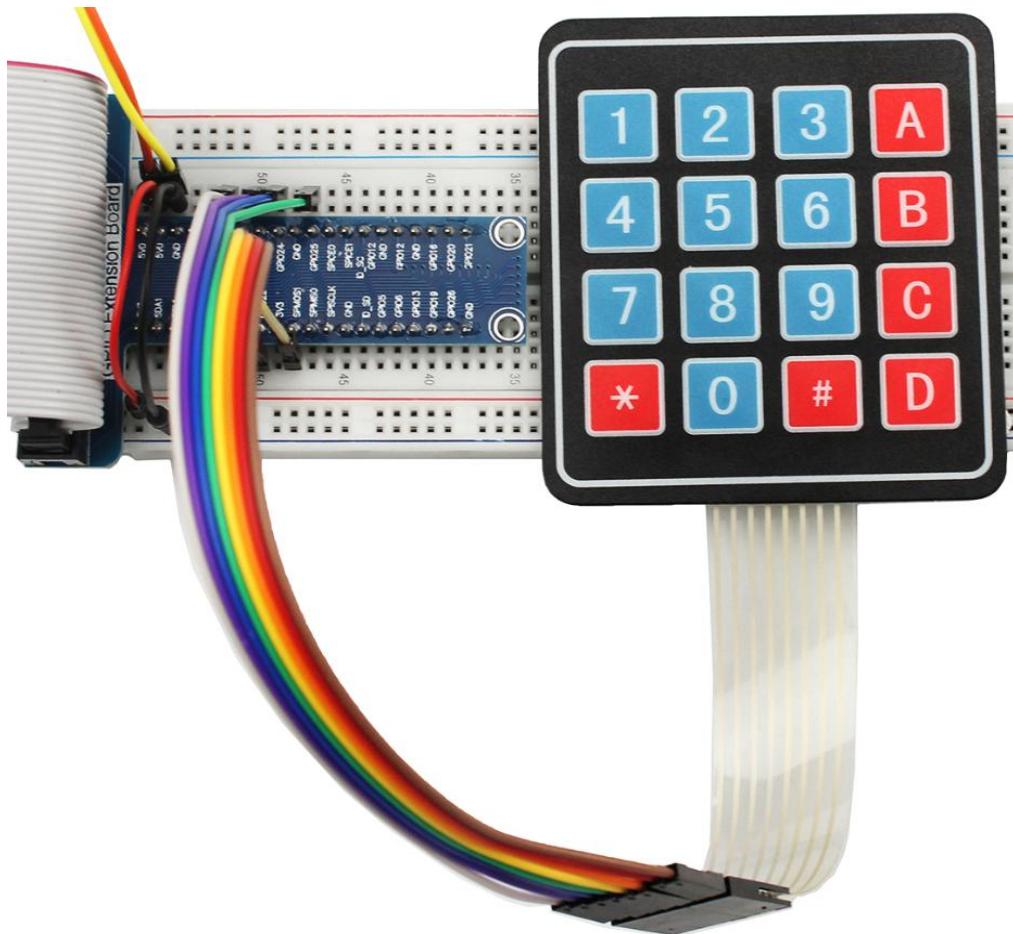


Verdrahtung Diagramm





Beispiel Abbildung



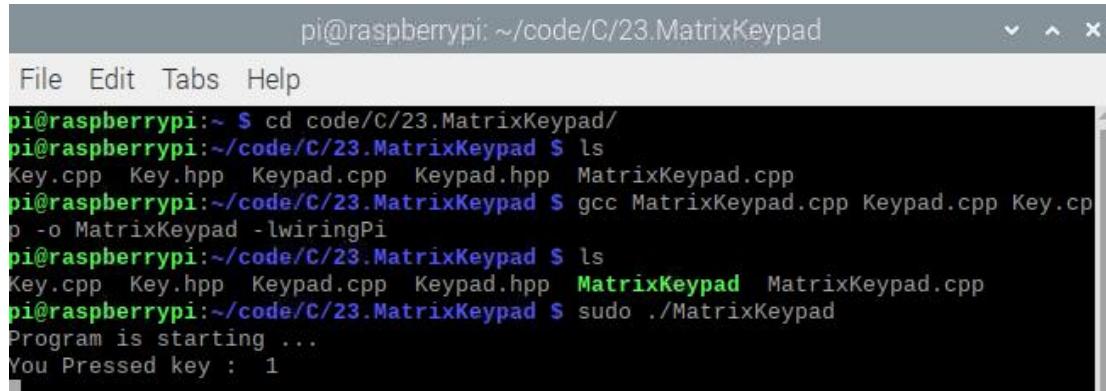
C code

Öffnen Sie das Terminal und geben Sie den Befehl "cd code / C / 23.MatrixKeypad /" ein, um das Codeverzeichnis "MatrixKeypad" aufzurufen;

Geben Sie den Befehl "ls" ein, um die Dateien "Key.cpp", "Key.hpp", "Keypad.cpp", "Keypad.hpp" und "MatrixKeypad.cpp" im Verzeichnis anzuzeigen;

```
pi@raspberrypi:~/code/C/23.MatrixKeypad
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/23.MatrixKeypad/
pi@raspberrypi:~/code/C/23.MatrixKeypad$ ls
Key.cpp  Key.hpp  Keypad.cpp  Keypad.hpp  MatrixKeypad.cpp
pi@raspberrypi:~/code/C/23.MatrixKeypad$
```

Geben Sie den Befehl "["gcc MatrixKeypad.cpp Keypad.cpp Key.cpp -o MatrixKeypad -lwiringPi"](#)" ein, generieren Sie "Key.cpp", "Key.hpp", "Keypad.cpp", "Keypad.hpp" und "MatrixKeypad.cpp" "Geben Sie in ausführbare Dateien" MatrixKeypad "den Befehl" ls "ein, um ihn anzuzeigen. Geben Sie "["sudo ./ Der Befehl MatrixKeypad"](#)" ein, um den Code auszuführen. Die Ergebnisse lauten wie folgt:



```

pi@raspberrypi: ~ cd code/C/23.MatrixKeypad
pi@raspberrypi:~/code/C/23.MatrixKeypad $ ls
Key.cpp Key.hpp Keypad.cpp Keypad.hpp MatrixKeypad.cpp
pi@raspberrypi:~/code/C/23.MatrixKeypad $ gcc MatrixKeypad.cpp Keypad.cpp Key.cpp -o MatrixKeypad -lwiringPi
pi@raspberrypi:~/code/C/23.MatrixKeypad $ ls
Key.cpp Key.hpp Keypad.cpp Keypad.hpp MatrixKeypad MatrixKeypad.cpp
pi@raspberrypi:~/code/C/23.MatrixKeypad $ sudo ./MatrixKeypad
Program is starting ...
You Pressed key : 1

```

Dieser Code wird verwendet, um alle Tastencodes der 4x4-Matrixtastatur abzurufen. Wenn eine der Tasten gedrückt wird, wird der Tastencode im Terminalfenster gedruckt.

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```

#include "Keypad.hpp"
#include <stdio.h>
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
char keys[ROWS][COLS] = { //key code
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}
};
byte rowPins[ROWS] = {1, 4, 5, 6 }; //connect to the row pinouts of the keypad
byte colPins[COLS] = {12,3, 2, 0 }; //connect to the column pinouts of the keypad
//create Keypad object
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

int main(){
    printf("Program is starting ... \n");

```

```

if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
    printf("setup wiringPi failed !");
    return 1;
}
char key = 0;
keypad.setDebounceTime(50);
while(1){
    key = keypad.getKey(); //get the state of keys
    if (key){           //if a key is pressed, print out its key code
        printf("You Pressed key : %c \n",key);
    }
}
return 1;
}

```

Code Interpretation

In diesem Projektcode verwenden wir zwei benutzerdefinierte Bibliotheksdateien "Keypad.hpp" und "Key.hpp". Sie befinden sich im selben Verzeichnis wie die Programmdateien "MatrixKeypad.cpp", "Keypad.cpp" und "Key.cpp". Die Bibliothekstastatur wird aus der Arduino Bibliothekstastatur transplantiert. Und diese Bibliotheksdatei bietet eine Methode zum Lesen der Tastatur. Mithilfe dieser Bibliothek können wir die Matrixtastatur leicht lesen. Definieren Sie zunächst die Informationen der in diesem Projekt verwendeten Matrixtastatur: die Anzahl der Zeilen und Spalten, den Code jeder Taste und den GPIO Pin, der mit jeder Spalte und jeder Zeile verbunden ist. Es ist erforderlich, die Header-Datei "Keypad.hpp" einzuschließen.

```

#include "Keypad.hpp"
#include <stdio.h>
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
char keys[ROWS][COLS] = { //key code
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}
};
byte rowPins[ROWS] = {1, 4, 5, 6 }; //connect to the row pinouts of the keypad
byte colPins[COLS] = {12,3, 2, 0 }; //connect to the column pinouts of the keypad

```

Instanziieren Sie dann basierend auf den obigen Informationen ein Keypad Klassenobjekt, um die Matrixtastatur zu bedienen.

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

Stellen Sie die entprelle Zeit auf 50 ms ein, und dieser Wert kann basierend auf der tatsächlichen Verwendung der Tastatur flexibel mit einer Standardzeit von 10 ms eingestellt werden.

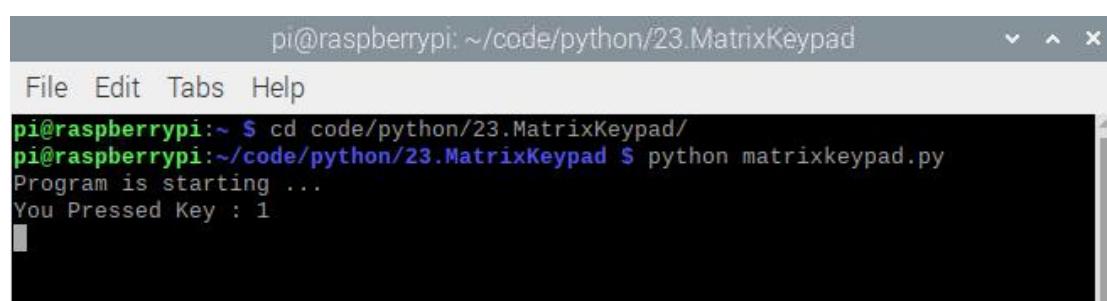
```
keypad.setDebounceTime(50);
```

Verwenden Sie im Zyklus "while" die Funktion "key = keypad.getKey()", um die Tastatur ständig zu lesen. Wenn eine Taste gedrückt wird, wird ihr Tastencode in der Variablen "Key" gespeichert und dann ausgedruckt.

```
while(1){  
    key = keypad.getKey(); //get the state of keys  
    if (key){ //if a key is pressed, print out its key code  
        printf("You Pressed key : %c \n",key);  
    }  
}
```

Python code

1. Verwenden Sie den Befehl "[cd code / python / 23.MatrixKeypad /](#)", um das Verzeichnis der Matrixschlüssel einzugeben;
2. Verwenden Sie den Befehl "[python matrixkeypad.py](#)", um den Code "matrixkeypad.py" auszuführen.



The screenshot shows a terminal window titled "pi@raspberrypi: ~ /code/python/23.MatrixKeypad". The window contains the following text:

```
pi@raspberrypi: ~ /code/python/23.MatrixKeypad  
File Edit Tabs Help  
pi@raspberrypi: ~ $ cd code/python/23.MatrixKeypad/  
pi@raspberrypi: ~/code/python/23.MatrixKeypad $ python matrixkeypad.py  
Program is starting ...  
You Pressed Key : 1
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import Keypad
ROWS = 4
COLS = 4
keys = [    '1','2','3','A',
           '4','5','6','B',
           '7','8','9','C',
           '*', '0','#','D'      ]
rowsPins = [12,16,18,22]
colsPins = [19,15,13,11]
def loop():
    keypad = Keypad.Keypad(keys,rowsPins,colsPins,ROWS,COLS)
    keypad.setDebounceTime(50)
    while(True):
        key = keypad.getKey()
        if(key != keypad.NULL):
            print ("You Pressed Key : %c "%(key))
    if __name__ == '__main__':
        print ("Program is starting ... ")
        try:
            loop()
        except KeyboardInterrupt:
            GPIO.cleanup()
```

Code Interpretation

```
import RPi.GPIO as GPIO
import Keypad
ROWS = 4
COLS = 4
keys = [    '1','2','3','A',
           '4','5','6','B',
           '7','8','9','C',
           '*', '0','#','D'      ]
rowsPins = [12,16,18,22]
colsPins = [19,15,13,11]
```

Instanziieren Sie dann basierend auf den obigen Informationen ein Keypad Klassenobjekt, um die Matrixtastatur zu bedienen.

def loop():

```
keypad = Keypad.Keypad(keys,rowsPins,colsPins,ROWS,COLS)
keypad.setDebounceTime(50)
```

Stellen Sie die entprelle Zeit auf 50 ms ein, und dieser Wert kann basierend auf der tatsächlichen Verwendung der Tastatur flexibel mit einer Standardzeit von 10 ms eingestellt werden.

while(True):

```
key = keypad.getKey()
if(key != keypad.NULL):
    print ("You Pressed Key : %c %(key))
```

Verwenden Sie im Zyklus "while" die Funktion "key = keypad.getKey()", um die Tastatur ständig zu lesen. Wenn eine Taste gedrückt wird, wird ihr Tastencode in der Variablen "Key" gespeichert und dann ausgedruckt.

Lektion 24 Ultraschall

Überblick

In dieser Lektion lernen Sie, wie Sie mit dem Ultraschallmodul die Entfernung messen und die Daten im Terminal ausdrucken.

Erforderliche Teile

1 x Raspberry Pi Entwicklungs-Board

1 x Ultraschallsensor

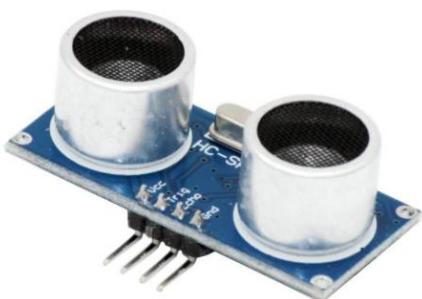
4 x Doppelt Männlich Jumper Kabel

1 x Steckbrett

Produkt Einführung

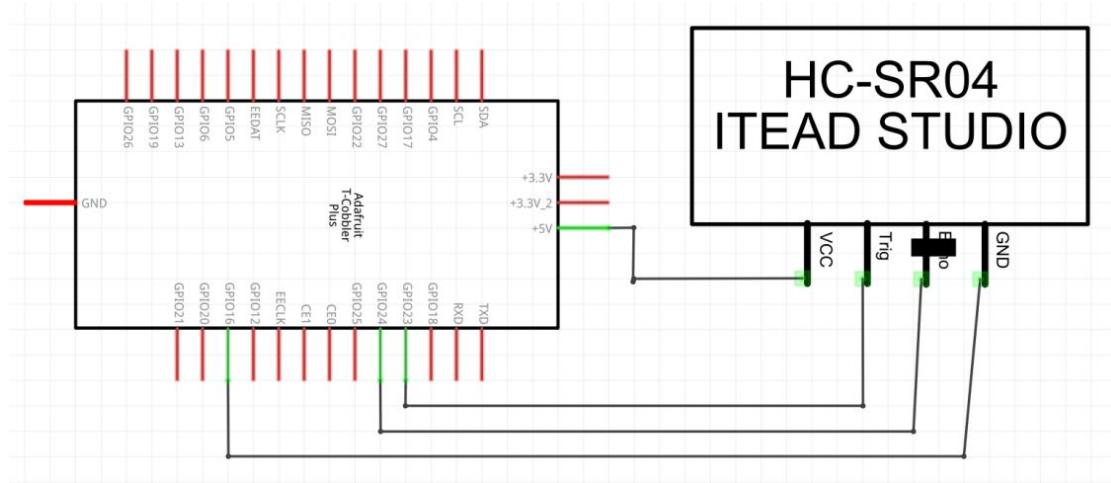
Ultrasonic ranging

HC-SR04 Ultrasonic Sensor ist ein Gerät, das als Abstandssensor in der Lage ist, Objekte berührungslos zu erkennen und ihre Entfernung zum Sensor zu messen. Die Abstandserfassung Ultraschallsensoren arbeiten nach dem Prinzip der Laufzeitmessung von Schallimpulsen einer bestimmten Frequenz. Wenn Ultraschallimpuls auf einen Gegenstand trifft, wird er reflektiert. Dieses Echo wird vom Sensor wieder aufgenommen und aus der Zeitspanne zwischen dem Senden und dem Empfangen des Schallimpulses, wird der Abstand zum Objekt berechnet.

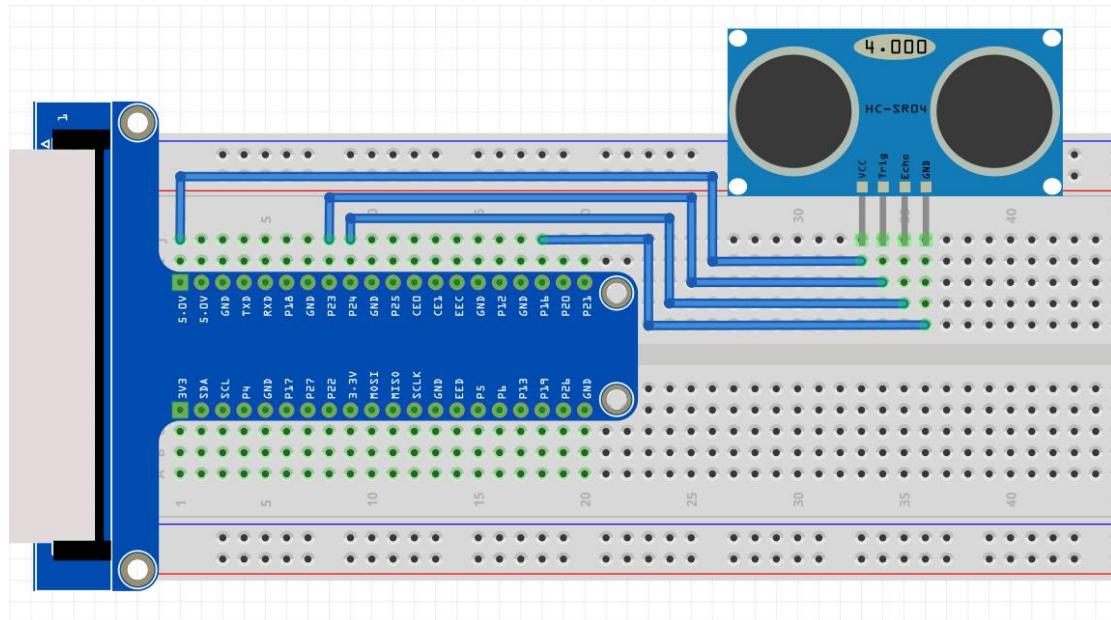




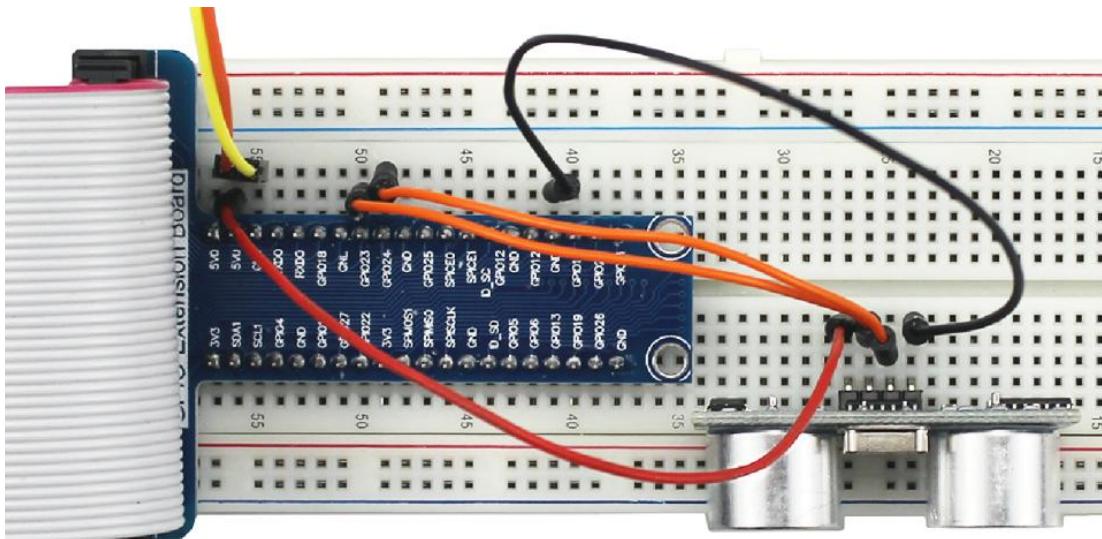
Schaltplan



Verdrahtung Diagramm



Beispiel Abbildung



C code

Öffnen Sie das Terminal und geben Sie den Befehl "cd code / C / 24.UltrasonicRanging /" ein, um das Codeverzeichnis "UltrasonicRanging" aufzurufen;

Geben Sie den Befehl "ls" ein, um die Datei "ultrasonicranging.c" im Verzeichnis anzuzeigen;

```
pi@raspberrypi:~/code/C/24.UltrasonicRanging
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/24.UltrasonicRanging/
pi@raspberrypi:~/code/C/24.UltrasonicRanging$ ls
ultrasonicranging.c
pi@raspberrypi:~/code/C/24.UltrasonicRanging$
```

Geben Sie den Befehl "gcc ultrasonicranging.c -o ultrasonicranging -lwiringPi" ein, um die ausführbare Datei "ultrasonicranging .c" zu generieren. Geben Sie den Befehl "ls" ein, um sie anzuzeigen. Geben Sie den Befehl "sudo ./ultrasonicranging" ein, um den Code auszuführen.

Die Ergebnisse sind unten gezeigt:

```
pi@raspberrypi:~/code/C/24.UltrasonicRanging
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/24.UltrasonicRanging/
pi@raspberrypi:~/code/C/24.UltrasonicRanging $ ls
ultrasonicranging.c
pi@raspberrypi:~/code/C/24.UltrasonicRanging $ gcc ultrasonicranging.c -o ultrasonicranging -lwiringPi
pi@raspberrypi:~/code/C/24.UltrasonicRanging $ ls
ultrasonicranging ultrasonicranging.c
pi@raspberrypi:~/code/C/24.UltrasonicRanging $ sudo ./ultrasonicranging
starting ...
The distance is : 13.35 cm
The distance is : 40.12 cm
The distance is : 19.01 cm
The distance is : 15.98 cm
The distance is : 15.10 cm
The distance is : 15.93 cm
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>
#include <sys/time.h>

#define trigPin 4
#define echoPin 5
#define MAX_DISTANCE 220          // define the maximum measured distance
#define timeOut MAX_DISTANCE*60 // calculate timeout according to the
maximum measured distance
//function pulseIn: obtain pulse time of a pin
int pulseIn(int pin, int level, int timeout);
float getSonar(){    // get the measurement results of ultrasonic module,with unit: cm
    long pingTime;
    float distance;
    digitalWrite(trigPin,HIGH); //trigPin send 10us high level
    delayMicroseconds(10);
    digitalWrite(trigPin,LOW);
    pingTime = pulseIn(echoPin,HIGH,timeOut);    //read plus time of echoPin
    distance = (float)pingTime * 340.0 / 2.0 / 10000.0; // the sound speed is
340m/s,and calculate distance
    return distance;
```

```
}
```

```
int main(){
    printf("starting ... \n");
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("failed !");
        return 1;
    }
    float distance = 0;
    pinMode(trigPin,OUTPUT);
    pinMode(echoPin,INPUT);
    while(1){
        distance = getSonar();
        printf("The distance is : %.2f cm\n",distance);
        delay(1000);
    }
    return 1;
}
```

```
int pulseIn(int pin, int level, int timeout)
{
    struct timeval tn, t0, t1;
    long micros;
    gettimeofday(&t0, NULL);
    micros = 0;
    while (digitalRead(pin) != level)
    {
        gettimeofday(&tn, NULL);
        if (tn.tv_sec > t0.tv_sec) micros = 1000000L; else micros = 0;
        micros += (tn.tv_usec - t0.tv_usec);
        if (micros > timeout) return 0;
    }
    gettimeofday(&t1, NULL);
    while (digitalRead(pin) == level)
    {
        gettimeofday(&tn, NULL);
        if (tn.tv_sec > t0.tv_sec) micros = 1000000L; else micros = 0;
        micros = micros + (tn.tv_usec - t0.tv_usec);
        if (micros > timeout) return 0;
    }
}
```

```
if (tn.tv_sec > t1.tv_sec) micros = 1000000L; else micros = 0;  
micros = micros + (tn.tv_usec - t1.tv_usec);  
return micros;  
}
```

Code Interpretation

Definieren Sie zunächst die Pin und den maximalen Messabstand.

```
#define trigPin 4  
#define echoPin 5  
#define MAX_DISTANCE 220
```

Wenn das Modul keinen hohen Pegel zurückgibt, kann das Modul nicht auf die Rückkehr des Signals warten. Daher müssen wir die dauerhafte Zeit über maximale Entfernung berechnen, nämlich "timOut". $timOut = 2 * MAX_DISTANCE / 100/340 * 1000000$. Der folgende konstante Teil entspricht ungefähr 58,8.

```
#define timeOut MAX_DISTANCE*60
```

Mit der Unterfunktion "getSonar ()" wird das Ultraschallmodul für eine Messung gestartet und der gemessene Abstand mit der Einheit cm zurückgegeben. Lassen Sie in dieser Funktion zuerst "trigPin" 10us hohen Pegel senden, um das Ultraschallmodul zu starten. Verwenden Sie dann "pulsIn ()", um das Ultraschallmodul zu lesen und die Dauer des hohen Pegels zurückzugeben. Berechnen Sie abschließend die gemessene Entfernung nach der Zeit.

```
float getSonar(){ // get the measurement results of ultrasonic module,with unit: cm  
    long pingTime;  
    float distance;  
    digitalWrite(trigPin,HIGH); //trigPin send 10us high level  
    delayMicroseconds(10);  
    digitalWrite(trigPin,LOW);  
    pingTime = pulseIn(echoPin,HIGH,timeOut); //read plus time of echoPin  
    distance = (float)pingTime * 340.0 / 2.0 / 10000.0; // the sound speed is  
    340m/s,and calculate distance  
    return distance;  
}
```

Ermitteln Sie schließlich in der "while" Schleife der Hauptfunktion den Messabstand und drucken Sie ihn ständig aus.

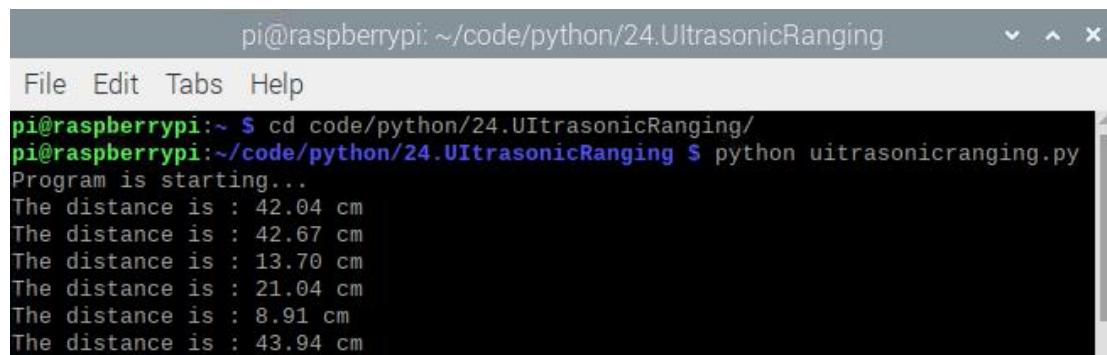
```
while(1){  
    distance = getSonar();  
    printf("The distance is : %.2f cm\n",distance);  
    delay(1000);  
}
```

Über die Funktion "pulsIn () " : Gibt die Länge des Impulses (in Mikrosekunden) oder 0 zurück, wenn vor dem Timeout kein Impuls abgeschlossen ist (vorzeichenlos lang).

```
int pulseIn(int pin, int level, int timeout);
```

Python code

1. Verwenden Sie den Befehl "[cd code / python / 24.UltrasonicRanging /](#)", um das Verzeichnis "UltrasonicRanging" aufzurufen;
2. Verwenden Sie den Befehl "[python ultrasonicranging.py](#)", um den Code "ultrasonicranging.py" auszuführen.



A terminal window titled "pi@raspberrypi: ~code/python/24.UltrasonicRanging". The window shows the command "python ultrasonicranging.py" being run and its output. The output displays the distance measured in centimeters for multiple pulses.

```
pi@raspberrypi: ~$ cd code/python/24.UltrasonicRanging/  
pi@raspberrypi:~/code/python/24.UltrasonicRanging $ python ultrasonicranging.py  
Program is starting...  
The distance is : 42.04 cm  
The distance is : 42.67 cm  
The distance is : 13.70 cm  
The distance is : 21.04 cm  
The distance is : 8.91 cm  
The distance is : 43.94 cm
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO  
import time  
trigPin = 16  
echoPin = 18  
MAX_DISTANCE = 220
```

```
timeOut = MAX_DISTANCE*60
def pulseIn(pin,level,timeOut):
    t0 = time.time()
    while(GPIO.input(pin) != level):
        if((time.time() - t0) > timeOut*0.000001):
            return 0;
    t0 = time.time()
    while(GPIO.input(pin) == level):
        if((time.time() - t0) > timeOut*0.000001):
            return 0;
    pulseTime = (time.time() - t0)*1000000
    return pulseTime
def getSonar():
    GPIO.output(trigPin,GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(trigPin,GPIO.LOW)
    pingTime = pulseIn(echoPin,GPIO.HIGH,timeOut)
    distance = pingTime * 340.0 / 2.0 / 10000.0
    return distance
def setup():
    print ('Program is starting...')
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(trigPin, GPIO.OUT)
    GPIO.setup(echoPin, GPIO.IN)
def loop():
    while(True):
        distance = getSonar()
        print ("The distance is : %.2f cm"%distance)
        time.sleep(1)
if __name__ == '__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        GPIO.cleanup()
```

Code Interpretation

```
import RPi.GPIO as GPIO
import time
trigPin = 16
echoPin = 18
MAX_DISTANCE = 220
```

Definieren Sie zunächst die Pin und den maximalen Messabstand.

```
timeOut = MAX_DISTANCE*60
```

Wenn das Modul keinen hohen Pegel zurückgibt, kann das Modul nicht auf die Rückkehr des Signals warten. Daher müssen wir die dauerhafte Zeit über maximale Entfernung berechnen, nämlich "timeOut(μ s)". "timeOut = 2 * MAX_DISTANCE / 100/340 * 1000000". Der folgende konstante Teil entspricht ungefähr 58,8.

```
def getSonar():
    GPIO.output(trigPin,GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(trigPin,GPIO.LOW)
    pingTime = pulseIn(echoPin,GPIO.HIGH,timeOut)
    distance = pingTime * 340.0 / 2.0 / 10000.0
    return distance
```

Mit der Unterfunktion "getSonar ()" wird das Ultraschallmodul für eine Messung gestartet und der gemessene Abstand mit der Einheit cm zurückgegeben. Lassen Sie in dieser Funktion zuerst "trigPin" 10us hohen Pegel senden, um das Ultraschallmodul zu starten. Verwenden Sie dann "pulsIn ()", um das Ultraschallmodul zu lesen und die Dauer des hohen Pegels zurückzugeben. Berechnen Sie abschließend die gemessene Entfernung nach der Zeit.

Lektion 25 Infrarot Sensor

Überblick

In dieser Lektion lernen Sie den Umgang mit Infrarot Bewegungssensor. Der Arbeitsmodus ist: Ausgabe eines hohen Pegels, wenn der menschliche Körper erfasst wird, und Ausgabe eines niedrigen Pegels, wenn der menschliche Körper nicht erfasst wird.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-Erweiterungskarte und Kabel

1 x Steckbrett

1 x 220Ω Widerstand

1 x LED

1 x HC SR501

Produkt Einführung

Infrared Sensor(HC SR501)

Der menschliche Körper hat eine konstante Körpertemperatur, normalerweise bei 37 Grad, so dass er Infrarotstrahlen mit einer spezifischen Wellenlänge von etwa 10UM aussendet. Der Infrarotsensor erfasst die vom menschlichen Körper emittierten Infrarotstrahlen. Etwa 10UM Infrarotlicht, das vom menschlichen Körper emittiert wird, wird durch einen Fresnel-Linse verstärkt und auf eine Infrarot-Erfassungsquelle fokussiert.

Infrarot-Induktionsquellen verwenden normalerweise pyroelektrische Komponenten. Solche Komponenten verlieren ihr Ladungsgleichgewicht, wenn sie Änderungen der Temperatur der Infrarotstrahlung des menschlichen Körpers erhalten und ihre

Ladungen nach außen abgeben und nachfolgende Schaltungen können nach Erkennung und Verarbeitung Alarmsignale erzeugen.

Modulparameter:

Betriebsspannung: DC 5V bis 20V

Statischer Stromverbrauch: 65 Mikroampere

Pegelausgang: 3,3 V hoch, 0 V niedrig

Verzögerungszeit: einstellbar (0,3 Sekunden bis 18 Sekunden)

Sperrzeit: 0,2 Sekunden

Triggermodus: L kann nicht wiederholt werden, H kann wiederholt werden, der Standardwert ist H (Auswahl der Sprungkappe)

Induktionsbereich: weniger als 120 Grad Kegelwinkel innerhalb von 7 Metern

Arbeitstemperatur: -15 ~ + 70 Grad

L: Nicht wiederholbarer Triggermodus. Nach dem Erfassen des menschlichen Körpers gibt das Modul einen hohen Pegel aus, und nach Ablauf der Verzögerungszeit gibt das Modul einen niedrigen Pegel aus. Während der Hochzeit erfasst der Sensor den menschlichen Körper nicht mehr.

H: Wiederholbarer Triggermodus. Der Unterschied zu L besteht darin, dass es den menschlichen Körper die ganze Zeit erfassen kann, das Modul nach dem Erfassen des menschlichen Körpers einen hohen Pegel ausgibt, bis der menschliche Körper verlässt, und dann das Timing startet und nach dem Verzögern der T-Zeit einen niedrigen Pegel ausgibt.

Induktionsblockierungszeit: Die Zeit nach der Induktion bleibt im Blockierungszustand, ohne dass das externe Signal einen hohen oder niedrigen Pegel ausgibt (weniger als die Verzögerungszeit).

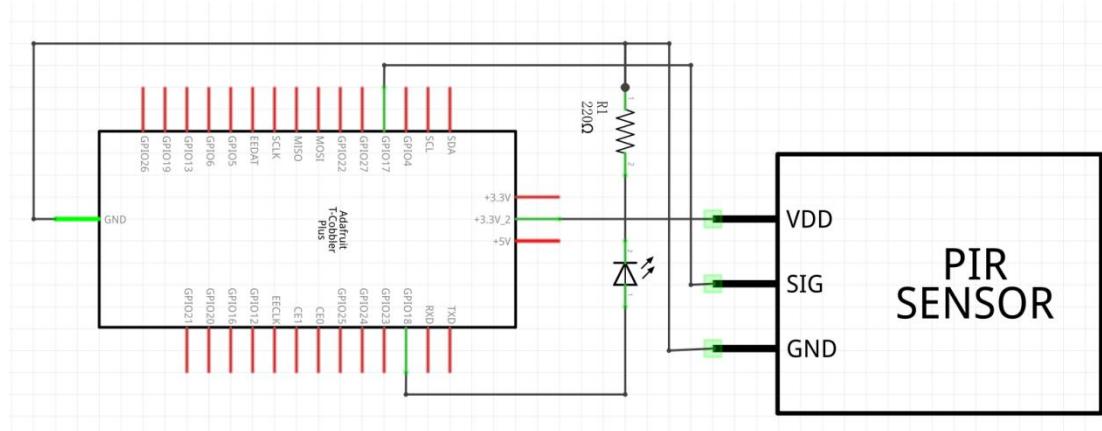
Initialisierungszeit: Es dauert ungefähr 1 Minute, bis das Modul nach dem Einschalten initialisiert ist. Während dieser Zeit wird abwechselnd hoch oder niedrig ausgegeben.

In Anbetracht der Eigenschaften dieses Sensors arbeitet der Sensor sehr empfindlich, wenn sich der menschliche Körper in der Nähe oder vom Rand entfernt befindet.

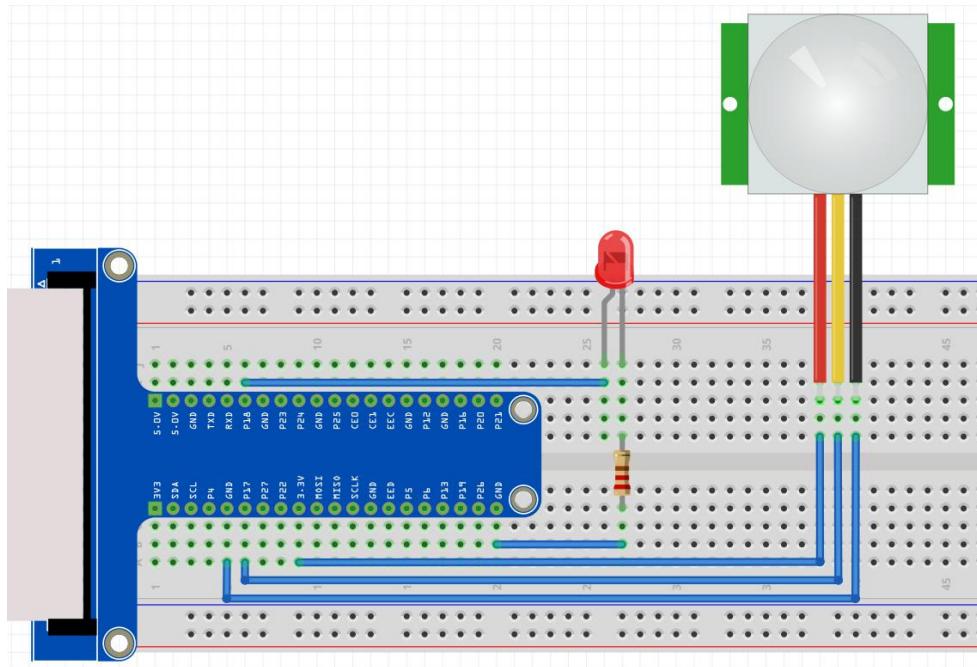
Wenn sich der menschliche Körper der vertikalen Richtung nähert oder von dieser weg bewegt, kann der Sensor nicht arbeiten. Wir können uns diesen Sensor als einfachen induktiven Schalter vorstellen, wenn er verwendet wird.



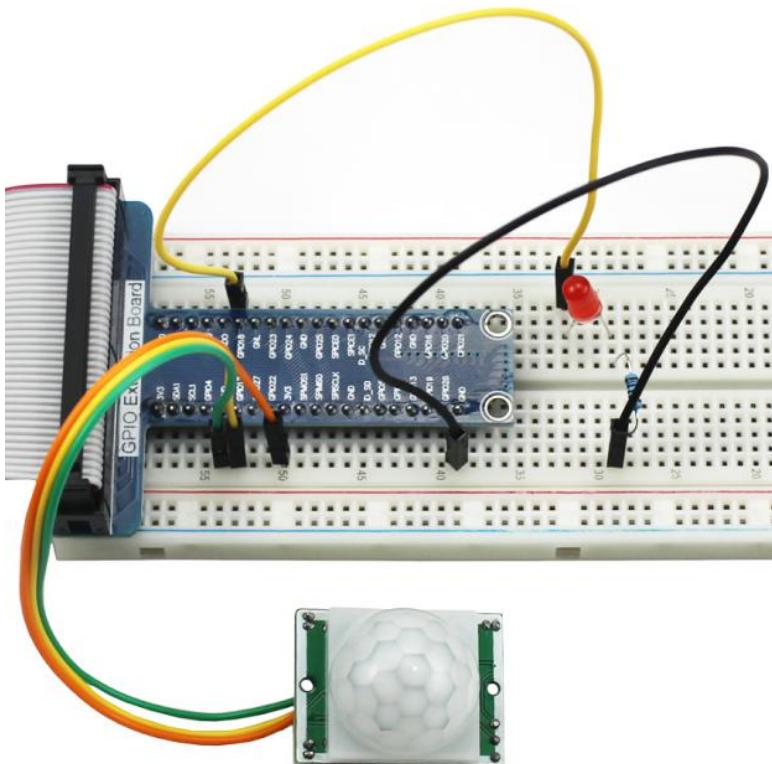
Schaltplan



Verdrahtung Diagramm



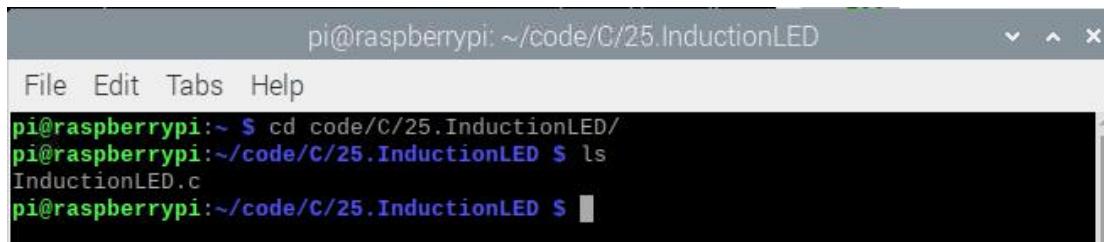
Beispiel Abbildung



C code

Öffnen Sie das Terminal und geben Sie den Befehl "[cd code / C / 25.InductionLED /](#)" ein, um das Codeverzeichnis "InductionLED" aufzurufen.

Geben Sie den Befehl "[ls](#)" ein, um die Datei "InductionLED.c" im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/C/25.InductionLED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/25.InductionLED/
pi@raspberrypi:~/code/C/25.InductionLED $ ls
InductionLED.c
pi@raspberrypi:~/code/C/25.InductionLED $
```

Geben Sie den Befehl "[gcc InductionLED.c -o InductionLED -lwiringPi](#)" ein, um die ausführbare Datei "InductionLED .c" "InductionLED" zu generieren, und geben Sie den Befehl "ls" ein, um sie anzuzeigen. Geben Sie "[sudo ./InductionLED](#)" command ein, um den Code auszuführen. Die Ergebnisse werden unten angezeigt:



```
pi@raspberrypi:~/code/C/25.InductionLED
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/25.InductionLED/
pi@raspberrypi:~/code/C/25.InductionLED $ ls
InductionLED.c
pi@raspberrypi:~/code/C/25.InductionLED $ gcc InductionLED.c -o InductionLED -lwiringPi
pi@raspberrypi:~/code/C/25.InductionLED $ ls
InductionLED InductionLED.c
pi@raspberrypi:~/code/C/25.InductionLED $ sudo ./InductionLED
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>

#define ledPin    1      //define the ledPin
#define sensorPin 0      //define the sensorPin
```

```
int main(void)
{
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("failed !");
        return 1;
    }

    pinMode(ledPin, OUTPUT);
    pinMode(sensorPin, INPUT);

    while(1){

        if(digitalRead(sensorPin) == HIGH){ //if read sensor for high level
            digitalWrite(ledPin, HIGH);    //led on
            printf("led on...\n");
        }
        else {
            digitalWrite(ledPin, LOW);    //led off
            printf("...led off\n");
        }
    }

    return 0;
}
```

Code Interpretation

Verwenden Sie in der Hauptfunktion zuerst "if (digitalRead (sensorPin) == HIGH)", um festzustellen, ob ein menschlicher Körper vorhanden ist, und kehren Sie auf ein hohes Niveau zurück, wenn ein menschlicher Körper erkannt wird. "digitalWrite (ledPin, HIGH)" wird verwendet, um das LED-Licht einzuschalten, wenn ein menschlicher Körper erkannt wird.

```
while(1){

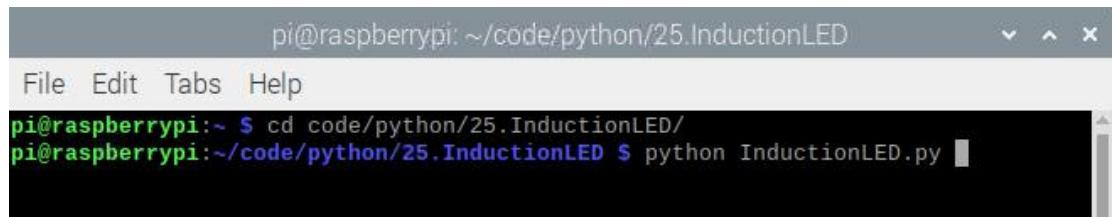
    if(digitalRead(sensorPin) == HIGH){ //if read sensor for high level
        digitalWrite(ledPin, HIGH);    //led on
        printf("led on...\n");
    }
}
```

```
    else {
        digitalWrite(ledPin, LOW); //led off
        printf("...led off\n");
    }
}
```

Der Code entspricht der LED-Lektion für die Tastensteuerung.

Python code

1. Verwenden Sie den Befehl "`cd code / python / 25.InductionLED /`", um das Verzeichnis des Infrarotsensors einzugeben.
2. Verwenden Sie den Befehl "`python InductionLED.py`", um den Code "InductionLED.py" auszuführen.



A screenshot of a terminal window titled "pi@raspberrypi: ~ /code/python/25.InductionLED". The window shows the command "cd code/python/25.InductionLED/" followed by "python InductionLED.py". The terminal is a standard Linux terminal with a dark background and light text.

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO

ledPin = 12      # define the ledPin
sensorPin = 11    # define the sensorPin

def setup():
    print ('starting...')
    GPIO.setmode(GPIO.BOARD)      # Numbers GPIOs by physical location
    GPIO.setup(ledPin, GPIO.OUT)    # Set ledPin's mode is output
    GPIO.setup(sensorPin, GPIO.IN)  # Set sensorPin's mode is input

def loop():
```

```
while True:  
    if GPIO.input(sensorPin)==GPIO.HIGH:  
        GPIO.output(ledPin,GPIO.HIGH)  
        print ('led on ...')  
    else :  
        GPIO.output(ledPin,GPIO.LOW)  
        print ('led off ...')  
  
def destroy():  
    GPIO.cleanup()                      # Release resource  
if __name__ == '__main__':      # Program start from here  
    setup()  
    try:  
        loop()  
    except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program  
destroy() will be executed.  
    destroy()
```

Code Interpretation

Verwenden Sie in der Hauptfunktion "loop ()" zuerst "if GPIO.input (sensorPin) == GPIO.HIGH", um festzustellen, ob ein menschlicher Körper vorhanden ist, und geben Sie einen hohen Wert zurück, wenn ein menschlicher Körper erkannt wird. Wenn ein menschlicher Körper erkannt wird, verwenden Sie die Funktion "GPIO.output (ledPin, GPIO.HIGH)", um das LED Licht einzuschalten.

```
def loop():  
    while True:  
        if GPIO.input(sensorPin)==GPIO.HIGH:  
            GPIO.output(ledPin,GPIO.HIGH)  
            print ('led on ...')  
        else :  
            GPIO.output(ledPin,GPIO.LOW)  
            print ('led off ...')
```

Der Code ist der gleiche wie der "Taste" Lektion Code.

Lektion26 MPU6050

Überblick

In dieser Lektion lernen Sie einen Lagerungssensor MPU6050 kennen, der einen Beschleunigungsmesser und ein Gyroskop integriert.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-Erweiterungskarte und Kabel

1 x Steckbrett

1 x MPU6050

Produkt Einführung

MPU6050

Die MPU6050 ist ein integrierter 3-Achsen Beschleunigungsmesser, ein 3-Achsen Winkelbeschleunigungsmesser (Gyroskop genannt) und ein digitaler Lagerungsprozessor (DMP). Im mpu6050 verwenden wir einen Gyroskopsensor zum Messen des Winkels und einen Beschleunigungssensor zum Messen der Beschleunigung.

Gyroskop: Das Prinzip eines Gyroskops besteht darin, dass sich die Richtung der Drehachse eines rotierenden Objekts nicht ändert, wenn es nicht durch äußere Kräfte beeinflusst wird. Verwenden Sie es aus diesem Grund, um die Richtung beizubehalten. Verwenden Sie dann verschiedene Methoden, um die von der Achse angegebene Richtung zu lesen und das Datensignal automatisch an das Steuerungssystem zu übertragen.

Beschleunigungssensor: Ein Beschleunigungssensor ist ein Sensor, der die Beschleunigung messen kann. Es besteht normalerweise aus Masse, Dämpfer, elastischem Element, empfindlichem Element und adaptiver Schaltung. Während der Beschleunigung verwendet der Sensor das zweite Newtonsche Gesetz, um den Beschleunigungswert durch Messen der Trägheitskraft auf die Masse zu erhalten. Abhängig von den empfindlichen Komponenten des Sensors umfassen übliche

Beschleunigungssensoren kapazitive, induktive, Dehnungs-, piezoresistive und piezoelektrische Sensoren.

6-Achsen Gyroskopsensor: Der DMP empfängt und verarbeitet Daten vom Gyroskop, Beschleunigungsmesser und externen Sensoren. Die Verarbeitungsergebnisse können aus DMP-Registern gelesen oder über FIFO gepuffert werden. Der DMP ist berechtigt, einen Interrupt über einen externen Pin der MPU zu generieren.

Pin-Name & Pin Nummer Beschreibung

VCC 1 ----- Positive Stromversorgung, 5V

GND 2 ----- Negative Stromversorgung

SCL3 I2C ----- Kommunikationstakt-Pin

SDA 4 I2C ----- Kommunikationsdaten-Pin

XDA 5 I2C ----- Host Daten Pin, kann mit anderen Geräten verbunden werden.

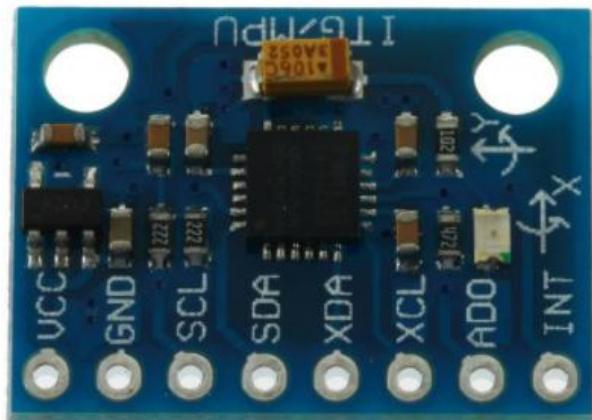
XCL 6 I2C ----- Master Clock Pin, kann an andere Geräte angeschlossen werden.

AD0 7 I2C ----- Adressbit-Steuerstift.

----- Niedrig: Geräteadresse ist 0x68

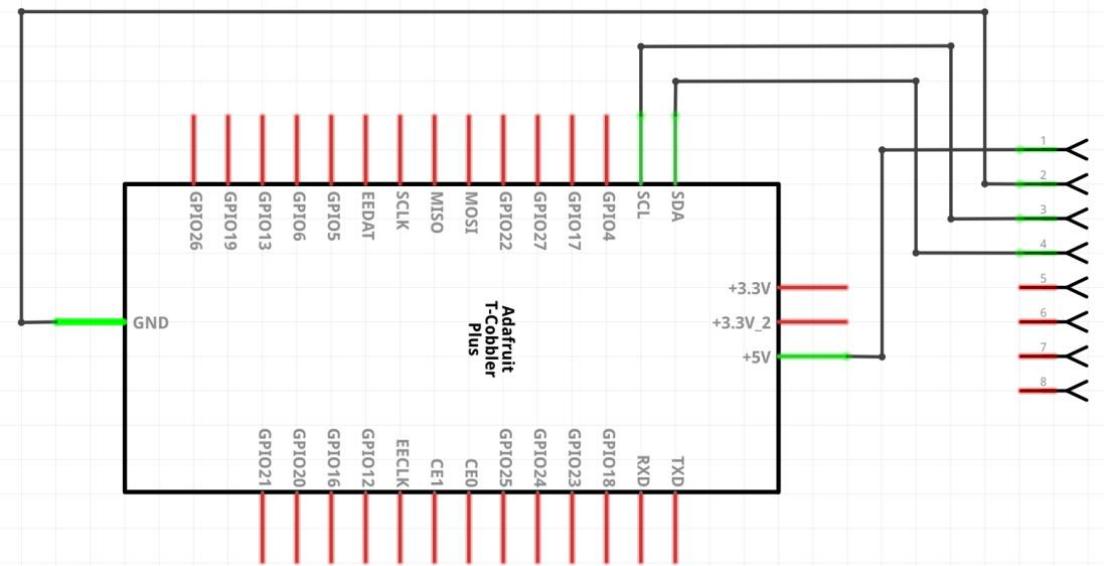
----- Hoch: Geräteadresse ist 0x69

INT 8 ----- Ausgangs Interrupt Pin

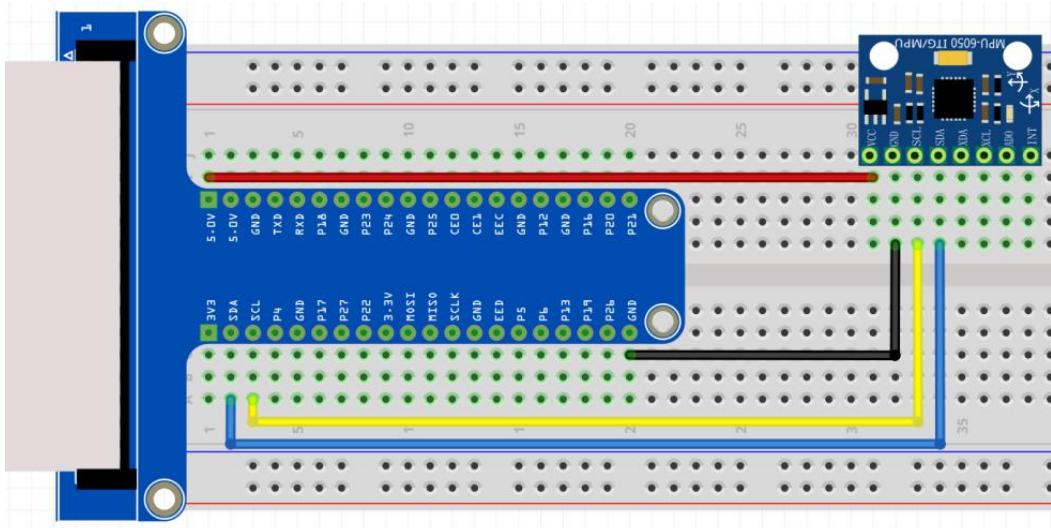




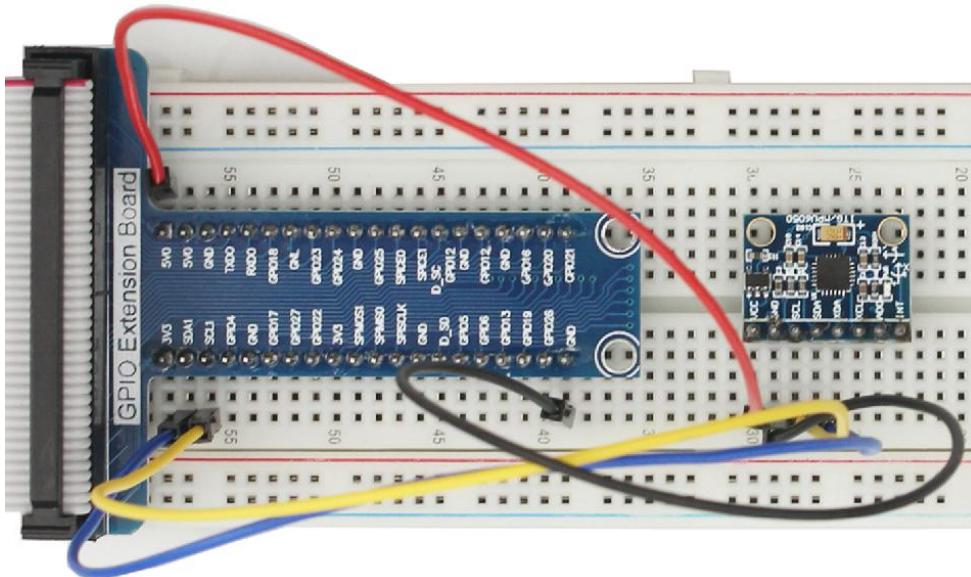
Schaltplan



Verdrahtung Diagramm



Beispiel Abbildung



C code

Öffnen Sie das Terminal und geben Sie den Befehl "[cd code / C / 26.MPU6050 /](#)" ein, um das Codeverzeichnis "MPU6050" aufzurufen.

Geben Sie den Befehl "[ls](#)" ein, um die Dateien "I2Cdev.cpp", "I2Cdev.h", "MPU6050.cpp", "MPU6050.h", "MPU6050RAW.cpp" im Verzeichnis anzuzeigen.



```
pi@raspberrypi:~/code/C/26.MPU6050
pi@raspberrypi:~/code/C/26.MPU6050$ ls
I2Cdev.cpp  I2Cdev.h  MPU6050.cpp  MPU6050.h  MPU6050RAW.cpp
pi@raspberrypi:~/code/C/26.MPU6050$
```

Geben Sie "[gcc I2Cdev.cpp MPU6050.cpp MPU6050RAW.cpp -o MPU6050RAW -lwiringPi](#)" ein, um die ausführbare Datei "MPU6050RAW" zu generieren, und geben Sie den Befehl "ls" ein, um sie anzuzeigen. Geben Sie "[sudo ./MPU6050RAW](#)" Befehl" ein, um den Code auszuführen.

Die Ergebnisse sind wie folgt:

```
pi@raspberrypi:~/code/C/26.MPU6050
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/26.MPU6050/
pi@raspberrypi:~/code/C/26.MPU6050$ ls
I2Cdev.cpp I2Cdev.h MPU6050.cpp MPU6050.h MPU6050RAW.cpp
pi@raspberrypi:~/code/C/26.MPU6050$ gcc I2Cdev.cpp MPU6050.cpp MPU6050RAW.cpp -o MPU6050RAW -lwiringPi
pi@raspberrypi:~/code/C/26.MPU6050$ ls
I2Cdev.cpp I2Cdev.h MPU6050.cpp MPU6050.h MPU6050RAW MPU6050RAW.cpp
pi@raspberrypi:~/code/C/26.MPU6050$ sudo ./MPU6050RAW
Initializing I2C devices...
Testing device connections...
MPU6050 connection failed
a/g:    128     108   19758      -700     -166     123
a/g:  0.01 g  0.01 g  1.21 g   -5.34 d/s  -1.27 d/s  0.94 d/s
a/g:    120      28   19208      -666     -125     130
a/g:  0.01 g  0.00 g  1.17 g   -5.08 d/s  -0.95 d/s  0.99 d/s
a/g:    120      66   18776      -674     -123     133
a/g:  0.01 g  0.00 g  1.15 g   -5.15 d/s  -0.94 d/s  1.02 d/s
a/g:    140      56   18584      -694     -159     131
a/g:  0.01 g  0.00 g  1.13 g   -5.30 d/s  -1.21 d/s  1.00 d/s
a/g:    180      20   19170      -722     -207     123
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <stdio.h>
#include <stdint.h>
#include <unistd.h>
#include "I2Cdev.h"
#include "MPU6050.h"

MPU6050 accelgyro;           //instantiate a MPU6050 class object

int16_t ax, ay, az;          //store acceleration data
int16_t gx, gy, gz;          //store gyroscope data

void setup() {
    // initialize device
    printf("Initializing I2C devices...\n");
    accelgyro.initialize();    //initialize MPU6050

    // verify connection
    printf("Testing device connections...\n");
```

```

printf(accelgyro.testConnection() ? "MPU6050 connection successful\n" :
"MPU6050 connection failed\n");
}

void loop() {
    // read raw accel/gyro measurements from device
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
    // display accel/gyro x/y/z values
    printf("a/g: %6hd %6hd %6hd    %6hd %6hd %6hd\n",ax,ay,az,gx,gy,gz);
    printf("a/g: %.2f g %.2f g %.2f g    %.2f d/s %.2f d/s %.2f d/s
\n", (float)ax/16384,(float)ay/16384,(float)az/16384,
        (float)gx/131,(float)gy/131,(float)gz/131);
}

int main()
{
    setup();
    while(1){
        loop();
    }
    return 0;
}

```

Code Interpretation

Im Code werden zwei Bibliotheksdateien "MPU6050.h" und "I2Cdev.h" verwendet. Sie werden wie andere kompiliert. Die MPU6050 Klasse wird zum Bedienen der MPU6050 verwendet. Wenn es verwendet wird, wird zuerst ein Objekt instanziert.

MPU6050 accelgyro;

In der Funktion "setup ()" wird die MPU6050 initialisiert und das Ergebnis beurteilt.

```

void setup() {
    // initialize device
    printf("Initializing I2C devices...\n");
    accelgyro.initialize();      //initialize MPU6050

    // verify connection
    printf("Testing device connections...\n");

```

```
    printf(accelgyro.testConnection() ? "MPU6050 connection successful\n" :  
          "MPU6050 connection failed\n");  
}
```

Lesen Sie in der Funktion "loop ()" die Rohdaten der MPU6050 und drucken Sie sie aus, konvertieren Sie die Rohdaten in die entsprechende Beschleunigung und Winkelgeschwindigkeit und drucken Sie die konvertierten Daten aus.

```
void loop() {  
    // read raw accel/gyro measurements from device  
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);  
    // display accel/gyro x/y/z values  
    printf("a/g: %6hd %6hd %6hd %6hd %6hd %6hd\n", ax, ay, az, gx, gy, gz);  
    printf("a/g: %.2f g %.2f g %.2f g %.2f d/s %.2f d/s %.2f d/s  
          \n", (float)ax/16384, (float)ay/16384, (float)az/16384,  
          (float)gx/131, (float)gy/131, (float)gz/131);  
}
```

Rufen Sie in der Hauptfunktion das Funktionssetup "setup ()" und die gelesenen Daten auf und drucken Sie dann die Funktion "loop ()".

```
int main()  
{  
    setup();  
    while(1){  
        loop();  
    }  
    return 0;  
}
```

Python code

1. Verwenden Sie den Befehl "[cd code / python / 26.MPU6050 /](#)", um das Verzeichnis der MPU6050 einzugeben.
2. Verwenden Sie den Befehl "[python MPU6050.py](#)", um den Code "MPU6050.py" auszuführen.

Nach Ausführung des Programms zeigt das Terminal die von der MPU6050 gelesenen Rohdaten sowie die transformierte Beschleunigung und Winkelgeschwindigkeit an.

Wie in der folgenden Abbildung gezeigt:

```
pi@raspberrypi: ~/code/python/26.MPU6050
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/26.MPU6050/
pi@raspberrypi:~/code/python/26.MPU6050 $ python MPU6050.py
tarting ...
a/g:184 34      19196   -86    -21     16
a/g:0.01 g     0.00 g   1.17 g  -0.66 d/s    -0.16 d/s     0.12 d/s
a/g:220 44      19172   -85    -20     15
a/g:0.01 g     0.00 g   1.17 g  -0.65 d/s    -0.15 d/s     0.11 d/s
a/g:188 50      19212   -86    -22     15
a/g:0.01 g     0.00 g   1.17 g  -0.66 d/s    -0.17 d/s     0.11 d/s
a/g:178 48      19174   -87    -22     15
a/g:0.01 g     0.00 g   1.17 g  -0.66 d/s    -0.17 d/s     0.11 d/s
a/g:188 64      19172   -86    -21     15
a/g:0.01 g     0.00 g   1.17 g  -0.66 d/s    -0.16 d/s     0.11 d/s
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import MPU6050RAW
import time
mpu = MPU6050RAW.MPU6050()          #instantiate a MPU6050 class object
accel = [0]*3                         #store accelerometer data
gyro = [0]*3                           #store gyroscope data
def setup():
    mpu.dmp_initialize()               #initialize MPU6050

def loop():
    while(True):
        accel = mpu.get_acceleration()    #get accelerometer data
        gyro = mpu.get_rotation()         #get gyroscope data
        print("a/g:%d\%d\%d\%d\%d\%d\n"
              "%(accel[0],accel[1],accel[2],gyro[0],gyro[1],gyro[2]))")
        print("a/g:%.2f g\%.2f g\%.2f g\%.2f g\%.2f d/s\%.2f d/s\%.2f\n"
              "d/s%(accel[0]/16384.0,accel[1]/16384.0,
              accel[2]/16384.0,gyro[0]/131.0,gyro[1]/131.0,gyro[2]/131.0))")
        time.sleep(0.1)

if __name__ == '__main__':      # Program start from here
    print("tarting ... ")
    setup()
    try:
```

```
loop()
except KeyboardInterrupt: # When 'Ctrl+C' is pressed, the program will exit.
    pass
```

Code Interpretation

Im Code wird das Modul "MPU6050RAW.py" verwendet. Dieses Modul enthält Klassen für den Betrieb der MPU6050.

'MPU6050RAW.py': Dies ist eine Bibliothek zum Bedienen der MPU6050. Sie können die MPU6050 direkt lesen und einstellen.

Konstruktor 'def dmp_initialize (self)': Dies ist eine Initialisierungsfunktion, mit der die MPU6050 aktiviert wird. Der Bereich des Beschleunigungsmessers beträgt ± 2 g und der Bereich des Gyroskops beträgt ± 250 Grad / Sekunde.

'def get_acceleration (self): & def get_rotation (self)': Erfasst die Rohdaten von Beschleunigungsmesser und Gyroskop.

Instanziieren Sie ein Objekt am Anfang des Codes:

```
mpu = MPU6050RAW.MPU6050()
```

Initialisieren Sie in der Setup-Funktion 'setup ()' die MPU6050.

```
def setup():
    mpu.dmp_initialize()
```

Lesen Sie in der Hauptfunktion 'loop ()' zuerst die Rohdaten der MPU6050 und drucken Sie sie aus, konvertieren Sie dann die Rohdaten in die entsprechende Beschleunigungs und Winkelgeschwindigkeit und drucken Sie schließlich die konvertierten Beschleunigungs und Winkelgeschwindigkeitsdaten aus.

```
def loop():
    while(True):
        accel = mpu.get_acceleration()      #get accelerometer data
        gyro = mpu.get_rotation()          #get gyroscope data
        print("a/g:%d\nt%d\nt%d\nt%d\nt%d\nt%d"
              "%(accel[0],accel[1],accel[2],gyro[0],gyro[1],gyro[2]))"
        print("a/g:%.2f g\nt%.2f g\nt%.2f g\nt%.2f g\nt%.2f d/s\nt%.2f d/s\nt%.2f
              d/s"%(accel[0]/16384.0,accel[1]/16384.0,accel[2]/16384.0,gyro[0]/131.0,gyro[1]/131.
              0,gyro[2]/131.0))
```

```
time.sleep(0.1)
```

Lektion 27 Sound Sensor

Überblick

In dieser Lektion lernen wir Schallsensoren kennen und steuern LED-Leuchten durch Ton.

Erforderliche Teile

1 x Raspberry Pi 4

1 x GPIO T-Erweiterungskarte und Kabel

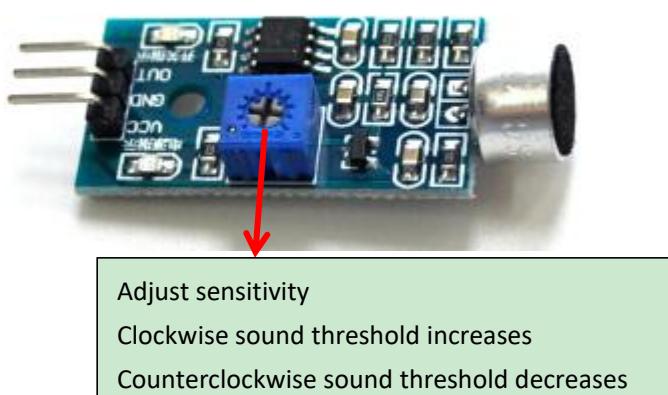
1 x Steckbrett

1 x Schallsensor

Produkt Einführung

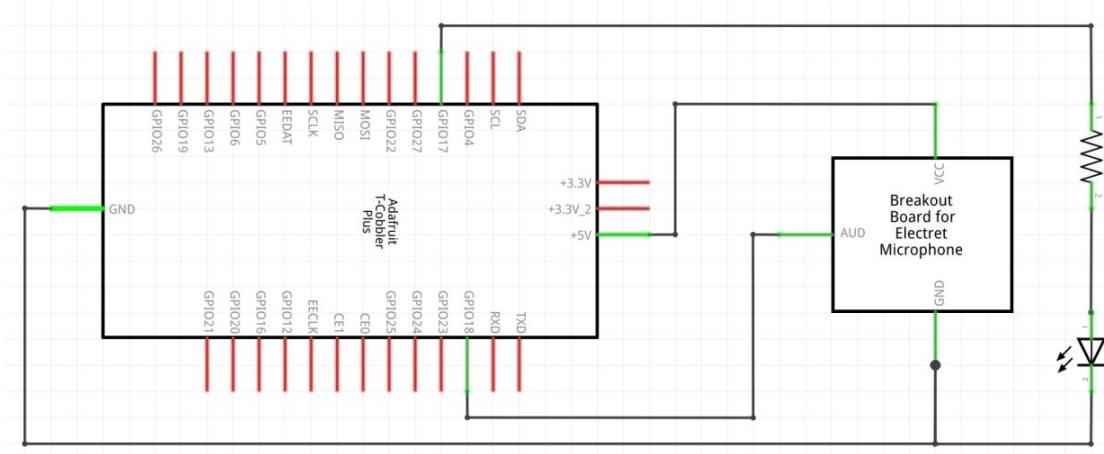
Sound Sensor

Der Schallsensor wird auch als Schallschalter bezeichnet. Wenn die Umgebung Schallintensität den eingestellten Schwellenwert nicht erreicht, gibt das Modul einen hohen Pegel aus. Wenn die Intensität des externen Umgebung Schallintensität den eingestellten Schwellenwert überschreitet, gibt das Modul einen niedrigen Pegel aus.

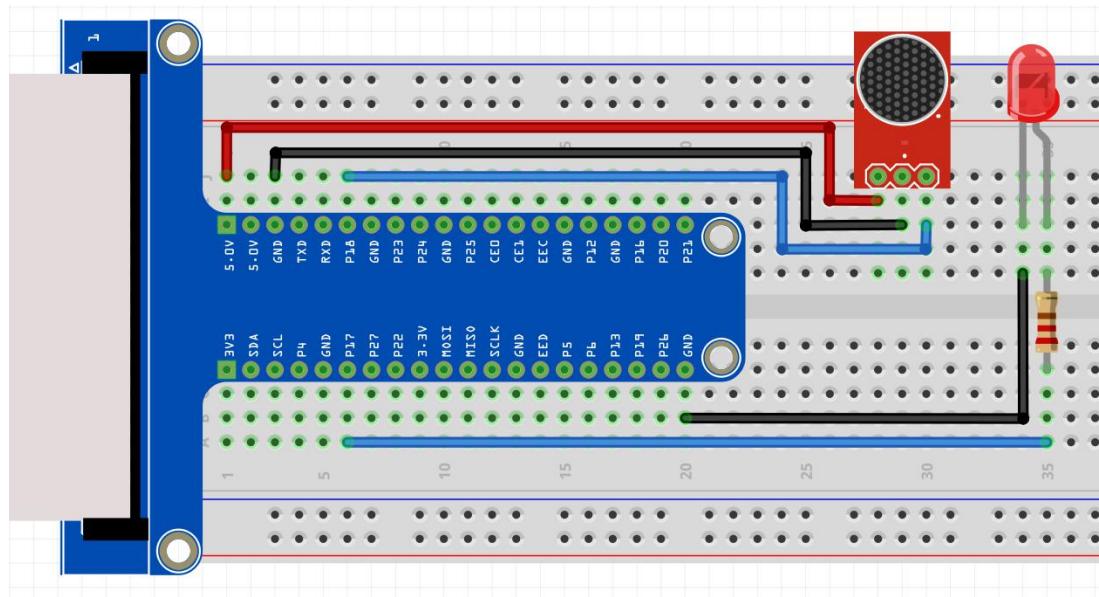




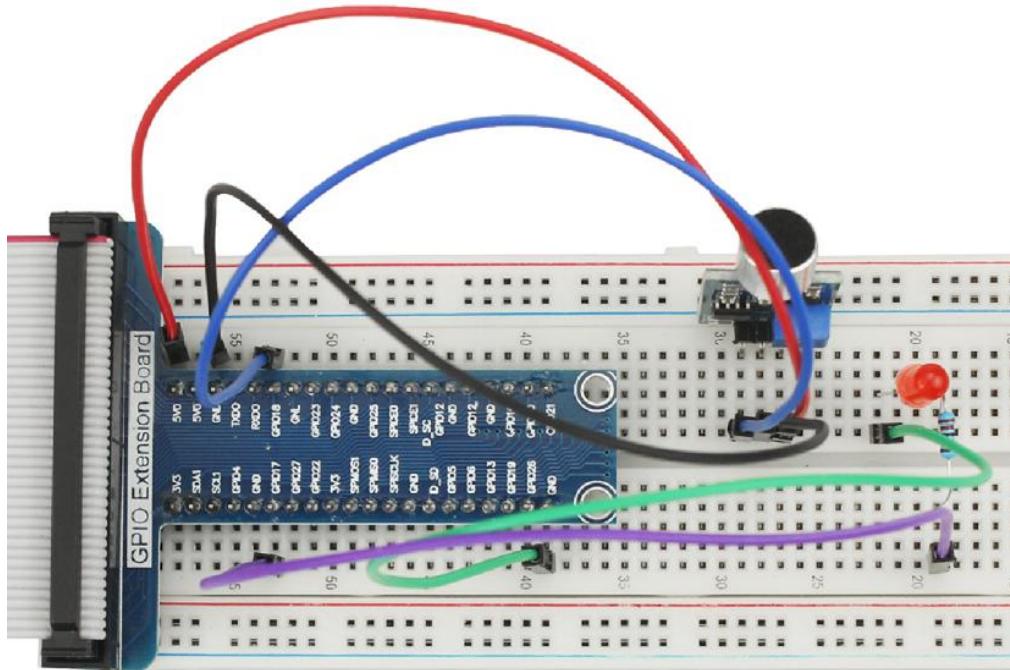
Schaltplan



Verdrahtung Diagramm



Beispiel Abbildung



C code

Öffnen Sie das Terminal und geben Sie den Befehl "cd code / C / 27.Sound /" ein, um das Codeverzeichnis "Sound" aufzurufen.

Geben Sie den Befehl "ls" ein, um die Datei "Sound.c" im Verzeichnis anzuzeigen.

```
pi@raspberrypi:~/code/C/27.Sound
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/27.Sound/
pi@raspberrypi:~/code/C/27.Sound $ ls
Sound.c
pi@raspberrypi:~/code/C/27.Sound $
```

Geben Sie den Befehl "`gcc Sound.c -o Sound -lwiringP`" ein, um die ausführbare Datei "Sound" von "Sound .c" zu generieren. Geben Sie den Befehl "`ls`" ein, um sie anzuzeigen. Geben Sie "`sudo ./Sound`" ein, um den Code auszuführen.

Das Ergebnis ist im Folgenden dargestellt:

```
pi@raspberrypi: ~/code/C/27.Sound
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/27.Sound/
pi@raspberrypi:~/code/C/27.Sound $ ls
Sound.c
pi@raspberrypi:~/code/C/27.Sound $ gcc Sound.c -o Sound -lwiringPi
pi@raspberrypi:~/code/C/27.Sound $ ls
Sound Sound.c
pi@raspberrypi:~/code/C/27.Sound $ sudo ./Sound
shock:
LED...LOW
shock:
LED...LOW
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>

#define ledPin      0      //define the ledPin
#define soundPin 1      //define the sensorPin

int main(void)
{
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("failed !");
        return 1;
    }

    pinMode(ledPin, OUTPUT);
    pinMode(soundPin, INPUT);

    while(1){
        printf("shock:\n");
        if(digitalRead(soundPin) == HIGH){ //if read sensor for high level
            digitalWrite(ledPin, HIGH);    //led on
            printf("LED...HIGH\n");
        }
        else {
```

```
    digitalWrite(ledPin, LOW); //led off
    printf("LED...LOW\n");
}
delay(500);
}

return 0;
}
```

Code Interpretation

Verwenden Sie in der Hauptfunktion die Anweisung "if (digitalRead (soundPin) == HIGH)", um festzustellen, ob Ton vorhanden ist, und geben Sie einen hohen Pegel zurück, wenn Ton vorhanden ist. Schalten Sie nach dem Erkennen des Tons die LED mit "digitalWrite (ledPin, HIGH)" ein.

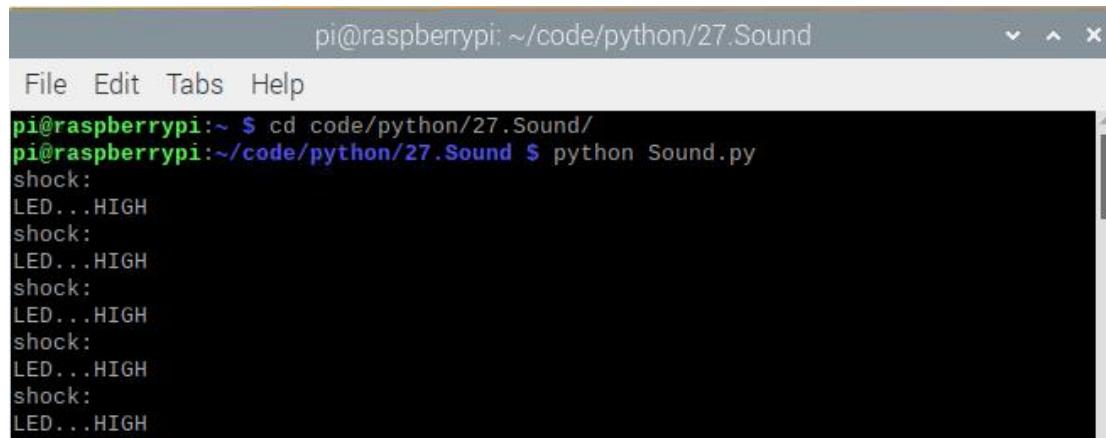
```
while(1){
    printf("shock:\n");
    if(digitalRead(soundPin) == HIGH){ //if read sensor for high level
        digitalWrite(ledPin, HIGH); //led on
        printf("LED...HIGH\n");
    }
    else {
        digitalWrite(ledPin, LOW); //led off
        printf("LED...LOW\n");
    }
    delay(500);
}
```

Der Code ist im Grunde der gleiche wie der Lektion 2 Taste & LED.

Python code

1. Verwenden Sie den Befehl "`cd code / python / 27.Sound /`", um das Verzeichnis des Soundsensors einzugeben.

2. Verwenden Sie den Befehl "[python Sound.py](#)", um den Code "Sound.py" auszuführen.



The screenshot shows a terminal window titled "pi@raspberrypi: ~/code/python/27.Sound". The window contains a menu bar with "File", "Edit", "Tabs", and "Help". Below the menu is a command-line interface. The user has navigated to the directory "/code/python/27.Sound/" and run the command "python Sound.py". The output of the program is displayed, showing repeated messages: "shock:" followed by "LED...HIGH".

```
pi@raspberrypi:~ $ cd code/python/27.Sound/
pi@raspberrypi:~/code/python/27.Sound $ python Sound.py
shock:
LED...HIGH
shock:
LED...HIGH
shock:
LED...HIGH
shock:
LED...HIGH
shock:
LED...HIGH
shock:
LED...HIGH
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import time
D0=12
LED=11
def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(D0,GPIO.IN)
    GPIO.setup(LED,GPIO.OUT)
    GPIO.output(LED,GPIO.LOW)

def loop():
    while True:
        print('shock!')
        PIN=GPIO.input(D0)
        if PIN == GPIO.HIGH :
            GPIO.output(LED,GPIO.HIGH)
            print("LED...HIGH")
        else  :
            GPIO.output(LED,GPIO.LOW)
            print("LED...LOW")
        time.sleep(0.5)
```

```
def destroy():
    GPIO.cleanup()

if __name__=='__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

Code Interpretation

Definieren Sie zunächst in der Funktion 'setup ()' den Schallsensor als Eingangsmodus und die LED als Ausgangsmodus.

```
def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(D0,GPIO.IN)
    GPIO.setup(LED,GPIO.OUT)
    GPIO.output(LED,GPIO.LOW)
```

In der 'while' Schleife der Hauptfunktion 'loop ()' wird der vom Schallsensor zurückgegebene Wert kontinuierlich erhalten. Wenn der Schallsensor einen hohen Pegel zurückgibt, wird das LED Licht eingeschaltet, andernfalls wird das LED Licht ausgeschaltet.

```
def loop():
    while True:
        print('shock:')
        PIN=GPIO.input(D0)
        if PIN == GPIO.HIGH :
            GPIO.output(LED,GPIO.HIGH)
            print("LED...HIGH")
        else  :
            GPIO.output(LED,GPIO.LOW)
            print("LED...LOW")
        time.sleep(0.5)
```

Lektion 28 RFID RC522

Überblick

In dieser Lektion lernen Sie, wie Sie RFID verwenden und wie Sie den RC522 RFID Kartenleser zum Lesen und Schreiben der M1-S50 Karte verwenden.

Erforderliche Teile

1 x Raspberry pi

1 x GPIO -Erweiterungskarte und Kabel

1 x Steckbrett

7 x Jumper M/F

1 x RC522 Modul

1 x Mifare1 S50 Nicht standardmäßige Karte

1 x Mifare1 S50 Standardkarte

Produkt Einführung

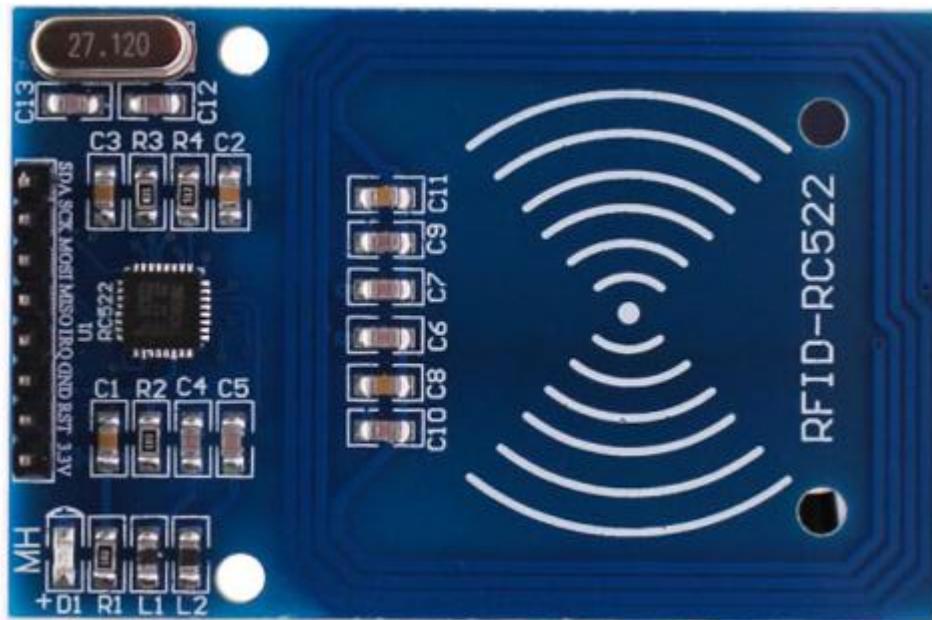
MFRC522

Der MFRC522 ist ein hochintegrierter Lese- / Schreib-IC für die kontaktlose Kommunikation bei 13,56 MHz.

Der interne Sender des MFRC522 kann eine Lese- / Schreibantenne ansteuern, die für die Kommunikation mit ISO / IEC 14443 A / MIFARE-Karten und -Transpondern ohne zusätzliche aktive Schaltkreise ausgelegt ist. Das Empfängermodul bietet eine robuste und effiziente Implementierung zum Demodulieren und Decodieren von Signalen von ISO / IEC 14443A / MIFARE-kompatiblen Karten und Transpondern. Das digitale Modul verwaltet die vollständige ISO / IEC 14443Aframing- und Fehlererkennungsfunktionalität (Parität und CRC). Dieses RFID-Modul verwendet MFRC522 als Steuerchip und SPI (Peripheral Interface Serial) als reservierte Schnittstelle.

Technische Daten:

Operating Voltage	13–26mA (DC) \3. 3V
Idle current	10–13mA (DC) \3. 3V
Sleep current in the	<80uA
Peak current	<30mA
Operating frequency	13. 56MHz
Supported card type	Mifare1 S50、Mifare1 S70、Mifare Ultralight、Mifare Pro、Mifare Desfire
Size	40mmX60mm
Operation temperature	20–80 degrees(Celsius)
Storage temperature	40–85 degrees (Celsius)
Operation humidity	5%–95% (Relative humidity)



Mifare1 S50 Card

Mifare S50 wird häufig als Mifare Standard mit einer Kapazität von 1 KByte bezeichnet. Und jede Karte hat eine 4-Byte globale eindeutige Identifikationsnummer (USN / UID), die 100.000 Mal umgeschrieben und unendlich oft gelesen werden kann. Die Lagerdauer beträgt 10 Jahre. Die normale Mifare1 S50-Karte und die nicht standardmäßige Mifare1 S50-Karte für das RFID Kit sind nachstehend aufgeführt. Die Kapazität des Mifare S50 (1 KByte) ist in 16 Sektoren unterteilt (Sektor 0 - Sektor 15). Jeder Sektor enthält 4 datablock (Block0-Block3. 64 Blöcke mit 16 Sektoren werden entsprechend der absoluten Adresse von 0 bis 63 nummeriert). Und

jeder Block enthält 16 Bytes (Byte0-Byte15), $64 * 16 = 1024$. Wie in der folgenden Tabelle gezeigt:

Sector No.	Block No.	Storage area	Block type	Absolute block No.
sector 0	block 0	vendor code	vendor block	0
	block 1		data block	1
	block 2		data block	2
	block 3	Password A-access control-password B	control block	3
sector 1	block 0		data block	4
	block 1		data block	5
	block 2		data block	6
	block 3	Password A-access control-password B	control block	7
.....
sector 15	block 0		data block	60
	block 1		data block	61
	block 2		data block	62
	block 3	Password A-access control-password B	control block	63

Jeder Sektor verfügt über eine Reihe unabhängiger Kennwort- und Zugriffskontrollen, die im letzten Block jedes Sektors abgelegt werden. Der Block wird auch als Sektoranhänger bezeichnet, dh Block 3 in jedem Sektor. In Sektor 0, Block 0 (nämlich absolute Adresse 0) von S50 wird der Herstellercode gespeichert, der verfestigt wurde und nicht geändert werden kann, und die Seriennummer der Karte wird hier gespeichert. Neben dem Hersteller und dem Steuerblock sind die restlichen Karten Datenblöcke, in denen Daten gespeichert werden können. Der Datenblock kann für zwei Arten von Anwendungen verwendet werden:

- (1) dient als allgemeine Datenspeicherung und kann zum Lesen und Schreiben betrieben werden.
- (2) wird als Datenwert verwendet und kann zum Initialisieren des Werts, Hinzufügen von Werten, Subtrahieren und Lesen des Werts verwendet werden.

Der Sektoranhängerblock in jedem Sektor ist der Steuerblock, einschließlich eines 6-Byte-Passworts A, einer 4-Byte-Zugriffssteuerung und eines 6-Byte-Passworts B. Der Steuerblock einer brandneuen Karte lautet beispielsweise wie folgt:

A0 A1 A2 A3 A4 A5	FF 07 80 69	B0 B1 B2 B3 B4 B5
password A	access control	password B

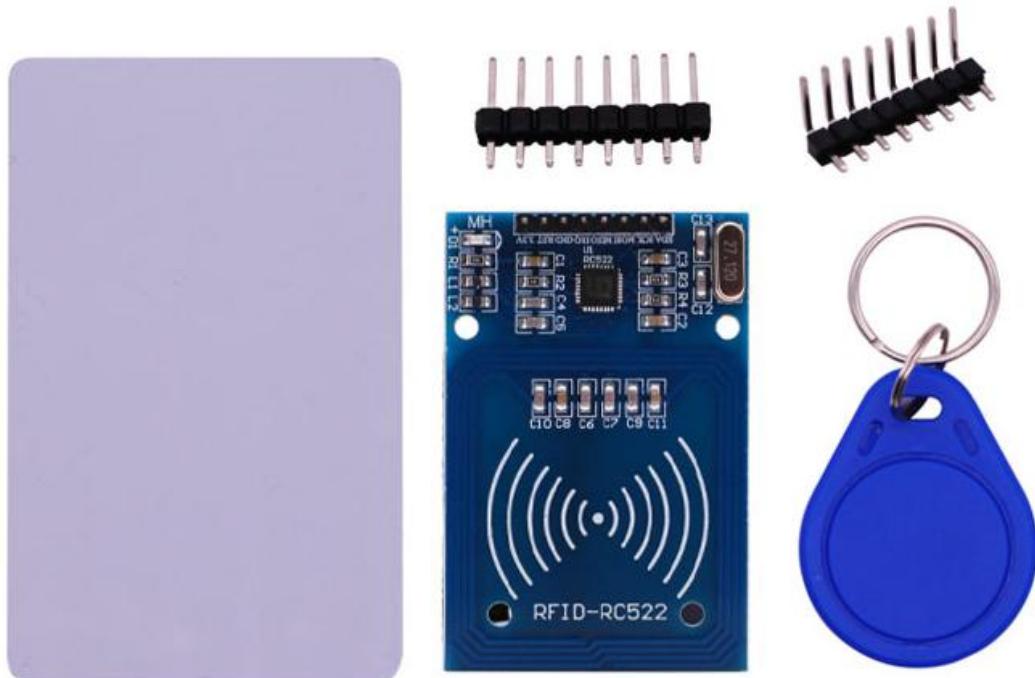
Das Standardkennwort einer brandneuen Karte lautet im Allgemeinen 0A1A2A3A4A5 für Kennwort A, B0B1B2B3B4B5 für Kennwort B oder sowohl Kennwort A als auch Kennwort B sind 6 FF. Die Zugriffssteuerung wird verwendet, um die Zugriffsbedingungen für jeden Block (einschließlich des Steuerblocks selbst) in einem Sektor festzulegen.

Blöcke von S50 sind in Datenblöcke und Steuerblöcke unterteilt. Es gibt vier Operationen "Lesen", "Schreiben", "Wert hinzufügen", "Wert subtrahieren (einschließlich Übertragung und Speicherung)" für Datenblöcke, und es gibt zwei Operationen ". Lesen und Schreiben für Steuerblöcke.

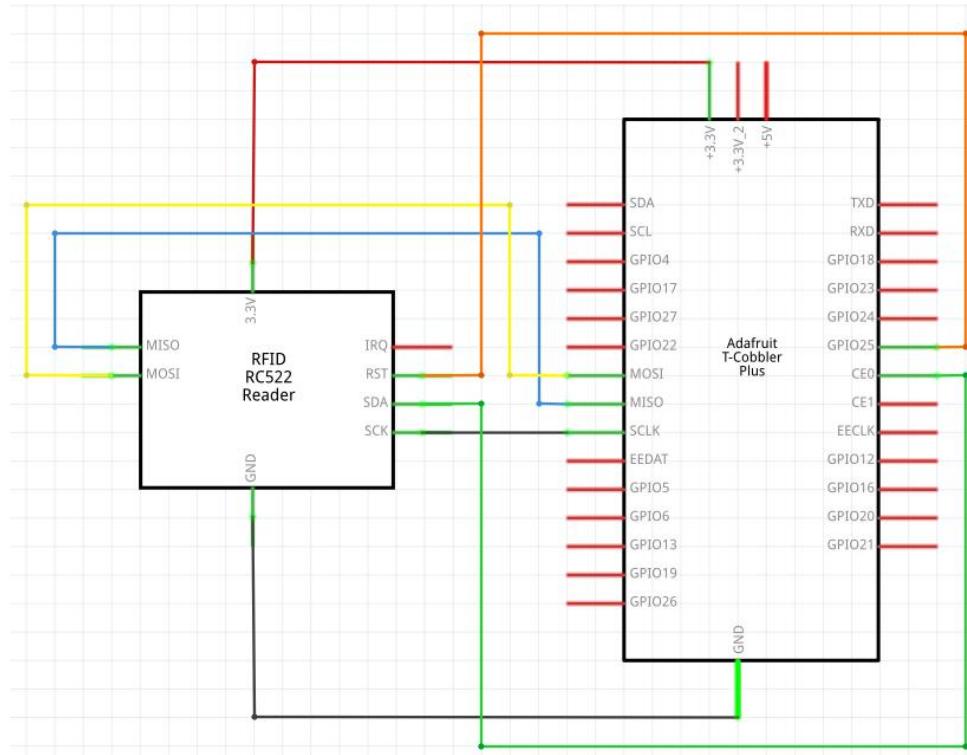
Weitere Informationen zum Festlegen von Datenblöcken und Steuerblöcken finden Sie im Datenblatt.

Standardmäßig können wir nach Überprüfung von Kennwort A oder Kennwort B Lese- oder Schreibvorgänge für Datenblöcke ausführen. Nach Überprüfung des Kennworts A können wir Lese- oder Schreibvorgänge ausführen, um Blöcke zu steuern. Aber Passwort A kann niemals gelesen werden. Wenn Sie Kennwort A überprüfen und dann Kennwort A vergessen, kann der Block nie wieder lesen.
Anfängern wird dringend empfohlen, den Inhalt von Steuerblöcken nicht zu ändern.

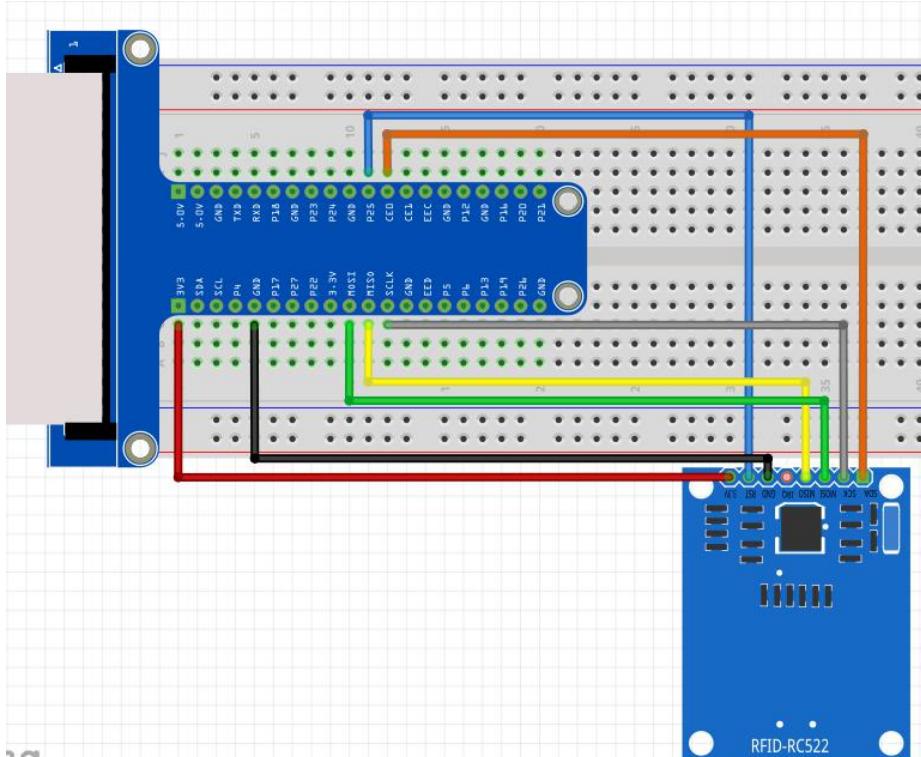
Für Mifare1 S50-Karten mit RFID Kits lauten die Standardkennwörter A und B FFFFFFFFFFFF.



Schaltplan

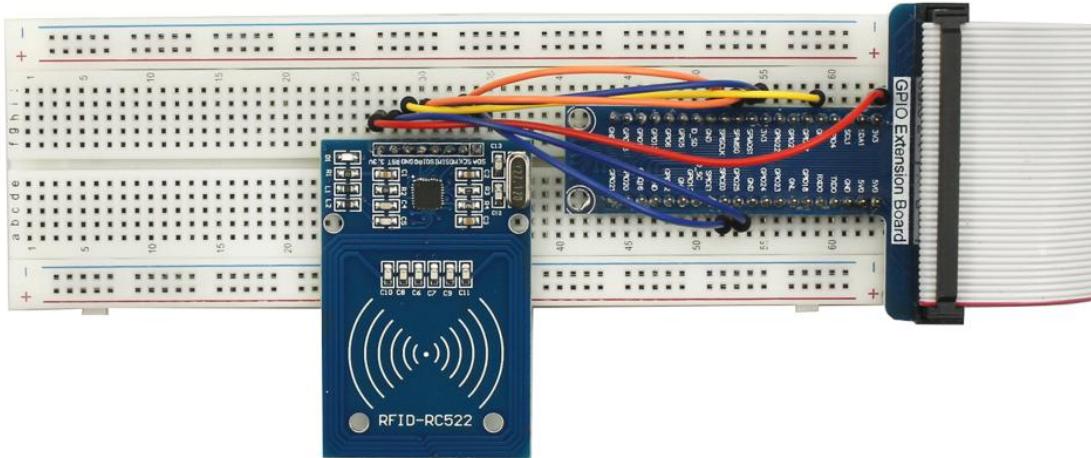


Verdrahtung Diagramm



Beispiel

Abbildung

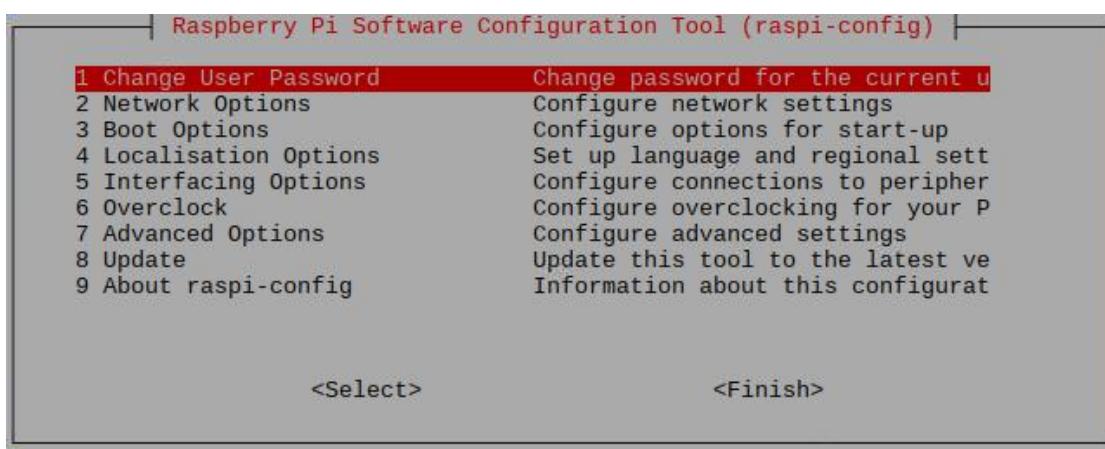


Configure SPI

Aktivieren Sie SPI

Der Himbeerkuchen der SPI-Schnittstelle ist standardmäßig geschlossen. Sie müssen es manuell öffnen. Sie können das SPI-Interface folgendermaßen aktivieren. Geben Sie den Befehl in das Terminal ein: sudo raspi-config

Öffnen Sie dann das folgende Dialogfeld:



Wählen Sie “5 Interfacing Options” “P4 SPI” “Yes” “Finish”, um Ihr RPi später neu zu starten. Dann wird das SPI Modul gestartet.

Installieren Sie das SPI-Py, PY-spidev-Modul

Geben Sie sudo apt-get install python-spidev python3-spidev im Terminal ein, um das Modul "Py-spidev" zu installieren.

Geben Sie den folgenden Befehl ein, um das SPI-Py Modul zu installieren:

Git-Klon <https://github.com/WayinTop/SPI-Py>

CD SPI-Py

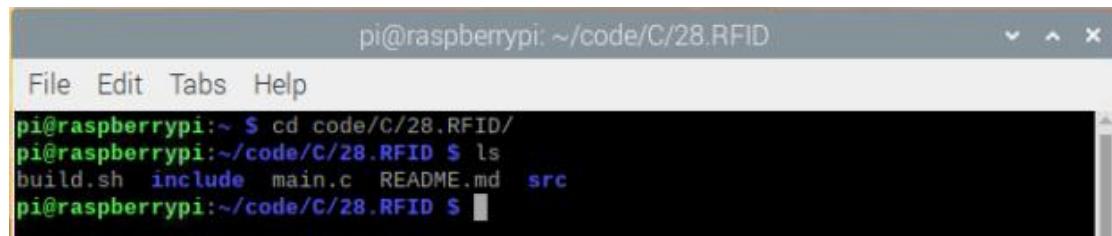
sudo python setup.py installieren

C Code

Der Projektcode verwendet den Befehlszeilenmodus für die Mensch-Computer-Interaktion zum Lesen und Schreiben der M1-S50-Karte.

Beobachten Sie zuerst das laufende Ergebnis und analysieren Sie dann den Code.

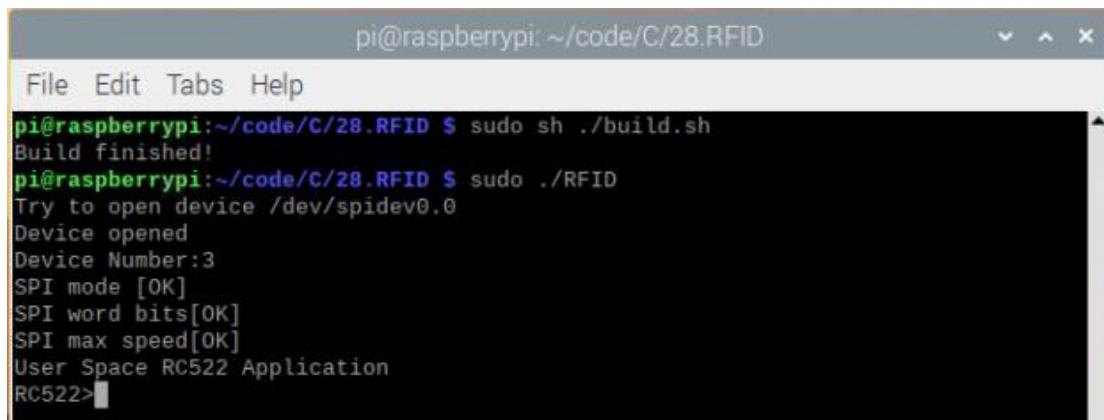
Geben Sie den Befehl “[cd code / C / 28.RFID /](#)” ein, um wie folgt in das RFID-Codeverzeichnis zu wechseln:



```
pi@raspberrypi: ~ cd code/C/28.RFID/
pi@raspberrypi: ~/code/C/28.RFID $ ls
build.sh  include  main.c  README.md  src
pi@raspberrypi: ~/code/C/28.RFID $
```

Verwenden Sie den Befehl “[sudo sh ./build.sh](#)”, um die ausführbare RFID-Datei zu generieren, und führen Sie die Datei dann mit “[sudo ./RFID](#)” aus.

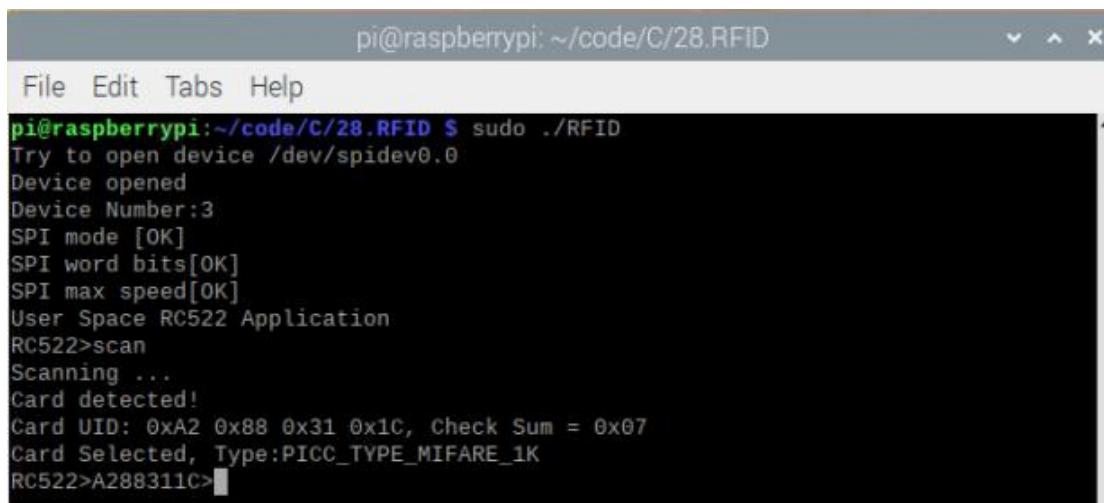
Das Ergebnis ist wie folgt:



```
pi@raspberrypi:~/code/C/28.RFID
File Edit Tabs Help
pi@raspberrypi:~/code/C/28.RFID $ sudo sh ./build.sh
Build finished!
pi@raspberrypi:~/code/C/28.RFID $ sudo ./RFID
Try to open device /dev/spidev0.0
Device opened
Device Number:3
SPI mode [OK]
SPI word bits[OK]
SPI max speed[OK]
User Space RC522 Application
RC522>
```

Geben Sie hier den Befehl “**quit**” ein, um das Programm zu beenden.

Geben Sie den Befehl “**scan**” ein, und das Programm erkennt, ob sich eine Karte in der Nähe des Erfassungsbereichs des MFRC522-Lesegeräts befindet. Legen Sie eine M1-S50-Karte in den Erfassungsbereich. Die folgenden Ergebnisse zeigen an, dass die M1-S50-Karte erkannt wurde, deren UID A288311C (HEX) lautet.



```
pi@raspberrypi:~/code/C/28.RFID
File Edit Tabs Help
pi@raspberrypi:~/code/C/28.RFID $ sudo ./RFID
Try to open device /dev/spidev0.0
Device opened
Device Number:3
SPI mode [OK]
SPI word bits[OK]
SPI max speed[OK]
User Space RC522 Application
RC522>scan
Scanning ...
Card detected!
Card UID: 0xA2 0x88 0x31 0x1C, Check Sum = 0x07
Card Selected, Type:PICC_TYPE_MIFARE_1K
RC522>A288311C>
```

Wenn sich die Karte im Erfassungsbereich befindet, können Sie die Karte mit dem folgenden Befehl lesen und schreiben.

```
Usage:  
    read <blockstart>  
    dump  
    halt  
    clean <blockaddr>  
    write <blockaddr> <data>
```

Im Befehl read <blockstart> ist der Parameter blockstart die Adresse des Datenblocks und der Bereich ist 0-63. Mit diesem Befehl werden alle Daten von der Blockstartadresse bis zum Ende des Sektors angezeigt. Beispielsweise enthält Sektor 0 den Datenblock 0,1,2,3. Mit dem Befehl "read 0" können alle Inhalte des Datenblocks 0,1,2,3 angezeigt werden. Mit dem Befehl "read 1" können alle Inhalte des Datenblocks 1,2,3 angezeigt werden.

Wie unten gezeigt:

```
RC522>A288311C>read 1  
read  
Auth Block (0x01) with key 0xFF 0xFF 0xFF 0xFF 0xFF ...OK  
Read block address 0x01 ....OK read 144 bits  
Read block address 0x02 ....OK read 144 bits  
Read block address 0x03 ....OK read 144 bits  
  0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : .....  
  16: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : .....  
  32: 00 00 00 00 00 ff 07 80 69 ff ff ff ff ff ff : .....i....  
RC522>A288311C>read 0  
read  
Auth Block (0x00) with key 0xFF 0xFF 0xFF 0xFF 0xFF ...OK  
Read block address 0x00 ....OK read 144 bits  
Read block address 0x01 ....OK read 144 bits  
Read block address 0x02 ....OK read 144 bits  
Read block address 0x03 ....OK read 144 bits  
  0: a2 88 31 1c 07 08 04 00 62 63 64 65 66 67 68 69 : ..1....bcdefghi  
  16: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : .....  
  32: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : .....  
  48: 00 00 00 00 00 ff 07 80 69 ff ff ff ff ff ff : .....i....  
RC522>A288311C>
```

Der Befehlsauszug wird verwendet, um den Inhalt aller Datenblöcke in allen Sektoren anzuzeigen.

Mit dem Befehl <Adresse> <Daten> wird "Daten" in den Datenblock mit der Adresse "Adresse" geschrieben. Der Adressbereich ist 0-63 und die Datenlänge ist 0-16. Wenn Sie beispielsweise die Zeichenfolge schreiben möchten In den Datenblock "Hallo" mit der Adresse "1" können Sie den folgenden Befehl eingeben: "write 1 hello".

```
RC522>A288311C>write 1 hello
write
Auth Block (0x01) with key 0xFF 0xFF 0xFF 0xFF 0xFF ...OK
Try to write block 1 with 5 byte data...OK
RC522>A288311C>■
```

Geben Sie den Befehl "read 0" ein, um den Inhalt des Sektors zu lesen und die gerade geschriebenen Daten zu überprüfen.

Die folgenden Ergebnisse zeigen, dass die Zeichenfolge "hello" erfolgreich in den Datenblock 1 geschrieben wurde.

```
RC522>A288311C>read 0
read
Auth Block (0x00) with key 0xFF 0xFF 0xFF 0xFF 0xFF ...OK
Read block address 0x00 ....OK read 144 bits
Read block address 0x01 ....OK read 144 bits
Read block address 0x02 ....OK read 144 bits
Read block address 0x03 ....OK read 144 bits
    0: a2 88 31 1c 07 08 04 00 62 63 64 65 66 67 68 69 : ..1.....bcdefghi
    16: 68 65 6c 6c 6f 00 00 00 00 00 00 00 00 00 00 00 : hello.....
    32: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : .....
    48: 00 00 00 00 00 ff 07 80 69 ff ff ff ff ff : .....i.....
RC522>A288311C>■
```

Mit dem Befehl clean <Adresse> wird der Inhalt des Datenblocks mit der Adresse "Adresse" entfernt. Wenn Sie beispielsweise den Inhalt des gerade geschriebenen Datenblocks 1 löschen möchten, können Sie den folgenden Befehl eingeben: "clean 1".

```
RC522>A288311C>clean 1
clean
Auth Block (0x01) with key 0xFF 0xFF 0xFF 0xFF 0xFF ...OK
Try to clean block 1...OK
RC522>A288311C>■
```

.Lesen Sie den Inhalt der Datenblöcke in diesem Sektor erneut, um zu testen, ob die Daten gelöscht wurden.

Die folgenden Ergebnisse zeigen an, dass der Inhalt von Datenblock 1 gelöscht wurde.

```
RC522>A288311C>read 0
read
Auth Block (0x00) with key 0xFF 0xFF 0xFF 0xFF 0xFF ...OK
Read block address 0x00 ....OK read 144 bits
Read block address 0x01 ....OK read 144 bits
Read block address 0x02 ....OK read 144 bits
Read block address 0x03 ....OK read 144 bits
  0: a2 88 31 1c 07 08 04 00 62 63 64 65 66 67 68 69 : ..1.....bcdefghi
  16: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : .....
  32: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : .....
  48: 00 00 00 00 00 00 ff 07 80 69 ff ff ff ff ff ff : .....i.....
RC522>A288311C>
```

Der Befehl "halt" wird verwendet, um den Auswahlstatus der Karte zu beenden.

```
RC522>A288311C>halt
halt
Halt...
Try to open device /dev/spidev0.0
Device opened
Device Number:6
SPI mode [OK]
SPI word bits[OK]
SPI max speed[OK]
RC522>
```

Das Folgende ist der Programmcode:

```
#include <stdio.h>
#include <stdint.h>
#include <unistd.h>
#include <string.h>
#include <getopt.h>
#include <stdlib.h>
#include "mfrc522.h"
#define DISP_COMMANDLINE() printf("RC522>")

int scan_loop(uint8_t *CardID);
int tag_select(uint8_t *CardID);

int main(int argc, char **argv) {
    MFRC522_Status_t ret;
    //Recognized card ID
    uint8_t CardID[5] = { 0x00, };
```

```
uint8_t tagType[16] = {0x00,};  
static char command_buffer[1024];  
  
ret = MFRC522_Init('B');  
if (ret < 0) {  
    printf("Failed to initialize.\r\nProgram exit.\r\n");  
    exit(-1);  
}  
  
printf("User Space RC522 Application\r\n");  
  
while (1) {  
    /*Main Loop Start*/  
    DISP_COMMANDLINE();  
  
    scanf("%os", command_buffer);  
    if (strcmp(command_buffer, "scan") == 0) {  
        puts("Scanning ... ");  
        while (1) {  
            ret = MFRC522_Request(PICC_REQIDL, tagType);  
            if (ret == MI_OK) {  
                printf("Card detected!\r\n");  
                ret = MFRC522_Anticoll(CardID);  
                if(ret == MI_OK){  
                    ret = tag_select(CardID);  
                    if (ret == MI_OK) {  
                        ret = scan_loop(CardID);  
                        if (ret < 0) {  
                            printf("Card error... \r\n");  
                            break;  
                        } else if (ret == 1) {  
                            puts("Halt...\r\n");  
                            break;  
                        }  
                    }  
                }  
            }  
        }  
    }  
    else{  
        printf("Get Card ID failed!\r\n");  
    }  
}
```

```
        MFRC522_Halt();
    }
    MFRC522_Halt();
    MFRC522_Init('B');
} else if (strcmp(command_buffer, "quit") == 0
    || strcmp(command_buffer, "exit") == 0) {
    return 0;
} else {
    puts("Unknown command");
    puts("scan:scan card and dump");
    puts("quit:exit program");
}
/*Main Loop End*/
}
int scan_loop(uint8_t *CardID) {

    while (1) {

        char input[32];
        int block_start;
        DISP_COMMANDLINE();
        printf("%02X%02X%02X%02X>", CardID[0], CardID[1], CardID[2],
CardID[3]);
        scanf("%s", input);
        puts((char*)input);
        if (strcmp(input, "halt") == 0) {
            MFRC522_Halt();
            return 1;
        } else if (strcmp(input, "dump") == 0) {
            if (MFRC522_Debug_CardDump(CardID) < 0)
                return -1;
        } else if (strcmp(input, "read") == 0) {
            scanf("%d", &block_start);
            if (MFRC522_Debug_DumpSector(CardID, block_start) < 0) {
                return -1;
            }
        } else if (strcmp(input, "clean") == 0) {
            char c;
            scanf("%d", &block_start);
```

```
while ((c = getchar()) != '\n' && c != EOF)
;
if (MFRC522_Debug_Clean(CardID, block_start)) {
    return -1;
}

} else if (strcmp(input, "write") == 0) {
    char write_buffer[256];
    size_t len = 0;
    scanf("%od", &block_start);
    scanf("%s", write_buffer);
    if (len >= 0) {
        if (MFRC522_Debug_Write(CardID, block_start, write_buffer,
            strlen(write_buffer)) < 0) {
            return -1;
        }
    }
} else {

    printf(
        "Usage:\r\n""\tread <blockstart>\r\n""\tdump\r\n""\thalt\r\n"
        "\tclean <blockaddr>\r\n""\twrite <blockaddr><data>\r\n");
    //return 0;
}
}
return 0;

}

int tag_select(uint8_t *CardID) {
    int ret_int;
    printf(
        "Card UID: 0x%02X 0x%02X 0x%02X 0x%02X, Check Sum =
0x%02X\r\n",
        CardID[0], CardID[1], CardID[2], CardID[3], CardID[4]);
    ret_int = MFRC522_SelectTag(CardID);
    if (ret_int == 0) {
        printf("Card Select Failed\r\n");
        return -1;
    } else {
        printf("Card Selected, Type:%s\r\n",

```

```

        MFRC522_TypeToString(MFRC522_ParseType(ret_int)));
    }
    ret_int = 0;
    return ret_int;
}

```

Code Interpretation

```

ret = MFRC522_Init('B');
if (ret < 0) {
    printf("Failed to initialize.\r\nProgram exit.\r\n");
    exit(-1);
}

```

Initialisieren Sie im Code zuerst den MFRC522. Wenn die Initialisierung fehlschlägt, wird das Programm beendet.

```

if (strcmp(command_buffer, "scan") == 0) {
    puts("Scanning ... ");
    while (1) {
        ret = MFRC522_Request(PICC_REQIDL, tagType);
        if (ret == MI_OK) {
            printf("Card detected!\r\n");
            ret = MFRC522_Anticoll(CardID);
            if (ret == MI_OK) {
                ret = tag_select(CardID);
                if (ret == MI_OK) {
                    ret = scan_loop(CardID);
                    if (ret < 0) {
                        printf("Card error...\r\n");
                        break;
                    } else if (ret == 1) {
                        puts("Halt...\r\n");
                        break;
                    }
                }
            }
        }
    }
}
else{
    printf("Get Card ID failed!\r\n");
}

```

```
        }
        MFRC522_Halt();
    }
    MFRC522_Halt();
    MFRC522_Init('B');
} else if (strcmp(command_buffer, "quit") == 0
           || strcmp(command_buffer, "exit") == 0) {
    return 0;
} else {
    puts("Unknown command");
    puts("scan:scan card and dump");
    puts("quit:exit program");
}
/*Main Loop End*/
}
```

Warten Sie in der Hauptfunktion auf die Befehlseingabe. Wenn der Befehl "scan" empfangen wird, erkennt die Funktion, ob sich eine Karte in der Nähe des Erfassungsbereichs befindet. Wenn eine Karte erkannt wird, wird die Karte ausgewählt und die Karten UID erfasst. Geben Sie dann die Funktion "scan_loop ()" ein. Wenn der Befehl "quit" oder "exit" empfangen wird, wird das Programm beendet.

```
int scan_loop(uint8_t *CardID) {

    while (1) {

        char input[32];
        int block_start;
        DISP_COMMANDLINE();
        printf("%02X%02X%02X%02X>", CardID[0], CardID[1], CardID[2],
CardID[3]);
        scanf("%s", input);
        puts((char*)input);
        if (strcmp(input, "halt") == 0) {
            MFRC522_Halt();
            return 1;
        } else if (strcmp(input, "dump") == 0) {
            if (MFRC522_Debug_CardDump(CardID) < 0)
                return -1;
        } else if (strcmp(input, "read") == 0) {
            scanf("%d", &block_start);
```

```

        if (MFRC522_Debug_DumpSector(CardID, block_start) < 0) {
            return -1;
        }
    } else if(strcmp(input, "clean") == 0){
        char c;
        scanf("%d", &block_start);
        while ((c = getchar()) != '\n' && c != EOF)
            ;
        if (MFRC522_Debug_Clean(CardID, block_start)) {
            return -1;
        }

    } else if (strcmp(input, "write") == 0) {
        char write_buffer[256];
        size_t len = 0;
        scanf("%d", &block_start);
        scanf("%s", write_buffer);
        if (len >= 0) {
            if (MFRC522_Debug_Write(CardID, block_start, write_buffer,
                strlen(write_buffer)) < 0) {
                return -1;
            }
        }
    } else {

        printf(
            "Usage:\r\n""\tread <blockstart>\r\n""\tdump\r\n""\thalt\r\n"
            "\tclean <blockaddr>\r\n""\twrite <blockaddr><data>\r\n");
        //return 0;
    }
}
return 0;
}

```

Die Funktion "scan_loop ()" erkennt die Befehle "write" , "clean" , "halt" , "dump" und führt die entsprechende Verarbeitung für jeden Befehl durch. Die Funktion jedes Befehls und die Methode wurden bereits vorgestellt.

Die Header Datei "mfrc522.h" enthält die zugehörige Operationsmethode für den MFRC522. Sie können die Datei öffnen, um alle Definitionen und Funktionen anzuzeigen.

Python code

Der Projektcode verwendet den Befehlszeilenmodus für die Mensch-Computer-Interaktion zum Lesen und Schreiben der M1-S50-Karte.

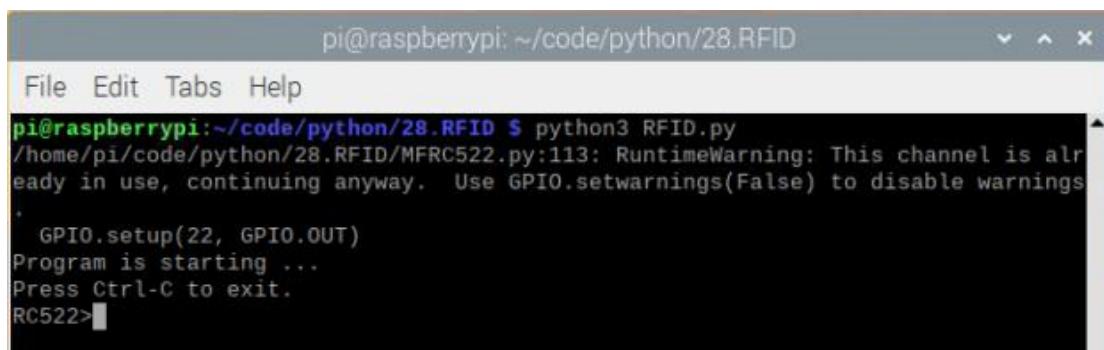
Beobachten Sie zuerst das laufende Ergebnis und analysieren Sie dann den Code.

Geben Sie den Befehl “[cd code / python / 28.RFID /](#)” ein, um zum Codeverzeichnis RFID.py zu wechseln (siehe folgende Abbildung):



```
pi@raspberrypi:~/code/python/28.RFID
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/28.RFID/
pi@raspberrypi:~/code/python/28.RFID $ ls
Dump.py  MFRC522.py  __pycache__  README.md  Read.py  RFID.py  Write.py
pi@raspberrypi:~/code/python/28.RFID $
```

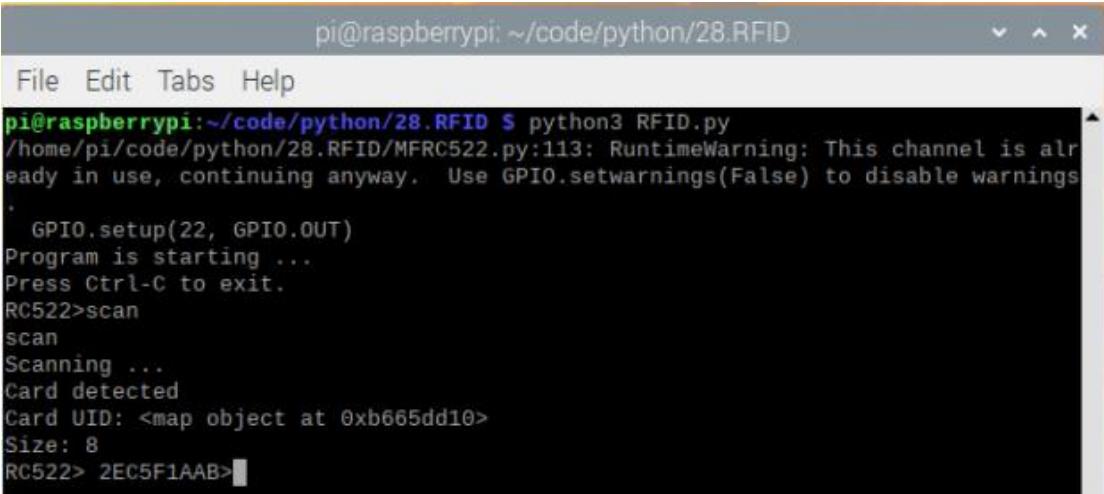
Geben Sie den Befehl “[python3 RFID](#)”.py ein und führen Sie den Code RFID.py aus. Das Ergebnis ist unten dargestellt:



```
pi@raspberrypi:~/code/python/28.RFID
File Edit Tabs Help
pi@raspberrypi:~/code/python/28.RFID $ python3 RFID.py
/home/pi/code/python/28.RFID/MFRC522.py:113: RuntimeWarning: This channel is already in use, continuing anyway.  Use GPIO.setwarnings(False) to disable warnings
  GPIO.setup(22, GPIO.OUT)
Program is starting ...
Press Ctrl-C to exit.
RC522>
```

Geben Sie den Befehl "scan" ein, und das Programm erkennt, ob sich eine Karte in der Nähe des Erfassungsbereichs des MFRC522 Lesegeräts befindet. Legen Sie eine M1-S50 Karte in den Erfassungsbereich. Die folgenden Ergebnisse zeigen an, dass die M1-S50-Karte erkannt wurde (Wenn beim Einsetzen der Karte in den Induktionsbereich keine Reaktion erfolgt, trennen Sie die Stromversorgung des

MFRC522 und schließen Sie sie wieder an, bevor Sie den Code ausführen).



```
pi@raspberrypi:~/code/python/28.RFID$ python3 RFID.py
/home/pi/code/python/28.RFID/MFRC522.py:113: RuntimeWarning: This channel is already in use, continuing anyway.  Use GPIO.setwarnings(False) to disable warnings
'
    GPIO.setup(22, GPIO.OUT)
Program is starting ...
Press Ctrl-C to exit.
RC522>scan
scan
Scanning ...
Card detected
Card UID: <map object at 0xb665dd10>
Size: 8
RC522> 2EC5F1AAB>
```

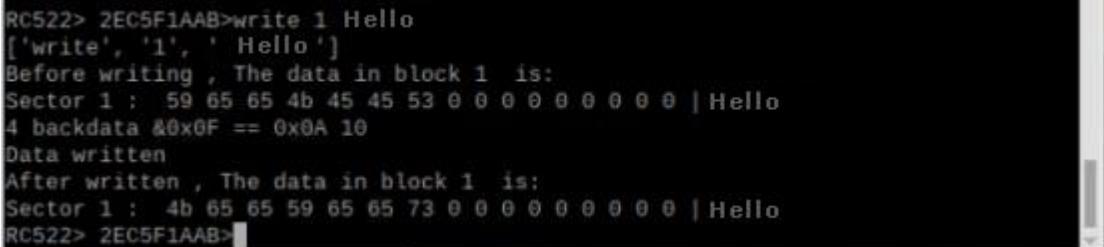
Wenn sich die Karte im Erfassungsbereich befindet, können Sie die Karte mit dem folgenden Befehl lesen und schreiben.



```
Usage:
    read <blockstart>
    dump
    halt
    clean <blockaddr>
    write <blockaddr> <data>

RC522> 2EC5F1AAB>
```

Mit dem Befehl <Adresse> <Daten> wird "Daten" in den Datenblock mit der Adresse "Adresse" geschrieben. Der Adressbereich ist 0-63 und die Datenlänge ist 0-16. Wenn Sie beispielsweise die Zeichenfolge schreiben möchten In den Datenblock "Hallo" mit der Adresse "1" können Sie den folgenden Befehl eingeben: "write 1 hello".



```
RC522> 2EC5F1AAB>write 1 Hello
['write', '1', 'Hello']
Before writing , The data in block 1  is:
Sector 1 :  59 65 65 4b 45 45 53 0 0 0 0 0 0 0 0 | Hello
4 backdata &0x0F == 0xA 10
Data written
After written , The data in block 1  is:
Sector 1 :  4b 65 65 59 65 65 73 0 0 0 0 0 0 0 0 | Hello
RC522> 2EC5F1AAB>
```

Im Befehl read <blockstart> ist der Parameter blockstart die Adresse des Datenblocks und der Bereich ist 0-63. Mit diesem Befehl werden die Daten des Datenblocks mit der Adresse "blockstart" gelesen. Wenn Sie beispielsweise den Befehl read 0

verwenden, kann der Inhalt von Datenblock 0 angezeigt werden. Mit dem Befehl read 1 kann der Inhalt von Datenblock 1 angezeigt werden. Wie unten gezeigt:

```
RC522> 2EC5F1AAB>read 1
['read', '1']
Sector 1 : 4b 65 65 59 65 65 73 0 0 0 0 0 0 0 0 | Hello
RC522> 2EC5F1AAB>read 0
['read', '0']
Sector 0 : 2 ec 5f 1a ab 8 4 0 62 63 64 65 66 67 68 69 | i_#bcdefghi
RC522> 2EC5F1AAB>|
```

Mit dem Befehl clean <Adresse> wird der Inhalt des Datenblocks mit der Adresse "Adresse" entfernt. Wenn Sie beispielsweise den Inhalt des gerade geschriebenen Datenblocks 1 löschen möchten, können Sie den folgenden Befehl eingeben: "clean 1".

```
RC522> 2EC5F1AAB>clean 1
['clean', '1']
Before cleaning , The data in block 1 is:
Sector 1 : 4b 65 65 59 65 65 73 0 0 0 0 0 0 0 0 | Hello
4 backdata &0x0F == 0xA 10
Data written
After cleaned , The data in block 1 is:
Sector 1 : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
RC522> 2EC5F1AAB>|
```

Der Befehl "**“halt”**" wird verwendet, um den Auswahlstatus der Karte zu beenden.

```
RC522> 2EC5F1AAB>halt
['halt']
RC522>|
```

Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import MFRC522
import sys
import os

# Create an object of the class MFRC522
mfrc = MFRC522.MFRC522()

def dis_CommandLine():
    print ("RC522>",end="")
```

```
def dis_CardID(cardID):
    print
    ("%2X%2X%2X%2X>"%(cardID[0],cardID[1],cardID[2],cardID[3],cardID[4]),
end="")

def setup():
    print ("Program is starting ... ")
    print ("Press Ctrl-C to exit.")
    pass

def loop():
    global mfrc
    while(True):
        dis_CommandLine()
        inCmd = input()
        print (inCmd)
        if (inCmd == "scan"):
            print ("Scanning ... ")
            mfrc = MFRC522.MFRC522()
            isScan = True
            while isScan:
                # Scan for cards
                (status,TagType) = mfrc.MFRC522_Request(mfrc.PICC_REQIDL)
                # If a card is found
                if status == mfrc.MI_OK:
                    print ("Card detected")
                # Get the UID of the card
                (status,uid) = mfrc.MFRC522_Anticoll()
                # If we have the UID, continue
                if status == mfrc.MI_OK:
                    print ("Card UID: "+ str(map(hex,uid)))
                    # Select the scanned tag
                    if mfrc.MFRC522_SelectTag(uid) == 0:
                        print ("MFRC522_SelectTag Failed!")
                    if cmdloop(uid) < 1 :
                        isScan = False

        elif inCmd == "quit":
            destroy()
            exit(0)
        else :
```

```
        print ("\tUnknown command\n"+ "\tscan:scan card and
dump\n"+ "\tquit:exit program\n")

def cmdloop(cardID):
    pass
    while(True):
        dis_CommandLine()
        dis_CardID(cardID)
        inCmd = input()
        cmd = inCmd.split(" ")
        print (cmd)
        if(cmd[0] == "read"):
            blockAddr = int(cmd[1])
            if((blockAddr<0) or (blockAddr>63)):
                print ("Invalid Address!")
            # This is the default key for authentication
            key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
            # Authenticate
            status = mfrc.MFRC522_Auth(mfrc.PICC_AUTHENT1A, blockAddr,
key, cardID)
            # Check if authenticated
            if status == mfrc.MI_OK:
                mfrc.MFRC522_Readstr(blockAddr)
            else:
                print ("Authentication error")
                return 0

        elif cmd[0] == "dump":
            # This is the default key for authentication
            key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
            mfrc.MFRC522_Dump_Str(key,cardID)

        elif cmd[0] == "write":
            blockAddr = int(cmd[1])
            if((blockAddr<0) or (blockAddr>63)):
                print ("Invalid Address!")
            data = [0]*16
            if(len(cmd)<2):
                data = [0]*16
            else:
```

```
data = cmd[2][0:17]
data = map(ord,data)
data = list(data)
lenData = len(list(data))
if lenData<16:
    data+=[0]*(16-lenData)
# This is the default key for authentication
key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
# Authenticate
status = mfrc.MFRC522_Auth(mfrc.PICC_AUTHENT1A, blockAddr,
key, cardID)
# Check if authenticated
if status == mfrc.MI_OK:
    print ("Before writing , The data in block %d  is: "%(blockAddr))
    mfrc.MFRC522_Readstr(blockAddr)
    mfrc.MFRC522_Write(blockAddr, data)
    print ("After written , The data in block %d  is: "%(blockAddr))
    mfrc.MFRC522_Readstr(blockAddr)
else:
    print ("Authentication error")
    return 0

elif cmd[0] == "clean":
    blockAddr = int(cmd[1])
    if((blockAddr<0) or (blockAddr>63)):
        print ("Invalid Address!")
    data = [0]*16
    # This is the default key for authentication
    key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
    # Authenticate
    status = mfrc.MFRC522_Auth(mfrc.PICC_AUTHENT1A, blockAddr,
key, cardID)
    # Check if authenticated
    if status == mfrc.MI_OK:
        print ("Before cleaning , The data in block %d  is: "%(blockAddr))
        mfrc.MFRC522_Readstr(blockAddr)
        mfrc.MFRC522_Write(blockAddr, data)
        print ("After cleaned , The data in block %d  is: "%(blockAddr))
        mfrc.MFRC522_Readstr(blockAddr)
    else:
```

```

        print ("Authentication error")
        return 0
    elif cmd[0] == "halt":
        return 0
    else :
        print ("Usage:\r\n""\tread
<blockstart>\r\n""\tdump\r\n""\thalt\r\n""\tclean <blockaddr>\r\n""\twrite
<blockaddr><data>\r\n")
def destroy():
    GPIO.cleanup()
if __name__ == "__main__":
    setup()
    try:
        loop()
    except KeyboardInterrupt: # Ctrl+C captured, exit
        destroy()

```

Code Interpretation

In the code, first create an MFRC522 class object.

```
mfrc = MFRC522.MFRC522()
```

Warten Sie in der Hauptfunktion auf die Befehlseingabe. Wenn der Befehl "scan" empfangen wird, erkennt die Funktion, ob sich eine Karte in der Nähe des Erfassungsbereichs befindet. Wenn eine Karte erkannt wird, wird die Karte ausgewählt und die Karten UID erfasst. Geben Sie dann die Funktion "camloop()" ein. Wenn der Befehl "quit" empfangen wird, wird das Programm beendet.

```

if (inCmd == "scan"):
    print ("Scanning ... ")
    mfrc = MFRC522.MFRC522()
    isScan = True
    while isScan:
        # Scan for cards
        (status,TagType) = mfrc.MFRC522_Request(mfrc.PICC_REQIDL)
        # If a card is found
        if status == mfrc.MI_OK:
            print ("Card detected")
            # Get the UID of the card
            (status,uid) = mfrc.MFRC522_Anticoll()

```

```
# If we have the UID, continue
if status == mfrc.MI_OK:
    print ("Card UID: " + str(map(hex,uid)))
    # Select the scanned tag
    if mfrc.MFRC522_SelectTag(uid) == 0:
        print ("MFRC522_SelectTag Failed!")
    if cmdloop(uid) < 1 :
        isScan = False

elif inCmd == "quit":
    destroy()
    exit(0)
else :
    print ("\tUnknown command\n"+ "\tscan:scan card and
dump\n"+ "\tquit:exit program\n")
```

Die Funktion cmdloop () erkennt den Befehl “read, write, clean, halt, dump und do” die entsprechende Verarbeitung für jeden Befehl durch.

```
def cmdloop( (cardID ):
pass
while( ( True ):
dis_CommandLine ()
dis_CardID( (cardID ) )
inCmd == raw_input ()
cmd == inCmd. .split( ("") )
print cmd
if( (cmd[ [0] ] == "read" ):
.....
elif cmd[ [0] ] == "dump": :
.....
elif cmd[ [0] ] == "write": :
.....
elif cmd[ [0] ] == "clean": :
.....
elif cmd[ [0] ] == "halt": :
return 0
else :::
print "Usage:\r\n""\tread <blockstart>\r\n""\tdump\r\n""\thalt\r\n"
"\tclean <blockaddr>\r\n""\twrite <blockaddr><data>\r\n"
```

Die Datei "MFRC522.py" enthält die zugehörige Operationsmethode für den MFRC522. Sie können die Datei öffnen, um alle Definitionen und Funktionen anzuzeigen.

Lektion 29 4N35

Überblick

In dieser Lektion lernen Sie, wie Sie 4N35 verwenden. 4N35 ist ein Fotokoppler für allgemeine Anwendungen. Es besteht aus einer Infrarot-LED und einem Silizium NPN Fototransistor. Wenn ein Eingangssignal an die LED im Eingangsanschluss angelegt wird, leuchtet die LED auf. Nach dem Empfang des optischen Signals wandelt der optische Empfänger es in ein elektrisches Signal um und gibt das Signal direkt oder nach Verstärkung des Signals auf einen digitalen Standardpegel aus. Der Übergang und die Übertragung von Elektrizität und Elektrizität sind abgeschlossen. Da Licht ein Übertragungsmedium ist, bedeutet dies, dass die Eingangs- und Ausgangsanschlüsse elektrisch isoliert sind. Daher wird dieser Vorgang auch als elektrische Isolation bezeichnet.

Erforderliche Teile

1 x Raspberry pi

1 x 4N35

1 x LED

1 x 220 Ohm Widerstand

1 x 1k ohm Resistor

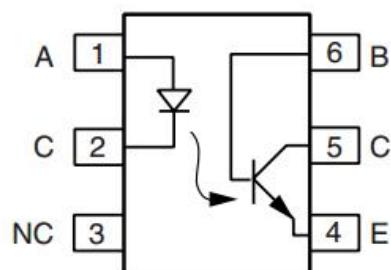
Some Jumper Wiress

Produkt Einführung

4N35

Der Fotokoppler hat die Aufgabe, die Verbindung zwischen der Signalquelle und dem Signalempfänger zu unterbrechen, um elektrische Störungen zu stoppen. Mit anderen Worten wird es verwendet, um Störungen durch externe elektrische Signale zu verhindern. 4N35 kann für AV-Konvertierungs-Audiokreise verwendet werden.

Es wird häufig zur elektrischen Isolierung gewöhnlicher Optokoppler verwendet.

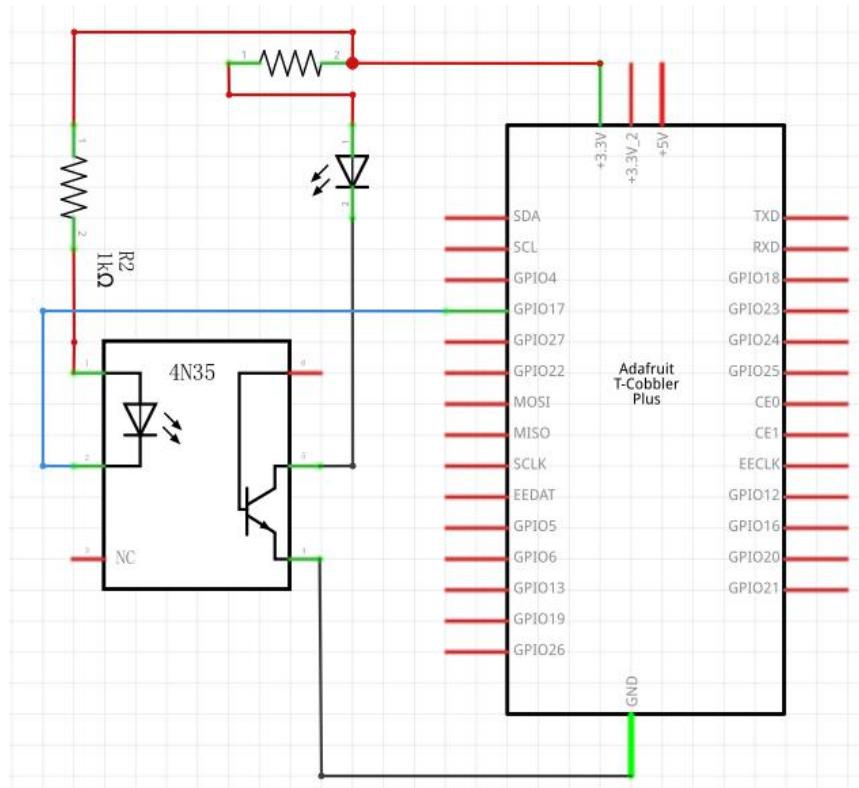


Das Obige ist die interne Struktur von 4N35. Die Pins 1 und 2 sind mit der Infrarot LED verbunden. Wenn die LED eingeschaltet ist, sendet sie Infrarotlicht aus. Um ein Durchbrennen der LED zu verhindern, wird normalerweise ein Widerstand (ca. 1 K) an Pin 1 angeschlossen. Der NPN-Fototransistor schaltet dann die Stromversorgung ein, wenn Licht empfangen wird. Dies kann verwendet werden, um die an den Fototransistor angeschlossene Last zu steuern. Selbst wenn ein Lastkurzschluss auftritt, wirkt sich dies nicht auf die Steuerplatine aus, wodurch eine gute elektrische Isolation erreicht wird.

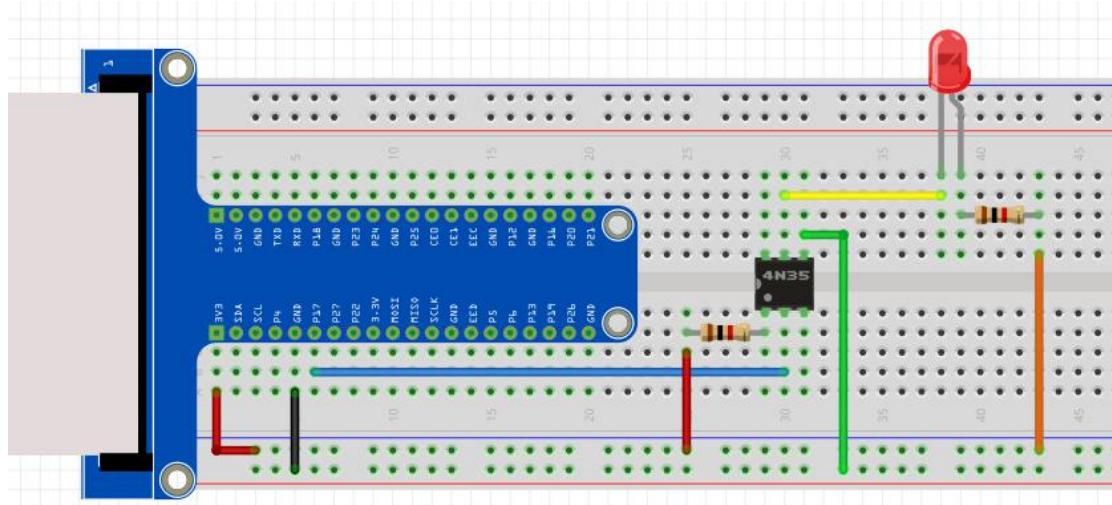


In dieser Lektion wird eine LED als Last verwendet, die an den NPN Fototransistor angeschlossen ist. Verbinden Sie Pin 2 von 4N35 mit Pin P17, verbinden Sie Pin 1 mit einem 1K Strombegrenzungswiderstand und verbinden Sie ihn dann mit 3,3 V. Verbinden Sie Pin 4 mit GND und Pin 5 mit der Kathode der LED. Verbinden Sie dann ein Ende des 220 Ohm Widerstands mit der Anode der LED und das andere Ende mit 3,3 V. Im Programm ist der P17 Pin Niederspannung und die Infrarot LED sendet Infrarotstrahlen aus. Der Fototransistor empfängt dann Infrarotlicht und wird erregt, und die LED-Kathode hat eine niedrige Spannung, um die LED einzuschalten. Sie können die LED auch über den Stromkreis steuern, indem Sie Pin 2 mit Masse verbinden, und sie leuchtet auf.

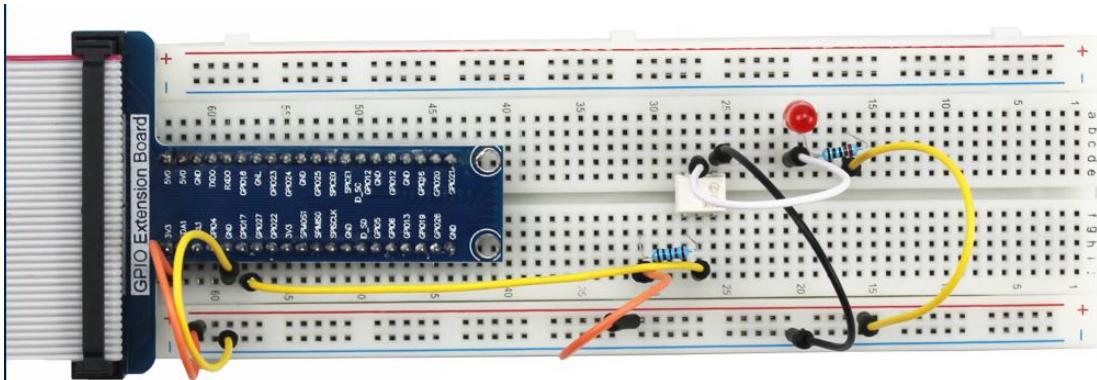
Schaltplan



Verdrahtung Diagramm



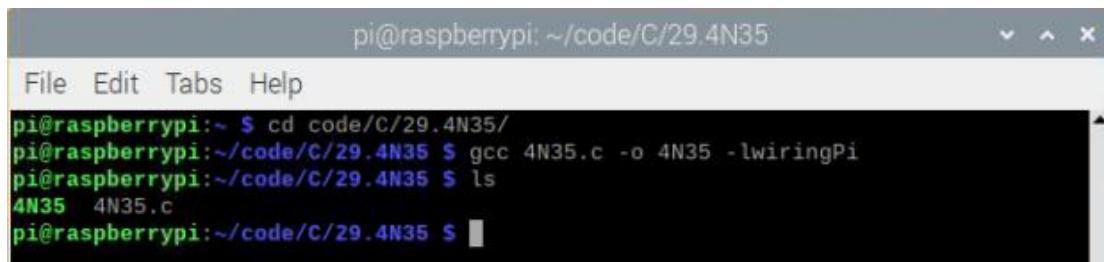
Beispiel Abbildung



C code

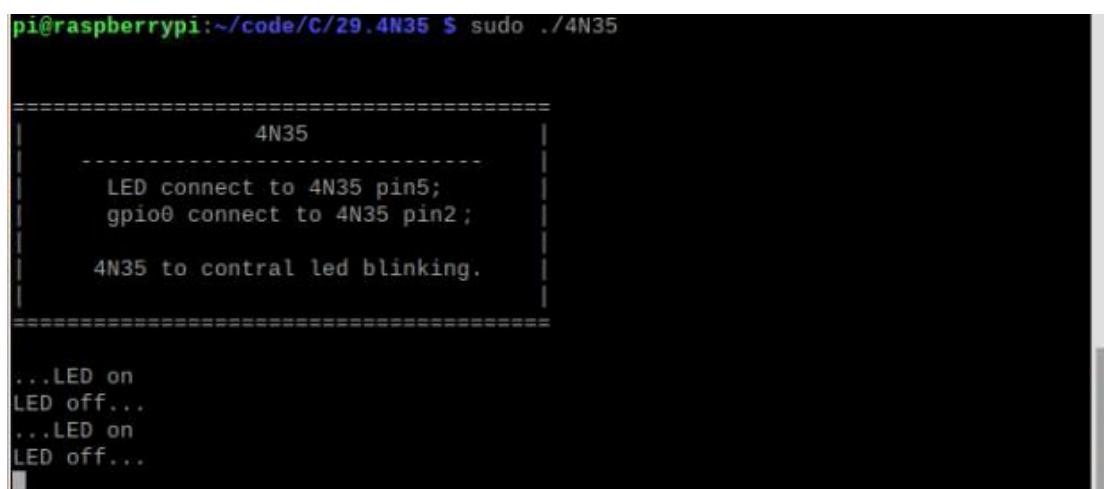
1. Öffnen Sie das Raspberry Pi-Terminal und geben Sie den Befehl “`cd code / C / 29.4N35 /`” ein, um zum Verzeichnis 4N35 zu wechseln.

2. Geben Sie den Befehl “`gcc 4N35.c -o 4N35 -lwiringPi`” ein, um eine ausführbare 4N35-Datei zu generieren. Wie nachfolgend dargestellt;



```
pi@raspberrypi:~/code/C/29.4N35
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/29.4N35/
pi@raspberrypi:~/code/C/29.4N35 $ gcc 4N35.c -o 4N35 -lwiringPi
pi@raspberrypi:~/code/C/29.4N35 $ ls
4N35  4N35.c
pi@raspberrypi:~/code/C/29.4N35 $
```

Geben Sie “`sudo ./4N35`” ein, um den Code auszuführen. Das Ausführungsergebnis ist in der folgenden Abbildung dargestellt:



```
pi@raspberrypi:~/code/C/29.4N35 $ sudo ./4N35
=====
        4N35
-----
    LED connect to 4N35 pin5;
    gpio0 connect to 4N35 pin2;

    4N35 to control led blinking.
=====
...LED on
LED off...
...LED on
LED off...
```

Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>

#define _4N35Pin      0

int main(void)
{
    // When initialize wiring failed, print message to screen
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }

    pinMode(_4N35Pin, OUTPUT);
```

```
printf("\n");
printf("\n");
printf("=====\\n");
printf("|          4N35          |\\n");
printf("|-----|\\n");
printf("|      LED connect to 4N35 pin5;    |\\n");
printf("|      gpio0 connect to 4N35 pin2;    |\\n");
printf("|          |\\n");
printf("|      4N35 to contral led blinking. |\\n");
printf("|          |\\n");
printf("=====|\\n");
printf("\n");
printf("\n");

while(1){
    // LED on
    digitalWrite(_4N35Pin, LOW);
    printf("...LED on\\n");
    delay(1000);
    // LED off
    digitalWrite(_4N35Pin, HIGH);
    printf("LED off...\\n");
    delay(1000);
}

return 0;
}
```

Code Interpretation

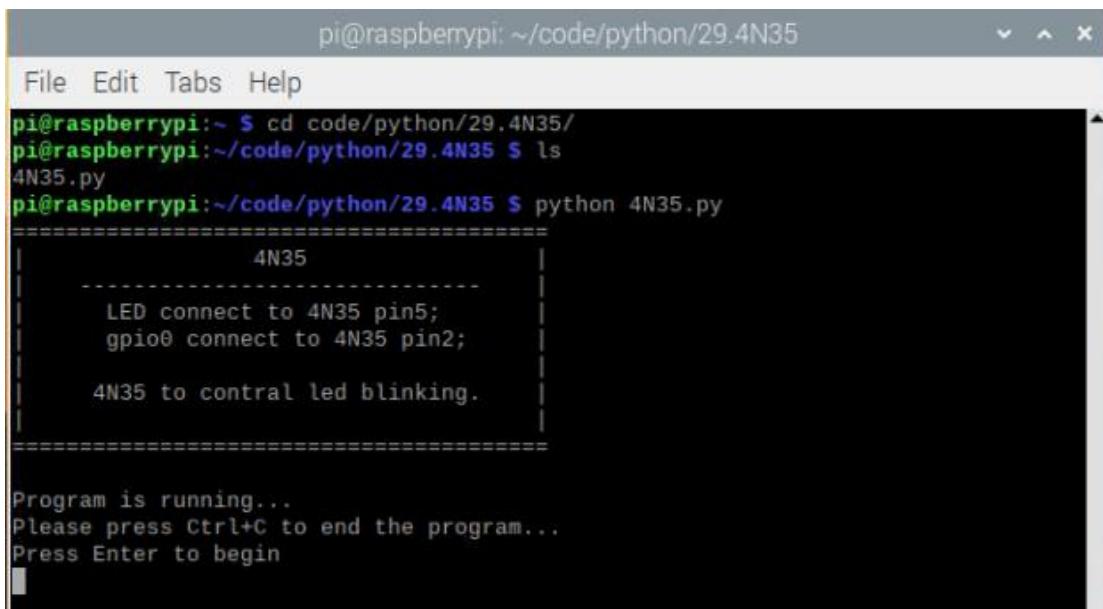
```
while(1){
    // LED on
    digitalWrite(_4N35Pin, LOW);
    printf("...LED on\\n");
    delay(1000);
    // LED off
    digitalWrite(_4N35Pin, HIGH);
    printf("LED off...\\n");
    delay(1000);
```

{

Das Programm ist sehr einfach. Wir haben alle Sätze in den vorherigen Lektionen erklärt. Die Hauptsache ist, die 'digitalWrite' Anweisung zu verwenden, um hohe und niedrige Pegel auszugeben, um die Funktion des Ein- und Ausschaltens der LED in der 'while' Schleife zu realisieren.

Python code

1. Öffnen Sie das Raspberry Pi-Terminal und geben Sie den Befehl “cd code / python / 29.4N35 /” ein, um zum Verzeichnis 4N35 zu wechseln.
2. Geben Sie den Befehl “python 4N35.py” ein, um den Code auszuführen. Wie nachfolgend dargestellt:



```
pi@raspberrypi:~/code/python/29.4N35
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/29.4N35/
pi@raspberrypi:~/code/python/29.4N35 $ ls
4N35.py
pi@raspberrypi:~/code/python/29.4N35 $ python 4N35.py
=====
|          4N35           |
|-----|
| LED connect to 4N35 pin5; |
| gpio0 connect to 4N35 pin2; |
|                               |
| 4N35 to control led blinking. |
|-----|
Program is running...
Please press Ctrl+C to end the program...
Press Enter to begin
```

Drücken Sie die "Enter" taste, um zu sehen, wie die LED blinkt (drücken Sie Ctrl+c, um den Lauf zu beenden). wie in der folgenden Abbildung gezeigt:

```
Program is running...
Please press Ctrl+C to end the program...
Press Enter to begin

...LED ON
LED OFF...
...LED ON
```

Das Folgende ist der Programmcode:

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import time

# Set #17 as 4N35 pin
Pin_4N35 = 17
# Define a function to print message at the beginning
def print_message():
    print ("====")
    print ("|          4N35          |")
    print ("| ----- |")
    print ("|      LED connect to 4N35 pin5;      |")
    print ("|      gpio0 connect to 4N35 pin2;      |")
    print ("|          |")
    print ("|      4N35 to contral led blinking.    |")
    print ("|          |")
    print ("====\n")
    print 'Program is running...'
    print 'Please press Ctrl+C to end the program...'
    raw_input ("Press Enter to begin\n")

# Define a setup function for some setup
```

```
def setup():
    # Set the GPIO modes to BCM Numbering
    GPIO.setmode(GPIO.BCM)
    # Set Pin_4N35's mode to output,
    # and initial level to High(3.3v)
    GPIO.setup(Pin_4N35, GPIO.OUT, initial=GPIO.HIGH)

    # Define a loop function for loop process
    def loop():
        # Print messages
        print_message()
        while True:
            print '...LED ON'
            # Turn on LED
            GPIO.output(Pin_4N35, GPIO.LOW)
            time.sleep(0.5)
            print 'LED OFF...'
            # Turn off LED
            GPIO.output(Pin_4N35, GPIO.HIGH)
            time.sleep(0.5)

    # Define a destroy function for clean up everything after
    # the script finished
    def destroy():
        # Turn off LED
        GPIO.output(Pin_4N35, GPIO.HIGH)
        # Release resource
        GPIO.cleanup()

    # If run this script directly, do:
if __name__ == '__main__':
    setup()
    try:
        loop()
    # When 'Ctrl+C' is pressed, the child program
    # destroy() will be executed.
    except KeyboardInterrupt:
        destroy()
```

Code Interpretation

In diesem Kurs wird kein neuer Satz verwendet, und das Programm enthält detaillierte Kommentarkommentare, sodass in diesem Kurs das Programm nicht erläutert wird. Wenn Sie Fragen haben, lesen Sie bitte die vorherigen Tutorials und Kommentare zum Übersetzungsprogramm

4N35 wird auch zum Ansteuern von Relais und Motorstromkreisen verwendet. Da keine direkte Verbindung zwischen Eingang und Ausgang besteht, wird die Steuerplatine auch dann nicht verbrannt, wenn am Ausgang ein Kurzschluss auftritt. Wenn Sie daran interessiert sind, können Sie experimentieren.

Lektion 30 NE555

Überblick

In dieser Lektion lernen Sie, wie Sie den Timer NE555 verwenden.

Erforderliche Teile

1 x Raspberry Pi

1 x NE555 Chip

1 x 220 Ohm Widerstand

1 x 1K Widerstand

2 x 10K Widerstand

2 x Kondensatoren (100nF)

20 x doppelte Männlich Jumper Kabel

1 x Steckbrett

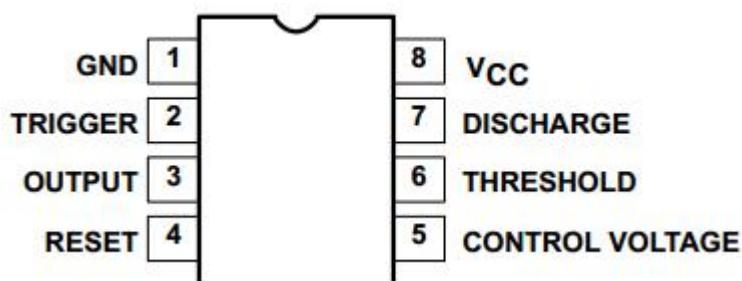
1 x LED

Produkt Einführung

Der NE555 Timer ist eine Hybridschaltung, die aus analogen und digitalen Schaltungen besteht. Es integriert analoge und logische Funktionen in unabhängige ICs und erweitert dadurch die Anwendung analoger integrierter Schaltungen erheblich. Es ist weit verbreitet in verschiedenen Zeitgebern, Impulsgeneratoren und Oszillatoren.

Der 555 Timer ist ein mittelgroßes IC-Gerät, das analoge und digitale Funktionen kombiniert. Es hat niedrige Kosten und zuverlässige Leistung. Es müssen nur externe Widerstände und Kondensatoren angeschlossen werden, um Multivibrator, monostabiles Flipflop, Schmitt-Trigger und andere Schaltungen zu implementieren, die Impulse erzeugen und umwandeln können. Es wird auch häufig als Zeitschaltuhr verwendet und ist in Instrumenten, Haushaltsgeräten, elektronischer Messung, automatischer Steuerung und anderen Bereichen weit verbreitet.

Pins und Funktionen:



Wie gezeigt, sind die Stifte sind mit dem 8-poligen zweireihig aufgebaut.

Pin 1 (GND): Masse

Pin 2 (TRIGGER): Eingang des Low-Side-Komparators

Pin 3 (OUTPUT): hat zwei Zustände 0 und 1, die durch den Eingangspegel bestimmt werden

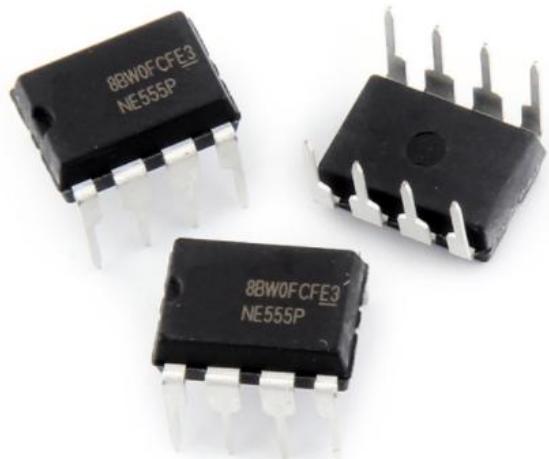
Pin 4 (RESET): Niedriger Pegel ausgeben, wenn niedriger Pegel bereitgestellt wird

Pin 5 (Steuerspannung): Ändern Sie den oberen und unteren Triggerwert

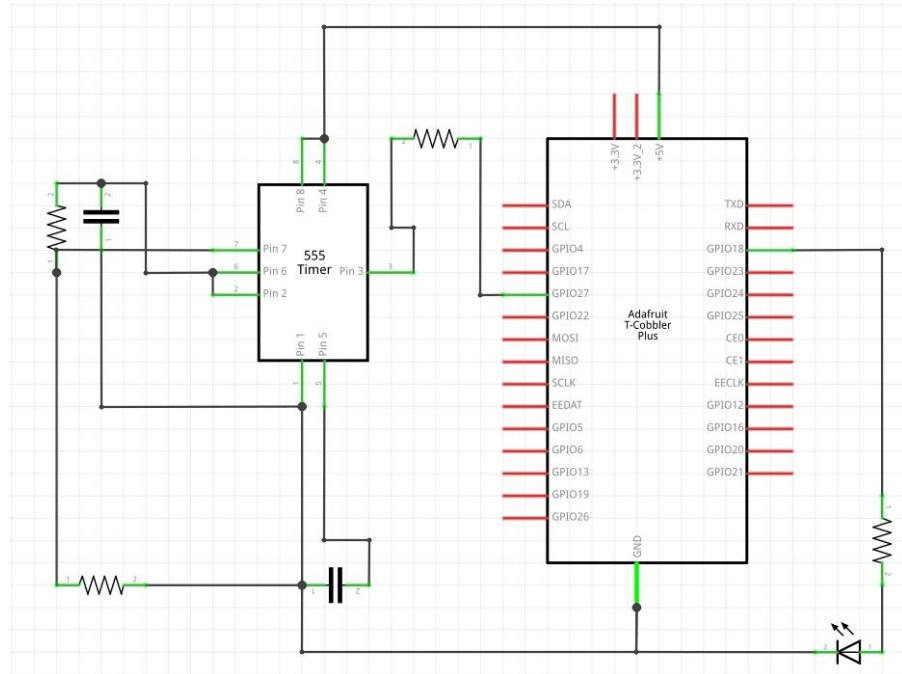
Pin 6 (THRESHOLD): Eingang des oberen Komparators

Pin 7 (ENTLADEN): Die beiden Zustände von Aufhängung und Masse werden auch durch den Ein- und Ausgang der internen Entladungsrohre bestimmt

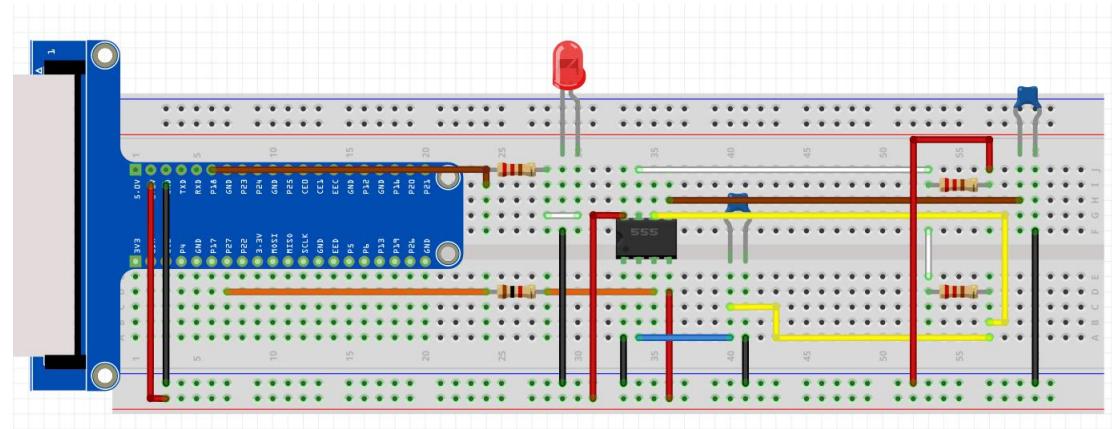
Pin 8 (VCC): Stromversorgung



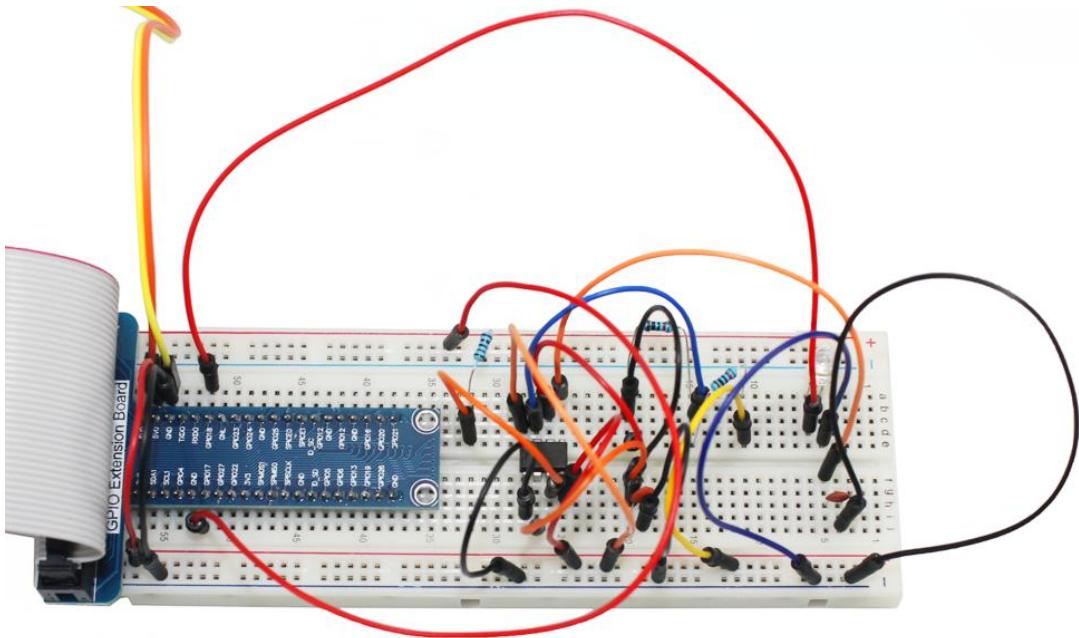
Schaltplan



Verdrahtung Diagramm



Beispiel Abbildung



C code

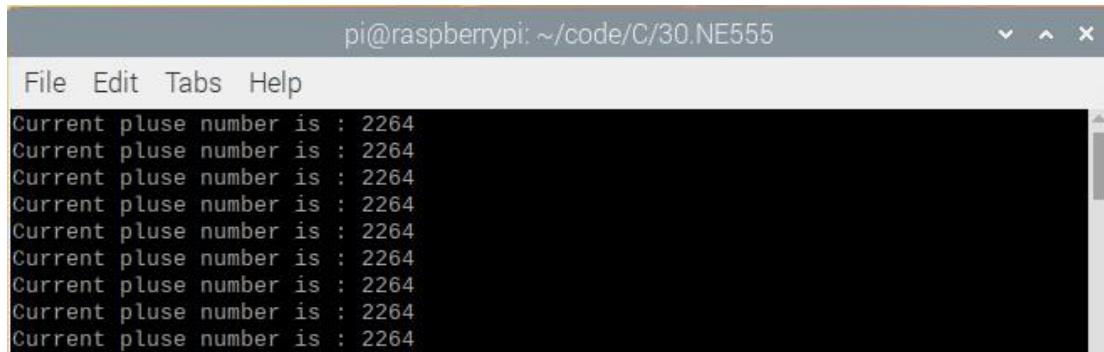
1. Öffnen Sie das Terminal und geben Sie den Befehl “`cd code / C / 30.NE555 /`” ein, um das Codeverzeichnis ne555.c aufzurufen.

2. Geben Sie den Befehl “`ls`” ein, um die Datei ne555.c im Verzeichnis anzuzeigen.

```
pi@raspberrypi: ~/code/C/30.NE555
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/C/30.NE555/
pi@raspberrypi:~/code/C/30.NE555 $ ls
ne555.c
pi@raspberrypi:~/code/C/30.NE555 $
```

3. Geben Sie den Befehl “`gcc ne555.c -o ne555 -lwiringPi`” ein, um die ausführbare Datei ne555.c ne555 zu generieren. Geben Sie den Befehl `ls` ein, um ihn anzuzeigen.

4. Geben Sie den Befehl “`sudo ./ne555`” ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi: ~ /code/C/30.NE555
File Edit Tabs Help
Current pluse number is : 2264
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include <wiringPi.h>

#define Pin0 0
#define led 1

static volatile int globalCounter = 0 ;

void exInt0_ISR(void) //GPIO0 interrupt service routine
{
    ++globalCounter;
}

int main (void)
{
    if(wiringPiSetup() < 0){
        fprintf(stderr, "Unable to setup wiringPi:%s\n",strerror(errno));
        return 1;
    }

    wiringPiISR(Pin0, INT_EDGE_FALLING, &exInt0_ISR);
    printf("wiringPi initialize successfully, GPIO %d(wiringPi pin)\n",led);
```

```

pinMode(led, OUTPUT);
while(1){
    if(globalCounter/100%2==0 )
    {
        digitalWrite(led, HIGH);
    }
    else
    {
        digitalWrite(led, LOW);

    }
    printf("Current pluse number is : %d\n", globalCounter);
}

return 0;
}

```

Code Interpretation

```

void exInt0_ISR(void) //GPIO0 interrupt service routine
{
    ++globalCounter;
}

```

Die Interrupt-Funktion wird einmal ausgeführt und der 'globalCounter' wird um eins erhöht.

```

while(1){
    if(globalCounter/100%2==0 )
    {
        digitalWrite(led, HIGH);
    }
    else
    {
        digitalWrite(led, LOW);

    }
    printf("Current pluse number is : %d\n", globalCounter);
}

```

Bestimmen Sie in der Schleife, ob 'g_count / 100% 2' gleich 0 ist, damit die LED blinkt, und drucken Sie den Wert von 'globalCounter'.

Python code

Öffnen Sie das Terminal und geben Sie mit dem Befehl “`cd code / python / 30.NE555 /`” das Codeverzeichnis ein.

Geben Sie den Befehl “ls” ein, um die Datei ne555.py im Verzeichnis anzuzeigen.

```
pi@raspberrypi: ~code/python/30.NE555
File Edit Tabs Help
pi@raspberrypi:~ $ cd code/python/30.NE555/
pi@raspberrypi:~/code/python/30.NE555 $ ls
ne555.py
pi@raspberrypi:~/code/python/30.NE555 $
```

Geben Sie den Befehl “`python3 ne555.py`” ein, um den Code auszuführen. Das Ergebnis ist wie folgt:

Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO  
SigPin = 11  
ledPin = 12  
g_count = 0  
def count(ev=None):
```

```

global g_count
g_count += 1

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(SigPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
        GPIO.add_event_detect(SigPin, GPIO.RISING, callback=count)
        GPIO.setup(ledPin, GPIO.OUT)
        GPIO.output(ledPin, GPIO.LOW)
def loop():
    while True:
        if(g_count/100%2==0):
            GPIO.output(ledPin, GPIO.HIGH)
        else:
            GPIO.output(ledPin, GPIO.LOW)
        print ('g_count = %d' % g_count)

def destroy():
    GPIO.cleanup()

if __name__ == '__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
destroy() will be executed.
        destroy()

```

Code Interpretation

```

def count(ev=None):
    global g_count
    g_count += 1

```

Interrupt-Funktion = einmal ausführen, um 'g_count' um 1 zu erhöhen;

```

def loop():
    while True:
        if(g_count/100%2==0):
            GPIO.output(ledPin, GPIO.HIGH)

```

```
else:  
    GPIO.output(ledPin, GPIO.LOW)  
    print ('g_count = %d' % g_count)
```

In der Schleife wird beurteilt, ob 'g_count / 100% 2' gleich 0 ist, damit die LED blinkt, und der Wert von 'g_count' wird gedruckt.

Lektion 31 Infrarot Fernbedienung

Überblick

In dieser Lektion lernen Sie, wie Sie mit Raspberry Pi LED Leuchten einer Infrarot Fernbedienung ansteuern.

Erforderliche Teile

1 x Raspberry Pi

1 x IR Fernbedienung

8 x doppelte Männlich Jumper Kabel

1 x Steckbrett

3 x LED

3 x 220 Ohm Widerstand

Produkt Einführung

Infrared Remote Control

Die Infrarot Fernsteuerungs Sendeschaltung verwendet Infrarot Leuchtdioden, um modulierte Infrarot Lichtwellen zu emittieren. Die Infrarot-Empfangsschaltung besteht aus Infrarot-Empfangsdioden, Trioden oder Silizium Fotozellen. Sie wandeln

das vom Infrarotsender emittierte Infrarotlicht in das entsprechende elektrische Signal um und senden Sie es dann an den hinteren Verstärker.

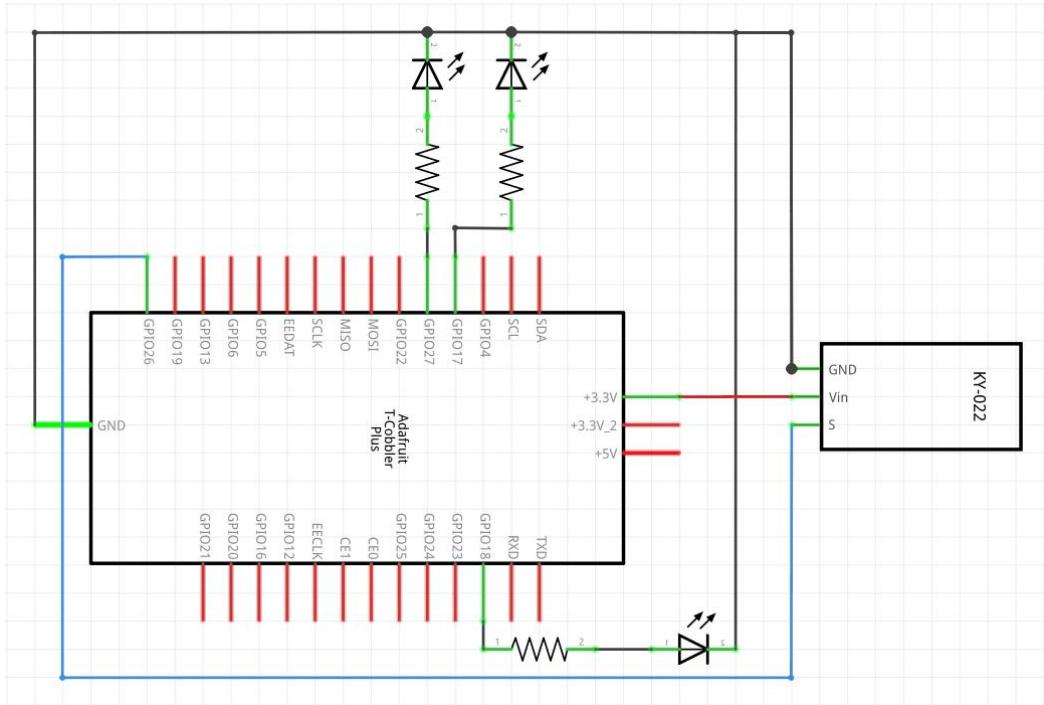
Der Sender besteht im Allgemeinen aus einer Befehlstaste (oder einem Joystick), einem Befehlscodierungssystem, einer Modulationsschaltung, einer Ansteuerschaltung und einer Sendeschaltung. Wenn die Befehlstaste gedrückt oder der Joystick gedrückt wird, erzeugt die Befehlskodierungsschaltung das erforderliche Befehlskodierungssignal, das Befehlskodierungssignal moduliert die Trägerwelle, und dann führt die Treiberschaltung eine Leistungsverstärkung durch, dann sendet die Sendeschaltung das befehlscodierte Signal, das eingestellt und formuliert wurde.

Die Empfangsschaltung besteht im Allgemeinen aus einer Empfangsschaltung, einer Verstärkungsschaltung, einer Modulationsschaltung, einer Befehlsdecodierungsschaltung, einer Ansteuerschaltung und einer Ausführungsschaltung (Mechanismus). Die Empfangsschaltung empfängt das vom Sender gesendete modulierte codierte Befehlssignal, verstärkt es und sendet es an die Demodulationsschaltung. Die Demodulationsschaltung demoduliert das modulierte befehlscodierte Signal, dh das codierte Signal wird wiederhergestellt. Der Befehlsdecodierer decodiert das codierte Befehlssignal, und schließlich steuert die Treiberschaltung die Ausführungsschaltung an, um die Betriebssteuerung verschiedener Befehle zu realisieren.

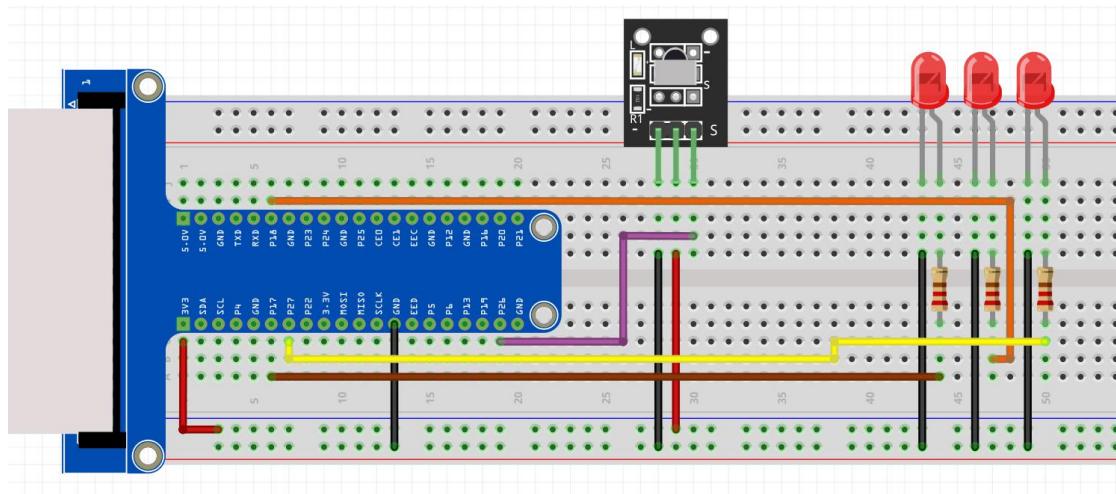




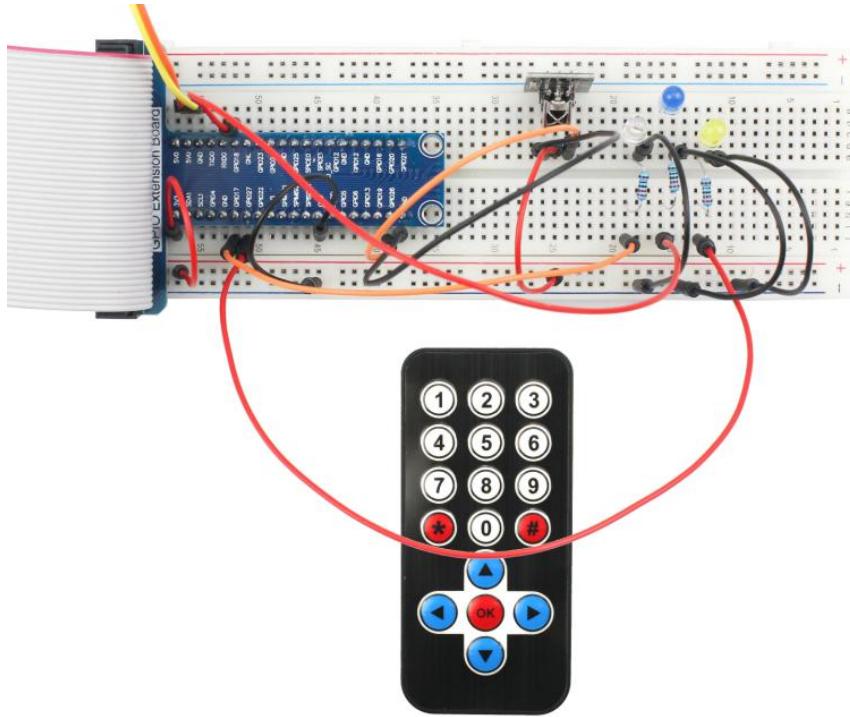
Schaltplan



Verdrahtung Diagramm



Beispiel Abbildung



C code

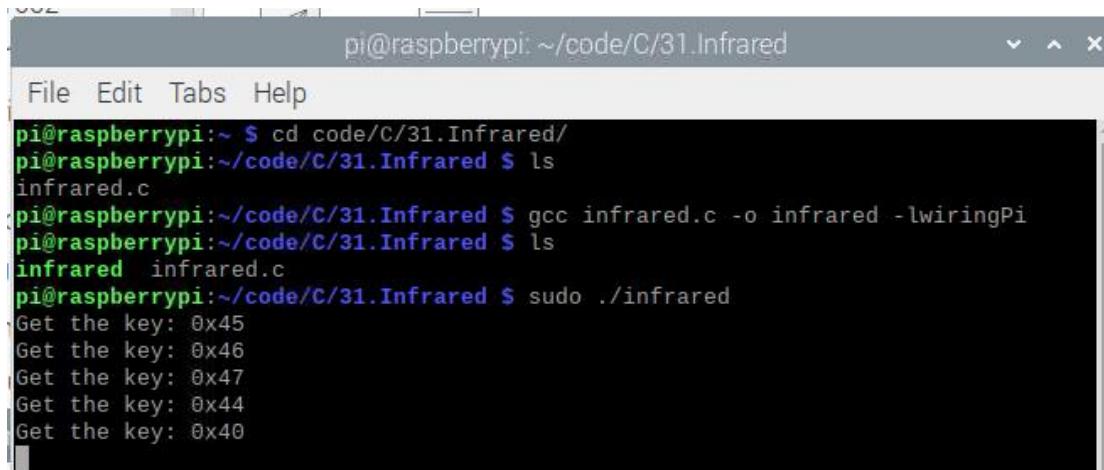
Öffnen Sie das Terminal und geben Sie den Befehl "[cd code / C / 31.Infrared /](#)" ein, um das Verzeichnis infrared.c code aufzurufen.

Geben Sie den Befehl "[ls](#)" ein, um die Datei "infrared.c" im Verzeichnis anzuzeigen.

```
pi@raspberrypi:~/code/C/31.Infrared
File Edit Tabs Help
pi@raspberrypi:~$ cd code/C/31.Infrared/
pi@raspberrypi:~/code/C/31.Infrared$ ls
infrared.c
pi@raspberrypi:~/code/C/31.Infrared$
```

Geben Sie den Befehl "[gcc infrared.c -o infrarot -lwiringPi](#)" ein, um die ausführbare Datei "infrared.c" "infrarot" zu generieren, und geben Sie den Befehl "ls" ein, um sie anzuzeigen.

Geben Sie den Befehl "[sudo ./infrared](#)" ein, um den Code auszuführen. Das Ergebnis ist wie folgt:



```
pi@raspberrypi:~/code/C/31.Infrared
pi@raspberrypi:~/code/C/31.Infrared$ cd code/C/31.Infrared/
pi@raspberrypi:~/code/C/31.Infrared$ ls
infrared.c
pi@raspberrypi:~/code/C/31.Infrared$ gcc infrared.c -o infrared -lwiringPi
pi@raspberrypi:~/code/C/31.Infrared$ ls
infrared infrared.c
pi@raspberrypi:~/code/C/31.Infrared$ sudo ./infrared
Get the key: 0x45
Get the key: 0x46
Get the key: 0x47
Get the key: 0x44
Get the key: 0x40
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>

#define Pin 25
#define delay_time 120
int LED[3]={0,1,2};
char i,idx,ent;
char count;
char data[4];

static int flag = 0;
int exec_cmd(char key_val)
{
    switch(key_val) {
        case 0x45://1
            digitalWrite(LED[0],HIGH);
            digitalWrite(LED[1],LOW);
            digitalWrite(LED[2],LOW);
            break;
        case 0x46://2
```

```
        digitalWrite(LED[0],LOW);
        digitalWrite(LED[1],HIGH);
        digitalWrite(LED[2],LOW);
        break;
    case 0x47://3
        digitalWrite(LED[0],LOW);
        digitalWrite(LED[1],LOW);
        digitalWrite(LED[2],HIGH);
        break;
    default:
        digitalWrite(LED[0],LOW);
        digitalWrite(LED[1],LOW);
        digitalWrite(LED[2],LOW);
        break;
    }

    return 0;
}

void setup()
{
    int i;
    pinMode(Pin, INPUT);
    for(i=0;i<3;i++)
    {
        pinMode(LED[i], OUTPUT);
    }
}

int main(int argc, char const *argv[])
{
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("failed !");
        return 1;
    }
    setup();

    while (1) {
        if (digitalRead(Pin) == LOW) {
            count = 0;
```

```

        while (digitalRead(Pin) == LOW && count++ < 200)    //9ms
            delayMicroseconds(delay_time);
        count = 0;
        while (digitalRead(Pin) == HIGH && count++ < 80)     //4.5ms
            delayMicroseconds(delay_time);
        idx = 0;
        cnt = 0;
        data[0]=0;
        data[1]=0;
        data[2]=0;
        data[3]=0;
        for (i =0;i<32;i++) {
            count = 0;
            while (digitalRead(Pin) == LOW && count++ < 15)   //0.56ms
                delayMicroseconds(delay_time);

            count = 0;
            while (digitalRead(Pin) == HIGH && count++ < 40) //0: 0.56ms;
1: 1.69ms delayMicroseconds(60); if (count > 25)
            delayMicroseconds(delay_time);
            if (count > 8)
                data[idx] |= 1<<cnt;
            if (cnt== 7) {
                cnt=0;
                idx++;
            } else {
                cnt++;
            }
            if ((data[0]+data[1] == 0xFF) && (data[2]+data[3]==0xFF))
{
    //check
        printf("Get the key: 0x%02x\n",data[2]);
        exec_cmd(data[2]);
    }
}
}
return 0;
}

```

Code Interpretation

Die Funktion "setup ()" setzt die Infrarot Fernbedienung auf den Eingangsmodus und die LED auf den Ausgangsmodus.

```
void setup()
{
    int i;
    pinMode(Pin, INPUT);
    for(i=0;i<3;i++)
    {
        pinMode(LED[i], OUTPUT);
    }
}
```

Die Funktion "exec_cmd (char key_val)" wird hauptsächlich zum Einschalten des LED Lichts verwendet. Zunächst wird beurteilt, ob der decodierte Wert gleich dem entsprechenden Codewert der Taste ist. Wenn es sich um den Codewert handelt, der der Taste "1, 2, 3" entspricht, schalten Sie das entsprechende LED Licht ein, andernfalls schalten Sie das LED Licht aus.

```
int exec_cmd(char key_val)
{
    switch(key_val) {
        case 0x45://1
            digitalWrite(LED[0],HIGH);
            digitalWrite(LED[1],LOW);
            digitalWrite(LED[2],LOW);
            break;
        case 0x46://2
            digitalWrite(LED[0],LOW);
            digitalWrite(LED[1],HIGH);
            digitalWrite(LED[2],LOW);
            break;
        case 0x47://3
            digitalWrite(LED[0],LOW);
            digitalWrite(LED[1],LOW);
            digitalWrite(LED[2],HIGH);
            break;
        default:
            digitalWrite(LED[0],LOW);
    }
}
```

```

        digitalWrite(LED[1],LOW);
        digitalWrite(LED[2],LOW);
        break;
    }

    return 0;
}

In der "for" Schleife der Hauptfunktion wird der Codewert des entsprechenden empfangenen Schlüsselwerts gemäß der ungefähren Zeitdifferenz zwischen den hohen und niedrigen Pegeln der empfangenen Infrarotfernbedienung decodiert.

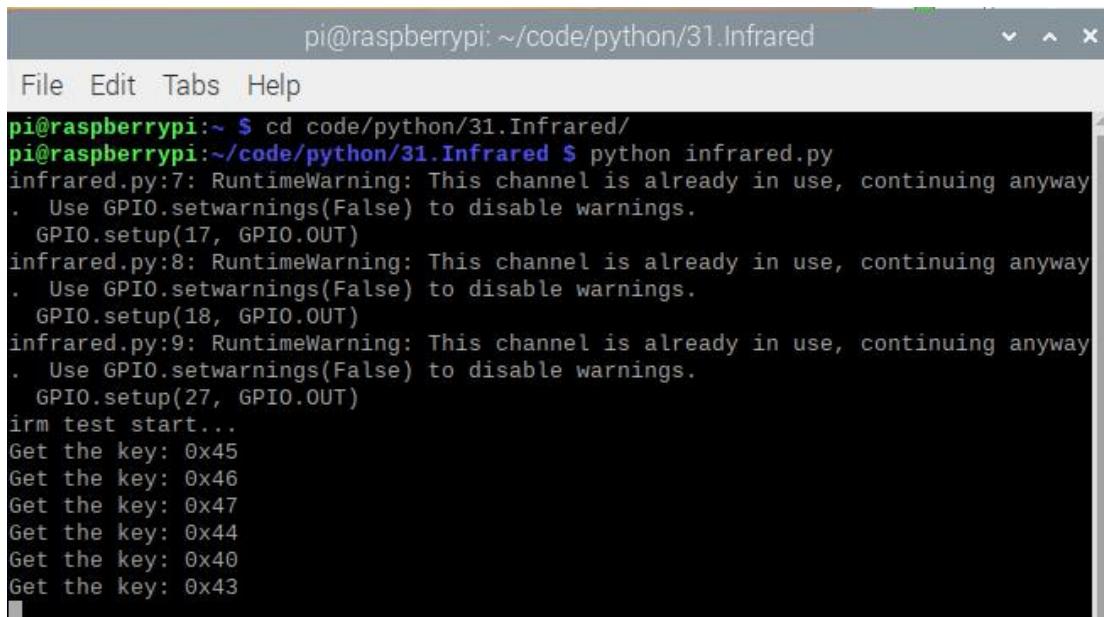
for (i =0;i<32;i++) {
    count = 0;
    while (digitalRead(Pin) == LOW && count++ < 15) //0.56ms
        delayMicroseconds(delay_time);

    count = 0;
    while (digitalRead(Pin) == HIGH && count++ < 40) //0: 0.56ms;
1: 1.69ms delayMicroseconds(60); if (count > 25)
    delayMicroseconds(delay_time);
    if (count > 8)
        data[idx] |= 1<<cnt;
    if (cnt== 7) {
        cnt=0;
        idx++;
    } else {
        cnt++;
    }
    if ((data[0]+data[1] == 0xFF) && (data[2]+data[3]==0xFF))
{
    //check
        printf("Get the key: 0x%02x\n",data[2]);
        exec_cmd(data[2]);
    }
}

```

Python code

1. Verwenden Sie den Befehl "`cd code / python / 31.Infrared /`", um zum Verzeichnis "Infrared" zu wechseln.
2. Verwenden Sie den Befehl "`python infrared.py`", um den Code "infrared.py" auszuführen.



The screenshot shows a terminal window titled "pi@raspberrypi: ~/code/python/31.Infrared". The window contains the following text:

```
pi@raspberrypi:~ $ cd code/python/31.Infrared/
pi@raspberrypi:~/code/python/31.Infrared $ python infrared.py
infrared.py:7: RuntimeWarning: This channel is already in use, continuing anyway
. Use GPIO.setwarnings(False) to disable warnings.
    GPIO.setup(17, GPIO.OUT)
infrared.py:8: RuntimeWarning: This channel is already in use, continuing anyway
. Use GPIO.setwarnings(False) to disable warnings.
    GPIO.setup(18, GPIO.OUT)
infrared.py:9: RuntimeWarning: This channel is already in use, continuing anyway
. Use GPIO.setwarnings(False) to disable warnings.
    GPIO.setup(27, GPIO.OUT)
irm test start...
Get the key: 0x45
Get the key: 0x46
Get the key: 0x47
Get the key: 0x44
Get the key: 0x40
Get the key: 0x43
```

Drücken Sie "Ctrl + c", um das Programm zu beenden. Das Folgende ist der Programmcode:

```
import RPi.GPIO as GPIO
import time
PIN = 26;
def setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(PIN,GPIO.IN,GPIO.PUD_UP)
    GPIO.setup(17, GPIO.OUT)
    GPIO.setup(18, GPIO.OUT)
    GPIO.setup(27, GPIO.OUT)
    print("irm test start...")

def exec_cmd(key_val):
    if(key_val==0x45):
        GPIO.output(17, GPIO.HIGH)
```

```
GPIO.output(18, GPIO.LOW)
GPIO.output(27, GPIO.LOW)
elif(key_val==0x46):
    GPIO.output(17, GPIO.LOW)
    GPIO.output(18, GPIO.HIGH)
    GPIO.output(27, GPIO.LOW)
elif(key_val==0x47):
    GPIO.output(17, GPIO.LOW)
    GPIO.output(18, GPIO.LOW)
    GPIO.output(27, GPIO.HIGH)
else :
    GPIO.output(17, GPIO.LOW)
    GPIO.output(18, GPIO.LOW)
    GPIO.output(27, GPIO.LOW)

def loop():
    while True:
        if GPIO.input(PIN) == 0:
            count = 0
            while GPIO.input(PIN) == 0 and count < 200:
                count += 1
                time.sleep(0.00006)

            count = 0
            while GPIO.input(PIN) == 1 and count < 80:
                count += 1
                time.sleep(0.00006)

        idx = 0
        cnt = 0
        data = [0,0,0,0]
        for i in range(0,32):
            count = 0
            while GPIO.input(PIN) == 0 and count < 15:
                count += 1
                time.sleep(0.00006)

            count = 0
            while GPIO.input(PIN) == 1 and count < 40:
                count += 1
```

```

time.sleep(0.00006)

if count > 8:
    data[idx] |= 1<<cnt
if cnt == 7:
    cnt = 0
    idx += 1
else:
    cnt += 1
if data[0]+data[1] == 0xFF and data[2]+data[3] == 0xFF:
    print("Get the key: 0x%02x" %data[2])
    exec_cmd(data[2])

def destroy():
    GPIO.output(PIN, GPIO.LOW)      # led off
    GPIO.cleanup()                  # Release resource

if __name__ == '__main__':      # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:   # When 'Ctrl+C' is pressed, the child program
destroy() will be executed.
    destroy()

```

Code Interpretation

Die Funktion "setup ()" setzt die Infrarot-Fernbedienung auf den Eingangsmodus und die LED auf den Ausgangsmodus.

```

def setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(PIN,GPIO.IN,GPIO.PUD_UP)
    GPIO.setup(17, GPIO.OUT)
    GPIO.setup(18, GPIO.OUT)
    GPIO.setup(27, GPIO.OUT)
    print("irm test start...")

```

Die Funktion "exec_cmd (key_val)" wird hauptsächlich zum Einschalten des LED-Lichts verwendet. Zunächst wird beurteilt, ob der decodierte Wert dem entsprechenden Codewert des Schlüssels entspricht. Wenn es sich um den Codewert

handelt, der der Taste "1, 2, 3" entspricht, schalten Sie das entsprechende LED-Licht ein, andernfalls schalten Sie es das LED Licht aus.

```
def exec_cmd(key_val):
    if(key_val==0x45):
        GPIO.output(17, GPIO.HIGH)
        GPIO.output(18, GPIO.LOW)
        GPIO.output(27, GPIO.LOW)
    elif(key_val==0x46):
        GPIO.output(17, GPIO.LOW)
        GPIO.output(18, GPIO.HIGH)
        GPIO.output(27, GPIO.LOW)
    elif(key_val==0x47):
        GPIO.output(17, GPIO.LOW)
        GPIO.output(18, GPIO.LOW)
        GPIO.output(27, GPIO.HIGH)
    else :
        GPIO.output(17, GPIO.LOW)
        GPIO.output(18, GPIO.LOW)
        GPIO.output(27, GPIO.LOW)
```

In der "for" Schleife der Hauptfunktion wird der Codewert des entsprechenden empfangenen Schlüsselwerts gemäß der ungefähren Zeitdifferenz zwischen den hohen und niedrigen Pegeln der empfangenen Infrarotfernbedienung decodiert.

```
for i in range(0,32):
    count = 0
    while GPIO.input(PIN) == 0 and count < 15:
        count += 1
        time.sleep(0.00006)

    count = 0
    while GPIO.input(PIN) == 1 and count < 40:
        count += 1
        time.sleep(0.00006)

    if count > 8:
        data[idx] |= 1<<cnt
    if cnt == 7:
        cnt = 0
```

```
    idx += 1
else:
    cnt += 1
```

Lektion 32 Summer Schweißen

Überblick

Wir werden Leiterplatten verwenden, um Schaltungen und Komponenten zu löten. Bis zum Ende dieses Kapitels möchte ich Ihnen helfen, das Entwerfen Ihrer eigenen Schaltungen zu beherrschen und Ideen für den Bau von Leiterplatten zu inspirieren. Um dieses Kapitel zu vervollständigen, müssen Sie die erforderlichen Lötgeräte vorbereiten, einschließlich Lötkolben und Draht sowie anderer Hilfsmaterialien. Wir haben für Sie Schweißmaterialien wie Universalplatine, Zinndraht, Stiftleiste und Buchsenkopf vorbereitet. Da die Temperatur des Lötkolbens Hunderte von Grad oder mehr erreichen kann, seien Sie während des Betriebs vorsichtig.

In dieser Lektion lernen wir, einen Stromkreis zu schweißen, der den Summer per Knopfdruck steuert. Der Summer ertönt, wenn die Taste gedrückt wird.

Diese Schaltung erfordert keine Programmierung und kann beim Einschalten arbeiten. Wenn die Taste nicht gedrückt wird, verbraucht der Stromkreis keinen Strom. Sie können es an einem Fahrrad, einer Schlafzimmertür oder an einem beliebigen Ort installieren.

Erforderliche Teile

1 x Pin Header

1 x Red LED

1 x 220 Ohm Widerstand

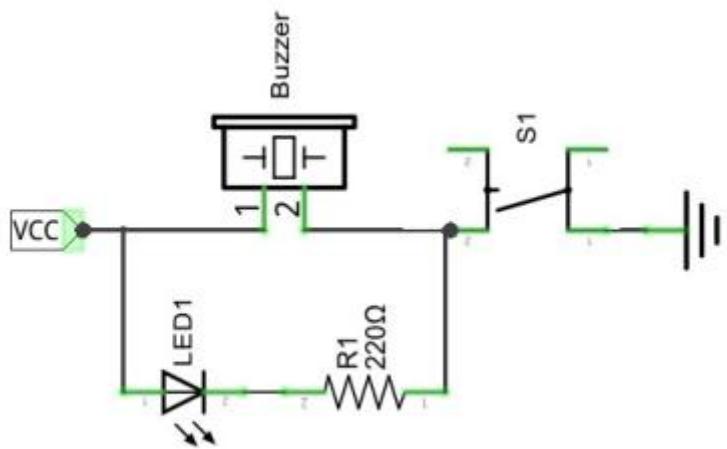
1 x aktiver Summer

1 x Taste

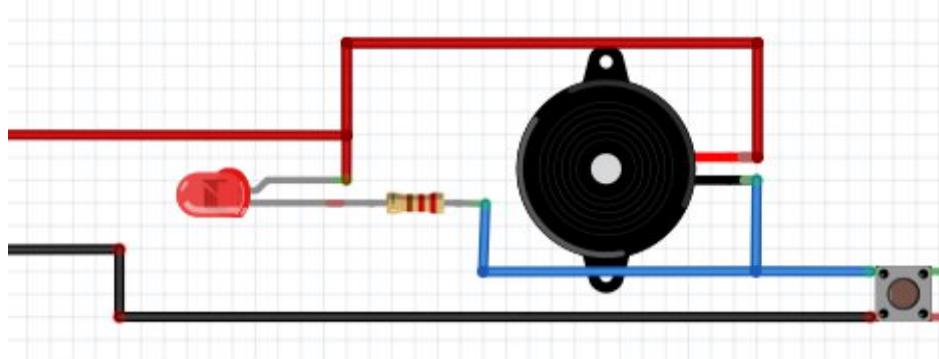
Verbindung Schaltung:

Wir werden die Universalplatine verwenden, um die folgenden Schaltkreise zu löten;

Schaltplan

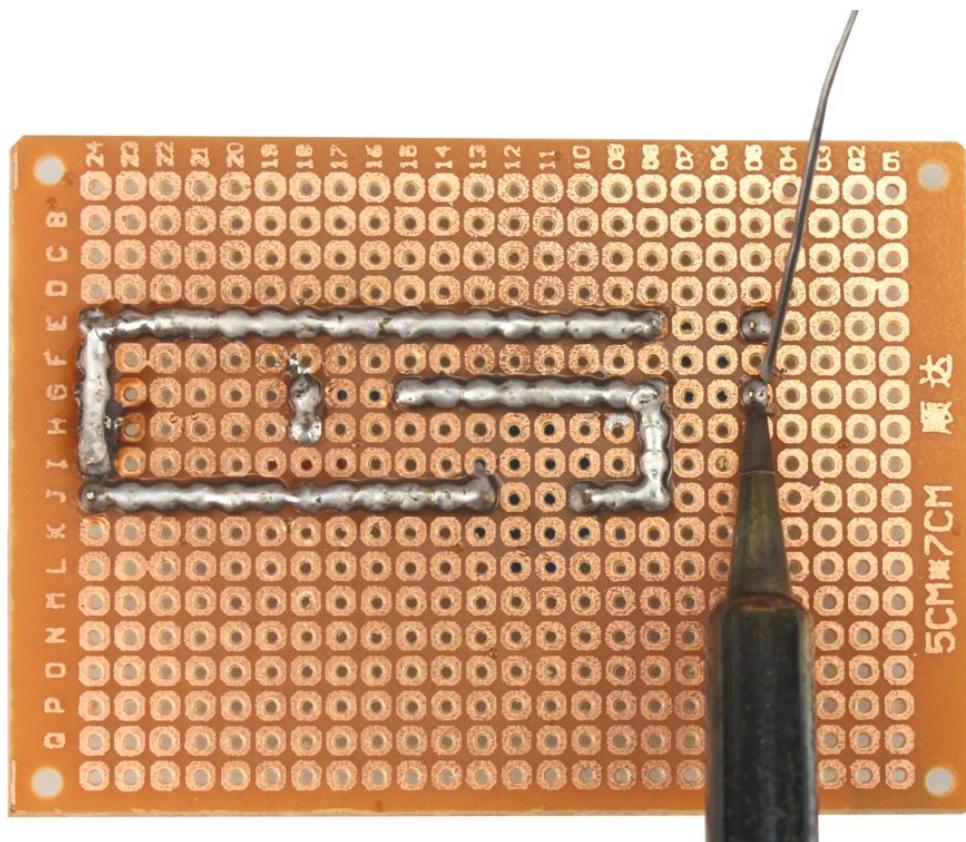


Verdrahtung Diagramm

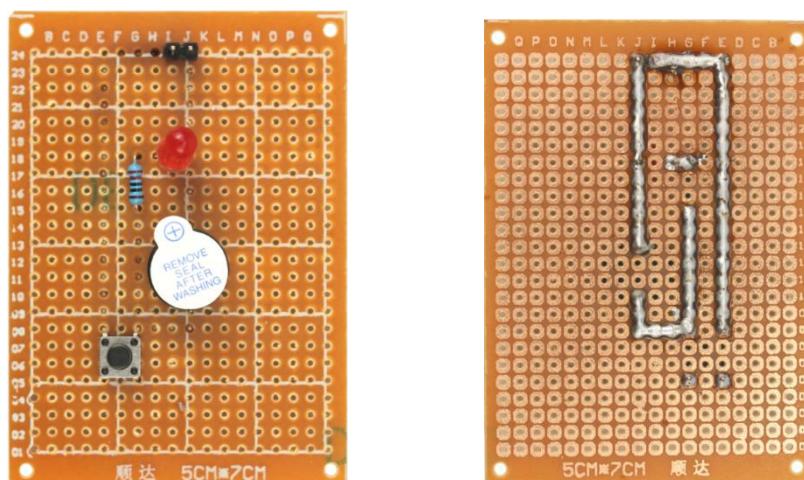


Löten Schaltkreis

Führen Sie die Komponenten in die Universalplatine ein, stecken Sie die Stifte von der Seite ohne Kupfer ein und verlöten Sie die Schaltung mit der Kupferseite, wie nachfolgend dargestellt.

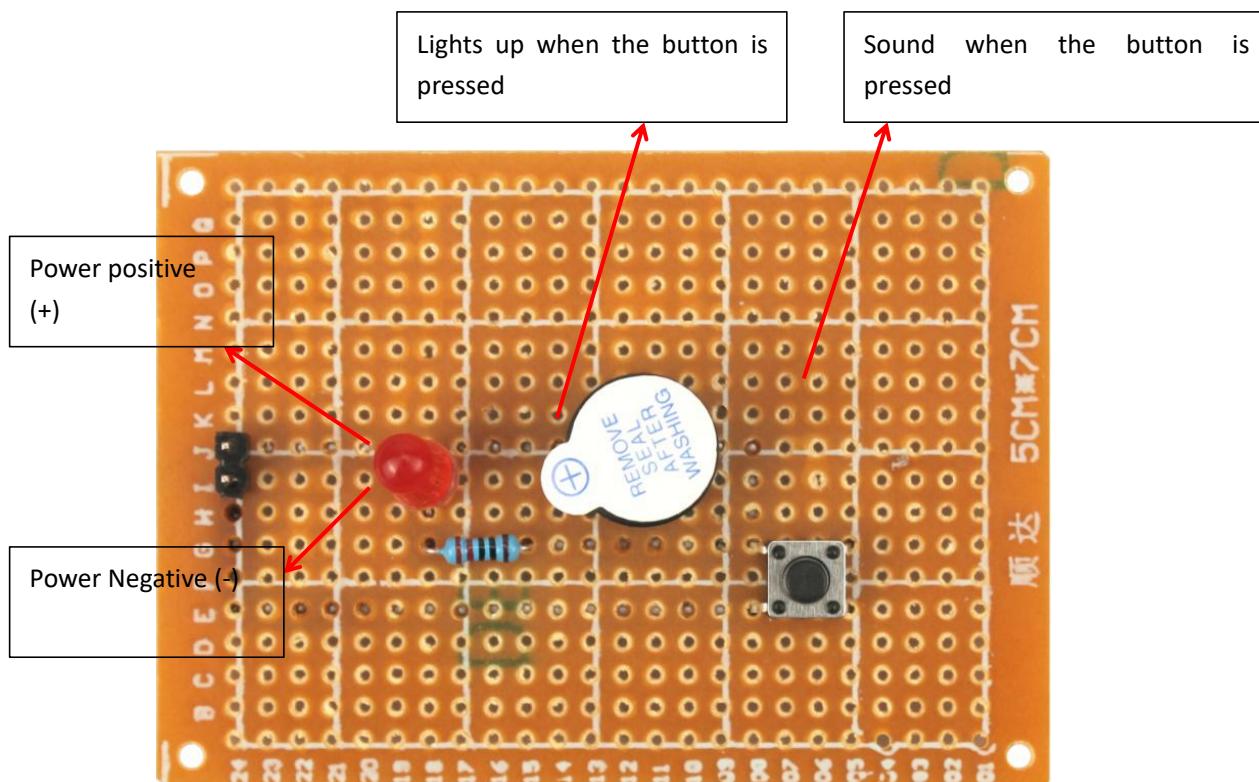


Das folgende ist die Abbildungen, nachdem das Löten beendet ist.



Ist die Schaltung erfolgreich verlötet?

Verbinden Sie die beiden Stifte des Stromkreises (3.3V~5V) mit der Stromversorgung. Das Netzteil kann über eine Batterie, ein Arduino-Entwicklungsboard, einen Raspberry Pi usw. mit Strom versorgt werden. Drücken Sie nach dem Anschließen an die Stromversorgung die Taste, die Betriebsanzeige-LED leuchtet auf und der Summer ertönt es wird fehlschlagen und Sie müssen die Schaltung erneut überprüfen.



Lektion 33 Fließendes Wasserlicht Schweißen

Überblick

In dieser Lektion lernen wir das Löten der blinkenden und laufenden LEDs. Da es 8 LEDs gibt, sollten wir 8 Arduino- oder Raspberry Pi I/O Ports verwenden, die zu viele I/O Ports belegen. Daher verwenden wir den 74HC595-Chip, um die I/O Ports zu erweitern.

Erforderliche Teile

7 x Header

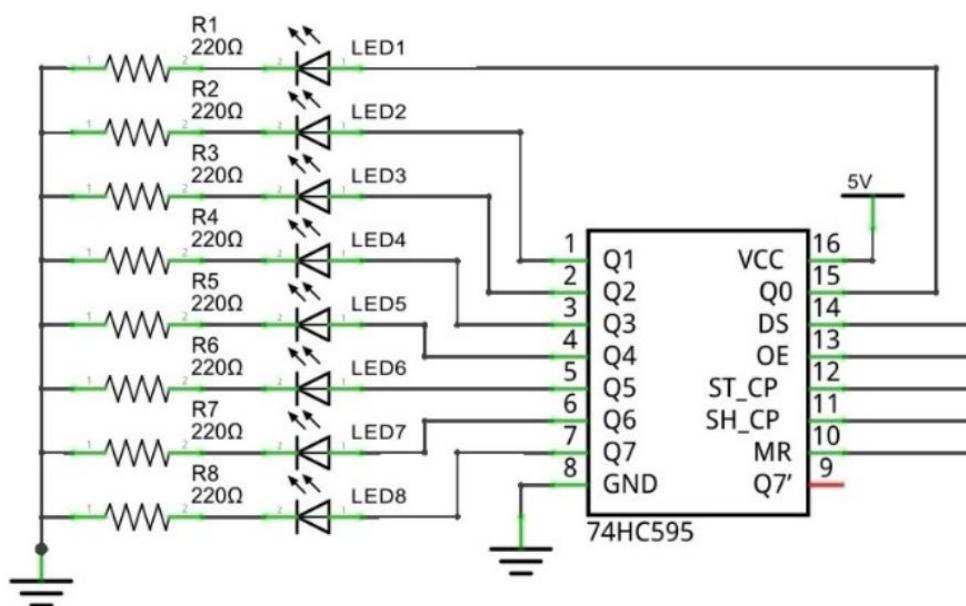
8 x 220 Ohm Widerstände

8 x LED

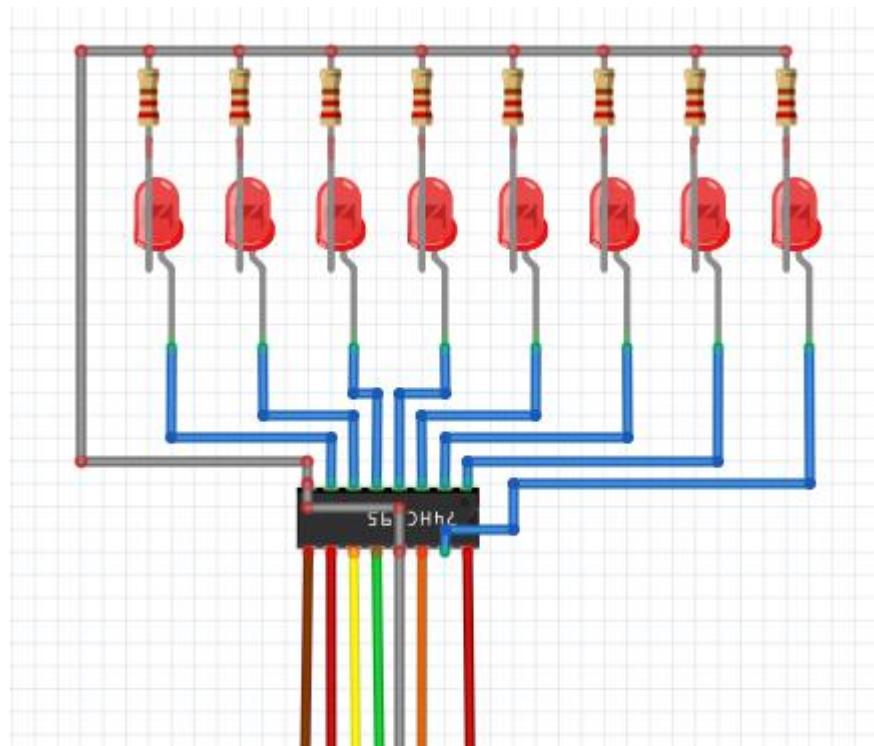
1 x 74HC595

Wie man es lötet?

Schaltplan

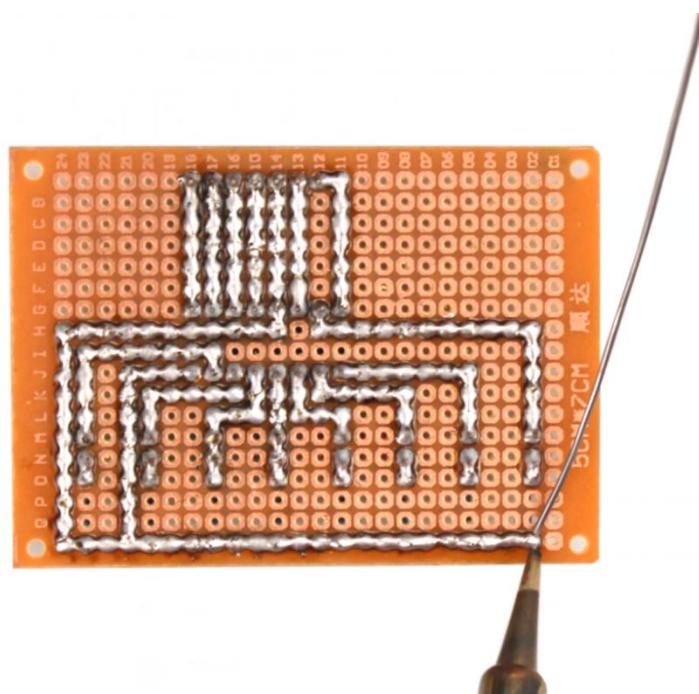


Verdrahtung Diagramm

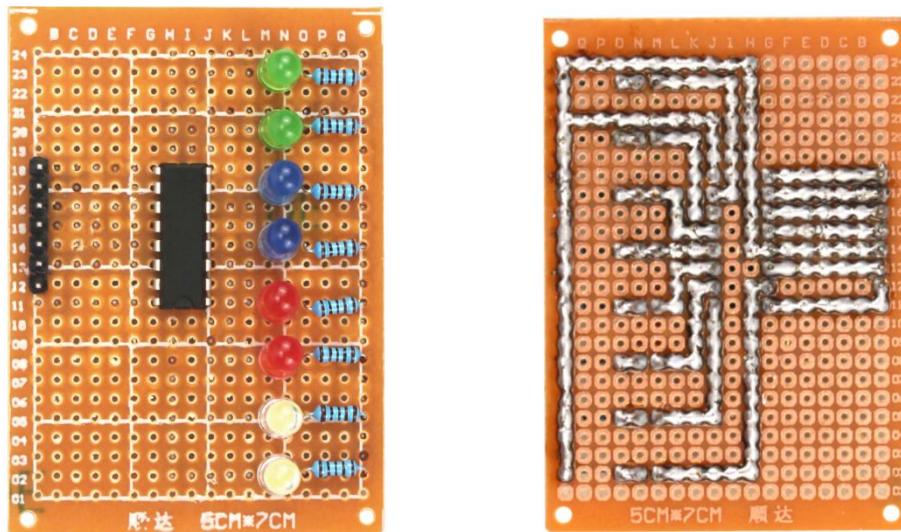


Welding Circuit

Insert the component into the universal board, insert the pins from the side without copper clad, and solder the circuit to the side with copper clad. As shown below.



Das folgende ist die Abbildungen, nachdem das Löten beendet ist.



Wie man es testet, ob die Schaltung erfolgreich verlötet wurde?

Verbinden Sie die Platine mit dem Raspberry Pi. Ausführliche Informationen zum Anschluss und zur Vorgehensweise finden Sie im Kurs „74HC595 und LED“.

