KAPITEL 11

Motoren

11.0 Einführung

In diesem Kapitel wollen wir uns ansehen, wie sich unterschiedliche Motortypen am Raspberry Pi betreiben lassen.

11.1 Servomotoren steuern

Sehen Sie sich das dazugehörige Video auf *http://razzpisampler.oreilly.com* an.

Problem

Sie möchten einen Raspberry Pi nutzen, um die Position eines Servomotors zu steuern.

Lösung

Nutzen Sie PWM, um die Impulsbreite am Servomotor und damit dessen Winkel zu ändern. Das funktioniert zwar, doch die erzeugte PWM ist nicht ganz stabil, weshalb der Servo leicht zittert. Eine alternative Lösung, die wesentlich stabilere Impulse erzeugt, bietet die ServoBlaster-Treibersoftware (Rezept 11.2).

Bei einem älteren Raspberry Pi 1 sollten Sie den Servo separat mit 5 V versorgen, da die auftretenden Stromspitzen den Raspberry Pi zum Absturz bringen oder überlasten können. Wenn Sie einen Raspberry Pi B+ oder neuer nutzen, sorgen die Verbesserungen an der Board-eigenen Spannungsversorgung dafür, dass Sie kleine Servos direkt über den 5 V-Pin des GPIO-Anschlusses versorgen können.

Abbildung 11-1 zeigt einen kleinen 9g-Servo (Anhang A, *Vermischtes*, Seite 453), der problemlos mit einem Raspberry Pi B+ läuft.

Simon Monk: Raspberry-Pi-Kochbuch (ISBN Print: 978-3-96009-033-5), 2017

Die Anschlüsse des Servos sind üblicherweise farblich gekennzeichnet: 5V mit rot, Masse mit braun und die Steuerung mit orange. 5V und Masse werden mit den 5V- und GND-Pins des GPIO-Anschlusses verbunden. Das Steuerungskabel wird mit Pin 18 verbunden.

Wenn Sie mit einer separaten Stromversorgung arbeiten, bietet ein Steckbrett eine gute Möglichkeit, alle Anschlüsse zusammenzuhalten.

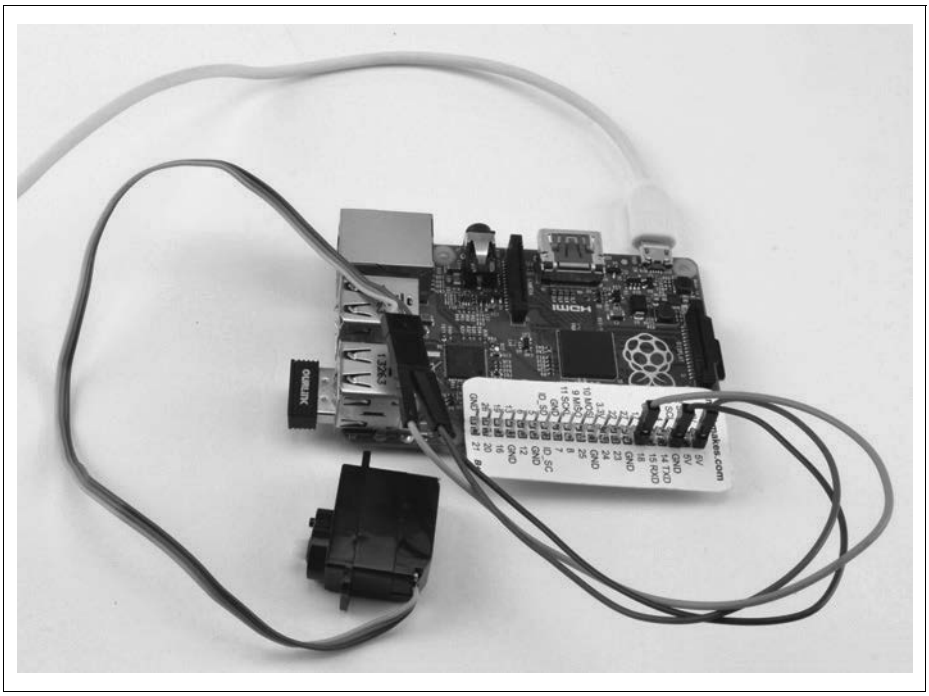


Abbildung 11-1: Direkt angeschlossener kleiner Servo am Rasbperry Pi B+

Für dieses Rezept benötigen Sie folgende Komponenten:

- 5-V-Servomotor (Anhang A, *Vermischtes*, Seite 453)
- Steckbrett und Schaltdrähte (Anhang A, *Prototyping-Zubehör*, Seite 450)
- 1-kΩ-Widerstand (Anhang A, *Widerstände und Kondensatoren*)
- 5-V-/1-A-Stromversorgung oder 4,8-V-Batteriepaket (Anhang A, *Vermischtes*, Seite 453)

Den Aufbau auf dem Steckbrett zeigt Abbildung 11-2.

Der 1-kΩ-Widerstand ist nicht unbedingt erforderlich, schützt den GPIO-Pin aber vor unerwartet hohen Strömen im Steuersignal, die auftreten können, wenn es einen Fehler am Servo gibt.

Wenn Sie möchten, können Sie den Servo auch über einen Akku-Pack betreiben. Ein Batteriehalter für vier AA-Akkus liefert um die 4,8V und reicht für ein Servo

aus. Vier Alkali-AA-Zellen liefern 6V und reichen für mehrere Servos, allerdings sollten Sie im Datenblatt prüfen, ob der Servo die 6V verträgt.

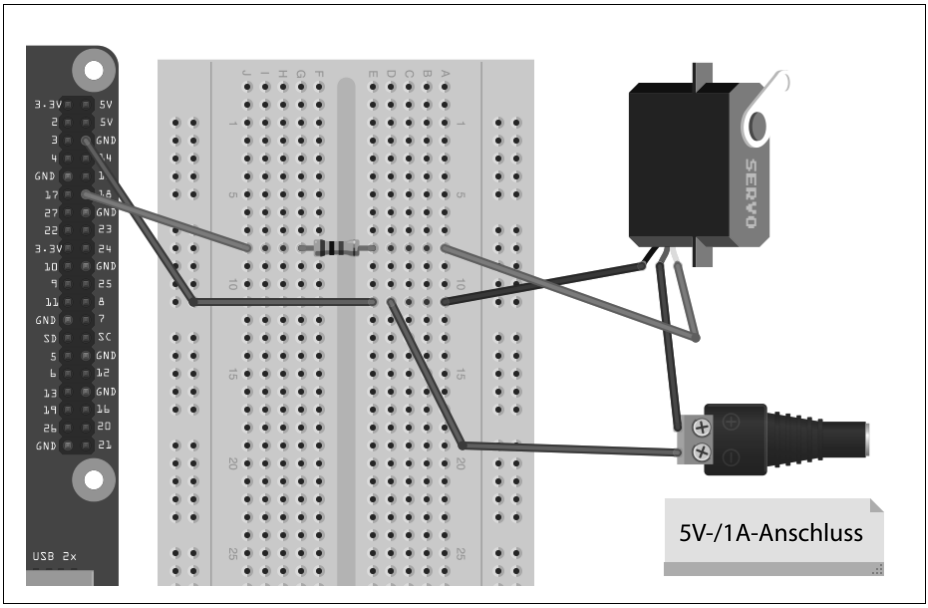


Abbildung 11-2: Einen Servomotor steuern

Die Benutzerschnittstelle zum Einstellen des Winkels basiert auf dem *gui_slider.py*-Programm zur Steuerung der Helligkeit einer LED (Rezept 10.9). Es wurde so modifiziert, dass mit dem Schieberegler ein Winkel von 0 bis 180 Grad eingestellt werden kann (Abbildung 11-3).

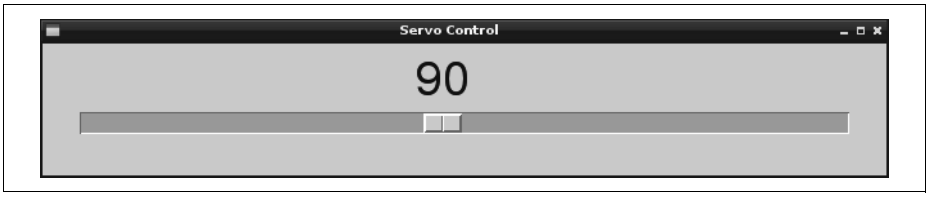


Abbildung 11-3: Benutzerschnittstelle zur Steuerung eines Servomotors

Öffnen Sie einen Editor (nano oder IDLE) und kopieren Sie den folgenden Code. Wie alle Programmbeispiele aus diesem Buch können Sie auch dieses aus dem Codebereich von *http://www.oreilly.de/raspberrykochbuch* herunterladen (unter dem Namen *servo.py*).

Beachten Sie, dass das Programm eine grafische Benutzerschnittstelle nutzt, d.h., Sie können es nicht über SSH ausführen.

Simon Monk: Raspberry-Pi-Kochbuch (ISBN Print: 978-3-96009-033-5), 2017

Sie müssen es in der Windows-Umgebung des Pi ausführen oder über VNC (Rezept 2.8), bzw. RDP (Rezept 2.9) fernsteuern. Sie müssen es außerdem als Superuser ausführen, also mit `sudo python servo.py` starten:

```
from Tkinter import *
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
pwm = GPIO.PWM(18, 100)
pwm.start(5)

class App:

    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        scale = Scale(frame, from_=0, to=180,
              orient=HORIZONTAL, command=self.update)
        scale.grid(row=0)

    def update(self, angle):
        duty = float(angle) / 10.0 + 2.5
        pwm.ChangeDutyCycle(duty)

root = Tk()
root.wm_title('Servo Control')
app = App(root)
root.geometry("200x50+0+0")
root.mainloop()
```

Diskussion

Servomotoren werden bei der (Fern-)Steuerung von Fahrzeugen und Robotern eingesetzt. Die meisten Servomotoren sind nicht *umlaufend*, d.h., sie drehen sich nicht komplett, sondern nur in einem Winkel bis zu etwa 180 Grad.

Die Position des Servomotors wird durch die Impulslänge gesetzt. Der Servo erwartet einen Impuls alle 20 Millisekunden. Wenn der Puls für eine Millisekunde high ist, ist sein Winkel bei 0. Bei 1,5 Millisekunden ist er in der Mittelstellung und bei 2 Millisekunden bei 180 Grad (Abbildung 11-4).

Beim Beispielprogramm wird eine PWM-Frequenz von 100 Hz festgelegt, d.h., der Servo erhält alle 10 Millisekunden einen Impuls. Tatsächlich werden dadurch auch Impulse erzeugt, die unter dem erwarteten Minimum von einer Millisekunde und über dem Maximum von zwei Millisekunden liegen.

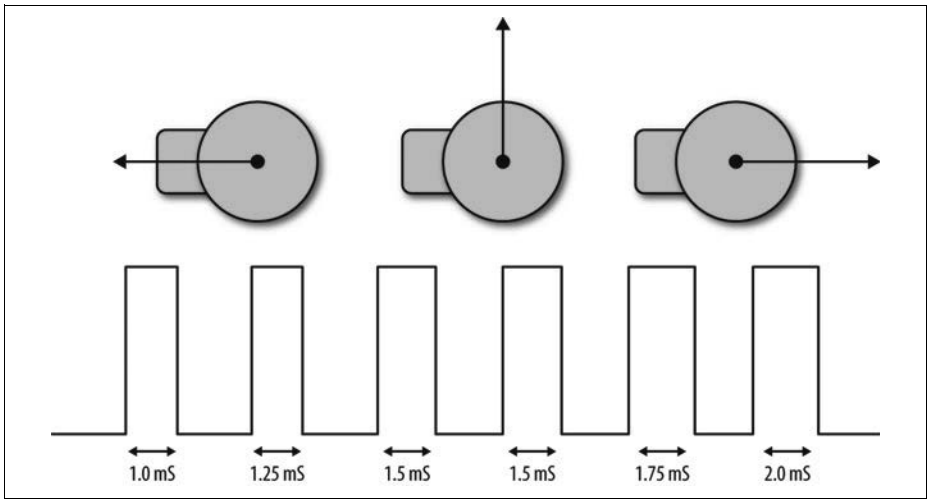


Abbildung 11-4: Servomotoren

Siehe auch

Wenn Sie mit mehreren Servomotoren arbeiten oder eine größere Stabilität und Präzision benötigen, können Sie ein spezielles Servo-Controller-Modul nutzen, wie es in Rezept 11.3 beschrieben wird.

Adafruit hat eine andere Methode der Servosteuerung (*http://bit.ly/17FVspk*) entwickelt.

Eine alternative Lösung bietet die ServoBlaster-Treibersoftware, die wesentlich stabilere Impulse erzeugt (Rezept 11.2).

11.2 Servomotoren präzise steuern

Problem

Die PWM-Funktion der `RPi.GPIO`-Bibliothek ist für Ihre Servoanwendung nicht präzise und »jitter-frei« genug.

Lösung

Installieren Sie den ServoBlaster-Gerätetreiber.

Die von Richard Hurst entwickelte ServoBlaster-Software nutzt die CPU des Raspberry Pi, um Impulse zu generieren, die wesentlich genauer sind, als dies mit der `RPi.GPIO`-Bibliothek möglich ist. Installieren Sie die Software mit den folgenden Befehlen und starten Sie den Raspberry Pi dann neu.

```
git clone git://github.com/richardghirst/PiBits.git
cd PiBits/ServoBlaster/user
sudo make
sudo make install
```

Das Programm aus Rezept 11.1 kann so angepasst werden, dass es den ServoBlaster-Code nutzt. Das modifizierte Programm finden Sie in der Datei *servo_blaster.py*. Es geht davon aus, dass der Servo-Steuerpin mit GPIO 18 verbunden ist.

```
from Tkinter import *
import os
import time
servo_min = 500 # uS
servo_max = 2500 # uS
servo = 2 # GPIO 18

def map(value, from_low, from_high, to_low, to_high):
  from_range = from_high - from_low
  to_range = to_high - to_low
  scale_factor = float(from_range) / float(to_range) return to_low + (value /
scale_factor)

def set_angle(angle):
  pulse = int(map(angle, 0, 180, servo_min, servo_max))
  command = "echo {}={}us > /dev/servoblaster".format(servo, pulse)
  os.system(command)

class App:
   def __init__(self, master):
      frame = Frame(master)
      frame.pack()
      scale = Scale(frame, from_=0, to=180,
                  orient=HORIZONTAL, command=self.update)
      scale.grid(row=0)

   def update(self, angle):
      set_angle(float(angle))

root = Tk()
root.wm_title('Servo Control')
app = App(root)
root.geometry("200x50+0+0")
root.mainloop()
```

Die Benutzerschnittstelle ist mit der aus Rezept 11.1 identisch. Die Unterschiede liegen in der Funktion `set_angle`. Diese Funktion nutzt eine Hilfsfunktion namens `map`, um einen Winkel mithilfe der Konstanten `servo_min` und `servo_max` in eine Impulsdauer umzurechnen. Daraus wird dann ein Befehl konstruiert, der über die Kommandozeile ausgeführt wird. Dieser Befehl beginnt mit einem `echo`, gefolgt von der Nummer des zu steuernden Servos, einem Gleichheitszeichen und der Impulsdauer in Mikrosekunden. Der Stringteil des Befehls wird dann an das Gerät */dev/servoblaster* weitergeleitet und die Position des Servos geändert.

ServoBlaster beenden

Wenn ServoBlaster, oder genauer gesagt der Dienst *servo.d*, läuft, können Sie die Servo-Pins für nichts anderes verwenden, und Audio funktioniert am Raspberry Pi auch nicht mehr. Sollen die Pins also für etwas anderes genutzt werden, müssen Sie ServoBlaster deaktivieren und den Raspberry Pi neu starten.

```
$ sudo update-rc.d servoblaster disable
$sudo reboot
```

Nach dem Neustart des Raspberry Pi hat ServoBlaster keine Kontrolle mehr über die Pins. Natürlich können Sie den ServoBlaster jederzeit wieder aktivieren:

```
$ sudo update-rc.d servoblaster enable
$ sudo reboot
```

Diskussion

Der ServoBlaster-Treiber ist sehr mächtig und erlaubt es Ihnen, nahzu alle GPIO-Pins zur Steuerung von Servos zu nutzen. Die Standardeinstellung definiert acht GPIO-Pins zur Servosteuerung. Jedem dieser Pins wird ein Kanal zugewiesen. Die Standardzuordnung zeigt Tabelle 11-1.

Tabelle 11-1: ServoBlaster-Standardzuordnung von Kanälen und Pins

Servokanal	GPIO-Pin
0	4
1	17
2	18
3	27
4	22
5	23
6	24
7	25

Der Anschluss so vieler Servos kann einen ziemlichen Kabelsalat verursachen. Ein Board wie das MonkMakes Servo Six vereinfacht die Verkabelung der Servos mit Ihrem Raspberry Pi deutlich.

Siehe auch

Die vollständige Dokumentation zum ServoBlaster finden Sie auf *https://github.com/richard- ghirst/PiBits/tree/master/ServoBlaster*.

Wenn Sie das präzise Timing des ServoBlasters nicht benötigen, kann auch die `RPi.GPIO`-Bibliothek die Impulse für Ihren Servo erzeugen (Rezept 11.1).

11.3 Eine große Anzahl von Servomotoren steuern

Problem

Sie möchten eine Reihe von Servomotoren mit hoher Präzision steuern.

Lösung

Zwar können Sie mit dem ServoBlaster-Code aus Rezept 11.2 viele Servos präzise steuern, das löst aber das Problem der Verkabelung nicht. Sie können das zwar auf einem Steckbrett aufbauen, doch das ist ziemlich chaotisch und irgendein Kabel steckt garantiert nicht richtig.

Um den Anschluss einfach zu gestalten, nutzen Sie ein Servomotor-HAT wie in Abbildung 11-5.

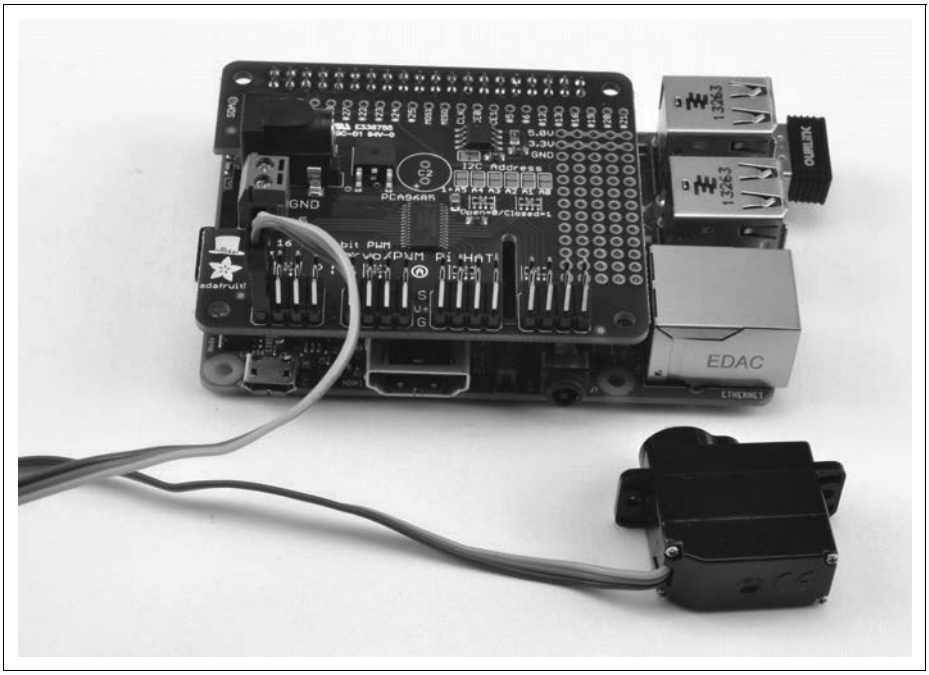


Abbildung 11-5: Adafruit-Servomotor-HAT

Dieses HAT ermöglicht die Steuerung von bis zu 16 Servomotoren oder PWM-Kanälen über die I2C-Schnittstelle des Raspberry Pi. Die Servos werden direkt mit dem HAT verbunden.

Die Spannungsversorgung der Logikschaltung des Moduls erfolgt über den 3,3-V-Anschluss des Raspberry Pi und ist völlig von den Servomotoren getrennt, die über eine externe 5-V-Quelle versorgt werden.

Wenn Sie möchten, können Sie den Servo auch über einen Akku-Pack betreiben. Ein Batteriehalter für vier AA-Akkus liefert um die 4,8 V und reicht für einen Servo aus. Vier Alkali-AA-Zellen liefern 6 V und reichen für mehrere Servos, allerdings sollten Sie im Datenblatt prüfen, ob der Servo die 6 V verträgt.

Die Anschlüsse des Servos sind praktischerweise so angeordnet, dass Sie direkt mit den Pins verbunden werden können. Achten Sie beim Anschluss auf die Polung.

Um die Adafruit-Software für dieses Modul nutzen zu können, müssen Sie I2C auf dem Raspberry Pi aktivieren (Rezept 9.3).

Die Adafruit-Bibliothek ist keine »richtige« Bibliothek mit Installationsskript, sondern besteht nur aus einem Verzeichnis, das eine Reihe von Dateien enthält. Wenn Sie die Bibliothek nutzen, müssen Sie sich daher in diesem Verzeichnis befinden, sonst findet Ihr Programm die Dateien nicht.

Um die gesamte Adafruit-Software-Bibliothek für den Raspberry Pi herunterzuladen, geben Sie Folgendes ein:

```
$ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
$ cd Adafruit-Raspberry-Pi-Python-Code
$ cd Adafruit_PWM_Servo_Driver
```

Mit den beiden letzten Zeilen erfolgt ein Wechsel in das Verzeichnis, das den PWM-Code enthält sowie ein entsprechendes Beispielprogramm von Adafruit, das Sie mit dem folgenden Befehl ausführen können:

```
$ sudo python Servo_Example.py
```

Im folgenden Programm wird unser Beispiel aus Rezept 11.2 so angepasst, dass Sie die Position des Servos über einen Schieberegler zwischen 0 und 180 Grad einstellen können. Die Programmdatei muss im Verzeichnis *Adafruit_PWM_Servo_Driver* abgelegt werden. Der Schieberegler ändert die Servoposition gleichzeitig für die beiden Kanäle 0 und 1, d.h., die beiden Servos bewegen sich parallel, wenn Sie den Schieberegler nutzen.

Öffnen Sie einen Editor (nano oder IDLE) und kopieren Sie den folgenden Code. Wie alle Programmbeispiele aus diesem Buch können Sie auch dieses aus dem Codebereich von *http://www.oreilly.de/raspberrykochbuch* herunterladen (unter dem Namen *servo_module.py*). Beachten Sie, dass das Programm eine grafische Benutzerschnittstelle nutzt und daher nicht über SSH ausgeführt werden kann. Sie müssen es entweder über die Windows-Umgebung auf dem Pi selbst ausführen oder es entfernt über VNC (Rezept 2.8) oder RDP (Rezept 2.9) steuern.

```
from Tkinter import *
from Adafruit_PWM_Servo_Driver import PWM
import time

pwm = PWM(0x40)
pwm.setPWMFreq(50)
```

```
class App:

    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        scale = Scale(frame, from_=0, to=180,
              orient=HORIZONTAL, command=self.update)
        scale.grid(row=0)

    def update(self, angle):
        pulse_len = int(float(angle) * 500.0 / 180.0) + 110
        pwm.setPWM(0, 0, pulse_len)
        pwm.setPWM(1, 0, pulse_len)

root = Tk()
root.wm_title('Servo Control')
app = App(root)
root.geometry("200x50+0+0")
root.mainloop()
```

Diskussion

Die erste Zeile nach dem Import erzeugt eine neue PWM-Instanz. Sie nutzt dazu die I2C-Adresse als Argument, in diesem Fall 0×40. Das Modul besitzt Lötpunkte, über die Sie die I2C-Adresse ändern können, falls es zu Konflikten mit anderen I2C-Geräten kommt oder wenn Sie mehrere dieser Module nutzen möchten.

In der nächsten Zeile wird eine PWM-Frequenz von 50 Hz festgelegt, mit der alle 20 Millisekunden ein Update-Impuls geliefert wird. Die Zeile, mit der die eigentliche PWM für einen bestimmten Kanal eingestellt wird, lautet folgendermaßen:

```
pwm.setPWM(0, 0, pulse_len)
```

Das erste Argument gibt den PWM-Kanal an, dessen Tastverhältnis geändert werden soll. Jede Periode wird in 4096 Ticks unterteilt. Das zweite Argument enthält den Tick, bei dem der Impuls beginnen soll. Dieser ist immer 0. Das dritte Argument gibt den Tick an, bei dem der Impuls enden soll. Die beiden Konstanten 500. 0 und 110 in der folgenden Zeile wurden durch Austesten ermittelt und sollen einen Standardservo so nah wie möglich an die 180 Grad bringen:

```
    pulse_len = int(float(angle) * 500.0 / 180.0) + 110
```

Bei der Wahl einer Stromversorgung für dieses Modul müssen Sie daran denken, dass ein normaler Servo leicht 400 mA zieht, wenn er sich bewegt, und unter Last noch mehr. Wenn Sie also viele Servos gleichzeitig bewegen möchten, benötigen Sie eine entsprechende Stromversorgung.

Siehe auch

Ein Servo-HAT ist eine gute Sache, wenn Ihr Raspberry Pi nahe der Servomotoren liegt. Sind die Motoren hingegen weiter vom Raspberry Pi weg, können Sie von Adafruit auch ein Servomodul erwerben (Produkt-ID 815), die die gleiche Servo-

Controller-Hardware verwendet wie das Servo-HAT, aber nur vier Pins hat, das die I2C-Schnittstelle des Boards mit dem I2C-Interface des Raspberry Pi verbinden.

Sehen Sie sich die Adafruit-Dokumentation) zu deren Raspberry Pi-Bibliothek an (*http://bit.ly/1iomEey*.

11.4 Die Geschwindigkeit eines Gleichstrommotors steuern

Problem

Sie möchten die Geschwindigkeit eines Gleichstrommotors mit dem Raspberry Pi steuern.

Lösung

Sie können den gleichen Entwurf wie in Rezept 10.5 verwenden, allerdings sollten Sie eine Diode zwischen den Motor schalten, damit Stromspitzen nicht den Transistor oder gar den Raspberry Pi beschädigen. Die 1N4001 ist dafür eine geeignete Diode. Sie hat an einer Seite einen Streifen. Achten Sie also auf die richtige Polung.

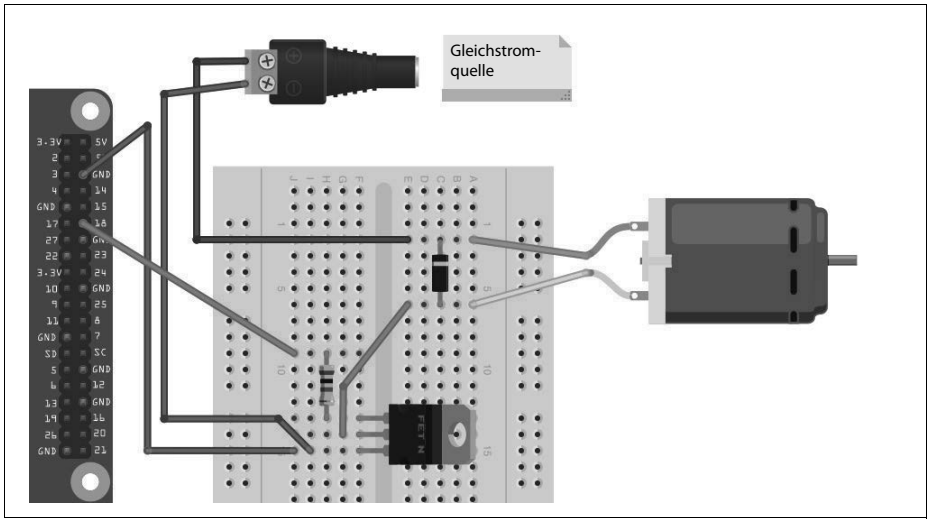


Abbildung 11-6: Gleichstrommotor mit hoher Leistung steuern

Sie benötigen folgende Komponenten:

- 3-V- bis 12-V-Gleichstrommotor
- Steckbrett und Schaltdrähte (Anhang A, *Prototyping-Zubehör*, Seite 450)

- 1-kΩ-Widerstand (Anhang A, *Widerstände und Kondensatoren*, Seite 451)
- MOSFET-Transistor FQP30N06L (Anhang A, *Transistoren und Dioden*, Seite 451)
- Diode 1N4001 (Anhang A, *Transistoren und Dioden*, Seite 451)
- Stromversorgung mit einer für den Motor ausreichenden Leistung

Zur Steuerung der Geschwindigkeit des Motors können Sie das Programm aus dem Codebereich von *http://www.oreilly.de/raspberrykochbuch* herunterladen, wo es unter dem Namen *gui_slider.py* zu finden ist. Beachten Sie, dass das Programm eine grafische Benutzerschnittstelle verwendet und daher nicht über SSH ausgeführt werden kann. Sie müssen es in der Windows-Umgebung auf dem Pi selbst ausführen oder es entfernt über VNC (Rezept 2.8) oder RDP (Rezept 2.9) steuern.

Diskussion

Bei einem Gleichstrommotor mit geringer Leistung (unter 200 mA) können Sie einen kleineren (und billigeren) Transistor wie den 2B3904 verwenden (Anhang A, *Transistoren und Dioden*, Seite 451). Den Aufbau für einen 2N3904 sieht auf einem Steckbrett so aus wie in (Abbildung 11-7).

Bei einem kleinen Motor reicht es wahrscheinlich aus, wenn Sie ihn über die 5-V-Leitung des GPIO-Anschlusses versorgen. Wenn der Raspberry Pi dabei abstürzt, müssen Sie eine externe Stromversorgung anschließen (Abbildung 11-6).

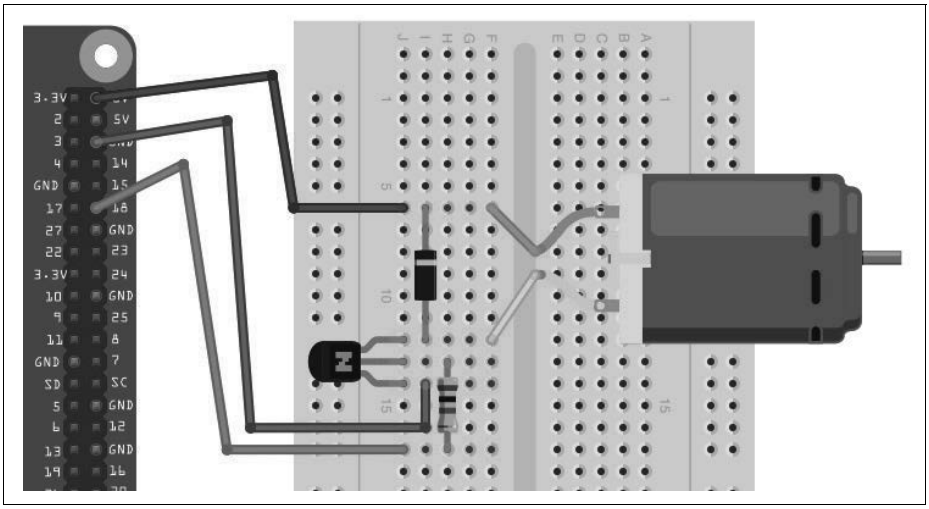


Abbildung 11-7: Gleichstrommotor mit geringer Leistung steuern

Siehe auch

Mit dieser Schaltung wird nur die Geschwindigkeit des Motors gesteuert. Sie kann nicht die Drehrichtung des Motors beeinflussen. Wie das funktioniert, wird in Rezept 11.5 erläutert.

Die Nutzung von Steckbrett und Schaltdrähten mit dem Raspberry Pi wird in Rezept 9.8 beschrieben.

11.5 Die Drehrichtung eines Gleichstrommotors steuern

Problem

Sie möchten sowohl die Geschwindigkeit als auch die Drehrichtung eines kleinen Gleichstrommotors steuern.

Lösung

Nutzen Sie einen H-Brücken-Chip oder ein H-Brücken-Modul.

Zur Steuerung des Motors stehen zwei Rezepte zur Auswahl. Beim ersten, dem »DIY-Ansatz«, werden ein Steckbrett und ein L293D-Chip verwendet. Beim zweiten Ansatz kommt ein fertiges H-Brücken-Modul von SparkFun zum Einsatz, das direkt mit dem Raspberry Pi verbunden wird.

Sowohl mit dem L293D als auch mit dem SparkFun-Modul können zwei Motoren ohne zusätzliche Hardware angesteuert werden.

Option 1: L293D-Chip und Steckbrett

Wenn Sie sich für die L293D-Variante entscheiden, benötigen Sie folgende Komponenten:

- 3-V- bis 12-V-Gleichstrommotor
- Steckbrett und Schaltdrähte (männlich/weiblich; siehe Anhang A, *Prototyping-Zubehör*, Seite 450)
- L293D-Chip (Anhang A, *Integrierte Schaltungen*, Seite 451)
- Stromversorgung mit einer für den Motor ausreichenden Leistung

Der Aufbau auf dem Steckbrett wird in Abbildung 11-8 gezeigt.

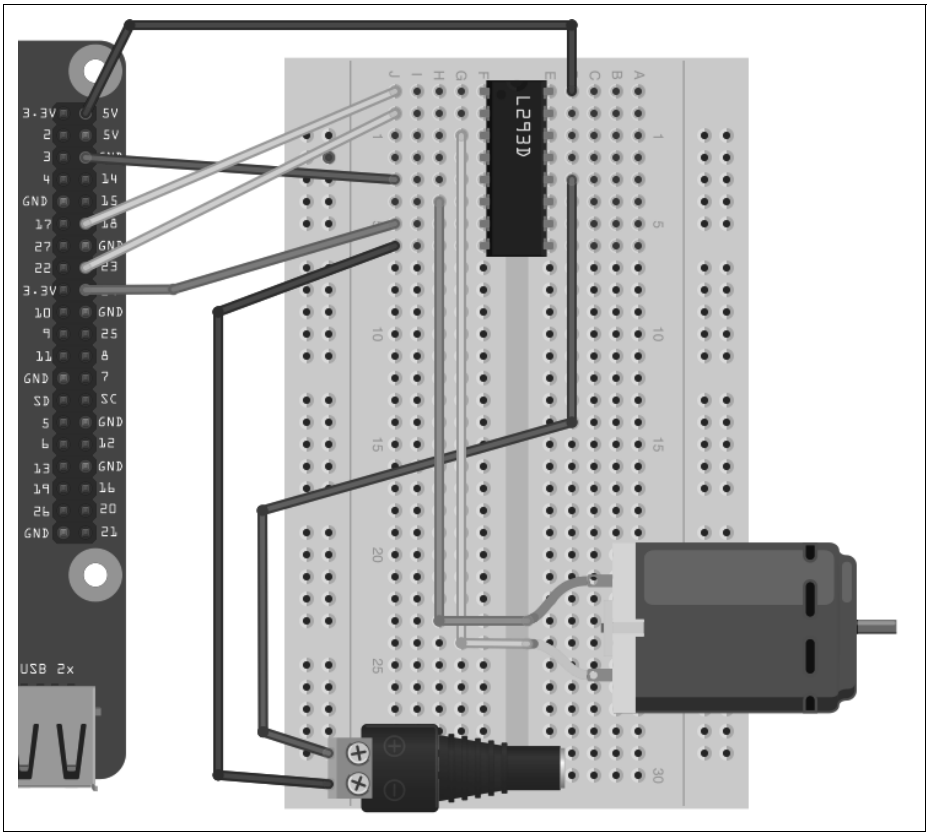


Abbildung 11-8: L293D-Chip zur Steuerung eines Motors

Stellen Sie sicher, dass Sie den Chip richtig herum anschließen. Er hat auf einer Seite eine kleine Einkerbung, die auf dem Steckbrett nach oben zeigen muss.

Option 2: Motor-Controller-Modul

Wenn Sie sich für einen SparkFun-Motor-Controller (oder ein vergleichbares Motor-Controller-Modul) entscheiden, benötigen Sie folgende Komponenten:

- 3-V- bis 12-V-Gleichstrommotor
- Schaltdrähte (weiblich/weiblich; Anhang A, *Prototyping-Zubehör*, Seite 450)
- Stiftleisten (Anhang A, *Vermischtes*, Seite 453)
- SparkFun-Motortreiber 1A Dual (Anhang A, *Module*, Seite 452)
- Stromversorgung mit einer für den Motor ausreichenden Leistung

Die Verdrahtung wird in Abbildung 11-9 gezeigt. Beachten Sie, dass der abgebildete Motor nur ein einfacher Gleichstrommotor ist. Wenn in einem Projekt Räder angetrieben werden sollen, würde man üblicherweise einen *Getriebemotor* verwen-

den, bei dem ein Motor und ein Getriebe kombiniert werden, um die Drehzahl zu reduzieren und das Drehmoment zu erhöhen.

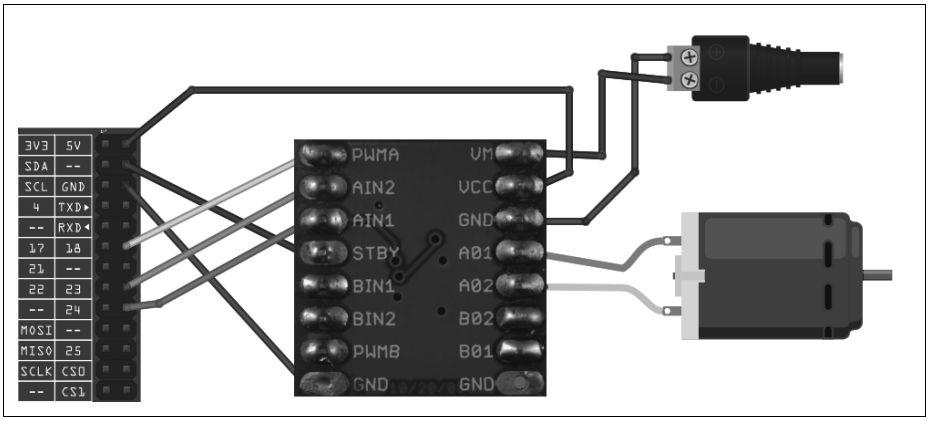


Abbildung 11-9: Verschaltung eines SparkFun-Motor-Controllers

Das Motor-Controller-Modul wird ohne Stiftleisten geliefert, die Sie selbst anbringen müssen, bevor Sie das Modul nutzen können.

Abbildung 11-10 zeigt die Verschaltung des Moduls mit einem kleinen Gleichstrom-Getriebemotor entsprechend der Schaltung aus Abbildung 11-9.

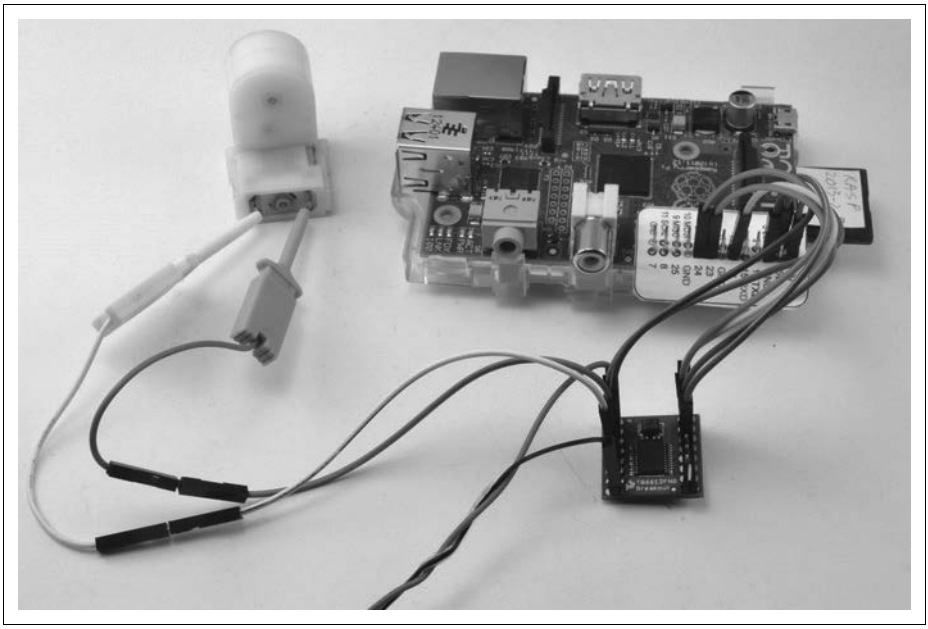


Abbildung 11-10: SparkFun-Motor-Controller mit Getriebemotor

Simon Monk: Raspberry-Pi-Kochbuch (ISBN Print: 978-3-96009-033-5), 2017

Software

Für welche Hardwarelösung Sie sich auch entscheiden, das Programm, mit dem Sie den Motor ausprobieren können, ist für beide identisch. Es ermöglicht die Eingabe des Buchstabens *f* (für forward, vorwärts) oder *r*, gefolgt von einer einzelnen Ziffer zwischen 0 und 9. Der Motor läuft dann entweder vor- oder rückwärts mit einer Geschwindigkeit zwischen 0 (Stopp) und 9 (Höchstgeschwindigkeit).

```
$ sudo python motor_control.py
Command, f/r 0..9, E.g. f5 :f5
Command, f/r 0..9, E.g. f5 :f1
Command, f/r 0..9, E.g. f5 :f2
Command, f/r 0..9, E.g. f5 :r2
```

Öffnen Sie einen Editor (nano oder IDLE) und kopieren Sie den folgenden Code. Wie alle Programmbeispiele aus diesem Buch können Sie auch dieses aus dem Codebereich von *http://www.oreilly.de/raspberrykochbuch* herunterladen (unter dem Namen *motor_control.py*). Das Programm nutzt die Kommandozeile, kann also auch über SSH ausgeführt werden.

Wenn Sie Python 3 nutzen, müssen Sie `raw_input` durch `input` ersetzen:

```
import RPi.GPIO as GPIO
import time

enable_pin = 18
in1_pin = 23
in2_pin =24
GPIO.setmode(GPIO.BCM)
GPIO.setup(enable_pin, GPIO.OUT)
GPIO.setup(in1_pin, GPIO.OUT)
GPIO.setup(in2_pin, GPIO.OUT)

pwm = GPIO.PWM(enable_pin, 500)
pwm.start(0)

def clockwise():
    GPIO.output(in1_pin, True)
    GPIO.output(in2_pin, False)

def counter_clockwise():
    GPIO.output(in1_pin, False)
    GPIO.output(in2_pin, True)

while True:
    cmd = raw_input("Command, f/r 0..9, E.g. f5 :")
    direction = cmd[0]
    if direction == "f":
        clockwise()
    else:
        counter_clockwise()
    speed = int(cmd[1]) * 10
    pwm.ChangeDutyCycle(speed)
```

Diskussion

Bevor wir uns ansehen, wie das Programm funktioniert, müssen wir zuerst verstehen, wie eine H-Brücke arbeitet.

Abbildung 11-11 zeigt, wie sie funktioniert. Im Diagramm werden Schalter anstelle von Transistoren verwendet. Durch die Umkehr der Polarität ändert die H-Brücke auch die Richtung, in der sich der Motor dreht.

In Abbildung 11-11 sind S1 und S4 geschlossen und S2 und S3 offen. Der Strom fließt von Anschluss A (positiv) über Anschluss B (negativ) durch den Motor. Wenn wir die Schalter so umlegen, dass S2/S3 geschlossen und S1/S4 offen sind, ist B positiv und A negativ und der Motor dreht sich in die andere Richtung.

Aber vielleicht ist Ihnen die Gefahr bei dieser Schaltung aufgefallen. Wenn S1 und S2 aus irgendeinem Grund beide gleichzeitig geschlossen sind, ist der positive Anschluss direkt mit dem negativen verbunden und es gibt einen Kurzschluss. Das Gleiche trifft zu, wenn S3 und S4 gleichzeitig geschlossen sind.

Sie können eine H-Brücke direkt über Transistoren aufbauen, doch der Einsatz eines H-Brücken-ICs wie der L293D-Chips ist wesentlich einfacher. Der Chip enthält darüber hinaus zwei H-Brücken, sodass Sie zwei Motoren steuern können.

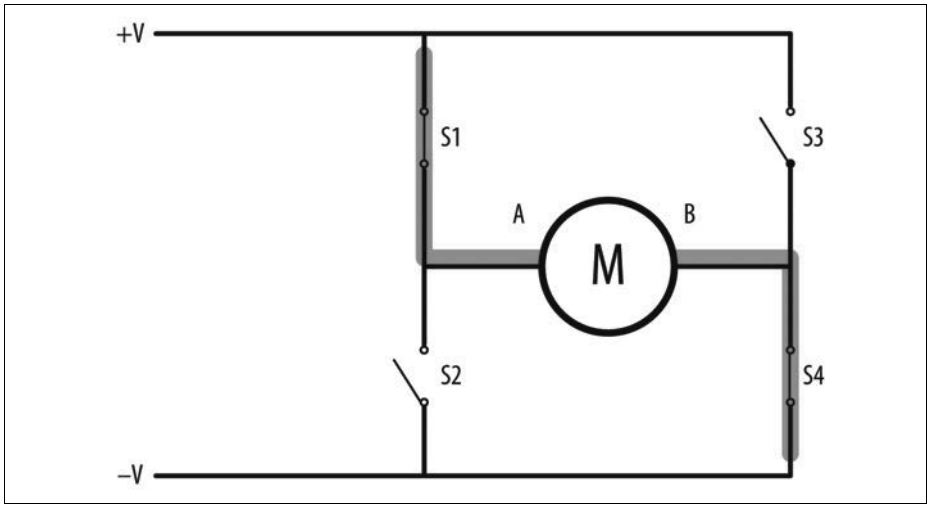


Abbildung 11-11: Eine H-Brücke

Der L293 nutzt drei Steuerpins für jeden der beiden Motorsteuerungskanäle. Der `Enable`-Pin aktiviert bzw. deaktiviert den Kanal als Ganzes. Im Beispielprogramm ist er mit einem PWM-Ausgang verbunden, der die Geschwindigkeit des Motors steuert. Die anderen beiden Pins (IN1 und IN2) steuern die Drehrichtung des Motors. Wie die beiden Pins genutzt werden, zeigen die Funktionen `clockwise` und `counter_clockwise`:

```
def clockwise():
    GPIO.output(in1_pin, True)
    GPIO.output(in2_pin, False)

def counter_clockwise():
    GPIO.output(in1_pin, False)
    GPIO.output(in2_pin, True)
```

Ist IN1 an und IN2 aus, dreht sich der Motor in die eine Richtung. Kehrt man die Zustände der beiden Pins um, dreht sich der Motor in die andere Richtung.

Statt eine L293D-Schaltung auf einem Steckbrett aufzubauen, können Sie bei eBay sehr kostengünstige Module erstehen. Diese Leiterplatten enthalten einen L293D und Schraubklemmen zum Anschluss der Motoren und Stiftleisten für die direkte Verbindung mit dem Raspberry Pi. Wenn Sie ein Controller-Modul für leistungsstarke Motoren benötigen, finden Sie Module, die nach dem gleichen Prinzip arbeiten, doch mit wesentlich höheren Strömen (20A und mehr) klarkommen. Polulu (*https://www.pololu.com*) bietet eine beeindruckende Auswahl solcher Motor-Controller-Boards an.

Siehe auch

Der Adafruit-Schrittmotor-HAT (Rezept 11.8) und das RasPiRobot-Board (Rezept 11.9) können ebenfalls genutzt werden, um Geschwindigkeit und Richtung eines Gleichstrommotors zu steuern.

Sehen Sie sich das L293D-Datenblatt (*http://bit.ly/18c4GKm*) und die Produktseite des SparkFun-Motor-Driver-Moduls (*http://bit.ly/ILHVkJ*)an.

Die Nutzung von Steckbrett und Schaltdrähten mit dem Raspberry Pi wird in Rezept 9.8 beschrieben.

11.6 Einen unipolaren Schrittmotor nutzen

Problem

Sie möchten einen unipolaren Schrittmotor mit 5 Anschlüssen mit dem Raspberry Pi steuern.

Lösung

Nutzen Sie einen ULN2803-Darlington-Treiber-Chip.

Bei den verfügbaren Motortechnologien liegen Schrittmotoren irgendwo zwischen Gleichstrom- und Servomotoren. Wie gewöhnliche Gleichstrommotoren können sie sich kontinuierlich drehen, gleichzeitig lassen sie sich aber auch sehr genau positionieren, indem man sie Schritt für Schritt in eine beliebige Richtung bewegt.

Für dieses Rezept benötigen Sie folgende Komponenten:

- 5-V-Schrittmotor, unipolar, mit 5 Pins (Anhang A, *Vermischtes*, Seite 453)
- ULN2803-Darlington-Treiber-IC (Anhang A, *Integrierte Schaltungen*, Seite 451)
- Steckbrett und Schaltdrähte (Anhang A, *Prototyping-Zubehör*, Seite 450)

Abbildung 11-12 zeigt die Verkabelung eines ULN2803. Beachten Sie, dass der integrierte Schaltkreis zwei solcher Motoren antreiben kann. Um einen zweiten Schrittmotor anzusteuern, müssen Sie vier weitere Pins des GPIO-Anschlusses mit den Pins 5 bis 8 des ULN2803 verbinden und den zweiten Motor mit den Pins 11 bis 14.

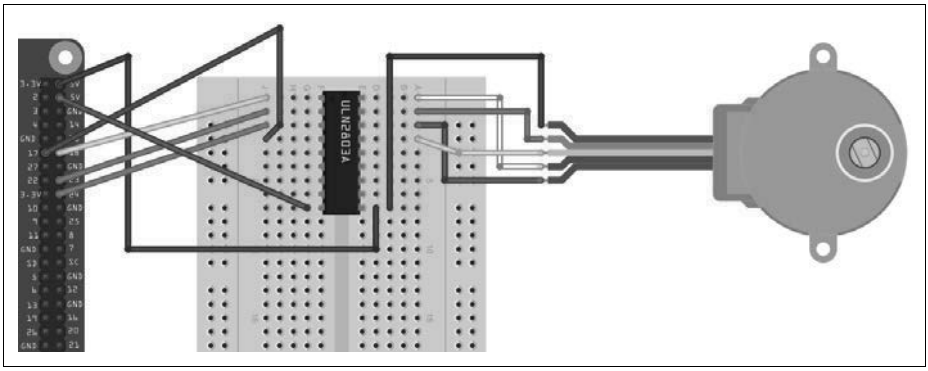


Abbildung 11-12: Einen unipolaren Schrittmotor mittels ULN2803 steuern

Die 5-V-Leitung des GPIO-Anschlusses sollte für einen kleinen Schrittmotor ausreichen. Kommt es beim Betrieb zu Problemen oder wird ein größerer Schrittmotor benötigt, müssen Sie eine separate Stromversorgung für den Motor verwenden (Pin 10 des ULN2803).

Öffnen Sie einen Editor (nano oder IDLE) und kopieren Sie den folgenden Code. Wie alle Programmbeispiele aus diesem Buch können Sie auch dieses aus dem Codebereich von *http://www.oreilly.de/raspberrykochbuch* herunterladen (unter dem Namen *stepper.py*). Das Programm nutzt die Kommandozeile, d.h., Sie können es über SSH ausführen.

Wenn Sie Python 3 nutzen, müssen Sie `raw_input` durch `input` ersetzen:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

coil_A_1_pin = 18
coil_A_2_pin = 23
coil_B_1_pin = 24
coil_B_2_pin = 17
```

```
GPIO.setup(coil_A_1_pin, GPIO.OUT)
GPIO.setup(coil_A_2_pin, GPIO.OUT)
GPIO.setup(coil_B_1_pin, GPIO.OUT)
GPIO.setup(coil_B_2_pin, GPIO.OUT)

forward_seq = ['1010', '0110', '0101', '1001']
reverse_seq = list(forward_seq) # to copy the list
reverse_seq.reverse()

def forward(delay, steps):
  for i in range(steps):
    for step in forward_seq:
      set_step(step)
      time.sleep(delay)

def backwards(delay, steps):
  for i in range(steps):
    for step in reverse_seq:
      set_step(step)
      time.sleep(delay)

def set_step(step):
  GPIO.output(coil_A_1_pin, step[0] == '1')
  GPIO.output(coil_A_2_pin, step[1] == '1')
  GPIO.output(coil_B_1_pin, step[2] == '1')
  GPIO.output(coil_B_2_pin, step[3] == '1')

while True:
  set_step('0000')
  delay = raw_input("Delay between steps (milliseconds)?")
  steps = raw_input("How many steps forward? ")
  forward(int(delay) / 1000.0, int(steps))
  set_step('0000')
  steps = raw_input("How many steps backwards? ")
  backwards(int(delay) / 1000.0, int(steps))
```

Wenn Sie das Programm ausführen, wird die Verzögerungszeit zwischen den Schritten angefordert. Der Wert muss hier 2 oder höher sein. Sie werden dann nach den Schritten in jede Richtung gefragt:

```
$ sudo python stepper.py
Delay between steps (milliseconds)?2
How many steps forward? 100
How many steps backwards? 100
Delay between steps (milliseconds)?10
How many steps forward? 50
How many steps backwards? 50
Delay between steps (milliseconds)?
```

Diskussion

Schrittmotoren nutzen einen gezahnten Rotor und Elektromagneten, um das Rad um jeweils einen *Schritt* zu bewegen (Abbildung 11-13). Beachten Sie, dass die Farben der Anschlüsse variieren.

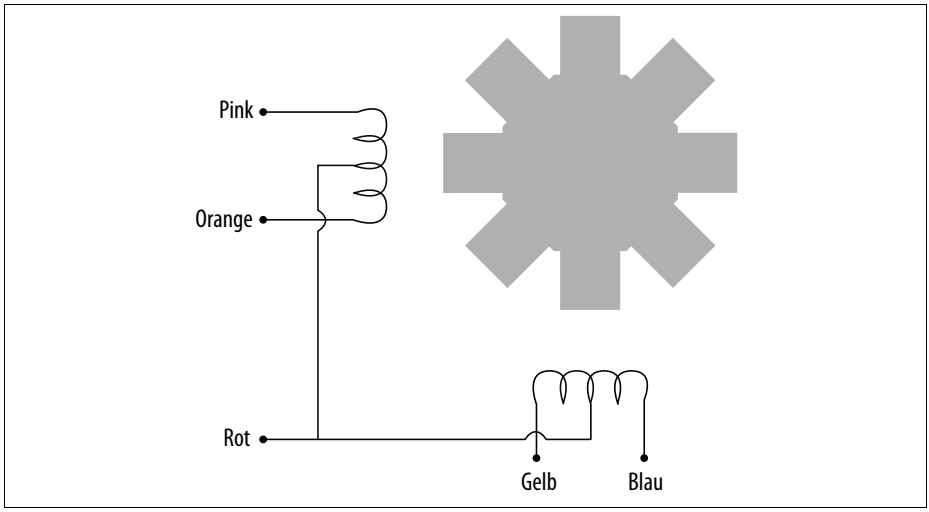


Abbildung 11-13: Ein Schrittmotor

Wenn die Spulen in einer bestimmten Reihenfolge eingeschaltet werden, dreht sich der Motor. Die Anzahl der Schritte, die ein Schrittmotor benötigt, um eine 360-Grad-Drehung durchzuführen, wird von der Anzahl der Zähne des Rotors bestimmt.

Im Beispielprogramm wird eine Liste von Strings für die Einschaltzustände genutzt, die für einen einzelnen Schritt notwendig sind:

```
forward_seq = ['1010', '0110', '0101', '1001']
```

Die Sequenz, die den Motor in entgegengesetzter Richtung laufen lässt, ist einfach die umgekehrte Abfolge der Vorwärtsbewegung.

Sie können die Funktionen `forward` und `backward` in Ihren Programmen nutzen, um den Motor schrittweise vor und zurück zu bewegen. Das erste Argument beider Funktionen ist die Verzögerung zwischen den einzelnen Schritten in Millisekunden. Der hier anzugebende Minimalwert hängt vom verwendeten Motor ab. Wenn er zu klein ist, dreht sich der Motor nicht. Üblicherweise reichen zwei Millisekunden aus. Der zweite Parameter gibt die Anzahl der gewünschten Schritte an.

```
def forward(delay, steps):
  for i in range(steps):
    for step in forward_seq:
      set_step(step)
      time.sleep(delay)
```

Bei der `forward`-Funktion werden zwei verschachtelte `for`-Scheifen genutzt. Die äußere geht die Anzahl der gewünschten Schritte durch, während die innere Schleife die Folge der notwendigen Motoraktionen durchläuft. Dabei wird für jede Aktion `set_Step` aufgerufen.

```
def set_step(step):
  GPIO.output(coil_A_1_pin, step[0] == '1')
  GPIO.output(coil_A_2_pin, step[1] == '1')
  GPIO.output(coil_B_1_pin, step[2] == '1')
  GPIO.output(coil_B_2_pin, step[3] == '1')
```

Die Funktion `set_step` setzt jeden der Steuerpins entsprechend dem im Argument übergebenen Muster auf High oder Low.

Die Hauptschleife setzt den Schritt zwischen der Vor- und Rückwärtsbewegung auf *0000*. Damit werden alle Ausgänge auf 0 gesetzt, wenn sich der Motor nicht dreht. Anderenfalls könnte eine der Spulen an sein und der Motor würde unnötig Strom ziehen.

Siehe auch

Das Ansteuern eines bipolaren Schrittmotors mit vier Anschlüssen wird in Rezept 11.7 erläutert.

Weitere Informationen zu Schrittmotoren, den unterschiedlichen Typen und zur Funktionsweise finden Sie auf Wikipedia. Dort wird auch das Aktivierungsmuster zur Ansteuerung des Motors mit einer schönen Animation erläutert.

Informationen zur Ansteuerung von Servomotoren finden Sie in Rezept 11.1. Die Steuerung von Gleichstrommotoren wird in den Rezepten Rezept 11.4 und Rezept 11.5 erläutert.

Die Nutzung von Steckbrett und Schaltdrähten mit dem Raspberry Pi wird in Rezept 9.8 beschrieben.

11.7 Einen bipolaren Schrittmotor nutzen

Problem

Sie möchten einen bipolaren Schrittmotor mit vier Anschlüssen mit einem Raspberry Pi steuern.

Lösung

Nutzen Sie einen L293D-H-Brücken-Treiber-Chip. Eine H-Brücke wird zur Ansteuerung eines bipolaren Schrittmotors benötigt, weil die Richtung des Stroms

durch die Windungen umgekehrt werden muss, ähnlich wie bei der Steuerung der Drehrichtung bei einem Gleichstrommotor (Rezept 11.5).

Für dieses Rezept benötigen Sie folgende Komponenten:

- 12-V-Schrittmotor, bipolar, mit 4 Pins (Anhang A, *Vermischtes*, Seite 453)
- L293D-H-Brücken-IC (Anhang A, *Integrierte Schaltungen*, Seite 451)
- Steckbrett und Schaltdrähte (Anhang A, *Prototyping-Zubehör*, Seite 450)

Der hier verwendete 12-V-Motor ist etwas größer als im vorangehenden Unipolar-Schrittmotor-Beispiel. Den Strom für den Motor liefert daher eine externe Stromversorgung und nicht der Raspberry Pi (Abbildung 11-14).

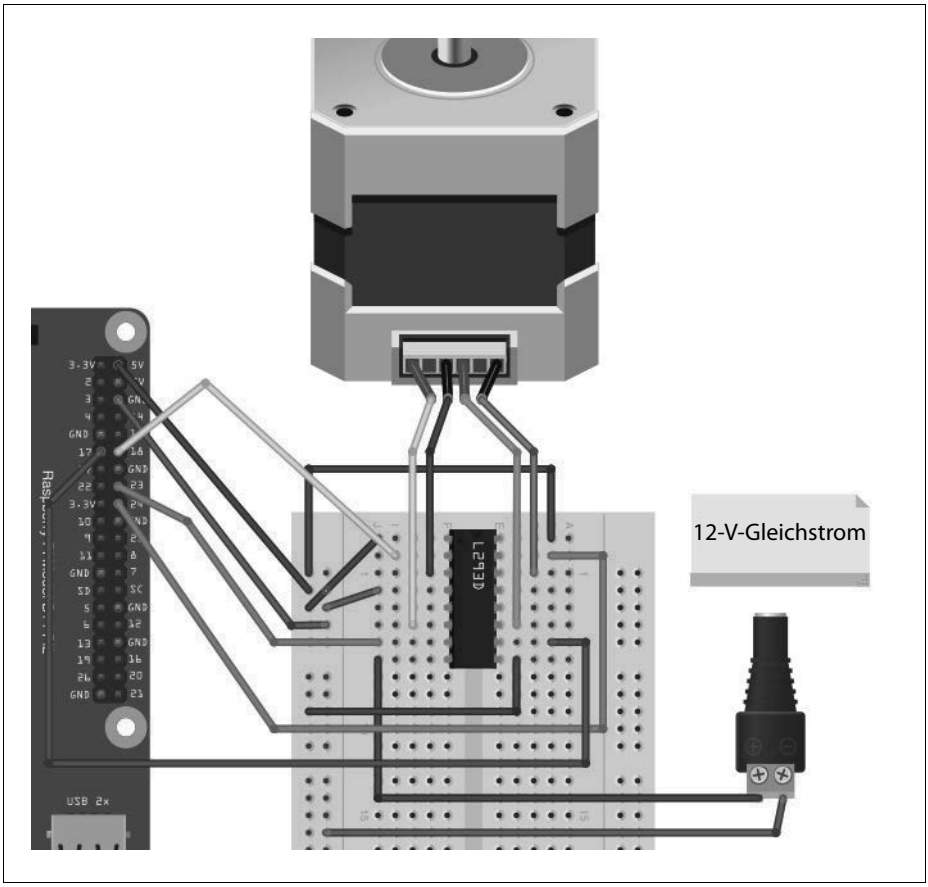


Abbildung 11-14: Einen L293D zur Steuerung eines bipolaren Schrittmotors nutzen

Simon Monk: Raspberry-Pi-Kochbuch (ISBN Print: 978-3-96009-033-5), 2017

Diskussion

Sie können genau das gleiche *stepper.py*-Programm nutzen, um diesen Schrittmotor zu steuern (Rezept 11.6). Beim Entwurf werden beide H-Brücken des L293D genutzt, d.h., Sie benötigen für jeden zu steuernden Motor einen dieser Chips.

Siehe auch

Die Steuerung eines unipolaren Schrittmotors mit fünf Anschlüssen wird in Rezept 11.6 erläutert.

Weitere Informationen zu Schrittmotoren, den unterschiedlichen Typen und zur Funktionsweise finden Sie auf Wikipedia. Dort wird auch das Aktivierungsmuster zur Ansteuerung des Motors mit einer schönen Animation erläutert.

Informationen zur Verwendung von Servomotoren finden Sie in Rezept 11.1; die Steuerung von Gleichstrommotoren wird in den Rezepten 11.4 und 11.5 erläutert.

Sie können einen Schrittmotor auch mit dem RasPiRobot-Board ansteuern (Rezept 11.9).

Die Nutzung von Steckbrett und Schaltdrähten mit dem Raspberry Pi wird in Rezept 9.8 beschrieben.

11.8 Einen bipolaren Schrittmotor mit einem Schrittmotor-HAT steuern

Problem

Sie möchten mehrere bipolare Schrittmotoren mit einem einzigen Interface-Board steuern.

Lösung

Verwenden Sie das Adafruit-Schrittmotor-HAT (Abbildung 11-5).

Dieses Board kann zwei bipolare Schrittmotoren ansteuern. Abbildung 11-15 zeigt das Board mit angeschlossenem bipolaren Schrittmotor. Eine Spule ist mit der Klemme M1 verbunden, die andere mit M2. Die Stromversorgung des Motors erfolgt separat über die Schraubklemmen rechts in Abbildung 11-15.

Dieses HAT nutzt I2C, d.h., Sie müssen I2C aktivieren (Rezept 9.3).

Um die Arbeit mit dem HAT zu erleichtern, hat Adafruit eine Python-Bibliothek entwickelt. Sie installieren sie mit den folgenden Befehlen:

```
$ git clone https://github.com/adafruit/Adafruit-Motor-HAT-Python-Library.git
$ sudo apt-get install python-dev
$ cd Adafruit-Motor-HAT-Python-Library/
$ sudo python setup.py install
```

Die Bibliothek enthält auch einige Beispiele. Wechseln Sie also in das entsprechende Verzeichnis und führen Sie das grundlegende Beispiel aus:

```
$ cd examples
$ sudo python StepperTest.py
```

I2C-Busse

Wenn Sie wie in Rezept 9.20 beschrieben ein eigenes HAT entwickelt und den I2C-Bus 0 aktiviert haben, dann müssen Sie diese Änderung in */boot/config.txt* rückgängig machen, da Adafruit den zu nutzenden I2C-Bus automatisch erkennt und den falschen wählt, wenn Bus 0 aktiv ist.

In */boot/config.txt* löschen Sie die folgende Zeile (oder kommentieren sie aus):

```
dtparam=i2c_vc=on
```

Danach starten Sie den Raspberry Pi neu.

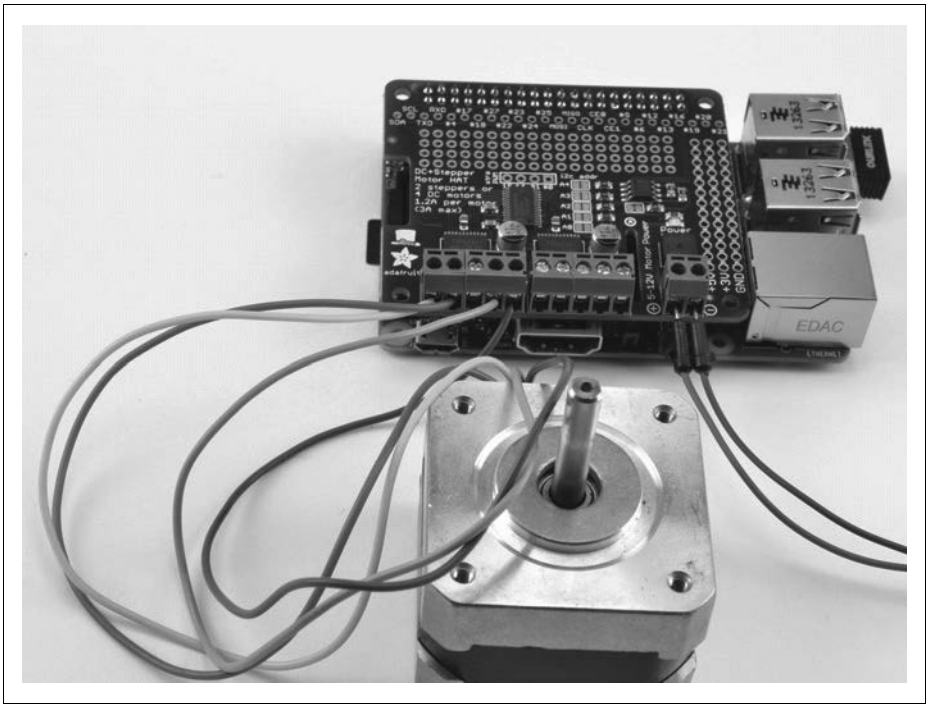


Abbildung 11-15: Adafruit-Schrittmotor-HAT zur Steuerung eines bipolaren Schrittmotors

Simon Monk: Raspberry-Pi-Kochbuch (ISBN Print: 978-3-96009-033-5), 2017

Diskussion

Wenn Sie das Programm ausführen, beginnt sich der Motor zu drehen und es werden vier verschiedene Schrittmodi durchlaufen. Sie können das nachfolgende, einfache Beispiel nutzen, um Ihren Schrittmotor zu steuern.

Öffnen Sie einen Editor (nano oder IDLE) und kopieren Sie den folgenden Code. Wie alle Programmbeispiele aus diesem Buch können Sie auch dieses aus dem Codebereich von *http://www.oreilly.de/raspberrykochbuch* herunterladen (unter dem Namen *stepper_hat.py*). Das Programm nutzt die Kommandozeile und kann daher über SSH ausgeführt werden.

```
from Adafruit_MotorHAT import Adafruit_MotorHAT
import time

HAT = Adafruit_MotorHAT
stepper_hat = Adafruit_MotorHAT()

stepper = stepper_hat.getStepper(200, 1) # 200 steps/rev, port 1 (M1, M2)

try:
     while True:
        speed = input("Enter stepper speed (rpm) ")
        stepper.setSpeed(speed)
        steps_forward = input("Steps forward ")
        stepper.step(steps_forward, HAT.FORWARD,  HAT.SINGLE)
        steps_forward = input("Steps reverse ")
        stepper.step(steps_forward, HAT.BACKWARD,  HAT.SINGLE)

finally:
     print("cleaning up")
     stepper_hat.getMotor(1).run(HAT.RELEASE)
```

Beim Import des `Adafruit_MotorHAT`-Moduls wird diesem der Name HAT zugewiesen, um den Quellcode etwas kompakter zu fassen.

Der `try/finally`-Block stellt sicher, das die Spannung an den Spulen unterbrochen wird, sobald das Programm mit Ctrl-C beendet wird.

Siehe auch

Eine Diskussion zum HAT-Standard und wie man ein eigenes HAT entwickelt, finden Sie in Rezept 9.20.

Weitere Informationen zu diesem HAT und der dazugehörigen Bibliothek finden Sie auf *https:// learn.adafruit.com/adafruit-dc-and-stepper-motor-hat-for-raspberry-pi/*.

Wie man einen Schrittmotor per L293D steuert, erläutert Rezept 11.7. Wie das mit einem RasPiRobot gemacht wird, erklärt Rezept 11.9.

11.9 Ein RasPiRobot-Board zur Steuerung eines bipolaren Schrittmotors nutzen

Sehen Sie sich das dazugehörige Video auf *http://razzpisampler.oreilly.com* an.

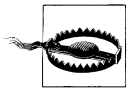
Problem

Sie möchten einen bipolaren Schrittmotor steuern und sowohl Motor als auch den Raspberry Pi mit nur einer Quelle mit Strom versorgen.

Lösung

Verwenden Sie ein RasPiRobot-Board V3.

Das RasPiRobot-Board nutzt die Stromversorgung von seinem DC-Anschluss, um den Motor zu versorgen, und regelt die gleiche Quelle auf 5 V herunter, um den Raspberry Pi zu versorgen. In diesem Fall liefert also die 12-V-Quelle den Strom sowohl für den Schrittmotor als auch für den Raspberry Pi.



Bei älteren Versionen des RasPiRobot-Boards (Version 1 oder 2) darf der Raspberry Pi *nicht* über den USB-Anschluss versorgt werden, wenn das RasPiRobot-Board eingeschaltet ist.

Bei Version 3 des Boards ist das kein Problem.

Verbinden Sie den Schrittmotor und die Stromversorgung mit dem RasPiRobot-Board, wie in Abbildung 11-16 gezeigt wird. Die Farben der Anschlüsse des 12-V-Schrittmotors von Adafruit sind (vom DC-Anschluss ausgehend von links nach rechts) gelb, rot, grau und grün.

Bevor Sie das Programm ausführen können, müssen Sie die Bibliothek für den RasPiRobot V3 mit den folgenden Befehlen installieren:

```
$ git clone https://github.com/simonmonk/raspirobotboard3.git
$ cd raspirobotboard3/python
$ sudo python setup.py install
```

Öffnen Sie einen Editor (nano oder IDLE) und kopieren Sie den folgenden Code. Wie alle Programmbeispiele aus diesem Buch können Sie auch dieses aus dem Codebereich von *http://www.oreilly.de/raspberrykochbuch* herunterladen (unter dem Namen *stepper_rrb.py*). Das Programm nutzt die Kommandozeile und kann daher über SSH ausgeführt werden.

Simon Monk: Raspberry-Pi-Kochbuch (ISBN Print: 978-3-96009-033-5), 2017

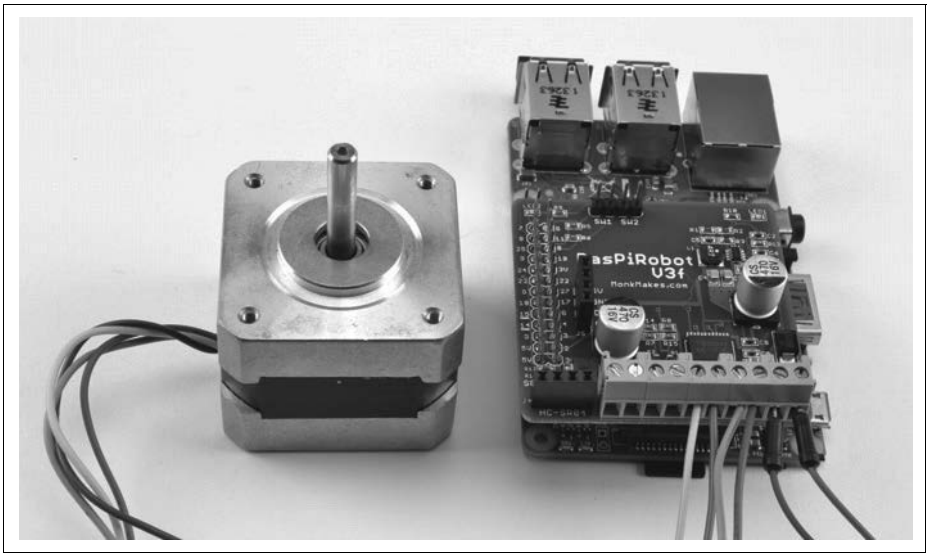


Abbildung 11-16: Ein RasPiRobot-Board zur Steuerung eines bipolaren Schrittmotors nutzen

Wenn Sie Python 3 nutzen, müssen Sie `raw_input` durch `input` ersetzen:

```
from rrb3 import *
import time

rr = RRB3(12.0, 12.0) # battery, motor

try:
    while True:
        delay = raw_input("Delay between steps (milliseconds)?")
        steps = raw_input("How many steps forward? ")
        rr.step_forward(int(delay) / 1000.0, int(steps))
        steps = raw_input("How many steps backwards? ")
        rr.step_reverse(int(delay) / 1000.0, int(steps))

finally:
GPIO.cleanup()
```

Diskussion

Sie werden feststellen, dass es einen Minimalwert für »`Delay between Steps`«, also für die Verzögerung zwischen zwei Schritten, gibt. Unterschreiten Sie diesen Wert, wird der Motor sich nicht bewegen, sondern nur zucken.

Siehe auch

Mehr Informationen zum RasPiRobot-Board und zu den dazugehörigen Projekten finden Sie auf der RasPiRobot-Website (*http://www.raspirobot.com*).

Wie sich ein Schrittmotor mit einem L293D auf einem Steckbrett betreiben lässt, wird in Rezept 11.7 erläutert.

11.10 Einen einfachen Robot-Rover bauen

Problem

Sie möchten einen Raspberry Pi als Controller für einen einfachen fahrenden Roboter nutzen.

Lösung

Nutzen Sie ein RasPiRobot-Board V3 als Schnittstelle zum Raspberry Pi, um zwei Motoren zu steuern, sowie ein Roboter-Chassis-Kit wie das Magician Chassis.

Für dieses Rezept benötigen Sie folgende Komponenten:

- RasPiRobot-Board V3 (Anhang A, *Module*, Seite 452)
- Magician Chassis mit Getriebemotoren (Anhang A, *Vermischtes*, Seite 453)
- 6xAA-Batteriehalter (Anhang A, *Vermischtes*, Seite 453)
- USB-WLAN-Adapter

Einen vollständigen Baukasten für diesen Roboter können Sie als MonkMakes RasPiRobot Rover Kit kaufen.

Im ersten Schritt müssen Sie das Magician Chassis zusammenbauen. Zum Chassis gehört ein Batteriehalter für vier AA-Batterien, doch um den Raspberry Pi mit zu versorgen, benötigen Sie einen solchen für sechs Batterien. Bei der entsprechenden Aufbauanweisung des Magician Chassis verwenden Sie also den 6xAA-Batteriehalter.

Verkabeln Sie es so, wie es in Abbildung 11-17 gezeigt wird.

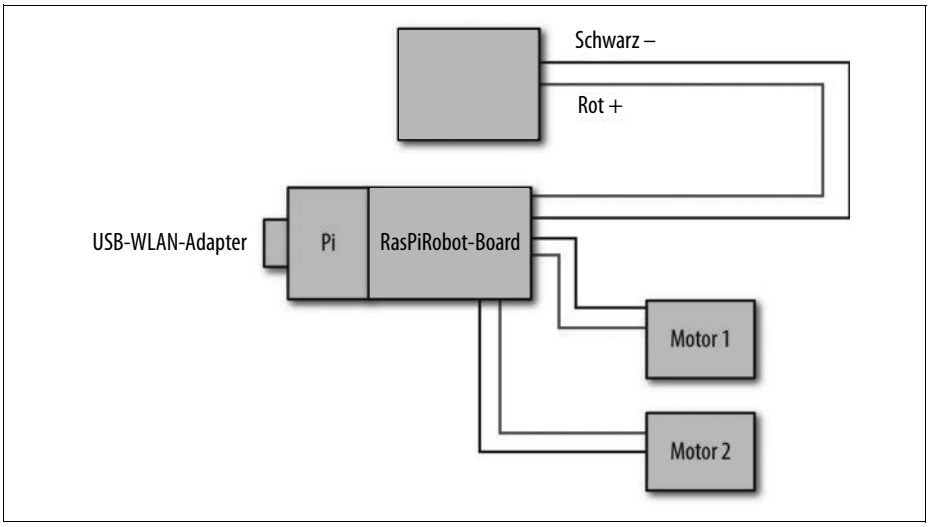


Abbildung 11-17: Verkabelung des fahrenden Roboters

Das Batteriepaket versorgt das RasPiRobot-Board, das wiederum den Raspberry Pi mit 5 V versorgt. Es wird also nur eine Stromversorgung benötigt.

Der fertige Rover sollte in etwa so wie in Abbildung 11-18 aussehen.

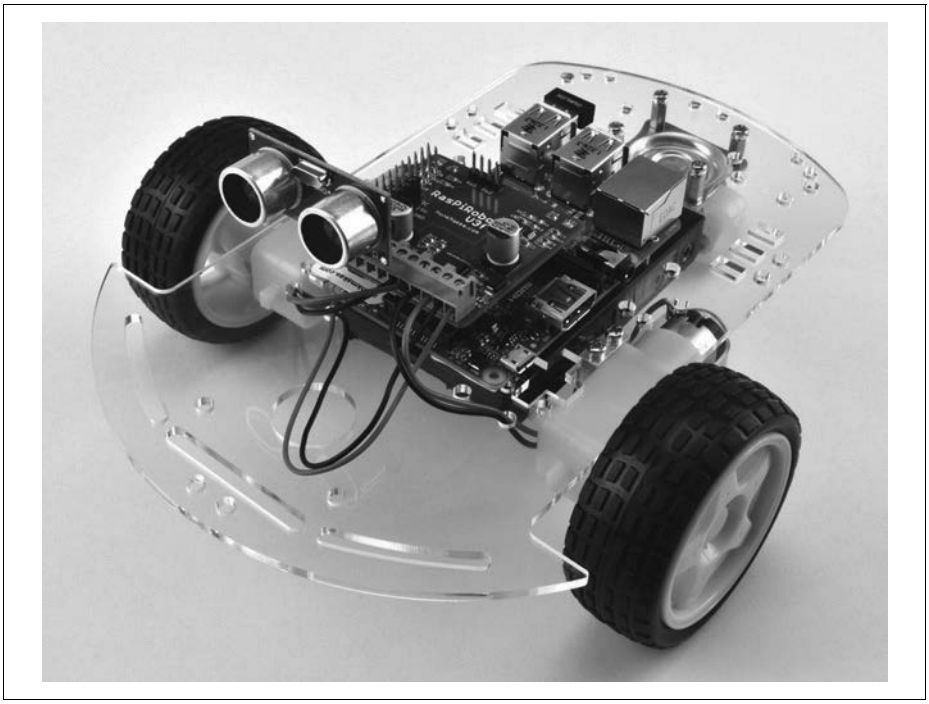


Abbildung 11-18: Der fertige Roboter

Um den Roboter zu bewegen, nutzen wir ein Programm, das es uns erlaubt, den Rover mit den Tasten eines Laptops (oder anderen Computers) zu steuern, der über SSH mit dem Raspberry Pi verbunden ist. Falls noch nicht geschehen, richten Sie WLAN (Rezept 2.5) und SSH (Rezept 2.7 auf Ihrem Raspberry Pi ein.

Öffnen Sie einen Editor (nano oder IDLE) und kopieren Sie den folgenden Code. Wie alle Programmbeispiele aus diesem Buch können Sie auch dieses aus dem Codebereich von *http://www.oreilly.de/raspberrykochbuch* herunterladen (unter dem Namen *rover.py*).

```
from rrb3 import *
import sys
import tty
import termios

rr = RRB3(9.0, 6.0) # battery, motor

UP=0
DOWN=1
RIGHT = 2
LEFT=3
print("Use the arrow keys to move the robot")
print("Press Ctrl-C to quit the program")

# Über diese Funktionen liest das Programm die Tastatur aus
def readchar():
  fd = sys.stdin.fileno()
  old_settings = termios.tcgetattr(fd)
  try:
        tty.setraw(sys.stdin.fileno())
        ch = sys.stdin.read(1)
  finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
  if ch == '0x03':
    raise KeyboardInterrupt
  return ch

def readkey(getchar_fn=None):
  getchar = getchar_fn or readchar
  c1 = getchar()
  if ord(c1) != 0x1b:
    return c1 c2 = getchar()
  if ord(c2) != 0x5b: return c1
    c3 = getchar()
  return ord(c3) - 65 # 0=nach oben, 1=nach unten, 2=rechts, 3=Links Pfeiltasten

# Steuert die Bewegung und gibt sie auf dem Bildschirm aus
try:
  while True:
    keyp = readkey()
    if keyp == UP:
            rr.forward(1)
            print 'forward'
```

```
        elif keyp == DOWN:
                rr.reverse(1)
                print 'backward'
        elif keyp == RIGHT:
                rr.right(1)
                print 'clockwise'
        elif keyp == LEFT:
                rr.left(1)
                print 'anti clockwise'
        elif ord(keyp) == 3:
                break

    except KeyboardInterrupt:
        GPIO.cleanup()
```

Um Tastatureingaben abfangen zu können, nutzt dieses Programm die *termios*-Bibliothek sowie die beiden Funktionen *readchar* und *readkey*.

Nach dem Import der Bibliotheken wird eine neue Instanz von RRB3 erzeugt. Als Parameter werden dabei die Batterie- und die Motorspannung übergeben (in diesem Fall 9V und 6V). Wenn Sie Motoren mit anderer Spannung verwenden, müssen Sie den zweiten Parameter entsprechend anpassen.

Die Hauptschleife überprüft, ob eine Taste gedrückt wurde, und sendet die entsprechenden Befehle für vorwärts, rückwärts, links und rechts an die RBB3-Bibliothek.

Diskussion

Sie können den Rover interessanter gestalten, indem Sie zusätzliche Peripheriegeräte hinzufügen. Sie könnten beispielsweise eine Webcam anschließen und ein Webstreaming aufsetzen, um aus Ihrem Rover eine fahrende Spycam zu machen (Rezept 4.4).

Die RBB3-Bibliothek unterstützt auch Entfernungsmesser vom Typ HC-SR04, die in einen Sockel auf dem RasPiRobot-Board V3 eingesteckt werden können. Auf diese Weise können Sie Hindernisse erkennen. Entsprechende Softwarebeispiele finden Sie in der RRB3-Bibliothek.

Siehe auch

Mehr Informationen zum RasPiRobot-Board und der RRB3-Bibliothek finden Sie auf GitHub (*https://github.com/simonmonk/raspirobotboard3*).