Leseprobe

Elektronikgrundlagen gemischt mit Bastelspaß: So spannend kann Lernen sein! In dieser Leseprobe erhalten Sie einen Überblick über das benötigte Zubehör, um die Buchprojekte mitzumachen. Außerdem lernen Sie, wie den Raspberry Pi einrichten und einen PWM-Controller erzeugen.

Leseprobe weiterempfehlen

Daniel Kampert, Christoph Scherbeck

Elektronik verstehen mit Raspberry Pi

361 Seiten, broschiert, in Farbe, Februar 2017
29,90 Euro, ISBN 978-3-8362-2869-5

www.rheinwerk-verlag.de/3602

1.7 Was benötige ich alles?

Bevor Sie loslegen können, verschaffen wir uns erst einmal einen kurzen Überblick über das benötigte Equipment.

1.7.1 Ein Raspberry Pi inklusive Zubehör

Dass Sie einen Raspberry Pi (siehe Abbildung 1.39) benötigen, dürfte für Sie bestimmt keine Überraschung sein. Sie benötigen außerdem ein passendes Netzteil und eine Micro-SD-Karte. Für die Arbeit mit diesem Buch benötigen Sie den Raspberry Pi mit einem Betriebssystem Ihrer Wahl (empfehlenswert ist die aktuelle Version des Betriebssystems *Raspbian*).

Abbildung 1.39 Der Raspberry Pi in der Version 3 für 38 EUR von Adafruit

Welche Version des Raspberry Pi Sie verwenden, bleibt Ihnen überlassen, allerdings sollten Sie nach Möglichkeit nicht den Raspberry Pi Zero verwenden. Der Zero ist wegen der fehlenden Schnittstellen für Anfänger nicht ganz so geeignet.

1.7.2 Ein Multimeter

Gerade wenn Sie Schaltungen aufbauen und kontrollieren wollen, ist es ratsam, ein Multimeter zur Hand zu haben. Da sich die Multimeter in ihren Grundfunktionen nicht stark unterscheiden, reicht für den Anfang ein preiswertes Gerät vollkommen aus. Bei einem Händler wie Reichelt Elektronik sind Multimeter bereits ab 10 EUR erhältlich.

Diese Multimeter sind aber schon von ihren Funktionen her eingeschränkt und unserer Meinung nach nicht empfehlenswert. Wenn Sie bereit sind, mindestens 30 EUR auszugeben, bekommen Sie ein anständiges Multimeter, das von den Funktionen her gut ausgestattet ist und erst einmal keine Wünsche offenlässt (siehe Abbildung 1.40).

Abbildung 1.40 Ein preiswertes Einsteigermultimeter von Adafruit für etwa 30 EUR

1.7.3 Externe Spannungsversorgung

Der Raspberry Pi eignet sich nur für kleine Schaltungen als Stromquelle. Bei größeren Spannungen und/oder Strömen kommt man um eine externe Spannungsversorgung nicht herum. Als externe Spannungsversorgung können Sie z. B. Batterien oder Netzgeräte von Smartphones oder alten Routern etc. verwenden (siehe Abbildung 1.41).

Abbildung 1.41 Ein 12-V-Steckernetzteil eines alten Routers

Den runden Stecker am Steckernetzteil können Sie z. B. einfach abschneiden und einen anderen Stecker anlöten, um das Steckernetzteil mit Ihrem Steckbrett oder Ähnlichem zu verbinden.

Diese Netzgeräte bzw. Batterien haben allerdings den Nachteil, dass sie meistens nur eine feste Spannung ausgeben und je nach Netzteil vom maximalen Strom her begrenzt sind.

Ein einstellbares Labornetzteil (siehe Abbildung 1.42) bietet Ihnen mehr Freiheiten, ist aber auch deutlich teurer. Solche Netzteile besitzen eine Kurzschlusssicherung. Das heißt, sie schalten ab, wenn der entnommene Strom zu groß wird, und die Ausgangsspannung kann in einem bestimmten Rahmen (z. B. 0–12 V) eingestellt werden.

Abbildung 1.42 Ein einstellbares Labornetzteil von Sparkfun

Das in Abbildung 1.42 gezeigte Netzteil bietet den Vorteil, dass der maximale Ausgangsstrom im Bereich von 0 bis 5 A und die maximale Ausgangsspannung im Bereich von 1 bis 30 V eingestellt werden können, wodurch Sie sehr flexibel sind.

1.7.4 Messleitungen

Um ein Labornetzteil oder ein Multimeter flexibel nutzen zu können, benötigen Sie in der Regel ein paar zusätzliche Kabel, um z. B. die Schaltung mit dem Netzteil zu verbinden.

Solche Messleitungen können Sie entweder selbst anfertigen (hierfür brauchen Sie allerdings einen Lötkolben, ein bisschen Löterfahrung und einen Schraubstock) oder als fertiges Set kaufen.

Abbildung 1.43 Messleitungen für z. B. Labornetzteile von Sparkfun für 3,40 EUR

Bei den Messleitungen aus Abbildung 1.43 handelt es sich um Leitungen mit einem 4-mm-Stift, der für eine 4-mm-Buchse verwendet wird, wie sie in jedem Multimeter oder Labornetzteil verbaut ist.

Um die Leitungen verwenden zu können, werden unter anderem Prüfspitzen oder Klemmen benötigt (siehe Abbildung 1.44). Sie können ebenfalls separat bestellt werden.

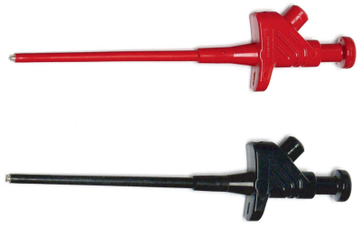


Abbildung 1.44 Passende Klemmen für einen 4-mm-Stift

1.7.5 Seitenschneider

Wenn Sie Schaltungen aufbauen möchten, benötigen Sie einen Seitenschneider, um z. B. Leitungen oder Anschlussbeinchen zu kürzen. Auch hier gibt es im Fachhandel eine große Auswahl an Seitenschneidern in verschiedenen Größen.

Zum Basteln reicht ein kleiner Seitenschneider für unter 20 EUR locker aus (siehe Abbildung 1.45).

Abbildung 1.45 Ein kleiner Elektronikerseitenschneider von Adafruit für 18 EUR

1.7.6 Steckbrett und Drahtbrücken

Ein Steckbrett (siehe Abbildung 1.46) ist eine ideale Möglichkeit, um Schaltungen schnell und einfach aufzubauen und zu testen.

Abbildung 1.46 Die große Variante des Steckbretts von Adafruit für 19 EUR

Ein Steckbrett erspart Ihnen lästige Lötarbeiten, und Sie können eventuell auftretende Fehler schnell beheben, da Sie die Bauteile ganz einfach umstecken können. Ein weiterer Vorteil ist, dass die Bauelemente für weitere Schaltungen wiederverwendet werden können. Solche Steckbretter gibt es in verschiedenen Größen. Welche Größe Sie kaufen, bleibt Ihnen überlassen.

Für die elektrischen Verbindungen auf dem Steckbrett benötigen Sie unbedingt sogenannte Drahtbrücken (siehe Abbildung 1.47), meistens die mit zwei Steckern (*male-male*), einige wenige auch mit Stecker und Buchse (*male-female*).

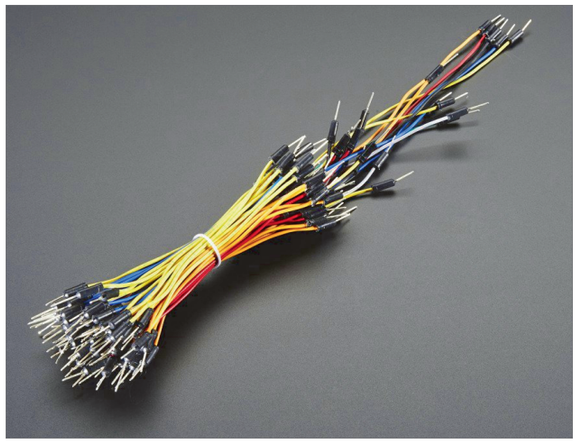


Abbildung 1.47 Notwendige Drahtbrücken für das Steckbrett, ebenfalls von Adafruit

1.7.7 Raspberry-Pi-Adapter für ein Steckbrett

Für Ihre ersten Gehversuche im Bereich der Elektronik wäre es äußerst praktisch, wenn Sie Ihren Raspberry Pi mit dem Steckbrett verbinden könnten. Speziell dafür gibt es diverse Adapterplatinen, die unter anderem bei *www.watterott.com* für kleines Geld bestellt werden können (siehe Abbildung 1.48).

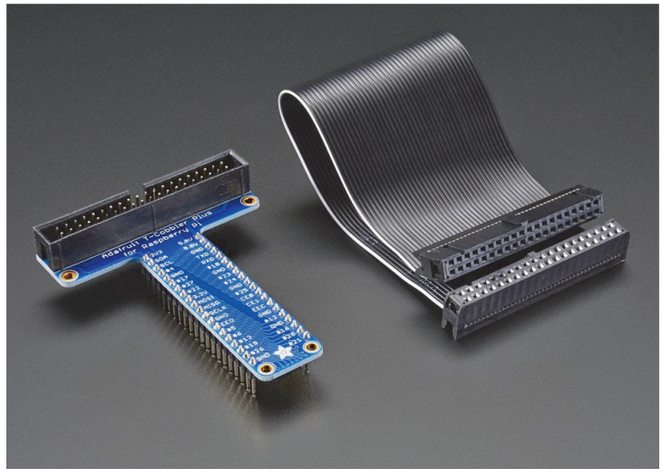


Abbildung 1.48 Raspberry-Pi-Steckbrettadapter von Adafruit

Auf diesen Adaptern ist nebenbei bemerkt noch die Pinbelegung aufgedruckt, sodass Sie diese immer griffbereit haben. Sie müssen bei der Bestellung nur aufpassen, dass Sie den richtigen Adapter für Ihre Raspberry-Pi-Version bestellen.

1.7.8 Lötkolben und Zubehör

Wenn Sie Ihre entwickelte und getestete Schaltung auf einer Leiterkarte aufbauen möchten, um die Schaltung z. B. in ein Gehäuse zu integrieren, werden Sie um einen Lötkolben nicht herumkommen.

Bei der Auswahl eines geeigneten Lötkolbens haben Sie die Wahl zwischen ungeregelten Lötkolben, die permanent eine feste Temperatur erreichen, oder einer geregelten Lötstation (siehe Abbildung 1.49), bei der die Temperatur variabel eingestellt werden kann.

Abbildung 1.49 Produktfoto einer für den Hobbybereich ausreichenden Lötstation für 72 EUR von Weller

Wir empfehlen, eine einstellbare Lötstation zu kaufen, da beim Löten, je nach Größe der Lötfläche, unterschiedliche Temperaturen verwendet werden müssen. Die teureren Lötstationen bieten etwas mehr Komfort beim Arbeiten. Hinzu kommen noch Kosten für Zubehör wie Lötzinn etc., sodass eine komplette Lötausrüstung gut über 100 EUR kosten kann.

Kapitel 2

Einrichtung

Auch wenn Sie mit diesem Buch in erster Linie etwas über Elektronik lernen möchten, müssen Sie sich jetzt mit den Grundlagen der Einrichtung des Raspberry Pi befassen. Im weiteren Verlauf des Buches setzen wir diese Kenntnisse voraus. Lesen Sie in diesem Kapitel unter anderem, wie Sie den Raspberry Pi einrichten, mit Ihrem Heimnetzwerk verbinden, eine SSH-Verbindung herstellen und Updates durchführen.

2.1 Installation

Wenn Sie den kleinen Mini-PC vor sich liegen haben, müssen Sie ihn startklar machen. Dazu benötigen Sie mindestens eine 8-GB-Micro-SD-Karte, ein Micro-USB-Kabel samt Netzteil, das mindestens 2 A Strom liefern kann, und idealerweise eine Tastatur samt Maus und einen HMDI-fähigen Bildschirm oder Fernseher.

Das Betriebssystem finden Sie auf der Raspberry-Pi-Webseite *www.raspberrypi.org* zum Download. Die heruntergeladene Image-Datei können Sie dann mit Tools wie Win32-DiskImager (Windows) oder Apple PiBaker (macOS) auf die SD-Karte schreiben.

Wir empfehlen Ihnen an dieser Stelle jedoch das Tool *PiBakery* für Windows, macOS und Linux. Das kleine Programm kann Raspian in der Full- oder in der Lite-Version installieren und erfordert nicht das vorige Herunterladen der Image-Datei, da diese in dem Tool integriert ist. Die Besonderheit an PiBakery ist, dass direkt bei der Installation des Images diverse Einstellungen mitgegeben werden können. So können Sie z. B. bei der Installation direkt die WLAN-Konfiguration vornehmen, Programme installieren und grundlegende Einstellungen vornehmen (siehe Abbildung 2.1).

Wählen Sie nach einem Klick auf Write die zuvor eingesteckte SD-Karte aus, und der Übertragungsvorgang startet.

Sobald der Vorgang abgeschlossen ist, haben Sie bereits ein funktionierendes Betriebssystem auf der SD-Karte und können diese in den Micro-SD-Slot Ihres Raspberry Pi stecken.

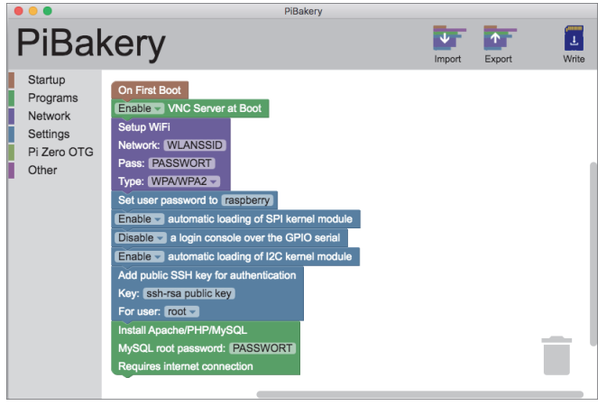


Abbildung 2.1 Bereits vor dem Überspielen der Image-Datei können Sie in PiBakery dem Betriebsystem zahlreiche Einstellungen mitgeben.

Schalten Sie den Raspberry Pi nun ein, indem Sie das Micro-USB-Kabel in den dafür vorgesehenen Port stecken. Sie sollten nun ebenfalls eine Tastatur per USB und einen Bildschirm per HDMI angeschlossen haben. Melden Sie sich nach dem Bootvorgang mit dem Nutzer `pi` und dem Passwort `raspberry` an. Beachten Sie, dass das Tastaturlayout möglicherweise noch auf eine amerikanische Tastatur eingestellt ist (Y und Z sind dann vertauscht).

Die Komplettlösung NOOBS

Auf der Homepage der Raspberry Pi Foundation finden Sie außerdem das sogenannte NOOBS. Dabei handelt es sich um ein fertiges Image, das eine Art Installationsassistenten mit an Bord hat. Damit können Sie jedes andere verfügbare Betriebssystem kinderleicht auf dem Raspberry Pi installieren. Wir empfehlen NOOBS trotzdem nicht uneingeschränkt, da es bei jedem Einschalten eine Wiederherstellungsoption anbietet, die eine gewisse Sicherheit suggeriert, in Wirklichkeit aber alle Daten und installierten Programme löscht und nur die Urversion von Raspbian wiederherstellt.

2.1.1 Einrichtung per raspi-config

Starten Sie den Raspberry Pi nun, indem Sie das Micro-USB-Kabel in die passende Buchse stecken. Nach einer kurzen Bootphase werden Sie gebeten, sich anzumelden. Das tun Sie bei Raspbian standardmäßig mit dem Benutzernamen `pi` und dem Passwort `raspberry`.

Rufen Sie nun mit folgendem Konsolenbefehl den Einrichtungsassistenten auf:

```
sudo raspi-config
```

Wir werden nun alle Einstellungen vornehmen, die Sie im Verlauf dieses Buchs benötigen werden. Sie können `raspi-config` aber auch jederzeit erneut aufrufen, um die Einstellungen nachträglich vorzunehmen.

Um den vollen Speicherplatz Ihrer SD-Karte nutzen zu können, sollten Sie als Erstes den Menüpunkt 1 Expand Filesystem auswählen (siehe Abbildung 2.2).

Die Standard-Anmeldedaten sind in jeder frischen Raspbian-Installation gleich. Mit dem Nutzer `pi` und dem Passwort `raspberry` kann sich jeder Bediener im Betriebssystem anmelden. Da dies also kein Geheimnis mehr ist, sollten Sie das Passwort ändern. Das erledigen Sie mit der Option 2 Change User Password.

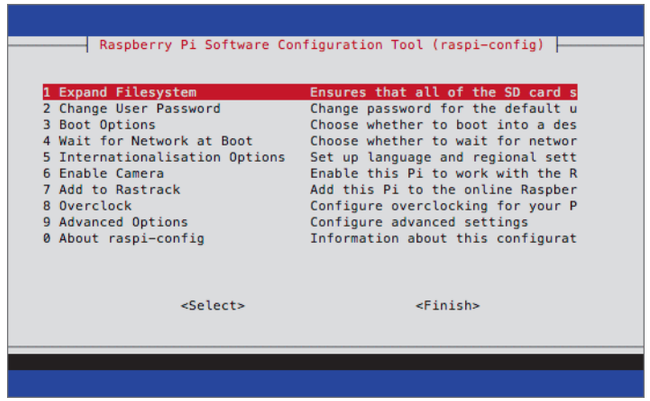


Abbildung 2.2 Das Einstellungsmenü »raspi-config« erlaubt es, viele grundlegende Einstellungen vorzunehmen.

Der Menüpunkt 3 Boot Options bietet die Wahl zwischen dem Boot in die Konsole mit automatischer Anmeldung sowie den automatischen Start der grafischen Oberfläche. Empfehlung: Lassen Sie die Einstellung unverändert.

Über Punkt 4 Wait for Network at Boot können Sie festlegen, ob der Raspberry Pi auf die erfolgreiche Anmeldung im Netzwerk wartet, was etwas länger dauert, oder den Bootvorgang etwas beschleunigt, ohne auf das Netzwerk zu warten. Empfehlung: Lassen Sie die Einstellung unverändert.

Unter dem Punkt 5 Internationalisation Options stellen Sie Tastaturlayout, Zeitzone, Systemsprache und WLAN-Kanäle für Ihr Heimatland aus.

Falls Sie später das Raspberry-Pi-Kameramodul nutzen möchten, können Sie die nötigen Treiber mit der Option 6 Enable Camera laden. Empfehlung: Lassen Sie die Einstellung vorerst unverändert.

7 Add to Rastrack: Rastrack war ein Projekt, das auf einer Landkarte alle Standorte von aktiven Raspberry Pis anzeigt. Das Projekt wird aber momentan nicht weiter fortgesetzt und hat den Dienst bis auf Weiteres eingestellt: *http://rastrack.co.uk/index.php*. Lassen Sie daher auch diese Option unberührt.

Über den Punkt 8 Overclock können Sie die Taktfrequenzen für CPU und Speicher erhöhen. Der Raspberry Pi 3 kann nicht mehr auf diese Weise übertaktet werden. Lassen Sie auch bei älteren Modellen diese Option unverändert.

Aktivieren Sie beziehungsweise überprüfen Sie unter dem Punkt 9 Advanced Options, ob die folgenden Punkte aktiviert (Einstellung Enable) sind:

- SSH
- Device Tree
- SPI
- I2C
- Serial
- 1-Wire

Mit Finish beenden Sie *raspi-config*. Der Raspberry Pi übernimmt alle Einstellungen nach einem Neustart. Alternativ können Sie alle Einstellungen auch in der grafischen Oberfläche von Raspbian vornehmen. Navigieren Sie dazu im Startmenü zum Menüpunkt Einstellungen • RaspberryPi-Konfiguration (siehe Abbildung 2.3).

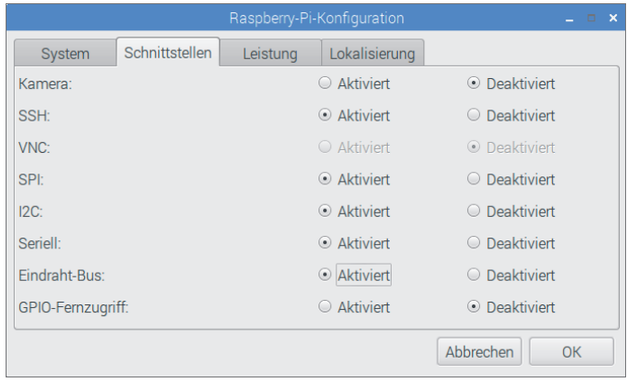


Abbildung 2.3 Die grafische Variante von »raspi-config« im X-Desktop

2.2 Eine WLAN-Verbindung zum Heimnetzwerk herstellen

Die Verbindung des Raspberry Pi in das Heimnetzwerk und damit ins Internet ist essenziell, und wir empfehlen Ihnen, vor dem Start der Elektronikprojekte eine Netzwerkverbindung herzustellen. Sie haben dadurch die Möglichkeit, die Pakete und das Betriebssystem jederzeit auf dem aktuellen Stand zu halten. Zudem können Sie den Raspberry Pi dadurch *kopflos* (headless) betreiben. Das bedeutet, dass Sie keinen Bildschirm sowie keine Tastatur und Maus an den Raspberry Pi anschließen müssen, um ihn zu bedienen. Ebenso wird die Dateiübertragung der *SSH* zum Kinderspiel. *SSH* steht für *Secure Shell*, ein Netzwerkprotokoll für verschlüsselte Verbindungen.

Die Einrichtung einer SSH-Verbindung erläutern wir in Abschnitt 2.3, »SSH-Verbindung herstellen und Dateien übertragen«.

Wir möchten direkt auf die Verbindung per WLAN eingehen, denn wenn Sie eine LAN-Verbindung via Ethernet-Kabel herstellen möchten, müssen Sie das Kabel bloß in die Ethernet-Schnittstelle des Raspberry Pi und in Ihren Router stecken. Eine weitere Einrichtung ist in der Regel nicht notwendig.

In den Raspberry Pi 3 ist bereits ein WLAN-Modul integriert. Für ihn benötigen Sie keine externen USB-Sticks mehr. Das ist bei den Vorgängermodellen sowie beim Raspberry Pi Zero jedoch anders. Hier benötigen Sie einen WLAN-USB-Dongle. Wir empfehlen Ihnen z. B. den *Edimax-WLAN-Stick*, den Sie unter seinem Namen problemlos bei Amazon finden können. Mittlerweile beeinhaltet Raspbian alle grundlegenden Treiber, sodass nach dem Anstecken des USB-Sticks keine weitere Installation von Treibern notwendig ist.

Egal ob Sie das integrierte WLAN des Raspberry Pi 3 oder einen USB-Dongle nutzen, die folgenden Schritte sind für alle Varianten gleich.

Starten Sie den Raspberry Pi, diesmal noch mit einem Bildschirm sowie einer Tastatur und Maus. Sofern Sie nicht direkt zum Desktop booten, müssen Sie sich mit Ihren Anmeldedaten einloggen und die grafische Oberfläche mit dem Befehl `startx` starten. Damit starten Sie den Desktop von Raspbian. Auch wenn Sie ein kompletter Neueinsteiger in die Linux-Welt sind, werden Sie sofort Ähnlichkeiten zu gängigen Betriebssystemen wie Windows erkennen.

Klicken Sie nun einfach oben rechts neben der Uhr auf das WLAN-Symbol (siehe Abbildung 2.4). Wählen Sie Ihr Heimnetzwerk aus, und geben Sie in dem darauffolgenden Eingabefeld Ihr Passwort ein. Nun ist Ihre Netzwerkverbindung bereits eingerichtet.

Wir gehen davon aus, dass Ihr Router mit *DHCP*, der Standardkonfiguration, konfiguriert ist, sodass der Raspberry Pi nun automatisch eine IP-Adresse vom Router zugewiesen bekommt.

WLAN-Empfangsreichweite des Raspberry Pi 3

Wir haben die Erfahrung gemacht, dass die Reichweite des eingebauten WLAN-Moduls des Raspberry Pi 3 nicht wirklich ideal ist. So sollte der Router am besten im gleichen Raum wie der Mini-PC aufgestellt sein. Anderenfalls werden Sie das Netzwerk nicht finden können oder mit Verbindungsabbrüchen zu kämpfen haben.

Abhilfe schafft hier ein USB-LAN-Stick. Selbstverständlich können Sie diesen auch im Raspberry Pi 3 verwenden.

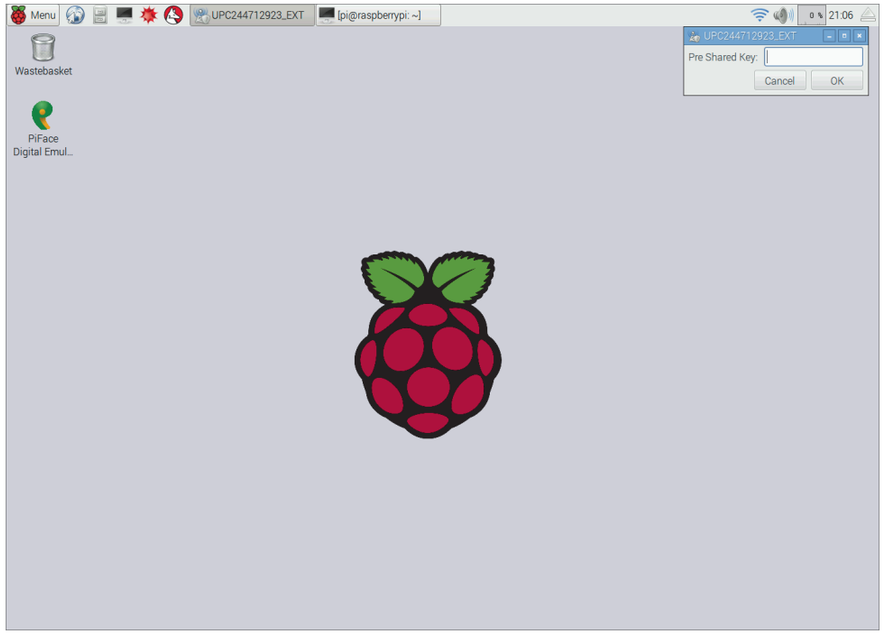


Abbildung 2.4 Nach einem Klick auf das WLAN-Symbol können Sie ein Netzwerk auswählen und sich mit Ihrem Passwort anmelden.

Welche IP-Adresse vergeben wurde, finden Sie heraus, indem Sie nach dem Herstellen der Verbindung nochmals mit dem Mauszeiger über das WLAN-Symbol gehen, im Terminal den Befehl `ifconfig` oder `ip addr` eingeben oder mit Ihrem PC die IP-Adressen im Router auslesen.

Sofern Sie sich noch in der grafischen Oberfläche befinden, starten Sie aus dem Startmenü ZUBEHÖR • LXTERMINAL, um ein Terminal-Fenster zu öffnen. Führen Sie darin nun den Befehl IFCONFIG aus. Sie erhalten eine Ausgabe wie in Abbildung 2.5.

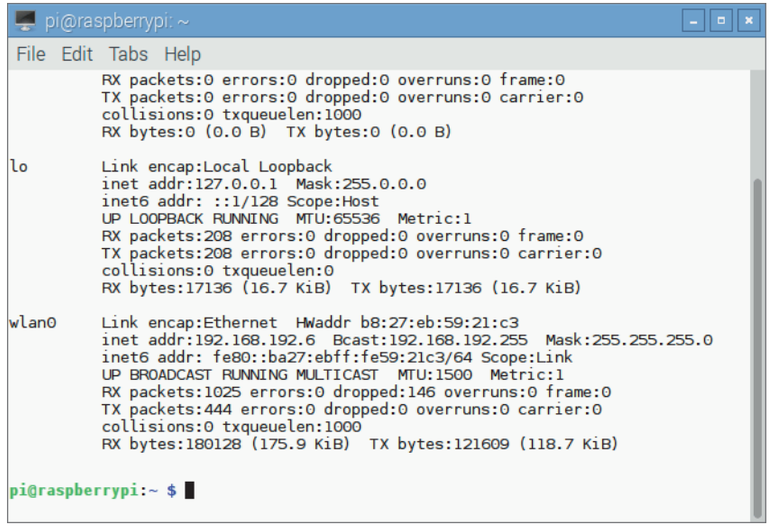


Abbildung 2.5 Die Ausgabe von »ifconfig«. Ihre Heimnetz-IP lautet in diesem Fall »192.168.192.6«.

2.3 SSH-Verbindung herstellen und Dateien übertragen

Wir empfehlen Ihnen, den Raspberry Pi *headless* zu betreiben. Das bedeutet, dass Sie ihn von Tastatur, Maus und Monitor befreien und ihn lediglich mit Versorgungsspannung und Netzwerkzugang ausstatten. Die Steuerung erfolgt dann über einen PC oder Laptop. Das erleichtert die parallele Recherche im Internet und das Kopieren von Programmcodes ungemein.

Der *headless*-Betrieb ist einfach einzurichten. Den Grundstein haben Sie bereits gelegt, als Sie in RASPI-CONFIG die Option SSH aktiviert haben.

Der Recher, von dem aus Sie den Raspberry Pi steuern wollen, benötigt dazu zusätzliche Software. Wir gehen im Folgenden kurz auf die Windows- und Mac-Tools ein, die Sie im Internet kostenlos herunterladen können.

Windows

Nutzen Sie für Windows das Konsolen-Tool *PuTTY* (*http://www.putty.org*). *PuTTY* stellt per SSH eine Verbindung zum Raspberry Pi her und erlaubt es Ihnen, die Shell, also die Konsole, so vom PC aus zu bedienen, als säßen Sie direkt am Raspberry Pi.

Zudem benötigen Sie noch ein Programm, um Dateien von Ihrem PC auf den Raspberry Pi zu schieben. Hier ist das Programm *FileZilla* zu empfehlen (*https://filezilla-project.org*). FileZilla ist ein FTP-Client, der das Protokoll *SFTP* unterstützt, das Sie benötigen, um Dateien auf den Raspberry Pi zu übertragen (siehe Abbildung 2.6).

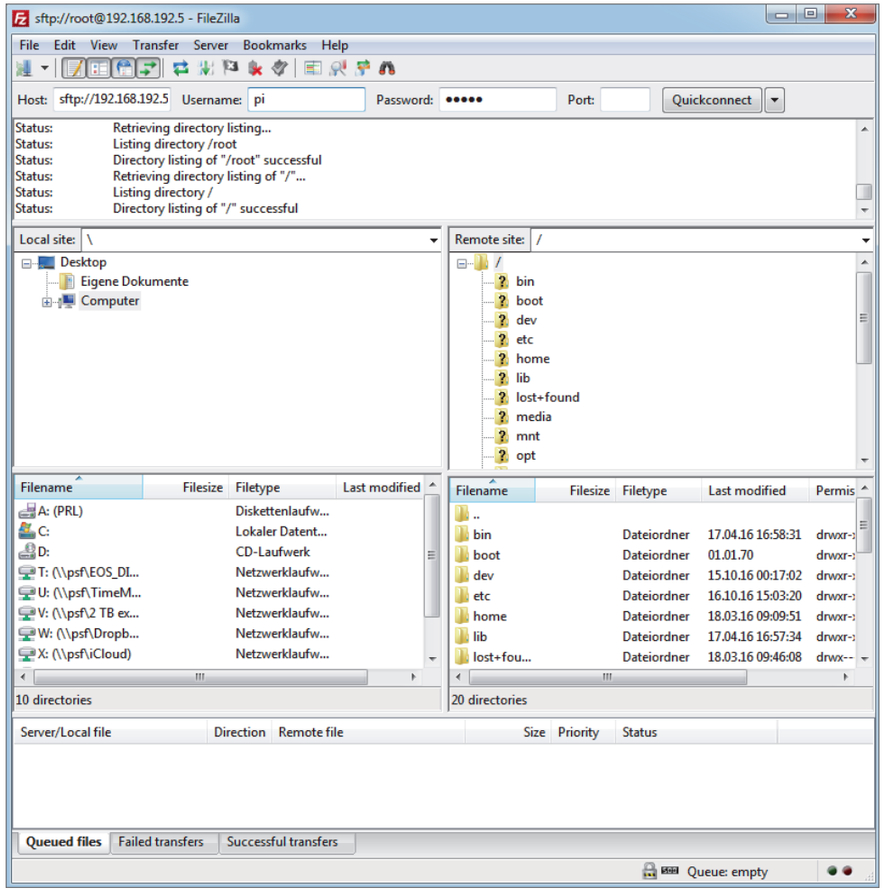


Abbildung 2.6 Der Dateimanager FileZilla erlaubt den unkomplizierten Datentransfer zwischen PC und Raspberry Pi.

Bei beiden Programmen müssen Sie die IP-Adresse eingeben, die Sie im vorigen Abschnitt ermittelt haben.

Nun können Sie mit PuTTy und FileZilla eine Verbindung zum Raspberry Pi herstellen (siehe Abbildung 2.7). Die Konsole können Sie nun wie gewohnt nutzen. Durch den Dateimanager schieben Sie nun einfach z. B. Python-Programme, die Sie am PC geschrieben haben, auf den Raspberry Pi.

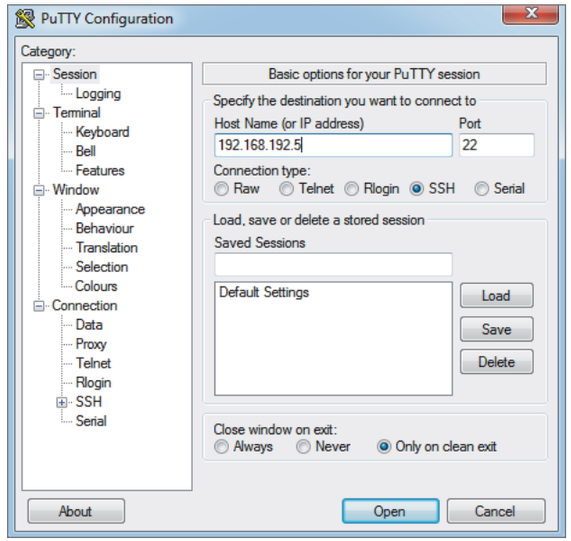


Abbildung 2.7 Die Verbindung von PuTTY zum Raspberry Pi ist simpel.

macOS

Falls Sie einen Rechner mit macOS nutzen, so können Sie das hauseigene TERMINAL einsetzen. Sie verbinden sich mit dem Befehl `ssh pi@192.168.192.5` mit dem Raspberry Pi.

Daraufhin können Sie sich mit Ihren Raspberry-Pi-Anmeldedaten einloggen (siehe Abbildung 2.8).

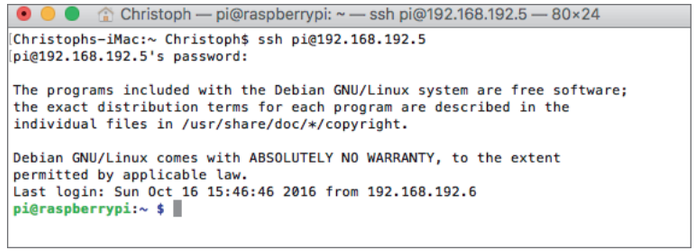


Abbildung 2.8 Die Anmeldung am Raspberry Pi über das macOS-Terminal

Für den Datentransfer können Sie das kostenlose Programm *Cyberduck* nutzen (*https://cyberduck.io*). Ähnlich wie beim Windows-Programm FileZilla handelt es sich bei Cyberduck um einen übersichtlichen Dateimanager für unterschiedliche Übertragungsprotokolle.

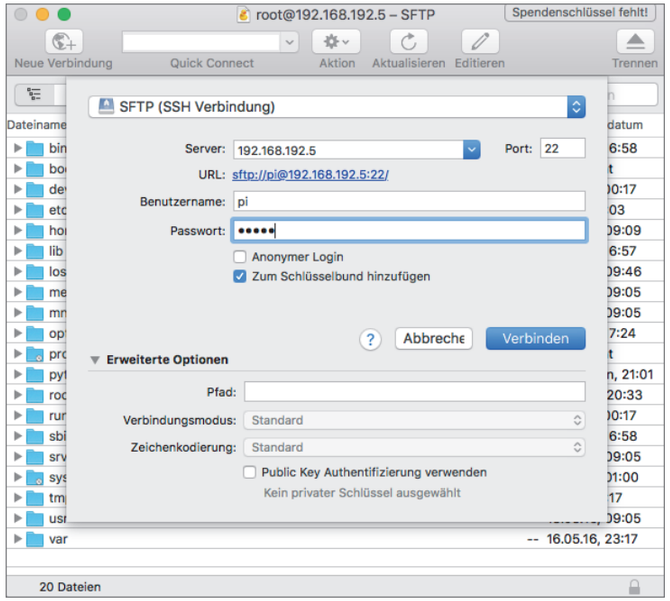


Abbildung 2.9 Das kostenlose Tool Cyberduck kann via SFTP eine Datenverbindung zum Raspberry Pi aufbauen.

2.4 Erste Schritte in Linux

Wir können Ihnen in diesem Buch keine grundlegende Linux-Anleitung bieten. Dazu ist das Betriebssystem schlichtweg zu umfangreich. Trotzdem stellen wir Ihnen einige notwendige Befehle und Werkzeuge vor, die Ihnen den Einstieg in die Linux-Welt erleichtern.

2.4.1 Root-Rechte

Sofern Sie keine Änderung vornehmen, können Sie sich nach der Installation von Raspbian lediglich als Benutzer `pi` anmelden. Dieser Nutzer besitzt keine Admin-Rolle, die in der Linux-Welt als *Root* bezeichnet wird. Viele Aktionen jedoch erfordern eben diese

Root-Rechte, so z. B. auch der Befehl `apt-get update`, der die Paketlisten aktualisiert. Dazu aber später mehr.

Wenn Sie den Befehl als Nutzer `pi` ausführen, erhalten Sie prompt eine Fehlermeldung, die Sie über Ihre fehlenden Rechte belehrt. Nun haben Sie die Möglichkeit, dem Nutzer `pi` kurzfristig Superuser-Rechte zu geben, sodass er diesen einen Befehl mit vollen Rechten ausführen darf. Dazu setzen Sie vor den eigentlichen Befehl das Kürzel `sudo` (*superuser do*). Damit sieht Ihr neuer Befehl also wie folgt aus:

```
sudo apt-get update
```

Ein weiterer Weg ist es, sich direkt als Root-User anzumelden. In einer frischen Raspbian-Installation müssen Sie jedoch dem Root-User zuerst ein Passwort zuweisen, da er noch über keines verfügt.

Führen Sie dazu den Befehl `sudo passwd root` aus. Sie werden nun zweimal nach einem Passwort gefragt, das Sie jetzt festlegen können und zur Sicherheit noch einmal wiederholen. Von nun an können Sie sich auch nach dem Bootvorgang mit dem Nutzer `root` und dem gewählten Passwort anmelden. Im laufenden Betrieb wechseln Sie mit dem Befehl `su root` beziehungsweise `su pi` zwischen den Nutzern.

Root-Rechte stellen ein Sicherheitsrisiko dar!

Wir empfehlen Ihnen dringendst, nicht dauerhaft die vollen Root-Rechte zu verwenden. Im Normalfall können Sie alle Aufgaben mit dem Nutzer `pi` und dem Zusatz `sudo` ausführen. Denn ein Root-User darf *alles*. Und zwar wirklich alles. Wer sich als Root-User anmeldet, dem gewährt Linux alle Rechte im gesamten System. Sie sollten wissen, was Sie tun, denn als Root-User wird Ihnen Linux nur selten die Frage »Sind Sie sicher?« stellen.

Bitte beachten Sie auch, dass nicht nur Sie dadurch über die vollen Rechte verfügen, sondern auch die Programme, die als Root-User gestartet wurden. Stellen Sie sich vor, Sie starten aus Versehen ein schädliches Programm mit Root-Rechten. Der Virus wird nun seine helle Freude an Ihrem System haben.

2.4.2 Software-Verwaltung

Das Raspberry-Pi-Betriebssystem wird nach wie vor fleißig gepflegt und verbessert, ebenso wie die Programmpakete, die Sie auf Ihrem System installieren. Daher ist es ratsam, regelmäßig eine Prüfung auf Updates durchzuführen.

Zum Aktualisieren Ihres Betriebssystems und der installierten Programme (Pakete) benötigen Sie zwei Befehle:

```
sudo apt-get update
```

```
sudo apt-get dist-upgrade
```

Der erste Befehl aktualisiert die Paketlisten; er lädt also eine Liste aller verfügbaren Programme herunter. Dadurch kann der darauffolgende Befehl die Paketliste mit den Programmen abgleichen, die Sie tatsächlich installiert haben. Sollten nun Versionsunterschiede festgestellt werden, so werden Ihnen diese angezeigt und Sie können alle notwendigen Updates durchführen. Führen Sie immer den `update`-Befehl direkt vor dem `dist-upgrade`-Befehl aus, denn der Abgleich mit den aktuellsten Paketversionen erfolgt immer über die lokale Liste. Daher benötigt die Ausführung dieser Befehle auch zwingend eine Internetverbindung.

Doch wozu sollten Sie Programme aktualisieren, wenn Sie nicht wissen, wie Sie überhaupt neue Programme installieren? Doch auch das erledigen Sie mit dem Paketmanager `apt-get`. Möchten Sie beispielsweise den Editor *Joe* installieren, so führen Sie folgenden Konsolenbefehl aus:

```
sudo apt-get install joe
```

Die Konsole zeigt Ihnen nun die Schritte an, die durchgeführt werden:

```
sudo apt-get install joe
  Die folgenden NEUEN Pakete werden installiert: joe
  Es müssen noch 0 B von 474 kB an Archiven herunter
  geladen werden.
```

Wenn Sie das Update zum ersten Mal nach der Raspbian-Installation durchführen beziehungsweise seit dem letzten Update schon lange keines mehr durchgeführt haben, wird das Update relativ lange dauern – durchaus auch eine Viertelstunde: Zeit kostet nicht nur der Download, der oft viele MByte umfasst, sondern auch das Dekomprimieren der Pakete und schließlich ihre Installation auf der SD-Karte. Sie können etwas Zeit sparen, wenn Sie vorher Programme deinstallieren, die Sie nicht benötigen, beispielsweise das Audio-Programm *Sonic Pi* oder das Computeralgebraprogramm *Mathematica*:

```
sudo apt-get remove sonic-pi wolfram-engine
```

2.4.3 Firmware- und Kernel-Updates

Der Befehl `sudo apt-get dist-upgrade` aktualisiert die komplette Raspbian-Distribution – mit zwei Ausnahmen. Nicht berücksichtigt werden der Kernel und die Firmware des Raspberry Pi. Der Kernel ist jener Teil von Linux, der alle essenziellen Low-Level-Funktionen enthält, also Speicherverwaltung, Prozessverwaltung, Hardware-Treiber etc. Die Firmware enthält Software für die GPU (*Graphics Processing Unit*) des Broadcom-System-on-a-Chip, also für die Recheneinheit des Raspberry Pi.

Um nun ein Update der Firmware und des Kernels durchzuführen, führen Sie den Befehl `sudo rpi-update` aus. `rpi-update` lädt dann die gerade aktuelle Firmware- und Kernelversion herunter und installiert die Dateien in die Verzeichnisse */boot* und */lib/modules/n.n.* Vorher wird der ursprüngliche Inhalt von */boot* nach */boot.bak* kopiert, sodass Sie ein Backup der bisherigen Kernel- und Firmware-Version haben. Wenn Sie das Update rückgängig machen möchten, kopieren Sie alle Dateien von */boot.bak* nach */boot* – einmal vorausgesetzt, es gibt nach dem Update keine Boot-Probleme. Bei unseren Tests hat aber immer alles klaglos funktioniert.

Vielleicht fragen Sie sich, warum Sie Firmware und Kernel nicht getrennt voneinander aktualisieren können. Die beiden Programme sind aufeinander abgestimmt. Deswegen aktualisiert `rpi-update` grundsätzlich immer beide Komponenten.

2.4.4 Navigation und Dateioperationen im Terminal

In Tabelle 2.1 finden Sie einige wichtige Befehle, um im Terminal durch die Verzeichnisstrukturen zu navigieren und Operationen wie das Kopieren und Löschen durchzuführen.

Befehl	Funktion
`cd`	Wechselt ins Home-Verzeichnis.
`cd XY`	Wechselt in das Verzeichnis *XY*.
`ls`	Listet den Verzeichnisinhalt auf.
`cp datei1 datei2`	Erstellt eine Kopie von *datei1* und gibt ihr den neuen Namen *datei2*.
`mv altername neuername`	Benennt die Datei *altername* in *neuername* um.
`mv datei1 datei2 datei3 verzeichnis`	Verschiebt die Dateien *datei1*, *datei2*, *datei3* in *verzeichnis*.
`rm datei.bak`	Löscht die Datei *datei.bak*.
`rm -r XY`	Löscht das Verzeichnis *XY*.

Tabelle 2.1 Übersicht einiger wichtiger Linux-Befehle

Befehl	Funktion
`touch XY.xyz`	Erzeugt die neue leere Datei *XY.xyz*.
`mkdir test`	Erzeugt das Verzeichnis *test*.

Tabelle 2.1 Übersicht einiger wichtiger Linux-Befehle (Forts.)

Für den Umgang mit diesem Buch sollte Ihnen dieser kleine Einblick reichen. Befehle und Konfigurationen, die für die einzelnen Projekte notwendig sind, erfahren Sie immer zu Beginn des jeweiligen Kapitels.

Wenn Sie einen kompletten Überblick über das Betriebssystem Linux haben möchten, können wir Ihnen das kostenlose Openbook zum Buch »Linux – Das umfassende Handbuch« vom Rheinwerk Verlag empfehlen:

http://openbook.rheinwerk-verlag.de/linux/

(Und wenn Sie lieber ein gedrucktes Buch in den Händen halten, können wir Ihnen »Einstieg in Linux« von den Autoren des Openbooks empfehlen oder das Handbuch »Linux« von Michael Kofler.)

6.4 Eine PWM mit einem PWM-Controller erzeugen

In Kapitel 3, »I/O-Grundlagen – die Ein- und Ausgänge des Raspberry Pi im Detail«, haben Sie gelernt, wie Sie mit dem Raspberry Pi eine *pulsweitenmodulierte Spannung* (PWM) erzeugen und einsetzen können. Bei der Erzeugung der PWM mit dem Raspberry Pi müssen Sie allerdings mehrere Einschränkungen hinnehmen:

- Der Raspberry Pi besitzt sehr wenige Pins für eine PWM, die durch die Hardware erzeugt wird.
- Jeder I/O kann mit einer Software-PWM versehen werden. Diese ist aber sehr ungenau und kann z. B. für die Ansteuerung eines Servomotors (das besprechen wir im Laufe dieses Abschnitts) nur sehr bedingt verwendet werden.

Wenn Sie diese Einschränkungen umgehen wollen, müssen Sie auf einen externen Chip zurückgreifen, den Sie mit dem Raspberry Pi verbinden.

Aus diesem Grund lernen Sie in diesem Abschnitt, wie Sie mit einem separaten Chip über den I^2C-Bus eine pulsweitenmodulierte Spannung erzeugen können. Solch eine Spannung wird üblicherweise in Kombination mit einem Motortreiber dazu verwendet, um elektrische Motoren in ihrer Drehzahl und Drehrichtung zu steuern.

6.4.1 Eine PWM erzeugen, um eine LED zu dimmen

Es gibt zwei Methoden, eine PWM mit dem Raspberry Pi zu erzeugen:

- durch Ein- und Ausschalten eines Pins mittels Software: Das ist die sogenannte Soft-PWM.
- mit Unterstützung von Hardware im Prozessor oder externen Bausteinen, die speziell darauf ausgelegt sind, eine PWM zu erzeugen

Die Erzeugung einer PWM mittels Software hat den schönen Effekt, dass auch Prozessoren oder Mikrocontroller ohne PWM-Hardware eine PWM erzeugen können. Allerdings ist die erzeugte Soft-PWM in ihren zeitlichen Eigenschaften bei Weitem nicht so präzise wie eine durch Hardware erzeugte PWM. So können die Periodendauer T und die Einschaltzeit t_1 unter Umständen massiv schwanken, wodurch Bauteile, die sehr genaue Timings bei den Pulsen benötigen (wie z. B. ein Servo), nicht vernünftig verwendet werden können.

In diesem Abschnitt verwenden wir einen 12-Bit-PWM-LED-Controller mit der Bezeichnung PCA9685, um eine präzise PWM zu erzeugen, mit der dann auch Servomotoren problemlos angesteuert werden können.

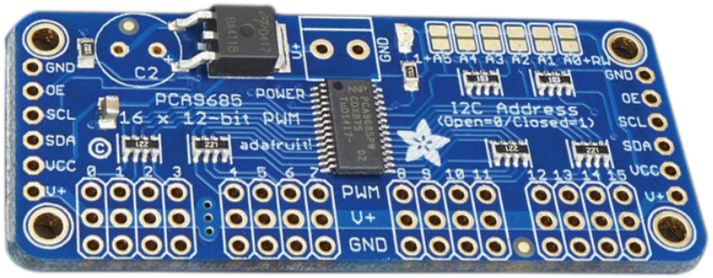


Abbildung 6.44 Der verwendete PWM-Controller

Der Controller aus Abbildung 6.44 ist in der Lage, 16 unabhängige PWM-Signale mit einer frei wählbaren PWM-Frequenz von 24 Hz bis zu 1,526 kHz zu generieren. Die erzeugte PWM besitzt dabei eine Auflösung von 12 Bit, wodurch Sie den Duty Cycle im Bereich von 0 bis 100 % mit einer Genauigkeit von 0,024 % einstellen können:

$$\Delta D = \frac{D_{Max}}{Auflösung} = \frac{100\%}{2^{12}\text{Bit}} = 0{,}024\frac{\%}{\text{Bit}}$$

Der Chip wird über den I²C-Bus konfiguriert und unterstützt Betriebsspannungen von 2,3–5,5 V.

Zudem sind die Eingangs-Pins des Chips 5-V-tolerant. Dies erlaubt es Ihnen, 5-V-Signale an den Chip anzulegen, selbst wenn der Chip nur mit 3,3 V betrieben wird.

In diesem Beispiel wird der Chip mit 3,3 V betrieben. Für die Ansteuerung eines Servos benötigen Sie allerdings noch eine weitere 5-V-Spannungsquelle. Hier bietet sich ein externes Netzteil als ideale Lösung an. Gerade bei Motoren sollten Sie nicht die Spannungsversorgung auf dem Raspberry Pi verwenden. Motoren haben nämlich die unangenehme Eigenschaft, Störungen in der Spannungsversorgung hervorzurufen, die dann so empfindliche Bauteile wie Prozessoren stören können, wodurch es zu Fehlfunktionen kommen kann.

Merke: Spannungsversorgung von Motoren

Motoren sollten immer vom digitalen Steuerkreis *entkoppelt*, also getrennt, werden, da sie sehr viele Störungen in der Stromversorgung verursachen. Je größer die Last an den Motoren ist oder je größer die Motoren sind, desto größer sind die Störungen in der Versorgung. Den Motoren machen solche Störungen nichts aus, aber gerade digitale Schaltkreise haben sehr oft Probleme mit solchen Störungen.

Weiterhin liefert der Raspberry Pi nicht genug Strom, um Motoren versorgen zu können. Wenn Sie also einen Motor direkt an den Raspberry Pi anschließen, kann es sein, dass der Raspberry Pi ausgeht, weil die Spannungsversorgung auf der Platine zusammenbricht.

Die einfachste Möglichkeit, um diese Probleme zu umgehen, ist die Verwendung eines zweiten Netzteils oder einer Batterie als Spannungsversorgung für den Motor.

Alles in allem bietet der Chip eine ganze Fülle von Funktionen, was Sie auch an dem 52-seitigen Datenblatt erkennen können. Um diesen Abschnitt nicht zu lang werden zu lassen, beschränken wir uns daher auf das Minimum, das Sie brauchen, um eine PWM zu erzeugen und um mit dieser PWM einen Servo anzusteuern. Wie bei allen anderen Chips ist auch bei diesem das Datenblatt des Herstellers sehr wichtig, um die Funktionen des Chips zu verstehen und um eine korrekte Kommunikation mit dem Chip zu implementieren.

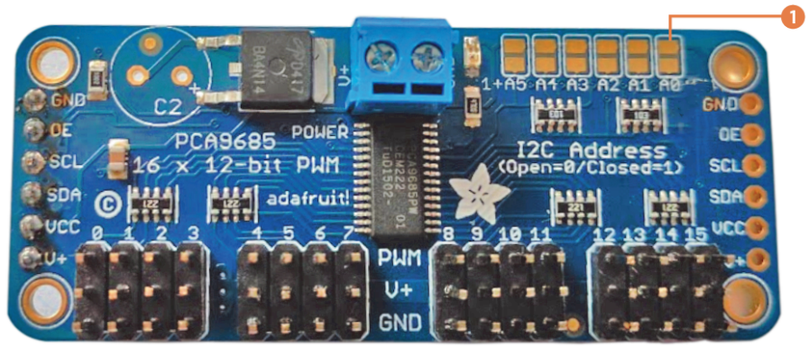


Abbildung 6.45 Das fertig zusammengebaute Breakout Board

Zuallererst muss die Platine aber erst einmal zusammengebaut werden, indem Sie die fehlenden Stiftleisten und die Klemme zur Spannungsversorgung der Servos anlöten (siehe Abbildung 6.45).

Der PWM-Controller besitzt insgesamt sechs Adress-Pins, mit denen Sie die I²C-Adresse des Chips einstellen können. Über die Lötfelder A5 bis A0 (1) können die Adress-Pins auf High gezogen werden, um die Adresse des Chips zu konfigurieren. Mithilfe dieser Adress-Pins können Sie bis zu 64 identische Chips (2^6) an einem Bus betreiben.

In dem hier vorgestellten Beispiel bleiben diese Lötfelder alle offen. Anschließend kann das Board so verdrahtet werden wie in Abbildung 6.46.

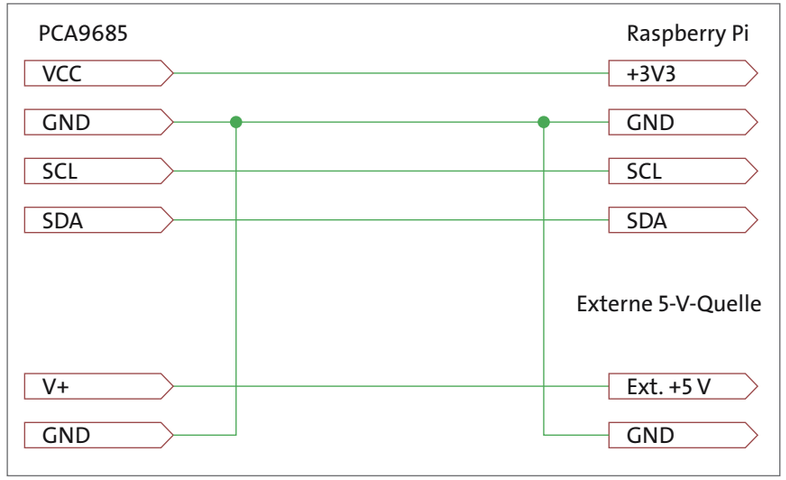


Abbildung 6.46 Das fertig angeschlossene Breakout Board

Als externe Spannungsversorgung können Sie z. B. ein einfaches Steckernetzteil (Maximalstrom mind. 500 mA) mit einem Hohlstecker verwenden (siehe Abbildung 6.47).

Für solche Hohlstecker gibt es schöne Adapter mit einer Schraubklemme, sodass Sie das Steckernetzteil ganz einfach mithilfe der Schraubklemme und ein paar Drahtbrücken mit dem Board verbinden können.

Abbildung 6.47 Hohlbuchse mit Schraubklemme, wie Sie sie bei verschiedenen Händlern bestellen können

Anschließend melden Sie sich auf dem Raspberry Pi an. Mithilfe des Befehls

```
$ i2cdetect -y 1
```

können Sie jetzt überprüfen, ob der Chip korrekt erkannt wird (siehe Abbildung 6.48).


```
root@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: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@raspberrypi:~#
```

Abbildung 6.48 Der Chip wurde korrekt erkannt und kann nun verwendet werden.

Die Adresse des Chips finden Sie in Fig. 4 auf der Seite 8 des offiziellen Herstellerdatenblattes bzw. hier in Abbildung 6.49.

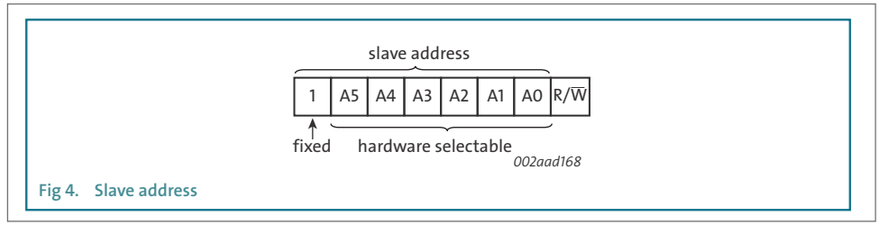


Abbildung 6.49 Die Adresse des Chips aus dem offiziellen Datenblatt des Herstellers

Wenn Sie alle sechs Adress-Pins offen lassen, lautet die I²C-Adresse $0100\,0000_2$ bzw. 0x40. Der Chip wird also korrekt am I²C-Bus erkannt.

Nun können Sie damit beginnen, ein Programm für diesen Chip zu schreiben. Dazu öffnen Sie im MENU • PROGRAMMING das Programm Python 3 und erstellen ein neues Python-Skript. Am Anfang des Programms importieren Sie wieder das Modul *python-smbus* und erzeugen ein Objekt der I²C-Schnittstelle:

```
import smbus
PCA8685 = smbus.SMBus(1)
```

Bevor Sie den PWM-Controller erstmalig verwenden, sollten Sie ihn einmal resetten, da der Controller alle Einstellungen speichert und Sie einen sicheren Ausgangszustand benötigen. Laut Datenblatt erfolgt ein sogenannter Software-Reset, also ein durch ein Programm ausgelöster Reset, indem der Wert 0x06 mit der Adresse 0x00 auf dem I²C-Bus geschrieben wird (siehe Abbildung 6.50).

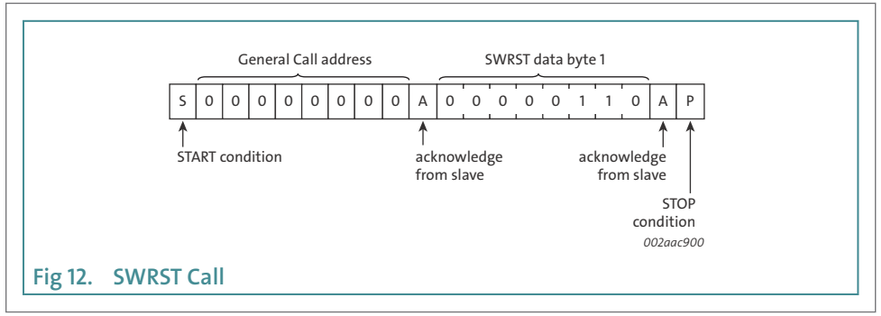


Abbildung 6.50 Das Kommando, um einen Software-Reset (SWRST) auszulösen

Der Reset betrifft alle am I²C-Bus angeschlossenen PCA9685-PWM-Controller. Da der Befehl in kein Register geschrieben wird, kann das Versenden des Befehls über die `write_byte(addr, data)`-Methode erfolgen:

```
PCA8685.write_byte(0x00, 0x06)
```

Mit dieser Methode senden Sie zwei Bytes an Daten, also einmal die Adresse 0x00 sowie den Reset-Befehl 0x06, auf den I²C-Bus. Der PWM-Controller empfängt die Daten, wertet sie entsprechend aus und setzt alle Einstellungen auf die Default-Einstellungen zurück.

Nun können Sie den Chip entsprechend konfigurieren. Dazu besitzt der Chip zwei Konfigurationsregister, die über die Registeradresse 0 und 1 angesprochen werden können. Nach dem Anlegen einer Versorgungsspannung wird eine Default-Konfiguration in den Chip geladen. Sämtliche Bits, die beim Anlegen einer Spannung auf einen bestimmten Wert gesetzt werden, werden mit einem * gekennzeichnet. Diese Konfiguration müssen Sie nun anpassen.

Hinweis: Bei den Chip-Registern immer den Überblick behalten

Je nach Komplexität besitzen integrierte Bausteine mitunter eine Vielzahl verschiedener Register, wie Sie bereits am DAC, ADC und jetzt dem PWM-Controller bemerkt haben. In jedem Datenblatt finden Sie ein *Register summary*, in der der Hersteller alle verfügbaren Register mit ihren Adressen auflistet.

In den Konfigurationsregistern können Sie eine softwareseitige Erweiterung der I²C-Adresse des Chips aktivieren. Durch diese Erweiterung ignoriert der Chip bestimmte Bits in den I²C-Adressen, wodurch es möglich ist, mehrere Chips gleichzeitig mit einer einzigen Adresse anzusteuern. Diese Funktion wird z. B. bei der Verwendung als LED-Controller für große LED-Wände verwendet, um gleich einen ganzen Block an LEDs in einer bestimmten Helligkeit leuchten zu lassen.

Diese Funktion wird in diesem Buch allerdings nicht benötigt und kann daher deaktiviert werden. Dazu müssen Sie das Bit *ALLCALL* im Register *MODE1* setzen und die Bits *SUB3* bis *SUB1* löschen (siehe Abbildung 6.51).

3	SUB1	R/W	0*	PCA9685 does not respond to I²C-bus subaddress 1.
			1	PCA9685 responds to I²C-bus subaddress 1.
2	SUB2	R/W	0*	PCA9685 does not respond to I²C-bus subaddress 2.
			1	PCA9685 responds to I²C-bus subaddress 2.
1	SUB3	R/W	0*	PCA9685 does not respond to I²C-bus subaddress 3.
			1	PCA9685 responds to I²C-bus subaddress 3.
0	ALLCALL	R/W	0	PCA9685 does not respond to LED All Call I²C-bus address.
			1*	PCA9685 responds to LED All Call I²C-bus address.

Abbildung 6.51 Mit diesen Bits können Sie ganze Gruppen von Controllern ansteuern

Sobald Sie eine Spannung an den Chip angelegt haben, befindet sich der Chip in einem Low-Power-Modus. In diesem Modus ist der Oszillator, also der Taktgeber des Chips, deaktiviert. Ohne Oszillator kann der Chip kein PWM-Signal erzeugen.

Um den Chip aus dem Low-Power-Modus in den Normalbetrieb zu versetzen, müssen Sie das *SLEEP*-Bit im Register *MODE1* löschen (siehe Abbildung 6.52).

4	SLEEP	R/W	0	Normal mode[2].
			1*	Low power mode. Oscillator off[3][4].

Abbildung 6.52 Die beiden Betriebsmodi des Chips

Sobald der Oszillator aktiviert wurde, benötigt er mindestens 500 µs, bevor er seine Zielfrequenz erreicht und benutzt werden kann. Diese Zeit müssen Sie warten, bevor Sie mit der Konfiguration des Chips fortfahren.

Alle weiteren Bits sind für Sie im Moment uninteressant und werden benötigt, wenn Sie z. B. einen externen Taktgeber oder mehrere Chips verwenden möchten. Das komplette Byte für das Register *MODE1* nimmt damit den folgenden Wert an:

$$0000\ 0001_2 = 0x01$$

Als Nächstes müssen Sie noch das Register *MODE2* (siehe Seite 16, Table 6 im Datenblatt des Herstellers bzw. Abbildung 6.53) entsprechend beschreiben. In diesem Register legen Sie fest, wie die Ausgangslogik des Chips arbeiten soll. Für uns sind nur das *OUT-*

DRV-Bit und das *INVRT*-Bit des Registers interessant, da diese Bits bestimmen, wie die Treiber der Ausgänge intern beschaltet werden.

2	OUTDRV[1]	R/W	0	The 16 LEDn outputs are configured with an open-drain structure.
			1*	The 16 LEDn outputs are configured with a totem pole structure.

Abbildung 6.53 Die beiden verfügbaren Ausgangstreiber des Chips

Die Hinweise unter der Tabelle im Datenblatt helfen Ihnen bei der Auswahl des richtigen Betriebsmodus. Da Sie lediglich eine LED anschließen wollen, ist es egal, welchen Ausgangstreiber Sie verwenden möchten. Aus diesem Grund können Sie den Default-Wert des Bits (1) in dem Register stehen lassen.

Hinweis: Ausgangsbeschaltung »totem pole« – was ist das?

Bei einer sogenannten *Totem-pole*-Beschaltung handelt es sich um eine Treiberstufe, die aus einem PNP- und einem NPN-Transistor besteht und für hohe Schaltgeschwindigkeiten und hohe Ströme optimiert worden ist. Solch eine Beschaltung wird häufig genutzt, wenn der Ausgangs-Pin sehr hohe Signalfrequenzen verarbeiten oder hohe Ströme treiben muss. Ein Nachteil dieser Beschaltung sind die höhere Komplexität und das nichtlineare Verhalten, wodurch sich dieser Ausgangstreiber nicht für die Übertragung von analogen Signalen eignet, da diese Treiberstufe analoge Signale mit unterschiedlichen Frequenzen wegen der Nichtlinearität unterschiedlich stark verstärkt.

Auf Seite 29 des Datenblattes gibt der Hersteller des Chips noch weitere Empfehlungen für die Beschaltung der Ausgänge des Controllers. Diese Empfehlungen finden Sie dort in Fig 13 bis Fig 15 bzw. hier in Abbildung 6.54 unten.

In Ihren Versuchen schließen Sie die LED bzw. den Servo direkt an den Ausgang des PWM-Controllers an. Sie müssen sich also den Punkt DIRECT CONNECTION TO LEDn in Table 12 des offiziellen Datenblattes anschauen, um zu prüfen, ob Sie weitere Komponenten benötigen.

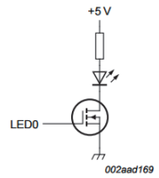
Bei einer direkten Verbindung zwischen LED und Pin des PWM-Controllers wird nur ein Widerstand zur Strombegrenzung benötigt. Jeder Pin kann bis zu 25 mA gegen Masse treiben.

Wenn Sie höhere Ströme benötigten, weil Sie z. B. eine High-Power-LED verwenden wollen, so müssen Sie einen zusätzlichen Treiber oder einen Transistor verwenden. Zusätzlich müssen Sie je nach Konfiguration der *INVRT*- oder *OUTDRV*-Bits einen Pull-up-Widerstand verwenden (siehe Table 12 in Abbildung 6.54).

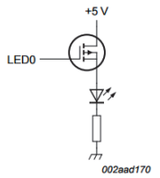
Table 12. Use of INVRT and OUTDRV based on connection to the LEDn outputs when $\overline{OE}$ = 0[1]

INVRT	OUTDRV	Direct connection to LEDn		External N-type driver		External P-type driver	
		Firmware	External pull-up resistor	Firmware	External pull-up resistor	Firmware	External pull-up resistor
0	0	formulas and LED output state values inverted	LED current limiting R[2]	formulas and LED output state values inverted	required	formulas and LED output state values apply	required
0	1	formulas and LED output state values inverted	LED current limiting R[2]	formulas and LED output state values apply[3]	not required[3]	formulas and LED output state values inverted	not required
1	0	formulas and LED output state values apply[2]	LED current limiting R	formulas and LED output state values apply	required	formulas and LED output state values inverted	required
1	1	formulas and LED output state values apply[2]	LED current limiting R	formulas and LED output state values inverted	not required	formulas and LED output state values apply[4]	not required[4]

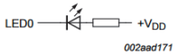
[1] When $\overline{OE}$ = 1, LED output state is controlled only by OUTNE[1:0] bits (MODE2 register).
[2] Correct configuration when LEDs directly connected to the LEDn outputs (connection to V_{DD} through current limiting resistor).
[3] Optimum configuration when external N-type (NPN, NMOS) driver used.
[4] Optimum configuration when external P-type (PNP, PMOS) driver used.



INVRT = 0
OUTDRV = 1
Fig 13. External N-type driver



INVRT = 1
OUTDRV = 1
Fig 14. External P-type driver



INVRT = 1
OUTDRV = 0
Fig 15. Direct LED connection

Abbildung 6.54 Die Beschaltung der Ausgänge bei verschiedenen Konfigurationen

Da wir aber jetzt erst einmal nur eine LED bzw. in Abschnitt 6.4.2 einen Servo ansteuern, müssen Sie lediglich den Vorwiderstand der LED berechnen. Verwenden Sie dafür die folgende Formel:

$$R_V = \frac{U - U_{LED}}{I_{LED}}$$

Für den Wert der Spannung U setzen Sie den Wert des High-Pegels der erzeugten PWN (also 3,3 V) ein, und die Spannung U_{LED} hängt von der gewählten LED ab. Bei einer roten LED beträgt die Spannung etwa 1,9 V. Für den Strom I_{LED} wählen wir den Wert 10 mA. Daraus resultiert dann der folgende Vorwiderstand:

$$R_V = \frac{3{,}3\,\text{V} - 1{,}9\,\text{V}}{0{,}01\,\text{A}} = 140\,\Omega$$

Die Beschaltung des Ausgangs des PWM-Controllers sieht damit so aus wie in Abbildung 6.55.

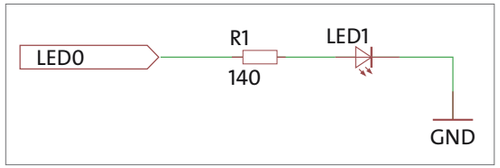


Abbildung 6.55 Die Ausgangsbeschaltung, um eine rote LED über eine PWM anzutreiben

Alle anderen Bits des *MODE2*-Registers sind für Sie uninteressant und können ignoriert werden. Da Sie den Default-Wert des *OUTDRV*-Bits verwenden, ergibt sich als Konfigurationsbyte für das Register *MODE2* der folgende Wert:

$$0000\,0100_2 = 0x04$$

Diese beiden Bits müssen Sie nun übertragen. Dazu benutzen Sie die `write_byte_data(addr, cmd, val)`-Methode des SMBus-Moduls:

```
PCA8685.write_byte_data(0x40, 0x00, 0x01)
time.sleep(0.01)
PCA8685.write_byte_data(0x40, 0x01, 0x04)
```

Über die Methode `sleep(seconds)` des Moduls `time` lassen Sie das Programm für eine bestimmte Zeit pausieren (hier 1 ms). Diese Zeit soll gewährleisten, dass der Oszillator des PWM-Controllers komplett eingeschwungen ist, bevor Sie mit dem Programm fortfahren.

Nun können Sie damit beginnen, eine PWM zu erzeugen. Im ersten Schritt erzeugen Sie eine PWM mit einer Frequenz von 100 Hz und einem Duty Cycle von 50 %, also einem Puls/Pause-Verhältnis von 1.

Im ersten Schritt müssen Sie die Frequenz der PWM einstellen. Der interne Taktgeber des PWM-Controllers arbeitet mit einer Frequenz von 25 MHz, und Sie benötigen eine Frequenz von 100 Hz. Damit Sie aus einer Frequenz von 25 MHz eine Frequenz von 100 Hz erzeugen, müssen Sie die Taktfrequenz des PWM-Controllers teilen. Dies geschieht mithilfe eines sogenannten *Prescalers*, also eines Vorteilers. Dieser Vorteiler besitzt ein Register, in das Sie den Teilungsfaktor hineinschreiben müssen.

Auf Seite 25 des offiziellen Datenblattes finden Sie eine Beschreibung des Prescaler-Registers und eine Formel, um den Teilungswert zu berechnen:

$$Prescale = \frac{Oszillatorfrequenz}{4096 \cdot Frequenz} - 1$$

Beim Anlegen einer Betriebsspannung wird das Prescaler-Register automatisch mit dem Wert 0x1E geladen, was einer PWM-Frequenz von 200 Hz entspricht. Sie benötigen allerdings eine Frequenz von 100 Hz:

$$Prescale = \frac{25\,\text{MHz}}{4096 \cdot 100\,\text{Hz}} - 1 \approx 60 = 0x3C$$

Da Sie in dem Register nur ganzzahlige Werte abspeichern können, müssen Sie das Ergebnis aus der Formel entsprechend runden. Den berechneten Wert müssen Sie nun in das Prescaler-Register schreiben, das laut Datenblatt die Adresse 0xFE besitzt (siehe Abbildung 6.56).

FEh	PRE_SCALE	7:0	PRE_SCALE[7:0]	R/W	0001 1110*	prescaler to program the PWM output frequency (default is 200 Hz)

Abbildung 6.56 Adresse und Default-Wert des Prescaler-Registers

```
PCA8685.write_byte_data(0x40, 0xFE, 0x3C)
```

Das Prescaler-Register kann aber nur beschrieben werden, wenn sich der Chip im Sleep-Modus befindet und der Oszillator pausiert wurde. Sie müssen also vor jeder Änderung der PWM-Frequenz den Chip in den Sleep-Modus versetzen, den Wert für den Prescaler übertragen und den Chip aus dem Sleep-Modus holen.

Um den PWM-Controller in den Sleep-Modus zu setzen, lesen Sie das *MODE1*-Register aus und setzen über eine Und-Verknüpfung das 4. Bit für den Sleep-Modus (siehe Abbildung 6.57).

4	SLEEP	R/W	0	Normal mode[2].
			1*	Low power mode. Oscillator off[3][4].

Abbildung 6.57 Das entsprechende Bit für den Sleep-Modus

Beim Auslesen des Registers müssen Sie das 8. Bit, das sogenannte Restart-Bit, ignorieren, da dieses Bit beim Auslesen nur einen Status signalisiert. Dieser Status darf am Ende nicht wieder in das Bit hineingeschrieben werden, da sich der PWM-Controller sonst je nach Wert des Bits unterschiedlich verhält (siehe Abbildung 6.58).

7	RESTART	R		Shows state of RESTART logic. See Section 7.3.1.1 for detail.
		W		User writes logic 1 to this bit to clear it to logic 0. A user write of logic 0 will have no effect. See Section 7.3.1.1 for detail.
			0*	Restart disabled.
			1	Restart enabled.

Abbildung 6.58 Das Restart-Bit besitzt unterschiedliche Funktionen, je nachdem, ob es gelesen oder geschrieben wird.

Über eine Und-Verknüpfung des ausgelesenen Wertes mit 01111111_2 = 0x7F filtern Sie das *RESTART*-Bit aus dem ausgelesenen Wert heraus:

```
Mode = PCA8685.read_byte_data(0x40, 0x00)
Mode = Mode & 0x7F
```

Nun müssen Sie noch das Bit für den Sleep-Modus setzen, indem Sie eine »1« in das 5. Bit schreiben. Das fünfte Bit (Bitposition 4) erreichen Sie, indem Sie eine 1 um vier Stellen nach links schieben:

```
SleepMode = Mode | (1 << 4)
```

Im nächsten Schritt übertragen Sie das neu erstellte Byte für das *MODE1*-Register zurück auf den PWM-Controller, um diesen in den Sleep-Modus zu versetzen:

```
PCA8685.write_byte_data(0x40, 0x00, SleepMode)
```

Anschließend übertragen Sie den berechneten Wert für den Prescaler:

```
PCA8685.write_byte_data(0x40, 0xFE, 0x3C)
```

Nach dem Übertragen des Prescaler-Wertes müssen Sie den PWM-Controller wieder aufwecken, indem Sie das Sleep-Bit löschen. Dafür übertragen Sie einfach den unter `Mode` abgespeicherten Wert zurück in das Register:

```
PCA8685.write_byte_data(0x40, 0x00, Mode)
time.sleep(0.01)
```

Hierbei dürfen Sie nicht vergessen, das Programm für mindestens 500 µs pausieren zu lassen, um dem Oszillator genug Zeit zum Einschwingen zu geben. Im letzten Schritt müssen Sie dann noch das Restart-Bit (8. Bit im *MODE1*-Register) setzen, um die PWM-Kanäle neu zu starten:

```
PCA8685.write_byte_data(ChipAdresse, 0x00, Mode | (1 << 8))
```

Achtung: PWM-Frequenz wird für alle Kanäle verwendet

Der verwendete Controller benutzt für jeden der 16 Kanäle dieselbe PWM-Frequenz. Es ist somit nicht möglich, zeitgleich unterschiedliche PWM-Frequenzen zu benutzen!

Damit wäre die PWM-Frequenz gesetzt. Im nächsten Schritt müssen Sie dann noch das Puls/Pause-Verhältnis einprogrammieren.

Aus der Register-Summary des PWM-Controllers geht hervor, dass der Chip für jeden Ausgang zwei 16-Bit-Register besitzt, über die Sie die On- und die Off-Zeit des entsprechenden Ausgangs bestimmen können:

- LEDx_ON_L – Offset 0. Byte
- LEDx_ON_H – Offset 1. Byte
- LEDx_OFF_L – Offset 2. Byte
- LEDx_OFF_H – Offset 3. Byte

Jedes der oben genannten Register besitzt einen Offset, der sich auf die Adresse des *ON_L*-Registers bezieht. Die Adresse des *LED0_ON_L*-Registers lautet z. B. 0x06, und das Register *LED0_OFF_H* besitzt einen Offset von 3 Bytes. Daher lautet die Adresse des Registers *LED0_OFF_H* 0x09.

Die Adresse des LEDx_ON_L-Registers können Sie sich entsprechend berechnen lassen, indem Sie wie folgt rechnen:

$$Basisadresse = 0x06 + n \cdot 4$$

Dabei steht der Wert 0x06 für die Adresse des ersten Registers von Kanal 0 und n für die Kanalnummer. Für den Kanal 3 würde damit als Basisadresse der Wert 0x12 herauskommen:

$$Basisadresse = 0x06 + 3 \cdot 4 = 18_{10} = 0x12$$

Mit dieser Methode können Sie direkt eine entsprechende Methode schreiben, um den Duty Cycle eines beliebigen Ausgangs zu setzen:

```
def SetDutyCycle(Pin, OnZeit):
  Basisadresse = 0x06 + Pin * 4
```

In jedes der Registerpaare des entsprechenden Channels müssen Sie nun die On- und Off-Zeit eintragen. Diese Zeiten entsprechen Prozentwerten zu der Periodendauer, also der Zeit von einer Flanke zu der nächsten, identischen Flanke des PWM-Signals (siehe Abbildung 6.59). Angenommen, Sie haben eine PWM-Frequenz von 100 Hz. Daraus ergibt sich dann eine Periodendauer von 10 ms:

$$T = \frac{1}{f} = \frac{1}{100\,\text{Hz}} = 10\,\text{ms}$$

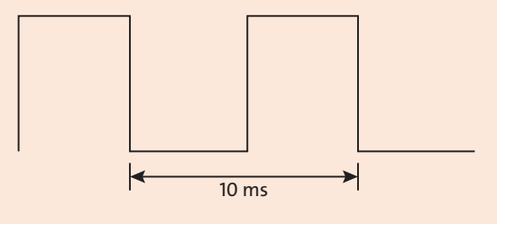


Abbildung 6.59 Periodendauer eines Signals

Diese Periodendauer entspricht nun 100% oder dem Wert 4095 (entspricht 2^{12} bzw. 4096 Werten). Das heißt, wenn Sie eine On-Zeit von 100% der Zeit einer Periode eingestellt haben, erzeugen Sie eine ideale Gleichspannung. Die PWM wird in dem PWM-Controller über einen Zähler erzeugt. Dieser Zähler zählt von 0 bis 4095 (12 Bit) und benötigt zwei Umschaltschwellen: eine Schwelle, um von Low nach High zu schalten, und eine Schwelle, um von High nach Low zu schalten. Diese beiden Schwellen müssen Sie über die Werte in LEDx_ON und LEDx_OFF definieren.

Für den Anfang wollen wir eine On-Zeit von 10% und eine Off-Zeit von 90% einstellen. Die On-Zeit entspricht in diesem Fall 10% der gesamten Periodendauer, also dem zehnten Teil des Maximums 4095. Der Zähler soll während der letzten 410 Schritte (~10% von 4095) einen High-Pegel ausgeben:

$$OnZeit = 4096 - \frac{2^{12}}{100\%} \cdot 10\% - 1 = 4095 - 408{,}6 \approx 3686 = 0xE66$$

Da Sie auch hier nur ganze Zahlen verwenden können, müssen Sie die Ergebnisse entsprechend runden. Damit schaltet der Zähler auf einen High-Pegel, sobald er den Wert 3686_{10} erreicht hat (siehe Abbildung 6.60).

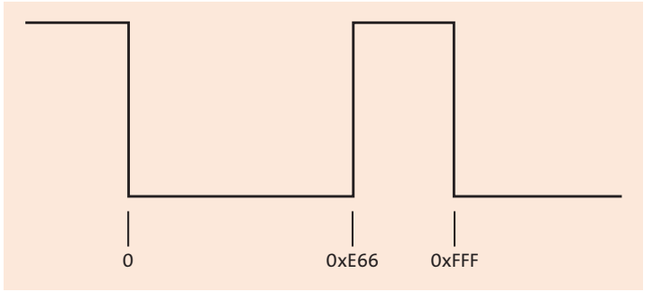


Abbildung 6.60 Die Umschaltschwellen des Controllers

Die zweite Umschaltschwelle ist dann das Ende der Periode, also der Wert 4095_{10}.

$$OffZeit = 2^{12} - 1 = 0xFFF$$

Diese Berechnung müssen Sie nun auf den Code übertragen und am Ende das Ergebnis in zwei einzelne Bytes aufteilen. Anschließend schreiben Sie die beiden Bytes in die entsprechenden Register. Dazu maskieren Sie die einzelnen Bytes der Werte und übertragen diese anschließend mit der `write_byte(addr, cmd, data)`-Methode in die entsprechenden Register des PWM-Controllers. Das Ganze machen Sie nun zweimal – einmal für die On-Zeit und einmal für die Off-Zeit:


```
LowByte = (0xFFF - OnZeit) & 0x00FF
HighByte = ((0xFFF - OnZeit) & 0xFF00) >> 8
PCA8685.write_byte_data(0x40, Basisadresse, LowByte)
PCA8685.write_byte_data(0x40, Basisadresse + 1, HighByte)

LowByte = 0xFFF & 0x00FF
HighByte = (0xFFF & 0xFF00) >> 8
PCA8685.write_byte_data(0x40, Basisadresse + 2, LowByte)
PCA8685.write_byte_data(0x40, Basisadresse + 3, HighByte)
```

Nun können Sie das Programm mit den berechneten Werten testen. Dazu müssen Sie lediglich nach der Initialisierung des PWM-Controllers die eben definierte Unterfunktion aufrufen und einen Pin und die On-Zeit übergeben:

```
SetDutyCycle(0, 0x199)
```

Das fertige Programm sieht nun wie folgt aus:

```
import smbus
import time
PCA8685 = smbus.SMBus(1)
PCA8685.write_byte(0x00, 0x06)

def SetDutyCycle(Pin, OnZeit):
  Basisadresse = 0x06 + Pin * 4
  LowByte = (0xFFF - OnZeit) & 0x00FF
  HighByte = ((0xFFF - OnZeit) & 0xFF00) >> 8
  PCA8685.write_byte_data(0x40, Basisadresse, LowByte)
  PCA8685.write_byte_data(0x40, Basisadresse + 1, HighByte)
  LowByte = 0xFFF & 0x00FF
  HighByte = (0xFFF & 0xFF00) >> 8
PCA8685.write_byte_data(0x40, Basisadresse + 2, LowByte)
PCA8685.write_byte_data(0x40, Basisadresse + 3, HighByte)
PCA8685.write_byte_data(0x40, 0x01, 0x04)
PCA8685.write_byte_data(0x40, 0x00, 0x01)
time.sleep(0.01)
Mode = PCA8685.read_byte_data(0x40, 0x00)
Mode = Mode & 0x7F
SleepMode = Mode | (1 << 4)
PCA8685.write_byte_data(0x40, 0x00, SleepMode)
PCA8685.write_byte_data(0x40, 0xFE, 0x3C)
PCA8685.write_byte_data(0x40, 0x00, Mode)
```

```
time.sleep(0.01)
PCA8685.write_byte_data(0x40, 0x00, Mode | (1 << 8))
SetDutyCycle(0, 0x199)
```

Wenn Sie das Programm nun abspeichern und über Run • Run Module starten, wird eine PWM mit einer Frequenz von 200 Hz und einem Duty Cycle von 10 % erzeugt, welche auf einem Oszilloskop so aussieht wie in Abbildung 6.61.

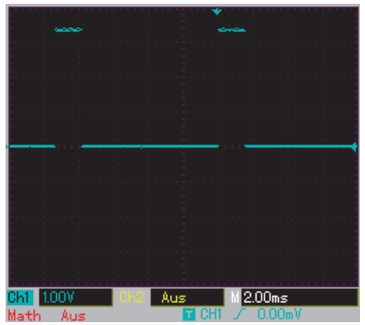


Abbildung 6.61 Oszilloskopbild der erzeugten PWM mit 10 % Duty Cycle

Wenn Sie nun die LED aus Abbildung 6.55 an den Ausgang des PWM-Controllers anschließen, sehen Sie, dass diese nicht leuchtet. Hier gilt nämlich: Je höher der Tastgrad ist, desto heller leuchtet die LED.

Bei einem Duty Cycle von 10 % beträgt der Mittelwert der Spannung 10 % der maximalen Spannung, also 320 mV. Anders ausgedrückt, können Sie auch sagen, dass sich eine PWM mit einem 10 %igen Duty Cycle wie eine Gleichspannung in der Höhe von 10 % der maximalen PWM-Spannung verhält.

Wenn Sie nun den Duty Cycle nach und nach erhöhen (z. B. über einen Timer), werden Sie sehen, dass die LED irgendwann anfängt zu leuchten und dann immer heller wird. Nachfolgend sehen Sie ein Beispiel, bei dem der Duty Cycle alle zwei Sekunden um etwa 10 % (409,6 ~ 409) erhöht wird:

```
for DutyCycle in range(11):
  DutyCycle = DutyCycle * 409
  print("DutyCycle = ", DutyCycle)
  SetDutyCycle(0, DutyCycle)
  time.sleep(2)
SetDutyCycle(0, 0)
PCA8685.close()
```


6.4.2 Eine PWM erzeugen, um einen Servomotor anzusteuern

Da Sie jetzt wissen, wie Sie eine PWM mit dem PCA9685 erzeugen, können wir uns damit beschäftigen, wie Sie einen Servo mit dem PWM-Controller ansteuern können. Doch bevor wir damit beginnen, fassen wir noch einmal die wesentlichen Merkmale eines Servomotors zusammen.

Servos sind, im Gegensatz zu normalen Motoren, in der Lage, den Drehwinkel der Motorachse sehr genau einzustellen. Aus diesem Grund werden Servomotoren z. B. im Modellbau verwendet, um die Ruder des Flugzeugs oder die Spurstange von ferngesteuerten Autos einzustellen.

Abbildung 6.62 Ein handelsüblicher Modellbauservo von Adafruit

Hinweis: analoge und digitale Servos

Neben den hier vorgestellten digitalen Servos gibt es auch noch analoge Servomotoren zu kaufen. Bei analogen Servos wird die Position der Achse über eine analoge Spannung eingestellt. Dementsprechend funktionieren diese Servos mit dem hier gezeigten Beispiel nicht. In diesem Abschnitt beschäftigen wir uns ausschließlich mit dem Ansteuern digitaler Servomotoren.

Ein Servo besteht in der Regel aus einem Motor, einem Drehgeber, um den aktuellen Drehwinkel zu erfassen, einem Getriebe und einer Steuerelektronik. Servos aus dem Modellbaubereich (siehe Abbildung 6.62) besitzen in der Regel drei standardisierte Anschlüsse, wobei die Farben der Leitungen je nach Hersteller unterschiedlich sein können (siehe Tabelle 6.3).

Leitungsfarbe	Signal
Rot	Betriebsspannung (4–6 V)
Schwarz	Masse
Weiß/Orange	PWM-Signal (5-V-Pegel)

Tabelle 6.3 Signale und Leitungsfarben von Servomotoren

Zur Ansteuerung eines Servos werden Impulse mit einer definierten Länge benötigt. Dieses Signal muss in der Regel einen High-Pegel von 5V aufweisen, und die Impulslänge liegt normalerweise zwischen 1 bis 2 ms bei einer Periodendauer von 20 ms, d. h. bei 50 Hz (siehe Abbildung 6.63).

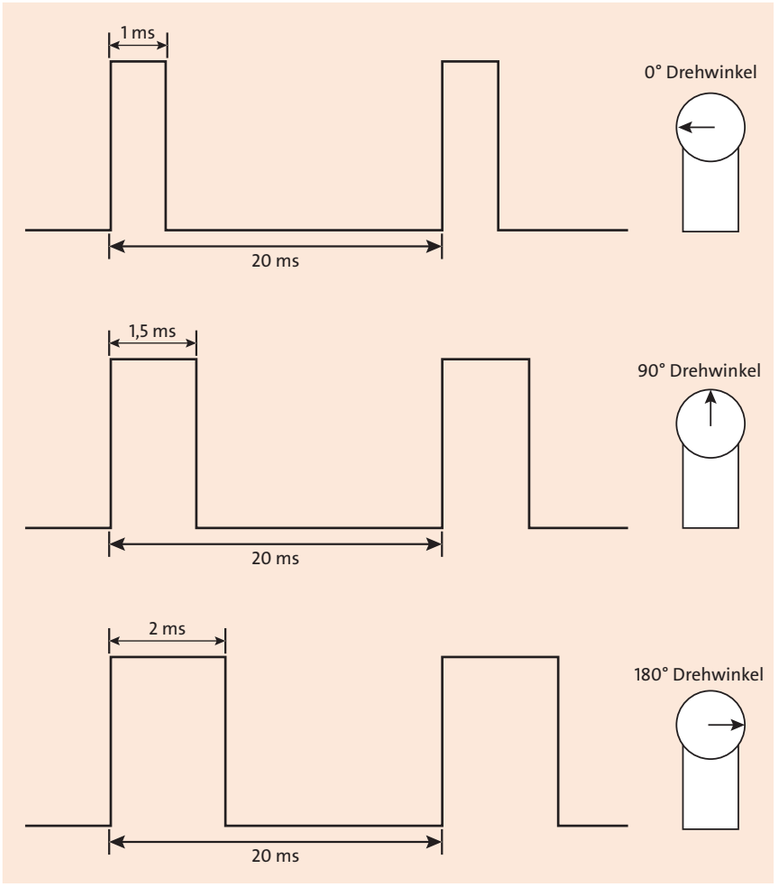


Abbildung 6.63 Die Positionen eines Servos in Abhängigkeit von der Pulslänge

Bei einer Pulslänge von 1 ms bzw. von 2 ms fährt der Servo zum linken bzw. rechten Anschlagspunkt. Der restliche Drehbereich teilt sich dann entsprechend auf die Differenz von 1 ms auf. Bei einer Pulsdauer von 1,5 ms wird dann die Mittelposition eingenommen. Die Erzeugung des PWM-Signals wird später der I²C-Servo-Controller übernehmen, und mit diesem PWM-Signal werden Sie dann einen Servo ansteuern.

Ein solcher Servo kann z. B. mit einem Ultraschallmodul bestückt werden (z. B. *Parallax PING*), um eine Art »Ultraschall-Radar« für einen Roboter zu bauen, wodurch der Roboter dann in der Lage ist, auf Hindernisse zu reagieren.

Nach dieser kurzen Wiederholung der Grundlagen (mehr Informationen finden Sie in Abschnitt 4.2, »Servomotoren«) können wir nun mit dem eigentlichen Thema beginnen: wie Sie einen Servo mit dem PWM-Controller ansteuern.

Dazu müssen Sie als Erstes eine Spannung von 5V an den V+-Eingang des Breakout Boards anlegen (siehe Abbildung 6.46). Hier empfiehlt es sich, eine separate Spannungsquelle zu verwenden, um Störungen durch den Servo in der Spannungsversorgung des Prozessors zu verhindern. Die Spannung können Sie z. B. durch vier in Reihe geschaltete Batterien oder ein Labornetzteil/Steckernetzteil erzeugen.

Den Servo schließen Sie dann an den Ausgang 0 des PWM-Controllers an. Dabei gilt die Belegung aus Tabelle 6.4.

Servo	PWM-Controller
Orange	PWM
Rot	Betriebsspannung (Mitte)
Schwarz/Braun	Masse

Tabelle 6.4 Anschlussbelegung eines Servos an das PCA9685-Breakout-Board

Wenn Sie mit der Verkabelung fertig sind, müssen Sie das erstellte Skript für den PCA9685 modifizieren. Dazu erstellen Sie ein neues Skript namens *Servo.py* und kopieren den Inhalt des PCA9685-Skripts in das Programm *Servo.py*.

Ein Servo benötigt in der Regel ein 50-Hz-Signal, für das Sie den Prescaler des PWM-Controllers neu berechnen müssen:

$$Prescale = \frac{25\,\text{MHz}}{4096 \cdot 50\,\text{Hz}} - 1 \approx 121 = 0x79$$

Diesen Prescaler-Wert müssen Sie nun in den Controller schreiben. Dazu wollen wir ebenfalls eine eigene Methode verwenden:

```
def SetPWMPrecsaler(Prescaler):
  Mode = PCA8685.read_byte_data(ChipAdresse, 0x00)
  Mode = Mode & 0x7F
  SleepMode = Mode | (1 << 4)
  PCA8685.write_byte_data(ChipAdresse, 0x00, SleepMode)
```

```
PCA8685.write_byte_data(ChipAdresse, 0xFE, Prescaler)
PCA8685.write_byte_data(ChipAdresse, 0x00, Mode)
time.sleep(0.01)
PCA8685.write_byte_data(ChipAdresse, 0x00, Mode | (1 << 8))
```

Die Methode `SetPWMPrescaler(Prescaler)` erwartet als Übergabeparameter den eben berechneten Wert für den Prescaler:

```
SetPWMPrescaler(0x79)
```

Über den Duty Cycle der PWM können Sie nun die Position des Servos bestimmen (siehe Abbildung 6.63).

Die Pulsweite variiert zwischen 1 ms und 2 ms und entspricht einem Drehwinkel von 0° bis 180° im Uhrzeigersinn. Eine Pulslänge von 1 ms entspricht bei einer Periodendauer von 20 ms einem Duty Cycle von 1/20, also 5%, und 2 ms sind dementsprechend 2/20, also 10%. In diesem Bereich müssen Sie den Duty Cycle der PWM variieren, um die Position des Servos zu verändern.

Als Werte für die On-Zeit ergeben sich somit folgende Werte:

$$0° \rightarrow \frac{2^{12}}{100\%} \cdot 5\% - 1 \approx 205_{10}$$

$$180° \rightarrow \frac{2^{12}}{100\%} \cdot 10\% - 1 \approx 409_{10}$$

Jeder Servo besitzt bei 0° und 180° Drehwinkel eine Sperre, die verhindert, dass sich der Servo weiterdreht.

Da das Programm die Position des Servos nicht auslesen kann, empfiehlt es sich, die Achse des Servos einmalig bei Programmstart in die Mitte zu drehen, damit der Servo eine bekannte Position einnimmt. Im schlimmsten Fall kann es sonst sein, dass sich die Achse des Servos an einem der Anschlagspunkte befindet und Sie die Achse weiterdrehen möchten und der Servo blockiert und kaputtgeht.

Info: Vorsicht bei der Steuerung von Motoren

Wenn Sie ein Programm schreiben, um Motoren (Gleichstrommotoren, Servos etc.) anzusteuern, ist es immer von Vorteil, wenn Sie den Motor beim Programmstart in eine bekannte Position drehen. Es kann z. B. passieren, dass Ihr Programm oder der Rechner, auf dem das Programm läuft, abstürzt und das Programm bei einem Neustart somit nicht weiß, wo sich der Motor befindet. Durch das Anfahren einer bekannten Position umgehen Sie dieses Problem geschickt.

Als Neutralposition können Sie z. B. die Mittelstellung anfahren lassen. Dazu benötigen Sie einen Impuls mit der Länge von 1,5 ms bzw. einem Duty Cycle von 7,5%:

$$\textit{Mittelstellung} \rightarrow \frac{2^{12}}{100\%} \cdot 7{,}5\% - 1 \approx 307_{10}$$

Diesen Wert müssen Sie nun an den PWM-Controller übertragen:

```
SetDutyCycle(0, 307)
```

Der komplette Drehbereich des Servos teilt sich linear zwischen dem rechten und dem linken Anschlagspunkt auf. Zwischen diesen beiden Anschlagspunkten liegt ein PWM-Bereich von 205 Schritten:

$$\Delta D = 409_{10} - 204_{10} = 205_{10}$$

Diese 205 Schritte stellen nun 180° dar, und dementsprechend dreht sich die Achse des Servos pro Schritt um 0,87° pro Schritt:

$$\alpha = \frac{180°}{205\ \text{Schritte}} = 0{,}87\frac{°}{\text{Schritt}}$$

Mit diesem Wert können Sie nun den Duty Cycle berechnen, der benötigt wird, um den Servo um einen bestimmten Winkel zu drehen. Angenommen, Sie wollen den Servo von der 0°-Position um 42° drehen, dann benötigen Sie folgenden Duty Cycle:

$$n = \frac{42°}{0{,}87\frac{°}{\text{Schritt}}} = 49\ \text{Schritte}$$

$$D = 205 + 49 = 254$$

Mit diesen Werten können Sie nun eine Methode schreiben, die einen Winkel als Übergabewert erwartet und dann den Servo entsprechend dreht:

```
def RotateServo(Winkel):
  DutyCycle = Winkel / 0.87
  SetDutyCycle(0, 204 + int(DutyCycle))
```

Wichtig ist, dass Sie den Wert `DutyCycle` vor der Übergabe an die `SetDutyCycle()`-Methode in einen Integer umwandeln, da die Methode keine Kommazahlen verarbeiten kann.

Über eine `for`-Schleife können Sie dann z. B. den kompletten Winkelbereich des Servos anfahren:

```
for Winkel in range(0, 180, 1):
  RotateServo(Winkel)
  time.sleep(0.1)
```

Die Verzögerungszeit von 0,1 s ist beliebig gewählt und kann dementsprechend verändert werden. Sie müssen dem Servo nur genug Zeit geben, um die Position anzufahren. Die benötigte Zeit lässt sich aus der Stellzeit ❶ des Servos berechnen, die im Datenblatt angegeben ist. Sie ist vom Drehwinkel abhängig (siehe Abbildung 6.64).

So benötigt der in Abbildung 6.64 gezeigte Servo bei einer Spannung von 6 V etwa 3 ms, um die Achse um 1° zu drehen:

$$t = \frac{0{,}18\ \text{s}}{60\,°} = 0{,}003\ \frac{\text{s}}{°}$$

Gewicht	61 g
Getriebe	Metall
Lagerart	Doppelt kugelgelagert
Servo-Technologie	Digital-Servo
Kategorie	Standard-Servo
Stecksystem	JR
Länge	40.7 mm
Breite	20 mm
Höhe	42.4 mm
Stell-Moment (4,8 V)	130 Ncm
Stell-Moment bei 6 V	160 Ncm
Herst.-Teilenr.	80101028
Stell-Zeit bei 6 V	0,18s 60 °
Stell-Zeit bei 4,8 V	0,20s 60 °

❶

Abbildung 6.64 Beispiel für die technischen Daten eines Servos

Wenn Sie nun den Duty Cycle der PWM und damit den Stellwinkel schneller als alle 0,003 s ändern, dann schafft der Servo es nicht, die Position anzufahren. Daher empfiehlt es sich immer, dem Servo ausreichend Zeit zu geben, um die gewünschte Position anzufahren. In Listing 6.1 sehen Sie das fertige Programm:

```
import smbus
import time
ChipAdresse = 0x40
PCA8685 = smbus.SMBus(1)
PCA8685.write_byte(0x00, 0x06)
def SetDutyCycle(Pin, OnZeit):
  Basisadresse = 0x06 + Pin * 4
  LowByte = (0xFFF - OnZeit) & 0x00FF
  HighByte = ((0xFFF - OnZeit) & 0xFF00) >> 8
  PCA8685.write_byte_data(ChipAdresse, Basisadresse, LowByte)
  PCA8685.write_byte_data(ChipAdresse, Basisadresse + 1, HighByte)
  LowByte = 0xFFF & 0x00FF
  HighByte = (0xFFF  & 0xFF00) >> 8
  PCA8685.write_byte_data(ChipAdresse, Basisadresse + 2, LowByte)
  PCA8685.write_byte_data(ChipAdresse, Basisadresse + 3, HighByte)
def SetPWMPrescaler(Prescaler):
  Mode = PCA8685.read_byte_data(ChipAdresse, 0x00)
  Mode = Mode & 0x7F
  SleepMode = Mode | (1 << 4)
  PCA8685.write_byte_data(ChipAdresse, 0x00, SleepMode)
  PCA8685.write_byte_data(ChipAdresse, 0xFE, Prescaler)
  PCA8685.write_byte_data(ChipAdresse, 0x00, Mode)
  time.sleep(0.01)
  PCA8685.write_byte_data(ChipAdresse, 0x00, Mode | (1 << 8))
def RotateServo(Winkel):
  DutyCycle = Winkel / 0.87
  SetDutyCycle(0, 204 + int(DutyCycle))
PCA8685.write_byte_data(ChipAdresse, 0x01, 0x04)
PCA8685.write_byte_data(ChipAdresse, 0x00, 0x01)
time.sleep(0.01)
SetPWMPrescaler(0x79)
RotateServo(0)
time.sleep(5)
RotateServo(90)
time.sleep(5)
RotateServo(180)
time.sleep(5)
while(True):
  for Winkel in range(0, 180, 1):
    RotateServo(Winkel)
    time.sleep(0.05)
  time.sleep(1)
  for Winkel in range(180, 0, -1):
    RotateServo(Winkel)
    time.sleep(0.05)
  time.sleep(1)
SetDutyCycle(0, 0)
```

Listing 6.1 Das Programm zur Ansteuerung von Servomotoren

Auf einen Blick

Inhalt

Index

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Daniel Kampert, Christoph Scherbeck

Elektronik verstehen mit Raspberry Pi

361 Seiten, broschiert, in Farbe, Februar 2017
29,90 Euro, ISBN 978-3-8362-2869-5

www.rheinwerk-verlag.de/3602

Daniel Kampert absolvierte eine Ausbildung zum Elektroniker für Geräte und Systeme und arbeitet derzeit in der Entwicklung der Firma Jenoptik Robot GmbH. Der begeisterte Bastler entwickelt beruflich wie privat elektronische Schaltungen und programmiert für diverse Plattformen wie Raspberry Pi, Atmel AVR u. a. In seinem Blog schreibt er regelmäßig Anleitungen zu Maker-Themen.

Christoph Scherbeck, Jahrgang 1985, ist gelernter Mechatroniker und Maschinenbautechniker sowie begeisterter Raspberry-Pi-Bastler. Als Bestsellerautor zum Raspberry Pi hat er sich bereits einen Namen gemacht, privat betreibt er die Website http://www.elektronx.de, auf der er seine Bastelprojekte mit dem Raspberry Pi beschreibt.