

Vorwort

Über SunFounder

SunFounder ist ein Unternehmen, das sich auf STEAM-Ausbildung mit Produkten konzentriert, wie Open-Source-Robotern, Entwicklungsboards, STEAM-Kits, Modulen, Tools und anderen intelligenten Geräten, die weltweit vertrieben werden. In SunFounder bemühen wir uns, Grund- und Mittelschülern sowie Hobbyisten durch STEAM-Ausbildung dabei zu helfen, die praktischen Praktiken und Fähigkeiten zur Problemlösung zu stärken. Auf diese Weise hoffen wir, Wissen zu verbreiten und Fertigkeitstraining voller Freude anzubieten, wodurch Ihr Interesse am Programmieren und Erstellen zu fördern und Sie einer faszinierenden Welt von Wissenschaft und Technik auszusetzen. Um die Zukunft der künstlichen Intelligenz zu erfassen, ist es dringend und sinnvoll, reichlich STEAM-Wissen zu erlernen.

Über das Da Vinci Kit

Dieses Da Vinci-Kit gilt für den Raspberry Pi 4 Modell B, 3 Modell A +, 3 Modell B +, 3 Modell B, 2 Modell B, 1 Modell B +, 1 Modell A +, zero W and zero. Es enthält verschiedene Komponenten und Chips, die dazu beitragen können, verschiedene interessante Phänomene zu erzeugen, die Sie durch einfache Operationen unter Anleitung von Versuchsanweisungen erhalten können. In diesem Prozess können Sie einige Grundkenntnisse über Programmieren erlernen. Sie können auch mehrere Anwendungen selbst erkunden. Jetzt mach es!

Kostenlose Unterstützung



Bei Fragen senden Sie bitte eine E-Mail an service@sunfounder.com.

Inhalte

Komponentenliste.....	1
Einführung.....	5
Was brauchen wir?.....	6
Erforderliche Komponenten.....	6
Vorbereitung.....	8
Installieren des Betriebssystems.....	8
Richten Sie Ihren Raspberry Pi ein.....	13
Wenn Sie einen Bildschirm haben.....	13
Wenn Sie keinen Bildschirm haben.....	14
Die IP-Adresse bekommen.....	14
Verwenden Sie die SSH-Fernbedienung.....	14
Für Linux- oder Mac OS X-Benutzer.....	14
Für Windows-Benutzer.....	16
Bibliotheken.....	19
RPi.GPIO.....	19
WiringPi.....	20
GPIO-Erweiterungskarte.....	22
Laden Sie den Code herunter.....	24
1 Ausgabe.....	25
1.1 Anzeigen.....	25
1.1.1 Blinkende LED.....	25
1.1.2 RGB-LED.....	38
1.1.3 LED-Balkendiagramm.....	49

1.1.4 7-Segment-Anzeige.....	58
1.1.5 4-stellige 7-Segment-Anzeige.....	69
1.1.6 LED-Punktmatrix.....	83
1.1.7 I2C LCD1602.....	96
1.2 Ton.....	103
1.2.1 Aktiver Summer.....	103
1.2.2 Passiver Summer.....	110
1.3 Treiber.....	119
1.3.1 Motor.....	119
1.3.2 Servo.....	130
1.3.3 Schrittmotor.....	139
1.3.4 Relais.....	153
2 Eingabe.....	161
2.1 Steuerungen.....	161
2.1.1 Taste.....	161
2.1.2 Schiebeschalter.....	169
2.1.3 Neigungsschalter.....	177
2.1.4 Potentiometer.....	185
2.1.5 Tastatur.....	200
2.1.6 Joystick.....	215
2.2 Sensoren.....	224
2.2.1 Fotowiderstand.....	224
2.2.2 Thermistor.....	231
2.2.3 DHT-11.....	241
2.2.4 PIR.....	254

2.2.5 Ultraschallsensormodul.....	265
2.2.6 MPU6050-Modul.....	275
2.2.7 MFRC522 RFID-Modul.....	288
3 Erweiterung.....	294
3.1 Anwendung.....	294
3.1.1 Zählgerät.....	294
3.1.2 Willkommen.....	301
3.1.3 Alarm umkehren.....	308
3.1.4 Smart Fan.....	320
3.1.5 Batterieanzeige.....	327
3.1.6 Bewegungssteuerung.....	333
3.1.7 Ampel.....	339
3.1.8 Überhitzungsmonitor.....	347
3.1.9 Passwortsperre.....	356
3.1.10 Alarmglocke.....	363
3.1.11 Morsekode-Generator.....	372
3.1.12 SPIEL – Nummer Vermutung.....	380
3.1.13 SPIEL - 10 Sekunden.....	391
3.1.14 SPIEL - Nicht nicht.....	397
3.2 Anhang.....	409
3.2.1 I2C-Konfiguration.....	409
3.2.2 SPI-Konfiguration.....	412
3.2.3 Remotedesktop.....	414
VNC.....	414
XRDP.....	418

Komponentenliste

Resistor (220R)

20 pcs



Resistor (1K)

10 pcs



Resistor (10K)

10 pcs



1N4007 Diode

2 pcs



Zener Diode

2 pcs



Green LED

4 pcs



Red LED

10 pcs



Yellow LED

4 pcs



Blue LED

4 pcs



RGB LED

1 pcs



S8050 Transistor

4 pcs



S8550 Transistor

4 pcs



Capacitor 0.1 uF

4 pcs



Capacitor 10uF

4 pcs



Photoresistor

1 pcs



Thermistor

1 pcs



L293D

1 pcs



ADC0834

1 pcs



74HC595

2 pcs



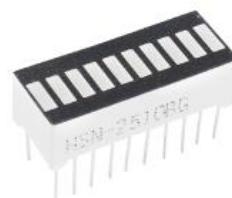
7-segment Display

1 pcs



LED Bar Graph

1 pcs



LED Matrix

1 pcs



Active Buzzer

2 pcs



4-Digit 7-segment Display

1 pcs



Tilt Switch

1 pcs



Potentiometer

3 pcs



Passive Buzzer

1 pcs



Button

4 pcs



Motor

1 pcs



Slide Switch

2 pcs



Keypad

1 pcs



I2C LCD 1602

1 pcs



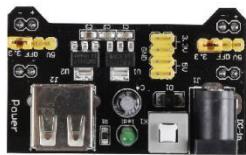
9G Servo

1 pcs



Breadboard Power Module

1 pcs



MFRC522 RFID Module

1 pcs



Relay

1 pcs



Stepping Motor Driver

1 pcs



Stepping Motor

1 pcs



Ultrasonic Ranging Module

1 pcs



Joystick

1 pcs



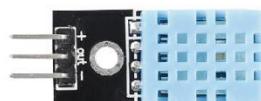
Infrare Motion Sensor

1 pcs



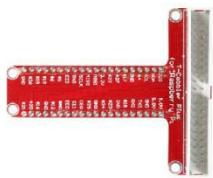
DHT-11

1 pcs



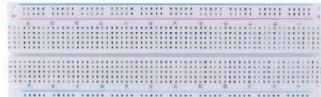
T-shape Extension Board

1 pcs



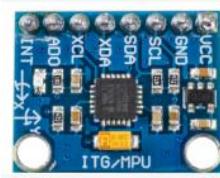
Breadboard

1 pcs



MPU6050 Module

1 pcs



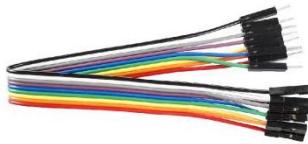
40 Pin GPIO Cable

1 pcs



Jump Wire F/M

10 pcs



9V Battery Cable

1 pcs



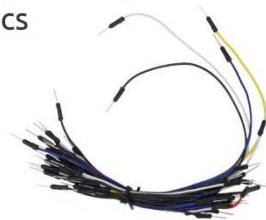
Jump Wire F/F

10 pcs



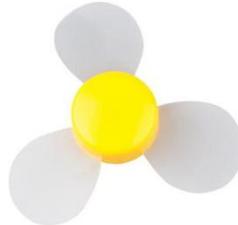
Jump Wire M/M

65 pcs



Fan

1 pcs



Hinweis: Überprüfen Sie bitte nach dem Öffnen der Verpackung, ob Komponentenanzahl der Produktbeschreibung entspricht und ob alle Komponenten in gutem Zustand sind.

Einführung

Da Vinci Kit ist ein Basis-Kit für intelligente Anfänger mit Projektszeitplan. Es enthält 26 häufig verwendete Eingangs- und Ausgangskomponenten und -module sowie eine Reihe grundlegender elektronischer Geräte (wie Widerstände, Kondensatoren), die Sie beim Programmierenlernen unterstützen können.

Nach dem Kit können Sie einige Grundkenntnisse über Raspberry Pi erlernen, einschließlich Installationsmethode von Raspberry Pi, Kenntnisse über Bash Shell und GPIO. Nachdem Sie dieses Wissen verstanden haben, können Sie mit der Programmierung beginnen.

Wenn Sie keinen Hardware Hintergrund Kenntnisse haben, bietet Ihnen dieses Kit Dokument 30 Lektionen zum Nachführen und Lernen, einschließlich 26 grundlegende I/o-Lektionen und 4 einfache praktische Beispiele. Beachten Sie bitte darauf, die Anordnung dieser Kurse basiert nicht auf Schwierigkeitsgrad, sondern Funktionen in der Praxis. Entsprechende Kurse finden Sie nach Ihren Bedürfnissen. Mit anderen Worten, selbst wenn Sie noch keinen vollständigen Kurs absolviert oder die Verwendung dieser Komponenten gemeistert haben, wird dieses Dokument eine wichtige Rolle spielen, um Sie bei der Durchführung praktischer Projekte in der Zukunft zu unterstützen.

Wir freuen uns auf Ihre Projekte und hoffen, dass Sie Ihre Erfolge oder Kreationen in unserem Forum teilen können, während Sie dieses Dokument lesen.

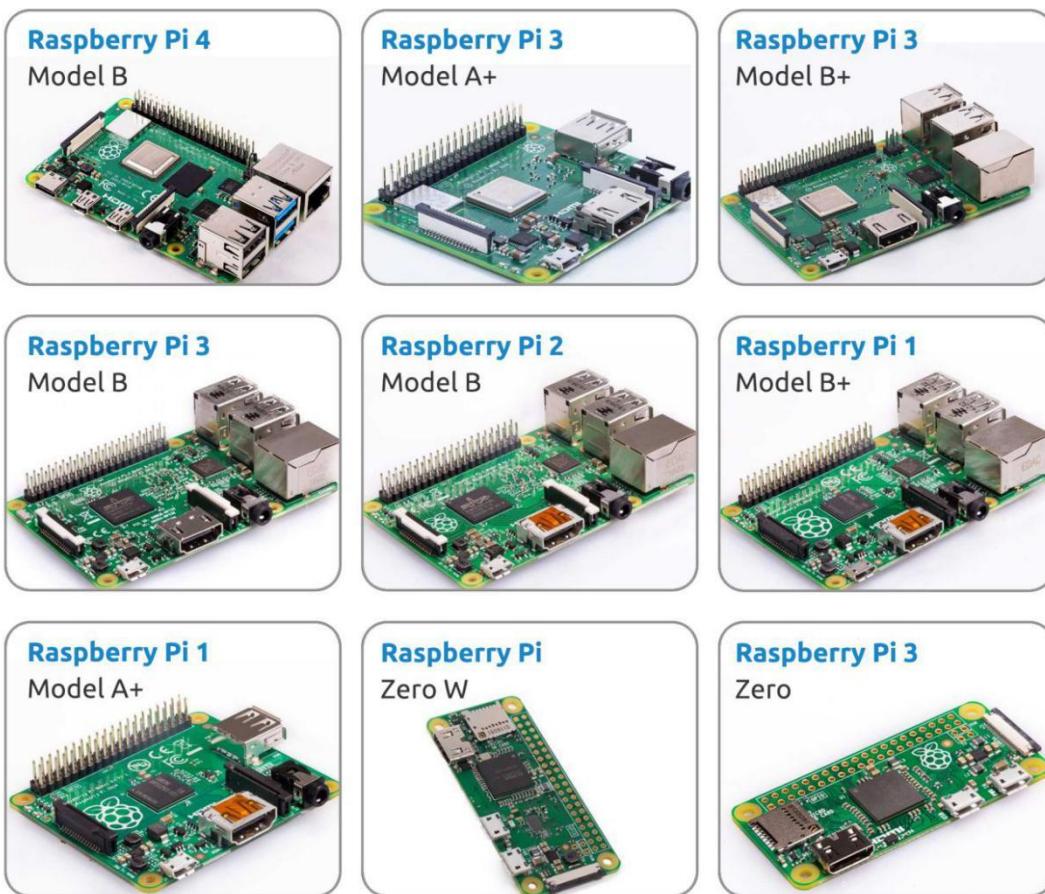
Was brauchen wir?

Erforderliche Komponenten

Raspberry -Pi

Der Raspberry Pi ist ein kostengünstiger Computer in Kreditkartengröße, der an einen Computermonitor oder Fernseher angeschlossen wird und eine Standard Tastatur und Maus verwendet. Es ist ein leistungsfähiges kleines Gerät, das Menschen mit allen Altern ermöglicht, Rechnung zu erforschen wie man in Sprachen wie Scratch und Python programmiert.

Unser Kit gilt für die folgenden Versionen des Produkts von Raspberry Pi:



Netzteil

Für den Anschluss an eine Steckdose verfügt der Raspberry Pi über einen Micro-USB-Anschluss (gleich wie bei vielen Mobiltelefonen). Sie benötigen eine Stromversorgung mit mindestens 2,5 Ampere.

Micro SD Karte

Ihr Raspberry Pi benötigt eine Micro-SD-Karte, um alle Dateien und das Raspberry Pi-Betriebssystem zu speichern. Sie benötigen eine Micro-SD-Karte mit einer Kapazität von mindestens 8 GB

Optionale Komponenten

Bildschirm

Um die Bildschirm Umgebung von Raspberry Pi anzuzeigen, brauchen Sie einen Bildschirm, es kann ein Fernsehbildschirm oder ein Computermonitor sein. Wenn der Bildschirm hat eingebaute Lautsprecher, spielt der Pi über den Ton damit ab.

Maus & Tastatur

Wenn Sie einen Bildschirm verwenden, werden auch eine USB-Tastatur und eine USB-Maus benötigt.

HDMI

Der Raspberry Pi verfügt über einen HDMI-Ausgangsanschluss, der mit HDMI-Anschläßen von meisten modernen Fernseh- und Computermonitoren kompatibel ist. Wenn Ihr Bildschirm nur über DVI- oder VGA-Anschlüsse verfügt, müssen Sie die entsprechende Konvertierungsleitung verwenden.

Tasche

Sie können den Raspberry Pi in eine Tasche stecken; Auf diese Weise können Sie Ihr Gerät schützen.

Ton oder Kopfhörer

Der Raspberry Pi ist mit einem Audioanschluss von ca. 3,5 mm ausgestattet, Sie können den verwenden, wenn Ihr Bildschirm keine eingebauten Lautsprecher hat oder wenn es keinen Bildschirmsoperation gibt.

Vorbereitung

In diesem Kapitel lernen wir zunächst den Raspberry Pi. Der Inhalt umfasst die Installation des Betriebssystems und Raspberry Pi-Netzwerks und das Öffnen des Terminals.

Hinweis: Das vollständige Tutorial bitte finden Sie auf der offiziellen Website des Raspberry Pi: <https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up>

Hinweis: Wenn Ihr Raspberry Pi eingerichtet ist, können Sie den Teil überspringen und mit dem nächsten Kapitel fortfahren.

Installieren des Betriebssystems

Erforderliche Komponenten

Jeglicher Raspberry -Pi	1 * Personal Computer
1 * Micro SD-Karte	

Schritt 1

Raspberry Pi haben ein grafisches SD-Karten-Schreibtool entwickelt, das auf Mac OS, Ubuntu 18.04 und Windows funktioniert und ist die einfachste Option für die meisten Benutzer, da es das Bild herunterladen und automatisch auf der SD-Karte installieren wird.

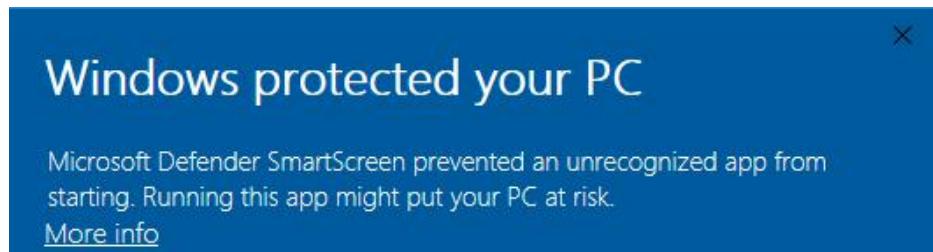
Besuchen Sie die Download-Seite: <https://www.raspberrypi.org/software/>. Klicken Sie auf den Link für den Raspberry Pi Imager, der Ihrem Betriebssystem entspricht. Wenn der Download abgeschlossen ist, klicken Sie darauf und das Installationsprogramm startet.



Schritt 2

Wenn Sie das Installationsprogramm starten, wird Ihr Betriebssystem möglicherweise versuchen, die Ausführung zu verhindern. Unter Windows erhalte ich beispielsweise die folgende Meldung:

Wenn dies angezeigt wird, klicken Sie auf **Weitere Informationen** und **um jeden fall ausführen**, befolgen Sie dann die Anweisungen, um den Raspberry Pi Imager zu installieren.

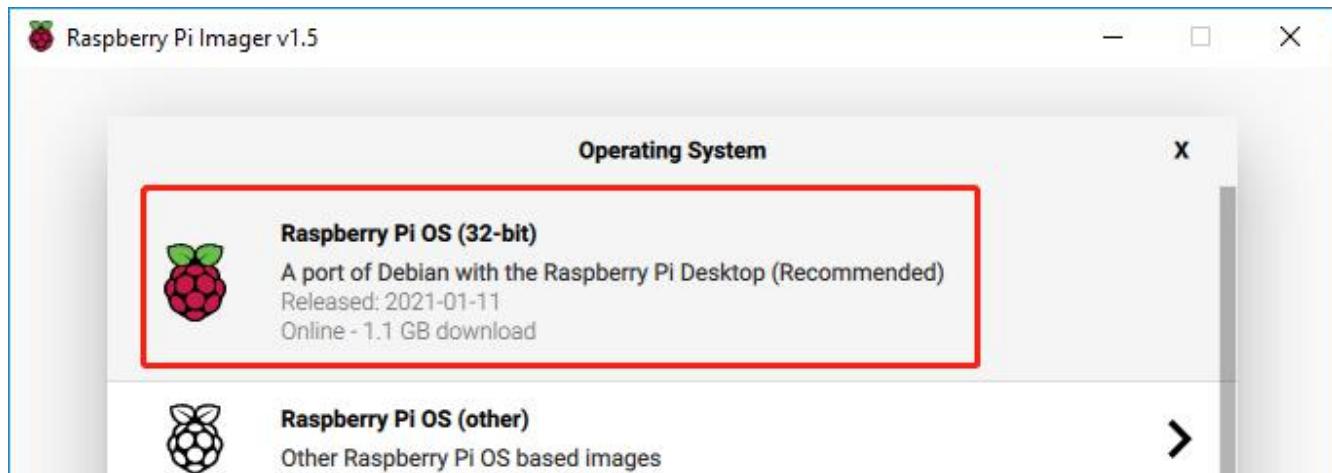


Schritt 3

Legen Sie Ihre SD-Karte in den SD-Kartensteckplatz des Computers oder Laptops ein.

Schritt 4

Wählen Sie im Raspberry Pi Imager das Betriebssystem aus, das Sie installieren möchten, und die SD-Karte, auf der Sie es installieren möchten.



Hinweis:

- 1) Sie müssen zum ersten Mal mit dem Internet verbunden sein.
- 2) Dieses Betriebssystem wird dann für die zukünftige Offline-Verwendung gespeichert (`C:\Users\yourusername\AppData\Local\Raspberry Pi\Imager\cache`, Wenn Sie die Software das nächste Mal öffnen, wird die Anzeige "Freigegeben: Datum, zwischengespeichert auf Ihrem Computer" angezeigt).

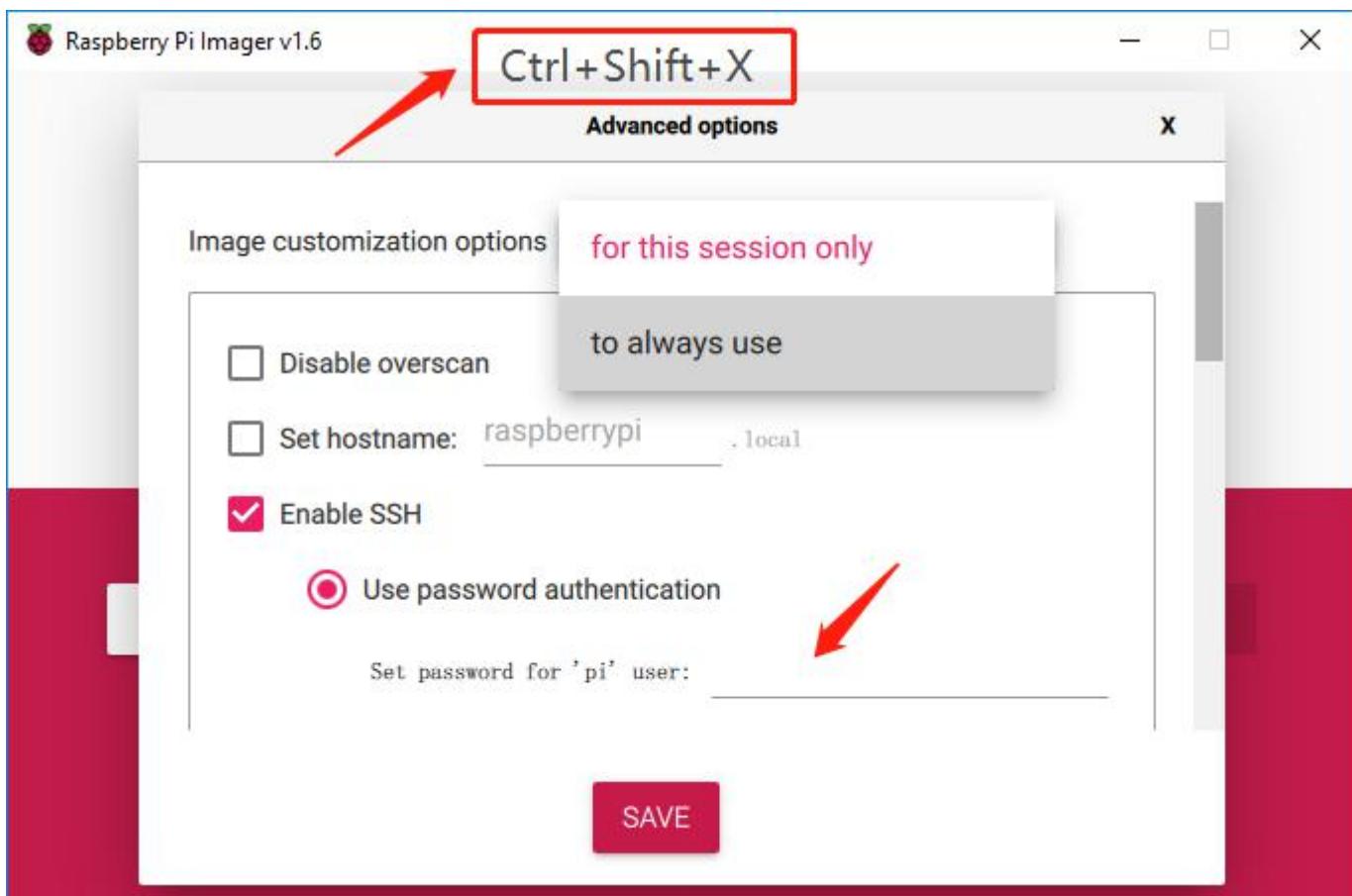
Schritt 5

Wählen Sie die momentan verwendete SD-Karte aus.



Schritt 6

Drücken Sie **Ctrl+Shift+X**, um die Seite Erweiterte Optionen zu öffnen, um SSH zu aktivieren und WLAN zu konfigurieren. Diese beiden Elemente müssen festgelegt werden und die anderen hängen von Ihrer Wahl ab. Sie können diese Bildanpassungsoptionen immer verwenden.

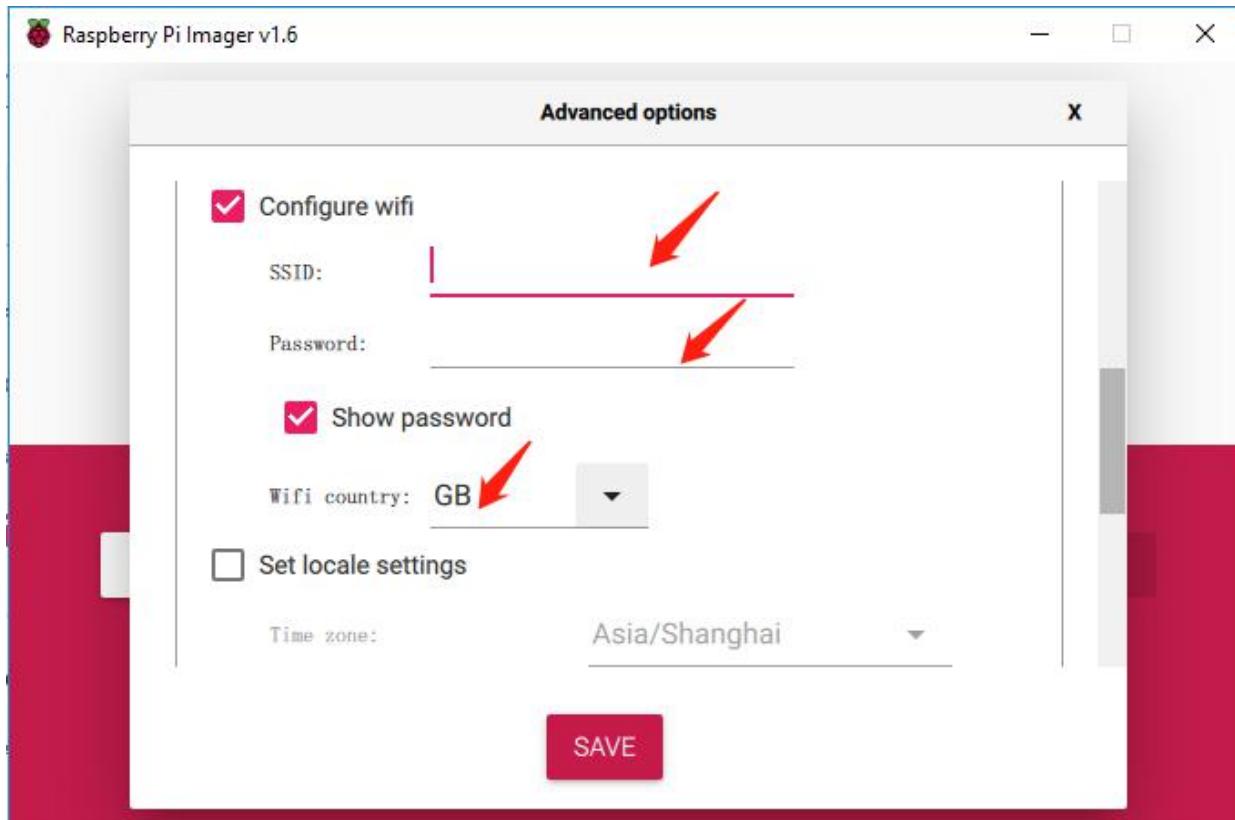


Scrollen Sie dann nach unten, um die WLAN-Konfiguration abzuschließen, und klicken Sie auf **SPEICHERN**.

Hinweis:

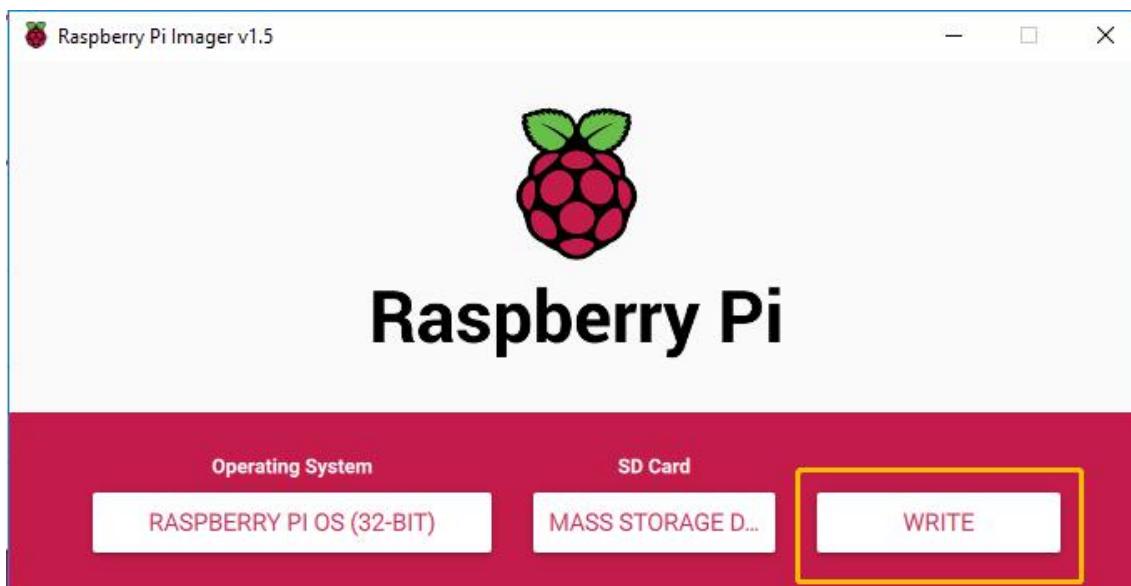
Wlan-Land sollte der zweistellige ISO/IEC alpha2-Kode eingestellt werden, für das Land, in dem Sie Ihren Raspberry Pi verwenden, lesen Sie bitte den folgenden Link:

https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements



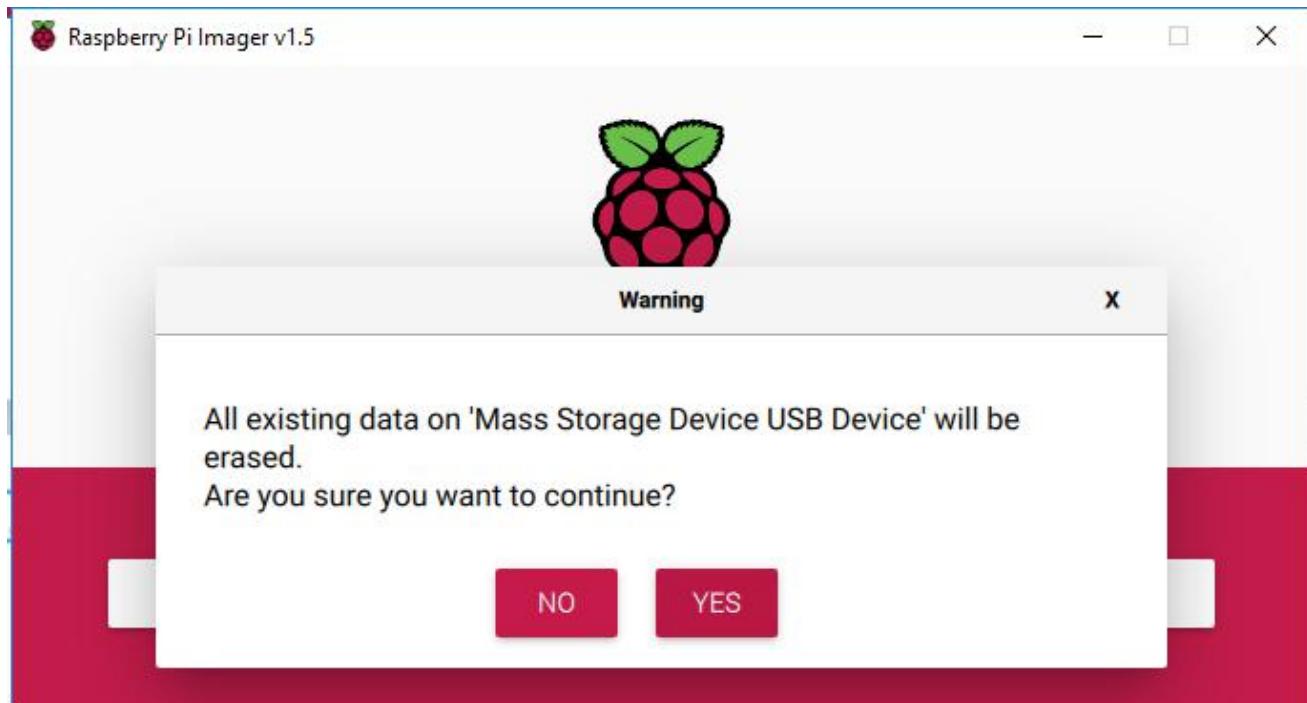
Schritt 7

Klicken Sie auf die Schaltfläche **SCHREIBEN(WRITE)**.



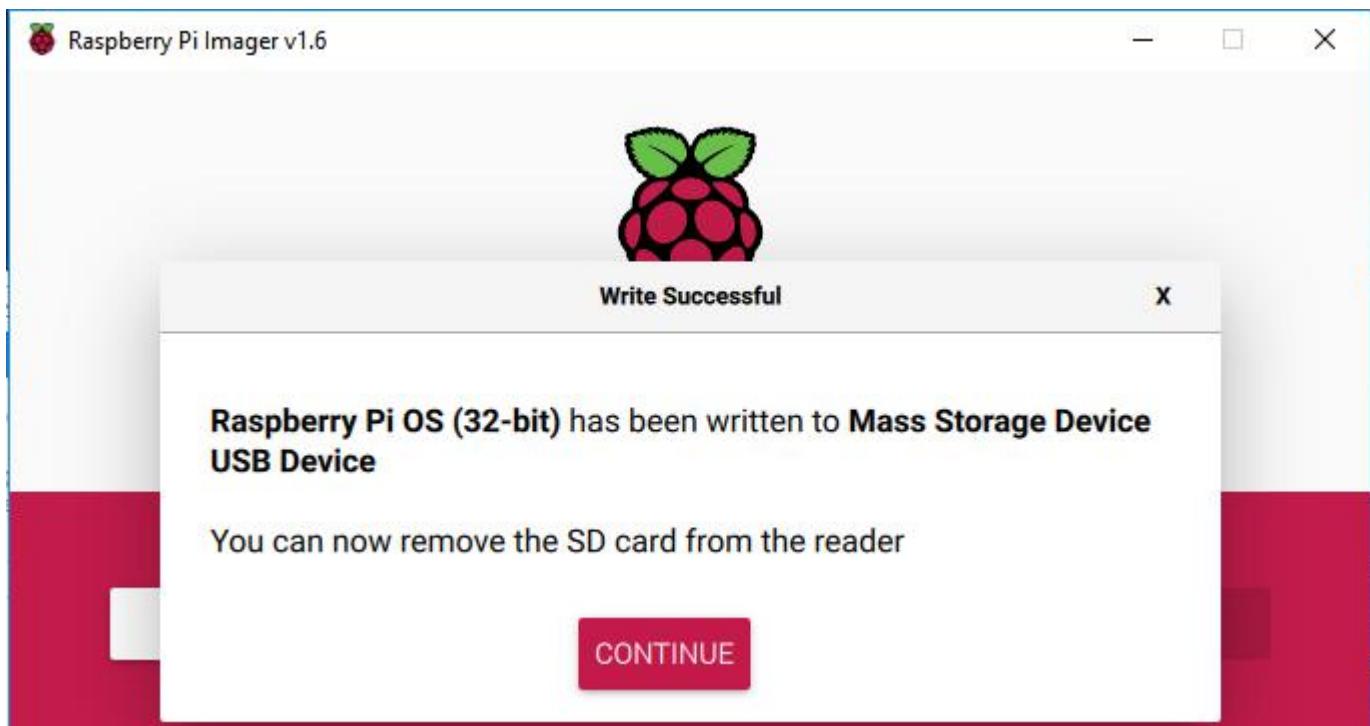
Schritt 8

Wenn auf Ihrer SD-Karte derzeit Dateien gespeichert sind, bitte speichern Sie diese möglicherweise zuerst, um zu verhindern, dass Sie sie dauerhaft verlieren. Wenn keine zu sichernde Datei vorhanden ist, klicken Sie auf **Ja(YES)**.



Schritt 9

Nach einer gewissen Wartezeit wird das folgende Fenster angezeigt und stellt den Abschluss des Schreibens dar.



Richten Sie Ihren Raspberry Pi ein

Wenn Sie einen Bildschirm haben

Wenn Sie einen Bildschirm haben, können Sie den Raspberry Pi problemlos bedienen.

Erforderliche Komponenten

Jeglicher Raspberry -Pi	1 * Netzteil
1 * Micro SD Karte	1 * Bildschirm Netzteil
1 * HDMI Kabel	1 *Bildschirm
1 *Maus	1 * Tastatur

- 1) Legen Sie die mit Raspberry Pi OS eingerichtete SD-Karte in den Micro-SD-Kartensteckplatz an der Unterseite Ihres Raspberry Pi ein.
- 2) Stecken Sie die Maus und Tastatur ein.
- 3) Schließen Sie den Bildschirm an den HDMI-Anschluss von Raspberry Pi an und stellen Sie sicher, dass Ihr Bildschirm an eine Wand Steckdose angeschlossen und eingeschaltet ist.

Hinweis: Wenn Sie einen Raspberry Pi 4 verwenden, müssen Sie den Bildschirm an den HDMI0 anschließen (der dem Stromanschluss am nächsten liegt).

- 4) Verwenden Sie das Netzteil und versorgen den Raspberry Pi mit Strom. Nach einigen Sekunden wird der Raspberry Pi OS-Bildschirm angezeigt.



Wenn Sie keinen Bildschirm haben

Wenn Sie keine Anzeige haben, können Sie sich aus der Ferne beim Raspberry Pi anmelden. Zuvor müssen Sie jedoch die IP-Adresse des Raspberry Pi abrufen.

Die IP-Adresse bekommen

Nachdem der Raspberry Pi mit WIFI verbunden ist, müssen wir die IP-Adresse davon erhalten. Es gibt viele Möglichkeiten, die IP-Adresse zu ermitteln, und zwei davon sind wie folgt aufgeführt.

1. Überprüfung über den Router

Wenn Sie berechtigt sind, sich beim Router anzumelden (z. B. in einem Heimnetzwerk), können Sie die Raspberry Pi zugewiesenen Adressen auf der Administrationsoberfläche des Routers überprüfen.

Der Default Hostname des Raspberry Pi-Betriebssystems ist **raspberrypi**, und bitte Sie ihn finden. (Wenn Sie das ArchLinuxARM-System verwenden, suchen Sie bitte `alarmpi`.)

2. Scannen von Netzwerksegmenten

Sie können auch Netzwerk Scannen verwenden, um die IP-Adresse von Raspberry Pi zu ermitteln. Sie können die Software anwenden, den **erweiterten IP-Scanner** usw.

Scannen Sie den eingestellten IP-Bereich, und alle angeschlossenen Gerätsnamen wird angezeigt. Ebenso, der Default Hostname des Raspberry Pi-Betriebssystems ist **raspberrypi**, wenn Sie ihn nicht geändert haben.

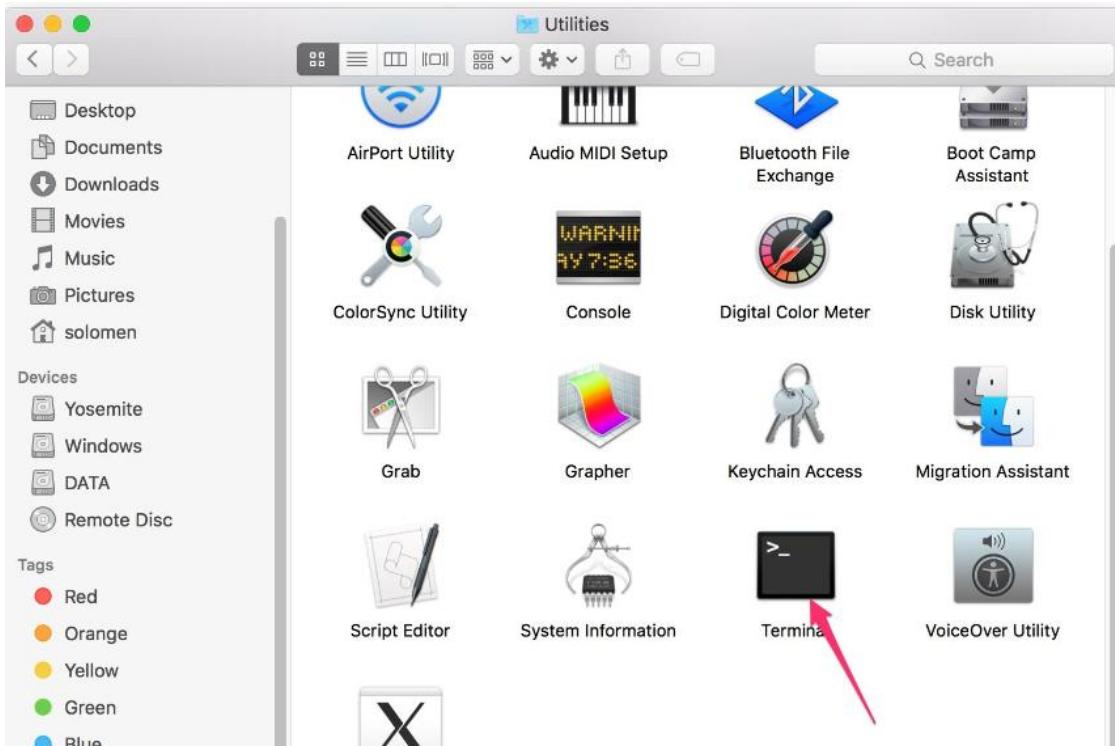
Verwenden Sie die SSH-Fernbedienung

Wir können die Bash Shell von Raspberry Pi öffnen, mit Anwendung von SSH. Bash ist die Default Standard Shell von Linux. Die Shell selbst ist ein in C geschriebenes Programm und stellt die Brücke zwischen den Kunden und Unix / Linux dar. Darüber hinaus kann es helfen, den größten Teil der erforderlichen Arbeit zu erledigen.

➤ Für Linux- oder Mac OS X-Benutzer

Schritt 1

Gehen Sie zu **Anwendungen-> Dienstprogramme**, suchen Sie das **Terminal** und öffnen Sie es.



Schritt 2

Geben Sie **ssh pi @ ip_address** ein. "Pi" ist Ihr Benutzername und "ip_address" ist Ihre IP-Adresse. Z. B. :

```
ssh pi@192.168.18.197
```

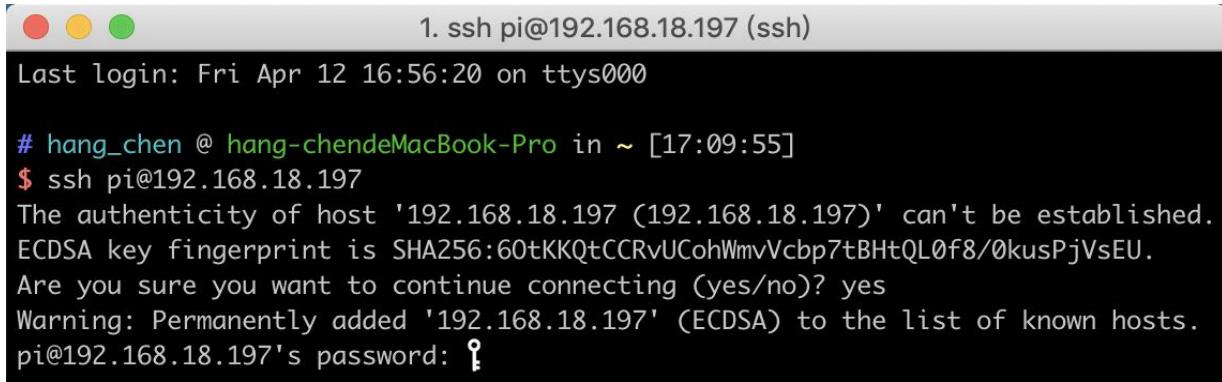
Schritt 3

Geben Sie "Ja" ein.

```
1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000
# hang_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]
$ ssh pi@192.168.18.197
The authenticity of host '192.168.18.197 (192.168.18.197)' can't be established.
ECDSA key fingerprint is SHA256:60tKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)?
```

Schritt 4

Geben Sie den Password ein und das Default Passwort lautet **raspberry**.

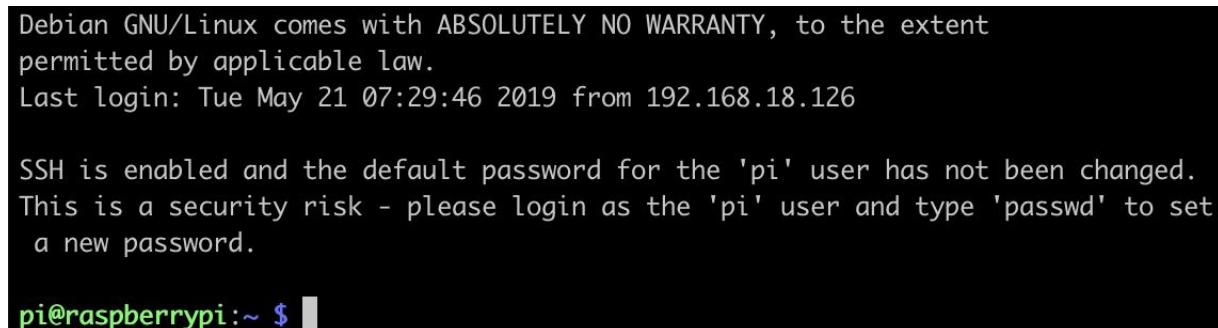


1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000

hang_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]
\$ ssh pi@192.168.18.197
The authenticity of host '192.168.18.197 (192.168.18.197)' can't be established.
ECDSA key fingerprint is SHA256:60tKKQtCCRvUCohWmvVcbp7tBHTQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.18.197' (ECDSA) to the list of known hosts.
pi@192.168.18.197's password: **¶**

Schritt 5

Wir verbinden jetzt den Raspberry Pi und sind bereit, mit dem nächsten Schritt fortzufahren.



Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue May 21 07:29:46 2019 from 192.168.18.126

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ \$

Hinweis: Wenn Sie das Kennwort eingeben, werden die Zeichen im Fenster nicht entsprechend angezeigt, was normal ist. Sie brauchen lediglich das richtige Passwort einzugeben.

➤ Für Windows-Benutzer

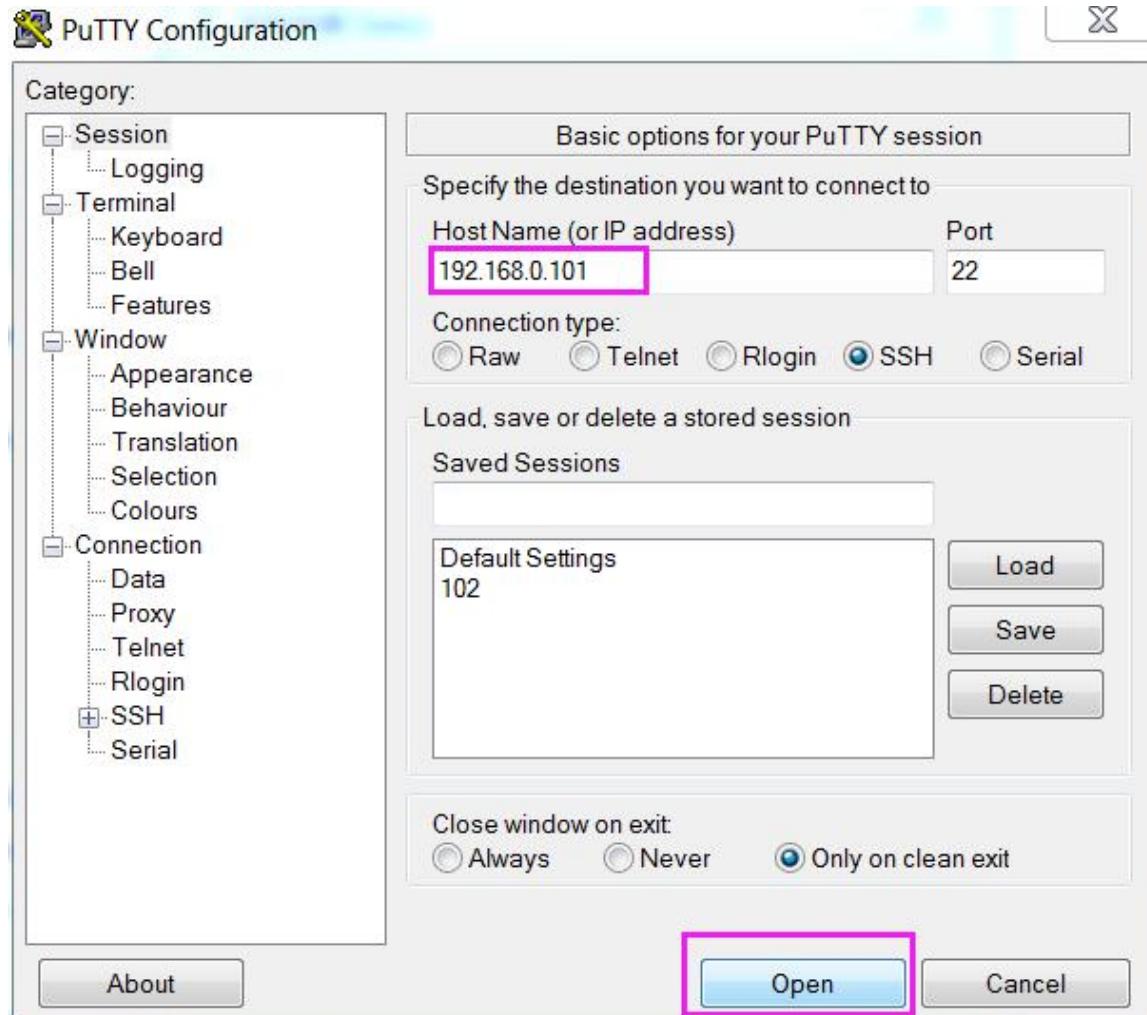
Wenn Sie ein Windows-Benutzer sind, können Sie SSH mit der Anwendung einiger Software verwenden. Hier empfehlen wir **PuTTY**.

Schritt 1

Laden Sie PuTTY herunter.

Schritt 2

Öffnen Sie PuTTY und klicken Sie auf **Sitzung** in der linken baumähnlichen Struktur. Geben Sie die IP-Adresse des RPi in das Textfeld unter **Hostname (oder IP-Adresse)** und **22** unter **Port** ein (Default ist 22).



Schritt 3

Klicken Sie auf **Öffnen**. Beachten Sie bitte, beim ersten Anmelden am Raspberry Pi mit der IP-Adresse eine Sicherheitserinnerung wird anzeigen. Klicken Sie einfach auf **Ja**.

Schritt 4

Wenn PuTTY-Fenster die Meldung "**Anmelden als:**" anzeigt, geben Sie "**pi**" (den Benutzernamen des RPis) und das Passwort "**raspberry**" (die Default Einstellung, wenn Sie diese nicht geändert haben) ein.

```

pi@raspberrypi: ~
login as: pi
pi@192.168.0.234's password: raspberry
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb 21 02:54:55 2017
pi@raspberrypi: ~ $ 
```

Schritt 5

Hier wird der Raspberry Pi angeschlossen und es ist Zeit, die nächsten Schritte durchzuführen.

Hinweis: Wenn Sie das Kennwort eingeben, werden die Zeichen im Fenster nicht entsprechend angezeigt, was normal ist. Sie brauchen lediglich das richtige Passwort einzugeben.

Hinweis: Wenn Sie mit der Verwendung des Befehlsfensters zur Raspberry Pi Steuerung nicht zufrieden sind, können Sie auch die Fernbedingungen verwenden, mit der wir die Dateien im Raspberry Pi einfach verwalten können.

Einzelheiten dazu finden Sie unter [3.2.3 Fern-Bildschirm](#)

Bibliotheken

Bei der Programmierung mit Raspberry Pi werden zwei wichtige Bibliotheken verwendet: WiringPi und RPi.GPIO. Das Raspberry Pi OS installiert sie standardmäßig, Sie können sie direkt verwenden.

RPi.GPIO

Wenn Sie ein Python-Benutzer sind, können Sie GPIOs mit der von RPi.GPIO bereitgestellten API programmieren.

RPi.GPIO ist ein Modul zur Steuerung von Raspberry Pi GPIO-Kanälen. Dieses Paket enthält eine Klasse zur Steuerung des GPIO einer Raspberry Pi. Beispiele und Dokumente finden Sie unter <http://sourceforge.net/p/raspberry-gpio-python/wiki/Home/>

Beim Testen ob RPi.GPIO installiert ist oder nicht, geben Sie bitte Python ein:

Python

```
pi@raspberrypi:~ $ python
Python 2.7.9 (default, Mar  8 2015, 00:52:26)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Geben Sie in Python CLI "import RPi.GPIO" ein. Wenn kein Fehler stellt, bedeutet, RPi.GPIO ist installiert.

RPi.GPIO importieren

```
pi@raspberrypi:~ $ python
Python 2.7.9 (default, Mar  8 2015, 00:52:26)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import RPi.GPIO
>>> 
```

Wenn Sie die Python-CLI beenden möchten, geben Sie ein:

exit()

```
>>> exit()
pi@raspberrypi:~ $ 
```

WiringPi

wiringPi ist eine GPIO-Bibliothek in C-Sprache, die auf die Raspberry Pi-Plattform angewendet wird. Es entspricht GUN Lv3. Die Funktionen in WiringPi sind ähnlich wie im Verkabelungssystem von Arduino. Die mit Arduino vertrauten Benutzer wird die einfachere Verwendung von wiringPi.

WiringPi enthält viele GPIO-Befehle, mit denen Sie alle Arten von Schnittstellen auf Raspberry Pi steuern können. Anhand der folgenden Anweisungen können Sie testen, ob die wiringPi-Bibliothek erfolgreich installiert wurde oder nicht.

```
gpio -v
```

```
pi@raspberrypi:~/davinci-kit-for-raspberry-pi/c/1.1.1 $ 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: 2048MB, Maker: Sony
  * Device tree is enabled.
  *--> Raspberry Pi 4 Model B Rev 1.1
  * This Raspberry Pi supports user-level GPIO access.
```

Hinweis:

Wenn Sie Raspberry Pi 4B verwenden, die GPIO-Version jedoch 2.50 ist, wird nach Ausführung des C-Sprachcodes keine Antwort ausgegeben, die GPIO-Pins funktionieren nicht. Da müssen Sie manuell auf Version 2.52 aktualisieren. Die Aktualisierungsschritte lauten wie folgt:

```
cd /tmp
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
```

Überprüfen Sie mit:

```
gpio -v
```

und sicherstellen, es ist Version 2.52.

gpio readall

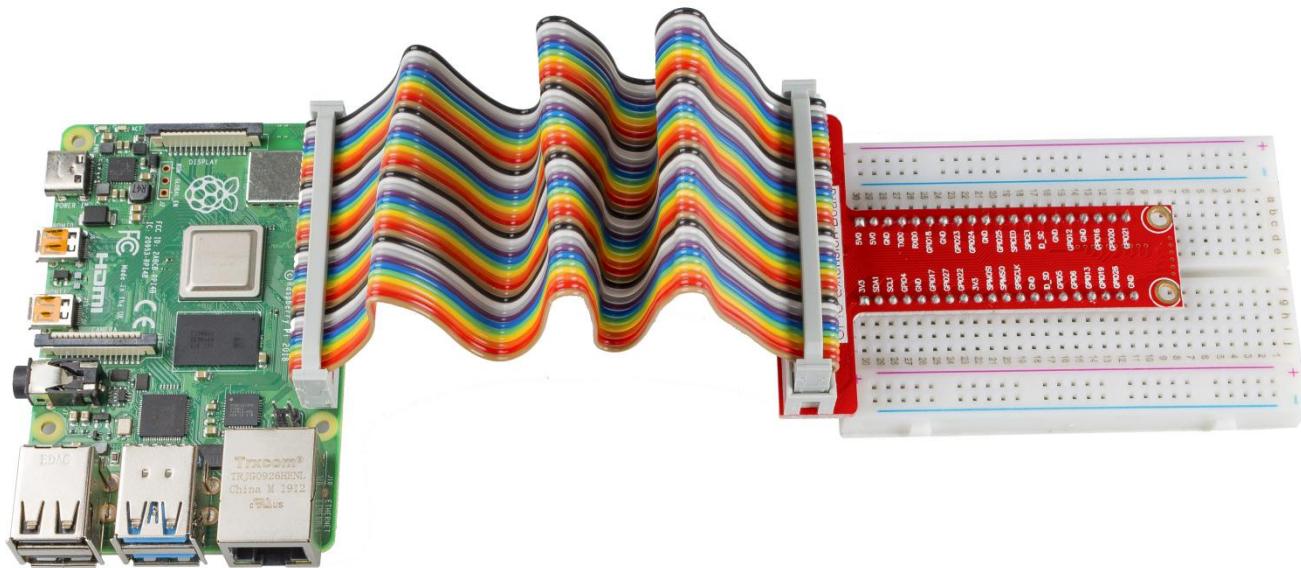
Pi 3												
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM		
		3.3v			1	2		5v				
2	8	SDA.1	ALT0	1	3	4		5V				
3	9	SCL.1	ALT0	1	5	6		0v				
4	7	GPIO. 7	IN	0	7	8	1	IN	TxD	15	14	
		0v			9	10	1	IN	RxD	16	15	
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1	18	
27	2	GPIO. 2	IN	0	13	14		0v				
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	4	23	
		3.3v			17	18	0	IN	GPIO. 5	5	24	
10	12	MOSI	ALT0	1	19	20		0v				
9	13	MISO	ALT0	1	21	22	0	IN	GPIO. 6	6	25	
11	14	SCLK	ALT0	0	23	24	1	OUT	CE0	10	8	
		0v			25	26	1	OUT	CE1	11	7	
0	30	SDA.0	IN	1	27	28	1	OUT	SCL.0	31	1	
5	21	GPIO.21	IN	0	29	30		0v				
6	22	GPIO.22	IN	0	31	32	0	IN	GPIO.26	26	12	
13	23	GPIO.23	IN	1	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	
Pi 3												
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM		

Weitere Informationen zu wiringPi finden Sie unter: <http://wiringpi.com/download-and-install/>

GPIO-Erweiterungskarte

Vor dem Erlernen der Befehle, müssen Sie zunächst mehr über die Pins von Raspberry Pi wissen, der für die nachfolgende Studie wichtige Rolle spielt.

Wir können die Pins Raspberry Pi leicht über das GPIO Erweiterungskarte zum Steckbrett führen, um GPIO-Schäden aus häufiges Ein- oder Ausstecken zu vermeiden. Dies ist unser 40-pin GPIO-Erweiterungskarte und GPIO-Kabel für Raspberry Pi Modell B +, 2 Modell B und 3, 4 Modell B.



Pin Nummer

Die Pins von Raspberry Pi können auf drei Arten benannt werden: WiringPi, BCM und Board. Unter diesen Benennungsmethoden verwendet die 40-pin GPIO-Erweiterungskarte die Benennungsmethode BCM. Für einige spezielle Pins wie den I2C-Port und den SPI-Port verwenden sie jedoch den Namen, der mit ihnen geliefert wird. Die folgende Tabelle zeigt die Benennungsmethoden von WiringPi, Karte und den systeminternen Name jedes Pins auf der GPIO-Erweiterungskarte. Beispielsweise für GPIO17 die Karte ist Benennungsmethode 11, die wiringPi-Benennungsmethode 0 und die intrinsische Benennungsmethode GPIO0.

Hinweis:

- 1) In der Sprache C wird die Benennungsmethode WiringPi verwendet.
- 2) In Python Language werden Karte und BCM als Benennungsmethoden verwendet, und die Funktion GPIO.setmode () wird verwendet, um sie einzustellen.

Name	WiringPi	Karte	BCM		Karte	WiringPi	Name
		GPIO-Erweiterungskarte					
3.3V	3V3	1	3V3	5.0V	2	5.0V	5V
SDA	8	3	SDA	5.0V	4	5.0V	5V
SCL	9	5	SCL	GND	6	GND	0V
GPIO7	7	7	GPIO4	TXD	8	15	TXD
0V	GND	9	GND	RXD	10	16	RXD
GPIO0	0	11	GPIO17	GPIO18	12	1	GPIO1
GPIO2	2	13	GPIO27	GND	14	GND	0V
GPIO3	3	15	GPIO22	GPIO23	16	4	GPIO4
3.3V	3.3V	17	3.3V	GPIO24	18	5	GPIO5
MOSI	12	19	MOSI	GND	20	GND	0V
MISO	13	21	MISO	GPIO25	22	6	GPIO6
SCLK	14	23	SCLK	CE0	24	10	CEO
0V	GND	25	GND	CE1	26	11	CE1
IN_SDA	30	27	EED	EEC	28	31	ID_SCL
GPIO21	21	29	GPIO5	GND	30	GND	0V
GPIO22	22	31	GPIO6	GPIO12	32	26	GPIO26
GPIO23	23	33	GPIO13	GND	34	GND	0V
GPIO24	24	35	GPIO19	GPIO16	36	27	GPIO27
GPIO25	25	37	GPIO26	GPIO20	38	28	GPIO28
0V	GND	39	GND	GPIO21	40	29	GPIO29

Laden Sie den Code herunter

Beachten Sie vor dem Herunterladen des Codes, dass der Beispielcode **NUR** unter Raspberry Pi OS getestet wird. Wir bieten zwei Methoden zum Download an:

Methode 1: git clon verwenden (empfohlen)

Melden Sie sich bei Raspberry Pi an und wechseln Sie dann das Verzeichnis in `/home/pi`.

```
cd /home/pi/
```

Hinweis: cd, um vom aktuellen Pfad in das beabsichtigte Verzeichnis zu wechseln. Informell geht es hier zum Pfad `/home/pi/`.

Klonen Sie das Repository von GitHub

```
git clone https://github.com/sunfounder/davinci-kit-for-raspberry-pi.git
```

Methode 2: Laden Sie den Kode herunter.

Laden Sie den Quellcode von github herunter:

<https://github.com/sunfounder/davinci-kit-for-raspberry-pi>

The screenshot shows the GitHub repository page for 'davinci-kit-for-raspberry-pi'. At the top, there's a header with the repository name, a 'Watch' button (0), a 'Star' button (0), and a 'Fork' button (0). Below the header, there are tabs for 'Code', 'Issues (0)', 'Pull requests (0)', 'Projects (0)', 'Wiki', 'Security', and 'Insights'. A summary bar below the tabs shows: 5 commits, 1 branch, 0 releases, 1 contributor, and GPL-2.0 license. Under the summary bar, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find File', and a prominent green 'Clone or download' button. To the right of the 'Clone or download' button is a 'Clone with HTTPS' section with a URL: <https://github.com/sunfounder/davinci-kit-for-raspberry-pi>. Below this are 'Open in Desktop' and 'Download ZIP' buttons, with 'Download ZIP' being highlighted with a red box. On the left, there's a list of files: 'xiemeiping upload codes' (a folder), 'c' (Upload code), 'python' (Upload code), 'LICENSE' (Upload code), 'README.md' (upload codes), and 'show' (Upload code). The 'README.md' file was uploaded 5 days ago.

1 Ausgabe

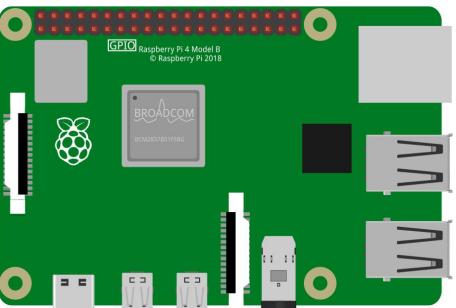
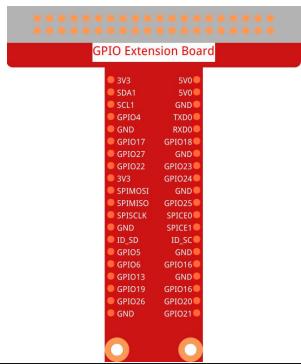
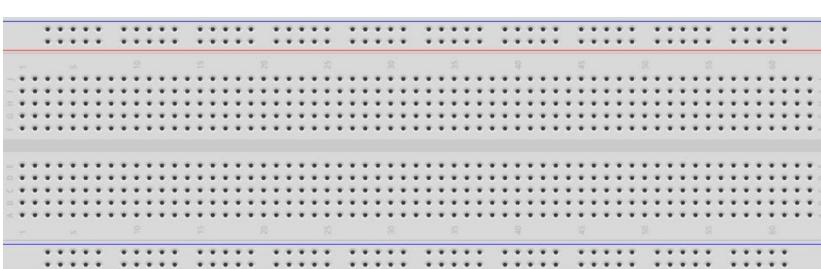
1.1 Anzeigen

1.1.1 Blinkende LED

Einführung

In dieser Lektion lernen wir, wie man durch Programmieren eine blinkende LED erzeugt. Durch Ihre Einstellungen kann Ihre LED eine Reihe interessanter Phänomene erzeugen. Jetzt mach es.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * LED
	 Pinout diagram for the GPIO Extension Board: 2V3, SDA1, SCL1, GND, GPIO24, TXD0, RXD0, GND, GPIO17, GPIO18, GND, GPIO27, GND, GPIO22, GPIO23, V2, GPIO24, GND, SPI_MOSI, GND, SPI_MISO, GPIO25, SPI_SCLK, SPI_CE0, GND, ID_SD, ID_SC, GPIO26, GPIO19, GPIO13, GPIO21, GPIO26, GPIO20, GND, GPIO21	
1 * 40-Pin Kabel	Mehrere Überbrückungsdrähte	
		
1 * Steckbrett	1 * Widerstand (220 Ω)	
		

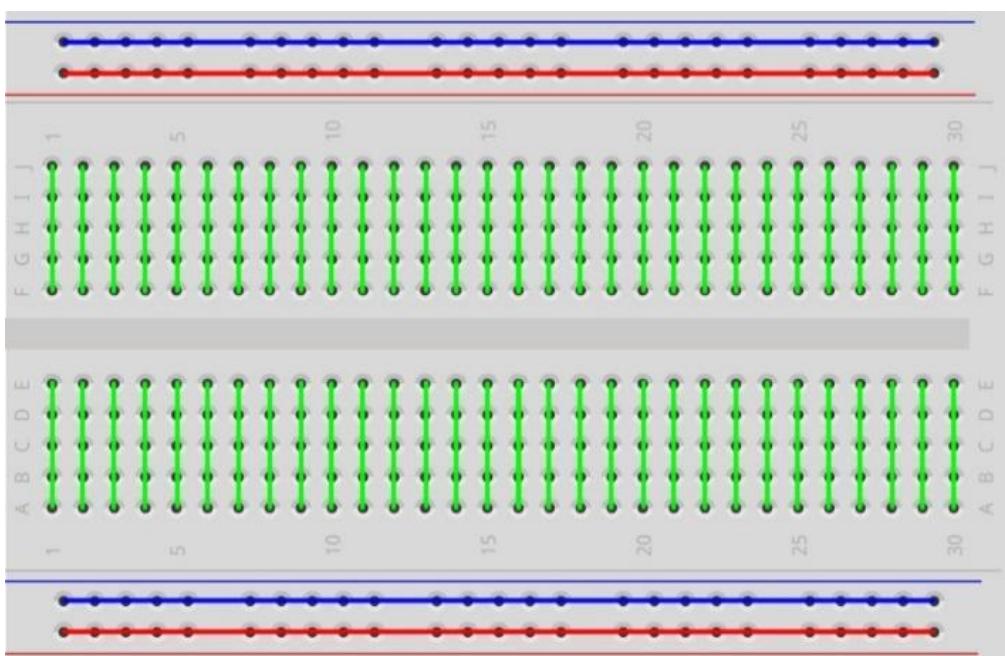
Hinweis: Um reibungslos vorgehen zu können, müssen Sie Ihre eigene Raspberry Pi, TF-Karte und Raspberry Pi-Leistung mitbringen.

Prinzip

Steckbrett

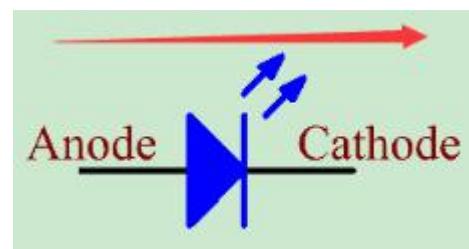
Ein Steckbrett ist eine Konstruktionsbasis für das Prototyping von Elektronik. Es wird verwendet, um Schaltungen schnell aufzubauen und zu testen, bevor ein Schaltungsentwurf abgeschlossen wird. Und es hat viele Löcher, in die oben erwähnte Komponenten wie ICs und Widerstände sowie Jumperdrähte eingesetzt werden können. Mit dem Steckbrett können Sie Komponenten einfach anschließen und entfernen.

Das Bild zeigt die interne Struktur eines Voll+ Steckbretts. Obwohl diese Löcher auf dem Steckbrett unabhängig voneinander zu sein scheinen, sind sie tatsächlich intern über Metallstreifen miteinander verbunden.



LED

LED ist eine Art Diode. Die LED leuchtet nur, wenn der lange Stift der LED mit der positiven Elektrode und der kurze Stift mit der negativen Elektrode verbunden ist.

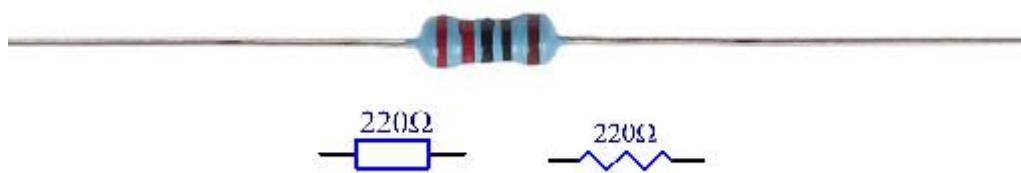


Die LED kann nicht direkt an die Stromversorgung angeschlossen werden, die Komponente kann daran beschädigt werden. Ein Widerstand mit $160\ \Omega$ oder mehr (Arbeit in 5V) muss in der LED-Schaltung in Reihe geschaltet werden.

Widerstand

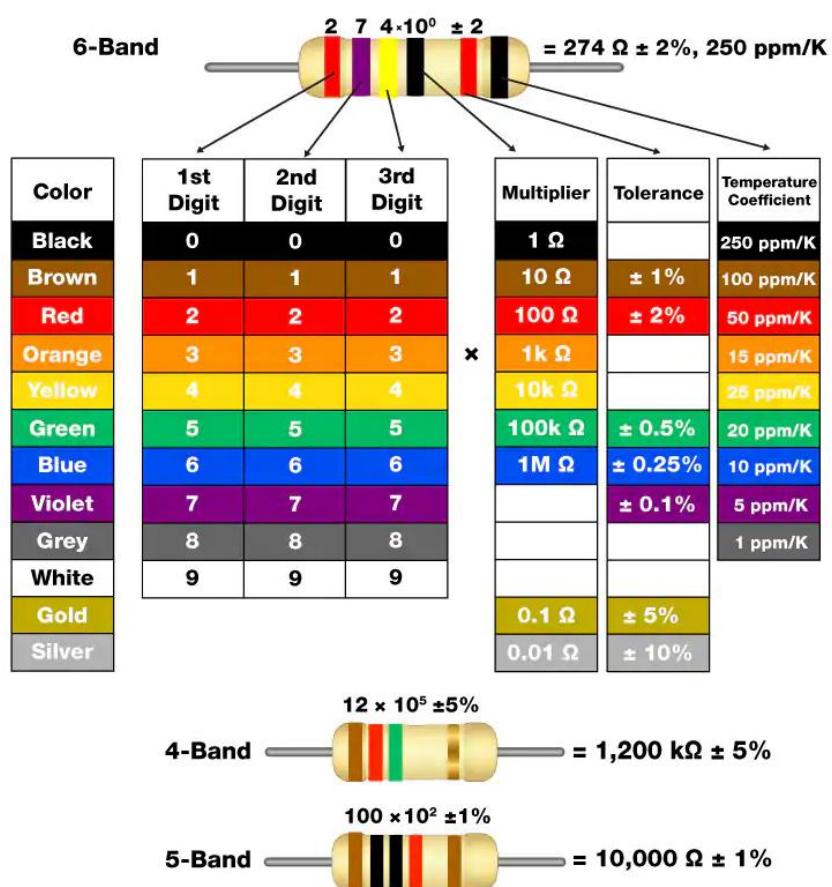
Der Widerstand ist ein elektronisches Element, das den Zweigstrom begrenzen kann. Ein fester Widerstand ist ein Typ von Widerstand, dessen Widerstand nicht geändert werden kann, während der eines Potentiometers oder eines variablen Widerstands eingestellt werden kann.

In diesem Kit wird ein fester Widerstand angewendet. In der Schaltung ist es wichtig, die angeschlossenen Komponenten zu schützen. Die folgenden Bilder zeigen ein reales Objekt, einen $220\ \Omega$ -Widerstand und zwei allgemein verwendete Schaltungssymbole des Widerstands. Ω ist die Widerstandseinheit und die größeren Einheiten umfassen $K\Omega$, $M\Omega$ usw. Ihre Beziehung kann wie folgt gezeigt werden: $1\ M\ \Omega = 1000\ K\Omega$, $1\ K\Omega = 1000\ \Omega$. Normalerweise ist der Widerstandswert darauf markiert. Wenn Sie diese Symbole in einer Schaltung sehen, bedeutet es gibt ein Widerstand.



Wenn wir einen Widerstand verwenden, müssen wir zuerst seinen Widerstand kennen. Hier sind zwei Methoden: Sie können die Bänder am Widerstand beobachten oder den Widerstand mit einem Multimeter messen. Es wird empfohlen, die erste Methode zu verwenden, da diese bequemer und schneller ist. Verwenden Sie ein Multimeter um den Wert zu messen.

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



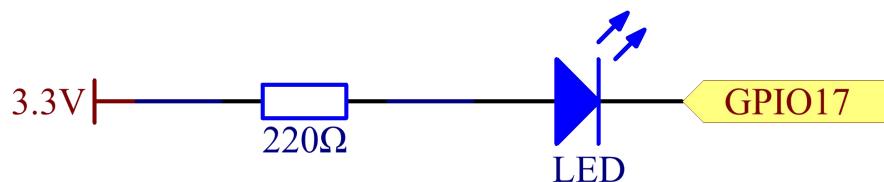
Schematische Darstellung

In diesem Experiment schließen Sie einen $220\ \Omega$ -Widerstand an die Anode (den langen Pin der LED) und dann den Widerstand an 3,3 V an und verbinden Sie die Kathode (den kurzen Pin) der LED mit GPIO17 von Raspberry Pi. Da um eine LED einzuschalten, muss der GPIO17-Niveau niedrig (0V) sein. Wir können dieses Phänomen durch Programmierung erhalten.

Hinweis: Pin11 bezieht sich von links nach rechts auf den 11. Pin des Raspberry Pi. Die entsprechenden **wiringPi** und **BCM**-Pin-Nummern sind in der folgenden Tabelle aufgeführt.

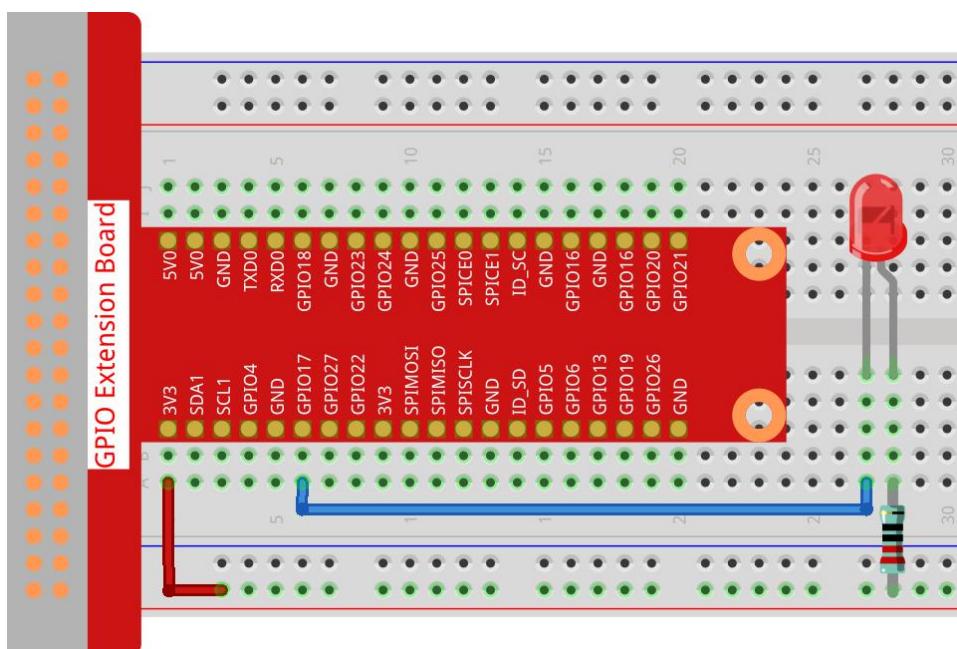
In den C-Sprachinhalten machen wir GPIO0 sz bis 0 in der wiringPi. Unter den Python-Sprachinhalten, BCM 17 ist 17 in der BCM-Spalte der folgenden Tabelle. Gleichzeitig sind sie gleich mit dem 11. Pin des Raspberry Pi, Pin 11.

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



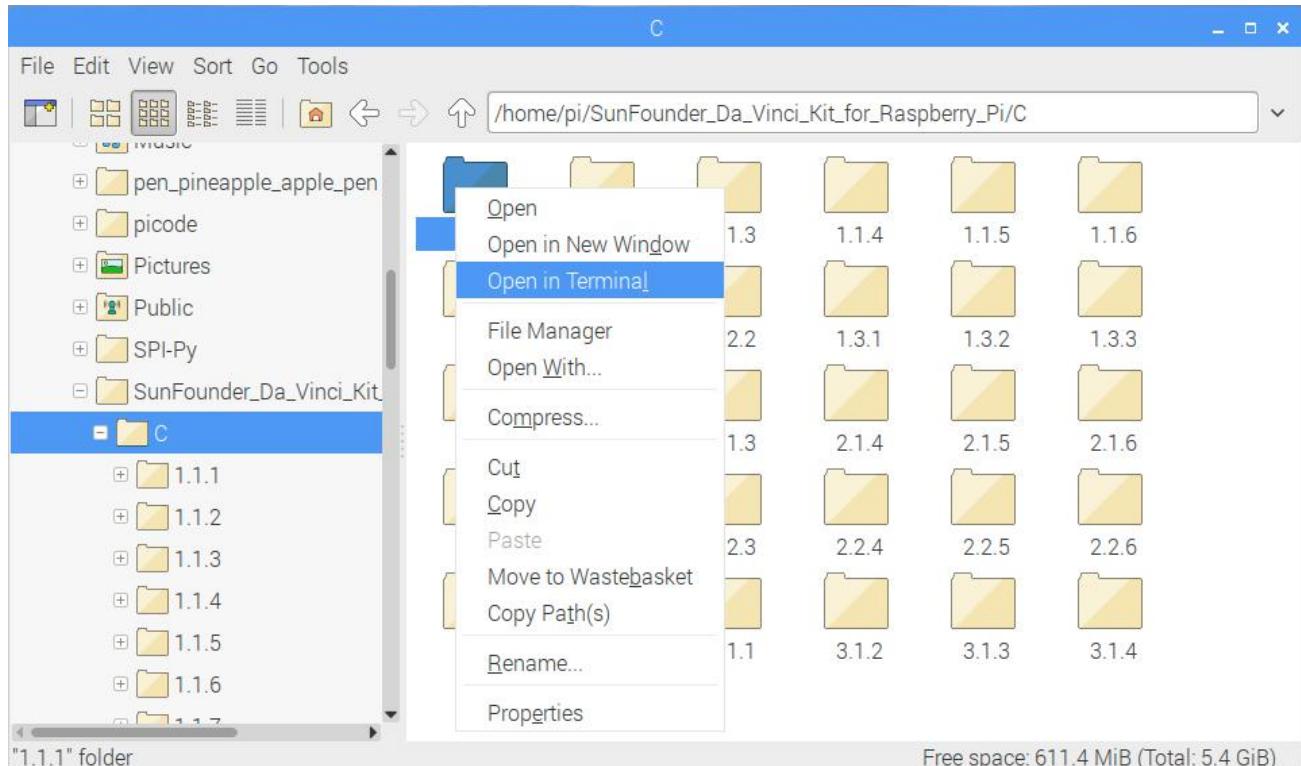
➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner des Codes.

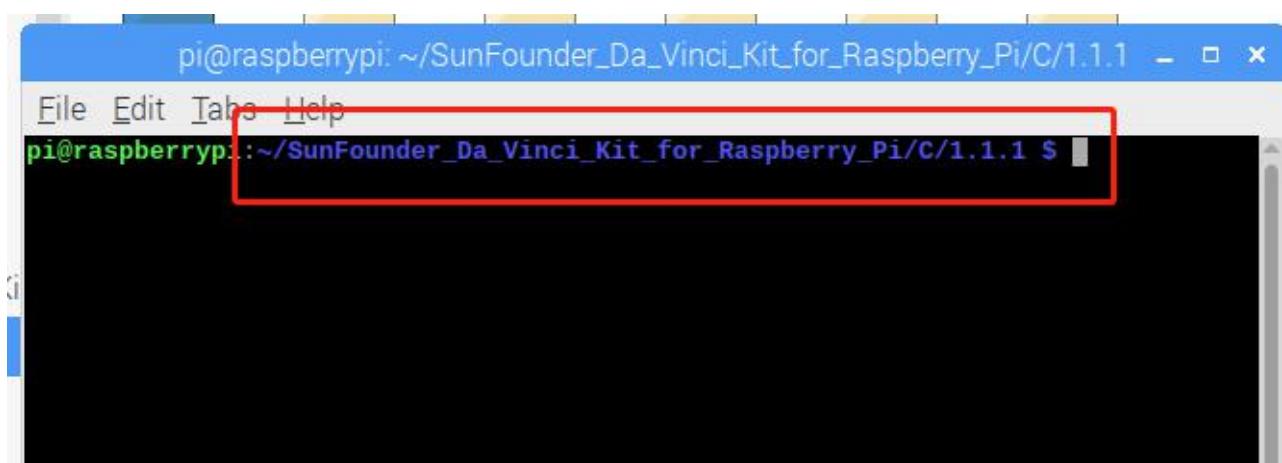
Wenn Sie einen Bildschirm verwenden, sind die folgenden Schritte empfohlen.

Gehen Sie zu **/home/pi/** und suchen Sie den Ordner **davinci-kit-for-raspberry-pi**.

Suchen Sie **C** im Ordner, klicken Sie mit der rechten Maustaste darauf und wählen Sie In Terminal öffnen.



Dann öffnet sich ein Fenster wie unten gezeigt. Nun haben Sie den Pfad des Codes **1.1.1_BlinkingLed.c** eingegeben.



In den folgenden Lektionen verwenden wir den Befehl, anstatt mit der rechten Maustaste zu klicken, um die Codedatei einzugeben,. Sie können jedoch die Methode auswählen, die Sie bevorzugen.

Wenn Sie sich beim Raspberry Pi aus der Ferne anmelden, wechseln Sie mit "cd" das Verzeichnis:

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/1.1.1/
```

Hinweis: Ändern Sie das Verzeichnis in den Codepfad in diesem Experiment durch CD.

In beiden Richtungen befinden Sie sich jetzt im Ordner C. Die nachfolgenden Verfahren basier auf diesen beiden Methoden, sind identisch. Lass uns weitermachen.

Schritt 3: Kompilieren Sie die Kode

```
gcc 1.1.1_BlinkingLed.c -o BlinkingLed -lwiringPi
```

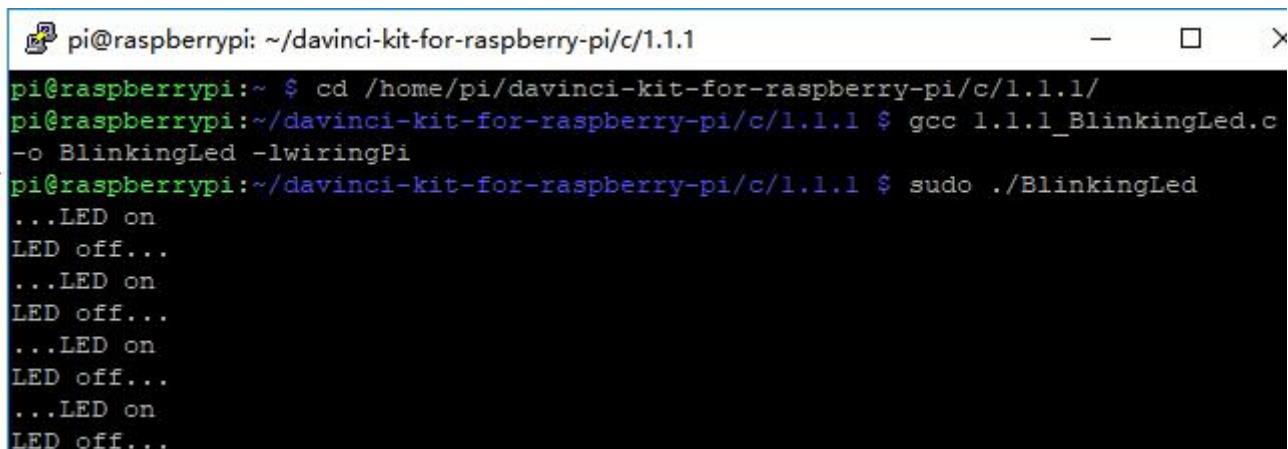
Hinweis: gcc ist die GNU Compilersammlung.. Hier funktioniert es wie das Kompilieren der C-Sprachdatei 1_BlinkingLed.c und das Ausgeben einer ausführbaren Datei.

Im Befehl bedeutet -o die Ausgabe (das Zeichen unmittelbar nach -o ist die Dateinamenausgabe nach der Kompilierung, und eine ausführbare Datei namens BlinkingLed wird hier generiert) , -lwiringPi ist das Laden der Bibliothek wiringPi (l ist die Abkürzung der Bibliothek).

Schritt 4: Führen Sie die Ausgabe der ausführbaren Datei im vorherigen Schritt aus.

```
sudo ./BlinkingLed
```

Hinweis: Um das GPIO zu steuern, müssen Sie das Programm mit dem Befehl sudo (Superuser do) ausführen. Der Befehl "./" gibt das aktuelle Verzeichnis an. Der gesamte Befehl ist für die Ausführung der **BlinkingLed** im aktuellen Verzeichnis.



```
pi@raspberrypi: ~ /davinci-kit-for-raspberry-pi/c/1.1.1
pi@raspberrypi:~ $ cd /home/pi/davinci-kit-for-raspberry-pi/c/1.1.1/
pi@raspberrypi:~/davinci-kit-for-raspberry-pi/c/1.1.1 $ gcc 1.1.1_BlinkingLed.c
-o BlinkingLed -lwiringPi
pi@raspberrypi:~/davinci-kit-for-raspberry-pi/c/1.1.1 $ sudo ./BlinkingLed
...LED on
LED off...
...LED on
LED off...
...LED on
LED off...
...LED on
LED off...
```

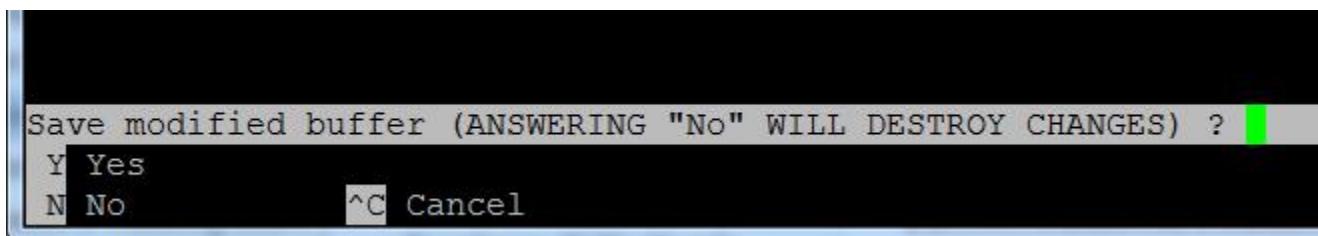
Nachdem der Kode ausgeführt wurde, blinkt die LED.

Wenn Sie die Kodedatei `1.1.1_BlinkingLed.c` bearbeiten möchten, drücken Sie **Ctrl + C**, um die Ausführung des Kodes zu beenden. Geben Sie dann den folgenden Befehl ein und öffnen es.

```
nano 1.1.1_BlinkingLed.c
```

Hinweis: nano ist ein Texteditor. Der Befehl wird verwendet, um die Kodedatei `1.1.1_BlinkingLed.c` mit diesem Tool zu öffnen.

Drücken Sie **Ctrl+X** für Ausfahrt Wenn Sie die Kode geändert haben, werden Sie gefragt, ob Sie die Änderungen speichern möchten oder nicht. Geben Sie **Y** (speichern) oder **N** (nicht speichern) ein. Drücken Sie dann die **Enter**, um den Vorgang zu beenden. Wiederholen Sie **Schritt 3** und **Schritt 4**, um den Effekt nach dem Ändern zu sehen.



Kode

Der Programmkode wird wie folgt angezeigt:

```
#include <wiringPi.h>
#include <stdio.h>
#define LedPin      0
int main(void)
{
    // When initialize wiring failed, print message to screen
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }
    pinMode(LedPin, OUTPUT); // Set LedPin as output to write value to it.
    while(1){
        // LED on
        digitalWrite(LedPin, LOW);
        printf("...LED on\n");
        delay(500);
        // LED off
        digitalWrite(LedPin, HIGH);
```

```
    printf("LED off...\n");
    delay(500);
}
return 0;
}
```

Kode Erklärung

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

Die Hardware-Laufwerksbibliothek wurde für die C-Sprache von Raspberry Pi entwickelt. Das Hinzufügen dieser Bibliothek fördert die Initialisierung der Hardware und die Ausgabe von I/O ports, PWM usw.

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

Standard I/O Bibliothek. Die Pintf-Funktion zum Drucken der auf dem Bildschirm angezeigten Daten wird von dieser Bibliothek realisiert. Es gibt viele andere Leistungsfunktionen, die Sie erkunden können.

```
#define LedPin 0
```

Pin GPIO17 der T_ Erweiterungskarte entspricht dem GPIO0 in WiringPi. Weisen Sie Ledpin GPIO0 zu, was GPIO0 im zukünftigen Kode darstellt.

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

Dadurch wird wiringPi initialisiert und geht davon aus, dass das aufrufende Programm das wiringPi Nummerierungsschema verwendet.

Diese Funktion muss mit Root-Rechten aufgerufen werden. Wenn die Initialisierung der Verkabelung fehlgeschlagen ist, drucken Sie die Nachricht auf dem Bildschirm. Die Funktion „ Rückgabe “ wird verwendet, um aus der aktuellen Funktion herauszuspringen. Wenn Sie die Funktion Rückgabe in main () verwenden, wird das Programm beendet.

```
pinMode(LedPin, OUTPUT);
```

Stellen Sie LedPin als Ausgabe ein, um einen Wert darauf zu schreiben.

```
digitalWrite(LedPin, LOW);
```

Stellen Sie GPIO0 auf 0V (niedriger Niveau) ein. Da die Kathode der LED mit GPIO0 verbunden ist, leuchtet die LED auf, wenn GPIO0 niedrig eingestellt ist. Im Gegenteil, stellen Sie GPIO0 als High-Level ein. digitalWrite (LedPin, HIGH): Die LED erlischt.

```
printf("...LED off\n");
```

Die printf-Funktion ist eine Standardbibliotheksfunktion und ihr Funktionsprototyp befindet sich in der Header-Datei "stdio.h". Die allgemeine Form des Aufrufs lautet: printf ("Format Control String", Spalten der Ausgabetafel). Die Formatsteuerzeichenfolge wird verwendet, um das Ausgabeformat anzugeben, das in Formatzeichenfolge und Nichtformatzeichenfolge unterteilt ist. Die Formatzeichenfolge beginnt mit '%', gefolgt von Formatzeichen, z. B. '% d' für die Ausgabe von Dezimalzahlen. Unformatierte Zeichenfolgen werden als Prototypen gedruckt. Hier wird eine nicht formatierte Zeichenfolge verwendet, gefolgt von "\ n", einem Zeilenumbruchzeichen, das den automatischen Zeilenumbruch nach dem Drucken einer Zeichenfolge darstellt.

```
delay(500);
```

Die Verzögerung (500) hält den aktuellen HIGH- oder LOW-Status für 500 ms.

Dies ist eine Funktion, die das Programm für einen bestimmten Zeitraum anhält. Und die Geschwindigkeit des Programms wird von unserer Hardware bestimmt. Hier schalten wir die LED ein oder aus. Wenn es keine Verzögerungsfunktion gibt, führt das Programm das gesamte Programm sehr schnell und kontinuierlich aus. Wir brauchen also die Verzögerungsfunktion, um das Programm schreiben und debuggen zu können.

```
return 0;
```

Normalerweise wird es hinter der Hauptfunktion platziert, was anzeigt, dass die Funktion bei erfolgreicher Ausführung 0 zurückgibt.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner des Codes und führen Sie ihn aus.

Wenn Sie einen Bildschirm verwenden, sind die folgenden Schritte empfohlen.

Suchen Sie 1.1.1_BlinkingLed.py und doppelklicken Sie darauf, um es zu öffnen. Jetzt bist du in der Datei.

Klicken Sie im Fenster auf **Run -> Run Module**. Der folgende Inhalt wird angezeigt.

Um die Ausführung zu stoppen, klicken Sie einfach auf die X-Schaltfläche oben rechts, um sie zu schließen, und kehren Sie dann zum Code zurück. Wenn Sie die Kode ändern, müssen Sie ihn zuerst speichern, bevor Sie auf **Run Module (F5)** klicken. Dann können Sie die Ergebnisse sehen.

Wenn Sie sich beim Raspberry Pi aus der Ferne anmelden, geben Sie den folgenden Befehl ein:

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Hinweis: Wechseln Sie in diesem Experiment per **cd** in den Pfad der Kode

Schritt 3: Führen Sie die Kode aus

```
sudo python3 1.1.1_BlinkingLed.py
```

Hinweis: Hier tun sudo - superuser do und python bedeutet, die Datei von Python auszuführen.

Nachdem der Kode ausgeführt wurde, blinkt die LED.

Schritt 4: Wenn Sie die Dodedatei [1.1.1_BlinkingLed.py](#), bearbeiten möchten, drücken Sie **Ctrl + C**, um die Ausführung der Kode zu beenden. Geben Sie dann den folgenden Befehl ein, um [1.1.1_BlinkingLed.py](#): zu öffnen:

```
nano 1.1.1_BlinkingLed.py
```

Hinweis: `nano` ist ein Texteditor. Mit dem Befehl wird die Dodedatei [1.1.1_BlinkingLed.py](#) by this tool.von diesem Tool geöffnet.

Drücken Sie **Ctrl+X** für Ausfahrt Wenn Sie die Kode geändert haben, werden Sie gefragt, ob Sie die Änderungen speichern möchten oder nicht. Geben Sie **Y** (speichern) oder **N** (nicht speichern) ein.

Drücken Sie dann **Enter**, um den Vorgang zu beenden. Geben Sie `nano 1.1.1_BlinkingLed.py` erneut ein, um den Effekt nach der Änderung zu sehen.

Kode

Das Folgende ist der Programmkode:

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time
LedPin = 17
def setup():
```

```

# Set the GPIO modes to BCM Numbering
GPIO.setmode(GPIO.BCM)
# Set LedPin's mode to output, and initial level to High(3.3v)
GPIO.setup(LedPin, GPIO.OUT, initial=GPIO.HIGH)

# Define a main function for main process
def main():
    while True:
        print ('...LED ON')
        # Turn on LED
        GPIO.output(LedPin, GPIO.LOW)
        time.sleep(0.5)
        print ('LED OFF...')
        # Turn off LED
        GPIO.output(LedPin, 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(LedPin, GPIO.HIGH)
    # Release resource
    GPIO.cleanup()

# If run this script directly, do:
if __name__ == '__main__':
    setup()
    try:
        main()
    # When 'Ctrl+C' is pressed, the program destroy() will be executed.
    except KeyboardInterrupt:
        destroy()

```

Kode Erklärung

```
#!/usr/bin/env python3
```

Wenn das System dies erkennt, durchsucht es den Installationspfad von Python in der Umgebung env und ruft dann den entsprechenden Interpreter auf, um den Vorgang abzuschließen. Dies soll verhindern, dass der Benutzer die Python nicht auf dem Defaultpfad /usr/bin installiert.

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

Importieren Sie auf diese Weise die RPi.GPIO-Bibliothek und definieren Sie dann eine Variable, GPIO, um RPi.GPIO in der folgenden Kode zu ersetzen.

```
import time
```

Zeitpaket importieren, für Zeitverzögerungsfunktion im folgenden Programm.

```
LedPin = 17
```

Die LED wird mit dem GPIO17 der T-förmigen Erweiterungskarte, nämlich BCM 17, verbunden.

```
def setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(LedPin, GPIO.OUT, initial=GPIO.HIGH)
```

Stellen Sie den LedPin-Modus auf Ausgabe und den AnfangsNiveau auf Hoch (3.3v).

Es gibt zwei Wisen, die IO -Pins eines Raspberry Pi in RPi.GPIO zu nummerieren: BOARD-Nummern und BCM-Nummern. In unseren Lektionen verwenden wir BCM-Nummern. Sie müssen jeden Kanal einrichten, den Sie als Eingang oder Ausgang verwenden.

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

Stellen Sie GPIO17 (BCM17) auf 0V (niedriger Niveau) ein. Da die Kathode der LED mit GPIO17 verbunden ist, leuchtet die LED auf.

```
time.sleep(0.5)
```

Verzögerung um 0,5 Sekunden. Hier ähnelt die Anweisung der Verzögerungsfunktion in der Sprache C, die Einheit ist die zweite.

```
def destroy():
    GPIO.cleanup()
```

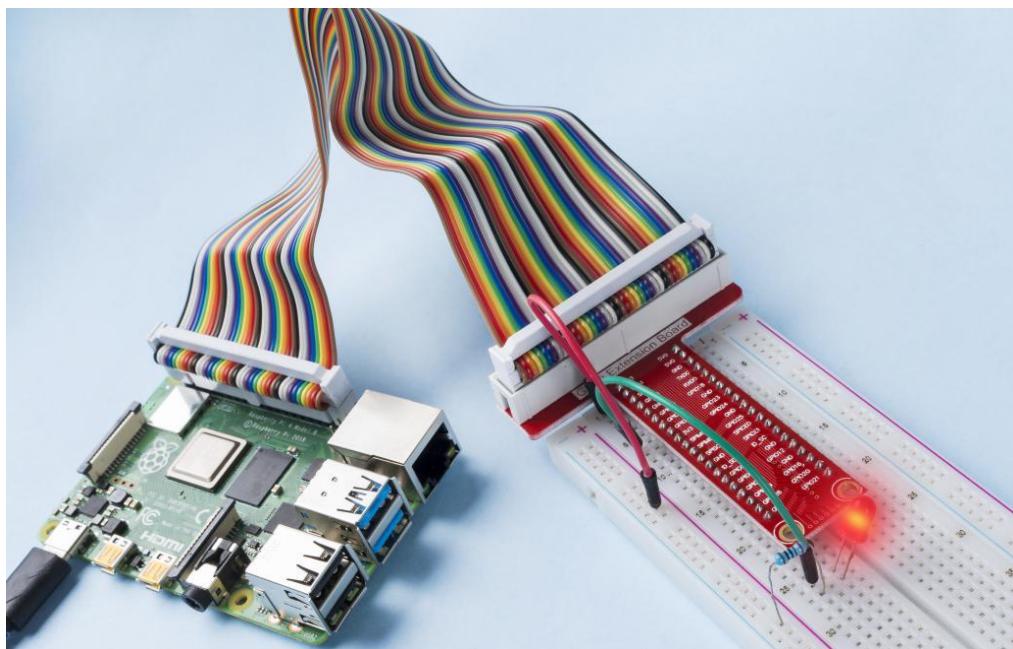
Definieren Sie eine Zerstörungsfunktion, um alles nach Abschluss des Skripts zu bereinigen.

```
if __name__ == '__main__':
    setup()
    try:
        main()
    # When 'Ctrl+C' is pressed, the program destroy() will be executed.
```

```
except KeyboardInterrupt:  
    destroy()
```

Dies ist die allgemeine Ausführungsstruktur der Kode Wenn das Programm gestartet wird, wird der Pin durch Ausführen von `setup ()` initialisiert und anschließend der Kode in der Funktion `main ()` ausgeführt, um den Pin auf hohe und niedrige Niveau zu setzen. Wenn 'Ctrl+C' gedrückt wird, wird das Programm `destroy ()` ausgeführt.

Phänomen Bild



1.1.2 RGB-LED

Einführung

In dieser Lektion steuern wir mit einer RGB-LED, die verschiedene Arten von Farben zu blinken.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * RGB-LED
1 * 40-Pin Kabel		Mehrere Überbrückungsdrähte
1 * Steckbrett		3 * Widerstand (220 Ω)

Prinzip

PWM

Die Pulsweitenmodulation oder PWM ist eine Technik, mit der analoge Ergebnisse mit digitalen Mitteln erzielt werden können. Die digitale Steuerung wird verwendet, um eine Rechteckwelle zu erzeugen, um ein Signal zwischen Ein und Aus zu schalten. Dieses Ein-Aus-Muster kann Spannungen zwischen Voll-Ein (5 Volt) und Aus (0 Volt) simulieren, mit der Änderung von Zeit, in dem das Signal an ist, gegenüber der Zeit,

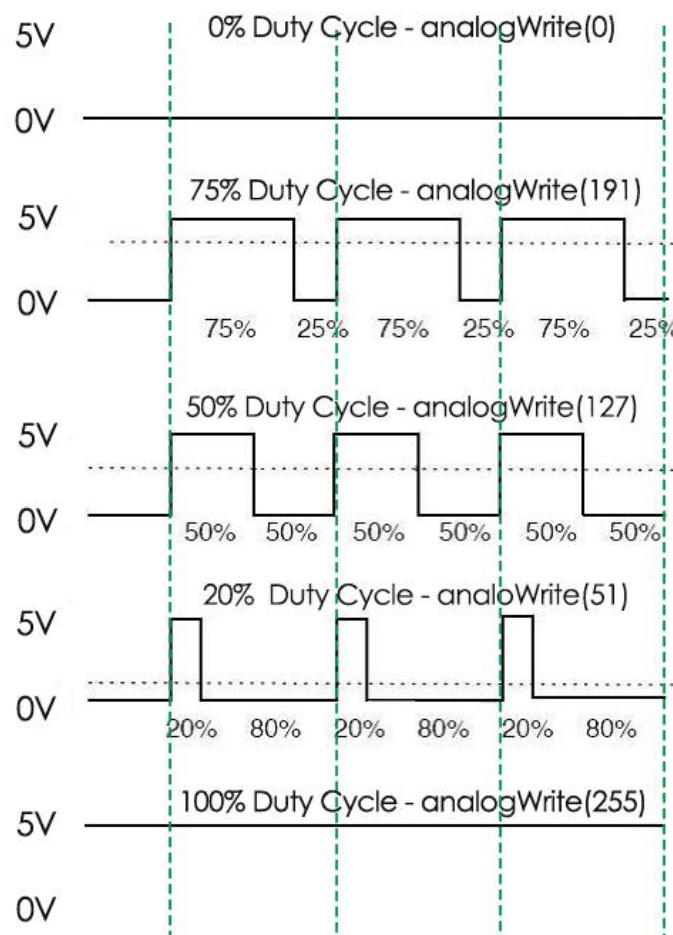
in der das Signal abläuft. Die Dauer der "Einschaltzeit" wird als Impulsbreite bezeichnet. Um unterschiedliche analoge Werte zu erhalten, ändern oder modulieren Sie diese Breite. Wenn Sie dieses Ein-Aus-Muster mit einem Gerät, beispielsweise einer LED, schnell genug wiederholen, sieht das Ergebnis folgendermaßen aus: Das Signal ist eine konstante Spannung zwischen 0 und 5 V, die die Helligkeit der LED steuert.

Arbeitszyklus

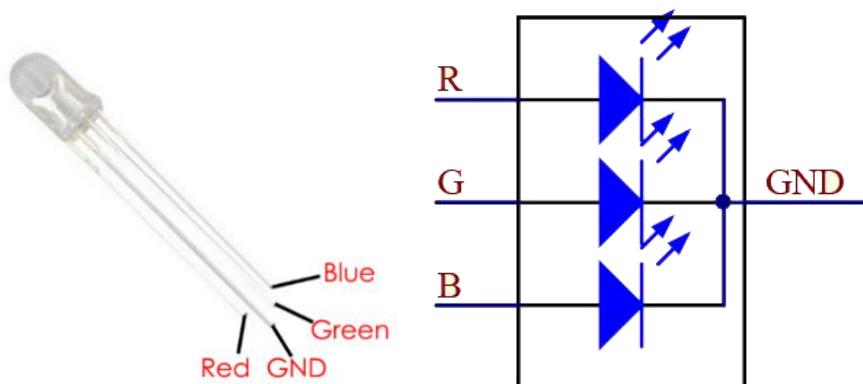
Ein Arbeitszyklus ist der Prozentsatz einer Periode, in der ein Signal aktiv ist. Eine Periode ist die Zeit, die ein Signal benötigt, um einen Ein- und Ausschaltzyklus abzuschließen. Als Formel kann ein Arbeitszyklus ausgedrückt werden als:

$$D = \frac{T}{P} \times 100\%$$

Wobei **D** das Tastverhältnis ist, **T** die Zeit ist, zu der das Signal aktiv ist, und **P** die Gesamtperiode des Signals ist. Ein Tastverhältnis von 60% bedeutet also, dass das Signal in 60% der Fälle eingeschaltet ist, in 40% der Fälle jedoch ausgeschaltet ist. Die "Pünktlichkeit" für einen Arbeitszyklus von 60% kann je nach Dauer des Zeitraums einen Bruchteil einer Sekunde, eines Tages oder sogar einer Woche betragen.



RGB LED

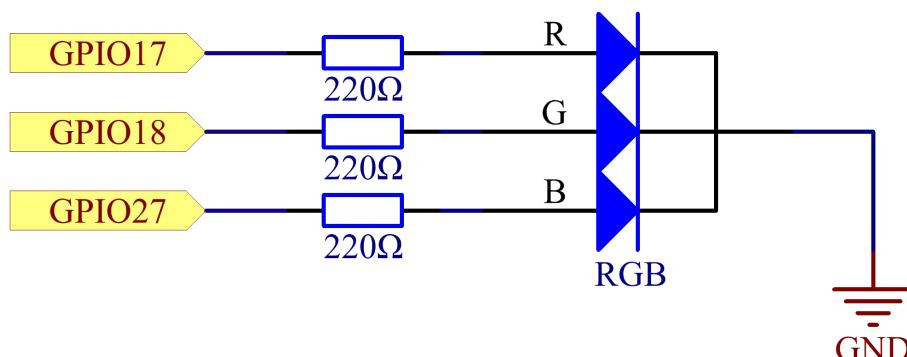


Die drei Primärfarben der RGB-LED können durch Helligkeit in verschiedene Farben gemischt werden. Die Helligkeit der LED kann mit PWM eingestellt werden. Raspberry Pi hat nur einen Kanal für die Hardware-PWM-Ausgabe, benötigt jedoch drei Kanäle zur Steuerung der RGB-LED, was bedeutet, dass es schwierig ist, die RGB-LED mit der Hardware-PWM von Raspberry Pi zu steuern. Glücklicherweise simuliert die softPwm-Bibliothek PWM (softPwm) durch Programmierung. Alles, was Sie tun müssen, ist die Titeldatei SoftPwm.h (C-Sprachbenutzer) und rufen Sie die API, die es bietet für die einfache Steuerung der RGB-LED durch die Mehrkanal-PWM-Ausgabe, um eine Vielzahl von Farben anzuzeigen.

Schematische Darstellung

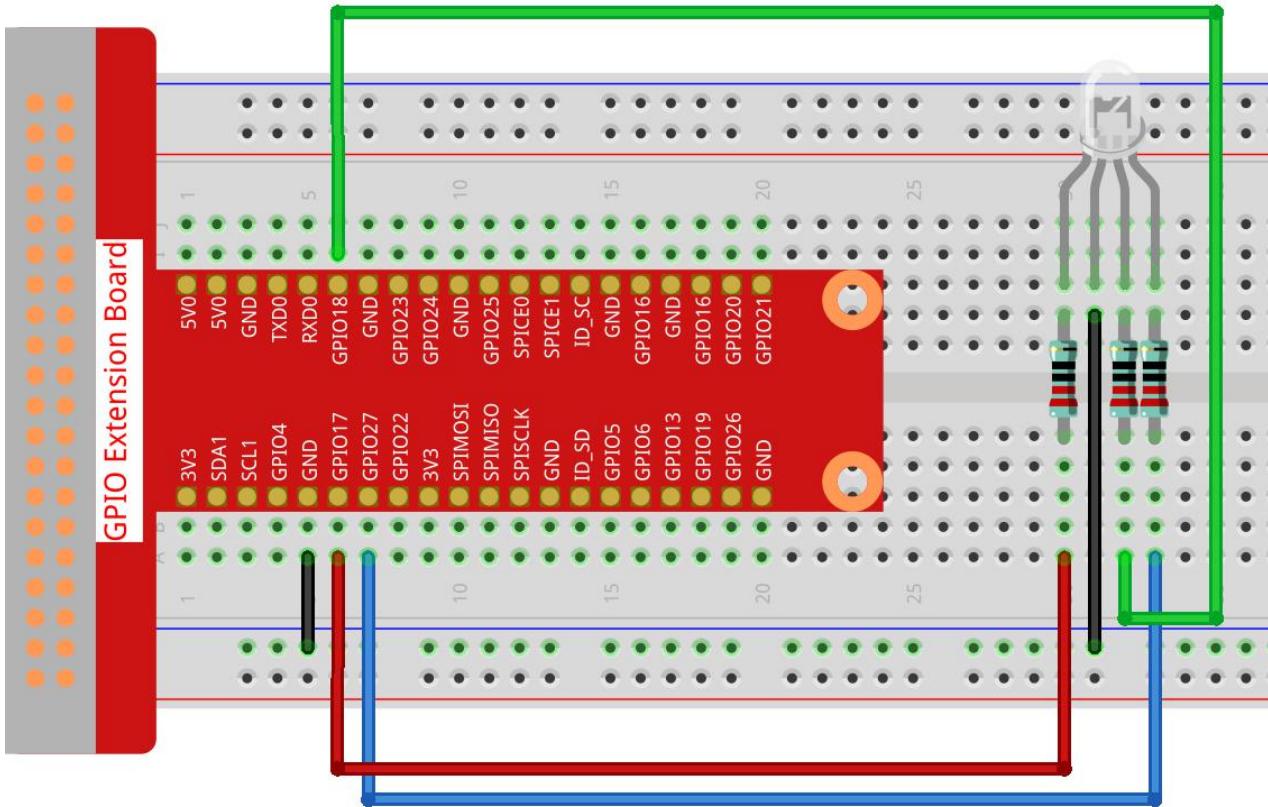
Nachdem Sie die Pins von R, G und B mit einem Strombegrenzungswiderstand verbunden haben, verbinden Sie sie mit dem GPIO17, GPIO18 bzw. GPIO27. Der längste Pin (GND) der LED ist mit dem GND des Raspberry Pi verbunden. Wenn die drei Pins unterschiedliche PWM-Werte erhalten, zeigt die RGB-LED unterschiedliche Farben an.

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode.

```
cd
```

Schritt 3: Kompilieren Sie die Kode

```
gcc 1.1.2_rgbLed.c -lwiringPi
```

Hinweis: Wenn die Anweisung "gcc" ausgeführt wird und "-o" nicht aufgerufen wird, wird ausführbare Datei "a.out" benannt.

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Nachdem die Kode ausgeführt wurde, sehen Sie, dass RGB Rot, Grün, Blau, Gelb, Pink und Cyan anzeigt.

Kode

```
#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>
#define uchar unsigned char
#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(uchar r_val, uchar g_val, uchar b_val){
    softPwmWrite(LedPinRed,   r_val);
    softPwmWrite(LedPinGreen, g_val);
    softPwmWrite(LedPinBlue,  b_val);
}

int main(void){

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

    ledInit();
    while(1){
        printf("Red\n");
        ledColorSet(0xff,0x00,0x00); //red
        delay(500);
        printf("Green\n");
        ledColorSet(0x00,0xff,0x00); //green
        delay(500);
    }
}
```

```

printf("Blue\n");
ledColorSet(0x00,0x00,0xff); //blue
delay(500);
printf("Yellow\n");
ledColorSet(0xff,0xff,0x00); //yellow
delay(500);
printf("Purple\n");
ledColorSet(0xff,0x00,0xff); //purple
delay(500);
printf("Cyan\n");
ledColorSet(0xc0,0xff,0x3e); //cyan
delay(500);
}
return 0;
}

```

Kode Erklärung

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

Bibliothek zur Realisierung der PWM-Funktion der Software.

```

void ledInit(void){
    softPwmCreate(LedPinRed, 0, 100);
    softPwmCreate(LedPinGreen,0, 100);
    softPwmCreate(LedPinBlue, 0, 100);
}

```

Die Funktion besteht darin, mit Software einen PWM-Pin zu erstellen und dessen Periode zwischen 0x100us und 100x100us einzustellen.

Der Prototyp der Funktion softPwmCreate (LedPinRed, 0, 100) lautet wie folgt:

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

Parameter-Pin: Jeder GPIO-Pin von Raspberry Pi kann als PWM-Pin gesetzt werden.

Parameter initialValue: Die anfängliche Impulsbreite ist der initialValue times100us.

Parameter pwmRange: Die Periode von PWM ist die pwmRange times100us.

```
void ledColorSet(uchar r_val, uchar g_val, uchar b_val){
```

```

softPwmWrite(LedPinRed,    r_val);
softPwmWrite(LedPinGreen,  g_val);
softPwmWrite(LedPinBlue,   b_val);
}

```

Diese Funktion dient zum Einstellen der Farben der LED. Bei Verwendung von RGB repräsentiert der formale Parameter **r_val** die Luminanz des roten, **g_val** des grünen, **b_val** des blauen.

Der Prototyp der Funktion softPwmWrite (LedPinBlue, b_val) lautet wie folgt:

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

Parameter-Pin: Jeder GPIO-Pin von Raspberry Pi kann als PWM-Pin gesetzt werden.

Parameterwert: Die Impulsbreite von PWM beträgt den Wert mal 100us. Beachten Sie, dass der Wert nur kleiner als der zuvor definierte pwmRange sein kann. Wenn er größer als pwmRange ist, erhält der Wert den festen Wert pwmRange.

```
ledColorSet(0xff,0x00,0x00);
```

Rufen Sie die zuvor definierte Funktion auf. Schreiben Sie 0xff in LedPinRed und 0x00 in LedPinGreen und LedPinBlue. Nach dem Ausführen der Kode leuchtet nur die rote LED auf. Wenn Sie LEDs in anderen Farben aufleuchten möchten, ändern Sie einfach die Parameter.

➤ Für Python-Sprachbenutzer

Schritt 2: Öffnen Sie die Kodedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Schritt 3: Ausführen.

```
sudo python3 1.1.2_rgbLed.py
```

Nachdem die Kode ausgeführt wurde, sehen Sie, dass RGB Rot, Grün, Blau, Gelb, Pink und Cyan anzeigt.

Kode

```

import RPi.GPIO as GPIO
import time
# Set up a color table in Hexadecimal
COLOR = [0xFF0000, 0x00FF00, 0x0000FF, 0xFFFF00, 0xFF00FF, 0x00FFFFFF]
# Set pins' channels with dictionary

```

```

pins = {'Red':17, 'Green':18, 'Blue':27}

def setup():
    global p_R, p_G, p_B
    GPIO.setmode(GPIO.BCM)
    # Set all LedPin's mode to output and initial level to High(3.3v)
    for i in pins:
        GPIO.setup(pins[i], GPIO.OUT, initial=GPIO.HIGH)

    p_R = GPIO.PWM(pins['Red'], 2000)
    p_G = GPIO.PWM(pins['Green'], 2000)
    p_B = GPIO.PWM(pins['Blue'], 2000)
    p_R.start(0)
    p_G.start(0)
    p_B.start(0)

# Define a MAP function for mapping values. Like from 0~255 to 0~100
def MAP(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

# Define a function to set up colors
def setColor(color):
    # configures the three LEDs' luminance with the inputted color value.
    R_val = (color & 0xFF0000) >> 16
    G_val = (color & 0x00FF00) >> 8
    B_val = (color & 0x0000FF) >> 0

    # Map color value from 0~255 to 0~100
    R_val = MAP(R_val, 0, 255, 0, 100)
    G_val = MAP(G_val, 0, 255, 0, 100)
    B_val = MAP(B_val, 0, 255, 0, 100)

    # Change the colors
    p_R.ChangeDutyCycle(R_val)
    p_G.ChangeDutyCycle(G_val)
    p_B.ChangeDutyCycle(B_val)

```

```

print ("color_msg: R_val = %s, G_val = %s, B_val = %s"%(R_val, G_val, B_val))

def main():
    while True:
        for color in COLOR:
            setColor(color)# change the color of the RGB LED
            time.sleep(0.5)

def destroy():
    # Stop all pwm channel
    p_R.stop()
    p_G.stop()
    p_B.stop()
    # Turn off all LEDs
    GPIO.output(pins, GPIO.HIGH)
    # Release resource
    GPIO.cleanup()

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

```

Kode Erklärung

```

p_R = GPIO.PWM(pins['Red'], 2000)
p_G = GPIO.PWM(pins['Green'], 2000)
p_B = GPIO.PWM(pins['Blue'], 2000)

p_R.start(0)
p_G.start(0)
p_B.start(0)

```

Rufen Sie die Funktion `GPIO.PWM ()` auf, um Rot, Grün und Blau als PWM-Pins zu definieren und die Frequenz der PWM-Pins auf 2000 Hz einzustellen. Verwenden Sie dann die Funktion `Start ()`, um den anfänglichen Arbeitszyklus auf Null zu setzen.

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

Definieren Sie eine MAP-Funktion zum Zuordnen von Werten. Zum Beispiel ist $x = 50$, $in_min = 0$, $in_max = 255$, $out_min = 0$, $out_max = 100$. Nach der Zuordnung der Kartenfunktion wird $(50-0) * (100-0)/(255-0) + 0 = 19.6$, zurückgegeben, was bedeutet, dass 50 in 0-255 19,6 in 0-100 entspricht.

```
def setColor(color):
    R_val = (color & 0xFF0000) >> 16
    G_val = (color & 0x00FF00) >> 8
    B_val = (color & 0x0000FF) >> 0
```

Konfiguriert die Luminanz der drei LEDs mit dem eingegebenen Farbwert. Weisen Sie R_val die ersten beiden Hexadezimalwerte zu, G_val die beiden mittleren und B_val die letzten beiden Werte. Wenn beispielsweise $color = 0xFF00FF$ ist, ist $R_val = 0xFF00FF \& 0xFF0000 >> 16 = 0xFF$, $G_val = 0x00$, $B_val = 0xFF$.

```
R_val = MAP(R_val, 0, 255, 0, 100)
G_val = MAP(G_val, 0, 255, 0, 100)
B_val = MAP(B_val, 0, 255, 0, 100)
```

Verwenden Sie die Zuordnungsfunktion, um den R-, G-, B-Wert zwischen 0 und 255 in den PWM-Arbeitszyklusbereich von 0 bis 100 abzubilden.

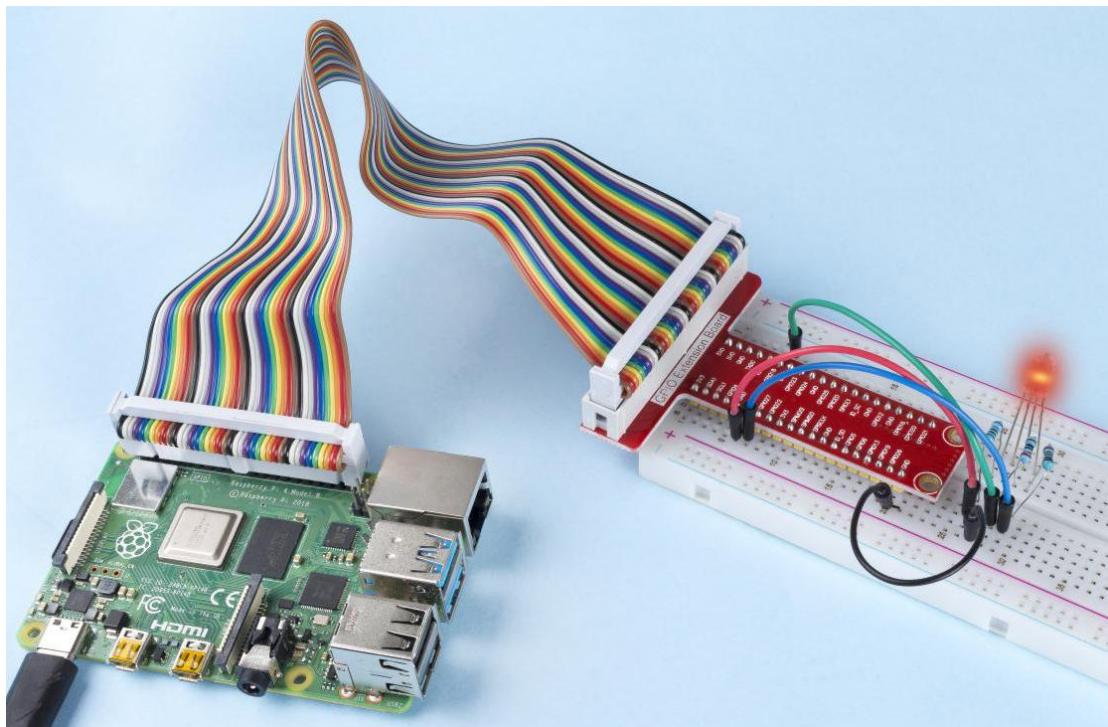
```
p_R.ChangeDutyCycle(R_val)
p_G.ChangeDutyCycle(G_val)
p_B.ChangeDutyCycle(B_val)
```

Weisen Sie den zugeordneten Tastverhältniswert dem entsprechenden PWM-Kanal zu, um die Luminanz zu ändern.

```
for color in COLOR:
    setColor(color)
    time.sleep(0.5)
```

Ordnen Sie jedes Element in der COLOR-Liste der jeweiligen Farbe zu und ändern Sie die Farbe der RGB-LED über die Funktion `setColor ()`.

Phänomen Bild

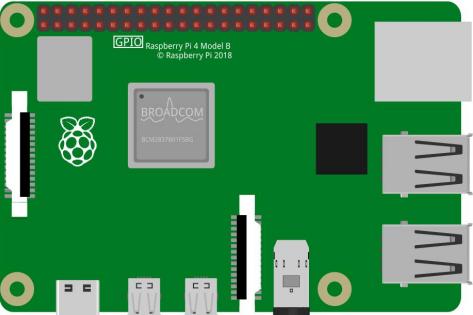
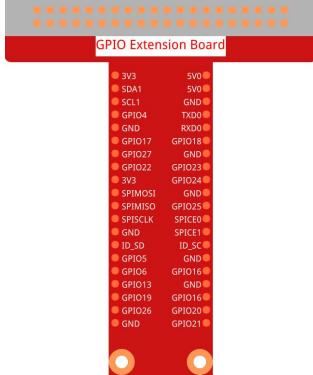
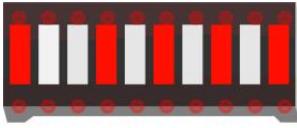
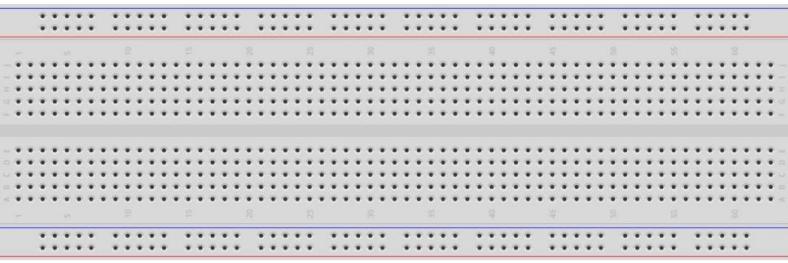


1.1.3 LED-Balkendiagramm

Einführung

In diesem Projekt beleuchten wir nacheinander die Lichter auf dem LED-Balkendiagramm.

Komponenten

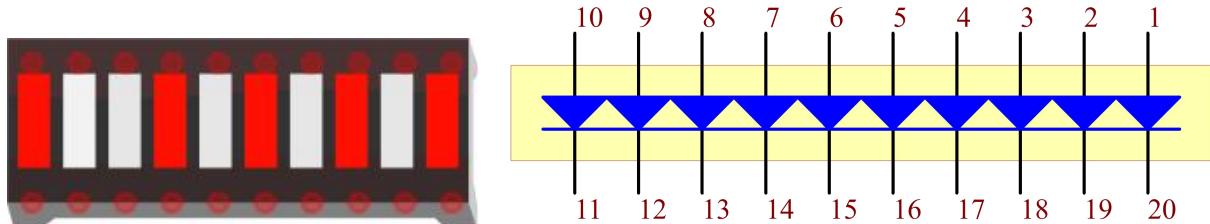
1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * LED-Bargraph
		
1 * 40-Pin Kabel		Mehrere Überbrückungsdrähte
		
1 * Steckbrett		10 * Widerstand (220 Ω)
		

Prinzip

LED-Balkendiagramm

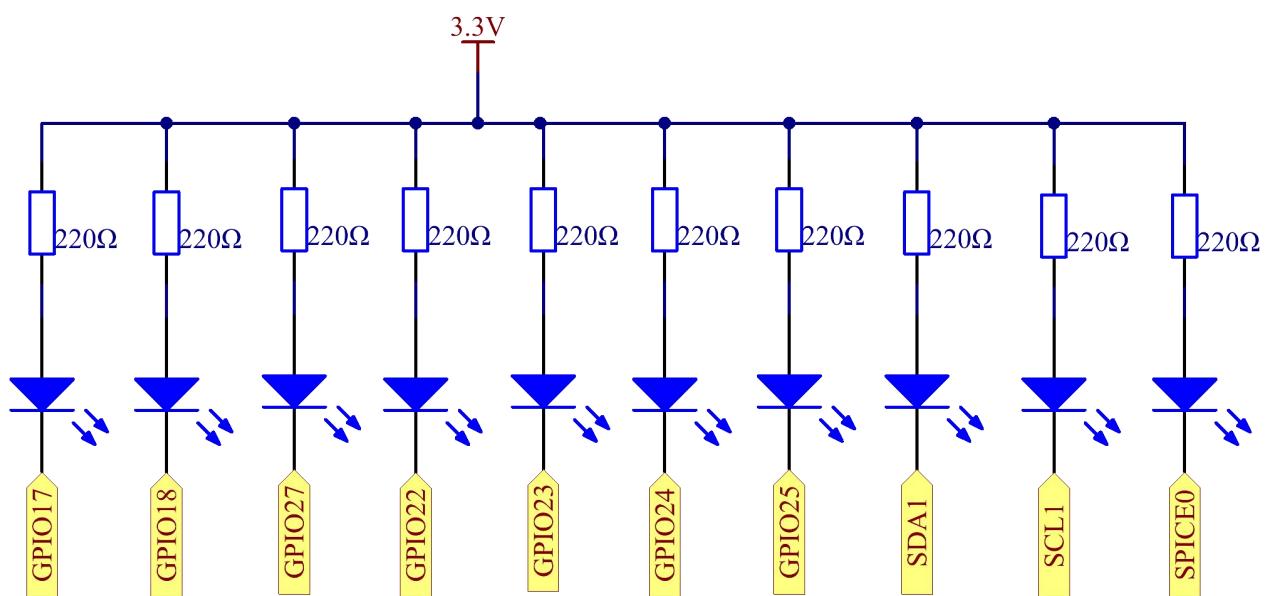
LED-Balkendiagramm ist ein LED-Array, das zur Verbindung mit einer elektronischen Schaltung oder einem Mikrocontroller verwendet wird. Es ist einfach, ein LED-Balkendiagramm mit der Schaltung zu verbinden, es ist wie 10 einzelne LEDs mit 10 Ausgangspins. Im Allgemeinen können wir das LED-Balkendiagramm als

Batteriestandsanzeige, Audiogeräte und industrielle Bedienfelder verwenden. Es gibt viele andere Anwendungen von LED-Balkendiagrammen.



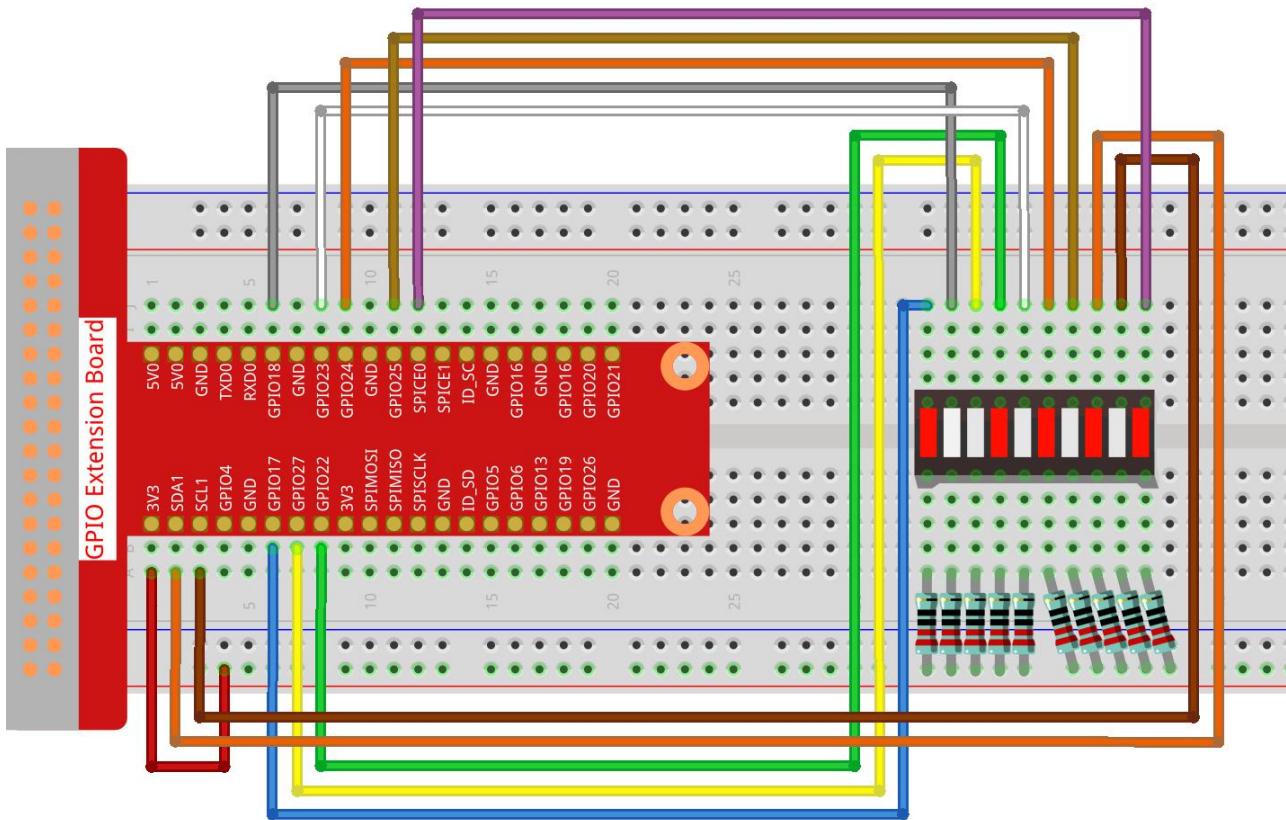
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24
GPIO25	Pin 22	6	25
SDA1	Pin 3	8	2
SCL1	Pin 5	9	3
SPICE0	Pin 24	10	8



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner des Codes.

```
cd ~/davinci-kit-for-raspberry-pi/c/1.1.3/
```

Schritt 3: Kompilieren Sie den Code.

```
gcc 1.1.3_LedBarGraph.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Nachdem die Kode ausgeführt wurde, werden die LEDs in der LED-Leiste regelmäßig ein- und ausgeschaltet.

Kode

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

int pins[10] = {0,1,2,3,4,5,6,8,9,10};
```

```
void oddLedBarGraph(void){  
    for(int i=0;i<5;i++){  
        int j=i*2;  
        digitalWrite(pins[j],HIGH);  
        delay(300);  
        digitalWrite(pins[j],LOW);  
    }  
}  
  
void evenLedBarGraph(void){  
    for(int i=0;i<5;i++){  
        int j=i*2+1;  
        digitalWrite(pins[j],HIGH);  
        delay(300);  
        digitalWrite(pins[j],LOW);  
    }  
}  
  
void allLedBarGraph(void){  
    for(int i=0;i<10;i++){  
        digitalWrite(pins[i],HIGH);  
        delay(300);  
        digitalWrite(pins[i],LOW);  
    }  
}  
  
int main(void)  
{  
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen  
        printf("setup wiringPi failed !");  
        return 1;  
    }  
    for(int i=0;i<10;i++) { //make led pins' mode is output  
        pinMode(pins[i], OUTPUT);  
    }  
}
```

```
    digitalWrite(pins[i],LOW);

}

while(1){

    oddLedBarGraph();

    delay(300);

    evenLedBarGraph();

    delay(300);

    allLedBarGraph();

    delay(300);

}

return 0;

}
```

Kode Erklärung

```
int pins[10] = {0,1,2,3,4,5,6,8,9,10};
```

Erstellen Sie ein Array und weisen Sie es der Pin-Nummer zu, die dem LED-Balkendiagramm (0,1,2,3,4,5,6,8,9,10) entspricht. Das Array wird zur Steuerung der LED verwendet.

```
void oddLedBarGraph(void){

    for(int i=0;i<5;i++){

        int j=i*2;

        digitalWrite(pins[j],HIGH);

        delay(300);

        digitalWrite(pins[j],LOW);

    }

}
```

Lassen Sie die LED an der ungeraden Stelle des LED-Balkendiagramms der Reihe nach leuchten.

```
void evenLedBarGraph(void){
```

```
for(int i=0;i<5;i++){  
    int j=i*2+1;  
    digitalWrite(pins[j],HIGH);  
    delay(300);  
    digitalWrite(pins[j],LOW);  
}  
}
```

Schalten Sie die LED auf der geraden Ziffer des LED-Balkendiagramms der Reihe nach ein.

```
void allLedBarGraph(void){  
    for(int i=0;i<10;i++){  
        digitalWrite(pins[i],HIGH);  
        delay(300);  
        digitalWrite(pins[i],LOW);  
    }  
}
```

Lassen Sie die LED auf dem LED-Balkendiagramm nacheinander leuchten.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner des Codes.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 1.1.3_LedBarGraph.py
```

Nachdem die Kode ausgeführt wurde, werden die LEDs in der LED-Leiste regelmäßig ein- und ausgeschaltet.

Kode

```
import RPi.GPIO as GPIO  
  
import time  
  
  
ledPins = [11, 12, 13, 15, 16, 18, 22, 3, 5, 24]
```

```
def oddLedBarGraph():
    for i in range(5):
        j = i*2
        GPIO.output(ledPins[j],GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(ledPins[j],GPIO.LOW)

def evenLedBarGraph():
    for i in range(5):
        j = i*2+1
        GPIO.output(ledPins[j],GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(ledPins[j],GPIO.LOW)

def allLedBarGraph():
    for i in ledPins:
        GPIO.output(i,GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(i,GPIO.LOW)

def setup():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)      # Numbers GPIOs by physical location
    for i in ledPins:
        GPIO.setup(i, GPIO.OUT)   # Set all ledPins' mode is output
        GPIO.output(i, GPIO.LOW)  # Set all ledPins to high(+3.3V) to off led

def loop():
    while True:
        oddLedBarGraph()
```

```

time.sleep(0.3)
evenLedBarGraph()
time.sleep(0.3)
allLedBarGraph()
time.sleep(0.3)

def destroy():
    for pin in ledPins:
        GPIO.output(pin, GPIO.LOW)      # turn off all leds
    GPIO.cleanup()                      # Release resource

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

```

Kode Erklärung

```
ledPins = [11, 12, 13, 15, 16, 18, 22, 3, 5, 24]
```

Erstellen Sie ein Array und weisen Sie es der Pin-Nummer zu, die dem LED-Balkendiagramm (11, 12, 13, 15, 16, 18, 22, 3, 5, 24) entspricht. Das Array wird zur Steuerung der LED verwendet.

```

def oddLedBarGraph():
    for i in range(5):
        j = i*2
        GPIO.output(ledPins[j],GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(ledPins[j],GPIO.LOW)

```

Lassen Sie die LED an der ungeraden Stelle des LED-Balkendiagramms der Reihe nach leuchten.

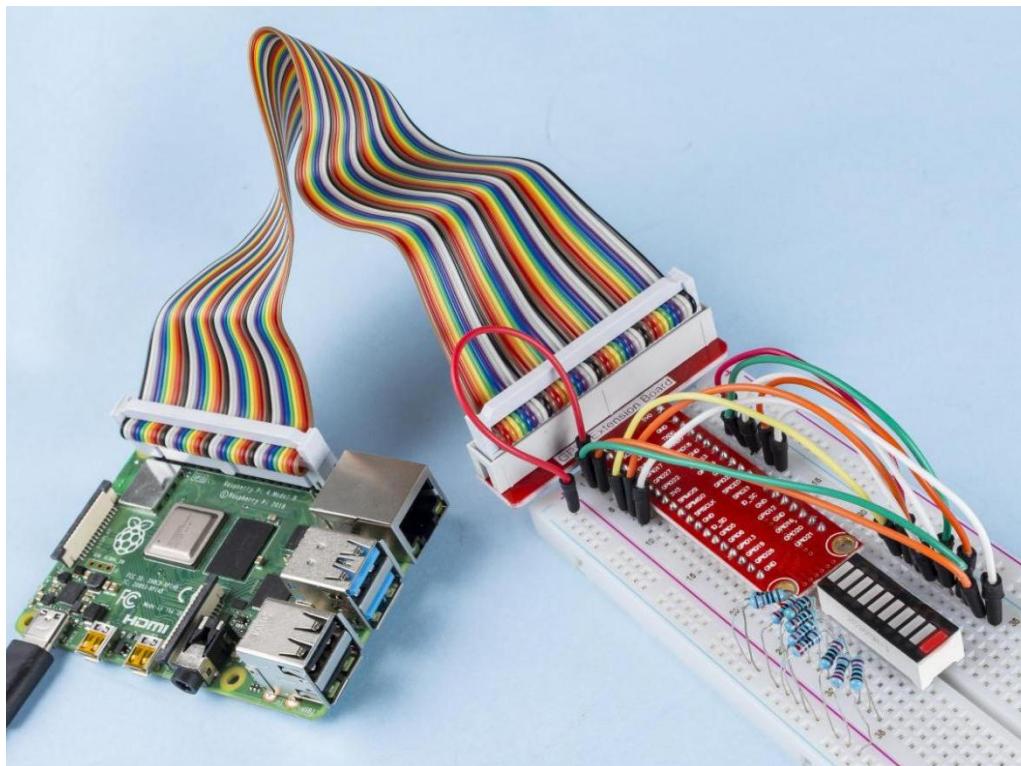
```
def evenLedBarGraph():
    for i in range(5):
        j = i*2+1
        GPIO.output(ledPins[j],GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(ledPins[j],GPIO.LOW)
```

Schalten Sie die LED auf der geraden Ziffer des LED-Balkendiagramms der Reihe nach ein.

```
def allLedBarGraph():
    for i in ledPins:
        GPIO.output(i,GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(i,GPIO.LOW)
```

Lassen Sie die LED auf dem LED-Balkendiagramm nacheinander leuchten.

Phänomen Bild

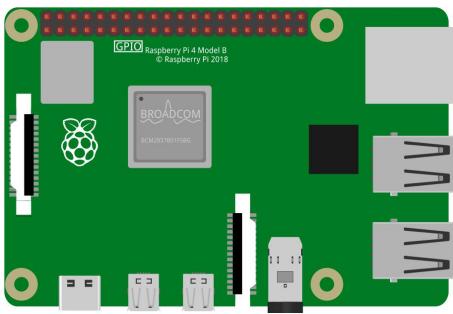
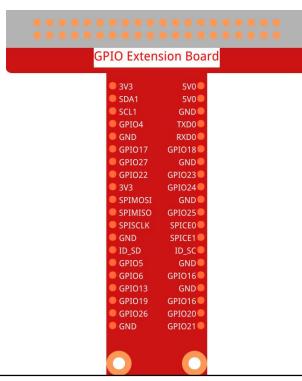
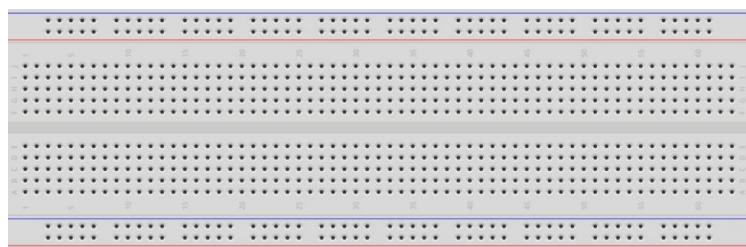
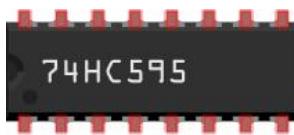


1.1.4 7-Segment-Anzeige

Einführung

Versuchen wir, eine 7-Segment-Anzeige anzutreiben, um eine Nummer von 0 bis 9 und von A bis F anzuzeigen.

Komponenten

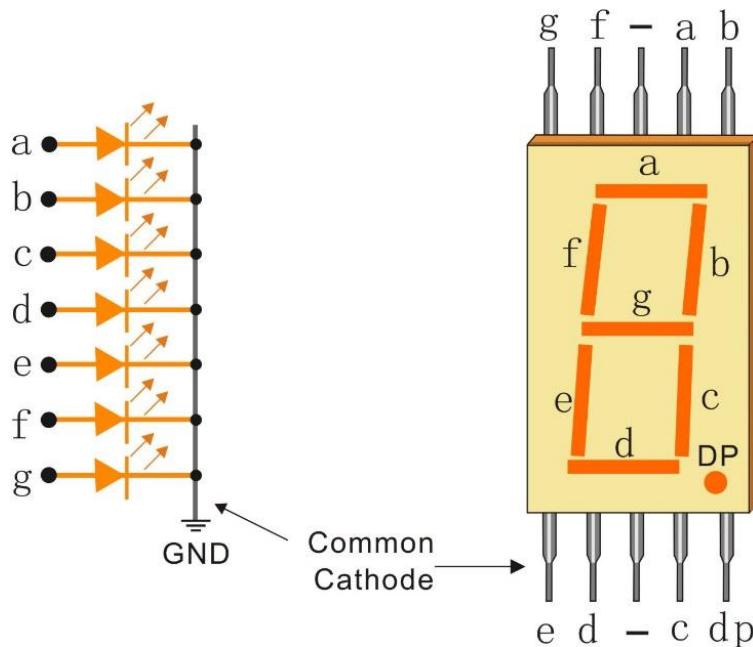
1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * 7-Segment-Anzeige																																								
	 GPIO Extension Board <table border="1"><tr><td>3V3</td><td>SVO</td></tr><tr><td>SDA1</td><td>GND</td></tr><tr><td>SCL1</td><td>TxD0</td></tr><tr><td>GPIO4</td><td>RxD0</td></tr><tr><td>GND</td><td>GPIO18</td></tr><tr><td>GPIO17</td><td>GPIO19</td></tr><tr><td>GPIO27</td><td>GND</td></tr><tr><td>GPIO22</td><td>GPIO23</td></tr><tr><td>3V</td><td>GPIO24</td></tr><tr><td>SPIMOSI</td><td>GND</td></tr><tr><td>SPIMISO</td><td>GPIO25</td></tr><tr><td>SPICLK</td><td>SPICE0</td></tr><tr><td>GND</td><td>SPICE1</td></tr><tr><td>ID_SD</td><td>ID_SC</td></tr><tr><td>GPIO5</td><td>GND</td></tr><tr><td>GPIO6</td><td>GPIO16</td></tr><tr><td>GPIO13</td><td>GND</td></tr><tr><td>GPIO19</td><td>GPIO16</td></tr><tr><td>GPIO26</td><td>GPIO20</td></tr><tr><td>GND</td><td>GPIO21</td></tr></table>	3V3	SVO	SDA1	GND	SCL1	TxD0	GPIO4	RxD0	GND	GPIO18	GPIO17	GPIO19	GPIO27	GND	GPIO22	GPIO23	3V	GPIO24	SPIMOSI	GND	SPIMISO	GPIO25	SPICLK	SPICE0	GND	SPICE1	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	
3V3	SVO																																									
SDA1	GND																																									
SCL1	TxD0																																									
GPIO4	RxD0																																									
GND	GPIO18																																									
GPIO17	GPIO19																																									
GPIO27	GND																																									
GPIO22	GPIO23																																									
3V	GPIO24																																									
SPIMOSI	GND																																									
SPIMISO	GPIO25																																									
SPICLK	SPICE0																																									
GND	SPICE1																																									
ID_SD	ID_SC																																									
GPIO5	GND																																									
GPIO6	GPIO16																																									
GPIO13	GND																																									
GPIO19	GPIO16																																									
GPIO26	GPIO20																																									
GND	GPIO21																																									
1 * 40-Pin Kabel		1 * Widerstand (220 Ω)																																								
																																										
1 * Steckbrett	Mehrere Überbrückungsdrähte	1 * 74HC595																																								
																																										

Prinzip

7-Segment-Anzeige

Ein 7-Segment-Display ist eine 8-förmige Komponente, die 7 LEDs enthält. Jede LED wird als Segment bezeichnet. Bei Erregung ist ein Segment Teil einer anzuzeigenden Ziffer.

Es gibt zwei Typen von Pin-Verbindungen: Common Cathode (CC) und Common Anode (CA). Wie der Name schon sagt, sind an einer CC-Anzeige alle Kathoden der 7 LEDs angeschlossen, wenn an einer CA-Anzeige alle Anoden der 7 Segmente angeschlossen sind. In diesem Kit verwenden wir der erstere.



Jede der LEDs im Display erhält ein Positionssegment, wobei einer der Verbindungsstifte aus dem rechteckigen Kunststoffgehäuse herausgeführt wird. Diese LED-Pins sind von "a" bis "g" bezeichnet und repräsentieren jede einzelne LED. Die anderen LED-Pins sind miteinander verbunden und bilden einen gemeinsamen Pin. Wenn Sie also die entsprechenden Pins der LED-Segmente in einer bestimmten Reihenfolge nach vorne vorspannen, werden einige Segmente heller und andere dunkel bleiben, wodurch das entsprechende Zeichen auf dem Display angezeigt wird.

Koden anzeigen

Damit Sie wissen können, wie 7-Segment-Anzeigen (Common Cathode) Numbers anzeigen, haben wir die folgende Tabelle gezeichnet. Nummer sind die Nummer 0-F, die auf der 7-Segment-Anzeige angezeigt werden. (DP) GFEDCBA bezieht sich auf die entsprechende LED, die auf 0 oder 1 gesetzt ist. Beispielsweise bedeutet 00111111, dass DP und G auf 0 gesetzt sind, während andere auf 1 gesetzt sind. Daher wird die Nummer 0 auf dem 7-Segment-Display angezeigt, während der HEX-Kode der Hexadezimalzahl entspricht.

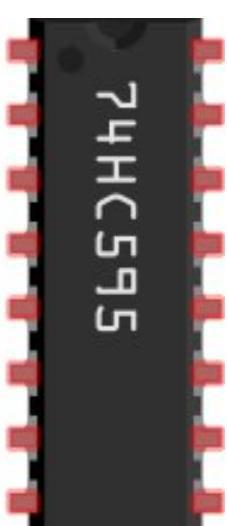
Nummer	Gemeisame Cathode		Nummer	Gemeisame Cathode	
	(DP) GFEDCBA	Hex-Kode		(DP) GFEDCBA	Hex-Kode
0	00111111	0x3f	A	01110111	0x77

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

74HC595

Der 74HC595 besteht aus einem 8-Bit-Schieberegister und einem Speicherregister mit parallelen Ausgängen mit drei Zuständen. Es wandelt den seriellen Eingang in einen parallelen Ausgang um, sodass Sie E / A-Ports einer MCU speichern können.

Wenn MR (Pin 10) einen hohen Niveaul und OE (Pin 13) einen niedrigen Niveaul aufweist, werden Daten in die ansteigende Flanke von SHcp eingegeben und gehen über die ansteigende Flanke von SHcp in das Speicherregister. Wenn die beiden Takte miteinander verbunden sind, ist das Schieberegister immer einen Impuls früher als das Speicherregister. Es gibt einen seriellen Verschiebungseingangspin (Ds), einen seriellen Ausgangspin (Q) und eine asynchrone Rücksetztaste (niedriger Niveaul) im Speicherregister. Das Speicherregister gibt einen Bus mit einem parallelen 8-Bit und in drei Zuständen aus. Wenn OE aktiviert ist (niedriger Niveaul), werden die Daten im Speicherregister an den Bus ausgegeben.



1	Q1	VCC	16
2	Q2	Q0	15
3	Q3	DS	14
4	Q4	CE	13
5	Q5	STcp	12
6	Q6	SHcp	11
7	Q7	MR	10
8	GND	Q7'	9

Pins von 74HC595 und ihre Funktionen:

Q0-Q7: Parallele 8-Bit-Datenausgangspins, mit denen 8 LEDs oder 8 Pins der 7-Segment-Anzeige direkt gesteuert werden können.

Q7': Serienausgangspin, verbunden mit DS eines anderen 74HC595, um mehrere 74HC595 in Reihe zu schalten

MR: Reset-Pin, aktiv bei niedrigem Niveau;

SH_{CP}: Zeitsequenz-Eingabe des Schieberegisters. Bei der ansteigenden Flanke bewegen sich die Daten im Schieberegister nacheinander um ein Bit, dh die Daten in Q1 sich zu Q2 und so weiter bewegen. Während der fallenden Flanke bleiben die Daten im Schieberegister unverändert.

ST_{CP}: Zeitsequenz-Eingabe des Speicherregisters. Bei steigender Flanke werden Daten im Schieberegister in das Speicherregister verschoben.

CE: Ausgangsfreigabepin, aktiv auf niedrigem Niveau.

DS: Serieller Dateneingangspin

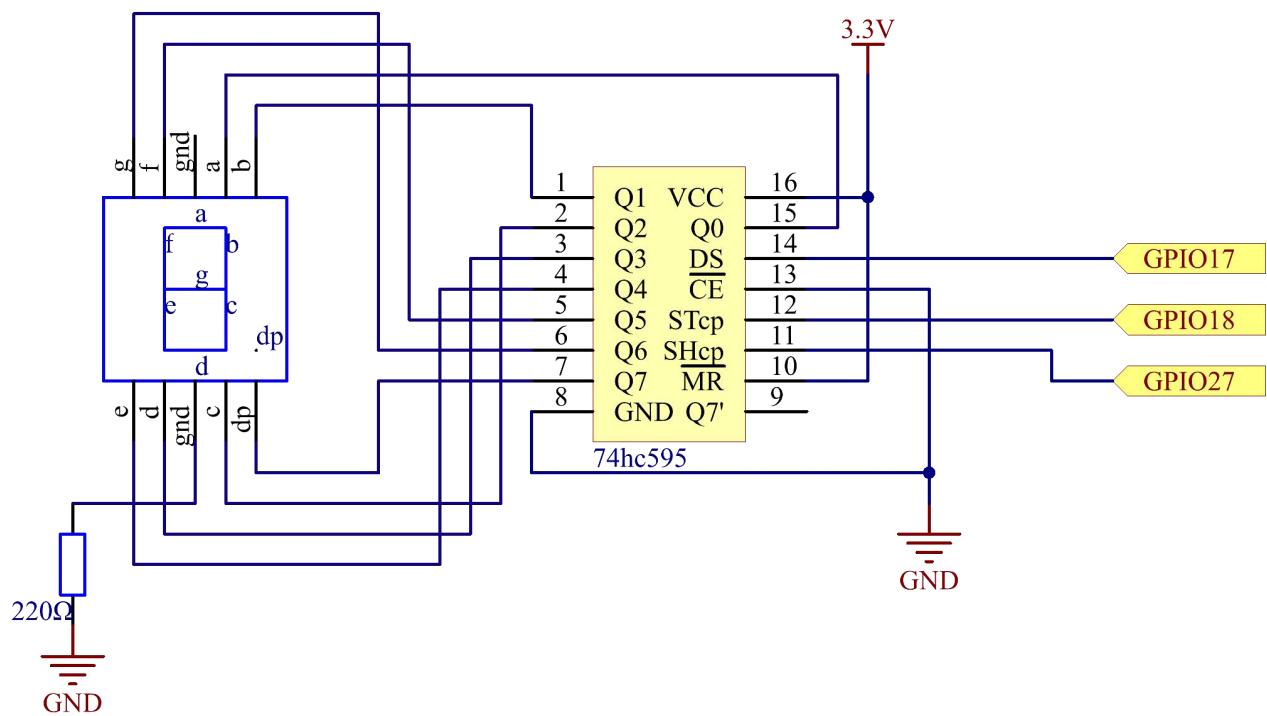
VCC: Positive Versorgungsspannung

GND: Boden

Schematische Darstellung

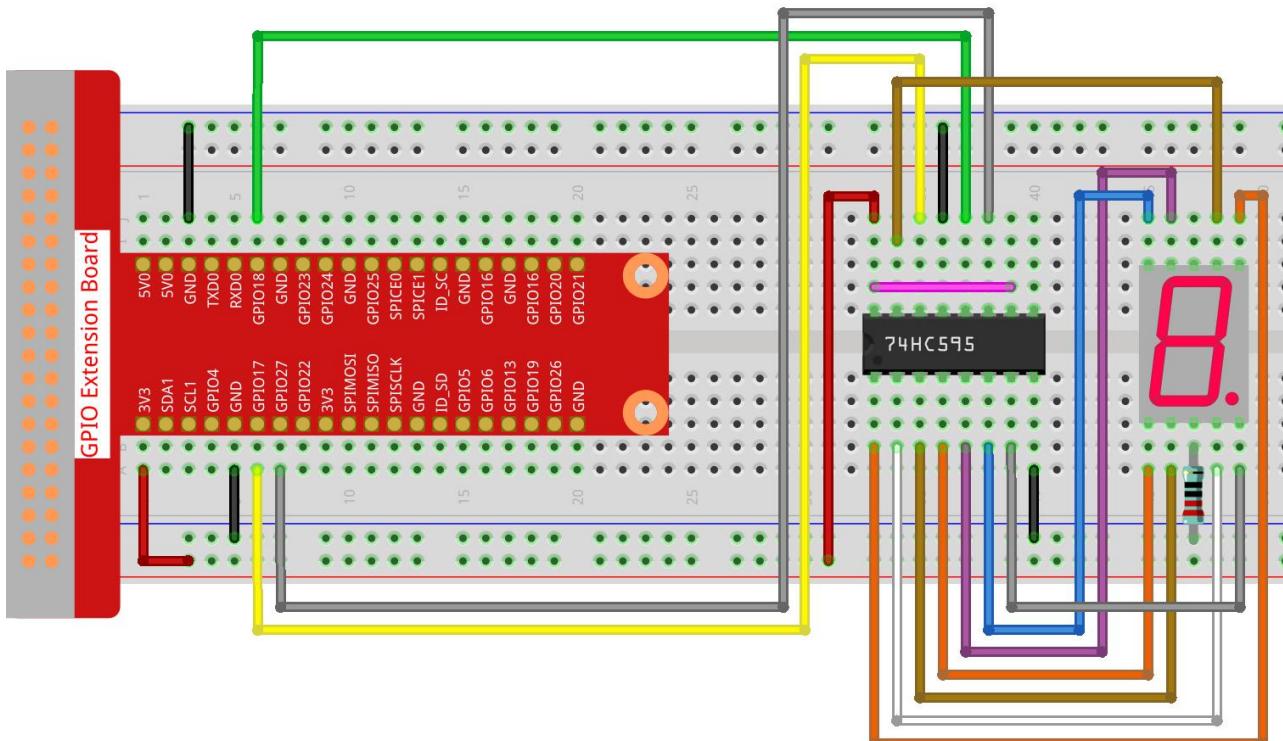
Verbinden Sie Pin ST_CP von 74HC595 mit Raspberry Pi GPIO18, SH_CP mit GPIO27, DS mit GPIO17 und parallele Ausgangsanschlüsse mit 8 Segmenten der LED-Segmentanzeige. Geben Sie Daten in den DS-Pin in das Schieberegister ein, wenn sich SH_{CP} (der Takteingang des Schieberegisters) an der ansteigenden Flanke befindet, und in das Speicherregister, wenn sich ST_{CP} (der Takteingang des Speichers) an der ansteigenden Flanke befindet. Anschließend können Sie die Zustände von SH_{CP} und ST_{CP} über die Raspberry Pi-GPIOs steuern, um die serielle Dateneingabe in eine parallele Datenausgabe umzuwandeln, um Raspberry Pi-GPIOs zu speichern und die Anzeige zu steuern.

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie in den Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/1.1.4/
```

Schritt 3: Kompilieren.

```
gcc 1.1.4_7-Segment.c -lwiringPi
```

Schritt 4: Führen Sie die obige ausführbare Datei aus.

```
sudo ./a.out
```

Nachdem der Code ausgeführt wurde, wird die 7-Segment-Anzeige 0-9, AF angezeigt.

Kode

```
#include <wiringPi.h>
#include <stdio.h>
#define SDI 0 //serial data input
#define RCLK 1 //memory clock input(STCP)
#define SRCLK 2 //shift register clock input(SHCP)
unsigned char SegCode[16] =
{0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

void init(void){
    pinMode(SDI, OUTPUT);
    pinMode(RCLK, OUTPUT);
    pinMode(SRCLK, OUTPUT);
    digitalWrite(SDI, 0);
    digitalWrite(RCLK, 0);
    digitalWrite(SRCLK, 0);
}

void hc595_shift(unsigned char dat){
    int i;
    for(i=0;i<8;i++){
        digitalWrite(SDI, 0x80 & (dat << i));
        digitalWrite(SRCLK, 1);
        delay(1);
        digitalWrite(SRCLK, 0);
    }
    digitalWrite(RCLK, 1);
    delay(1);
    digitalWrite(RCLK, 0);
}
```

```

int main(void){
    int i;
    if(wiringPiSetup() == -1){ //when initialize wiring failed, print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    init();
    while(1){
        for(i=0;i<16;i++){
            printf("Print %1X on Segment\n", i); // %X means hex output
            hc595_shift(SegCode[i]);
            delay(500);
        }
    }
    return 0;
}

```

Kode Erklärung

```

unsigned char SegCode[16] =
{0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

```

Ein Segmentcode-Array von 0 bis F in hexadezimaler Darstellung (gemeinsame Kathode).

```

void init(void){
    pinMode(SDI, OUTPUT);
    pinMode(RCLK, OUTPUT);
    pinMode(SRCLK, OUTPUT);
    digitalWrite(SDI, 0);
    digitalWrite(RCLK, 0);
    digitalWrite(SRCLK, 0);
}

```

Setzen Sie ds, st_cp, sh_cp drei Pins auf OUTPUT und den Anfangszustand auf 0.

```
void hc595_shift(unsigned char dat){}
```

Zuweisen eines 8-Bit-Werts zum Schieberegister des 74HC595.

```
digitalWrite(SDI, 0x80 & (dat << i));
```

Ordnen Sie die Datendaten SDI (DS) in Bits zu. Hier nehmen wir an, dass dat = 0x3f (0011 1111, wenn i = 2, 0x3f 2 Bits nach links (<<) verschiebt. 1111 1100 (0x3f << 2) & 1000 0000 (0x80) = 1000 0000 ist wahr.

```
digitalWrite(SRCLK, 1);
```

Der Anfangswert von SRCLK wurde auf 0 gesetzt, und hier wird er auf 1 gesetzt, um einen Anstiegsflankenimpuls zu erzeugen und dann das DS-Datum in das Schieberegister zu verschieben.

```
digitalWrite(RCLK, 1);
```

Der Anfangswert von RCLK wurde auf 0 gesetzt, und hier wird er auf 1 gesetzt, um eine ansteigende Flanke zu erzeugen und dann Daten vom Schieberegister zum Speicherregister zu verschieben.

```
while(1){  
    for(i=0;i<16;i++){  
        printf("Print %1X on Segment\n", i); // %X means hex output  
        hc595_shift(SegCode[i]);  
        delay(500);  
    }  
}
```

In dieser for Schleife verwenden wir "%1X" , um i als Hexadezimalzahl auszugeben. Wenden Sie i an, um den entsprechenden Segmentkode im Segkode [] -Array zu finden, und verwenden Sie hc595_shift (), um den Segkode in das Schieberegister des 74HC595 zu übergeben.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie in den Ordner der Kodeoo.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Ausführen.

```
sudo python3 1.1.4_7-Segment.py
```

Nachdem der Code ausgeführt wurde, wird die 7-Segment-Anzeige 0-9, AF angezeigt.

Kode

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

```

# Set up pins
SDI    = 17
RCLK   = 18
SRCLK = 27

# Define a segment code from 0 to F in Hexadecimal
segCode = [0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71]

def setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(SDI, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(RCLK, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(SRCLK, GPIO.OUT, initial=GPIO.LOW)

# Shift the data to 74HC595
def hc595_shift(dat):
    for bit in range(0, 8):
        GPIO.output(SDI, 0x80 & (dat << bit))
        GPIO.output(SRCLK, GPIO.HIGH)
        time.sleep(0.001)
        GPIO.output(SRCLK, GPIO.LOW)
        GPIO.output(RCLK, GPIO.HIGH)
        time.sleep(0.001)
        GPIO.output(RCLK, GPIO.LOW)

def main():
    while True:
        # Shift the code one by one from segCode list
        for code in segCode:
            hc595_shift(code)
            print ("segCode[%s]: 0x%02X"%(segCode.index(code), code)) # %02X means
double digit HEX to print
            time.sleep(0.5)

def destroy():
    GPIO.cleanup()

if __name__ == '__main__':
    setup()
    try:

```

```
main()
except KeyboardInterrupt:
    destroy()
```

Kode Erklärung

```
segCode = [0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71]
```

Ein Segmentcode-Array von 0 bis F in hexadezimaler Darstellung (gemeinsame Kathode).

```
def setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(SDI, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(RCLK, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(SRCLK, GPIO.OUT, initial=GPIO.LOW)
```

Setzen Sie ds, st_cp, sh_cp auf drei Pins und den Ausgangszustand auf niedrigen Niveau.

```
GPIO.output(SDI, 0x80 & (dat << bit))
```

Ordnen Sie die Datendaten SDI (DS) in Bits zu. Hier nehmen wir an, dass dat = 0x3f (0011 1111, wenn Bit = 2, 0x3f 2 Bits nach rechts (<<) verschiebt. 1111 1100 (0x3f << 2) & 1000 0000 (0x80) = 1000 0000 ist wahr.

```
GPIO.output(SRCLK, GPIO.HIGH)
```

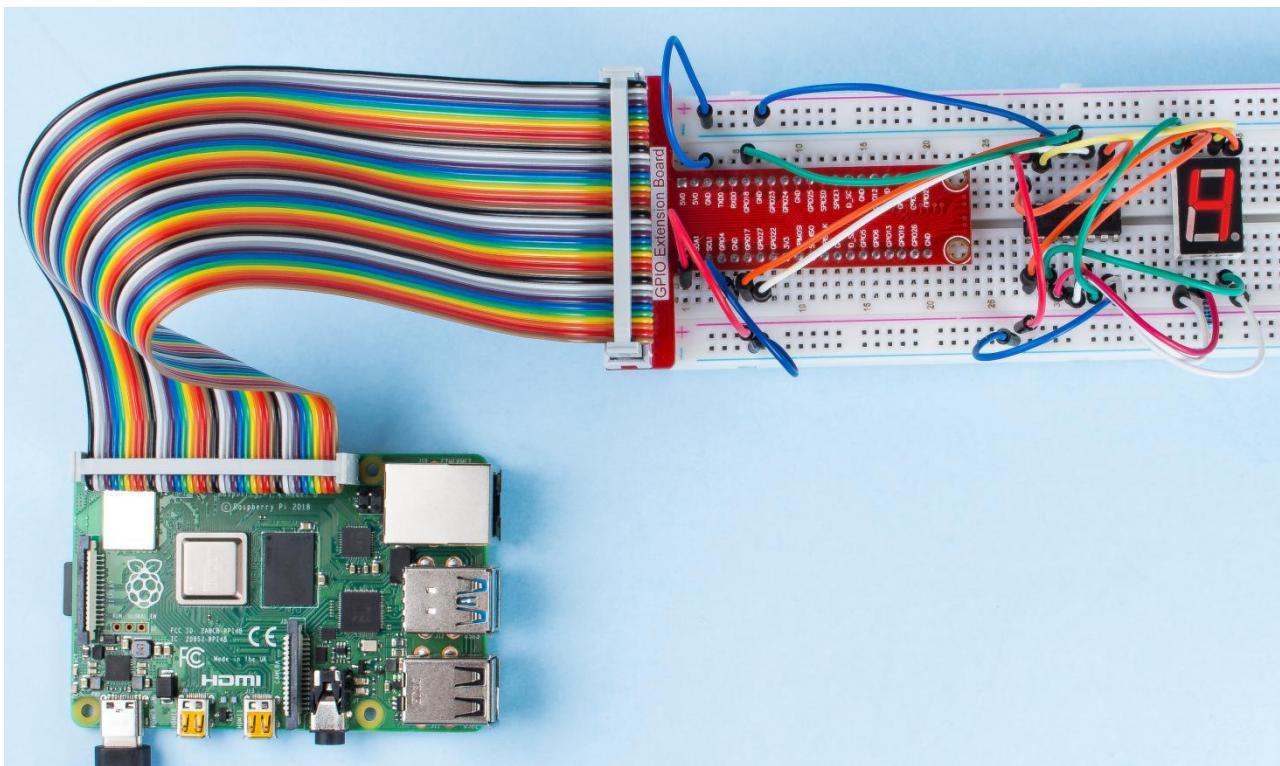
Der Anfangswert von SRCLK war auf NIEDRIG gesetzt, und hier wird er auf HIGH gesetzt, um einen ansteigenden Flankenimpuls zu erzeugen und dann das DS-Datum in das Schieberegister zu verschieben.

```
GPIO.output(RCLK, GPIO.HIGH)
```

Der Anfangswert von RCLK wurde auf NIEDRIG gesetzt, und hier wird er auf HOCH gesetzt, um eine ansteigende Flanke zu erzeugen und dann Daten vom Schieberegister zum Speicherregister zu verschieben.

Hinweis: Das hexadezimale Format der Nummer 0 bis 15 ist (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

Phänomen Bild



1.1.5 4-stellige 7-Segment-Anzeige

Einführung

Folgen Sie mir als Nächstes, um zu versuchen, die 4-stellige 7-Segment-Anzeige zu steuern.

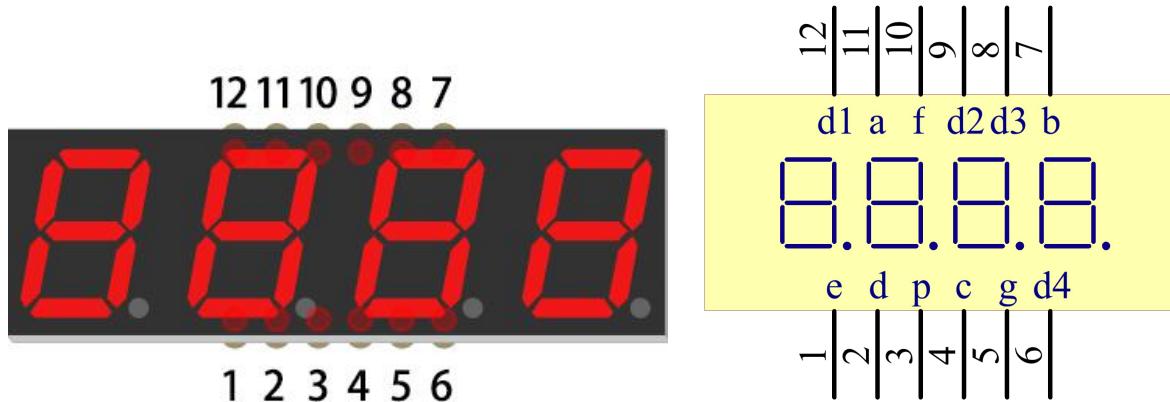
Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * 4-stellige 7-Segment-Anzeige
	 GPIO Extension Board Pinout: 3V3, SDA1, GND, SCL1, GPIO4, TXD0, GND, GPIO10, RXD0, GND, GPIO18, GPIO17, GND, GPIO23, GPIO22, GPIO24, 3V3, GND, SPI MOSI, GND, SPI MISO, GPIO25, SPI SCLK, SPI CE0, GND, ID_SD, ID_SC, GND, GPIO16, GND, GPIO13, GND, GPIO19, GPIO16, GND, GPIO26, GPIO16, GND, GPIO21, GND	
1 * 40-Pin Kabel		1 * 74HC595
1 * Steckbrett		Mehrere Überbrückungsdrähte

Prinzip

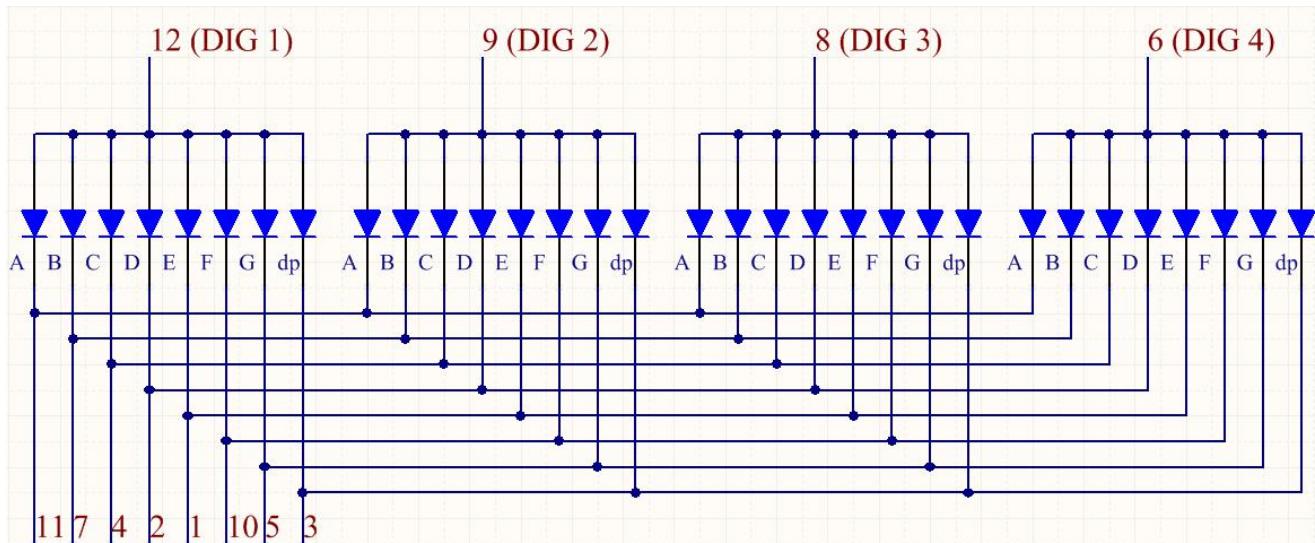
4-stellige 7-Segment-Anzeige

Die 4-stellige 7-Segment-Anzeige besteht aus vier 7-Segment-Anzeigen, die zusammenarbeiten.



Das 4-Digital-7-Segment-Display arbeitet unabhängig. Es verwendet das Prinzip der menschlichen visuellen Persistenz, um die Zeichen jedes 7-Segments schnell in einer Schleife anzuzeigen und fortlaufende Zeichenfolgen zu bilden.

Wenn beispielsweise "1234" auf dem Display angezeigt wird, wird "1" auf dem ersten 7-Segment angezeigt und "234" wird nicht angezeigt. Nach einer gewissen Zeit zeigt das zweite 7-Segment "2", das 1. 3. 4. des 7-Segments zeigt nicht usw. Die vier Digitalanzeigen werden nacheinander angezeigt. Dieser Vorgang ist sehr kurz (normalerweise 5 ms), und aufgrund des optischen Nachleuchteffekts und des Prinzips der visuellen Rückstände können wir vier Zeichen gleichzeitig sehen.



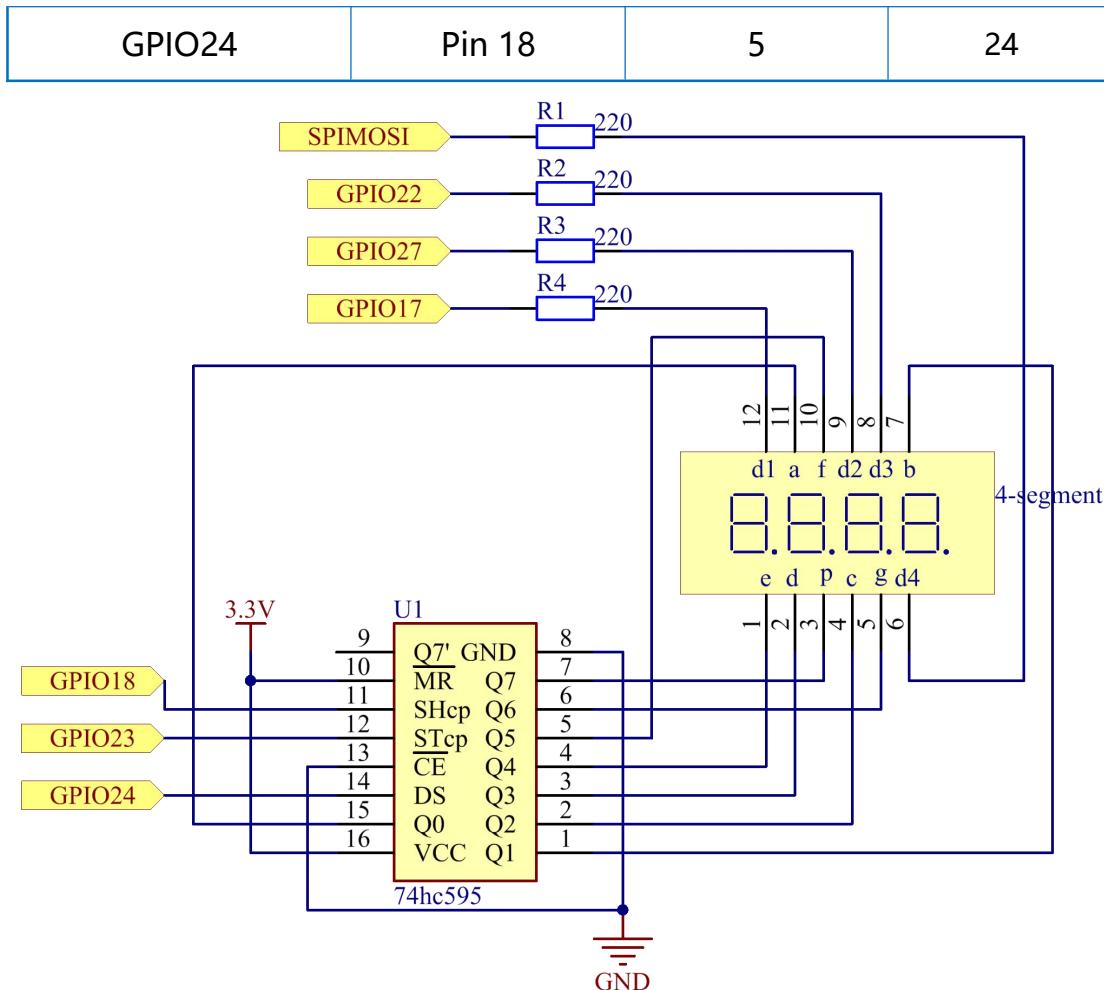
Koden anzeigen

Um Ihnen zu zeigen, wie 7-Segment-Anzeigen (Gemeinsame Anode) Nummer anzeigen, haben wir die folgende Tabelle gezeichnet. Nummer sind die Nummer 0-F, die auf der 7-Segment-Anzeige angezeigt werden. (DP) GFEDCBA bezieht sich auf die entsprechende LED, die auf 0 oder 1 gesetzt ist. Beispielsweise bedeutet 11000000, dass DP und G auf 1 gesetzt sind, während andere auf 0 gesetzt sind. Daher wird die Nummer 0 auf dem 7-Segment-Display angezeigt, während der HEX-Kode der Hexadezimalzahl entspricht.

Nummer	Gemeinsame Anode		Nummer	Gemeinsame Anode	
	(DP) GFEDCBA	Hex-Code		(DP) GFEDCBA	Hex-Code
0	11000000	0xc0	A	10001000	0x88
1	11111001	0xf9	B	10000011	0x83
2	10100100	0xa4	C	11000110	0xc6
3	10110000	0xb0	D	10100001	0xa1
4	10011001	0x99	E	10000110	0x86
5	10010010	0x92	F	10001110	0x8e
6	10000010	0x82	.	01111111	0x7f
7	11111000	0xf8			
8	10000000	0x80			
9	10010000	0x90			

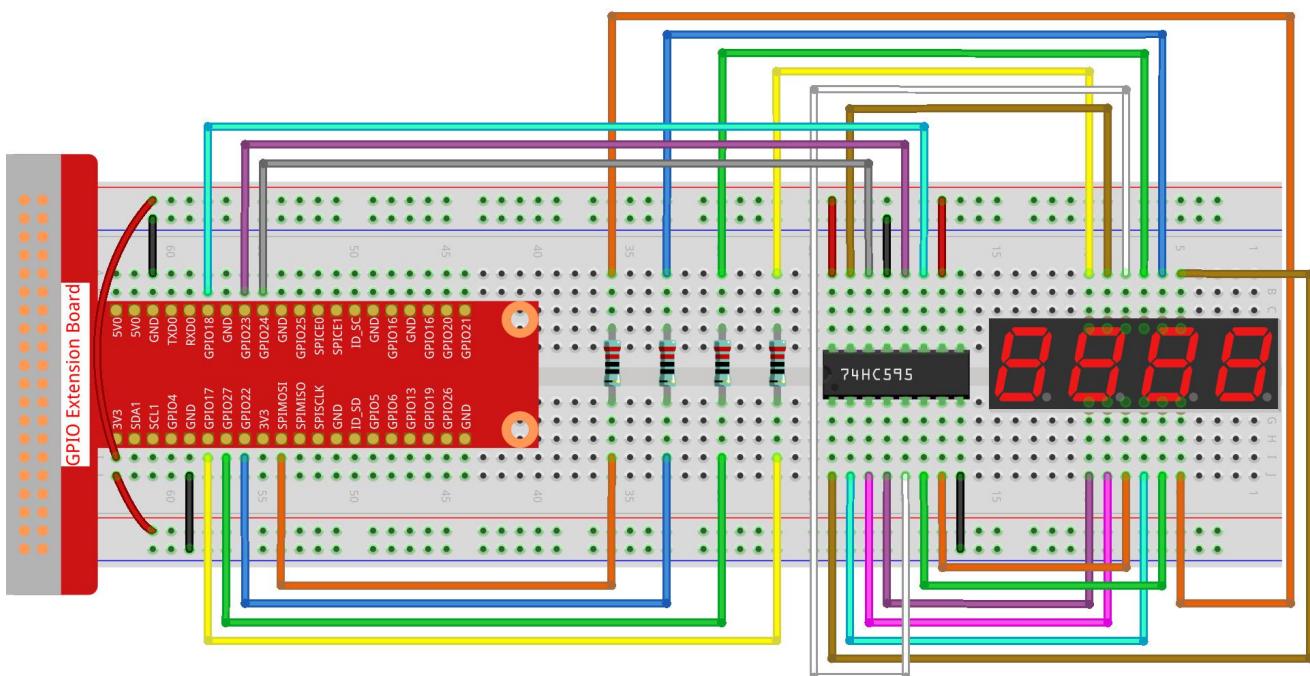
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22
SPIMOSI	Pin 19	12	10
GPIO18	Pin 12	1	18
GPIO23	Pin 16	4	23



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner des Codes.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/1.1.5/
```

Schritt 3: Kompilieren Sie den Code.

```
gcc 1.1.5_4-Digit.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Nachdem der Code ausgeführt wurde, nimmt das Programm eine Zählung vor, die um 1 pro Sekunde erhöht wird, und die 4-stellige 7-Segment-Anzeige zeigt die Zählung an.

Kode

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

#define SDI 5
#define RCLK 4
#define SRCLK 1

const int placePin[] = {12, 3, 2, 0};
unsigned char number[] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90};

int counter = 0;

void pickDigit(int digit)
{
    for (int i = 0; i < 4; i++)
    {
        digitalWrite(placePin[i], 0);
    }
    digitalWrite(placePin[digit], 1);
}

void hc595_shift(int8_t data)
```

```
{  
    int i;  
    for (i = 0; i < 8; i++)  
    {  
        digitalWrite(SDI, 0x80 & (data << i));  
        digitalWrite(SRCLK, 1);  
        delayMicroseconds(1);  
        digitalWrite(SRCLK, 0);  
    }  
    digitalWrite(RCLK, 1);  
    delayMicroseconds(1);  
    digitalWrite(RCLK, 0);  
}  
  
void clearDisplay()  
{  
    int i;  
    for (i = 0; i < 8; i++)  
    {  
        digitalWrite(SDI, 1);  
        digitalWrite(SRCLK, 1);  
        delayMicroseconds(1);  
        digitalWrite(SRCLK, 0);  
    }  
    digitalWrite(RCLK, 1);  
    delayMicroseconds(1);  
    digitalWrite(RCLK, 0);  
}  
  
void loop()  
{  
    while(1){  
        clearDisplay();  
        pickDigit(0);  
        hc595_shift(number[counter % 10]);  
  
        clearDisplay();  
        pickDigit(1);  
        hc595_shift(number[counter % 100 / 10]);  
    }  
}
```

```

clearDisplay();
pickDigit(2);
hc595_shift(number[counter % 1000 / 100]);

clearDisplay();
pickDigit(3);
hc595_shift(number[counter % 10000 / 1000]);
}

void timer(int timer1)
{
    if (timer1 == SIGALRM)
    {
        counter++;
        alarm(1);
        printf("%d\n", counter);
    }
}

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

    pinMode(SDI, OUTPUT);
    pinMode(RCLK, OUTPUT);
    pinMode(SRCLK, OUTPUT);

    for (int i = 0; i < 4; i++)
    {
        pinMode(placePin[i], OUTPUT);
        digitalWrite(placePin[i], HIGH);
    }

    signal(SIGALRM, timer);
    alarm(1);
    loop();
}

```

Kode Erklärung

```
const int placePin[] = {12, 3, 2, 0};
```

Diese vier Pins steuern die gemeinsamen Anodenpins der vierstelligen 7-Segment-Anzeigen.

```
unsigned char number[] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90};
```

Ein Segmentkode-Array von 0 bis 9 in hexadezimaler Darstellung (gemeinsame Anode).

```
void pickDigit(int digit)
{
    for (int i = 0; i < 4; i++)
    {
        digitalWrite(placePin[i], 0);
    }
    digitalWrite(placePin[digit], 1);
}
```

Wählen Sie den Ort des Wertes. Es gibt nur einen Ort, der jedes Mal aktiviert werden sollte. Der aktivierte Ort wird hoch geschrieben.

```
void loop()
{
    while(1){
        clearDisplay();
        pickDigit(0);
        hc595_shift(number[counter % 10]);

        clearDisplay();
        pickDigit(1);
        hc595_shift(number[counter % 100 / 10]);

        clearDisplay();
        pickDigit(2);
        hc595_shift(number[counter % 1000 / 100]);

        clearDisplay();
        pickDigit(3);
        hc595_shift(number[counter % 10000 / 1000]);
    }
}
```

```
}
```

Mit dieser Funktion wird die auf der 4-stelligen 7-Segment-Anzeige angezeigte Nummer eingestellt.

Klare Anzeige (): Schreiben Sie in 11111111, um diese acht LEDs auf der 7-Segment-Anzeige auszuschalten und den angezeigten Inhalt zu löschen.

pickDigit (0): Wählen Sie die vierte 7-Segment-Anzeige.

hc595_shift (Nummer [Zähler% 10]): Die Nummer in der einzelnen Ziffer des Zählers wird im vierten Segment angezeigt.

```
signal(SIGALRM, timer);
```

Dies ist eine vom System bereitgestellte Funktion. Der Prototyp der Kode lautet:

```
sig_t signal(int signum,sig_t handler);
```

Nach dem Ausführen des Signals () hält der Prozess, sobald er das entsprechende Signal (in diesem Fall SIGALRM) erhalten hat, die vorhandene Aufgabe sofort an und verarbeitet die eingestellte Funktion (in diesem Fall Timer (sig)).

```
alarm(1);
```

Dies ist auch eine vom System bereitgestellte Funktion. Der Kode-Prototyp ist

```
unsigned int alarm (unsigned int seconds);
```

Es erzeugt nach einer bestimmten Anzahl von Sekunden ein SIGALRM-Signal.

```
void timer(int timer1)
{
    if (timer1 == SIGALRM)
    {
        counter++;
        alarm(1);
        printf("%d\n", counter);
    }
}
```

Wir verwenden die obigen Funktionen, um die Timer-Funktion zu implementieren.

Nachdem der Alarm () das SIGALRM-Signal erzeugt hat, wird die Timer-Funktion aufgerufen. Addiere 1 zum Zähler und die Funktion Warnung (1) wird nach 1 Sekunde wiederholt aufgerufen.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 1.1.5_4-Digit.py
```

Nachdem die Kode ausgeführt wurde, nimmt das Programm eine Zählung vor, die um 1 pro Sekunde erhöht wird, und die 4-stellige Anzeige zeigt die Zählung an.

Kode

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

SDI = 24
RCLK = 23
SRCLK = 18

placePin = (10, 22, 27, 17)
number = (0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90)

counter = 0
timer1 = 0

def clearDisplay():
    for i in range(8):
        GPIO.output(SDI, 1)
        GPIO.output(SRCLK, GPIO.HIGH)
        GPIO.output(SRCLK, GPIO.LOW)
        GPIO.output(RCLK, GPIO.HIGH)
        GPIO.output(RCLK, GPIO.LOW)

def hc595_shift(data):
    for i in range(8):
        GPIO.output(SDI, 0x80 & (data << i))
        GPIO.output(SRCLK, GPIO.HIGH)
        GPIO.output(SRCLK, GPIO.LOW)
        GPIO.output(RCLK, GPIO.HIGH)
        GPIO.output(RCLK, GPIO.LOW)
```

```
def pickDigit(digit):
    for i in placePin:
        GPIO.output(i,GPIO.LOW)
        GPIO.output(placePin[digit], GPIO.HIGH)

def timer():
    global counter
    global timer1
    timer1 = threading.Timer(1.0, timer)
    timer1.start()
    counter += 1
    print("%d" % counter)

def loop():
    global counter
    while True:
        clearDisplay()
        pickDigit(0)
        hc595_shift(number[counter % 10])

        clearDisplay()
        pickDigit(1)
        hc595_shift(number[counter % 100//10])

        clearDisplay()
        pickDigit(2)
        hc595_shift(number[counter % 1000//100])

        clearDisplay()
        pickDigit(3)
        hc595_shift(number[counter % 10000//1000])

def setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(SDI, GPIO.OUT)
    GPIO.setup(RCLK, GPIO.OUT)
    GPIO.setup(SRCLK, GPIO.OUT)
    for i in placePin:
        GPIO.setup(i, GPIO.OUT)
```

```

global timer1
timer1 = threading.Timer(1.0, timer)
timer1.start()

def destroy():  # When "Ctrl+C" is pressed, the function is executed.
    global timer1
    GPIO.cleanup()
    timer1.cancel() # cancel the timer

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

```

Kode Erklärung

placePin = (10, 22, 27, 17)

Diese vier Pins steuern die gemeinsamen Anodenpins der vierstelligen 7-Segment-Anzeigen.

number = (0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90)

Ein Segmentkode-Array von 0 bis 9 in hexadezimaler Darstellung (gemeinsame Anode).

```

def clearDisplay():
    for i in range(8):
        GPIO.output(SDI, 1)
        GPIO.output(SRCLK, GPIO.HIGH)
        GPIO.output(SRCLK, GPIO.LOW)
        GPIO.output(RCLK, GPIO.HIGH)
        GPIO.output(RCLK, GPIO.LOW)

```

Schreiben Sie achtmal "1" in SDI., Damit die acht LEDs auf dem 7-Segment-Anzeige erlöschen, um den angezeigten Inhalt zu löschen.

```

def pickDigit(digit):
    for i in placePin:

```

```
GPIO.output(i,GPIO.LOW)
GPIO.output(placePin[digit], GPIO.HIGH)
```

Wählen Sie den Ort des Wertes. Es gibt nur einen Ort, der jedes Mal aktiviert werden sollte. Der aktivierte Ort wird hoch geschrieben.

```
def loop():
    global counter
    while True:
        clearDisplay()
        pickDigit(0)
        hc595_shift(number[counter % 10])

        clearDisplay()
        pickDigit(1)
        hc595_shift(number[counter % 100//10])

        clearDisplay()
        pickDigit(2)
        hc595_shift(number[counter % 1000//100])

        clearDisplay()
        pickDigit(3)
        hc595_shift(number[counter % 10000//1000])
```

Mit dieser Funktion wird die auf der 4-stelligen 7-Segment-Anzeige angezeigte Nummer eingestellt.

Starten Sie zuerst die vierte Segmentanzeige und schreiben Sie die einstellige Nummer. Dann starten Sie die Anzeige des dritten Segments und geben Sie die Zehnerstelle ein. Starten Sie danach die zweite bzw. die erste Segmentanzeige und schreiben Sie die Hunderter- bzw. Tausenderstellen. Da die Aktualisierungsgeschwindigkeit sehr hoch ist, sehen wir eine vollständige vierstellige Anzeige.

```
timer1 = threading.Timer(1.0, timer)
timer1.start()
```

Das Modul Threading ist das übliche Threading-Modul in Python Tim und Timer ist die Unterklasse davon.

Der Prototyp der Kode ist:

```
class threading.Timer(interval, function, args=[], kwargs={})
```

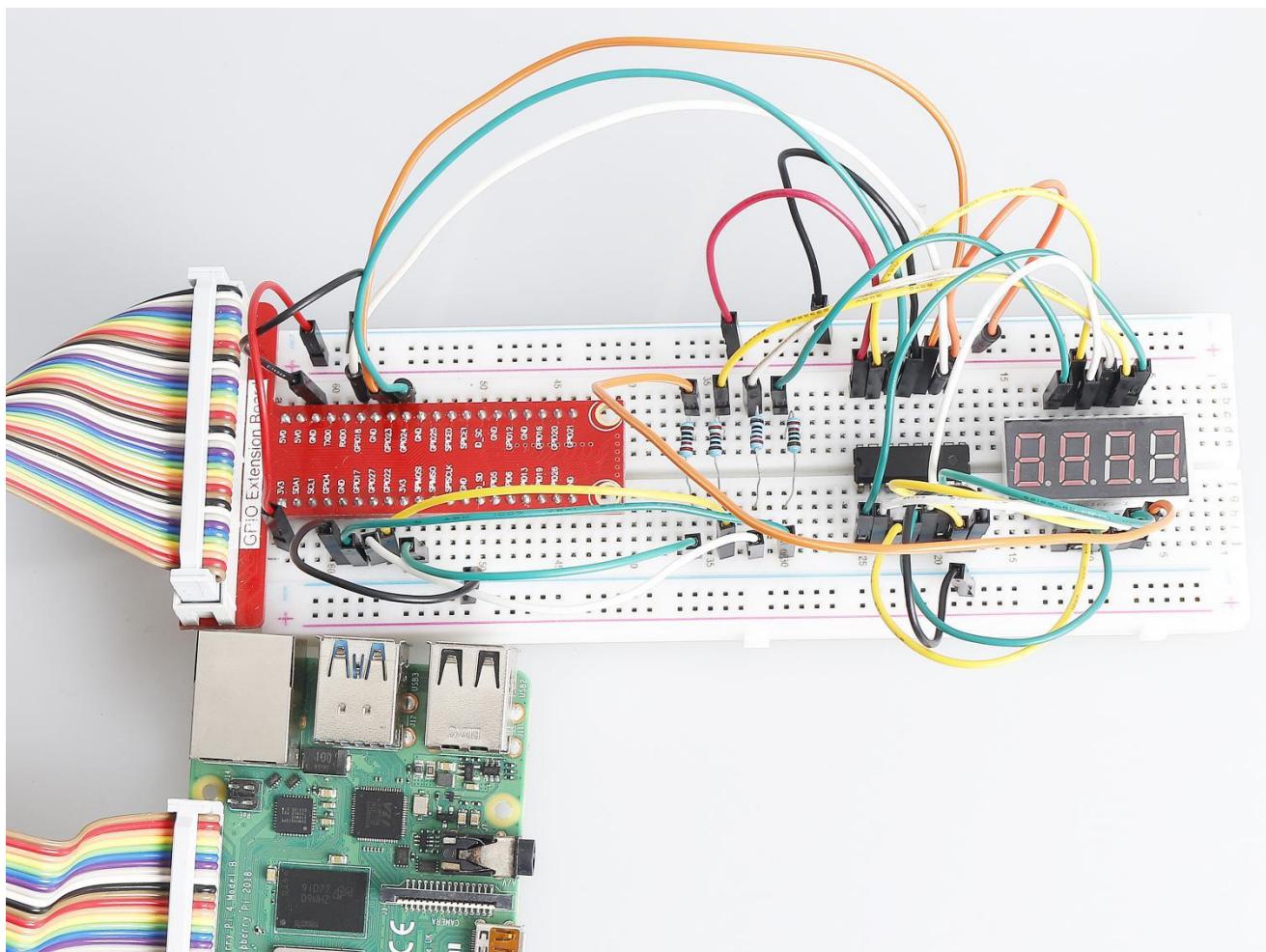
Nach dem Intervall wird die Funktion ausgeführt. Hier beträgt das Intervall 1.0, und die Funktion ist timer () .

start () bedeutet, dass der Timer an diesem Punkt startet.

```
def timer():
    global counter
    global timer1
    timer1 = threading.Timer(1.0, timer)
    timer1.start()
    counter += 1
    print("%d" % counter)
```

Nachdem der Timer 1,0 Sekunden erreicht hat, wird die Timer-Funktion aufgerufen. Addiere 1 zum Zähler und der Timer wird erneut verwendet, um sich jede Sekunde wiederholt auszuführen.

Phänomen Bild



1.1.6 LED-Punktmatrix

Einführung

Wie der Name schon sagt, eine LED-Punktmatrix ist eine Matrix aus LEDs. Das Aufleuchten und Dimmen der LEDs formuliert unterschiedliche Zeichen und Muster.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * LED-Punktmatrix																																								
	 Pinout diagram: <table border="1"><tr><td>3V3</td><td>5V</td></tr><tr><td>SDA1</td><td>GND</td></tr><tr><td>SCL1</td><td></td></tr><tr><td>GPIO4</td><td>TxD0</td></tr><tr><td>GND</td><td>RxD0</td></tr><tr><td>GPIO17</td><td>GPIO18</td></tr><tr><td>GPIO27</td><td>GND</td></tr><tr><td>GPIO22</td><td>GPIO23</td></tr><tr><td>3V3</td><td>GPIO24</td></tr><tr><td>SPI_MOSI</td><td>GND</td></tr><tr><td>SPI_MISO</td><td>GPIO25</td></tr><tr><td>SPI_SCLK</td><td>SPICE0</td></tr><tr><td>GND</td><td>SPICE1</td></tr><tr><td>ID_SD</td><td>ID_SC</td></tr><tr><td>GPIO9</td><td>GND</td></tr><tr><td>GPIO6</td><td>GPIO16</td></tr><tr><td>GPIO13</td><td>GND</td></tr><tr><td>GPIO19</td><td>GPIO16</td></tr><tr><td>GPIO26</td><td>GPIO20</td></tr><tr><td>GND</td><td>GPIO21</td></tr></table>	3V3	5V	SDA1	GND	SCL1		GPIO4	TxD0	GND	RxD0	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GPIO24	SPI_MOSI	GND	SPI_MISO	GPIO25	SPI_SCLK	SPICE0	GND	SPICE1	ID_SD	ID_SC	GPIO9	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	
3V3	5V																																									
SDA1	GND																																									
SCL1																																										
GPIO4	TxD0																																									
GND	RxD0																																									
GPIO17	GPIO18																																									
GPIO27	GND																																									
GPIO22	GPIO23																																									
3V3	GPIO24																																									
SPI_MOSI	GND																																									
SPI_MISO	GPIO25																																									
SPI_SCLK	SPICE0																																									
GND	SPICE1																																									
ID_SD	ID_SC																																									
GPIO9	GND																																									
GPIO6	GPIO16																																									
GPIO13	GND																																									
GPIO19	GPIO16																																									
GPIO26	GPIO20																																									
GND	GPIO21																																									
1 * 40-Pin Kabel		Mehrere Überbrückungsdrähte																																								
1 * Steckbrett		2 * 74HC595																																								

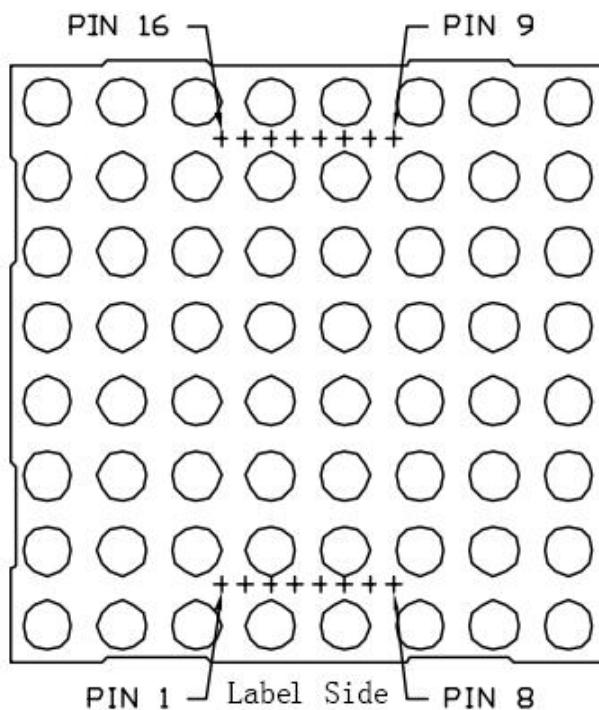
Prinzip

LED-Punktmatrix

Im Allgemeinen kann die LED-Punktmatrix in zwei Typen eingeteilt werden: gemeinsame Kathode (CC) und gemeinsame Anode (CA). Sie sehen sich sehr ähnlich, aber innerlich liegt der Unterschied. Sie können durch Test erkennen. In diesem Kit wird eine CA verwendet. Sie können 788BS an der Seite beschriftet sehen.

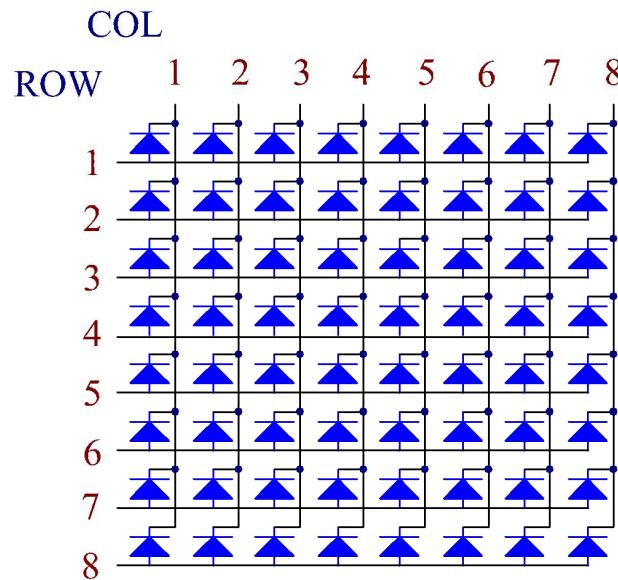
Siehe die Abbildung unten. Die Pins sind an den beiden Enden hinten angeordnet. Nehmen Sie die Etikettenseite als Referenz: Die Pins an diesem Ende sind Pin 1-8 und die anderen sind Pin 9-16.

Die Außenansicht:



Unten zeigen die Abbildungen ihre interne Struktur. Sie können in einer CA-LED-Punktmatrix sehen, dass ROW die Anode der LED darstellt und COL die Kathode ist. es ist das Gegenteil für einen CC. Eines ist gemeinsam: Für beide Typen sind Pin 13, 3, 4, 10, 6, 11, 15 und 16 alle COL, wenn Pin 9, 14, 8, 12, 1, 7, 2 und 5 alle COL sind REIHE. Wenn Sie die erste LED in der oberen linken Ecke einschalten möchten, setzen Sie für eine CA-LED-Punktmatrix einfach Pin 9 als High und Pin 13 als Low und für eine CC-Pin Pin 13 als High und Pin 9 als NIEDRIG . Wenn Sie die gesamte erste Spalte für CA aufleuchten möchten, setzen Sie Pin 13 auf Niedrig und REIHE 9, 14, 8, 12, 1, 7, 2 und 5 auf Hoch. Wenn Sie für CC Pin 13 auf Hoch und setzen REIHE 9, 14, 8, 12, 1, 7, 2 und 5 als niedrig. Betrachten Sie die folgenden Abbildungen zum besseren Verständnis.

Die interne Ansicht:



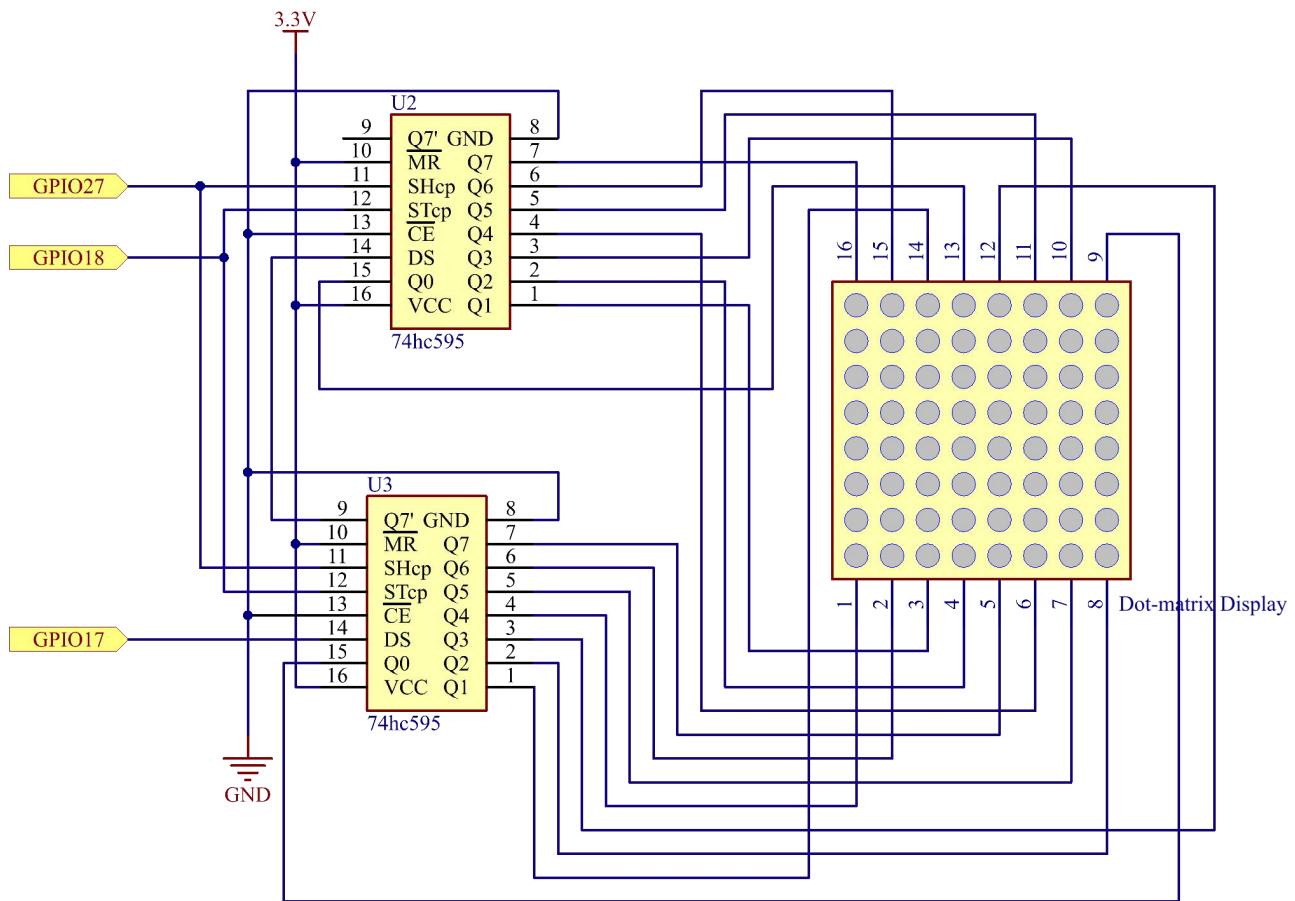
Pin-Nummerierung entsprechend den obigen Zeilen und Spalten:

MIT DEM	1	2	3	4	5	6	7	8
REIHE	13	3	4	10	6	11	15	16
Pin Nr.	1	2	3	4	5	6	7	8
Pin Nr.	9	14	8	12	1	7	2	5

Zusätzlich werden hier zwei 74HC595-Chips verwendet. Eine besteht darin, die Zeilen der LED-Punktmatrix zu steuern, während die andere die Spalten steuert.

Schematische Darstellung

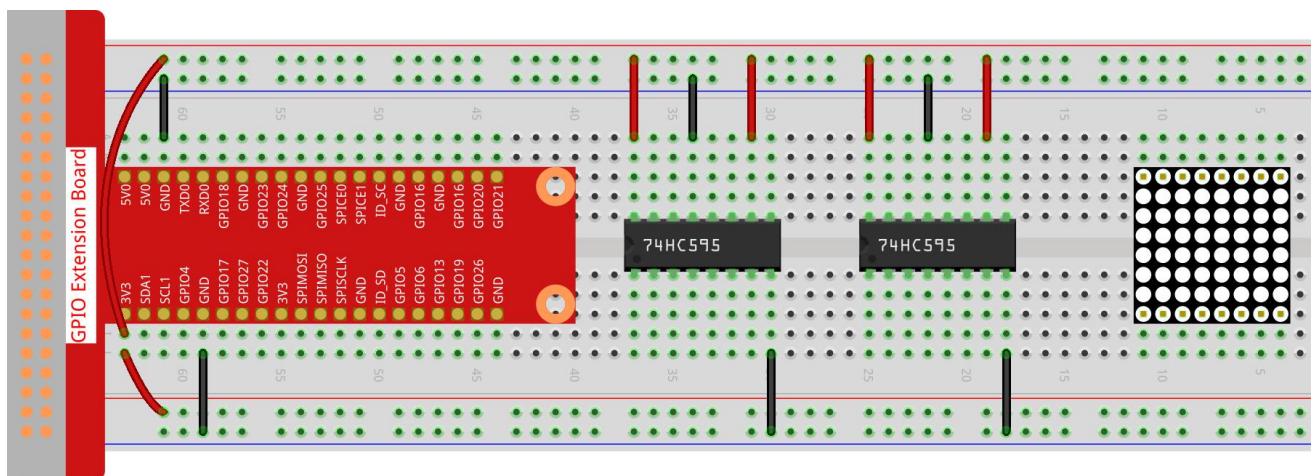
T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27



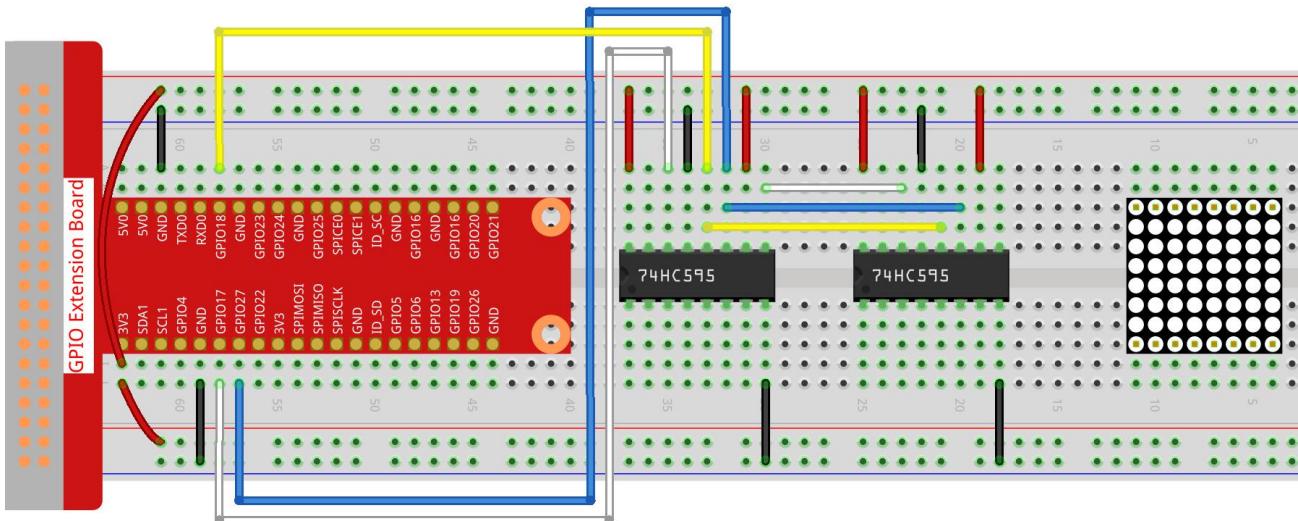
Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf. Da die Verkabelung kompliziert ist, machen wir es Schritt für Schritt. Setzen Sie zuerst den T-Cobbler, die LED-Punktmatrix und zwei 74HC595-Chips in das Steckbrett ein. Verbinden Sie die 3,3 V und GND des T-Cobbler mit den Löchern auf den beiden Seiten der Platine und schließen Sie dann Pin 16 und 10 der beiden 74HC595-Chips an VCC, Pin 13 und Pin 8 an GND an.

Hinweis: Im Fritzing-Bild oben befindet sich die Seite mit der Beschriftung unten.

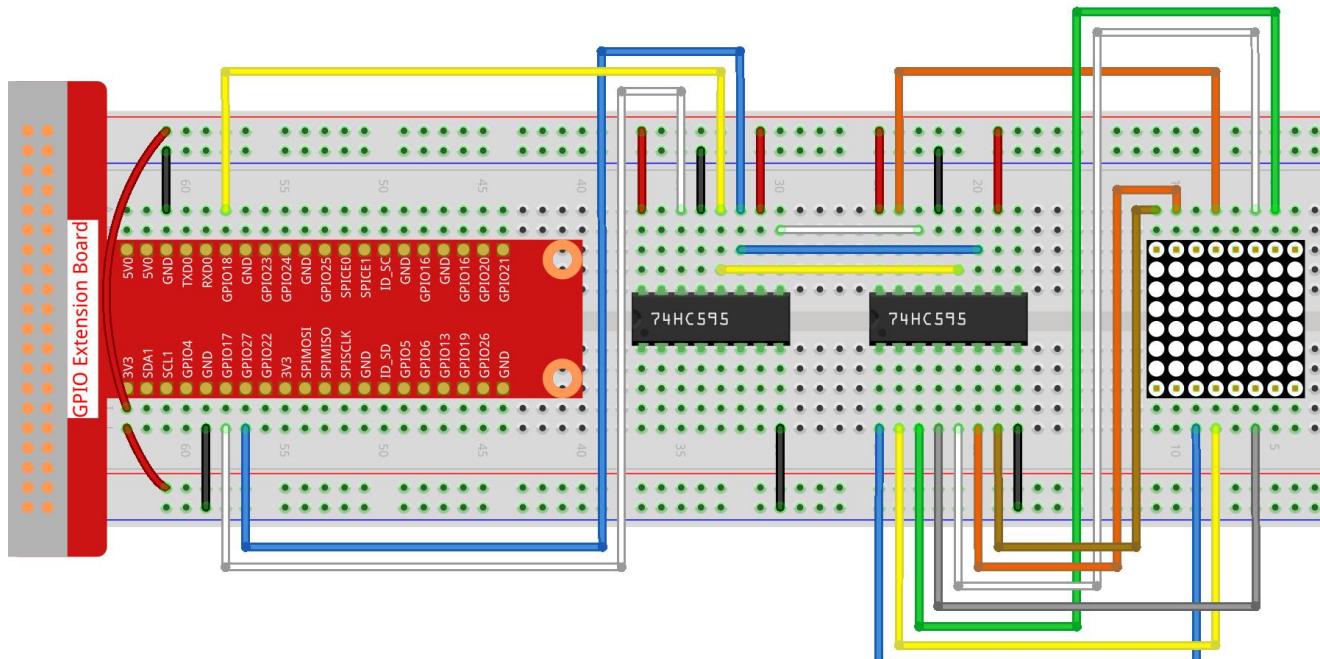


Schritt 2: Verbinden Sie Pin 11 der beiden 74HC595 miteinander und dann mit GPIO27. dann Pin 12 der beiden Chips und GPIO18; Als nächstes Pin 14 des 74HC595 auf der linken Seite an GPIO17 und Pin 9 an Pin 14 des zweiten 74HC595.



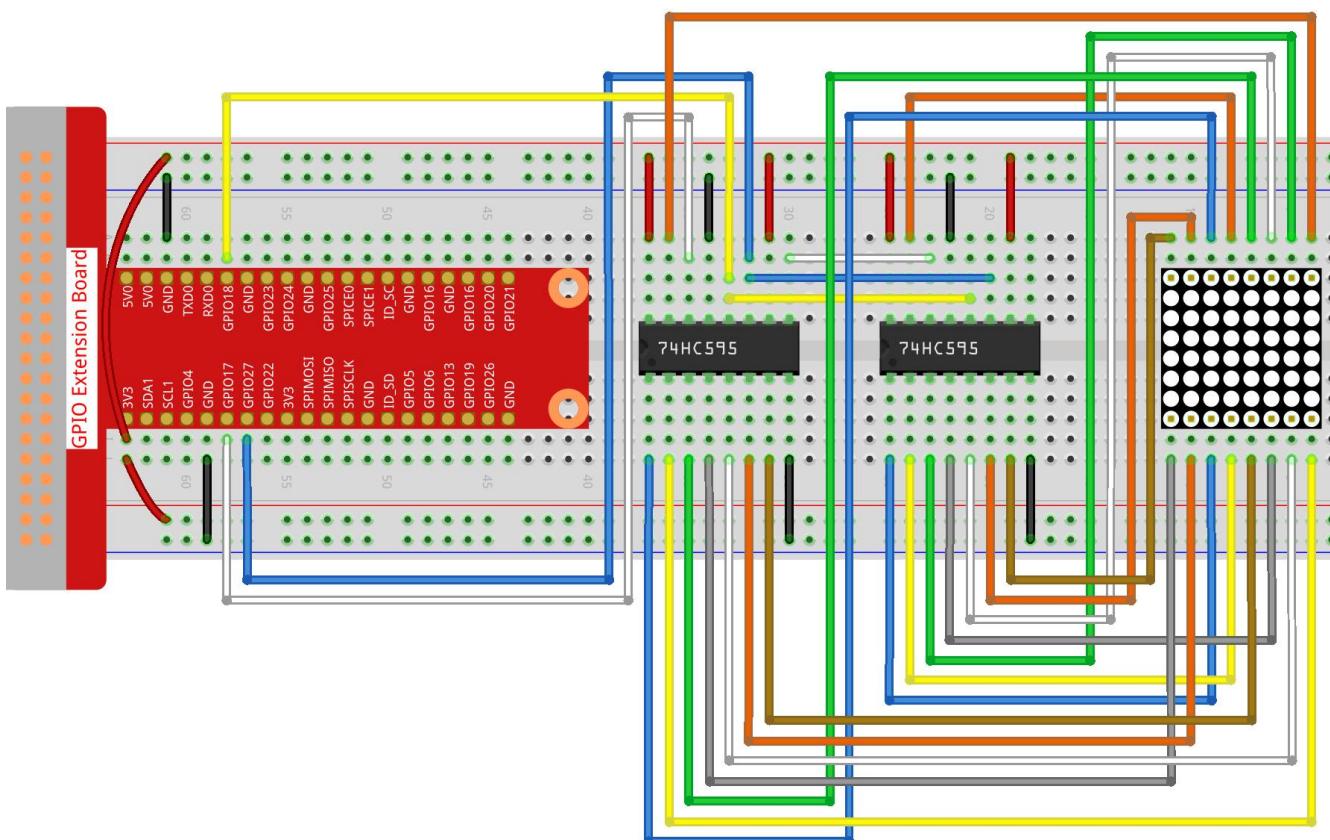
Schritt 3: Der 74HC595 auf der rechten Seite steuert die Spalten der LED-Punktmatrix. Die Zuordnung finden Sie in der folgenden Tabelle. Daher werden die Q0-Q7-Pins des 74HC595 mit Pin 13, 3, 4, 10, 6, 11, 15 bzw. 16 abgebildet.

74HC595	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
LED- Punktmatrix	13	3	4	10	6	11	15	16



Schritt 4: Verbinden Sie nun die REIHEN der LED-Punktmatrix. Der 74HC595 auf der linken Seite steuert die REIHE der LED-Punktmatrix. Die Zuordnung finden Sie in der folgenden Tabelle. Wir können sehen, dass Q0-Q7 des 74HC595 auf der linken Seite mit Pin 9, 14, 8, 12, 1, 7, 2 bzw. 5 abgebildet sind.

74HC595	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
LED-Punktmatrix	9	14	8	12	1	7	2	5



➤ Für Benutzer in C-Sprache

Schritt 5: Wechseln Sie in den Codeordner.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/1.1.6/
```

Schritt 6: Kompilieren.

```
gcc 1.1.6_LedMatrix.c -lwiringPi
```

Schritt 7: Ausführen.

```
sudo ./a.out
```

Nachdem die Kode ausgeführt wurde, leuchtet die LED-Punktmatrix Zeile für Zeile und Spalte für Spalte auf und aus.

Kode

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

#define SDI 0 //serial data input
#define RCLK 1 //memory clock input(STCP)
#define SRCLK 2 //shift register clock input(SHCP)

unsigned char code_H[20] =
{0x01,0xff,0x80,0xff,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0xff,0xff,0xff,0xff,0x
ff};

unsigned char code_L[20] =
{0x00,0x7f,0x00,0xfe,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
bf,0x7f};

void init(void){
    pinMode(SDI, OUTPUT);
    pinMode(RCLK, OUTPUT);
    pinMode(SRCLK, OUTPUT);

    digitalWrite(SDI, 0);
    digitalWrite(RCLK, 0);
    digitalWrite(SRCLK, 0);
}

void hc595_in(unsigned char dat){
    int i;
    for(i=0;i<8;i++){
        digitalWrite(SDI, 0x80 & (dat << i));
        digitalWrite(SRCLK, 1);
        delay(1);
        digitalWrite(SRCLK, 0);
    }
}

void hc595_out(){
    digitalWrite(RCLK, 1);
    delay(1);
    digitalWrite(RCLK, 0);
}
```

```

int main(void){
    int i;
    if(wiringPiSetup() == -1){ //when initialize wiring failed, print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    init();
    while(1){
        for(i=0;i<sizeof(code_H);i++){
            hc595_in(code_L[i]);
            hc595_in(code_H[i]);
            hc595_out();
            delay(100);
        }

        for(i[sizeof(code_H)];i>=0;i--){
            hc595_in(code_L[i]);
            hc595_in(code_H[i]);
            hc595_out();
            delay(100);
        }
    }

    return 0;
}

```

Kode Erklärung

```

unsigned char code_H[20] =
{0x01,0xff,0x80,0xff,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0xff,0xff,0xff,0xff,0x
ff};

unsigned char code_L[20] =
{0x00,0x7f,0x00,0xfe,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
bf,0x7f};

```

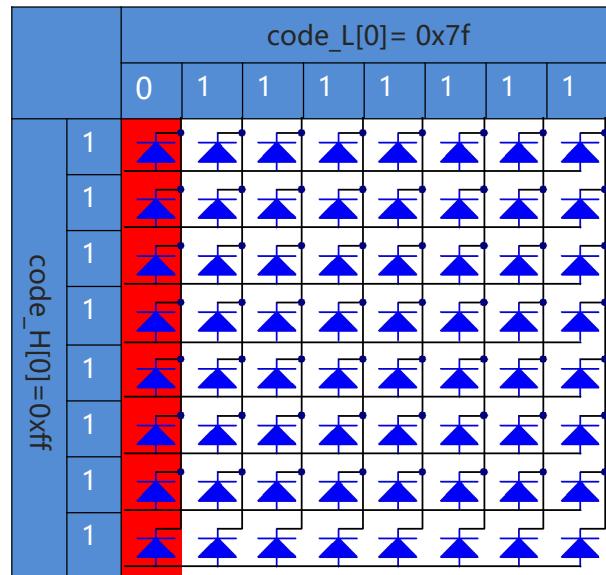
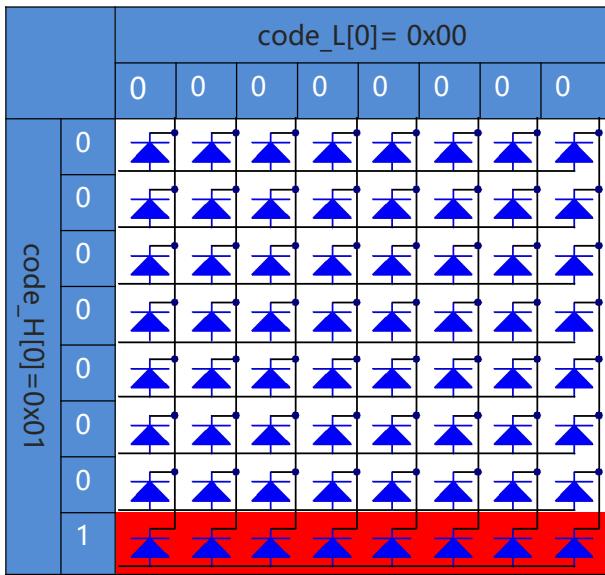
Der Array-Kode_H repräsentiert die Elemente der LED-Punktmatrixzeile, und der Array-Code_L bezieht sich auf die Elemente der Spalte. Wenn Zeichen angezeigt werden, werden ein Element in Zeile und eines in Spalte erfasst und den beiden HC595-Chips zugewiesen. Somit ist ein Muster auf der LED-Punktmatrix gezeigt.

Nehmen Sie als Beispiele die erste Nummer von code_H, 0x01 und die erste Nummer von code_L, 0x00.

0x01 konvertiert in Binär wird 00000001; 0x00 konvertiert in Binär wird 0000 0000.

In diesem Kit wird eine LED-Punktmatrixanzeige mit gemeinsamer Anode verwendet, sodass nur die acht LEDs in der achten Reihe aufleuchten.

Wenn die Bedingungen, dass Code_H 0xff und Code_L 0x7f ist, gleichzeitig erfüllt sind, leuchten diese 8 LEDs in der ersten Spalte.



```
void hc595_in(unsigned char dat){
    int i;
    for(i=0;i<8;i++){
        digitalWrite(SDI, 0x80 & (dat << i));
        digitalWrite(SRCLK, 1);
        delay(1);
        digitalWrite(SRCLK, 0);
```

Schreiben Sie den Wert von dat, um SDI des HC595 Stück für Stück anzuhängen. Der Anfangswert von SRCLK wurde auf 0 gesetzt, und hier wird er auf 1 gesetzt, um einen ansteigenden Flankenimpuls zu erzeugen und dann das PinSDI (DS) -Datum in das Schieberegister zu verschieben.

```
void hc595_out(){
    digitalWrite(RCLK, 1);
    delay(1);
    digitalWrite(RCLK, 0);
```

Der Anfangswert von RCLK wurde auf 0 gesetzt, und hier wird er auf 1 gesetzt, um eine ansteigende Flanke zu erzeugen und dann Daten vom Schieberegister zum Speicherregister zu verschieben.

```
while(1){  
    for(i=0;i<sizeof(code_H);i++){  
        hc595_in(code_L[i]);  
        hc595_in(code_H[i]);  
        hc595_out();  
        delay(100);  
    }  
}
```

In dieser Schleife werden diese 20 Elemente in den beiden Arrays Kode_L und code_H nacheinander auf die beiden 74HC595-Chips hochgeladen. Rufen Sie dann die Funktion hc595_out () auf, um Daten vom Schieberegister zum Speicherregister zu verschieben.

➤ Für Python-Sprachbenutzer

Schritt 5: Gehen Sie in den Kode-Ordner.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Schritt 6: Ausführen.

```
sudo python3 1.1.6_LedMatrix.py
```

Nachdem die Kode ausgeführt wurde, leuchtet die LED-Punktmatrix Zeile für Zeile und Spalte für Spalte auf und aus.

Kode

```
import RPi.GPIO as GPIO  
import time  
SDI = 17  
RCLK = 18  
SRCLK = 27  
  
# we use BX matrix, ROW for anode, and COL for cathode  
# ROW +--+  
code_H =  
[0x01,0xff,0x80,0xff,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0xff,0xff,0xff,0xff,0xff,0x  
ff]  
# COL ----
```



```
main()
except KeyboardInterrupt:
    destroy()
```

Kode Erklärung

```
code_H =
[0x01,0xff,0x80,0xff,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0xff,0xff,0xff,0xff,0x
ff]
code_L =
[0x00,0x7f,0x00,0xfe,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
bf,0x7f]
```

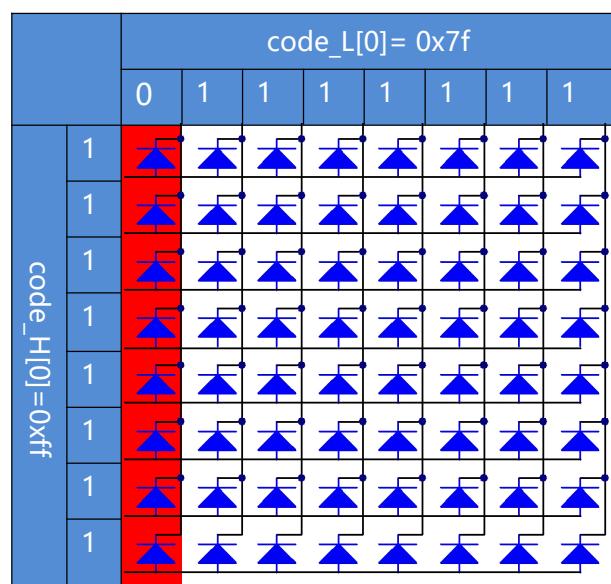
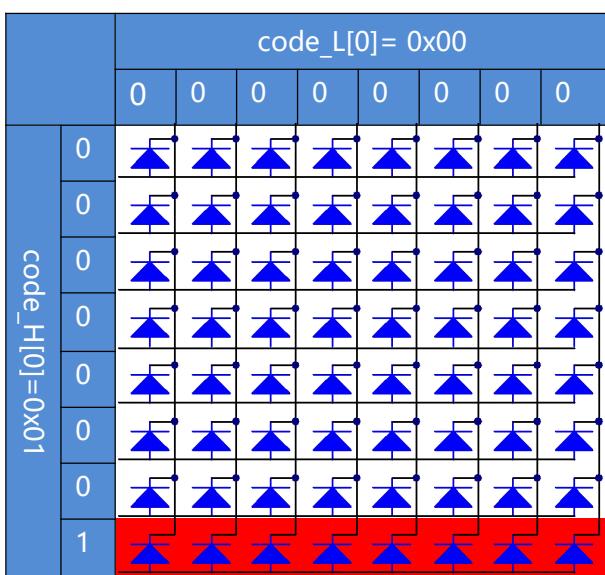
Das Array code_H repräsentiert die Elemente der Matrix-Zeile, und das Array code_L bezieht sich auf die Elemente der Spalte. Wenn Zeichen angezeigt werden, werden ein Element in Zeile und eines in Spalte erfasst und den beiden HC595-Chips zugewiesen. Somit ist ein Muster auf der LED-Punktmatrix gezeigt.

Nehmen Sie als Beispiele die erste Nummer von code_H, 0x01 und die erste Nummer von code_L, 0x00.

0x01 konvertiert in Binär wird 00000001; 0x00 konvertiert in Binär wird 0000 0000.

In diesem Kit wird eine gemeinsame Anoden-LED-Punktmatrix angewendet, sodass nur die acht LEDs in der achten Reihe aufleuchten.

Wenn die Bedingungen, dass Code H 0xff und Code_L 0x7f ist, gleichzeitig erfüllt sind, leuchten diese 8 LEDs in der ersten Spalte.



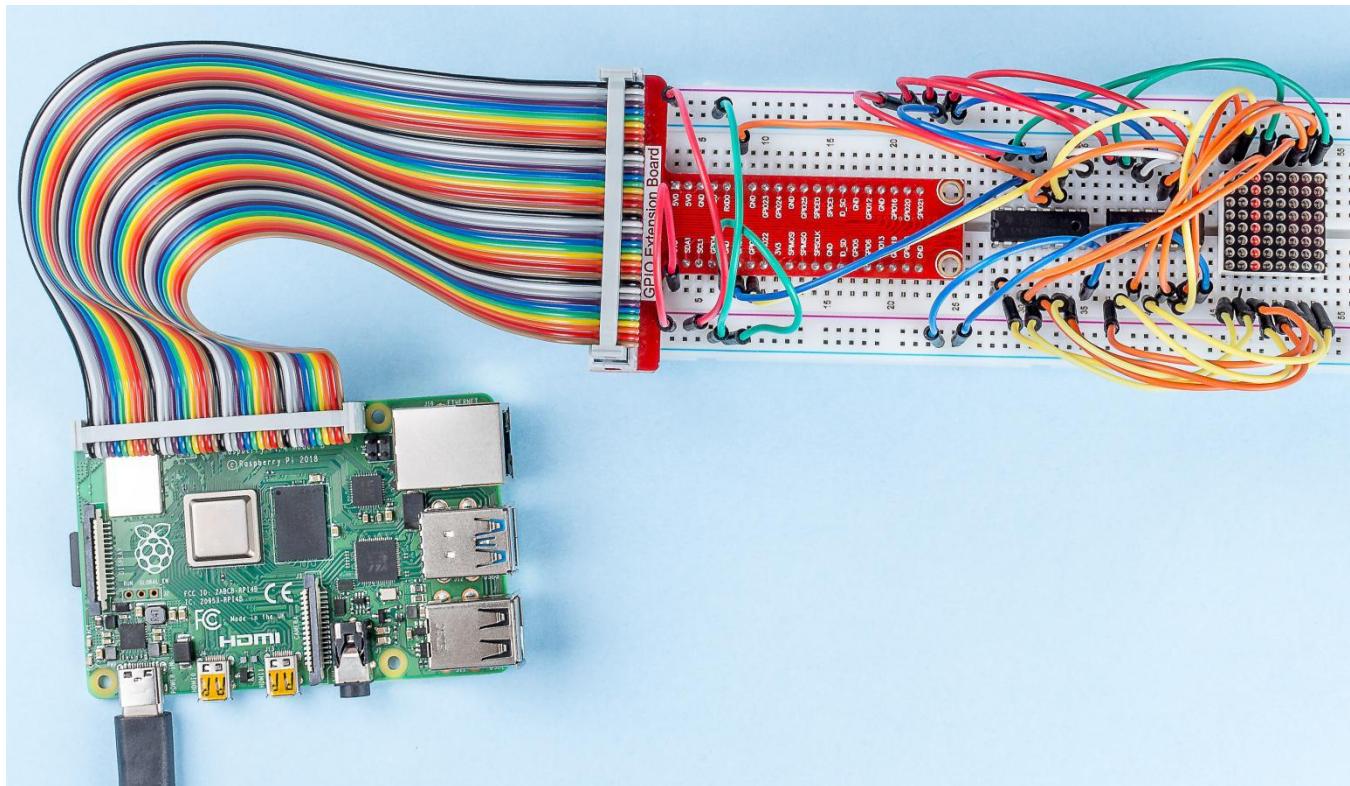
```
for i in range(0, len(code_H)):
```

```
hc595_shift(code_L[i])  
hc595_shift(code_H[i])
```

In dieser Schleife werden diese 20 Elemente in den beiden Arrays Kode_L und Kode_H nacheinander auf den HC595-Chip hochgeladen.

Hinweis: Wenn Sie Zeichen in der LED-Punktmatrix anzeigen möchten, lesen Sie bitte einen Python-Code: https://github.com/sunfounder/SunFounder_Dot_Matrix

Phänomen Bild

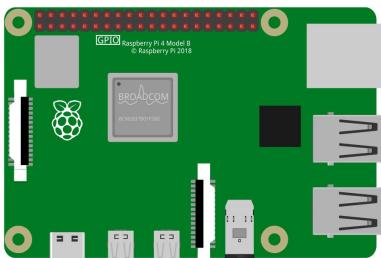
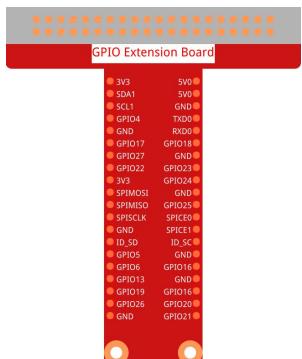
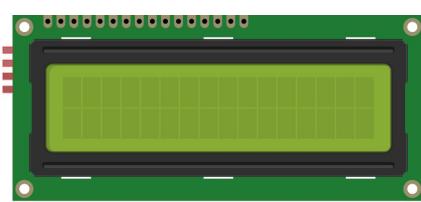
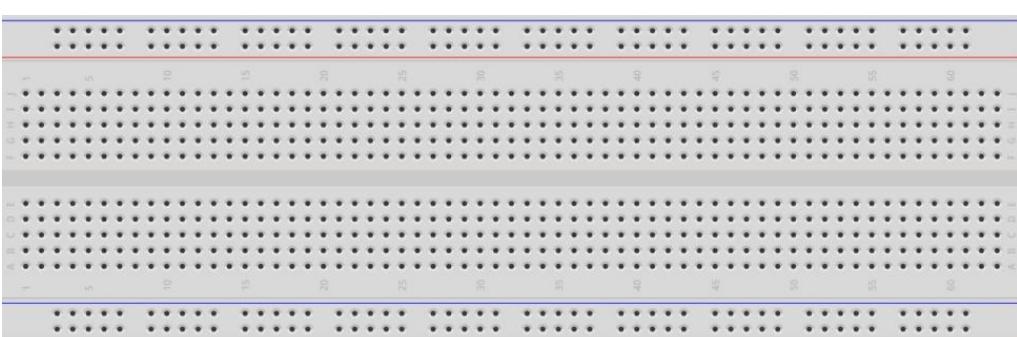


1.1.7 I2C LCD1602

Einführung

LCD1602 ist eine Flüssigkristallanzeige vom Zeichentyp, die gleichzeitig 32 (16 * 2) Zeichen anzeigen kann.

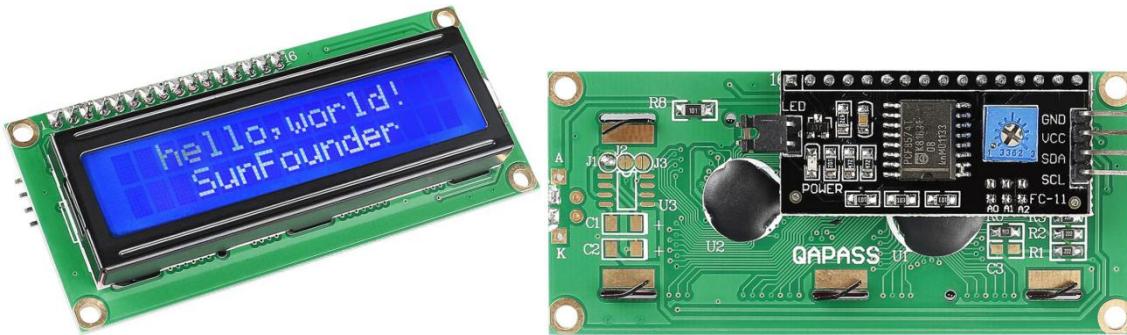
Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * I2C LCD1602																																												
	 <table border="1"> <thead> <tr> <th>Pin</th> <th>Name</th> </tr> </thead> <tbody> <tr><td>3V3</td><td>SVB</td></tr> <tr><td>5V</td><td>SVB</td></tr> <tr><td>3V3</td><td>SVB</td></tr> <tr><td>GND</td><td>GND</td></tr> <tr><td>GPIO4</td><td>TXDO</td></tr> <tr><td>GND</td><td>RXD0</td></tr> <tr><td>GPIO17</td><td>GPIO18</td></tr> <tr><td>GPIO27</td><td>GND</td></tr> <tr><td>GPIO22</td><td>GPIO23</td></tr> <tr><td>3V3</td><td>GPIO24</td></tr> <tr><td>GPIO00</td><td>GPIO25</td></tr> <tr><td>SPICSO</td><td>GPIO26</td></tr> <tr><td>SPISCLK</td><td>SPICSD</td></tr> <tr><td>GND</td><td>SPICEO</td></tr> <tr><td>ID_SD</td><td>ID_SC</td></tr> <tr><td>GPIO5</td><td>GND</td></tr> <tr><td>GPIO6</td><td>GPIO16</td></tr> <tr><td>GPIO13</td><td>GND</td></tr> <tr><td>GPIO19</td><td>GPIO10</td></tr> <tr><td>GPIO26</td><td>GPIO20</td></tr> <tr><td>GND</td><td>GPIO21</td></tr> </tbody> </table>	Pin	Name	3V3	SVB	5V	SVB	3V3	SVB	GND	GND	GPIO4	TXDO	GND	RXD0	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GPIO24	GPIO00	GPIO25	SPICSO	GPIO26	SPISCLK	SPICSD	GND	SPICEO	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO10	GPIO26	GPIO20	GND	GPIO21	
Pin	Name																																													
3V3	SVB																																													
5V	SVB																																													
3V3	SVB																																													
GND	GND																																													
GPIO4	TXDO																																													
GND	RXD0																																													
GPIO17	GPIO18																																													
GPIO27	GND																																													
GPIO22	GPIO23																																													
3V3	GPIO24																																													
GPIO00	GPIO25																																													
SPICSO	GPIO26																																													
SPISCLK	SPICSD																																													
GND	SPICEO																																													
ID_SD	ID_SC																																													
GPIO5	GND																																													
GPIO6	GPIO16																																													
GPIO13	GND																																													
GPIO19	GPIO10																																													
GPIO26	GPIO20																																													
GND	GPIO21																																													
1 * 40-Pin Kabel		Mehrere Überbrückungsdrähte																																												
																																														
1 * Steckbrett																																														

Prinzip

I2C LCD1602

Wie wir alle wissen, bereichern LCD und einige andere Anzeiger die Mensch-Maschine-Interaktion erheblich, weisen jedoch eine gemeinsame Schwäche auf. Wenn sie mit einem Controller verbunden sind, werden mehrere E / A des Controllers belegt, der nicht so viele äußere Ports hat. Es schränkt auch andere Funktionen der Steuerung ein. Daher wurde LCD1602 mit einem I2C-Bus entwickelt, um das Problem zu lösen.



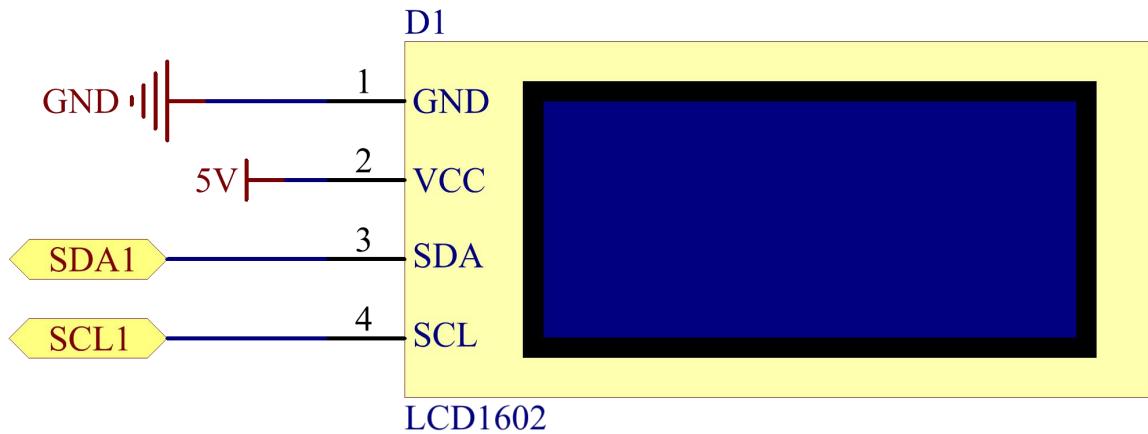
I2C-Kommunikation

Der I2C-Bus (Inter-Integrated Circuit) ist ein sehr beliebter und leistungsstarker Bus für die Kommunikation zwischen einem Master-Gerät (oder Master-Geräten) und einem oder mehreren Slave-Geräten.

Der I2C-Hauptcontroller kann zur Steuerung des E / A-Expanders, verschiedener Sensoren, des EEPROM, des ADC / DAC usw. verwendet werden. Alle diese werden nur von den beiden Pins des Hosts gesteuert, der seriellen Datenleitung (SDA1) und der seriellen Takteleitung (SCL1).

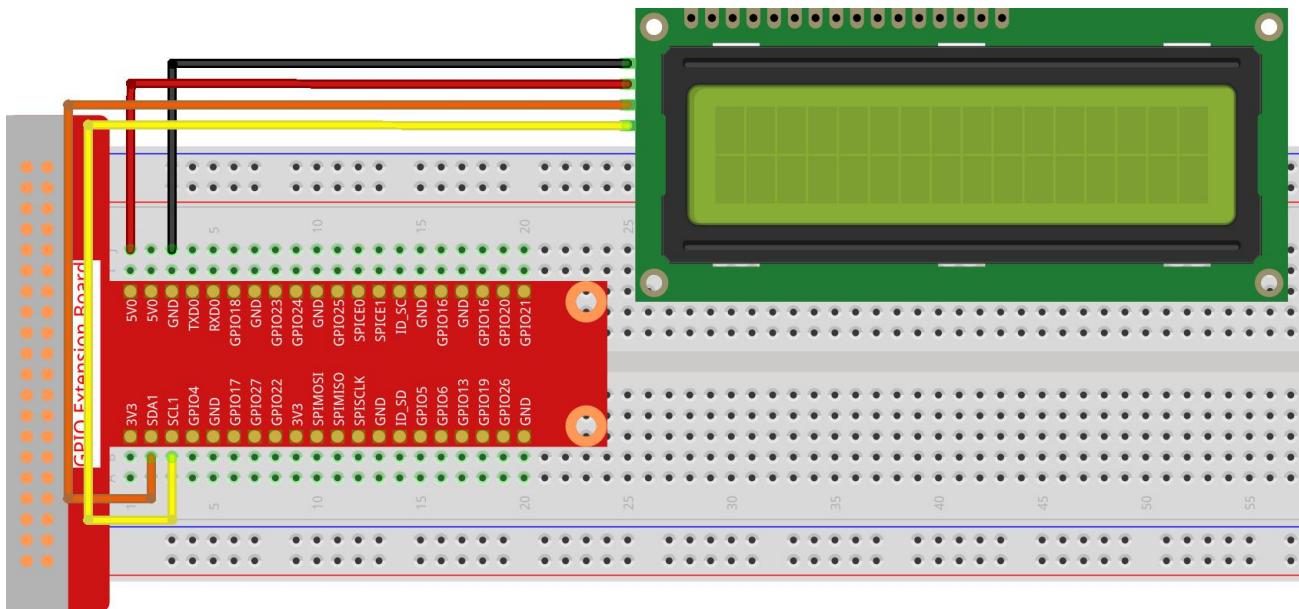
Schematische Darstellung

T-Karte Name	physisch
SDA1	Pin 3
SCL1	Pin 5



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



Schritt 2: I2C einrichten (siehe Anhang. Wenn Sie I2C eingestellt haben, überspringen Sie diesen Schritt.)

➤ Für Benutzer in C-Sprache

Schritt 3: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/1.1.7/
```

Schritt 4: Kompilieren.

```
gcc 1.1.7_Lcd1602.c -lwiringPi
```

Schritt 5: Ausführen.

```
sudo ./a.out
```

Nachdem die Kode ausgeführt wurde, werden auf dem LCD "Grüße" und "Von SunFounder" angezeigt.

Kode

Hinweis: Keine der folgenden Funktionen mit Ellipsen ist abgeschlossen. Sie können den vollständigen Kode mit dem Befehl nano 1.1.7 _lcd1602.c in der Bash-Oberfläche anzeigen.

```
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPi2C.h>
#include <string.h>

int LCDAddr = 0x27;
int BLEN = 1;
int fd;

void write_word(int data){.....}
void send_command(int comm){.....}
void send_data(int data){.....}
void init(){.....}
void clear(){.....}
void write(int x, int y, char data[]){.....}

void main(){
    fd = wiringPi2CSetup(LCDAddr);
    init();
    write(0, 0, "Greetings!");
    write(1, 1, "From SunFounder");
}
```

Kode Erklärung

```
void write_word(int data){.....}
void send_command(int comm){.....}
void send_data(int data){.....}
void init(){.....}
void clear(){.....}
void write(int x, int y, char data[]){.....}
```

Diese Funktionen werden zur Steuerung des Open Source-Codes I2C LCD1602 verwendet. Sie ermöglichen die einfache Verwendung von I2C LCD1602.

Unter diesen Funktionen wird init () zur Initialisierung verwendet, clear () wird zum Löschen des Bildschirms verwendet, write () wird zum Schreiben der angezeigten Elemente verwendet und andere Funktionen unterstützen die obigen Funktionen.

```
fd = wiringPil2CSetup(LCDAddr);
```

Diese Funktion initialisiert das I2C-System mit dem angegebenen Gerätesymbol. Der Prototyp der Funktion:

```
int wiringPil2CSetup(int devId);
```

Parameter devId ist die Adresse des I2C-Geräts. Sie kann über den Befehl i2cdetect (siehe Anhang) ermittelt werden. Die devId des I2C LCD1602 ist im Allgemeinen 0x27.

```
void write(int x, int y, char data[]){}
```

In dieser Funktion ist data [] das Zeichen, das auf dem LCD gedruckt werden soll, und die Parameter x und y bestimmen die Druckposition (Zeile y + 1, Spalte x + 1 ist die Startposition des zu druckenden Zeichens).

➤ Für Python-Sprachbenutzer

Schritt 3: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 4: Ausführen.

```
sudo python3 1.1.7_Lcd1602.py
```

Nachdem die Kode ausgeführt wurde, werden auf dem LCD "Grüße" und "Von SunFounder" angezeigt.

Kode

```
import LCD1602
import time

def setup():
    LCD1602.init(0x27, 1)    # init(slave address, background light)
    LCD1602.write(0, 0, 'Greetings!!')
    LCD1602.write(1, 1, 'from SunFounder')
    time.sleep(2)
```

```
def destroy():
    LCD1602.clear()

if __name__ == "__main__":
    try:
        setup()
    except KeyboardInterrupt:
        destroy()
```

Kode Erklärung

```
import LCD1602
```

Diese Datei ist eine Open Source-Datei zur Steuerung von I2C LCD1602. Es ermöglicht uns die einfache Verwendung von I2C LCD1602.

```
LCD1602.init(0x27, 1)
```

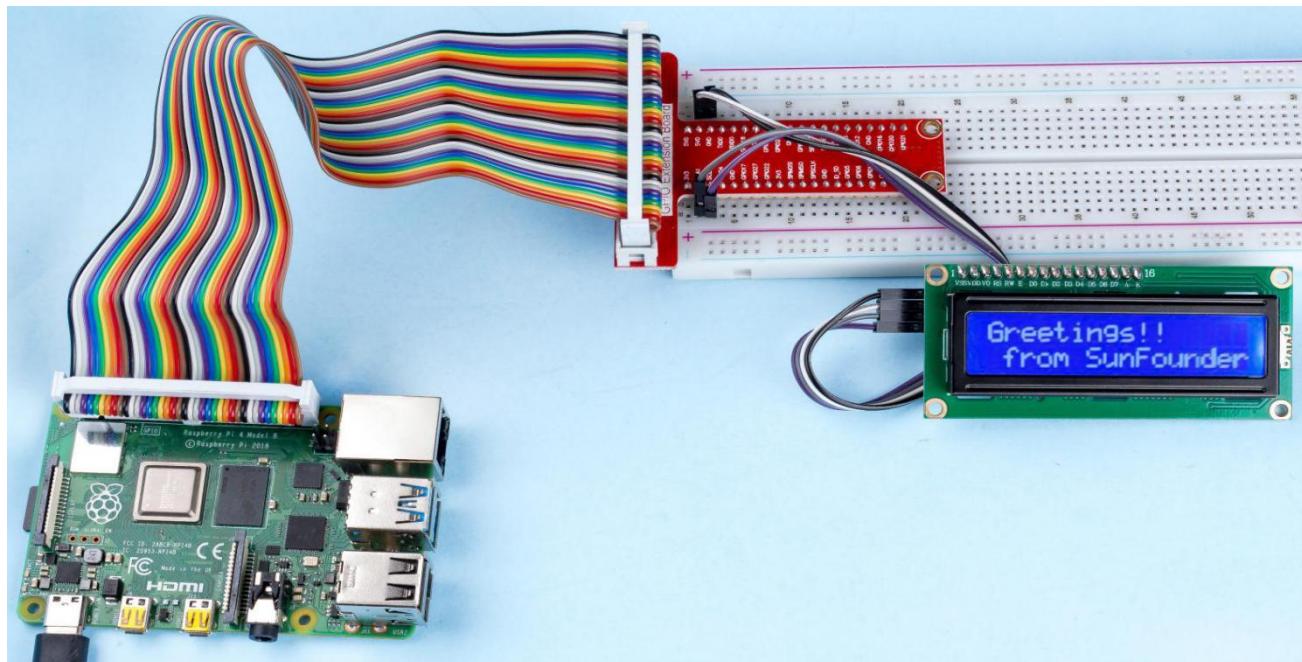
Die Funktion initialisiert das I2C-System mit dem angegebenen Gerätesymbol. Der erste Parameter ist die Adresse des I2C-Geräts, die mit dem Befehl i2cdetect erkannt werden kann (Einzelheiten siehe Anhang). Die Adresse des I2C LCD1602 lautet im Allgemeinen 0x27.

```
LCD1602.write(0, 0, 'Greetings!!')
```

Innerhalb dieser Funktion 'Grüße !!' ist das Zeichen, das in der Zeile 0 + 1, Spalte 0 + 1 auf dem LCD gedruckt werden soll.

Jetzt können Sie „Grüße! Von SunFounder“ auf dem LCD angezeigt.

Phänomen Bild



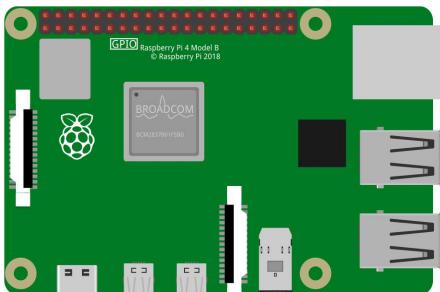
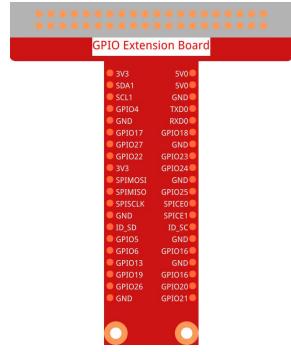
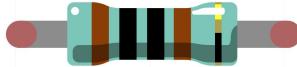
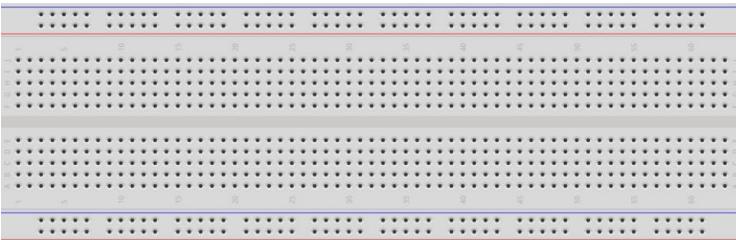
1.2 Ton

1.2.1 Aktiver Summer

Einführung

In dieser Lektion lernen wir, wie man einen aktiven Summer ansteuert, um mit einem PNP-Transistor zu piepen.

Komponenten

1 * Raspberry Pi	1 * T- Erweiterungskarte	1 * Aktiver Summer
	 Pinout: <ul style="list-style-type: none">3V3 5V0SPI01 GNDSCL1 GNDGPIO4 TXD0GND RXD0GPIO17 GPIO18GPIO27 GNDGPIO22 GPIO23GPIO25 GNDSPI0SO GNDSPI0SI GPIO25SPI0CLK SPICE0GND SPICE1ID_SD ID_SEGPIO3 GNDGPIO2 GNDGPIO13 GNDGPIO19 GPIO10GPIO26 GPIO20GND GPIO21	
1 * 40-Pin Kabel		1 * Widerstand (1 kΩ)
		
1 * Steckbrett		Mehrere Überbrückungsdrähte
		
		1 * S8550 PNP-Transistor
		

Prinzip

Summer

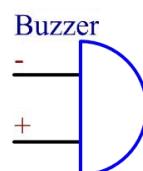
Als eine Art elektronischer Summer mit integrierter Struktur werden Summer, die mit Gleichstrom versorgt werden, häufig in Computern, Druckern, Fotokopierern, Alarmen, elektronischem Spielzeug, elektronischen Kraftfahrzeugen, Telefonen, Zeitschaltuhren und anderen elektronischen Produkten oder Sprachgeräten verwendet. Summer können in aktive und passive unterteilt werden (siehe folgendes Bild). Drehen Sie den Summer so, dass seine Stifte nach oben zeigen, und der Summer mit einer grünen Leiterplatte ist ein passiver Summer, während der mit einem schwarzen Band umschlossene ein aktiver ist.

Der Unterschied zwischen einem aktiven und einem passiven Summer:



Der Unterschied zwischen einem aktiven und einem passiven Summer besteht darin, dass ein aktiver Summer über eine integrierte Oszillationsquelle verfügt, sodass er bei Elektrifizierung Geräusche erzeugt. Ein passiver Summer verfügt jedoch nicht über eine solche Quelle, sodass bei Verwendung von Gleichstromsignalen kein Piepton ertönt. Stattdessen müssen Sie Rechteckwellen verwenden, deren Frequenz zwischen 2K und 5K liegt, um sie anzutreiben. Der aktive Summer ist aufgrund mehrerer eingebauter Schwingkreise oft teurer als der passive.

Das Folgende ist das elektrische Symbol eines Summers. Es hat zwei Stifte mit positiven und negativen Polen. Mit einem + in der Oberfläche steht die Anode und das andere ist die Kathode.

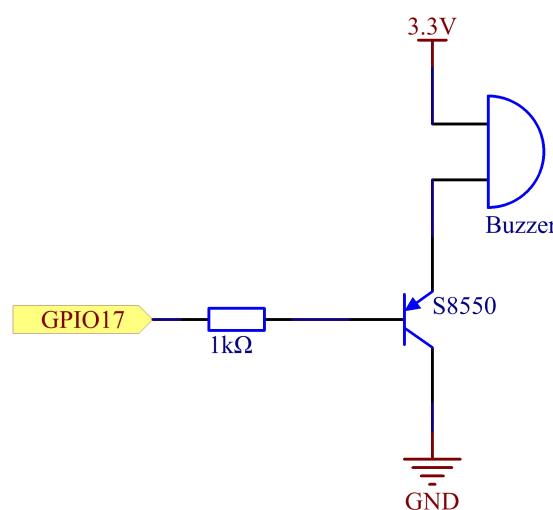


Sie können die Pins des Summers überprüfen, je länger die Anode und je kürzer die Kathode ist. Bitte verwechseln Sie sie beim Anschließen nicht, da sonst kein Summer ertönt.

Schematische Darstellung

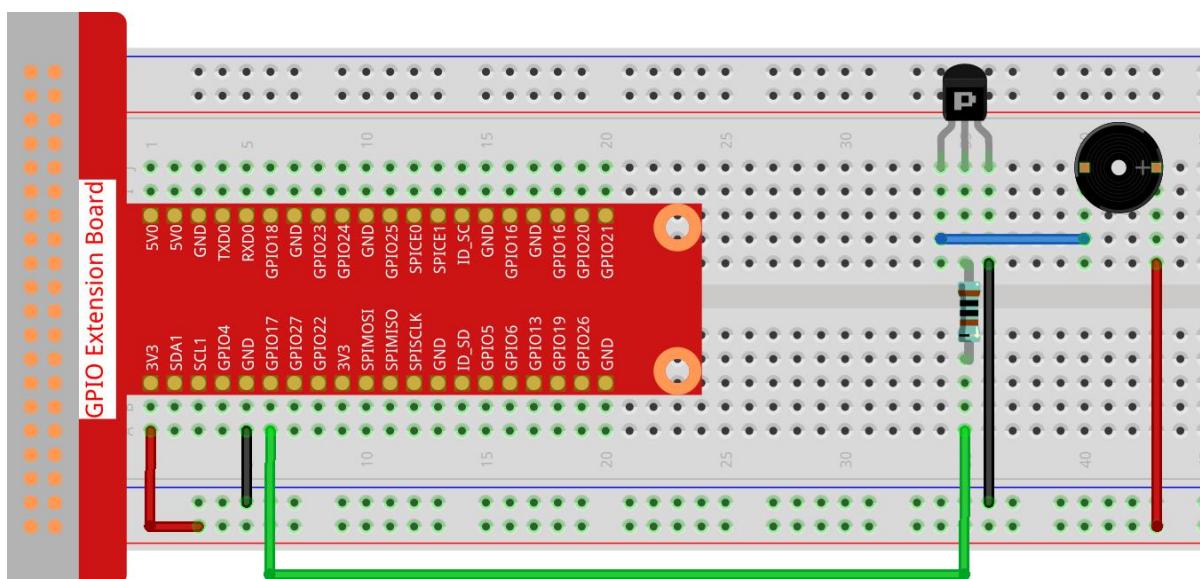
In diesem Experiment werden ein aktiver Summer, ein PNP-Transistor und ein 1k-Widerstand zwischen der Basis des Transistors und GPIO verwendet, um den Transistor zu schützen. Wenn der GPIO17 des Raspberry Pi-Ausgangs durch Programmierung mit einem niedrigen Niveau (0V) versorgt wird, leitet der Transistor aufgrund der Stromsättigung und der Summer gibt Geräusche aus. Wenn jedoch die IO des Raspberry Pi mit einem hohen Niveau versorgt wird, wird der Transistor abgeschaltet und der Summer gibt keine Geräusche von sich.

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf. (Achten Sie auf die Pole des Summers: Der mit dem + Etikett ist der positive Pol und der andere der negative.)



Für Benutzer in C-Sprache

Schritt 2: Öffnen Sie die Codedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/1.2.1/
```

Schritt 3: Kompilieren Sie den Code.

```
gcc 1.2.1_ActiveBuzzer.c -lwiringPi
```

Schritt 4: Führen Sie die obige ausführbare Datei aus.

```
sudo ./a.out
```

Die Kode läuft, der Summer piept.

Kode

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

#define BeepPin 0
int main(void){
    if(wiringPiSetup() == -1){ //when initialize wiring failed, print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }

    pinMode(BeepPin, OUTPUT);    //set GPIO0 output
    while(1){
        //beep on
        printf("Buzzer on\n");
        digitalWrite(BeepPin, LOW);
        delay(100);
        printf("Buzzer off\n");
        //beep off
        digitalWrite(BeepPin, HIGH);
        delay(100);
    }
    return 0;
}
```

Kode Erklärung

```
digitalWrite(BeepPin, LOW);
```

In diesem Experiment verwenden wir einen aktiven Summer, der beim Anschließen an den Gleichstrom automatisch einen Ton erzeugt. Diese Skizze dient dazu, den I/O - Port auf einen niedrigen Niveau (0V) einzustellen, um so den Transistor zu verwalten und den Summer piepen zu lassen.

```
digitalWrite(BeepPin, HIGH);
```

Um den /O -Anschluss auf einen hohen Niveau (3,3V) einzustellen, wird der Transistor nicht erregt und der Summer piept nicht.

➤ Für Python-Sprachbenutzer

Schritt 2: Öffnen Sie die Kodedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Schritt 3: Ausführen.

```
sudo python3 1.2.1_ActiveBuzzer.py
```

Die Kode läuft, der Summer piept.

Kode

```
import RPi.GPIO as GPIO
import time

# Set GPIO17 as buzzer pin
BeepPin = 17

def setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(BeepPin, GPIO.OUT, initial=GPIO.HIGH)

def main():
    while True:
        # Buzzer on (Beep)
        print ('Buzzer On')
        GPIO.output(BeepPin, GPIO.LOW)
        time.sleep(0.1)
        # Buzzer off
        print ('Buzzer Off')
        GPIO.output(BeepPin, GPIO.HIGH)
```

```
GPIO.output(BeepPin, GPIO.HIGH)
time.sleep(0.1)

def destroy():
    # Turn off buzzer
    GPIO.output(BeepPin, GPIO.HIGH)
    # Release resource
    GPIO.cleanup()

# If run this script directly, do:
if __name__ == '__main__':
    setup()
    try:
        main()
    # When 'Ctrl+C' is pressed, the program
    # destroy() will be executed.
    except KeyboardInterrupt:
        destroy()
```

Kode Erklärung

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

Stellen Sie den Summer Pin auf einen niedrigen Niveau ein, damit der Summer piept.

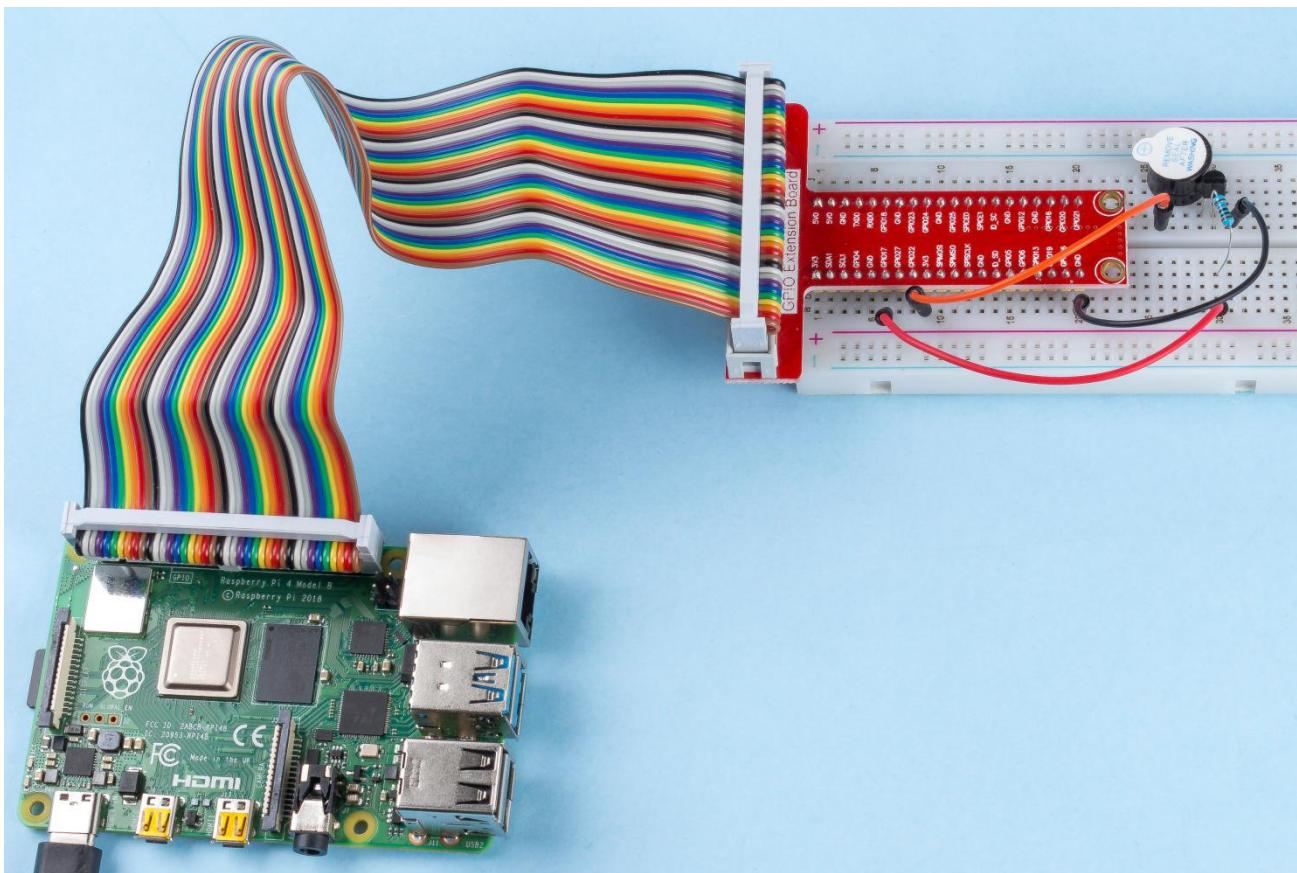
```
time.sleep(0.1)
```

Warten Sie 0,1 Sekunden. Ändern Sie die Schaltfrequenz, indem Sie diesen Parameter ändern. **Hinweis:** Nicht die Schallfrequenz. Der aktive Summer kann die Schallfrequenz nicht ändern.

```
GPIO.output(BeepPin, GPIO.HIGH)
```

Schließen Sie den Summer.

Phänomen Bild

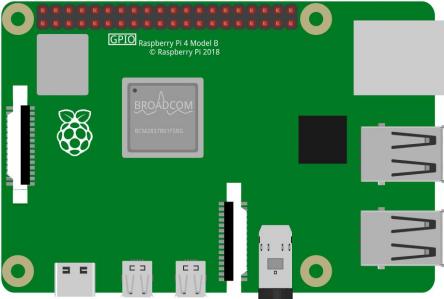
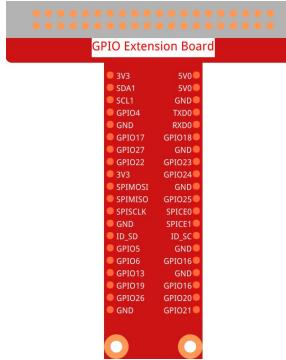
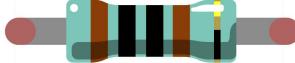
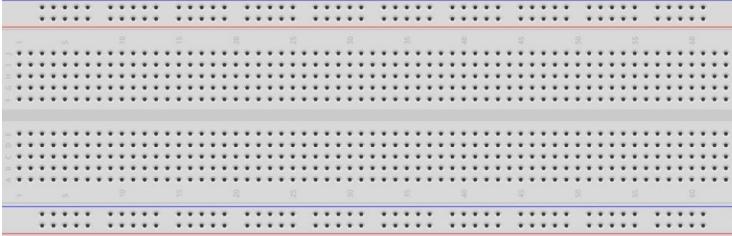


1.2.2 Passiver Summer

Einführung

In dieser Lektion lernen wir, wie man einen passiven Summer dazu bringt, Musik zu spielen.

Komponenten

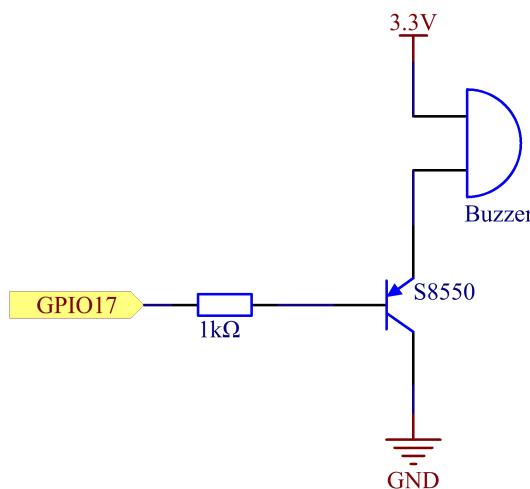
1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Passiver Summer																																										
	 <table border="1"> <thead> <tr> <th colspan="2">GPIO Extension Board</th> </tr> </thead> <tbody> <tr><td>3V3</td><td>AVG</td></tr> <tr><td>GND</td><td>5V</td></tr> <tr><td>SCL1</td><td>GND</td></tr> <tr><td>GPIO4</td><td>TxD0</td></tr> <tr><td>GND</td><td>RxD0</td></tr> <tr><td>GPIO17</td><td>GPIO18</td></tr> <tr><td>GPIO27</td><td>GND</td></tr> <tr><td>GPIO22</td><td>GPIO23</td></tr> <tr><td>3V3</td><td>GPIO24</td></tr> <tr><td>GND</td><td>GPIO25</td></tr> <tr><td>SPI_MISO</td><td>GPIO26</td></tr> <tr><td>SPI_SCLK</td><td>SPI_CE0</td></tr> <tr><td>GND</td><td>SPI_CE1</td></tr> <tr><td>ID_SD</td><td>ID_SC</td></tr> <tr><td>GPIO5</td><td>GND</td></tr> <tr><td>GPIO6</td><td>GPIO16</td></tr> <tr><td>GPIO13</td><td>GND</td></tr> <tr><td>GPIO19</td><td>GPIO18</td></tr> <tr><td>GPIO26</td><td>GPIO20</td></tr> <tr><td>GND</td><td>GPIO21</td></tr> </tbody> </table>	GPIO Extension Board		3V3	AVG	GND	5V	SCL1	GND	GPIO4	TxD0	GND	RxD0	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GPIO24	GND	GPIO25	SPI_MISO	GPIO26	SPI_SCLK	SPI_CE0	GND	SPI_CE1	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO18	GPIO26	GPIO20	GND	GPIO21	
GPIO Extension Board																																												
3V3	AVG																																											
GND	5V																																											
SCL1	GND																																											
GPIO4	TxD0																																											
GND	RxD0																																											
GPIO17	GPIO18																																											
GPIO27	GND																																											
GPIO22	GPIO23																																											
3V3	GPIO24																																											
GND	GPIO25																																											
SPI_MISO	GPIO26																																											
SPI_SCLK	SPI_CE0																																											
GND	SPI_CE1																																											
ID_SD	ID_SC																																											
GPIO5	GND																																											
GPIO6	GPIO16																																											
GPIO13	GND																																											
GPIO19	GPIO18																																											
GPIO26	GPIO20																																											
GND	GPIO21																																											
1 * 40-Pin Kabel	1 * Widerstand (1 kΩ)																																											
																																												
1 * Steckbrett	Mehrere Überbrückungsdrähte																																											
																																												
	1 * S8550 PNP-Transistor																																											
																																												

Schematische Darstellung

In diesem Experiment werden ein passiver Summer, ein PNP-Transistor und ein 1k-Widerstand zwischen der Basis des Transistors und GPIO verwendet, um den Transistor zu schützen.

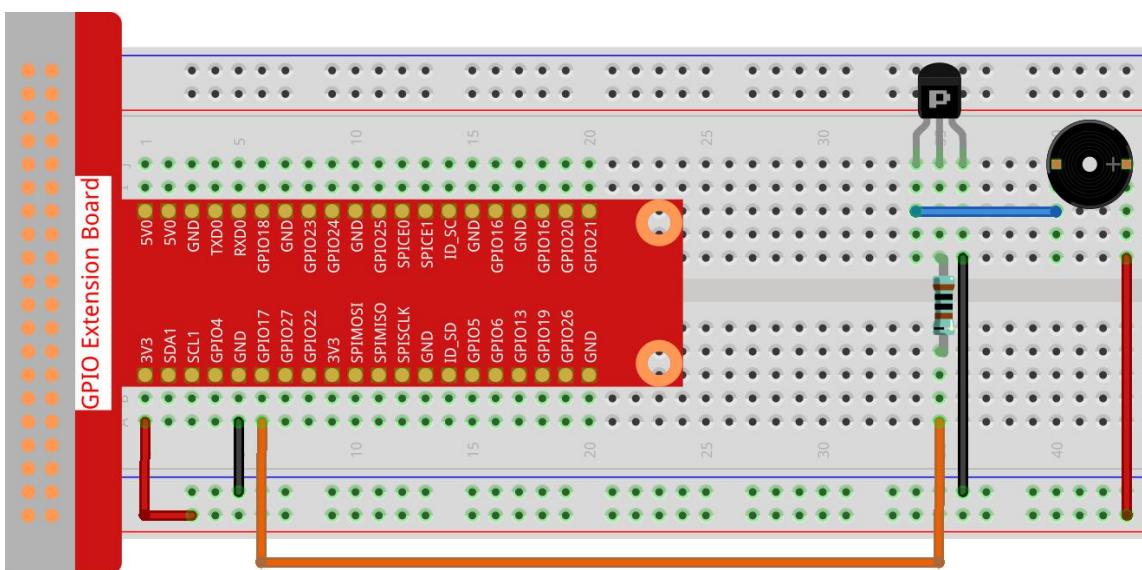
Wenn GPIO17 unterschiedliche Frequenzen erhält, gibt der passive Summer unterschiedliche Töne aus. Auf diese Weise spielt der Summer Musik.

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/1.2.2/
```

Schritt 3: Kompilieren.

```
gcc 1.2.2_PassiveBuzzer.c -lwiringPi
```

Schritt 4: Ausführen.

```
sudo ./a.out
```

Die Kode läuft, der Summer spielt ein Musikstück.

Kode

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

#define BuzPin    0

#define CL1    131
#define CL2    147
#define CL3    165
#define CL4    175
#define CL5    196
#define CL6    221
#define CL7    248

#define CM1    262
#define CM2    294
#define CM3    330
#define CM4    350
#define CM5    393
#define CM6    441
#define CM7    495

#define CH1    525
#define CH2    589
#define CH3    661
#define CH4    700
#define CH5    786
#define CH6    882
#define CH7    990
```

```

int song_1[] = {CM3,CM5,CM6,CM3,CM2,CM3,CM5,CM6,CH1,CM6,CM5,CM1,CM3,CM2,
                CM2,CM3,CM5,CM2,CM3,CM3,CL6,CL6,CL6,CM1,CM2,CM3,CM2,CL7,
                CL6,CM1,CL5};

int beat_1[] = {1,1,3,1,1,3,1,1,1,1,1,1,3,1,1,3,1,1,1,1,1,1,2,1,1,
                1,1,1,1,1,3};

int song_2[] = {CM1,CM1,CM1,CL5,CM3,CM3,CM1,CM1,CM3,CM5,CM5,CM4,CM3,CM2,
                CM2,CM3,CM4,CM4,CM3,CM2,CM3,CM1,CM1,CM3,CM2,CL5,CL7,CM2,CM1
};

int beat_2[] = {1,1,1,3,1,1,1,3,1,1,1,1,1,3,1,1,2,1,1,1,3,1,1,1,3,3,2,3};

int main(void)
{
    int i, j;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }

    if(softToneCreate(BuzPin) == -1){
        printf("setup softTone failed !");
        return 1;
    }

    while(1){
        printf("music is being played...\n");

        for(i=0;i<sizeof(song_1)/4;i++){
            softToneWrite(BuzPin, song_1[i]);
            delay(beat_1[i] * 500);
        }

        for(i=0;i<sizeof(song_2)/4;i++){
            softToneWrite(BuzPin, song_2[i]);
            delay(beat_2[i] * 500);
        }
    }
}

```

```
    return 0;
}
```

Kode Erklärung

```
#define CL1 131
#define CL2 147
#define CL3 165
#define CL4 175
#define CL5 196
#define CL6 221
#define CL7 248

#define CM1 262
#define CM2 294
...
```

Diese Frequenzen jeder Note sind wie gezeigt. CL bezieht sich auf tiefe Note, CM mittlere Note, CH hohe Note, 1-7 entsprechen den Noten C, D, E, F, G, A, B.

```
int song_1[] = {CM3,CM5,CM6,CM3,CM2,CM3,CM5,CM6,CH1,CM6,CM5,CM1,CM3,CM2,
                CM2,CM3,CM5,CM2,CM3,CM3,CL6,CL6,CL6,CM1,CM2,CM3,CM2,CL7,
                CL6,CM1,CL5};
int beat_1[] = {1,1,3,1,1,3,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,1,1,
                1,1,1,1,1,1,3};
```

Das Array song_1 [] speichert eine Musikpartitur eines Gelieds, in der sich beat_1 [] auf der Schlagart jeder Note im Gelied bezieht (0,5 S für jeder Schlagart).

```
if(softToneCreate(BuzPin) == -1){
    printf("setup softTone failed !");
    return 1;
```

Dadurch wird ein softwaregesteuerter Ton Pin erstellt. Sie können einen beliebigen GPIO-Pin verwenden. Die Pin-Nummerierung entspricht der von Ihnen verwendeten Funktion wiringPiSetup (). Der Rückgabewert ist 0 für Erfolg. Alles andere und Sie sollten die globale Fehlerbehebung überprüfen, um festzustellen, was schief gelaufen ist.

```
for(i=0;i<sizeof(song_1)/4;i++){
    softToneWrite(BuzPin, song_1[i]);
```

```
    delay(beat_1[i] * 500);
}
```

Verwenden Sie eine for-Anweisung, um song_1 abzuspielen.

In der Urteilsbedingung wird **i<sizeof(song_1)/4**, "devide by 4" verwendet, da das Array song_1[] ein Array des Datentyps ganzer Nummer ist und jedes Element vier Bytes in Anspruch nimmt.

Die Anzahl der Elemente in song_1 (die Anzahl der Noten) wird erhalten, indem sizeof(song_4) um 4 geteilt wird.

Damit jede Note für Schläge * 500ms gespielt werden kann, wird die Funktionsverzögerung (Beat_1 [i] * 500) aufgerufen.

Der Prototyp von softToneWrite(BuzPin, song_1[i]):

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

Dadurch wird der Tonfrequenzwert am angegebenen Pin aktualisiert. Der Ton hört erst auf zu spielen, wenn Sie die Frequenz auf 0 eingestellen.

➤ Für Python-Sprachbenutzer

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Ausführen.

```
sudo python3 1.2.2_PassiveBuzzer.py
```

Die Kode läuft, der Summer spielt ein Musikstück.

Kode

```
import RPi.GPIO as GPIO
import time

Buzzer = 11

CL = [0, 131, 147, 165, 175, 196, 211, 248]      # Frequency of Bass tone in C major
CM = [0, 262, 294, 330, 350, 393, 441, 495]      # Frequency of Midrange tone in C major
CH = [0, 525, 589, 661, 700, 786, 882, 990]      # Frequency of Treble tone in C major

song_1 = [ CM[3], CM[5], CM[6], CM[3], CM[2], CM[3], CM[5], CM[6], # Notes of song1
          CH[1], CM[6], CM[5], CM[1], CM[3], CM[2], CM[2], CM[3],
          CM[5], CM[2], CM[3], CM[3], CL[6], CL[6], CM[1],
```

```
CM[2], CM[3], CM[2], CL[7], CL[6], CM[1], CL[5] ]
```

```
beat_1 = [ 1, 1, 3, 1, 1, 3, 1, 1, # Beats of song 1, 1 means 1/8 beat
           1, 1, 1, 1, 1, 1, 3, 1,
           1, 3, 1, 1, 1, 1, 1, 1,
           1, 2, 1, 1, 1, 1, 1, 1,
           1, 1, 3 ]
```

```
song_2 = [ CM[1], CM[1], CM[1], CL[5], CM[3], CM[3], CM[3], CM[1], # Notes of song2
           CM[1], CM[3], CM[5], CM[5], CM[4], CM[3], CM[2], CM[2],
           CM[3], CM[4], CM[4], CM[3], CM[2], CM[3], CM[1], CM[1],
           CM[3], CM[2], CL[5], CL[7], CM[2], CM[1] ]
```

```
beat_2 = [ 1, 1, 2, 2, 1, 1, 2, 2, # Beats of song 2, 1 means 1/8 beat
           1, 1, 2, 2, 1, 1, 3, 1,
           1, 2, 2, 1, 1, 2, 2, 1,
           1, 2, 2, 1, 1, 3 ]
```

```
def setup():
    GPIO.setmode(GPIO.BOARD)      # Numbers GPIOs by physical location
    GPIO.setup(Buzzer, GPIO.OUT)   # Set pins' mode is output
    global Buzz                  # Assign a global variable to replace GPIO.PWM
    Buzz = GPIO.PWM(Buzzer, 440)  # 440 is initial frequency.
    Buzz.start(50)               # Start Buzzer pin with 50% duty cycle
```

```
def loop():
    while True:
        print ('\n      Playing song 1...')
        for i in range(1, len(song_1)):
            # Play song 1
            Buzz.ChangeFrequency(song_1[i]) # Change the frequency along the song note
            time.sleep(beat_1[i] * 0.5)    # delay a note for beat * 0.5s
        time.sleep(1)                  # Wait a second for next song.
```

```
        print ('\n\n      Playing song 2...')
        for i in range(1, len(song_2)):
            # Play song 1
            Buzz.ChangeFrequency(song_2[i]) # Change the frequency along the song note
            time.sleep(beat_2[i] * 0.5)    # delay a note for beat * 0.5s
```

```
def destroy():
    Buzz.stop()                  # Stop the buzzer
```

```

GPIO.output(Buzzer, 1)      # Set Buzzer pin to High
GPIO.cleanup()               # Release resource

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

```

Kode Erklärung

CL = [0, 131, 147, 165, 175, 196, 211, 248]	# Frequency of Bass tone in C major
CM = [0, 262, 294, 330, 350, 393, 441, 495]	# Frequency of Midrange tone in C major
CH = [0, 525, 589, 661, 700, 786, 882, 990]	# Frequency of Treble tone in C major

Dies sind die Frequenzen jeder Note. Die erste 0 ist das Überspringen von CL[0], so dass die Nummer 1-7 dem CDEFGAB des Tons entspricht.

```

song_1 = [ CM[3], CM[5], CM[6], CM[3], CM[2], CM[3], CM[5], CM[6],
           CH[1], CM[6], CM[5], CM[1], CM[3], CM[2], CM[2], CM[3],
           CM[5], CM[2], CM[3], CM[3], CL[6], CL[6], CL[6], CM[1],
           CM[2], CM[3], CM[2], CL[7], CL[6], CM[1], CL[5] ]

```

Diese Arrays sind die Noten eines Gelieds.

```

beat_1 = [ 1, 1, 3, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
           1, 3, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
           1, 1, 3 ]

```

Jeder Klangschlag (jede Nummer) repräsentiert den $\frac{1}{8}$ -Schlage oder 0,5s

```

Buzz = GPIO.PWM(Buzzer, 440)
Buzz.start(50)

```

Definieren Sie den Pin-Summer als PWM-Pin, stellen Sie dann seine Frequenz auf 440 ein und Buzz.start (50) wird zum Ausführen von PWM verwendet. Und stellen Sie auch den Arbeitszyklus auf 50% ein.

```

for i in range(1, len(song_1)):
    Buzz.ChangeFrequency(song_1[i])

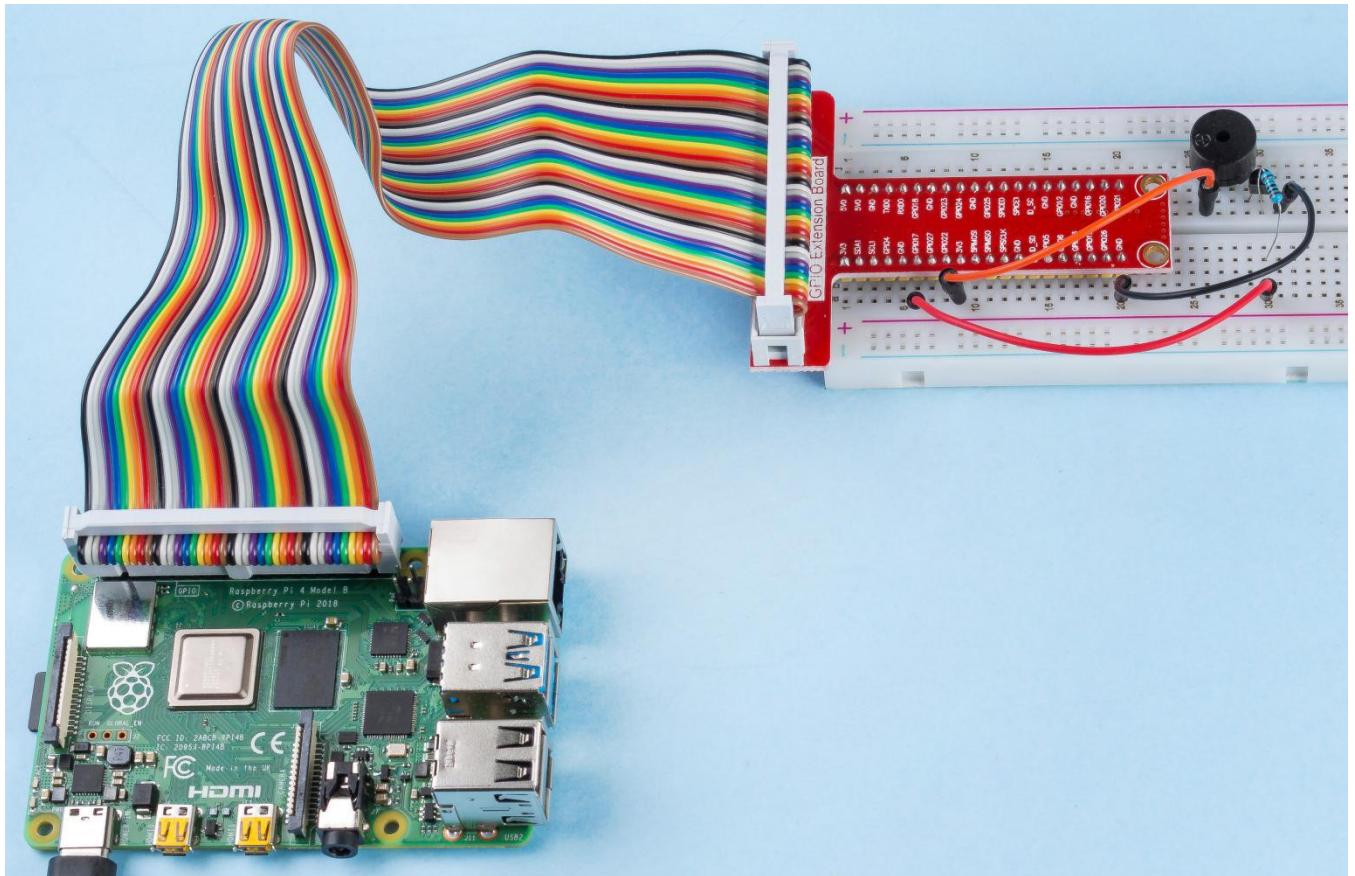
```

```
time.sleep(beat_1[i] * 0.5)
```

Führen Sie eine for-Schleife aus, dann spielt der Summer die Noten im Array song_1 [] mit den Schlägen im Array beat_1 [].

Jetzt können Sie den passiven Summer Musik spielen hören.

Phänomen Bild



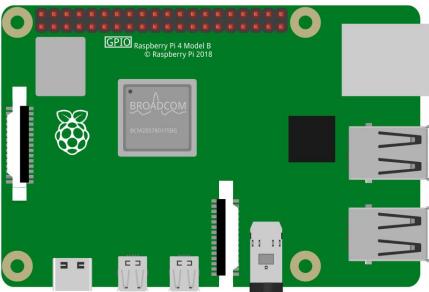
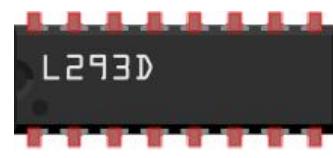
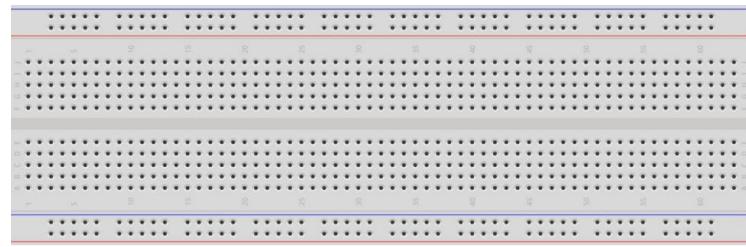
1.3 Treiber

1.3.1 Motor

Einführung

In dieser Lektion lernen wir, mit L293D einen Gleichstrommotor antreiben und ihn im und gegen den Uhrzeigersinn drehen. Da der Gleichstrommotor aus Sicherheitsgründen einen größeren Strom benötigt, verwenden wir hier das Stromversorgungsmodul zur Versorgung der Motoren.

Komponenten

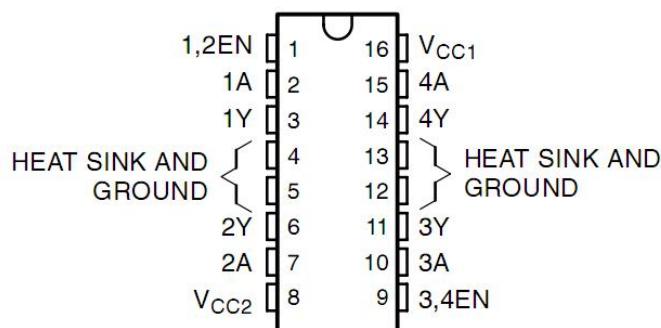
1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Power Module (mit 9V Batterie und Schnalle)
		
1 * 40-Pin Kabel	1 * L293D	
		
1 * Steckbrett	Mehrere Überbrückungsdrähte	
		
1 * Gleichstrommotor		

Prinzip

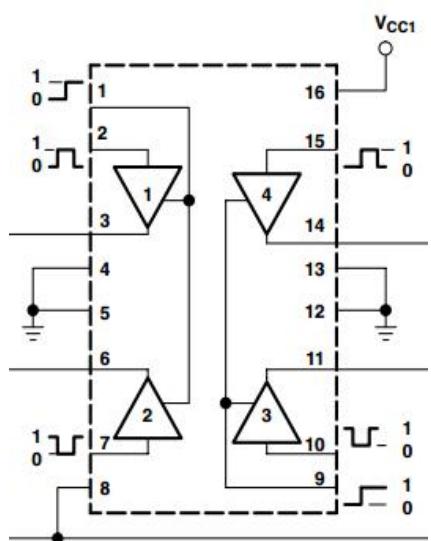
L293D

L293D ist ein 4-Kanal-Motortreiber, der durch einen Chip mit hoher Spannung und hohem Strom integriert ist. Es ist für den Anschluss an Standard-DTL-, TTL-Logikniveau und ansteuerungsinduktive Lasten (wie Relaisspulen, Gleichstrom-, Schrittmotoren) sowie Leistungsschalttransistoren usw. ausgelegt. Gleichstrommotoren sind Geräte, die elektrische Gleichstromenergie in mechanische Energie umwandeln. Sie werden im elektrischen Antrieb wegen ihrer überlegenen Geschwindigkeitsregelungsleistung häufig verwendet.

Siehe die Abbildung der Pins unten. Der L293D verfügt über zwei Pins (Vcc1 und Vcc2) für die Stromversorgung. Vcc2 wird verwendet, um den Motor mit Strom zu versorgen, während Vcc1 dient, um den Chip zu versorgen. Da hier ein kleiner Gleichstrommotor verwendet wird, verbinden Sie beide Pins mit + 5V.



Das Folgende ist die interne Struktur von L293D. Pin EN ist ein Freigabepin und funktioniert nur mit hohem Niveau. A steht für Eingabe und Y für Ausgabe. Sie können die Beziehung zwischen ihnen unten rechts sehen. Wenn Pin EN auf Hohe Niveau steht und A auf High steht, gibt Y Hohe Niveau aus. Wenn A niedrig ist, gibt Y einen niedrigen Niveau aus. Wenn Pin EN auf niedrigen Niveau steht, funktioniert der L293D nicht.



INPUTS†		OUTPUT Y
A	EN	
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)

Gleichspannungs Motor



Dies ist ein 5V Gleichstrommotor. Es dreht sich, wenn Sie den beiden Anschlüssen des Kupferblechs einen hohen und einen niedrigen Niveau geben. Der Einfachheit halber können Sie die Pin daran schweißen.

Größe: 25 * 20 * 15MM

Betriebsspannung: 1-6V

Freilaufstrom (3 V): 70 m
min

A Freilaufgeschwindigkeit (3 V): 13000 U /

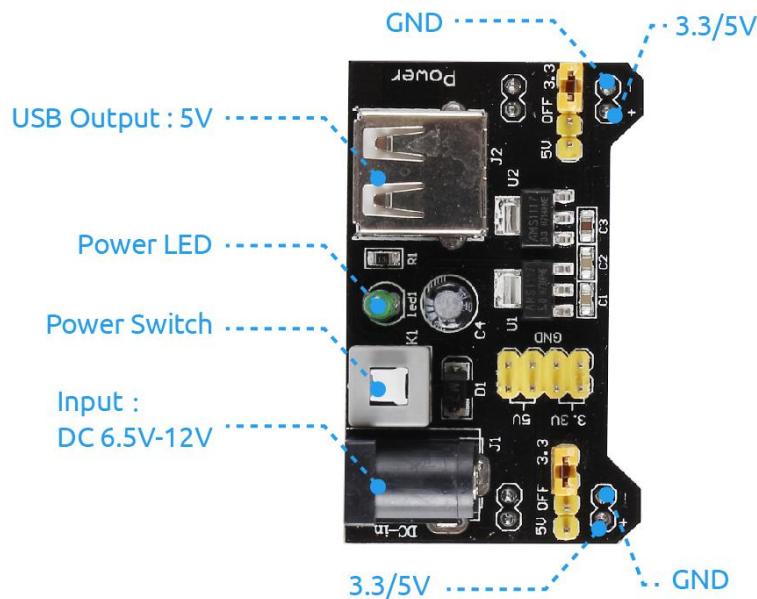
Blockierstrom (3 V): 800 mA

Wellendurchmesser: 2 mm

Energieversorgung Modul

In diesem Experiment werden große Ströme benötigt, um den Motor anzutreiben, insbesondere wenn er startet und stoppt, was die normale Arbeit von Raspberry Pi stark beeinträchtigt. Da versorgen wir diesen Motor separat mit Strom, damit er sicher und gleichmäßig läuft.

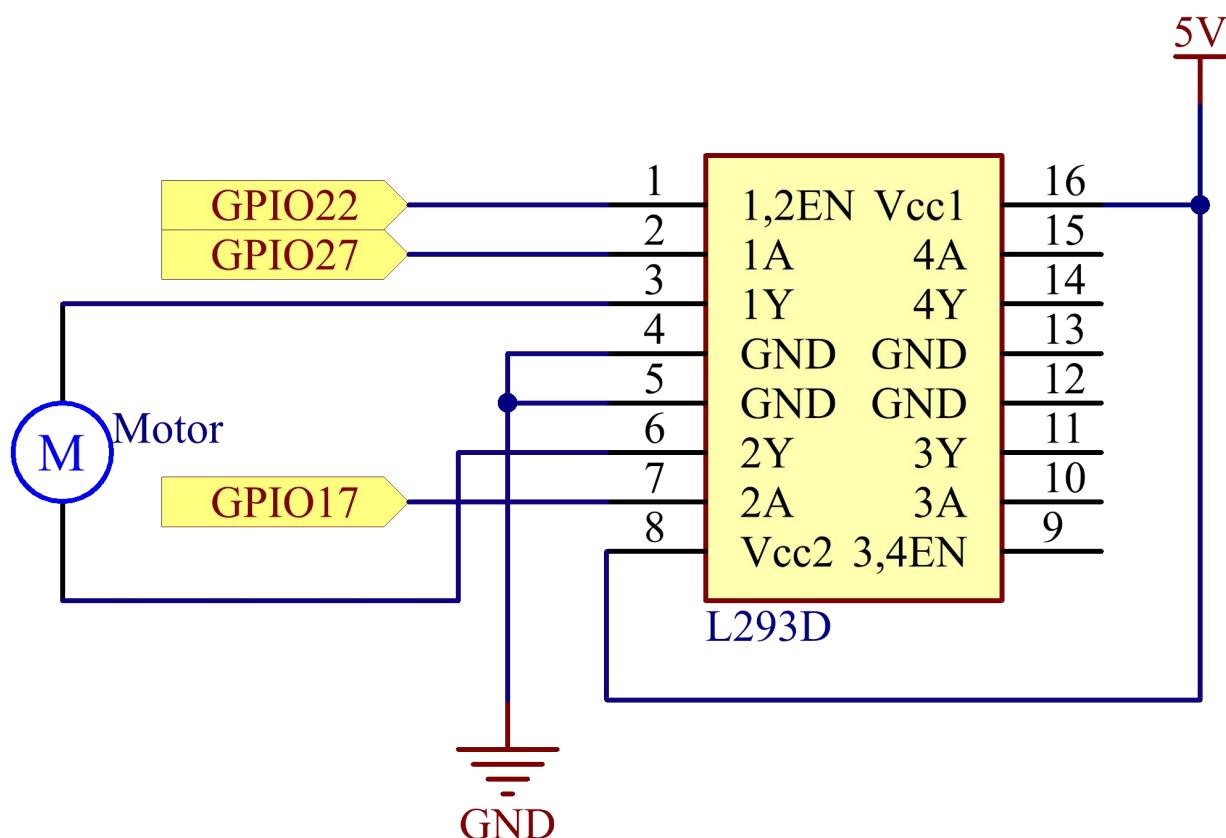
Sie können es einfach in das Steckbrett einstecken, um Strom zu liefern. Es liefert eine Spannung von 3,3V und 5V, und Sie können entweder über eine mitgelieferte Überbrückungskappe anschließen.



Schematische Darstellung

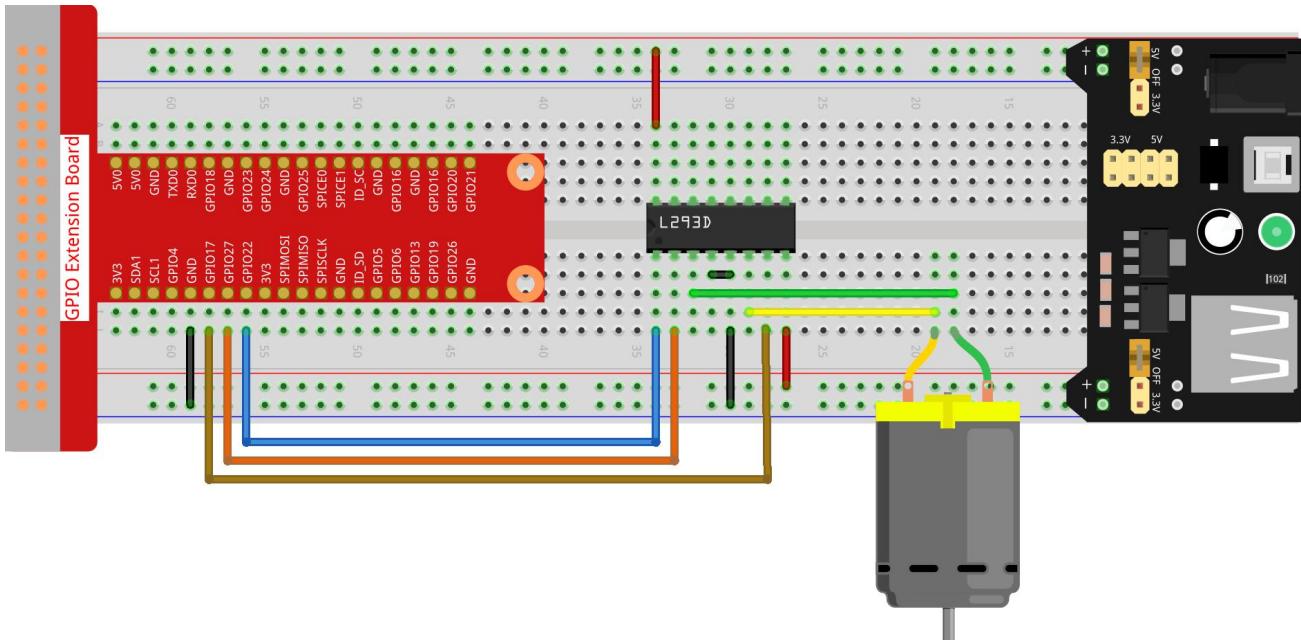
Stecken Sie das Netzteilmodul in das Steckbrett und setzen Sie die Überbrückungskappe auf 5V, dann wird eine Spannung von 5V ausgegeben. Verbinden Sie Pin 1 des L293D mit GPIO22 und stellen Sie ihn auf Hohe Niveau ein. Verbinden Sie Pin2 mit GPIO27 und Pin7 mit GPIO17 und setzen Sie dann einen Pin hoch, während der andere niedrig ist. So können Sie die Drehrichtung des Motors ändern.

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



Hinweis: Das Leistungsmodul kann eine 9-V-Batterie mit der im Kit enthaltenen 9V-Batterieschnalle anlegen. Setzen Sie die Überbrückungskappe des Leistungsmoduls in die 5V-Busleisten des Steckbretts ein.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie in den Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/1.3.1/
```

Schritt 3: Kompilieren.

```
gcc 1.3.1_Motor.c -lwiringPi
```

Schritt 4: Führen Sie die obige ausführbare Datei aus.

```
sudo ./a.out
```

Während die Kode läuft, dreht sich der Motor zuerst 5 Sekunden lang im Uhrzeigersinn und stoppt dann 5 Sekunden lang. Danach dreht er sich 5 Sekunden lang gegen den Uhrzeigersinn. Anschließend stoppt der Motor für 5S. Diese Reihe von Aktionen wird wiederholt ausgeführt.

Kode

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

#define MotorPin1      0
#define MotorPin2      2
#define MotorEnable     3

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

    pinMode(MotorPin1, OUTPUT);
    pinMode(MotorPin2, OUTPUT);
    pinMode(MotorEnable, OUTPUT);

    while(1){
        printf("Clockwise\n");
        digitalWrite(MotorEnable, HIGH);
        digitalWrite(MotorPin1, HIGH);
        digitalWrite(MotorPin2, LOW);
        for(i=0;i<3;i++){
            delay(1000);
        }

        printf("Stop\n");
        digitalWrite(MotorEnable, LOW);
        for(i=0;i<3;i++){
            delay(1000);
        }

        printf("Anti-clockwise\n");
        digitalWrite(MotorEnable, HIGH);
        digitalWrite(MotorPin1, LOW);
        digitalWrite(MotorPin2, HIGH);
        for(i=0;i<3;i++){
            delay(1000);
        }
    }
}
```

```
printf("Stop\n");
digitalWrite(MotorEnable, LOW);
for(i=0;i<3;i++){
    delay(1000);
}
return 0;
}
```

Kode Erklärung

```
digitalWrite(MotorEnable, HIGH);
```

Aktivieren Sie den L239D.

```
digitalWrite(MotorPin1, HIGH);
digitalWrite(MotorPin2, LOW);
```

Stellen Sie einen hohen Niveau für 2A ein (Pin 7); Da sich 1,2EN (Pin 1) auf einem hohen Niveau befindet, gibt 2Y einen hohen Niveau aus.

Stellen Sie einen niedrigen Niveau für 1A ein, dann gibt 1Y einen niedrigen Niveau aus und der Motor dreht sich.

```
for(i=0;i<3;i++){
    delay(1000);
}
```

Diese Schleife soll 3 * 1000ms verzögern.

```
digitalWrite(MotorEnable, LOW)
```

Wenn sich 1,2EN (Pin1) auf einem niedrigen Niveau befindet, funktioniert L293D nicht. Motor stoppt sich zu drehen.

```
digitalWrite(MotorPin1, LOW)
digitalWrite(MotorPin2, HIGH)
```

Den Stromfluss des Motors umkehren, dann dreht sich der Motor umgekehrt.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie in den Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Schritt 3: Ausführen.

```
sudo python3 1.3.1_Motor.py
```

Während die Kode läuft, dreht sich der Motor zuerst 5 Sekunden lang im Uhrzeigersinn und stoppt dann 5 Sekunden lang. Danach dreht er sich 5 Sekunden lang gegen den Uhrzeigersinn. Anschließend stoppt der Motor für 5S. Diese Reihe von Aktionen wird wiederholt ausgeführt.

Kode

```
import RPi.GPIO as GPIO
import time

# Set up pins
MotorPin1    = 17
MotorPin2    = 27
MotorEnable = 22

def setup():
    # Set the GPIO modes to BCM Numbering
    GPIO.setmode(GPIO.BCM)
    # Set pins to output
    GPIO.setup(MotorPin1, GPIO.OUT)
    GPIO.setup(MotorPin2, GPIO.OUT)
    GPIO.setup(MotorEnable, GPIO.OUT, initial=GPIO.LOW)

# Define a motor function to spin the motor
# direction should be
# 1(clockwise), 0(stop), -1(counterclockwise)
def motor(direction):
    # Clockwise
    if direction == 1:
        # Set direction
        GPIO.output(MotorPin1, GPIO.HIGH)
        GPIO.output(MotorPin2, GPIO.LOW)
        # Enable the motor
        GPIO.output(MotorEnable, GPIO.HIGH)
```

```

print ("Clockwise")
# Counterclockwise
if direction == -1:
    # Set direction
    GPIO.output(MotorPin1, GPIO.LOW)
    GPIO.output(MotorPin2, GPIO.HIGH)
    # Enable the motor
    GPIO.output(MotorEnable, GPIO.HIGH)
    print ("Counterclockwise")
# Stop
if direction == 0:
    # Disable the motor
    GPIO.output(MotorEnable, GPIO.LOW)
    print ("Stop")

def main():
    # Define a dictionary to make the script more readable
    # CW as clockwise, CCW as counterclockwise, STOP as stop
    directions = {'CW': 1, 'CCW': -1, 'STOP': 0}
    while True:
        # Clockwise
        motor(directions['CW'])
        time.sleep(5)
        # Stop
        motor(directions['STOP'])
        time.sleep(5)
        # Anticlockwise
        motor(directions['CCW'])
        time.sleep(5)
        # Stop
        motor(directions['STOP'])
        time.sleep(5)

def destroy():
    # Stop the motor
    GPIO.output(MotorEnable, GPIO.LOW)
    # Release resource
    GPIO.cleanup()

# If run this script directly, do:

```

```
if __name__ == '__main__':
    setup()
    try:
        main()
    # When 'Ctrl+C' is pressed, the program
    # destroy() will be executed.
    except KeyboardInterrupt:
        destroy()
```

Kode Erklärung

```
def motor(direction):
    # Clockwise
    if direction == 1:
        # Set direction
        GPIO.output(MotorPin1, GPIO.HIGH)
        GPIO.output(MotorPin2, GPIO.LOW)
        # Enable the motor
        GPIO.output(MotorEnable, GPIO.HIGH)
        print ("Clockwise")
    ...
    ...
```

Erstellen Sie eine Funktion, motor (), deren Variable die Richtung ist. Wenn die Bedingung erfüllt ist, dass Richtung = 1 erfüllt ist, dreht sich der Motor im Uhrzeigersinn. Wenn die Richtung = -1 ist, dreht sich der Motor gegen den Uhrzeigersinn. und unter der Bedingung, dass Richtung = 0 ist, hört es auf, sich zu drehen.

```
def main():
    # Define a dictionary to make the script more readable
    # CW as clockwise, CCW as counterclockwise, STOP as stop
    directions = {'CW': 1, 'CCW': -1, 'STOP': 0}
    while True:
        # Clockwise
        motor(directions['CW'])
        time.sleep(5)
        # Stop
        motor(directions['STOP'])
        time.sleep(5)
        # Anticlockwise
        motor(directions['CCW'])
```

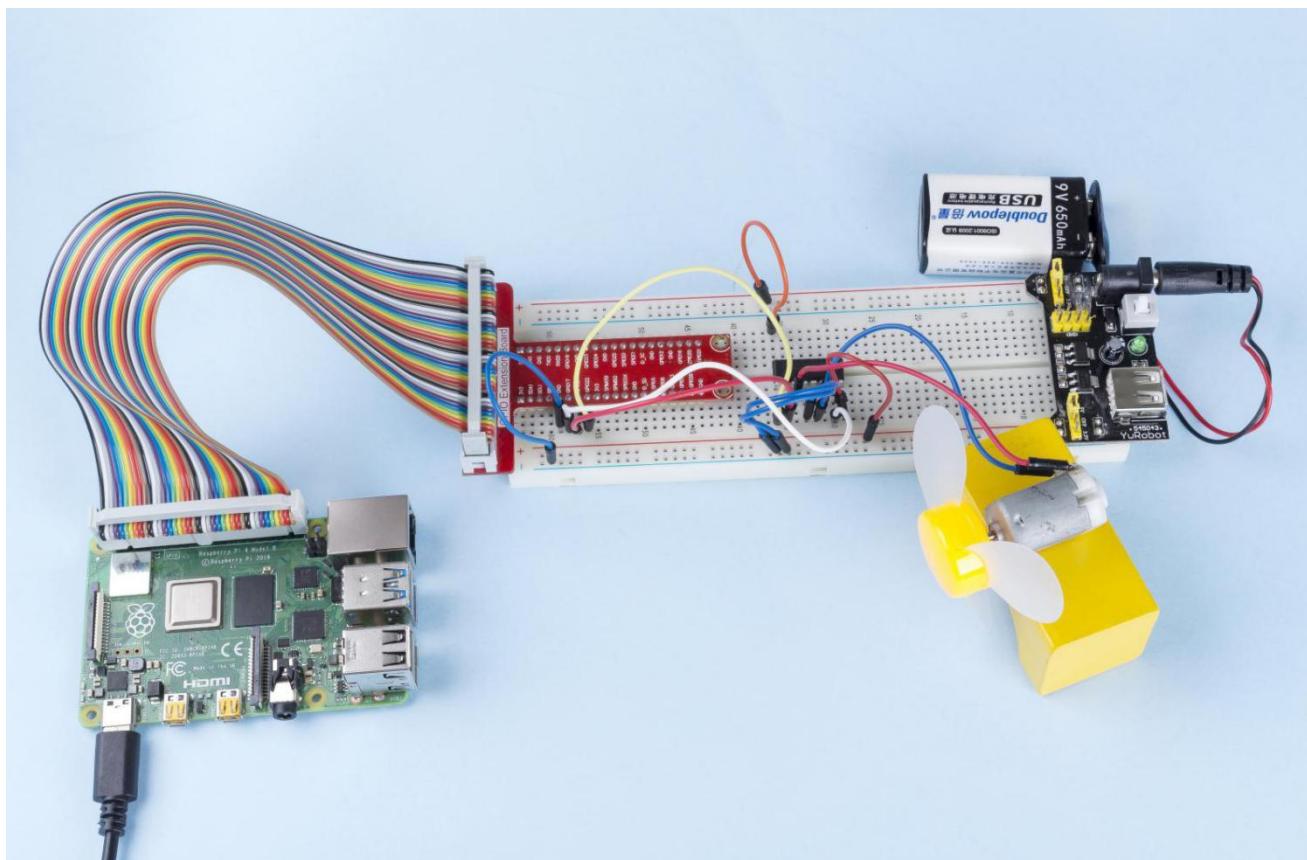
```
time.sleep(5)  
# Stop  
motor(directions['STOP'])  
time.sleep(5)
```

In der Hauptfunktion()-erstellen Sie ein Array, Richtungen[], in dem CW gleich 1 ist, der Wert von CCW -1 ist und die Nummer 0 auf Stop verweist.

Während die Kode läuft, dreht sich der Motor zuerst 5 Sekunden lang im Uhrzeigersinn und stoppt dann 5 Sekunden lang. Danach dreht er sich 5 Sekunden lang gegen den Uhrzeigersinn. Anschließend stoppt der Motor für 5s. Diese Reihe von Aktionen wird wiederholt ausgeführt.

Jetzt sollte sich das Motorblatt drehen.

Phänomen Bild

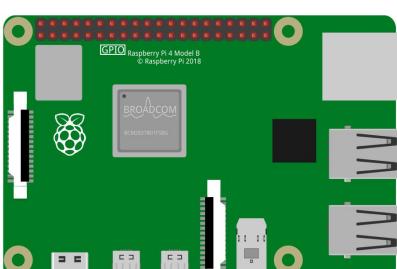
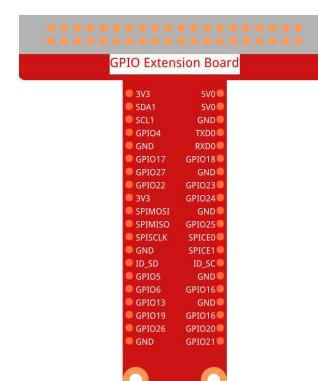
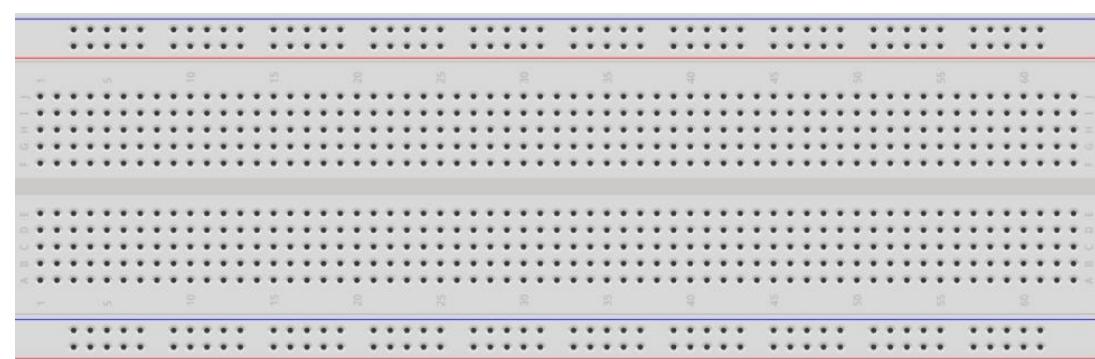


1.3.2 Servo

Einführung

In dieser Lektion lernen wir, wie man das Servo dreht.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Servo																																																									
	 <table border="1"> <tr><td>3V3</td><td>SDA1</td><td>5V0</td></tr> <tr><td>SCL1</td><td>5V0</td><td>GND</td></tr> <tr><td>GPIO4</td><td>TX00</td><td></td></tr> <tr><td>GND</td><td>RX00</td><td></td></tr> <tr><td>GPIO17</td><td>GPIO18</td><td></td></tr> <tr><td>GPIO27</td><td>GND</td><td></td></tr> <tr><td>GPIO22</td><td>GPIO23</td><td></td></tr> <tr><td>3V3</td><td>GPIO24</td><td></td></tr> <tr><td>SPI MOSI</td><td>GND</td><td></td></tr> <tr><td>SPI MISO</td><td>GPIO25</td><td></td></tr> <tr><td>SPI SCLK</td><td>SPI CE0</td><td></td></tr> <tr><td>GND</td><td>SPI CE1</td><td></td></tr> <tr><td>ID_SD</td><td>ID_SC</td><td></td></tr> <tr><td>GPIO5</td><td>GND</td><td></td></tr> <tr><td>GPIO6</td><td>GPIO16</td><td></td></tr> <tr><td>GPIO13</td><td>GND</td><td></td></tr> <tr><td>GPIO19</td><td>GPIO16</td><td></td></tr> <tr><td>GPIO26</td><td>GPIO20</td><td></td></tr> <tr><td>GND</td><td>GPIO21</td><td></td></tr> </table>	3V3	SDA1	5V0	SCL1	5V0	GND	GPIO4	TX00		GND	RX00		GPIO17	GPIO18		GPIO27	GND		GPIO22	GPIO23		3V3	GPIO24		SPI MOSI	GND		SPI MISO	GPIO25		SPI SCLK	SPI CE0		GND	SPI CE1		ID_SD	ID_SC		GPIO5	GND		GPIO6	GPIO16		GPIO13	GND		GPIO19	GPIO16		GPIO26	GPIO20		GND	GPIO21		
3V3	SDA1	5V0																																																									
SCL1	5V0	GND																																																									
GPIO4	TX00																																																										
GND	RX00																																																										
GPIO17	GPIO18																																																										
GPIO27	GND																																																										
GPIO22	GPIO23																																																										
3V3	GPIO24																																																										
SPI MOSI	GND																																																										
SPI MISO	GPIO25																																																										
SPI SCLK	SPI CE0																																																										
GND	SPI CE1																																																										
ID_SD	ID_SC																																																										
GPIO5	GND																																																										
GPIO6	GPIO16																																																										
GPIO13	GND																																																										
GPIO19	GPIO16																																																										
GPIO26	GPIO20																																																										
GND	GPIO21																																																										
Mehrere Überbrückungsdrähte																																																											
1 * 40-Pin Kabel																																																											
1 * Steckbrett																																																											

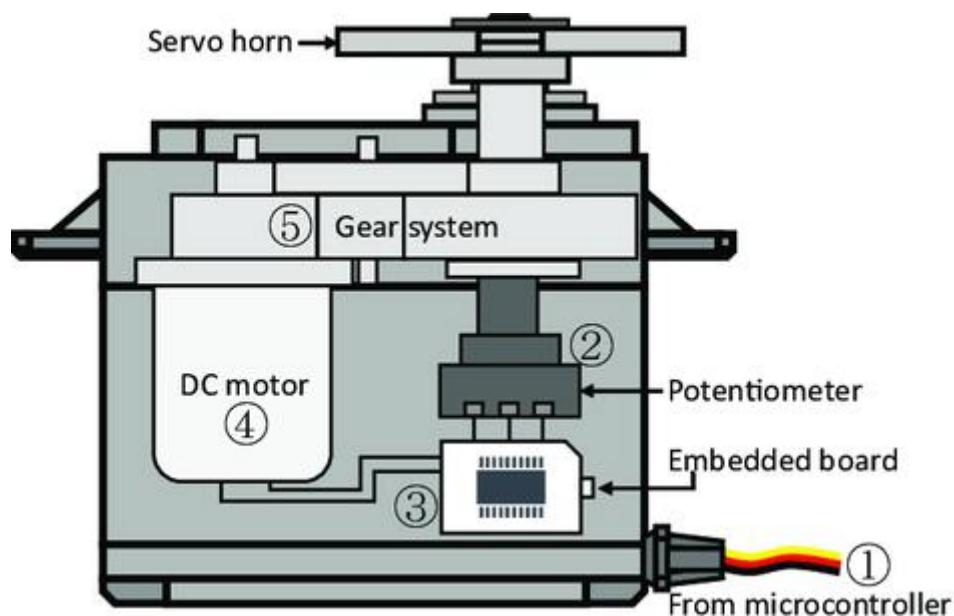
Prinzip

Servo

Ein Servo besteht im Allgemeinen aus folgenden Teilen: Gehäuse, Welle, Getriebe, Potentiometer, Gleichstrommotor und eingebettete Platine.

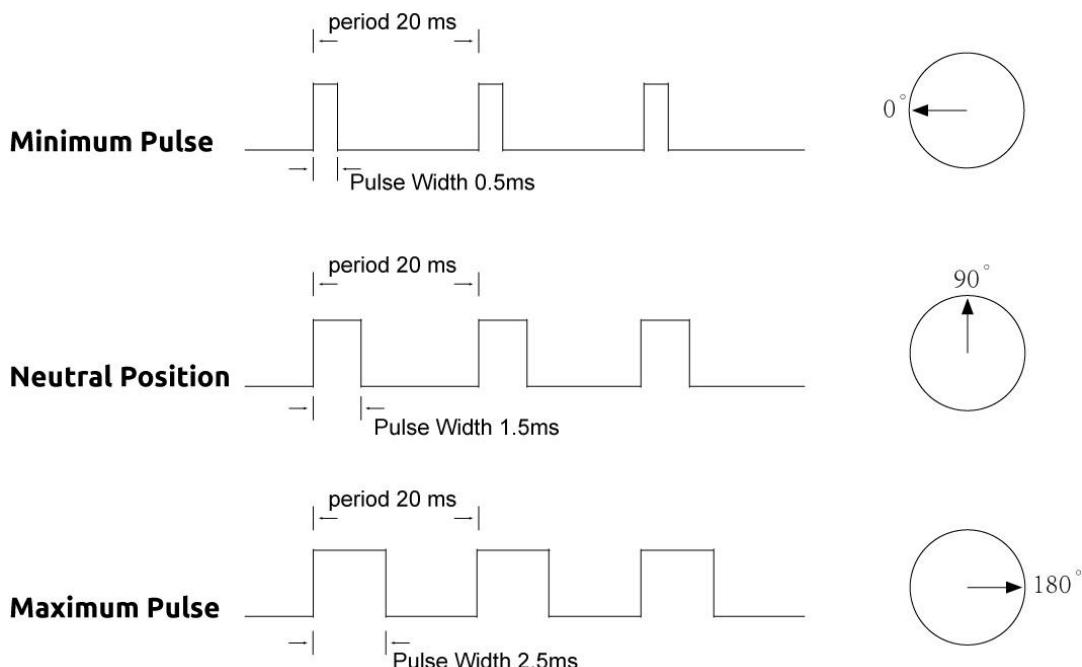


Das funktioniert so: Der Mikrocontroller sendet PWM-Signale an das Servo, und dann empfängt die im Servo eingebettete Karte die Signale über den Signal Pin und steuert den Motor im Inneren, um sich zu drehen. Infolgedessen treibt der Motor das Zahnradssystem an und motiviert dann die Welle nach dem Abbremsen. Die Welle und das Potentiometer des Servos sind miteinander verbunden. Wenn sich die Welle dreht, treibt sie das Potentiometer an, sodass das Potentiometer ein Spannungssignal an die eingebettete Platine ausgibt. Dann bestimmt das Board die Richtung und Geschwindigkeit der Drehung basierend auf der aktuellen Position, so dass es genau an der richtigen Position wie definiert anhalten und dort halten kann.



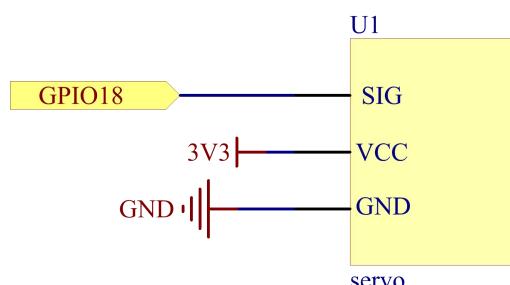
Der Winkel wird durch die Dauer eines Impulses bestimmt, der an den Steuerdraht angelegt wird. Dies wird als Pulsweitenmodulation bezeichnet. Das Servo erwartet alle 20 ms einen Impuls. Die Länge des Impulses bestimmt, wie weit sich der Motor dreht. Zum Beispiel bringt ein Impuls von 1,5 ms den Motor in die 90-Grad-Position (neutrale Position).

Wenn ein Impuls an ein Servo gesendet wird, das weniger als 1,5 ms beträgt, dreht sich das Servo in eine Position und hält seine Ausgangswelle einige Grad gegen den Uhrzeigersinn vom Neutralpunkt entfernt. Wenn der Impuls breiter als 1,5 ms ist, tritt das Gegenteil auf. Die minimale Breite und die maximale Impulsbreite, die das Servo anweisen, sich in eine gültige Position zu drehen, sind Funktionen jedes Servos. **Im Allgemeinen ist der minimale Impuls ungefähr 0,5 ms breit und der maximale Impuls ist 2,5 ms breit.**



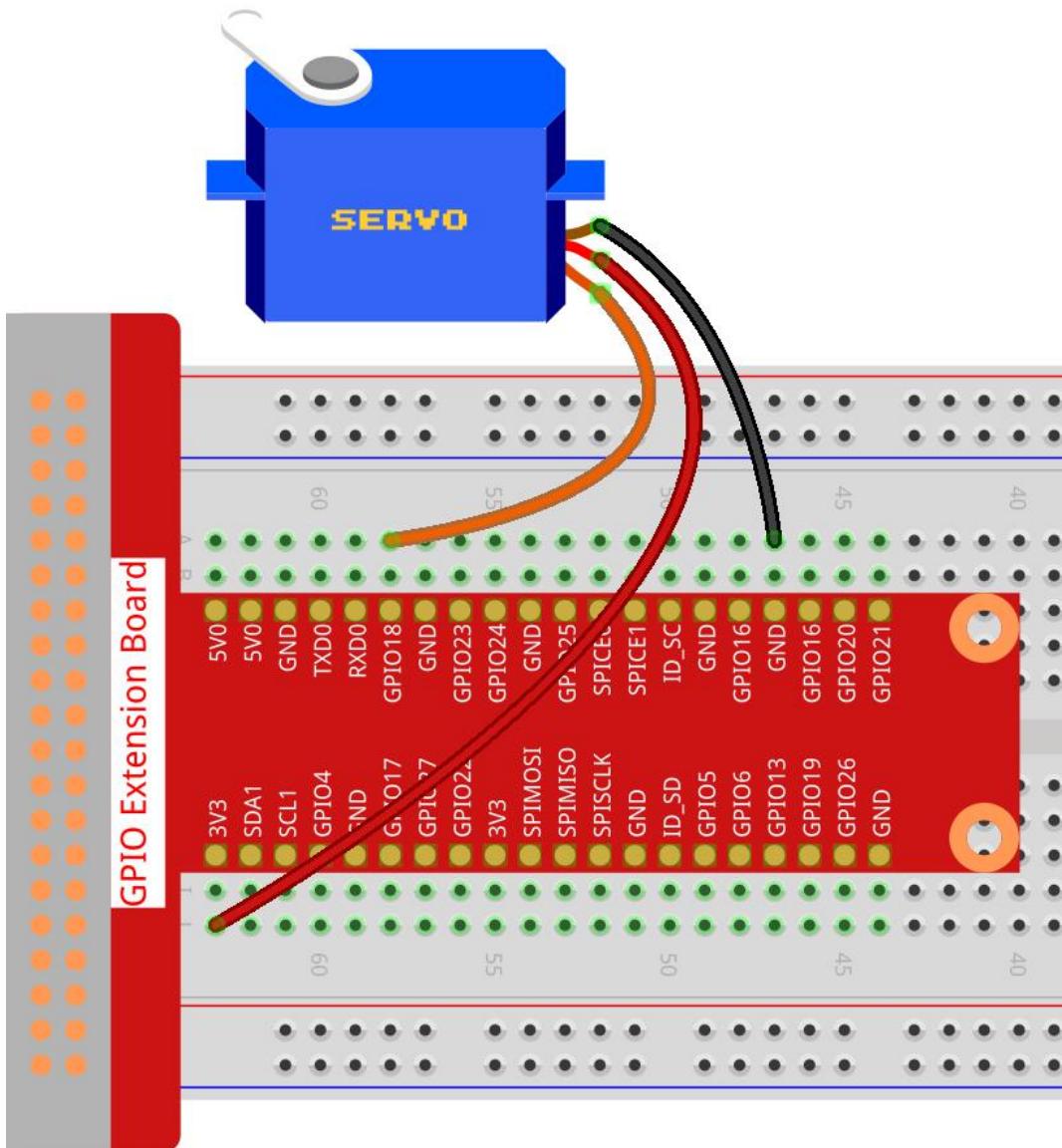
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO18	Pin 12	1	18



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner des Codes.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/1.3.2
```

Schritt 3: Kompilieren Sie den Code.

gcc 1.3.2_Servo.c -lwiringPi

Schritt 4: Führen Sie die ausführbare Datei aus.

sudo ./a.out

Nachdem das Programm ausgeführt wurde, dreht sich das Servo kreisförmig von 0 Grad auf 180 Grad und dann von 180 Grad auf 0 Grad.

Kode

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

#define ServoPin 1           //define the servo to GPIO1
long Map(long value,long fromLow,long fromHigh,long toLow,long toHigh){
    return (toHigh-toLow)*(value-fromLow) / (fromHigh-fromLow) + toLow;
}
void setAngle(int pin, int angle){ //Create a funtion to control the angle of the servo.
    if(angle < 0)
        angle = 0;
    if(angle > 180)
        angle = 180;
    softPwmWrite(pin,Map(angle, 0, 180, 5, 25));
}

int main(void)
{
    int i;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    softPwmCreate(ServoPin, 0, 200); //initialize PMW pin of servo
    while(1){
        for(i=0;i<181;i++){ // Let servo rotate from 0 to 180.
            setAngle(ServoPin,i);
            delay(2);
        }
        delay(1000);
        for(i=181;i>-1;i--){ // Let servo rotate from 180 to 0.
            setAngle(ServoPin,i);
            delay(2);
        }
        delay(1000);
    }
    return 0;
}
```

Kode Erklärung

```
long Map(long value,long fromLow,long fromHigh,long toLow,long toHigh){  
    return (toHigh-toLow)*(value-fromLow) / (fromHigh-fromLow) + toLow;  
}
```

Erstellen Sie eine Map() - Funktion, um den Wert im folgenden Kode zuzuordnen.

```
void setAngle(int pin, int angle){ //Create a function to control the angle of the servo.  
    if(angle < 0)  
        angle = 0;  
    if(angle > 180)  
        angle = 180;  
    softPwmWrite(pin,Map(angle, 0, 180, 5, 25));  
}
```

Erstellen Sie eine Funktion, setAngle (), um den Winkel zum Servo zu schreiben.

```
softPwmWrite(pin,Map(angle,0,180,5,25));
```

Diese Funktion kann das Einschaltzeitdauer der PWM ändern.

Damit sich das Servo auf 0 bis 180 ° dreht, sollte sich die Impulsbreite im Bereich von 0,5 ms bis 2,5 ms ändern, wenn die Periode 20 ms beträgt. In der Funktion softPwmCreate () haben wir festgelegt, dass der Zeitraum 200x100us = 20ms beträgt. Daher müssen wir 0 ~ 180 bis 5x100us ~ 25x100us zuordnen.

Der Prototyp dieser Funktion ist unten dargestellt.

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

Parameter-Pin: Jeder GPIO-Pin von Raspberry Pi kann als PWM-Pin gesetzt werden.

Parameter initialValue: Die anfängliche Impulsbreite ist der initialValue mal 100us.

Parameter pwmRange: Die Periode von PWM ist die pwmRange mal 100us.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner des Codes.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 1.3.2_Servo.py
```

Nachdem das Programm ausgeführt wurde, dreht sich das Servo kreisförmig von 0 Grad auf 180 Grad und dann von 180 Grad auf 0 Grad.

Kode

```

import RPi.GPIO as GPIO
import time

SERVO_MIN_PULSE = 500
SERVO_MAX_PULSE = 2500
ServoPin = 18

def map(value, inMin, inMax, outMin, outMax):
    return (outMax - outMin) * (value - inMin) / (inMax - inMin) + outMin

def setup():
    global p
    GPIO.setmode(GPIO.BCM)      # Numbers GPIOs by BCM
    GPIO.setup(ServoPin, GPIO.OUT)  # Set ServoPin's mode is output
    GPIO.output(ServoPin, GPIO.LOW) # Set ServoPin to low
    p = GPIO.PWM(ServoPin, 50)    # set Frequency to 50Hz
    p.start(0)                  # Duty Cycle = 0

def setAngle(angle):      # make the servo rotate to specific angle (0-180 degrees)
    angle = max(0, min(180, angle))
    pulse_width = map(angle, 0, 180, SERVO_MIN_PULSE, SERVO_MAX_PULSE)
    pwm = map(pulse_width, 0, 20000, 0, 100)
    p.ChangeDutyCycle(pwm)#map the angle to duty cycle and output it

def loop():
    while True:
        for i in range(0, 181, 5):  #make servo rotate from 0 to 180 deg
            setAngle(i)      # Write to servo
            time.sleep(0.002)
        time.sleep(1)
        for i in range(180, -1, -5): #make servo rotate from 180 to 0 deg
            setAngle(i)
            time.sleep(0.001)
        time.sleep(1)

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

if __name__ == '__main__':    #Program start from here

```

```

setup()
try:
    loop()
except KeyboardInterrupt: # When 'Ctrl+C' is pressed, the program destroy() will be
executed.
destroy()

```

Kode Erklärung

```

p = GPIO.PWM(ServoPin, 50)      # set Frequency to 50Hz
p.start(0)                      # Duty Cycle = 0

```

Stellen Sie den ServoPin auf den PWM-Pin, dann die Frequenz auf 50 Hz und die Periode auf 20 ms.

p.start (0): Führen Sie die PWM-Funktion, und setzen Sie den Anfangswert auf 0.

```

def setAngle(angle):      # make the servo rotate to specific angle (0-180 degrees)
    angle = max(0, min(180, angle))
    pulse_width = map(angle, 0, 180, SERVO_MIN_PULSE, SERVO_MAX_PULSE)
    pwm = map(pulse_width, 0, 20000, 0, 100)
    p.ChangeDutyCycle(pwm)#map the angle to duty cycle and output it

```

Erstellen Sie eine Funktion, setAngle (), um einen Winkel von 0 bis 180 in das Servo zu schreiben.

```

Winkel = max (0, min (180, Winkel))

```

Diese Kode wird verwendet, um den Winkel im Bereich von 0 bis 180 ° zu begrenzen.

Die Funktion min () gibt das Minimum der Eingabewerte zurück. Wenn $180 < \text{Winkel}$, dann 180 zurückgeben, wenn nicht, Winkel zurückgeben .

Die max () -Methode gibt das maximale Element in einem iterierbaren oder größten von zwei oder mehr Parametern zurück. Wenn $0 > \text{Winkel}$, dann 0 zurückgeben, wenn nicht, Winkel zurückgeben.

```

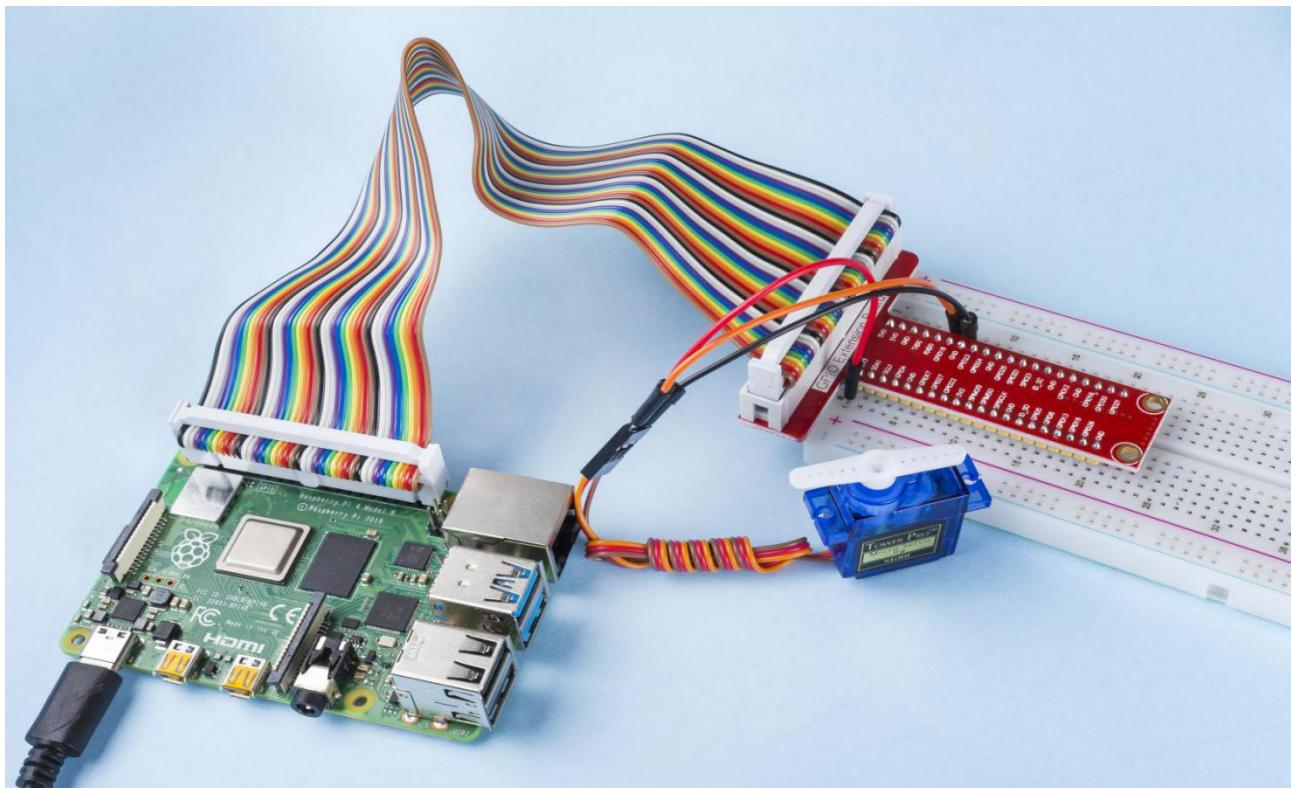
pulse_width = map(angle, 0, 180, SERVO_MIN_PULSE, SERVO_MAX_PULSE)
pwm = map(pulse_width, 0, 20000, 0, 100)
p.ChangeDutyCycle(pwm)

```

Um einen Bereich von 0 bis 180 ° zum Servo zu rendern, wird die Impulsbreite des Servos auf 0,5 ms (500 us) bis 2,5 ms (2500 us) eingestellt.

Die Periode der PWM beträgt 20 ms (20000us), daher beträgt das Tastverhältnis der PWM $(500/20000)\% - (2500/20000)\%$, und der Bereich 0 bis 180 wird auf 2,5 bis 12,5 abgebildet.

Phänomen Bild



1.3.3 Schrittmotor

Einführung

Schrittmotoren können aufgrund ihres einzigartigen Designs ohne Rückkopplungsmechanismen mit hoher Genauigkeit gesteuert werden. Die Welle eines Schrittmachers, der mit einer Reihe von Magneten montiert ist, wird von einer Reihe elektromagnetischer Spulen gesteuert, die in einer bestimmten Reihenfolge positiv und negativ geladen werden und diese in kleinen "Schritten" präzise vorwärts oder rückwärts bewegen.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Schrittmotor
	 GPIO Extension Board Pinout: 3V3 SDA1 SVO SDA2 SVO SCL1 GND SCL2 GND GND RXD0 GPIO17 GPIO18 GND GPIO27 GPIO28 GND GPIO22 GPIO23 GND 3V3 GPIO25 GND SPIMOSI GPIO26 GND SPISCLK SPICEO GND GND SPICEO GND ID_SD ID_SC GND GPIO5 GPIO16 GND GPIO6 GPIO17 GND GPIO13 GPIO18 GND GPIO19 GPIO16 GND GPIO26 GPIO23 GND GND GPIO22 GND	
1 * 40-Pin Kabel		Mehrere Überbrückungsdrähte
1 * Steckbrett		1 * ULN2003

Prinzip

Schrittmotor

Es gibt zwei Typen von Steppern, Unipolare und Bipolare, und es ist sehr wichtig zu wissen, mit welchem Typ Sie arbeiten. In diesem Experiment verwenden wir einen unipolaren Stepper.

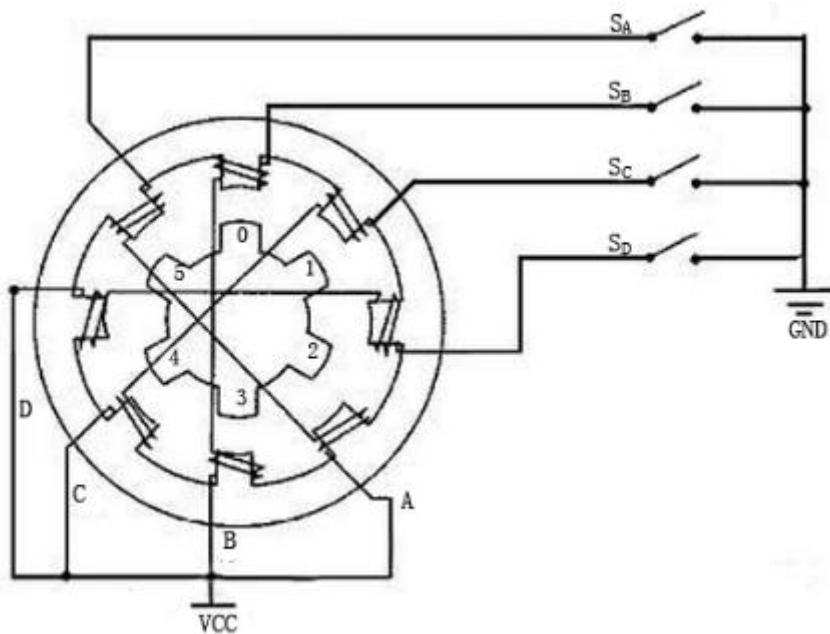
Der Schrittmotor ist ein Vierphasenmotor, der eine Gleichstromversorgung mit Unipolarität verwendet. Solange Sie alle Phasenwicklungen des Motors durch eine geeignete Zeitfolge elektrifizieren, können Sie ihn Schritt für Schritt drehen lassen. Das schematische Diagramm eines vierphasigen reaktiven Schrittmotors:



In der Abbildung befindet sich in der Mitte des Motors ein Rotor - ein zahnradförmiger Permanentmagnet. Um den Rotor herum sind 0 bis 5 Zähne. Dann weiter draußen gibt es 8 Magnetpole, wobei jeweils zwei gegenüberliegende durch Spulenwicklung verbunden sind. Sie bilden also vier Paare von A nach D, was als Phase bezeichnet wird. Es verfügt über vier Anschlusskabel, die mit den Schaltern SA, SB, SC und SD verbunden werden können. Daher sind die vier Phasen in der Schaltung parallel und die zwei Magnetpole in einer Phase sind in Reihe geschaltet.

So funktioniert ein 4-Phasen-Schrittmotor:

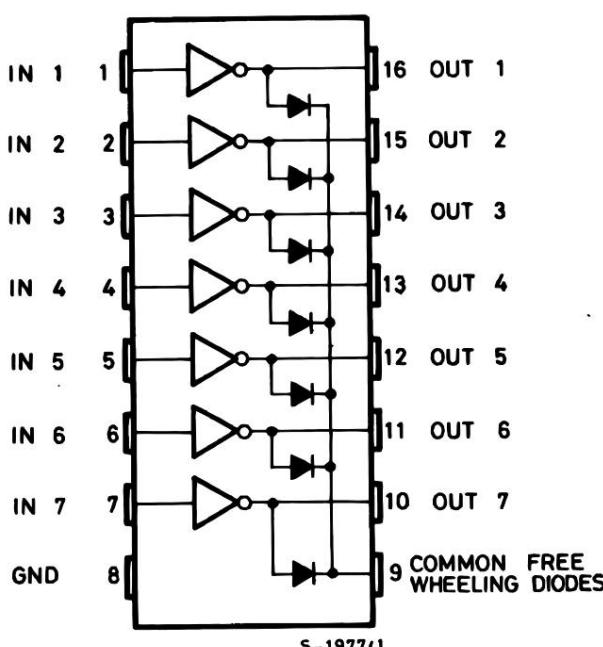
Wenn der Schalter SB eingeschaltet ist, sind die Schalter SA, SC und SD ausgeschaltet, und die B-Phasen-Magnetpole sind auf die Zähne 0 und 3 des Rotors ausgerichtet. Gleichzeitig erzeugen Zahn 1 und 4 versetzte Zähne mit C- und D-Phasenpolen. Zahn 2 und 5 erzeugen versetzte Zähne mit D- und A-Phasenpolen. Wenn der Schalter SC eingeschaltet ist, die Schalter SB, SA und SD ausgeschaltet sind, dreht sich der Rotor unter dem Magnetfeld der C-Phasenwicklung und dem zwischen Zahn 1 und 4. Dann richten sich Zahn 1 und 4 an den Magnetpolen der C-Phasenwicklung aus. Während Zahn 0 und 3 versetzte Zähne mit A- und B-Phasenpolen erzeugen, erzeugen Zahn 2 und 5 versetzte Zähne mit den Magnetpolen von A- und D-Phasenpolen. Die ähnliche Situation geht weiter und weiter. Schalten Sie die Phasen A, B, C und D nacheinander ein, und der Rotor dreht sich in der Reihenfolge A, B, C und D.



Der Vierphasen-Schrittmotor verfügt über drei Betriebsarten: einfach vierstufig, doppelt vierstufig und achtstufig. Der Schrittewinkel für den einzelnen vierstufigen und den doppelten vierstufigen ist gleich, aber das Antriebsmoment für den einzelnen vierstufigen ist kleiner. Der Schrittewinkel des Achtstufens ist halb so groß wie der des Einzel-Vier-Stufen- und des Doppel-Vier-Stufen-Winkels. Somit kann der achtstufige Betriebsmodus ein hohes Antriebsmoment beibehalten und die Steuergenauigkeit verbessern.

Der von uns verwendete Stator des 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.**

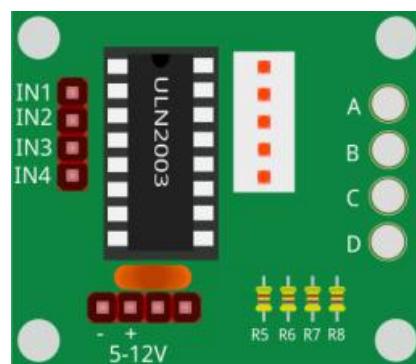
ULN2003



Um den Motor in der Schaltung anzubringen, muss eine Treiberplatine verwendet werden. Der Schrittmotortreiber-ULN2003 ist eine 7-Kanal-Wechselrichterschaltung. Das heißt, wenn sich der Eingangspin auf einem hohen Niveau befindet, befindet sich der Ausgangspin von ULN2003 auf einem niedrigen Niveau und umgekehrt. Wenn wir IN1 mit hohem Niveau und IN2, IN3 und IN4 mit niedrigem Niveau versorgen, befindet sich das Ausgangsende OUT1 auf niedrigem Niveau und alle anderen Ausgangsende auf hohem Niveau.

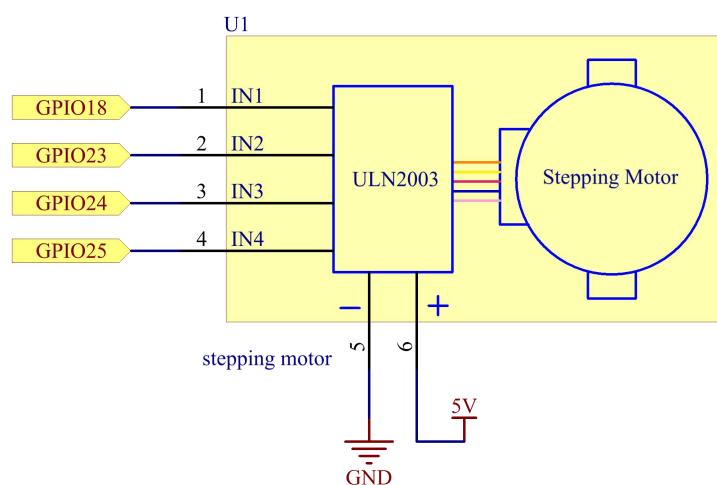
Die interne Struktur des Chips ist wie folgt dargestellt.

Der Schrittmotortreiber, der aus dem ULN2003-Chip und 4 LEDs besteht, ist wie folgt dargestellt. Auf der Platine fungieren IN1, IN2, IN3 und IN4 als Eingang und die vier LEDs A, B, C, D sind die Anzeigen des Eingangspins. Zusätzlich sind OUT1, OUT2, OUT3 und OUT4 mit SA, SB, SC und SD am Schrittmotortreiber verbunden. Wenn der Wert von IN1 auf einen hohen Niveau eingestellt ist, leuchtet A auf; Schalter SA ist eingeschaltet und der Schrittmotor dreht sich einen Schritt. Der ähnliche Fall wiederholt sich immer weiter. Geben Sie dem Schrittmotor daher einfach eine bestimmte Zeitfolge, er dreht sich Schritt für Schritt. Der ULN2003 wird hier verwendet, um bestimmte Zeitabläufe für den Schrittmotor bereitzustellen.



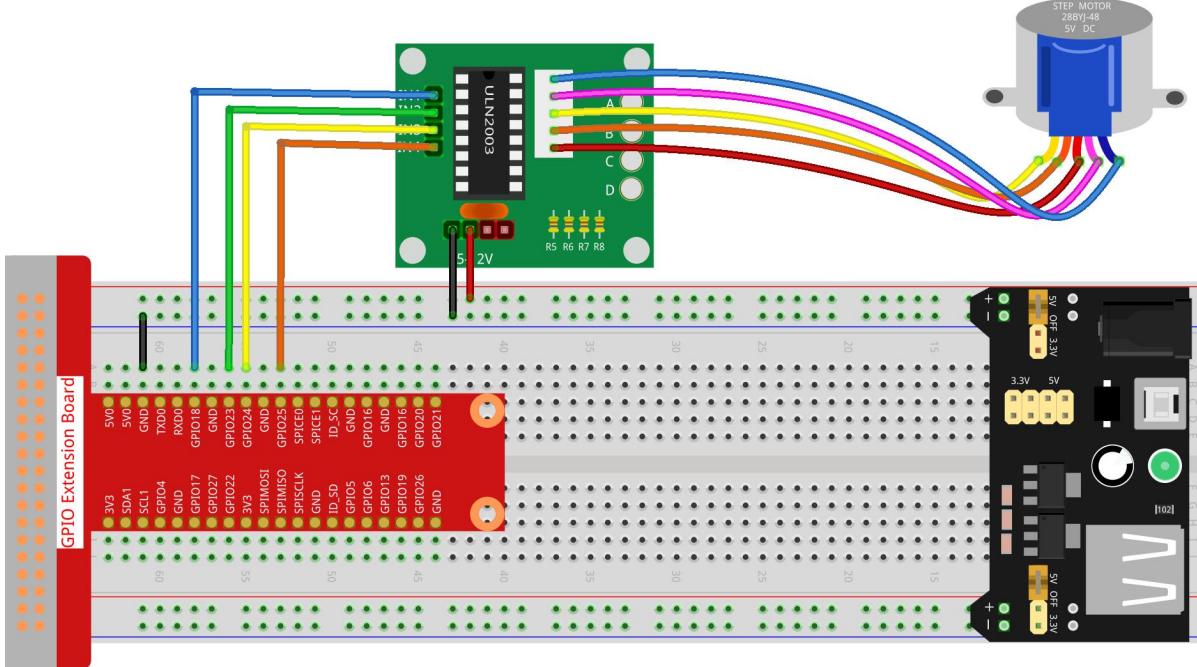
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO18	Pin 12	1	18
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24
GPIO25	Pin 22	6	25



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner des Codes.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/1.3.3/
```

Schritt 3: Kompilieren Sie den Code.

```
gcc 1.3.3_StepperMotor.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

sudo ./a.out

Während der Code ausgeführt wird, dreht sich der Schrittmotor je nach Eingabe 'a' oder 'c' im oder gegen den Uhrzeigersinn.

Kode

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

const int motorPin[] = {1, 4, 5, 6};
int rolePerMinute = 15;
int stepsPerRevolution = 2048;
int stepSpeed = 0;
```

```

void rotary(char direction){
    if(direction == 'c'){
        for(int j=0;j<4;j++){
            for(int i=0;i<4;i++)
                {digitalWrite(motorPin[i],0x99>>j & (0x08>>i));}
            delayMicroseconds(stepSpeed);
        }
    }
    else if(direction =='a'){
        for(int j=0;j<4;j++){
            for(int i=0;i<4;i++)
                {digitalWrite(motorPin[i],0x99<<j & (0x80>>i));}
            delayMicroseconds(stepSpeed);
        }
    }
}

void loop()
{
    char direction = '0';
    while (1)
    {
        printf("select motor direction a=anticlockwise, c=clockwise: ");
        direction=getchar();
        if (direction == 'c')
        {
            printf("motor running clockwise\n");
            break;
        }
        else if (direction == 'a')
        {
            printf("motor running anti-clockwise\n");
            break;
        }
        else
        {
            printf("input error, please try again!\n");
        }
    }
    while(1)
}

```

```

    {
        rotary(direction);
    }
}

void main(void)
{
    if (wiringPiSetup() == -1)
    {
        printf("setup wiringPi failed !");
        return;
    }
    for (int i = 0; i < 4; i++)
    {
        pinMode(motorPin[i], OUTPUT);
    }
    stepSpeed = (60000000 / rolePerMinute) / stepsPerRevolution;
    loop();
}

```

Kode Erklärung

```

int rolePerMinute = 15;
int stepsPerRevolution = 2048;
int stepSpeed = 0;

```

rolePerMinute: Umdrehungen pro Minute sollte die Drehzahl des in diesem Kit verwendeten Schrittmotors 0 bis 17 betragen.

stepPerRevolution: Die Anzahl der Schritte für jede Umdrehung und der in diesem Kit verwendete Schrittmotor benötigen 2048 Schritte pro Umdrehung.

stepSpeed: Die für jeden Schritt verwendete Zeit. In main () weisen wir ihnen die folgenden Werte zu: $\lceil (60000000 / \text{rolePerMinute}) / \text{stepPerRevolution} \rceil$ (60.000.000 us = 1 Minute)

```

void loop()
{
    char direction = '0';
    while (1)

```

```
{
    printf("select motor direction a=anticlockwise, c=clockwise: ");
    direction=getchar();
    if (direction == 'c')
    {
        printf("motor running clockwise\n");
        break;
    }
    else if (direction == 'a')
    {
        printf("motor running anti-clockwise\n");
        break;
    }
    else
    {
        printf("input error, please try again!\n");
    }
}
while(1)
{
    rotary(direction);
}
}
```

Die Funktion loop () ist grob in zwei Teile unterteilt (zwischen zwei während (1)): Der erste Teil besteht darin, den Schlüsselwert zu erhalten. Wenn 'a' oder 'c' erhalten wird, verlassen Sie die Schleife und stoppen Sie die Eingabe.
Der zweite Teil ruft Drehung (Richtung) auf, um den Schrittmotor laufen zu lassen.

```
void rotary(char direction){
    if(direction == 'c'){
        for(int j=0;j<4;j++){
            for(int i=0;i<4;i++)
                {digitalWrite(motorPin[i],0x99>>j & (0x08>>i));}
            delayMicroseconds(stepSpeed);
        }
    }
    else if(direction =='a'){
        for(int j=0;j<4;j++){
            for(int i=0;i<4;i++)
                {digitalWrite(motorPin[i],0x99<<j & (0x80>>i));}
    }}
```

```
    delayMicroseconds(stepSpeed);  
}  
}  
}
```

Damit sich der Schrittmotor **im Uhrzeigersinn** dreht, sollte der Füllstandsstatus von motorPin in der folgenden Tabelle angezeigt werden:

	MotorPin A.	MotorPin B.	MotorPin C.	MotorPin D.
Schritt 1	HOCH	NIEDRIG	NIEDRIG	HOCH
Schritt 2	HOCH	HOCH	NIEDRIG	NIEDRIG
Schritt 3	NIEDRIG	HOCH	HOCH	NIEDRIG
Schritt 4	NIEDRIG	NIEDRIG	HOCH	HOCH
Schritt 5 (Schritt 1)	HOCH	NIEDRIG	NIEDRIG	HOCH

Daher wird das potentielle Schreiben von MotorPin unter Verwendung einer zweischichtigen for-Schleife implementiert.

In Schritt 1 ist $j = 0$, $i = 0 \sim 4$.

motorPin [0] wird in der hohen Ebene geschrieben ($10011001 \& 00001000 = 1$)

motorPin [1] wird auf dem niedrigen Niveau geschrieben ($10011001 \& 00000100 = 0$)

motorPin [2] wird in der niedrigen Ebene geschrieben ($10011001 \& 00000010 = 0$)

motorPin [3] wird in der hohen Ebene geschrieben ($10011001 \& 00000001 = 1$)

In Schritt 2 ist $j = 1$, $i = 0 \sim 4$.

motorPin [0] wird in der hohen Ebene geschrieben (01001100 & 00001000 = 1)

motorPin [1] wird auf dem niedrigen Niveau geschrieben (01001100 & 00000100 = 1)

USW.

Damit sich der Schrittmotor **gegen den Uhrzeigersinn** dreht, wird der Füllstandsstatus von motorPin in der folgenden Tabelle angezeigt.

	MotorPin A.	MotorPin B.	MotorPin C.	MotorPin D.
Schritt 1	HOCH	NIEDRIG	NIEDRIG	HOCH
Schritt 2	NIEDRIG	NIEDRIG	HOCH	HOCH
Schritt 3	NIEDRIG	HOCH	HOCH	NIEDRIG
Schritt 4	HOCH	HOCH	NIEDRIG	NIEDRIG
Schritt 5 (1)	HOCH	NIEDRIG	NIEDRIG	HOCH

In Schritt 1 ist $j = 0$, $i = 0 \sim 4$.

`motorPin [0]` wird in der hohen Ebene geschrieben ($10011001 \& 10000000 = 1$)

`motorPin [1]` wird auf dem niedrigen Niveau geschrieben ($10011001 \& 01000000 = 0$)

In Schritt 2 ist , $j = 1$, $i = 0 \sim 4$.

`motorPin [0]` wird in der hohen Ebene geschrieben ($00110010 \& 10000000 = 0$)

`motorPin [1]` wird auf dem niedrigen Niveau geschrieben ($00110010 \& 01000000 = 0$)

usw.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner des Codes.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 1.3.3_StepperMotor.py
```

Während der Code läuft, dreht sich der Schrittmotor abhängig von Ihrer Eingabe 'a' oder 'c' im oder gegen den Uhrzeigersinn.

Kode

```
import RPi.GPIO as GPIO
from time import sleep
```

```
motorPin = (18,23,24,25)
rolePerMinute = 15
stepsPerRevolution = 2048
```

```

stepSpeed = (60/rolePerMinute)/stepsPerRevolution

def setup():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)
    for i in motorPin:
        GPIO.setup(i, GPIO.OUT)

def rotary(direction):
    if(direction == 'c'):
        for j in range(4):
            for i in range(4):
                GPIO.output(motorPin[i],0x99>>j & (0x08>>i))
                sleep(stepSpeed)

    elif(direction == 'a'):
        for j in range(4):
            for i in range(4):
                GPIO.output(motorPin[i],0x99<<j & (0x80>>i))
                sleep(stepSpeed)

def loop():
    while True:
        direction = input('select motor direction a=anticlockwise, c=clockwise: ')
        if(direction == 'c'):
            print('motor running clockwise\n')
            break
        elif(direction == 'a'):
            print('motor running anti-clockwise\n')
            break
        else:
            print('input error, please try again!')
    while True:
        rotary(direction)

def destroy():
    GPIO.cleanup()

if __name__ == '__main__':
    setup()

```

```
try:  
    loop()  
except KeyboardInterrupt:  
    destroy()
```

Kode Erklärung

```
rolePerMinute = 15  
SchrittePerRevolution = 2048  
SchrittGeschwindigkeit = (60 / rolePerMinute) / stepPerRevolution
```

rolePerMinute: Umdrehungen pro Minute sollte die Drehzahl des in diesem Kit verwendeten Schrittmotors 0 bis 17 betragen.

stepPerRevolution: Die Anzahl der Schritte für jede Umdrehung und der in diesem Kit verwendete Schrittmotor benötigen 2048 Schritte pro Umdrehung.

Schritt Geschwindigkeit: Die Zeit, die für jeden Schritt verwendet wird, und wir weisen ihnen die Werte zu: $\lceil (60 / \text{rolePerMinute}) / \text{stepPerRevolution} \rceil$ (60s = 1minute).

```
def loop():  
    while True:  
        direction = input('select motor direction a=anticlockwise, c=clockwise: ')  
        if(direction == 'c'):  
            print('motor running clockwise\n')  
            break  
        elif(direction == 'a'):  
            print('motor running anti-clockwise\n')  
            break  
        else:  
            print('input error, please try again!')  
    while True:  
        rotary(direction)
```

Die Funktion `loop()` ist grob in zwei Teile unterteilt (in zwei Teilen, während (1)): Der erste Teil besteht darin, den Schlüsselwert zu erhalten. Wenn 'a' oder 'c' erhalten wird, verlassen Sie die Schleife und stoppen Sie die Eingabe. Der zweite Teil ruft Drehung (Richtung) auf, um den Schrittmotor laufen zu lassen.

```
def rotary(direction):  
    if(direction == 'c'):
```

```

for j in range(4):
    for i in range(4):
        GPIO.output(motorPin[i],0x99>>j & (0x08>>i))
        sleep(stepSpeed)

    elif(direction == 'a'):
        for j in range(4):
            for i in range(4):
                GPIO.output (motorPin[i],0x99<<j & (0x80>>i))
                Schlaf (stepSpeed)

```

Um den Schrittmotor im Uhrzeigersinn drehen zu lassen, wird der Niveau status von motorPin in der folgenden Tabelle angezeigt:

	MotorPin A.	MotorPin B.	MotorPin C.	MotorPin D.
Schritt 1	HOCH	NIEDRIG	NIEDRIG	HOCH
Schritt 2	HOCH	HOCH	NIEDRIG	NIEDRIG
Schritt 3	NIEDRIG	HOCH	HOCH	NIEDRIG
Schritt 4	NIEDRIG	NIEDRIG	HOCH	HOCH
Schritt 5 (1)	HOCH	NIEDRIG	NIEDRIG	HOCH

Daher wird das potentielle Schreiben von MotorPin unter Verwendung einer zweischichtigen for-Schleife implementiert.

In Schritt 1 ist $j = 0$, $i = 0 \sim 4$.

motorPin [0] wird in der hohen Ebene geschrieben ($10011001 \& 00001000 = 1$)

motorPin [1] wird auf dem niedrigen Niveau geschrieben ($10011001 \& 00000100 = 0$)

motorPin [2] wird in der niedrigen Ebene geschrieben ($10011001 \& 00000010 = 0$)

motorPin [3] wird in der hohen Ebene geschrieben ($10011001 \& 00000001 = 1$)

In Schritt 2 ist $j = 1$, $i = 0 \sim 4$.

motorPin [0] wird in der hohen Ebene geschrieben ($01001100 \& 00001000 = 1$)

motorPin [1] wird auf dem niedrigen Niveau geschrieben ($01001100 \& 00000100 = 1$)

Um den Schrittmotor gegen den Uhrzeigersinn drehen zu lassen, wird der Füllstandsstatus von motorPin in der folgenden Tabelle angezeigt.

	MotorPin A.	MotorPin B.	MotorPin C.	MotorPin D.
Schritt 1	HOCH	NIEDRIG	NIEDRIG	HOCH
Schritt 2	NIEDRIG	NIEDRIG	HOCH	HOCH
Schritt 3	NIEDRIG	HOCH	HOCH	NIEDRIG
Schritt 4	HOCH	HOCH	NIEDRIG	NIEDRIG
Schritt 5 (1)	HOCH	NIEDRIG	NIEDRIG	HOCH

In Schritt 1 ist $j = 0$, $i = 0 \sim 4$.

`motorPin [0]` wird in der hohen Ebene geschrieben ($10011001 \& 10000000 = 1$)

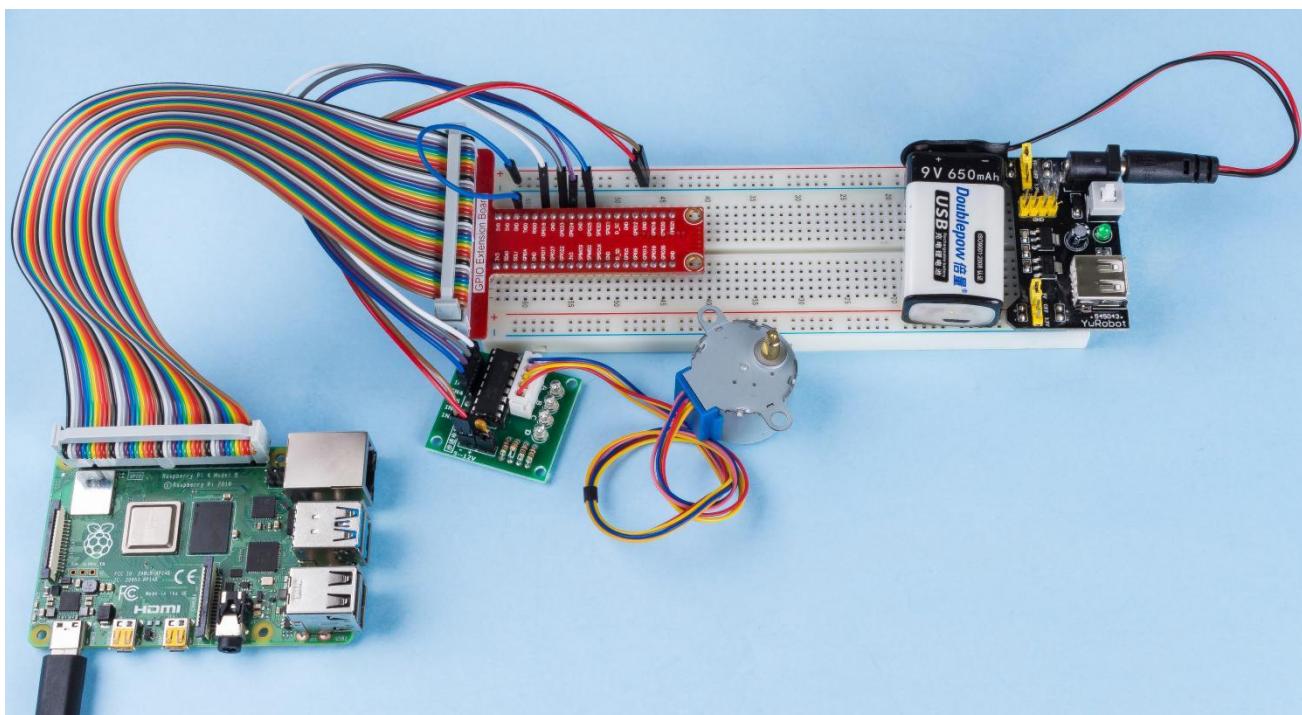
`motorPin [1]` wird auf dem niedrigen Niveau geschrieben ($10011001 \& 01000000 = 0$)

In Schritt 2 ist $j = 1$, $i = 0 \sim 4$.

`motorPin [0]` wird in der hohen Ebene geschrieben ($00110010 \& 10000000 = 0$)

`motorPin [1]` wird auf dem niedrigen Niveau geschrieben ($00110010 \& 01000000 = 0$)

Phänomen Bild

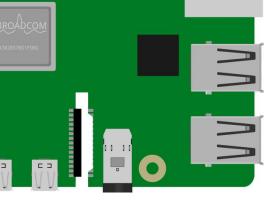
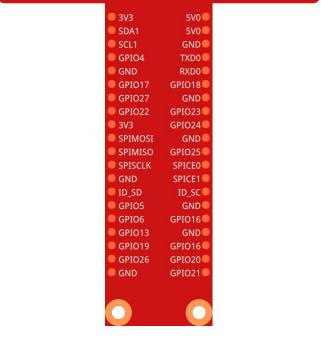
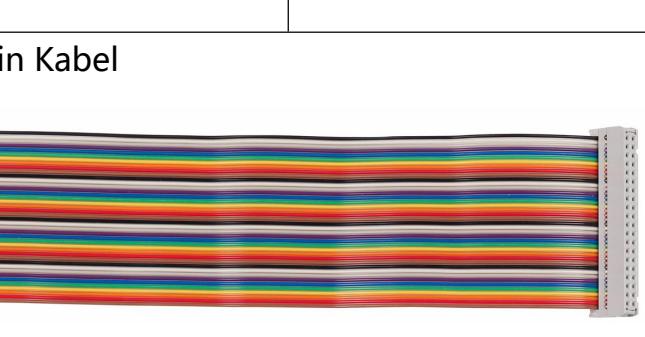
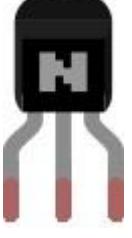
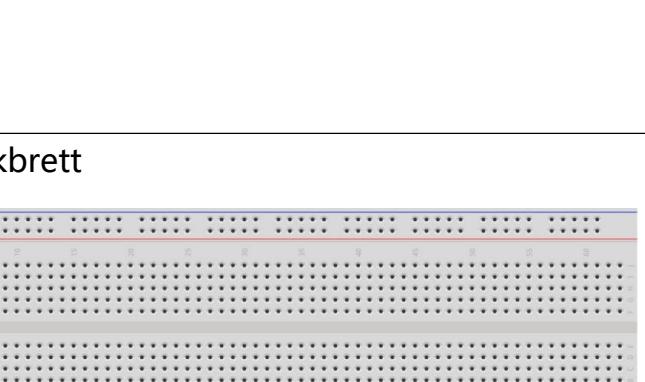
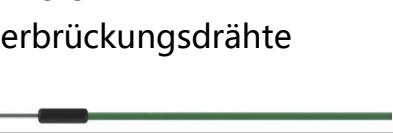
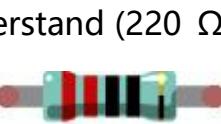


1.3.4 Relais

Einführung

In dieser Lektion lernen wir, ein Relais zu verwenden. Es ist eine der am häufigsten verwendeten Komponenten im automatischen Steuerungssystem. Wenn die Spannung, der Strom, die Temperatur, der Druck usw. den vorgegebenen Wert erreichen, überschreiten oder unterschreiten, wird das Relais den Stromkreis anschließen oder unterbrechen, um das Gerät zu steuern und zu schützen.

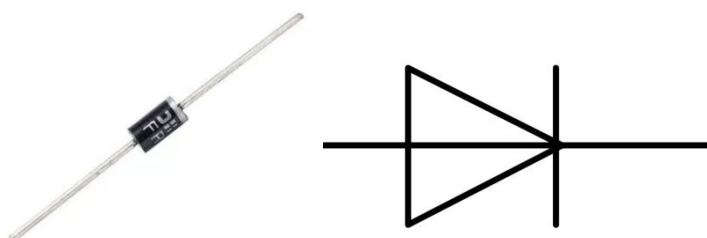
Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Relais																																								
	 GPIO Extension Board <table border="1"><tr><td>3V3</td><td>5V0</td></tr><tr><td>SDA1</td><td>5V0</td></tr><tr><td>SCL1</td><td>GND</td></tr><tr><td>GPIO4</td><td>TXD0</td></tr><tr><td>GND</td><td>RXD0</td></tr><tr><td>GPIO17</td><td>GPIO18</td></tr><tr><td>GPIO27</td><td>GND</td></tr><tr><td>GPIO22</td><td>GPIO23</td></tr><tr><td>3V3</td><td>GND</td></tr><tr><td>SPIMOSI</td><td>GND</td></tr><tr><td>SPIMISO</td><td>GPIO25</td></tr><tr><td>SPI5CLK</td><td>SPICE0</td></tr><tr><td>GND</td><td>SPICE1</td></tr><tr><td>ID_SD</td><td>ID_SC</td></tr><tr><td>GPIO5</td><td>GND</td></tr><tr><td>GPIO6</td><td>GPIO16</td></tr><tr><td>GPIO13</td><td>GND</td></tr><tr><td>GPIO19</td><td>GPIO16</td></tr><tr><td>GPIO26</td><td>GPIO20</td></tr><tr><td>GND</td><td>GPIO21</td></tr></table>	3V3	5V0	SDA1	5V0	SCL1	GND	GPIO4	TXD0	GND	RXD0	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GND	SPIMOSI	GND	SPIMISO	GPIO25	SPI5CLK	SPICE0	GND	SPICE1	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	 SONGLE 3A 250VAC 30VDC SRS-05VDC-SH
3V3	5V0																																									
SDA1	5V0																																									
SCL1	GND																																									
GPIO4	TXD0																																									
GND	RXD0																																									
GPIO17	GPIO18																																									
GPIO27	GND																																									
GPIO22	GPIO23																																									
3V3	GND																																									
SPIMOSI	GND																																									
SPIMISO	GPIO25																																									
SPI5CLK	SPICE0																																									
GND	SPICE1																																									
ID_SD	ID_SC																																									
GPIO5	GND																																									
GPIO6	GPIO16																																									
GPIO13	GND																																									
GPIO19	GPIO16																																									
GPIO26	GPIO20																																									
GND	GPIO21																																									
1 * 40-Pin Kabel	1 * 1N4007 Diode	1 * LED																																								
																																										
1 * Steckbrett	1 * S8050 NPN-Transistor																																									
	Mehrere Überbrückungsdrähte																																									
	1 * Widerstand (220 Ω)																																									
	1 * Widerstand 1KΩ																																									

Prinzip

Diode

Eine Diode ist eine zweipolare Komponente in der Elektronik mit einem unidirektionalen Stromfluss. Es bietet einen geringen Widerstand in Richtung des Stromflusses und einen hohen Widerstand in der entgegengesetzten Richtung. Dioden werden meistens verwendet, um Schäden an Bauteilen zu vermeiden, insbesondere aufgrund elektromotorischer Kraft in Schaltkreisen, die normalerweise polarisiert sind.



Die beiden Anschlüsse einer Diode sind polarisiert, wobei das positive Ende als Anode und das negative Ende als Kathode bezeichnet wird. Die Kathode besteht üblicherweise aus Silber oder hat ein Farbband. Die Steuerung der Stromflussrichtung ist eines der Hauptmerkmale von Dioden - der Strom in einer Diode fließt von Anode zu Kathode. Das Verhalten einer Diode ähnelt dem Verhalten eines Rückschlagventils. Eine der wichtigsten Eigenschaften einer Diode ist die nichtlineare Stromspannung. Wenn eine höhere Spannung an die Anode angeschlossen ist, fließt Strom von Anode zu Kathode, und der Prozess wird als Vorwärtsvorspannung bezeichnet. Wenn jedoch die höhere Spannung an die Kathode angeschlossen ist, leitet die Diode keine Elektrizität, und der Prozess wird als Sperrvorspannung bezeichnet.

Relais

Wie wir vielleicht wissen, ist Relais ein Gerät, das verwendet wird, um eine Verbindung zwischen zwei oder mehr Punkten oder Geräten als Reaktion auf das angelegte Eingangssignal herzustellen. Mit anderen Worten, Relais stellen eine Isolation zwischen der Steuerung und dem Gerät bereit, da Geräte sowohl mit Wechselstrom als auch mit Gleichstrom arbeiten können. Sie empfangen jedoch Signale von einem Mikrocontroller, der mit Gleichstrom arbeitet, weshalb ein Relais erforderlich ist, um die Lücke zu schließen. Das Relais ist äußerst nützlich, wenn Sie eine große Menge an Strom oder Spannung mit einem kleinen elektrischen Signal steuern müssen.

Jedes Relais besteht aus 5 Teilen:

1. Elektromagnet - Er besteht aus einem Eisenkern, der durch eine Drahtspule gewickelt ist. Wenn Elektrizität durchgelassen wird, wird sie magnetisch. Daher wird es Elektromagnet genannt.

2. Anker - Der bewegliche Magnetstreifen wird als Anker bezeichnet. Wenn Strom durch sie fließt, wird die Spule erregt, wodurch ein Magnetfeld erzeugt wird, das verwendet wird, um die normalerweise offenen (N/O) oder normalerweise geschlossenen (N/C) Punkte zu erzeugen oder zu brechen. Der Anker kann sowohl mit Gleichstrom (DC) als auch mit Wechselstrom (AC) bewegt werden.

3. Feder - Wenn am Elektromagneten keine Ströme durch die Spule fließen, zieht die Feder den Anker weg, sodass der Stromkreis nicht abgeschlossen werden kann.

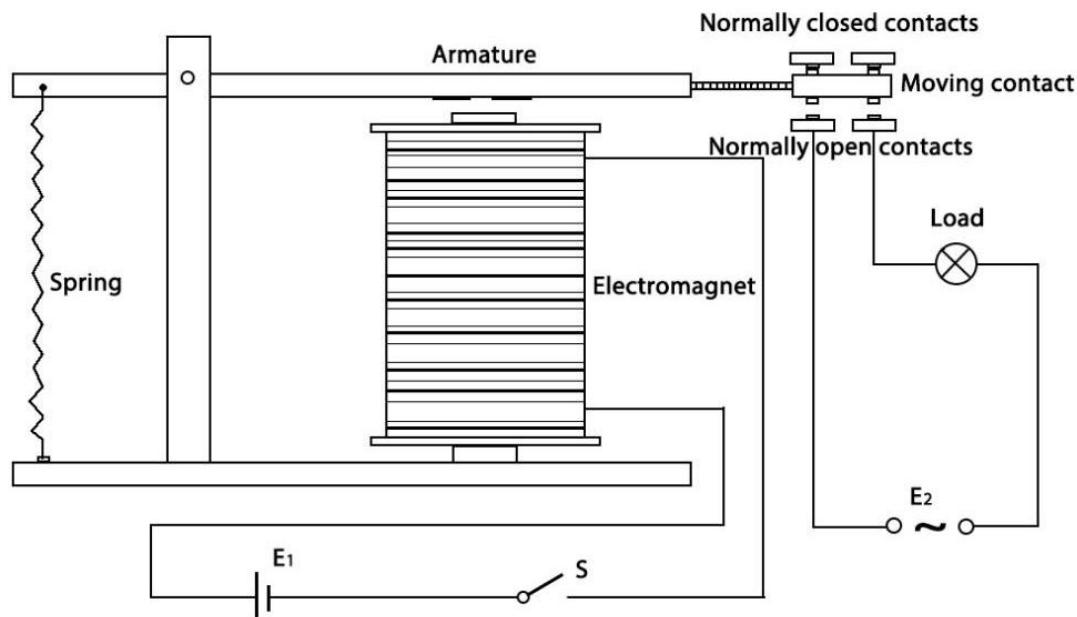
4. Satz elektrischer Kontakte - Es gibt zwei Kontaktpunkte:

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

5. Formrahmen - Die Relais sind zum Schutz mit Kunststoff bedeckt.

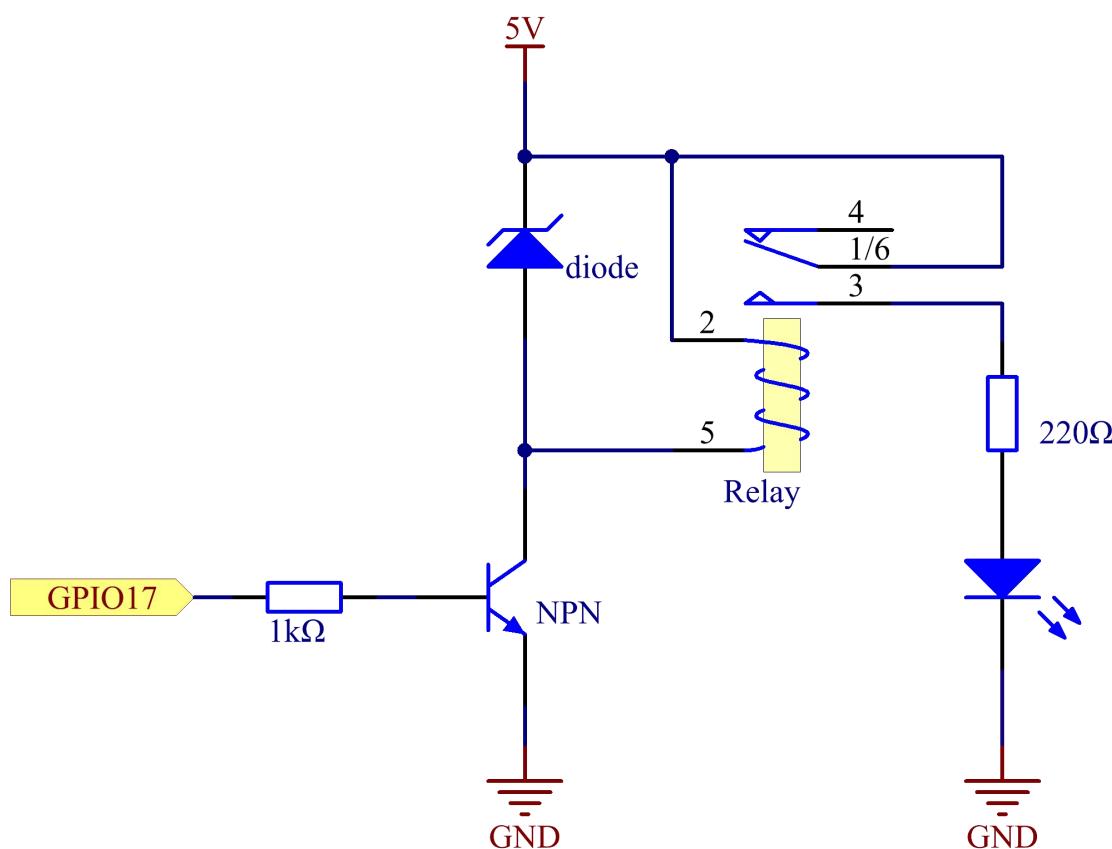
Funktionieren des Relais

Das Funktionsprinzip des Relais ist einfach. Wenn das Relais mit Strom versorgt wird, fließen Ströme durch die Steuerspule. Infolgedessen beginnt der Elektromagnet zu erregen. Dann wird der Anker von der Spule angezogen, bewegliche Kontakt zusammenziehen, wodurch eine Verbindung mit den normalerweise offenen Kontakten hergestellt wird. Der Stromkreis mit der Last wird also erregt. Ein Unterbrechen des Stromkreises wäre dann ein ähnlicher Fall, da der bewegliche Kontakt unter der Kraft der Feder zu den normalerweise geschlossenen Kontakten hochgezogen wird. Auf diese Weise kann das Ein- und Ausschalten des Relais den Zustand eines Lastkreises steuern.



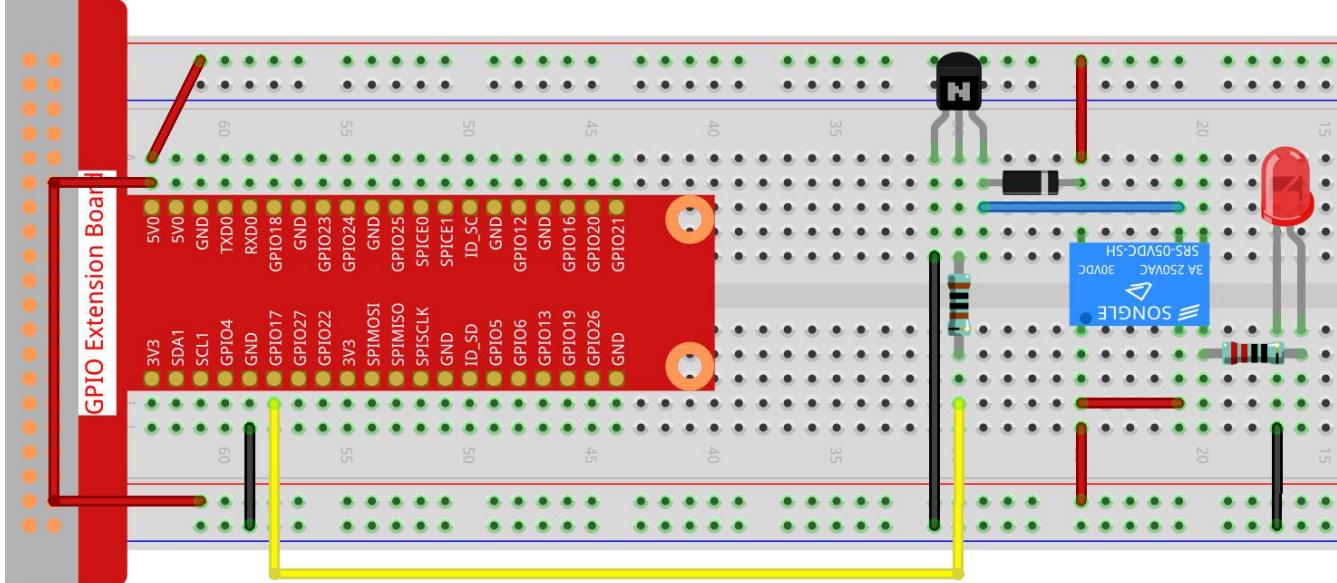
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Öffnen Sie die Kodedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/1.3.4
```

Schritt 3: Kompilieren Sie den Code.

```
gcc 1.3.4_Relay.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Nachdem die Kode ausgeführt wurde, leuchtet die LED auf. Außerdem können Sie ein Ticktock hören, das durch das Unterbrechen des normalerweise geschlossenen Kontakts und das Schließen des normalerweise offenen Kontakts verursacht wird.

Kode

```
#include <wiringPi.h>
#include <stdio.h>
#define RelayPin 0

int main(void){
    if(wiringPiSetup() == -1){ //when initialize wiring failed, print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    pinMode(RelayPin, OUTPUT); //set GPIO17(GPIO0) output
    while(1){
        // Tick
    }
}
```

```
printf("Relay Open.....\n");
digitalWrite(RelayPin, LOW);
delay(1000);
// Tock
printf(".....Relay Close\n");
digitalWrite(RelayPin, HIGH);
delay(1000);
}

return 0;
}
```

Kode Erklärung

```
digitalWrite(RelayPin, LOW);
```

Stellen Sie den I/O -Anschluss auf einen niedrigen Niveau (0V) ein, damit der Transistor nicht erregt und die Spule nicht mit Strom versorgt wird. Es gibt keine elektromagnetische Kraft, daher öffnet sich das Relais und die LED leuchtet nicht.

```
digitalWrite(RelayPin, HIGH);
```

Stellen Sie den I/O -Anschluss auf einen hohen Niveau (5V) ein, um den Transistor mit Strom zu versorgen. Die Spule des Relais wird mit Strom versorgt und erzeugt elektromagnetische Kraft. Das Relais schließt, die LED leuchtet auf.

➤ Für Python-Benutzer:

Schritt 2: Öffnen Sie die Codedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Schritt 3: Ausführen.

```
sudo python3 1.3.4_Relay.py
```

Während der Code läuft, leuchtet die LED. Außerdem können Sie ein Ticktock hören, das durch das Unterbrechen des normalerweise geschlossenen Kontakts und das Schließen des normalerweise offenen Kontakts verursacht wird.

Kode

```
#!/usr/bin/env python3
```

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

```
import time

# Set GPIO17 as control pin
relayPin = 17

# Define a setup function for some setup
def setup():
    # Set the GPIO modes to BCM Numbering
    GPIO.setmode(GPIO.BCM)
    # Set relayPin's mode to output,
    # and initial level to High(3.3v)
    GPIO.setup(relayPin, GPIO.OUT, initial=GPIO.HIGH)

# Define a main function for main process
def main():
    while True:
        print ('Relay open...')
        # Tick
        GPIO.output(relayPin, GPIO.LOW)
        time.sleep(1)
        print ('...Relay close')
        # Tock
        GPIO.output(relayPin, GPIO.HIGH)
        time.sleep(1)

# Define a destroy function for clean up everything after
# the script finished
def destroy():
    # Turn off LED
    GPIO.output(relayPin, GPIO.HIGH)
    # Release resource
    GPIO.cleanup()

# If run this script directly, do:
if __name__ == '__main__':
    setup()
    try:
        main()
    # When 'Ctrl+C' is pressed, the child program
    # destroy() will be executed.
```

```
except KeyboardInterrupt:  
    destroy()
```

Kode Erklärung

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

Stellen Sie die Pins des Transistors auf einen niedrigen Niveau ein, damit das Relais geöffnet wird. Die LED leuchtet nicht.

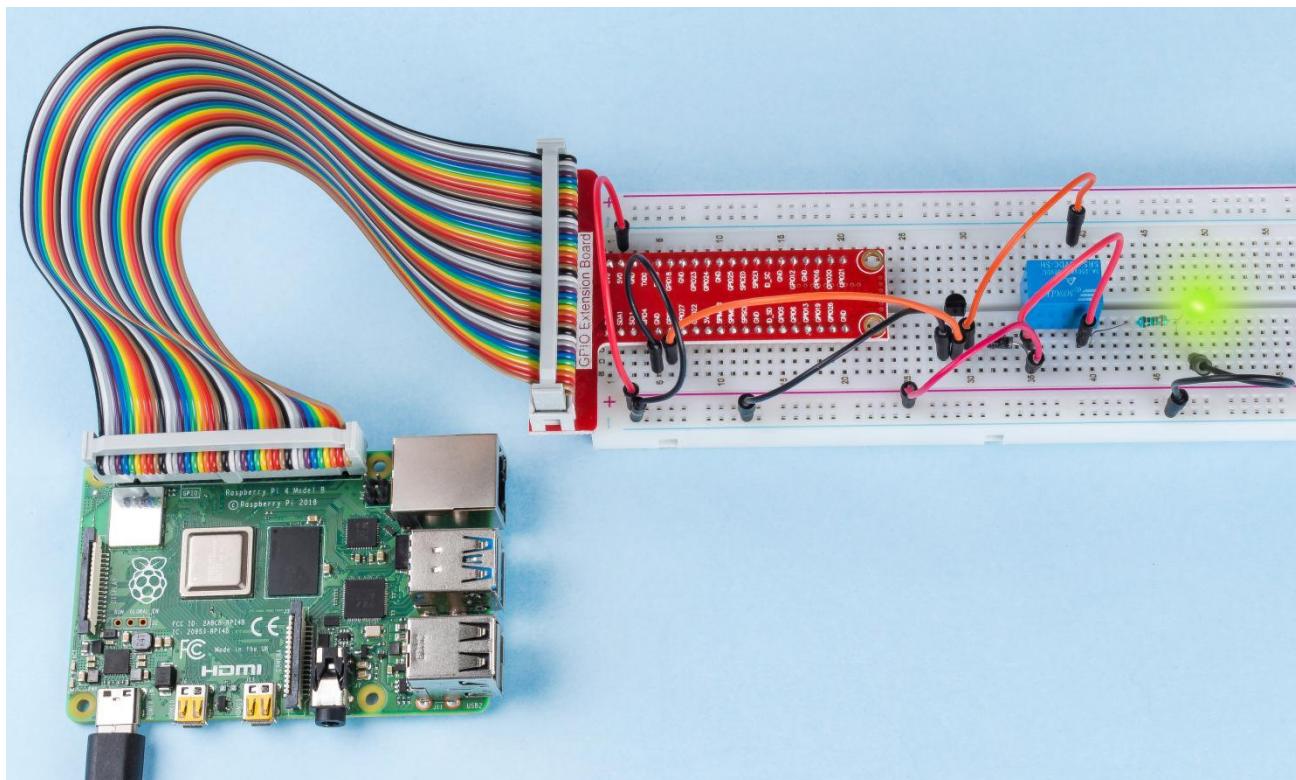
```
time.sleep(1)
```

1 Sekunde warten.

```
GPIO.output(relayPin, GPIO.HIGH)
```

Stellen Sie die Pins des Transistors auf einen niedrigen Niveau ein, um das Relais zu betätigen. Die LED leuchtet auf.

Phänomen Bild



2 Eingabe

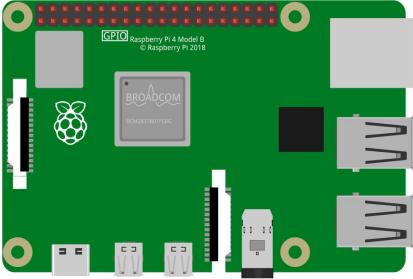
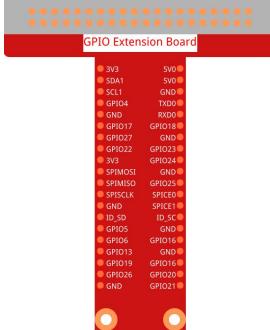
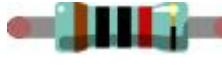
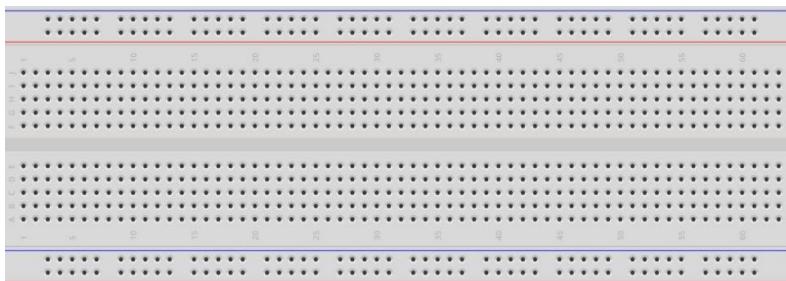
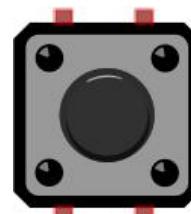
2.1 Steuerungen

2.1.1 Taste

Einführung

In dieser Lektion lernen wir, wie Sie die LED mithilfe einer Taste ein- oder ausschalten.

Komponenten

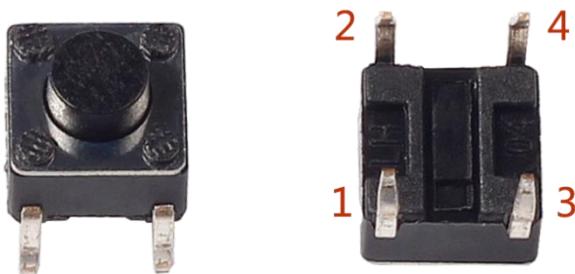
1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * LED																																																												
	 <table border="1"> <tr><th></th><th>3V3</th><th>5V0</th></tr> <tr><td>SDA1</td><td>GND</td><td></td></tr> <tr><td>SCL1</td><td>GND</td><td></td></tr> <tr><td>GPIO0</td><td>TX00</td><td></td></tr> <tr><td>GPIO1</td><td>TX00</td><td></td></tr> <tr><td>GPIO17</td><td>GPIO18</td><td></td></tr> <tr><td>GPIO27</td><td>GND</td><td></td></tr> <tr><td>GPIO22</td><td>GPIO23</td><td></td></tr> <tr><td>3V</td><td>GPIO24</td><td></td></tr> <tr><td>SPI_MOSI</td><td>GND</td><td></td></tr> <tr><td>SPI_MISO</td><td>GND</td><td></td></tr> <tr><td>SPI_SCLK</td><td>SPI_CE0</td><td></td></tr> <tr><td>GND</td><td>SPI_CE1</td><td></td></tr> <tr><td>ID_SD</td><td>ID_SC</td><td></td></tr> <tr><td>GPIO5</td><td>GND</td><td></td></tr> <tr><td>GPIO6</td><td>GND</td><td></td></tr> <tr><td>GPIO13</td><td>GND</td><td></td></tr> <tr><td>GPIO19</td><td>GPIO16</td><td></td></tr> <tr><td>GPIO26</td><td>GPIO20</td><td></td></tr> <tr><td>GND</td><td>GPIO21</td><td></td></tr> </table>		3V3	5V0	SDA1	GND		SCL1	GND		GPIO0	TX00		GPIO1	TX00		GPIO17	GPIO18		GPIO27	GND		GPIO22	GPIO23		3V	GPIO24		SPI_MOSI	GND		SPI_MISO	GND		SPI_SCLK	SPI_CE0		GND	SPI_CE1		ID_SD	ID_SC		GPIO5	GND		GPIO6	GND		GPIO13	GND		GPIO19	GPIO16		GPIO26	GPIO20		GND	GPIO21		
	3V3	5V0																																																												
SDA1	GND																																																													
SCL1	GND																																																													
GPIO0	TX00																																																													
GPIO1	TX00																																																													
GPIO17	GPIO18																																																													
GPIO27	GND																																																													
GPIO22	GPIO23																																																													
3V	GPIO24																																																													
SPI_MOSI	GND																																																													
SPI_MISO	GND																																																													
SPI_SCLK	SPI_CE0																																																													
GND	SPI_CE1																																																													
ID_SD	ID_SC																																																													
GPIO5	GND																																																													
GPIO6	GND																																																													
GPIO13	GND																																																													
GPIO19	GPIO16																																																													
GPIO26	GPIO20																																																													
GND	GPIO21																																																													
1 * 40-Pin Kabel		1 * Widerstand 10 kΩ																																																												
																																																														
1 * Steckbrett		1 * Widerstand (220 Ω)																																																												
																																																														
		Mehrere Überbrückungsdrähte																																																												
																																																														
		1 * Taste																																																												
																																																														

Prinzip

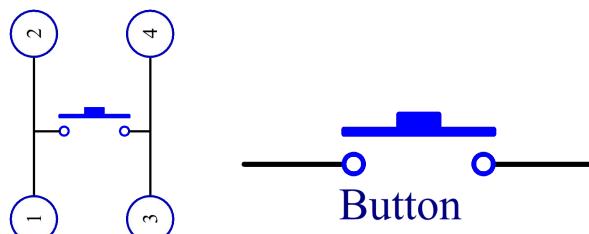
Taste

Die Taste ist eine übliche Komponente zur Steuerung elektronischer Geräte. Es wird normalerweise als Schalter zum Anschließen oder Unterbrechen von Stromkreisen verwendet. Obwohl die Tasten in verschiedenen Größen und Formen erhältlich sind, wird hier ein 6-mm-Miniknopf verwendet, wie in den folgenden Bildern gezeigt.

Zwei Pins auf der linken Seite sind verbunden, und der eine auf der rechten Seite ähnelt dem linken, der unten gezeigt wird:



Das unten gezeigte Symbol wird normalerweise verwendet, um eine Schaltfläche in Schaltkreisen darzustellen.



Wenn die Taste gedrückt wird, werden die 4 Pins verbunden, wodurch der Stromkreis geschlossen wird.

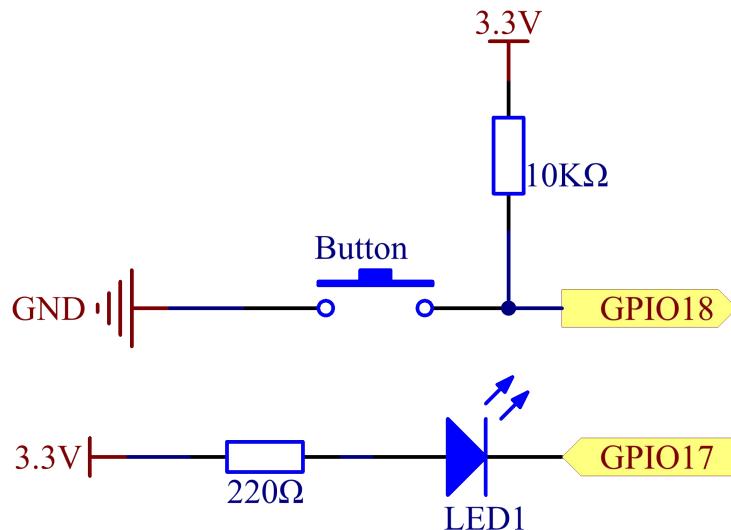
Schematische Darstellung

Verwenden Sie eine normalerweise geöffnete Taste als Eingang für Raspberry Pi. Die Verbindung ist in der folgenden schematischen Darstellung dargestellt. Wenn die Taste gedrückt wird, wird der GPIO18 auf einen niedrigen Niveau (0V) eingestellt. Wir können den Zustand des GPIO18 durch Programmierung erkennen. Das heißt, wenn der GPIO18 auf einen niedrigen Niveau wechselt, bedeutet dies, dass die Taste gedrückt wird. Sie können den entsprechenden Kode ausführen, wenn die Taste gedrückt wird, und dann leuchtet die LED auf.

Hinweis: Der längere Pin der LED ist die Anode und der kürzere ist die Kathode.

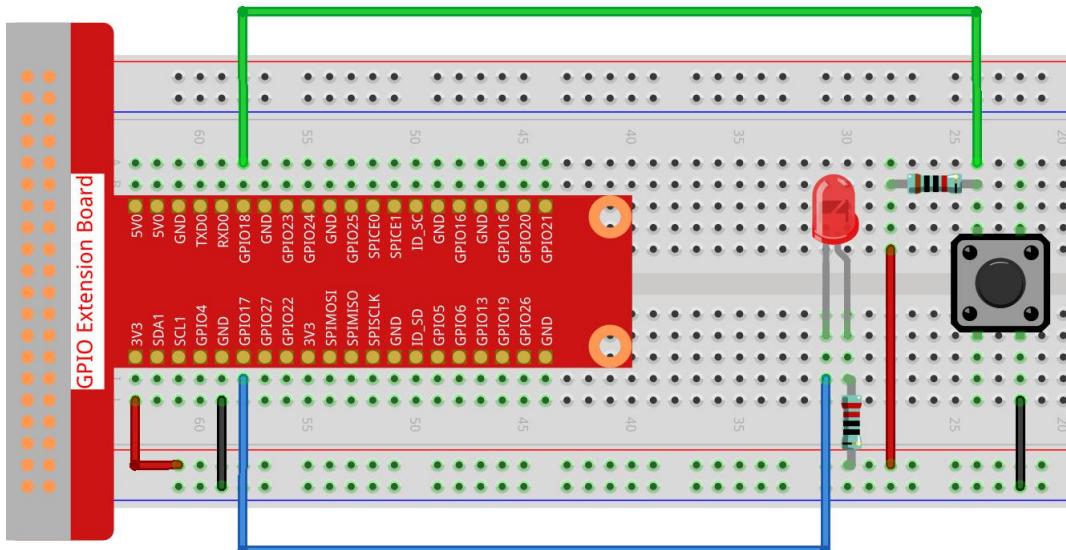
T-Karte Name	physisch	wiringPi	BCM
--------------	----------	----------	-----

GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Öffnen Sie die Kodedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.1.1/
```

Hinweis: Wechseln Sie in diesem Experiment in den Pfad der Kode mit **cd**.

Schritt 3: Kompilieren Sie den Code.

```
gcc 2.1.1_Button.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Nachdem die Kode ausgeführt wurde, drücken Sie die Taste. Die LED leuchtet auf. Andernfalls wird ausgeschaltet.

Kode

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

#define LedPin      0
#define ButtonPin   1

int main(void){
    // When initialize wiring failed, print message to screen
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }

    pinMode(LedPin, OUTPUT);
    pinMode(ButtonPin, INPUT);
    digitalWrite(LedPin, HIGH);

    while(1){
        // Indicate that button has pressed down
        if(digitalRead(ButtonPin) == 0){
            // Led on
            digitalWrite(LedPin, LOW);
            // printf("...LED on\n");
        }
        else{
            // Led off
            digitalWrite(LedPin, HIGH);
            // printf("LED off...\n");
        }
    }
    return 0;
}
```

Kode Erklärung

```
#define LedPin 0
```

Der Pin GPIO17 in der T_Extension-Karte entspricht dem GPIO0 im wiringPi.

```
#define ButtonPin 1
```

ButtonPin ist mit GPIO1 verbunden.

```
pinMode(LedPin, OUTPUT);
```

Stellen Sie LedPin als Ausgabe ein, um ihm einen Wert zuzuweisen.

```
pinMode(ButtonPin, INPUT);
```

Legen Sie ButtonPin als Eingabe fest, um den Wert von ButtonPin zu lesen.

```
while(1){  
    // Indicate that button has pressed down  
    if(digitalRead(ButtonPin) == 0){  
        // Led on  
        digitalWrite(LedPin, LOW);  
        // printf("...LED on\n");  
    }  
    else{  
        // Led off  
        digitalWrite(LedPin, HIGH);  
        // printf("LED off...\n");  
    }  
}
```

if (digitalRead (ButtonPin) == 0: Überprüfen Sie, ob die Taste gedrückt wurde. Führen Sie digitalWrite (LedPin, LOW) aus, wenn die Taste gedrückt wird, um die LED zu leuchten.

➤ Für Python-Sprachbenutzer

Schritt 2: Öffnen Sie die Kodedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Schritt 3: Führen Sie die Kode aus.

```
sudo python3 2.1.1_Button.py
```

Drücken Sie nun die Taste und die LED leuchtet auf. Drücken Sie die Taste erneut und die LED erlischt. Gleichzeitig wird der Status der LED auf dem Bildschirm gedruckt.

Kode

```
import RPi.GPIO as GPIO
import time

LedPin = 17 # Set GPIO17 as LED pin
BtnPin = 18 # Set GPIO18 as button pin

# Set Led status to True(OFF)
Led_status = True

# Define a setup function for some setup
def setup():
    # Set the GPIO modes to BCM Numbering
    GPIO.setmode(GPIO.BCM)
    # Set LedPin's mode to output,
    # and initial level to high (3.3v)
    GPIO.setup(LedPin, GPIO.OUT, initial=GPIO.HIGH)
    # Set BtnPin's mode to input,
    # and pull up to high (3.3V)
    GPIO.setup(BtnPin, GPIO.IN)

# Define a callback function for button callback
def swLed(ev=None):
    global Led_status
    # Switch led status(on-->off; off-->on)
    Led_status = not Led_status
    GPIO.output(LedPin, Led_status)
    if Led_status:
        print ('LED OFF...')
    else:
        print ('...LED ON')

# Define a main function for main process
def main():
    # Set up a falling detect on BtnPin,
    # and callback function to swLed
    GPIO.add_event_detect(BtnPin, GPIO.FALLING, callback=swLed)
```

```

while True:
    # Don't do anything.
    time.sleep(1)

# Define a destroy function for clean up everything after
# the script finished
def destroy():
    # Turn off LED
    GPIO.output(LedPin, GPIO.HIGH)
    # Release resource
    GPIO.cleanup()

# If run this script directly, do:
if __name__ == '__main__':
    setup()
    try:
        main()
    # When 'Ctrl+C' is pressed, the program
    # destroy() will be executed.
    except KeyboardInterrupt:
        destroy()

```

Kode Erklärung

LedPin = 17

Stellen Sie GPIO17 als LED-Pin ein

BtnPin = 18

Stellen Sie GPIO18 als Tasten Pin ein

GPIO.add_event_detect(BtnPin, GPIO.FALLING, callback=swLed)

Richten Sie eine Fallerkennung für BtnPin ein. Wenn sich der Wert von BtnPin von einem hohen auf einen niedrigen Wert ändert, bedeutet dies, dass die Taste gedrückt wird. Der nächste Schritt ist das Aufrufen der Funktion swled.

```

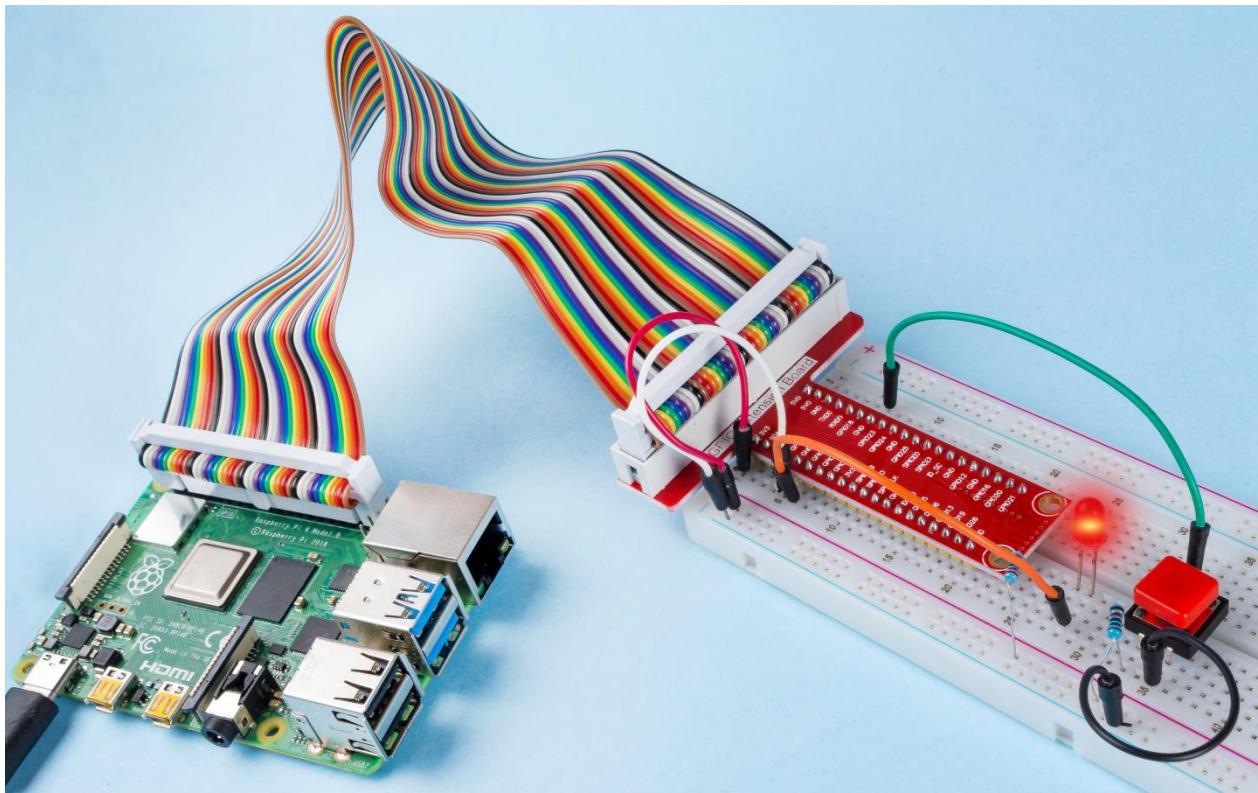
def swLed(ev=None):
    global Led_status
    # Switch led status(on-->off; off-->on)
    Led_status = not Led_status

```

```
GPIO.output(LedPin, Led_status)
```

Definieren Sie eine Rückruffunktion als Tastenrückruf. Wenn die Taste beim ersten Mal gedrückt wird und die Bedingung, nicht `Led_status`, falsch ist, wird die Funktion `GPIO.output()` aufgerufen, um die LED zu beleuchten. Wenn die Taste erneut gedrückt wird, wird der Status der LED von falsch in wahr umgewandelt, sodass die LED erlischt.

Phänomen Bild

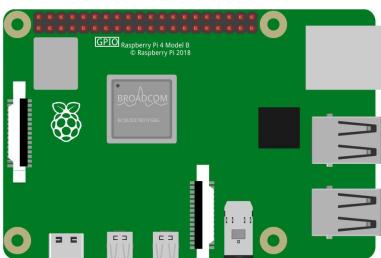
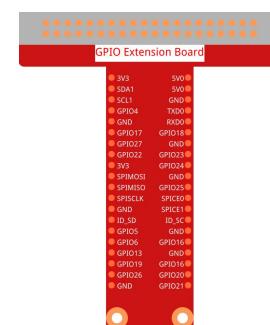
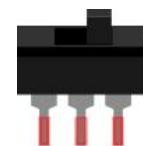
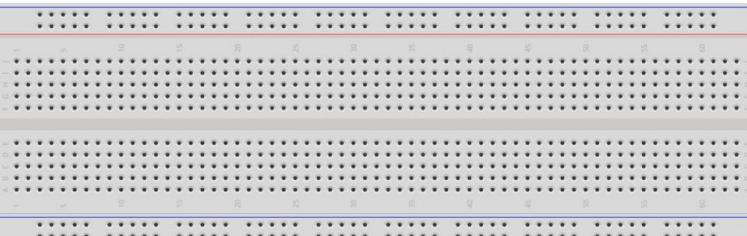
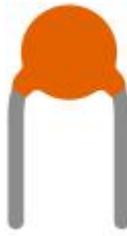


2.1.2 Schiebeschalter

Einführung

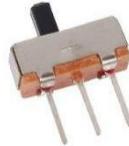
In dieser Lektion lernen wir, wie man einen Schiebeschalter benutzt. Normalerweise wird der Schiebeschalter als Netzschalter auf die Leiterplatte gelötet, aber hier müssen wir ihn in das Steckbrett einsetzen, damit er möglicherweise nicht festgezogen wird. Und wir verwenden es auf dem Steckbrett, um seine Funktion zu zeigen.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Schiebeschalter
	 GPIO Extension Board Pinout: ● 3V3 SVO(B) ● SDI1 SVO(B) ● SCL1 GND(B) ● SDO4 TXD0(B) ● GND RVDD(B) ● GPIO17 GPIO18(B) ● GPIO27 GND(B) ● GPIO22 GPIO23(B) ● GND GND(B) ● SPIMOSI GND(B) ● SPIMISO GPIO25(B) ● SPISCLK SPICE0(B) ● GND SVO(B) ● ID_SD ID_SD(B) ● GPIO5 GND(B) ● GPIO13 GND(B) ● GPIO19 GND(B) ● GPIO26 GPIO20(B) ● GND GPIO21(B)	
1 * 40-Pin Kabel	2 * LED	
		
1 * Steckbrett	1 * 104 Kondensator	
		
Mehrere Überbrückungsdrähte		
2 * Widerstand (220 Ω)		
1 * Widerstand 10 kΩ		

Prinzip

Schiebeschalter

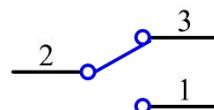


Ein Schiebeschalter dient, wie der Name schon sagt, dazu, die Schaltleiste zu schieben, um den Stromkreis anzuschließen oder zu unterbrechen, und weitere Schaltkreise. Die am häufigsten verwendeten Typen sind SPDT, SPTT, DPDT, DPTT usw. Der Schiebeschalter wird üblicherweise in Niederspannungsschaltungen verwendet. Es hat die Merkmale Flexibilität und Stabilität und ist in elektrischen Instrumenten und elektrischem Spielzeug weit verbreitet.

So funktioniert es: Stellen Sie den mittleren Pin als festen Pin ein. Wenn Sie den Schieber nach links ziehen, sind die beiden Pins links verbunden. Wenn Sie es nach rechts ziehen, sind die beiden Pins rechts verbunden. Somit funktioniert es als Schalter, der Schaltkreise verbindet oder trennt. Siehe die folgende Abbildung:



Das Schaltungssymbol des Schiebeschalters ist wie folgt dargestellt. Der Pin2 in der Abbildung bezieht sich auf den mittleren Pin.



Kondensator

Der Kondensator ist eine Komponente, die Energie in Form von elektrischer Ladung speichern oder eine Potentialdifferenz (statische Spannung) zwischen ihren Platten erzeugen kann, ähnlich wie eine kleine wiederaufladbare Batterie.

Standardkapazitätseinheiten

Microfarad (μF) $1\mu\text{F} = 1/1,000,000 = 0.000001 = 10^{-6} \text{ F}$

Nanofarad (nF) $1\text{nF} = 1/1,000,000,000 = 0.000000001 = 10^{-9}\text{F}$

Picofarad (pF) $1\text{pF} = 1/1,000,000,000,000 = 0.000000000001 = 10^{-12}\text{F}$

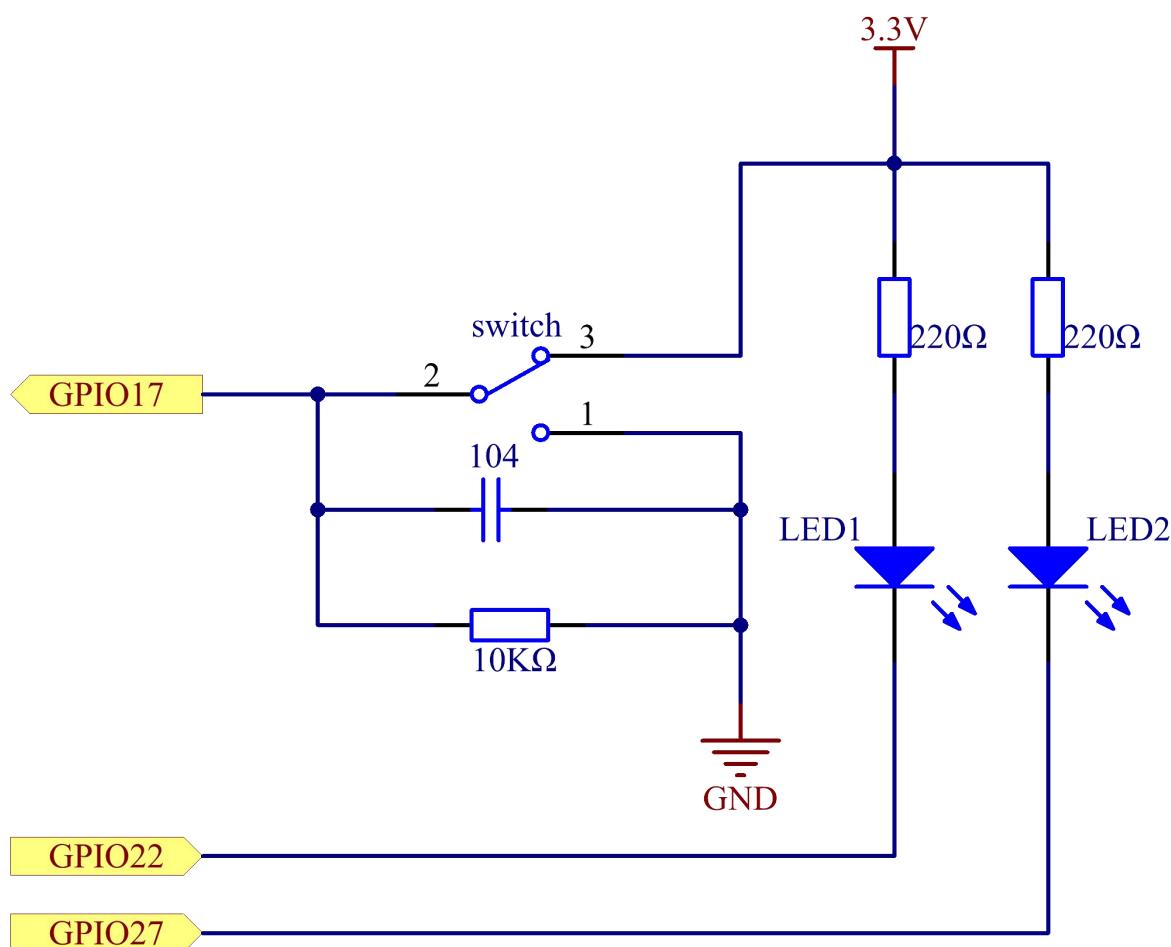
Hinweis: Hier verwenden wir **104 Kondensatoren (10 x 10⁴PF)**. Genau wie beim Widerstandsring helfen die Nummer auf den Kondensatoren beim Ablesen der Werte, die auf der Platine montiert wurden. Die ersten beiden Ziffern stellen den

Wert dar und die letzte Ziffer der Nummer bedeutet den Multiplikator. Somit repräsentiert 104 eine Potenz von 10×10 zu 4 (in pF) gleich wie 100 nF.

Schematische Darstellung

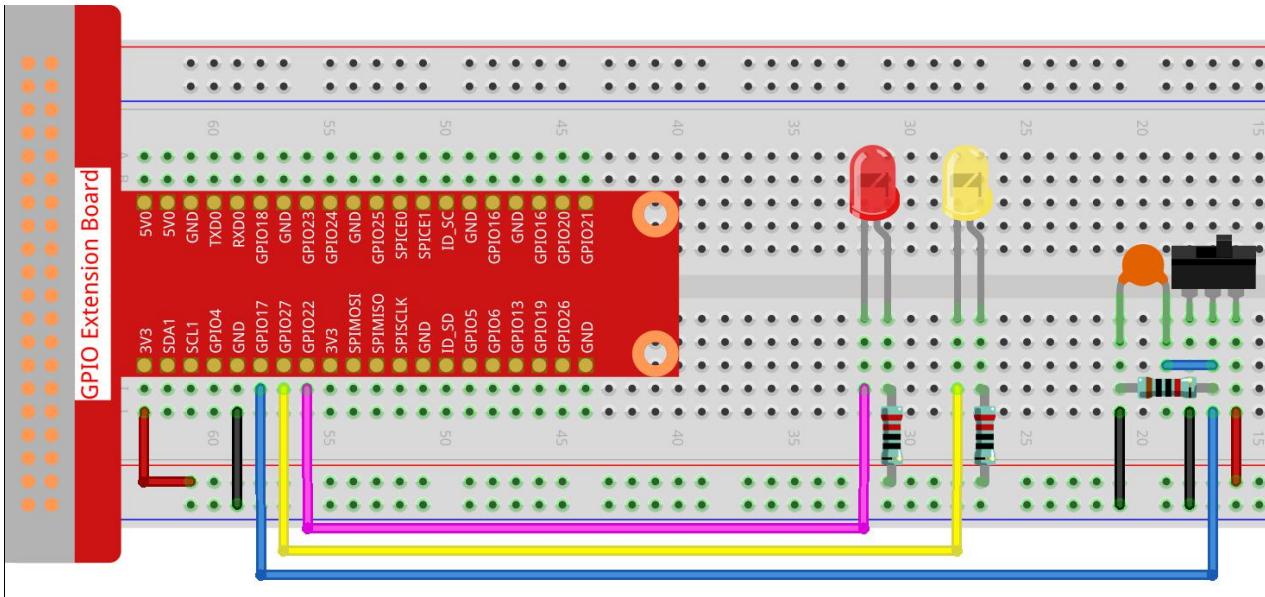
Verbinden Sie den mittleren Pin des Schiebeschalters mit GPIO17 und zwei LEDs mit Pin GPIO22 bzw. GPIO27. Wenn Sie dann an der Folie ziehen, leuchten die beiden LEDs abwechselnd auf.

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.1.2
```

Schritt 3: Kompilieren.

```
gcc 2.1.2_Slider.c -lwiringPi
```

Schritt 4: Führen Sie die obige ausführbare Datei aus.

```
sudo ./a.out
```

Während der Code ausgeführt wird, schalten Sie den Schalter links ein, und die gelbe LED leuchtet auf. rechts leuchtet das rote Licht auf.

Kode

```
#include <wiringPi.h>
#include <stdio.h>
#define slidePin      0
#define led1          3
#define led2          2

int main(void)
{
    // When initialize wiring failed, print message to screen
    if(wiringPiSetup() == -1){

```

```

printf("setup wiringPi failed !");
return 1;
}
pinMode(slidePin, INPUT);
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
while(1){
    // slide switch high, led1 on
    if(digitalRead(slidePin) == 1){
        digitalWrite(led1, LOW);
        digitalWrite(led2, HIGH);
        printf("LED1 on\n");
    }
    // slide switch low, led2 on
    if(digitalRead(slidePin) == 0){
        digitalWrite(led2, LOW);
        digitalWrite(led1, HIGH);
        printf(".....LED2 on\n");
    }
}
return 0;
}

```

Kode Erklärung

```

if(digitalRead(slidePin) == 1){
    digitalWrite(led1, LOW);
    digitalWrite(led2, HIGH);
    printf("LED1 on\n");
}

```

Wenn der Schieber nach rechts gezogen wird, sind der mittlere und der rechte Stift verbunden. Der Raspberry Pi liest einen hohen Niveau am mittleren Pin, sodass die LED1 an und die LED2 aus ist

```

if(digitalRead(slidePin) == 0){
    digitalWrite(led2, LOW);
    digitalWrite(led1, HIGH);
    printf(".....LED2 on\n");
}

```

Wenn der Schieber nach links gezogen wird, sind der mittlere und der linke Pins verbunden. Der Raspberry Pi zeigt einen niedrigen Wert an, sodass die LED2 leuchtet und die LED1 aus ist

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie in den Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Schritt 3: Ausführen.

```
sudo python3 2.1.2_Slider.py
```

Während die Kode ausgeführt wird, schalten Sie den Schalter links ein, und die gelbe LED leuchtet auf. rechts leuchtet das rote Licht auf.

Kode

```
import RPi.GPIO as GPIO
import time

# Set GPIO17 as slide switch pin, GPIO22 as led1 pin, GPIO27 as led2 pin
slidePin = 17
led1Pin = 22
led2Pin = 27

# Define a setup function for some setup
def setup():
    # Set the GPIO modes to BCM Numbering
    GPIO.setmode(GPIO.BCM)
    # Set slidePin input
    # Set ledPin output,
    # and initial level to High(3.3v)
    GPIO.setup(slidePin, GPIO.IN)
    GPIO.setup(led1Pin, GPIO.OUT, initial=GPIO.HIGH)
    GPIO.setup(led2Pin, GPIO.OUT, initial=GPIO.HIGH)

# Define a main function for main process
def main():
    while True:
        # slide switch high, led1 on
        if GPIO.input(slidePin) == 1:
            print('    LED1 ON    ')
```

```

GPIO.output(led1Pin, GPIO.LOW)
GPIO.output(led2Pin, GPIO.HIGH)

# slide switch low, led2 on
if GPIO.input(slidePin) == 0:
    print('    LED2 ON    ')
    GPIO.output(led2Pin, GPIO.LOW)
    GPIO.output(led1Pin, 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(led1Pin, GPIO.HIGH)
    GPIO.output(led2Pin, GPIO.HIGH)
    # Release resource
    GPIO.cleanup()

# If run this script directly, do:
if __name__ == '__main__':
    setup()
    try:
        main()
    # When 'Ctrl+C' is pressed, the program
    # destroy() will be executed.
    except KeyboardInterrupt:
        destroy()

```

Kode Erklärung

```

if GPIO.input(slidePin) == 1:
    GPIO.output(led1Pin, GPIO.LOW)
    GPIO.output(led2Pin, GPIO.HIGH)

```

Wenn der Schieber nach rechts gezogen wird, sind der mittlere und der rechte Pins verbunden. Der Raspberry Pi liest einen hohen Niveau am mittleren Pin, sodass die LED1 an und die LED2 aus ist.

```

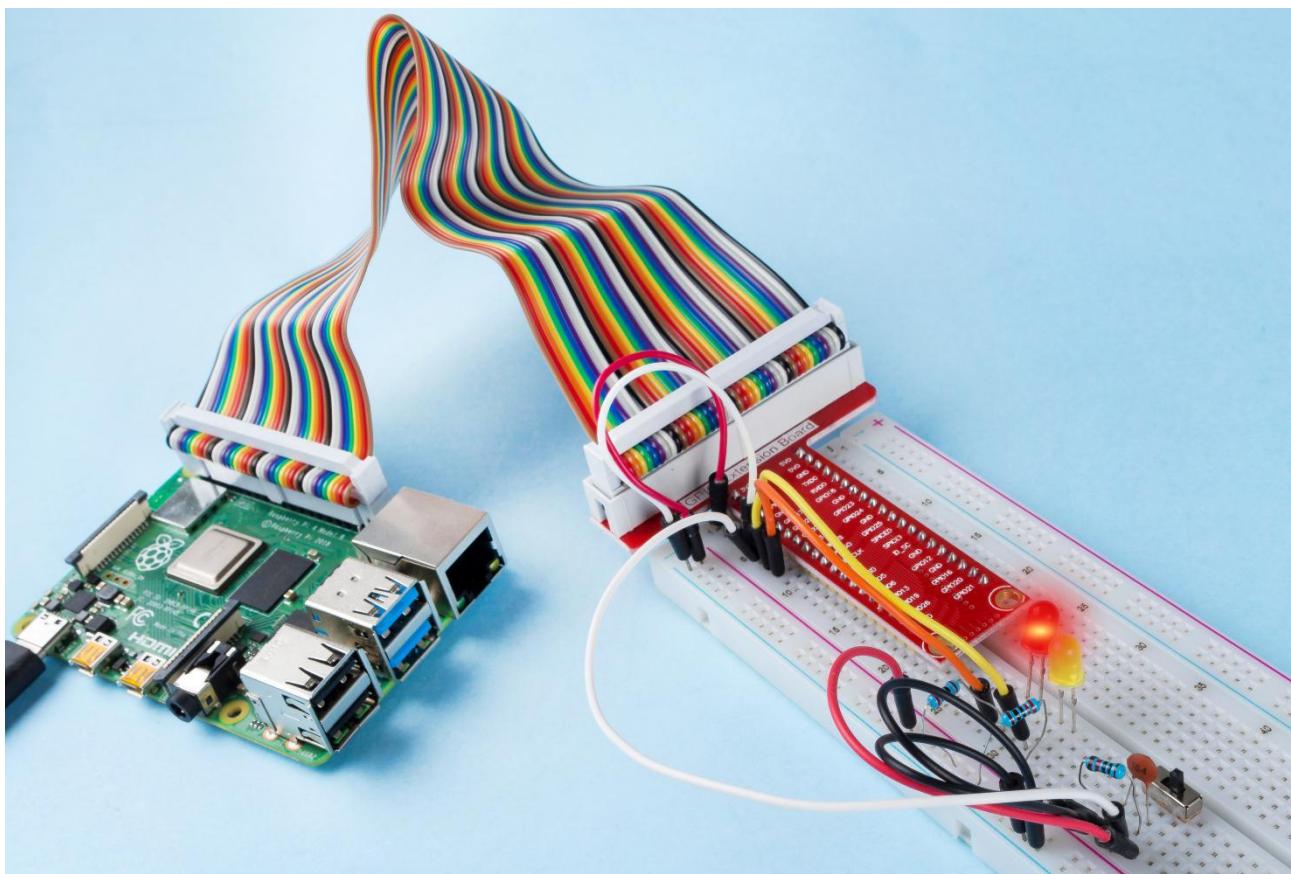
if GPIO.input(slidePin) == 0:

```

```
GPIO.output(led2Pin, GPIO.LOW)  
GPIO.output(led1Pin, GPIO.HIGH)
```

Wenn der Schieber nach links gezogen wird, sind der mittlere und der linke Pins verbunden. Der Raspberry Pi zeigt einen niedrigen Wert an, sodass die LED2 leuchtet und die LED1 aus ist.

Phänomen Bild

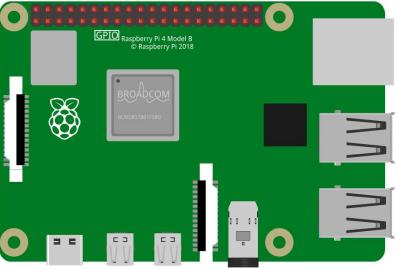
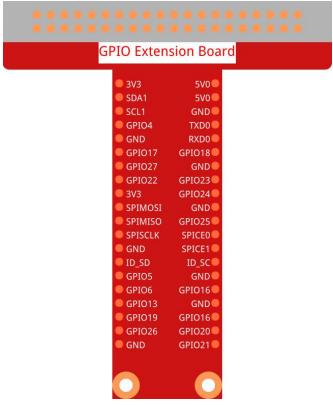
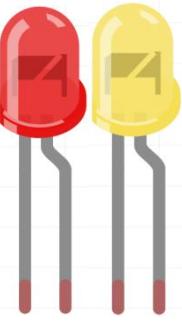


2.1.3 Neigungsschalter

Einführung

Dies ist ein Kugelkippschalter mit einer Metallkugel im Inneren. Es wird verwendet, um Neigungen eines kleinen Winkels zu erfassen.

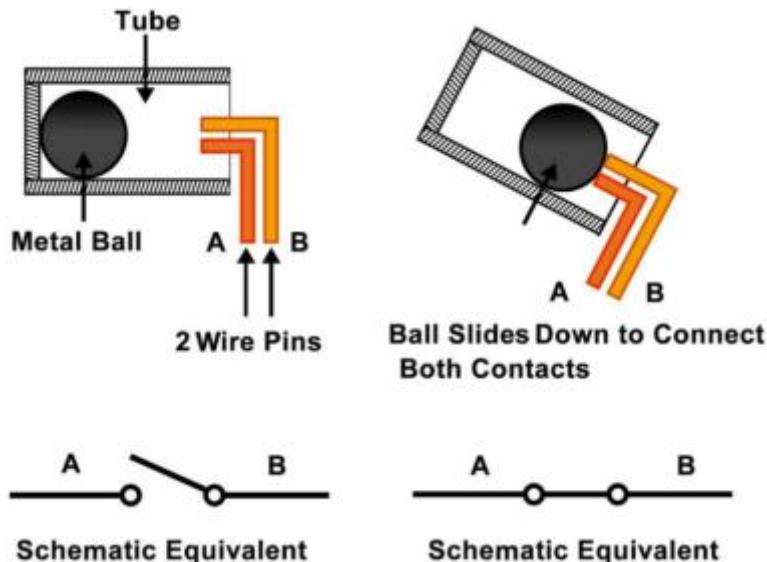
Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Neigungsschalter	2 * LED																																								
	 GPIO Extension Board <table border="1"><tr><td>3V3</td><td>5V0</td></tr><tr><td>SDA1</td><td>5V0</td></tr><tr><td>SCL1</td><td>GND</td></tr><tr><td>GPIO4</td><td>TxD0</td></tr><tr><td>GND</td><td>RxD0</td></tr><tr><td>GPIO17</td><td>GPIO18</td></tr><tr><td>GPIO27</td><td>GND</td></tr><tr><td>GPIO22</td><td>GPIO23</td></tr><tr><td>3V3</td><td>GPIO24</td></tr><tr><td>SPIMOSI</td><td>GND</td></tr><tr><td>SPIMISO</td><td>GPIO25</td></tr><tr><td>SPISCLK</td><td>SPICE0</td></tr><tr><td>GND</td><td>SPICE1</td></tr><tr><td>ID_SD</td><td>ID_SC</td></tr><tr><td>GPIO5</td><td>GND</td></tr><tr><td>GPIO6</td><td>GPIO16</td></tr><tr><td>GPIO13</td><td>GND</td></tr><tr><td>GPIO19</td><td>GPIO16</td></tr><tr><td>GPIO26</td><td>GPIO20</td></tr><tr><td>GND</td><td>GPIO21</td></tr></table>	3V3	5V0	SDA1	5V0	SCL1	GND	GPIO4	TxD0	GND	RxD0	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GPIO24	SPIMOSI	GND	SPIMISO	GPIO25	SPISCLK	SPICE0	GND	SPICE1	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21		
3V3	5V0																																										
SDA1	5V0																																										
SCL1	GND																																										
GPIO4	TxD0																																										
GND	RxD0																																										
GPIO17	GPIO18																																										
GPIO27	GND																																										
GPIO22	GPIO23																																										
3V3	GPIO24																																										
SPIMOSI	GND																																										
SPIMISO	GPIO25																																										
SPISCLK	SPICE0																																										
GND	SPICE1																																										
ID_SD	ID_SC																																										
GPIO5	GND																																										
GPIO6	GPIO16																																										
GPIO13	GND																																										
GPIO19	GPIO16																																										
GPIO26	GPIO20																																										
GND	GPIO21																																										
1 * 40-Pin Kabel	Mehrere Überbrückungsdrähte																																										
1 * Steckbrett	2 * Widerstand (220 Ω)																																										
	1 * Widerstand 10 k Ω																																										

Prinzip

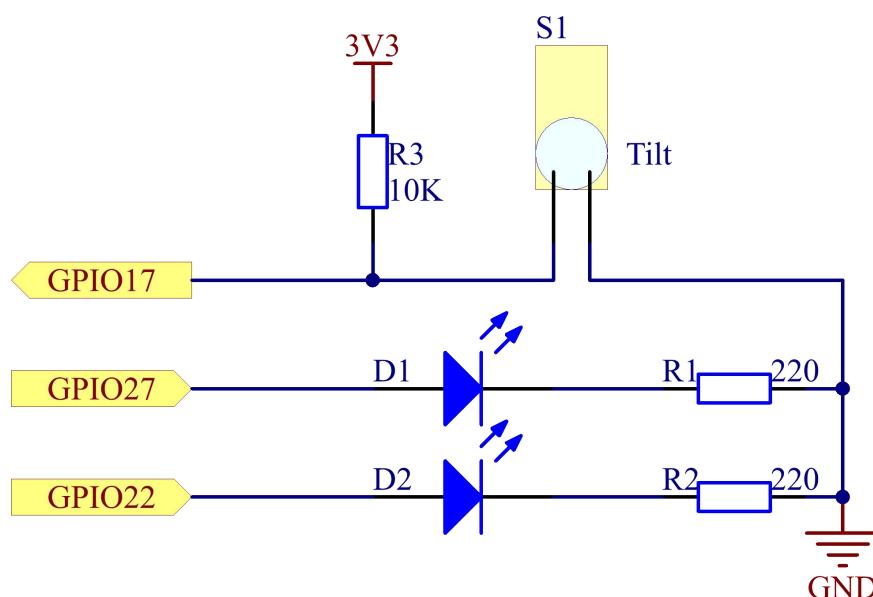
Neigung

Das Prinzip ist sehr einfach. Wenn der Schalter in einem bestimmten Winkel gekippt wird, rollt die Kugel im Inneren nach unten und berührt die beiden Kontakte, die mit den Pins außen verbunden sind, wodurch Schaltkreise ausgelöst werden. Andernfalls bleibt der Ball von den Kontakten fern und unterbricht so die Stromkreise.



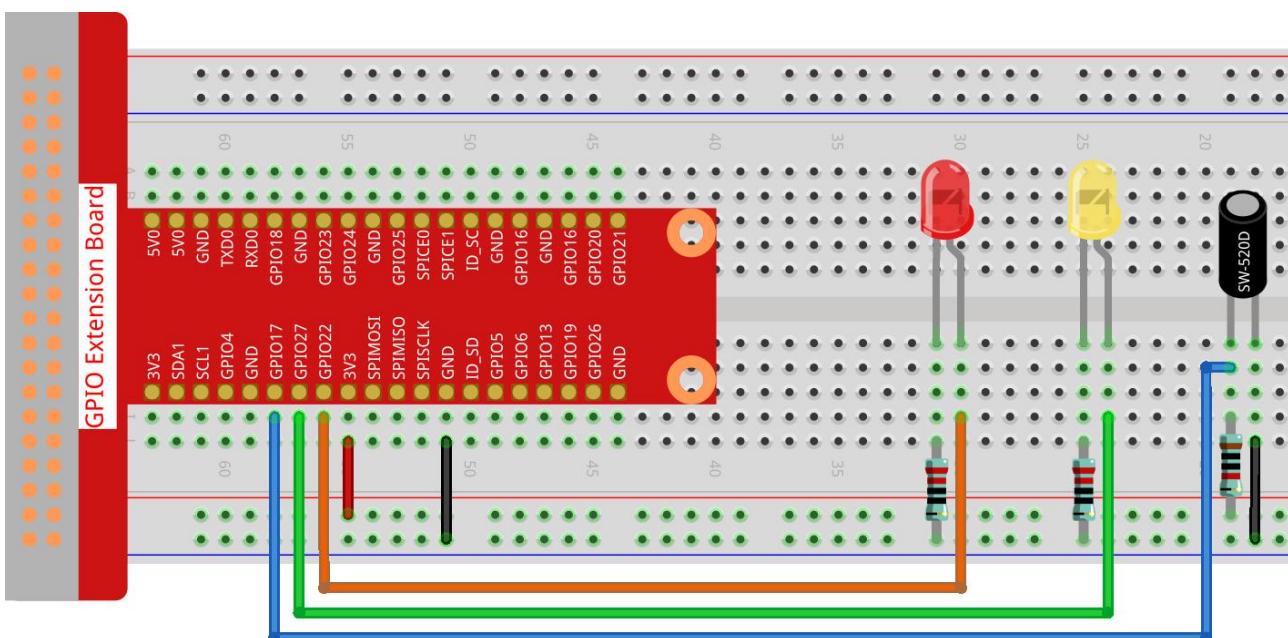
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.1.3/
```

Schritt 3: Kompilieren.

gcc 2.1.3_Tilt.c -lwiringPi

Schritt 4: Ausführen.

sudo ./a.out

Wenn Sie die Neigung horizontal platzieren, leuchtet die grüne LED auf. Wenn Sie es kippen, "Neigung!" wird auf dem Bildschirm gedruckt und die rote LED leuchtet auf. Stellen Sie es wieder horizontal auf und die grüne LED leuchtet wieder auf.

Kode

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

#define TiltPin      0
#define Gpin         2
#define Rpin         3

void LED(char* color)
```

```
{
    pinMode(Gpin, OUTPUT);
    pinMode(Rpin, OUTPUT);
    if (color == "RED")
    {
        digitalWrite(Rpin, HIGH);
        digitalWrite(Gpin, LOW);
    }
    else if (color == "GREEN")
    {
        digitalWrite(Rpin, LOW);
        digitalWrite(Gpin, HIGH);
    }
    else
        printf("LED Error");
}

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

    pinMode(TiltPin, INPUT);
    LED("GREEN");

    while(1){
        if(0 == digitalRead(TiltPin)){
            delay(10);
            if(0 == digitalRead(TiltPin)){
                LED("RED");
                printf("Tilt!\n");
            }
        }
        else if(1 == digitalRead(TiltPin)){
            delay(10);
            if(1 == digitalRead(TiltPin)){
                LED("GREEN");
            }
        }
    }
}
```

```

    }
}

return 0;
}
}
```

Kode Erklärung

```

void LED(char* color)
{
    pinMode(Gpin, OUTPUT);
    pinMode(Rpin, OUTPUT);
    if (color == "RED")
    {
        digitalWrite(Rpin, HIGH);
        digitalWrite(Gpin, LOW);
    }
    else if (color == "GREEN")
    {
        digitalWrite(Rpin, LOW);
        digitalWrite(Gpin, HIGH);
    }
    else
        printf("LED Error");
}
```

Definieren Sie eine Funktions-LED (), um die beiden LEDs ein- oder auszuschalten. Wenn die Parameterfarbe ROT ist, leuchtet die rote LED auf. Wenn die Parameterfarbe GRÜN ist, leuchtet die grüne LED ebenfalls auf.

```

while(1){
    if(0 == digitalRead(TiltPin)){
        delay(10);
        if(0 == digitalRead(TiltPin)){
            LED("RED");
            printf("Tilt!\n");
        }
    }
    else if(1 == digitalRead(TiltPin)){
        delay(10);
        if(1 == digitalRead(TiltPin)){
```

```

        LED("GREEN");
    }
}
}
```

Wenn der Lesewert des Neigungsschalters 0 ist, bedeutet dies, dass der Neigungsschalter gekippt ist. Dann schreiben Sie den Parameter "ROT" in die Funktions-LED, damit die rote LED aufleuchtet. Andernfalls leuchtet die grüne LED.

➤ Für Python-Sprachbenutzer

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Ausführen.

```
sudo python3 2.1.3_Tilt.py
```

Wenn Sie die Neigung horizontal platzieren, leuchtet die grüne LED auf. Wenn Sie es kippen, "Neigung!" wird auf dem Bildschirm gedruckt und die rote LED leuchtet auf. Stellen Sie es wieder horizontal auf und die grüne LED leuchtet auf.

Kode

```

import RPi.GPIO as GPIO

TiltPin = 11
Gpin   = 13
Rpin   = 15

def setup():
    GPIO.setmode(GPIO.BOARD)      # Numbers GPIOs by physical location
    GPIO.setup(Gpin, GPIO.OUT)     # Set Green Led Pin mode to output
    GPIO.setup(Rpin, GPIO.OUT)     # Set Red Led Pin mode to output
    GPIO.setup(TiltPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)  # Set BtnPin's mode is
input, and pull up to high level(3.3V)
    GPIO.add_event_detect(TiltPin, GPIO.BOTH, callback=detect, bouncetime=200)

def Led(x):
    if x == 0:
        GPIO.output(Rpin, 1)
        GPIO.output(Gpin, 0)
    if x == 1:
        GPIO.output(Rpin, 0)
```

```

GPIO.output(Gpin, 1)

def Print(x):
    if x == 0:
        print ('*****')
        print (' * Tilt! *')
        print ('*****')

def detect(chn):
    Led(GPIO.input(TiltPin))
    Print(GPIO.input(TiltPin))

def loop():
    while True:
        pass

def destroy():
    GPIO.output(Gpin, GPIO.HIGH)      # Green led off
    GPIO.output(Rpin, GPIO.HIGH)      # Red led off
    GPIO.cleanup()                   # Release resource

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

```

Kode Erklärung

```
GPIO.add_event_detect (TiltPin, GPIO.BOTH, callback=detect, bouncetime=200)
```

Richten Sie eine Erkennung auf TiltPin und eine Rückruffunktion zur Erkennung ein.

```

def Led(x):
    if x == 0:
        GPIO.output(Rpin, 1)
        GPIO.output(Gpin, 0)
    if x == 1:

```

```
GPIO.output(Rpin, 0)
GPIO.output(Gpin, 1)
```

Definieren Sie eine Funktion Led (), um die beiden LEDs ein- oder auszuschalten. Wenn x = 0 ist, leuchtet die rote LED auf. Andernfalls leuchtet die grüne LED.

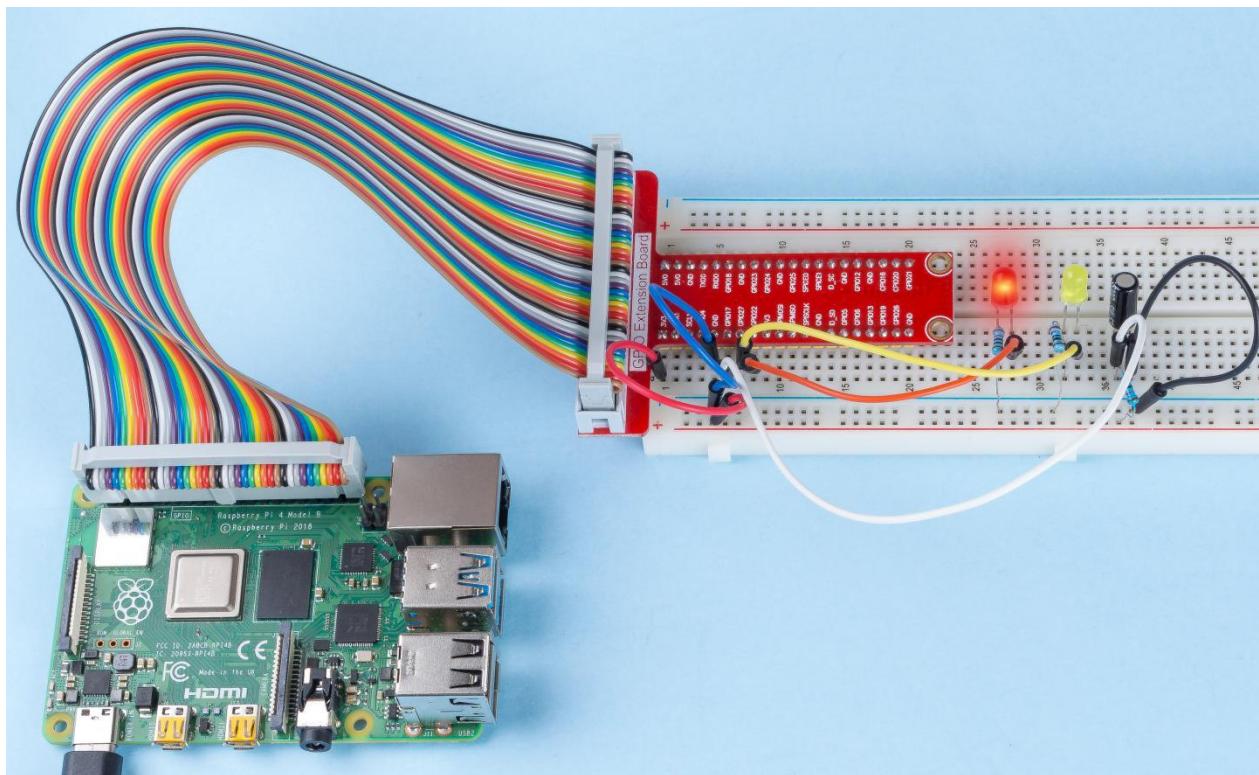
```
def Print(x):
    if x == 0:
        print ('*****')
        print ('* Tilt! *')
        print ('*****')
```

Erstellen Sie eine Funktion, Print (), um die obigen Zeichen auf dem Bildschirm zu drucken.

```
def detect(chn):
    Led(GPIO.input(TiltPin))
    Print(GPIO.input(TiltPin))
```

Definieren Sie eine Rückruffunktion für den Neigungsrückruf. Holen Sie sich den Lesewert des Neigungsschalters, dann steuert die Funktion LED () das Ein- oder Ausschalten der beiden LEDs, abhängig vom Lesewert des Neigungsschalters.

Phänomen Bild

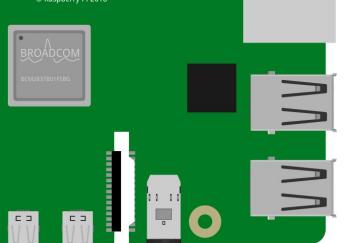
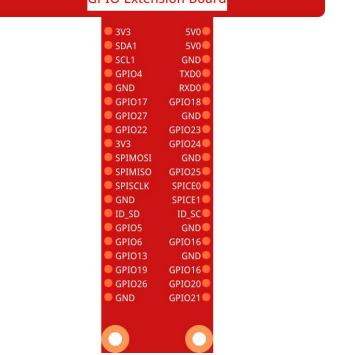
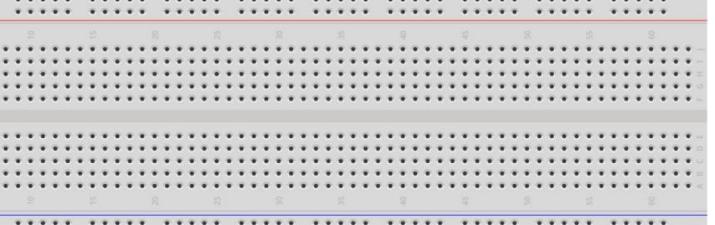


2.1.4 Potentiometer

Einführung

Die ADC-Funktion kann verwendet werden, um analoge Signale in digitale Signale umzuwandeln, und in diesem Experiment wird ADC0834 verwendet, um die Funktion zu erhalten, an der ADC beteiligt ist. Hier implementieren wir diesen Prozess mithilfe eines Potentiometers. Das Potentiometer ändert die physikalische Größe - Spannung, die von der ADC-Funktion umgewandelt wird.

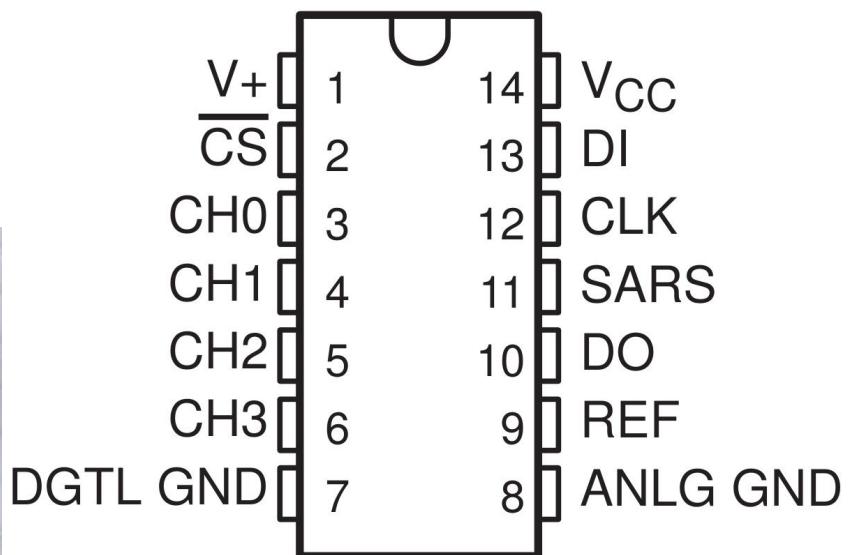
Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Potentiometer																																								
	 <p>GPIO Extension Board</p> <table border="1"> <tbody> <tr><td>3V3</td><td>SVO</td></tr> <tr><td>SDA1</td><td>SVO</td></tr> <tr><td>SCL1</td><td>GND</td></tr> <tr><td>GPIO4</td><td>TXD0</td></tr> <tr><td>GND</td><td>RXD0</td></tr> <tr><td>GPIO17</td><td>GPIO18</td></tr> <tr><td>GPIO27</td><td>GND</td></tr> <tr><td>GPIO22</td><td>GPIO23</td></tr> <tr><td>3V3</td><td>GPIO24</td></tr> <tr><td>SPI_MOSI</td><td>GND</td></tr> <tr><td>SPI_MISO</td><td>GPIO25</td></tr> <tr><td>SPI_SCLK</td><td>SPICER0</td></tr> <tr><td>GND</td><td>SPICER1</td></tr> <tr><td>ID_SD</td><td>ID_SC</td></tr> <tr><td>GPIO5</td><td>GND</td></tr> <tr><td>GPIO6</td><td>GPIO16</td></tr> <tr><td>GPIO13</td><td>GND</td></tr> <tr><td>GPIO19</td><td>GPIO16</td></tr> <tr><td>GPIO26</td><td>GPIO20</td></tr> <tr><td>GND</td><td>GPIO21</td></tr> </tbody> </table>	3V3	SVO	SDA1	SVO	SCL1	GND	GPIO4	TXD0	GND	RXD0	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GPIO24	SPI_MOSI	GND	SPI_MISO	GPIO25	SPI_SCLK	SPICER0	GND	SPICER1	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	
3V3	SVO																																									
SDA1	SVO																																									
SCL1	GND																																									
GPIO4	TXD0																																									
GND	RXD0																																									
GPIO17	GPIO18																																									
GPIO27	GND																																									
GPIO22	GPIO23																																									
3V3	GPIO24																																									
SPI_MOSI	GND																																									
SPI_MISO	GPIO25																																									
SPI_SCLK	SPICER0																																									
GND	SPICER1																																									
ID_SD	ID_SC																																									
GPIO5	GND																																									
GPIO6	GPIO16																																									
GPIO13	GND																																									
GPIO19	GPIO16																																									
GPIO26	GPIO20																																									
GND	GPIO21																																									
1 * 40-Pin Kabel		1 * ADC0834																																								
																																										
1 * Steckbrett	1 * Widerstand (220 Ω)	Mehrere Überbrückungsdrähte																																								
																																										
		1 * LED																																								
																																										

Prinzip

ADC0834

ADC0834 ist ein 8-Bit-Analog-Digital-Wandler mit sukzessiver Approximation, der mit einem eingangskonfigurierbaren Mehrkanal-Multiplexer und einem seriellen Ein- / Ausgang ausgestattet ist. Der serielle Ein- / Ausgang ist für die Schnittstelle mit Standardschieberegistern oder Mikroprozessoren konfiguriert.



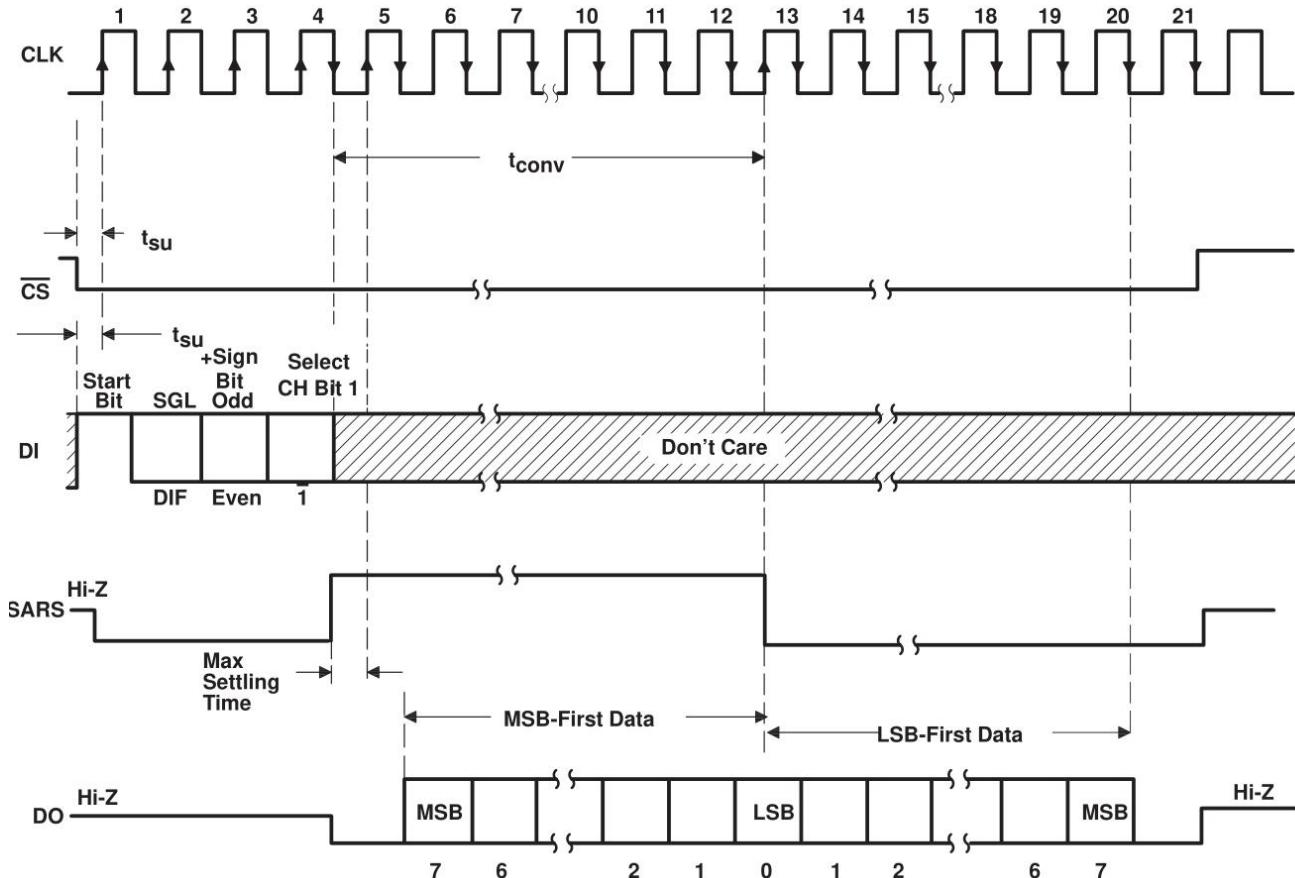
Betriebsablauf

Eine Konvertierung wird eingeleitet, indem CS auf niedrig gesetzt wird, wodurch alle Logikschaltungen aktiviert werden. CS muss für den gesamten Konvertierungsprozess niedrig gehalten werden. Ein Takteingang wird dann vom Prozessor empfangen. Bei jedem Übergang von niedrig nach hoch des Takteingangs werden die Daten auf DI in das Multiplexer-Adressschieberegister getaktet. Die erste Logik hoch am Eingang ist das Startbit. Auf das Startbit folgt ein 3- bis 4-Bit-Zuweisungswort. Bei jedem aufeinanderfolgenden Übergang von niedrig nach hoch des Takteingangs werden das Startbit und das Zuweisungswort durch das Schieberegister verschoben. Wenn das Startbit in den Startort des Multiplexerregisters verschoben wird, wird der Eingangskanal ausgewählt und die Umwandlung beginnt. Der SAR-Statu-Ausgang (SARS) geht auf High, um anzudeuten, dass eine Konvertierung läuft, und DI in das Multiplexer-Schieberegister ist während der Konvertierungsdauer deaktiviert.

Ein Intervall von einer Taktperiode wird automatisch eingefügt, damit sich der ausgewählte Multiplexkanal einstellen kann. Der Datenausgang DO kommt aus dem hochohmigen Zustand heraus und liefert ein führendes Tief für diese Eintaktperiode der Multiplexer-Einschwingzeit. Der SAR-Komparator vergleicht aufeinanderfolgende Ausgänge von der Widerstandsleiter mit dem eingehenden analogen Signal. Der Komparatortaustausch zeigt an, ob der Analogeingang größer oder kleiner als der

Widerstandsleiterausgang ist. Während der Konvertierung werden die Konvertierungsdaten gleichzeitig vom DO-Ausgangspin ausgegeben, wobei das höchstwertige Bit (MSB) zuerst angezeigt wird.

Nach acht Taktperioden ist die Konvertierung abgeschlossen und der SARS-Ausgang wird niedrig. Schließlich werden die niedrigstwertigen Bit-First-Daten nach dem MSB-First-Datenstrom ausgegeben.



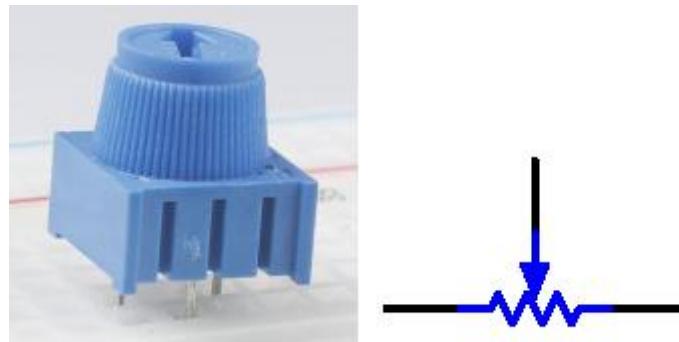
ADC0834 MUX ADDRESS CONTROL LOGIC TABLE

MUX ADDRESS			CHANNEL NUMBER			
SGL/ \overline{DIF}	ODD/EVEN	SELECT BIT 1	O	1	2	3
L	L	L	+/-	-		
		H	-/+	+	-	
	H	L	-/-		-	+
	H	H	H			
H	L	L	+/-			
		H	-/+	+	+	
	H	L	-/-			
	H	H	H			

H = high level, L = low level, – or + = polarity of selected input pin

Potentiometer

Das Potentiometer ist auch eine Widerstandskomponente mit 3 Anschlüssen und sein Widerstandswert kann gemäß einigen regelmäßigen Abweichungen eingestellt werden. Das Potentiometer besteht normalerweise aus einem Widerstand und einer beweglichen Bürste. Wenn sich die Bürste entlang des Widerstands bewegt, gibt es abhängig von der Verschiebung einen bestimmten Widerstand oder eine bestimmte Spannung.



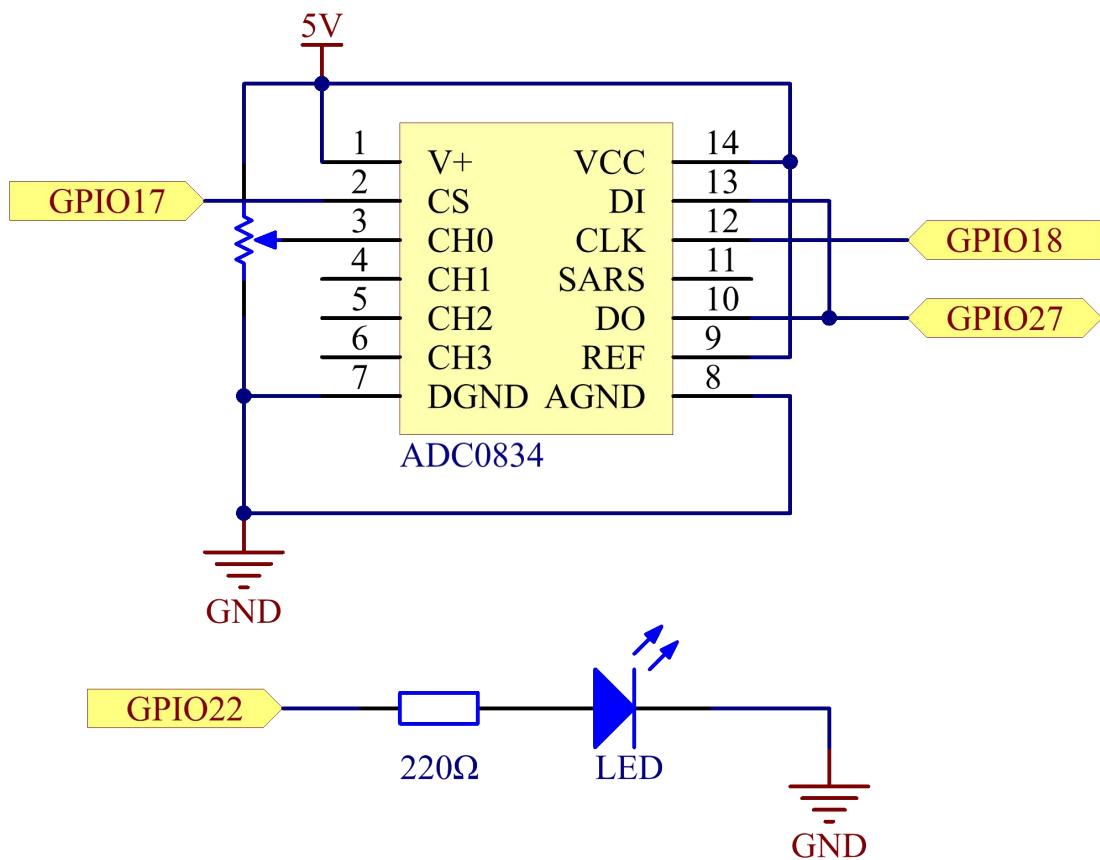
Die Funktionen des Potentiometers in der Schaltung sind wie folgt:

1. Dient als Spannungsteiler

Das Potentiometer ist ein stufenlos einstellbarer Widerstand. Wenn Sie die Welle oder den Schiebegriff des Potentiometers einstellen, gleitet der bewegliche Kontakt auf dem Widerstand. Zu diesem Zeitpunkt kann eine Spannung ausgegeben werden, die von der an das Potentiometer angelegten Spannung und dem Winkel abhängt, in den sich der bewegliche Arm gedreht hat, oder von der Entfernung, um die er sich bewegt.

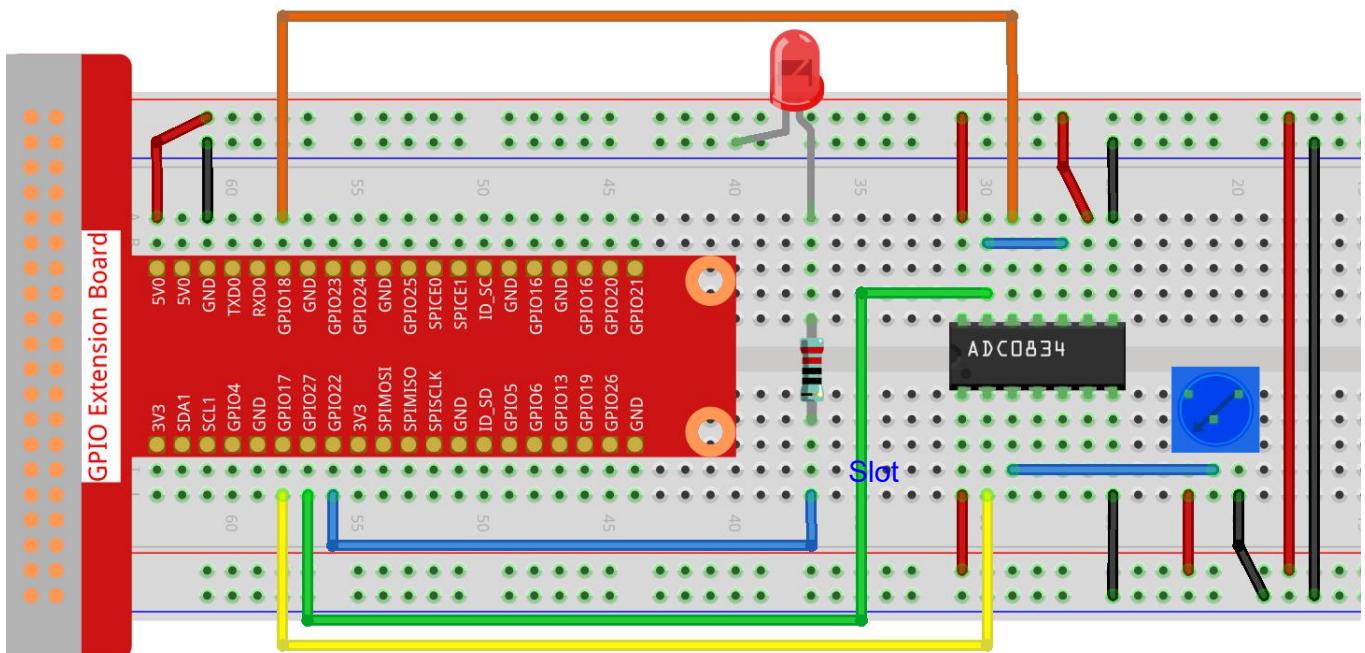
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27
GPIO22	Pin15	3	22



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



Hinweis: Bitte platzieren Sie den Chip unter Bezugnahme auf die entsprechende Position auf dem Bild. Beachten Sie, dass sich die Rillen auf dem Chip beim Platzieren links befinden sollten. .

➤ Für Benutzer in C-Sprache

Schritt 2: Öffnen Sie die Kodedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.1.4/
```

Schritt 3: Kompilieren Sie die Kode

```
gcc 2.1.4_Potentiometer.c -lwiringPi
```

Schritt 4: Ausführen.

```
sudo ./a.out
```

Nachdem der Kode ausgeführt wurde, drehen Sie die Taste am Potentiometer. Die Intensität der LED ändert sich entsprechend.

Kode

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

typedef unsigned char uchar;
typedef unsigned int uint;

#define ADC_CS    0
#define ADC_CLK   1
#define ADC_DIO   2
#define LedPin    3

uchar get_ADC_Result(uint channel)
{
    uchar i;
    uchar dat1=0, dat2=0;
    int sel = channel > 1 & 1;
    int odd = channel & 1;

    pinMode(ADC_DIO, OUTPUT);
    digitalWrite(ADC_CS, 0);
    // Start bit
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    //Single End mode
```

```

digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
// ODD
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,odd);  delayMicroseconds(2);
digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
//Select
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,sel);  delayMicroseconds(2);
digitalWrite(ADC_CLK,1);

digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,1);    delayMicroseconds(2);

for(i=0;i<8;i++)
{
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);    delayMicroseconds(2);

    pinMode(ADC_DIO, INPUT);
    dat1=dat1<<1 | digitalRead(ADC_DIO);
}

for(i=0;i<8;i++)
{
    dat2 = dat2 | ((uchar)(digitalRead(ADC_DIO))<<i);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);    delayMicroseconds(2);
}

digitalWrite(ADC_CS,1);
pinMode(ADC_DIO, OUTPUT);
return(dat1==dat2) ? dat1 : 0;
}

int main(void)
{
    uchar analogVal;
}

```

```

if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
    printf("setup wiringPi failed !");
    return 1;
}
softPwmCreate(LedPin, 0, 100);
pinMode(ADC_CS, OUTPUT);
pinMode(ADC_CLK, OUTPUT);

while(1){
    analogVal = get_ADC_Result(0);
    printf("Current analogVal : %d\n", analogVal);
    softPwmWrite(LedPin, analogVal);
    delay(100);
}
return 0;
}

```

Kode Erklärung

```

#define ADC_CS 0
#define ADC_CLK 1
#define ADC_DIO 2
#define LedPin 3

```

Definieren Sie CS, CLK, DIO von ADC0834 und verbinden Sie sie mit GPIO0, GPIO1 bzw. GPIO2. Schließen Sie dann die LED an GPIO3 an.

```

uchar get_ADC_Result(uint channel)
{
    uchar i;
    uchar dat1=0, dat2=0;
    int sel = channel > 1 & 1;
    int odd = channel & 1;

    pinMode(ADC_DIO, OUTPUT);
    digitalWrite(ADC_CS, 0);
    // Start bit
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    //Single End mode

```

```

digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
// ODD
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,odd);  delayMicroseconds(2);
digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
//Select
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,sel);  delayMicroseconds(2);
digitalWrite(ADC_CLK,1);

digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
for(i=0;i<8;i++)
{
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);    delayMicroseconds(2);

pinMode(ADC_DIO, INPUT);
dat1=dat1<<1 | digitalRead(ADC_DIO);
}

for(i=0;i<8;i++)
{
    dat2 = dat2 | ((uchar)(digitalRead(ADC_DIO))<<i);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);    delayMicroseconds(2);
}

digitalWrite(ADC_CS,1);
pinMode(ADC_DIO, OUTPUT);
return(dat1==dat2) ? dat1 : 0;
}

```

Es gibt eine Funktion von ADC0834, um die Analog-Digital-Wandlung zu erhalten.
Der spezifische Workflow lautet wie folgt:

digitalWrite(ADC_CS, 0);

Stellen Sie CS auf einen niedrigen Wert ein und aktivieren Sie die AD-Konvertierung.

```
// Start bit
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
```

Wenn der Übergang von niedrig zu hoch des Takteingangs zum ersten Mal auftritt, setzen Sie DIO als Startbit auf 1. In den folgenden drei Schritten gibt es 3 Zuweisungswörter.

```
//Single End mode
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
```

Sobald der von niedrig zu hoch Übergang des Takteingangs zum zweiten Mal erfolgt, setzen Sie DIO auf 1 und wählen Sie den SGL-Modus.

```
// ODD
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,odd);  delayMicroseconds(2);
digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
```

Einmal zum dritten Mal auftritt, wird der Wert von DIO durch die Variable **ungerade** gesteuert.

```
//Select
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,sel);  delayMicroseconds(2);
digitalWrite(ADC_CLK,1);
```

Wenn der Impuls von CLK zum vierten Mal von einem niedrigen auf einen hohen Niveau umgewandelt wird, wird der Wert von DIO durch die Variable **sel** gesteuert.

Unter der Bedingung, dass Kanal = 0, sel = 0, ungerade = 0 ist, lauten die Betriebsformeln bezüglich **sel** und **ungerade** wie folgt:

```
int sel = channel > 1 & 1;
int odd = channel & 1;
```

Wenn die Bedingung erfüllt ist, dass Kanal = 1, sel = 0, ungerade = 1 ist, lesen Sie bitte die folgende Adresssteuerungslogiktabelle. Hier wird CH1 gewählt und das Startbit wird in den Startort des Multiplexerregisters verschoben und die Umwandlung beginnt.

MUX ADDRESS			CHANNEL NUMBER			
SGL/DIF	ODD/EVEN	SELECT BIT 1	O	1	2	3
L	L	L	+	-		
L	L	H			+	-
L	H	L	-	+		
L	H	H			-	+
H	L	L	+			
H	L	H			+	
H	H	L		+		
H	H	H				+

```
digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
```

Hier setzen Sie DIO zweimal auf 1, bitte ignorieren Sie es.

```
for(i=0;i<8;i++)
{
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);    delayMicroseconds(2);

    pinMode(ADC_DIO, INPUT);
    dat1=dat1<<1 | digitalRead(ADC_DIO);
}
```

Stellen Sie in der ersten for() - Anweisung DIO in den Eingangsmodus, sobald der fünfte Impuls von CLK von einem hohen Niveau in einen niedrigen Niveau umgewandelt wurde. Dann beginnt die Konvertierung und der konvertierte Wert wird in der Variablen dat1 gespeichert. Nach acht Taktperioden ist die Konvertierung abgeschlossen.

```
for(i=0;i<8;i++)
{
    dat2 = dat2 | ((uchar)(digitalRead(ADC_DIO))<<i);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);    delayMicroseconds(2);
}
```

Geben Sie in der zweiten for() - Anweisung die konvertierten Werte nach weiteren acht Taktperioden über DO aus und speichern Sie sie in der Variablen dat2.

```
digitalWrite(ADC_CS,1);
```

```
pinMode(ADC_DIO, OUTPUT);
return(dat1==dat2) ? dat1 : 0;
```

return(dat1==dat2) ? dat1 : 0 wird verwendet, um den während der Konvertierung erhaltenen Wert mit dem Ausgabewert zu vergleichen. Wenn sie gleich sind, geben Sie den Konvertierungswert dat1 aus. Andernfalls wird 0 ausgegeben. Hier ist der Workflow von ADC0834 abgeschlossen.

```
softPwmCreate(LedPin, 0, 100);
```

Die Funktion besteht darin, mithilfe von Software einen PWM-Pin, LedPin, zu erstellen, dann die anfängliche Impulsbreite auf 0 zu setzen und die PWM-Periode 100 x 100us zu betragen.

```
while(1){
    analogVal = get_ADC_Result(0);
    printf("Current analogVal : %d\n", analogVal);
    softPwmWrite(LedPin, analogVal);
    delay(100);
}
```

Lesen Sie im Hauptprogramm den Wert von Kanal 0 ab, der mit einem Potentiometer verbunden wurde. Speichern Sie den Wert in der Variablen analogVal und schreiben Sie ihn in LedPin. Jetzt können Sie sehen, wie sich die Helligkeit der LED mit dem Wert des Potentiometers ändert.

➤ Für Python-Benutzer

Schritt 2: Öffnen Sie die Kodedatei

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Ausführen.

```
sudo python3 2.1.4_Potentiometer.py
```

Nachdem der Kode ausgeführt wurde, drehen Sie die Taste am Potentiometer. Die Intensität der LED ändert sich entsprechend.

Kode

```
#!/usr/bin/env python3
```

```
import RPi.GPIO as GPIO
import ADC0834
import time
```

```
LedPin = 22
```

```
def setup():
    global led_val
    # Set the GPIO modes to BCM Numbering
    GPIO.setmode(GPIO.BCM)
    # Set all LedPin's mode to output and initial level to High(3.3v)
    GPIO.setup(LedPin, GPIO.OUT, initial=GPIO.HIGH)
    ADC0834.setup()
    # Set led as pwm channel and frequece to 2KHz
    led_val = GPIO.PWM(LedPin, 2000)

    # Set all begin with value 0
    led_val.start(0)

# Define a MAP function for mapping values. Like from 0~255 to 0~100
def MAP(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def destroy():
    # Stop all pwm channel
    led_val.stop()
    # Release resource
    GPIO.cleanup()

def loop():
    while True:
        res = ADC0834.getResult()
        print ('res = %d' % res)
        R_val = MAP(res, 0, 255, 0, 100)
        led_val.ChangeDutyCycle(R_val)
        time.sleep(0.2)

if __name__ == '__main__':
    setup()
    try:
```

```

loop()
except KeyboardInterrupt: # When 'Ctrl+C' is pressed, the program destroy() will
be executed.

destroy()

```

Kode Erklärung

```
import ADC0834
```

Importieren ADC0834-Bibliothek Sie können den Inhalt der Bibliothek überprüfen, indem Sie den Befehl nano ADC0834.py aufrufen.

```

def setup():
    global led_val
    # Set the GPIO modes to BCM Numbering
    GPIO.setmode(GPIO.BCM)
    # Set all LedPin's mode to output and initial level to High(3.3v)
    GPIO.setup(LedPin, GPIO.OUT, initial=GPIO.HIGH)
    ADC0834.setup()
    # Set led as pwm channel and frequece to 2KHz
    led_val = GPIO.PWM(LedPin, 2000)

    # Set all begin with value 0
    led_val.start(0)

```

Definieren Sie in setup() die Benennungsmethode als BCM, legen Sie LedPin als PWM-Kanal fest und rendern Sie eine Frequenz von 2Khz.

ADC0834.setup(): Initialisieren Sie ADC0834 und verbinden Sie das definierte CS, CLK, DIO von ADC0834 mit GPIO17, GPIO18 bzw. GPIO27.

```

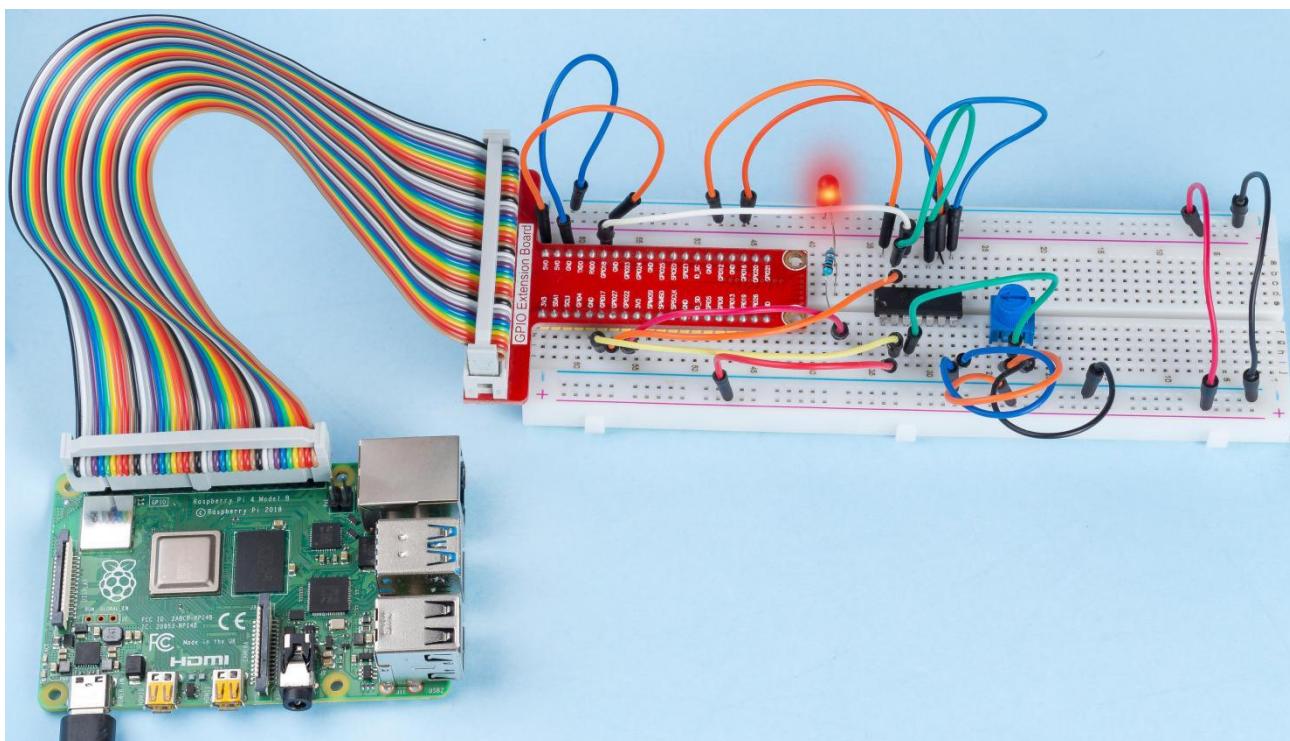
def loop():
    while True:
        res = ADC0834.getResult()
        print ('res = %d' % res)
        R_val = MAP(res, 0, 255, 0, 100)
        led_val.ChangeDutyCycle(R_val)
        time.sleep(0.2)

```

Mit der Funktion `getResult()` werden die Analogwerte der vier Kanäle von ADC0834 gelesen. Standardmäßig liest die Funktion den Wert von CH0. Wenn Sie andere Kanäle lesen möchten, geben Sie bitte die Kanalnummer in `()` ein, z. `getResult(1)`.

Die Funktion `loop()` liest zuerst den Wert von CH0 und weist ihn dann der Variablen `res` zu. Rufen Sie danach die Funktion `MAP` auf, um den Lesewert des Potentiometers auf $0 \sim 100$ abzubilden. Dieser Schritt wird verwendet, um den Arbeitszyklus von `LedPin` zu steuern. Jetzt können Sie sehen, dass sich die Helligkeit der LED mit dem Wert des Potentiometers ändert.

Phänomen Bild

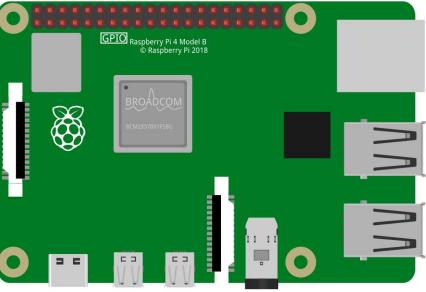
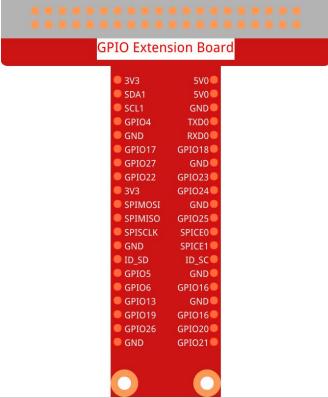
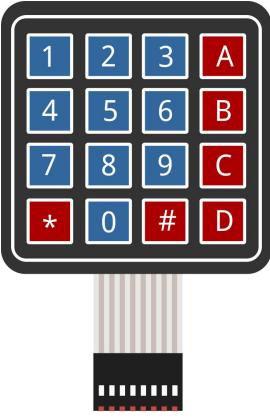
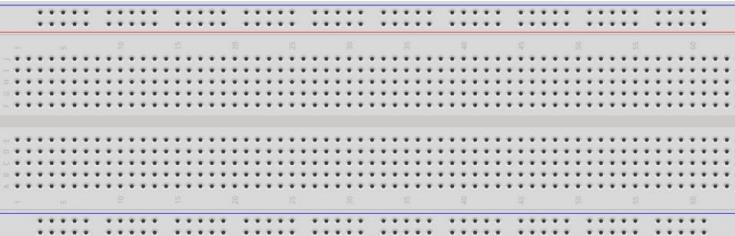
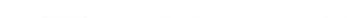


2.1.5 Tastatur

Einführung

Eine Tastatur ist eine rechteckige Anordnung von Schaltflächen. In diesem Projekt werden wir Eingabezeichen verwenden.

Komponenten

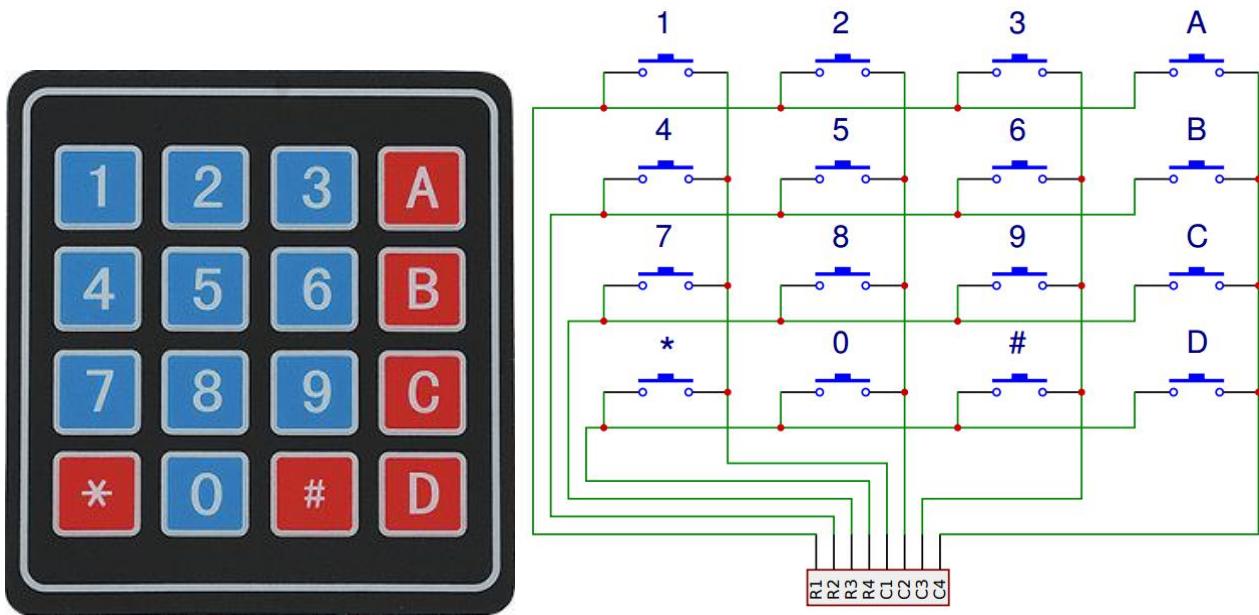
1 * Raspberry Pi	1 * T- Erweiterungskarte	1 * Tastatur																																								
	 GPIO Extension Board <table border="1"><tr><td>3V3</td><td>5V0</td></tr><tr><td>SDA1</td><td>5V0</td></tr><tr><td>SCL1</td><td>GND</td></tr><tr><td>GPIO4</td><td>TXD0</td></tr><tr><td>GND</td><td>RXD0</td></tr><tr><td>GPIO17</td><td>GPIO18</td></tr><tr><td>GPIO27</td><td>GND</td></tr><tr><td>GPIO22</td><td>GPIO23</td></tr><tr><td>3V3</td><td>GPIO24</td></tr><tr><td>SPI_MOSI</td><td>GND</td></tr><tr><td>SPI_MISO</td><td>GPIO25</td></tr><tr><td>SPI_SCLK</td><td>SPICEO</td></tr><tr><td>GND</td><td>SPI_CE</td></tr><tr><td>ID_SD</td><td>ID_SC</td></tr><tr><td>GPIO5</td><td>GND</td></tr><tr><td>GPIO6</td><td>GPIO16</td></tr><tr><td>GPIO13</td><td>GND</td></tr><tr><td>GPIO19</td><td>GPIO16</td></tr><tr><td>GPIO26</td><td>GPIO20</td></tr><tr><td>GND</td><td>GPIO21</td></tr></table>	3V3	5V0	SDA1	5V0	SCL1	GND	GPIO4	TXD0	GND	RXD0	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GPIO24	SPI_MOSI	GND	SPI_MISO	GPIO25	SPI_SCLK	SPICEO	GND	SPI_CE	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	
3V3	5V0																																									
SDA1	5V0																																									
SCL1	GND																																									
GPIO4	TXD0																																									
GND	RXD0																																									
GPIO17	GPIO18																																									
GPIO27	GND																																									
GPIO22	GPIO23																																									
3V3	GPIO24																																									
SPI_MOSI	GND																																									
SPI_MISO	GPIO25																																									
SPI_SCLK	SPICEO																																									
GND	SPI_CE																																									
ID_SD	ID_SC																																									
GPIO5	GND																																									
GPIO6	GPIO16																																									
GPIO13	GND																																									
GPIO19	GPIO16																																									
GPIO26	GPIO20																																									
GND	GPIO21																																									
1 * Steckbrett		8 * Widerstand 10KΩ																																								
																																										
1 * 40-Pin Kabel		Mehrere Überbrückungsdrähte																																								
																																										

Prinzip

Tastenfeld

Eine Tastatur ist eine rechteckige Anordnung von 12 oder 16 AUS- (EIN) Tasten. Der Zugriff auf ihre Kontakte erfolgt über einen Header, der zum Anschluss mit einem Flachbandkabel oder zum Einsetzen in eine Leiterplatte geeignet ist. Bei einigen

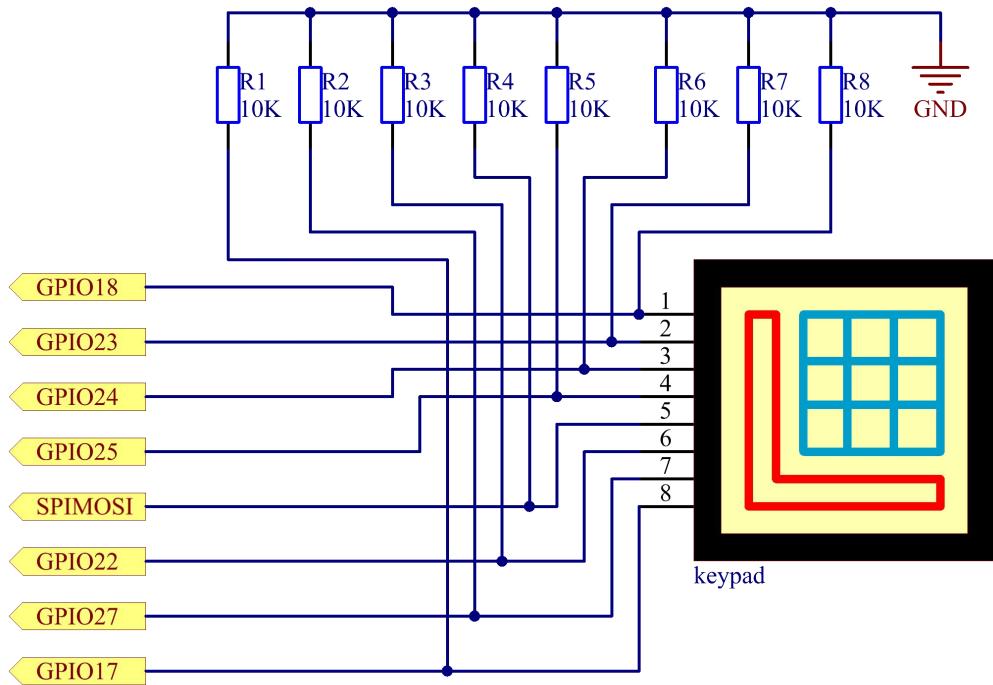
Tastaturen ist jede Taste mit einem separaten Kontakt in der Kopfzeile verbunden, während alle Tasten eine gemeinsame Masse haben.



Häufiger sind die Tasten Matrix kodiert, was bedeutet, dass jede von ihnen ein eindeutiges Leiterpaar in einer Matrix überbrückt. Diese Konfiguration eignet sich zum Abfragen durch einen Mikrocontroller, der so programmiert werden kann, dass er nacheinander einen Ausgangsimpuls an jeden der vier horizontalen Drähte sendet. Während jedes Impulses werden die verbleibenden vier vertikalen Drähte nacheinander überprüft, um festzustellen, welcher, falls vorhanden, ein Signal führt. Pullup- oder Pulldown-Widerstände sollten zu den Eingangsleitungen hinzugefügt werden, um zu verhindern, dass sich die Eingänge des Mikrocontrollers unvorhersehbar verhalten, wenn kein Signal vorhanden ist.

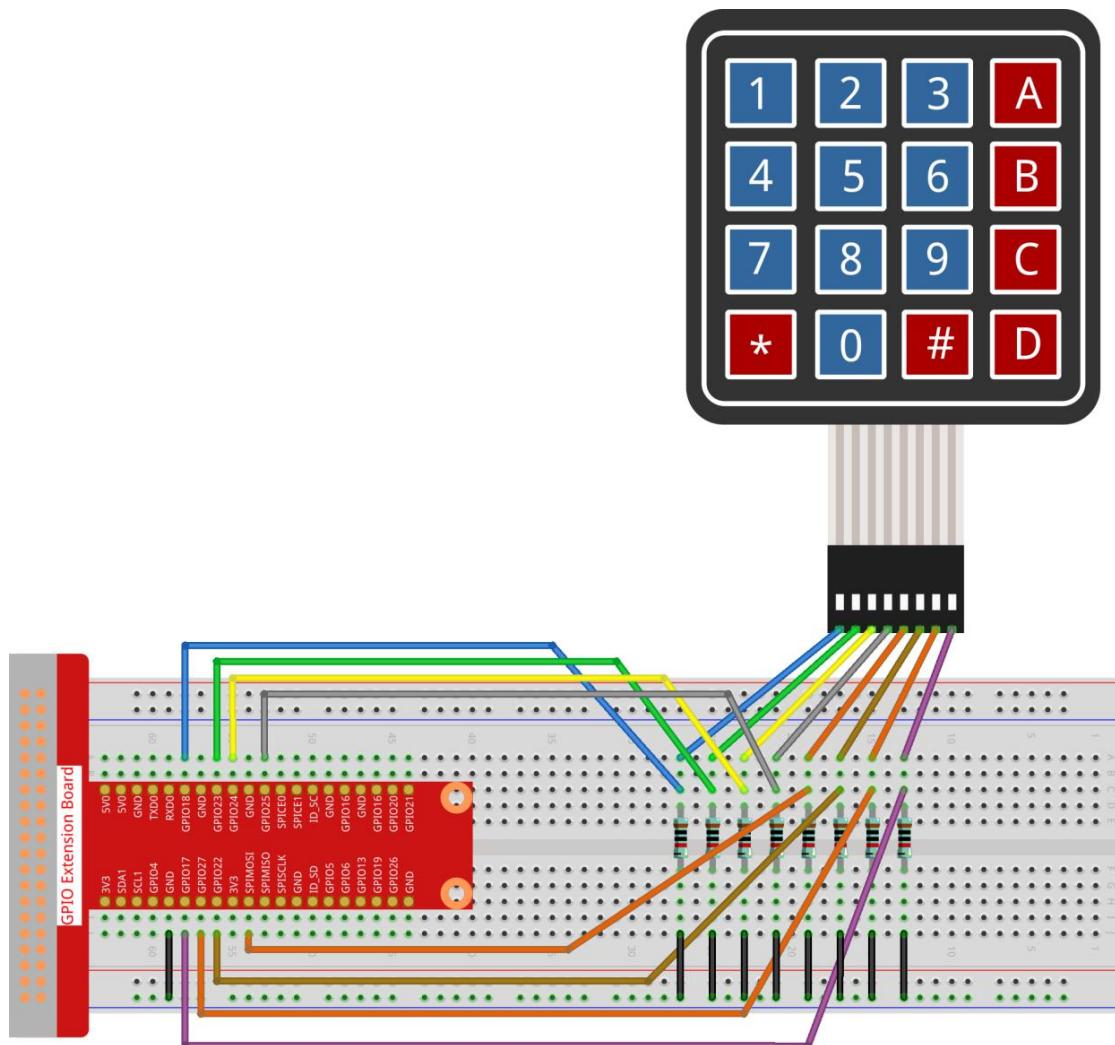
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO18	Pin 12	1	18
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24
GPIO25	Pin 22	6	25
SPI MOSI	Pin 19	12	10
GPIO22	Pin 15	3	22
GPIO27	Pin 13	2	27
GPIO17	Pin 11	0	17



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Öffnen Sie die Kodedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.1.5/
```

Schritt 3: Kompilieren Sie die Kode

```
gcc 2.1.5_Keypad.cpp -lwiringPi
```

Schritt 4: Ausführen.

```
sudo ./a.out
```

Nachdem die Kode ausgeführt wurde, werden die Werte der gedrückten Tasten auf der Tastatur (Tastenwert) auf dem Bildschirm gedruckt.

Kode

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

#define ROWS 4
#define COLS 4
#define BUTTON_NUM (ROWS * COLS)

unsigned char KEYS[BUTTON_NUM] {
    '1','2','3','A',
    '4','5','6','B',
    '7','8','9','C',
    '*', '0', '#', 'D'};

unsigned char rowPins[ROWS] = {1, 4, 5, 6};
unsigned char colPins[COLS] = {12, 3, 2, 0};

void keyRead(unsigned char* result);
bool keyCompare(unsigned char* a, unsigned char* b);
void keyCopy(unsigned char* a, unsigned char* b);
void keyPrint(unsigned char* a);
void keyClear(unsigned char* a);
int keyIndexOf(const char value);

void init(void) {
```

```

for(int i=0 ; i<4 ; i++) {
    pinMode(rowPins[i], OUTPUT);
    pinMode(colPins[i], INPUT);
}
}

int main(void){
    unsigned char pressed_keys[BUTTON_NUM];
    unsigned char last_key_pressed[BUTTON_NUM];

    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    init();
    while(1){
        keyRead(pressed_keys);
        bool comp = keyCompare(pressed_keys, last_key_pressed);
        if (!comp){
            keyPrint(pressed_keys);
            keyCopy(last_key_pressed, pressed_keys);
        }
        delay(100);
    }
    return 0;
}

void keyRead(unsigned char* result){
    int index;
    int count = 0;
    keyClear(result);
    for(int i=0 ; i<ROWS ; i++ ){
        digitalWrite(rowPins[i], HIGH);
        for(int j =0 ; j < COLS ; j++){
            index = i * ROWS + j;
            if(digitalRead(colPins[j]) == 1){
                result[count]=KEYS[index];
            }
        }
    }
}

```

```

        count += 1;
    }
}
delay(1);
digitalWrite(rowPins[i], LOW);
}
}

bool keyCompare(unsigned char* a, unsigned char* b){
    for (int i=0; i<BUTTON_NUM; i++){
        if (a[i] != b[i]){
            return false;
        }
    }
    return true;
}

void keyCopy(unsigned char* a, unsigned char* b){
    for (int i=0; i<BUTTON_NUM; i++){
        a[i] = b[i];
    }
}

void keyPrint(unsigned char* a){
    if (a[0] != 0){
        printf("%c",a[0]);
    }
    for (int i=1; i<BUTTON_NUM; i++){
        if (a[i] != 0){
            printf(", %c",a[i]);
        }
    }
    printf("\n");
}

void keyClear(unsigned char* a){
    for (int i=0; i<BUTTON_NUM; i++){

```

```

    a[i] = 0;
}
}

int keyIndexOf(const char value){
    for (int i=0; i<BUTTON_NUM; i++){
        if ((const char)KEYS[i] == value){
            return i;
        }
    }
    return -1;
}

```

Kode Erklärung

```

unsigned char KEYS[BUTTON_NUM] {
    '1','2','3','A',
    '4','5','6','B',
    '7','8','9','C',
    '*','0','#','D';

unsigned char rowPins[ROWS] = {1, 4, 5, 6};
unsigned char colPins[COLS] = {12, 3, 2, 0};

```

Deklarieren Sie jede Taste der Matrixtastatur zu den Array- keys[] und definieren Sie die Pins für jede Zeile und Spalte.

```

while(1){
    keyRead(pressed_keys);
    bool comp = keyCompare(pressed_keys, last_key_pressed);
    if (!comp){
        keyPrint(pressed_keys);
        keyCopy(last_key_pressed, pressed_keys);
    }
    delay(100);
}

```

Dies ist der Teil der Hauptfunktion, der den Tastenwert liest und druckt.

Die Funktion keys[] liest den Status jeder Taste.

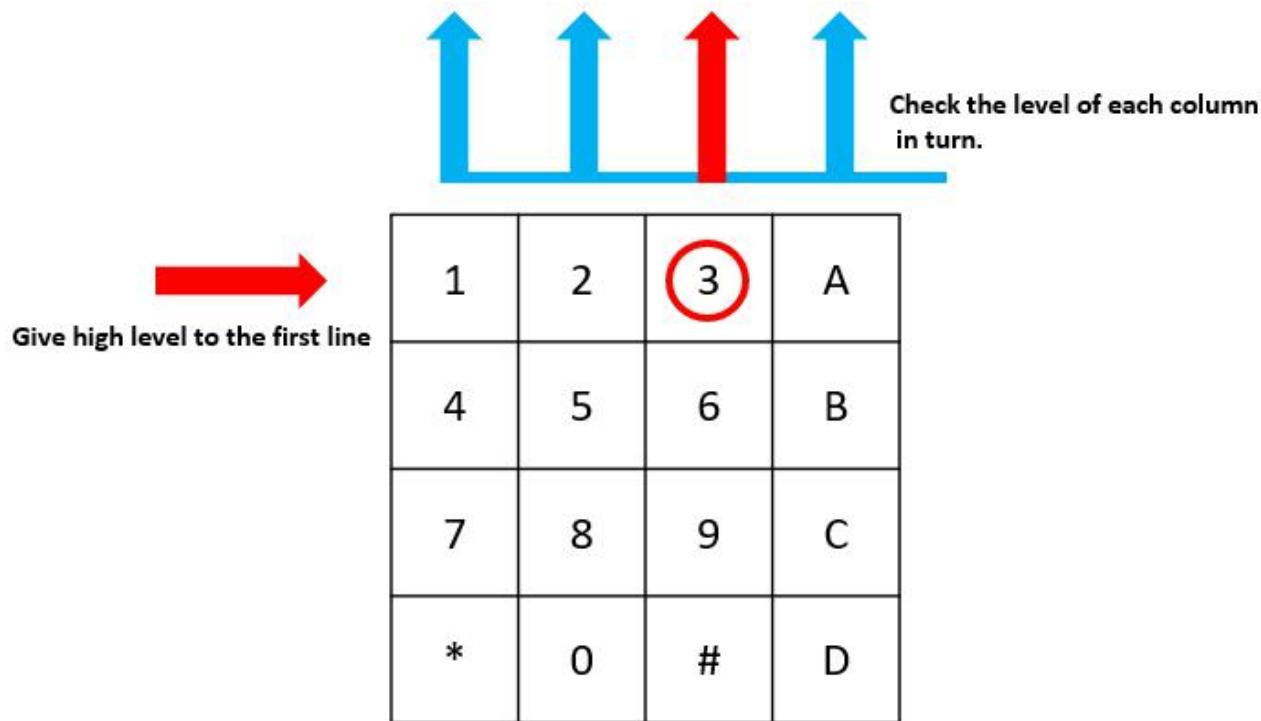
Mit KeyCompare() und keyCopy () wird beurteilt, ob sich der Status einer Schaltfläche geändert hat (dh eine Schaltfläche wurde gedrückt oder losgelassen).

keyPrint() druckt den Tastenwert der Taste, deren aktueller Niveau hoch ist (die Taste wird gedrückt).

```
void keyRead(unsigned char* result){  
    int index;  
    int count = 0;  
    keyClear(result);  
    for(int i=0 ; i<ROWS ; i++ ){  
        digitalWrite(rowPins[i], HIGH);  
        for(int j =0 ; j < COLS ; j ++){  
            index = i * ROWS + j;  
            if(digitalRead(colPins[j]) == 1){  
                result[count]=KEYS[index];  
                count += 1;  
            }  
        }  
        delay(1);  
        digitalWrite(rowPins[i], LOW);  
    }  
}
```

Diese Funktion weist jeder Zeile nacheinander eine hohe Ebene zu, und wenn die Taste in der Spalte gedrückt wird, erhält die Spalte, in der sich die Taste befindet, eine hohe Ebene. Nach der zweischichtigen Schleifenbeurteilung generiert die Schlüsselzustandskomplilierung ein Array (reasult[]).

Beim Drücken von Taste 3:



The button whose value is "3" is pressed.

RowPin [0] schreibt auf der hohen Ebene und colPin[2] erhält die hohe Ebene. ColPin [0], colPin[1], colPin[3] erhalten den niedrigen Wert.

Dies gibt uns 0,0,1,0. Wenn rowPin[1], rowPi [2] und rowPin[3] auf hoher Ebene geschrieben werden, wird colPin[0] ~ colPin[4] auf niedriger Ebene.

Nach Abschluss der Schleifenbeurteilung wird ein Array generiert:

```
result[BUTTON_NUM] {
    0, 0, 1, 0,
    0, 0, 0, 0,
    0, 0, 0, 0,
    0, 0, 0, 0};
```

```
bool keyCompare(unsigned char* a, unsigned char* b){
    for (int i=0; i<BUTTON_NUM; i++){
        if (a[i] != b[i]){
            return false;
        }
    }
    return true;
}
```

```
void keyCopy(unsigned char* a, unsigned char* b){
    for (int i=0; i<BUTTON_NUM; i++){
        a[i] = b[i];
    }
}
```

Diese beiden Funktionen werden verwendet, um zu beurteilen, ob sich der Tastenstatus geändert hat. Wenn Sie beispielsweise Ihre Hand loslassen, wenn Sie '3' oder '2' drücken, gibt keyCompare () false zurück.

Mit KeyCopy() wird der aktuelle Schaltflächenwert für ein Array (last_key_pressed[BUTTON_NUM]) nach jedem Vergleich neu geschrieben. So können wir sie beim nächsten Mal vergleichen.

```
void keyPrint(unsigned char* a){
    //printf("{}");
    if (a[0] != 0){
        printf("%c",a[0]);
    }
    for (int i=1; i<BUTTON_NUM; i++){
        if (a[i] != 0){
            printf(", %c",a[i]);
        }
    }
    printf("\n");
}
```

Mit dieser Funktion wird der Wert der aktuell gedrückten Taste gedruckt. Wenn die Taste '1' gedrückt wird, wird die '1' gedruckt. Wenn die Taste '1' gedrückt wird und die Taste '3' gedrückt wird, wird die '1, 3' gedruckt.

➤ Für Python-Sprachbenutzer

Schritt 2: Öffnen Sie die Kodedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Step 3: Run.

```
sudo python3 2.1.5_Keypad.py
```

Nachdem die Kode ausgeführt wurde, werden die Werte der gedrückten Tasten auf der Tastatur (Tastenwert) auf dem Bildschirm gedruckt.

Kode

```
import RPi.GPIO as GPIO
import time

class Keypad():

    def __init__(self, rowsPins, colsPins, keys):
        self.rowsPins = rowsPins
        self.colsPins = colsPins
        self.keys = keys
        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.rowsPins, GPIO.OUT, initial=GPIO.LOW)
        GPIO.setup(self.colsPins, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

    def read(self):
        pressed_keys = []
        for i, row in enumerate(self.rowsPins):
            GPIO.output(row, GPIO.HIGH)
            for j, col in enumerate(self.colsPins):
                index = i * len(self.colsPins) + j
                if (GPIO.input(col) == 1):
                    pressed_keys.append(self.keys[index])
            GPIO.output(row, GPIO.LOW)
        return pressed_keys

    def setup():
        global keypad, last_key_pressed
        rowsPins = [18,23,24,25]
        colsPins = [10,22,27,17]
        keys = ["1","2","3","A",
               "4","5","6","B",
               "7","8","9","C",
               "*","0","#","D"]
        keypad = Keypad(rowsPins, colsPins, keys)
        last_key_pressed = []
```

```

def loop():
    global keypad, last_key_pressed
    pressed_keys = keypad.read()
    if len(pressed_keys) != 0 and last_key_pressed != pressed_keys:
        print(pressed_keys)
    last_key_pressed = pressed_keys
    time.sleep(0.1)

# Define a destroy function for clean up everything after the script finished
def destroy():
    # Release resource
    GPIO.cleanup()

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

```

Kode Erklärung

```

def setup():
    global keypad, last_key_pressed
    rowsPins = [18,23,24,25]
    colsPins = [10,22,27,17]
    keys = ["1","2","3","A",
            "4","5","6","B",
            "7","8","9","C",
            "*","0","#","D"]
    keypad = Keypad(rowsPins, colsPins, keys)
    last_key_pressed = []

```

Deklarieren Sie jede Taste der Matrixtastatur zu den Array- keys[] und definieren Sie die Pins für jede Zeile und Spalte.

```

def loop():
    global keypad, last_key_pressed

```

```
pressed_keys = keypad.read()
if len(pressed_keys) != 0 and last_key_pressed != pressed_keys:
    print(pressed_keys)
last_key_pressed = pressed_keys
time.sleep(0.1)
```

Dies ist der Teil der Hauptfunktion, der den Tastenwert liest und druckt.

Die Funktion keys[] liest den Status jeder Taste.

Die Anweisung if len(pressed_keys) != 0 und last_key_pressed != pressed_keys wird zur Beurteilung verwendet

ob eine Taste gedrückt wird und der Status der gedrückten Taste. (Wenn Sie '3' drücken, während Sie '1' drücken, ist das Urteil haltbar.)

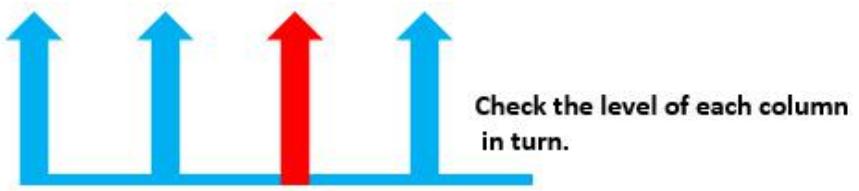
Drückt den Wert der aktuell gedrückten Taste, wenn die Bedingung haltbar ist.

Die Anweisung last_key_pressed = pressed_keys weist einem Array last_key_pressed den Status jeder Beurteilung zu, um die nächste Runde der bedingten Beurteilung zu erleichtern.

```
def read(self):
    pressed_keys = []
    for i, row in enumerate(self.rowsPins):
        GPIO.output(row, GPIO.HIGH)
        for j, col in enumerate(self.colsPins):
            index = i * len(self.colsPins) + j
            if (GPIO.input(col) == 1):
                pressed_keys.append(self.keys[index])
        GPIO.output(row, GPIO.LOW)
    return pressed_keys
```

Diese Funktion weist jeder Zeile nacheinander eine hohe Ebene zu, und wenn die Schaltfläche in der Spalte gedrückt wird, erhält die Spalte, in der sich die Taste befindet, eine hohe Ebene. Nachdem die Zweischichtschleife beurteilt wurde, wird der Wert der Schaltfläche, deren Status 1 ist, im Array press_keys gespeichert.

Wenn Sie die Taste '3' drücken:



Give high level to the first line

1	2	3	A
4	5	6	B
7	8	9	C
*	0	#	D

The button whose value is "3" is pressed.

rowPins[0] wird auf hoher Ebene geschrieben, und colPins[2] wird auf hoher Ebene geschrieben.

colPins[0] 、 colPins[1] 、 colPins[3] erhalten einen niedrigen Wert.

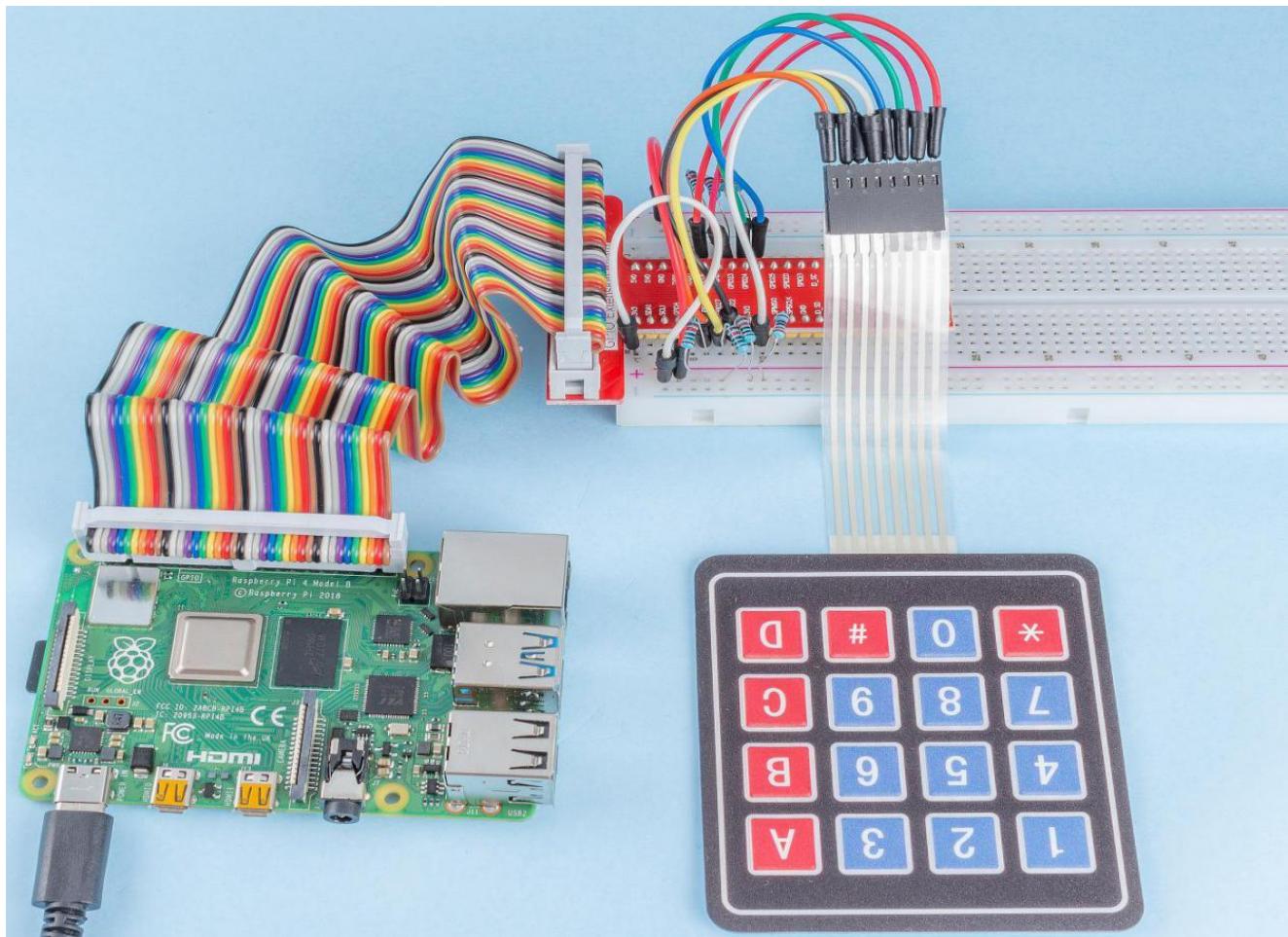
Es gibt vier Zustände: 0, 0, 1, 0; und wir schreiben '3' in press_keys.

Wenn rowPins[1], rowPins[2], rowPins[3] auf eine hohe Ebene geschrieben werden, erhalten colPins[0] ~ colPins[4] eine niedrige Ebene.

Die Schleife wurde gestoppt, dort wird press_keys = '3' zurückgegeben.

Wenn Sie die Tasten '1' und '3' drücken, wird pressed_keys = ['1','3'] zurückgegeben.

Phänomen Bild

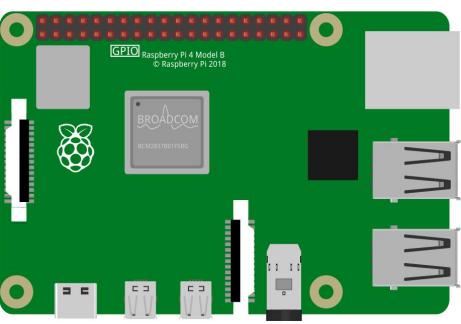
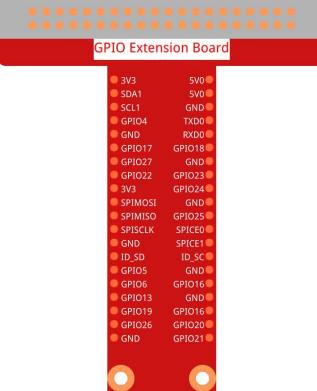
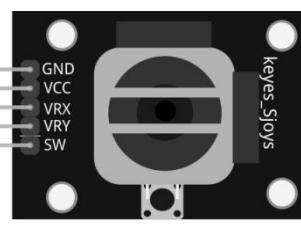
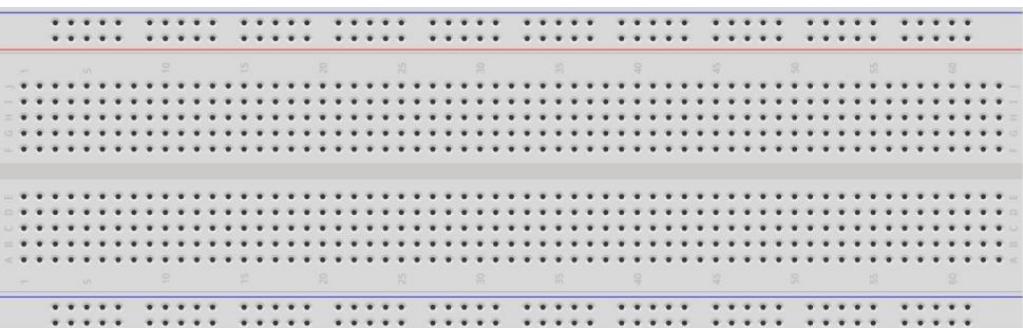


2.1.6 Joystick

Einführung

In diesem Projekt lernen wir, wie der Joystick funktioniert. Wir manipulieren den Joystick und zeigen die Ergebnisse auf dem Bildschirm an.

Komponenten

1 * Raspberry Pi	1 * T- Erweiterungskarte	1 * Joystick
		
1 * 40-Pin Kabel		1 * Widerstand 10 kΩ
		
1 * Steckbrett		1 * ADC0834
		
		Mehrere Überbrückungsdrähte
		

Prinzip

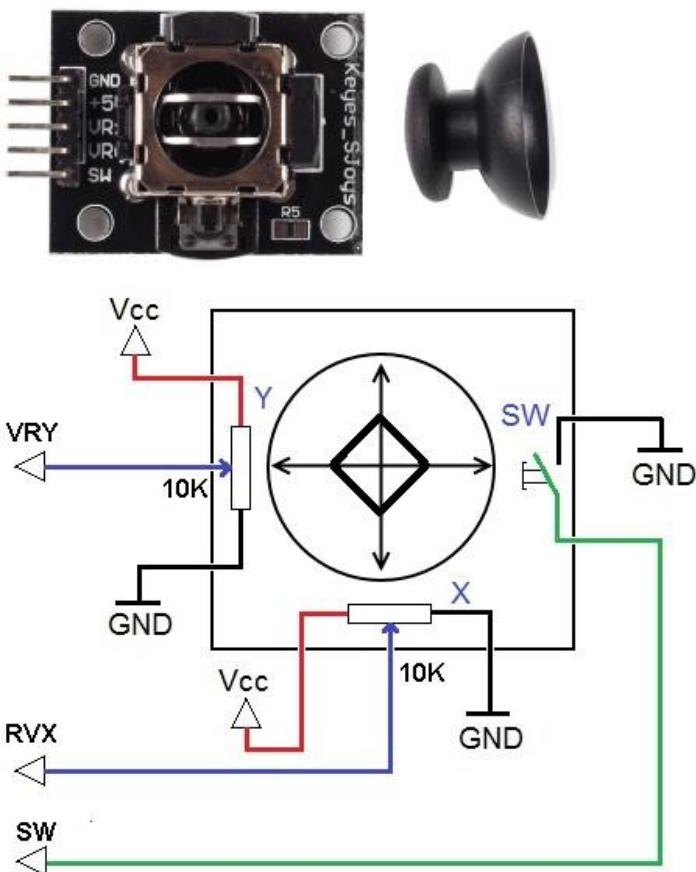
Joystick

Die Grundidee eines Joysticks besteht darin, die Bewegung eines Sticks in elektronische Informationen umzuwandeln, die ein Computer verarbeiten kann.

Um dem Computer einen vollständigen Bewegungsbereich zu übermitteln, muss ein Joystick die Position des Sticks auf zwei Achsen messen - der X-Achse (von links nach rechts) und der Y-Achse (nach oben und unten). Genau wie in der Grundgeometrie bestimmen die XY-Koordinaten genau die Position des Sticks.

Um die Position des Steuerknüppels zu bestimmen, überwacht das Joystick-Steuerungssystem einfach die Position jeder Welle. Das herkömmliche analoge Joystick-Design macht dies mit zwei Potentiometern oder variablen Widerständen.

Der Joystick verfügt außerdem über einen digitalen Eingang, der beim Herunterdrücken des Joysticks betätigt wird.

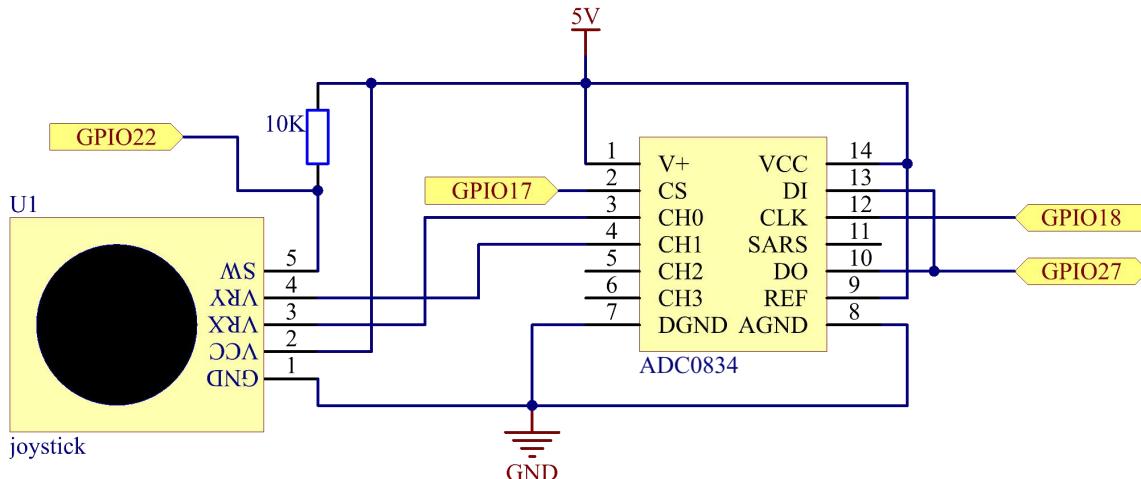


Schematische Darstellung

Wenn die Daten des Joysticks gelesen werden, gibt es einige Unterschiede zwischen den Achsen: Die Daten der X- und Y-Achse sind analog. Daher muss ADC0834 verwendet werden, um den Analogwert in einen Digitalwert umzuwandeln. Die Daten

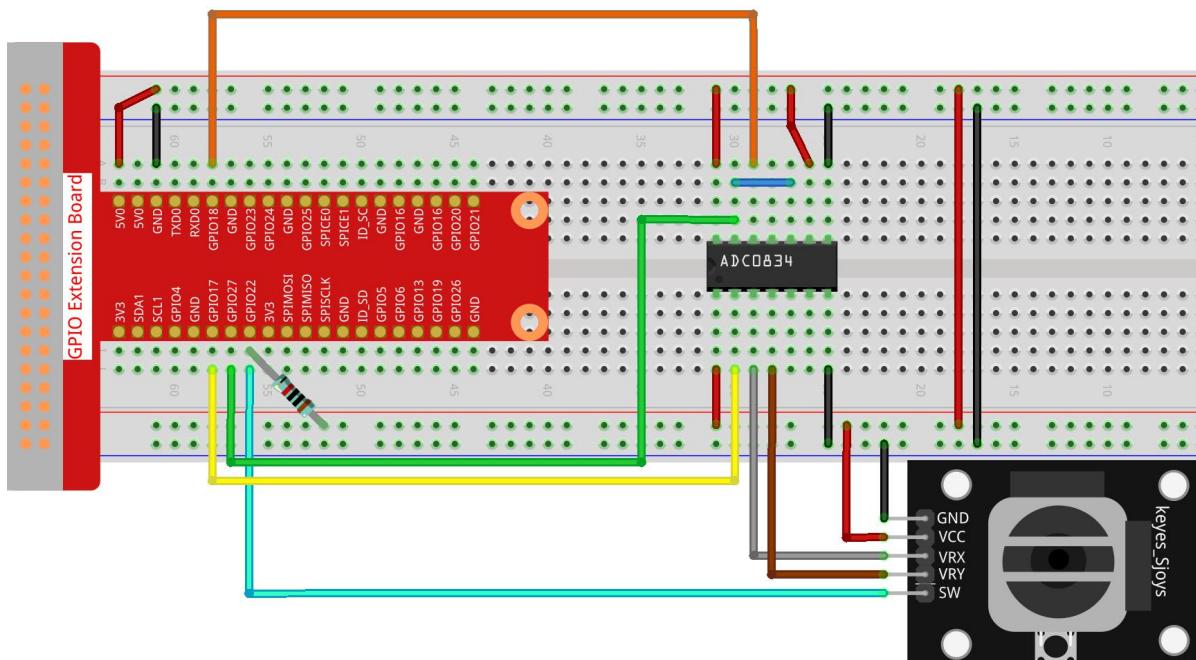
der Z-Achse sind digital, sodass Sie den GPIO direkt zum Lesen oder den ADC zum Lesen verwenden können.

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27
GPIO22	Pin15	3	22



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.1.6/
```

Schritt 3: Kompilieren Sie die Kode.

```
gcc 2.1.6_Joystick.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Nachdem die Kode ausgeführt wurde, drehen Sie den Joystick, und die entsprechenden Werte von x, y, Btn werden auf dem Bildschirm angezeigt.

Kode

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

typedef unsigned char uchar;
typedef unsigned int uint;

#define ADC_CS    0
#define ADC_CLK   1
#define ADC_DIO   2
#define BtnPin    3

uchar get_ADC_Result(uint channel)
{
    uchar i;
    uchar dat1=0, dat2=0;
    int sel = channel > 1 & 1;
    int odd = channel & 1;
    pinMode(ADC_DIO, OUTPUT);
    digitalWrite(ADC_CS, 0);
    // Start bit
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    //Single End mode
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
```

```

// ODD
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,odd);  delayMicroseconds(2);
digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
//Select
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,sel);  delayMicroseconds(2);
digitalWrite(ADC_CLK,1);
digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
digitalWrite(ADC_CLK,0);
digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
for(i=0;i<8;i++)
{
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);    delayMicroseconds(2);
    pinMode(ADC_DIO, INPUT);
    dat1=dat1<<1 | digitalRead(ADC_DIO);
}
for(i=0;i<8;i++)
{
    dat2 = dat2 | ((uchar)(digitalRead(ADC_DIO))<<i);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);    delayMicroseconds(2);
}
digitalWrite(ADC_CS,1);
pinMode(ADC_DIO, OUTPUT);
return(dat1==dat2) ? dat1 : 0;
}
int main(void)
{
    uchar x_val;
    uchar y_val;
    uchar btn_val;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("setup wiringPi failed !");
        return 1;
}

```

```

pinMode(BtnPin, INPUT);
pullUpDnControl(BtnPin, PUD_UP);
pinMode(ADC_CS, OUTPUT);
pinMode(ADC_CLK, OUTPUT);

while(1){
    x_val = get_ADC_Result(0);
    y_val = get_ADC_Result(1);
    btn_val = digitalRead(BtnPin);
    printf("x = %d, y = %d, btn = %d\n", x_val, y_val, btn_val);
    delay(100);
}
return 0;
}

```

Kode Erklärung

```

uchar get_ADC_Result(uint channel)
{
    uchar i;
    uchar dat1=0, dat2=0;
    int sel = channel > 1 & 1;
    int odd = channel & 1;
    pinMode(ADC_DIO, OUTPUT);
    digitalWrite(ADC_CS, 0);
    // Start bit
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);

    //Single End mode
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    .....
}

```

Der Arbeitsprozess der Funktion ist in 2.1.4 Potentiometer beschrieben.

```

while(1){
    x_val = get_ADC_Result(0);
    y_val = get_ADC_Result(1);
}

```

```
btn_val = digitalRead(BtnPin);
printf("x = %d, y = %d, btn = %d\n", x_val, y_val, btn_val);
delay(100);
}
```

VRX und VRY des Joysticks sind mit CH0 bzw. CH1 des ADC0834 verbunden. Daher wird die Funktion getResult() aufgerufen, um die Werte von CH0 und CH1 zu lesen. Dann sollten die gelesenen Werte in den Variablen x_val und y_val gespeichert werden. Lesen Sie außerdem den Wert von SW des Joysticks und speichern Sie ihn in der Variablen Btn_val. Schließlich sollen die Werte von x_val, y_val und Btn_val mit der Funktion print() gedruckt werden.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Ausführen.

```
sudo python3 2.1.6_Joystick.py
```

Nachdem die Kode ausgeführt wurde, drehen Sie den Joystick, und die entsprechenden Werte von x, y, Btn werden auf dem Bildschirm angezeigt.

Kode

```
#!/usr/bin/env python3

import RPi.GPIO as GPIO
import ADC0834
import time

BtnPin = 22

def setup():
    # Set the GPIO modes to BCM Numbering
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(BtnPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    ADC0834.setup()

def destroy():
    # Release resource
    GPIO.cleanup()
```

```

def loop():
    while True:
        x_val = ADC0834.getResult(0)
        y_val = ADC0834.getResult(1)
        Btn_val = GPIO.input(BtnPin)
        print ('X: %d  Y: %d  Btn: %d' % (x_val, y_val, Btn_val))
        time.sleep(0.2)

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

```

Kode Erklärung

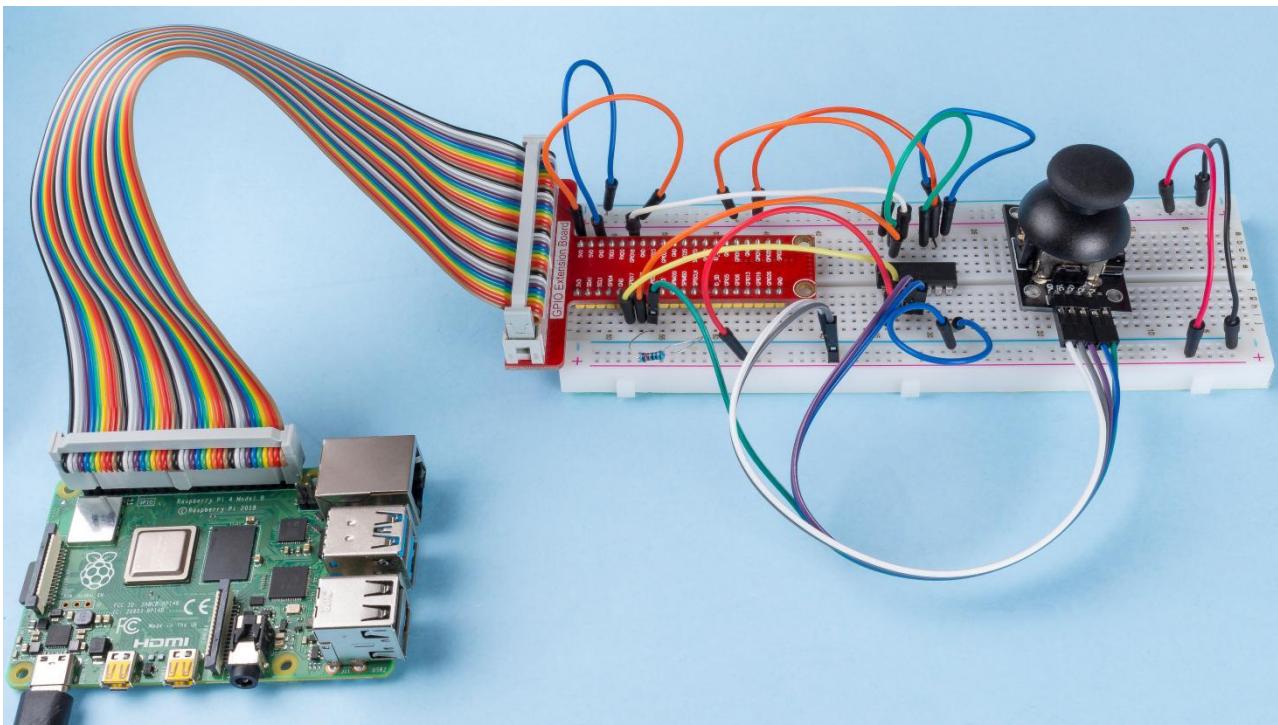
```

def loop():
    while True:
        x_val = ADC0834.getResult(0)
        y_val = ADC0834.getResult(1)
        Btn_val = GPIO.input(BtnPin)
        print ('X: %d  Y: %d  Btn: %d' % (x_val, y_val, Btn_val))
        time.sleep(0.2)

```

VRX und VRY des Joysticks sind mit CH0 bzw. CH1 des ADC0834 verbunden. Daher wird die Funktion getResult() aufgerufen, um die Werte von CH0 und CH1 zu lesen. Dann sollten die gelesenen Werte in den Variablen x_val und y_val gespeichert werden. Lesen Sie außerdem den Wert von SW des Joysticks und speichern Sie ihn in der Variablen Btn_val. Schließlich sollen die Werte von x_val, y_val und Btn_val mit der Funktion print() gedruckt werden.

Phänomen Bild



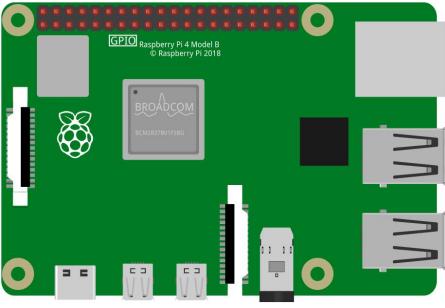
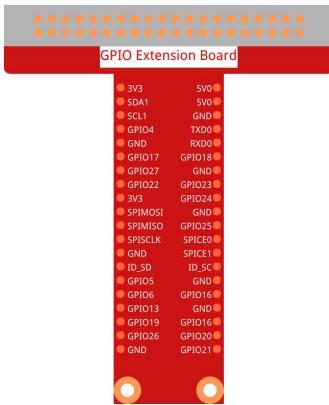
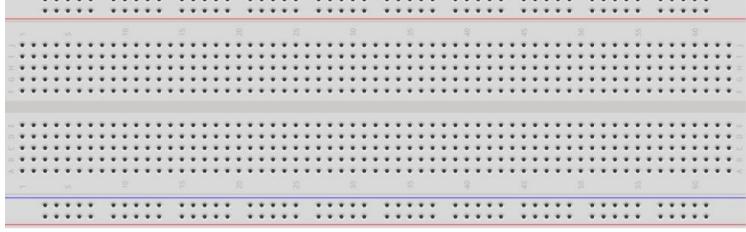
2.2 Sensoren

2.2.1 Fotowiderstand

Einführung

Der Fotowiderstand ist eine häufig verwendete Komponente der Umgebungslichtintensität im Leben. Es hilft dem Controller, Tag und Nacht zu erkennen und Lichtsteuerungsfunktionen wie Nachtlampen zu realisieren. Dieses Projekt ist dem Potentiometer sehr ähnlich, und Sie könnten denken, es ändert die Spannung, um Licht zu erfassen.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Fotowiderstand
	 GPIO Extension Board Pinout: 3V3 SDA1 5V0 SCL1 GND GPIO4 TXD0 GND RXD0 GPIO17 GPIO18 GPIO27 GND GPIO22 GPIO23 3V3 GND SPI-MOSI GND SPI-MISO GPIO25 SPI-SCLK SPICE0 GND SPICE1 TxD GND GPIO5 GND GPIO6 GPIO16 GPIO13 GND GPIO19 GPIO16 GPIO26 GPIO20 GND GPIO21	
1 * 40-Pin Kabel		1 * ADC0834
		
1 * Steckbrett		1 * LED
		
		Mehrere Überbrückungsdrähte
		
		1 * Widerstand (220 Ω)
		
		1 * Widerstand 10 k Ω
		

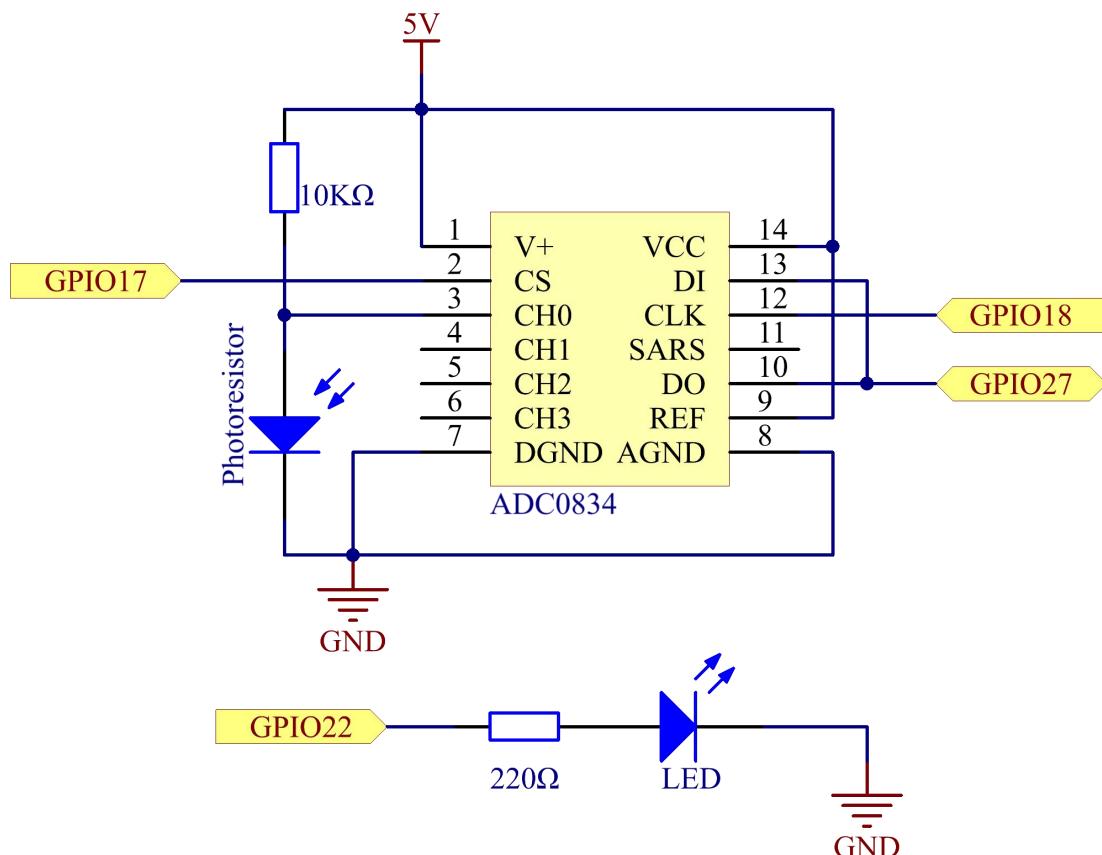
Prinzip

Ein Fotowiderstand oder eine Fotozelle ist ein lichtgesteuerter variabler Widerstand. Der Widerstand eines Fotowiderstands nimmt mit zunehmender Intensität des einfallenden Lichts ab; mit anderen Worten, es zeigt Fotoleitfähigkeit. Ein Fotowiderstand kann in lichtempfindlichen Detektorschaltungen sowie in licht- und dunkelheitsaktivierten Schaltkreisen eingesetzt werden.



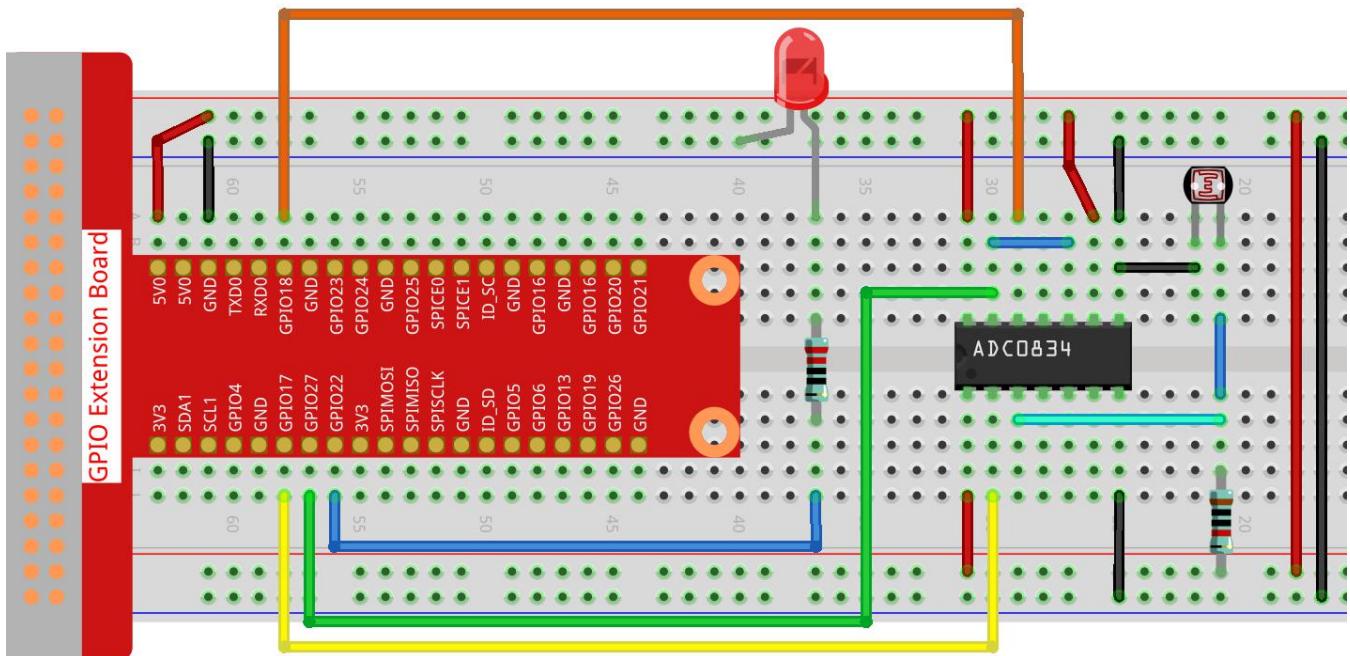
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27
GPIO22	Pin14	3	22



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.2.1/
```

Schritt 3: Kompilieren Sie der Kode

```
gcc 2.2.1_Photoresistor.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Wenn die Kode ausgeführt wird, variiert die Helligkeit der LED in Abhängigkeit von der Lichtintensität, die der Fotowiderstand erfasst.

Kode

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

typedef unsigned char uchar;
typedef unsigned int uint;

#define ADC_CS 0
#define ADC_CLK 1
```

```
#define      ADC_DIO    2
#define      LedPin     3

uchar get_ADC_Result(uint channel)
{
    uchar i;
    uchar dat1=0, dat2=0;
    int sel = channel > 1 & 1;
    int odd = channel & 1;

    pinMode(ADC_DIO, OUTPUT);
    digitalWrite(ADC_CS, 0);
    // Start bit
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);

    //Single End mode
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);

    // ODD
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,odd);  delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);

    //Select
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,sel);  delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);

    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);    delayMicroseconds(2);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);

    for(i=0;i<8;i++)
    {
        digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
        digitalWrite(ADC_CLK,0);    delayMicroseconds(2);

        pinMode(ADC_DIO, INPUT);
        dat1=dat1<<1 | digitalRead(ADC_DIO);
    }
}
```

```

}

for(i=0;i<8;i++)
{
    dat2 = dat2 | ((uchar)(digitalRead(ADC_DIO))<<i);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);    delayMicroseconds(2);
}

digitalWrite(ADC_CS,1);
pinMode(ADC_DIO, OUTPUT);
return(dat1==dat2) ? dat1 : 0;
}

int main(void)
{
    uchar analogVal;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    softPwmCreate(LedPin, 0, 100);
    pinMode(ADC_CS, OUTPUT);
    pinMode(ADC_CLK, OUTPUT);

    while(1){
        analogVal = get_ADC_Result(0);
        printf("Current analogVal : %d\n", analogVal);
        softPwmWrite(LedPin, analogVal);
        delay(100);
    }
    return 0;
}

```

Kode Erklärung

Die Koden hier sind die gleichen wie in 2.1.4 Potentiometer. Wenn Sie weitere Fragen haben, lesen Sie bitte die Kode-Erklärung von 2.1.4 Potentiometer.c für Details.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 2.2.1_Photoresistor.py
```

Wenn die Kode ausgeführt wird, variiert die Helligkeit der LED in Abhängigkeit von der Lichtintensität, die der Fotowiderstand erfasst.

Kode

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import ADC0834
import time
LedPin = 22
def setup():
    global led_val
    # Set the GPIO modes to BCM Numbering
    GPIO.setmode(GPIO.BCM)
    # Set all LedPin's mode to output and initial level to High(3.3v)
    GPIO.setup(LedPin, GPIO.OUT, initial=GPIO.HIGH)
    ADC0834.setup()
    # Set led as pwm channel and freuece to 2KHz
    led_val = GPIO.PWM(LedPin, 2000)
    # Set all begin with value 0
    led_val.start(0)
def destroy():
    # Stop all pwm channel
    led_val.stop()
    # Release resource
    GPIO.cleanup()
def loop():
    while True:
        analogVal = ADC0834.getResult()
        print ('analog value = %d' % analogVal)
        led_val.ChangeDutyCycle(analogVal*100/255)
        time.sleep(0.2)
if __name__ == '__main__':
    setup()
```

```

try:
    loop()
except KeyboardInterrupt: # When 'Ctrl+C' is pressed, the program destroy() will be
executed.
    destroy()

```

Kode Explanation

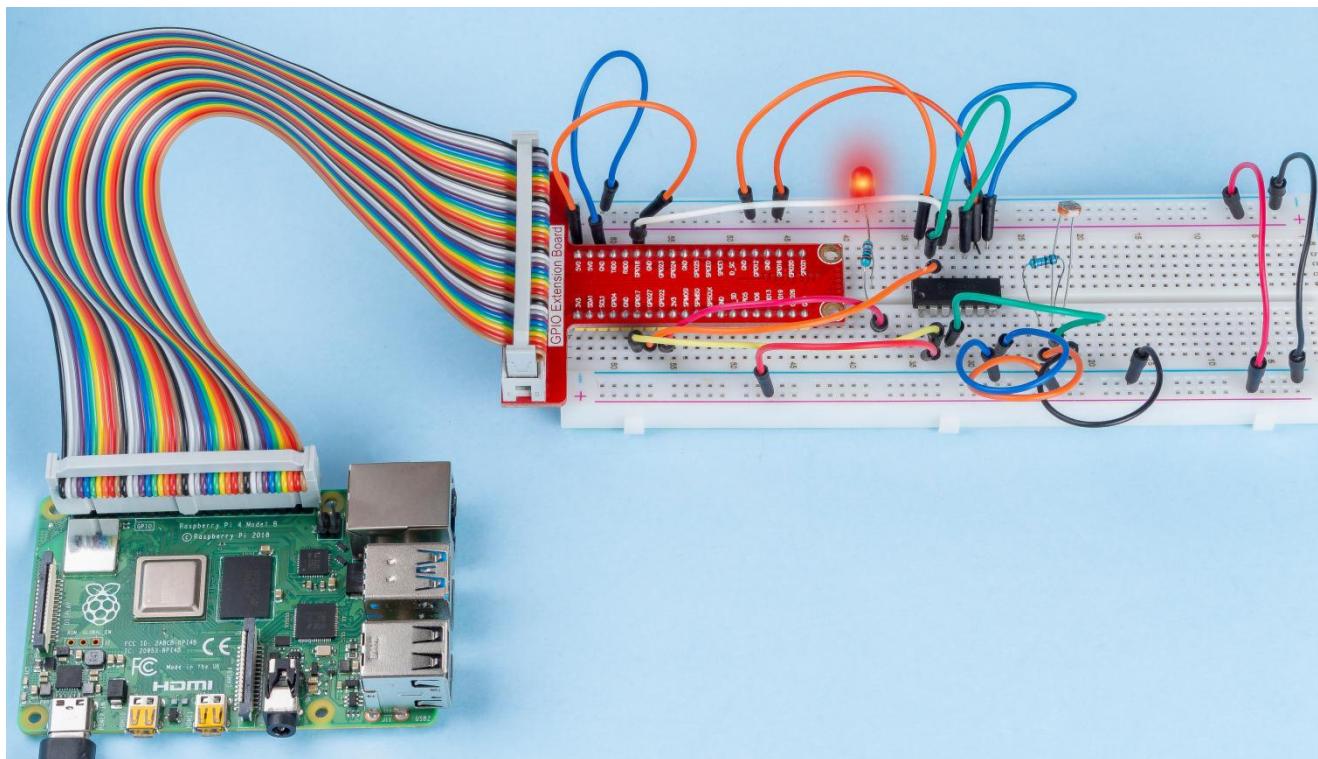
```

def loop():
    while True:
        analogVal = ADC0834.getResult()
        print ('analog value = %d' % analogVal)
        led_val.ChangeDutyCycle(analogVal*100/255)
        time.sleep(0.2)

```

Lesen Sie den Analogwert von CH0 von ADC0834 ab. Standardmäßig wird mit der Funktion getResult () der Wert von CH0 gelesen. Wenn Sie also andere Kanäle lesen möchten, geben Sie bitte 1, 2 oder 3 in () der Funktion getResult () ein. Als nächstes müssen Sie den Wert über die Druckfunktion drucken. Da das sich ändernde Element das Tastverhältnis von LedPin ist, wird die Berechnungsformel $\text{analogVal} * 100/255$ benötigt, um analogVal in Prozent umzuwandeln. Schließlich wird ChangeDutyCycle () aufgerufen, um den Prozentsatz in LedPin zu schreiben.

Phänomen Bild

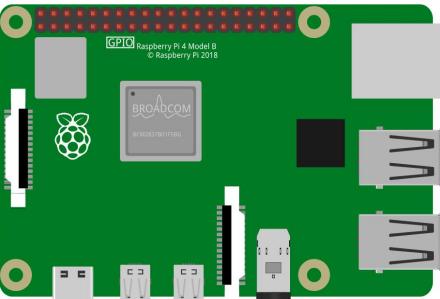
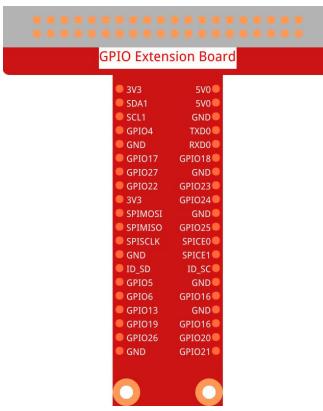


2.2.2 Thermistor

Einführung

Genau wie der Fotowiderstand Licht erfassen kann, ist der Thermistor ein temperaturempfindliches elektronisches Gerät, mit dem Funktionen der Temperaturregelung wie z. B. ein Wärmealarm realisiert werden können.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Thermistor																																								
	 GPIO Extension Board <table border="1"><tr><td>3V3</td><td>5V0</td></tr><tr><td>SDA1</td><td>GND</td></tr><tr><td>SCL1</td><td></td></tr><tr><td>GPIO4</td><td>TXD0</td></tr><tr><td>GND</td><td>RXD0</td></tr><tr><td>GPIO17</td><td>GPIO18</td></tr><tr><td>GPIO27</td><td>GND</td></tr><tr><td>GPIO22</td><td>GPIO23</td></tr><tr><td>3V</td><td>GPIO24</td></tr><tr><td>SPI_MISO</td><td>GND</td></tr><tr><td>SPI_MOSI</td><td>GPIO25</td></tr><tr><td>SPI_SCLK</td><td>SPICEO</td></tr><tr><td>GND</td><td>SPICE1</td></tr><tr><td>ID_SD</td><td>ID_SC</td></tr><tr><td>GPIO5</td><td>GND</td></tr><tr><td>GPIO6</td><td>GPIO16</td></tr><tr><td>GPIO13</td><td>GND</td></tr><tr><td>GPIO19</td><td>GPIO16</td></tr><tr><td>GPIO26</td><td>GPIO20</td></tr><tr><td>GND</td><td>GPIO21</td></tr></table>	3V3	5V0	SDA1	GND	SCL1		GPIO4	TXD0	GND	RXD0	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V	GPIO24	SPI_MISO	GND	SPI_MOSI	GPIO25	SPI_SCLK	SPICEO	GND	SPICE1	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	
3V3	5V0																																									
SDA1	GND																																									
SCL1																																										
GPIO4	TXD0																																									
GND	RXD0																																									
GPIO17	GPIO18																																									
GPIO27	GND																																									
GPIO22	GPIO23																																									
3V	GPIO24																																									
SPI_MISO	GND																																									
SPI_MOSI	GPIO25																																									
SPI_SCLK	SPICEO																																									
GND	SPICE1																																									
ID_SD	ID_SC																																									
GPIO5	GND																																									
GPIO6	GPIO16																																									
GPIO13	GND																																									
GPIO19	GPIO16																																									
GPIO26	GPIO20																																									
GND	GPIO21																																									
1 * ADC0834																																										
1 * 40-Pin Kabel	Mehrere Überbrückungsdrähte																																									
1 * Steckbrett		1 * Widerstand 10 kΩ																																								
																																										

Prinzip

Ein Thermistor ist ein wärmeempfindlicher Widerstand, der eine präzise und vorhersagbare Widerstandsänderung proportional zu kleinen Temperaturänderungen aufweist. Wie sehr sich sein Widerstand ändert, hängt von seiner einzigartigen Zusammensetzung ab. Thermistoren sind Teile einer größeren Gruppe passiver Komponenten. Und im Gegensatz zu ihren Gegenstücken mit aktiven Komponenten können passive Geräte keine Leistungsverstärkung oder Verstärkung für eine Schaltung bereitstellen.

Der Thermistor ist ein empfindliches Element und es gibt zwei Arten: den negativen Temperaturkoeffizienten (NTC) und den positiven Temperaturkoeffizienten (PTC), auch bekannt als NTC und PTC. Sein Widerstand variiert erheblich mit der Temperatur. Der Widerstand des PTC-Thermistors steigt mit der Temperatur, während der Zustand des NTC dem ersten entgegengesetzt ist. In diesem Experiment verwenden wir NTC.



Das Prinzip ist, dass sich der Widerstand des NTC-Thermistors mit der Temperatur der äußeren Umgebung ändert. Es erfasst die Echtzeit-Temperatur der Umgebung. Wenn die Temperatur höher wird, nimmt der Widerstand des Thermistors ab. Anschließend werden die Spannungsdaten vom A / D-Adapter in digitale Größen umgewandelt. Die Temperatur in Celsius oder Fahrenheit wird über die Programmierung ausgegeben.

In diesem Experiment werden ein Thermistor und ein 10k-Pull-up-Widerstand verwendet. Jeder Thermistor hat einen normalen Widerstand. Hier sind es 10 kOhm, die unter 25 Grad Celsius gemessen werden.

Hier ist die Beziehung zwischen dem Widerstand und der Temperatur:

$$R_T = R_N \exp^{B(1/T_K - 1/T_N)}$$

R_T ist der Widerstand des NTC-Thermistors bei einer Temperatur von T_K .

R_N ist der Widerstand des NTC-Thermistors unter der Nenntemperatur T_N . Hier beträgt der numerische Wert von R_N 10k.

T_K ist eine Kelvin-Temperatur und die Einheit ist K. Hier beträgt der numerische Wert von T_K 273.15 + Grad Celsius.

T_N ist eine Kelvin-Nenntemperatur; Das Gerät ist auch K. Hier beträgt der numerische Wert von T_N 273.15+25.

Und \mathbf{B} (beta), die Materialkonstante des NTC-Thermistors, wird auch als Wärmeempfindlichkeitsindex mit einem numerischen Wert von 3950 bezeichnet.

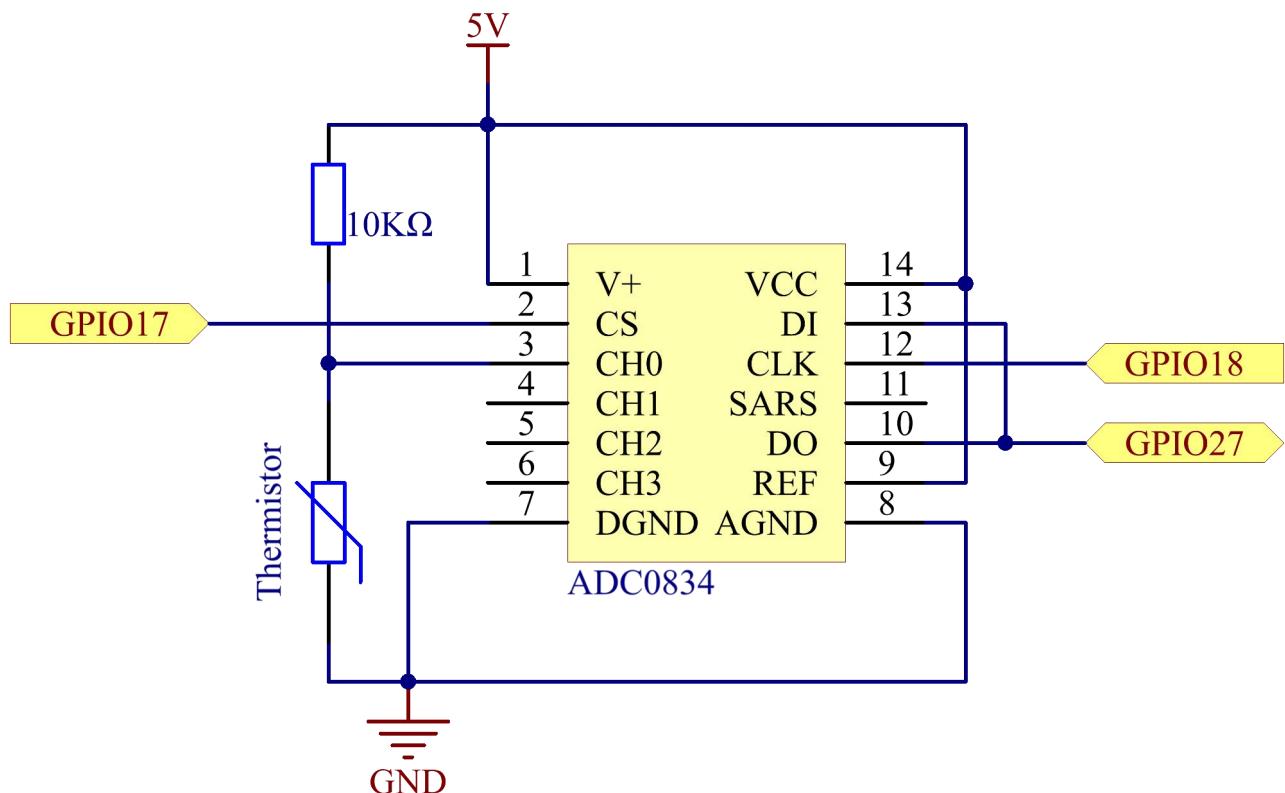
exp ist die Abkürzung für exponentiell, und die Basisnummer ist eine natürliche Nummer und entspricht ungefähr 2,7.

Konvertieren Sie diese Formel $T_K=1/(\ln(R_T/R_N)/B+1/T_N)$, um eine Kelvin-Temperatur zu erhalten, die minus 273,15 Grad Celsius entspricht.

Diese Beziehung ist eine empirische Formel. Sie ist nur dann genau, wenn Temperatur und Widerstand im effektiven Bereich liegen.

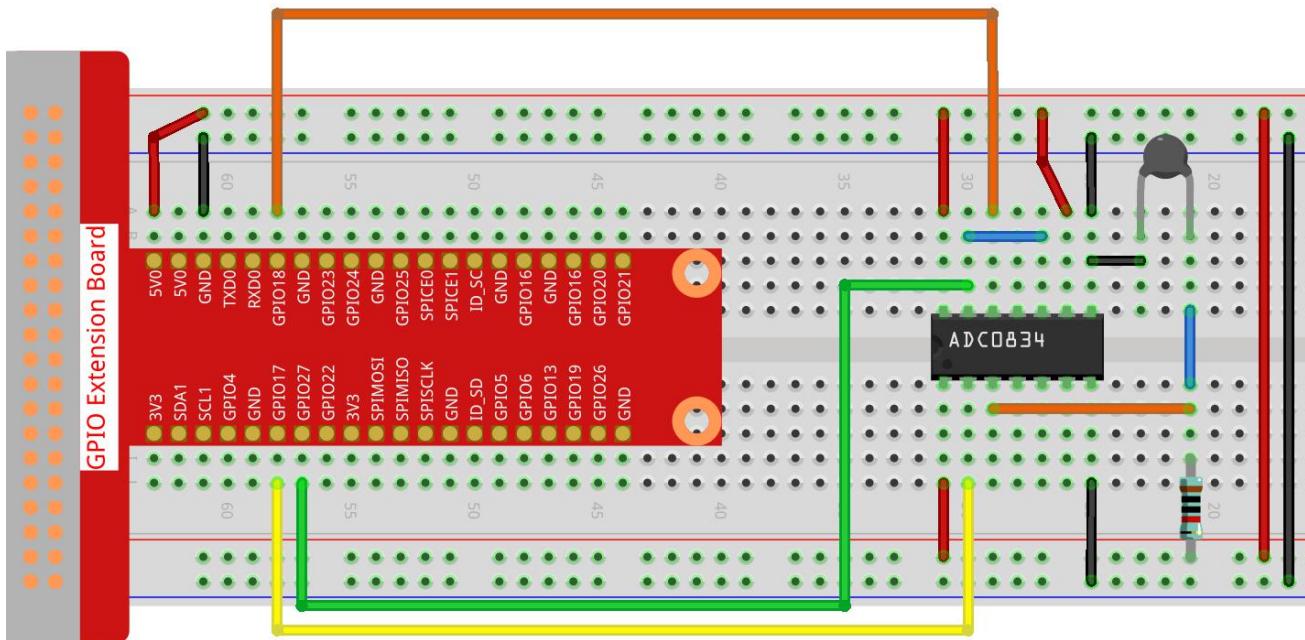
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.2.2/
```

Schritt 3: Kompilieren Sie die Kode.

```
gcc 2.2.2_Thermistor.c -lwiringPi -lm
```

Hinweis: **-lm** dient zum Laden der Bibliotheksmathematik. Nicht weglassen, sonst wird ein Fehler gemacht.

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Während die Kode ausgeführt wird, erkennt der Thermistor die Umgebungstemperatur, die nach Abschluss der Programmberchnung auf dem Bildschirm angezeigt wird.

Kode

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

typedef unsigned char uchar;
```

```

typedef unsigned int uint;

#define      ADC_CS      0
#define      ADC_CLK     1
#define      ADC_DIO     2

uchar get_ADC_Result(uint channel)
{
    uchar i;
    uchar dat1=0, dat2=0;
    int sel = channel > 1 & 1;
    int odd = channel & 1;

    pinMode(ADC_DIO, OUTPUT);
    digitalWrite(ADC_CS, 0);
    // Start bit
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);

    //Single End mode
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);

    // ODD
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,odd);  delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);    delayMicroseconds(2);

    //Select
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,sel);  delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);

    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);

    for(i=0;i<8;i++)
    {
        digitalWrite(ADC_CLK,1);    delayMicroseconds(2);
        digitalWrite(ADC_CLK,0);    delayMicroseconds(2);
    }
}

```

```

pinMode(ADC_DIO, INPUT);
dat1=dat1<<1 | digitalRead(ADC_DIO);
}

for(i=0;i<8;i++)
{
    dat2 = dat2 | ((uchar)(digitalRead(ADC_DIO))<<i);
    digitalWrite(ADC_CLK,1);      delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);      delayMicroseconds(2);
}

digitalWrite(ADC_CS,1);
pinMode(ADC_DIO, OUTPUT);
return(dat1==dat2) ? dat1 : 0;
}

int main(void)
{
    unsigned char analogVal;
    double Vr, Rt, temp, cel, Fah;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    pinMode(ADC_CS,  OUTPUT);
    pinMode(ADC_CLK, OUTPUT);

    while(1){
        analogVal = get_ADC_Result(0);
        Vr = 5 * (double)(analogVal) / 255;
        Rt = 10000 * (double)(Vr) / (5 - (double)(Vr));
        temp = 1 / (((log(Rt/10000)) / 3950)+(1 / (273.15 + 25)));
        cel = temp - 273.15;
        Fah = cel * 1.8 +32;
        printf("Celsius: %.2f C  Fahrenheit: %.2f F\n", cel, Fah);
        delay(100);
    }
    return 0;
}

```

Kode Erklärung

```
#include <math.h>
```

Es gibt eine C-Numerik-Bibliothek, die eine Reihe von Funktionen deklariert, um allgemeine mathematische Operationen und Transformationen zu berechnen.

```
analogVal = get_ADC_Result(0);
```

Mit dieser Funktion wird der Wert des Thermistors abgelesen.

```
Vr = 5 * (double)(analogVal) / 255;
Rt = 10000 * (double)(Vr) / (5 - (double)(Vr));
temp = 1 / (((log(Rt/10000)) / 3950)+(1 / (273.15 + 25)));
cel = temp - 273.15;
Fah = cel * 1.8 +32;
printf("Celsius: %.2f C  Fahrenheit: %.2f F\n", cel, Fah);
```

Diese Berechnungen wandeln die Thermistorwerte in Celsiuswerte um.

```
Vr = 5 * (double)(analogVal) / 255;
Rt = 10000 * (double)(Vr) / (5 - (double)(Vr));
```

Diese beiden Kodezeilen berechnen die Spannungsverteilung mit dem analogen Lesewert, um Rt (Widerstand des Thermistors) zu erhalten.

```
temp = 1 / (((log(Rt/10000)) / 3950)+(1 / (273.15 + 25)));
```

Diese Kode bezieht sich auf das Einsticken von Rt in die Formel $T_K=1/(\ln(R_T/R_N)/B+1/T_N)$, um die Kelvin-Temperatur zu erhalten.

```
temp = temp - 273.15;
```

Wandeln Sie die Kelvin-Temperatur in Grad Celsius um.

```
Fah = cel * 1.8 +32;
```

Konvertieren Sie Grad Celsius in Fahrenheit.

```
printf("Celsius: %.2f C  Fahrenheit: %.2f F\n", cel, Fah);
```

Drucken Sie Celsius, Fahrenheit und ihre Einheiten auf dem Display.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus

```
sudo python3 2.2.2_Thermistor.py
```

Während die Kode ausgeführt wird, erkennt der Thermistor die Umgebungstemperatur, die nach Abschluss der Programmberchnung auf dem Bildschirm angezeigt wird.

Kode

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import RPi.GPIO as GPIO
import ADC0834
import time
import math

def init():
    ADC0834.setup()

def loop():
    while True:
        analogVal = ADC0834.getResult()
        Vr = 5 * float(analogVal) / 255
        Rt = 10000 * Vr / (5 - Vr)
        temp = 1/(((math.log(Rt / 10000)) / 3950) + (1 / (273.15+25)))
        Cel = temp - 273.15
        Fah = Cel * 1.8 + 32
        print ('Celsius: %.2f °C  Fahrenheit: %.2f °F' % (Cel, Fah))
        time.sleep(0.2)

if __name__ == '__main__':
    init()
    try:
        loop()
    except KeyboardInterrupt:
        ADC0834.destroy()
```

Kode Erklärung

```
import math
```

Es gibt eine numerische Bibliothek, die eine Reihe von Funktionen deklariert, um allgemeine mathematische Operationen und Transformationen zu berechnen.

```
analogVal = ADC0834.getResult()
```

Mit dieser Funktion wird der Wert des Thermistors abgelesen.

```
Vr = 5 * float(analogVal) / 255  
Rt = 10000 * Vr / (5 - Vr)  
temp = 1/(((math.log(Rt / 10000)) / 3950) + (1 / (273.15+25)))  
Cel = temp - 273.15  
Fah = Cel * 1.8 + 32  
print ('Celsius: %.2f °C  Fahrenheit: %.2f °F' % (Cel, Fah))
```

Diese Berechnungen wandeln die Thermistorwerte in Grad Celsius und Fahrenheit um.

```
Vr = 5 * float(analogVal) / 255  
Rt = 10000 * Vr / (5 - Vr)
```

Diese beiden Kodezeilen berechnen die Spannungsverteilung mit dem analogen Lesewert, um Rt (Widerstand des Thermistors) zu erhalten.

```
temp = 1/(((math.log(Rt / 10000)) / 3950) + (1 / (273.15+25)))
```

Diese Kode bezieht sich auf das Einsticken von Rt in die Formel $T_K=1/(\ln(R_T/R_N)/B+1/T_N)$, um die Kelvin-Temperatur zu erhalten.

```
temp = temp - 273.15
```

Wandeln Sie die Kelvin-Temperatur in Grad Celsius um.

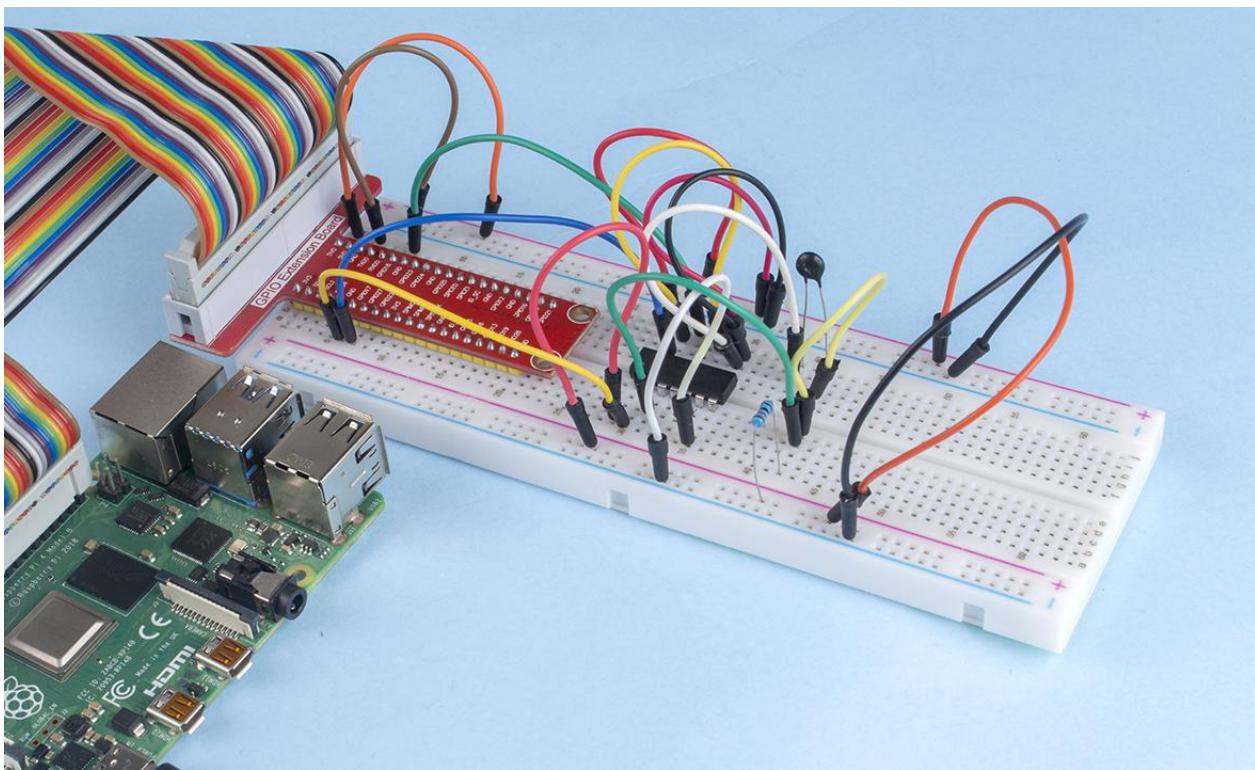
```
Fah = Cel * 1.8 + 32
```

Konvertieren Sie den Celsius-Grad in Fahrenheit-Grad.

```
print ('Celsius: %.2f °C  Fahrenheit: %.2f °F' % (Cel, Fah))
```

Drucken Sie Celsius, Fahrenheit und ihre Einheiten auf der Anzeige.

Phänomen Bild



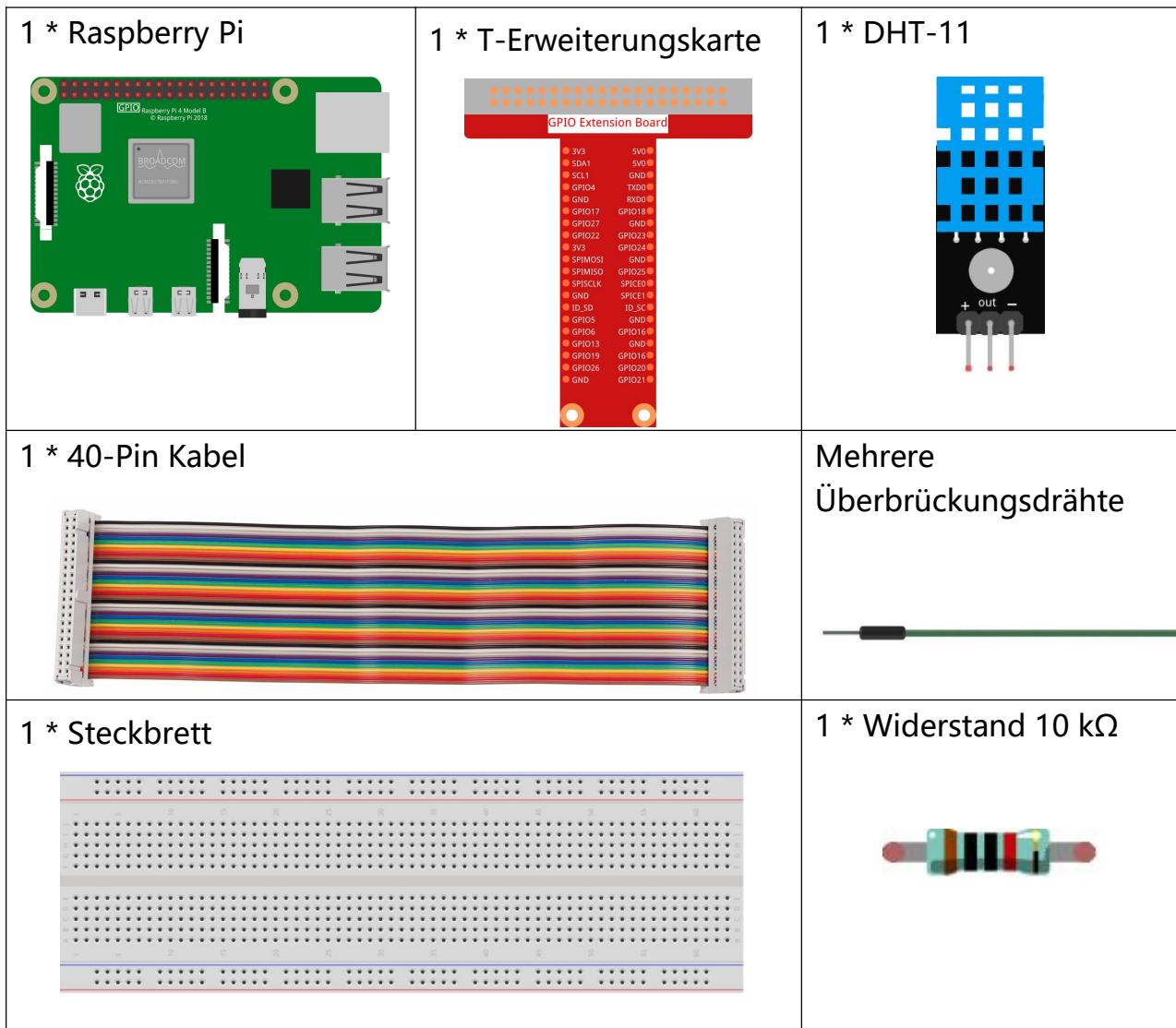
2.2.3 DHT-11

Einführung

Der digitale Temperatur- und Feuchtigkeitssensor DHT11 ist ein Verbundsensor, der einen kalibrierten digitalen Signalausgang für Temperatur und Luftfeuchtigkeit enthält. Die Technologie einer speziellen Sammlung digitaler Module und die Technologie der Temperatur- und Feuchtigkeitsmessung werden angewendet, um sicherzustellen, dass das Produkt eine hohe Zuverlässigkeit und ausgezeichnete Stabilität aufweist.

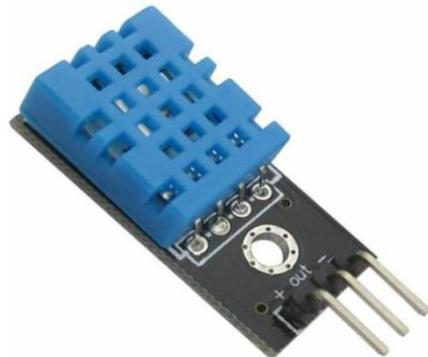
Die Sensoren umfassen einen Nasselement-Widerstandssensor und einen NTC-Temperatursensor und sind mit einem Hochleistungs-8-Bit-Mikrocontroller verbunden.

Komponenten



Prinzip

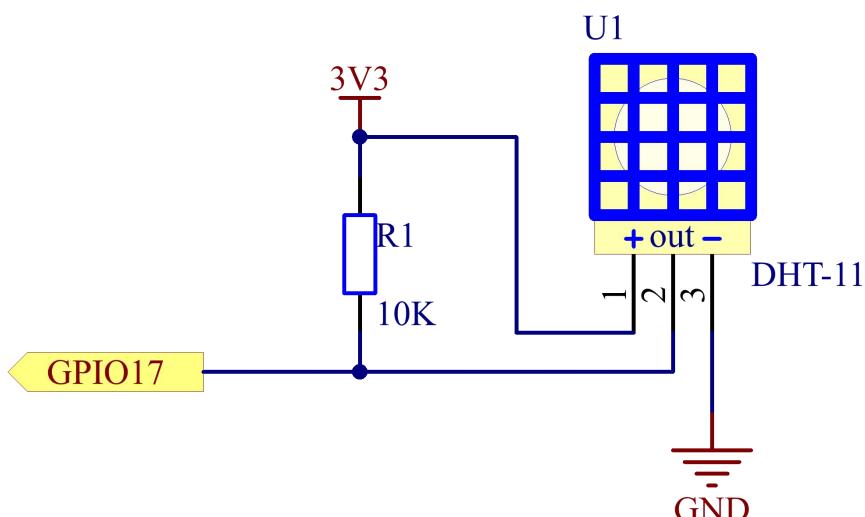
Der DHT11 ist ein grundlegender, äußerst kostengünstiger digitaler Temperatur- und Feuchtigkeitssensor. Es verwendet einen kapazitiven Feuchtigkeitssensor und einen Thermistor, um die Umgebungsluft zu messen, und spuckt ein digitales Signal auf den Datenstift aus (es werden keine analogen Eingangsstifte benötigt).



Es sind nur drei Pins verfügbar: VCC, GND und DATA. Der Kommunikationsprozess beginnt damit, dass die DATA-Leitung Startsignale an DHT11 sendet, und DHT11 empfängt die Signale und gibt ein Antwortsignal zurück. Dann empfängt der Host das Antwortsignal und beginnt mit dem Empfang von 40-Bit-Feuchtigkeitsdaten (8-Bit-Feuchtigkeits-Ganzzahl + 8-Bit-Feuchtigkeits-Dezimalzahl + 8-Bit-Temperatur-Ganzzahl + 8-Bit-Temperatur-Dezimalzahl + 8-Bit-Prüfsumme). Weitere Informationen finden Sie im DHT11-Datenblatt.

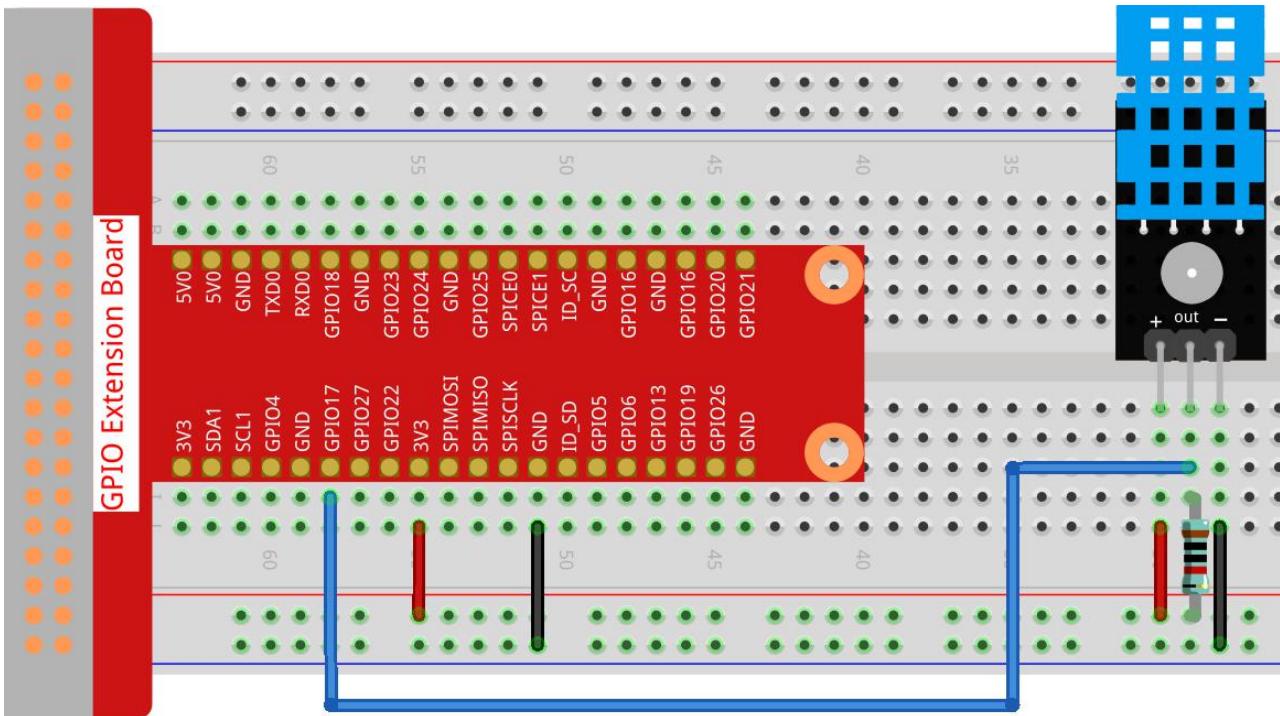
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.2.3/
```

Schritt 3: Kompilieren Sie die Kode.

```
gcc 2.2.3_DHT.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Nachdem die Kode ausgeführt wurde, druckt das Programm die von DHT11 erfasste Temperatur und Luftfeuchtigkeit auf dem Computerbildschirm.

Kode

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

#define maxTim 85
#define dhtPin 0
```

```

int dht11_dat[5] = {0,0,0,0,0};

void readDht11() {
    uint8_t laststate = HIGH;
    uint8_t counter = 0;
    uint8_t j = 0, i;
    float Fah; // fahrenheit
    dht11_dat[0] = dht11_dat[1] = dht11_dat[2] = dht11_dat[3] = dht11_dat[4] = 0;
    // pull pin down for 18 milliseconds
    pinMode(dhtPin, OUTPUT);
    digitalWrite(dhtPin, LOW);
    delay(18);
    // then pull it up for 40 microseconds
    digitalWrite(dhtPin, HIGH);
    delayMicroseconds(40);
    // prepare to read the pin
    pinMode(dhtPin, INPUT);

    // detect change and read data
    for (i=0; i< maxTim; i++) {
        counter = 0;
        while (digitalRead(dhtPin) == laststate) {
            counter++;
            delayMicroseconds(1);
            if (counter == 255) {
                break;
            }
        }
        laststate = digitalRead(dhtPin);

        if (counter == 255) break;
        // ignore first 3 transitions
        if ((i >= 4) && (i%2 == 0)) {
            // shove each bit into the storage bytes
            dht11_dat[j/8] <<= 1;
            if (counter > 50)
                dht11_dat[j/8] |= 1;
            j++;
        }
    }
}

```

```

// check we read 40 bits (8bit x 5 ) + verify checksum in the last byte
// print it out if data is good
if ((j >= 40) &&
    (dht11_dat[4] == ((dht11_dat[0] + dht11_dat[1] + dht11_dat[2] + dht11_dat[3])
& 0xFF)) ) {
    Fah = dht11_dat[2] * 9. / 5. + 32;
    printf("Humidity = %d.%d %% Temperature = %d.%d *C (%.1f *F)\n",
           dht11_dat[0], dht11_dat[1], dht11_dat[2], dht11_dat[3], Fah);
}
}

int main (void) {
    if(wiringPiSetup() == -1){ //when initialize wiring failed, print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    while (1) {
        readDht11();
        delay(500); // wait 1sec to refresh
    }
    return 0 ;
}

```

Kode Erklärung

```

void readDht11() {
    uint8_t laststate = HIGH;
    uint8_t counter = 0;
    uint8_t j = 0, i;
    float Fah; // fahrenheit
    dht11_dat[0] = dht11_dat[1] = dht11_dat[2] = dht11_dat[3] = dht11_dat[4] = 0;
    // ...
}

```

Diese Funktion wird verwendet, um die Funktion von DHT11 zu realisieren.

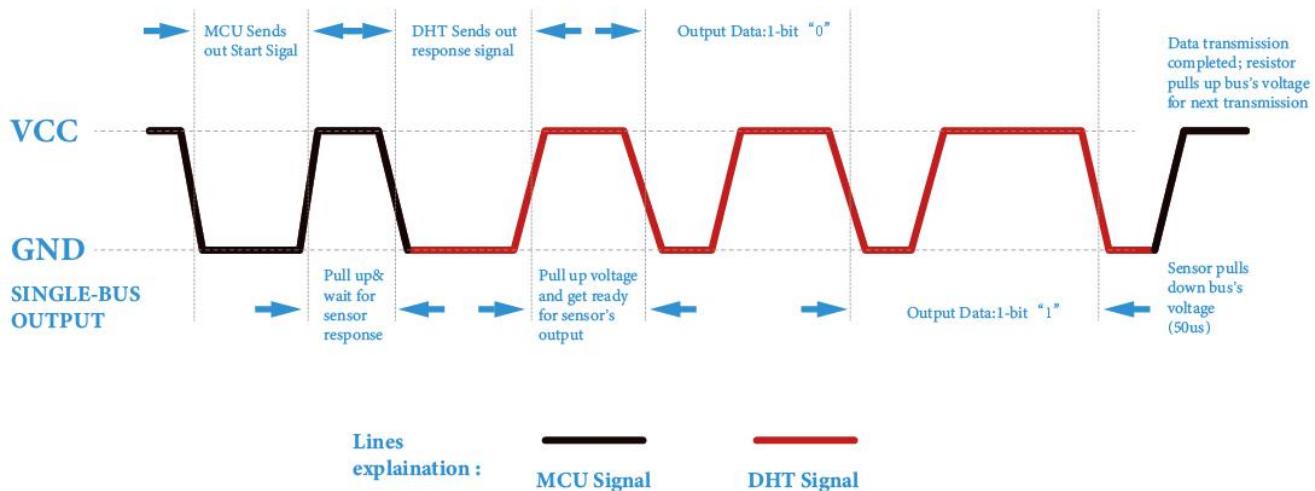
Es kann im Allgemeinen in 3 Teile unterteilt werden:

1. Bereiten Sie sich darauf vor, den Stift zu lesen:

```
// pull pin down for 18 milliseconds
```

```
pinMode(dhtPin, OUTPUT);
digitalWrite(dhtPin, LOW);
delay(18);
// then pull it up for 40 microseconds
digitalWrite(dhtPin, HIGH);
delayMicroseconds(40);
// prepare to read the pin
pinMode(dhtPin, INPUT);
```

Sein Kommunikationsfluss wird durch den Arbeitszeitpunkt bestimmt.



Wenn DHT11 startet, sendet die MCU ein Signal mit niedrigem Niveau und hält das Signal dann für 40us auf hohem Niveau. Danach beginnt die Erkennung des Zustands der externen Umgebung.

2. Daten lesen:

```
// detect change and read data
for ( i=0; i< maxTim; i++) {
    counter = 0;
    while (digitalRead(dhtPin) == laststate) {
        counter++;
        delayMicroseconds(1);
        if (counter == 255) {
            break;
        }
    }
    laststate = digitalRead(dhtPin);
    if (counter == 255) break;
    // ignore first 3 transitions
```

```

if ((i >= 4) && (i%2 == 0)) {
    // shove each bit into the storage bytes
    dht11_dat[j/8] <<= 1;
    if (counter > 50)
        dht11_dat[j/8] |= 1;
    j++;
}

```

Die Schleife speichert die erkannten Daten im Array dht11_dat []. DHT11 überträgt Daten von jeweils 40 Bit. Die ersten 16 Bits beziehen sich auf die Luftfeuchtigkeit, die mittleren 16 Bits auf die Temperatur und die letzten acht Bits werden zur Überprüfung verwendet. Das Datenformat ist:

8-Bit-Feuchtigkeits-Integer-Daten + 8-Bit-Feuchtigkeits-Dezimaldaten + 8-Bit-Temperatur-Integer-Daten + 8-Bit-Temperatur-Dezimaldaten + 8-Bit-Prüfbit.

3. Luftfeuchtigkeit und Temperatur drucken.

```

// check we read 40 bits (8bit x 5 ) + verify checksum in the last byte
// print it out if data is good
if ((j >= 40) &&
    (dht11_dat[4] == ((dht11_dat[0] + dht11_dat[1] + dht11_dat[2] + dht11_dat[3])
& 0xFF)) ) {
    Fah = dht11_dat[2] * 9. / 5. + 32;
    printf("Humidity = %d.%d %% Temperature = %d.%d *C (%.1f *F)\n",
           dht11_dat[0], dht11_dat[1], dht11_dat[2], dht11_dat[3], Fah);
}

```

Wenn der Datenspeicher bis zu 40 Bit beträgt, überprüfen Sie die Gültigkeit der Daten über das **Prüfbit (dht11_dat[4])** und drucken Sie dann Temperatur und Luftfeuchtigkeit aus.

Wenn die empfangenen Daten beispielsweise 00101011 (8-Bit-Wert der Feuchtigkeits-Ganzzahl) 00000000 (8-Bit-Wert der Feuchtigkeits-Dezimalzahl) 00111100 (8-Bit-Wert der Temperatur-Ganzzahl) 00000000 (8-Bit-Wert der Temperatur-Dezimalzahl) 01100111 (Bit prüfen)

Berechnung:

00101011+00000000+00111100+00000000=01100111.

Das Endergebnis ist gleich mit den Prüfbitdaten, dann sind die empfangenen Daten korrekt:

Luftfeuchtigkeit = 43% , Temperatur = 60 * C.

Wenn es nicht gleich mit den Prüfbitdaten ist, ist die Datenübertragung nicht normal und die Daten werden erneut empfangen.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 2.2.3_DHT.py
```

Nachdem die Kode ausgeführt wurde, druckt das Programm die von DHT11 erfasste Temperatur und Luftfeuchtigkeit auf dem Computerbildschirm.

Kode

```
import RPi.GPIO as GPIO
import time

dhtPin = 17

GPIO.setmode(GPIO.BCM)

MAX_UNCHANGE_COUNT = 100

STATE_INIT_PULL_DOWN = 1
STATE_INIT_PULL_UP = 2
STATE_DATA_FIRST_PULL_DOWN = 3
STATE_DATA_PULL_UP = 4
STATE_DATA_PULL_DOWN = 5

def readDht11():
    GPIO.setup(dhtPin, GPIO.OUT)
    GPIO.output(dhtPin, GPIO.HIGH)
    time.sleep(0.05)
    GPIO.output(dhtPin, GPIO.LOW)
    time.sleep(0.02)
    GPIO.setup(dhtPin, GPIO.IN, GPIO.PUD_UP)
```

```

unchanged_count = 0
last = -1
data = []
while True:
    current = GPIO.input(dhtPin)
    data.append(current)
    if last != current:
        unchanged_count = 0
        last = current
    else:
        unchanged_count += 1
        if unchanged_count > MAX_UNCHANGE_COUNT:
            break

state = STATE_INIT_PULL_DOWN

lengths = []
current_length = 0

for current in data:
    current_length += 1

    if state == STATE_INIT_PULL_DOWN:
        if current == GPIO.LOW:
            state = STATE_INIT_PULL_UP
        else:
            continue
    if state == STATE_INIT_PULL_UP:
        if current == GPIO.HIGH:
            state = STATE_DATA_FIRST_PULL_DOWN
        else:
            continue
    if state == STATE_DATA_FIRST_PULL_DOWN:
        if current == GPIO.LOW:
            state = STATE_DATA_PULL_UP
        else:
            continue
    if state == STATE_DATA_PULL_UP:
        if current == GPIO.HIGH:

```

```

        current_length = 0
        state = STATE_DATA_PULL_DOWN
    else:
        continue
    if state == STATE_DATA_PULL_DOWN:
        if current == GPIO.LOW:
            lengths.append(current_length)
            state = STATE_DATA_PULL_UP
        else:
            continue
    if len(lengths) != 40:
        #print ("Data not good, skip")
        return False

shortest_pull_up = min(lengths)
longest_pull_up = max(lengths)
halfway = (longest_pull_up + shortest_pull_up) / 2
bits = []
the_bytes = []
byte = 0

for length in lengths:
    bit = 0
    if length > halfway:
        bit = 1
    bits.append(bit)
#print ("bits: %s, length: %d" % (bits, len(bits)))
for i in range(0, len(bits)):
    byte = byte << 1
    if (bits[i]):
        byte = byte | 1
    else:
        byte = byte | 0
    if ((i + 1) % 8 == 0):
        the_bytes.append(byte)
        byte = 0
#print (the_bytes)
checksum = (the_bytes[0] + the_bytes[1] + the_bytes[2] + the_bytes[3]) & 0xFF
if the_bytes[4] != checksum:
    #print ("Data not good, skip")

```

```

        return False

    return the_bytes[0], the_bytes[2]

def main():

    while True:
        result = readDht11()
        if result:
            humidity, temperature = result
            print ("humidity: %s %%,  Temperature: %s C`" % (humidity, temperature))
            time.sleep(1)

    def destroy():
        GPIO.cleanup()

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

```

Kode Erklärung

```

def readDht11():
    GPIO.setup(dhtPin, GPIO.OUT)
    GPIO.output(dhtPin, GPIO.HIGH)
    time.sleep(0.05)
    GPIO.output(dhtPin, GPIO.LOW)
    time.sleep(0.02)
    GPIO.setup(dhtPin, GPIO.IN, GPIO.PUD_UP)
    unchanged_count = 0
    last = -1
    data = []
    #...

```

Diese Funktion wird verwendet, um die Funktionen von DHT11 zu implementieren. Es speichert die erkannten Daten im Array `the_bytes[]`. DHT11 überträgt Daten von jeweils 40 Bit. Die ersten 16 Bits beziehen sich auf die Luftfeuchtigkeit, die mittleren

16 Bits auf die Temperatur und die letzten acht Bits werden zur Überprüfung verwendet. Das Datenformat ist:

8-Bit-Feuchtigkeits-Integer-Daten + 8-Bit-Feuchtigkeits-Dezimaldaten + 8-Bit-Temperatur-Integer-Daten + 8-Bit-Temperatur-Dezimaldaten + 8-Bit-Prüfbit.

Wenn die Gültigkeit über das Prüfbit erkannt wird, gibt die Funktion zwei Ergebnisse zurück: 1. Fehler; 2. Luftfeuchtigkeit und Temperatur.

```
checksum = (the_bytes[0] + the_bytes[1] + the_bytes[2] + the_bytes[3]) & 0xFF  
if the_bytes[4] != checksum:  
    #print ("Data not good, skip")  
    return False  
  
return the_bytes[0], the_bytes[2]
```

Wenn das Empfangsdatum beispielsweise 00101011 (8-Bit-Wert der Feuchtigkeits-Ganzzahl) 00000000 (8-Bit-Wert der Feuchtigkeits-Dezimalzahl) 00111100 (8-Bit-Wert der Temperatur-Ganzzahl) 00000000 (8-Bit-Wert der Temperatur-Dezimalzahl) 01100111 (Bit prüfen)

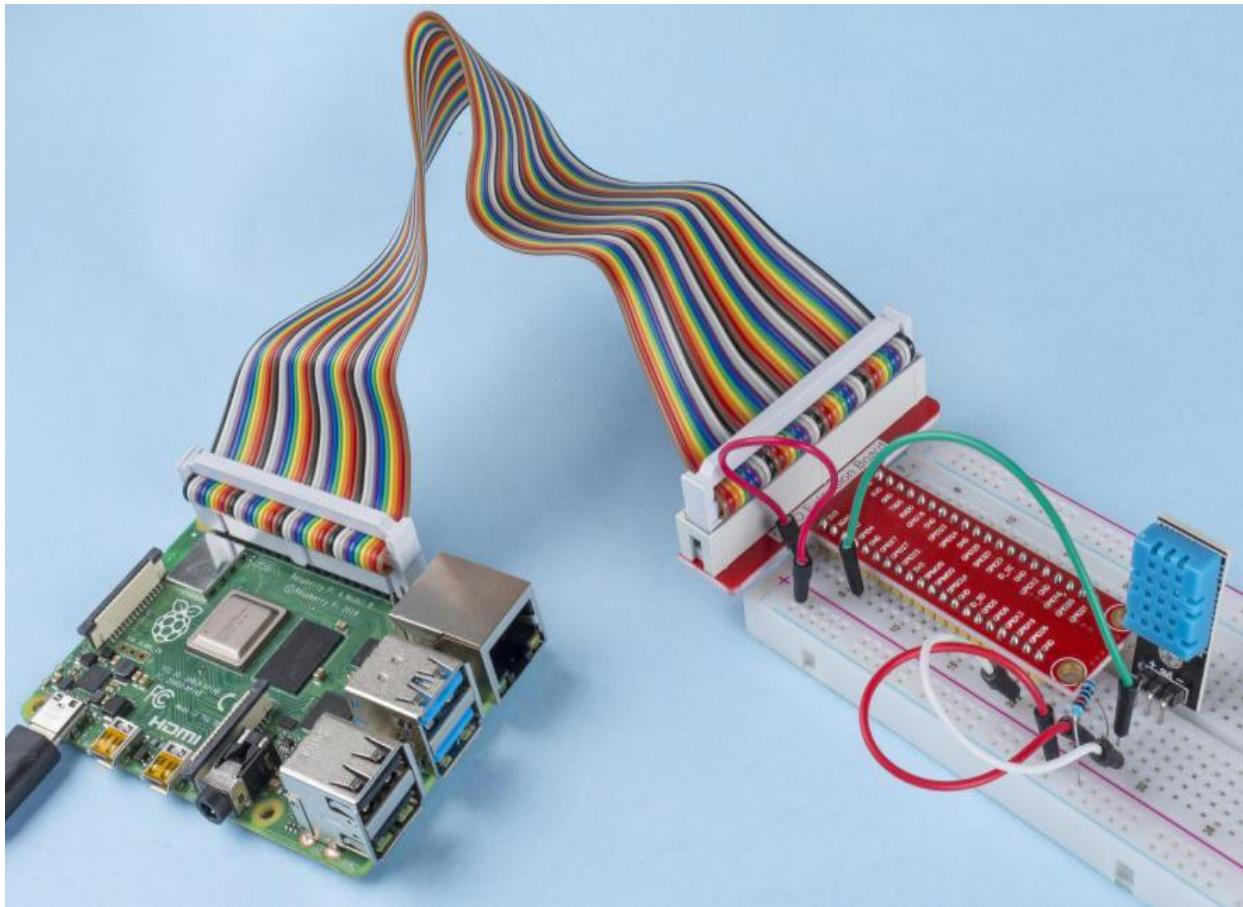
Berechnung:

$$00101011 + 00000000 + 00111100 + 00000000 = 01100111.$$

Wenn das Endergebnis den Prüfbitdaten entspricht, ist die Datenübertragung abnormal: return False.

Wenn das Endergebnis den Prüfbitdaten entspricht, sind die empfangenen Daten korrekt, dann werden die_Bytes [0] und die_Bytes [2] zurückgegeben und "Luftfeuchtigkeit = 43% , Temperatur = 60 ° C" ausgegeben.

Phänomen Bild

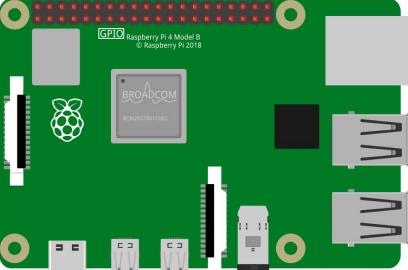
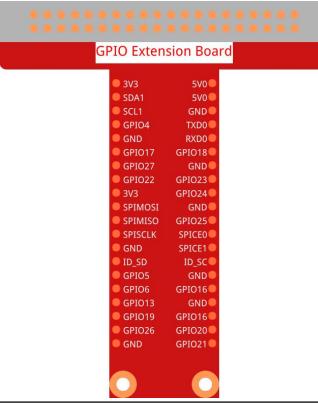
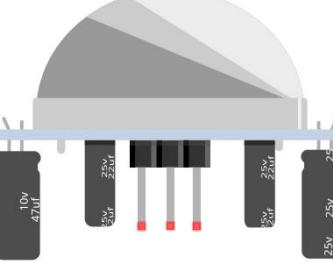
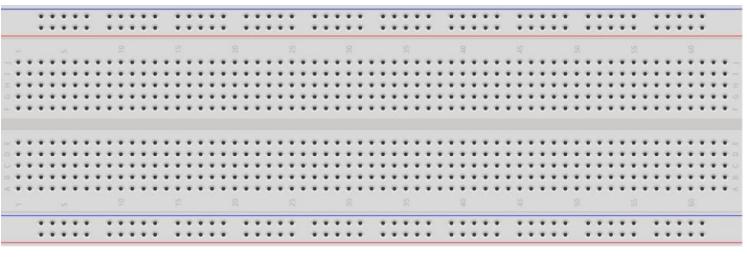


2.2.4 PIR

Einführung

In diesem Projekt werden wir ein Gerät unter Verwendung der pyroelektrischen Infrarotsensoren des menschlichen Körpers herstellen. Wenn sich jemand der LED nähert, leuchtet die LED automatisch auf. Wenn nicht, geht das Licht aus. Dieser Infrarot-Bewegungssensor ist ein Typ Sensor, der das von Mensch und Tier emittierte Infrarot erfassen kann.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * PIR-Sensormodul																																								
	 GPIO Extension Board <table border="1"><tr><td>3V3</td><td>5V0</td></tr><tr><td>SDA1</td><td>GND</td></tr><tr><td>SCL1</td><td></td></tr><tr><td>GPIO4</td><td>TXDD</td></tr><tr><td>GND</td><td>RXDD</td></tr><tr><td>GPIO17</td><td>GPIO18</td></tr><tr><td>GPIO27</td><td>GND</td></tr><tr><td>GPIO22</td><td>GPIO23</td></tr><tr><td>3V3</td><td>GPIO24</td></tr><tr><td>SPIMOSI</td><td>GND</td></tr><tr><td>SPIMISO</td><td>GPIO25</td></tr><tr><td>SPISCLK</td><td>SPICE0</td></tr><tr><td>GND</td><td>SPICE1</td></tr><tr><td>ID_SD</td><td>ID_SC</td></tr><tr><td>GPIO5</td><td>GND</td></tr><tr><td>GPIO6</td><td>GPIO16</td></tr><tr><td>GPIO13</td><td>GND</td></tr><tr><td>GPIO19</td><td>GPIO16</td></tr><tr><td>GPIO26</td><td>GPIO20</td></tr><tr><td>GND</td><td>GPIO21</td></tr></table>	3V3	5V0	SDA1	GND	SCL1		GPIO4	TXDD	GND	RXDD	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GPIO24	SPIMOSI	GND	SPIMISO	GPIO25	SPISCLK	SPICE0	GND	SPICE1	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	
3V3	5V0																																									
SDA1	GND																																									
SCL1																																										
GPIO4	TXDD																																									
GND	RXDD																																									
GPIO17	GPIO18																																									
GPIO27	GND																																									
GPIO22	GPIO23																																									
3V3	GPIO24																																									
SPIMOSI	GND																																									
SPIMISO	GPIO25																																									
SPISCLK	SPICE0																																									
GND	SPICE1																																									
ID_SD	ID_SC																																									
GPIO5	GND																																									
GPIO6	GPIO16																																									
GPIO13	GND																																									
GPIO19	GPIO16																																									
GPIO26	GPIO20																																									
GND	GPIO21																																									
1 * 40-Pin Kabel		Mehrere Überbrückungsdrähte																																								
																																										
1 * Steckbrett	3 * 220 Widerstand	1 * RGB-LED																																								
																																										

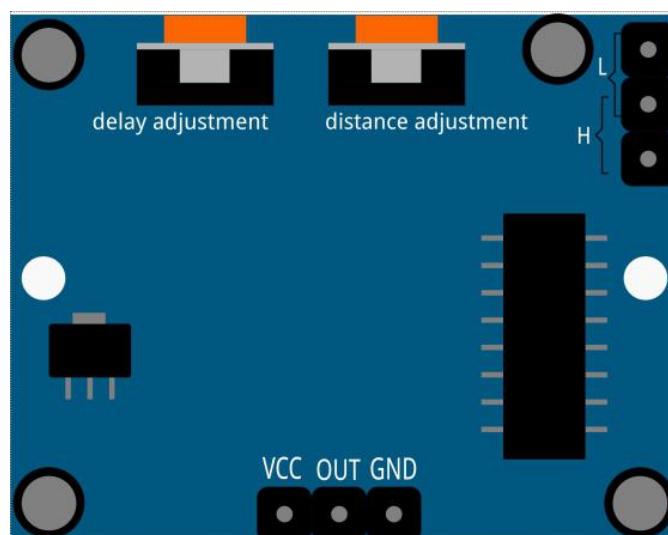
Prinzip

Der PIR-Sensor erfasst Infrarot-Wärmestrahlung, mit der das Vorhandensein von Organismen erfasst werden kann, die Infrarot-Wärmestrahlung emittieren.

Der PIR-Sensor ist in zwei Steckplätze unterteilt, die an einen Differenzverstärker angeschlossen sind. Wenn sich ein stationäres Objekt vor dem Sensor befindet, empfangen die beiden Schlitze die gleiche Strahlungsmenge und der Ausgang ist Null. Wenn sich ein sich bewegendes Objekt vor dem Sensor befindet, empfängt einer der Schlitze mehr Strahlung als der andere, wodurch der Ausgang stark oder niedrig schwankt. Diese Änderung der Ausgangsspannung ist ein Ergebnis der Bewegungserkennung.



Nach dem Verdrahten des Sensormoduls erfolgt eine einminütige Initialisierung. Während der Initialisierung wird das Modul in Intervallen 0 bis 3 Mal ausgegeben. Dann befindet sich das Modul im Standby-Modus. Bitte halten Sie die Interferenz von Lichtquellen und anderen Quellen von der Oberfläche des Moduls fern, um Fehlfunktionen durch das Störsignal zu vermeiden. Und Sie sollten das Modul besser ohne zu viel Wind verwenden, da der Wind auch den Sensor stören kann.



Abstandseinstellung

Durch Drehen des Knopfes des Potentiometers zur Entfernungseinstellung im Uhrzeigersinn vergrößert sich der Bereich der Erfassungsentfernung und der maximale Erfassungsentfernungsreichbereich beträgt etwa 0 bis 7 Meter. Wenn Sie ihn gegen den Uhrzeigersinn drehen, verringert sich die Reichweite der Erfassungsentfernung, und die minimale Reichweite der Erfassungsentfernung beträgt etwa 0 bis 3 Meter.

Verzögerungseinstellung

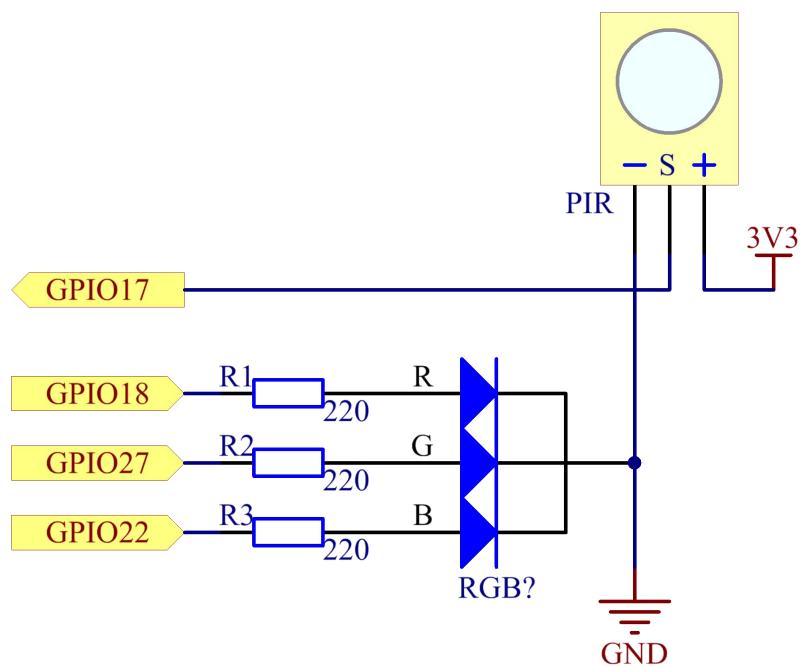
Drehen Sie die Taste des Potentiometers für die Verzögerungseinstellung im Uhrzeigersinn. Sie können auch sehen, wie die Erfassungsverzögerung zunimmt. Das Maximum der Erfassungsverzögerung kann bis zu 300 s erreichen. Im Gegenteil, wenn Sie es gegen den Uhrzeigersinn drehen, können Sie die Verzögerung um mindestens 5 Sekunden verkürzen.

Zwei Triggermodus: (Auswahl verschiedener Modus mit der Überbrückungskappe).

- ✧ **H: Wiederholbarer Triggermodus**, nachdem der menschliche Körper erfasst wurde, gibt das Modul einen hohen Niveau aus. Wenn während der nachfolgenden Verzögerungszeit jemand den Erfassungsbereich betritt, bleibt der Ausgang auf dem hohen Niveau.
- ✧ **L: Nicht wiederholbarer Triggermodus**, der einen hohen Niveau ausgibt, wenn er den menschlichen Körper erfasst. Nach der Verzögerung wechselt der Ausgang automatisch von einem hohen zu einem niedrigen Niveau.

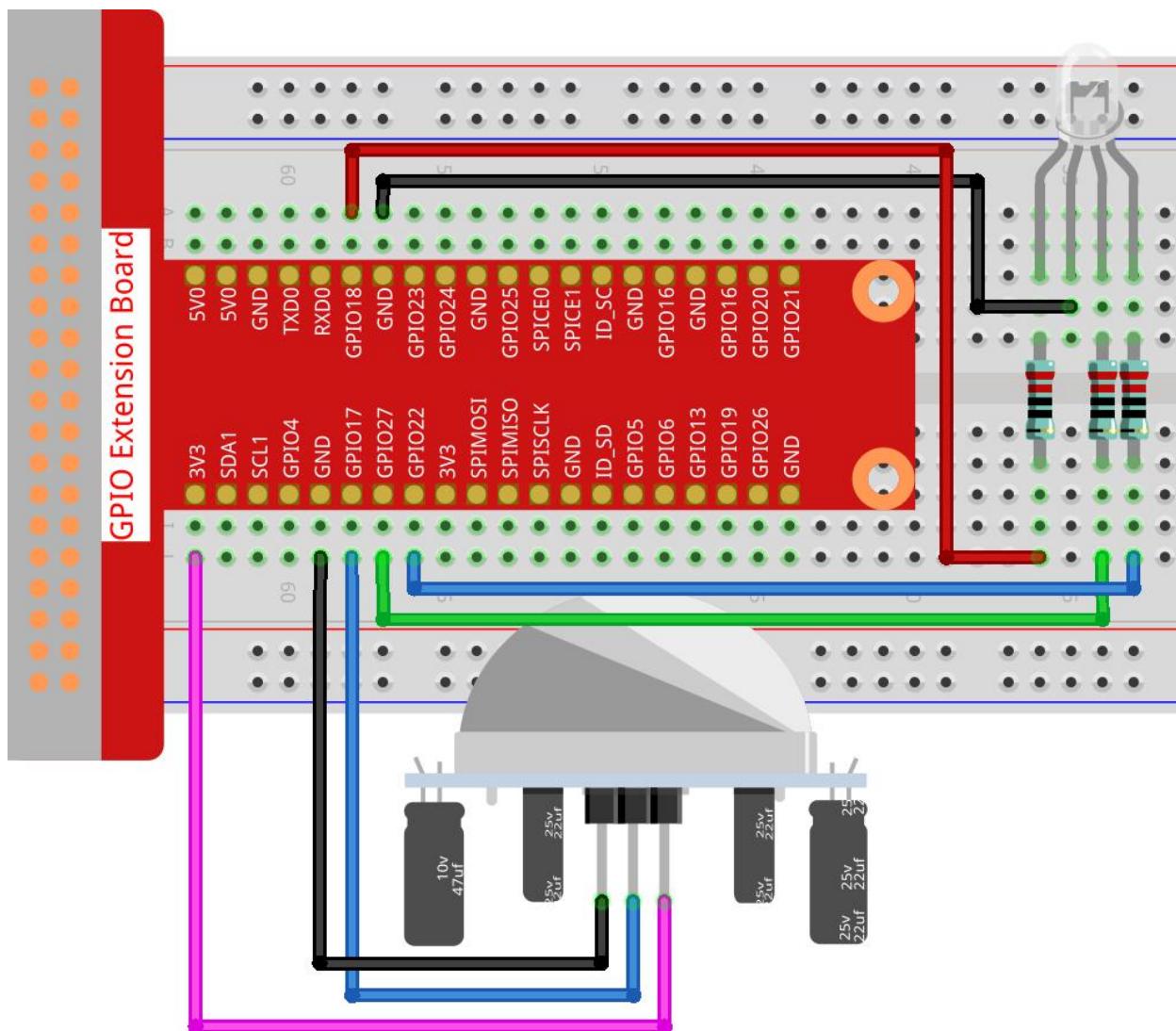
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin12	1	18
GPIO27	Pin13	2	27
GPIO22	Pin15	3	22



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.2.4/
```

Schritt 3: Kompilieren Sie die Kode.

```
gcc 2.2.4_PIR.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Nachdem die Kode ausgeführt wurde, erkennt PIR die Umgebung und lässt die RGB-LED gelb leuchten, wenn sie erkennt, dass jemand vorbeigeht. Das PIR-Modul verfügt über zwei Potentiometer: Das eine dient zum Einstellen der Empfindlichkeit und das andere zum Einstellen des Erfassungsabstands. Damit das PIR-Modul besser funktioniert, müssen Sie versuchen, diese beiden Potentiometer einzustellen.

Kode

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

#define uchar unsigned char

#define pirPin    0      //the pir connect to GPIO0
#define redPin    1
#define greenPin  2
#define bluePin   3

void ledInit(void){
    softPwmCreate(redPin, 0, 100);
    softPwmCreate(greenPin,0, 100);
    softPwmCreate(bluePin, 0, 100);
}

void ledColorSet(uchar r_val, uchar g_val, uchar b_val){
```

```
softPwmWrite(redPin,    r_val);
softPwmWrite(greenPin,  g_val);
softPwmWrite(bluePin,   b_val);

}

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

    ledInit();
    pinMode(pirPin, INPUT);
    while(1){
        pir_val = digitalRead(pirPin);
        if(pir_val== 1){ //if read pir is HIGH level
            ledColorSet(0xff,0xff,0x00);
        }
        else {
            ledColorSet(0x00,0x00,0xff);
        }
    }
    return 0;
}
```

Kode Erklärung

```
void ledInit(void);  
  
void ledColorSet(uchar r_val, uchar g_val, uchar b_val);
```

Mit dieser Kode wird die Farbe der RGB-LED eingestellt. Weitere Informationen finden Sie unter **1.1.2-RGB-LED**.

```
int main(void)  
{  
    int pir_val;  
    //.....  
    pinMode(pirPin, INPUT);  
    while(1){  
        pir_val = digitalRead(pirPin);  
        if(pir_val== 1){ //if read pir is HIGH level  
            ledColorSet(0xff,0xff,0x00);  
        }  
        else {  
            ledColorSet(0x00,0x00,0xff);  
        }  
    }  
    return 0;  
}
```

Wenn PIR das menschliche Infrarotspektrum erkennt, sendet die RGB-LED das gelbe Licht aus. Wenn nicht, wird das blaue Licht ausgesendet.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 2.2.4_PIR.py
```

Nachdem die Kode ausgeführt wurde, erkennt PIR die Umgebung und lässt die RGB-LED gelb leuchten, wenn sie erkennt, dass jemand vorbeigeht. Das PIR-Modul verfügt über zwei Potentiometer: Das eine dient zum Einstellen der Empfindlichkeit und das

andere zum Einstellen des Erfassungsabstands. Damit das PIR-Modul besser funktioniert, müssen Sie versuchen, diese beiden Potentiometer einzustellen.

Kode

```
import RPi.GPIO as GPIO
import time

rgbPins = {'Red':18, 'Green':27, 'Blue':22}
pirPin = 17      # the pir connect to pin17

def setup():
    global p_R, p_G, p_B
    GPIO.setmode(GPIO.BCM)      # Set the GPIO modes to BCM Numbering
    GPIO.setup(pirPin, GPIO.IN)  # Set pirPin to input
    # Set all LedPin's mode to output and initial level to High(3.3v)
    for i in rgbPins:
        GPIO.setup(rgbPins[i], GPIO.OUT, initial=GPIO.HIGH)

    # Set all led as pwm channel and freuece to 2KHz
    p_R = GPIO.PWM(rgbPins['Red'], 2000)
    p_G = GPIO.PWM(rgbPins['Green'], 2000)
    p_B = GPIO.PWM(rgbPins['Blue'], 2000)

    # Set all begin with value 0
    p_R.start(0)
    p_G.start(0)
    p_B.start(0)

# Define a MAP function for mapping values. Like from 0~255 to 0~100
def MAP(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
```

```
# Define a function to set up colors

def setColor(color):

    # configures the three LEDs' luminance with the inputted color value .

        # Devide colors from 'color' veriable

    R_val = (color & 0xFF0000) >> 16
    G_val = (color & 0x00FF00) >> 8
    B_val = (color & 0x0000FF) >> 0

    # Map color value from 0~255 to 0~100

    R_val = MAP(R_val, 0, 255, 0, 100)
    G_val = MAP(G_val, 0, 255, 0, 100)
    B_val = MAP(B_val, 0, 255, 0, 100)

#Assign the mapped duty cycle value to the corresponding PWM channel to change the
luminance.

p_R.ChangeDutyCycle(R_val)
p_G.ChangeDutyCycle(G_val)
p_B.ChangeDutyCycle(B_val)

#print ("color_msg: R_val = %s, G_val = %s, B_val = %s"%(R_val, G_val, B_val))

def loop():

    while True:

        pir_val = GPIO.input(pirPin)

        if pir_val==GPIO.HIGH:

            setColor(0xFFFF00)

        else :

            setColor(0x0000FF)

def destroy():

    p_R.stop()
    p_G.stop()
    p_B.stop()
```

```

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()

```

Kode Erklärung

```

rgbPins = {'Red':18, 'Green':27, 'Blue':22}

def setup():
    global p_R, p_G, p_B
    GPIO.setmode(GPIO.BCM)
    # .....
    for i in rgbPins:
        GPIO.setup(rgbPins[i], GPIO.OUT, initial=GPIO.HIGH)
    p_R = GPIO.PWM(rgbPins['Red'], 2000)
    p_G = GPIO.PWM(rgbPins['Green'], 2000)
    p_B = GPIO.PWM(rgbPins['Blue'], 2000)
    p_R.start(0)
    p_G.start(0)
    p_B.start(0)

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

def setColor(color):

```

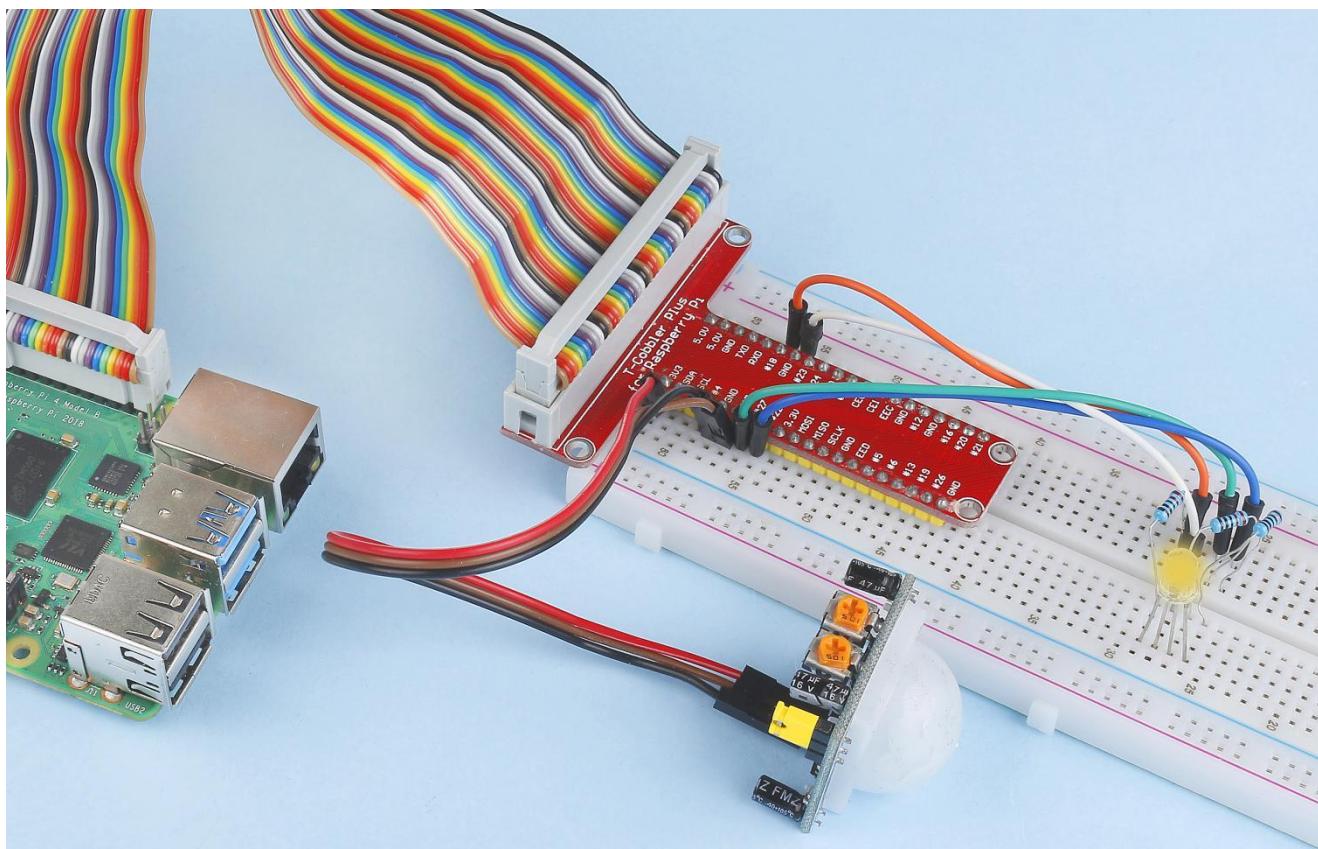
...

Mit dieser Kode wird die Farbe der RGB-LED eingestellt. Weitere Informationen finden Sie unter **1.1.2-RGB-LED**.

```
def loop():
    while True:
        pir_val = GPIO.input(pirPin)
        if pir_val==GPIO.HIGH:
            setColor(0xFFFF00)
        else :
            setColor(0x0000FF)
```

Wenn PIR das menschliche Infrarotspektrum erkennt, sendet die RGB-LED das gelbe Licht aus. Wenn nicht, wird das blaue Licht ausgesendet.

Phänomen Bild

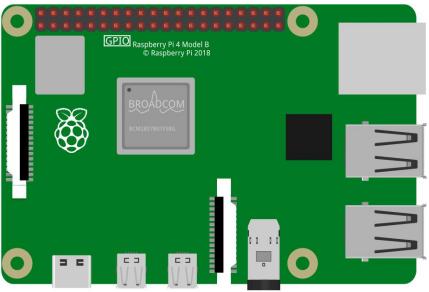
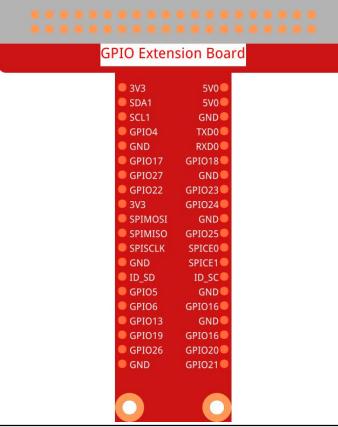
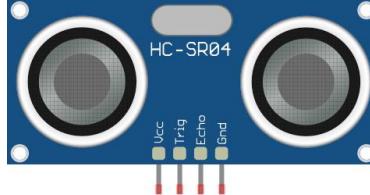
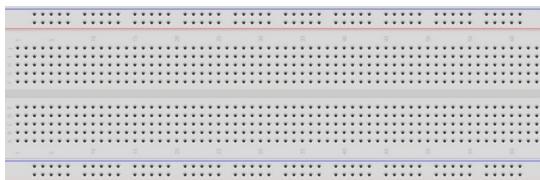


2.2.5 Ultraschallsensormodul

Einführung

Der Ultraschallsensor verwendet Ultraschall, um Objekte genau zu erfassen und die Entfernung zu messen. Es sendet Ultraschallwellen aus und wandelt sie in elektronische Signale um.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * HC-SR04																																										
	 <table border="1"> <thead> <tr> <th colspan="2">GPIO Extension Board</th> </tr> </thead> <tbody> <tr><td>3V3</td><td>5V0</td></tr> <tr><td>SDA1</td><td>5V0</td></tr> <tr><td>SCL1</td><td>GND</td></tr> <tr><td>GPIO4</td><td>TxD0</td></tr> <tr><td>GND</td><td>RxD0</td></tr> <tr><td>GPIO17</td><td>GPIO18</td></tr> <tr><td>GPIO27</td><td>GND</td></tr> <tr><td>GPIO22</td><td>GPIO23</td></tr> <tr><td>3V3</td><td>GPIO24</td></tr> <tr><td>SPI_MOSI</td><td>GND</td></tr> <tr><td>SPI_MISO</td><td>GPIO25</td></tr> <tr><td>SPI_SCLK</td><td>SPICE0</td></tr> <tr><td>GND</td><td>SPICE1</td></tr> <tr><td>ID_SD</td><td>ID_SC</td></tr> <tr><td>GPIO5</td><td>GND</td></tr> <tr><td>GPIO6</td><td>GPIO16</td></tr> <tr><td>GPIO13</td><td>GND</td></tr> <tr><td>GPIO19</td><td>GPIO16</td></tr> <tr><td>GPIO26</td><td>GPIO20</td></tr> <tr><td>GND</td><td>GPIO21</td></tr> </tbody> </table>	GPIO Extension Board		3V3	5V0	SDA1	5V0	SCL1	GND	GPIO4	TxD0	GND	RxD0	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GPIO24	SPI_MOSI	GND	SPI_MISO	GPIO25	SPI_SCLK	SPICE0	GND	SPICE1	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	
GPIO Extension Board																																												
3V3	5V0																																											
SDA1	5V0																																											
SCL1	GND																																											
GPIO4	TxD0																																											
GND	RxD0																																											
GPIO17	GPIO18																																											
GPIO27	GND																																											
GPIO22	GPIO23																																											
3V3	GPIO24																																											
SPI_MOSI	GND																																											
SPI_MISO	GPIO25																																											
SPI_SCLK	SPICE0																																											
GND	SPICE1																																											
ID_SD	ID_SC																																											
GPIO5	GND																																											
GPIO6	GPIO16																																											
GPIO13	GND																																											
GPIO19	GPIO16																																											
GPIO26	GPIO20																																											
GND	GPIO21																																											
Mehrere Überbrückungsdrähte																																												
1 * 40-Pin Kabel		1 * Steckbrett																																										
																																												

Prinzip

Ultraschall

Das Ultraschall-Entfernungsmessmodul bietet eine berührungslose Messfunktion von 2cm bis 400 cm, und die Entfernungsgenauigkeit kann bis zu 3mm betragen. Es kann sicherstellen, dass das Signal innerhalb von 5m stabil ist und das Signal nach 5m allmählich geschwächt wird, bis die 7m-Position verschwindet.

Das Modul enthält Ultraschallsender, Empfänger und Steuerschaltung. Die Grundprinzipien sind wie folgt:

- (1) Verwenden Sie ein IO -Flipflop, um ein Signal mit hohem Niveau von mindestens 10us zu verarbeiten.

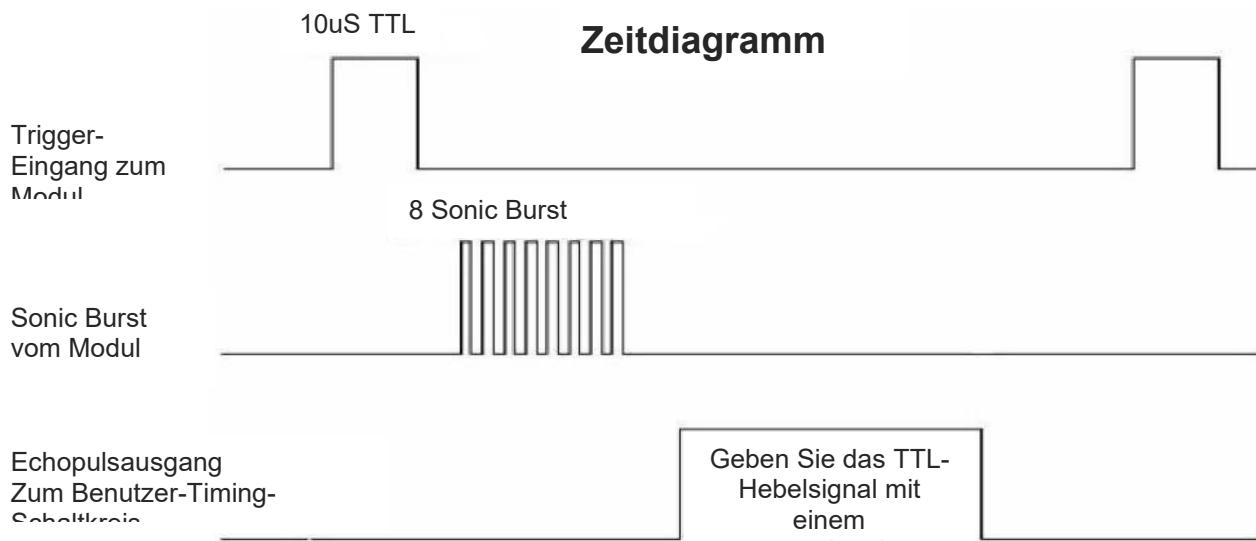
- (2) Das Modul sendet automatisch acht 40kHz und erkennt, ob ein Impulssignal zurückkehrt.
- (3) Wenn das Signal zurückkehrt und den hohen Niveau überschreitet, ist die IO - Dauer des hohen Ausgangs die Zeit von der Übertragung der Ultraschallwelle bis zu ihrer Rückkehr. Hier Testabstand = (hohe Zeit x Schallgeschwindigkeit (340 m / s) / 2.



TRIG	Impulseingang auslösen
ECHO	Echopulsausgang
GND	Boden
VCC	Liefern

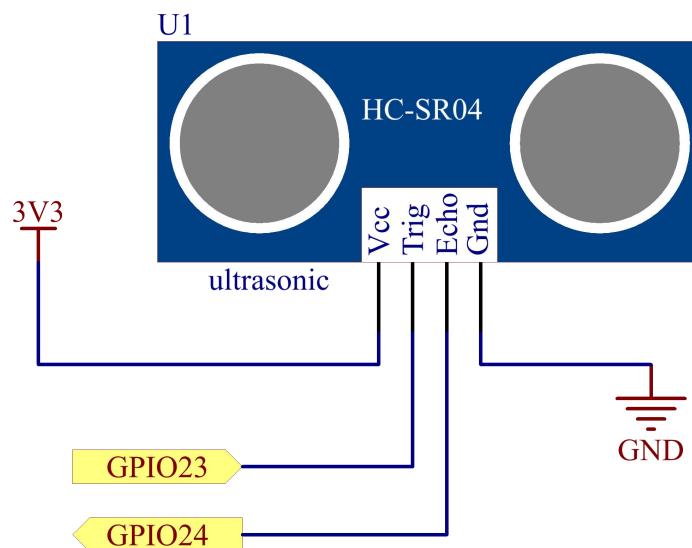
Das Zeitdiagramm ist unten dargestellt. Sie müssen nur einen kurzen 10us-Impuls für den Triggereingang liefern, um die Entfernungsmessung zu starten. Anschließend sendet das Modul einen Ultraschallstoß von 8 Zyklen bei 40 kHz und erhöht sein Echo. Sie können den Bereich über das Zeitintervall zwischen dem Senden des Triggersignals und dem Empfangen des Echosignals berechnen.

Formel: $us / 58 = \text{Zentimeter}$ oder $us / 148 = \text{Zoll}$; oder: der Bereich = hohe Niveauzeit * Geschwindigkeit (340M/S) / 2; Es wird empfohlen, einen Messzyklus über 60ms zu verwenden, um Signalkollisionen des Triggersignals und des Echosignals zu vermeiden.



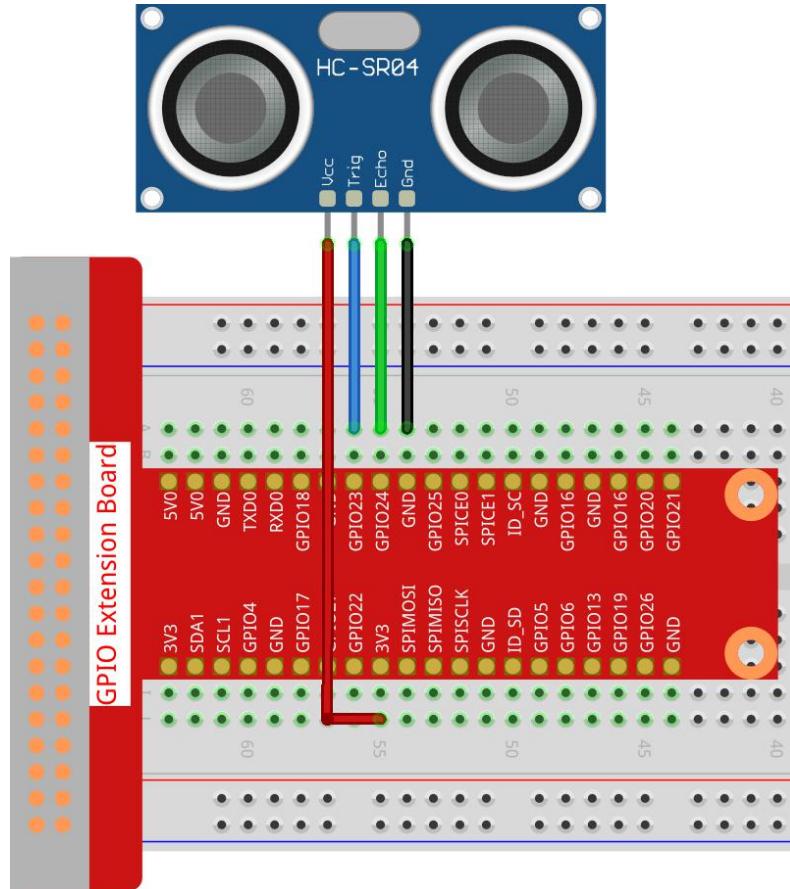
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.2.5/
```

Schritt 3: Kompilieren Sie die Kode.

```
gcc 2.2.5_Ultrasonic.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Wenn die Kode ausgeführt wird, erkennt das Ultraschallsensormodul den Abstand zwischen dem vorausfahrenden Hindernis und dem Modul selbst. Anschließend wird der Abstandswert auf dem Bildschirm gedruckt.

Kode

```
#include <wiringPi.h>
#include <stdio.h>
#include <sys/time.h>

#define Trig    4
#define Echo   5
```

```
void ultraInit(void)
{
    pinMode(Echo, INPUT);
    pinMode(Trig, OUTPUT);
}

float disMeasure(void)
{
    struct timeval tv1;
    struct timeval tv2;
    long time1, time2;
    float dis;

    digitalWrite(Trig, LOW);
    delayMicroseconds(2);

    digitalWrite(Trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig, LOW);

    while(!(digitalRead(Echo) == 1));
    gettimeofday(&tv1, NULL);

    while(!(digitalRead(Echo) == 0));
    gettimeofday(&tv2, NULL);

    time1 = tv1.tv_sec * 1000000 + tv1.tv_usec;
    time2 = tv2.tv_sec * 1000000 + tv2.tv_usec;

    dis = (float)(time2 - time1) / 1000000 * 34000 / 2;

    return dis;
}

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

```

ultraInit();

while(1){
    dis = disMeasure();
    printf("%0.2f cm\n\n",dis);
    delay(300);
}

return 0;
}

```

Kode Explanation

```

void ultraInit(void)
{
    pinMode(Echo, INPUT);
    pinMode(Trig, OUTPUT);
}

```

Initialisieren Sie den Ultraschall Pin. Stellen Sie währenddessen Echo auf Eingabe und Trig auf Ausgabe.

```
float disMeasure(void){};
```

Diese Funktion wird verwendet, um die Funktion des Ultraschallsensors durch Berechnung der Rückerkennungsentfernung zu realisieren.

```

def distance():
struct timeval tv2;

```

Strukturzeitwert ist eine Struktur, die zum Speichern der aktuellen Zeit verwendet wird. Die vollständige Struktur ist wie folgt:

```

struct timeval
{
    __time_t tv_sec;          /* Seconds. */
    __suseconds_t tv_usec;   /* Microseconds. */
};

```

Hier repräsentiert tv_sec die Sekunden, die Epoch beim Erstellen des Strukturzeitwerts verbracht hat. Tv_usec steht für Mikrosekunden oder einen Bruchteil von Sekunden.

```
digitalWrite(Trig, HIGH);
```

```
delayMicroseconds(10);  
digitalWrite(Trig, LOW);
```

Ein 10us Ultraschallimpuls wird gesendet.

```
while(!(digitalRead(Echo) == 1));  
gettimeofday(&tv1, NULL);
```

Diese leere Schleife wird verwendet, um sicherzustellen, dass beim Senden des Triggersignals kein störendes Echosignal vorhanden ist, und um dann die aktuelle Zeit zu erhalten.

```
while(!(digitalRead(Echo) == 0));  
gettimeofday(&tv2, NULL);
```

Diese leere Schleife wird verwendet, um sicherzustellen, dass der nächste Schritt erst ausgeführt wird, wenn das Echosignal empfangen wird, und um dann die aktuelle Zeit abzurufen.

```
time1 = tv1.tv_sec * 1000000 + tv1.tv_usec;  
time2 = tv2.tv_sec * 1000000 + tv2.tv_usec;
```

Konvertieren Sie die von struct timeval gespeicherte Zeit in eine volle Mikrosekundenzeit.

```
dis = (float)(time2 - time1) / 1000000 * 34000 / 2;
```

Die Entfernung wird durch das Zeitintervall und die Geschwindigkeit der Schallausbreitung berechnet. Die Schallgeschwindigkeit in der Luft: 34000cm / s.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 2.2.5_Ultrasonic.py
```

Wenn die Kode ausgeführt wird, erkennt das Ultraschallsensormodul den Abstand zwischen dem vorausfahrenden Hindernis und dem Modul selbst. Anschließend wird der Abstandswert auf dem Bildschirm gedruckt.

Kode

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

```
import time

TRIG = 16
ECHO = 18

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(TRIG, GPIO.OUT)
    GPIO.setup(ECHO, GPIO.IN)

def distance():
    GPIO.output(TRIG, 0)
    time.sleep(0.000002)

    GPIO.output(TRIG, 1)
    time.sleep(0.00001)
    GPIO.output(TRIG, 0)

    while GPIO.input(ECHO) == 0:
        a = 0
        time1 = time.time()
        while GPIO.input(ECHO) == 1:
            a = 1
            time2 = time.time()

        during = time2 - time1
    return during * 340 / 2 * 100

def loop():
    while True:
        dis = distance()
        print ('Distance: %.2f' % dis )
        time.sleep(0.3)

def destroy():
    GPIO.cleanup()

if __name__ == "__main__":
    setup()
    try:
```

```
loop()
except KeyboardInterrupt:
    destroy()
```

Kode Erklärung

```
def distance():
```

Diese Funktion wird verwendet, um die Funktion des Ultraschallsensors durch Berechnung der Rückerkennungsentfernung zu realisieren.

```
GPIO.output(TRIG, 1)
time.sleep(0.00001)
GPIO.output(TRIG, 0)
```

Dies sendet einen 10us Ultraschallimpuls aus.

```
while GPIO.input(ECHO) == 0:
    a = 0
    time1 = time.time()
```

Diese leere Schleife wird verwendet, um sicherzustellen, dass beim Senden des Triggersignals kein störendes Echosignal vorhanden ist, und um dann die aktuelle Zeit zu erhalten.

```
while GPIO.input(ECHO) == 1:
    a = 1
    time2 = time.time()
```

Diese leere Schleife wird verwendet, um sicherzustellen, dass der nächste Schritt erst ausgeführt wird, wenn das Echosignal empfangen wird, und um dann die aktuelle Zeit abzurufen.

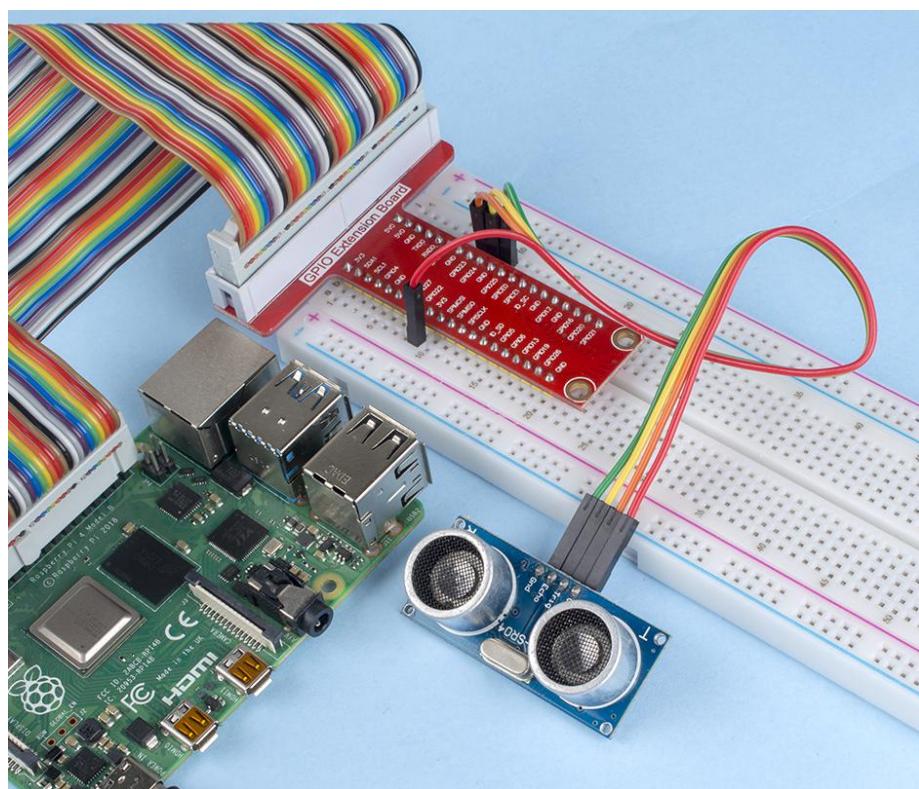
```
during = time2 - time1
```

Führen Sie die Intervallberechnung durch.

```
return during * 340 / 2 * 100
```

Die Entfernung wird unter Berücksichtigung des Zeitintervalls und der Schallausbreitungsgeschwindigkeit berechnet. Die Schallgeschwindigkeit in der Luft: 340 m / s.

Phänomen Bild



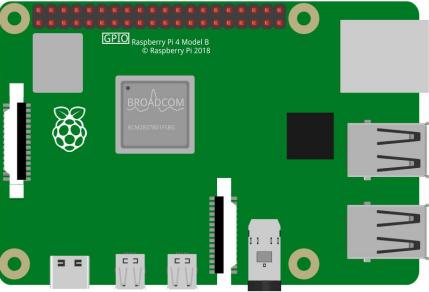
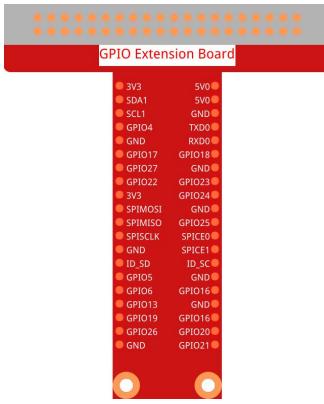
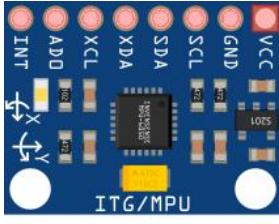
2.2.6 MPU6050-Modul

Einführung

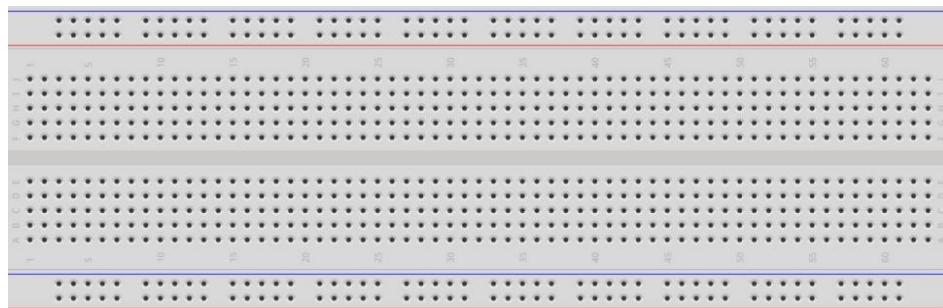
Die MPU-6050 ist das weltweit erste und einzige 6-Achsen-Bewegungsverfolgungsgerät (3-Achsen-Gyroskop und 3-Achsen-Beschleunigungsmesser), das für Smartphones, Tablets und tragbare Sensoren entwickelt wurde, die diese Funktionen aufweisen, einschließlich geringer Leistung, geringer Kosten und hoher Leistung Leistungsanforderungen.

Verwenden Sie in diesem Experiment I2C, um die Werte des dreiachsigem Beschleunigungssensors und des dreiachsigem Gyroskops für MPU6050 zu erhalten und auf dem Bildschirm anzuzeigen.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * MPU6050																																										
	 <table border="1"> <thead> <tr> <th colspan="2">GPIO Extension Board</th> </tr> </thead> <tbody> <tr><td>3V3</td><td>5V0</td></tr> <tr><td>SDA1</td><td>GND</td></tr> <tr><td>SCL1</td><td>GND</td></tr> <tr><td>GPIO4</td><td>TXDD</td></tr> <tr><td>GND</td><td>RXDD</td></tr> <tr><td>GPIO17</td><td>GPIO18</td></tr> <tr><td>GPIO27</td><td>GND</td></tr> <tr><td>GPIO22</td><td>GPIO23</td></tr> <tr><td>3V3</td><td>GND</td></tr> <tr><td>SPIMOSI</td><td>GPIO24</td></tr> <tr><td>SPIMOSO</td><td>GPIO25</td></tr> <tr><td>SPI5CLK</td><td>SPICEO</td></tr> <tr><td>GND</td><td>SPICEI</td></tr> <tr><td>ID_SD</td><td>ID_SC</td></tr> <tr><td>GPIO5</td><td>GND</td></tr> <tr><td>GPIO6</td><td>GPIO16</td></tr> <tr><td>GPIO13</td><td>GND</td></tr> <tr><td>GPIO19</td><td>GPIO16</td></tr> <tr><td>GPIO26</td><td>GPIO20</td></tr> <tr><td>GND</td><td>GPIO21</td></tr> </tbody> </table>	GPIO Extension Board		3V3	5V0	SDA1	GND	SCL1	GND	GPIO4	TXDD	GND	RXDD	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GND	SPIMOSI	GPIO24	SPIMOSO	GPIO25	SPI5CLK	SPICEO	GND	SPICEI	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	 <p>ITG/MPU</p>
GPIO Extension Board																																												
3V3	5V0																																											
SDA1	GND																																											
SCL1	GND																																											
GPIO4	TXDD																																											
GND	RXDD																																											
GPIO17	GPIO18																																											
GPIO27	GND																																											
GPIO22	GPIO23																																											
3V3	GND																																											
SPIMOSI	GPIO24																																											
SPIMOSO	GPIO25																																											
SPI5CLK	SPICEO																																											
GND	SPICEI																																											
ID_SD	ID_SC																																											
GPIO5	GND																																											
GPIO6	GPIO16																																											
GPIO13	GND																																											
GPIO19	GPIO16																																											
GPIO26	GPIO20																																											
GND	GPIO21																																											
Mehrere Überbrückungsdrähte																																												
1 * 40-Pin Kabel																																												

1 * Steckbrett



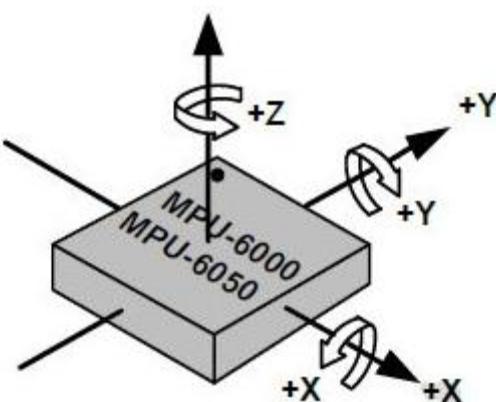
Prinzip

MPU6050

Die MPU-6050 ist ein 6-Achsen-Bewegungsverfolgungsgerät (kombiniert 3-Achsen-Gyroskop, 3-Achsen-Beschleunigungsmesser).

Seine drei Koordinatensysteme sind wie folgt definiert:

Legen Sie die MPU6050 flach auf den Tisch und stellen Sie sicher, dass das Gesicht mit dem Etikett nach oben zeigt und sich ein Punkt auf dieser Oberfläche in der oberen linken Ecke befindet. Dann ist die aufrechte Richtung nach oben die z-Achse des Chips. Die Richtung von links nach rechts wird als X-Achse angesehen. Dementsprechend ist die Richtung von hinten nach vorne als Y-Achse definiert.

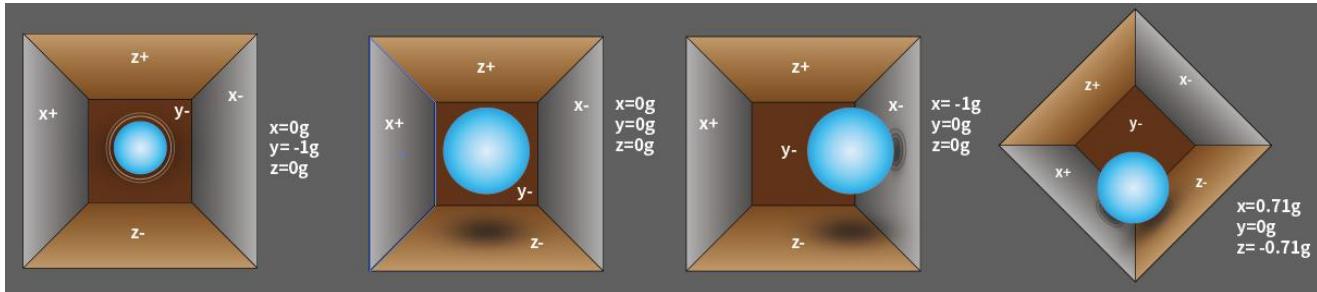


3-Achsbeschleunigungsmesser

Der Beschleunigungsmesser arbeitet nach dem Prinzip des piezoelektrischen Effekts, der Fähigkeit bestimmter Materialien, als Reaktion auf angelegte mechanische Beanspruchung eine elektrische Ladung zu erzeugen.

Stellen Sie sich hier eine quaderförmige Kisten vor, in der sich eine kleine Kugel befindet, wie im obigen Bild. Die Wände dieser Kisten bestehen aus piezoelektrischen Kristallen. Immer wenn Sie die Kisten kippen, muss sich der Ball aufgrund der Schwerkraft in Richtung der Neigung bewegen. Die Wand, mit der die Kugel kollidiert,

erzeugt winzige piezoelektrische Ströme. Es gibt insgesamt drei Paare gegenüberliegender Wände in einem Quader. Jedes Paar entspricht einer Achse im 3D-Raum: X-, Y- und Z-Achse. Abhängig vom Strom, der von den piezoelektrischen Wänden erzeugt wird, können wir die Neigungsrichtung und ihre Größe bestimmen.



Wir können die MPU6050 verwenden, um ihre Beschleunigung auf jeder Koordinatenachse zu erfassen (im stationären Desktop-Zustand beträgt die Beschleunigung der Z-Achse 1 Schwerkrafteinheit und die X- und Y-Achse 0). Wenn es gekippt ist oder sich in einem schwerelosen / übergewichtigen Zustand befindet, ändert sich der entsprechende Wert.

Es gibt vier Arten von Messbereichen, die programmgesteuert ausgewählt werden können: +/-2g, +/-4g, +/-8g und +/-16g (standardmäßig 2 g), die jeder Genauigkeit entsprechen. Die Werte reichen von -32768 bis 32767.

Der Messwert des Beschleunigungsmessers wird in einen Beschleunigungswert umgewandelt, indem der Messwert vom Messbereich auf den Messbereich abgebildet wird.

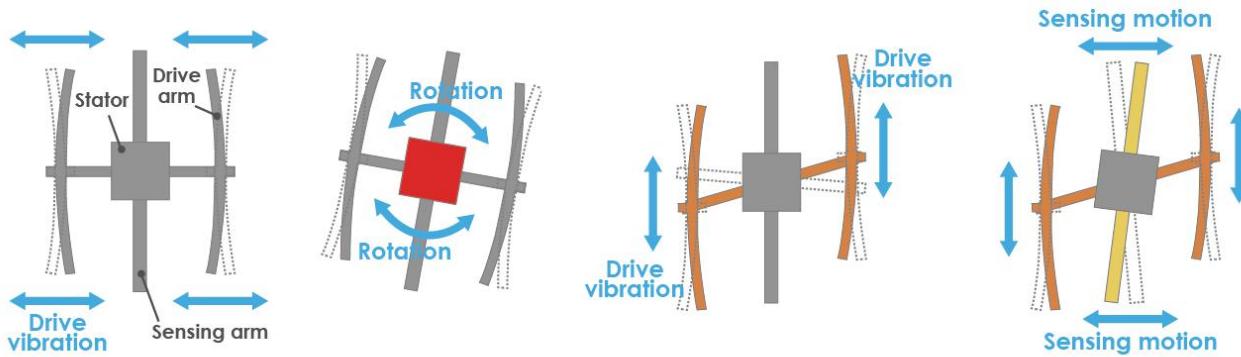
Beschleunigung = (Rohdaten der Beschleunigungsmesserachse / 65536 * voller Beschleunigungsbereich) g

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

$$\begin{aligned} \text{Beschleunigung entlang der X-Achse} &= (16384/65536 * 4) \text{ g} \\ &= 1 \text{ g} \end{aligned}$$

3-Achsengyroskop

Gyroskope arbeiten nach dem Prinzip der Coriolis-Beschleunigung.. Stellen Sie sich vor, es gibt eine gabelartige Struktur, die sich ständig hin und her bewegt. Es wird mit piezoelektrischen Kristallen an Ort und Stelle gehalten. Immer wenn Sie versuchen, diese Anordnung zu kippen, erfahren die Kristalle eine Kraft in Neigungsrichtung. Dies wird durch die Trägheit der beweglichen Gabel verursacht. Die Kristalle erzeugen somit einen Strom, der mit dem piezoelektrischen Effekt übereinstimmt, und dieser Strom wird verstärkt.



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

2. Direction of rotation

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

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

Das Gyroskop verfügt außerdem über vier Arten von Messbereichen: +/- 250, +/- 500, +/- 1000, +/- 2000. Die Berechnungsmethode und die Beschleunigung sind grundsätzlich konsistent.

Die Formel zum Umwandeln des Messwerts in die Winkelgeschwindigkeit lautet wie folgt:

Winkelgeschwindigkeit = (Rohdaten der Gyroskopachse / 65536 * Gyroskopbereich im vollen Maßstab) ° / s

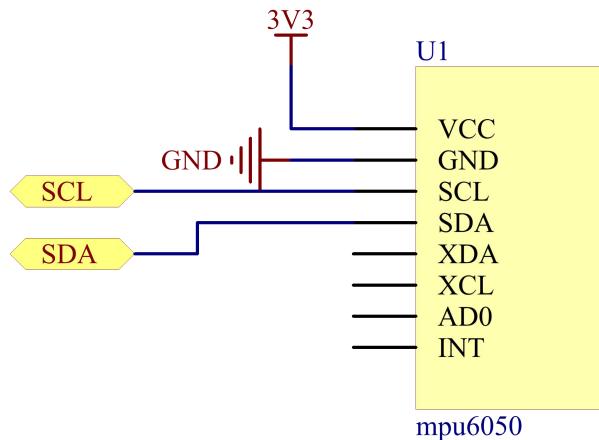
Die X-Achse, zum Beispiel die Rohdaten der X-Achse des Beschleunigungsmessers, ist 16384 und reicht von +/- 250 °/s:

$$\begin{aligned} \text{Winkelgeschwindigkeit entlang der X-Achse} &= (16384/65536 * 500) \text{ °/s} \\ &= 125 \text{ °/s} \end{aligned}$$

Schematische Darstellung

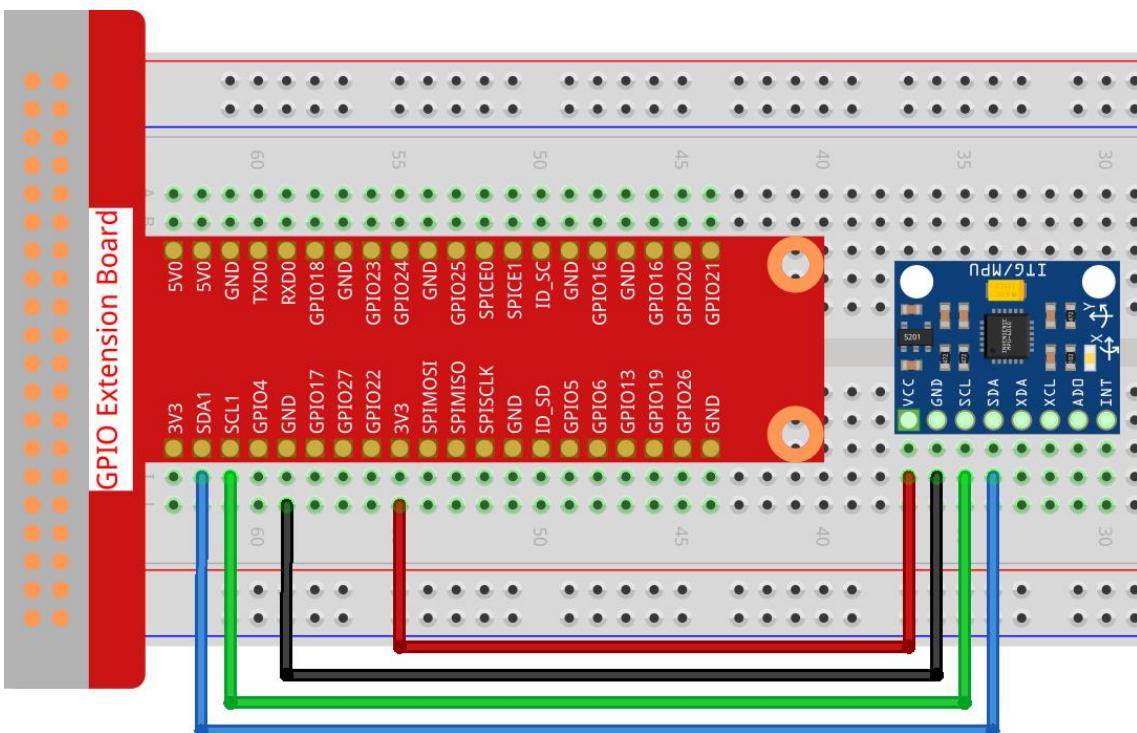
Die MPU6050 kommuniziert mit dem Mikrocontroller über die I2C-Busschnittstelle. Der SDA1 und der SCL1 müssen mit dem entsprechenden Pin verbunden werden.

T-Karte Name	physisch
SDA1	Pin 3
SCL1	Pin 5



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



Schritt 2: I2C einrichten (siehe Anhang. Wenn Sie I2C eingestellt haben, überspringen Sie diesen Schritt.)

➤ Für Benutzer in C-Sprache

Schritt 3: Gehen Sie zum Ordner der Kode.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.2.6/
```

Schritt 4: Kompilieren Sie der Kode

```
gcc 2.2.6_mpu6050.c -lwiringPi -lm
```

Schritt 5: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Wenn die Kode ausgeführt wird, werden der Ablenkwinkel der x-Achse, der y-Achse und die Beschleunigung sowie die Winkelgeschwindigkeit auf jeder von der MPU6050 gelesenen Achse nach der Berechnung auf dem Bildschirm gedruckt.

Kode

```
#include <wiringPi2C.h>
#include <wiringPi.h>
#include <stdio.h>
#include <math.h>
int fd;
int acclX, acclY, acclZ;
int gyroX, gyroY, gyroZ;
double acclX_scaled, acclY_scaled, acclZ_scaled;
double gyroX_scaled, gyroY_scaled, gyroZ_scaled;

int read_word_2c(int addr)
{
    int val;
    val = wiringPi2CReadReg8(fd, addr);
    val = val << 8;
    val += wiringPi2CReadReg8(fd, addr+1);
    if (val >= 0x8000)
        val = -(65536 - val);
    return val;
}

double dist(double a, double b)
{
    return sqrt((a*a) + (b*b));
}

double get_y_rotation(double x, double y, double z)
{
    double radians;
    radians = atan2(x, dist(y, z));
    return -(radians * (180.0 / M_PI));
}
```

```

double get_x_rotation(double x, double y, double z)
{
    double radians;
    radians = atan2(y, dist(x, z));
    return (radians * (180.0 / M_PI));
}

int main()
{
    fd = wiringPi2CSetup (0x68);
    wiringPi2CWriteReg8 (fd, 0x6B, 0x00); // disable sleep mode
    printf("set 0x6B=%X\n", wiringPi2CReadReg8 (fd, 0x6B));

    while(1) {

        gyroX = read_word_2c(0x43);
        gyroY = read_word_2c(0x45);
        gyroZ = read_word_2c(0x47);

        gyroX_scaled = gyroX / 131.0;
        gyroY_scaled = gyroY / 131.0;
        gyroZ_scaled = gyroZ / 131.0;

        // Print values for the X, Y, and Z axes of the gyroscope sensor.
        printf("My gyroX_scaled: %f\n", gyroX_scaled);
        printf("My gyroY_scaled: %f\n", gyroY_scaled);
        printf("My gyroZ_scaled: %f\n", gyroZ_scaled);

        acclX = read_word_2c(0x3B);
        acclY = read_word_2c(0x3D);
        acclZ = read_word_2c(0x3F);

        acclX_scaled = acclX / 16384.0;
        acclY_scaled = acclY / 16384.0;
        acclZ_scaled = acclZ / 16384.0;

        // Print the X, Y, and Z values of the acceleration sensor.
        printf("My acclX_scaled: %f\n", acclX_scaled);
        printf("My acclY_scaled: %f\n", acclY_scaled);
        printf("My acclZ_scaled: %f\n", acclZ_scaled);
    }
}

```

```

printf("My X rotation: %f\n", get_x_rotation(acclX_scaled, acclY_scaled, acclZ_scaled));
printf("My Y rotation: %f\n", get_y_rotation(acclX_scaled, acclY_scaled, acclZ_scaled));

delay(100);
}

return 0;
}

```

Kode Erklärung

```

int read_word_2c(int addr)
{
    int val;
    val = wiringPiI2CReadReg8(fd, addr);
    val = val << 8;
    val += wiringPiI2CReadReg8(fd, addr+1);
    if (val >= 0x8000)
        val = -(65536 - val);
    return val;
}

```

Lesen Sie die von der MPU6050 gesendeten Sensordaten.

```

double get_y_rotation(double x, double y, double z)
{
    double radians;
    radians = atan2(x, dist(y, z));
    return -(radians * (180.0 / M_PI));
}

```

Wir erhalten den Ablenkwinkel auf der Y-Achse.

```

double get_x_rotation(double x, double y, double z)
{
    double radians;
    radians = atan2(y, dist(x, z));
    return (radians * (180.0 / M_PI));
}

```

Berechnen Sie den Ablenkwinkel der x-Achse.

```

gyroX = read_word_2c(0x43);
gyroY = read_word_2c(0x45);
gyroZ = read_word_2c(0x47);

gyroX_scaled = gyroX / 131.0;
gyroY_scaled = gyroY / 131.0;
gyroZ_scaled = gyroZ / 131.0;

// Werte für die X-, Y- und Z-Achse des Gyroskopsensors drucken.

printf("My gyroX_scaled: %f\n", gyroX_scaled);
printf("My gyroY_scaled: %f\n", gyroY_scaled);
printf("My gyroZ_scaled: %f\n", gyroZ_scaled);

```

Lesen Sie die Werte der x-Achse, der y-Achse und der z-Achse auf dem Gyroskopsensor, konvertieren Sie die Metadaten in Winkelgeschwindigkeitswerte und drucken Sie sie dann aus.

```

acclX = read_word_2c(0x3B);
acclY = read_word_2c(0x3D);
acclZ = read_word_2c(0x3F);

acclX_scaled = acclX / 16384.0;
acclY_scaled = acclY / 16384.0;
acclZ_scaled = acclZ / 16384.0;

//Print the X, Y, and Z values of the acceleration sensor.

printf("My acclX_scaled: %f\n", acclX_scaled);
printf("My acclY_scaled: %f\n", acclY_scaled);
printf("My acclZ_scaled: %f\n", acclZ_scaled);

```

Lesen Sie die Werte der x-, y- und z-Achse auf dem Beschleunigungssensor ab, konvertieren Sie die Metadaten in beschleunigte Geschwindigkeitswerte (Schwerkrafteinheit) und drucken Sie sie dann aus.

```

printf("My X rotation: %f\n", get_x_rotation(acclX_scaled, acclY_scaled, acclZ_scaled));
printf("My Y rotation: %f\n", get_y_rotation(acclX_scaled, acclY_scaled, acclZ_scaled));

```

Drucken Sie die Ablenkinkel der x- und y-Achse.

➤ Für Python-Sprachbenutzer

Schritt 3: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo python3 2.2.6_mpu6050.py
```

Wenn die Kode ausgeführt wird, werden der Ablenkwinkel der x- und y-Achse sowie die Beschleunigung und Winkelgeschwindigkeit auf jeder von MPU6050 gelesenen Achse nach der Berechnung auf dem Bildschirm gedruckt.

Kode

```
import smbus
import math
import time

# Power management registers
power_mgmt_1 = 0x6b
power_mgmt_2 = 0x6c

def read_byte(adr):
    return bus.read_byte_data(address, adr)

def read_word(adr):
    high = bus.read_byte_data(address, adr)
    low = bus.read_byte_data(address, adr+1)
    val = (high << 8) + low
    return val

def read_word_2c(adr):
    val = read_word(adr)
    if (val >= 0x8000):
        return -((65535 - val) + 1)
    else:
        return val

def dist(a,b):
    return math.sqrt((a*a)+(b*b))

def get_y_rotation(x,y,z):
    radians = math.atan2(x, dist(y,z))
    return -math.degrees(radians)
```

```

def get_x_rotation(x,y,z):
    radians = math.atan2(y, dist(x,z))
    return math.degrees(radians)

bus = smbus.SMBus(1) # or bus = smbus.SMBus(1) for Revision 2 boards
address = 0x68      # This is the address value read via the i2cdetect command

# Now wake the 6050 up as it starts in sleep mode
bus.write_byte_data(address, power_mgmt_1, 0)

while True:
    time.sleep(0.1)
    gyro_xout = read_word_2c(0x43)
    gyro_yout = read_word_2c(0x45)
    gyro_zout = read_word_2c(0x47)

    print ("gyro_xout : ", gyro_xout, " scaled: ", (gyro_xout / 131))
    print ("gyro_yout : ", gyro_yout, " scaled: ", (gyro_yout / 131))
    print ("gyro_zout : ", gyro_zout, " scaled: ", (gyro_zout / 131))

    accel_xout = read_word_2c(0x3b)
    accel_yout = read_word_2c(0x3d)
    accel_zout = read_word_2c(0x3f)

    accel_xout_scaled = accel_xout / 16384.0
    accel_yout_scaled = accel_yout / 16384.0
    accel_zout_scaled = accel_zout / 16384.0

    print ("accel_xout: ", accel_xout, " scaled: ", accel_xout_scaled)
    print ("accel_yout: ", accel_yout, " scaled: ", accel_yout_scaled)
    print ("accel_zout: ", accel_zout, " scaled: ", accel_zout_scaled)

    print ("x rotation: " , get_x_rotation(accel_xout_scaled, accel_yout_scaled,
    accel_zout_scaled))
    print ("y rotation: " , get_y_rotation(accel_xout_scaled, accel_yout_scaled,
    accel_zout_scaled))
    time.sleep(0.5)

```

Kode Erklärung

```
def read_word(adr):
    high = bus.read_byte_data(address, adr)
    low = bus.read_byte_data(address, adr+1)
    val = (high << 8) + low
    return val

def read_word_2c(adr):
    val = read_word(adr)
    if (val >= 0x8000):
        return -((65535 - val) + 1)
    else:
        return val
```

Lesen Sie die von der MPU6050 gesendeten Sensordaten.

```
def get_y_rotation(x,y,z):
    radians = math.atan2(x, dist(y,z))
    return -math.degrees(radians)
```

Berechnen Sie den Ablenkwinkel der y-Achse.

```
def get_x_rotation(x,y,z):
    radians = math.atan2(y, dist(x,z))
    return math.degrees(radians)
```

Berechnen Sie den Ablenkwinkel der x-Achse.

```
gyro_xout = read_word_2c(0x43)
gyro_yout = read_word_2c(0x45)
gyro_zout = read_word_2c(0x47)

print ("gyro_xout : ", gyro_xout, " scaled: ", (gyro_xout / 131))
print ("gyro_yout : ", gyro_yout, " scaled: ", (gyro_yout / 131))
print ("gyro_zout : ", gyro_zout, " scaled: ", (gyro_zout / 131))
```

Lesen Sie die Werte der x-Achse, der y-Achse und der z-Achse auf dem Gyroskopsensor, konvertieren Sie die Metadaten in Winkelgeschwindigkeitswerte und drucken Sie sie dann aus.

```
accel_xout = read_word_2c(0x3b)
```

```

accel_yout = read_word_2c(0x3d)
accel_zout = read_word_2c(0x3f)

accel_xout_scaled = accel_xout / 16384.0
accel_yout_scaled = accel_yout / 16384.0
accel_zout_scaled = accel_zout / 16384.0

print ("accel_xout: ", accel_xout, " scaled: ", accel_xout_scaled)
print ("accel_yout: ", accel_yout, " scaled: ", accel_yout_scaled)
print ("accel_zout: ", accel_zout, " scaled: ", accel_zout_scaled)

```

Lesen Sie die Werte der x-, y- und z-Achse auf dem Beschleunigungssensor ab, konvertieren Sie die Elemente in den Wert für die beschleunigte Geschwindigkeit (Schwerkrafteinheit) und drucken Sie sie aus.

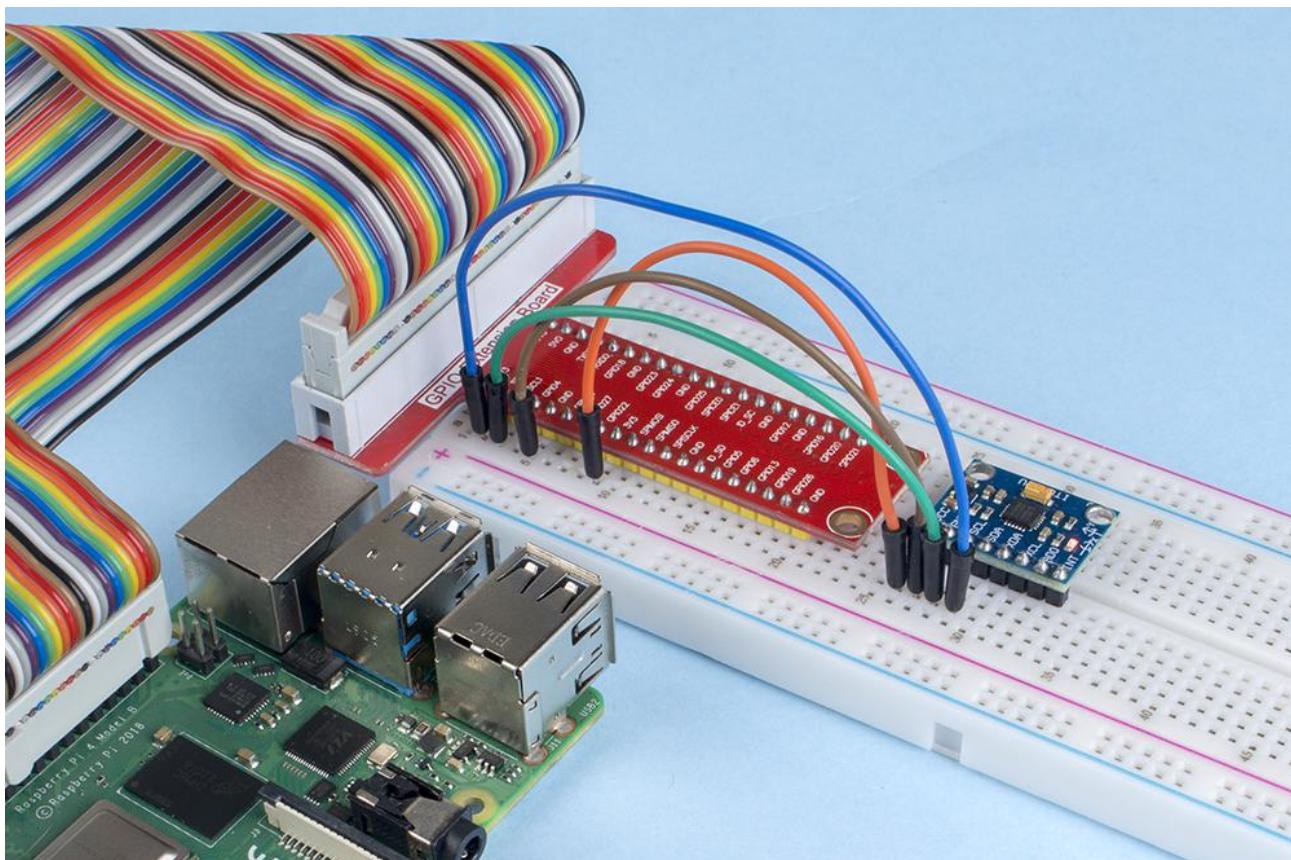
```

print ("x rotation: " , get_x_rotation(accel_xout_scaled, accel_yout_scaled,
accel_zout_scaled))
print ("y rotation: " , get_y_rotation(accel_xout_scaled, accel_yout_scaled,
accel_zout_scaled))

```

Drucken Sie die Ablenkwinkel der x- und y-Achse.

Phänomen Bild



2.2.7 MFRC522 RFID-Modul

Einführung

Radio Frequency Identification (RFID) bezieht sich auf Technologien, die die drahtlose Kommunikation zwischen einem Objekt (oder Tag) und einem Abfragegerät (oder Lesegerät) verwenden, um solche Objekte automatisch zu verfolgen und zu identifizieren.

Einige der häufigsten Anwendungen für diese Technologie sind Lieferketten für den Einzelhandel, militärische Lieferketten, automatisierte Zahlungsmethoden, Gepäckverfolgung und -verwaltung, Dokumentenverfolgung und pharmazeutische Verwaltung, um nur einige zu nennen.

In diesem Projekt werden wir RFID zum Lesen und Schreiben verwenden.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * RFID RC522 (mit weißer Karte und Schlüsselanhänger)
	 GPIO Extension Board Pinout: 3V3 SDA1 5V0 GND SCL1 GND GPIO4 DIO09 RXD0 GND DIO08 TXD0 GPIO17 GPIO18 GND GPIO27 GPIO23 GND 3V3 GPIO24 GND SPIMOSI GND SPIMISO GPIO25 GND SPISCLK SPICEO0 GND SPICE10 ID_SD ID_SC GND GPIO5 GPIO16 GND GPIO6 GPIO17 GND GPIO13 GND GPIO19 GPIO16 GND GPIO26 GPIO20 GND GND GPIO21 GND	 RFID-RC522 Pinout: 3.3V RST GND GND IRQ HSPI MOSI SCK SDA
Mehrere Überbrückungsdrähte		
1 * 40-Pin Kabel		
1 * Steckbrett		

Prinzip

RFID

Radio Frequency Identification (RFID) bezieht sich auf Technologien, bei denen eine drahtlose Kommunikation zwischen einem Objekt (oder Tag) und einem Abfragegerät (oder Lesegerät) verwendet wird, um solche Objekte automatisch zu verfolgen und zu identifizieren. Die Tag-Übertragungsreichweite ist auf mehrere Meter vom Lesegerät begrenzt. Eine klare Sichtlinie zwischen Lesegerät und Etikett ist nicht unbedingt erforderlich.

Die meisten Etikette enthalten mindestens eine integrierte Schaltung (IC) und eine Antenne. Der Mikrochip speichert Informationen und ist für die Verwaltung der Hochfrequenzkommunikation mit dem Lesegerät verantwortlich. Passive Etiketts haben keine unabhängige Energiequelle und sind auf ein externes elektromagnetisches Signal angewiesen, das vom Lesegerät bereitgestellt wird, um ihren Betrieb zu betreiben. Aktive Etiketts enthalten eine unabhängige Energiequelle, z. B. eine Batterie. Dann können sie eine erhöhte Verarbeitung, Übertragungsfähigkeit und Reichweite aufweisen.



MFRC522

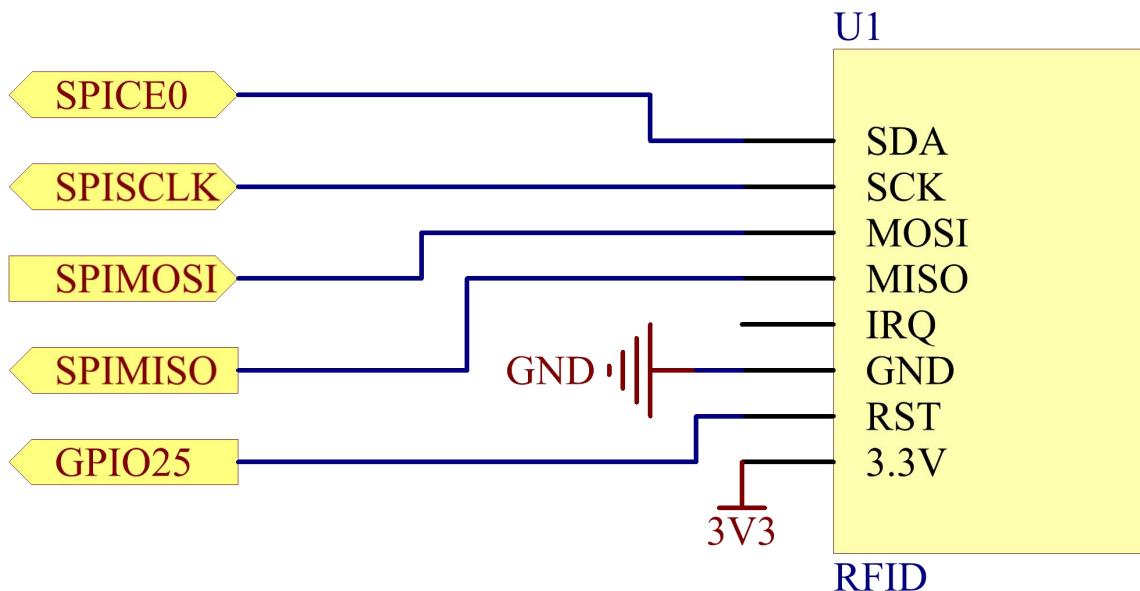
MFRC522 ist eine Art integrierter Lese- und Schreibkartenchip. Es wird üblicherweise im Radio bei 13,56 MHz verwendet. Es wurde von der NXP Company eingeführt und ist ein berührungsloser, kostengünstiger und kleiner Niederspannungs-Kartenchip, der die beste Wahl für intelligente Instrumente und tragbare Handheld-Geräte darstellt.

Der MF RC522 verwendet ein fortschrittliches Modulations- und Demodulationskonzept, das in allen Arten von passiven kontaktlosen Kommunikationsmethoden und -protokollen mit 13,56 MHz vollständig dargestellt wird. Darüber hinaus unterstützt es den schnellen CRYPTO1-

Verschlüsselungsalgorithmus zur Überprüfung von MIFARE-Produkten. Der MFRC522 unterstützt auch die berührungslose Hochgeschwindigkeitskommunikation der MIFARE-Serie mit einer bidirektionalen Datenübertragungsrate von bis zu 424 kbit / s. Als neues Mitglied der hochintegrierten 13,56-MHz-Lesekartenserie ist der MF RC522 dem vorhandenen MF RC500 und MF RC530 sehr ähnlich, es gibt jedoch auch große Unterschiede. Es kommuniziert mit dem Host-Computer über die serielle Art und Weise, die weniger Verkabelung erfordert. Sie können zwischen SPI-, I2C- und seriellem UART-Modus (ähnlich wie RS232) wählen, um die Verbindung zu reduzieren, Platz auf der Leiterplatte zu sparen (kleinere Größe) und Kosten zu senken.

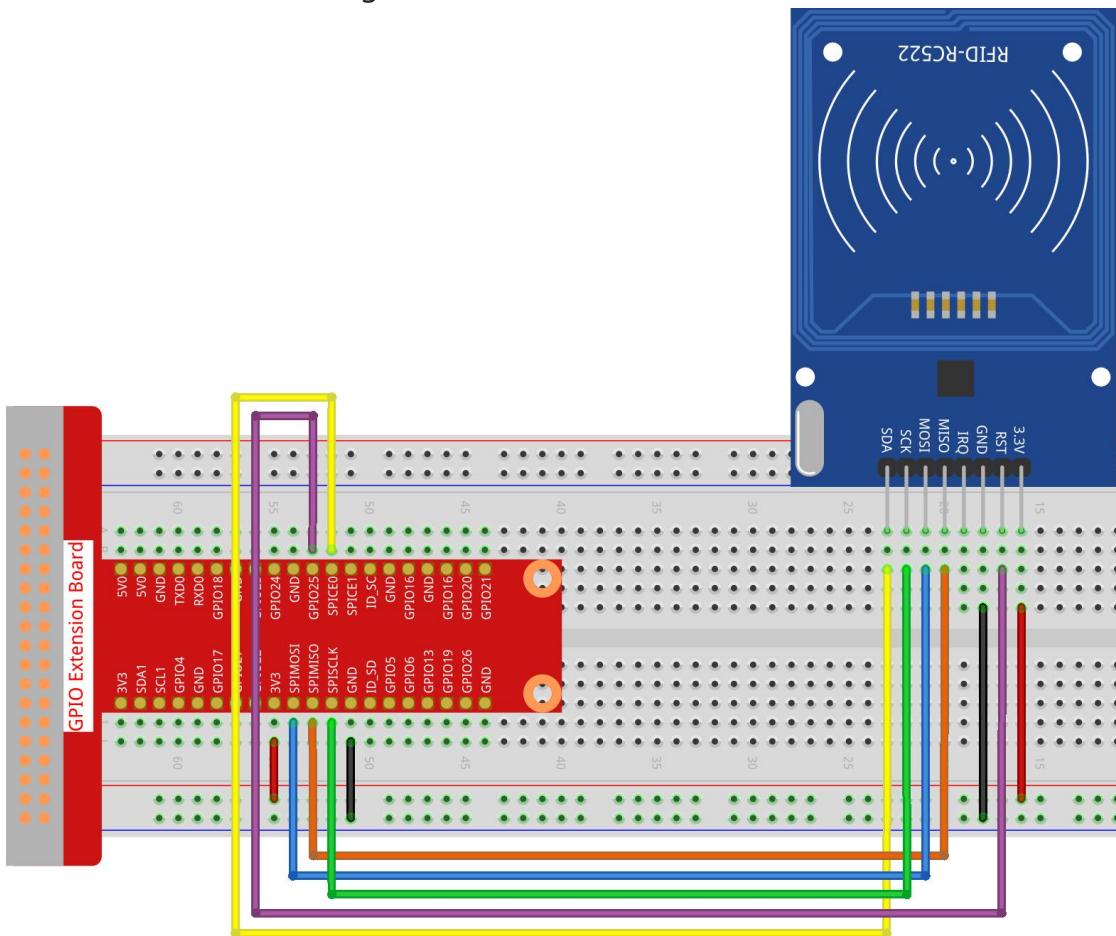
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
SPICE0	Pin 24	10	8
SPISCLK	Pin 23	14	11
SPIMOSI	Pin 19	12	10
SPIMISO	Pin 21	13	9
GPIO25	Pin 22	6	25



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



Schritt 2: Richten Sie SPI ein (weitere Informationen finden Sie im Anhang. Wenn Sie SPI eingestellt haben, überspringen Sie diesen Schritt.)

➤ Für Benutzer in C-Sprache

Schritt 3: Gehen Sie zum Ordner der Kode.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/2.2.7/
```

Schritt 4: Kompilieren Sie die Kode

lesen lassen
schreiben lassen

Hinweis: Es gibt zwei Beispiele, mit denen Sie die Karten-ID lesen oder schreiben können. Sie können je nach Bedarf eines davon auswählen.

Schritt 5: Führen Sie die ausführbare Datei aus.

```
sudo ./read  
sudo ./write
```

Kode Erklärung

```
InitRc522();
```

Mit dieser Funktion wird das RFID RC522-Modul initialisiert.

```
uint8_t read_card_data();
```

Diese Funktion wird zum Lesen der Daten der Karte verwendet. Wenn der Lesevorgang erfolgreich ist, wird "1" zurückgegeben.

```
uint8_t write_card_data(uint8_t *data);
```

Diese Funktion wird zum Schreiben der Kartendaten verwendet und gibt "1" zurück, wenn der Schreibvorgang erfolgreich ist. * Daten sind die Informationen, die auf die Karte geschrieben werden.

➤ Für Python-Sprachbenutzer

Schritt 3: Gehen Sie zum Ordner der Kode.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/2.2.7
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo python3 2.2.7_read.py  
sudo python3 2.2.7_write.py
```

Hinweis: Es gibt zwei Beispiele, mit denen Sie die Karten-ID lesen oder schreiben können. Sie können je nach Bedarf eines davon auswählen.

Kode Erklärung

```
RC522()
```

Instanziieren Sie die rc522-Klasse.

```
RC522.Pcd_start()
```

RFID initialisieren

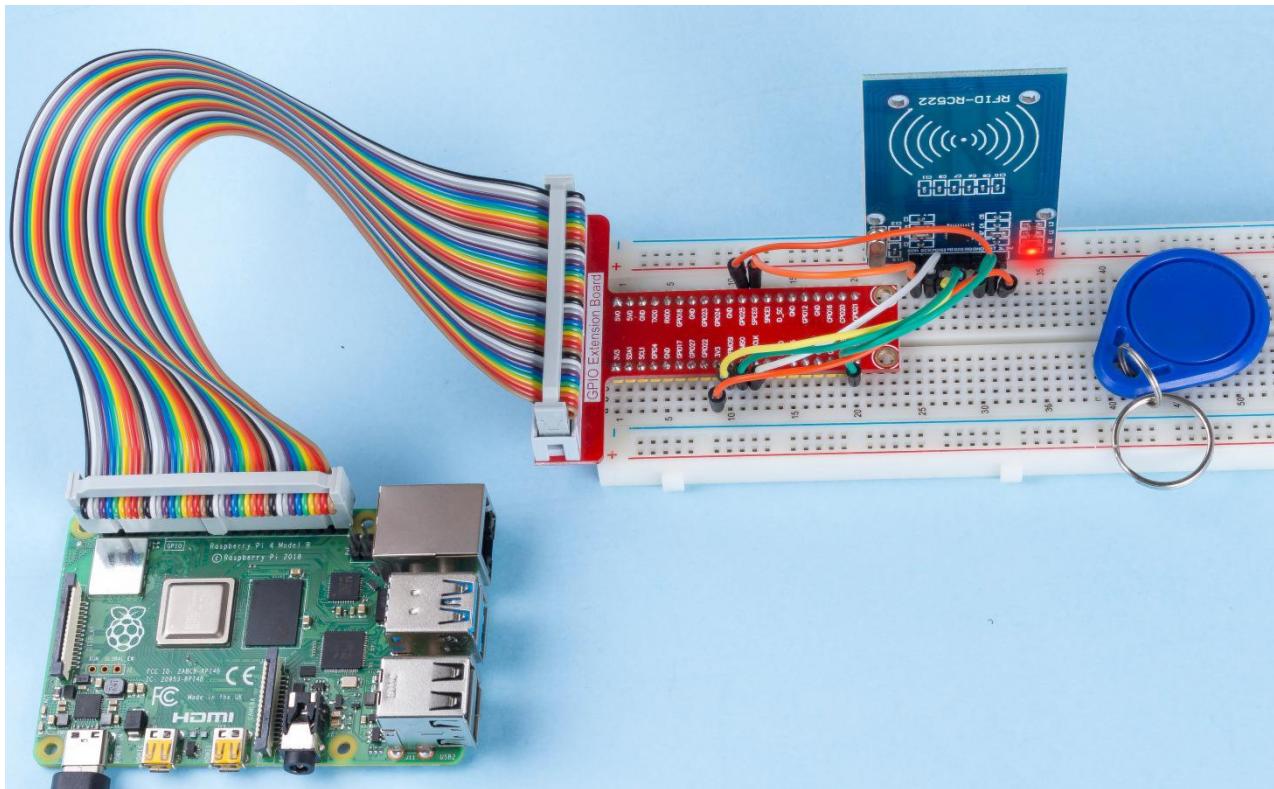
```
RC522.read_card_data(addr)
```

Mit dieser Funktion werden die Kartendaten gelesen. Wenn das Lesen erfolgreich ist, wird "1" zurückgegeben. addr ist die Adresse der Karte.

RC522.write_card_data(addr, data)

Diese Funktion wird zum Schreiben von Kartendaten verwendet. Wenn der Schreibvorgang erfolgreich ist, wird "1" zurückgegeben. addr ist die Adresse der Karte und data sind die Informationen, die auf die Karte geschrieben werden sollen.

Phänomen Bild



3 Erweiterung

Dieses Kapitel zeigt einige sehr lustige Expansionsexperimente. Folgende Punkte sind zu beachten:

- 1) Die Kode und die Verkabelung sind viel komplizierter, und Sie benötigen mehr Geduld, um diese Experimente abzuschließen.
- 2) Die vollständige Kode wird nicht auf dem Dokument angegeben, sodass Sie in den Kodeordner gehen können, um den vollständigen Kode anzuzeigen.

3.1 Anwendung

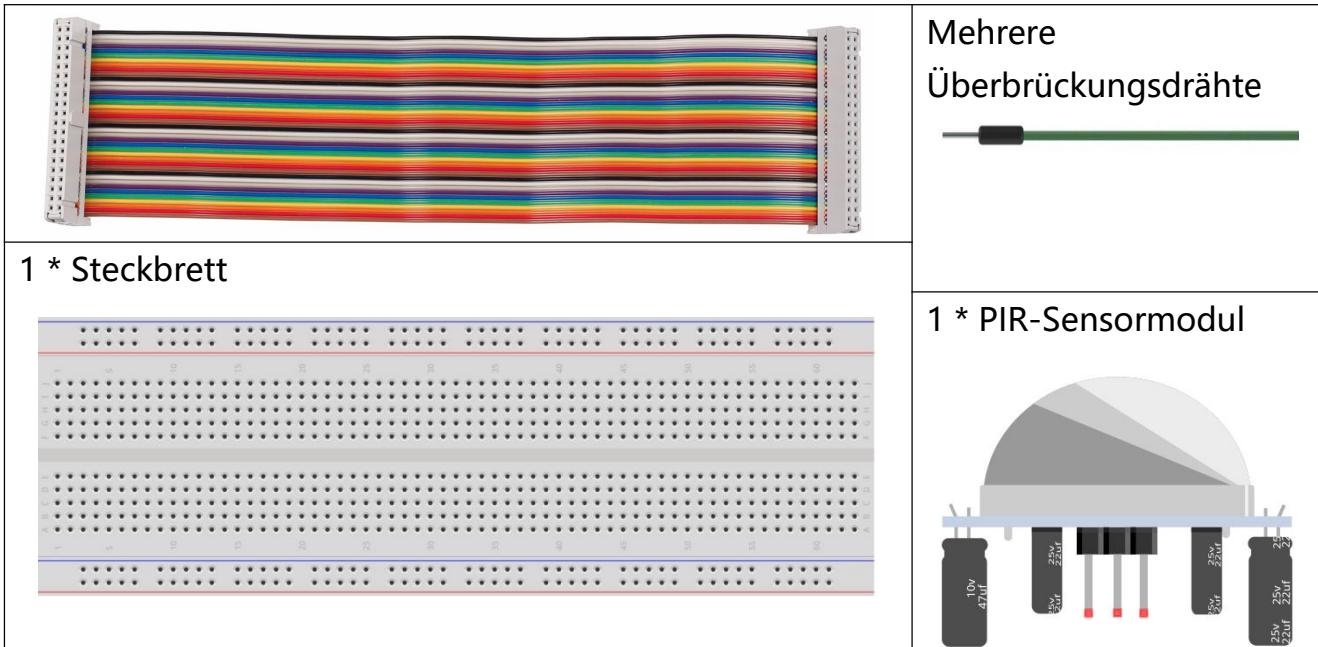
3.1.1 Zählgerät

Einführung

Hier werden wir ein Zählersystem mit Nummeranzeige herstellen, das aus einem PIR-Sensor und einer 4-stelligen Segmentanzeige besteht. Wenn der PIR feststellt, dass jemand vorbeikommt, addiert die Nummer auf der 4-stelligen Segmentanzeige 1. Mit diesem Zähler können Sie die Anzahl der Personen zählen, die durch den Durchgang gehen.

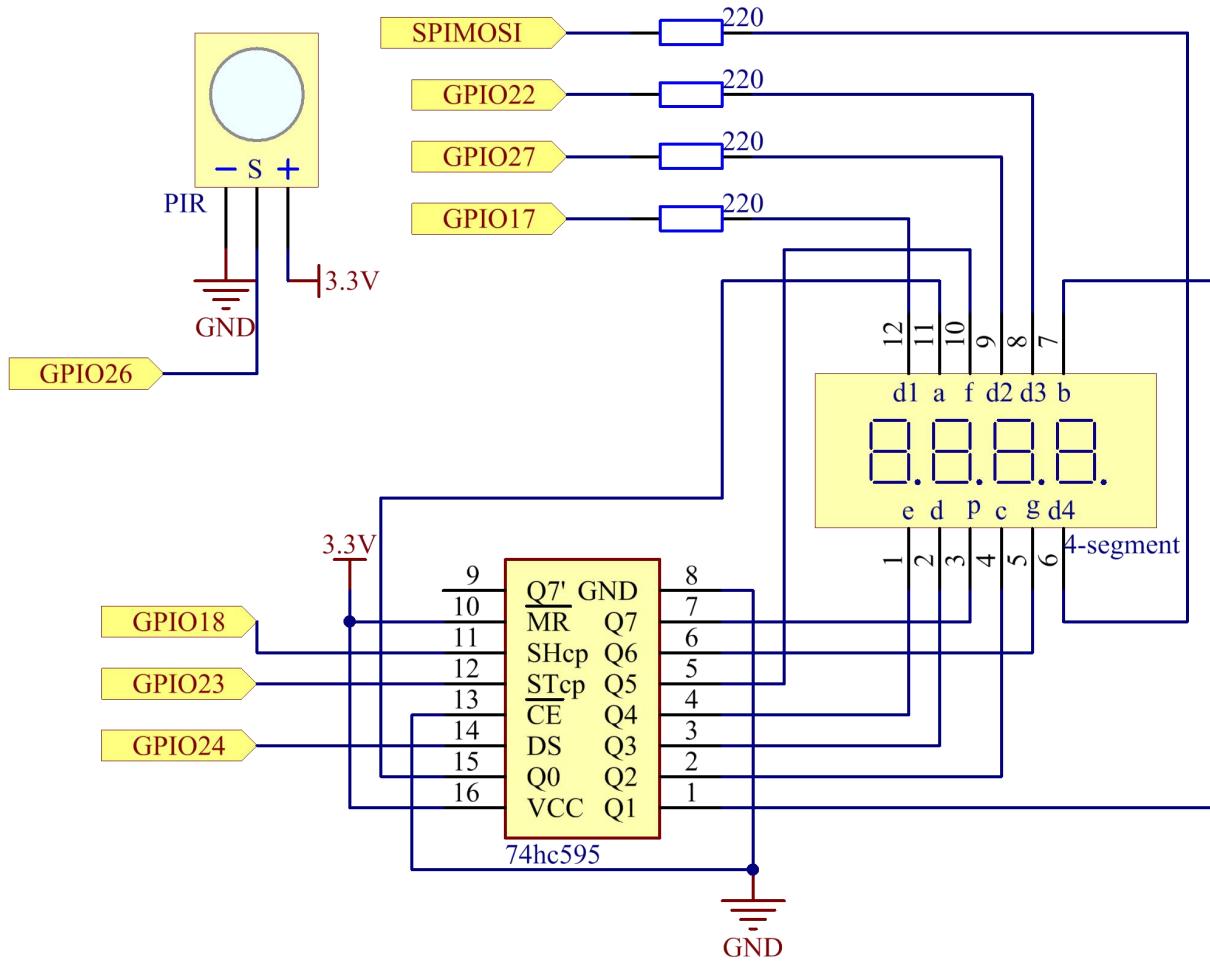
Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * 4-stellige 7-Segment-Anzeige																																								
	 <table border="1"> <tr><td>3V3</td><td>5V0</td></tr> <tr><td>GPIO2</td><td>5V0</td></tr> <tr><td>GND</td><td>GND</td></tr> <tr><td>GPIO4</td><td>TX00</td></tr> <tr><td>GND</td><td>RX00</td></tr> <tr><td>GPIO17</td><td>GPIO18</td></tr> <tr><td>GPIO27</td><td>GND</td></tr> <tr><td>GPIO22</td><td>GPIO23</td></tr> <tr><td>3V3</td><td>GPIO24</td></tr> <tr><td>SPIMOSI</td><td>GND</td></tr> <tr><td>SPIMISO</td><td>GPIO25</td></tr> <tr><td>SPISCLK</td><td>SPICE0</td></tr> <tr><td>GND</td><td>SPICE1</td></tr> <tr><td>ID_SD</td><td>ID_SC</td></tr> <tr><td>GPIO5</td><td>GND</td></tr> <tr><td>GPIO6</td><td>GPIO16</td></tr> <tr><td>GPIO13</td><td>GND</td></tr> <tr><td>GPIO19</td><td>GPIO16</td></tr> <tr><td>GPIO26</td><td>GPIO20</td></tr> <tr><td>GND</td><td>GPIO21</td></tr> </table>	3V3	5V0	GPIO2	5V0	GND	GND	GPIO4	TX00	GND	RX00	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GPIO24	SPIMOSI	GND	SPIMISO	GPIO25	SPISCLK	SPICE0	GND	SPICE1	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	
3V3	5V0																																									
GPIO2	5V0																																									
GND	GND																																									
GPIO4	TX00																																									
GND	RX00																																									
GPIO17	GPIO18																																									
GPIO27	GND																																									
GPIO22	GPIO23																																									
3V3	GPIO24																																									
SPIMOSI	GND																																									
SPIMISO	GPIO25																																									
SPISCLK	SPICE0																																									
GND	SPICE1																																									
ID_SD	ID_SC																																									
GPIO5	GND																																									
GPIO6	GPIO16																																									
GPIO13	GND																																									
GPIO19	GPIO16																																									
GPIO26	GPIO20																																									
GND	GPIO21																																									
1 * 40-Pin Kabel		4 * Widerstand (220 Ω)																																								
		1 * 74HC595																																								



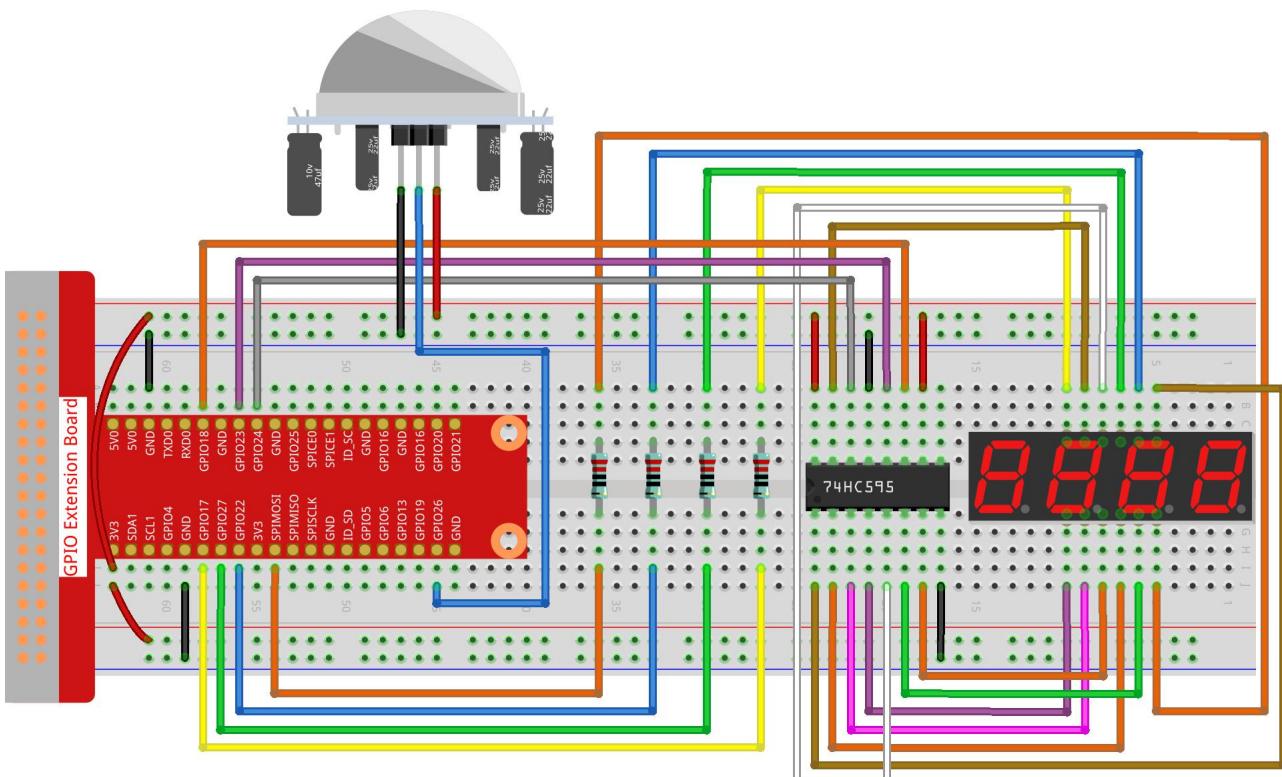
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22
SPIMOSI	Pin 19	12	10
GPIO18	Pin 12	1	18
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24
GPIO26	Pin 37	25	26



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.1/
```

Schritt 3: Kompilieren Sie die Kode

```
gcc 3.1.1_CountingDevice.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Wenn der PIR nach dem Ausführen der Kode feststellt, dass jemand vorbeikommt, addiert die Nummer auf der 4-stelligen Segmentanzeige 1.

Kode Erklärung

```
void display()
{
    clearDisplay();
    pickDigit(0);
    hc595_shift(number[counter % 10]);

    clearDisplay();
    pickDigit(1);
    hc595_shift(number[counter % 100 / 10]);

    clearDisplay();
    pickDigit(2);
    hc595_shift(number[counter % 1000 / 100]);

    clearDisplay();
    pickDigit(3);
    hc595_shift(number[counter % 10000 / 1000]);
}
```

Starten Sie zuerst die vierte Segmentanzeige und schreiben Sie die einstellige Nummer. Dann starten Sie die Anzeige des dritten Segments und geben Sie die Zehnerstelle ein. Starten Sie danach die zweite bzw. die erste Segmentanzeige und schreiben Sie die Hunderter- bzw. Tausenderstellen. Da die Aktualisierungsgeschwindigkeit sehr hoch ist, sehen wir eine vollständige vierstellige Anzeige.

```
void loop(){
```

```

int currentState =0;
int lastState=0;
while(1){
    display();
    currentState=digitalRead(sensorPin);
    if((currentState==0)&&(lastState==1)){
        counter +=1;
    }
    lastState=currentState;
}

```

Dies ist die Hauptfunktion: Zeigen Sie die Nummer auf der 4-stelligen Segmentanzeige an und lesen Sie den PIR-Wert. Wenn der PIR feststellt, dass jemand vorbeikommt, addiert die Nummer auf der 4-stelligen Segmentanzeige 1.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 3.1.1_CountingDevice.py
```

Wenn der PIR nach dem Ausführen der Kode feststellt, dass jemand vorbeikommt, addiert die Nummer auf der 4-stelligen Segmentanzeige 1.

Kode Erklärung

Basierend auf der **1.1.5 4-stelligen 7-Segment-Anzeige** wird in dieser Lektion ein PIR-Modul hinzugefügt, um die automatische Zählung von Lektion 1.1.5 in Zählerkennung zu ändern. Wenn der PIR feststellt, dass jemand vorbeikommt, addiert die Nummer auf der 4-stelligen Segmentanzeige 1.

```

def display():
    global counter
    clearDisplay()
    pickDigit(0)
    hc595_shift(number[counter % 10])

    clearDisplay()
    pickDigit(1)
    hc595_shift(number[counter % 100//10])

```

```
clearDisplay()  
pickDigit(2)  
hc595_shift(number[counter % 1000//100])
```

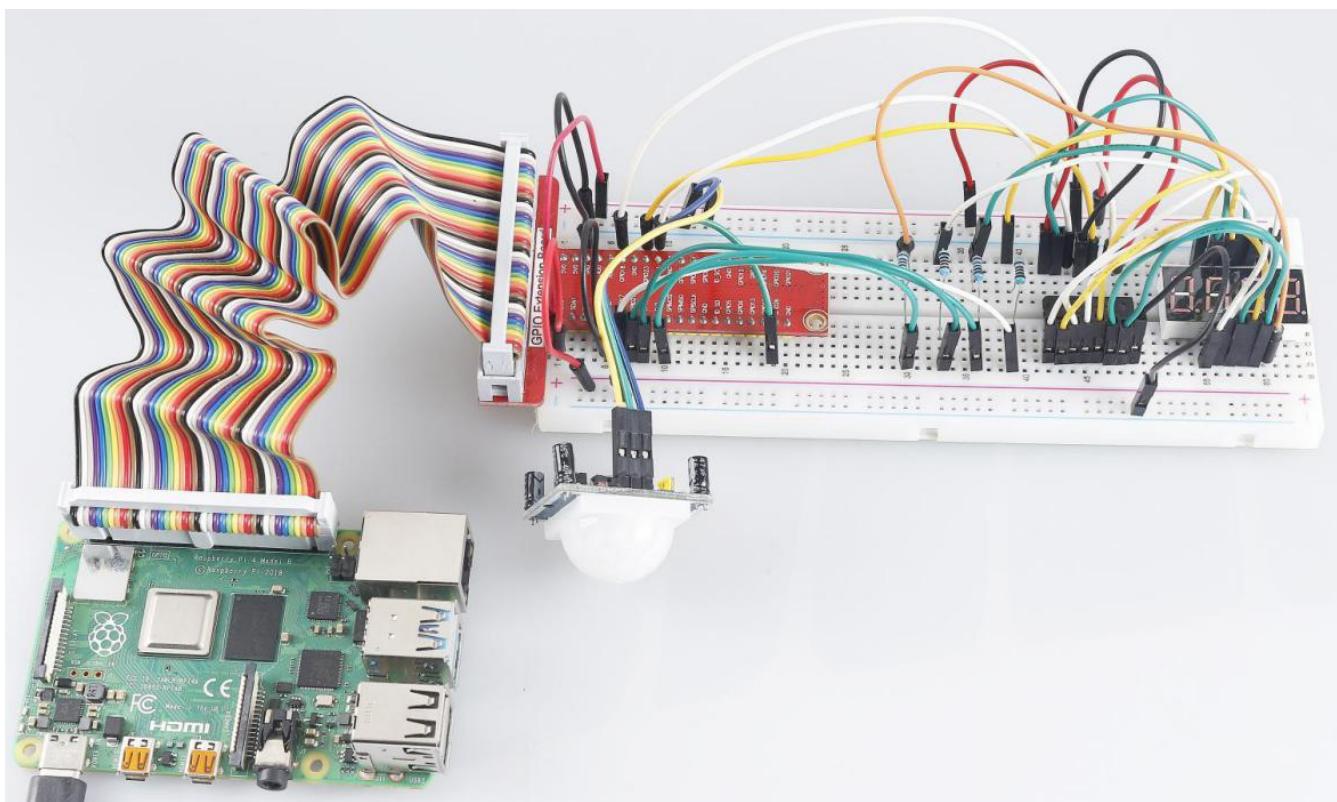
```
clearDisplay()  
pickDigit(3)  
hc595_shift(number[counter % 10000//1000])
```

Starten Sie zuerst die vierte Segmentanzeige und schreiben Sie die einstellige Nummer. Dann starten Sie die Anzeige des dritten Segments und geben Sie die Zehnerstelle ein. Starten Sie danach die zweite bzw. die erste Segmentanzeige und schreiben Sie die Hunderter- bzw. Tausenderstellen. Da die Aktualisierungsgeschwindigkeit sehr hoch ist, sehen wir eine vollständige vierstellige Anzeige.

```
def loop():  
    global counter  
    currentState = 0  
    lastState = 0  
    while True:  
        display()  
        currentState=GPIO.input(sensorPin)  
        if (currentState == 0) and (lastState == 1):  
            counter +=1  
        lastState=currentState
```

Dies ist die Hauptfunktion: Zeigen Sie die Nummer auf der 4-stelligen Segmentanzeige an und lesen Sie den PIR-Wert. Wenn der PIR feststellt, dass jemand vorbeikommt, addiert die Nummer auf der 4-stelligen Segmentanzeige 1.

Phänomen Bild

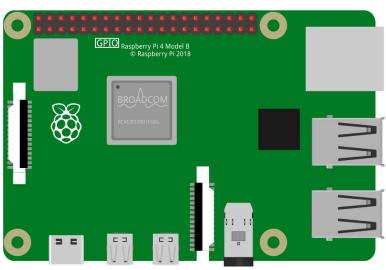
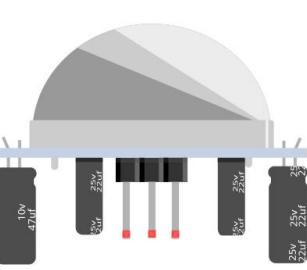
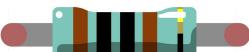


3.1.2 Willkommen

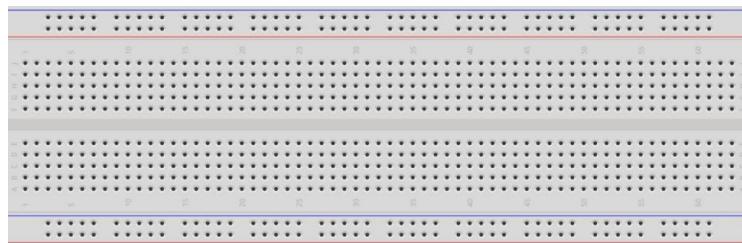
Einführung

In diesem Projekt werden wir PIR verwenden, um die Bewegung von Fußgängern zu erfassen, und Servos, LED und Summer verwenden, um die Arbeit der Sensortür des Supermarkts zu simulieren. Wenn der Fußgänger innerhalb des Erfassungsbereichs des PIR erscheint, leuchtet die Anzeigelampe, die Tür wird geöffnet und der Summer ertönt die Öffnungsglocke.

Komponenten

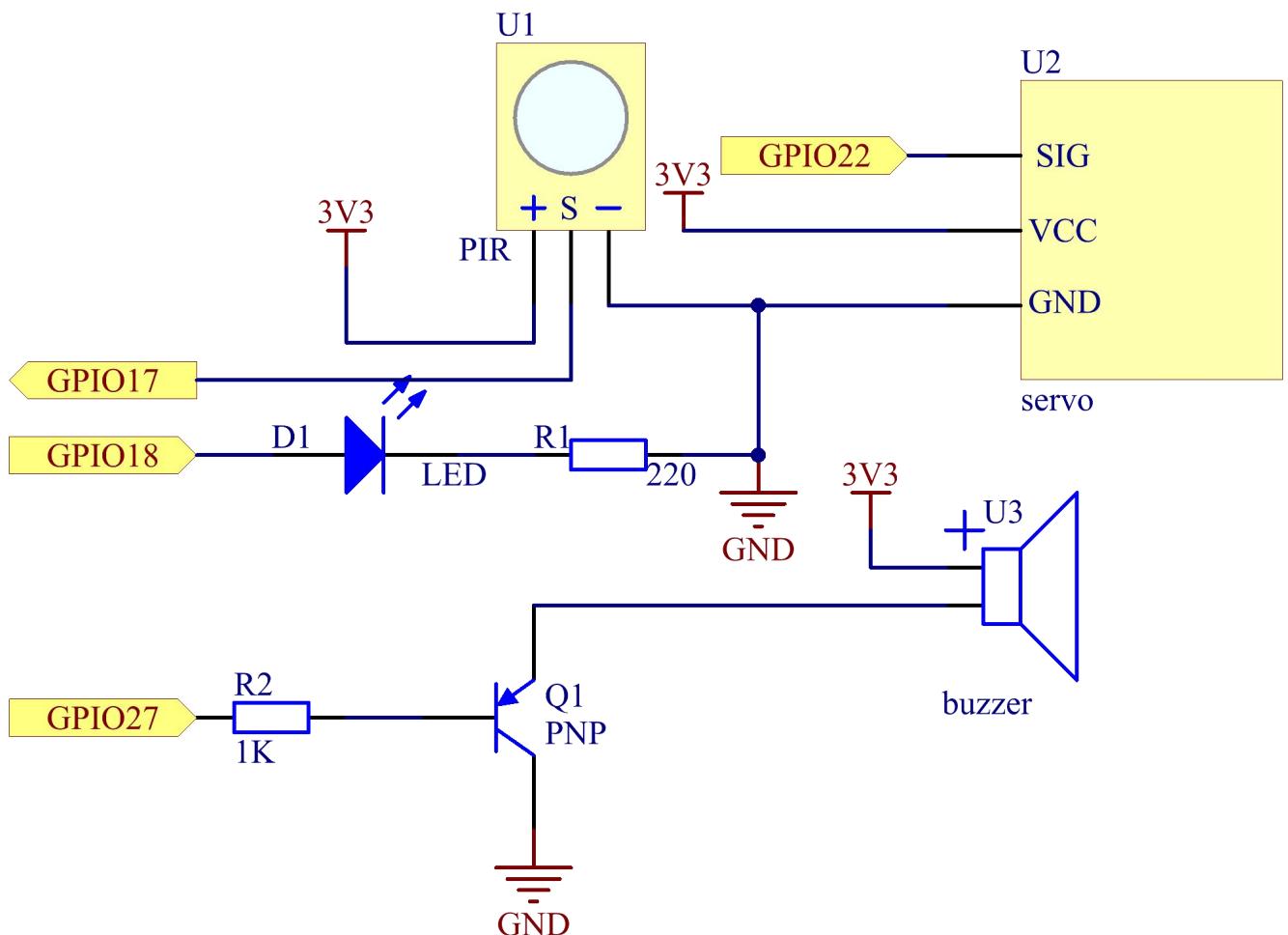
1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * PIR
		
1 * Servo	1 * Passiver Summer	1 * S8550 PNP-Transistor
		
1 * Widerstand 1 kΩ	1 * Widerstand 220Ω	Mehrere Überbrückungsdrähte
		
1 * 40-Pin Kabel		

1 * Steckbrett



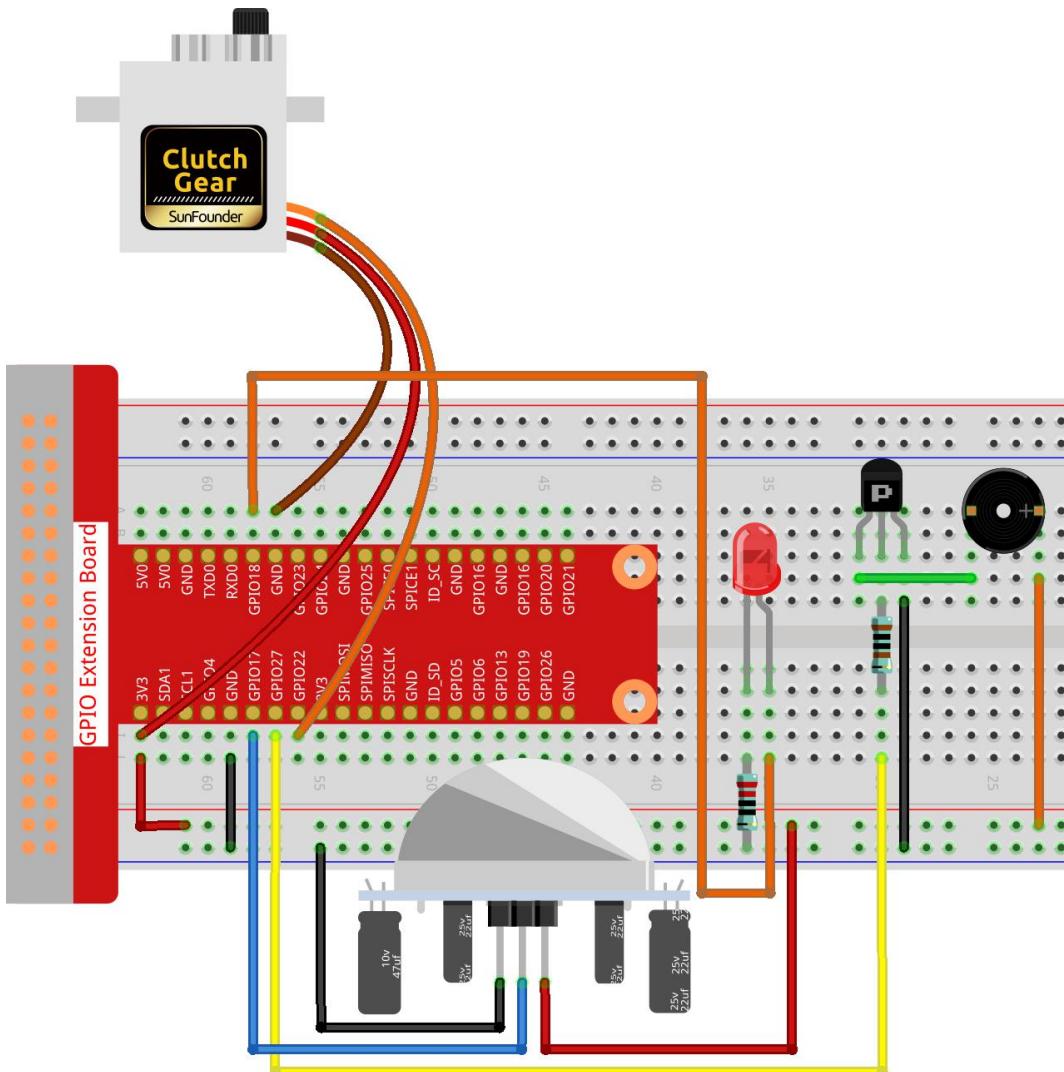
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO18	Pin 12	1	18
GPIO17	Pin 11	0	17
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.2/
```

Schritt 3: Kompilieren.

```
gcc 3.1.2_Welcome.c -lwiringPi
```

Schritt 4: Ausführen.

```
sudo ./a.out
```

Wenn der PIR-Sensor nach dem Ausführen der Kode jemanden erkennt, der vorbeikommt, öffnet sich die Tür automatisch (vom Servo simuliert), schaltet die Anzeige ein und spielt die Türklingelmusik ab. Nachdem die Türklingelmusik

abgespielt wurde, schließt das System automatisch die Tür und schaltet die Anzeigelampe aus, um auf das nächste Mal zu warten, wenn jemand vorbeikommt.

Kode Erklärung

```
void setAngle(int pin, int angle){    //Create a function to control the angle of the servo.
    if(angle < 0)
        angle = 0;
    if(angle > 180)
        angle = 180;
    softPwmWrite(pin,Map(angle, 0, 180, 5, 25));
}
```

Erstellen Sie eine Funktion, setAngle, um den Winkel im Servo von 0-180 zu schreiben.

```
void doorbell(){
    for(int i=0;i<sizeof(song)/4;i++){
        softToneWrite(BuzPin, song[i]);
        delay(beat[i] * 250);
}
```

Erstellen Sie eine Funktion, Türklingel, damit der Summer Musik abspielen kann.

```
void closedoor(){
    digitalWrite(ledPin, LOW);    //led off
    for(int i=180;i>-1;i--){
        setAngle(servoPin,i);
        delay(1);
    }
}
```

Erstellen Sie eine Closedoor-Funktion, um das Schließen der Tür zu simulieren, schalten Sie die LED aus und lassen Sie das Servo von 180 Grad auf 0 Grad drehen.

```
void opendoor(){
    digitalWrite(ledPin, HIGH);    //led on
    for(int i=0;i<181;i++){
        setAngle(servoPin,i);
        delay(1);
    }
    doorbell();
    closedoor();
```

}

Die Funktion opendoor () besteht aus mehreren Teilen: Schalten Sie die Anzeigelampe ein, schalten Sie das Servo ein (simulieren Sie das Öffnen der Tür), spielen Sie die Klingelmusik des Supermarkts und rufen Sie nach dem Abspielen der Musik die Funktion geschlossene Tür () auf.

```
int main(void)
{
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    if(softToneCreate(BuzPin) == -1){
        printf("setup softTone failed !");
        return 1;
    }
    .....
}
```

Initialisieren Sie in der Funktion main () die Bibliothek wiringPi und richten Sie softTone ein. Setzen Sie dann ledPin in den Ausgabestatus und pirPin in den Eingabestatus. Wenn der PIR-Sensor jemanden erkennt, der vorbeikommt, wird die Funktion opendoor aufgerufen, um das Öffnen der Tür zu simulieren.

➤ Für Python-Sprachbenutzer

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Ausführen.

```
sudo python3 3.1.2_Welcome.py
```

Wenn der PIR-Sensor nach dem Ausführen der Kode jemanden erkennt, der vorbeikommt, öffnet sich die Tür automatisch (vom Servo simuliert), schaltet die Anzeige ein und spielt die Türklingelmusik ab. Nachdem die Türklingelmusik abgespielt wurde, schließt das System automatisch die Tür und schaltet die Anzeigelampe aus, um auf das nächste Mal zu warten, wenn jemand vorbeikommt.

Kode Erklärung

```
def setup():
    global p
    global Buzz
    # Assign a global variable to replace GPIO.PWM
    GPIO.setmode(GPIO.BCM)      # Numbers GPIOs by physical location
```

```

GPIO.setup(ledPin, GPIO.OUT)      # Set ledPin's mode is output
GPIO.setup(pirPin, GPIO.IN)       # Set sensorPin's mode is input
GPIO.setup(buzPin, GPIO.OUT)      # Set pins' mode is output
Buzz = GPIO.PWM(buzPin, 440)     # 440 is initial frequency.
Buzz.start(50)                  # Start Buzzer pin with 50% duty ration
GPIO.setup(servoPin, GPIO.OUT)    # Set servoPin's mode is output
GPIO.output(servoPin, GPIO.LOW)   # Set servoPin to low
p = GPIO.PWM(servoPin, 50)       # set Freqeuce to 50Hz
p.start(0)                      # Duty Cycle = 0

```

Diese Anweisungen werden verwendet, um die Pins jeder Komponente zu initialisieren.

```

def setAngle(angle):          # make the servo rotate to specific angle (0-180 degrees)
    angle = max(0, min(180, angle))
    pulse_width = map(angle, 0, 180, SERVO_MIN_PULSE, SERVO_MAX_PULSE)
    pwm = map(pulse_width, 0, 20000, 0, 100)
    p.ChangeDutyCycle(pwm)#map the angle to duty cycle and output it

```

Erstellen Sie eine Funktion von servowrite, um den Winkel in das Servo zu schreiben, der 0-180 ist.

```

def doorbell():
    for i in range(1, len(song)):      # Play song 1
        Buzz.ChangeFrequency(song[i])  # Change the frequency along the song note
        time.sleep(beat[i] * 0.25)     # delay a note for beat * 0.25s

```

Erstellen Sie eine Funktion, Türklingel, damit der Summer Musik abspielen kann.

```

def closedoor():
    GPIO.output(ledPin, GPIO.LOW)
    Buzz.ChangeFrequency(1)
    for i in range(180, -1, -1): #make servo rotate from 180 to 0 deg
        setAngle(i)
        time.sleep(0.001)

```

Schließen Sie die Tür und schalten Sie die Kontrollleuchte aus.

```

def opendoor():
    GPIO.output(ledPin, GPIO.LOW)
    for i in range(0, 181, 1):  #make servo rotate from 0 to 180 deg

```

```

setAngle(i)      # Write to servo
time.sleep(0.001)
doorbell()
closedoor()

```

Die Funktion opendoor () besteht aus mehreren Teilen: Schalten Sie die Anzeigelampe ein, schalten Sie das Servo ein (um das Öffnen der Tür zu simulieren), spielen Sie die Klingelmusik des Supermarkts und rufen Sie nach dem Spielen die Funktion Closedoor() auf Musik.

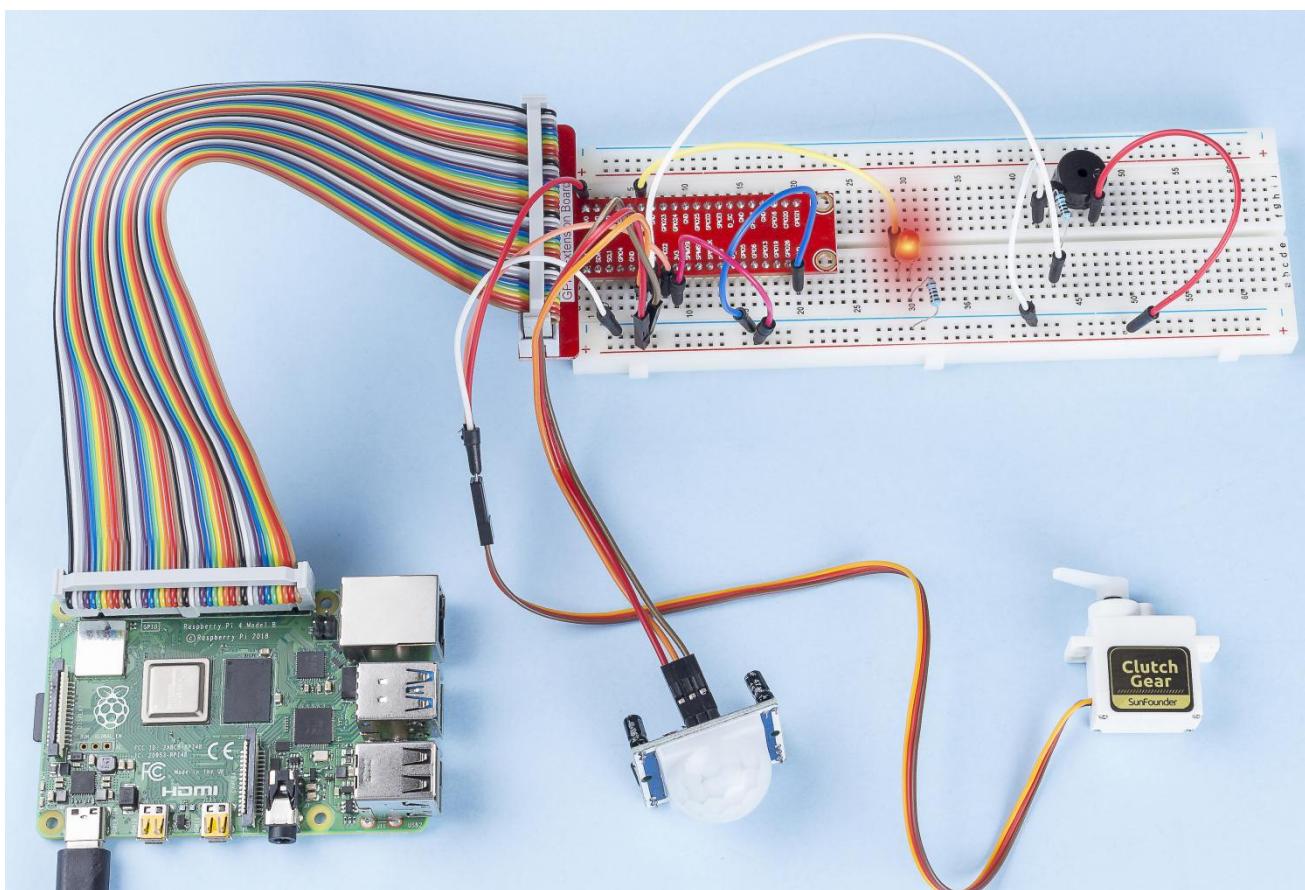
```

def loop():
while True:
    if GPIO.input(pirPin)==GPIO.HIGH:
        opendoor()

```

Wenn RIP erkennt, dass jemand vorbeikommt, ruft es die Funktion opendoor() auf.

Phänomen Bild



3.1.3 Alarm umkehren

Einführung

In diesem Projekt werden wir LCD-, Summer- und Ultraschallsensoren verwenden, um ein Rückwärtshilfesystem herzustellen. Wir können es auf das ferngesteuerte Fahrzeug setzen, um den tatsächlichen Vorgang des Rückwärtsfahrens des Autos in die Garage zu simulieren.

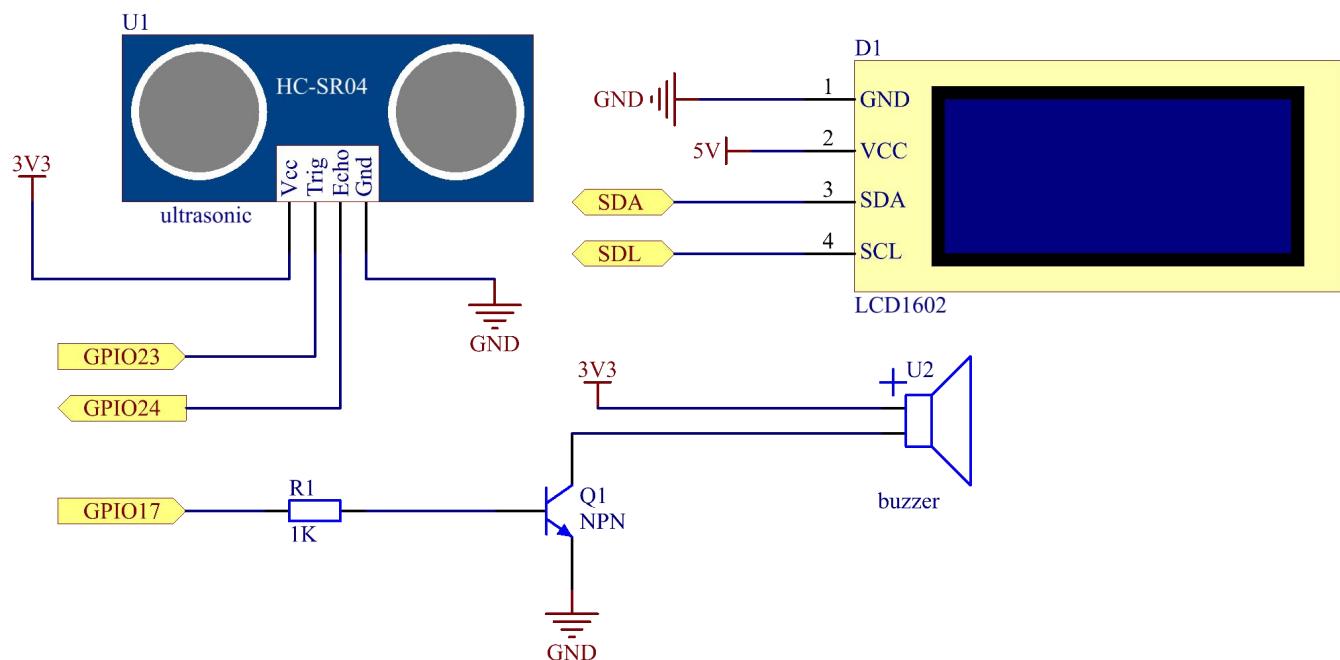
Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Aktiver Summer
	 GPIO Extension Board Pinout: 3V3 SDA1 SVO GND GND SCL1 GND TxD0 TXD0 GND RXD0 GPIO17 GPIO18 GPIO27 GND 3V GND GND GND SPIMOSI GND SPIMISO GPIO25 SPISCLK SPICE0 GND SPICE1 ID_SD ID_SD GND GND GPIO5 GPIO16 GPIO13 GND GPIO19 GPIO16 GPIO26 GPIO20 GND GPIO21	
1 * HC SR04	1 * I2C LCD1602	1 * S8050 NPN-Transistor
1 * 40-Pin Kabel		Mehrere Überbrückungsdrähte
1 * Steckbrett		1 * Widerstand (1 kΩ)

Schematische Darstellung

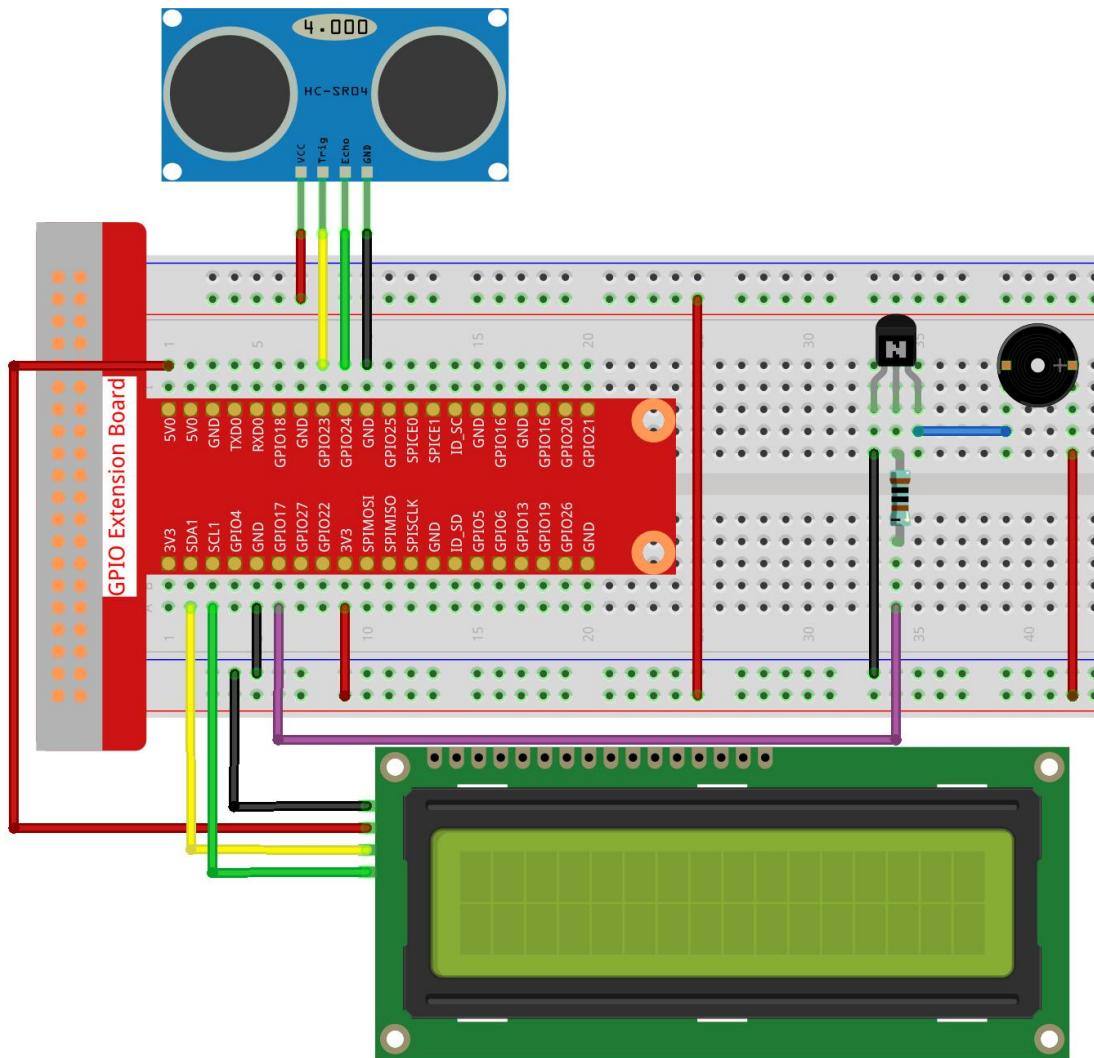
Der Ultraschallsensor erkennt den Abstand zwischen sich und dem Hindernis, der in Form einer Kode auf dem LCD angezeigt wird. Gleichzeitig ließ der Ultraschallsensor den Summer einen sofortigen Ton unterschiedlicher Frequenz je nach Entfernungswert ausgeben.

T-Karte Name	physisch	wiringPi	BCM
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24
GPIO17	Pin 11	0	17
SDA1	Pin 3		
SCL1	Pin 5		



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.3/
```

Schritt 3: Kompilieren.

gcc 3.1.3_ReversingAlarm.c -lwiringPi

Schritt 4: Ausführen.

sudo ./a.out

Während die Kode ausgeführt wird, erkennt das Ultraschallsensormodul die Entfernung zum Hindernis und zeigt dann die Informationen zur Entfernung auf dem LCD1602 an. Außerdem gibt der Summer einen Warnton aus, dessen Frequenz sich mit der Entfernung ändert.

Kode

Hinweis: Die folgenden der Kode sind unvollständig. Wenn Sie die vollständigen Kode überprüfen möchten, wird empfohlen, den Befehl nano 3.1.1_ReversingAlarm.c zu verwenden.

```
#include <wiringPi.h>
#include <stdio.h>
#include <sys/time.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <string.h>

#define Trig    4
#define Echo    5
#define Buzzer  0

int LCDAddr = 0x27;
int BLEN = 1;
int fd;

//here is the function of LCD
void write_word(int data){.....}

void send_command(int comm){.....}

void send_data(int data){.....}

void lcdInit(){.....}

void clear(){.....}

void write(int x, int y, char data[]){.....}

//here is the function of Ultrasonic
void ultraInit(void){.....}

float disMeasure(void){.....}

//here is the main function
int main(void)
```

```
{
    float dis;
    char result[10];
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }

    pinMode(Buzzer,OUTPUT);
    fd = wiringPiI2CSetup(LCDAddr);
    lcdInit();
    ultraInit();

    clear();
    write(0, 0, "Ultrasonic Starting");
    write(1, 1, "By Sunfounder");

    while(1){
        dis = disMeasure();
        printf("%.2f cm \n",dis);
        digitalWrite(Buzzer,LOW);
        if (dis > 400){
            clear();
            write(0, 0, "Error");
            write(3, 1, "Out of range");
            delay(500);
        }
        else
        {
            clear();
            write(0, 0, "Distance is");
            sprintf(result,"% .2f cm",dis);
            write(5, 1, result);

            if(dis>=50)
            {delay(500);}
            else if(dis<50 & dis>20) {
                for(int i=0;i<2;i++){
                    digitalWrite(Buzzer,HIGH);
                    delay(50);
                }
            }
        }
    }
}
```

```

        digitalWrite(Buzzer,LOW);
        delay(200);
    }
}

else if(dis<=20){
    for(int i=0;i<5;i++){
        digitalWrite(Buzzer,HIGH);
        delay(50);
        digitalWrite(Buzzer,LOW);
        delay(50);
    }
}

return 0;
}

```

Kode Erklärung

```

pinMode(Buzzer,OUTPUT);
fd = wiringPiI2CSetup(LCDAddr);
lcdInit();
ultraInit();

```

In diesem Programm wenden wir frühere Komponenten synthetisch an. Hier verwenden wir Summer, LCD und Ultraschall. Wir können sie auf die gleiche Weise wie zuvor initialisieren.

```

dis = disMeasure();
printf("%.2f cm \n",dis);
digitalWrite(Buzzer,LOW);
if (dis > 400){
    write(0, 0, "Error");
    write(3, 1, "Out of range");
}
else
{
    write(0, 0, "Distance is");
    sprintf(result,"% .2f cm",dis);
    write(5, 1, result);
}

```

}

Hier erhalten wir den Wert des Ultraschallsensors und die Entfernung durch Berechnung.

Wenn der Entfernungswert größer als der zu erkennende Bereichswert ist, wird eine Fehlermeldung auf dem LCD gedruckt. Wenn der Abstandswert innerhalb des Bereichs liegt, werden die entsprechenden Ergebnisse ausgegeben.

```
sprintf(result, "%.2f cm", dis);
```

Da der Ausgabemodus des LCD nur den Zeichentyp unterstützt und die Variable den Wert des Float-Typs nicht speichert, müssen wir sprintf () verwenden. Die Funktion konvertiert den Float-Typ-Wert in ein Zeichen und speichert ihn in der String-Variablen result[]. %.2f bedeutet, zwei Dezimalstellen beizubehalten.

```
if(dis>=50)
{delay(500);}
else if(dis<50 & dis>20) {
    for(int i=0;i<2;i++){
        digitalWrite(Buzzer,HIGH);
        delay(50);
        digitalWrite(Buzzer,LOW);
        delay(200);
    }
}
else if(dis<=20){
    for(int i=0;i<5;i++){
        digitalWrite(Buzzer,HIGH);
        delay(50);
        digitalWrite(Buzzer,LOW);
        delay(50);
    }
}
```

Diese Beurteilungsbedingung wird verwendet, um das Geräusch des Summers zu steuern. Je nach Entfernungsunterschied kann es in drei Fälle unterteilt werden, in denen unterschiedliche Schallfrequenzen auftreten. Da der Gesamtwert der Verzögerung 500 beträgt, können alle Fälle ein Intervall von 500 ms für den Ultraschallsensor bereitstellen.

➤ Für Python-Sprachbenutzer

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Ausführen.

```
sudo python3 3.1.3_ReversingAlarm.py
```

Während die Kode ausgeführt wird, erkennt das Ultraschallsensormodul die Entfernung zum Hindernis und zeigt dann die Informationen zur Entfernung auf dem LCD1602 an. Außerdem gibt der Summer einen Warnton aus, dessen Frequenz sich mit der Entfernung ändert.

Kode

```
import LCD1602
import time
import RPi.GPIO as GPIO

TRIG = 16
ECHO = 18
BUZZER = 11

def lcdsetup():
    LCD1602.init(0x27, 1)      # init(slave address, background light)
    LCD1602.clear()
    LCD1602.write(0, 0, 'Ultrasonic Starting')
    LCD1602.write(1, 1, 'By SunFounder')
    time.sleep(2)

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(TRIG, GPIO.OUT)
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.setup(BUZZER, GPIO.OUT, initial=GPIO.LOW)
    lcdsetup()

def distance():
    GPIO.output(TRIG, 0)
    time.sleep(0.000002)

    GPIO.output(TRIG, 1)
```

```

time.sleep(0.00001)
GPIO.output(TRIG, 0)

while GPIO.input(ECHO) == 0:
    a = 0
time1 = time.time()
while GPIO.input(ECHO) == 1:
    a = 1
time2 = time.time()

during = time2 - time1
return during * 340 / 2 * 100

def destroy():
    GPIO.output(BUZZER, GPIO.LOW)
    GPIO.cleanup()
    LCD1602.clear()

def loop():
    while True:
        dis = distance()
        print (dis, 'cm')
        print ('')
        GPIO.output(BUZZER, GPIO.LOW)
        if (dis > 400):
            LCD1602.clear()
            LCD1602.write(0, 0, 'Error')
            LCD1602.write(3, 1, 'Out of range')
            time.sleep(0.5)
        else:
            LCD1602.clear()
            LCD1602.write(0, 0, 'Distance is')
            LCD1602.write(5, 1, str(round(dis,2)) + ' cm')
            if(dis>=50):
                time.sleep(0.5)
            elif(dis<50 and dis>20):
                for i in range(0,2,1):
                    GPIO.output(BUZZER, GPIO.HIGH)
                    time.sleep(0.05)
                    GPIO.output(BUZZER, GPIO.LOW)

```

```

        time.sleep(0.2)
elif(dis<=20):
    for i in range(0,5,1):
        GPIO.output(BUZZER, GPIO.HIGH)
        time.sleep(0.05)
        GPIO.output(BUZZER, GPIO.LOW)
        time.sleep(0.05)

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

```

Kode Erklärung

```

def lcdsetup():
    LCD1602.init(0x27, 1)      # init(slave address, background light)

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(TRIG, GPIO.OUT)
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.setup(BUZZER, GPIO.OUT, initial=GPIO.LOW)
    lcdsetup()

```

In diesem Programm wenden wir die zuvor verwendeten Komponenten synthetisch an. Hier verwenden wir Summer, LCD und Ultraschall. Wir können sie auf die gleiche Weise wie zuvor initialisieren.

```

dis = distance()
print (dis, 'cm')
print ('')
GPIO.output(BUZZER, GPIO.LOW)
if (dis > 400):
    LCD1602.clear()
    LCD1602.write(0, 0, 'Error')
    LCD1602.write(3, 1, 'Out of range')

```

```

time.sleep(0.5)
else:
    LCD1602.clear()
    LCD1602.write(0, 0, 'Distance is')
    LCD1602.write(5, 1, str(round(dis,2)) + ' cm')

```

Hier erhalten wir die Werte des Ultraschallsensors und die Entfernung durch Berechnung. Wenn der Entfernungswert größer als der zu erkennende Wertebereich ist, wird eine Fehlermeldung auf dem LCD gedruckt. Und wenn der Abstand innerhalb des Arbeitsbereichs liegt, werden die entsprechenden Ergebnisse ausgegeben.

LCD1602.write(5, 1, str(round(dis,2)) + ' cm')

Da der LCD-Ausgang nur Zeichtypen unterstützt, müssen wir **str ()** verwenden, um numerische Werte in Zeichen umzuwandeln. Wir werden es auf zwei Dezimalstellen runden.

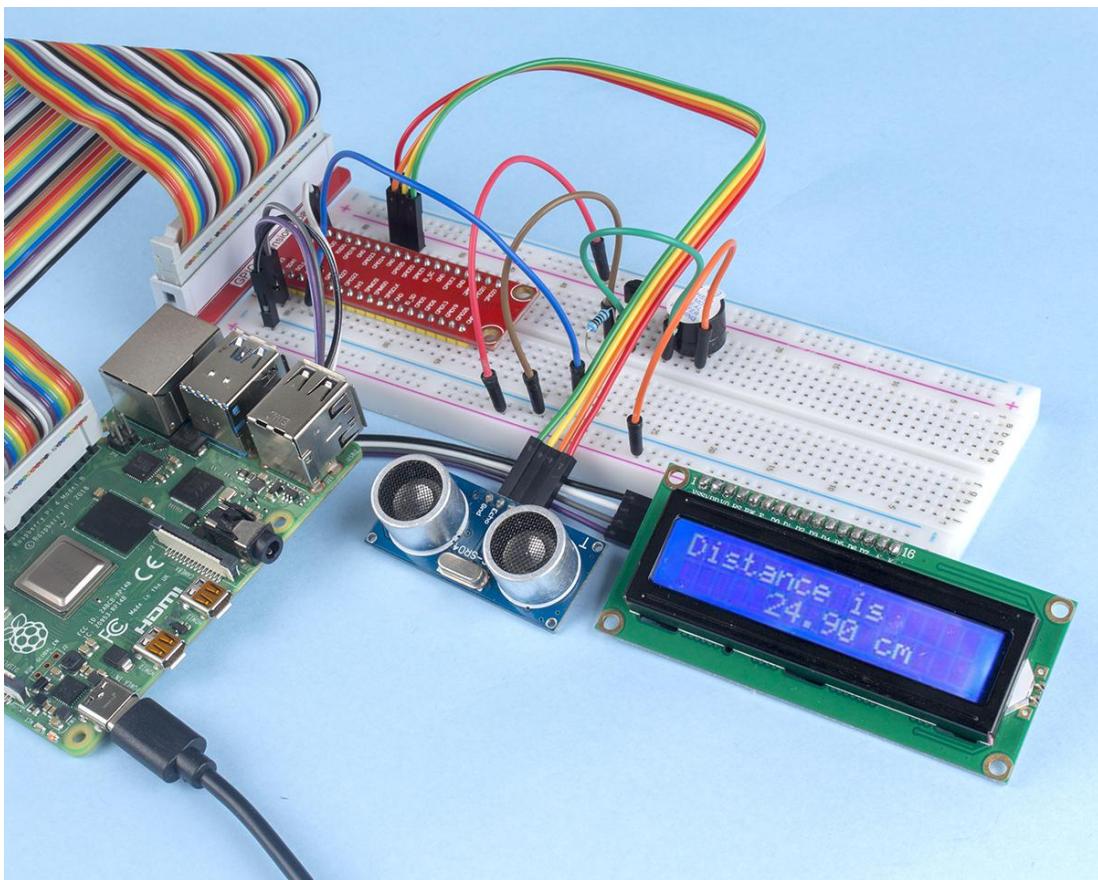
```

if(dis>=50)
{delay(500);}
else if(dis<50 & dis>20) {
    for(int i=0;i<2;i++){
        digitalWrite(Buzzer,HIGH);
        delay(50);
        digitalWrite(Buzzer,LOW);
        delay(200);
    }
}
else if(dis<=20){
    for(int i=0;i<5;i++){
        digitalWrite(Buzzer,HIGH);
        delay(50);
        digitalWrite(Buzzer,LOW);
        delay(50);
    }
}

```

Diese Beurteilungsbedingung wird verwendet, um das Geräusch des Summers zu steuern. Je nach Entfernungsunterschied kann es in drei Fälle unterteilt werden, in denen unterschiedliche Schallfrequenzen auftreten. Da der Gesamtwert der Verzögerung 500 beträgt, können alle ein Intervall von 500 ms bereitstellen, damit der Ultraschallsensor funktioniert.

Phänomen Bild

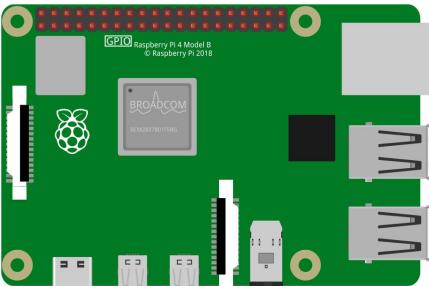
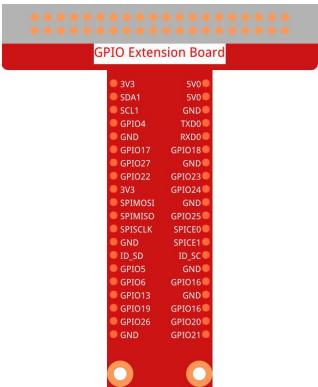
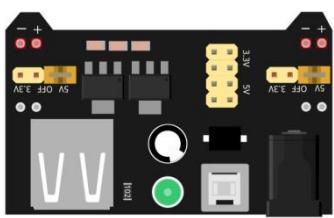
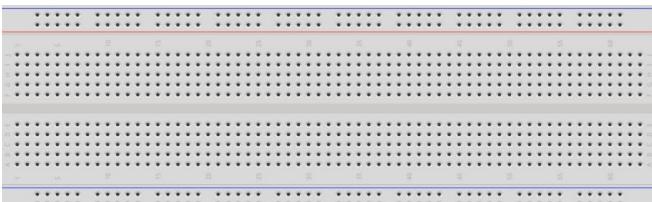
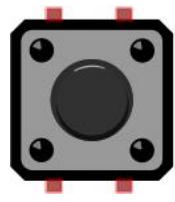
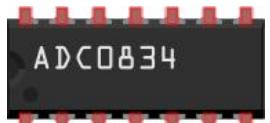
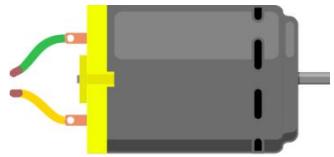


3.1.4 Smart Fan

Einführung

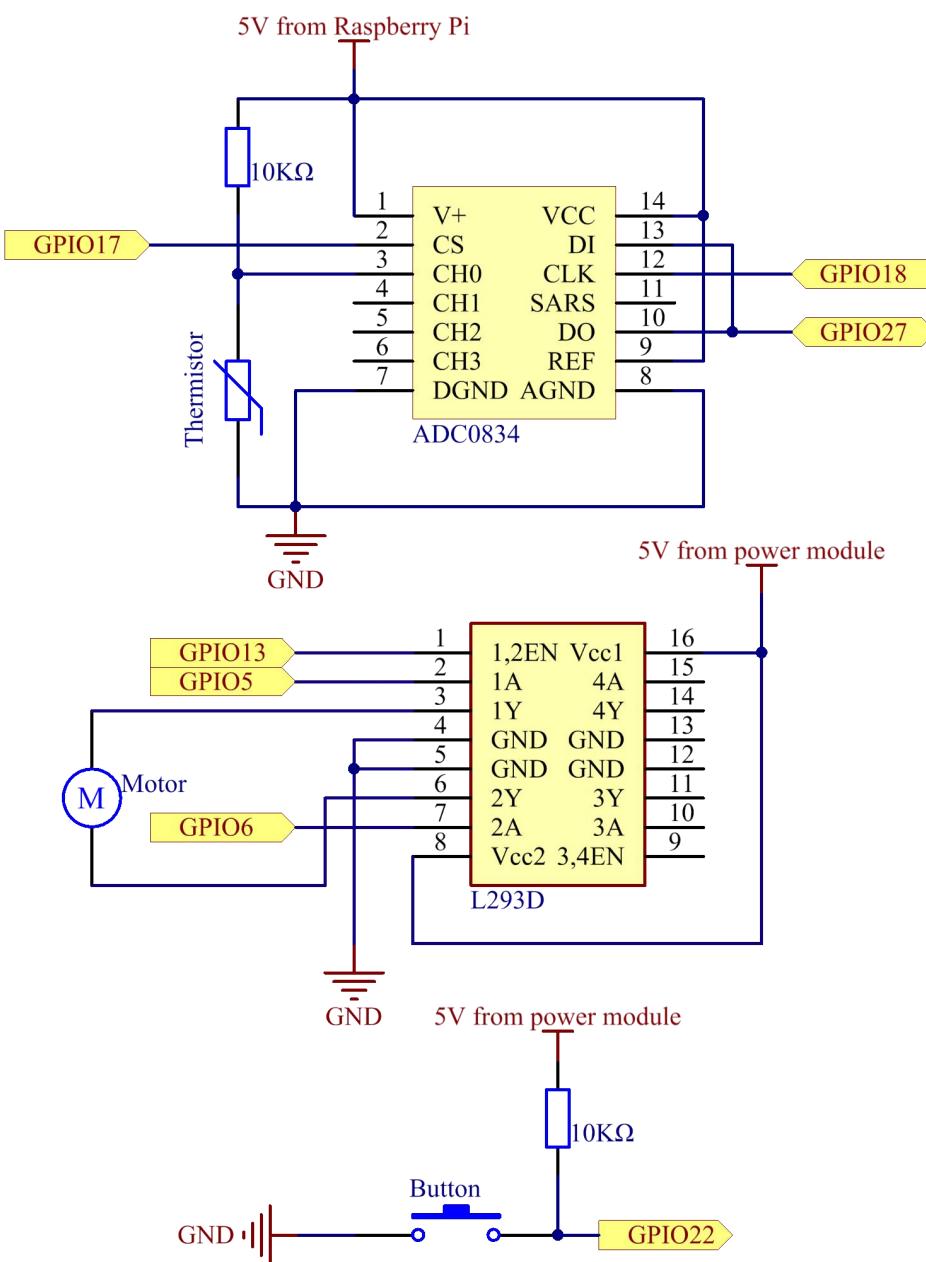
In diesem Kurs werden wir Motoren, Tasten und Thermistoren verwenden, um einen manuellen + automatischen intelligenten Lüfter herzustellen, dessen Windgeschwindigkeit einstellbar ist.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Leistungsmodul
	 GPIO Extension Board Pinout: 3V3 SDA1 SVO GND SCL1 GND 3V3 GPIO4 TXD0 GND RXD0 3V3 GPIO17 GPIO18 GND GPIO22 GPIO23 3V3 GPIO24 GND SPIMOSI GND SPIMISO GND SPISCLK GND SPICEO ID_SD ID_SC GND GPIO16 GND GPIO13 GND GPIO19 GND GPIO26 GND GPIO20 GND GPIO21	
1 * 40-Pin Kabel	1 * Thermistor	1 * L293D
		
1 * Steckbrett	1 * Taste	1 * ADC0834
		
		1 * Gleichstrommotor
		
		Mehrere Überbrückungsdrähte
		

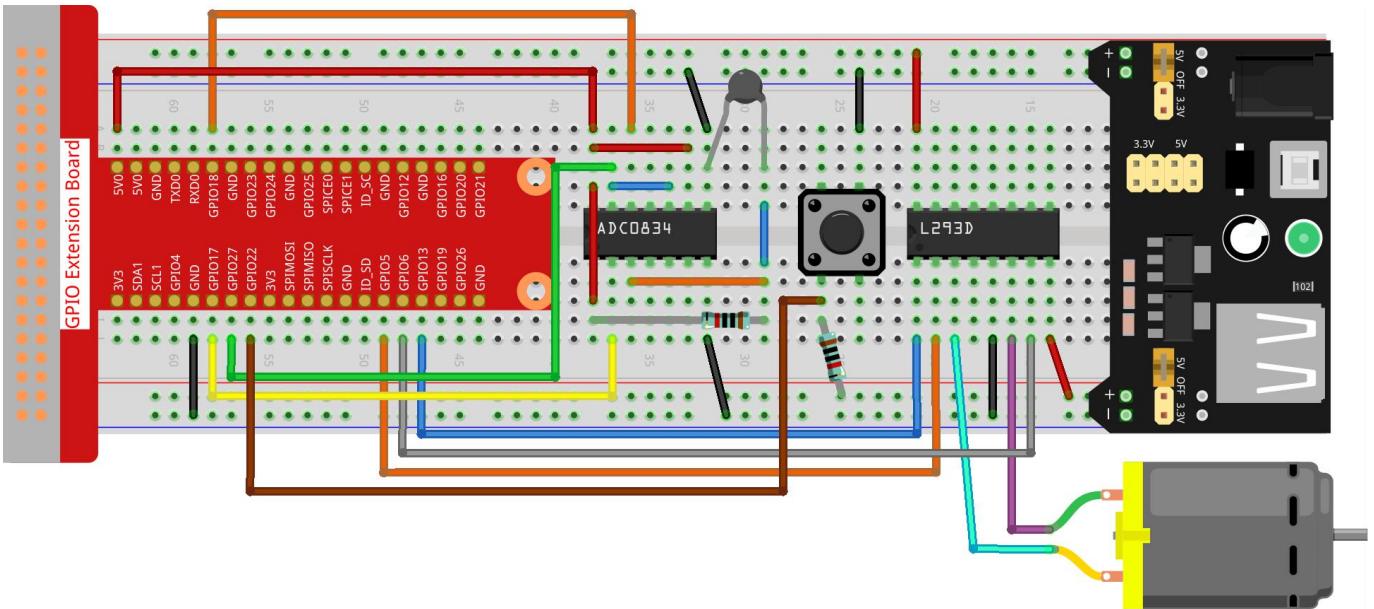
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22
GPIO5	Pin 29	21	5
GPIO6	Pin 31	22	6
GPIO13	Pin 33	23	13



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



Hinweis: Das Leistungsmodul kann eine 9-V-Batterie mit der im Kit enthaltenen 9V-Batterieschnalle anlegen. Setzen Sie die Überbrückungskappe des Leistungsmoduls in die 5V-Busleisten des Steckbretts ein.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie in den Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.4/
```

Schritt 3: Kompilieren.

```
gcc 3.1.4_SmartFan.c -lwiringPi -lm
```

Schritt 4: Führen Sie die obige ausführbare Datei aus.

sudo ./a.out

Starten Sie den Lüfter, während der Kode ausgeführt wird, indem Sie die Taste drücken. Jedes Mal, wenn Sie drücken, wird 1 Geschwindigkeitsstufe nach oben oder unten eingestellt. Es gibt **5** Arten von Geschwindigkeitsstufen: **0~4**. Wenn Sie die vierte Geschwindigkeitsstufe einstellen und die Taste drücken, arbeitet der Lüfter nicht mehr mit einer Windgeschwindigkeit von **0**.

Sobald die Temperatur länger als 2 °C , steigt oder fällt, wird die Geschwindigkeit automatisch um 1 Grad schneller oder langsamer.

Kode Erklärung

```
int temperature(){
    unsigned char analogVal;
    double Vr, Rt, temp, cel, Fah;
    analogVal = get_ADC_Result(0);
    Vr = 5 * (double)(analogVal) / 255;
    Rt = 10000 * (double)(Vr) / (5 - (double)(Vr));
    temp = 1 / (((log(Rt/10000)) / 3950)+(1 / (273.15 + 25)));
    cel = temp - 273.15;
    Fah = cel * 1.8 +32;
    int t=cel;
    return t;
}
```

Temperature() wandelt die vom ADC0834 gelesenen Thermistorwerte in Temperaturwerte um. Weitere Informationen finden Sie unter **2.2.2 Thermistor**.

```
int motor(int level){
    if(level==0){
        digitalWrite(MotorEnable,LOW);
        return 0;
    }
    if (level>=4){
        level =4;
    }
    digitalWrite(MotorEnable,HIGH);
    softPwmWrite(MotorPin1, level*25);
    return level;
}
```

Diese Funktion steuert die Drehzahl des Motors. Der Bereich der **Stufe: 0-4** (Stufe 0 stoppt den Arbeitsmotor). Eine Stufeneinstellung steht für eine 25%ige Änderung der Windgeschwindigkeit.

```
int main(void)
{
    setup();
    int currentState,lastState=0;
    int level = 0;
```

```

int currentTemp,markTemp=0;
while(1){
    currentState=digitalRead(BtnPin);
    currentTemp=temperture();
    if (currentTemp<=0){continue;}
    if (currentState==1&&lastState==0){
        level=(level+1)%5;
        markTemp=currentTemp;
        delay(500);
    }
    lastState=currentState;
    if (level!=0){
        if (currentTemp-markTemp<=-2){
            level=level-1;
            markTemp=currentTemp;
        }
        if (currentTemp-markTemp>=2){
            level=level+1;
            markTemp=currentTemp;
        }
    }
    level=motor(level);
}
return 0;
}

```

Die Funktion **main()** enthält den gesamten Programmablauf wie folgt:

- 1) Lesen Sie ständig den Tastenstatus und die aktuelle Temperatur ab.
- 2) Jede Presse erreicht Stufe+1 und gleichzeitig wird die Temperatur aktualisiert. Die **Stufe** reicht von 1~4..
- 3) Während der Lüfter arbeitet (der Niveau ist **nicht 0**), wird die Temperatur erfasst. Eine Änderung von **2°C+** bewirkt das Auf und Ab des Levels.
- 4) Der Motor ändert die Drehzahl mit der **Stufe**.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie in den Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Schritt 3: Ausführen.

```
sudo python3 3.1.4_SmartFan.py
```

Starten Sie den Lüfter, während der Kode ausgeführt wird, indem Sie die Taste drücken. Jedes Mal, wenn Sie drücken, wird 1 Geschwindigkeitsstufe nach oben oder unten eingestellt. Es gibt 5 Arten von Geschwindigkeitsstufen: 0~4. Wenn Sie die vierte Geschwindigkeitsstufe einstellen und die Taste drücken, arbeitet der Lüfter nicht mehr mit einer Windgeschwindigkeit von 0.

Sobald die Temperatur länger als 2 °C, steigt oder fällt, wird die Geschwindigkeit automatisch um 1 Grad schneller oder langsamer.

Kode Erklärung

```
def temperature():
    analogVal = ADC0834.getResult()
    Vr = 5 * float(analogVal) / 255
    Rt = 10000 * Vr / (5 - Vr)
    temp = 1/(((math.log(Rt / 10000)) / 3950) + (1 / (273.15+25)))
    Cel = temp - 273.15
    Fah = Cel * 1.8 + 32
    return Cel
```

Temperatur() wandelt die von **ADC0834** gelesenen Thermistorwerte in Temperaturwerte um. Weitere Informationen finden Sie unter **2.2.2 Thermistor**.

```
def motor(level):
    if level == 0:
        GPIO.output(MotorEnable, GPIO.LOW)
        return 0
    if level >= 4:
        level = 4
        GPIO.output(MotorEnable, GPIO.HIGH)
        p_M1.ChangeDutyCycle(level*25)
    return level
```

Diese Funktion steuert die Drehzahl des Motors. Der Bereich des **Hebels**: 0-4 (Stufe 0 stoppt den Arbeitsmotor). Eine Stufeneinstellung steht für eine 25%ige Änderung der Windgeschwindigkeit.

```
def main():
    lastState=0
    level=0
    markTemp = temperature()
    while True:
```

```

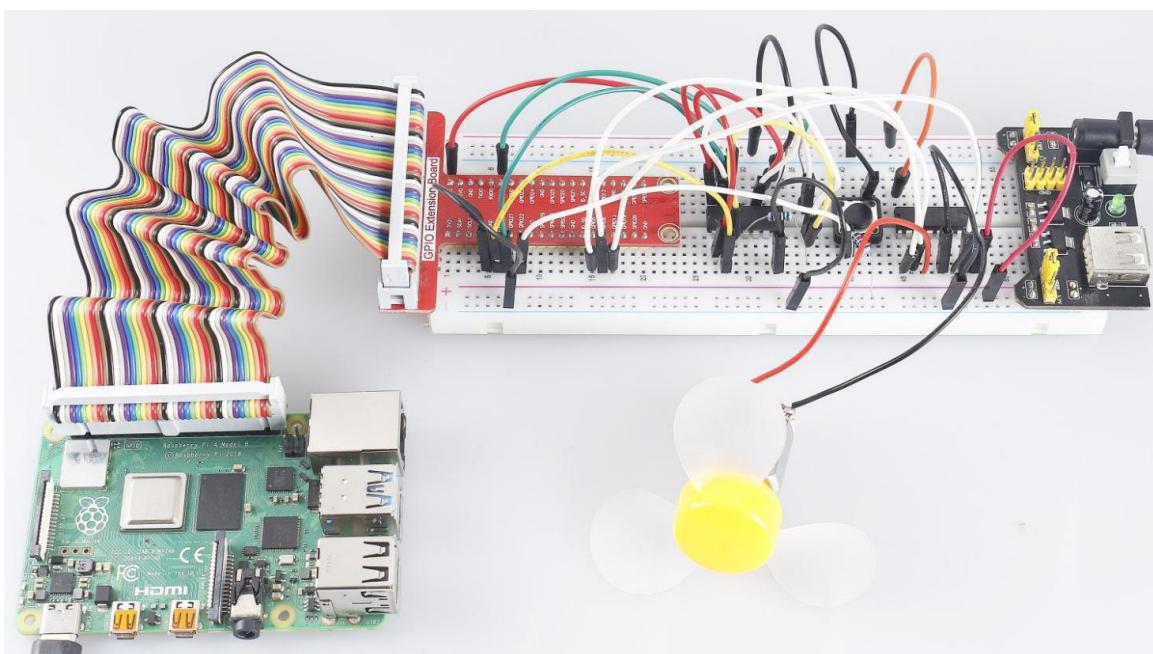
currentState =GPIO.input(BtnPin)
currentTemp=temperature()
if currentState == 1 and lastState == 0:
    level=(level+1)%5
    markTemp = currentTemp
    time.sleep(0.5)
lastState=currentState
if level!=0:
    if currentTemp-markTemp <= -2:
        level = level -1
        markTemp=currentTemp
    if currentTemp-markTemp >= 2:
        level = level +1
        markTemp=currentTemp
level = motor(level)

```

Die Funktion **main()** enthält den gesamten Programmablauf wie folgt:

- 1) Lesen Sie ständig den Tastenstatus und die aktuelle Temperatur ab.
- 2) Jede Presse erreicht Stufe+1 und gleichzeitig wird die Temperatur aktualisiert. Die **Stufe** reicht von 1~4
- 3) Während der Lüfter arbeitet (der Niveau ist **nicht 0**), wird die Temperatur erfasst. Eine Änderung von **2°C+** bewirkt das Auf und Ab des Levels.
- 4) Der Motor ändert die Drehzahl mit der **Stufe**.

Phänomen Bild



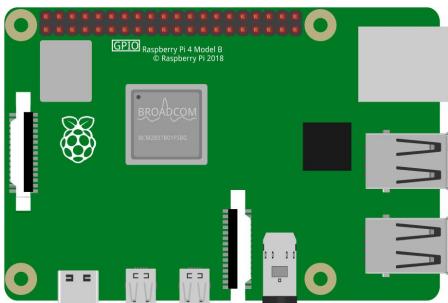
3.1.5 Batterieanzeige

Einführung

In diesem Kurs erstellen wir ein Batterieanzeigegerät, mit dem der Batteriestand auf dem LED-Balkendiagramm visuell angezeigt werden kann.

Komponenten

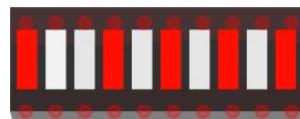
1 * Raspberry Pi



1 * T-Erweiterungskarte



1 * LED-Bargraph



1 * ADC0834



10 * Widerstand (220 Ω)



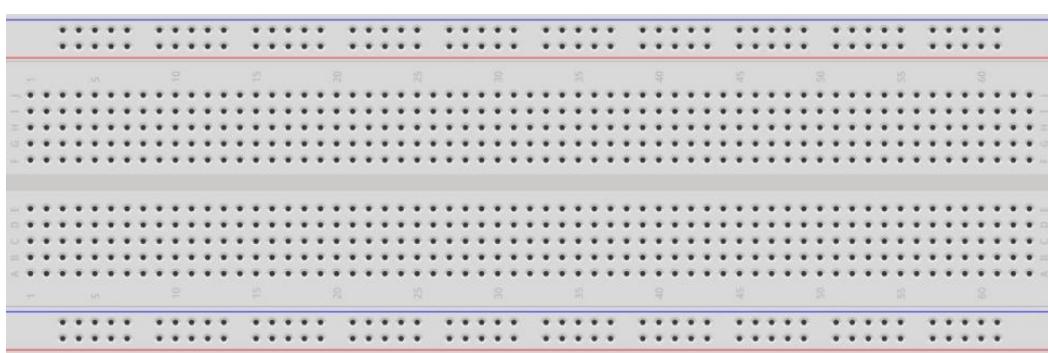
1 * 40-Pin Kabel



Mehrere
Überbrückungsdrähte

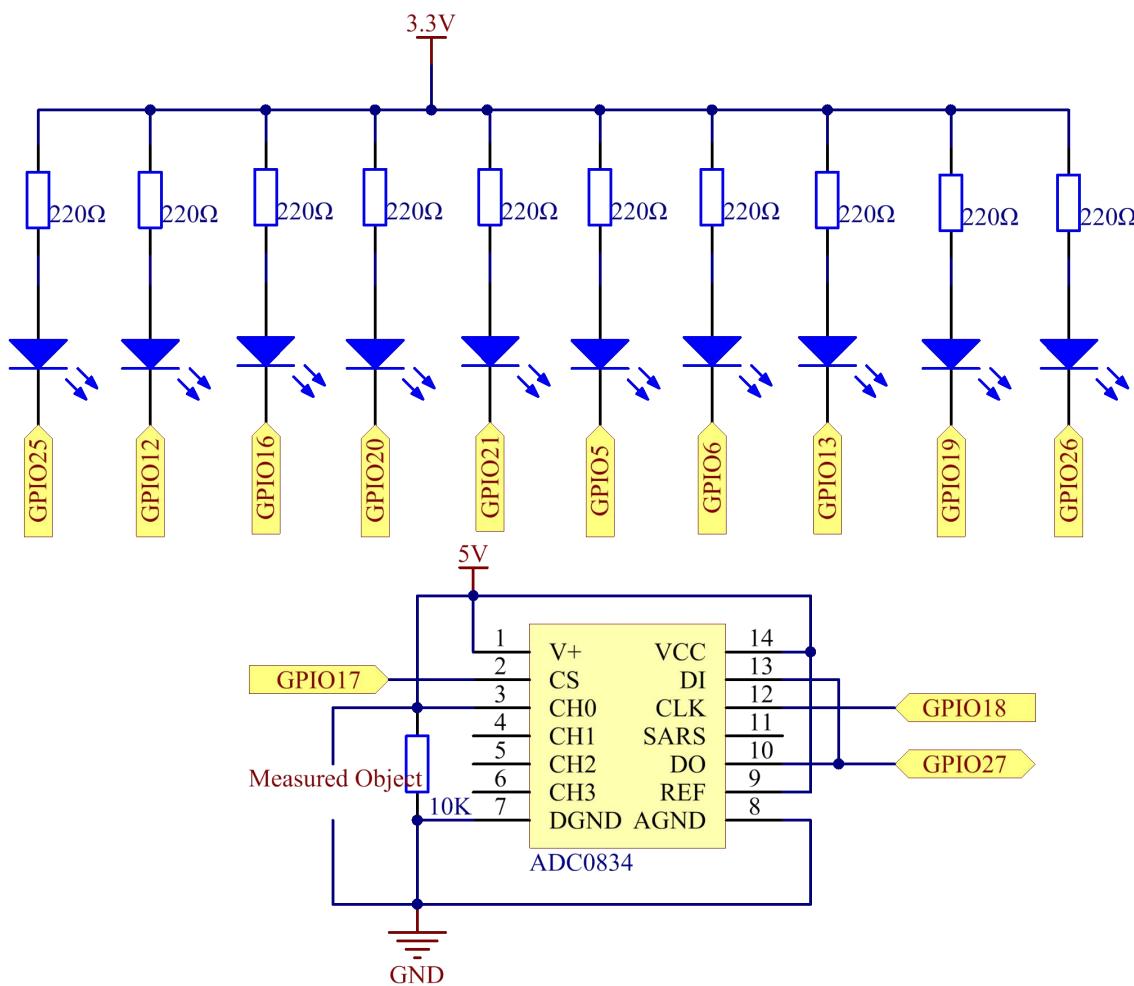


1 * Steckbrett



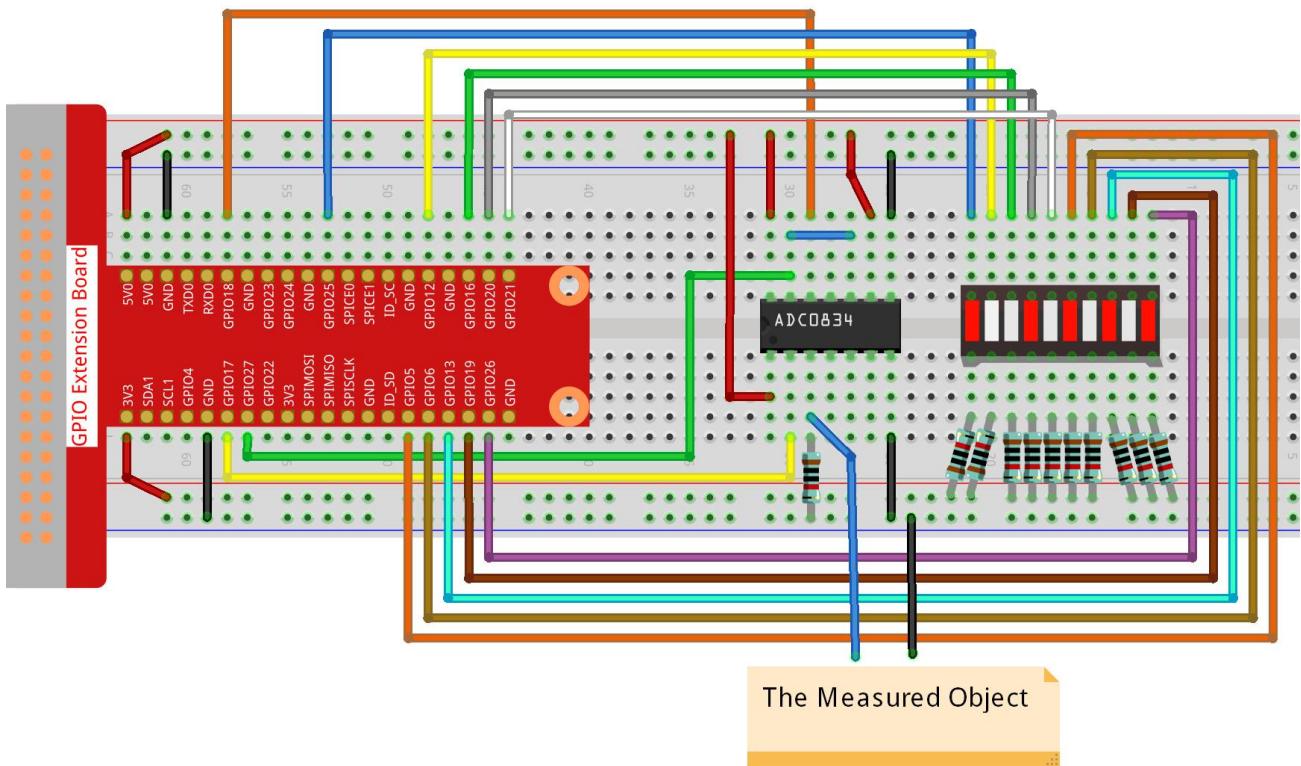
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27
GPIO25	Pin 22	6	25
GPIO12	Pin 32	26	12
GPIO16	Pin 36	27	16
GPIO20	Pin 38	28	20
GPIO21	Pin 40	29	21
GPIO5	Pin 29	21	5
GPIO6	Pin 31	22	6
GPIO13	Pin 33	23	13
GPIO19	Pin 35	24	19
GPIO26	Pin 37	25	26



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.5/
```

Schritt 3: Kompilieren Sie der Kode

```
gcc 3.1.5_BatteryIndicator.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Nachdem das Programm ausgeführt wurde, geben Sie dem 3. Pin von ADC0834 und dem GND separat einen Anschlussdraht und führen Sie sie dann separat zu den beiden Polen einer Batterie. Sie können sehen, dass die entsprechende LED auf dem LED-Balkendiagramm leuchtet, um den LeistungsNiveau anzugeben (Messbereich: 0-5V).

Kode Erklärung

```
void LedBarGraph(int value){
    for(int i=0;i<10;i++){
        digitalWrite(pins[i],HIGH);
    }
}
```

```
for(int i=0;i<value;i++){
    digitalWrite(pins[i],LOW);
}
}
```

Diese Funktion dient für die Steuerung von Ein- und Ausschalten der 10 LEDs am LED-Bargraphen. Wir geben diesen 10 LEDs hohe Niveau, damit sie zuerst ausgeschaltet sind, und entscheiden dann, wie viele LEDs aufleuchten, indem wir den empfangenen Analogwert ändern.

```
int main(void)
{
    uchar analogVal;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    pinMode(ADC_CS, OUTPUT);
    pinMode(ADC_CLK, OUTPUT);
    for(int i=0;i<10;i++){ //make led pins' mode is output
        pinMode(pins[i], OUTPUT);
        digitalWrite(pins[i],HIGH);
    }
    while(1){
        analogVal = get_ADC_Result(0);
        LedBarGraph(analogVal/25);
        delay(100);
    }
    return 0;
}
```

analogVal erzeugt Werte (**0-255**) mit variierenden Spannungswerten (**0-5V**). Wenn beispielsweise 3V an einer Batterie erkannt werden, wird der entsprechende Wert **152** auf dem Voltmeter angezeigt.

Die **10** LEDs auf dem LED-Balkendiagramm dienen zur Anzeige der **analogVal** Messwerte. $255/10 = 25$, also alle **25** erhöht sich der Analogwert, eine weitere LED leuchtet auf, z. B. wenn "analogVal = 150 (ca. 3 V), leuchten 6 LEDs".

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 3.1.5_BatteryIndicator.py
```

Nachdem das Programm ausgeführt wurde, geben Sie dem 3. Pin von ADC0834 und dem GND separat einen Anschlussdraht und führen Sie sie dann separat zu den beiden Polen einer Batterie. Sie können sehen, dass die entsprechende LED auf dem LED-Balkendiagramm leuchtet, um den LeistungsNiveau anzuzeigen (Messbereich: 0-5V).

Kode Erklärung

```
def LedBarGraph(value):  
    for i in ledPins:  
        GPIO.output(i,GPIO.HIGH)  
    for i in range(value):  
        GPIO.output(ledPins[i],GPIO.LOW)
```

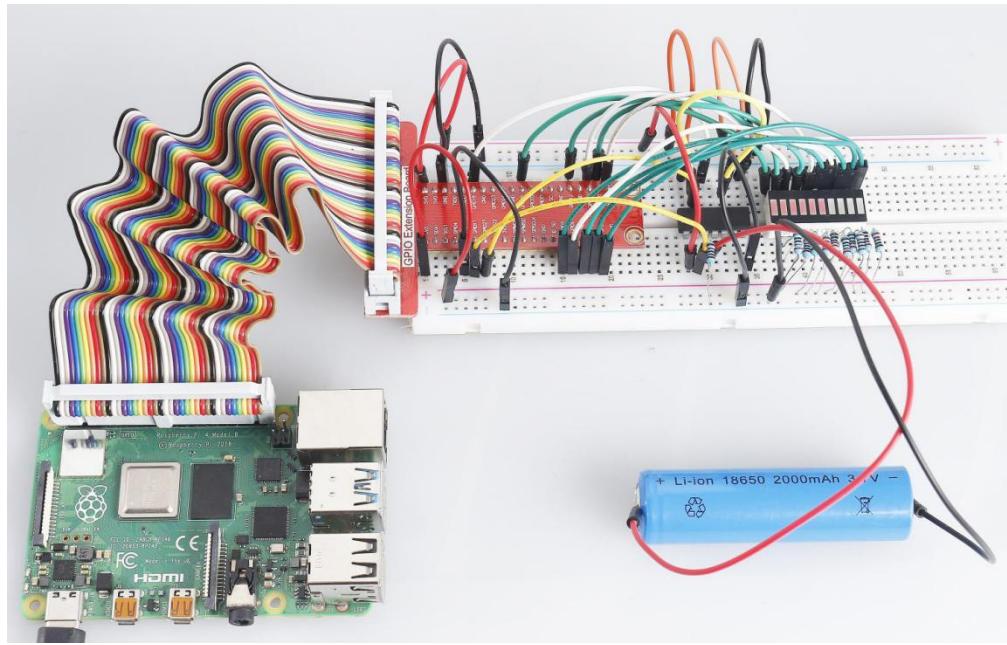
Diese Funktion dient für die Steuerung von Ein- und Ausschalten der **10** LEDs am LED-Bargraphen. Wir geben diesen **10** LEDs hohe Niveau, damit sie zuerst **ausgeschaltet** sind, und entscheiden dann, wie viele LEDs aufleuchten, indem wir den empfangenen Analogwert ändern.

```
def loop():  
    while True:  
        analogVal = ADC0834.getResult()  
        LedBarGraph(int(analogVal/25))
```

analogVal erzeugt Werte (**0-255**) mit variierenden Spannungswerten (**0-5V**). Wenn beispielsweise 3V an einer Batterie erkannt werden, wird der entsprechende Wert **152** auf dem Voltmeter angezeigt.

Die **10** LEDs auf dem LED-Balkendiagramm dienen zur Anzeige der **analogVal** Messwerte. $255/10 = 25$, also alle **25** erhöht sich der Analogwert, eine weitere LED leuchtet auf, z. B. wenn "analogVal = 150 (ca. 3 V), leuchten 6 LEDs".

Phänomen Bild

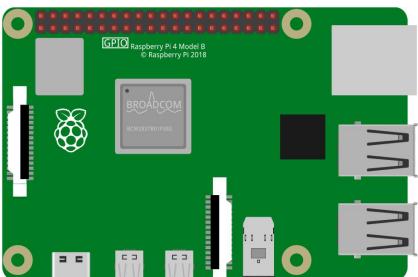
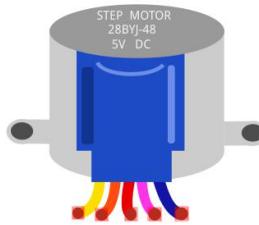
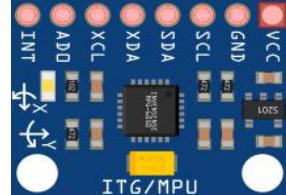
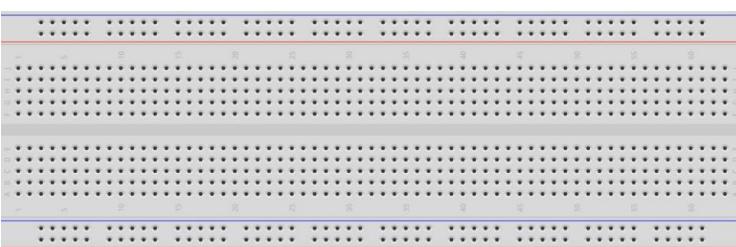
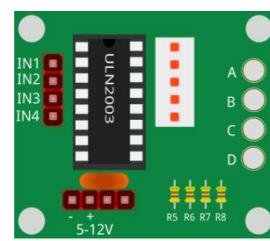


3.1.6 Bewegungssteuerung

Einführung

In dieser Lektion werden wir ein einfaches Bewegungserfassungs- und Steuergerät herstellen. Die MPU6050 wird als Sensor und der Schrittmotor als gesteuertes Gerät verwendet. Mit der am Handschuh montierten MPU6050 können Sie den Schrittmotor durch Drehen Ihres Handgelenks steuern.

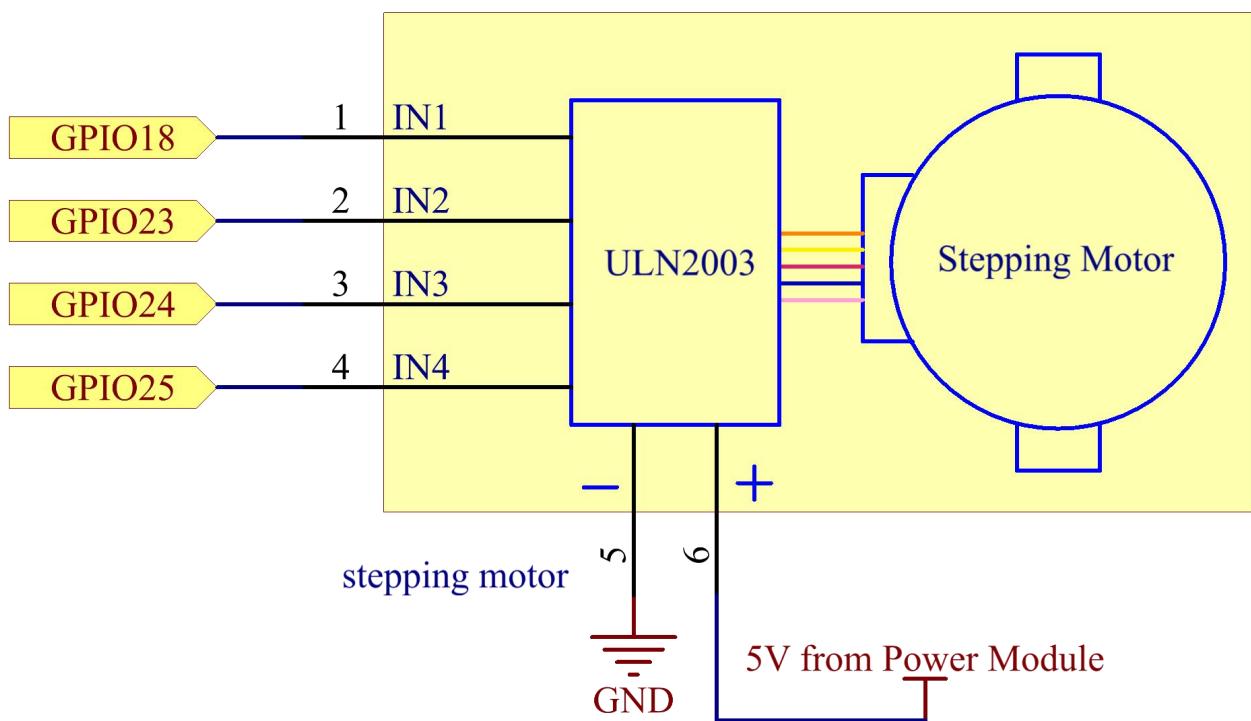
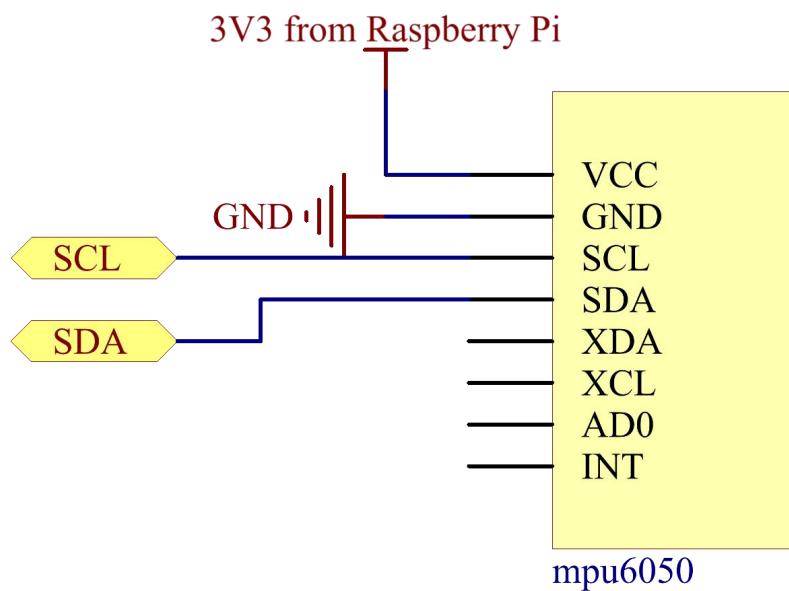
Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Schrittmotor																																										
	 <table border="1"> <tr><th colspan="2">GPIO Extension Board</th></tr> <tr><td>3V3</td><td>SV0</td></tr> <tr><td>SDA1</td><td>GND</td></tr> <tr><td>SCL1</td><td>I2C0</td></tr> <tr><td>GPIO4</td><td>I2C1</td></tr> <tr><td>GND</td><td>I2C2</td></tr> <tr><td>GPIO17</td><td>GPIO18</td></tr> <tr><td>GPIO27</td><td>GND</td></tr> <tr><td>GPIO22</td><td>GPIO23</td></tr> <tr><td>3V3</td><td>GPIO24</td></tr> <tr><td>SPI_MOSI</td><td>GND</td></tr> <tr><td>SPI_MISO</td><td>GND</td></tr> <tr><td>SPI_SCLK</td><td>SPI_CE0</td></tr> <tr><td>GND</td><td>SPI_CE1</td></tr> <tr><td>ID_SD</td><td>ID_SC</td></tr> <tr><td>GPIO5</td><td>GND</td></tr> <tr><td>GPIO6</td><td>GPIO7</td></tr> <tr><td>GPIO13</td><td>GND</td></tr> <tr><td>GPIO19</td><td>GPIO16</td></tr> <tr><td>GPIO26</td><td>GPIO20</td></tr> <tr><td>GND</td><td>GPIO21</td></tr> </table>	GPIO Extension Board		3V3	SV0	SDA1	GND	SCL1	I2C0	GPIO4	I2C1	GND	I2C2	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V3	GPIO24	SPI_MOSI	GND	SPI_MISO	GND	SPI_SCLK	SPI_CE0	GND	SPI_CE1	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO7	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	
GPIO Extension Board																																												
3V3	SV0																																											
SDA1	GND																																											
SCL1	I2C0																																											
GPIO4	I2C1																																											
GND	I2C2																																											
GPIO17	GPIO18																																											
GPIO27	GND																																											
GPIO22	GPIO23																																											
3V3	GPIO24																																											
SPI_MOSI	GND																																											
SPI_MISO	GND																																											
SPI_SCLK	SPI_CE0																																											
GND	SPI_CE1																																											
ID_SD	ID_SC																																											
GPIO5	GND																																											
GPIO6	GPIO7																																											
GPIO13	GND																																											
GPIO19	GPIO16																																											
GPIO26	GPIO20																																											
GND	GPIO21																																											
1 * 40-Pin Kabel	Mehrere Überbrückungsdrähte	1 * MPU6050-Modul																																										
																																												
1 * Steckbrett	1 * ULN2003-Modul																																											
																																												

Schematische Darstellung

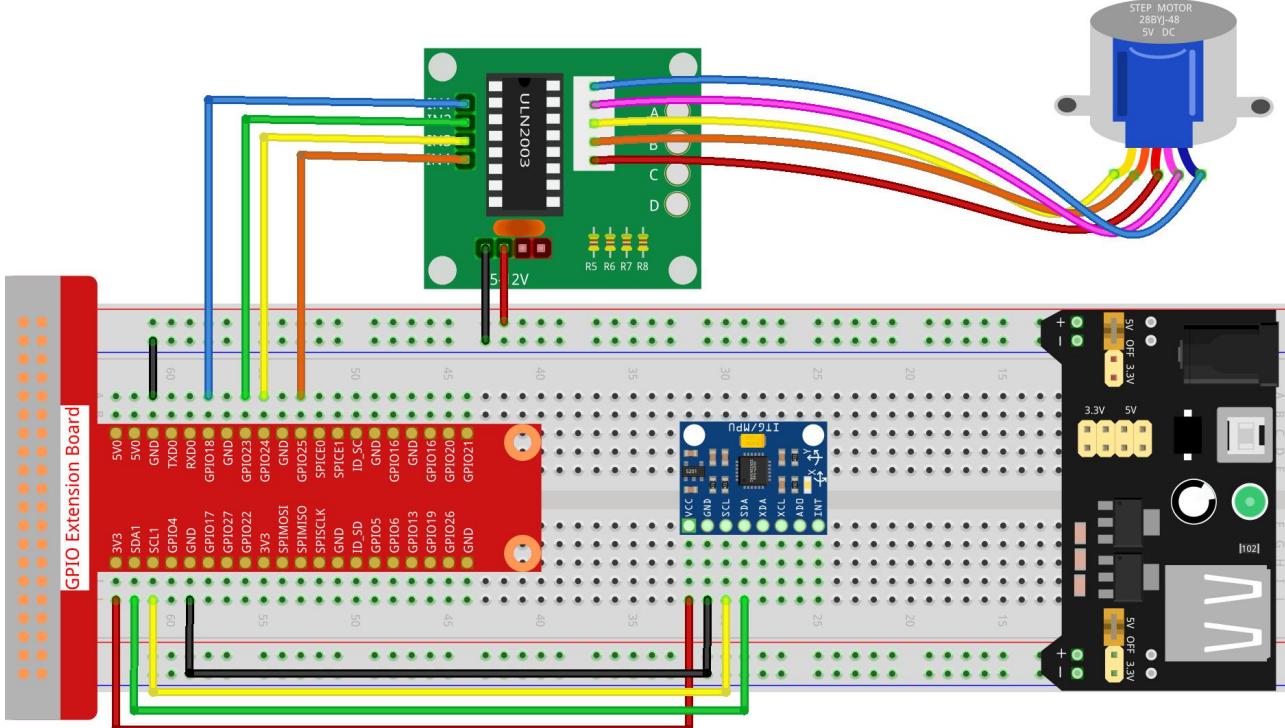
T-Karte Name	physisch	wiringPi	BCM

GPIO18	Pin 12	1	18
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24
GPIO25	Pin 22	6	25
SDA1	Pin 3		
SCL1	Pin 5		



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.6/
```

Schritt 3: Kompilieren Sie die Kode.

```
gcc 3.1.6_MotionControl.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Wenn der Neigungswinkel von **mpu6050** auf der Y-Achse größer als **45 °C**, ist, dreht sich der Schrittmotor während der Kode gegen den Uhrzeigersinn. Bei weniger als **-45 °C**, dreht sich der Schrittmotor im Uhrzeigersinn.

Kode Erklärung

```
double mpu6050(){
    acclX = read_word_2c(0x3B);
    acclY = read_word_2c(0x3D);
    acclZ = read_word_2c(0x3F);
    acclX_scaled = acclX / 16384.0;
    acclY_scaled = acclY / 16384.0;
    acclZ_scaled = acclZ / 16384.0;
    double angle=get_y_rotation(acclX_scaled, acclY_scaled, acclZ_scaled);
    return angle;
```

}

mpu6050 erhält den Neigungswinkel in Richtung der Y-Achse.

```
void rotary(char direction){
    if(direction == 'c'){
        for(int j=0;j<4;j++){
            for(int i=0;i<4;i++)
                {digitalWrite(motorPin[i],0x99>>j & (0x08>>i));}
            delayMicroseconds(stepSpeed);
        }
    }
    else if(direction =='a'){
        for(int j=0;j<4;j++){
            for(int i=0;i<4;i++)
                {digitalWrite(motorPin[i],0x99<<j & (0x80>>i));}
            delayMicroseconds(stepSpeed);
        }
    }
}
```

Wenn die empfangene **Richtung Taste** 'c' ist, dreht sich der Schrittmotor im Uhrzeigersinn. Wenn der **Taste** 'a' ist, dreht sich der Motor gegen den Uhrzeigersinn. Weitere Informationen zur Berechnung der Drehrichtung des Schrittmotors finden Sie unter **1.3.3 Schrittmotor**.

```
int main()
{
    setup();
    double angle;
    while(1) {
        angle = mpu6050();
        if (angle >=45){rotary('a');}
        else if (angle<=-45){rotary('c');}
    }
    return 0;
}
```

Der Neigungswinkel in Richtung der Y-Achse wird von **mpu6050** abgelesen. Wenn er größer als **45°C**, ist, dreht sich der Schrittmotor gegen den Uhrzeigersinn. Bei weniger als **45°C**, dreht sich der Schrittmotor im Uhrzeigersinn.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 3.1.6_MotionControl.py
```

Wenn der Neigungswinkel von **mpu6050** auf der Y-Achse größer als **45 °C**, ist, dreht sich der Schrittmotor während der Kode gegen den Uhrzeigersinn. Bei weniger als **45 °C**, dreht sich der Schrittmotor im Uhrzeigersinn.

Kode Erklärung

```
def mpu6050():
    accel_xout = read_word_2c(0x3b)
    accel_yout = read_word_2c(0x3d)
    accel_zout = read_word_2c(0x3f)
    accel_xout_scaled = accel_xout / 16384.0
    accel_yout_scaled = accel_yout / 16384.0
    accel_zout_scaled = accel_zout / 16384.0
    angle=get_y_rotation(accel_xout_scaled, accel_yout_scaled, accel_zout_scaled)
    return angle
```

mpu6050 erhält den Neigungswinkel in Richtung der Y-Achse.

```
def rotary(direction):
    if(direction == 'c'):
        for j in range(4):
            for i in range(4):
                GPIO.output(motorPin[i],0x99>>j & (0x08>>i))
                time.sleep(stepSpeed)

    elif(direction == 'a'):
        for j in range(4):
            for i in range(4):
                GPIO.output(motorPin[i],0x99<<j & (0x80>>i))
                time.sleep(stepSpeed)
```

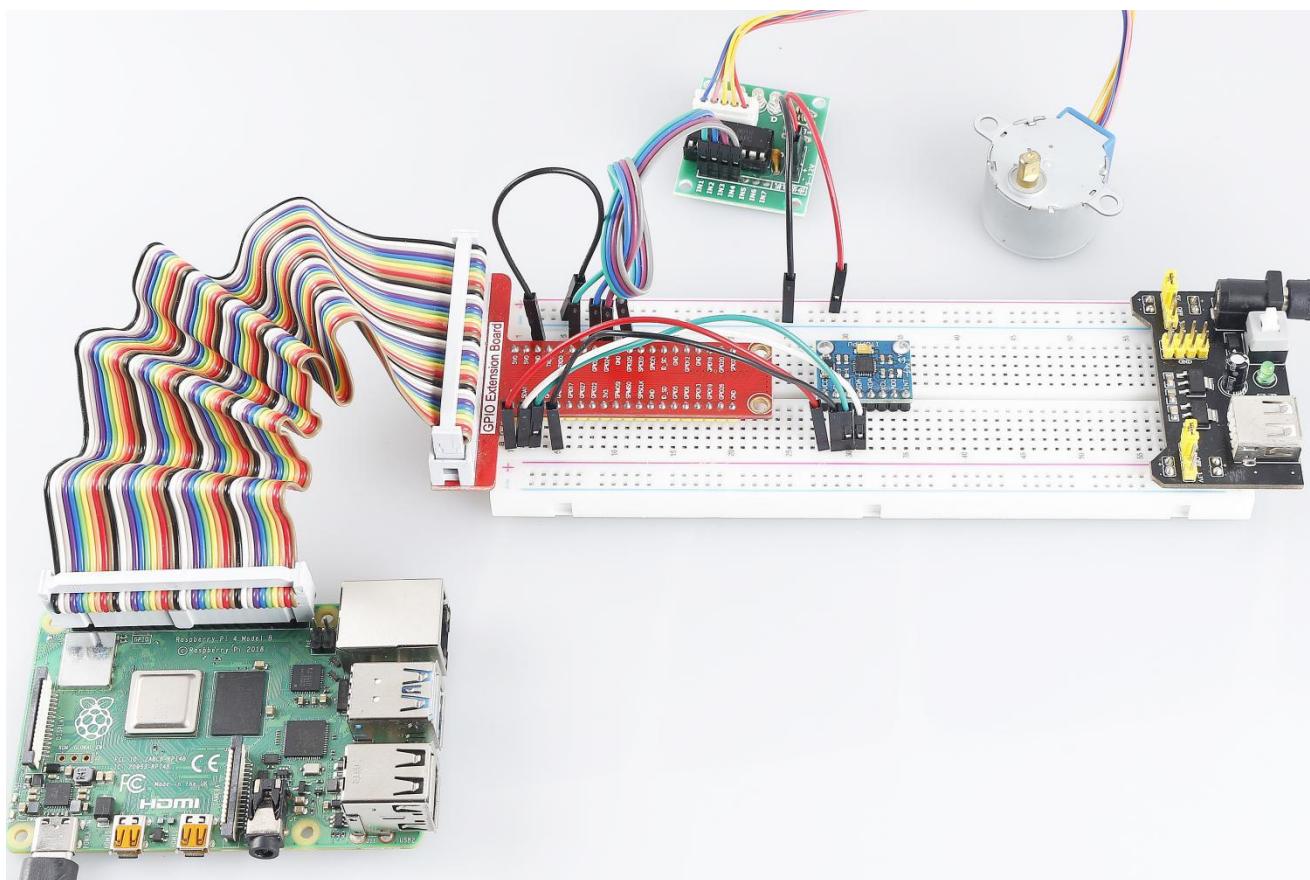
Wenn die empfangene **Richtung Taste** 'c' ist, dreht sich der Schrittmotor im Uhrzeigersinn. Wenn der **Taste** 'a' ist, dreht sich der Motor gegen den Uhrzeigersinn. Weitere Informationen zur Berechnung der Drehrichtung des Schrittmotors finden Sie unter **1.3.3 Schrittmotor**.

```
def loop():
    while True:
```

```
angle=mpu6050()
if angle >=45 :
    rotary('a')
elif angle <=-45:
    rotary('c')
```

Der Neigungswinkel in Richtung der **Y-Achse** wird aus **mpu6050** abgelesen. Wenn er größer als **45°C**, ist, wird rotary () aufgerufen, damit sich der Schrittmotor gegen den Uhrzeigersinn dreht. Bei weniger als **45°C**, dreht sich der Schrittmotor im Uhrzeigersinn.

Phänomen Bild

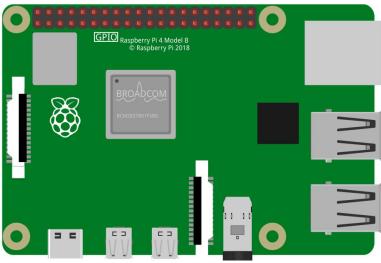
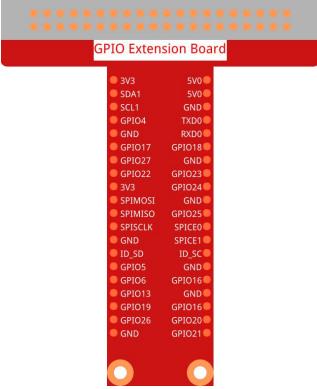
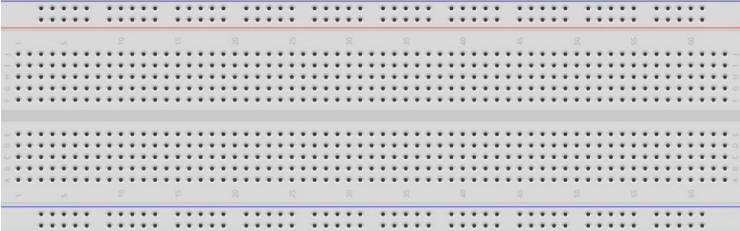
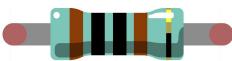


3.1.7 Ampel

Einführung

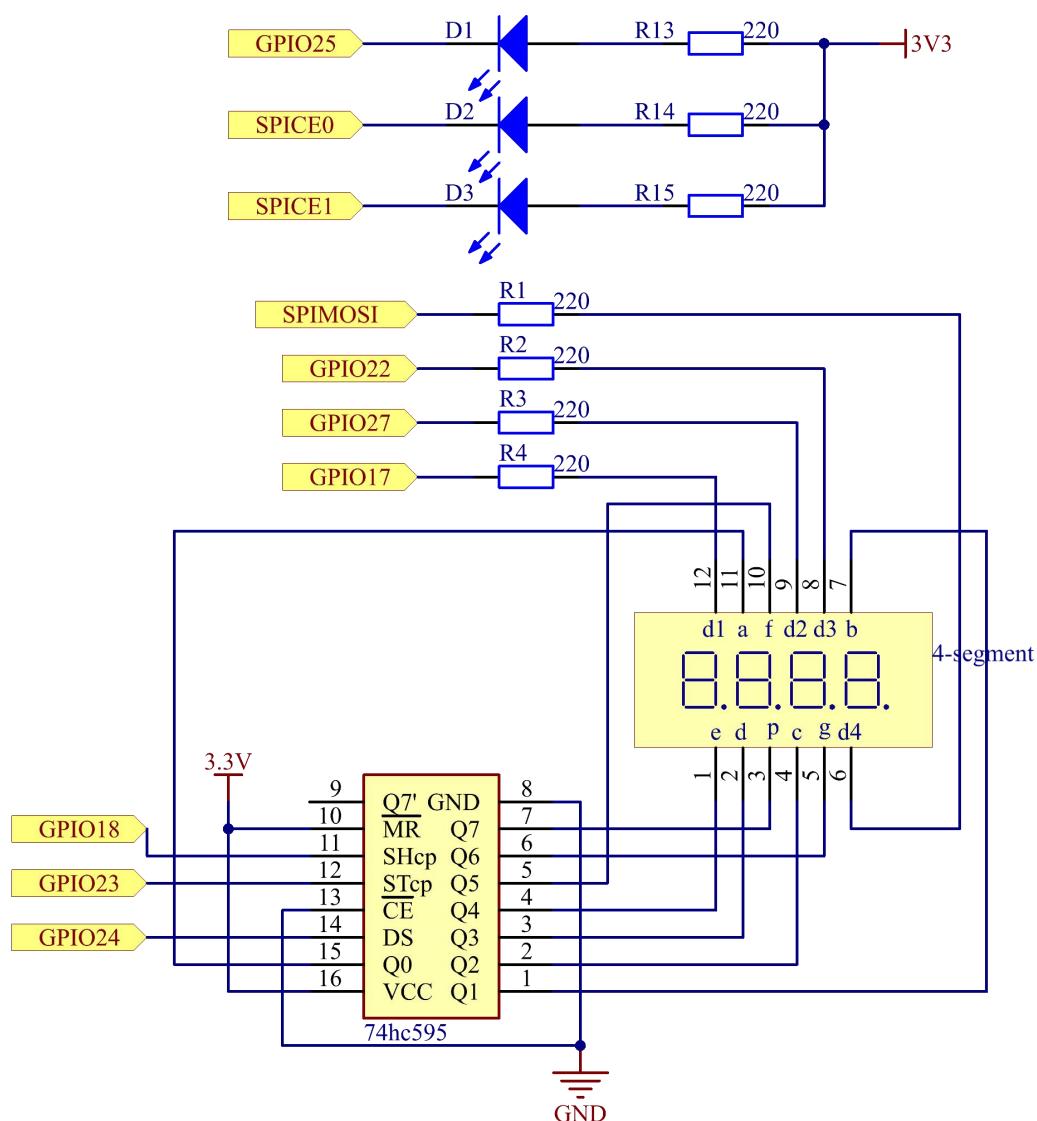
In diesem Projekt werden wir dreifarbig LED-Lichter verwenden, um den Wechsel der Ampeln zu realisieren, und eine vierstellige 7-Segment-Anzeige wird verwendet, um das Timing jedes Verkehrszustands anzuzeigen.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * 4-stellige 7-Segment-Anzeige
	 GPIO Extension Board Pinout: 3V3, SDA1, SCL1, GND, GPIO4, GND, TXD0, RXD0, GPIO17, GND, GPIO27, GND, GPIO22, GND, SPI2, GND, SPIMOSI, GND, SPIMISO, GND, SPI5CLK, GND, GND, SPICE0, GND, ID_SD, GND, SPICE1, GND, SPIOS, GND, GPIO6, GND, GPIO16, GND, GPIO13, GND, GPIO16, GND, GPIO26, GND, GPIO20, GND, GPIO21, GND	
1 * 40-Pin Kabel	4 * S8550 PNP-Transistor	
		
1 * Steckbrett	3 * LED	
		
Mehrere Überbrückungsdrähte		
11 * Widerstand (220 Ω)		
4 * Widerstand 1KΩ		

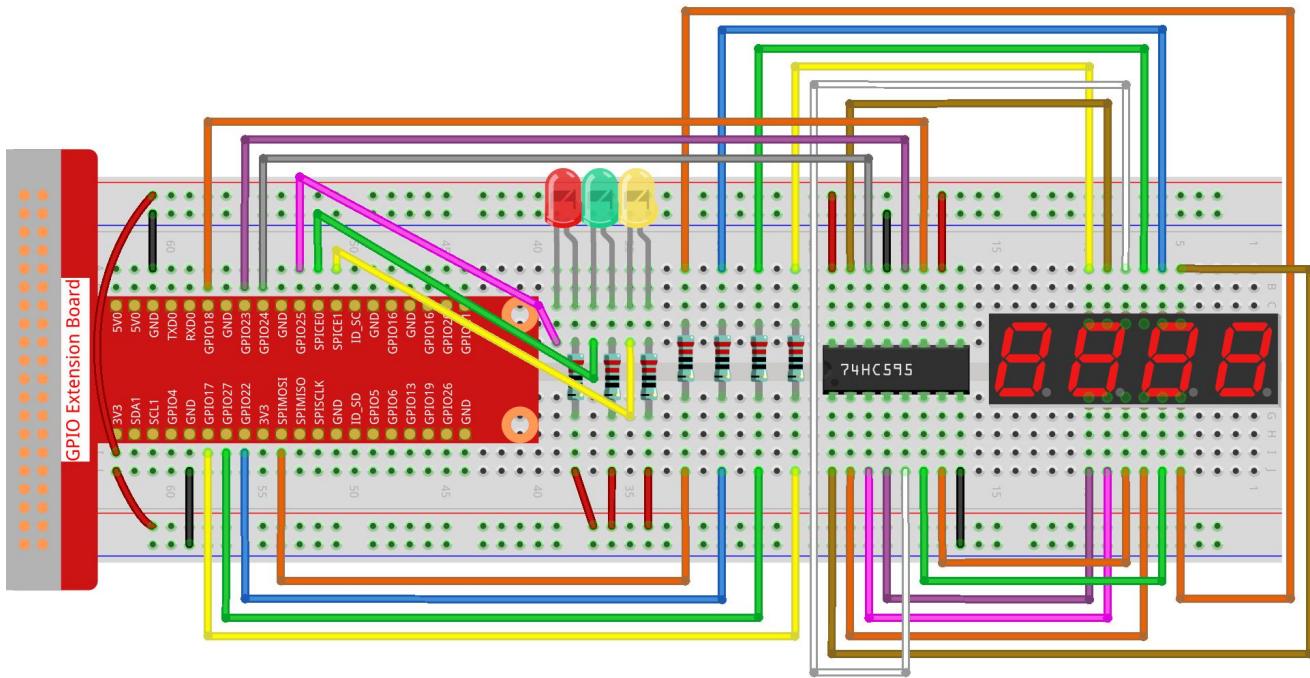
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22
SPI-MOSI	Pin 19	12	10
GPIO18	Pin 12	1	18
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24
GPIO25	Pin 22	6	25
SPI-CE0	Pin 24	10	8
SPI-CE1	Pin 26	11	7



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.7/
```

Schritt 3: Kompilieren.

```
gcc 3.1.7_TrafficLight.c -lwiringPi
```

Schritt 4: Ausführen.

```
sudo ./a.out
```

Während die Kode ausgeführt wird, simulieren LEDs den Farbwechsel von Ampeln. Zuerst leuchtet die rote LED 60 Sekunden lang, dann leuchtet die grüne LED 30 Sekunden lang. Als nächstes leuchtet die gelbe LED 5 Sekunden lang auf. Danach leuchtet die rote LED erneut für 60s. Auf diese Weise wird diese Reihe von Aktionen wiederholt ausgeführt.

Kode Erklärung

```
#define SDI 5
#define RCLK 4
#define SRCLK 1

const int placePin[] = {12, 3, 2, 0};
unsigned char number[] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90};

void pickDigit(int digit);
void hc595_shift(int8_t data);
void clearDisplay();
void display();
```

Diese Kode werden verwendet, um die Funktion der Nummeranzeige von 4-stelligen 7-Segment-Anzeigen zu realisieren. Weitere Informationen finden Sie in **Kapitel 1.1.5** des Dokuments. Hier verwenden wir die Kode, um den Countdown der Ampelzeit anzuzeigen.

```
const int ledPin[]={6,10,11};

int colorState = 0;

void lightup()
{
    for(int i=0;i<3;i++){
        digitalWrite(ledPin[i],HIGH);
    }
    digitalWrite(ledPin[colorState],LOW);
}
```

Die Kode dienen zum Ein- und Ausschalten der LED.

```
int greenLight = 30;
int yellowLight = 5;
int redLight = 60;
int colorState = 0;
char *lightColor[]={"Red","Green","Yellow"};
int counter = 60;
```

```

void timer(int timer1){      //Timer function
    if(timer1 == SIGALRM){
        counter--;
        alarm(1);
        if(counter == 0){
            if(colorState == 0) counter = greenLight;
            if(colorState == 1) counter = yellowLight;
            if(colorState == 2) counter = redLight;
            colorState = (colorState+1)%3;
        }
        printf("counter : %d \t light color: %s \n",counter,lightColor[colorState]);
    }
}

```

Die Kode dienen zum Ein- und Ausschalten des Timers. Weitere Informationen finden Sie in Kapitel 1.1.5. Wenn der Timer auf Null zurückkehrt, wird colorState umgeschaltet, um die LED zu wechseln, und der Timer wird einem neuen Wert zugewiesen.

```

void loop()
{
    while(1){
        display();
        lightup();
    }
}

int main(void)
{
    //...
    signal(SIGALRM,timer);
    alarm(1);
    loop();
    return 0;
}

```

Der Timer wird in der Funktion main() gestartet. Verwenden Sie in der Funktion loop() die Schleife **while(1)** und rufen Sie die Funktionen von 4-stelligem 7-Segment und LED auf.

➤ Für Python-Sprachbenutzer

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Ausführen.

```
sudo python3 3.1.7_TrafficLight.py
```

Während die Kode ausgeführt wird, simulieren LEDs den Farbwechsel von Ampeln. Zuerst leuchtet die rote LED 60 Sekunden lang, dann leuchtet die grüne LED 30 Sekunden lang. Als nächstes leuchtet die gelbe LED 5 Sekunden lang auf. Danach leuchtet die rote LED erneut für 60s. Auf diese Weise wird diese Reihe von Aktionen wiederholt ausgeführt. Währenddessen zeigt die 4-stellige 7-Segment-Anzeige kontinuierlich die Countdown-Zeit an.

Kode Erklärung

```
SDI = 24      #serial data input(DS)
RCLK = 23     #memory clock input(STCP)
SRCLK = 18    #shift register clock input(SHCP)
number = (0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90)
placePin = (17,27,22,10)

def clearDisplay():
def hc595_shift(data):
def pickDigit(digit):
def display():
```

Diese Kode werden verwendet, um die Funktion der Nummeranzeige eines 4-stelligen 7-Segments zu realisieren. Weitere Informationen finden Sie in Kapitel 1.1.5 des Dokuments. Hier verwenden wir die Kode, um den Countdown der Ampelzeit anzuzeigen.

```
ledPin =(22,24,26)
colorState=0

def lightup():
    global colorState
    for i in range(0,3):
        GPIO.output(ledPin[i], GPIO.HIGH)
        GPIO.output(ledPin[colorState], GPIO.LOW)
```

Die Kode dienen zum Ein- und Ausschalten der LED.

```

greenLight = 30
yellowLight = 5
redLight = 60
lightColor=("Red","Green","Yellow")

colorState=0
counter = 60
timer1 = 0

def timer():      #timer function
    global counter
    global colorState
    global timer1
    timer1 = threading.Timer(1.0,timer)
    timer1.start()
    counter-=1
    if (counter is 0):
        if(colorState is 0):
            counter= greenLight
        if(colorState is 1):
            counter=yellowLight
        if (colorState is 2):
            counter=redLight
        colorState=(colorState+1)%3
    print ("counter : %d    color: %s %(counter,lightColor[colorState]))
```

Die Kode dienen zum Ein- und Ausschalten des Timers. Weitere Informationen finden Sie in Kapitel 1.1.5. Wenn der Timer auf Null zurückkehrt, wird colorState umgeschaltet, um die LED zu wechseln, und der Timer wird einem neuen Wert zugewiesen.

```

def setup():
    # ...
    global timer1
    timer1 = threading.Timer(1.0,timer)
    timer1.start()

def loop():
    while True:
```

```

display()
lightup()

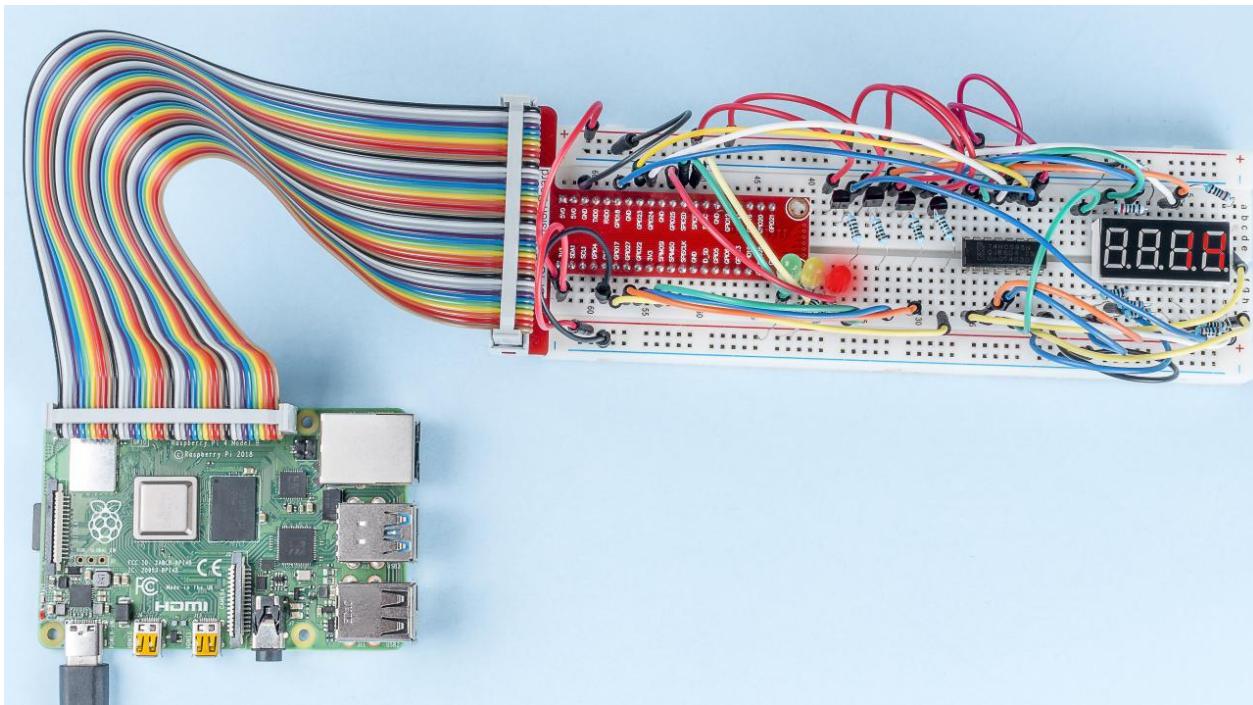
def destroy():    # When "Ctrl+C" is pressed, the function is executed.
    global timer1
    GPIO.cleanup()
    timer1.cancel()      #cancel the timer

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

```

Starten Sie in der Funktion `setup()` den Timer. In der Funktion `loop()` wird eine **Weile True** verwendet: Rufen Sie die relativen Funktionen von 4-stelligem 7-Segment und LED kreisförmig auf.

Phänomen Bild

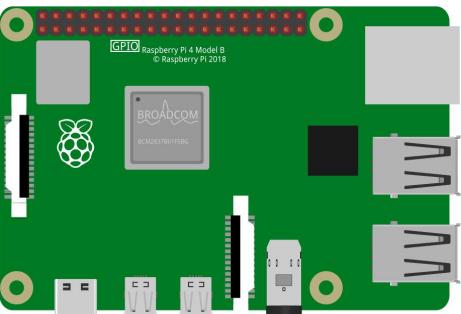
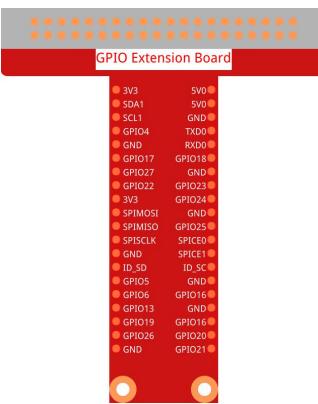
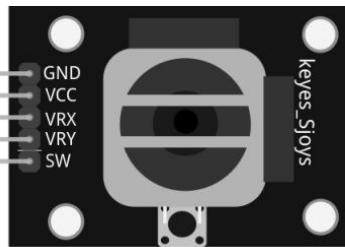
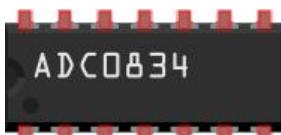
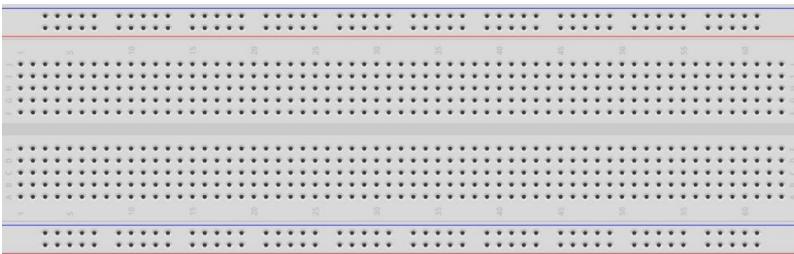
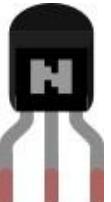


3.1.8 Überhitzungsmonitor

Einführung

Möglicherweise möchten Sie ein Überhitzungsüberwachungsgerät herstellen, das für verschiedene Situationen gilt, z. B. im Werk, wenn wir einen Alarm und das rechtzeitige automatische Ausschalten der Maschine bei Überhitzung des Stromkreises wünschen. In dieser Lektion verwenden wir Thermistor, Joystick, Summer, LED und LCD, um ein intelligentes Temperaturüberwachungsgerät zu erstellen, dessen Schwelle einstellbar ist.

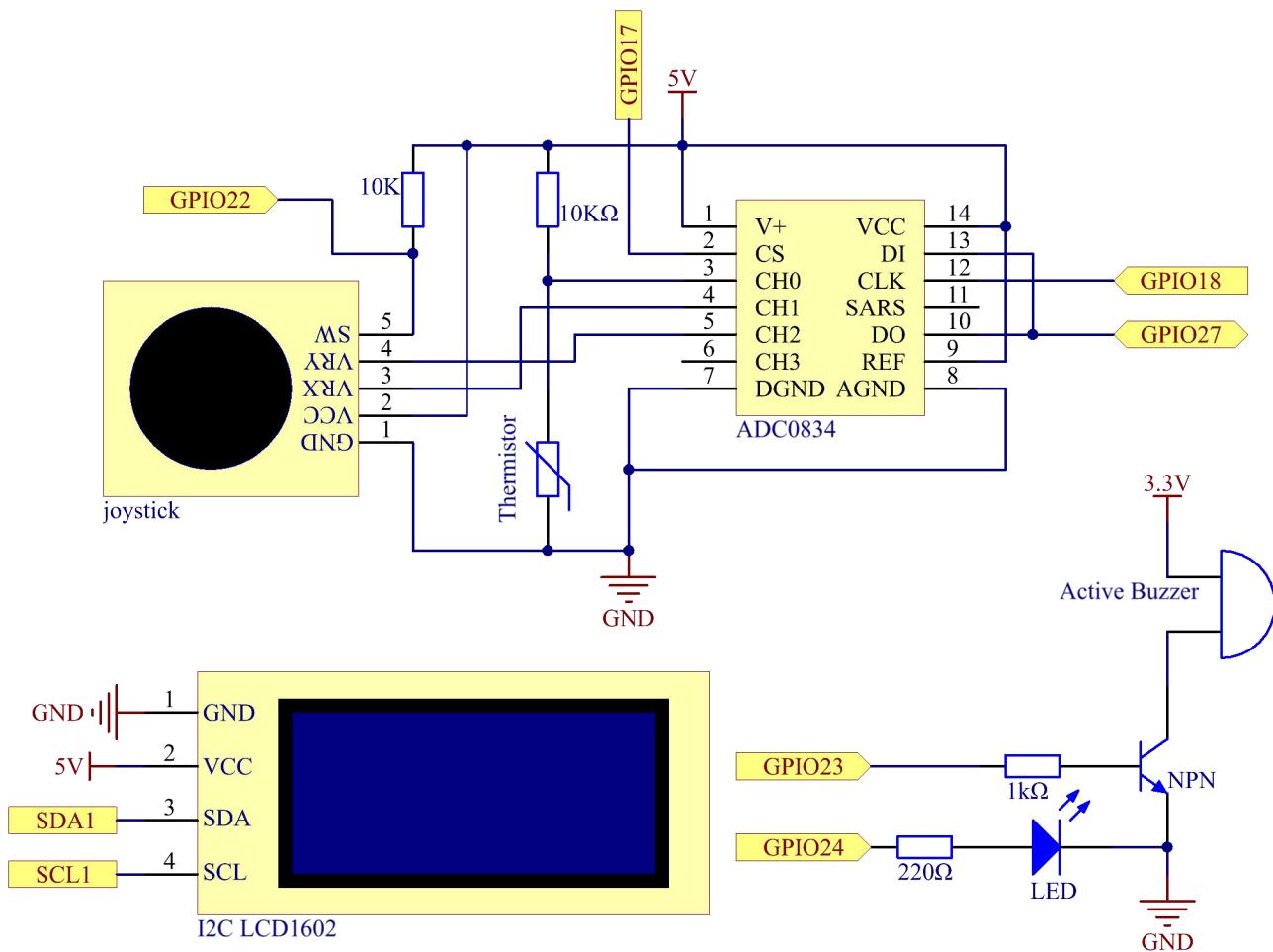
Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Joystick																																												
	 <table border="1"> <tr><th colspan="2">GPIO Extension Board</th></tr> <tr><td>3V3</td><td>SVO</td></tr> <tr><td>SDA1</td><td>SVO</td></tr> <tr><td>SCL1</td><td>GND</td></tr> <tr><td>GPIO4</td><td>TX00</td></tr> <tr><td>GND</td><td>RX00</td></tr> <tr><td>GPIO17</td><td>GPIO18</td></tr> <tr><td>GPIO27</td><td>GND</td></tr> <tr><td>GPIO22</td><td>GPIO23</td></tr> <tr><td>3V</td><td>GPIO24</td></tr> <tr><td>SPI MOSI</td><td>GND</td></tr> <tr><td>SPI MISO</td><td>GPIO25</td></tr> <tr><td>SPI SCLK</td><td>SPICE0</td></tr> <tr><td>GND</td><td>SPICE1</td></tr> <tr><td>ID_SD</td><td>ID_SC</td></tr> <tr><td>GPIO5</td><td>GND</td></tr> <tr><td>GPIO6</td><td>GPIO16</td></tr> <tr><td>GPIO13</td><td>GND</td></tr> <tr><td>GPIO19</td><td>GPIO16</td></tr> <tr><td>GPIO26</td><td>GPIO16</td></tr> <tr><td>GND</td><td>GPIO20</td></tr> <tr><td></td><td>GPIO21</td></tr> </table>	GPIO Extension Board		3V3	SVO	SDA1	SVO	SCL1	GND	GPIO4	TX00	GND	RX00	GPIO17	GPIO18	GPIO27	GND	GPIO22	GPIO23	3V	GPIO24	SPI MOSI	GND	SPI MISO	GPIO25	SPI SCLK	SPICE0	GND	SPICE1	ID_SD	ID_SC	GPIO5	GND	GPIO6	GPIO16	GPIO13	GND	GPIO19	GPIO16	GPIO26	GPIO16	GND	GPIO20		GPIO21	
GPIO Extension Board																																														
3V3	SVO																																													
SDA1	SVO																																													
SCL1	GND																																													
GPIO4	TX00																																													
GND	RX00																																													
GPIO17	GPIO18																																													
GPIO27	GND																																													
GPIO22	GPIO23																																													
3V	GPIO24																																													
SPI MOSI	GND																																													
SPI MISO	GPIO25																																													
SPI SCLK	SPICE0																																													
GND	SPICE1																																													
ID_SD	ID_SC																																													
GPIO5	GND																																													
GPIO6	GPIO16																																													
GPIO13	GND																																													
GPIO19	GPIO16																																													
GPIO26	GPIO16																																													
GND	GPIO20																																													
	GPIO21																																													
1 * 40-Pin Kabel	1 * ADC0834																																													
																																														
1 * Steckbrett	1 * LED	1 * S8050 NPN-Transistor																																												
																																														
1 * Widerstand (220 Ω)	1 * Widerstand 1KΩ	2 * Widerstand 10KΩ																																												
																																														



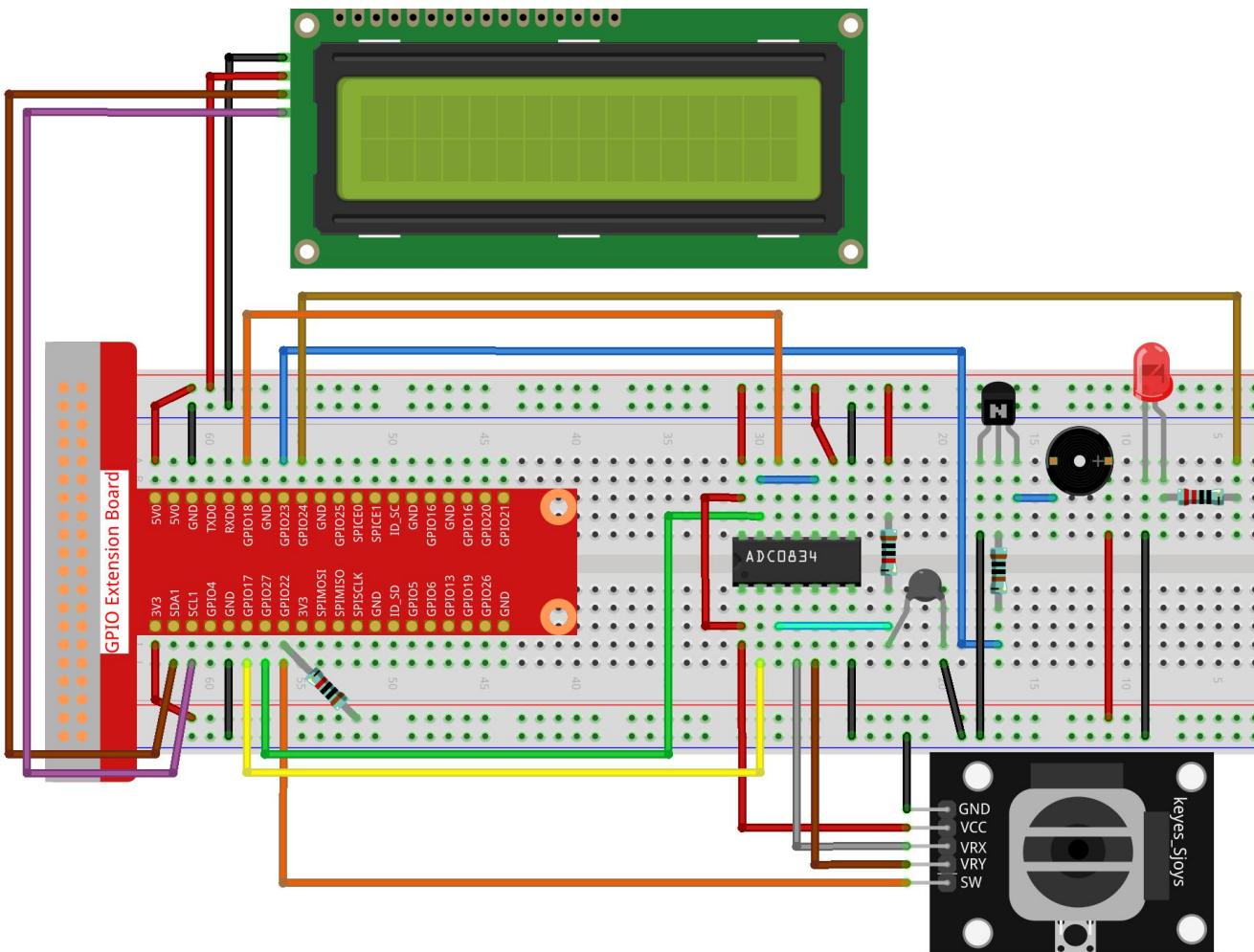
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27
GPIO22	Pin15	3	22
GPIO23	Pin16	4	23
GPIO24	Pin18	5	24
SDA1	Pin 3		
SCL1	Pin 5		



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.8/
```

Schritt 3: Kompilieren Sie der Kode.

```
gcc 3.1.8_OverheatMonitor.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Während die Kode ausgeführt wird, werden die aktuelle Temperatur und der Hochtemperaturschwellenwert **40** auf dem **I2C LCD1602** angezeigt. Wenn die aktuelle Temperatur größer als der Schwellenwert ist, werden der Summer und die LED gestartet, um Sie zu alarmieren.

Der **Joystick** dient zum Drücken, um die Hochtemperaturschwelle anzupassen. Durch Umschalten des **Joysticks** in Richtung X-Achse und Y-Achse kann der aktuelle

Hochtemperaturschwellenwert angepasst (nach oben oder unten gedreht) werden. Drücken Sie den **Joystick** erneut, um den Schwellenwert auf den Anfangswert zurückzusetzen.

Kode Erklärung

```
int get_joystick_value(){
    uchar x_val;
    uchar y_val;
    x_val = get_ADC_Result(1);
    y_val = get_ADC_Result(2);
    if (x_val > 200){
        return 1;
    }
    else if(x_val < 50){
        return -1;
    }
    else if(y_val > 200){
        return -10;
    }
    else if(y_val < 50){
        return 10;
    }
    else{
        return 0;
    }
}
```

Diese Funktion liest die Werte von X und Y. Wenn **X> 200** ist, wird "**1**" zurückgegeben. **X<50**, return "**-1**"; **y> 200**, geben Sie "**-10**" zurück, und **y<50**, geben Sie "**10**" zurück.

```
void upper_tem_setting(){
    write(0, 0, "Upper Adjust:");
    int change = get_joystick_value();
    upperTem = upperTem + change;
    char str[6];
    sprintf(str,3,"%d",upperTem);
    write(0,1,str);
    int len;
    len = strlen(str);
    write(len,1,");
```

```
delay(100);
}
```

This function is for adjusting the threshold and displaying it on the I2C LCD1602.

```
double temperature(){
    unsigned char temp_value;
    double Vr, Rt, temp, cel, Fah;
    temp_value = get_ADC_Result(0);
    Vr = 5 * (double)(temp_value) / 255;
    Rt = 10000 * (double)(Vr) / (5 - (double)(Vr));
    temp = 1 / (((log(Rt/10000)) / 3950)+(1 / (273.15 + 25)));
    cel = temp - 273.15;
    Fah = cel * 1.8 +32;
    return cel;
}
```

Lesen Sie den Analogwert des **CH0** (Thermistor) von **ADC0834** ab und wandeln Sie ihn dann in einen Temperaturwert um.

```
void monitoring_temp(){
    char str[6];
    double cel = temperature();
    sprintf(str, "%2f", cel);
    write(0, 0, "Temp: ");
    write(6, 0, str);
    sprintf(str, "%d", upperTem);
    write(0, 1, "Upper: ");
    write(7, 1, str);
    delay(100);
    if(cel >= upperTem){
        digitalWrite(buzzPin, HIGH);
        digitalWrite(LedPin, HIGH);
    }
    else if(cel < upperTem){
        digitalWrite(buzzPin, LOW);
        digitalWrite(LedPin, LOW);
    }
}
```

Während die Kode ausgeführt wird, werden die aktuelle Temperatur und der Hochtemperaturschwellenwert **40** auf dem **I2C LCD1602** angezeigt. Wenn die

aktuelle Temperatur größer als der Schwellenwert ist, werden der Summer und die LED gestartet, um Sie zu alarmieren.

```
int main(void)
{
    setup();
    int lastState =1;
    int stage=0;
    while (1)
    {
        int currentState = digitalRead(Joy_BtnPin);
        if(currentState==1 && lastState == 0){
            stage=(stage+1)%2;
            delay(100);
            lcd_clear();
        }
        lastState=currentState;
        if (stage==1){
            upper_tem_setting();
        }
        else{
            monitoring_temp();
        }
    }
    return 0;
}
```

Die Funktion main () enthält den gesamten Programmablauf wie folgt:

- 1) Wenn das Programm startet, ist der Anfangswert der **Stufe 0**, und die aktuelle Temperatur und der Hochtemperaturschwellenwert **40** werden auf dem **I2C LCD1602** angezeigt. Wenn die aktuelle Temperatur größer als der Schwellenwert ist, werden der Summer und die LED gestartet, um Sie zu alarmieren.
- 2) Drücken Sie den Joystick, und die **Stufe** ist **1**, und Sie können die Hochtemperaturschwelle einstellen. Durch Umschalten des Joysticks in Richtung X-Achse und Y-Achse kann der aktuelle Schwellenwert angepasst (nach oben oder unten gedreht) werden. Drücken Sie den Joystick erneut, um den Schwellenwert auf den Anfangswert zurückzusetzen.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 3.1.8_OverheatMonitor.py
```

Während die Kode ausgeführt wird, werden die aktuelle Temperatur und der Hochtemperaturschwellenwert **40** auf dem **I2C LCD1602** angezeigt. Wenn die aktuelle Temperatur größer als der Schwellenwert ist, werden der Summer und die LED gestartet, um Sie zu alarmieren.

Der **Joystick** dient zum Drücken, um die Hochtemperaturschwelle anzupassen. Durch Umschalten des **Joysticks** in Richtung X-Achse und Y-Achse kann der aktuelle Hochtemperaturschwellenwert angepasst (nach oben oder unten gedreht) werden. Drücken Sie den **Joystick** erneut, um den Schwellenwert auf den Anfangswert zurückzusetzen.

Kode Erklärung

```
def get_joystick_value():
    x_val = ADC0834.getResult(1)
    y_val = ADC0834.getResult(2)
    if(x_val > 200):
        return 1
    elif(x_val < 50):
        return -1
    elif(y_val > 200):
        return -10
    elif(y_val < 50):
        return 10
    else:
        return 0
```

Diese Funktion liest die Werte von X und Y. Wenn **X> 200** ist, wird "**1**" zurückgegeben. **X<50**, **return "-1"**; **y> 200**, geben Sie "**-10**" zurück, und **y<50**, geben Sie "**10**" zurück.

```
def upper_tem_setting():
    global upperTem
    LCD1602.write(0, 0, 'Upper Adjust: ')
    change = int(get_joystick_value())
    upperTem = upperTem + change
    LCD1602.write(0, 1, str(upperTem))
    LCD1602.write(len(strUpperTem), 1, '')
```

```
time.sleep(0.1)
```

Diese Funktion dient zum Einstellen des Schwellenwerts und zum Anzeigen auf dem I2C LCD1602.

```
def temperature():
    analogVal = ADC0834.getResult()
    Vr = 5 * float(analogVal) / 255
    Rt = 10000 * Vr / (5 - Vr)
    temp = 1/(((math.log(Rt / 10000)) / 3950) + (1 / (273.15+25)))
    Cel = temp - 273.15
    Fah = Cel * 1.8 + 32
    return round(Cel,2)
```

Lesen Sie den Analogwert des **CH0** (Thermistor) von **ADC0834** ab und wandeln Sie ihn dann in einen Temperaturwert um.

```
def monitoring_temp():
    global upperTem
    Cel=temperature()
    LCD1602.write(0, 0, 'Temp: ')
    LCD1602.write(0, 1, 'Upper: ')
    LCD1602.write(6, 0, str(Cel))
    LCD1602.write(7, 1, str(upperTem))
    time.sleep(0.1)
    if Cel >= upperTem:
        GPIO.output(buzzPin, GPIO.HIGH)
        GPIO.output(ledPin, GPIO.HIGH)
    else:
        GPIO.output(buzzPin, GPIO.LOW)
        GPIO.output(ledPin, GPIO.LOW)
```

Während die Kode ausgeführt wird, werden die aktuelle Temperatur und der Hochtemperaturschwellenwert **40** auf dem **I2C LCD1602** angezeigt. Wenn die aktuelle Temperatur größer als der Schwellenwert ist, werden der Summer und die LED gestartet, um Sie zu alarmieren.

```
def loop():
    lastState=1
    stage=0
    while True:
        currentState=GPIO.input(Joy_BtnPin)
        if currentState==1 and lastState ==0:
```

```

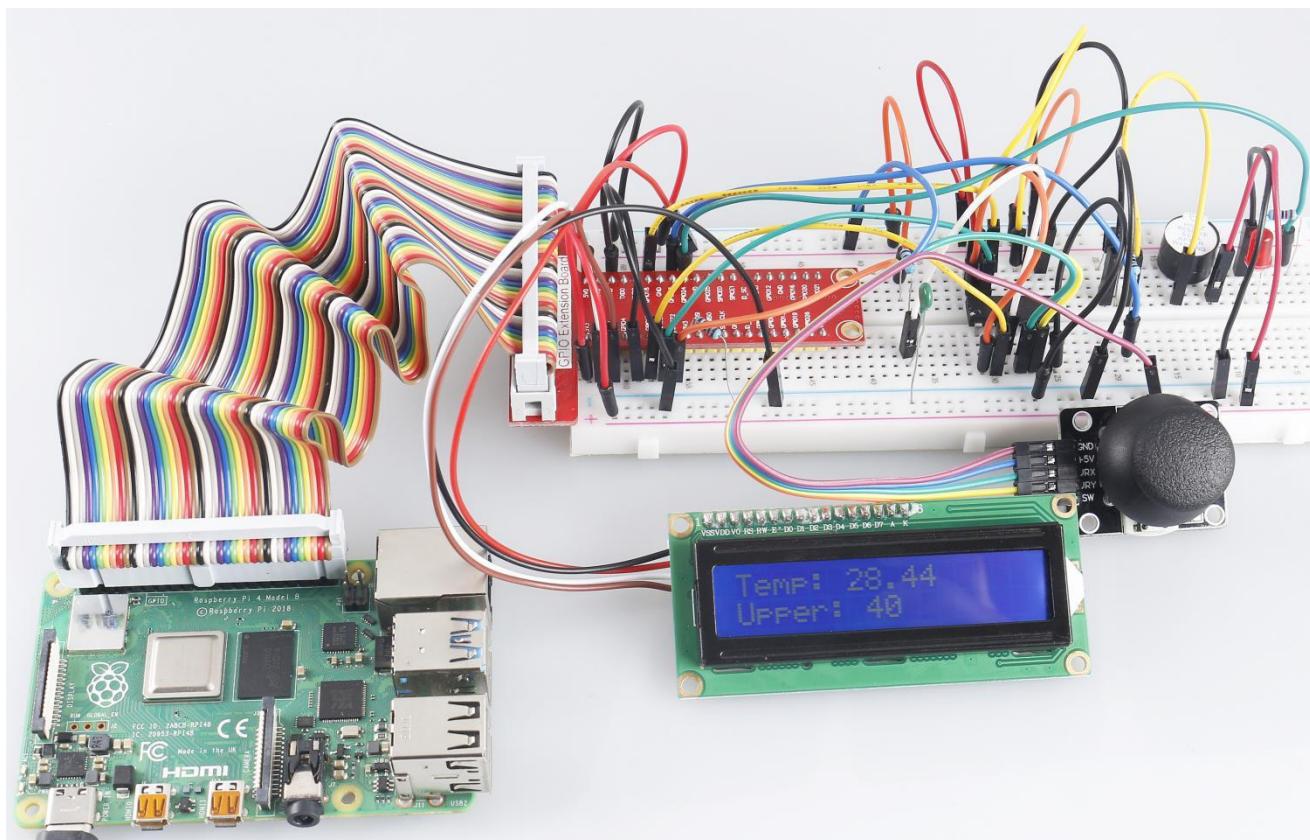
stage=(stage+1)%2
time.sleep(0.1)
LCD1602.clear()
lastState=currentState
if stage == 1:
    upper_tem_setting()
else:
    monitoring_temp()

```

Die Funktion main () enthält den gesamten Programmablauf wie folgt:

- 1) Wenn das Programm startet, ist der Anfangswert der **Stufe 0**, und die aktuelle Temperatur und der Hochtemperaturschwellenwert **40** werden auf dem **I2C LCD1602** angezeigt. Wenn die aktuelle Temperatur größer als der Schwellenwert ist, werden der Summer und die LED gestartet, um Sie zu alarmieren.
- 2) Drücken Sie den Joystick, und die **Stufe** ist **1**, und Sie können die Hochtemperaturschwelle einstellen. Durch Umschalten des Joysticks in Richtung X-Achse und Y-Achse kann der aktuelle Hochtemperaturschwellenwert angepasst (nach oben oder unten gedreht) werden. Drücken Sie den Joystick erneut, um den Schwellenwert auf den Anfangswert zurückzusetzen.

Phänomen Bild



3.1.9 Passwortsperre

Einführung

In diesem Projekt werden wir eine Tastatur und ein LCD verwenden, um ein Nummerschloss herzustellen. Auf dem LCD wird eine entsprechende Aufforderung angezeigt, Ihr Passwort auf der Tastatur einzugeben. Wenn das Passwort korrekt eingegeben wurde, wird „Richtig“ angezeigt.

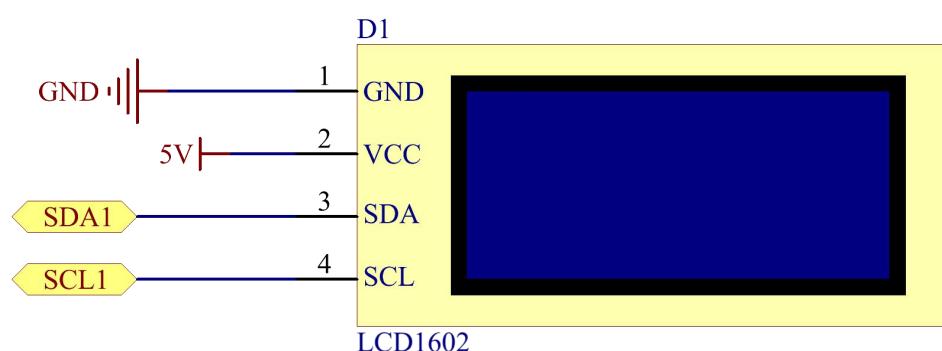
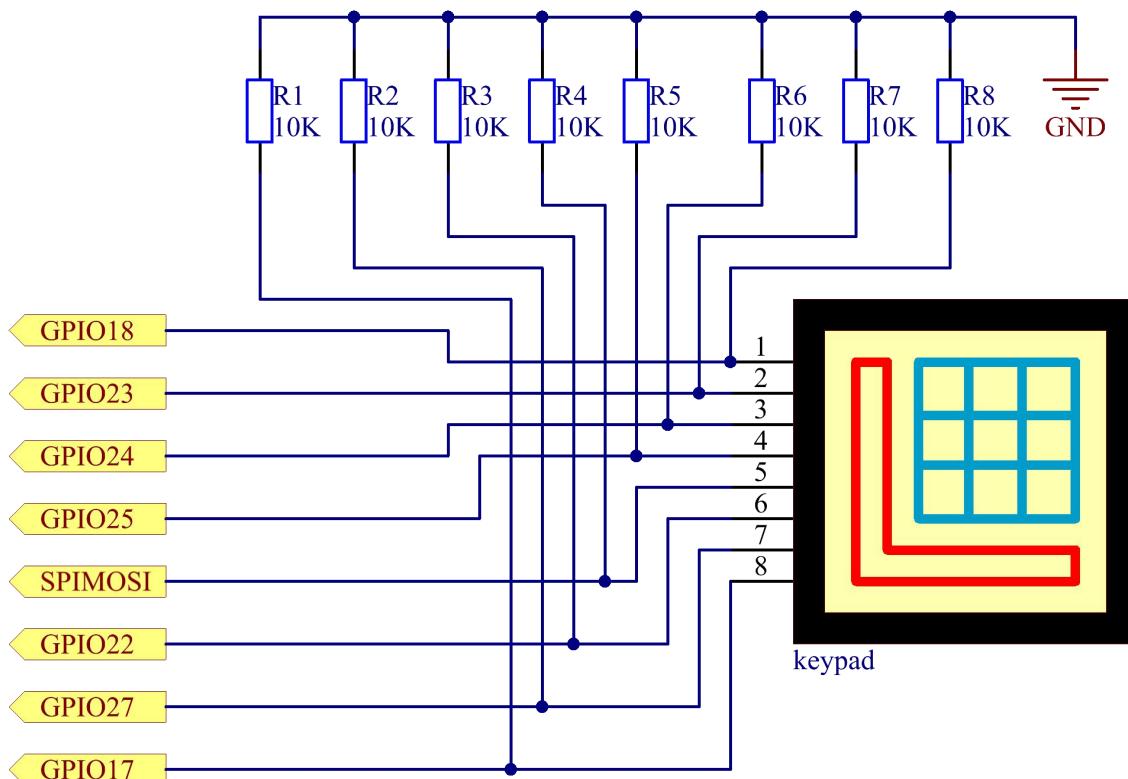
Auf der Grundlage dieses Projekts können wir zusätzliche elektronische Komponenten wie Summer, LED usw. hinzufügen, um verschiedene experimentelle Phänomene für die Passworteingabe hinzuzufügen.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * I2C LCD1602
	 T-Cobbler Plus for Raspberry Pi Pinout table: 3V3 SDA SCL GND VIO TxD GND RxD #17 #18 #27 GND #22 #23 3.3V GND MOSI MISO SCLK CE0 EEO GND #6 #13 #25 #26 GND #20 #21	
1 * 40-Pin Kabel	Mehrere Überbrückungsdrähte	1 * Tastatur
1 * Steckbrett		8 * Widerstand 10KΩ

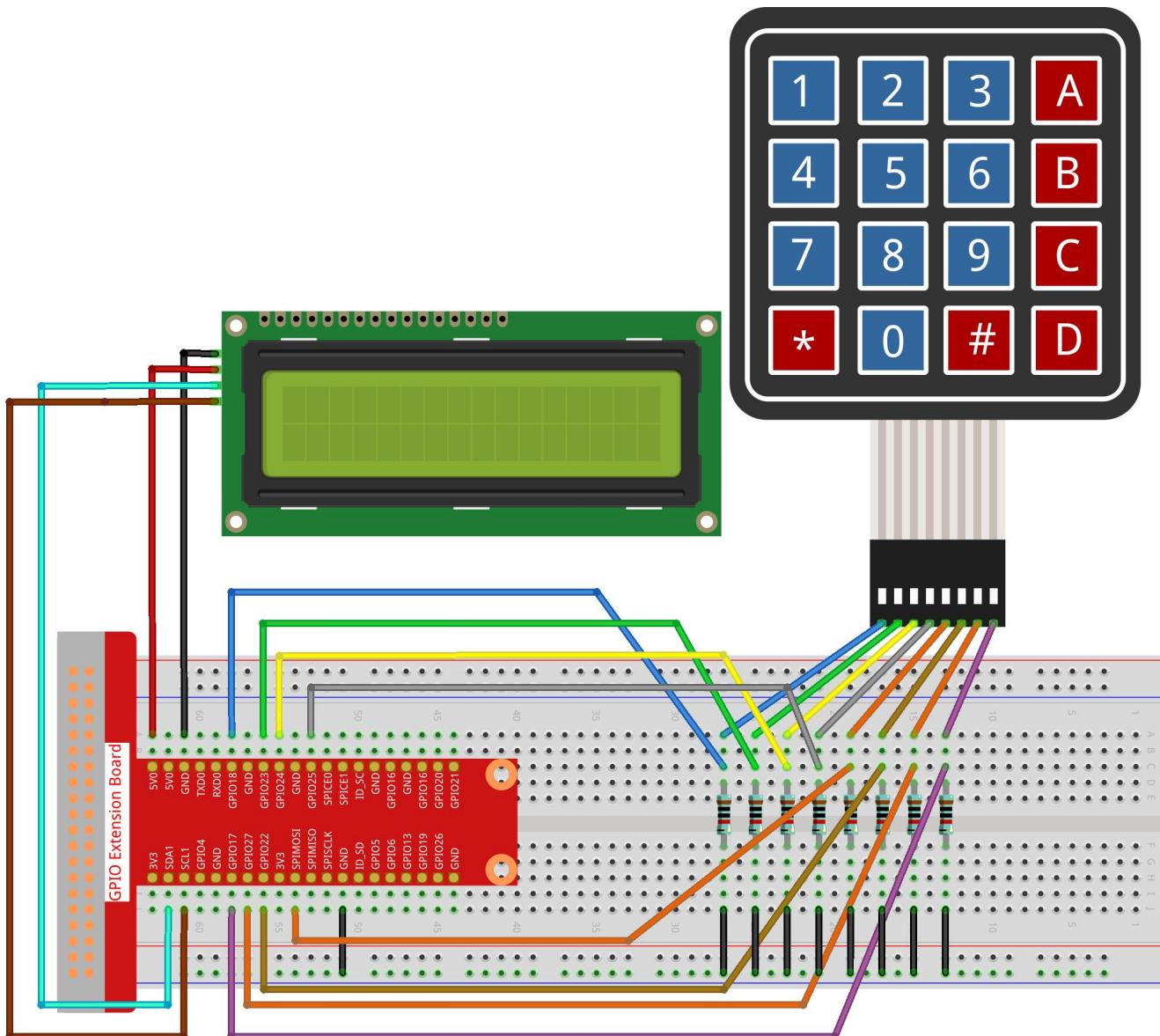
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO18	Pin 12	1	18
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24
GPIO25	Pin 22	6	25
GPIO17	Pin 11	0	17
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22
SPI MOSI	Pin 19	12	10
SDA1	Pin 3		
SCL1	Pin 5		



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.9/
```

Schritt 3: Kompilieren.

```
gcc 3.1.9_PasswordLock.cpp -lwiringPi
```

Schritt 4: Ausführen.

```
sudo ./a.out
```

Nachdem der Kode ausgeführt wurde, wird die Tastatur über die Tastatur eingegeben. Wenn auf dem LCD1602 die Meldung „RICHTIG“ angezeigt wird, ist das Kennwort nicht falsch. Andernfalls wird "FALSCHER SCHLÜSSEL" angezeigt.

Kode Erklärung

```
#define ROWS 4
#define COLS 4
#define BUTTON_NUM (ROWS * COLS)
#define LENS 4

unsigned char KEYS[BUTTON_NUM] {
    '1','2','3','A',
    '4','5','6','B',
    '7','8','9','C',
    '*', '0', '#', 'D'};

char password[LENS]={'1','9','8','4'};
```

Hier definieren wir die Länge des Kennworts LENS, des Schlüssel-Array-Tastenschlüssel-Arrays KEYS und des Arrays, in dem das richtige Kennwort gespeichert ist.

```
void keyRead(unsigned char* result);
bool keyCompare(unsigned char* a, unsigned char* b);
void keyCopy(unsigned char* a, unsigned char* b);
void keyPrint(unsigned char* a);
void keyClear(unsigned char* a);
int keyIndexOf(const char value);
```

Es gibt eine Deklaration der Unterfunktionen des Matrix-Tastaturkode. Weitere Informationen finden Sie in **Kapitel 2.1.5** dieses Dokuments.

```
void write_word(int data);
void send_command(int comm);
void send_data(int data);
void lcdInit();
void clear();
void write(int x, int y, char const data[]);
```

Es gibt eine Erklärung der Unterfunktionen der LCD1062-Kode. Weitere Informationen finden Sie in **Kapitel 1.1.7** dieses Dokuments.

```
while(1){
    keyRead(pressed_keys);
    bool comp = keyCompare(pressed_keys, last_key_pressed);
    .....
    testword[keyIndex]=pressed_keys[0];
    keyIndex++;
    if(keyIndex==LENS){
        if(check()==0){
            clear();
            write(3, 0, "WRONG KEY!");
            write(0, 1, "please try again");
        }
        .....
    }
}
```

Lesen Sie den Schlüsselwert und speichern Sie ihn im Testarray-Testwort. Wenn die Anzahl der gespeicherten Schlüsselwerte mehr als 4 beträgt, wird die Richtigkeit des Kennworts automatisch überprüft und die Überprüfungsergebnisse werden auf der LCD-Oberfläche angezeigt.

```
int check(){
    for(int i=0;i<LENS;i++){
        if(password[i]!=testword[i])
            {return 0;}
    }
    return 1;
}
```

Überprüfen Sie die Richtigkeit des Passworts. Geben Sie 1 zurück, wenn das Passwort korrekt eingegeben wurde, und 0, wenn nicht.

➤ Für Python-Sprachbenutzer

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Ausführen.

```
sudo python3 3.1.9_PasswordLock.py
```

Nachdem der Kode ausgeführt wurde, wird über die Tastatur das Kennwort eingegeben: 1984. Wenn auf dem LCD1602 die Meldung „RICHTIG“ angezeigt wird, ist das Kennwort nicht falsch. Andernfalls wird "FALSCHER SCHLÜSSEL" angezeigt.

Kode Erklärung

```
LENS = 4
password=['1','9','8','4']
.....
rowsPins = [18,23,24,25]
colsPins = [10,22,27,17]
keys = ["1","2","3","A",
        "4","5","6","B",
        "7","8","9","C",
        "*","0","#","D"]
```

Hier definieren wir die Länge des Passworts LENS, die Array-Tasten, in denen die Matrix-Tastaturtasten gespeichert sind, und das Array-Passwort, in dem das richtige Passwort gespeichert ist.

```
class Keypad():
    def __init__(self, rowsPins, colsPins, keys):
        self.rowsPins = rowsPins
        self.colsPins = colsPins
        self.keys = keys
        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.rowsPins, GPIO.OUT, initial=GPIO.LOW)
        GPIO.setup(self.colsPins, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
    ...
```

Diese Klasse ist der Kode, der die Werte der gedrückten Tasten liest. Weitere Informationen finden Sie in **Kapitel 2.1.5** dieses Dokuments.

```
while(True):
    pressed_keys = keypad.read()
    if len(pressed_keys) != 0 and last_key_pressed != pressed_keys:
        LCD1602.clear()
        LCD1602.write(0, 0, "Enter password:")
        LCD1602.write(15-keyIndex, 1, pressed_keys)
        testword[keyIndex]=pressed_keys
        keyIndex+=1
```

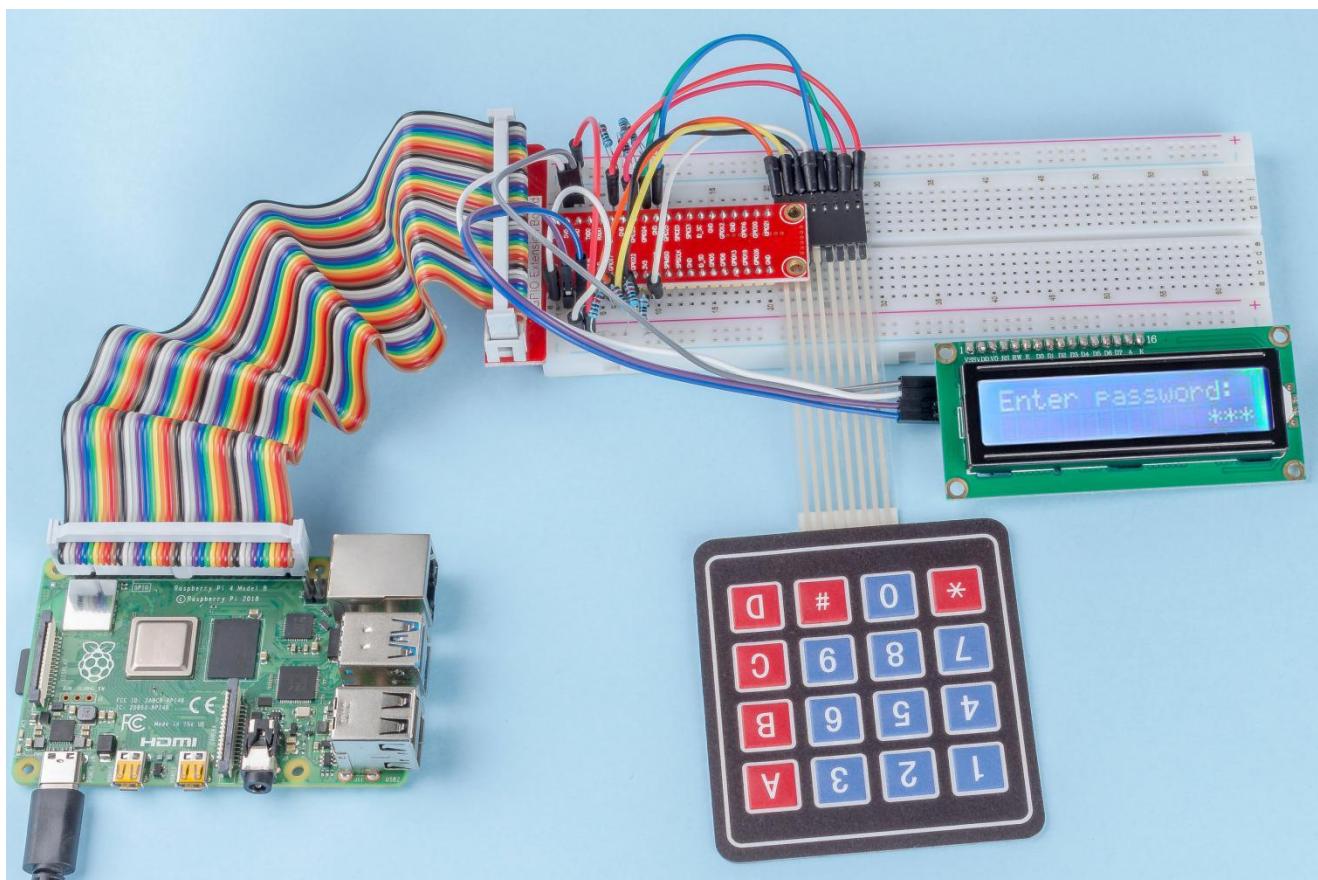
...

Lesen Sie den Schlüsselwert und speichern Sie ihn im Testarray-Testwort. Wenn die Anzahl der gespeicherten Schlüsselwerte mehr als 4 beträgt, wird die Richtigkeit des Kennworts automatisch überprüft und die Überprüfungsergebnisse werden auf der LCD-Oberfläche angezeigt.

```
def check():
    for i in range(0,LENS):
        if(password[i]!=testword[i]):
            return 0
    return 1
```

Überprüfen Sie die Richtigkeit des Passworts. Geben Sie 1 zurück, wenn das Passwort korrekt eingegeben wurde, und 0, wenn nicht.

Phänomen Bild

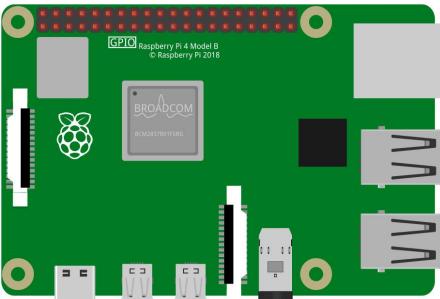
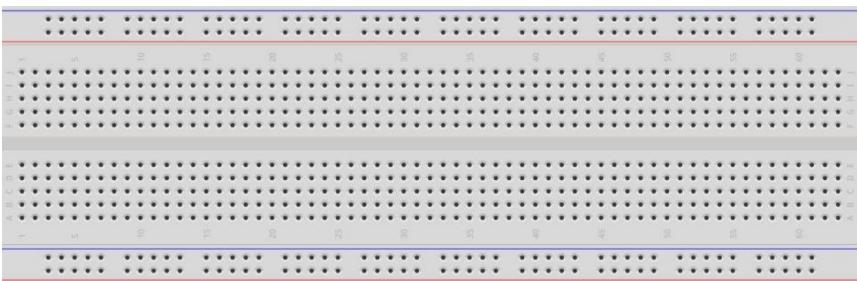
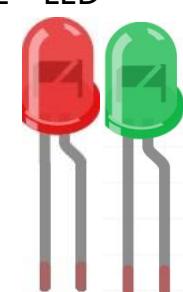
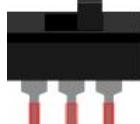


3.1.10 Alarmglocke

Einführung

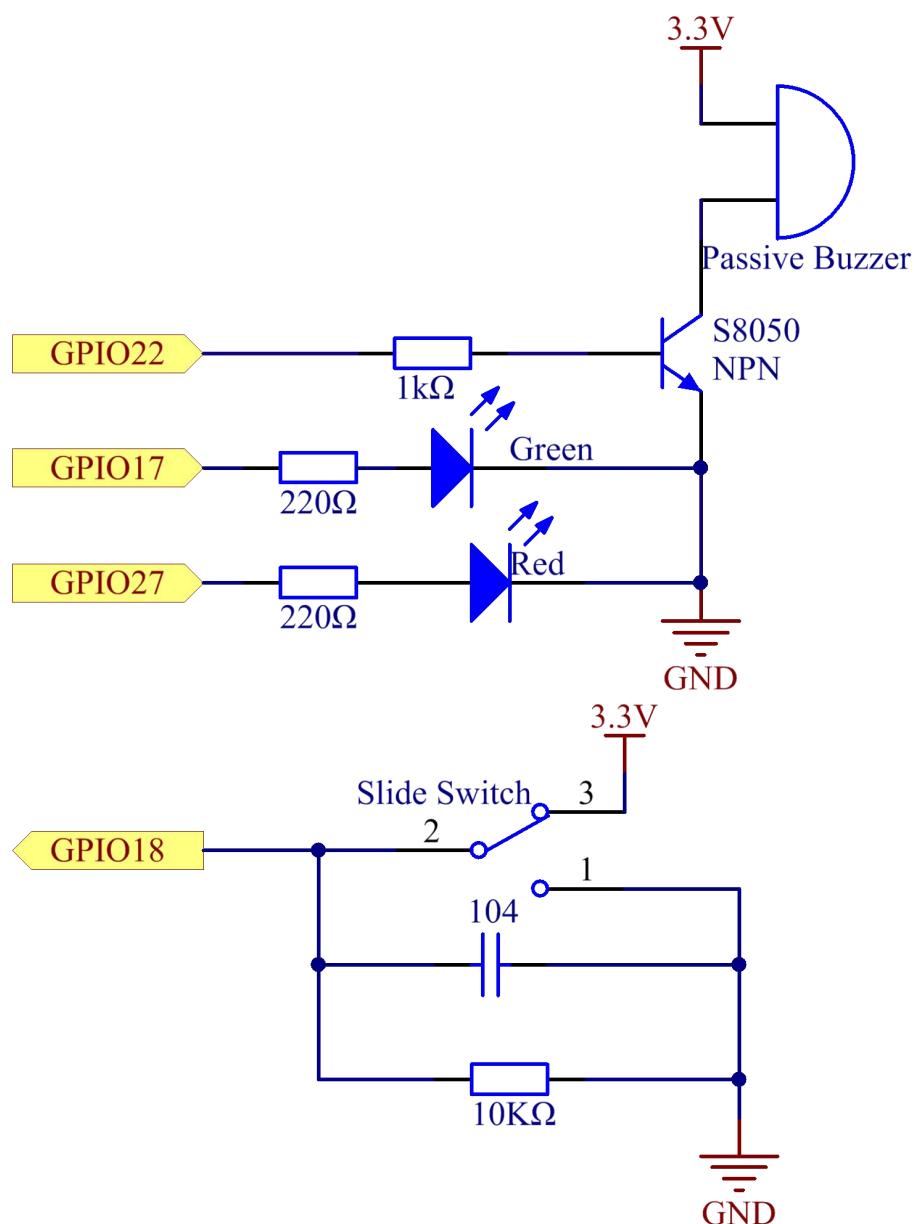
In diesem Kurs erstellen wir ein manuelles Alarmgerät. Sie können den Kippschalter durch einen Thermistor oder einen lichtempfindlichen Sensor ersetzen, um einen Temperaturalarm oder einen Lichtalarm auszulösen.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Passiver Summer
	 Pinout: <ul style="list-style-type: none">● 3V3 SVO● SDA1 SVO● SCL1 GND● GPIO4 TXD0● GND RXD0● GPIO17 GPIO18● GPIO27 GND● GPIO22 GPIO23● GND GPIO24● SPIMOSI GND● SPIMISO GPIO15● SPISCLK SPICE0● GND SPICE1● ID_SD GND● GPIO5 GPIO16● GPIO13 GPIO17● GPIO19 GPIO16● GPIO26 GPIO20● GND GPIO21	
1 * 40-Pin Kabel		1 * Widerstand (1 kΩ)
		
1 * Steckbrett		2 * Widerstand (220 Ω) 1 * Widerstand 10 kΩ
		 
Mehrere Überbrückungsdrähte	2 * LED	1 * Schiebeschalter
		
	1 * S85050 NPN-Transistor	1 * 104 Kondensator
		

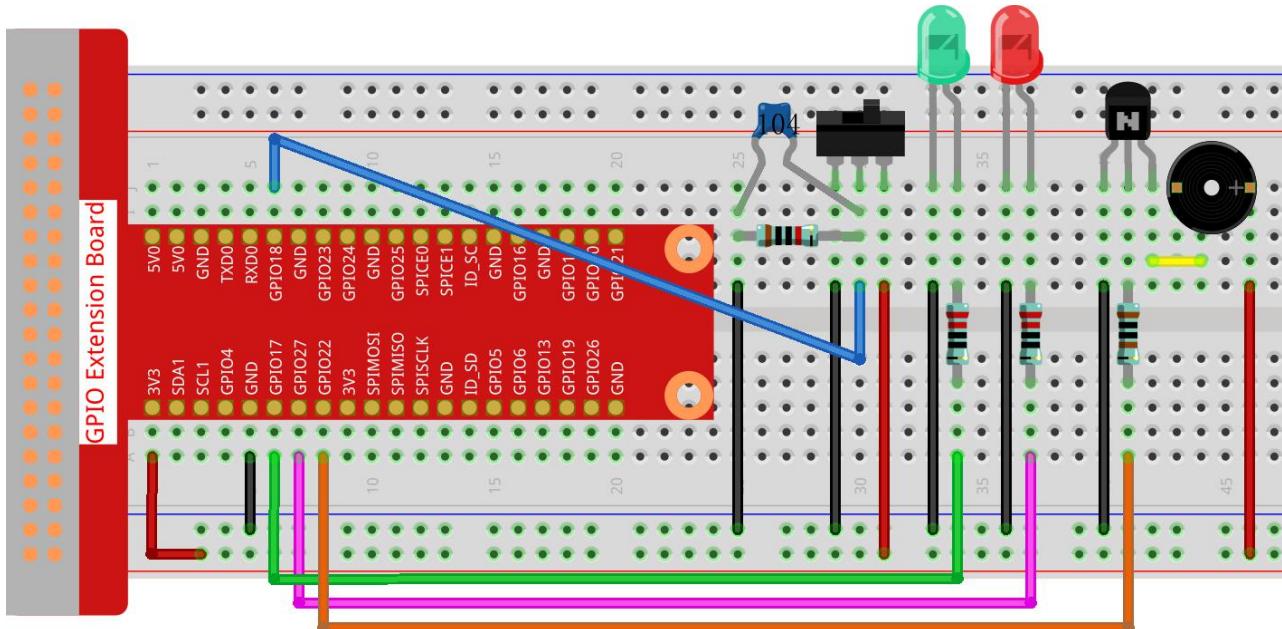
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.10/
```

Schritt 3: Kompilieren.

```
gcc 3.1.10_AlarmBell.c -lwiringPi -lpthread
```

Schritt 4: Ausführen.

```
sudo ./a.out
```

Nach dem Start des Programms wird der Kippschalter nach rechts umgeschaltet und der Summer gibt Alarmtöne aus. Gleichzeitig blinken die roten und grünen LEDs mit einer bestimmten Frequenz.

Kode Erklärung

```
#include <pthread.h>
```

In dieser Kode verwenden Sie eine neue Bibliothek, pthread.h, die aus einer Reihe allgemeiner Thread-Bibliotheken besteht und Multithreading realisieren kann. Wir fügen den Parameter **-lpthread** zur Kompilierungszeit hinzu, damit die LED und der Summer unabhängig voneinander arbeiten können.

```
void *ledWork(void *arg){  
    while(1)
```

```
{
    if(flag==0){
        pthread_exit(NULL);
    }
    digitalWrite(ALedPin,HIGH);
    delay(500);
    digitalWrite(ALedPin,LOW);
    digitalWrite(BLedPin,HIGH);
    delay(500);
    digitalWrite(BLedPin,LOW);
}
}
```

Die Funktion ledWork() hilft beim Einstellen des Arbeitszustands dieser beiden LEDs: Sie leuchtet die grüne LED 0,5 Sekunden lang auf und erlischt dann. In ähnlicher Weise leuchtet die rote LED 0,5 Sekunden lang auf und erlischt dann.

```
void *buzzWork(void *arg){
while(1)
{
    if(flag==0){
        pthread_exit(NULL);
    }
    if((note>=800)|| (note<=130)){
        pitch = -pitch;
    }
    note=note+pitch;
    softToneWrite(BeepPin,note);
    delay(10);
}
}
```

Mit der Funktion summWork() wird der Arbeitszustand des Summers eingestellt. Hier stellen wir die Frequenz zwischen 130 und 800 ein, um sie in einem Intervall von 20 zu akkumulieren oder abzunehmen.

```
void on(){
flag = 1;
if(softToneCreate(BeepPin) == -1){
    printf("setup softTone failed !");
    return;
}
```

```

pthread_t tLed;
pthread_create(&tLed,NULL,ledWork,NULL);
pthread_t tBuzz;
pthread_create(&tBuzz,NULL,buzzWork,NULL);
}

```

In der Funktion on ():

- 1) Definieren Sie die Markierung „flag = 1“, die das Ende des Kontrollthreads angibt.
- 2) Erstellen Sie einen softwaregesteuerten Ton-Pin **BeepPin**.
- 3) Erstellen Sie zwei separate Threads, damit die LED und der Summer gleichzeitig arbeiten können.

pthread_t tLed: Deklariert einen Thread **tLed**.

pthread_create(&tLed,NULL,ledWork,NULL): Erstellen Sie den Thread und sein Prototyp lautet wie folgt:

```
int pthread_create (pthread_t * tidp einschränken, const pthread_attr_t * einschränken_attr, void * (* start_rtn) (void *), void * arg einschränken);
```

Geben Sie den Wert zurück

Wenn dies erfolgreich ist, geben Sie "0" zurück. Andernfalls geben Sie die Fallzahl "-1" zurück.

Parameter

Der erste Parameter ist ein Zeiger auf die Thread-ID.

Der zweite wird verwendet, um das Thread-Attribut festzulegen.

Die dritte ist die Startadresse der Thread-Running-Funktion.

Der letzte ist derjenige, der die Funktion ausführt.

```

void off(){
    flag = 0;
    softToneStop(BeepPin);
    digitalWrite(ALedPin,LOW);
    digitalWrite(BLedPin,LOW);
}

```

Die Funktion Off() definiert "flag=0", um die Threads **ledWork** und **BuzzWork** zu verlassen und dann den Summer und die LED auszuschalten.

```

int main(){
    setup();
    int lastState = 0;
}

```

```

while(1){
    int currentState = digitalRead(switchPin);
    if ((currentState == 1)&&(lastState==0)){
        on();
    }
    else if((currentState == 0)&&(lastState==1)){
        off();
    }
    lastState=currentState;
}
return 0;
}

```

Main() enthält den gesamten Prozess des Programms: Lesen Sie zuerst den Wert des Schiebeschalters; Wenn der Kippschalter nach rechts umgeschaltet ist (der Messwert ist 1), wird die Funktion on () aufgerufen, der Summer wird zur Ausgabe von Tönen angesteuert und die rote und die grüne LED blinken. Andernfalls funktionieren der Summer und die LED nicht.

➤ Für Python-Sprachbenutzer

Schritt 2: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Ausführen.

```
sudo python3 3.1.10_AlarmBell.py
```

Nach dem Start des Programms wird der Kippschalter nach rechts umgeschaltet und der Summer gibt Alarmtöne aus. Gleichzeitig blinken die roten und grünen LEDs mit einer bestimmten Frequenz.

Kode Erklärung

importieren Threading

Hier importieren wir das **Threading**-Modul und es ermöglicht Ihnen, mehrere Dinge gleichzeitig zu tun, während normale Programme Kode nur von oben nach unten ausführen können. Bei **Threading**-Modulen können die LED und der Summer separat arbeiten.

```

def ledWork():
    while flag:
        GPIO.output(ALedPin,GPIO.HIGH)
        time.sleep(0.5)

```

```
GPIO.output(ALedPin,GPIO.LOW)
GPIO.output(BLedPin,GPIO.HIGH)
time.sleep(0.5)
GPIO.output(BLedPin,GPIO.LOW)
```

Die Funktion ledWork() hilft beim Einstellen des Arbeitszustands dieser beiden LEDs: Sie leuchtet die grüne LED 0,5 Sekunden lang auf und erlischt dann. In ähnlicher Weise leuchtet die rote LED 0,5 Sekunden lang auf und erlischt dann.

```
def buzzerWork():
    global pitch
    global note
    while flag:
        if note >= 800 or note <=130:
            pitch = -pitch
            note = note + pitch
            Buzz.ChangeFrequency(note)
            time.sleep(0.01)
```

Mit der Funktion summWork() wird der Arbeitszustand des Summers eingestellt. Hier stellen wir die Frequenz zwischen 130 und 800 ein, um sie in einem Intervall von 20 zu akkumulieren oder abzunehmen.

```
def on():
    global flag
    flag = 1
    Buzz.start(50)
    tBuzz = threading.Thread(target=buzzerWork)
    tBuzz.start()
    tLed = threading.Thread(target=ledWork)
    tLed.start()
```

In der Funktion on ():

- 1) Definieren Sie die Markierung „flag = 1“, die das Ende des Kontrollthreads angibt.
- 2) Starten Sie den Buzz und stellen Sie den Arbeitszyklus auf 50% ein.
- 3) Erstellen Sie **2** separate Threads, damit die LED und der Summer gleichzeitig arbeiten können.

tBuzz = threading.Thread (target = buzzerWork): Erstellen Sie den Thread und sein Prototyp lautet wie folgt:

```
class threading.Thread (group=None, target=None, name=None, args=(), kwargs={}, *, daemon=None)
```

Unter den Konstruktionsmethoden ist der Hauptparameter **target**. Wir müssen dem Ziel ein aufrufbares Objekt zuweisen (hier sind die Funktionen **ledWork** und **BuzzWork**).

Nach **start()** wird aufgerufen, um das Thread-Objekt zu starten. Beispiel: `tBuzz.start ()` wird verwendet, um den neu installierten tBuzz-Thread zu starten.

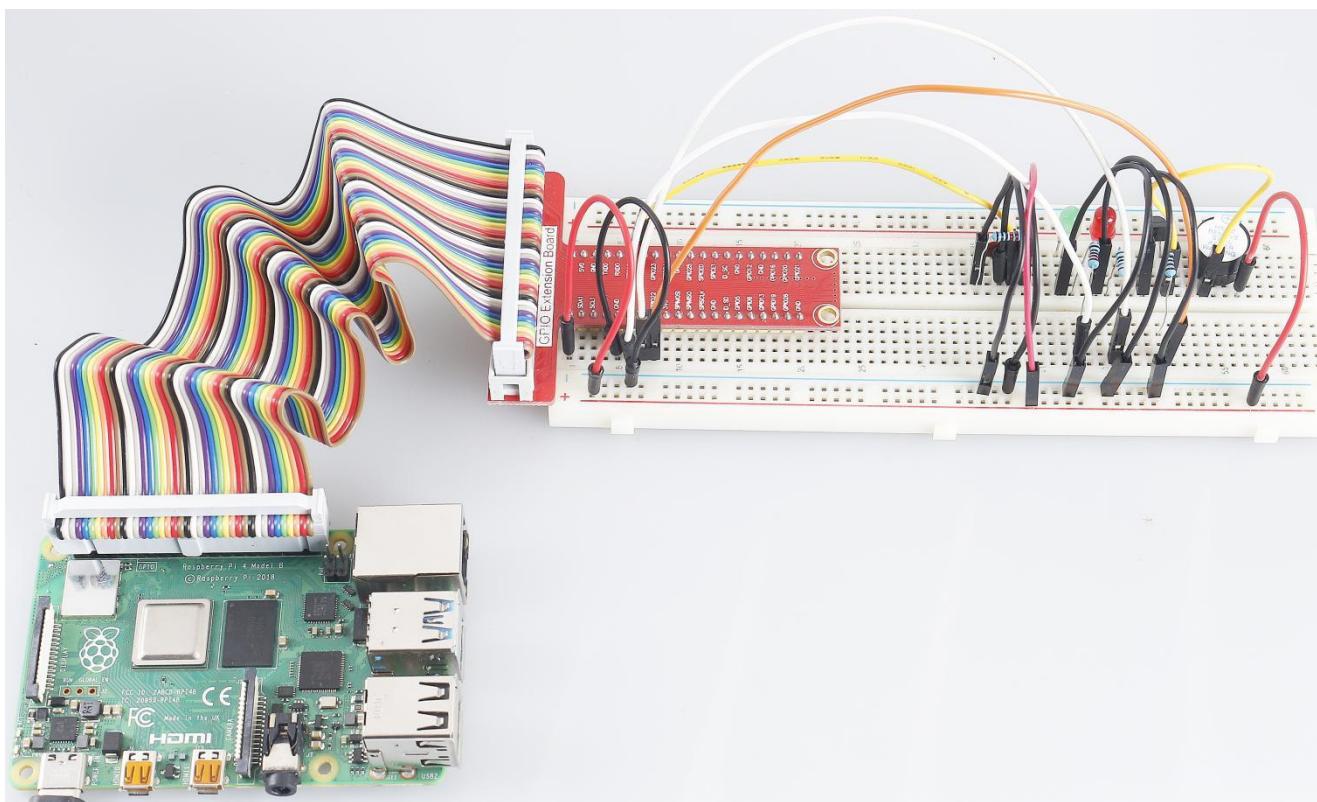
```
def off():
    global flag
    flag = 0
    Buzz.stop()
    GPIO.output(ALedPin,GPIO.LOW)
    GPIO.output(BLedPin,GPIO.LOW)
```

Die Funktion Off() definiert "flag=0", um die Threads **ledWork** und **BuzzWork** zu verlassen und dann den Summer und die LED auszuschalten.

```
def main():
    lastState=0
    while True:
        currentState =GPIO.input(switchPin)
        if currentState == 1 and lastState == 0:
            on()
        elif currentState == 0 and lastState == 1:
            off()
        lastState=currentState
```

Main() enthält den gesamten Prozess des Programms: Lesen Sie zuerst den Wert des Schiebeschalters; Wenn der Kippschalter nach rechts umgeschaltet ist (der Messwert ist 1), wird die Funktion on () aufgerufen, der Summer wird zur Ausgabe von Tönen angesteuert und die rote und die grüne LED blinken. Andernfalls funktionieren der Summer und die LED nicht.

Phänomen Bild

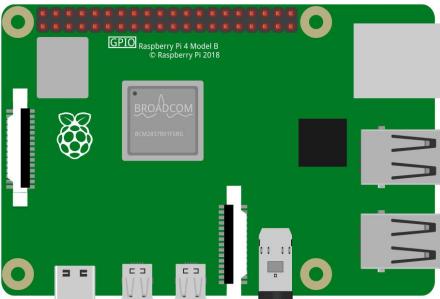
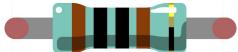
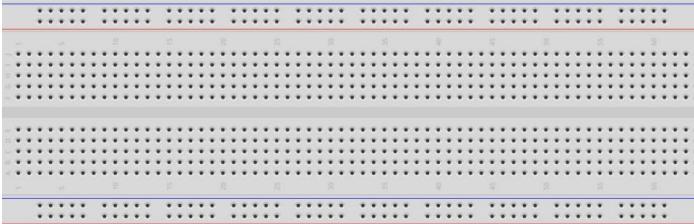


3.1.11 Morsekode-Generator

Einführung

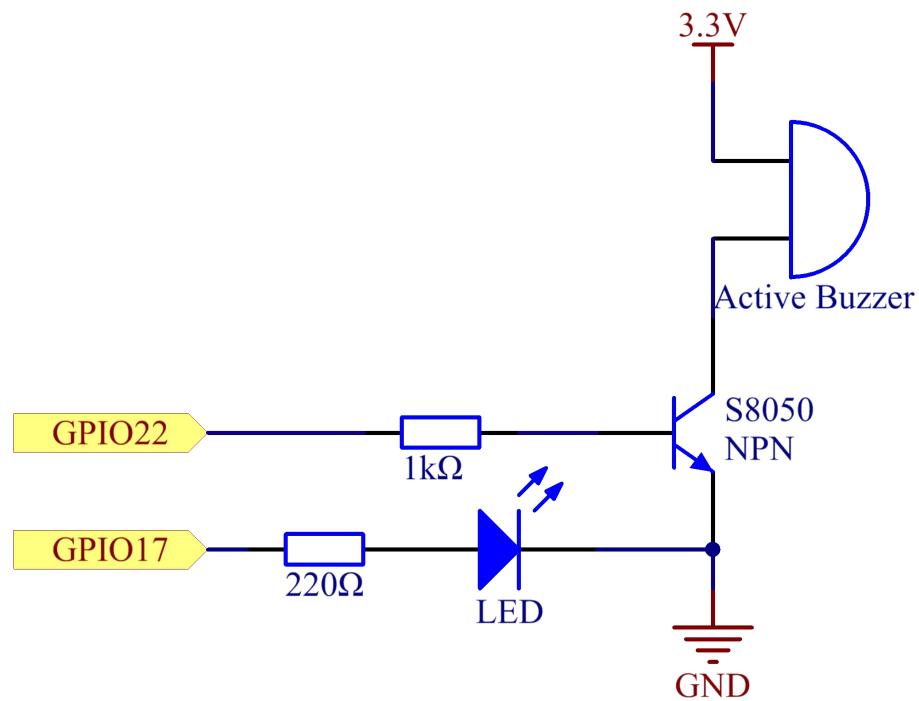
In dieser Lektion erstellen wir einen Morsekode-Generator, in den Sie eine Reihe englischer Buchstaben in den Raspberry Pi eingeben, damit er als Morsekode angezeigt wird.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Aktiver Summer																																												
	 <table border="1"> <tr><th colspan="2">GPIO Extension Board</th></tr> <tr><td>3V</td><td>SV</td></tr> <tr><td>SDA1</td><td>SVD</td></tr> <tr><td>SCL1</td><td>GND</td></tr> <tr><td>GPIO4</td><td>TXDD</td></tr> <tr><td>GND</td><td>RXD0</td></tr> <tr><td>GPIO17</td><td>GPIO18</td></tr> <tr><td>GND</td><td>GPIO19</td></tr> <tr><td>GPIO27</td><td>GND</td></tr> <tr><td>GPIO22</td><td>GPIO23</td></tr> <tr><td>3V</td><td>GPIO24</td></tr> <tr><td>SDA1</td><td>GND</td></tr> <tr><td>SCL1</td><td>GPIO25</td></tr> <tr><td>GPIO4</td><td>SPICEMOSI</td></tr> <tr><td>GND</td><td>SPIMISO</td></tr> <tr><td>ID_SD</td><td>SPISCLK</td></tr> <tr><td>GPIO5</td><td>GND</td></tr> <tr><td>GPIO6</td><td>ID_SO</td></tr> <tr><td>GPIO13</td><td>SPICECE10</td></tr> <tr><td>GPIO19</td><td>GPIO16</td></tr> <tr><td>GPIO26</td><td>GPIO20</td></tr> <tr><td>GND</td><td>GPIO21</td></tr> </table>	GPIO Extension Board		3V	SV	SDA1	SVD	SCL1	GND	GPIO4	TXDD	GND	RXD0	GPIO17	GPIO18	GND	GPIO19	GPIO27	GND	GPIO22	GPIO23	3V	GPIO24	SDA1	GND	SCL1	GPIO25	GPIO4	SPICEMOSI	GND	SPIMISO	ID_SD	SPISCLK	GPIO5	GND	GPIO6	ID_SO	GPIO13	SPICECE10	GPIO19	GPIO16	GPIO26	GPIO20	GND	GPIO21	
GPIO Extension Board																																														
3V	SV																																													
SDA1	SVD																																													
SCL1	GND																																													
GPIO4	TXDD																																													
GND	RXD0																																													
GPIO17	GPIO18																																													
GND	GPIO19																																													
GPIO27	GND																																													
GPIO22	GPIO23																																													
3V	GPIO24																																													
SDA1	GND																																													
SCL1	GPIO25																																													
GPIO4	SPICEMOSI																																													
GND	SPIMISO																																													
ID_SD	SPISCLK																																													
GPIO5	GND																																													
GPIO6	ID_SO																																													
GPIO13	SPICECE10																																													
GPIO19	GPIO16																																													
GPIO26	GPIO20																																													
GND	GPIO21																																													
1 * 40-Pin Kabel		1 * Widerstand (1 kΩ)																																												
																																														
1 * Steckbrett	1 * LED	2 * Widerstand (220 Ω)																																												
																																														
		Mehrere Überbrückungsdrähte																																												
																																														
		1 * S85050 NPN- Transistor																																												
																																														

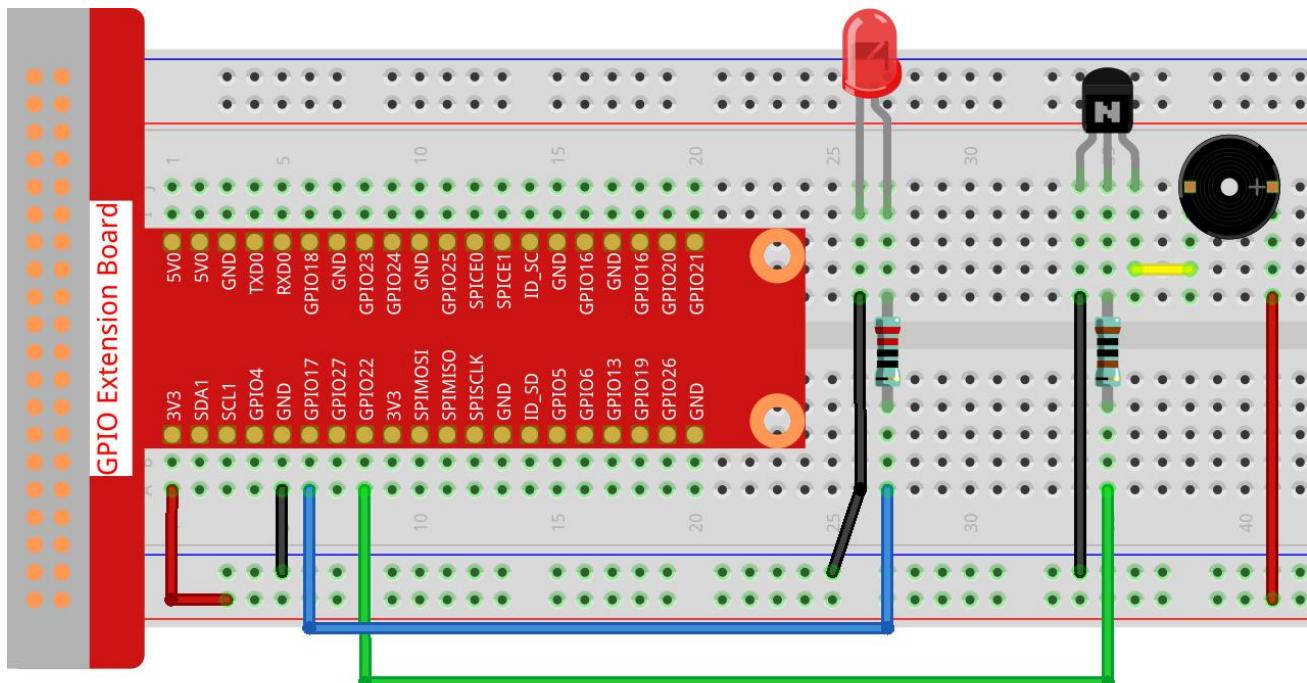
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO22	Pin 15	3	22



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf. (Achten Sie auf die Pole des Summers: Der mit dem + Etikett ist der positive Pol und der andere der negative.)



➤ Für Benutzer in C-Sprache

Schritt 2: Öffnen Sie die Kodedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.11/
```

Schritt 3: Kompilieren Sie den Kode.

```
gcc 3.1.11_MorseCodeGenerator.c -lwiringPi
```

Schritt 4: Führen Sie die obige ausführbare Datei aus.

```
sudo ./a.out
```

Geben Sie nach dem Ausführen des Programms eine Reihe von Zeichen ein, und der Summer und die LED senden die entsprechenden Morsekodesignale.

Kode Erklärung

```
struct MORSE{
    char word;
    unsigned char *code;
};

struct MORSE morseDict[]=
{
    {'A',"01"}, {'B',"1000"}, {"C","1010"}, {"D","100"}, {"E","0"}, {"F,"0010"}, {"G","110"}, {"H","0000"}, {"I","00"}, {"J","0111"}, {"K","101"}, {"L","0100"}, {"M","11"}, {"N","10"}, {"O","111"}, {"P","0110"}, {"Q","1101"}, {"R","010"}, {"S","000"}, {"T","1"}, {"U","001"}, {"V","0001"}, {"W","011"}, {"X","1001"}, {"Y","1011"}, {"Z","1100"}, {"'1',"01111"}, {"'2',"00111"}, {"'3',"00011"}, {"'4',"00001"}, {"'5',"00000"}, {"'6',"10000"}, {"'7',"11000"}, {"'8',"11100"}, {"'9',"11110"}, {"'0',"11111"}, {"'?' ,"001100"}, {"'/' , "10010"}, {"'.' , "110011"}, {"'.' , "010101"}, {"'!' , "101010"}, {"'!' , "101011"}, {"'@' , "011010"}, {"'!' , "111000"}}
```

Diese Struktur MORSE ist das Wörterbuch des Morsecodes, das die Zeichen AZ, die Nummer 0-9 und die Markierungen "?" Enthält. "/" ":" ";" ":" ";" "!" "@" .

```
char *lookup(char key,struct MORSE *dict,int length)
{
    for (int i=0;i<length;i++)
    {
        if(dict[i].word==key){
            return dict[i].code;
```

```
}
```

```
}
```

```
}
```

Die Funktion **lookup()** funktioniert durch „Überprüfen des Wörterbuchs“. Definieren Sie **einen Schlüssel**, suchen Sie die gleichen Wörter wie **den Schlüssel** in der Struktur **morseDict** und geben Sie die entsprechenden Informationen zurück - **"Kode"** des bestimmten Wortes.

```
void on(){
    digitalWrite(ALedPin,HIGH);
    digitalWrite(BeepPin,HIGH);
}
```

Erstellen Sie eine Funktion **auf()**, um den Summer und die LED zu starten.

```
void off(){
    digitalWrite(ALedPin,LOW);
    digitalWrite(BeepPin,LOW);
}
```

Die Funktion **off ()** schaltet den Summer und die LED aus.

```
void beep(int dt){
    on();
    delay(dt);
    off();
    delay(dt);
}
```

Definieren Sie einen Funktionston **()**, damit der Summer und die LED in einem bestimmten Intervall von **dt** ertönen und blinken.

```
void morsecode(char *code){
    int pause = 250;
    char *point = NULL;
    int length = sizeof(morseDict)/sizeof(morseDict[0]);
    for (int i=0;i<strlen(code);i++)
    {
        point=lookup(code[i],morseDict,length);
        for (int j=0;j<strlen(point);j++){
            if (point[j]=='0')
            {
                beep(pause/2);
            }
        }
    }
}
```

```
        }else if(point[j]=='1')  
        {  
            beep(pause);  
        }  
        delay(pause);  
    }  
}
```

Die Funktion `morsecode()` wird verwendet, um den Morsecode von Eingabezeichen zu verarbeiten, indem die "1" der Kode weiterhin Töne oder Lichter aussendet und die "0" in Kürze Töne oder Lichter aussendet, z. B. "SOS" eingibt und dort wird ein Signal sein, das drei kurze, drei lange und dann drei kurze Segmente "·" enthält.

```
int toupper(int c)
{
    if ((c >= 'a') && (c <= 'z'))
        return c + ('A' - 'a');
    return c;
}

char *strupr(char *str)
{
    char *origin=str;
    for (; *str != '\0'; str++)
        *str = toupper(*str);
    return origin;
}
```

Vor dem Codieren müssen Sie die Buchstaben in Großbuchstaben vereinheitlichen.

```
void main(){
    setup();
    char *code;
    int length=8;
    code = (char*)malloc(sizeof(char)*length);
    while (1){
        printf("Please input the messenger:");
        scanf("%s",code);
        code=strupr(code);
        printf("%s\n",code);
        morsecode(code);
    }
}
```

```
}
```

Wenn Sie die relevanten Zeichen mit der Tastatur eingeben, konvertiert code = strupper(code) die Eingabebuchstaben in ihre Großbuchstaben.

Printf() druckt dann den Klartext auf dem Computerbildschirm, und die Funktion morsecod() bewirkt, dass der Summer und die LED Morsecode ausgeben.

Beachten Sie, die Länge des Eingabezeichens darf **die Länge** nicht überschreiten(kann überarbeitet werden).

➤ Für Python-Sprachbenutzer

Schritt 2: Öffnen Sie die Kodedatei.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Schritt 3: Ausführen.

```
sudo python3 3.1.11_MorseCodeGenerator.py
```

Geben Sie nach dem Ausführen des Programms eine Reihe von Zeichen ein, und der Summer und die LED senden die entsprechenden Morsecodesignale.

Kode Erklärung

```
MORSECODE = {  
    'A':'01', 'B':'1000', 'C':'1010', 'D':'100', 'E':'0', 'F':'0010', 'G':'110',  
    'H':'0000', 'I':'00', 'J':'0111', 'K':'101', 'L':'0100', 'M':'11', 'N':'10',  
    'O':'111', 'P':'0110', 'Q':'1101', 'R':'010', 'S':'000', 'T':'1',  
    'U':'001', 'V':'0001', 'W':'011', 'X':'1001', 'Y':'1011', 'Z':'1100',  
    '1':'01111', '2':'00111', '3':'00011', '4':'00001', '5':'00000',  
    '6':'10000', '7':'11000', '8':'11100', '9':'11110', '0':'11111',  
    '?':'001100', '/':'10010', ' ':'110011', '!'':010101', ',:'101010',  
    '!':101011', '@':011010', '':111000',  
}
```

Diese Struktur MORSE ist das Wörterbuch des Morsecodes, das die Zeichen AZ, die Nummer 0-9 und die Markierungen "?" enthält.

```
def on():  
    GPIO.output(BeepPin, 1)  
    GPIO.output(ALedPin, 1)
```

Die Funktion on() startet den Summer und die LED.

```
def off():  
    GPIO.output(BeepPin, 0)
```

```
GPIO.output(ALedPin, 0)
```

Mit der Funktion off () werden der Summer und die LED ausgeschaltet.

```
def beep(dt): # x for delay time.  
    on()  
    time.sleep(dt)  
    off()  
    time.sleep(dt)
```

Definieren Sie einen Funktionston (), damit der Summer und die LED in einem bestimmten Intervall von **dt** ertönen und blinken.

```
def morsecode(code):  
    pause = 0.25  
    for letter in code:  
        for tap in MORSECODE[letter]:  
            if tap == '0':  
                beep(pause/2)  
            if tap == '1':  
                beep(pause)  
    time.sleep(pause)
```

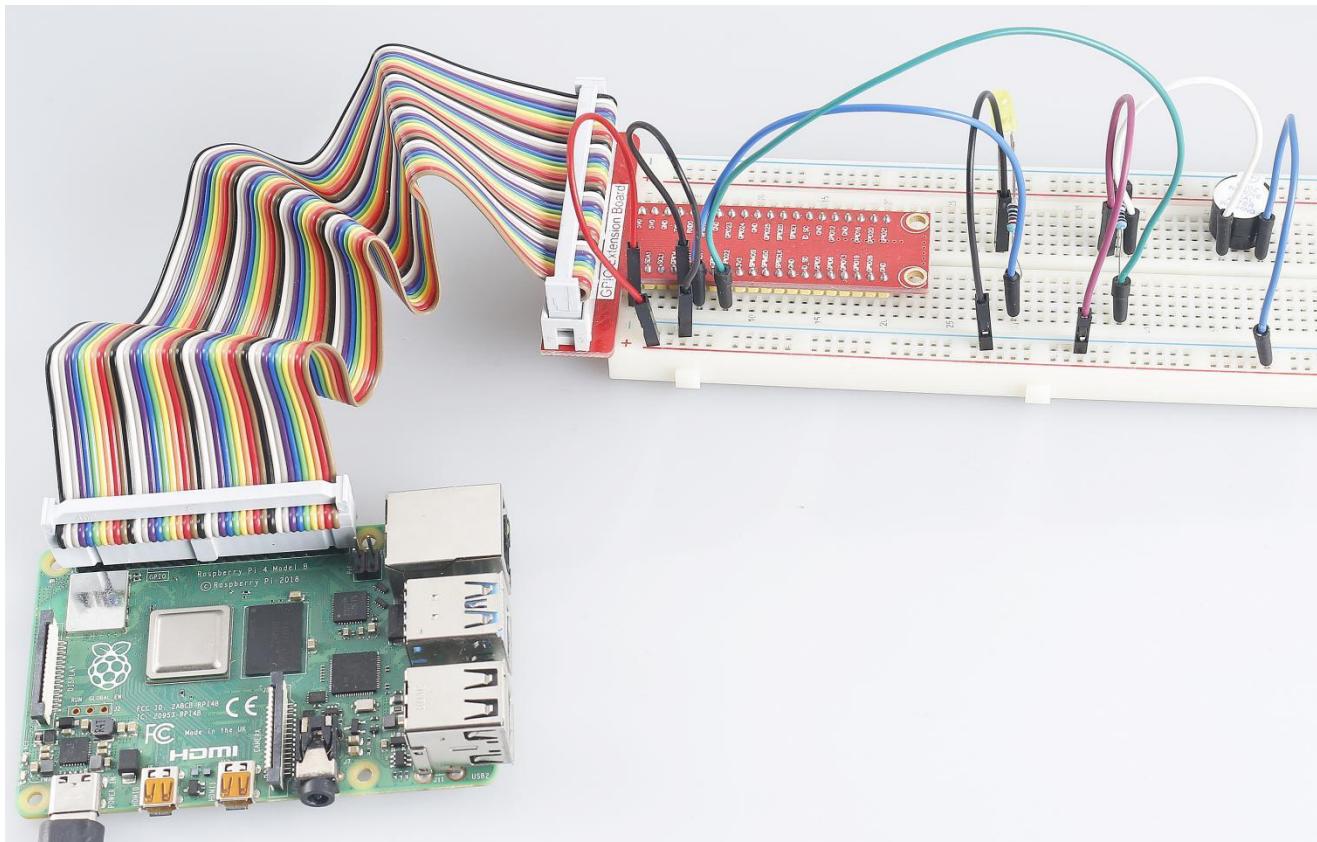
Die Funktion morsecode() wird verwendet, um den Morsecode von Eingabezeichen zu verarbeiten, indem die "1" der Kode weiterhin Töne oder Lichter aussendet und die "0" in Kürze Töne oder Lichter aussendet, z. B. "SOS" eingibt und dort wird ein Signal sein, das drei kurze, drei lange und dann drei kurze Segmente "·" enthält.

```
def main():  
    while True:  
        code=input("Please input the messenger:")  
        code = code.upper()  
        print(code)  
        morsecode(code)
```

Wenn Sie die relevanten Zeichen mit der Tastatur eingeben, konvertiert Upper() die Eingabebuchstaben in ihre Großbuchstaben.

Printf() druckt dann den Klartext auf dem Computerbildschirm, und die Funktion morsecod() bewirkt, dass der Summer und die LED Morsecode ausgeben.

Phänomen Bild

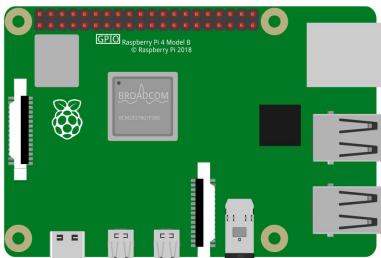
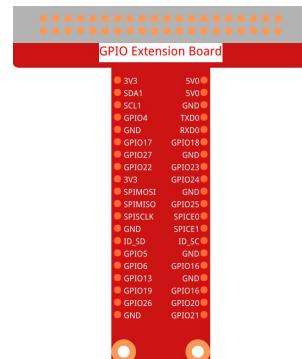
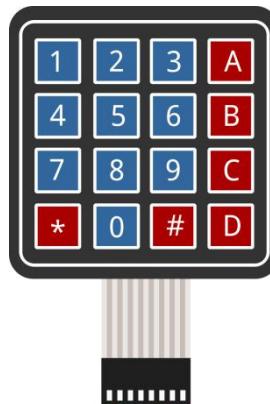
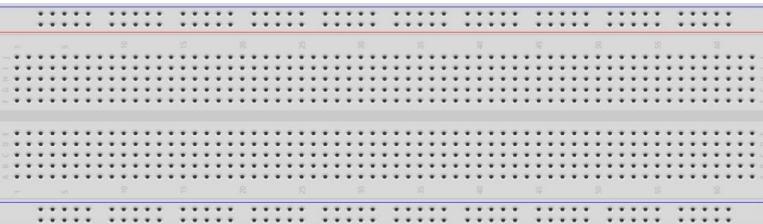
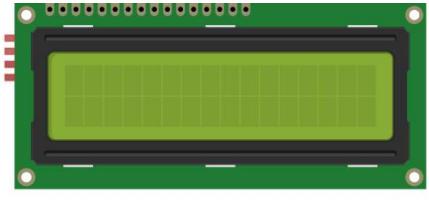


3.1.12 SPIEL – Nummer Vermutung

Einführung

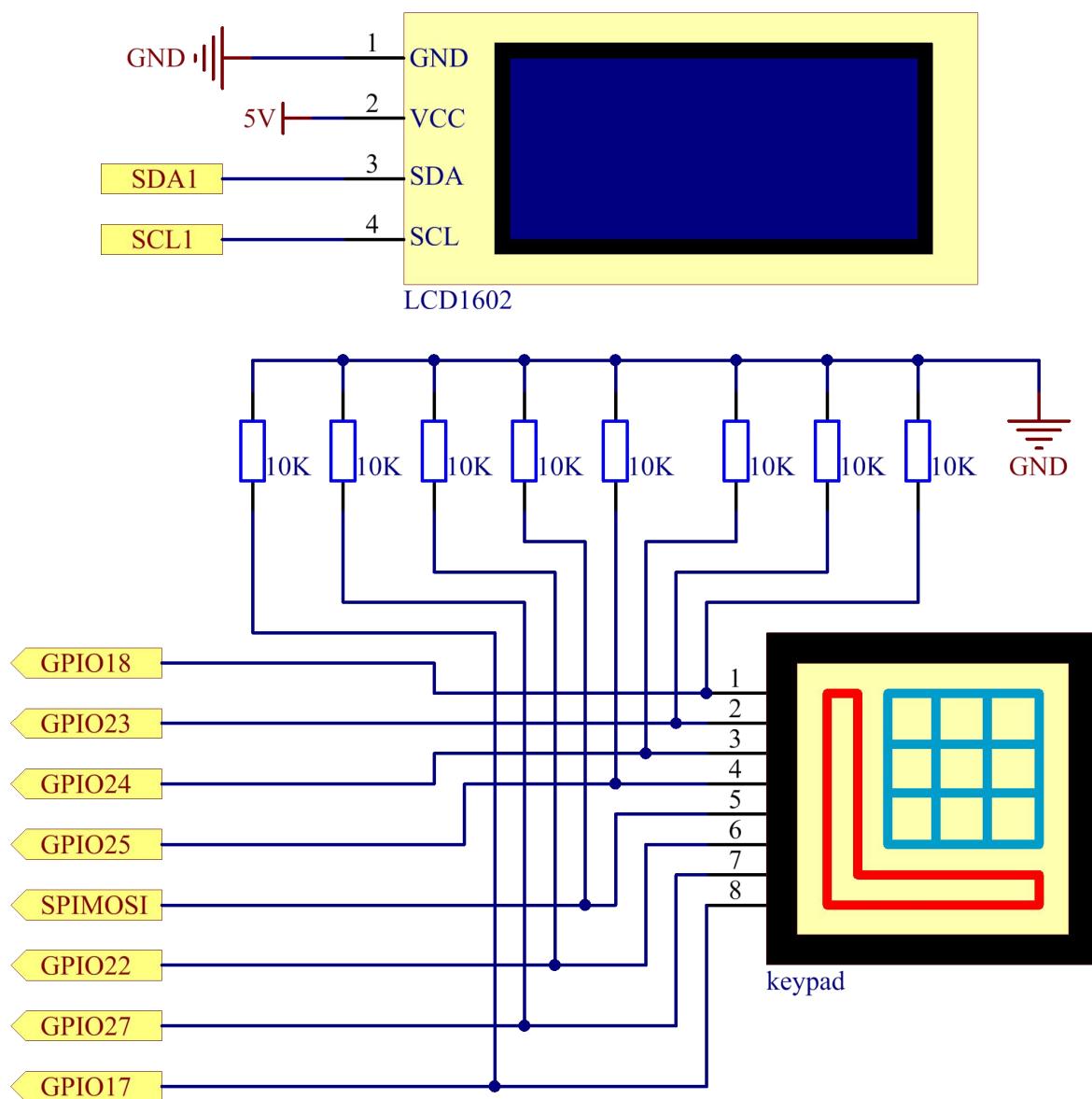
Nummer Vermutung ist ein lustiges Partyspiel, bei dem Sie und Ihre Freunde abwechselnd eine Nummer eingeben (0~99). Die Reichweite wird mit der Eingabe der Nummer kleiner, bis ein Spieler das Rätsel richtig beantwortet. Dann wird der Spieler besiegt und bestraft. Wenn zum Beispiel die Glücksnummer 51 ist, die die Spieler nicht sehen können, und der Spieler ① 50 eingibt, ändert sich die Eingabeaufforderung des Nummernbereichs auf 50~99; Wenn der Spieler ② 70 eingibt, kann der Nummernbereich zwischen 50 und 70 liegen. Wenn der Spieler ③ 51 eingibt, ist dieser Spieler der Unglückliche. Hier verwenden wir die Tastatur zur Eingabe von Nummer und das LCD zur Ausgabe der Ergebnisse.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * Tastatur
		
1 * 40-Pin Kabel		Mehrere Überbrückungsdrähte
		
1 * Steckbrett		8 * Widerstand 10KΩ
		
		1 * I2C LCD1602
		

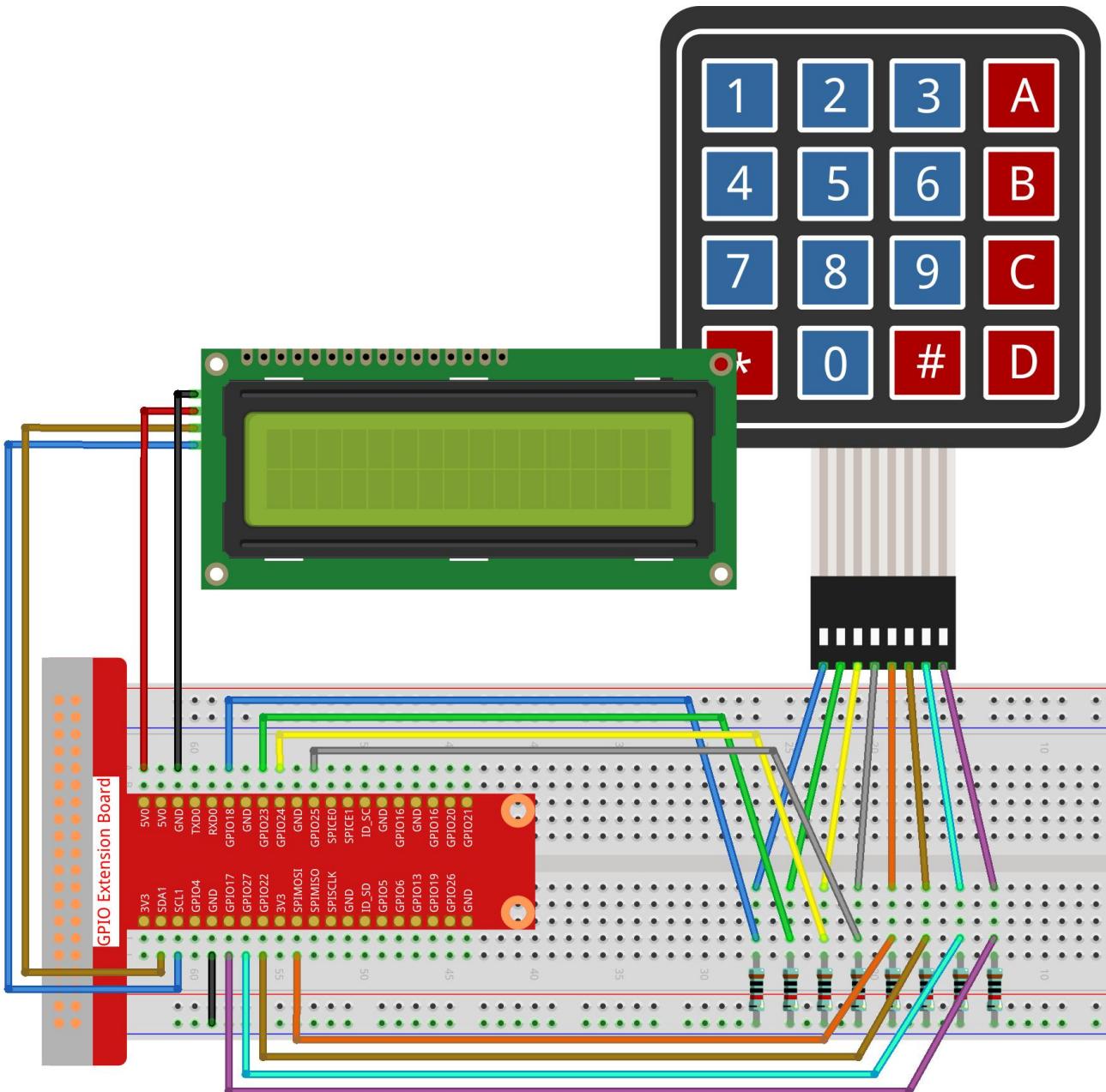
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO18	Pin 12	1	18
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24
GPIO25	Pin 22	6	25
SPIMOSI	Pin 19	12	10
GPIO22	Pin 15	3	22
GPIO27	Pin 13	2	27
GPIO17	Pin 11	0	17
SDA1	Pin 3	SDA1(8)	SDA1(2)
SCL1	Pin 5	SCL1(9)	SDA1(3)



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



Schritt 2: I2C einrichten (siehe Anhang. Wenn Sie I2C eingestellt haben, überspringen Sie diesen Schritt.)

➤ Für Benutzer in C-Sprache

Schritt 3: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.12/
```

Schritt 4: Kompilieren.

```
gcc 3.1.12_GAME_GuessNumber.c -lwiringPi
```

Schritt 5: Ausführen.

```
sudo ./a.out
```

Nachdem das Programm ausgeführt wurde, wird die erste Seite auf dem LCD angezeigt:

```
Herzlich willkommen!  
Drücken Sie A, los geht's!
```

Drücken Sie 'A' und das Spiel startet und die Spieleseite erscheint auf dem LCD.

```
Nummer eingeben:  
0 <Punkt> 99
```

Zu Beginn des Spiels wird eine Zufallsnummer "**Punkt**" erzeugt, die jedoch nicht auf dem LCD angezeigt wird. Sie müssen sie nur erraten. Die eingegebene Nummer wird am Ende der ersten Zeile angezeigt, bis die endgültige Berechnung abgeschlossen ist. (Drücken Sie 'D', um den Vergleich zu starten. Wenn die Eingangsnummer größer als **10** ist, wird der automatische Vergleich gestartet.)

Der Nummernkreis von 'Punkt' wird in der zweiten Zeile angezeigt. Und Sie müssen die Nummer innerhalb des Bereichs eingeben. Wenn Sie eine Nummer eingeben, wird der Bereich enger. Wenn Sie die Glücksnummer glücklicherweise oder unglücklicherweise erhalten haben, wird "Sie haben sie!" angezeigt.

Kode Erklärung

Am Anfang des Codes stehen die Funktionsfunktionen der **Tastatur** und des **I2C LCD1602**. Weitere Informationen hierzu finden Sie in den Tasten **1.1.7 I2C LCD1602** und **2.1.5**.

Hier müssen wir Folgendes wissen:

```
/************/  
//Start from here  
/************/  
void init(void){  
    fd = wiringPiI2CSetup(LCDAddr);  
    lcd_init();  
    lcd_clear();  
    for(int i=0 ; i<4 ; i++) {  
        pinMode(rowPins[i], OUTPUT);  
        pinMode(colPins[i], INPUT);  
    }  
    lcd_clear();
```

```
write(0, 0, "Welcome!");
write(0, 1, "Press A to go!");
}
```

Diese Funktion wird verwendet, um zunächst **I2C LCD1602** und **Tastatur** zu definieren und "Willkommen!" und "Drücken Sie A, um zu gehen!" anzuzeigen.

```
void init_new_value(void){
    srand(time(0));
    pointValue = rand()%100;
    upper = 99;
    lower = 0;
    count = 0;
    printf("point is %d\n",pointValue);
}
```

Die Funktion erzeugt die Zufallsnummer '**Punkt**' und setzt den Bereichshinweis des Punktes zurück.

```
bool detect_point(void){
    if(count > pointValue){
        if(count < upper){
            upper = count;
        }
    }
    else if(count < pointValue){
        if(count > lower){
            lower = count;
        }
    }
    else if(count == pointValue){
        count = 0;
        return 1;
    }
    count = 0;
    return 0;
}
```

`detect_point()` vergleicht die eingegebene Nummer mit dem erzeugten "Punkt". Wenn das Vergleichsergebnis ist, dass sie nicht gleich sind, weist **oberen** und **unteren** Werten zu und gibt '**0**' zurück. Andernfalls wenn das Ergebnis gleich sind, gibt es '**1**' zurück,

```
void lcd_show_input(bool result){
    char *str=NULL;
    str =(char*)malloc(sizeof(char)*3);
    lcd_clear();
    if (result == 1){
        write(0,1,"You've got it!");
        delay(5000);
        init_new_value();
        lcd_show_input(0);
        return;
    }
    write(0,0,"Enter number:");
    Int2Str(str,count);
    write(13,0,str);
    Int2Str(str,lower);
    write(0,1,str);
    write(3,1," <Point<");
    Int2Str(str,upper);
    write(12,1,str);
}
```

Diese Funktion dient zum Anzeigen der Spieleseite. Beachten Sie die Funktion **Int2Str(str,count)**, die diese Variablen **count**, **lower** und **superior** von Integer in Character String konvertiert, um die korrekte Anzeige von lcd zu gewährleisten.

```
int main(){
    unsigned char pressed_keys[BUTTON_NUM];
    unsigned char last_key_pressed[BUTTON_NUM];
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    init();
    init_new_value();
    while(1){
        keyRead(pressed_keys);
        bool comp = keyCompare(pressed_keys, last_key_pressed);
        if (!comp){
            if(pressed_keys[0] != 0){
                bool result = 0;
```

```

if(pressed_keys[0] == 'A'){
    init_new_value();
    lcd_show_input(0);
}
else if(pressed_keys[0] == 'D'){
    result = detect_point();
    lcd_show_input(result);
}
else if(pressed_keys[0] >='0' && pressed_keys[0] <= '9'){
    count = count * 10;
    count = count + (pressed_keys[0] - 48);
    if (count>=10){
        result = detect_point();
    }
    lcd_show_input(result);
}
keyCopy(last_key_pressed, pressed_keys);
}
delay(100);
}
return 0;
}

```

Main () enthält den gesamten Prozess des Programms, wie unten gezeigt:

- 1) Initialisieren Sie **I2C LCD1602** und **Tastatur**.
- 2) Verwenden Sie **init_new_value()**, um eine Zufallsnummer **0-99** zu erstellen.
- 3) Beurteilen Sie, ob die Taste gedrückt wurde, und lassen Sie die Taste ablesen.
- 4) Wenn die Taste '**A**' gedrückt wird, erscheint eine Zufallsnummer **0-99** und das Spiel beginnt.
- 5) Wenn festgestellt wird, dass die Taste '**D**' gedrückt wurde, geht das Programm in die Ergebnisbeurteilung ein und zeigt das Ergebnis auf dem LCD an. Dieser Schritt hilft Ihnen, das Ergebnis auch zu beurteilen, wenn Sie nur eine Nummer und dann die Taste '**D**' drücken.
- 6) Wenn die Taste **0-9** gedrückt wird, wird der Zählerwert geändert. Wenn die Anzahl größer als **10** ist, beginnt das Urteil.
- 7) Die Änderungen des Spiels und seiner Werte werden auf dem **LCD1602** angezeigt.

➤ Für Python-Sprachbenutzer

Schritt 3: Verzeichnis wechseln.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 4: Ausführen.

```
sudo python3 3.1.12_GAME_GuessNumber.py
```

Nachdem das Programm ausgeführt wurde, wird die erste Seite auf dem LCD angezeigt:

```
Herzlich willkommen!  
Drücken Sie A, los geht's!
```

Drücken Sie 'A' und das Spiel startet und die Spieleseite erscheint auf dem LCD.

```
Nummer eingeben:  
0 <Punkt> 99
```

Zu Beginn des Spiels wird eine Zufallsnummer "**Punkt**" erzeugt, die jedoch nicht auf dem LCD angezeigt wird. Sie müssen sie nur erraten. Die eingegebene Nummer wird am Ende der ersten Zeile angezeigt, bis die endgültige Berechnung abgeschlossen ist. (Drücken Sie 'D', um den Vergleich zu starten. Wenn die Eingangsnummer größer als **10** ist, wird der automatische Vergleich gestartet.)

Der Nummernkreis von 'Punkt' wird in der zweiten Zeile angezeigt. Und Sie müssen die Nummer innerhalb des Bereichs eingeben. Wenn Sie eine Nummer eingeben, wird der Bereich enger. Wenn Sie die Glücksnummer glücklicherweise oder unglücklicherweise erhalten haben, wird "Sie haben sie!" angezeigt.

Kode Erklärung

Am Anfang des Codes stehen die Funktionsfunktionen der **Tastatur** und des **I2C LCD1602**. Weitere Informationen hierzu finden Sie in den Tasten **1.1.7 I2C LCD1602** und **2.1.5**.

Hier müssen wir Folgendes wissen:

```
def init_new_value():  
    global pointValue,upper,count,lower  
    pointValue = random.randint(0,99)  
    upper = 99  
    lower = 0  
    count = 0
```

```
print('point is %d' %(pointValue))
```

Die Funktion erzeugt die Zufallsnummer '**Punkt**' und setzt den Bereichshinweis des Punktes zurück.

```
def detect_point():
    global count,upper,lower
    if count > pointValue:
        if count < upper:
            upper = count
    elif count < pointValue:
        if count > lower:
            lower = count
    elif count == pointValue:
        count = 0
    return 1
count = 0
return 0
```

`detect_point()` vergleicht die eingegebene Nummer (Anzahl) mit dem erzeugten "**Punkt**". Wenn das Vergleichsergebnis ist, dass sie nicht gleich sind, weist **oberen** und **unteren** Werten zu und gibt '0' zurück. Andernfalls wenn das Ergebnis gleich sind, gibt es '1' zurück,

```
def lcd_show_input(result):
    LCD1602.clear()
    if result == 1:
        LCD1602.write(0,1,'You have got it!')
        time.sleep(5)
        init_new_value()
        lcd_show_input(0)
    return
    LCD1602.write(0,0,'Enter number:')
    LCD1602.write(13,0,str(count))
    LCD1602.write(0,1,str(lower))
    LCD1602.write(3,1,' < Point < ')
    LCD1602.write(13,1,str.upper))
```

Diese Funktion dient zum Anzeigen der Spieleseite.

`str(count)`: Da `write()` nur den Datentyp - Zeichenfolge - unterstützen kann, wird `str()` benötigt, um die Nummer in eine Zeichenfolge umzuwandeln.

```
def loop():
```

```

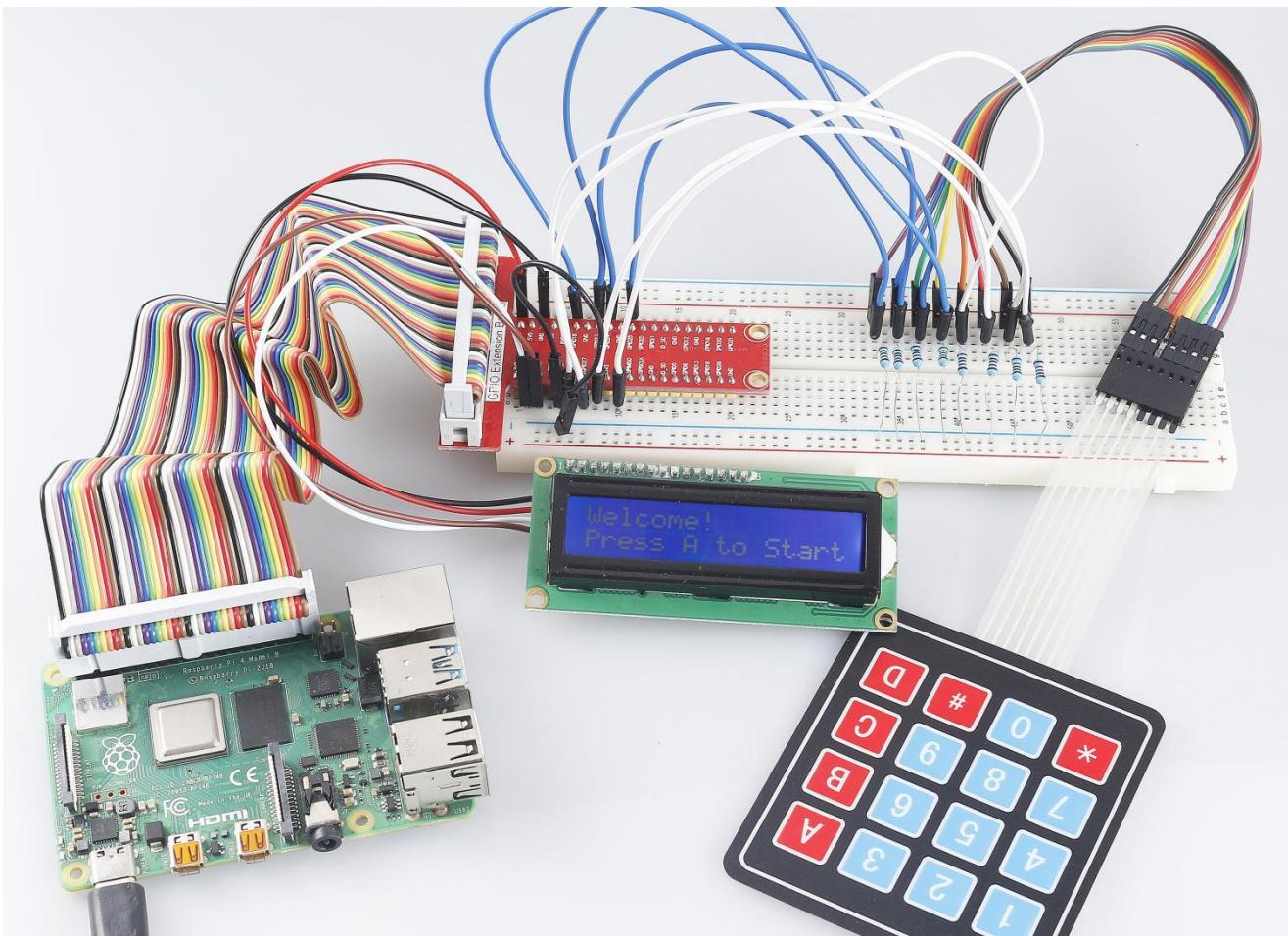
global keypad, last_key_pressed, count
while(True):
    result = 0
    pressed_keys = keypad.read()
    if len(pressed_keys) != 0 and last_key_pressed != pressed_keys:
        if pressed_keys == ["A"]:
            init_new_value()
            lcd_show_input(0)
        elif pressed_keys == ["D"]:
            result = detect_point()
            lcd_show_input(result)
        elif pressed_keys[0] in keys:
            if pressed_keys[0] in list(["A", "B", "C", "D", "#", "*"]):
                continue
            count = count * 10
            count += int(pressed_keys[0])
            if count >= 10:
                result = detect_point()
                lcd_show_input(result)
            print(pressed_keys)
        last_key_pressed = pressed_keys
    time.sleep(0.1)

```

Main () enthält den gesamten Prozess des Programms, wie unten gezeigt:

- 1) Initialisieren Sie **I2C LCD1602** und **Tastatur**.
- 2) Beurteilen Sie, ob die Taste gedrückt wurde, und lassen Sie die Taste ablesen.
- 3) Wenn die Taste '**A**' gedrückt wird, erscheint eine Zufallsnummer **0-99** und das Spiel beginnt.
- 4) Wenn festgestellt wird, dass die Taste '**D**' gedrückt wurde, geht das Programm in die Ergebnisbeurteilung ein.
- 5) Wenn die Taste **0-9** gedrückt wird, wird der Zählerwert geändert. Wenn die Anzahl größer als **10** ist, beginnt das Urteil.
- 6) Die Änderungen des Spiels und seiner Werte werden auf dem **LCD1602** angezeigt.

Phänomen Bild



3.1.13 SPIEL - 10 Sekunden

Einführung

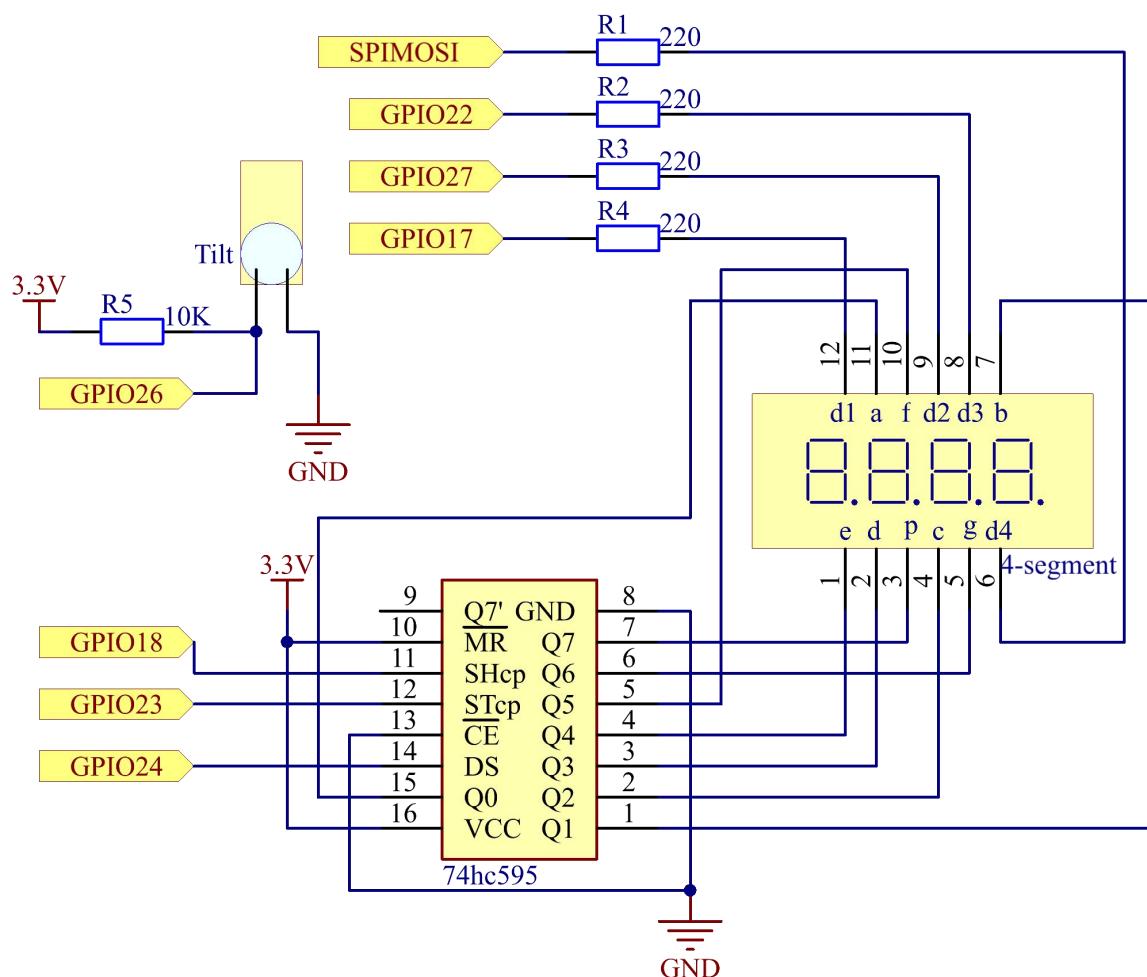
Als nächstes folge mir, um ein Spielgerät zu bauen, das deine Konzentration herausfordert. Binden Sie den Neigungsschalter an einen Stock, um einen Zauberstab herzustellen. Schütteln Sie den Stab, die 4-stellige Segmentanzeige beginnt zu zählen. Durch erneutes Schütteln wird die Zählung beendet. Wenn es Ihnen gelingt, die angezeigte Anzahl bei **10,00** zu halten, gewinnen Sie. Sie können das Spiel mit Ihren Freunden spielen, um zu sehen, wer der Zeitassistent ist.

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * 4-stellige 7-Segment-Anzeige
	 GPIO Extension Board Pinout: 3V3 SVO SDA1 SVO SCL1 GND GPI04 TXD0 GND RXD0 GPI017 GPIO18 GPI027 GND GPI022 GPIO23 3V3 GPIO24 SPIMOSI GND SPIMISO GPIO25 SPISCLK SPICO0 GND SPICE1 ID_S0 ID_S0 GPI05 GND GPI06 GPIO16 GPI013 GND GPI019 GPIO16 GPI026 GPIO20 GND GPIO21	
1 * 40-Pin Kabel	1 * 74HC595	1 * Neigungsschalter
1 * Steckbrett	Mehrere Überbrückungsdrähte	4 * Widerstand (220 Ω)
		1 * Widerstand 10 kΩ

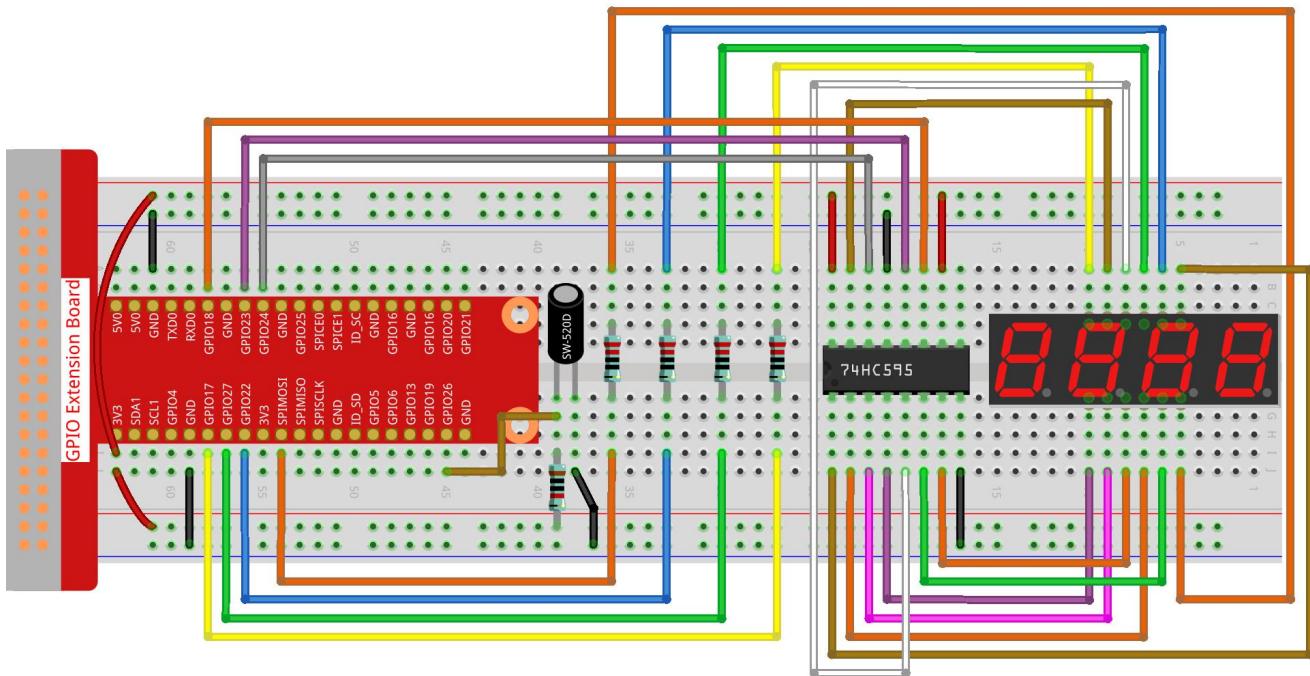
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22
SPIMOSI	Pin 19	12	10
GPIO18	Pin 12	1	18
GPIO23	Pin 16	4	23
GPIO24	Pin 18	5	24
GPIO26	Pin 37	25	26



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 2: Gehen Sie zum Ordner der Kode.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.13/
```

Schritt 3: Kompilieren Sie die Kode.

```
gcc 3.1.13_GAME_10Second.c -lwiringPi
```

Schritt 4: Führen Sie die ausführbare Datei aus.

```
sudo ./a.out
```

Schütteln Sie den Stab, die 4-stellige Segmentanzeige beginnt zu zählen. Durch erneutes Schütteln wird die Zählung beendet. Wenn es Ihnen gelingt, die angezeigte Anzahl bei **10,00** zu halten, gewinnen Sie. Schütteln Sie es noch einmal, um die nächste Runde des Spiels zu starten.

Kode Erklärung

```
void stateChange(){
    if (gameState == 0){
        counter = 0;
        delay(1000);
        ularm(10000,10000);
    }else{
        alarm(0);
```

```

    delay(1000);
}
gameState = (gameState + 1)%2;
}

```

Das Spiel ist in zwei Modi unterteilt:

gameState=0 ist der "Start" -Modus, in dem die Zeit zeitlich festgelegt und auf der Segmentanzeige angezeigt wird und der Kippschalter geschüttelt wird, um in den "Show" -Modus zu wechseln.

GameState=1 ist der "Show" -Modus, der das Timing stoppt und die Zeit auf der Segmentanzeige anzeigt. Durch erneutes Schütteln des Neigungsschalters wird der Timer zurückgesetzt und das Spiel neu gestartet.

```

void loop(){
    int currentState =0;
    int lastState=0;
    while(1){
        display();
        currentState=digitalRead(sensorPin);
        if((currentState==0)&&(lastState==1)){
            stateChange();
        }
        lastState=currentState;
    }
}

```

Loop() ist die Hauptfunktion. Zunächst wird die Zeit auf der 4-Bit-Segmentanzeige angezeigt und der Wert des Neigungsschalters gelesen. Wenn sich der Status des Neigungsschalters geändert hat, wird stateChange () aufgerufen.

➤ Für Python-Sprachbenutzer

Schritt 2: Gehen Sie zum Ordner der Kode

```
cd /home/pi/davinci-kit-for-raspberry-pi/python/
```

Schritt 3: Führen Sie die ausführbare Datei aus.

```
sudo python3 3.1.13_GAME_10Second.py
```

Schütteln Sie den Stab, die 4-stellige Segmentanzeige beginnt zu zählen. Durch erneutes Schütteln wird die Zählung beendet. Wenn es Ihnen gelingt, die angezeigte Anzahl bei **10,00** zu halten, gewinnen Sie. Schütteln Sie es noch einmal, um die nächste Runde des Spiels zu starten.

Kode Erklärung

```
def stateChange():
    global gameState
    global counter
    global timer1
    if gameState == 0:
        counter = 0
        time.sleep(1)
        timer()
    elif gameState ==1:
        timer1.cancel()
        time.sleep(1)
    gameState = (gameState+1)%2
```

Das Spiel ist in zwei Modi unterteilt:

gameState=0 ist der "Start" -Modus, in dem die Zeit zeitlich festgelegt und auf der Segmentanzeige angezeigt wird und der Kippschalter geschüttelt wird, um in den "Show" -Modus zu wechseln.

GameState=1 ist der "Show" -Modus, der das Timing stoppt und die Zeit auf der Segmentanzeige anzeigt. Durch erneutes Schütteln des Neigungsschalters wird der Timer zurückgesetzt und das Spiel neu gestartet.

```
def loop():
    global counter
    currentState = 0
    lastState = 0
    while True:
        display()
        currentState=GPIO.input(sensorPin)
        if (currentState == 0) and (lastState == 1):
            stateChange()
        lastState=currentState
```

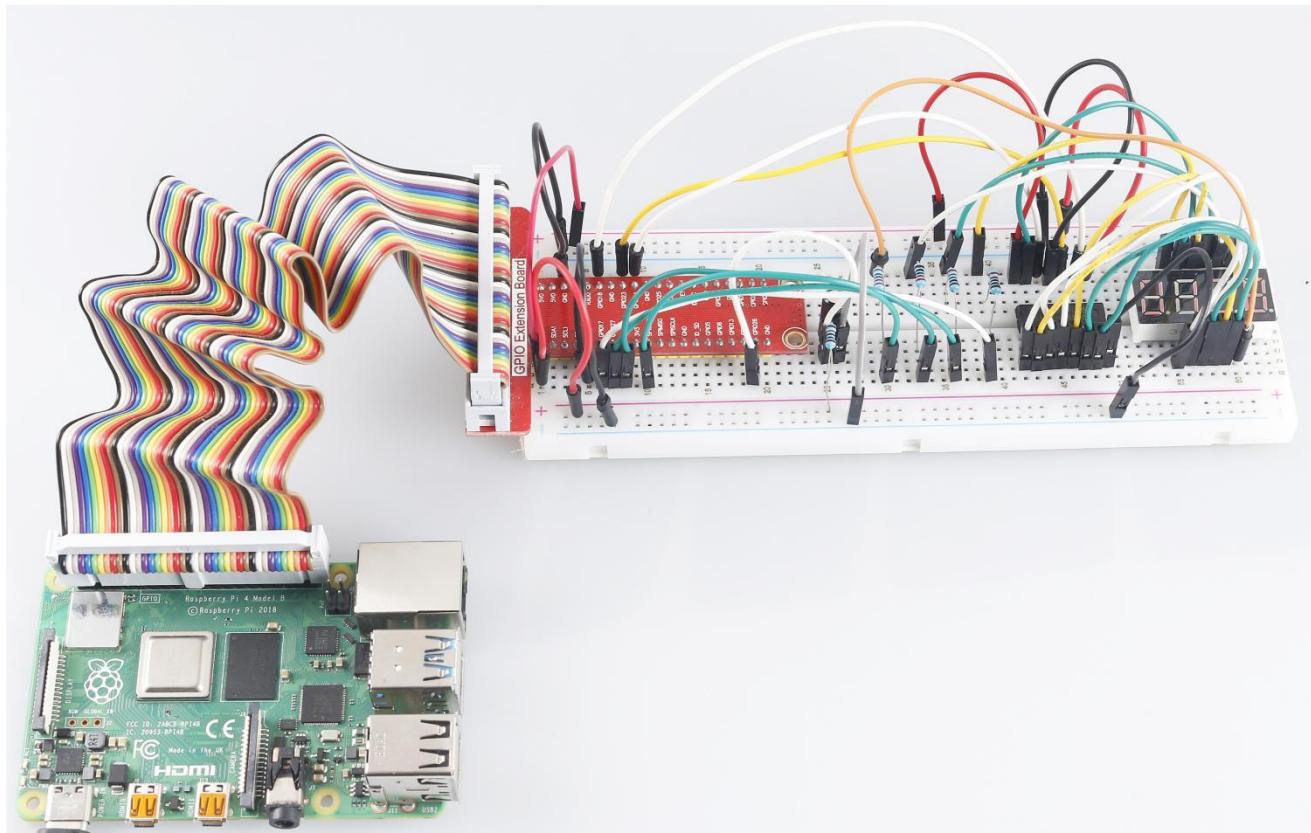
Loop() ist die Hauptfunktion. Zunächst wird die Zeit auf der 4-Bit-Segmentanzeige angezeigt und der Wert des Neigungsschalters gelesen. Wenn sich der Status des Neigungsschalters geändert hat, wird stateChange () aufgerufen.

```
def timer():
    global counter
    global timer1
    timer1 = threading.Timer(0.01, timer)
```

```
timer1.start()  
counter += 1
```

Nachdem das Intervall 0,01 s erreicht hat, wird die Timerfunktion aufgerufen. Addiere 1 zum Zähler und der Timer wird erneut verwendet, um sich alle 0,01 Sekunden wiederholt auszuführen.

Phänomen Bild



3.1.14 SPIEL - Nicht nicht

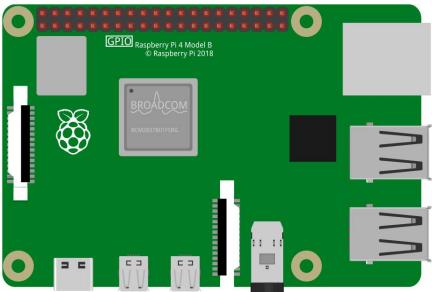
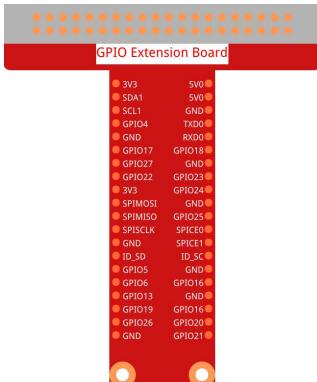
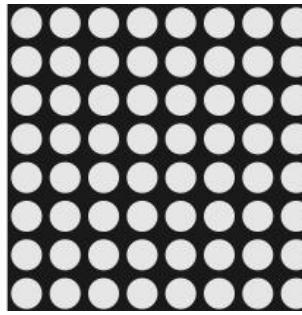
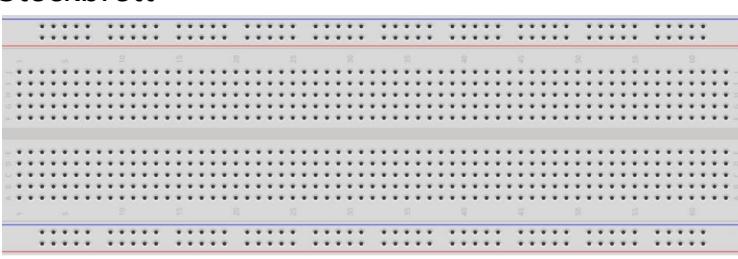
Einführung

In dieser Lektion werden wir ein interessantes Spielgerät herstellen und es "Nicht nicht" nennen.

Während des Spiels aktualisiert die Punktmatrix einen Pfeil nach dem Zufallsprinzip. Sie müssen die Taste innerhalb einer begrenzten Zeit in die entgegengesetzte Richtung des Pfeils drücken. Wenn die Zeit abgelaufen ist oder wenn die Taste in die gleiche Richtung wie der Pfeil gedrückt wird, sind Sie raus.

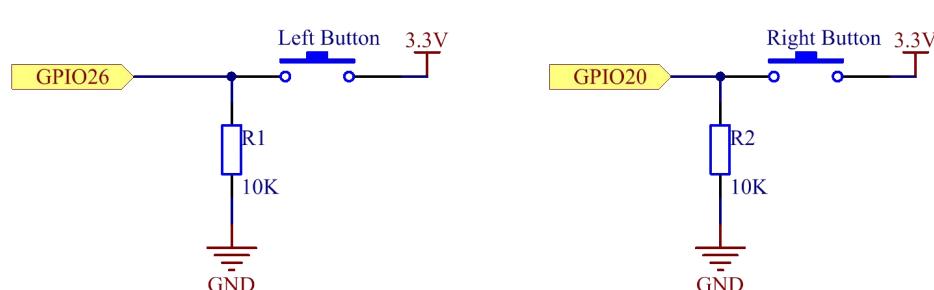
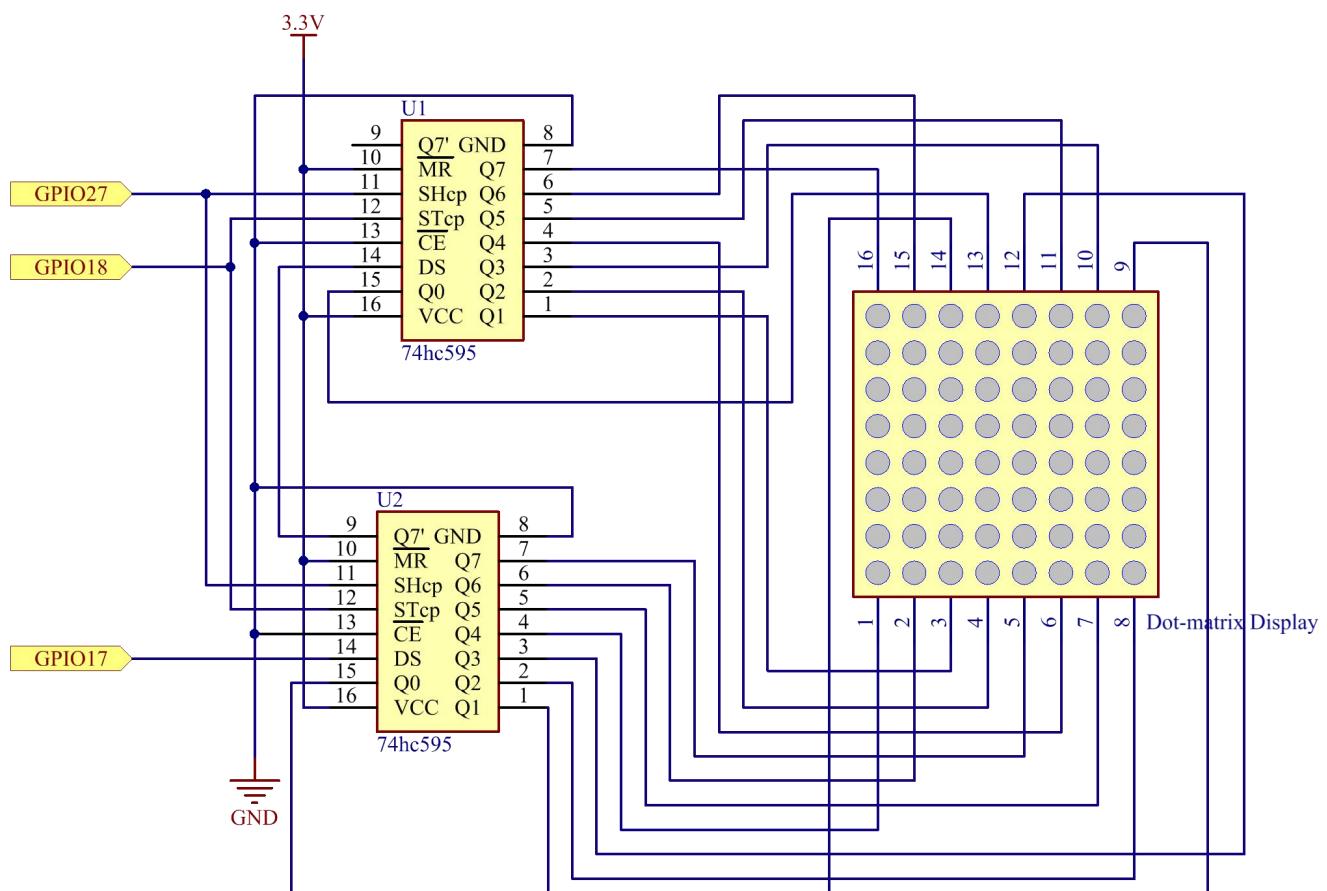
Dieses Spiel kann wirklich Ihr umgekehrtes Denken üben, und jetzt sollen wir es versuchen?

Komponenten

1 * Raspberry Pi	1 * T-Erweiterungskarte	1 * LED-Punktmatrix
	 GPIO Extension Board Pinout: 3V3, SDA1, 5V0, GND, GPIO4, TXD0, GND, RXD0, GPIO17, GPIO18, GPIO27, GND, GPIO22, GPIO23, 3V3, GND, SPI_MOSI, GND, SPI_MISO, GPIO25, SPI_SCLK, SPI_CE0, GND, ID_SD, ID_SC, GPIO9, GND, GPIO6, GPIO16, GPIO13, GND, GPIO19, GPIO16, GPIO26, GPIO20, GND, GPIO21	
1 * 40-Pin Kabel	Mehrere Überbrückungsdrähte	2 * Taste
		
1 * Steckbrett	2 * 74HC595	2 * Widerstand 10KΩ
		

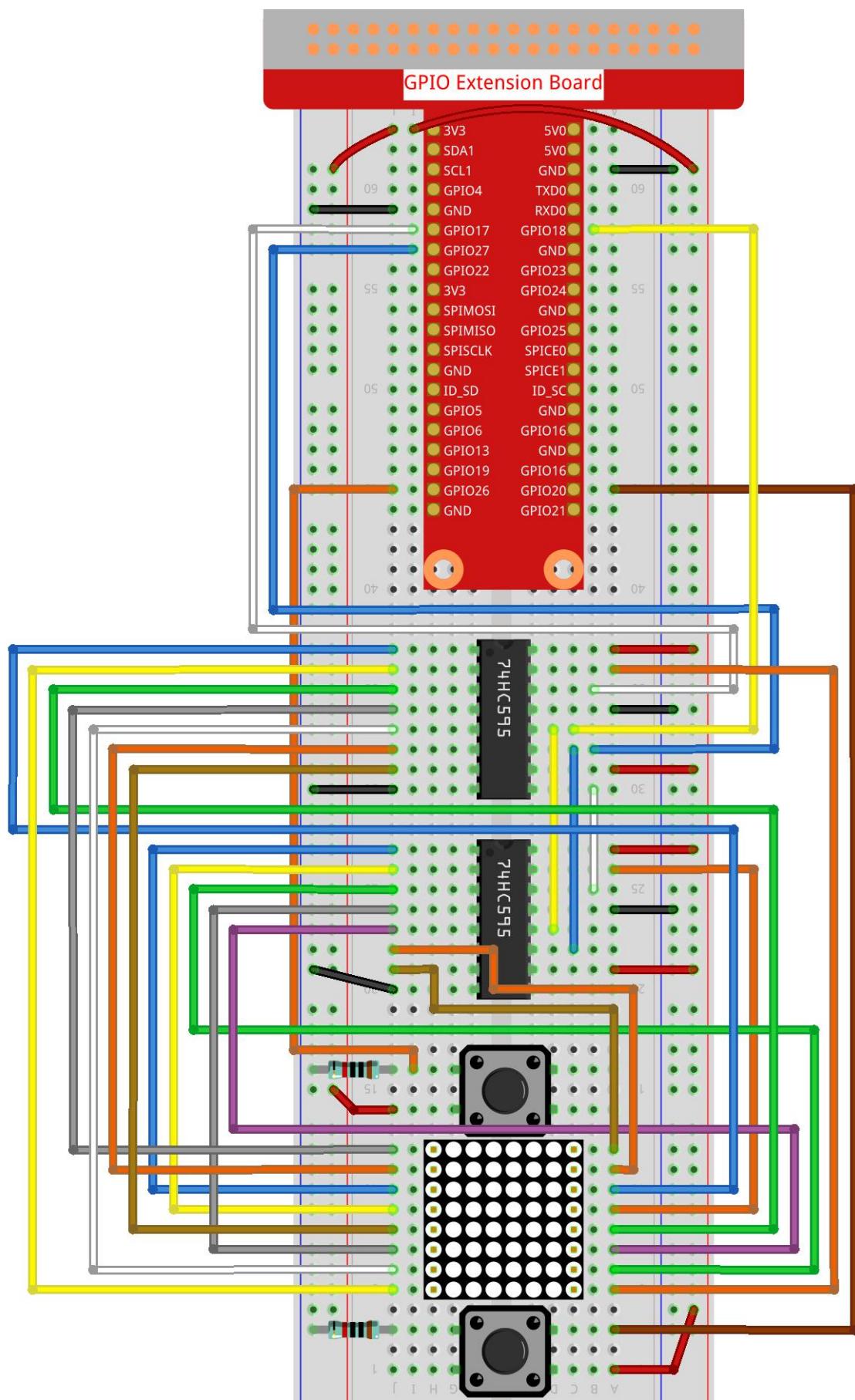
Schematische Darstellung

T-Karte Name	physisch	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18
GPIO27	Pin 13	2	27
GPIO20	Pin 38	28	20
GPIO26	Pin 37	25	26



Experimentelle Verfahren

Schritt 1: Bauen Sie die Schaltung auf.



➤ Für Benutzer in C-Sprache

Schritt 5: Wechseln Sie in den Kodeordner.

```
cd /home/pi/davinci-kit-for-raspberry-pi/c/3.1.14/
```

Schritt 6: Kompilieren.

```
gcc 3.1.14_GAME_NotNot.c -lwiringPi
```

Schritt 7: Ausführen.

```
sudo ./a.out
```

Nach dem Start des Programms wird ein Pfeil nach links oder rechts in der Punktmatrix nach dem Zufallsprinzip aktualisiert. Sie müssen die Taste innerhalb einer begrenzten Zeit in die entgegengesetzte Richtung des Pfeils drücken. Dann erscheint "√" auf der Punktmatrix. Wenn die Zeit abgelaufen ist oder wenn die Taste in die gleiche Richtung wie der Pfeil gedrückt wird, sind Sie ausgeschaltet und die Punktmatrix zeigt „x“ an. Sie können auch 2 neue Schaltflächen hinzufügen oder durch Joystick-Tasten für Auf, Ab, Links und Rechts ersetzen - 4 Richtungen, um die Schwierigkeit des Spiels zu erhöhen.

Kode Erklärung

Basierend auf der **1.1.6 LED-Punktmatrix** werden in dieser Lektion **2** Tasten hinzugefügt, um ein amüsantes Spielgerät zu erstellen. Wenn Sie mit der Punktmatrix nicht sehr vertraut sind, lesen Sie bitte **1.1.6 LED-Punktmatrix**.

Der gesamte Programmprozess ist wie folgt:

1. Wählen Sie zufällig eine Pfeilrichtung und generieren Sie **Timer 1**.
2. Zeigen Sie das Pfeilbild auf der Punktmatrix an.
3. Beurteilen Sie die Tasteneingabe. Wenn die Taste gedrückt wird oder **Timer 1** daran erinnert, dass die Zeit abgelaufen ist, beginnt die Beurteilung.
4. Zeigen Sie das Bild anhand eines Bewertungsergebnisses an. In der Zwischenzeit **Timer 2** generieren.
5. Führen Sie **Schritt 1** erneut aus, wenn **Timer 2** daran erinnert, dass die Zeit abgelaufen ist.

```
struct GLYPH{  
    char *word;  
    unsigned char code[8];  
};
```

```

struct GLYPH arrow[2]=
{
    {"right",{0xFF,0xEF,0xDF,0x81,0xDF,0xEF,0xFF,0xFF}},
    // {"down",{0xFF,0xEF,0xC7,0xAB,0xEF,0xEF,0xEF,0xFF}},
    // {"up",{0xFF,0xEF,0xEF,0xEF,0xAB,0xC7,0xEF,0xFF}},
    {"left",{0xFF,0xF7,0xFB,0x81,0xFB,0xF7,0xFF,0xFF}}
};

struct GLYPH check[2]=
{
    {"wrong",{0xFF,0xBB,0xD7,0xEF,0xD7,0xBB,0xFF,0xFF}},
    {"right",{0xFF,0xFF,0xF7,0xEB,0xDF,0xBF,0xFF,0xFF}}
};

```

Die GLYPH-Struktur funktioniert wie ein Wörterbuch: Das **Wort** Attribut entspricht dem **Schlüssel** im Wörterbuch. Das **Kode**-Attribut entspricht dem Wert.

Hier wird Kode verwendet, um ein Array für die Punktmatrix zum Anzeigen von Bildern zu speichern (ein 8x8-Bit-Array).

Hier kann der Array-**Pfeil** verwendet werden, um das Pfeilmuster in Aufwärts-, Abwärts-, Links- und Rechtsrichtung auf der LED-Punktmatrix anzuzeigen.

Jetzt werden **unten** und **oben** kommentiert und bei Bedarf auskommentiert.

Die Array-**Prüfung** wird verwendet, um diese beiden Bilder anzuzeigen: "x" und "√".

```

char *lookup(char *key,struct GLYPH *glyph,int length){
    for (int i=0;i<length;i++)
    {
        if(strcmp(glyph[i].word,key)==0){
            return glyph[i].code;
        }
    }
}

```

Die Funktion **lookup()** funktioniert durch „Überprüfen des Wörterbuchs“. Definieren Sie einen Schlüssel, suchen Sie die gleichen Wörter wie der Schlüssel in der Struktur **GLYPH *glyph** und geben Sie die entsprechenden Informationen zurück - "code" des bestimmten Wortes.

Die Funktion **strcmp()** wird verwendet, um die Identität von zwei Zeichenfolgen **glyph[i].word** und Schlüssel zu vergleichen; Wenn die Identität beurteilt wird, geben Sie den **glyph[i].code** zurück (wie gezeigt).

```
void display(char *glyphCode){
    for(int i;i<8;i++){
        hc595_in(glyphCode[i]);
        hc595_in(0x80>>i);
        hc595_out();
    }
}
```

Zeigen Sie das angegebene Muster in der Punktmatrix an.

```
void createGlyph(){
    srand(time(NULL));
    int i=rand()%(sizeof(arrows)/sizeof(arrows[0]));
    waypoint=arrows[i].word;
    stage="PLAY";
    alarm(2);
}
```

Mit der Funktion **createGlyph()** wird zufällig eine Richtung ausgewählt (das Wortattribut eines Elements im array **arrows[]**: "left", "right"...). Stellen Sie die Bühne auf „PLAY“ und starten Sie eine 2-Sekunden-Weckerfunktion.

srand(time(NULL)): Initialisiert zufällige Seeds, die von der Systemuhr stammen.

(sizeof(arrows)/sizeof(arrows[0])): Ermittelt die Länge des Arrays, das Ergebnis ist 2.

rand()%2: Der Rest ist 0 oder 1, erhalten durch Teilen einer generierten Zufallsnummer durch 2.

waypoint=arrows[i].word: Das Ergebnis sollte "rechts" oder "links" sein.

```
void checkPoint(char *inputKey){
    alarm(0)==0;
    if(inputKey==waypoint||inputKey=="empty")
    {
        waypoint="wrong";
    }
    else{
        waypoint="right";
    }
    stage="CHECK";
    alarm(1);
}
```

Mit `checkPoint()` wird die Tasteneingabe überprüft. Wenn die Taste nicht gedrückt wird oder die Taste in die gleiche Richtung wie der Pfeil gedrückt wird, ist das Ergebnis des Wegpunkts falsch und auf der Punktmatrix wird „x“ angezeigt. Andernfalls ist der Wegpunkt richtig und die Punktmatrix zeigt „√“ an. Hier ist die **Stufe CHECK** und es kann eine 1-Sekunden-Weckerfunktion eingestellt werden.

`alarm()` wird auch als „Wecker“ bezeichnet, bei dem ein Timer eingestellt werden kann, und sendet **SIGALRM**-Signale an den Fortschritt, wenn die definierte Zeit abgelaufen ist.

```
void getKey(){
    if (digitalRead(AButtonPin)==1&&digitalRead(BButtonPin)==0)
        {checkPoint("right");}
    else if (digitalRead(AButtonPin)==0&&digitalRead(BButtonPin)==1)
        {checkPoint("left");}
}
```

`getKey()` liest die Zustände dieser beiden Schaltflächen; Wenn die rechte Taste gedrückt wird, ist der Parameter der Funktion `checkPoint()` **rechts** und wenn die **linke** Taste gedrückt wird, bleibt der Parameter links.

```
void timer(){
    if (stage=="PLAY"){
        checkPoint("empty");
    }
    else if(stage=="CHECK"){
        createGlyph();
    }
}
```

Zuvor wurde `timer()` aufgerufen, wenn die Alarmzeit abgelaufen ist. Im Modus „PLAY“ soll dann `checkPoint()` aufgerufen werden, um das Ergebnis zu beurteilen. Wenn das Programm auf den Modus „CHECK“ eingestellt ist, sollte die Funktion `createGlyph()` aufgerufen werden, um neue Muster auszuwählen.

```
void main(){
    setup();
    signal(SIGALRM,timer);
    createGlyph();
    char *code = NULL;
    while(1){
        if (stage == "PLAY")
        {
```

```

        code=lookup(waypoint,arrow,sizeof(arrow)/sizeof(arrow[0]));
        display(code);
        getKey();
    }
    else if(stage == "CHECK")
    {
        code = lookup(waypoint,check,sizeof(check)/sizeof(check[0]));
        display(code);
    }
}
}

```

Die Funktionsweise des Funktionssignals (SIGALRM, Timer): Aufruf der Funktion timer(), wenn ein SIGALRM-Signal (vom Weckerfunktionsalarm() erzeugt) empfangen wird.

Wenn das Programm startet, rufen Sie zunächst einmal createGlyph() auf und starten Sie dann die Schleife.

In der Schleife: Im PLAY-Modus zeigt die Punktmatrix Pfeilmuster an und überprüft den Schaltflächenstatus. Im CHECK-Modus wird "x" oder "√" angezeigt.

➤ Für Python-Sprachbenutzer

Schritt 5: Gehen Sie in den Kode-Ordner.

```
cd /home/pi/davinci-kit-for-raspberry-pi/python
```

Schritt 6: Ausführen.

```
sudo python3 3.1.14_GAME_NonNot.py
```

Nach dem Starten des Programms erscheint auf der Punktmatrix ein Pfeil nach rechts oder links. Sie müssen die Taste innerhalb einer begrenzten Zeit in die entgegengesetzte Richtung des Pfeils drücken. Dann erscheint "√" auf der Punktmatrix. Wenn die Zeit abgelaufen ist oder wenn die Taste in die gleiche Richtung wie der Pfeil gedrückt wird, sind Sie ausgeschaltet und die Punktmatrix zeigt „x“ an. Sie können auch 2 neue Schaltflächen hinzufügen oder durch Joystick-Tasten für Auf, Ab, Links und Rechts ersetzen - 4 Richtungen, um die Schwierigkeit des Spiels zu erhöhen.

Kode Erklärung

Basierend auf der **1.1.6 LED-Punktmatrix** werden in dieser Lektion **2** Tasten hinzugefügt, um ein amüsantes Spielgerät zu erstellen. Wenn Sie mit der Punktmatrix nicht sehr vertraut sind, lesen Sie bitte **1.1.6 LED-Punktmatrix**.

Der gesamte Programmprozess ist wie folgt:

1. Wählen Sie zufällig eine Pfeilrichtung und generieren Sie **Timer 1**.
2. Zeigen Sie das entsprechende Pfeilbild in der Punktmatrix an.
3. Beurteilen Sie die Tasteneingabe. Wenn die Taste gedrückt wird oder **Timer 1** daran erinnert, dass die Zeit abgelaufen ist, beginnt die Beurteilung.
4. Zeigen Sie das Bild anhand eines Bewertungsergebnisses an. In der Zwischenzeit **Timer 2** generieren.
5. Führen Sie **Schritt 1** erneut aus, wenn **Timer 2** daran erinnert, dass die Zeit abgelaufen ist.

```
def main():
    creatGlyph()
    while True:
        if stage == "PLAY":
            display(arrow[waypoint])
            getKey()
        elif stage == "CHECK":
            display(check[waypoint])
```

Main() enthält den gesamten laufenden Prozess.

Wenn das Programm startet, rufen Sie zunächst einmal createGlyph() auf und starten Sie dann die Schleife.

In der Schleife: Im PLAY-Modus zeigt die Punktmatrix Pfeilmuster an und überprüft den Schaltflächenstatus. Im CHECK-Modus wird "x" oder "√" angezeigt.

```
arrow={
    "#down": [0xFF, 0xEF, 0xC7, 0xAB, 0xEF, 0xEF, 0xEF, 0xFF],
    "#up": [0xFF, 0xEF, 0xEF, 0xEF, 0xAB, 0xC7, 0xEF, 0xFF],
    "right": [0xFF, 0xEF, 0xDF, 0x81, 0xDF, 0xEF, 0xFF, 0xFF],
    "left": [0xFF, 0xF7, 0xFB, 0x81, 0xFB, 0xF7, 0xFF, 0xFF]
}
check={
    "wrong": [0xFF, 0xBB, 0xD7, 0xEF, 0xD7, 0xBB, 0xFF, 0xFF],
    "right": [0xFF, 0xFF, 0xF7, 0xEB, 0xDF, 0xBF, 0xFF, 0xFF]
}
```

Hier kann der **Wörterbuch** Pfeil verwendet werden, um das Pfeilmuster in Aufwärts-, Abwärts-, Links- und Rechtsrichtung auf der LED-Punktmatrix anzuzeigen.

Jetzt werden unten und oben kommentiert und bei Bedarf auskommentiert.

Die **Wörterbuch** Prüfung wird verwendet, um diese beiden Bilder anzuzeigen: "x" und "√".

```
def display(glyphCode):
    for i in range(0, 8):
        hc595_shift(glyphCode[i])
        hc595_shift(0x80 >> i)
        GPIO.output(RCLK, GPIO.HIGH)
        GPIO.output(RCLK, GPIO.LOW)
```

Zeigen Sie das angegebene Muster in der Punktmatrix an.

```
def creatGlyph():
    global waypoint
    global stage
    global timerPlay
    waypoint=random.choice(list(arrow.keys()))
    stage = "PLAY"
    timerPlay = threading.Timer(2.0, timeOut)
    timerPlay.start()
```

Mit der Funktion **createGlyph()** wird zufällig eine Richtung ausgewählt (das Wortattribut eines Elements im array **arrow[]**: "left", "right"...). Stellen Sie die Bühne auf „PLAY“ und starten Sie eine 2-Sekunden-Weckerfunktion.

arrow.keys(): Wählen Sie die Tasten "rechts" und "links" im Pfeilarray.

list(arrow.keys()): Kombinieren Sie diese Schlüssel zu einem Array.

random.choice(list(arrow.keys())): Wählen Sie zufällig ein Element im Array aus.

Das Ergebnis von **waypoint=random.choice(list(arrow.keys()))** sollte also "rechts" oder "links" sein.

```
def checkPoint(inputKey):
    global waypoint
    global stage
    global timerCheck
    if inputKey == "empty" or inputKey == waypoint:
        waypoint = "wrong"
    else:
        waypoint = "right"
    timerPlay.cancel()
```

```
stage = "CHECK"  
timerCheck = threading.Timer(1.0, creatGlyph)  
timerCheck.start()
```

Mit `checkPoint()` wird der aktuelle Status der Tasteneingabe ermittelt:

Wenn keine Taste gedrückt wird oder die Taste in die gleiche Richtung wie der Pfeil gedrückt wird, ist der zugewiesene Wert des **Wegpunkts falsch** und auf der Punktmatrix wird "x" angezeigt.

Andernfalls ist der Wegpunkt richtig und "√" wird angezeigt.

Jetzt ist die Stufe **CHECK** und startet einen 1-Sekunden-Timer **timerCheck**, um die Funktion `creatGlyph ()` in einer Sekunde aufzurufen.

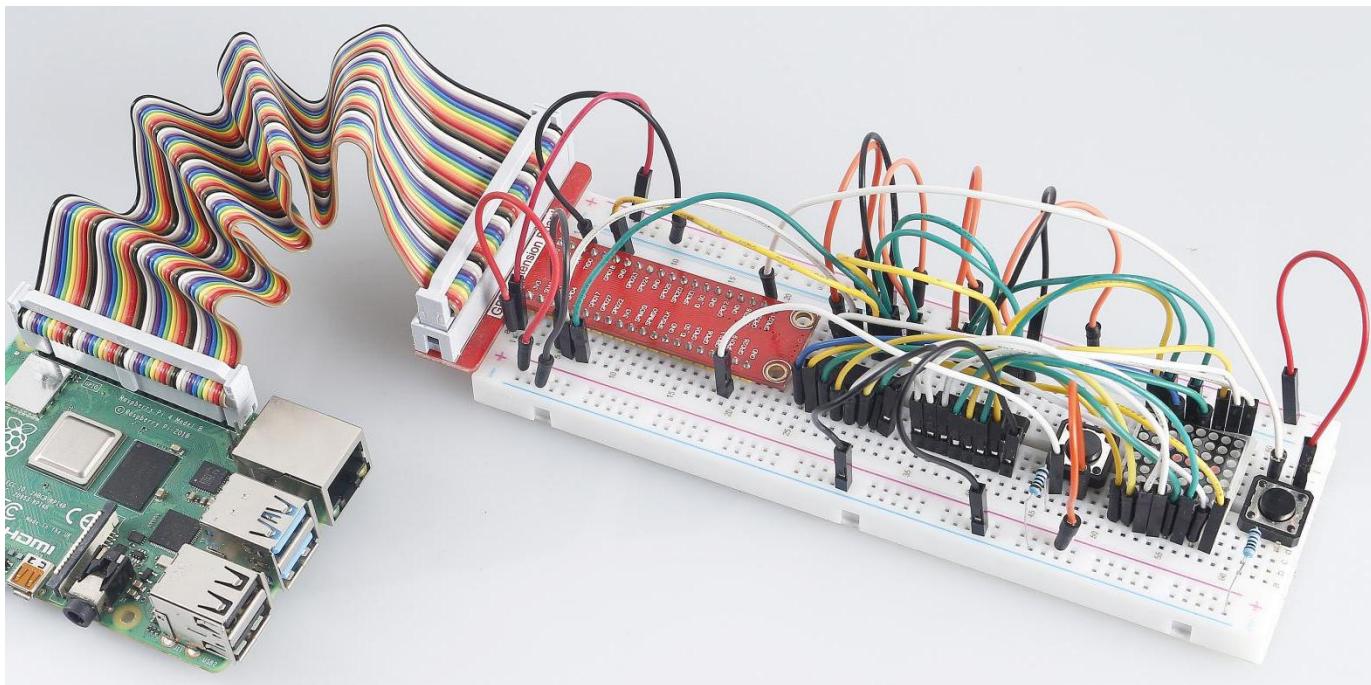
```
def timeOut():  
    checkPoint("empty")
```

Setzen Sie im Funktionszeitlimit() den Parameter von `checkPoint()` auf "**leer**".

```
def getKey():  
    if GPIO.input(AButtonPin)==1 and GPIO.input(BButtonPin)==0:  
        checkPoint("right")  
    elif GPIO.input(AButtonPin)==0 and GPIO.input(BButtonPin)==1:  
        checkPoint("left")
```

`getKey()` liest den Status dieser beiden Schaltflächen, und wenn die rechte Schaltfläche gedrückt wird, ist der Parameter von `checkPoint()` **richtig**. Wenn die **linke** Taste gedrückt wird, bleibt der Parameter übrig.

Phänomen Bild



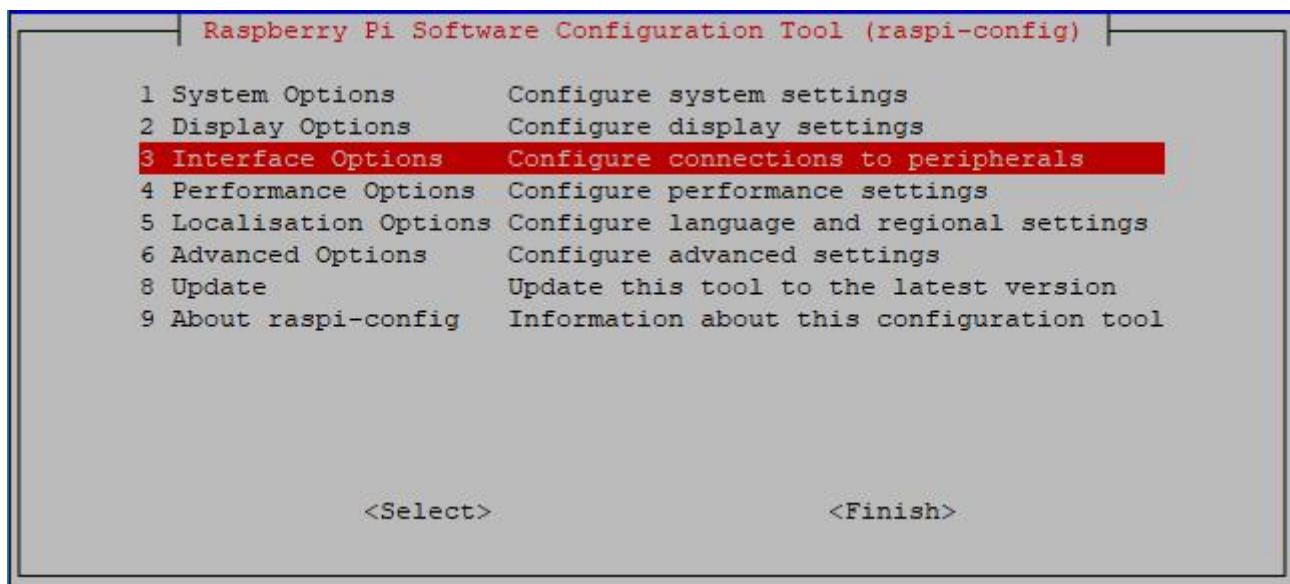
3.2 Anhang

3.2.1 I2C-Konfiguration

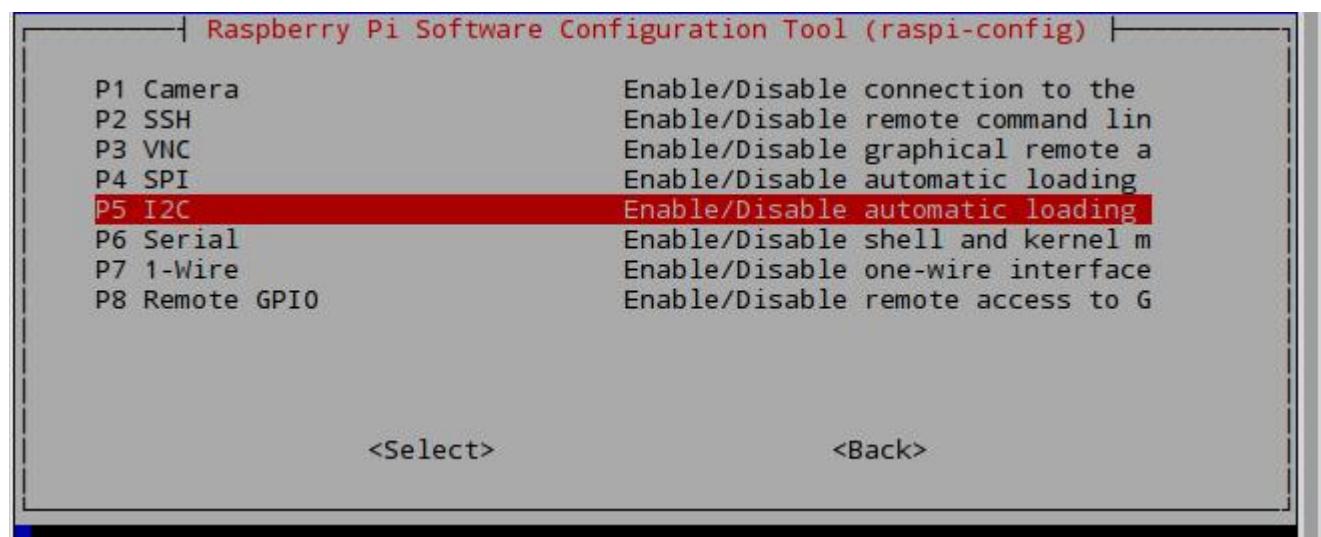
Schritt 1: Aktivieren Sie den I2C-Port Ihres Raspberry Pi (Wenn Sie ihn aktiviert haben, überspringen Sie diesen; wenn Sie nicht wissen, ob Sie dies getan haben oder nicht, fahren Sie bitte fort).

```
sudo raspi-config
```

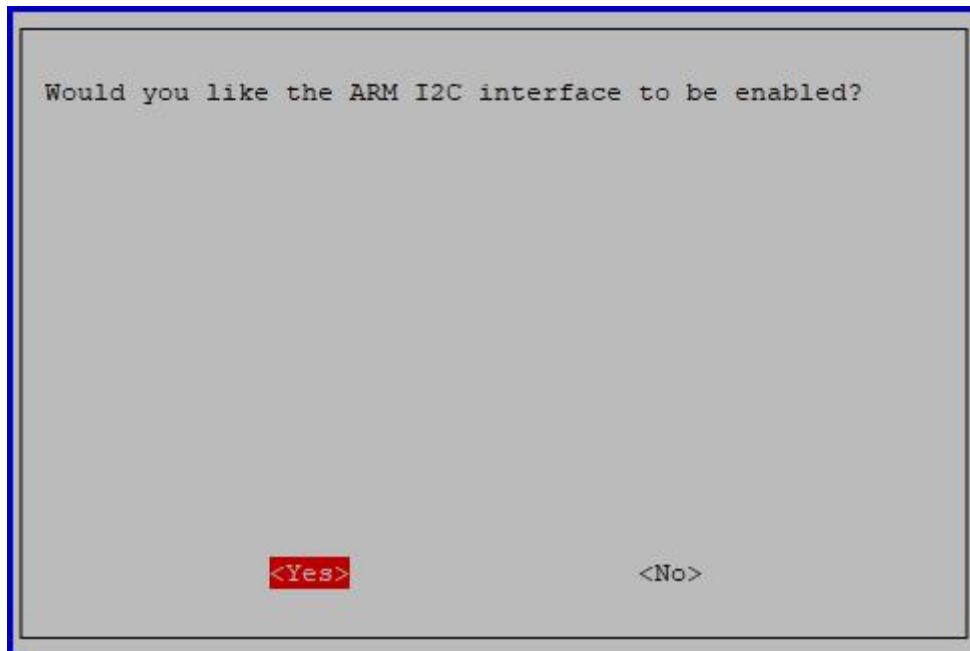
3 Schnittstellenoptionen



P5 I2C



<Yes>, dann<Ok> -><Finish>



Schritt 2: Überprüfen Sie, ob die i2c-Module geladen und aktiv sind.

```
lsmod | grep i2c
```

Dann erscheinen die folgenden Kode (die Nummer kann unterschiedlich sein)

```
i2c_dev          6276    0
i2c_bcm2708      4121    0
```

Schritt 3: Installieren Sie i2c-tools.

```
sudo apt-get install i2c-tools
```

Schritt 4: Überprüfen Sie die Adresse des I2C-Geräts.

```
i2cdetect -y 1      # For Raspberry Pi 2 and higher version
i2cdetect -y 0      # For Raspberry Pi 1
pi@raspberrypi ~ $ i2cdetect -y 1
  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -----
10: -----
20: -----
30: -----
40: ----- 48 -----
50: -----
60: -----
70: -----
```

Wenn ein I2C-Gerät angeschlossen ist, sind die Ergebnisse wie oben gezeigt ähnlich - da die Adresse des Geräts 0x48 lautet, wird 48 gedruckt.

Schritt 5:

Für Benutzer in C-Sprache: Installieren Sie libi2c-dev.

```
sudo apt-get install libi2c-dev
```

Für Python-Benutzer: Installieren Sie smbus für I2C.

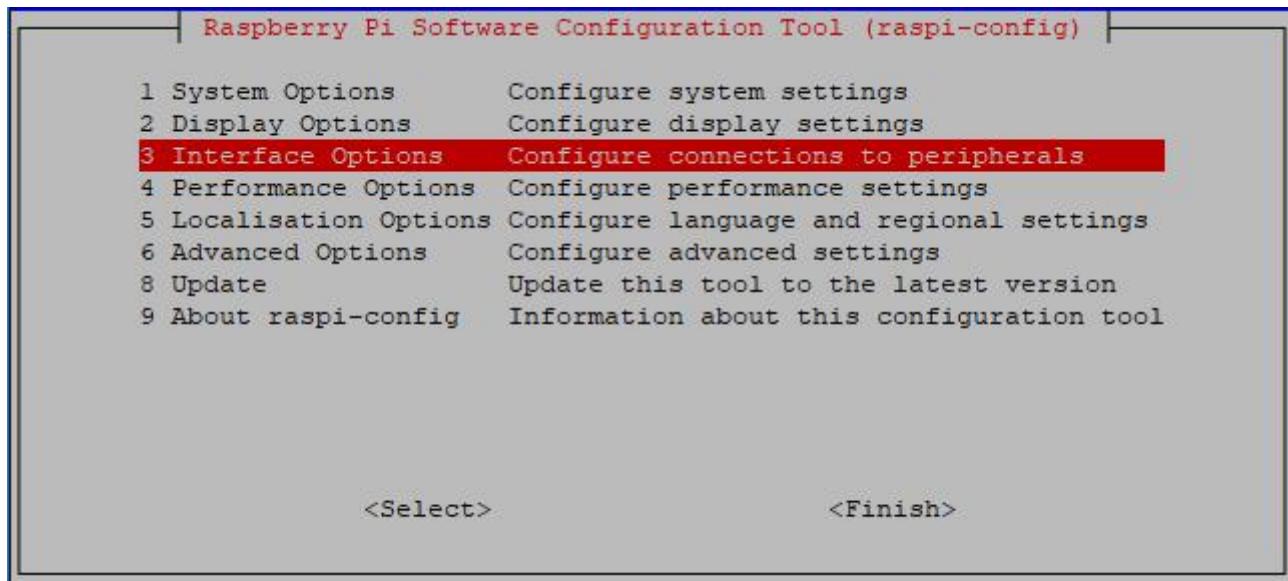
```
sudo apt-get install python-smbus
```

3.2.2 SPI-Konfiguration

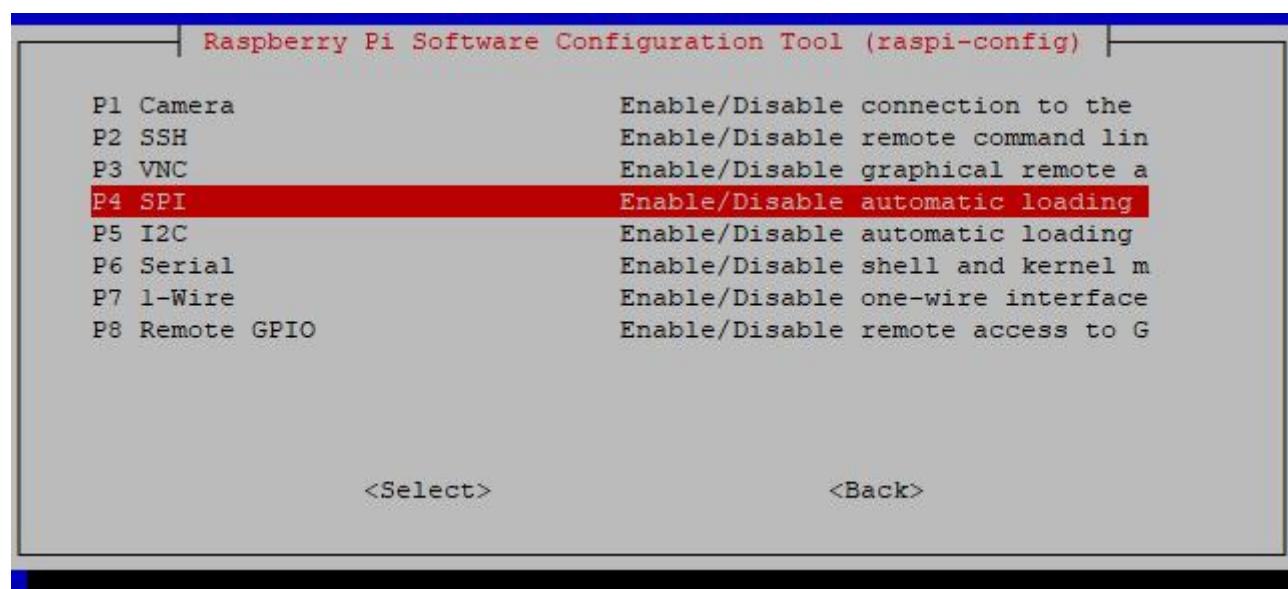
Schritt 1: Aktivieren Sie den SPI-Port Ihres Raspberry Pi (Wenn Sie ihn aktiviert haben, überspringen Sie diesen; wenn Sie nicht wissen, ob Sie dies getan haben oder nicht, fahren Sie bitte fort).

```
sudo raspi-config
```

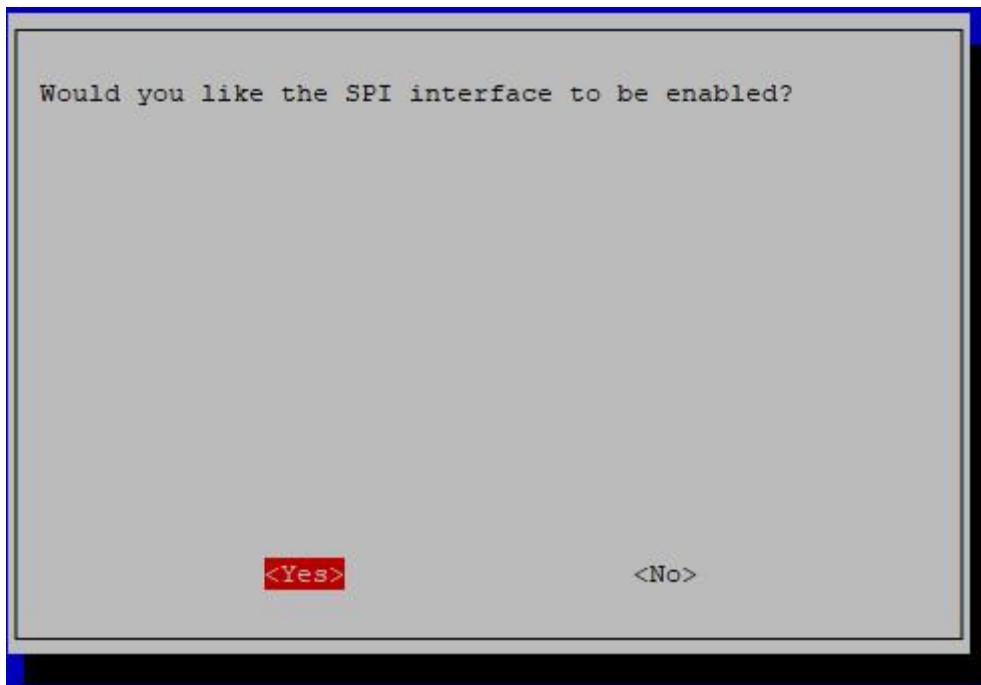
3 Schnittstellenoptionen



P4 SPI



<YES>, dann klick<OK> und<Finish> .



Schritt 2: Überprüfen Sie, ob die i2c-Module geladen und aktiv sind.

```
ls /dev/sp*
```

Dann erscheinen die folgenden Kode (die Nummer kann unterschiedlich sein).

```
/dev/spidev0.0  /dev/spidev0.1
```

Schritt 3: Installieren Sie das SPI-Py Modul.

```
git clone https://github.com/lthiery/SPI-Py.git  
cd SPI-Py  
sudo python3 setup.py install
```

Hinweis: Dieser Schritt ist für Python-Benutzer gedacht. Wenn Sie die Sprache C verwenden, überspringen Sie diese bitte.

3.2.3 Remotedesktop

Es gibt zwei Möglichkeiten, den Desktop des Raspberry Pi fernzusteuern: mit **VNC** und **XRDP**.

VNC

Sie können die Funktion des Remotedesktops über VNC verwenden.

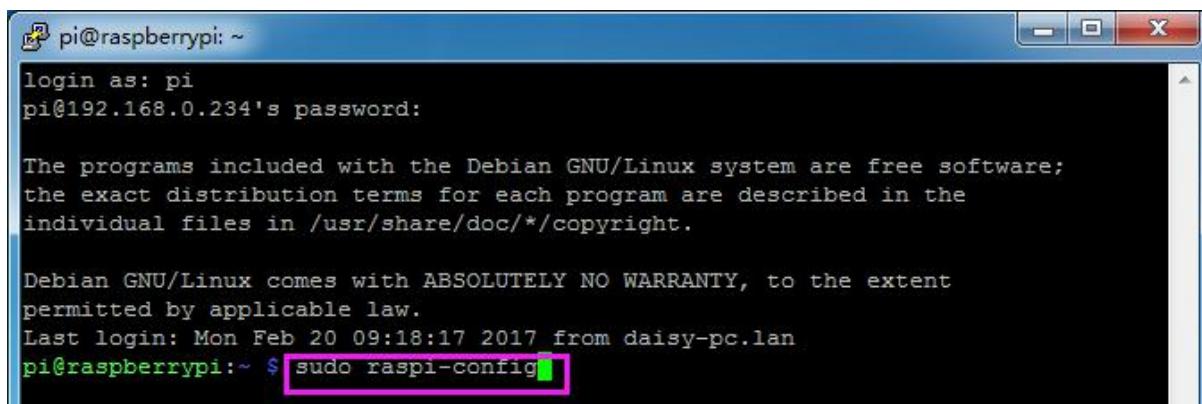
Aktivieren Sie den VNC-Dienst

Der VNC-Dienst wurde im System installiert. Mit Default ist VNC deaktiviert. Sie müssen es in der Konfiguration aktivieren.

Schritt 1

Geben Sie den folgenden Befehl ein:

```
sudo raspi-config
```



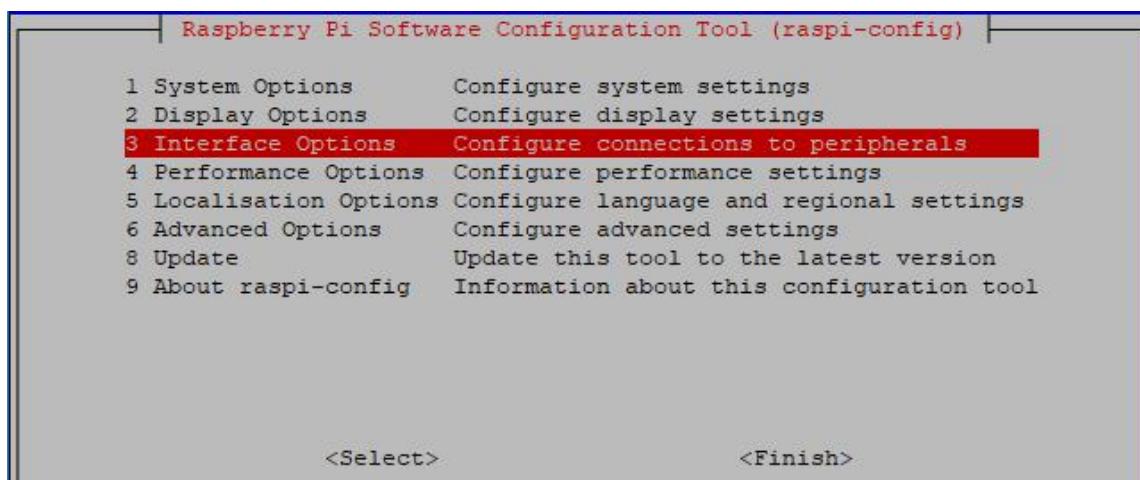
```
pi@raspberrypi: ~
login as: pi
pi@192.168.0.234's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Feb 20 09:18:17 2017 from daisy-pc.lan
pi@raspberrypi:~ $ sudo raspi-config
```

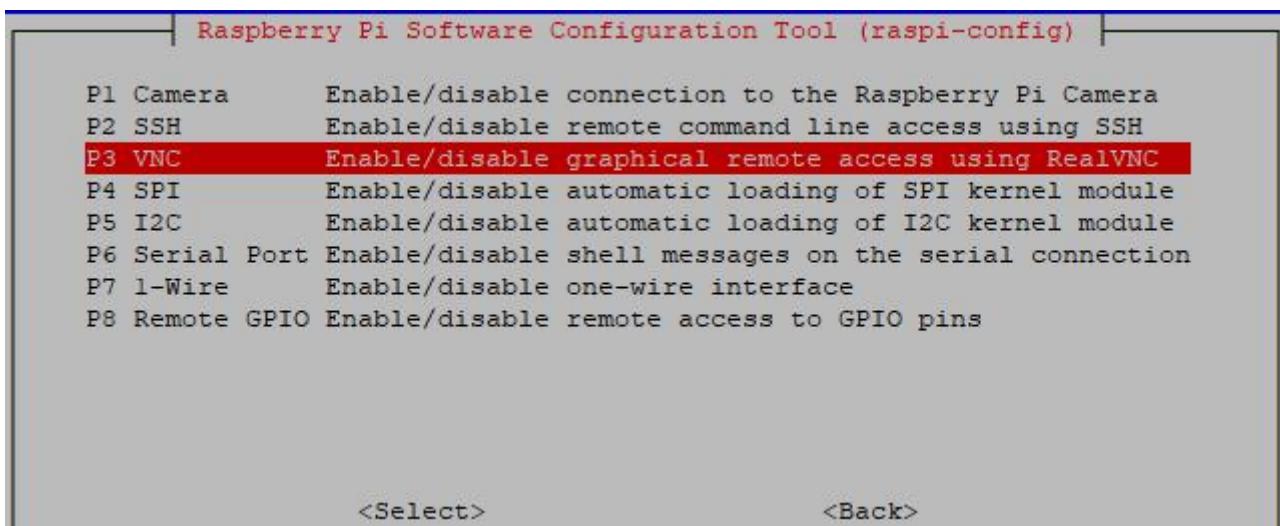
Schritt 2

Wählen Sie **3 Schnittstellenoptionen**, indem Sie die Abwärtspfeiltaste auf Ihrer Tastatur drücken und dann die **Eingabetaste** drücken.



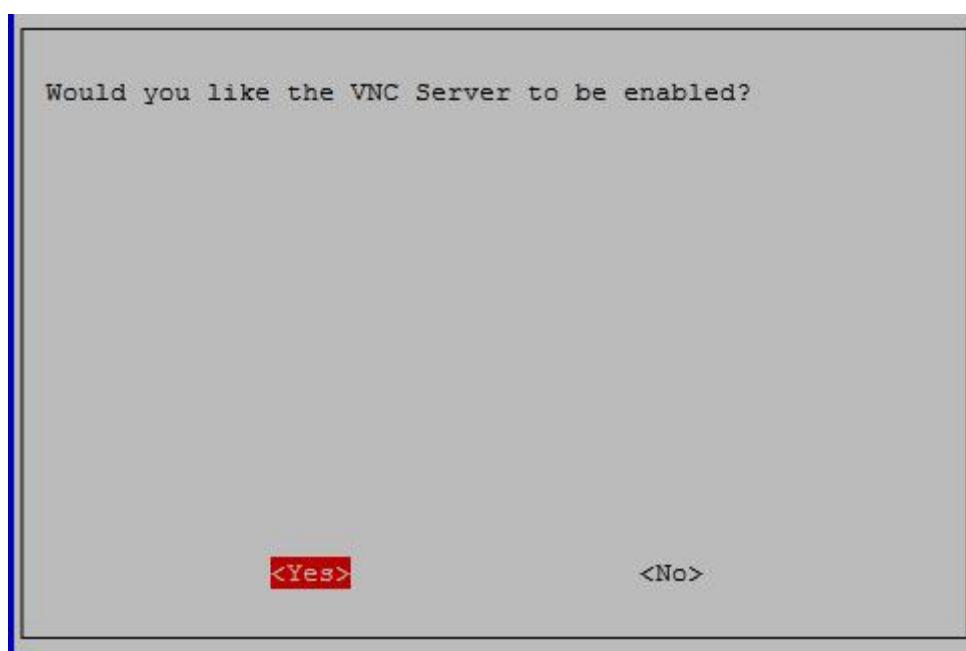
Schritt 3

P3 VNC



Schritt 4

Wählen Sie **Ja** -> **OK** -> **Fertig** stellen, um die Konfiguration zu beenden.



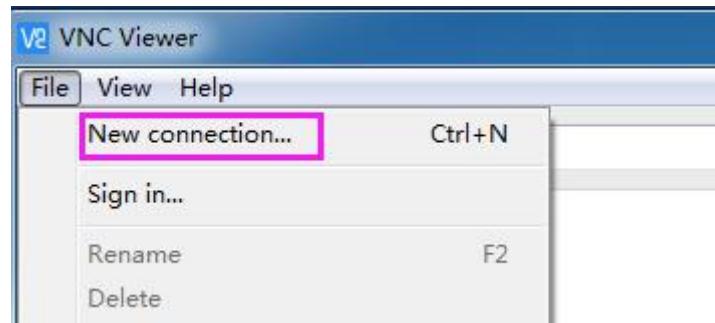
Melden Sie sich bei VNC an

Schritt 1

Sie müssen den **VNC Viewer** auf einem PC installieren. Öffnen Sie die Installation nach Abschluss der Installation.

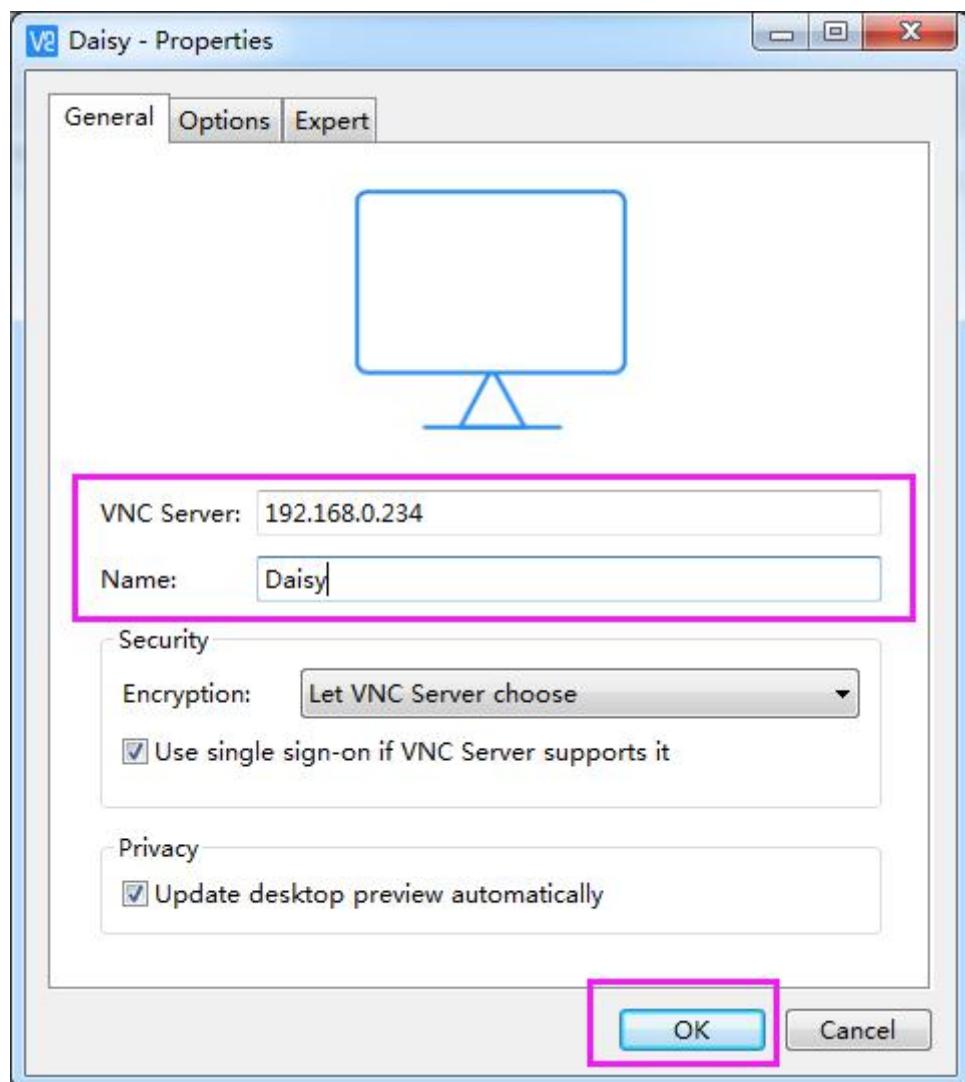
Schritt 2

Wählen Sie dann "Neue Verbindung".



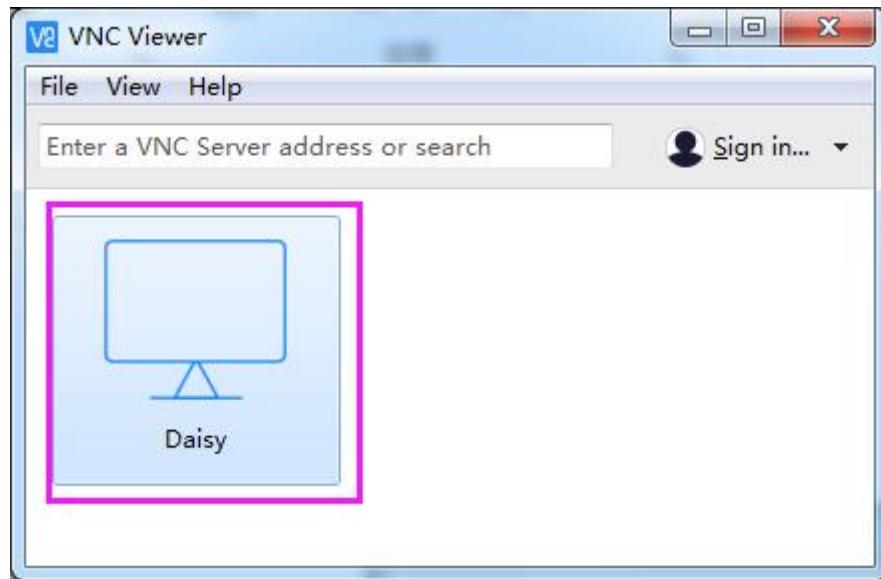
Schritt 3

Geben Sie die IP-Adresse des Raspberry Pi und einen beliebigen **Namen** ein.



Schritt 4

Doppelklicken Sie auf die gerade erstellte **Verbindung**:



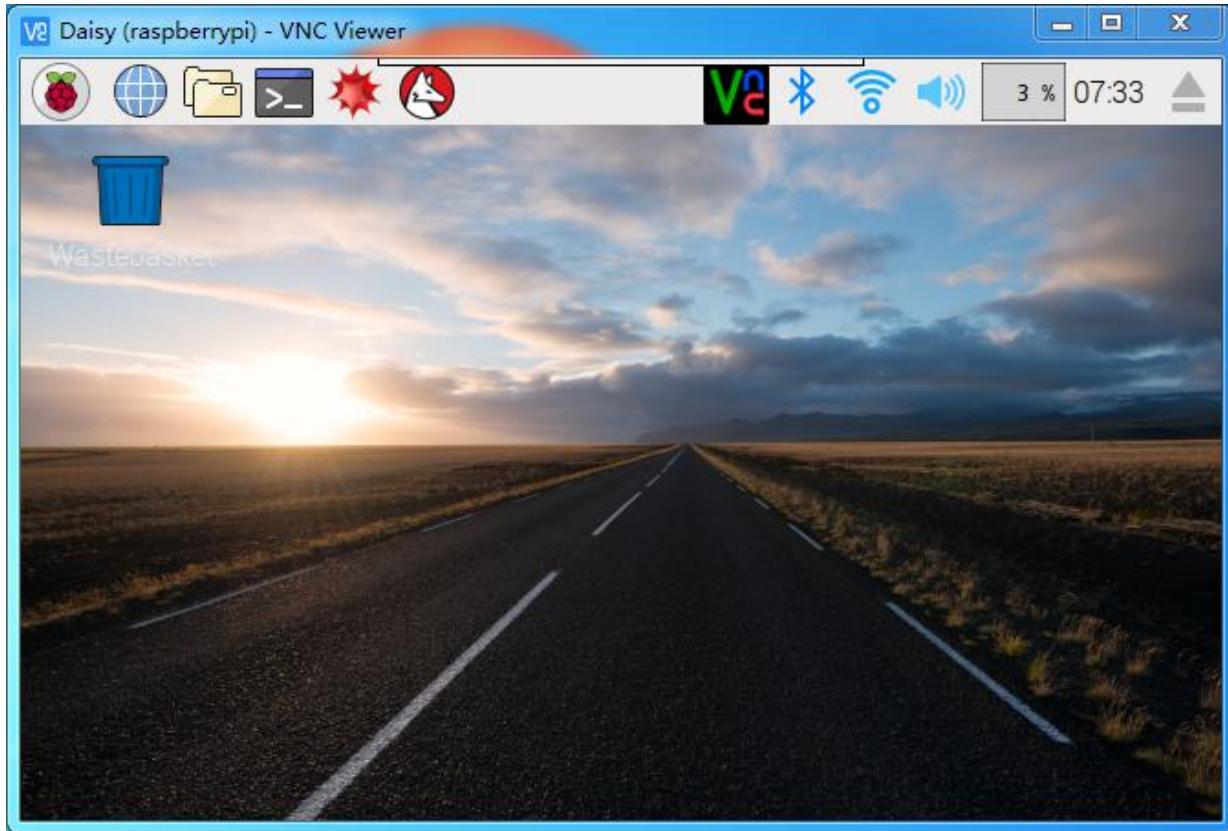
Schritt 5

Geben Sie Benutzername (**pi**) und Passwort ein (**raspberry** mit Default).



Schritt 6

Jetzt können Sie den Desktop des Raspberry Pi sehen:



XRDP

xrdp bietet eine grafische Anmeldung an Remotecomputern mit RDP (Microsoft Remote Desktop Protocol).

Installieren Sie XRDP

Schritt 1

Melden Sie sich mit SSH bei Raspberry Pi an.

Schritt 2

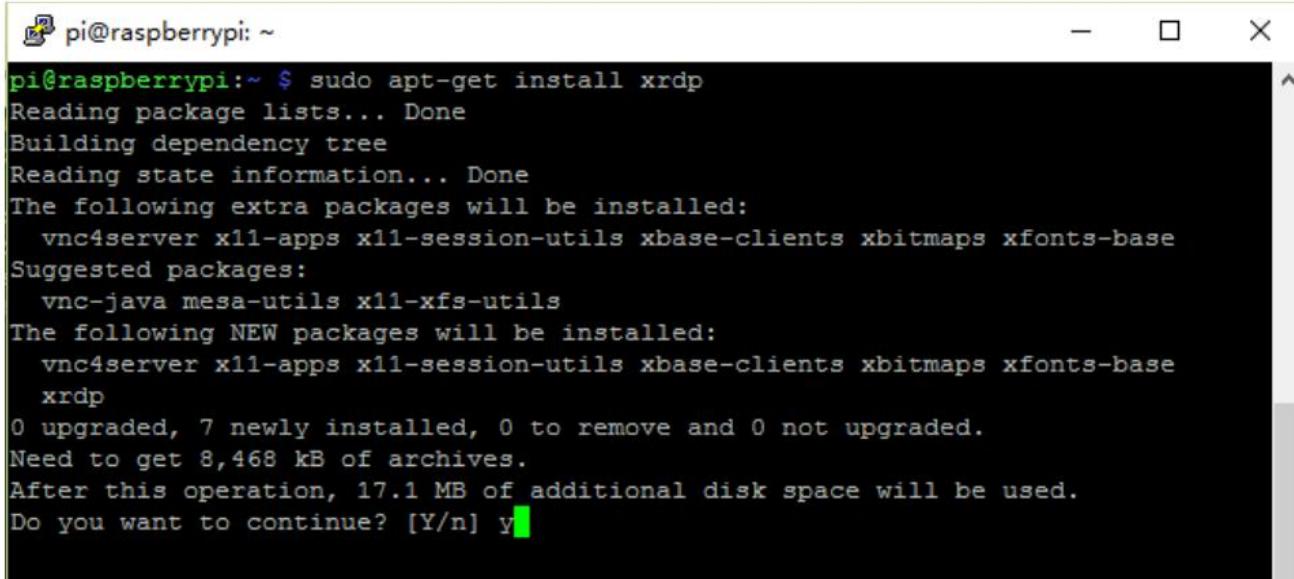
Geben Sie die folgenden Anweisungen ein, um XRDP zu installieren.

```
sudo apt-get update  
sudo apt-get install xrdp
```

Schritt 3

Später beginnt die Installation.

Geben Sie "Y" ein und drücken Sie zur Bestätigung die Taste "Enter".



A screenshot of a terminal window titled "pi@raspberrypi: ~". The window contains the following text output from a sudo apt-get install xrdp command:

```
pi@raspberrypi:~ $ sudo apt-get install xrdp
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  vnc4server x11-apps x11-session-utils xbase-clients xbitmaps xfonts-base
Suggested packages:
  vnc-java mesa-utils x11-xfs-utils
The following NEW packages will be installed:
  vnc4server x11-apps x11-session-utils xbase-clients xbitmaps xfonts-base
  xrdp
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.
Need to get 8,468 kB of archives.
After this operation, 17.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Schritt 4

Wenn Sie die Installation abgeschlossen haben, sollten Sie sich mit Windows-Remotedesktopanwendungen bei Ihrem Raspberry Pi anmelden.

Melden Sie sich bei XRDП an

Schritt 1

Wenn Sie Windows-Benutzer sind, können Sie die mit Windows gelieferte Remotedesktopfunktion verwenden. Wenn Sie Mac-Benutzer sind, können Sie Microsoft Remote Desktop aus dem APP Store herunterladen und verwenden, und es gibt keinen großen Unterschied dazwischen. Das nächste Beispiel ist der Windows-Remotedesktop.

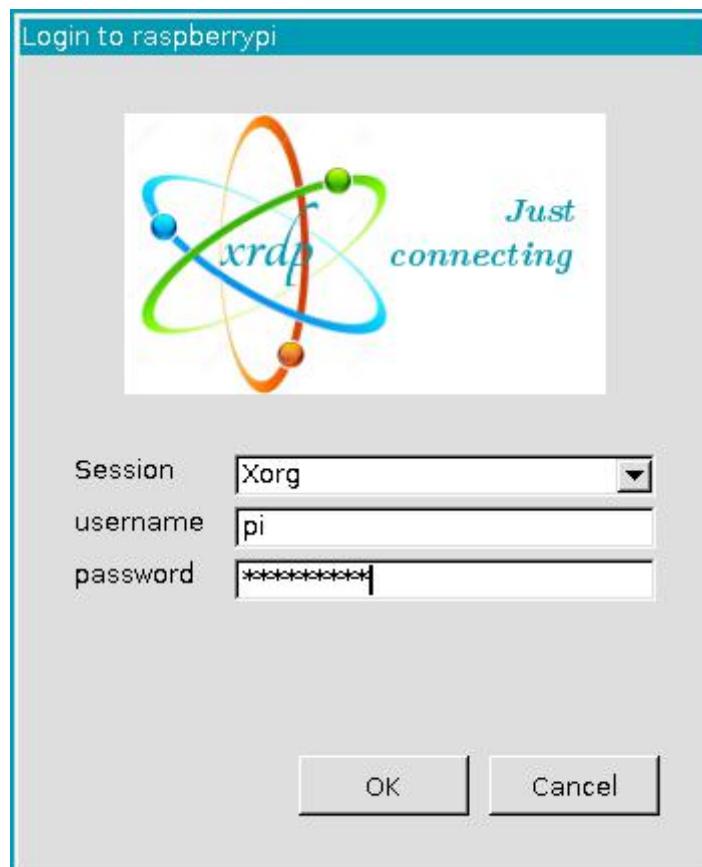
Schritt 2

Geben Sie "mstsc" in "Ausführen" (WIN + R) ein, um die Remotedesktopverbindung zu öffnen, und geben Sie die IP-Adresse von Raspberry Pi ein. Klicken Sie dann auf "Verbinden".



Schritt 3

Dann erscheint die xrdp-Anmeldeseite. Bitte geben Sie Ihren Benutzernamen und Ihr Passwort ein. Danach klicken Sie bitte auf "OK". Wenn Sie sich zum ersten Mal anmelden, lautet Ihr Benutzername "pi" und das Passwort "raspberry".



Schritt 4

Hier können Sie sich erfolgreich über den Remotedesktop bei RPi anmelden.



Urheberrechtshinweis

Alle Inhalte, einschließlich, aber nicht beschränkt auf Texte, Bilder und Kode in diesem Handbuch, sind Eigentum der SunFounder Unternehmen. Sie sollten es nur für persönliche Studien, Nachforschungen, Genüsse oder andere nichtkommerzielle oder gemeinnützige Zwecke gemäß den entsprechenden Bestimmungen und Urheberrechtsgesetzen verwenden, ohne die gesetzlichen Rechte des Autors und der relevanten Rechteinhaber zu verletzen. Für jede Person oder Organisation, die diese ohne Erlaubnis für kommerzielle Zwecke nutzt, behält sich das Unternehmen das Recht vor, rechtliche Schritte einzuleiten.