Kapitel 1

Kauf und Inbetriebnahme

Der Raspberry Pi ist ein winziger Computer. Seine Grundfläche ist etwas größer als eine Kreditkarte. In ein Gehäuse verpackt, hat der Computer das Volumen von zwei Smartphones. Das eigentliche Grundgerät kostet je nach Händler etwa 35 EUR. Zusätzlich brauchen Sie in der Regel ein Netzteil, ein Gehäuse, eine SD-Speicherkarte und eventuell ein paar Kabel. Die Gesamtinvestition liegt also deutlich unter 100 EUR.

Dafür erhalten Sie einen vollwertigen, Linux-basierten Computer mit einer ARM-CPU, den Sie zur Steuerung elektrischer Geräte, für Versuchsaufbauten, als Mini-Server oder als kleines Multimedia-Center in der Art des Apple TV einsetzen können. Prinzipiell kann der Raspberry Pi sogar als Ersatz für einen gewöhnlichen PC verwendet werden. Allerdings kann der Raspberry Pi hier, was die Geschwindigkeit betrifft, nicht mit modernen Rechnern mithalten.

Dieses Kapitel gibt Tipps zum Kauf des Raspberry Pi samt des erforderlichen Zubehörs. Außerdem erfahren Sie, wie Sie auf Ihrem Notebook oder PC eine SD-Karte so einrichten, dass Sie diese als Betriebssystem für Ihren Raspberry Pi verwenden können. Sobald Sie diesen Schritt geschafft haben, können Sie Ihren Raspberry Pi erstmals starten und verwenden. Die ersten Schritte unter Raspbian, dem beliebtesten Betriebssystem für den Raspberry Pi, beschreibt dann das nächste Kapitel.

Gewissermaßen als Zuckerl für Linux-Experten enthält dieses Kapitel auch eine Anleitung, wie Sie Raspbian auf einen USB-Stick anstelle der SD-Karte installieren können. Der größte Vorteil dieser Vorgehensweise besteht darin, dass ein USB-Stick in der Regel zuverlässiger arbeitet als eine SD-Karte. Allerdings wird die Installation dadurch etwas komplizierter, weswegen Linux- bzw. Raspberry-Pi-Einsteiger vorerst von dieser Installationsvariante absehen sollten.

1.1 Kauf

Sofern Sie noch keinen Raspberry Pi besitzen, steht zuerst der Kauf an. Beachten Sie, dass Sie den Raspberry Pi ohne jedes Zubehör erhalten – es sei denn, Sie entscheiden sich für ein in der Regel überteuertes Komplettpaket! Zur Inbetriebnahme benötigen Sie deswegen auch ein Netzteil, eine SD-Karte, eine Tastatur und eine Maus mit USB-Anschluss, einen Monitor mit HDMI-Eingang sowie die dazugehörigen Kabel.

Bezugsquellen

Den Raspberry Pi sowie die gerade aufgezählten Zubehörteile können Sie unkompliziert im Internet erwerben. Neben Amazon und großen Elektronik-Händlern wie Conrad oder Pollin gibt es auch eine Menge kleinere Web-Shops, die sich auf Elektronikbastler und die sogenannte Maker-Szene spezialisiert haben. Beachten Sie beim Einkauf immer den jeweiligen Firmenstandort! Manche besonders günstige Angebote werden aus asiatischen Ländern versandt. Das kann nicht nur lange dauern, sondern auch zu Zollproblemen führen.

Raspberry-Pi-Modelle

Vom Raspberry Pi sind verschiedene Modelle erhältlich, von denen wir Ihnen hier die wichtigsten präsentieren:

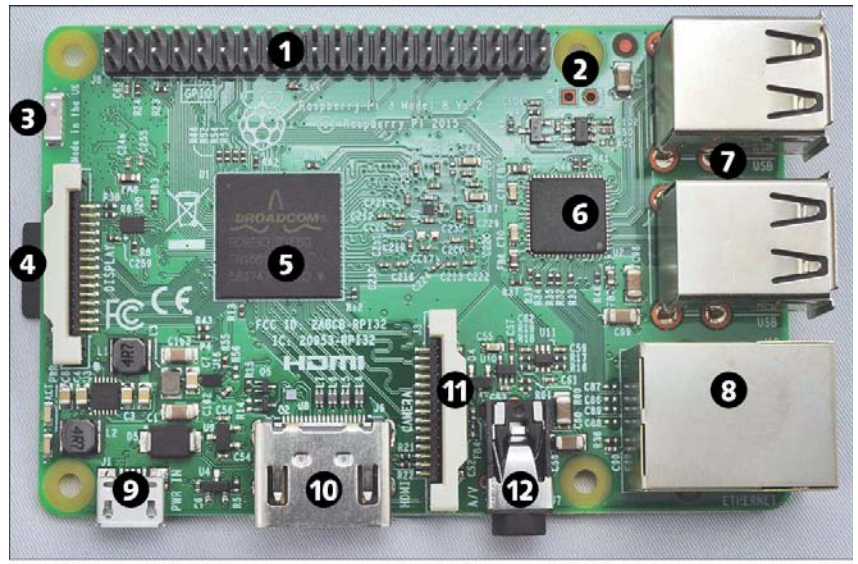
- **Raspberry Pi 3, Modell B (RPi3-B):** Dieses seit Februar 2016 verfügbare Modell ist der zurzeit leistungsfähigste Raspberry Pi (siehe Abbildung 1.1). Eine neue 64-Bit-CPU mit einer Taktfrequenz von 1,2 GHz machen den Raspberry Pi deutlich schneller als das Vorgängermodell.

  Der Rechner verfügt über vier USB-2.0-Anschlüsse, einen 100-MBit-Netzwerkanschluss, je einen WLAN- und Bluetooth-Adapter sowie über eine 40-Pin-Steckerleiste mit GPIOs (General Purpose Input/Output). Die Rechenleistung stellt ein Broadcom-BCM2837-SoC (System-on-a-Chip) zur Verfügung: Er enthält vier CPU-Cores in ARMv8-Architektur sowie einem Broadcom Video-Core IV mit H.264-Encoder/Decoder. Die Leistungsaufnahme des Minirechners ohne Peripheriegeräte beträgt je nach CPU-Auslastung zwischen 2,5 und 4,5 Watt.

- **Raspberry Pi 2, Modell B (RPi2-B):** Dieses Modell wurde im Februar 2015 vorgestellt. Seine CPU basiert auf der ARMv7-Architektur und ist mit 900 MHz getaktet. Im Vergleich zum RPi3-B fehlen der WLAN- und Bluetooth-Adapter.

- **Raspberry Pi Zero:** Diese seit Dezember 2015 lieferbare Variante des Raspberry Pi wurde auf das absolute Minimum geschrumpft (siehe Abbildung 1.2): Anstelle einer normalen HDMI-Buchse gibt es deren Mini-Variante. Es gibt zwei Micro-USB-Buchsen: eine für die Stromversorgung und eine zur Datenübertragung. Weitere USB-Buchsen wurden ebenso eliminiert wie die Ethernet-Buchse und der analoge Audio-Ausgang. Anstelle der GPIO-Steckerleiste gibt es nur noch 40 Lötpunkte.

  Einen Kameranschluss stellt erst die ab Mai 2016 verfügbare erneuerte Zero-Variante zur Verfügung. Dabei ist aber zu beachten, dass zum Anschluss der Kamera nicht das Standardkabel verwendet werden kann, sondern eines mit dem besonders kleinen FPC-Anschluss. Ein geeignetes Kabel erhalten Sie in der Regel zusammen mit dem Raspberry Pi Zero.

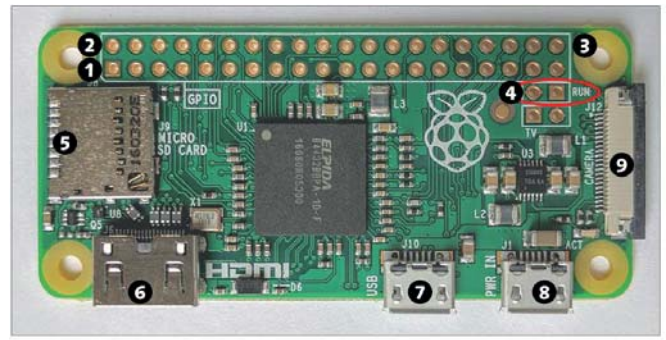


1 GPIO-Header J8 40-polig
2 Run-Header (P6)
3 Keramikantenne für WLAN und Bluetooth
4 DSI-Anschluss für Displays
5 SoC BCM2837
6 LAN-/USB-Controller
7 2x 2 USB-Ports
8 RJ45-LAN-Anschluss
9 Micro-USB-Buchse 5 V
10 HDMI-Anschluss
11 CSI-Anschluss für Camera-Board
12 3,5-mm-Audio- und Videoausgang

Abbildung 1.1 Der Raspberry Pi 3 (Modell B)

Das Zero-Modell verwendet den vom Raspberry Pi 1 bekannten SoC BCM2835 mit nur einem CPU-Core bei einer Taktfrequenz von 1 GHz. Der Arbeitsspeicher wurde auf 512 MByte reduziert.

Diesen Nachteilen stehen einige Vorteile gegenüber: Der Preis des Zero-Modells wurde auf sagenhafte fünf Euro reduziert. Die Leistungsaufnahme beträgt nicht einmal 1 Watt. Die Platine ist weniger als halb so groß wie die des Raspberry Pi 2, Modell B.



1 Pin 1
2 Pin 2
3 Pin 40
4 Run-Header (Reset)
5 Micro-SD
6 Mini-HDMI
7 Micro-USB-Daten
8 Micro-USB-Stromversorgung
9 Kameraanschluss

Abbildung 1.2 Der Raspberry Pi Zero

- **Raspberry Pi 1, Modell B+ (RPi1-B+):** Das Vorgängermodell zum Raspberry Pi 2 stimmt mit diesem in seinen Maßen und Anschlussmöglichkeiten vollständig überein. Allerdings verfügt der eingebaute BCM2835 nur über einen CPU-Core in ARMv6-Architektur, der standardmäßig mit 700 MHz getaktet wird. Zudem ist der Arbeitsspeicher nur 512 MByte groß.
- **Raspberry Pi 1, Modell A+ (RPi1-A+):** Diese Variante zum Raspberry Pi 1, Modell B+, ist mit weniger Speicher (256 MByte) ausgestattet. Es gibt nur einen USB-Anschluss und keinen Netzwerkanschluss. Diesen Nachteilen stehen auch Vorteile gegenüber: Das Modell ist nicht nur billiger und kleiner, es hat auch eine deutlich geringere Leistungsaufnahme von circa einem Watt.
- **Raspberry Pi 1, Modelle A und B (RPi1-A und RPi1-B):** Diese recht alten Modelle sind kaum mehr bei Händlern erhältlich. Der Hauptunterschied im Vergleich zu den Modellen A+ und B+ besteht darin, dass die GPIO-Steckerleiste nur 26 Pins umfasst.
- **Raspberry Pi 1, Compute Module:** Bei dieser Raspberry-Pi-Variante wurde das gesamte Innenleben des Raspberry Pi auf einer deutlich kleineren Platine realisiert, die die Form eines DDR2-SODIMM-Speicherriegels hat und somit weniger als halb so groß wie der originale Raspberry Pi ist. Das Compute Module enthält standardmäßig einen 4 GByte großen Flash-Speicher und macht mehr Steuerungs-Pins des BCM2835 zugänglich, bietet also mehr GPIOs. Wirklich genutzt werden kann dieser Raspberry Pi allerdings nur in Kombination mit einem I/O-Board, das die Anschlüsse nach außen führt. Das Compute Module ist vor allem für die industrielle Nutzung gedacht, z. B. wenn der Raspberry Pi zur Steuerung eines in hohen Stückzahlen produzierten Geräts verwendet werden soll.

	Modell A/A+ 1 × USB	**Modell B/B+** 2/4 × USB, Ethernet	**Zero** 65 mm × 35 mm, 1 × Micro-USB, 1 × Mini-HDMI	**Compute Module** 68 mm × 30 mm, inkl. 4 GB eMMC
Version 1 **BCM2835** 1 Core, ARMv6	**April 2012 (A)** **Nov. 2014 (A+)** 700 MHz, 256 MB	**April 2012 (B)** **Juli 2014 (B+)** 700 MHz, 512 MB	**Nov. 2015** **Mai 2016** 1 GHz, 512 MB	**Juni 2014** 700 MHz, 512 MB
Version 2 **BCM 2836** 4 Cores, ARMv7		**Feb. 2015** 900 MHz, 1 GB		
Version 3 **BCM 2837** 4 Cores, ARMv8		**Feb. 2016** 1,2 GHz, 1 GB		

Abbildung 1.3 Überblick über die bis Mitte 2016 vorgestellten Raspberry-Pi-Modelle, jeweils mit Taktfrequenz und RAM-Größe

USB-Mängel

Die Modelle RPi1-B+, RPi2-B und RPi3-B verfügen über vier USB-Anschlüsse. Sie sollten sich aber darüber im Klaren sein, wie diese Anschlüsse technisch realisiert sind: *Ein* USB-Kanal, den der BCM283x zur Verfügung stellt, führt zu einem internen Hub. Dieser ist dann mit den vier USB-Anschlüssen *und* dem Ethernet-Anschluss verbunden. Mit anderen Worten: Alle USB-Geräte und der Ethernet-Anschluss *teilen* sich die Bandbreite eines USB-2.0-Kanals.

Kaufempfehlung

Der Raspberry Pi Zero ist zwar das billigste Modell, aber aus unserer Sicht für die Mehrheit der Bastler nicht die beste Variante: Das Hauptproblem besteht darin, dass es viel weniger Anschlüsse gibt. Adapter-Stecker und der für die Inbetriebnahme erforderliche USB-Hub machen die anfängliche Kostenersparnis teilweise gleich wieder zunichte. Das Anlöten von Kontaktstiften oder anderen Bauteilen an die Lötpunkte des Zero-Modells ist für Einsteiger auch nicht der ideale Start.

Wir raten Ihnen für Ihre ersten Raspberry-Pi-Experimente zum Raspberry Pi 3, Modell B. Für ein paar Euro mehr erhalten Sie einen Rechner, dessen Bedienung auch im Desktop-Betrieb Spaß macht und der aufgrund seiner vielen Anschlüsse wesentlich einfacher zu beschalten ist. Wenn Sie später ein Projekt durchführen, bei dem die geringe Größe oder Leistungsaufnahme des Zero-Modells ein Vorteil ist, können Sie immer noch ein Exemplar dieses Mini-Modells erwerben.

Wenn Sie schon ein älteres Raspberry-Pi-Modell zu Hause haben, spricht natürlich nichts dagegen, dieses weiterzuverwenden. Zum Experimentieren und Basteln ist die im Vergleich zu aktuellen Modellen geringere Geschwindigkeit zumeist immer noch mehr als ausreichend.

Die Anschlüsse des Raspberry Pi 3 (Modell B)

Modell B des Raspberry Pi 3 bietet die folgenden Anschlussmöglichkeiten (siehe Abbildung 1.4):

- einen Micro-USB-Anschluss zur Stromversorgung (5 V, 1 bis 2,5 A, entspricht 5 bis 12,5 Watt). Der tatsächliche Stromverbrauch hängt von der CPU-Auslastung und dem Leistungsbedarf der USB-Geräte ab.
- vier gewöhnliche USB-2.0-Anschlüsse für Tastatur, Maus und andere USB-Geräte. Der RPi3-B kann über diese USB-Anschlüsse insgesamt 1200 mA weitergeben.
- einen HDMI-Ausgang für Bild und Ton, Auflösung bis zu 1920×1200 Pixel

- einen kombinierten Audio/Video-Ausgang für einen vierpoligen 3,5-mm-Klinkenstecker. Wenn das Video-Signal nicht genutzt werden soll, kann das Audio-Signal auch mit jedem dreipoligen 3,5-mm-Klinkenstecker abgegriffen werden.
- einen Micro-SD-Slot (SDHC)
- einen Ethernet-Anschluss (10/100 MBit)
- eine Steckerleiste mit 40 Pins (der sogenannte »J8-Header«) für allgemeine Zwecke (General Purpose Input/Output inklusive UART, I^2C-Bus, SPI-Bus, I^2S-Audio). Eine detaillierte technische Beschreibung der GPIO-Pins folgt in Kapitel 11, »Hardware-Einstieg«.
- einen integrierten WLAN-Adapter (leider ohne Anschlussmöglichkeit für eine externe Antenne)
- einen integrierten Bluetooth-Adapter

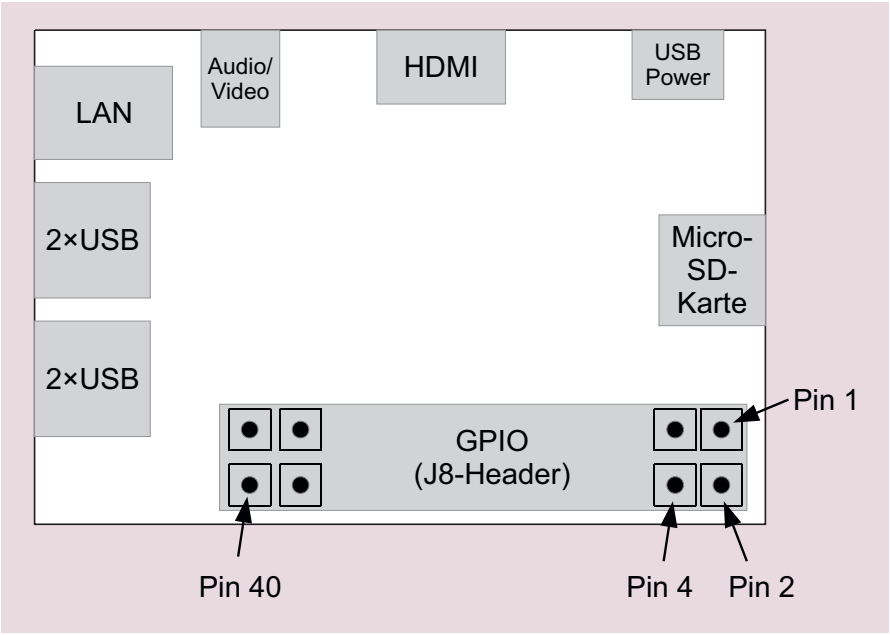


Abbildung 1.4 Schematische Darstellung der Raspberry-Pi-Anschlüsse (Raspberry Pi 2 und 3, jeweils Modell B, bzw. Raspberry Pi 1, Modell B+, Sicht von oben)

Die Anschlüsse des Raspberry Pi 2 (Modell B) und des Raspberry Pi 1 (Modell B+)

Die Modelle RPi2-B und RPi1-B+ verfügen prinzipiell über dieselben Anschlüsse wie der RPi3-B. Es gibt aber zwei Unterschiede:

- Die WLAN- und Bluetooth-Adapter fehlen.
- Die USB-Anschlüsse können standardmäßig nur 600 mA weitergeben. Wenn Sie energiehungrige USB-Geräte mit Strom versorgen möchten, müssen Sie in `/boot/config.txt` die Einstellung `usb_max_current=1` einbauen (siehe auch Abschnitt 5.8, »Systemeinstellungen in config.txt«). Der maximal zulässige Strom beträgt dann wie beim RPi3-B 1200 mA.

Die Anschlüsse anderer Raspberry-Pi-Modelle

Für das ältere Modell B des Raspberry Pi 1, das zwar nicht mehr verkauft wird, aber noch vielfach im Einsatz ist, gelten einige Abweichungen im Vergleich zur obigen Liste:

- Strombedarf 700 mA bis 1 A (ergibt 3,5 bis 5 Watt Leistung)
- nur zwei USB-Anschlüsse mit einem maximalen Ausgangsstrom von je 100 mA
- SD-Karten-Slot für gewöhnliche SD-Karten
- getrennte Audio- und Video-Ausgänge
- eine Steckerleiste mit nur 26 Pins (der sogenannte »P1-Header«)
- andere Anordnung der Anschlüsse, erfordert daher auch ein anderes Gehäuse als bei den neueren Modellen

Bei den Modellen A bzw. A+ müssen Sie auf den Ethernet-Anschluss verzichten. Außerdem gibt es nur einen USB-Anschluss.

Diese Einschränkungen gelten auch für das Zero-Modell. Dort kommt erschwerend hinzu, dass die USB- und HDMI-Anschlüsse in der Micro- bzw. Mini-Ausführung vorliegen. Zum Anschluss gewöhnlicher USB- oder HDMI-Kabel benötigen Sie Adapter.

Der analoge Audio-Ausgang fehlt beim Zero-Modell komplett. Den Composite-Video-Ausgang können Sie zur Not selbst realisieren – dafür gibt es entsprechende Lötpunkte. Dafür ist das Zero-Modell besonders sparsam und begnügt sich mit weniger als einem Watt Leistungsaufnahme.

Netzteil

Das Netzteil ist entscheidend dafür, dass der Raspberry Pi stabil und zuverlässig funktioniert. Achten Sie beim Kauf des Netzteils darauf, dass dieses ausreichend leistungsstark ist. Das Modell B des Raspberry Pi 3 benötigt zumindest 1000 mA Strom, das ergibt bei einer Spannung von 5 V eine Leistungsaufnahme von 5 W. (Im Leerlauf benötigt dieses Modell nur ca. 2,5 W. Die Leistungsaufnahme steigt aber, wenn die CPU ausgelastet wird.) Beim Modell B des Raspberry Pi 2 ist die maximale Grundlast mit 600 mA etwas niedriger.

Dazu kommt noch der Strombedarf der angeschlossenen USB-Geräte sowie anderer Komponenten, die mit dem Raspberry Pi verbunden sind: So dürfen über GPIO-Pins bis zu 50 mA fließen. Wenn der Raspberry Pi über ein HDMI-Kabel mit einem Monitor verbunden ist, kostet das ca. 50 mA Strom. Das Kameramodul für den Raspberry Pi benötigt weitere 250 mA.

Entscheiden Sie sich also für ein Netzteil, das zumindest 1000 mA (also 5 W) zur Verfügung stellen kann. Beim RPi3-B, RPi2-B bzw. RPi1-B+ kann der Gesamtstrom auf über 2000 mA ansteigen (also auf mehr als 10 W), wenn Sie viele bzw. leistungsstarke USB-Geräte nutzen. Typische Handy-Netzteile sind ungeeignet, auch wenn diese vielleicht mit dem richtigen USB-Kabel ausgestattet sind und die Verlockung daher groß ist, das Netzteil einer neuen Verwendung zuzuführen! Die Raspberry Pi Foundation empfiehlt für den RPi3-B generell ein Netzteil von 2,5 A, was nach unseren Erfahrungen aber übertrieben ist.

Grundsätzlich ist der Raspberry Pi für den Dauerbetrieb ausgelegt. Viele Raspberry-Pi-Anwendungen setzen voraus, dass der Raspberry Pi Tag und Nacht läuft. Glücklicherweise verbraucht der Raspberry Pi dabei nur etwas mehr Strom als viele andere Geräte im Stand-by-Betrieb. Dennoch summiert sich der Strombedarf über ein Jahr gerechnet auf rund 35 bis 50 Kilowattstunden. Bei einem Strompreis von 20 Cent/kWh betragen die Stromkosten für den Raspberry Pi (ohne Zusatzgeräte) also rund 6 bis 10 Euro pro Jahr.

Akku- und Batteriebetrieb

Im Vergleich zu einem gewöhnlichen Computer verbraucht der Raspberry Pi zwar nur wenig Strom, für den Akku- oder Batteriebetrieb ist die Leistungsaufnahme aber dennoch recht hoch. Tipps, wie Sie Ihren Raspberry Pi zumindest etliche Stunden lang ohne Netzanschluss betreiben können, finden Sie in Abschnitt 11.4, »Stromversorgung«. Für besonders energieeffiziente Anwendungen empfiehlt sich das Zero-Modell mit weniger als 1 Watt Leistungsaufnahme. Im Leerlaufbetrieb und ohne HDMI- und USB-Geräte beträgt der Energiebedarf sogar nur ein halbes Watt.

Ein/Aus-Schalter

Allen Raspberry-Pi-Modellen fehlt ein Ein/Aus-Schalter. Zum Einschalten stecken Sie das Micro-USB-Kabel zur Stromversorgung an. Um den Raspberry Pi auszuschalten, fahren Sie nach Möglichkeit zuerst das laufende Betriebssystem herunter, z. B. durch Abmelden im Startmenü oder mit dem Kommando `halt`. Anschließend lösen Sie das Micro-USB-Kabel für die Stromversorgung. Eine Anleitung, wie Sie Ihren Raspberry Pi über einen Taster ausschalten oder neu starten können, finden Sie in Abschnitt 20.3, »Reset/Shutdown-Taste«.

SD-Karte

Der Raspberry Pi verfügt nicht wie ein gewöhnlicher Computer über eine Festplatte oder eine SSD. Stattdessen dient eine SD-Karte als Datenspeicher für das Betriebssystem sowie für Ihre Daten. Die Form der SD-Karte hängt vom Modell ab:

- **Raspberry Pi 3, Raspberry Pi 2, Raspberry Pi Zero sowie Raspberry Pi 1, Modelle A+ und B+:** Für alle aktuellen Modelle des Raspberry Pi brauchen Sie eine Micro-SD-Karte. In der Regel ist es zweckmäßig, eine Micro-SD-Karte mit einem Adapter für das Standardformat zu erwerben. Den Adapter benötigen Sie, damit Sie die Micro-SD-Karte in den SD-Karten-Slot Ihres gewöhnlichen Computers einführen und dort beschreiben können.
- **Raspberry Pi 1, Modelle A und B:** Die älteren Raspberry-Pi-Modelle erwarten eine SD-Karte im Standardformat. Mini- oder Micro-SD-Karten können mit einem Adapter verwendet werden.

Unabhängig vom Format muss die SD-Karte dem SDHC-Standard entsprechen. Der neuere SDXC-Standard für SD-Karten mit mehr als 32 GByte wird offiziell nicht unterstützt! Tatsächlich können auch derartige SD-Karten verwendet werden, sofern Sie auf die NOOBS-Installationsvariante verzichten oder sicherstellen, dass Sie ein VFAT-Dateisystem (nicht ExFAT) für die Installation verwenden.

Probleme mit SD-Karten

Wenn man den diversen Raspberry-Pi-Diskussionsforen trauen kann, sind defekte SD-Karten die häufigste Fehlerursache auf dem Raspberry Pi. Das hat sich auch bei unseren Tests bestätigt. Kümmern Sie sich regelmäßig um Backups Ihrer Daten, und halten Sie für den Notfall eine SD-Reservekarte bereit.

Auch wenn wir diesbezüglich keine negativen Erfahrungen gemacht haben, existieren offensichtlich auch vereinzelt SD-Karten, die inkompatibel zum Raspberry Pi sind und überhaupt nicht funktionieren. Informationen zu diesem Problem können Sie auf der folgenden Seite nachlesen. Dort finden Sie in einer Art Datenbank unzählige Erfahrungsberichte zu diversen SD-Karten:

http://elinux.org/RPi_SD_cards

SD-Karten gibt es in unterschiedlichen Geschwindigkeitsklassen – Class 4, 6 oder 10. Class 6 bedeutet beispielsweise, dass eine Schreibgeschwindigkeit von zumindest 6 MByte pro Sekunde garantiert wird. Das klingt gut, ist aber weniger als ein Zehntel dessen, was bei Festplatten üblich ist, von SSDs gar nicht zu sprechen! Wenn Sie also Wert auf einen zügigen Start des Raspberry Pi legen bzw. häufig größere Datenmengen lesen oder schreiben möchten, sollten Sie eine möglichst schnelle SD-Karte verwenden, also Class 10.

Bleibt noch die optimale Größe der SD-Karte zu klären: Wenn Sie Raspbian einsetzen möchten, also das gängigste Linux-System für den Raspberry Pi, dann beträgt das unterste Limit 4 GByte. Besser ist es, Sie entscheiden sich gleich für etwas mehr Speicherplatz, z. B. für 8 GByte.

Gehäuse

Für Versuchsaufbauten auf Ihrem Schreibtisch können Sie auf ein Gehäuse verzichten. Sollten Sie aber vorhaben, Ihren Raspberry Pi im Rahmen eines Projekts dauerhauft einzusetzen (beispielsweise als Multimedia-Center im Wohnzimmer), ist ein Gehäuse empfehlenswert.

Im Internet gibt es eine große Auswahl an Gehäusen für den Raspberry Pi. Beim Kauf müssen Sie unbedingt darauf Rücksicht nehmen, welches Raspberry-Pi-Modell Sie einsetzen. Achten Sie auch darauf, dass das Gehäuse Belüftungsschlitze aufweist! Der Raspberry Pi läuft mangels Lüfter und anderer bewegter Teile vollkommen lautlos, produziert aber durchaus Abwärme. In einem Gehäuse ohne Luftzirkulation riskieren Sie ein vorzeitiges Ableben Ihres neuen Gadgets!

Sofern die Belüftung gewährleistet ist, benötigt der Raspberry Pi für den normalen Betrieb keine Kühlung. Es besteht allerdings die Möglichkeit, die CPU höher zu takten und damit die Geschwindigkeit des Raspberry Pi zu steigern (siehe Abschnitt 4.13, »Overclocking«). Sollten Sie sich dazu entschließen, ist es empfehlenswert, die CPU, den USB/LAN-Controller und den Spannungswandler mit passiven Kühlkörpern auszustatten. Diese leiten die Wärme besser ab.

Tastatur und Maus

Nahezu jede handelsübliche USB-Tastatur und -Maus eignet sich als Eingabegerät für den Raspberry Pi. Bei den Raspberry-Pi-Modellen A und B der Version 1 müssen Sie allerdings darauf achten, dass der Strombedarf jeweils nicht mehr als 100 mA betragen darf: Falls Ihre Tastatur bzw. Maus mehr Strom brauchen, müssen Sie die Geräte über einen aktiven USB-Hub mit dem Raspberry Pi verbinden.

Möglicherweise fragen Sie sich, wie Sie herausfinden, wie groß die Leistungsaufnahme Ihrer Tastatur bzw. Maus ist. Ein entsprechend genaues Datenblatt steht leider selten zur Verfügung. Gewissheit können Sie nur durch Ausprobieren oder mit einem USB-Strommessgerät erlangen. Sollte Ihr Raspberry Pi nicht stabil laufen bzw. sollten Tastatur und Maus gar nicht funktionieren, dann wird Ihre erste Zusatzinvestition ein aktiver USB-Hub sein.

Persönlich haben wir für unsere Experimente unter anderem eine schon etwas ältere Apple-Aluminium-Tastatur mit USB-Anschluss und eine preisgünstige Logitech-OEM-Maus verwendet.

Längerfristig können Sie den Raspberry Pi natürlich auch mit einer Bluetooth-Maus und -Tastatur steuern. Das erfordert aber einen USB-Bluetooth-Adapter (außer beim RPi3-B) sowie die oft hakelige Konfiguration der Bluetooth-Geräte. Tipps zur Bluetooth-Konfiguration folgen in Abschnitt 2.5, »Bluetooth-Konfiguration«.

USB-Hub

Wir haben es bereits erwähnt: Die älteren Modelle des Raspberry Pi verfügen über nur zwei USB-2.0-Anschlüsse. Noch größere Einschränkungen gelten für die A-Modelle sowie für den Raspberry Pi Zero, die nur einen einzigen USB-Anschluss vorsehen.

Aber nicht nur die Anzahl der USB-Anschlüsse ist limitiert, sondern auch der Strom, den der Raspberry Pi den USB-Geräten liefern kann. Ältere Raspberry-Pi-Modelle können USB-Geräten maximal 100 mA Strom zur Verfügung stellen – und das auch nur, wenn das Netzteil für den Raspberry Pi korrekt bemessen ist. Bei einer Spannung von 5 V ergibt sich daraus eine maximal zulässige Leistungsaufnahme von 0,5 W pro Gerät. Für viele USB-Geräte ist das zu wenig. Das gilt insbesondere für externe Festplatten, aber auch für manche Tastaturen und WLAN-Adapter.

Leider ist es nahezu unmöglich, den Strombedarf bzw. die Leistungsaufnahme eines USB-Geräts vor dem Kauf in Erfahrung zu bringen. Sie können den Erfahrungsberichten anderer Raspberry-Pi-Anwender vertrauen, eine wirklich zuverlässige Informationsquelle ist das aber nicht. Oder Sie probieren es einfach selbst aus: Wenn das USB-Gerät funktioniert und Ihr Raspberry Pi danach problemlos startet und über längere Zeit absturzfrei läuft, ist alles in Ordnung.

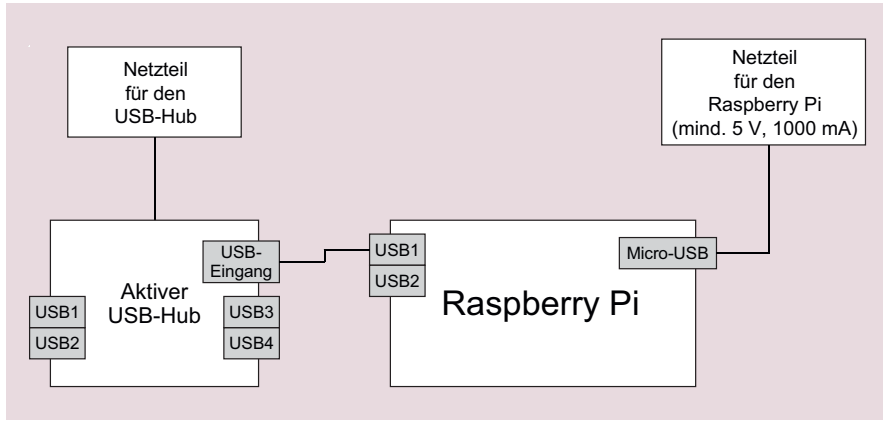


Abbildung 1.5 Raspberry Pi 1, Modell B, mit aktivem USB-Hub zur gleichzeitigen Verwendung von fünf USB-Geräten

Bei den aktuellen Modellen ist die USB-Situation wesentlich entspannter: Sie können bis zu vier Geräte gleichzeitig anschließen und diese mit insgesamt bis zu 1200 mA

Strom versorgen. Das setzt aber ein entsprechend großzügig dimensioniertes Netzteil voraus, denn der Raspberry Pi selbst benötigt ebenfalls mindestens 600 mA. Mit angeschlossenem HDMI-Monitor und einer Kamera braucht er sogar bis zu 900 mA.

Wenn Sie viele energiehungrige USB-Geräte gleichzeitig verwenden möchten, führt an einem aktiven USB-Hub kein Weg vorbei (siehe Abbildung 1.5). *Aktiv* bedeutet in diesem Fall, dass der USB-Hub über eine eigene Stromversorgung verfügt.

WLAN- und Bluetooth-Adapter

Der Raspberry Pi 3, Modell B, enthält integrierte WLAN- und Bluetooth-Adapter. Wenn Sie andere Raspberry-Pi-Modelle per Funk in das lokale Netzwerk integrieren möchten oder Bluetooth-Geräte nutzen möchten, benötigen Sie einen USB-WLAN- bzw. USB-Bluetooth-Adapter. Diese winzigen USB-Stöpsel sind für wenige Euro erhältlich. Konfigurationsanleitungen finden Sie in Abschnitt 5.5, »Netzwerkkonfiguration«, und in Abschnitt 5.6, »Bluetooth«. Beachten Sie aber, dass nicht jedes Gerät kompatibel zum Raspberry Pi ist! Recherchieren Sie unbedingt vor dem Kauf, z. B. auf den folgenden Seiten:

http://elinux.org/RPi_USB_Wi-Fi_Adapters
http://elinux.org/RPi_USB_Bluetooth_adapters

Was Sie sonst noch brauchen

Der Raspberry Pi ist zwar ein selbstständiger Computer, um ihn in Betrieb zu nehmen, benötigen Sie aber einen zweiten Computer: Dort laden Sie die Image-Datei mit dem Betriebssystem des Raspberry Pi herunter und übertragen das Image auf die SD-Karte. Dieser Vorgang wird im nächsten Abschnitt ausführlich beschrieben. Sollte Ihr Hauptcomputer über keinen SD-Slot verfügen, müssen Sie sich ein USB-SD-Karten-Lesegerät besorgen, das Sie für wenige Euro in jedem Elektronik-Shop erhalten.

Auch für den weiteren Betrieb ist ein regulärer Computer hilfreich: Sobald auf Ihrem Raspberry Pi Linux läuft, können Sie die meisten Administrationsaufgaben auch über eine Netzwerkverbindung erledigen. Diese Vorgehensweise ist oft viel komfortabler als das direkte Arbeiten auf dem Raspberry Pi.

Für erste Experimente ist es ausgesprochen praktisch, wenn Sie Ihren Raspberry Pi über ein Netzwerkkabel mit dem lokalen Netzwerk, also z. B. mit Ihrem Router, verbinden können. Damit hat Ihr Minicomputer ohne umständliche Konfigurationsarbeiten sofort Netzwerk- und Internetzugang. Das ist unter anderem auch deswegen zweckmäßig, weil der Raspberry Pi über keine batteriegepufferte Uhr verfügt und die aktuelle Uhrzeit aus dem Netzwerk bezieht. Mit der manchmal hakeligen WLAN-Konfiguration sollten Sie warten, bis Sie mit der Nutzung des Raspberry Pi vertraut sind.

Wenn Sie den Raspberry Pi für Elektronikprojekte einsetzen, benötigen Sie dazu natürlich die entsprechenden Bauteile, außerdem ein Multimeter, ein Steckboard für Versuchsaufbauten etc. Detaillierte Anleitungen für alle erdenklichen Einsatzzwecke folgen im dritten Teil dieses Buchs.

1.2 Raspberry-Pi-Distributionen

Der Raspberry Pi wird ohne Betriebssystem ausgeliefert. Bevor Sie mit ihm arbeiten können, müssen Sie sich für ein Betriebssystem entscheiden: Für den Raspberry Pi gibt es nämlich nicht nur eines, sondern es steht gleich eine ganze Menge von Betriebssystemen zur Auswahl. Nahezu alle diese Betriebssysteme basieren auf Linux. In der Linux-Welt ist es üblich, das eigentliche Betriebssystem sowie alle dafür verfügbaren Programme als *Distribution* zu bezeichnen. Die folgende Liste zählt die wichtigsten Distributionen auf, die für den Raspberry Pi geeignet sind:

- **Raspbian:** Raspbian ist die populärste Linux-Distribution für den Raspberry Pi. Das Wortgebilde *Raspbian* setzt sich aus »Raspberry Pi« und »Debian« zusammen. Fast alle Kapitel dieses Buchs beziehen sich auf Raspbian. Auch im Internet setzen fast alle Anleitungen und Tipps voraus, dass Sie Raspbian verwenden. Diverse Raspberry-Pi-Zusatzpakete stehen ausschließlich für Raspbian zur Verfügung (z. B. Mathematica) bzw. müssen beim Einsatz anderer Distributionen extra kompiliert werden. Für Raspberry-Pi-Einsteiger gibt es somit keinen plausiblen Grund, eine andere Distribution zu verwenden.
- **Arch Linux ARM:** Die Arch-Linux-Distribution richtet sich an fortgeschrittene Linux-Anwender. Die ARM-Version ist eine Arch-Variante, die speziell für Minicomputer wie den Raspberry Pi und das BeagleBoard optimiert wurde.
- **RISC OS:** RISC OS ist ein Betriebssystem, das von der englischen Firma Acorn entwickelt und in seiner ersten Version 1987 freigegeben wurde. Das Betriebssystem ist vor allem für IT-Historiker interessant. RISC OS basiert nicht auf Linux.
- **Ubuntu:** Die Raspberry-Pi-Modelle ab der Version 2 sind auch mit Ubuntu kompatibel. Allerdings ist nicht jede der vielen Ubuntu-Varianten für den Betrieb auf dem Raspberry Pi geeignet. Relativ gute Erfahrungen haben wir mit Ubuntu MATE gemacht (siehe Kapitel 6, »Ubuntu«).
- **Windows 10:** Etwas überraschend ist auch Microsoft auf den Raspberry-Pi-Zug aufgesprungen und bietet die kostenlose Windows-Version »Windows 10 IoT Core« an. Wie Ubuntu setzt auch Windows zumindest die Version 2 des Raspberry Pi voraus. Im Vergleich zu Linux ist der Windows-Betrieb allerdings recht umständlich, mit vielen Einschränkungen verbunden und nur für Entwickler mit Visual-Studio-Erfahrung gedacht. Lesen Sie mehr zu diesem Thema in Kapitel 7.

- **Volumio und Pi Musicbox:** Diese beiden Distributionen machen aus Ihrem Raspberry Pi einen Audio-Player für Ihre Stereoanlage. Beide Distributionen werden über einen Webbrowser bedient, z. B. auf dem Smartphone im WLAN zu Hause. Eine kurze Beschreibung finden Sie in Kapitel 8, »Audio-Player mit Smartphone-Fernbedienung«.
- **OpenELEC, OSMC und RasPlex:** Diese Distributionen sind speziell dazu gedacht, aus Ihrem Raspberry Pi ein Multimedia-Center zu machen. OpenELEC und RasPlex beschreiben wir im Detail in Kapitel 9, »Multimedia-Center mit Kodi und OpenELEC«, bzw. in Kapitel 10, »Multimedia-System mit Plex«.

Eine eindrucksvolle Liste mit rund 50 für den Raspberry Pi geeigneten Distributionen finden Sie hier:

http://elinux.org/RPi_Distributions

Allerdings ist nicht jede der dort aufgeführten Distributionen so ausgereift wie Raspbian. Ein Teil der genannten Distributionen wird schon jetzt nicht mehr gewartet, und es ist zu befürchten, dass dies in Zukunft sogar für die Mehrheit der genannten Distributionen zutrifft. Wenn Sie sich für eine derartige Distribution entscheiden, werden Sie nicht mit Sicherheits-Updates und Bugfixes versorgt.

1.3 NOOBS-Installation

Die Installation eines Betriebssystems für den Raspberry Pi erfolgt anders als auf gewöhnlichen Computern: Der Raspberry Pi verfügt über kein CD/DVD-Laufwerk, das zur Installation verwendet werden könnte, und auch das Booten über einen USB-Stick samt Installationsprogramm ist nicht vorgesehen.

Stattdessen müssen Sie die für den Raspberry Pi vorgesehene SD-Karte auf Ihrem regulären Notebook oder Desktop-Computer vorbereiten. Dazu gibt es zwei grundlegend unterschiedliche Vorgehensweisen: Entweder kopieren Sie die Dateien des in diesem Abschnitt beschriebenen NOOBS-Installationsprogramms direkt auf die SD-Karte, oder Sie laden sich eine sogenannte Image-Datei Ihrer Lieblingsdistribution herunter und schreiben diese auf die SD-Karte. Der Umgang mit Image-Dateien ist ein wenig komplizierter und wird im nächsten Abschnitt ausführlich erklärt.

Wie groß ist groß genug?

Zur NOOBS-Installation von Raspbian benötigen Sie eine zumindest 8 GByte große SD-Karte. Ohne NOOBS reichen zur Not 4 GByte aus, bei den meisten Multimedia-Distributionen funktioniert sogar eine noch kleinere SD-Karte. Eine großzügig dimensionierte SD-Karte gibt Ihnen aber mehr Spielraum, um später eigene Dateien, Filme oder Audio-Dateien direkt auf dem Raspberry Pi zu speichern.

SD-Karte formatieren

Bevor Sie auf Ihrem Notebook oder PC die Installationsdateien oder ein Image auf eine SD-Karte schreiben, müssen Sie die Karte formatieren. Das klingt nach einer trivialen Aufgabe, tatsächlich bereitet das Formatieren von SD-Karten aber überraschend viele Schwierigkeiten. Mit den Bordmitteln von Windows, OS X und Linux gehen Sie so vor:

- **Windows:** Unter Windows klicken Sie die SD-Karte (Wechseldatenträger) im Windows Explorer mit der rechten Maustaste an und führen Formatieren aus. Als Dateisystem verwenden Sie FAT32 (Standard).
- **OS X:** Unter OS X starten Sie das Festplattendienstprogramm, wählen die SD-Karte aus, wechseln in das Dialogblatt Löschen und klicken dort auf den gleichnamigen Button. Als Dateisystem verwenden Sie MS-DOS-Dateisystem (FAT).
- **Linux:** Unter Linux formatieren Sie die SD-Karte am einfachsten in einem Terminalfenster. Dazu stellen Sie zuerst mit `mount` sicher, dass momentan keines der auf der SD-Karte befindlichen Dateisysteme verwendet wird. Gegebenenfalls lösen Sie diese Dateisysteme mit `umount verzeichnis`.

  Außerdem müssen Sie den Device-Namen Ihrer SD-Karte feststellen. Dazu nehmen Sie das Kommando `lsblk` zu Hilfe. Es gibt einen Überblick über die Device-Namen aller Festplatten, SSDs, USB-Sticks und SD-Karten. Anhand der Größe lässt sich die SD-Karte in der Regel eindeutig ermitteln. Oft wird der Device-Name `sdb` oder `sdc` lauten, unter Umständen auch `mmcblk0`. Nach diesen Vorbereitungsarbeiten führen Sie drei Kommandos aus:

  ```
  parted /dev/xxx mklabel msdos
  parted /dev/xxx 'mkpart primary fat32 1MiB -1MiB'
  mkfs.vfat -F 32 /dev/xxxyy
  ```

  Mit dem ersten `parted`-Kommando erzeugen Sie eine neue Partitionstabelle auf der SD-Karte. Das zweite `parted`-Kommando legt eine Partition an. `mkfs.vfat` richtet darin ein FAT-Dateisystem ein. Bei den beiden `parted`-Kommandos geben Sie anstelle von `xxx` den Device-Namen der SD-Karte an. An `mkfs.vfat` übergeben Sie den Device-Namen der neuen Partition. Dieser lautet z. B. `/dev/sdc1` oder `/dev/mmcblk0p1`. Wenn Sie sich unsicher sind, rufen Sie vorher nochmals `lsblk` auf. Sollte Linux das Kommando `parted` nicht kennen, installieren Sie vorher das gleichnamige Paket.

Mitunter treten beim Formatieren Probleme auf, insbesondere dann, wenn die SD-Karte bereits für eine Raspberry-Pi-Installation verwendet wurde und daher Linux-Partitionen enthält, die Windows- oder OS-X-Rechner nicht erkennen. Viele Raspberry-Pi-Webseiten empfehlen deshalb, anstelle der Formatierwerkzeuge Ihres Betriebssystems das Formatierprogramm der *SD Association* (siehe Abbildung 1.6)

einzusetzen. Dieses Programm steht für Windows und OS X auf den folgenden Seiten kostenlos zum Download zur Verfügung:

https://www.sdcard.org/downloads/formatter_4/eula_windows
https://www.sdcard.org/downloads/formatter_4/eula_mac

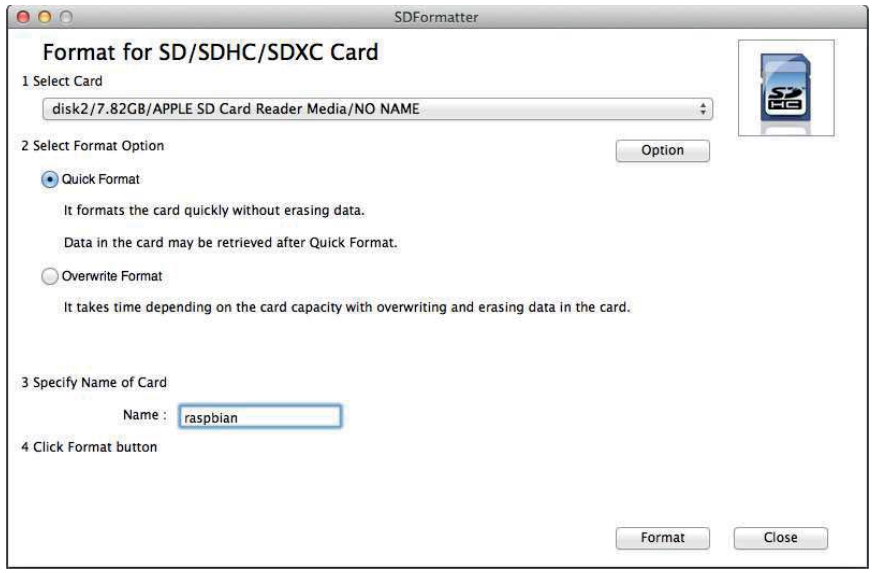


Abbildung 1.6 Das »SDFormatter«-Programm unter OS X

SD-Karten mit mehr als 32 GByte

Eigentlich unterstützt der Raspberry Pi nur SD-Karten bis zu 32 GByte. Tatsächlich läuft Raspbian aber auch auf größeren SD-Karten. Aufpassen müssen Sie aber bei der Installation: Wenn Sie eine NOOBS-Installation durchführen möchten, dann muss die SD-Karte ein VFAT-Dateisystem enthalten. Der SDXC-Standard sieht für SD-Karten mit mehr als 32 GByte aber das ExFAT-Dateisystem vor, und sowohl Windows als auch das Formatierprogramm der *SD Association* erzwingt dieses Format auch. (Sie haben keine Wahlmöglichkeit.)

Um dennoch eine NOOBS-Installation durchzuführen, müssen Sie unter Windows ein Formatierprogramm verwenden, das auch bei großen SD-Karten VFAT unterstützt. Bewährt hat sich bei unseren Tests `guiformat.exe` (siehe auch Kapitel 41, »DLNA-Server«):

http://www.ridgecrop.demon.co.uk/index.htm?guiformat.htm

USB-SD-Card-Reader

Die meisten Notebooks besitzen einen Slot für SD-Karten in Standardgröße. Bei Micro-SD-Karten müssen Sie in der Regel einen SD-Kartenadapter verwenden, der bei vielen Micro-SD-Karten gleich mitgeliefert wird.

Sollten Sie Ihre SD-Karte auf einem Rechner formatieren bzw. beschreiben wollen, der über keinen Slot für eine SD-Karte verfügt, benötigen Sie einen SD-Karten-Reader. Mit diesen mitunter winzigen Geräten können Sie SD-Karten via USB ansteuern (siehe Abbildung 1.7).

Abbildung 1.7 Eine SD-Karte in Standardgröße, eine Micro-SD-Karte, ein SD-Karten-Adapter sowie ein winziger USB-Adapter für Micro-SD-Karten

Ärger mit dem Schreibschutz

Auf SD-Karten befindet sich am linken Rand ein winziger mechanischer Schalter, um den Schreibschutz zu aktivieren. Wenn der Schreibschutz aktiv ist und der SD-Slot diesen Zustand auch berücksichtigt, kann der Inhalt der Karte nur gelesen, aber nicht verändert werden. Naturgemäß kann die Karte dann auch nicht formatiert werden. Werfen Sie bei entsprechenden Fehlermeldungen also einen Blick auf diesen Schalter. Damit die SD-Karte verändert werden darf, muss sich der Schalter oben befinden, also in der Nähe der Kontaktleiste.

Einer unserer Testrechner, ein sechs Jahre alter iMac von Apple, hatte offensichtlich Probleme mit der korrekten Erkennung dieses Read-only-Schalters. Immer wieder meldete das Festplattendienstprogramm, die SD-Karte könne nur beschrieben, aber nicht verändert werden, obwohl sich der Read-only-Schalter an der richtigen Position befand. Abhilfe: Nach mehrmaligem Ein- und Ausstecken der SD-Karte ist es uns letztlich immer wieder gelungen, die offensichtlich defekte Mechanik des Rechners zu überlisten.

NOOBS

NOOBS (*New Out Of Box Software*) ist keine Raspberry-Pi-Distribution, sondern vielmehr eine Sammlung von Installationsdateien, die auf eine leere, vorher formatierte

SD-Karte kopiert werden. Beim ersten Start von der SD-Karte können Sie eine oder mehrere Distributionen auswählen und installieren. Momentan umfasst die Palette Raspbian, Arch, OpenELEC, OSMC, Pidora und RISC OS. Allerdings sind nur für Raspbian die Installationsdateien direkt enthalten. Alle anderen Distributionen werden aus dem Internet heruntergeladen – und nur dann angezeigt, wenn beim Start des Raspberry Pi eine Internetverbindung besteht.

NOOBS richtet sich speziell an Raspberry-Pi-Einsteiger. Der größte Vorteil von NOOBS besteht darin, dass es sich nicht um eine Image-Datei handelt. Das vereinfacht das Einrichten der SD-Karte erheblich. NOOBS kann als ZIP-Datei von der folgenden Webseite heruntergeladen werden:

http://www.raspberrypi.org/downloads

Beim Download haben Sie die Wahl zwischen zwei Versionen:

- **Standard-Version:** Die rund 1 GByte große Offline-Version enthält die Installationsdateien für Raspbian. Dessen Installation kann dann ohne Netzwerkverbindung durchgeführt werden. Da der Inhalt des Installationsprogramms in Form einer Recovery-Partition auf der SD-Karte verbleibt, sollten Sie die Standard-Version nur verwenden, wenn Ihre SD-Karte zumindest 8 GByte groß ist.
- **NOOBS Lite:** Die Netzwerkversion umfasst nur bescheidene 30 MByte. Das reicht gerade aus, um den Raspberry Pi zu booten und ein Menü anzuzeigen, über das man die Installation einer Raspberry-Pi-Distribution starten kann. Die erforderlichen Installationsdateien werden danach aus dem Internet heruntergeladen. Das funktioniert allerdings nur dann, wenn Sie Ihren Raspberry Pi mit einem Netzwerkkabel an das lokale Netzwerk anschließen.

Egal für welche Variante Sie sich entscheiden, Sie müssen nun die SD-Karte formatieren und dann alle Dateien aus der heruntergeladenen ZIP-Datei auf die SD-Karte kopieren. Stellen Sie sicher, dass die Dateien `recovery.*` direkt auf der SD-Karte gespeichert werden, nicht in einem Unterverzeichnis! Denken Sie daran, im Dateimanager die SD-Karte per Kontextmenü auszuwerfen, bevor Sie die SD-Karte aus dem Slot entfernen.

Nach diesen Vorbereitungsarbeiten schließen Sie Ihren Raspberry Pi an einen Monitor an, verbinden Maus und Tastatur und stecken die SD-Karte mit den Kontakten nach oben in den SD-Slot. Wenn möglich, verbinden Sie den Raspberry Pi außerdem über ein Netzwerkkabel mit einem Switch/Hub im lokalen Netzwerk.

Erst nachdem Sie alle anderen Kabel verbunden haben, stecken Sie auch das Micro-USB-Kabel der Stromversorgung an. Auf dem Bildschirm sollte nun für circa zwei Sekunden ein buntes Quadrat angezeigt werden. Wenige Sekunden später erscheint das NOOBS-Fenster, in dem Sie die Sprache, das Tastaturlayout und das zu installierende Betriebssystem auswählen (siehe Abbildung 1.8).

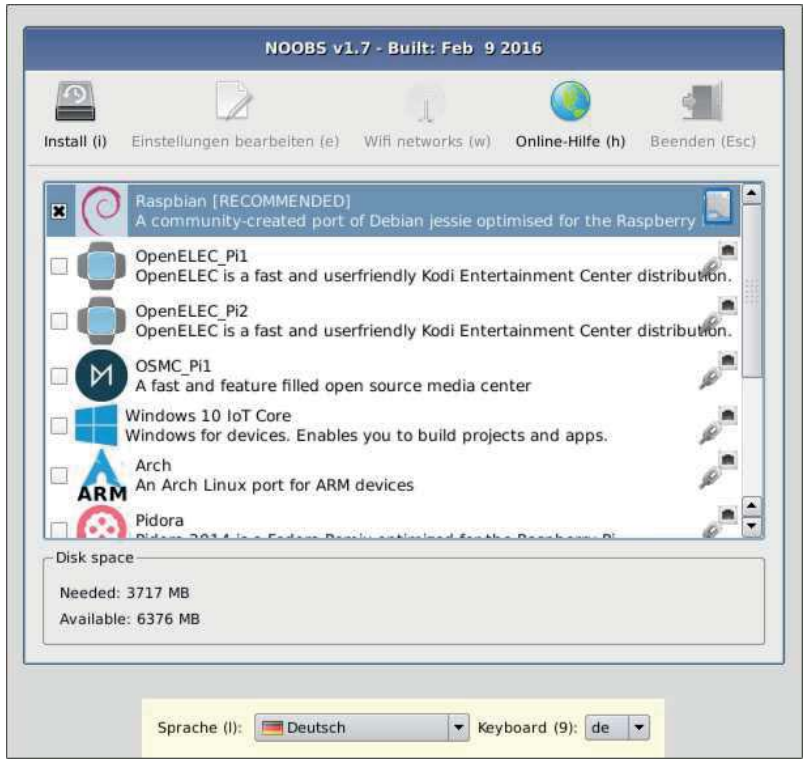


Abbildung 1.8 Das NOOBS-Installationsprogramm

In der Regel werden Sie im NOOBS-Menü nur den ersten Eintrag, RASPBIAN, auswählen. Fortgeschrittene Linux-Anwender werden vielleicht an der Zusatzoption DATA PARTITION Gefallen finden: Ist diese Option aktiv, dann richtet NOOBS während der Installation eine zweite, 512 MByte große Partition mit einem `ext4`-Dateisystem ein. Dieses Dateisystem können Sie dann z. B. als Datenspeicher verwenden, der unabhängig von der Systempartition ist. Die Nutzung dieser Partition erfordert allerdings etwas Linux-Know-how. Außerdem verringert sich die Größe der Systempartition um ein halbes GByte.

Grundsätzlich ist es auch möglich, mehrere Distributionen auf einmal zu installieren. In diesem Fall erscheint jedes Mal beim Start des Raspberry Pi ein Boot-Menü, in dem Sie das zu startende Betriebssystem auswählen. Parallelinstallationen haben allerdings den Nachteil, dass sich alle Betriebssysteme den Platz auf der Festplatte teilen. Es ist nachträglich nicht ohne Weiteres möglich, ein Betriebssystem zu entfernen und den freien Platz einem anderen Betriebssystem zuzuweisen. Deswegen raten wir Ihnen von Mehrfachinstallationen ab. Wenn Sie ein anderes Betriebssystem ausprobieren möchten, ist es besser, dafür eine zweite oder dritte SD-Karte zu verwenden.

Varianten für den Raspberry Pi 1, 2 und 3

Beachten Sie, dass es bei einigen Distributionen mehrere Varianten gibt, die mit Pi1, Pi2 oder Pi3 gekennzeichnet sind. Dabei handelt es sich um Distributionen, die speziell für ein Prozessormodell optimiert sind. Sie laufen ausschließlich auf der dafür vorgesehenen Raspberry-Pi-Version. Achten Sie darauf, dass Sie die Variante auswählen, die zu Ihrem Modell passt!

Mit dem Button INSTALL starten Sie nun die Installation. Während der Installation, die für Raspbian circa eine Viertelstunde dauert, zeigt das Installationsprogramm einen Fortschrittsbalken (siehe Abbildung 1.9).

Abbildung 1.9 Statusanzeige während der Installation

Nach Abschluss der Installation erscheint auf dem Bildschirm die Nachricht OS(ES) INSTALLED SUCCESSFULLY. Sobald Sie diese Meldung mit OK bestätigen, wird der Raspberry Pi neu gestartet.

Die grafische Benutzeroberfläche erscheint anfänglich mit englischen Dialogen. Zur Spracheinstellung und zur weiteren Konfiguration führen Sie im Startmenü PREFERENCES • RASPBERRY PI CONFIGURATION aus. Im Detail wird dieses Konfigurationsprogramm in Kapitel 2, »Erste Schritte in Raspbian«, beschrieben. Dort stellen wir Ihnen außerdem Raspbian näher vor und helfen Ihnen, das Betriebssystem schnell kennenzulernen.

Wenn etwas schiefgeht

Sollte die Installation aus irgendeinem Grund scheitern, können Sie jederzeit von vorne beginnen. Sie schalten also den Raspberry Pi aus, stecken die SD-Karte wieder in Ihr Notebook oder in Ihren PC, formatieren die Karte neu und kopieren dann nochmals den Inhalt der NOOBS-ZIP-Datei dorthin.

Schwieriger wird es, wenn es Hardware-Probleme gibt, d. h., wenn Sie z. B. kein stabiles Bild auf dem Monitor sehen, der Monitor nur 640×480 Pixel anzeigt oder Ihr Raspberry Pi während der Installation abstürzt. Für solche Fälle bietet Abschnitt 4.14, »Notfall-Tipps«, Hilfestellungen. Drei Tipps gleich vorweg: Stellen Sie sicher, dass die Stromversorgung ausreichend ist; probieren Sie es mit einer anderen SD-Karte; verwenden Sie einen aktiven USB-Hub zum Anschluss von Tastatur und Maus.

1.4 Image-Datei auf eine SD-Karte schreiben

Das im vorigen Abschnitt beschriebene NOOBS-Konzept besteht darin, dass Sie zuerst einige Dateien auf eine formatierte SD-Karte schreiben. Der Raspberry Pi kann diese Dateien ausführen und dann im zweiten Schritt das Betriebssystem installieren. Diese Vorgehensweise ist einfach, hat aber zwei Nachteile: Zum einen dauert der Installationsprozess länger als notwendig und zum anderen ist ein Teil der SD-Karte auch nach der Installation blockiert, weil die Installationsdateien auf einer Recovery-Partition verbleiben.

Aus diesem Grund stellen viele Raspberry-Pi-Projekte ihre Distributionen in Form sogenannter Image-Dateien zur Verfügung. Eine Image-Datei ist eine blockweise Kopie der Daten, die sich auf einer SD-Karte befinden. Die Image-Datei enthält mehrere Partitionen sowie die darauf befindlichen Dateisysteme. Ein weiterer Vorteil dieses blockbasierten Ansatzes besteht darin, dass dieser unabhängig von der Größe der SD-Karte funktioniert, also auch bei SD-Karten mit mehr als 32 GByte.

Entscheidend ist, dass Sie die Image-Datei nicht als solche in das Dateisystem der SD-Karte kopieren dürfen. Vielmehr müssen Sie den *Inhalt* der Image-Datei blockweise auf die SD-Karte schreiben. Das können Sie nicht im Dateimanager Ihres Betriebssystems machen; vielmehr benötigen Sie dazu ein Spezialprogramm. In diesem Abschnitt stellen wir Ihnen entsprechende Programme für Windows, OS X und Linux vor.

Image-Dateien herunterladen

Woher bekommen Sie die erforderliche Image-Datei? Für einige ausgewählte, besonders populäre Raspberry-Pi-Distributionen, unter anderem für Raspbian, Pidora,

OpenELEC und Raspbmc, finden Sie auf der folgenden Webseite Download-Links für Image-Dateien:

http://www.raspberrypi.org/downloads

Für alle anderen Distributionen müssen Sie auf der jeweiligen Projektseite nach der Image-Datei suchen. Die Image-Dateien sind häufig in eine ZIP-Datei verpackt. Sie müssen also das ZIP-Archiv entpacken. Üblicherweise erkennen Sie die Image-Datei an der Kennung `.img`.

Bei manchen Distributionen gibt es ähnlich wie bei NOOBS zwei Varianten der Image-Datei: Das oft deutlich größere Image enthält die komplette Distribution und ist für Offline-Installationen geeignet. Die kleinere Variante enthält hingegen nur das Grundgerüst der Distribution. Der verbleibende Rest wird beim ersten Start aus dem Internet heruntergeladen. Welche Variante für Sie besser ist, hängt davon ab, ob Ihr Raspberry Pi eine Netzwerkanbindung per Kabel hat. In diesem Fall können Sie dem kleineren Image den Vorzug geben.

Manche Distributionen werden zudem in zwei Varianten angeboten, die für die Version 1 bzw. die Version 2 des Raspberry Pi optimiert sind. Das ist insofern zweckmäßig, weil sich mit dem Versionssprung die interne CPU-Architektur geändert hat (ARMv6/ARMv7). Sie müssen in diesem Fall unbedingt die richtige Variante auswählen! Derart optimierte Distributionen lassen sich nur auf passenden Raspberry-Pi-Modellen starten.

Image-Datei unter Windows auf eine SD-Karte übertragen

Unabhängig davon, ob Sie unter Windows, OS X oder Linux arbeiten, sollten Sie auf jeden Fall zuerst die SD-Karte formatieren. Theoretisch wäre das gar nicht notwendig: Beim Schreiben der Image-Datei werden ohnedies die Partitionstabelle und alle Dateisysteme überschrieben, die sich auf der SD-Karte befinden. In der Praxis hat sich aber gezeigt, dass Image-Writer viel seltener Probleme verursachen, wenn die SD-Karte leer und frisch formatiert ist.

Der populärste Image-Writer für Windows heißt *Win32 Disk Imager* und kann von der folgenden Webseite kostenlos heruntergeladen werden:

http://sourceforge.net/projects/win32diskimager

Das Programm wird nach den Windows-üblichen Rückfragen installiert. Der vom Installationsprogramm angebotene sofortige Start wird aber an mangelnden Rechten scheitern. Sie müssen das Programm nämlich mit Administratorrechten ausführen. Dazu suchen Sie im Startmenü bzw. in der Liste der Programme nach dem Eintrag Win32DiskImager, klicken diesen mit der rechten Maustaste an und wählen den Eintrag Als Administrator ausführen.

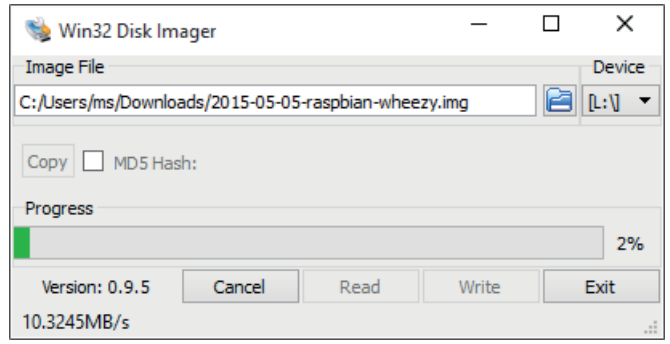


Abbildung 1.10 Win32 Disk Imager

In dem kleinen Programm wählen Sie zuerst die Image-Datei aus und dann das Laufwerk, wohin das Image geschrieben werden soll (siehe Abbildung 1.10). Aus Sicherheitsgründen stehen im Laufwerkslistenfeld nur SD-Karten und USB-Sticks zur Auswahl, aber keine Festplatten. Wenn Sie die Integrität Ihres Downloads überprüfen möchten, klicken Sie die Option MD5 Hash an. Das Programm errechnet dann eine Prüfsumme der Image-Datei, die Sie mit einer Prüfsumme vergleichen können, die oft auf der Download-Seite angegeben ist. Write schreibt die Image-Datei auf die SD-Karte. Das dauert wegen der zumeist bescheidenen Schreibgeschwindigkeit vieler SD-Karten mehrere Minuten.

Image-Datei unter OS X auf eine SD-Karte übertragen

Für OS X stehen diverse Programme zur Auswahl, um Image-Dateien auf eine SD-Karte zu übertragen. Die besten Erfahrungen haben wir mit *ApplePi-Baker* gemacht. Dieses Programm können Sie hier kostenlos herunterladen:

http://www.tweaking4all.nl/download/raspberrypi/ApplePi-Baker.zip

Nach dem Start des Programms wählen Sie im Listenfeld Pi-Crust die Device-Datei Ihrer SD-Karte aus (siehe Abbildung 1.11). Vorsicht, das Listenfeld enthält auch USB-Festplatten! Im Feld Pi-Ingredients klicken Sie rechts vom Textfeld IMG file auf den Button `...` und wählen dann die Image-Datei aus. Den eigentlichen Schreibprozess starten Sie schließlich mit dem Button IMG to SD-Card. Bevor der ApplePi-Baker mit seiner Arbeit beginnt, müssen Sie noch Ihr Passwort angeben. Der Schreibprozess erfordert Administratorrechte. Vergessen Sie nicht, die fertige SD-Karte zuerst im Finder auszuwerfen, bevor Sie die Karte aus dem Slot entfernen!

Das Programm bietet zwei Zusatzfunktionen: Mit dem Button Prep NOOBS Card können Sie eine SD-Karte formatieren. Backup SD-Card erstellt ein Backup von der eingelegten SD-Karte und speichert dieses als Image-Datei, die optional in einem komprimierten Archiv verpackt wird.

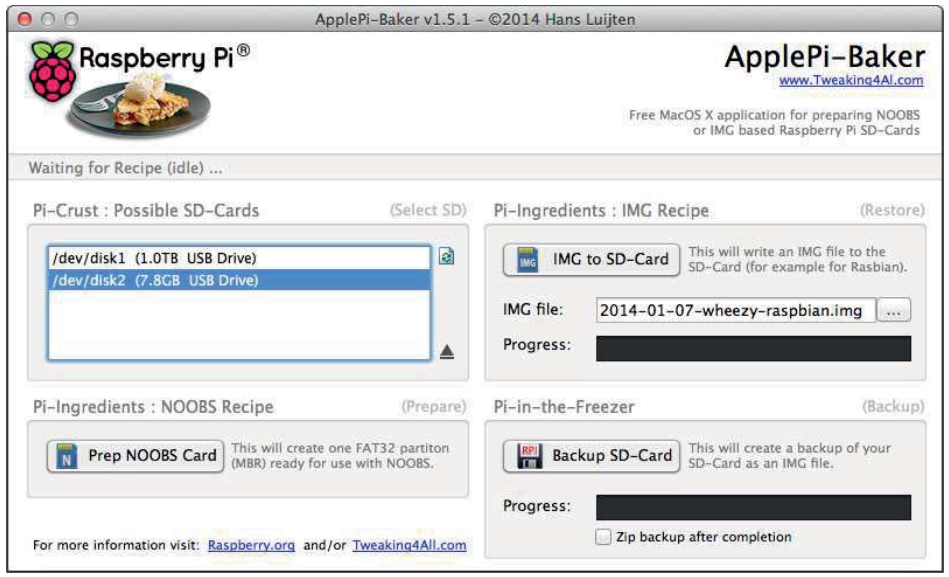


Abbildung 1.11 Der ApplePi-Baker

Roher Apfelkuchen

Bei unseren Tests ist es mehrfach vorgekommen, dass ApplePi-Baker bereits nach einer Sekunde behauptete, die SD-Karte sei fertig. Tatsächlich hatte der Schreibvorgang noch gar nicht begonnen. Abhilfe: Stellen Sie sicher, dass Sie die SD-Karte vorher formatieren, z. B. im Festplattendienstprogramm von OS X oder mit dem SD-Formatter – dann funktioniert es!

OS-X-Experten können die SD-Karte natürlich auch im Terminal erstellen. Dazu ermitteln Sie zuerst mit `diskutil list` den Device-Namen des Datenträgers und lösen dann mit `diskutil unmountDisk` alle eventuell aktiven Partitionen des Datenträgers aus dem Verzeichnisbaum. Anschließend schreiben Sie mit `sudo dd` die Image-Datei direkt auf das Device der Festplatte. Anstelle von `disk<n>` geben Sie dabei aber `rdisk<n>` an. Damit sprechen Sie das *raw disk device* an, was erheblich schneller geht. Passen Sie aber auf, dass Sie sich beim `if`-Parameter nicht vertippen! Wenn Sie hier irrtümlich das falsche Device angeben, überschreiben Sie unrettbar Ihre Festplatte!

Das folgende Listing illustriert den Vorgang. Es gibt drei Datenträger: eine interne SSD (disk0), eine Backup-Festplatte (disk1) und die SD-Karte (disk2). Während der Ausführung von `sudo dd` gibt es leider keinerlei Feedback. Der Prozess dauert einige Minuten. Denken Sie daran, die SD-Karte anschließend im Finder auszuwerfen, bevor Sie sie aus dem SD-Slot entfernen.


```
diskutil list
  /dev/disk0
     #:                       TYPE NAME       SIZE       IDENTIFIER
     0:  GUID_partition_scheme                 512.1 GB   disk0
     1:                        EFI EFI        209.7 MB   disk0s1
     2:                  Apple_HFS ssd        510.5 GB   disk0s2
     3:                Apple_Boot Recovery    1.2 GB     disk0s3
  /dev/disk1
     #:                       TYPE NAME       SIZE       IDENTIFIER
     0:  GUID_partition_scheme                 1.0 TB     disk1
     1:                        EFI EFI        209.7 MB   disk1s1
     2:                  Apple_HFS backup1    999.9 GB   disk1s2
  /dev/disk2
     #:                       TYPE NAME       SIZE       IDENTIFIER
     0: FDisk_partition_scheme                 7.8 GB     disk2
     1:                 DOS_FAT_32 NO NAME    7.8 GB     disk2s1

diskutil unmountDisk /dev/disk2
  Unmount of all volumes on disk2 was successful

sudo dd if=Downloads/raspbian.img of=/dev/rdisk2 bs=1m
  2825+0 records in, 2825+0 records out
  2962227200 bytes transferred in 278.172140 secs
    (10648900 bytes/sec)
```

Image-Datei unter Linux auf eine SD-Karte übertragen

Für Linux gibt es leider keine anerkannte Benutzeroberfläche, die beim Beschreiben von SD-Karten hilft. Das auf *http://elinux.org* erwähnte Programm ImageWriter wird nicht mehr gepflegt und steht nur in alten Linux-Distributionen zur Verfügung. Sie müssen die SD-Karte daher in einem Terminal beschreiben. Das ist nicht schwierig: Wie unter OS X müssen Sie aber aufpassen, dass Ihnen keine Tippfehler unterlaufen!

Zuerst ermitteln Sie mit `lsblk` die Device-Namen aller Datenträger. Mit `umount` lösen Sie alle Dateisysteme der SD-Karte aus dem Verzeichnisbaum. Außer bei fabrikneuen SD-Karten sollten Sie auf der SD-Karte eine neue Partitionstabelle einrichten. Auf das eigentliche Formatieren können Sie diesmal aber verzichten. Von diesem Detail abgesehen wurde die Vorgehensweise in Abschnitt 1.3, »NOOBS-Installation«, bereits beschrieben. Die folgenden Kommandos zeigen als Wiederholung nochmals die notwendigen Schritte. Beachten Sie, dass die Device-Namen auf Ihrem Linux-Rechner möglicherweise anders lauten! Im folgenden Beispiel ist `/dev/mmcblk0` der Device-Name der SD-Karte. Alle Kommandos müssen mit root-Rechten ausgeführt werden, unter Ubuntu also mit `sudo`.

```
lsblk
  NAME         MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
  sda            8:0    0 119,2G  0 disk
    sda1         8:1    0 143,1M  0 part /boot/efi
    sda2         8:2    0   2,8G  0 part [SWAP]
    ...
  mmcblk0      179:0    0   7,3G  0 disk
    mmcblk0p1  179:1    0    56M  0 part /media/kofler/boot
    mmcblk0p2  179:2    0   2,7G  0 part /media/kofler/fc254b57

umount /media/kofler/*
parted /dev/mmcblk0 mklabel msdos
```

Um die Image-Datei zu übertragen, verwenden Sie auch unter Linux das Kommando `dd`. Mit dem Parameter `if` geben Sie den Ort der Image-Datei an, mit `of` den Device-Namen der SD-Karte. Wie unter OS X gibt es auch unter Linux während der Ausführung von `dd` keinerlei Feedback. Sie müssen einfach einige Minuten abwarten, bis das Kommando abgeschlossen ist.

```
dd if=raspbian.img of=/dev/mmcblk0 bs=1M
  2825+0 Datensätze ein
  2825+0 Datensätze aus
  2962227200 Bytes (3,0 GB) kopiert, 262,231 s, 11,3 MB/s
```

Distributionsspezifische Installationsprogramme

Da es immer wieder Raspberry-Pi-Einsteiger gibt, die das Übertragen einer Image-Datei auf die SD-Karte überfordert, sind manche Distributoren dazu übergegangen, eigene Installationsprogramme anzubieten, die direkt unter Windows, OS X und fallweise unter Linux auszuführen sind. Beispielsweise gibt es derartige Installationshilfen für OSMC: Das ist eine Linux-Distribution, die speziell dafür gedacht ist, aus dem Raspberry Pi ein Multimedia-System zu machen.

Leider sind uns beim Test dieser Installationsprogramme immer wieder Fehler aufgefallen. Insofern erscheint uns eine Image-Datei die einfachere Lösung, zumal das Beschreiben einer SD-Karte, wie Sie gerade gesehen haben, wirklich keine Hexerei ist.

1.5 Installation auf einen USB-Stick

Üblicherweise verwendet der Raspberry Pi eine SD-Karte als einzigen Datenspeicher: Die SD-Karte enthält sowohl das Betriebssystem (oft Raspbian) als auch Ihre eigenen Daten, z. B. mit dem Raspberry Pi erstellte Fotos, Messdaten etc. Optional kann ein USB-Stick als zusätzlicher Datenspeicher verwendet werden.

Abweichend von diesem Standardszenario besteht auch die Möglichkeit, Linux direkt auf einen USB-Stick zu installieren. Die SD-Karte wird weiterhin benötigt, weil Raspbian von dort die für den Boot-Prozess erforderlichen Dateien liest. Aber alle weiteren Linux-Dateien und -Programme werden in der Folge direkt vom USB-Stick gelesen. Anstelle eines USB-Sticks können Sie auch eine USB-Festplatte mit eigener Stromversorgung verwenden.

Nur für Fortgeschrittene

Dieser Abschnitt richtet sich explizit an Leser bzw. Leserinnen, die bereits Linux- und Raspberry-Pi-Erfahrung haben. Viele Details, die in diesem Abschnitt vorkommen, werden erst in den weiteren Kapiteln dieses Buchs erklärt. Raspberry-Pi-Einsteiger sind gut beraten, vorerst eine normale Installation auf eine SD-Karte durchzuführen. Die Vorteile einer USB-Stick-Installation sind ohnedies nur in speziellen Anwendungsszenarien spürbar.

Vor- und Nachteile

Die Verwendung eines USB-Sticks anstelle einer SD-Karte hat einige Vorteile:

- USB-Sticks sind mitunter zuverlässiger als SD-Karten. Das gilt insbesondere dann, wenn Sie auf Ihrem Raspberry Pi Programme ausführen, die häufig große Datenmengen speichern bzw. ändern. Aus unserer Sicht ist das der entscheidende Punkt.
- Die Übertragungsgeschwindigkeit von bzw. zu USB-Sticks ist etwas höher als bei SD-Karten. Ihr Raspberry Pi bootet schneller, Programme werden flotter gestartet. In der Praxis ist der Geschwindigkeitszuwachs freilich kleiner als erwartet. Limitierende Faktoren bleiben das USB-System und die CPU-Geschwindigkeit des Raspberry Pi. Die Boot-Geschwindigkeit spielt zudem nur eine untergeordnete Rolle, weil der Raspberry Pi in den meisten Anwendungen ohnedies im Dauerbetrieb läuft und nicht ständig herunter- und wieder hochgefahren wird.

Dem stehen die folgenden Nachteile gegenüber:

- Für die erste Phase des Boot-Prozesses wird weiterhin eine SD-Karte benötigt. Die erforderlichen Dateien beanspruchen weniger als 30 MByte. Den verbleibenden Platz können Sie immerhin als zusätzlichen Datenspeicher verwenden.
- Der USB-Stick blockiert einen USB-Slot.
- Installation und Konfiguration sind etwas komplizierter.

USB-Stick vorbereiten

Wir gehen im Folgenden davon aus, dass Sie Raspbian installieren möchten. Prinzipiell ist eine USB-Stick-Installation natürlich auch für andere Distributionen möglich, allerdings sind dann unter Umständen kleine Abweichungen erforderlich.

Als Erstes müssen Sie Ihren USB-Stick so vorbereiten wie eine SD-Karte. Sofern es sich nicht um ein fabrikneues Modell handelt, formatieren Sie den USB-Stick. Anschließend übertragen Sie die Raspbian-Image-Datei auf den USB-Stick. Dazu können Sie dieselben Programme wie zum Beschreiben einer SD-Karte verwenden, also z. B. den Win32 Disk Imager, den ApplePi-Baker oder das Kommando `dd`.

SD-Karte vorbereiten

Wie bereits erwähnt wurde, benötigen Sie zusätzlich zum USB-Stick auch eine SD-Karte, wobei ein kleines Modell mit z. B. 1 GByte Speicher vollkommen ausreichend ist. Nachdem Sie die SD-Karte formatiert haben, kopieren Sie *alle* Dateien aus der Boot-Partition des USB-Sticks dorthin. Es handelt sich dabei insbesondere um die Dateien `bootcode.bin`, `kernel.img`, `config.txt`, `cmdline.txt` und `start*.*`. Diese Dateien müssen direkt auf der SD-Karte gespeichert werden, also nicht in einem Verzeichnis.

Nun laden Sie die Datei `cmdline.txt` der SD-Karte in einen beliebigen Texteditor (siehe Abbildung 1.12). Diese Datei enthält in einer einzigen, sehr langen Zeile diverse Optionen, die beim Hochfahren des Raspberry Pi an den Kernel übergeben werden. Sie müssen im Editor nun *eine* Option verändern: Anstelle von `root=/dev/mmcblk0p2` muss es `root=/dev/sda2` heißen. Diese Änderung bewirkt, dass Linux die zweite Partition des USB-Sticks als Systempartition verwendet, nicht wie sonst üblich die zweite Partition der SD-Karte. Nachdem Sie `cmdline.txt` gespeichert haben, stecken Sie die SD-Karte und den USB-Stick an Ihren Raspberry Pi und starten den Minicomputer.

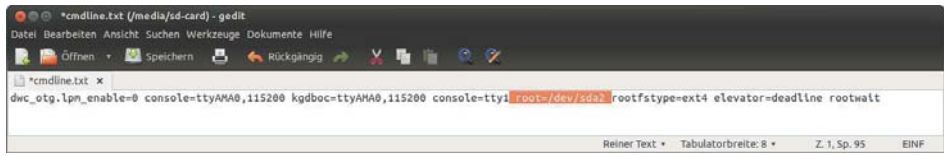


Abbildung 1.12 »cmdline.txt« in einem Editor verändern

cmdline-Syntax

Achten Sie darauf, dass die Datei `cmdline.txt` weiterhin aus nur einer Zeile bestehen darf. Manche Editoren umbrechen den Text und machen aus der Optionszeile mehrere kürzere Zeilen. In diesem Fall würde nur die erste Zeile berücksichtigt und der Boot-Vorgang würde scheitern!

Systempartition vergrößern

Sofern Ihnen bei den Vorbereitungsarbeiten keine Fehler unterlaufen sind, verläuft der Startprozess genauso wie bei der Verwendung einer normalen SD-Karte, nur ein klein wenig schneller. Beim ersten Start erscheint das Konfigurationsprogramm `raspi-config`, dessen Bedienung in Kapitel 2, »Erste Schritte in Raspbian«, beschrieben ist.

Allerdings gibt es einen Menüpunkt in `raspi-config`, der für die USB-Stick-Installation nicht zutrifft: Bei einer SD-Karteninstallation können Sie mit EXPAND FILESYSTEM die Systempartition so weit vergrößern, dass sie den gesamten zur Verfügung stehenden Platz auf der SD-Karte füllt. Bei der USB-Stick-Installation funktioniert dieses Kommando leider nicht. Unabhängig davon, wie groß der USB-Stick ist, beträgt die Größe der Systempartition ca. 2,6 GByte, von denen noch ca. 450 MByte frei sind.

Damit Sie den ganzen Platz Ihres USB-Sticks verwenden können, müssen Sie daher selbst Hand anlegen. Die folgenden Kommandos zeigen, wie Sie mit dem Programm `fdisk` die Systempartition zuerst löschen und dann neu anlegen, wobei Sie unbedingt exakt dieselbe Startposition verwenden müssen. `fdisk` führen Sie wahlweise direkt in der Textkonsole oder in einem Terminalfenster aus.

```
sudo fdisk /dev/sda
Command (m for help): p
  Disk /dev/sda: 31.4 GB, 31440961536 bytes
  64 heads, 32 sectors/track, 29984 cylinders,
    total 61408128 sectors
  Units = sectors of 1 * 512 = 512 bytes
  Sector size (logical/physical): 512 bytes / 512 bytes
  I/O size (minimum/optimal): 512 bytes / 512 bytes
  Disk identifier: 0x000981cb
     Device Boot  Start     End  Blocks   Id  System
  /dev/sda1         8192  122879   57344    c  W95 FAT32 (LBA)
  /dev/sda2       122880 5785599 2831360   83  Linux

Command (m for help): d
  Partition number (1-4): 2

Command (m for help): n
  Partition type:
     p   primary (1 primary, 0 extended, 3 free)
     e   extended
  Select (default p): p
  Partition number (1-4, default 2): 2
  First sector (2048-61408127, default 2048): 122880
  Last sector, +sectors or +size{K,M,G} (122880-61408127,
    default 61408127): <Return>
  Using default value 61408127
```

```
Command (m for help): p
  Disk /dev/sda: 31.4 GB, 31440961536 bytes
  64 heads, 32 sectors/track, 29984 cylinders,
    total 61408128 sectors
  Units = sectors of 1 * 512 = 512 bytes
  Sector size (logical/physical): 512 bytes / 512 bytes
  I/O size (minimum/optimal): 512 bytes / 512 bytes
  Disk identifier: 0x000981cb
     Device Boot Start      End   Blocks   Id  System
  /dev/sda1       8192   122879    57344    c  W95 FAT32 (LBA)
  /dev/sda2     122880 61408127 30642624   83  Linux

Command (m for help): w
  The partition table has been altered!
  Calling ioctl() to re-read partition table.
  WARNING: Re-reading the partition table failed with error 16:
  Das Gerät oder die Ressource ist belegt.  The kernel still uses
  the old table. The new table will be used at the next reboot or
  after you run partprobe(8) or kpartx(8)
```

Nun die ausführliche Erklärung, was hier vor sich geht: `fdisk` ist ein Kommando zur Partitionierung von Festplatten und anderen Datenträgern. Beim Start muss der sogenannte Device-Name des Geräts angegeben werden, also eine Spezialdatei, über die Linux auf die Festplatte zugreifen kann. Für den USB-Stick des Raspberry Pi lautet der Device-Name `/dev/sda`.

`fdisk` ist ein interaktives Programm, in dem Sie Kommandos ausführen. Die Eingabe von Kommandos erfolgt durch Buchstaben und ↵. Das erste Kommando `p` (*print*) listet die aktuelle Partitionstabelle auf. Es gibt also zwei Partitionen: eine kleine Partition mit einem Windows-Dateisystem, das in unserem Setup gar nicht verwendet wird, und eine größere Linux-Partition. Alle Start- und Endpositionen jeder Partition werden in Blöcken angegeben, wobei jeder Block 512 Byte groß ist. Davon abweichend wird die Größe der Partition in der Spalte `Blocks` hingegen in Vielfachen von 1024 Byte angegeben.

Die wichtigste Information für Sie ist die Startposition der zweiten Partition beim Block 122880. Sollte sich das Raspbian-Installations-Image nach Erscheinen dieses Buchs ändern, kann es sein, dass `fdisk` bei Ihnen eine andere Position anzeigt. Diese Position müssen Sie sich merken.

`d` (*delete*) löscht nun die zweite Partition. Keine Angst, sofern Sie die weitere Anleitung exakt befolgen, verlieren Sie dabei keine Daten! Denn bereits im nächsten Schritt wird die Partition mit `n` (*new*) wieder neu angelegt. `fdisk` fragt nun zuerst nach der Partitionsnummer (2), dann nach dem Partitionstyp (`p` für *primary*) und schließlich nach der Startposition der neuen Partition: Jetzt ist es entscheidend, dass

Sie exakt dieselbe Startposition wie bisher angeben, in unserem Beispiel also 122880 Blöcke. Unkompliziert ist die Frage nach der Endposition: `fdisk` schlägt den letzten Block des USB-Sticks vor, und Sie bestätigen diese Position einfach mit ↵.

Bis jetzt haben Sie alle Änderungen nur im Speicher durchgeführt. Erst mit `w` (*write*) wird die neue Partitionstabelle tatsächlich auf dem USB-Stick gespeichert. Jetzt gibt es also kein Zurück mehr. Der Schreibvorgang endet mit einer Warnung: Da der USB-Stick momentan aktiv genutzt wird, kann der Linux-Kernel die geänderte Partitionstabelle vorerst noch nicht berücksichtigen. Sie müssen Ihren Raspberry Pi daher nun neu starten, am einfachsten mit `sudo reboot`.

Nach dem Neustart erkennt Linux die neue Partitionsgröße. Unverändert geblieben ist aber das Dateisystem, das weiterhin nur rund 2,6 GByte der Partition nutzt. Daher ist nun ein letzter Schritt erforderlich: Die Anpassung des Dateisystems an die neue Partitionsgröße. Dazu führen Sie das folgende Kommando in der Textkonsole oder in einem Terminal aus:

```
sudo resize2fs /dev/sda2
  resize2fs 1.42.5 (29-Jul-2012)
  Das Dateisystem auf /dev/sda2 ist auf / eingehängt;
  Online-Grössenveränderung nötig
  old_desc_blocks = 1, new_desc_blocks = 2
  Das Dateisystem auf /dev/sda2 ist nun 7660656 Blöcke groß.
```

Zuletzt sollten Sie sich vergewissern, ob alles funktioniert hat. Dazu führen Sie das Kommando `df -h` aus. Es gibt einen Überblick über alle aktiven Dateisysteme. Gleich die erste Zeile des Ergebnisses zeigt, dass das Dateisystem in der Systempartition auf unserem 32-GByte-USB-Stick nun rund 29 GByte beträgt. Vielleicht fragen Sie sich, wo die übrigen drei GByte geblieben sind. Der Grund für die Diskrepanz sind unterschiedliche Rechenweisen. Datenträgerhersteller rechnen immer dezimal. Ein USB-Stick mit 32 GByte umfasst demnach rund 32.000.000.000 Byte. `df` rechnet hingegen binär. Ein GByte entspricht dort 2^{30} Byte, also 1.073.741.824 Byte.

Die restlichen Zeilen des `df`-Ergebnisses betreffen größtenteils temporäre bzw. virtuelle Dateisysteme. Interessant wird es erst wieder bei den letzten beiden Zeilen. Die `/boot`-Partition stammt von der SD-Karte und enthält die für den Boot-Prozess erforderlichen Daten. Über das Verzeichnis `/media/boot` ist außerdem die erste Partition des USB-Sticks zugänglich. Diese enthält ebenfalls Boot-Dateien, die aber ungenutzt sind.

```
df -h
  Dateisystem    Größe Benutzt Verf. Verw% Eingehängt auf
  rootfs           29G    2.0G   26G    8% /
  /dev/root        29G    2.0G   26G    8% /
  devtmpfs        211M       0  211M    0% /dev
```

```
tmpfs            44M    288K    44M     1% /run
...
/dev/mmcblk0p1  7.3G     22M  7.3G     1% /boot
/dev/sda1        56M     19M    38M    34% /media/boot
```

Kapitel 16

Erweiterungsboards

In diesem Kapitel behandeln wir Erweiterungsboards, die speziell für den Raspberry Pi entwickelt wurden. Diese Boards erleichtern in erster Linie den Zugang zu GPIO-Ports und den dort verfügbaren Systemen und Kommunikationstechnologien (SPI, I^2C usw.).

Für viele Einsatzzwecke, die wir in den vorangegangenen Kapiteln behandelt haben, gibt es spezielle Boards. So sparen Sie sich die Anschaffung von und den Aufbau mit Einzelbausteinen. Gerade für Einsteiger ist es sinnvoll, auf eines der zahlreichen Boards zurückzugreifen. Viele Boards können Sie in bereits bestückter Ausführung oder als Bausatz kaufen (siehe Abbildung 16.1).

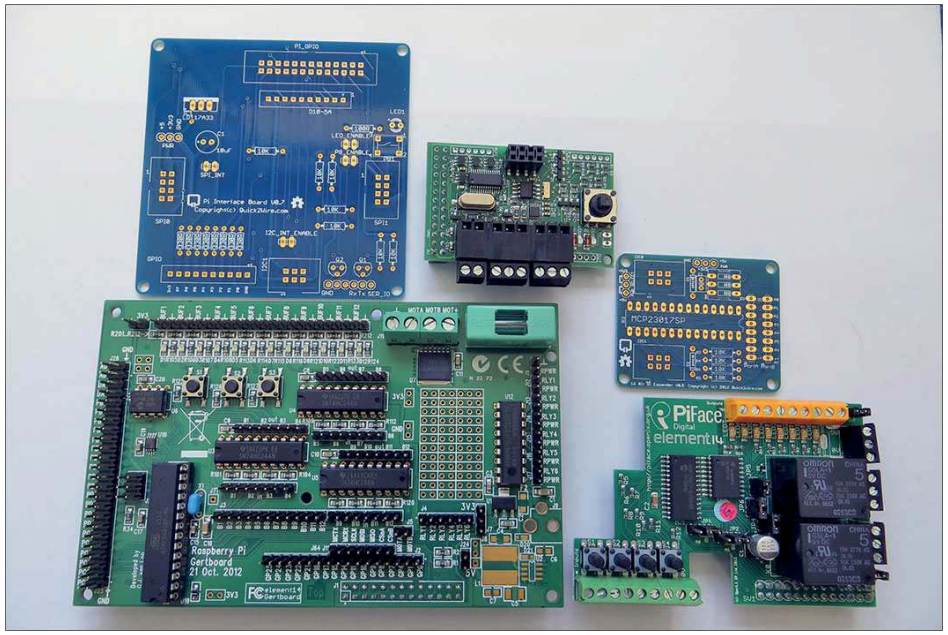


Abbildung 16.1 Eine Auswahl einiger verfügbarer Erweiterungsboards

Es muss nicht direkt die Mammuterweiterung *Gertboard* sein, oftmals reicht auch z. B. ein *PiFace* für die ersten Hardware-Versuche. Besonders Hardware-Neulinge schützen damit ihren wertvollen Raspberry Pi vor eventuellen Schäden. Oftmals sind die Erwei-

terungsboards durch Puffer, Optokoppler oder Transistoren abgesichert, und hinter den Boards kann sorgenfrei losgebastelt werden.

Auf einige der wichtigsten und hilfreichsten Boards möchten wir auf den nächsten Seiten eingehen. Vorweg ein erster Überblick:

- Das *Gertboard* ist eine Allround-Lösung, da es so gut wie alle Möglichkeiten der GPIO-Schnittstelle auf einer Leiterkarte abbildet.
- Das *RasPiComm* legt gezielt Wert auf den einfachen Umgang mit den Kommunikationsprotokollen.
- Suchen Sie nach einem Board zur Ansteuerung von Motoren oder Relais, dann bieten sich das *RTK Motor Controller Board* von Adafruit oder das *Step Your Pi Board* für Schrittmotoren von ModMyPi an. Das *PiFaceDigital 2* bietet unter anderem fertig bestückte Relais.

Kompatibilität mit dem Raspberry Pi 3

Die folgenden Abschnitte enthalten immer wieder Anmerkungen zur Kompatibilität mit dem Raspberry Pi 3. Einige Erweiterungen sind direkt kompatibel. Bei den meisten anderen Boards ist es möglich, durch einen sogenannten *Stacking Header* die ersten 26 Pins etwas höher zu legen und dort wie gewohnt alte Erweiterungen aufzustecken. Einen solchen Adapter finden Sie beispielsweise bei EXP-Tech.de:

http://www.exp-tech.de/stacking-header-for-raspberry-pi-b-2x20-extra-tall-header

16.1 Das Gertboard

Das Gertboard ist eines der ersten und umfangreichsten Erweiterungsboards, die für den Raspberry Pi erhältlich sind. Entwickelt wurde das Board von Gert van Loo, einem Entwickler des Raspberry Pi, der dem Board auch seinen Namen verlieh. Das Gertboard zählt zu den Allroundern und ist perfekt geeignet, um erste Prototypen der eigenen Projekte zu erstellen. Durch seine extra abgesicherten Ein- und Ausgänge bleibt der Raspberry Pi auch im Falle von elektrischen Fehlern in der Regel geschützt.

Im Verlauf dieses Kapitels werden Sie einige Bilder finden, die das Gertboard auf dem alten Raspberry Pi B zeigen. Das Gertboard ist für alle Raspberry-Pi-Modelle uneingeschränkt verwendbar. Es gibt noch keine Version des Gertboard, das an die neue 40-polige Steckerleiste angepasst ist. Da die ersten sechsundzwanzig Pins der Steckerleiste jedoch denen der alten Modelle entsprechen, kann mit diesen das Gertboard auch weiterhin noch perfekt zum Experimentieren verwendet werden.

Das Gertboard stellt die folgenden Funktionen zur Verfügung:

- 12 geschützte Ein- und Ausgänge
- 3 Taster
- 6 Open-Collector-Ausgänge (50 V, 0,5 A)
- Motortreiber für maximal 18 V und 2 A
- ATmega328-Microcontroller zur Auslagerung von Programmen
- 12 Status-LEDs
- ein 8-Bit-D/A-Wandler
- ein 10-Bit-A/D-Wandler

Inbetriebnahme

In der aktuell verfügbaren Version des Gertboards wird das komplette Board auf den Raspberry Pi gesteckt (siehe Abbildung 16.2). Frühere Versionen benötigten eine Flachbandleitung, um das Gertboard mit dem Raspberry Pi zu verbinden. Das Gertboard ist auch mit dem Modell 2 oder B+ kompatibel. Stecken Sie den Steckverbinder einfach auf die ersten 26 Pins der 40-poligen Steckerleiste (siehe Abbildung 16.3).

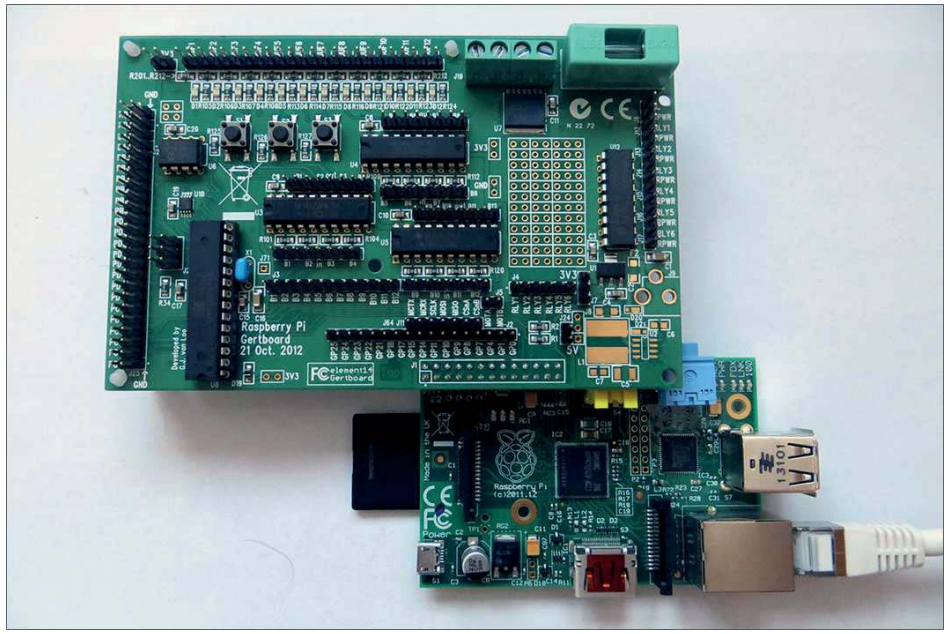


Abbildung 16.2 Auf dem Raspberry Pi montiertes Gertboard

Das Gertboard ist in verschiedene Funktionsblöcke aufgeteilt (siehe Abbildung 16.4). Die beiliegenden Verbindungskabel und Jumper machen es Ihnen möglich, gezielt einzelne Blöcke zu aktivieren bzw. miteinander zu verbinden. Wir beginnen nun als

Beispiel, die einzelnen Blöcke in Betrieb zu nehmen und deren Funktion am Raspberry Pi zu testen. Ein umfangreiches Handbuch sowie die Belegung und das Layout des Gertboards finden Sie in der offiziellen PDF-Datei:

http://www.element14.com/community/servlet/JiveServlet/previewBody/51727-102-1-265829/Gertboard_UM_with_python.pdf

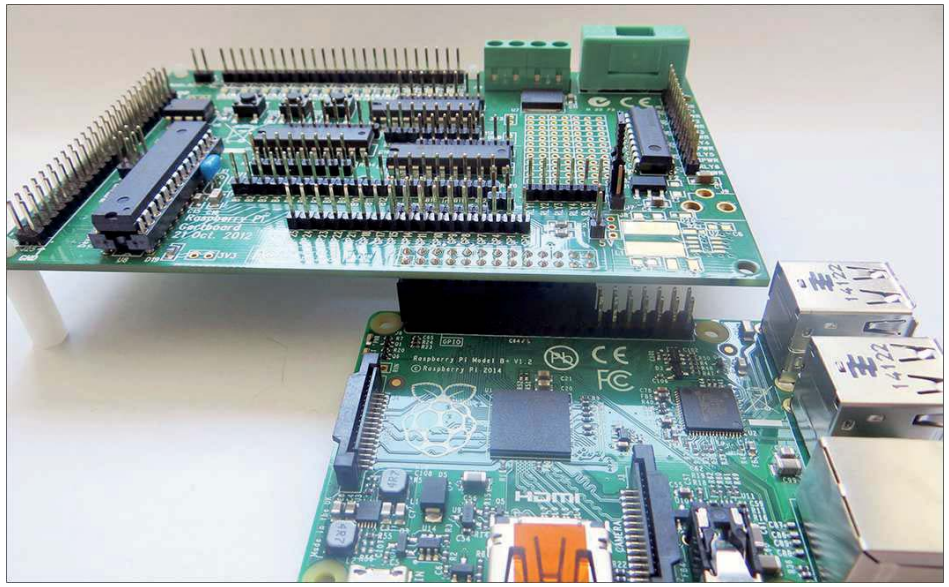


Abbildung 16.3 Das Gertboard kann problemlos auf den Raspberry Pi 2 oder 3 gesteckt werden.

Die drei Taster

Das Gertboard enthält drei Druckknöpfe, die bei Betätigung den entsprechenden GPIO-Pin über einen 1-k-Widerstand gegen Masse ziehen. Die Taster sind nicht fest an bestimmte Ports gebunden. Hier kommen die mitgelieferten Kabel und Jumper zum Einsatz, um den Tastern einen beliebigen GPIO-Pin zuzuweisen.

Um die Taster mit dem Raspberry Pi zu verbinden, benötigen Sie die mitgelieferten Jumper und Jumper-Kabel. Werfen Sie einen Blick auf das gesamte Gertboard, so finden Sie dort die Steckerleiste *J2*. Diese Leiste ist mit GP1, GP17 usw. beschriftet. Dies spiegelt die GPIO-Pins des Raspberry Pi wider. Die Bezeichnung dieser Pins ist in der BCM-Variante gewählt. In diesem Beispiel möchten wir die drei Taster den GPIO-Pins 11, 13 und 15 zuweisen. Dies entspricht den BCM-Namen 17, 27 und 22. Wir haben diese Konstellation gewählt, um auf eine Besonderheit hinzuweisen: GPIO 27 trägt seinen Namen erst seit der Raspberry-Pi-Revision 2. Die alte Bezeichnung war GPIO 21. Auf dem aktuellen Gertboard ist allerdings der GP27 noch weiterhin mit GP21 beschriftet!

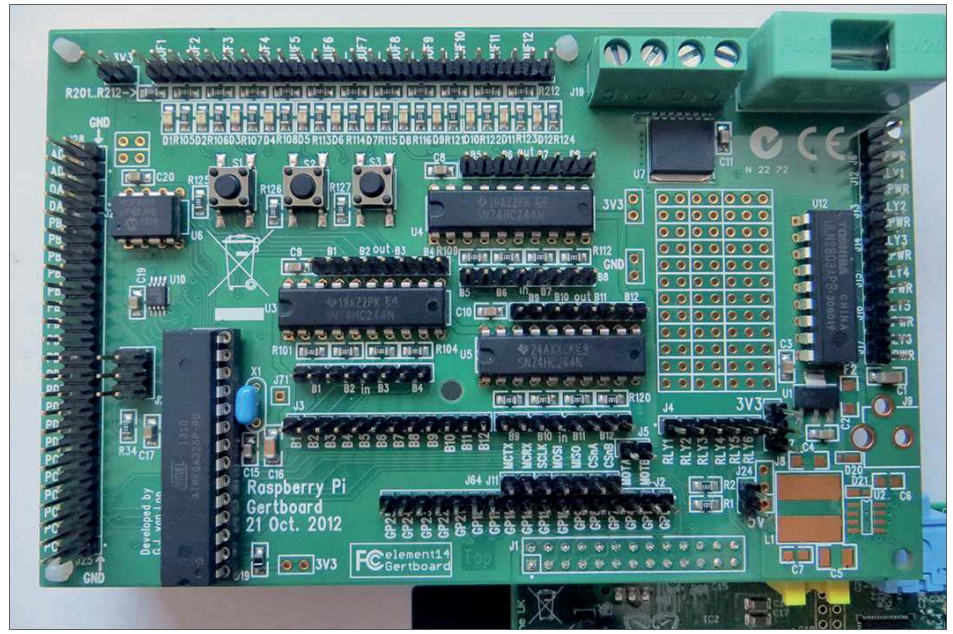


Abbildung 16.4 Das Gertboard in der Detailansicht

Vorsicht, irreführende Pin-Bezeichnungen!

Auch die Beschriftung der Pins GP 0 und GP 1 bezieht sich auf die kaum mehr gebräuchliche Revision 1 des Modells B. Im Modell 3 des Raspberry Pi entsprechen diese Pins den Ports GPIO 2 und GPIO 3.

Die Steckerleiste *J3* enthält die mit B1 bis B3 gekennzeichneten Pins. Diese sind bereits mit den Tastern verbunden. Sie müssen nun lediglich folgende Verbindungen durch die Jumper-Kabel herstellen:

- J3-B1 zu J2-GP17
- J3-B2 zu J2-GP21
- J3-B3 zu J2-GP22

Bereits jetzt könnten die Taster als Eingabeknöpfe am Raspberry Pi verwendet werden. Das Gertboard bietet allerdings zusätzlich noch die Option, den Status der Taster durch die verbauten LEDs anzuzeigen. Platzieren Sie hierzu den Jumper auf der Leiste *J7* (siehe Abbildung 16.5). Hierbei handelt es sich um die Spannungsversorgung von 3,3 V, die durch das Platzieren des Jumpers auf das Board geführt wird. Sobald der Jumper platziert ist, sollten alle LEDs auf dem Gertboard leuchten.

Betrachten Sie das Gertboard aus dem Blickwinkel der Abbildung, so befindet sich unter dem mit *U3* markierten Gebiet eine weitere Steckleiste, die die Pins B1, B2, B3

und B4 mit dem Hinweis *out* beinhaltet. Stecken Sie je einen Jumper über die Pins B1 bis B3 (siehe Abbildung 16.5).

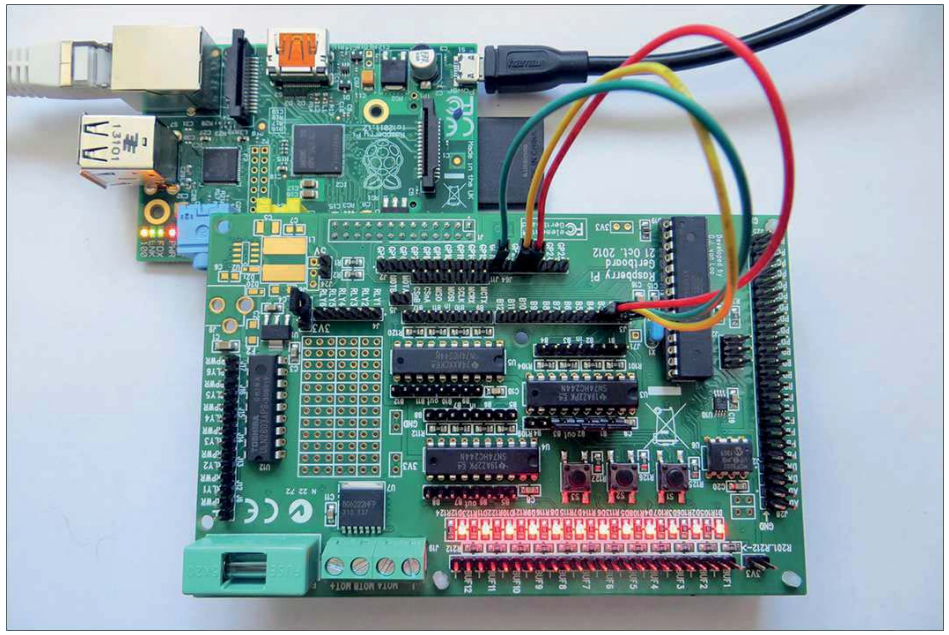


Abbildung 16.5 Verdrahtung des Gertboards zur Inbetriebnahme der Taster

Ein Tastendruck zieht jetzt den entsprechenden Raspberry-Pi-Pin gegen Masse. Dieses Verhalten spiegeln auch die LEDs wider. Sobald ein Taster betätigt ist, erlischt die entsprechende LED. Die Tasteneingaben verarbeiten Sie in einem Python-Script:

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(27, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(22, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    if not GPIO.input(17):
        print("S1")
        sleep(0.5)
    elif not GPIO.input(27):
        print("S2")
        sleep(0.5)
```



```
    elif not GPIO.input(22):
        print("S3")
        sleep(0.5)
```

Dieses kleine Beispielprogramm zeigt Ihnen, wie die Taster des Gertboards in Python verwendet werden können. Achten Sie darauf, die internen Pull-up-Widerstände zu aktivieren: Auf dem Gertboard selbst sind an den Tastern keine Pull-up-Widerstände verbaut. Um mit der Bezeichnung der Pins nicht durcheinanderzugeraten, bietet es sich bei der Arbeit mit dem Gertboard an, stets die BCM-Bezeichnungen der GPIO-Ports zu verwenden, da diese auf dem Gertboard aufgedruckt sind.

Digitale Ein- und Ausgänge und Leuchtdioden

Neben den drei Tastern auf dem Gertboard gibt es natürlich auch die Möglichkeit, jedes beliebige digitale Signal mit einem 3,3-V-Pegel zu verarbeiten. Hierzu werden mittels Jumpern auf dem Gertboard die Ports als Ein- oder Ausgang bestimmt. Als Beispiel dafür erstellen wir eine kleine Schaltung, in der durch einen Taster ein Signal erzeugt wird, woraufhin der Raspberry Pi eine externe LED schaltet.

Werfen Sie noch einen Blick auf das Board: Dort sind insgesamt zwölf gepufferte Ein- bzw. Ausgänge verfügbar. Den Zugang dazu finden Sie in der Pin-Leiste *J3*. Die Leiste *J2* führt alle GPIO-Ports als Pins aus dem Gertboard heraus. Verbinden Sie für diese Schaltung den Pin J2-GP22 mit J3-B4. Der vierte der zwölf verfügbaren I/Os auf dem Gertboard ist nun mit dem GPIO-Pin BCM 22 verbunden.

Im nächsten Schritt legen Sie fest, ob B4 ein Ein- oder Ausgang werden soll. Dies geschieht über die mitgelieferten Jumper: Stecken Sie den Jumper auf die Position U3-out-B4. Diese befindet sich oberhalb des ICs, der mit *U3* auf dem Gertboard eingezeichnet ist. Durch das Platzieren dieses Jumpers kann B4 nun als Ausgang genutzt werden. Die Verbindung nach *außen* geschieht über die Pins unterhalb der LEDs. Diese sind mit BUF1 bis BUF12 markiert. Hierbei handelt es sich um die Ausgänge bzw. Eingänge der Puffer-ICs.

Generell gilt: Die Puffer-ICs trennen jegliche Schaltung, die hinter den BUF*n*-Pins liegt, von den eigentlichen GPIO-Ports des Raspberry Pi. Sie dienen somit als Schutz vor Fehlbeschaltung. Zu guter Letzt setzen Sie auf die unteren beiden Pins der Leiste *J7* einen Jumper und versorgen so das Gertboard mit 3,3 V Betriebsspannung.

Wenn Sie die oben beschriebene Verdrahtung vornehmen, lässt sich der Puffer B4 bereits durch den Raspberry Pi schalten. Das Ausgangssignal wird bei einem aktiven Ausgang nun an BUF4 anliegen. Um dieses Beispiel ein wenig zu verdeutlichen, bauen wir wieder ein kleines Experiment auf: Wir nutzen einen der Taster, um dem Raspberry Pi ein Signal zu senden. Ein Python-Programm wertet das Signal aus und bringt eine externe LED zum Leuchten.

Dazu fügen Sie zu der oben beschriebenen Verdrahtung noch die Verbindung eines Tasters zu einem GPIO-Port hinzu. Wenn Sie den Taster S3 nutzen möchten, verbinden Sie also die Pins J3-B3 mit J2-GP17. Zusätzlich nehmen Sie nun eine LED und verbinden die Anode über einen Vorwiderstand mit BUF4 und die Kathode mit dem danebenliegenden GND-Pin (siehe Abbildung 16.6).

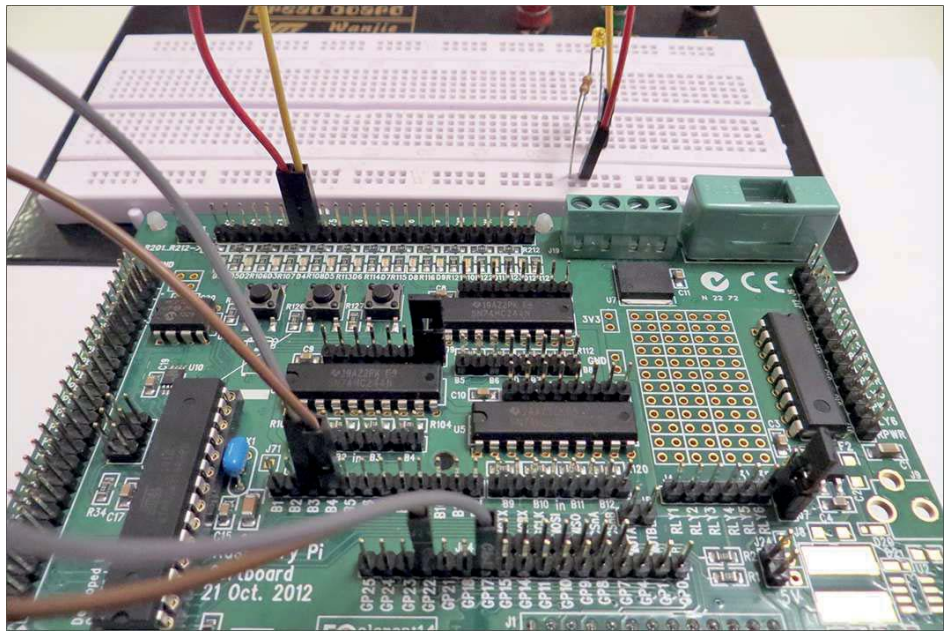


Abbildung 16.6 Die gesamte Verdrahtung zum Testen der digitalen Ein-/Ausgänge

Das dazugehörige Python-Programm kann in einer Minimalausführung wie folgt aussehen:

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(22, GPIO.OUT)

while True:
    if not GPIO.input(17):
        GPIO.output(22, True)
    else:
        GPIO.output(22, False)
    sleep (0.1)
```


Beachten Sie, dass die gepufferten Ein- und Ausgänge des Gertboards ebenfalls mit maximal 3,3 V zu belasten sind. Zum Schalten von höheren Spannungen nutzen Sie den Open-Collector-Treiber.

Der Open-Collector-Treiber

Auf dem Gertboard sind sechs Open-Collector-Ausgänge verfügbar. Diese werden durch IC *ULN2803A* realisiert. Dank dieses Darlington-Arrays ist es möglich, Spannungen von bis zu 50 V mit einer Strombelastung von 500 mA pro Ausgang zu schalten. Eine detaillierte Beschreibung des ULN2803A samt Beschaltungs- und Verwendungsbeispielen finden Sie in Abschnitt 13.1, »Leuchtdioden (LEDs)«.

In diesem Abschnitt erstellen wir eine kleine Schaltung, die es ermöglicht, externe Spannungen mit dem Gertboard zu schalten. Stellen Sie dazu mit einem Jumper-Kabel folgende Verbindung auf dem Gertboard her:

- J2-GP7 zu J4-RLY1

Damit kann nun bereits über GPIO 7 der Port 1 des ULN2803A geschaltet werden. In unserem Beispiel schalten wir eine LED über ein externes Netzteil. Dazu wird J12-RLY1 mit der Kathode der LED verbunden. Ein externes Netzteil versorgt die LED an der Anode sowie J6-RPWR auf dem Gertboard mit der externen Spannung. Empfehlenswert für dieses Beispiel sind 3 bis 5 V. J6-GND wird mit der Masse des Netzteils verbunden (siehe Abbildung 16.7).

Jetzt fehlt nur noch das dazugehörende Programm: Sofern Sie die Anschlüsse wie oben beschrieben vorgenommen haben, können Sie den Python-Code aus dem Beispiel in Abschnitt 13.1, »Leuchtdioden (LEDs)«, unverändert übernehmen.

Der Motortreiber

Neben dem ULN2803A kann auch der integrierte Motortreiber BD6222HFP größere Lasten schalten. Den Motortreiber finden Sie physisch in dem mit U7 markierten Gebiet auf dem Gertboard. In der Platinenversion *21. Oct. 2012* des Gertboards ist der oben genannte BD6222HFP als Treiber verbaut. Die Vorgängerversion verwendete den L6203. In der Bedienung macht sich dies nicht bemerkbar, wohl aber bei den maximal zulässigen Spannungen und Strömen: Der BD6222HFP kann mit 18 V und maximal 2 A an seinen Motorausgängen umgehen. Der alte L6203 konnte aufgrund seiner größeren Bauform sogar Spannungen bis 48 V und Ströme bis 5 A schalten.

Angesteuert wird der Motortreiber lediglich über zwei Pins: *MOTA* und *MOTB* in der Pin-Leiste J5. Die Ausgänge des Treibers sind als Schraubanschlüsse neben der Sicherung ausgeführt. Die 2-A-Feinsicherung sorgt für den Schutz vor Überstrom am Treiberausgang. Zudem hat der BD6222HFP einen internen Überhitzungsschutz. Lesen Sie dazu gegebenenfalls vorab Abschnitt 13.3, »Elektromotoren«!

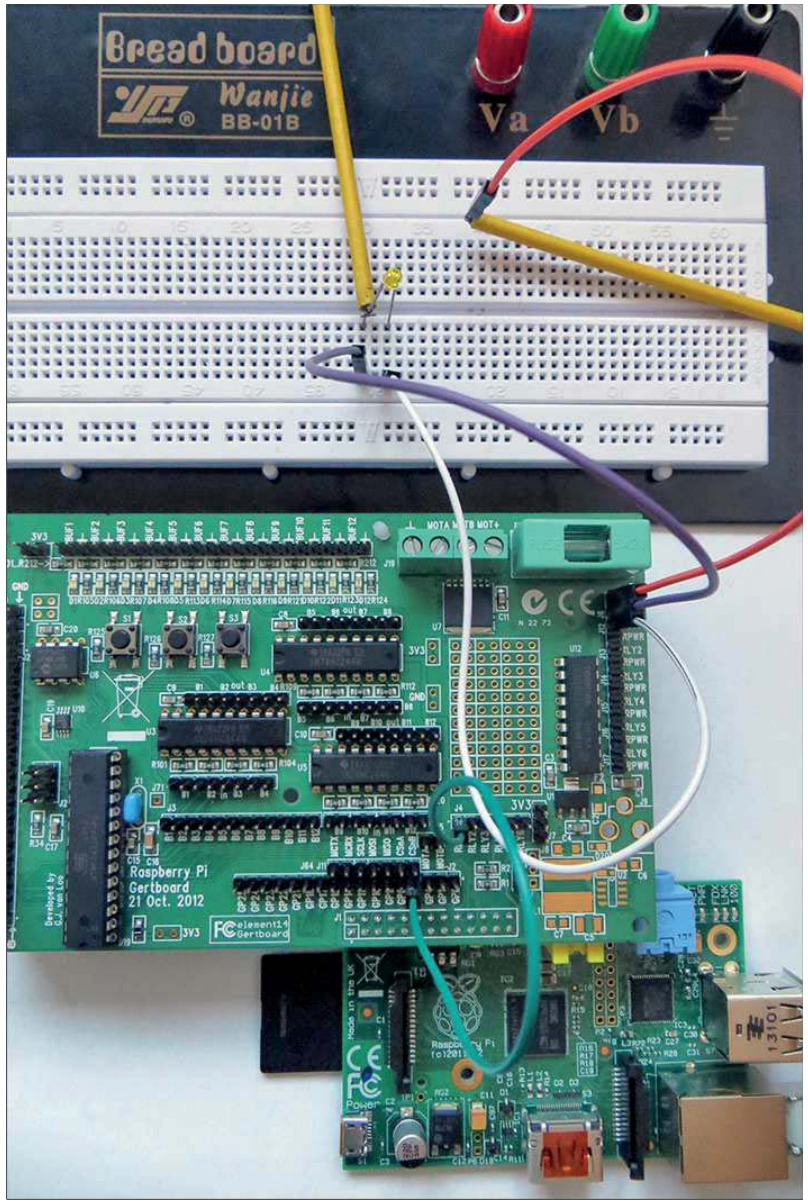


Abbildung 16.7 Eine LED wird über ein externes Netzteil am ULN2803A des Gertboards betrieben.

Um den Motortreiber auszuprobieren, benötigen wir ein externes Netzteil, das die Motorbetriebsspannung liefert, sowie einen Gleichstrommotor. Die Drehrichtung des Motors bestimmen Sie über die Taster S1 und S3 auf dem Gertboard. Stellen Sie mit den Jumper-Kabeln die beiden folgenden Verbindungen her:

- J2-GP17 zu J5-MOTA
- J2-GP18 zu J5-MOTB
- J3-B1 zu J2-GP23
- J3-B2 zu J2-GP24

Die Schraubklemmen in J19 belegen Sie wie folgt (siehe Abbildung 16.8):

- GND mit der Masse des Netzteils
- MOT+ mit dem Pluspol des Netzteils
- MOTA mit Motorleitung 1
- MOTB mit Motorleitung 2

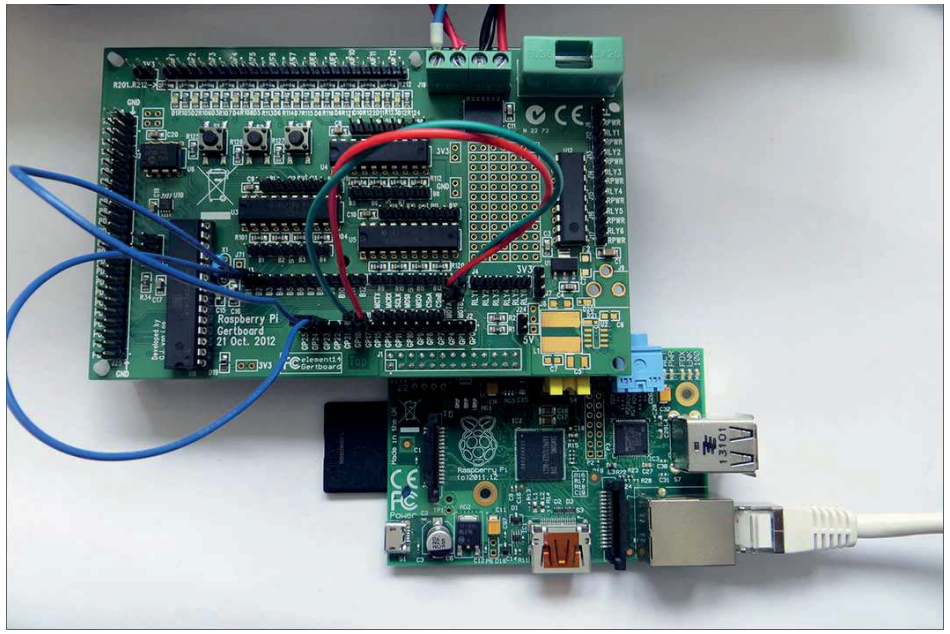


Abbildung 16.8 Verkabelung des Gertboards zur Nutzung des Motortreibers und der Taster

Warnung

Bei unseren Experimenten mit dem Motortreiber auf dem Gertboard haben wir einen Treiber zerstört – und das trotz sachgemäßer Verkabelung und Handhabung. Der Schaden zeigte sich durch einen Kurzschluss zwischen MOT+ und GND der Schraubklemme auf dem Gertboard. Wie die Diskussion im Raspberry-Pi-Forum zeigt, scheint dies kein Einzelfall zu sein:

http://www.raspberrypi.org/forums/viewtopic.php?f=42&t=38188

Im dazugehörenden Python-Programm werden die beiden Eingänge für die Taster (`b1` und `b2`) mit einem internen Pull-up-Widerstand versehen. Die Pins in den Variablen `mota` und `motb` werden als Ausgang definiert, da diese den Motortreiber ansteuern.

Wir nutzen in diesem Programm Interrupts zur Flankenerkennung: Die Eingänge `b1` und `b2` werden auf steigende sowie fallende Flanken überwacht. Die Callback-Funktionen `mot_v` und `mot_z` unterscheiden anhand der `if`-Abfrage, ob für den Pin eine positive oder negative Flanke vorlag.

Was erwartet Sie nach dem Start des Programms? Drücken Sie den Taster S1, so dreht der Motor in eine Richtung. Lassen Sie den Taster los, so bleibt er stehen. Ebenso gilt dies für den Taster S2 in die entgegengesetzte Richtung.

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
import time
mota = 17
motb = 18
b1 = 23
b2 = 24
GPIO.setmode(GPIO.BCM)
GPIO.setup(b1, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(b2, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(mota, GPIO.OUT)
GPIO.setup(motb, GPIO.OUT)

def mot_v( pin ):
    if GPIO.input(pin) == False:
        GPIO.output(mota, GPIO.HIGH)
        GPIO.output(motb, GPIO.LOW)
    else:
        GPIO.output(mota, GPIO.LOW)
        GPIO.output(motb, GPIO.LOW)
    return

def mot_z( pin ):
    if GPIO.input(pin) == False:
        GPIO.output(mota, GPIO.LOW)
        GPIO.output(motb, GPIO.HIGH)
    else:
        GPIO.output(mota, GPIO.LOW)
        GPIO.output(motb, GPIO.LOW)

GPIO.add_event_detect(b1, GPIO.BOTH, bouncetime=50)
GPIO.add_event_callback(b1, mot_v)
```



```
GPIO.add_event_detect(b2, GPIO.BOTH, bouncetime=50)
GPIO.add_event_callback(b2, mot_z)

try:
  while True:
    time.sleep(5)
except KeyboardInterrupt:
  GPIO.cleanup()
  sys.exit()
```

Motorgeschwindigkeit steuern

Um auch die Motorgeschwindigkeit zu steuern, generieren Sie an den Motortreibereingängen MOTA und MOTB ein PWM-Signal. Eine entsprechende Anleitung finden Sie in Abschnitt 13.3, »Elektromotoren«.

Der Analog-Digital-Wandler

Als A/D-Wandler ist der MCP3002 im Gertboard verbaut. Dieses SPI-Bauteil stammt aus der gleichen Familie wie der in Abschnitt 14.2 vorgestellte MCP3008. Der MCP3002 jedoch ist nur ein Dual-Channel-ADC, also ein A-D-Wandler mit zwei Kanälen, während der MCP3008 gleich über acht Kanäle verfügt. Beide Bausteine haben eine Auflösung von 10 Bit. Um den ADC des Gertboards zu nutzen, stellen Sie folgende Jumper-Verbindungen her:

- J2-GP11 zu J11-SCLK
- J2-GP10 zu J11-MOSI
- J2-GP9 zu J11-MISO
- J2-GP8 zu J11-CSnA

Auf der Pin-Leiste J28 am linken oberen Rand des Gertboards finden Sie die Pins *AD0* und *AD1*. Dies sind die beiden Kanäle des MCP3002. Die darunterliegenden Pins *DA0* und *DA1* gehören zum D-A-Wandler, der im nächsten Abschnitt behandelt wird.

Zum Ausprobieren des A/D-Wandlers benötigen Sie ein Potenziometer. Wir haben ein Modell mit einer Reichweite von 0 Ω bis 1 kΩ verwendet. Das Potenziometer schließen Sie nun wie folgt an das Gertboard an (siehe Abbildung 16.9):

- Pin 1 an den rechten Pin von J28-AD0 (GND)
- Pin 2, also den Schleifer des Potis, an den linken Pin von J28-AD0
- Pin 3 an 3,3 V (ganz oben links auf dem Gertboard)

Denken Sie daran, dass Sie für dieses und das nächste Kapitel die SPI-Schnittstelle auf dem Raspberry Pi freischalten müssen (siehe Abschnitt 14.2, »Der Analog-Digital-Wandler MCP3008«). Auch den Steuerungscode können Sie aus diesem Abschnitt

übernehmen. Einige Details müssen allerdings verändert werden: Da im Gertboard ein A/D-Wandler sowie ein D/A-Wandler verbaut sind, die *beide* über die SPI-Schnittstelle verbunden sind, sind zwei SPI-Kanäle in Verwendung. Der hier verwendete A/D-Wandler liegt an SPI-Kanal 0, der D/A-Wandler an Kanal 1. Dies führt zu folgender Konfiguration für das Beispielprogramm:

```
spi.open(0,0)
```

Das vollständige Programm für ein Poti an AD0 sieht wie folgt aus:

```
#!/usr/bin/python3
import spidev
import time

spi = spidev.SpiDev()
spi.open(0, 0)
while True:
    antwort = spi.xfer([1, 128, 0])
    if 0 <= antwort[1] <=3:
        wert = ((antwort[1] * 256) + antwort[2]) * 0.00322
        print(wert, " V")
        time.sleep(1)
```

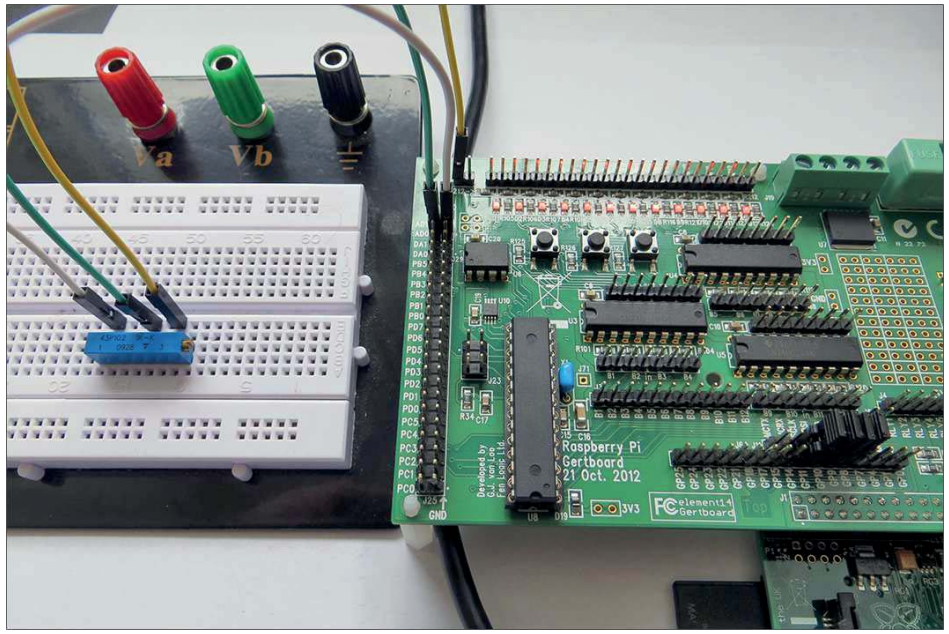


Abbildung 16.9 Jumper-Positionen und Poti zur Nutzung des A/D-Wandlers

Wenn Sie den Kanal DA1 verwenden möchten, müssen Sie die Konfigurationsbits gemäß dem Datenblatt anpassen:

http://ww1.microchip.com/downloads/en/DeviceDoc/21294E.pdf

Für das Beispielprogramm heißt das, dass der Funktion `xfer` folgende Parameter mitgegeben werden müssen:

```
spi.xfer([1, 192, 0])
```

Der Digital-Analog-Wandler

Als D/A-Wandler ist im Gertboard je nach Verfügbarkeit ein Baustein aus der MCP48XX-Familie verbaut, also beispielsweise ein MCP4822, MCP4812 oder ein MCP4802. Unser Gertboard enthielt einen MCP4802. Prüfen Sie dies anhand der Bauteilbeschriftung des ICs in dem mit *U10* markierten Gebiet (siehe Abbildung 16.10). Die Bauteile unterscheiden sich durch ihre Auflösung, die beim MCP4802 8 Bit, beim MCP4812 10 Bit und beim MCP4822 12 Bit beträgt.

Alle drei möglicherweise verbauten Bausteine besitzen zwei Ausgangskanäle. Eine grundsätzliche Einführung in die Funktionen des D/A-Wandlers finden Sie in Abschnitt 14.3, »Der Digital-Analog-Wandler MCP4811«. Der dort verwendete MCP4811 hat allerdings nur einen Kanal. Dennoch ist die Handhabung gut übertragbar, insbesondere was die Berechnung der Auflösung und die Steuerung der verschiedenen Betriebsmodi betrifft.

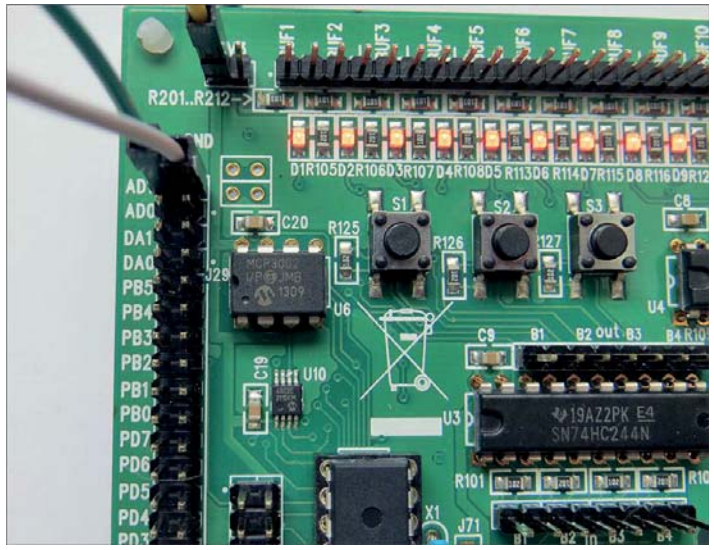


Abbildung 16.10 Auf unserem Gertboard ist ein MCP4802 verbaut. Das ist an der Beschriftung im Gebiet U10 zu erkennen.

Um den Zugriff auf den D/A-Wandler zu ermöglichen, sind die Jumper wie folgt zu setzen:

- J2-GP11 zu J11-SCLK
- J2-GP10 zu J11-MOSI
- J2-GP9 zu J11-MISO
- J2-GP7 zu J11-CSnB

Die erzeugte Analogspannung wird an J28-DA0 respektive DA1 ausgegeben. Um die Funktion des folgenden Python-Programms zu verifizieren, schließen Sie an den beiden Pins von DA0 ein Multimeter zur Spannungsmessung an. Der Code entspricht weitgehend dem Beispiel aus Abschnitt 14.3, »Der Digital-Analog-Wandler MCP4811«. Beachten Sie aber, dass wir mit `spi.open(0, 1)` den SPI-Kanal 1 ansprechen, um den D/A-Wandler des Gertboards anzusprechen. Damit ergibt sich der folgende Python-Code:

```
#!/usr/bin/python3
import spidev
import time
import RPi.GPIO as GPIO

ce = 7
GPIO.setmode(GPIO.BCM)
GPIO.setup(ce, GPIO.OUT)
spi = spidev.SpiDev()
spi.open(0,1)

GPIO.output(ce, True)
GPIO.output(ce, False)
spi.writebytes([0b10110001, 0b00000000])
GPIO.output(ce, True)
```

Beachten Sie die Zeile `spi.writebytes([0b10110001, ..])`: Das erste Bit der Bitfolge, also das höchstwertige, ist in diesem Fall eine 1. Damit wird der Ausgang DA1 auf dem Gertboard angesprochen. DA0 erreichen Sie, wenn Sie das erste Bit in eine 0 abändern. Das ist ein Unterschied zum MCP4811, der hier zwingend eine 0 benötigte, da er nur einen Kanal besitzt.

Nach dem Start des Programms zeigt Ihnen Ihr Multimeter eine Spannung an. Die effektive Ausgangsspannung müssen Sie individuell für den auf Ihrem Gertboard verbauten D/A-Wandler errechnen (siehe Tabelle 16.1). Das komplette Datenblatt der MCP48X2-Familie finden Sie hier:

http://ww1.microchip.com/downloads/en/DeviceDoc/22249A.pdf

Modell	Verstärkungsfaktor	LSB-Größe
MCP4802 (n=8)	1×	2,048 V / 256 = 8 mV
	2×	4,096 V / 256 = 16 mV
MCP4812 (n=10)	1×	2,048 V / 1024 = 2 mV
	2×	4,096 V / 1024 = 4 mV
MCP4822 (n=12)	1×	2,048 V / 4096 = 0,5 mV
	2×	4,096 V / 4096 = 1 mV

Tabelle 16.1 Schrittweiten der MCP48X2-Familie. Der MCP3002 hat beispielsweise eine Schrittweite von 8 mV ohne aktivierten Gain-Modus (Verstärkungsfaktor). (Quelle: Datenblatt des Herstellers)

16.2 Der ATmega auf dem Gertboard

Neben den im vorigen Abschnitt beschriebenen Hardware-Funktionen ermöglicht das Gertboard auch die Nutzung eines Arduino-kompatiblen Mikrocontrollers. Das Gertboard enthält dazu einen vormontierten ATmega168- oder ATmega328-Mikrocontroller. Dadurch können Sie Arduino-Programme auf den Mikrocontroller laden und ausführen. Der auf unserem Gertboard verbaute ATmega328P hat 2 kByte RAM, einen Flash-Speicher von 32 kByte und arbeitet aufgrund der 3,3-V-Versorgungsspannung mit ca. 12 MHz.

Im Gertboard-Handbuch sind einige Beispielprogramme samt Verkabelung beschrieben. Eines davon nehmen wir in diesem Abschnitt als Beispiel und gehen detailliert auf die Hard- und Software-Installationen ein, die hierfür erforderlich sind. Die Pins des ATmega sind in der Leiste *J23* aufgeführt (siehe Abbildung 16.12).

Wenn Sie den vollen Umfang der Mikrocontroller-Funktionen nutzen möchten, führt kein Weg an der Lektüre des Datenblatts des ATmega vorbei:

http://www.atmel.com/images/atmel-2545-8-bit-avr-microcontroller-atmega48-88-168_datasheet.pdf

Hello World!

Als *Hello-World*-Projekt nutzen wir das Arduino-Programm *Blink*. Dieses lässt eine einfache LED auf dem Gertboard blinken, gesteuert allerdings durch den ATmega-Mikrocontroller. Denken Sie dieses Prinzip weiter, so kann der ATmega zahlreiche Aufgaben für Sie erledigen und den Raspberry Pi so entlasten. Zu den Einsatzmöglichkeiten zählen unter anderem:

- Analogwertverarbeitung
- Timer
- Counter
- Speichern von Zuständen im EEPROM
- Porterweiterung
- allgemeine Logikverarbeitung

Für unser Beispielprojekt beginnen wir mit der Verkabelung. Diese ist nur einmalig notwendig, um das Mikrocontroller-Programm über SPI vom Raspberry Pi auf den ATmega zu laden. Danach können bei Bedarf Leitungen entfernt werden.

Sie benötigen vier Jumper-Kabel, mit denen Sie die Verbindungen von der Leiste *J2* zur Leiste *J23* herstellen (siehe Abbildung 16.11). Die Pins der SPI-Schnittstelle sind leider nicht beschriftet – orientieren Sie sich also am Schaltplan und an dem Foto des Versuchsaufbaus (siehe Abbildung 16.12).

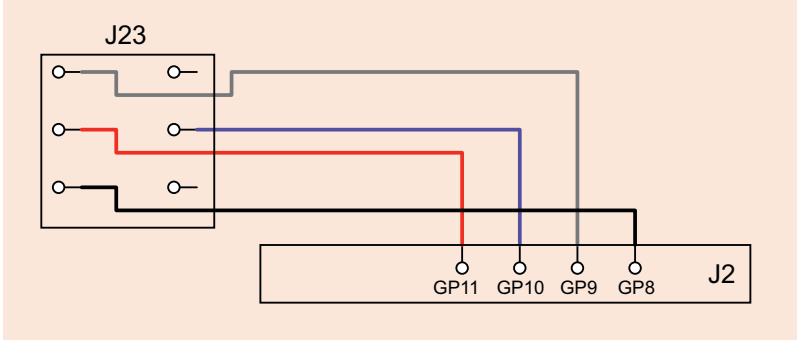


Abbildung 16.11 Schematische Darstellung der erforderlichen Gertboard-Verbindungen zur Programmierung des ATmega

Für unser Beispiel benötigen Sie außerdem ein Jumper-Kabel von J29-PB5 auf BUF1 in der darüberliegenden Leiste. Dieses Kabel verbindet einen Ausgang des ATmega mit der LED D1.

Stromversorgung für den Mikrocontroller

Setzen Sie im letzten Schritt einen Jumper auf die oberen beiden Pins der 3-Pin-Leiste J7. Dies versorgt die Bauteile auf dem Board mit einer 3,3-V-Spannung. Dieser Jumper wird im Gertboard-Handbuch nicht in den Schaltplänen dargestellt. Ohne die Verbindung schlägt aber jeder Programmierversuch fehl!

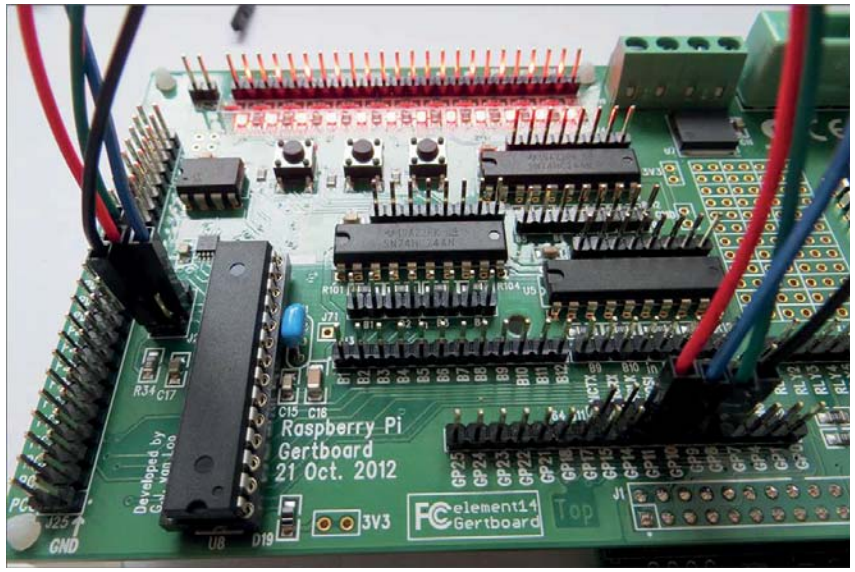


Abbildung 16.12 Die vier nötigen Leitungen von der SPI-Schnittstelle des Raspberry Pi zum ATmega. Der Mikrocontroller ist ganz links im Bild zu sehen.

avrdude

Zur Übertragung von Mikrocontroller-Code vom Raspberry Pi auf den Mikrocontroller des Gertboards ist ein spezielles Programm notwendig. Es wurde von Gordon Henderson entwickelt, dem die Raspberry-Pi-Community auch die WiringPi-Bibliothek zu verdanken hat:

https://projects.drogon.net/raspberry-pi/gertboard

Zur Installation führen Sie die folgenden Kommandos aus:

```
cd /tmp
wget http://project-downloads.drogon.net/gertboard/\
       avrdude_5.10-4_armhf.deb
sudo dpkg -i avrdude_5.10-4_armhf.deb
sudo chmod 4755 /usr/bin/avrdude
```

Im nächsten Schritt werden einige Einstellungen am System vorgenommen sowie ein paar Gertboard-spezifische Parameter im Arduino-IDE angepasst. Eine detaillierte Beschreibung aller Änderungen, die das Script `setup.sh` ausführt, finden Sie auf der oben erwähnten Webseite von Gordon Henderson.

```
cd /tmp
wget http://project-downloads.drogon.net/gertboard/setup.sh
chmod +x setup.sh
sudo ./setup.sh
```

Führen Sie nun mit `reboot` einen Neustart des Raspberry Pi durch. Sobald Sie sich wieder angemeldet haben, führen Sie das Kommando `avrsetup` aus:

```
avrsetup
```

Nun werden Sie nach Ihrem ATmega-Modell gefragt. Prüfen Sie die Bauteilbeschriftung, und drücken Sie [1] für ATmega328P oder [2] für ATmega168. Dem Mikrocontroller werden nun einige Parameter für den weiteren Betrieb übermittelt. Sollte die daraufhin erscheinende Meldung mit *Looks all OK - Happy ATmega programming!* enden, so ist die Einrichtung des Mikrocontrollers geglückt und Sie können mit der Programmierung beginnen.

Die Arduino-IDE

Nach diesen Vorbereitungsarbeiten erfolgt die eigentliche Mikrocontroller-Programmierung in der Arduino-IDE, also einer grafischen Entwicklungsumgebung. Diese installieren Sie wie folgt:

```
sudo apt-get install arduino
```

Da es sich hierbei um ein grafisches Programm handelt, müssen Sie spätestens jetzt einen Monitor an den Raspberry Pi anschließen oder eine VNC-Verbindung herstellen. Rufen Sie Elektronik • Arduino IDE im Startmenü Ihres Desktops auf. Nach dem ersten Start müssen Sie einmalig zwei Einstellungen vornehmen:

- Navigieren Sie zu Tools • Board, und aktivieren Sie den Eintrag Gertboard with ATmega168 (GPIO) beziehungsweise Gertboard with ATmega328 (GPIO) (siehe Abbildung 16.13).
- Außerdem wählen Sie unter Tools • Programmer den Eintrag Raspberry Pi GPIO als Programmierschnittstelle aus (siehe Abbildung 16.14).

Nun öffnen Sie mit Datei • Beispiele • 01 Basics • Blink das fertige Blink-Beispielprogramm. Sie sehen den C-Programmcode in einem neuen Fenster und können diesen gegebenenfalls verändern. Dieses Beispiel funktioniert so, wie es ist, daher können Sie direkt mit dem Upload fortfahren. Wählen Sie nun im neuen Fenster Datei • Upload mit Programmer, um das kompilierte Programm mit dem speziellen Raspberry-Pi-Uploader auf den Mikrocontroller zu übertragen (siehe Abbildung 16.15).

Erscheint keine Fehlermeldung im unteren Bereich des Fensters, so ist die Übertragung abgeschlossen. Die LED auf dem Gertboard blinkt nun.

Werfen Sie einen Blick in das englische Handbuch des Gertboard! In ihm sind einige weitere Beispielprogramme beschrieben. Dort finden Sie auch eine Menge Informationen zur Pin-Belegung sowie zu den Unterschieden in den Pin-Bezeichnungen

zwischen dem Arduino und der Mikrocontroller-Programmierung auf dem Gertboard:

http://www.element14.com/community/servlet/JiveServlet/previewBody/51727-102-1-265829/Gertboard_UM_with_python.pdf

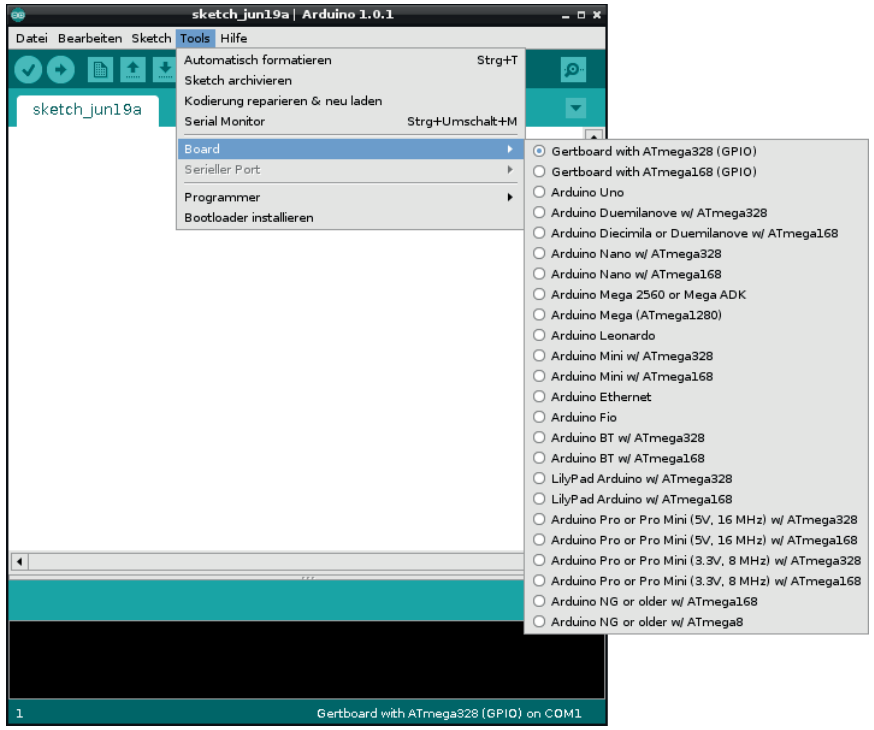


Abbildung 16.13 Auswahl des verwendeten Mikrocontrollers

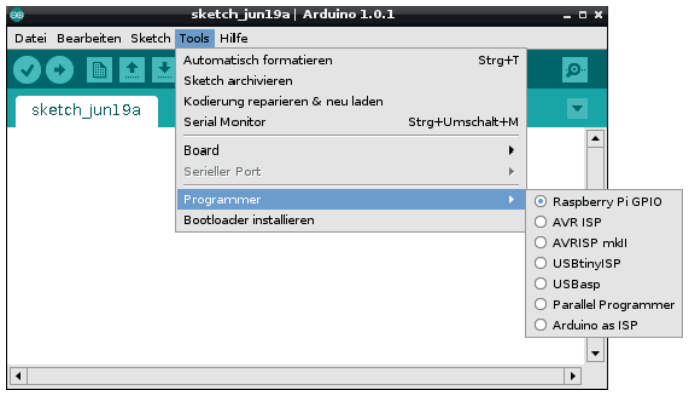


Abbildung 16.14 Wahl der Programmierschnittstelle

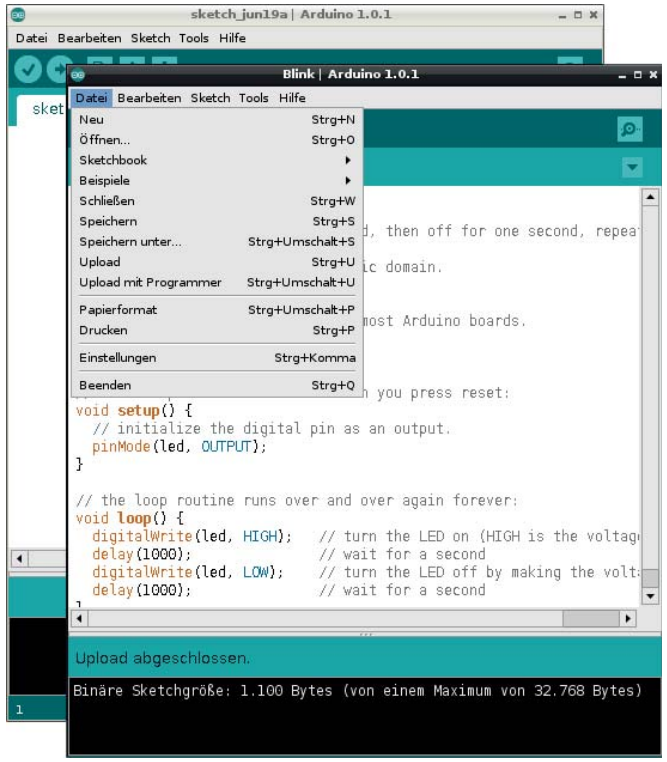


Abbildung 16.15 Upload des Programms auf den ATmega

16.3 Logic-Level-Converter

Nicht direkt eine Raspberry-Pi-Erweiterung, aber ein sehr nützlicher kleiner Helfer ist ein Logic-Level-Converter, zu Deutsch: Pegelwandler. Diese kleinen Module erleichtern die Kommunikation mit Plattformen unterschiedlicher TTL-Spannungslevel ungemein. Ein Arduino zum Beispiel führt an allen seinen Ausgangs-Pins 5 V. Ein direktes Signal vom Arduino zum Raspberry Pi würde das Aus für unseren geliebten Mini-PC bedeuten. Ein Pegelwandler kann zwischen die beiden Platinen geschaltet werden und wandelt so 5-V-Signale in 3,3-V-Signale und sogar umgekehrt um. Auch einige 5-V-Sensoren können so am Raspberry Pi verwendet werden.

Wir nutzen den 4-Kanal-Konverter von Adafruit. Dieser trägt den Namen *Adafruit 4-channel I2C-safe Bi-directional Logic Level Converter - BSS138* und unterstützt sogar die I^2C- und SPI-Kommunikation zweier Systeme mit unterschiedlichen Spannungspegeln.

Die Verwendung eines Pegelwandlers ist denkbar einfach. Das kleine Board besitzt eine *Low-Voltage*-Seite und eine *High-Voltage*-Seite. Jede Seite besitzt einen LV- bezie-

hungsweise HV-Pin, der mit dem jeweiligen Spannungspegel der verwendeten Hardware verbunden wird.

Im Beispiel »Raspberry Pi und Arduino« schließen Sie die 3,3 V des Raspberry Pi an LV und die 5 V des Arduino an HV an. Sehr praktisch: Die HV-Seite des Moduls kann bis zu 10 V vertragen. Die beiden GND-Pins verbinden Sie mit den Massen ihrer Geräte.

Wenn Sie nun an einen der mit A1–A4 markierten Pins einen GPIO-Pin des Raspberry Pi anschließen und auf High schalten, so führt der gegenüberliegende Pin (mit B1–B4 markiert) einen 5-V-Pegel.

Das von uns verwendete Modell ist bidirektional. Das bedeutet, dass im Umkehrschluss ein 5-V- bis 10-V-Pegel an einem Pin der HV-Seite 3,3 V auf der LV-Seite erzeugt. Somit ist eine gefahrlose Kommunikation zwischen dem Raspberry Pi und anderen Einplatinencomputern möglich (siehe Abbildung 16.16).

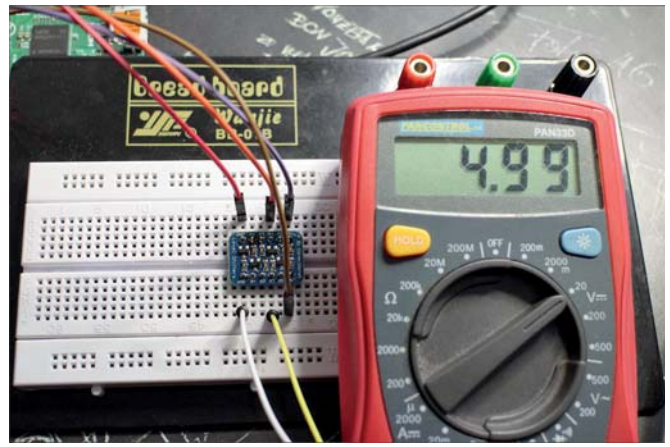


Abbildung 16.16 Ein Logic-Level-Converter wandelt die 3,3 V eines GPIO-Pins in 5 V um.

16.4 RasPiComm

Das RasPiComm ist eines der kleineren Erweiterungsboards für den Raspberry Pi. Der Hersteller *Amescon* bietet auf seiner Webseite Bezugsquellen, Software, Support und einige Beispielprogramme an:

http://www.amescon.com

Das RasPiComm ist eine kleine Platine, die auf die GPIO-Pin-Leiste gesteckt wird (siehe Abbildung 16.17). Entwickelt wurde sie für Modelle mit 26-poliger Stiftleiste. RasPiComm ist jedoch auch direkt kompatibel mit dem Raspberry Pi 3 (siehe Abbildung 16.18).

Einmal montiert und eingerichtet, bietet es folgende Features:

- eine RS-485-Schnittstelle
- eine RS-232-Schnittstelle
- eine Echtzeituhr mit Batterie
- einen I^2C-Steckverbinder
- zwei 5-V-Ausgänge
- einen 5-Wege-Joystick

Abbildung 16.17 Das auf das alte Modell B gesteckte RasPiComm

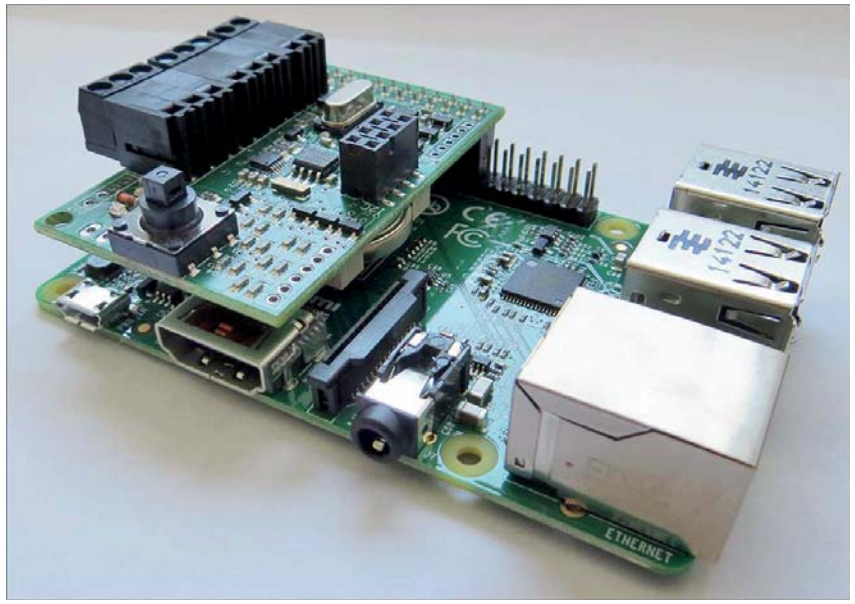


Abbildung 16.18 Das RasPiComm auf dem Modell 3

Nach dem Aufstecken des Boards laden Sie das Einrichtungs-Script des Herstellers herunter und führen es aus:

```
wget http://downloads.amescon.com/rpc_setup.sh
chmod +x ./rpc_setup.sh
sudo ./rpc_setup.sh
```

Dabei erscheint ein kleines Auswahlmenü, in dem Sie den Revisionsstand des Raspberry Pi ausgewählen müssen. Zudem können Sie alle Änderungen über dieses Menü wieder rückgängig machen. Nach einem Neustart des Raspberry Pi können die Funktionen des RasPiComm genutzt werden.

Der Joystick und die 5-V-Ausgänge

Die Taster des Joysticks sind mit fünf GPIO-Ports verbunden (siehe Tabelle 16.2). Die Richtungsangaben gelten aus der Aufnahmeperspektive der beiden obigen Fotos.

GPIO (BCM)	Richtung
4	unten
22	herunterdrücken
23	links
24	oben
25	rechts

Tabelle 16.2 Die Richtungen des Joysticks an den GPIO-Ports

Links neben dem Joystick befindet sich eine nicht bestückte 4-Pin-Leiste mit der Beschriftung *out*. Dabei handelt es sich um zwei Ausgänge mit jeweils einem GND-Anschluss. Sind die Ausgänge aktiviert, so schalten diese 5 V und können mit maximal 100 mA belastet werden. Laut Hersteller können z. B. 5-V-Relais direkt angeschlossen werden, da bereits Schutzdioden im RasPiComm integriert sind. Die Ausgänge sind auf die GPIO-Ports BCM 18 und BCM 21 gelegt.

Die Echtzeituhr

Die integrierte Hardware-Echtzeituhr verspricht eine Laufzeit von circa zehn Jahren. Auch diese Uhr wird über das Einrichtungs-Script eingebunden. Eine detaillierte Anleitung, wie Sie die Uhr stellen und auslesen, finden Sie in Abschnitt 15.10, »Hardware Real Time Clock«.

Die Schnittstellen

Die RS-485-Schnittstelle wird über den mittleren Block der schwarzen Schraubklemmen angeschlossen. Sie verspricht eine Baudrate von bis zu 230.400 Baud. In Linux ist die Schnittstelle unter `/dev/ttyRPC0` anzusprechen. Sie können diese Schnittstelle z. B. für spezielle Schrittmotoren mit RS-485-Elektronik nutzen.

Der untere Dreierblock der Schraubklemmen führt die RS232-Schnittstelle mit den Anschlüssen TX, RX und GND aus. Damit sind Geschwindigkeiten bis zu 115.200 Baud möglich. Nutzen können Sie diese Schnittstelle für die Kommunikation mit dem PC oder einem Mikrocontroller, beispielsweise mittels `minicom` (siehe Abschnitt 14.5, »UART«).

Der Schraubklemmenzweierblock, der auf den Fotos jeweils links oben zu sehen ist (siehe Abbildung 16.17 und Abbildung 16.18), beinhaltet je einen Anschluss für 5 V und GND. Wird der Raspberry Pi über die Mini-USB-Buchse versorgt, so liegen hier auch 5 V an. Umgekehrt kann der Raspberry Pi auch über diese beiden Klemmen mit seiner Betriebsspannung versorgt werden.

Die I^2C-Pins können über eine 2×4-Buchsenleiste genutzt werden. Die SPI-Schnittstelle ist als unbestücktes Lochraster ausgeführt. Anwendungsbeispiele finden Sie im offiziellen Forum oder im GitHub-Repository:

https://github.com/amescon/rpc-python-examples
http://www.amescon.com/de/forum.aspx

16.5 PiFace Digital 2

Das *PiFace Digital 2* ist ein kleines Erweiterungsboard für digitale Ein- und Ausgänge (siehe Abbildung 16.19). Es legt den Fokus auf das sichere Basteln mit externer Peripherie. Durch die vielen Schraub- und Steckverbinder kann auch der Lötkolben kalt bleiben und die gewünschte Hardware kinderleicht an das PiFace Digital angeschlossen werden. Das PiFace Digital 2 ist der Nachfolger des ersten Modells und wurde für das Layout des Raspberry Pi 3 sowie für die Modelle A+, B+ und 2 angepasst. Neben ein paar kleinen Änderungen im Layout der PiFace-Platine selbst ist die größte Neuerung der nun 40-polige Verbindungsstecker. Die Anzahl der Ein- und Ausgänge hat sich nicht geändert.

Falls Sie ein PiFace Digital der ersten Generation besitzen, können Sie dieses auch noch auf dem Raspberry Pi 3 verwenden. Nutzen Sie dazu einen Stacking-Header und kleine Kunststoff-Abstandshalter, damit die Platine sicher aufliegt. Da die ersten sechsundzwanzig Pins der alten Modelle dem neuen Modell 2 gleichen, sind keine Kompatibilitätsprobleme zu erwarten.

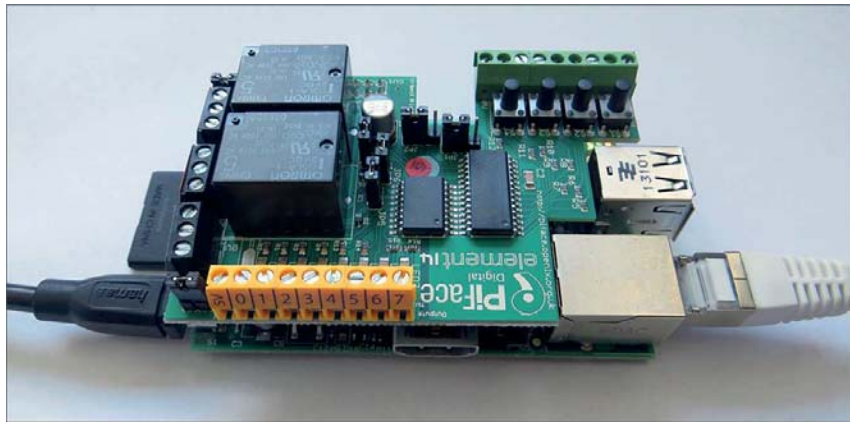


Abbildung 16.19 Das alte PiFace Digital auf dem Raspberry Pi B

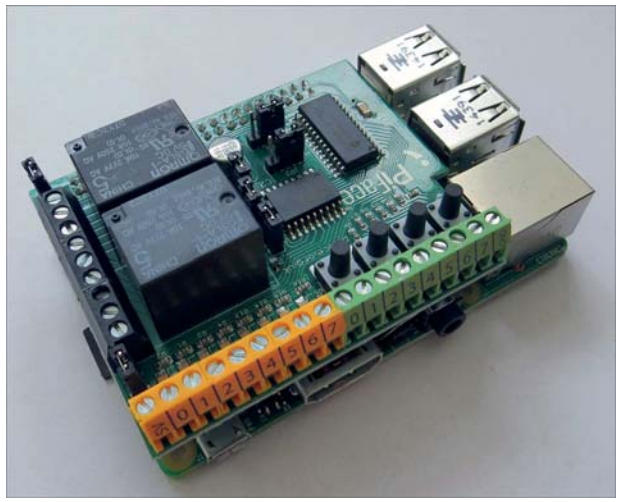


Abbildung 16.20 Das PiFace Digital 2 auf dem Raspberry Pi 3

Kaufinformationen, weiterführende Informationen und Herstellerhinweise sind auf der offiziellen Website zu finden:

http://www.piface.org.uk/products/piface_digital

Das PiFace Digital 2 bietet folgende technische Merkmale:

- zwei Wechslerrelais für maximal 20 V und 5 A
- vier Druckknöpfe, die als Signaleingang dienen
- acht frei belegbare digitale Eingänge
- acht Open-Collector-Ausgänge mit acht Status-LEDs
- einen grafischen Emulator des PiFace

Als Treiber-ICs sind ein MCP23S17 und ein ULN2803 verbaut. Die Installation erfolgt durch Aufstecken des Erweiterungsboards auf die GPIO-Steckerleiste. Das Board bedeckt den kompletten Raspberry Pi und enthält Aussparungen für die USB- und LAN-Buchsen.

Nach dem Aufstecken installieren Sie die dazugehörende Software. Diese beinhaltet unter anderem einen Emulator, auf den wir gleich eingehen, sowie eine Python-Bibliothek, die den Umgang mit den Ein- und Ausgängen sehr komfortabel gestaltet.

```
sudo apt-get install python3-pifacedigital-emulator
```

Ob die Installation erfolgreich verlaufen ist, testen Sie mit einem einfachen Beispielprogramm. Dazu führen Sie den nachfolgenden Befehl aus und achten auf die LEDs des PiFace:

```
python /usr/share/doc/python-pifacedigitalio/examples/blink.py
```

Sollte eine von ihnen nun blinken, ist alles in Ordnung. Andernfalls sollten Sie als Erstes überprüfen, ob Sie SPI aktiviert haben (siehe Abschnitt 14.1, »SPI«).

Der PiFace Digital Emulator

Der PiFace-Emulator mit dem Kommandonamen `pifacedigital-emulator` stellt das PiFace-Board auf dem Bildschirm dar und zeigt Eingangssignale und Tastendrücke an (siehe Abbildung 16.21). Zudem können Sie über Schaltflächen die Ausgänge des PiFace kontrollieren.

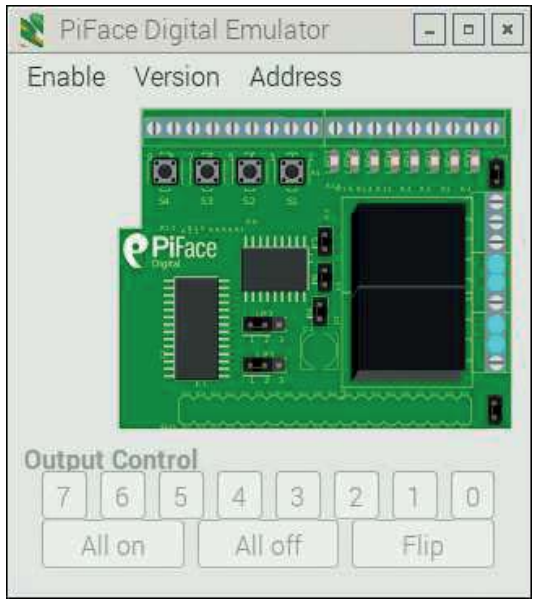


Abbildung 16.21 Das Emulations-Tool für PiFace Digital

Zum Ausprobieren aktivieren Sie über den Menüpunkt Enable die beiden Funktionen Output Control und Input Pullups. Nun drücken Sie einige der kleinen Taster auf dem PiFace-Board. Im Emulator sehen Sie, welcher Taster betätigt wurde. Wenn Sie mit der Maus eine der nummerierten Schaltflächen anklicken, so leuchtet am PiFace-Board die LED des entsprechenden Ausgangs. Der Ausgang ist also geschaltet. Die Relais sind den Ausgängen 0 und 1 zugeordnet. Klicken Sie auf diese beiden Schaltflächen, so hören Sie die Relais klacken.

Die PiFace-Bibliothek

Mit dem zuvor installierten Paket wurde eine bedienerfreundliche Python-Library installiert. Da im PiFace Digital 2 mit dem MCP23S17 ein Portexpander verbaut ist, würde sich die Handhabung des Boards ohne passende Bibliothek recht schwierig gestalten. Der MCP23S17 gleicht dem MCP23017, verwendet zur Kommunikation jedoch SPI statt I^2C. Die Python-Bibliothek verpackt die SPI-Kommunikation in praktische Funktionen. Damit kann das PiFace Digital 2 ähnlich komfortabel wie mit dem RPi.GPIO-Modul gesteuert werden.

Geladen wird die Bibliothek mit `import pifacedigitalio`. Um die Funktionen noch bequemer aufzurufen, geben Sie dem Ganzen noch einen kürzeren Namen. Importieren Sie die PiFace-Bibliothek z. B. als `pf`:

```
import pifacedigitalio as pf
```

Nach der Initialisierung durch `pf.init()` können die weiteren Funktionen verwendet werden. Wir beschreiben die Handhabung hier an einem kleinen Beispiel: Unser Ziel ist es, den Status der Taste *S1* auszulesen und das Relais *K0* entsprechend zu steuern. Sobald der Taster *S1* betätigt wird, zieht das Relais *K0* an; lassen Sie *S1* los, fällt auch das Relais wieder ab.

```
#!/usr/bin/python3
import pifacedigitalio as pf
from time import sleep

pf.init()
while True:
    if pf.digital_read(0):
        pf.digital_write(0, 1)
    else:
        pf.digital_write(0, 0)
    sleep(0.1)
```

Das Beispiel zeigt die grundlegende Handhabung der Bibliothek. Das Auslesen der Eingänge geschieht über `pf.digital_read(0)`. In diesem Fall wird der Status von Eingang 0 abgefragt. Ersetzen Sie die 0 durch einen Wert von 0 bis 7 für die entspre-

chenden acht Eingänge. Die Ausgabe der Funktion ist bei anliegendem Signal *1*, bei keinem Signal *0*. Dies kann auch einfach kontrolliert werden:

```
print pf.digital_read(0)
```

Die vier Onboard-Taster liegen an den Eingängen 0 bis 3 parallel. Damit können weiterhin externe Signale an diese Eingänge angeklemmt werden. Werden nun aber die Taster betätigt, wird das externe Signal womöglich überstimmt.

Das Schalten der Ausgänge erfolgt nach dem gleichen Schema. Die folgende Funktion setzt den Ausgang 0 von Low auf High:

```
pf.digital_write(0,1)
```

Der erste Übergabeparameter bestimmt den Ausgang (Wertebereich 0 bis 7), der zweite Parameter definiert den gewünschten Status: 1 für High, 0 für Low.

Die beiden Relais sind an den Ausgängen 0 und 1 parallel geschaltet. Trotzdem ist es möglich, die Ausgänge 0 und 1 der gelben Schraubklemmenleiste zu verwenden. Wichtig zu wissen ist, dass die acht Ausgänge kein Signal ausgeben, sondern im angesteuerten Zustand gegen Masse schalten. Die darüberliegenden LEDs zeigen immer den Schaltzustand der Ausgänge an.

Interrupts und Events in der PiFace-Bibliothek

Da alle Ein- und Ausgänge des PiFace durch SPI gesteuert werden, können die gewohnten Event- und Interrupt-Funktionen zur Überwachung eines Eingangs nicht verwendet werden. Aber auch hierfür hat die PiFace-Bibliothek eine hauseigene Lösung parat: Das folgende Python-Programm überwacht CPU-schonend die Eingänge 0 und 3, also die Taster *S1* und *S4*.

```
#!/usr/bin/python3
import pifacedigitalio as pf
import sys, time
pf.init()
pfc = pf.PiFaceDigital()

def an(event):
    pf.digital_write(0, 1)

def aus(event):
        pf.digital_write(0,0)

listener = pf.InputEventListener(chip=pfc)
listener.register(0, pf.IODIR_FALLING_EDGE, an)
listener.register(3, pf.IODIR_RISING_EDGE, aus)
listener.activate()
```



```
try:
    while True:
        time.sleep(5)
except KeyboardInterrupt:
    listener.deactivate()
    sys.exit()
```

In der aktuellen Version der PiFace-Bibliothek sind die Flanken genau entgegengesetzt zu verwenden. Ein Tasterdruck erzeugt eine fallende Flanke, das Loslassen eine steigende. Damit funktioniert das Programm wie folgt: Drücken Sie den Taster *S1*, so zieht das Relais *K0* an. Um das Relais zurückzusetzen, drücken Sie den Taster *S4* und lassen ihn wieder los. Beim Loslassen fällt das Relais wieder ab. Damit haben Sie beide möglichen Flanken durch Python erfasst.

PiFace Rack und die Jumper

Der PiFace-Hersteller bietet auch ein *PiFace Rack* an. Hierbei handelt es sich um eine Platine, mit der Sie bis zu vier PiFace-Platinen gleichzeitig an einen Raspberry Pi anschließen können. Aus technischer Sicht ist dies dank der SPI-Kommunikation kein Problem. Sollten mehrere Boards verwendet werden, so kann die Adresse jedes einzelnen Boards über die Jumper *JP1* und *JP2* in der Mitte des PiFace eingestellt werden (siehe Tabelle 16.3). Auch in Python kann mit der Bibliothek gezielt jedes Board angesprochen werden:

```
pf.digital_read(0, 1)     # liest Eingang 0 auf Board 1 aus
pf.digital_write(0, 1, 2) # setzt Ausgang 0 des Boards 2 auf High
```

Board-Nummer	JP1	JP2
0	0	0
1	1	0
2	0	1
3	1	1

Tabelle 16.3 Adressierung mehrerer PiFace-Boards durch Jumper

Weitere Infos zum PiFace Rack finden Sie auf der Herstellerseite:

http://www.piface.org.uk/products/piface_rack

Auf dem PiFace Digital 2 befinden sich noch fünf weitere Jumper:

- **JP3** stellt die Verbindung zur 5-V-Schiene des Raspberry Pi her.
- **JP4** versorgt die Freilaufdioden des Darlington-Arrays ULN2803. Dieser Jumper muss nur entfernt werden, wenn die zu schaltende Spannung an den Ausgängen höher als 5 V ist.
- **JP5 und JP6** können entfernt werden, um die Relais aus der Schaltung zu trennen.
- **JP7** schaltet beim Entfernen die Versorgung aller Ausgänge und der Relais ab.

PiFace Control and Display

Die Hersteller des PiFace bieten außerdem ein *PiFace Control and Display* an. Dabei handelt es sich ebenfalls um ein aufsteckbares Board, in das ein 16 × 2-LC-Display verbaut ist. Weitere Features sind ein Infrarotempfänger, ein 3-Wege-Navigationsknopf, fünf Taster sowie eine Python-Bibliothek zur Steuerung dieser Funktionen.

16.6 Quick2Wire Interface Board

Das *Quick2Wire Interface Board* ist ein verhältnismäßig kleines Erweiterungsboard. Es bietet nur wenige eigene Features, sondern legt den Fokus darauf, die Schnittstellen des Raspberry Pi einfach zugänglich zu machen. Das Quick2Wire-Board enthält:

- acht Ein- und Ausgänge
- eine I^2C-Schnittstelle
- zwei SPI-Schnittstellen
- eine serielle UART-Schnittstelle
- Anschlüsse für 3,3 V, 5 V und GND
- einen Taster und eine LED

Das Quick2Wire-Board wird über ein Flachbandkabel mit dem Raspberry Pi verbunden (siehe Abbildung 16.22). Die Steckverbinder des Kabels sind an den Enden etwas breiter und passen somit nicht auf die ersten 26 Pins des neuen Raspberry-Pi-Modells. Auch hier ist ein Stacking-Header notwendig (siehe Abbildung 16.23).

Das Board wird als Bausatz geliefert, das heißt, Sie müssen alle notwendigen Bauteile selbst auf die Leiterplatte löten. Das sollte allerdings auch mit Hobby-Löt-Equipment kein Problem darstellen, da die Löt-Pads großzügig und gut zugänglich gestaltet wurden. Unter dem folgenden Link finden Sie zudem eine detaillierte Anleitung zum Auflöten der Bauteile. Das Bestücken der Platine ist in ca. 30 Minuten erledigt.

Anleitungen, Bezugsquellen sowie den Schaltplan finden Sie auf der Herstellerseite:

http://quick2wire.com/products/quick2wire-interface-board-kit

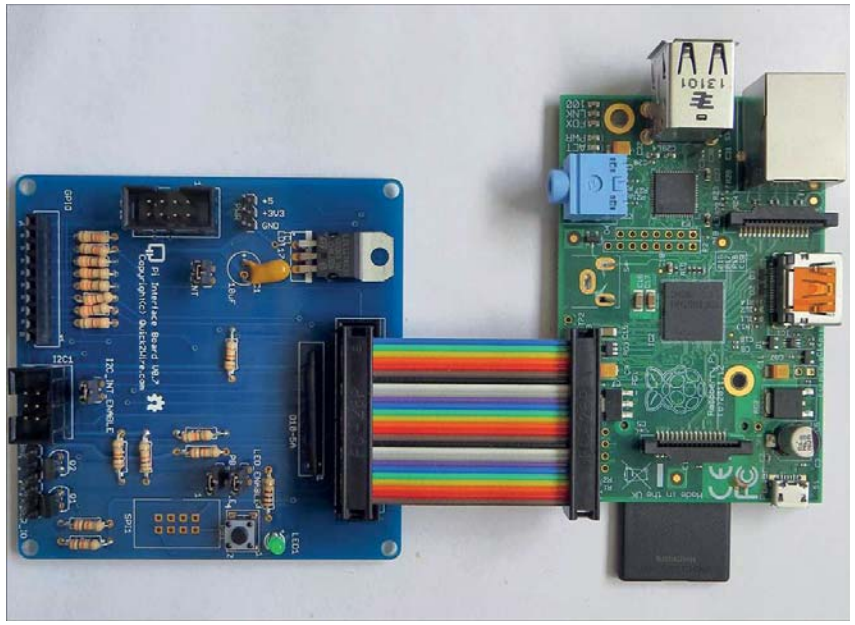


Abbildung 16.22 Das fertig bestückte Quick2Wire-Board mit dem Modell B

Abbildung 16.23 Das Quick2Wire-Board wird mit einem Stacking-Header an das Modell 3 angeschlossen.

Anschlüsse

Die acht ausgeführten GPIO-Pins (siehe Tabelle 16.4) sind durch strombegrenzende Widerstände sowie Schutzdioden gegen Fehlbeschaltung abgesichert. Die I^2C-Schnittstelle wird durch einen sechspoligen Wannenstecker einfach zugänglich gemacht. Dort sind SDA, SCL, 3,3 V, 5 V und GND als Pin verfügbar.

Bezeichnung bei Quick2Wire	GPIO-Pin (BCM)
P0 (Taster)	17
P1 (LED)	18
P2	27 (Rev2) bzw. 21 (Rev1)
P3	15
P4	23
P5	24
P6	25
P7	4

Tabelle 16.4 Quick2Wire-Ausgänge und ihre GPIO-Pin-Zuordnung

Auch die SPI-Schnittstellen SPI0 und SPI1 sind über Wannenstecker zugänglich. Es wird allerdings nur ein Stecker mitgeliefert, den wir an SPI0 aufgelötet haben.

Die UART-Kommunikation erfolgt über eine sechspolige Steckerleiste. Die Besonderheit hier ist, dass durch einen Pegelwandler auf dem Board auch 5-V-Komponenten angeschlossen werden können. Ein 5-V-Signal wird so auf 3,3 V reduziert, bevor es zum Raspberry Pi gelangt und dort Schaden anrichten kann.

Die LED ist mit Pin 12 (BCM 18) des Raspberry Pi verbunden. Parallel dazu liegt der Ausgang P1. Sofern Sie die WiringPi-Bibliothek installiert haben, können Sie die LED und den Ausgang P1 auf Anhieb testen:

```
gpio -g mode 18  out
gpio -g write 18 1
```

Die LED sollte nun leuchten. Parallel dazu führt der Pin 1 nun ein High-Signal. Sie können den Jumper *LED_ENABLE* abziehen, um die LED aus der Schaltung zu entfernen. Das Gleiche gilt für den Taster, der an Pin 11 (BCM 17) angeschlossen ist. Diesen deaktivieren Sie durch das Entfernen des Jumpers *PB_ENABLE*.

Die Jumper *SPI_INT* und *I2C_INT_ENABLE* aktivieren im gesetzten Zustand die entsprechenden Interrupt-Funktionen. Sie benötigen dafür einen Baustein, der Interrupt-Signale liefert. Sowohl das unten beschriebene *Quick2Wire-Port-Expander-Board* als auch die Python-3-Bibliothek von Quick2Wire unterstützen diese Funktion.

Quick2Wire-Bibliothek

Quick2Wire stellt eine eigene Python-Bibliothek zur Verfügung:

https://github.com/quick2wire/quick2wire-python-api

Die Bibliothek setzt Python 3 voraus. Der Einsatz der Bibliothek ist optional. Alle Funktionen des Quick2Wire-Boards können ebenso gut über die in diesem Buch erläuterten Standardverfahren genutzt werden.

Das Quick2Wire-Port-Expander-Board

Das *Quick2Wire-Port-Expander-Board* ist eine Erweiterung für das soeben vorgestellte Interface-Board (siehe Abbildung 16.24). Es besitzt einen MCP23017 als Port-Erweiterungsbaustein und ist durch das mitgelieferte Kabel einfach anzuschließen. Das Board stellt Ihnen 16 zusätzliche Ein- und Ausgänge zur Verfügung, ohne dass Sie sich die Schaltung aus Einzelkomponenten zusammenbauen müssen.

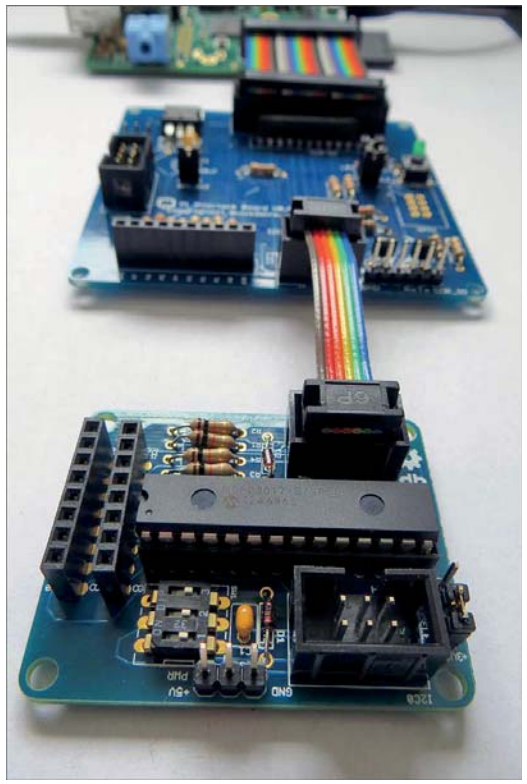


Abbildung 16.24 Im Hintergrund der Raspberry Pi, in der Mitte das Quick2Wire-Board, vorne der Quick2Wire-Port-Expander

Auch dieses Board wird als Bausatz geliefert und muss von Ihnen selbst verlötet werden. Quick2Wire bietet hierfür eine hervorragende Anleitung:

http://quick2wire.com/products/quick2wire-i2c-port-expander-board-kit

Einmal angesteckt, können die zusätzlichen Ports durch die Steckverbinder am Expander-Board erreicht werden. Zur Steuerung der neuen Ports verwenden Sie entweder die im vorigen Abschnitt erwähnte Quick2Wire-Bibliothek oder die Python-Bibliothek `smbus`, deren Einsatz wir bereits in Abschnitt 14.4, »I^2C«, erläutert haben.

16.7 StromPi 2 – USV und Wide-Range-Spannungsversorgung

Die Erweiterungsplatine StromPi 2 von Joy-IT widmet sich zwei Schwachstellen des Raspberry Pi zugleich. Zum einen erlaubt sie den Anschluss von Spannungsquellen von bis zu 61 V, zum anderen dient StromPi 2 als unterbrechungsfreie Stromversorgung (USV). StromPi 2 ist also die Platine der Wahl, wenn Sie Ihren Raspberry Pi an einer Autobatterie (12 V) im LKW (24 V) oder an anderen Spannungsquellen jenseits der 5 V betreiben möchten.

Die Platine kann bis zu 3 A an den Raspberry Pi liefern und enthält eine Reset-Funktion, auf die wir später noch eingehen werden. Auch ist es möglich, die USB-Ports des Raspberry Pi mithilfe des StromPi 2 zu High-Power-USB-Ports aufzurüsten, die dann bis zu 3 A Strom an angeschlossene USB-Geräte liefern können.

Installation und Modusauswahl

Um StromPi 2 mit dem Raspberry Pi zu verbinden, stecken Sie die Platine auf die GPIO-Leiste J8 (siehe Abbildung 16.25). Sie können nun durch die Position der beiden Modus-Jumperbrücken zwischen dem Wide-Range- und dem USV-Modus wählen:

- Im WIDE-Modus können Sie an der Schraubklemme eine Spannungsquelle im Bereich von 6 bis 61 V anschließen. Liegt diese Spannung an, so startet der Raspberry Pi und kann verwendet werden. StromPi 2 besitzt einen Spannungsregler, welcher die Winde-Range-Spannung zuverlässig auf 5 V herunterregelt.

  Sie können ebenfalls ein Micro-USB-Kabel an die mit *In* beschriftete Buchse des StromPi 2 einstecken. In diesem Fall wird die Spannungsquelle am Micro-USB-Anschluss bevorzugt. Fällt diese aus, so wird nahtlos auf den Wide-Range-Eingang umgeschaltet. Wenn Sie am Wide-Range-Eingang z. B. eine Autobatterie anschließen, so ist der Raspberry Pi sicher vor Stromausfällen geschützt.
- Im USV-Modus kann der Raspberry Pi nur über eine Versorgungsspannung am Micro-USB-Port *In* gestartet werden (siehe Abbildung 16.26). Es wird nur auf den Wide-Range-Eingang zurückgegriffen, wenn die Spannung am Micro-USB-Port ausfällt. Dieser Modus begünstigt die Verwendung eines kleinen LiPo-Akkus, da der Betrieb im USV-Modus nur 20 µA bis 80 µA benötigt.

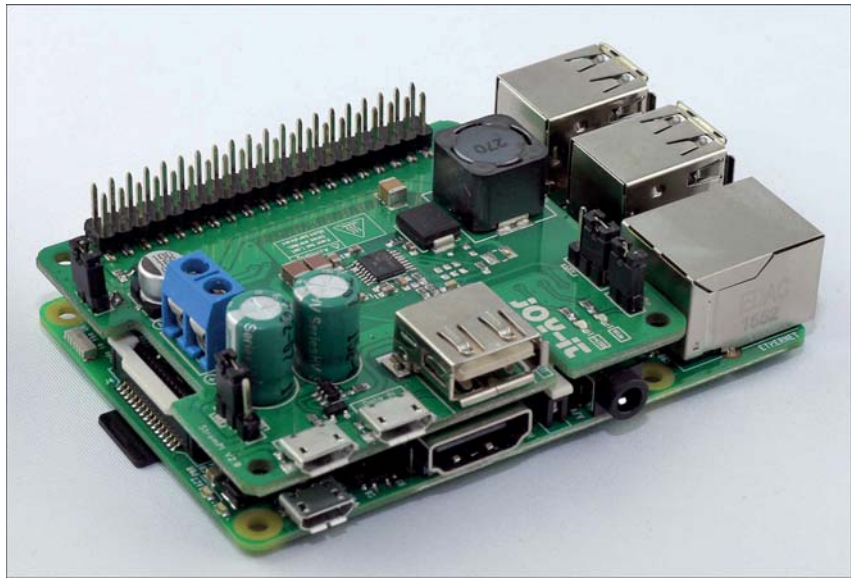
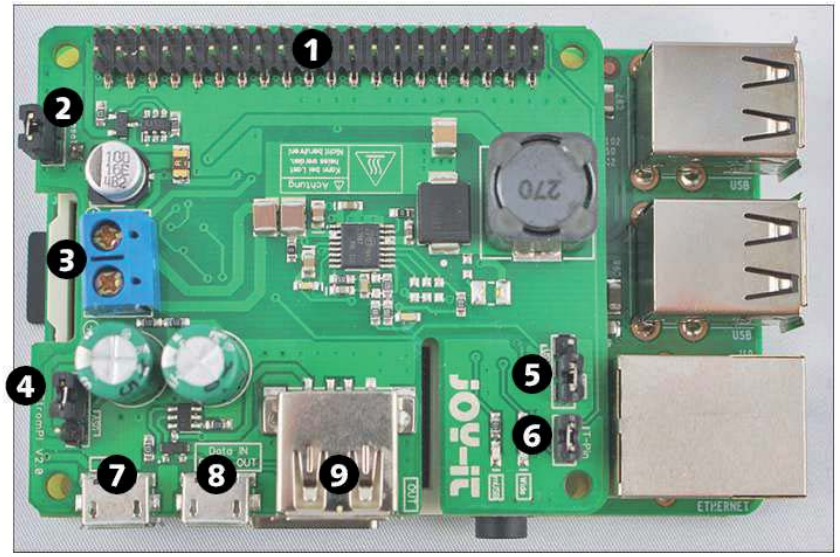


Abbildung 16.25 StromPi 2 passt perfekt auf den Raspberry Pi 2 und 3.



❶ GPIO-Port
❷ Reset-Umschalter
❸ Wide-Range-Spannungseingang (6 V bis 61 V) [sekundär]
❹ 2. Modus-Umschalter [USV|Wide]
❺ 1. Modus-Umschalter [USV|Wide]
❻ T-Pin
❼ Micro-USB-Spannungseingang (5 V) [primär]
❽ Daten-/Stromausgang zum Verbinden mit dem Raspberry Pi
❾ High-Power-USB-Ausgang

Abbildung 16.26 StromPi 2 im Überblick (Quelle: Joy-IT.net)

High-Power-USB und Reset

Die große USB-Buchse kann bereits als High-Power-USB-Port genutzt werden und liefert bis zu 3 A. Sie können auch die weiteren USB-Ports des Raspberry Pi aufrüsten, indem Sie mit einem Micro-USB-Kabel den Port *Data In Power Out* von StromPi 2 mit einer der USB-Buchsen des Raspberry Pi verbinden (siehe Abbildung 16.27). Die Voraussetzung hierfür ist natürlich, dass Ihre gewählte Primärspannungsquelle in der Lage ist, den benötigten Strom zu liefern.

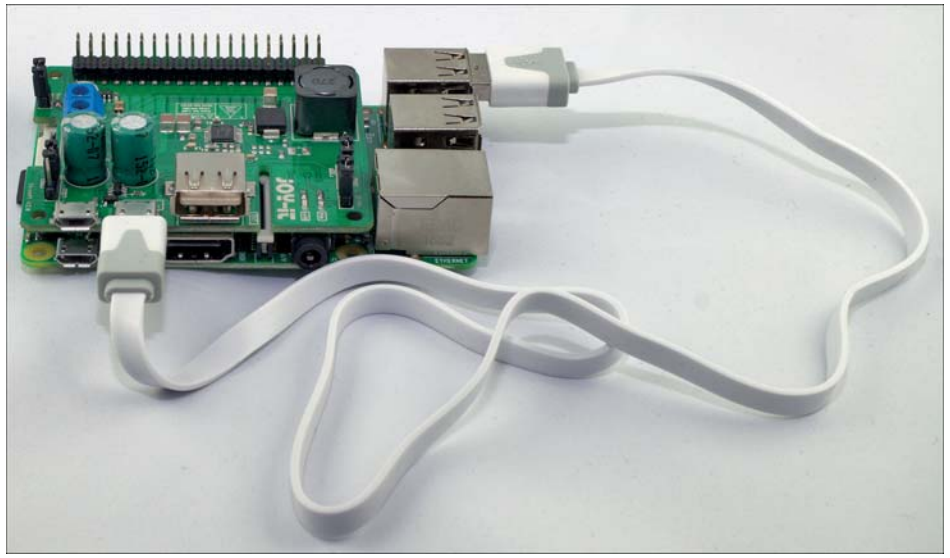


Abbildung 16.27 Durch diese Verbindung liefern auch die USB-Ports des Raspberry Pi bis zu 3 A Strom.

StromPi 2 kann den Raspberry Pi wieder neu starten, wenn eine unterbrochene Micro-USB-Spannung wiederhergestellt wurde. Dazu muss der Reset-Jumper auf StromPi 2 installiert sein (siehe Abbildung 16.26). Eine denkbare Anwendung ist das schonende Herunterfahren nach Ausfall der Hauptspannung und ein erneutes Hochfahren des Raspberry Pi, sobald diese wiederhergestellt wurde. Dieses Szenario können Sie mit der Software realisieren, die wir im folgenden Abschnitt beschreiben.

Die Software

Über die Webseite des Herstellers Joy-IT kann eine Software bezogen werden, die weitere Überwachungseinstellungen zulässt:

http://downloads.joy-it.net/download/37-strompi-2/90-strompi-2-software

Nach dem Download der Software erhalten Sie zwei kleine Python-Programme: `poweralarm.py` und `powershutdown.py`. Ersteres sendet eine E-Mail sobald StromPi 2 vom Micro-USB-Port auf den Wide-Range-Eingang umgeschaltet hat – mit anderen

Worten: sobald es einen Stromausfall gab. Sie müssen für diese Funktion nur die mitgelieferte Datei `sendmail.py` editieren und um die Zugangsdaten Ihres Mail-Accounts ergänzen.

Das zweite Programm fährt den Raspberry Pi nach Wegfall der Hauptspannungsversorgung am Micro-USB-Port schonend herunter, um Datenverluste zu vermeiden.

Beide Programme werden über den Wechsel der Versorgungspannung mithilfe des GPIO-Pins 21 benachrichtigt. Der Jumper *T-Pin* auf der StromPi-Platine stellt eine Verbindung zwischen dem Signalausgang von StromPi 2 und dem GPIO-Pin 21 des Raspberry Pi her. Wenn Sie diese Funktion nicht benötigen, den GPIO-Pin 21 anderweitig nutzen möchten oder einen eigenen Alarm-Pin nutzen möchten, so entfernen Sie den Jumper und nutzen gegebenenfalls ein Jumper-Wire zu einem beliebigen Pin. In diesem Fall müssen Sie jedoch die beiden Python-Programme an den neuen Pin anpassen, zum Beispiel `GPIO_TPIN = 3`.

Der Hersteller listet auf seiner Webseite eine Reihe von Bezugsquellen für den StromPi 2:

http://joy-it.net/bezugsquellen

16.8 GertDuino

Neben dem Gertboard hat der Entwickler Gert van Loo auch das *GertDuino* entworfen. Hierbei handelt es sich um einen Arduino-Klon, der auf den Raspberry Pi gesteckt werden kann (siehe Abbildung 16.28). Ähnlich wie die Erweiterung *Alamode* kann das GertDuino vom Raspberry Pi programmiert werden. Zudem bietet dieses Board zwei Knöpfe und insgesamt sechs LEDs, die mit den Ausgängen des GertDuino angesteuert werden können.

Zusätzlich zum verbauten ATmega-328 befindet sich noch ein ATmega-48 auf dem Board. Beide Mikrocontroller können auf die Schnittstellen des GertDuino zugreifen, unterscheiden sich aber in ihrer Taktfrequenz und im Stromverbrauch. Alle weiteren Spezifikationen, Anschlüsse und auch das Layout entsprechen dem Arduino Uno. Damit ist das GertDuino voll kompatibel zu allen Arduino-Uno-Shields.

Aber worin liegt nun der Reiz, diese beiden Systeme zu verbinden? Wozu einen Arduino auf den Raspberry Pi stecken, wenn Letzterer doch *fast* alles kann? Dem Arduino, also ebenso dem GertDuino, fehlt es sicherlich an der vergleichbar riesigen Rechenleistung des Raspberry Pi. Allerdings punktet das Arduino-System mit seiner einfachen Programmierbarkeit und der Vielfalt an digitalen und analogen Schnittstellen. Ein ATmega benötigt kein riesiges Betriebssystem wie der Raspberry Pi und kann dadurch zeitkritische Aufgaben präziser erledigen.

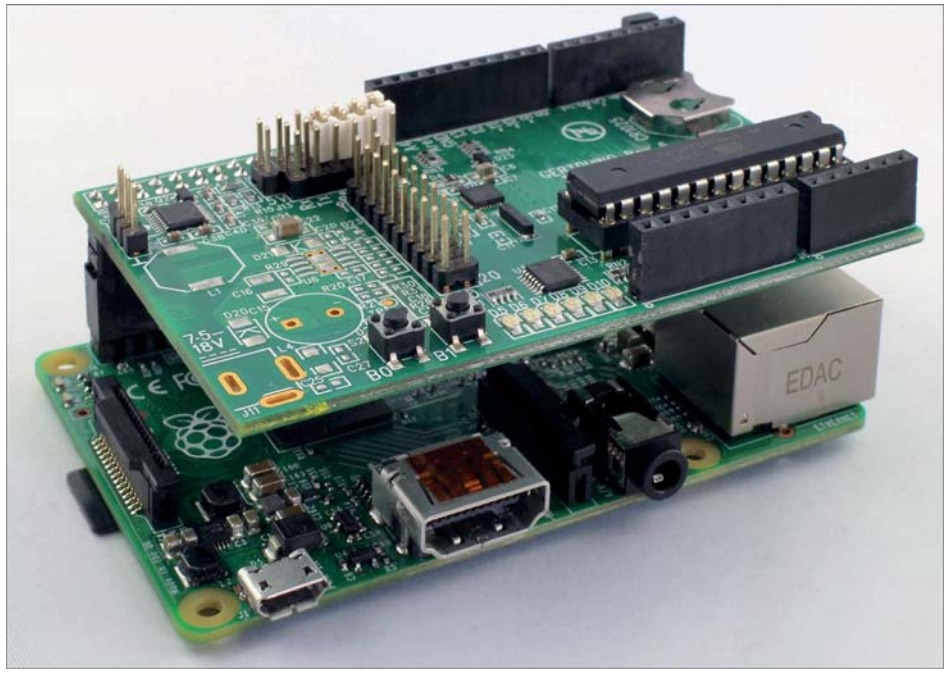


Abbildung 16.28 Das GertDuino auf dem Raspberry Pi 3

Im folgenden Beispiel nutzen wir einen analogen Eingang des GertDuino, um eine Spannung auszulesen. Diese übergeben wir über die serielle Schnittstelle an den Raspberry Pi. Spinnen Sie dieses Projekt weiter, so können Sie die vergleichbar einfache Aufgabe des *Analogwert-Auslesens* dem GertDuino überlassen, während der Raspberry Pi diese Werte auf einem Webserver zur Verfügung stellt. So werden die beiden Systeme ein perfektes Team, in dem jedes der beiden Boards seine Stärken ausspielen kann.

Die Einrichtung

Bevor Sie allerdings mit dem GertDuino arbeiten können, müssen Sie es einrichten. Zu allererst legen wir Ihnen die offizielle Anleitung ans Herz, die die Grundfunktionen und den Aufbau der Platine beschreibt:

http://www.farnell.com/datasheets/1778121.pdf

Beginnen Sie damit, das GertDuino auf den Raspberry Pi zu stecken. Stellen Sie sicher, dass der Raspberry Pi dabei ausgeschaltet ist. Sie wundern sich vielleicht anfangs über den nur 26-poligen Anschlussstecker. Dieser passt sowohl auf die alten Raspberry-Pi-Modelle als auch auf das aktuelle Modell 3. Die fehlenden Pins bedeuten keinerlei Funktionseinbußen, da die Kommunikation zwischen dem GertDuino und dem Raspberry Pi über die UART-Schnittstelle verläuft. Diese ist in den ersten 26 Pins mit inbegriffen.

Nun installieren Sie die Arduino-IDE und *avrdude*, wie wir dies in Abschnitt 16.2, »Der ATmega auf dem Gertboard«, beschrieben haben.

Die vier mitgelieferten Jumper stecken Sie auf die Programmierstellung. Zusätzlich stellen Sie durch zwei gekreuzte Brücken die UART-Verbindung zwischen Raspberry Pi und dem ATmega328 her (siehe Abbildung 16.29).

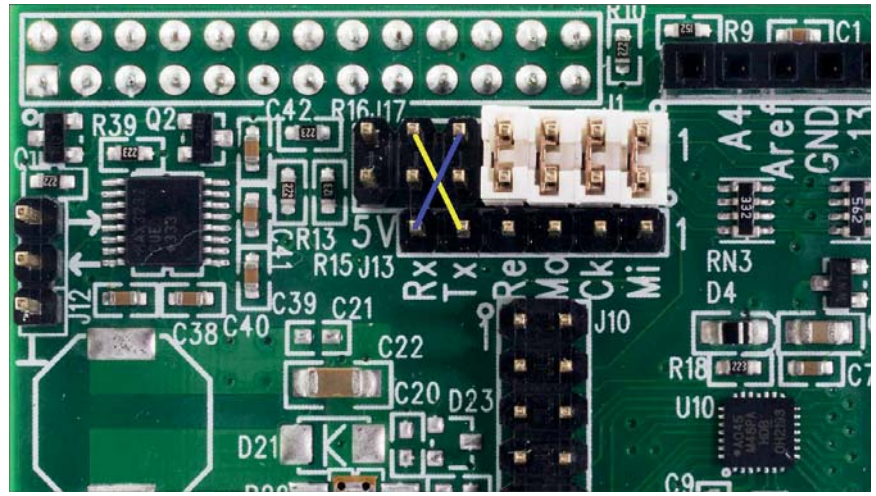


Abbildung 16.29 Alle vier Programmier-Jumper und in Gelb/Blau die benötigte UART-Verbindung

Anschließend muss der ATmega-328 des GertDuino auf eine Taktfrequenz von 16 MHz gestellt werden. Dies geschieht über ein Kommando, das im Terminal auszuführen ist:

```
avrdude -qq -c gpio -p atmega328p -U lock:w:0x3F:m \
  -U efuse:w:0x07:m - U lfuse:w:0xE7:m -U hfuse:w:0xD9:m
```

Die Entwicklungsumgebung starten Sie im Raspbian-Menü mit Entwicklung • Arduino-IDE. Die Einrichtung und der Upload der Programme erfolgt ebenfalls so wie in Abschnitt 16.2, »Der ATmega auf dem Gertboard«, beschrieben.

Bei unseren Versuchen gab es allerdings zunächst ein Problem mit der Kommunikation: So war es anfangs nicht möglich, eine *saubere* UART-Kommunikation aufzubauen, da sowohl der Raspberry Pi als auch das GertDuino nur Wirrwarr empfingen. Der Fehler lag in einem fehlerhaften Parameter der Arduino-IDE. Sie können das Problem bereits beheben, bevor es auftritt. Dazu laden Sie die Anpassungen von Gordon Henderson herunter:

```
cd /tmp
wget http://project-downloads.drogon.net/gertboard/boards.txt
wget http://project-downloads.drogon.net/gertboard/\
  programmers.txt
```

In einem Editor suchen Sie in `boards.txt` die Zeile `gert328.build.f_cpu=12000000L` und ändern den Wert in `16000000L`. Durch die Anpassung stimmt nun auch die Taktfrequenz der Programmkonfiguration mit den tatsächlichen 16 MHz des ATmega328 überein. Danach speichern und schließen Sie die Datei und kopieren beide Dateien in den Arduino-IDE-Ordner.

```
cd /usr/share/arduino/hardware/arduino
sudo mv boards.txt board.txt.bak
sudo mv /tmp/boards.txt .
sudo mv programmers.txt programmers.txt.bak
sudo mv /tmp/programmers.txt .
```

Der Arduino Sketch

Beginnen wir nun mit dem Arduino Sketch. Dieses Programm fällt sehr simpel aus, da es nur einen Wert ausliest, diesen in eine Spannung umrechnet und über UART verschickt.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int analog = analogRead(A0);
  float spannung = analog *(5.0/1023.0);
  delay(1000);
  Serial.println(spannung);
}
```

Der Pin A0 des GertDuino erwartet nun eine Spannung von 0 bis maximal 5 V, die Sie z. B. mit einem Potenziometer als Spannungsteiler erzeugen können. Falls Sie Hilfe beim Anschluss des Potenziometers benötigen, hilft ein Blick in die sehr gute Arduino-Datenbank. Unter folgendem Link finden Sie auch einen Anschlussplan für den Arduino Uno und das Potenziometer. Die Anschlüsse sind direkt auf das GertDuino übertragbar:

http://www.arduino.cc/en/Tutorial/ReadAnalogVoltage

Vergewissern Sie sich, dass sich die vier Jumper noch immer in der Programmierstellung befinden und dass die Arduino-IDE wie in Abschnitt 16.2, »Der ATmega auf dem Gertboard«, beschrieben eingestellt ist. Laden Sie das Programm nun über den Menüpunkt Datei • Upload mit Programmer auf das GertDuino.

Das Python-Programm auf dem Raspberry Pi

Das kleine C-Programm befindet sich nun im ATmega328. Im nächsten Schritt erstellen Sie ein Python-Script für den Raspberry Pi. Es soll den Inhalt der zuvor gesendeten Variable `spannung` empfangen:

```
#!/usr/bin/python3
# Datei gertduino.py
import serial, import time, RPi.GPIO as GPIO

# Reset-Modus des Arduino abschalten
GPIO.setmode(GPIO.BCM)
reset = 8
GPIO.setup(reset, GPIO.OUT)
GPIO.output(reset, GPIO.HIGH)
ser = serial.Serial("/dev/ttyAMA0", timeout = 10)
ser.baudrate = 9600
while True:
    daten = ser.readline()
    print("Spannung am Arduino: ", daten.decode('utf-8'))
    time.sleep(.5)
ser.close()
```

Um das Programm fehlerfrei ausführen zu können, muss der UART-Port freigeschaltet sein und *pySerial* installiert werden. Folgen Sie dazu den Anweisungen in Abschnitt 14.5, »UART«.

Zu Beginn des Programms wird GPIO 8 aktiviert. Dies ist nötig, damit bei gesetztem Reset-Jumper (*Re* der vier Programmier-Jumper) das GertDuino-Board aus dem Reset-Modus aufwacht. Alternativ können Sie vor dem Start des Programms den Jumper entfernen. Lesen Sie die letzten Seiten der zuvor verlinkten GertDuino-Anleitung für weitere Details.

Programme starten

Mit dem Start des Python-Programms lösen Sie also den ATmega aus dem Reset-Schlaf. Dadurch beginnt dieser sofort mit der Ausführung des übertragenen Sketches. Das GertDuino-Programm läuft nun in einer Endlosschleife und sendet jede Sekunde den aktuellen Spannungswert von A0 an den Raspberry Pi. Eine Konsolenausgabe des Python-Programms zeigt Ihnen ebenfalls im Sekundentakt diesen Wert an (siehe Abbildung 16.30).

```
root@pi:/python# python3 gd.py
Spannung am Arduino:  1.52

Spannung am Arduino:  1.52

Spannung am Arduino:  1.53

Spannung am Arduino:  1.53

Spannung am Arduino:  1.54

Spannung am Arduino:  1.55

Spannung am Arduino:  1.55
```

Abbildung 16.30 Konsolenausgabe des Python-Programms

16.9 Raspberry-Pi-HATs

Etwa gleichzeitig mit dem Erscheinen des Raspberry Pi B+ und der damit verbundenen Layout-Änderung der Platine hat die Raspberry-Pi-Stiftung eine Richtlinie zur Standardisierung von Erweiterungsboards entworfen. Dieser Standard nennt sich *HAT*, was für *Hardware Attached on Top* steht. Das Ziel dieses Standards ist es, dass Anwender zukünftige Erweiterungsboards sehr einfach in Betrieb nehmen können und die Kompatibilität gewährleistet wird.

Nun ist nicht jeder Hersteller verpflichtet, seine Boards nach dem HAT-Standard zu entwerfen. Jedoch darf er sein Board dann auch nicht HAT nennen. Was braucht man also, damit ein Board ein richtiger HAT wird?

- **Mechanische Spezifikationen:** Der HAT-Standard gibt die Geometrie der Erweiterungsplatine vor. So benötigt ein HAT die richtigen Abmaße, Bohrungen sowie die passende Form und Aussparungen. Zudem *muss* das Board Kontakte für die komplette 40-polige GPIO-Leiste enthalten.
- **Elektrische Spezifikationen:** Ein Board muss Vorgaben zur I^2C0-Schnittstelle sowie zur Möglichkeit des Backpowering erfüllen, um sich HAT nennen zu dürfen. Dazu zählen ein EEPROM sowie die Strombelastbarkeit beim Backpowering.
- **Das EEPROM:** Speziell zu erwähnen ist das für HATs erforderliche EEPROM. Ein EEPROM ist ein kleiner Speicherbaustein. Dieser soll in Zukunft einzigartige Informationen zu jedem HAT beinhalten. Dazu gehören eine Identifikationsnummer, Herstellerinformationen sowie eine GPIO-Konfiguration. Letztere soll beim Booten des Raspberry Pi mit aufgestecktem HAT die GPIO-Pins bereits korrekt konfigurieren. Somit entfällt das manuelle Einrichten der benötigten Pins (auch interner Pull-ups), und es vermeidet Fehler, wie z. B. Eingangs-Pins, die eigentlich Ausgänge sein müssten.

Alle Spezifikationen und bislang verfügbaren Infos zum Thema HATs finden Sie im offiziellen GitHub-Repository:

https://github.com/raspberrypi/hats

Prototyping HATs

Prototyping HATs sind leere, unbestückte Leiterkarten, die den HAT-Standard erfüllen. Das heißt, dass diese Boards auf die 40-polige Steckerleiste aufgesteckt werden können und sich genau mit der äußeren Form des Raspberry Pi 3 decken (siehe Abbildung 16.31).

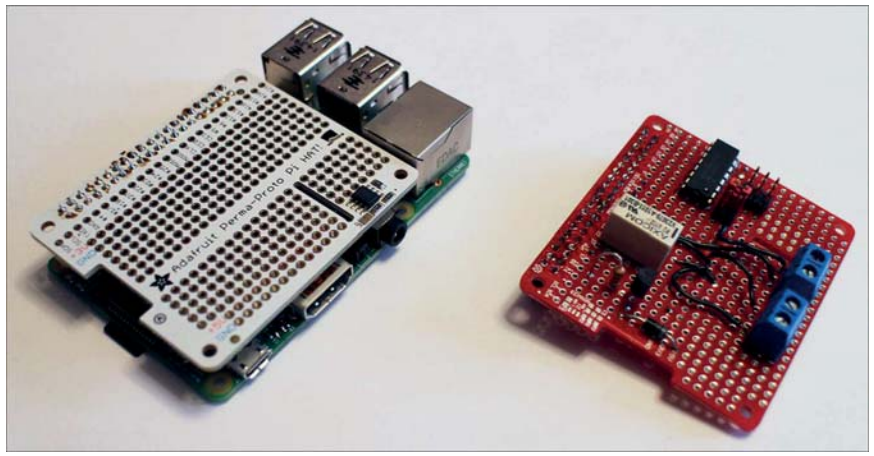


Abbildung 16.31 Zwei Prototyping-HATS: links auf dem Raspberry Pi 3, rechts bereits mit einer kleinen Schaltung bestückt

Diese kleinen und recht preiswerten HATs sind ideal für Bastelprojekte, da sie sehr einfach verlötet werden können und aufgrund des geringen Preises problemlos für jedes neue Projekt angeschafft werden können. Das spart Platz und sorgt für Ordnung im Leitungs-Chaos.

Wir haben uns das *Watterott Prototyping-HAT* sowie das *Adafruit Perma-Proto Pi HAT* angesehen. Produktinformationen finden Sie unter den folgenden Links:

http://www.watterott.com/de/RPi-Proto-HAT-ID-EEPROM
https://www.adafruit.com/products/2314

Beide Boards werden, sofern gewünscht, mit bereits bestücktem EEPROM, jedoch ohne verlöteten Steckverbinder verkauft. Jedoch sind die vierzig Lötpunkte schnell abgearbeitet. Es gibt jeweils bereits belegte Lochreihen für GND, 5 V und 3,3 V. Alle anderen Löcher können von Ihnen beliebig für Ihre Schaltung verwendet werden.

HATs und das EEPROM

Auch wir haben der neuen EEPROM-Unterstützung, die der Raspberry Pi ab Modell B+ bietet, anfangs zu wenig Aufmerksamkeit geschenkt. Einsteiger sind mit diesem Thema sicherlich auch ein wenig überfordert, da es bislang nur sparsame Informationen hierzu gibt. Doch das soll kein Hindernis sein. Das HAT-EEPROM verdient Aufmerksamkeit und bringt einige tolle Funktionen auf die HAT-Boards.

Beginnen wir mit einem kleinen Beispiel: Sie entwerfen eine Schaltung, die einige LEDs leuchten lassen soll und ein Relais schalten kann. Diese Schaltung löten Sie auf ein leeres Prototyping-HAT, das ein EEPROM besitzt. Sobald alles wie gewünscht funktioniert, möchten Sie Ihrem eigenen Erweiterungsboard einige Einstellungen mitgeben, die permanent auf dem Board gespeichert bleiben. Kramen Sie das Board nämlich nach einiger Zeit wieder aus der Bastelkiste und stecken es auf Ihren Raspberry Pi, so wird dieser direkt beim Boot-Vorgang alle nötigen Einstellungen für Ihr Board laden und ist somit direkt einsatzbereit.

So weit die Theorie. Die Praxis erfordert jedoch ein wenig Geduld.

EEPROM flashen

Wir gehen nun davon aus, dass Sie ein Board mit einem korrekt angeschlossenen EEPROM verwenden. Sollten Sie ein eigenes EEPROM nutzen und anschließen wollen, so schauen Sie sich zunächst die wichtigen Informationen zum EEPROM-Typ sowie zur Verdrahtung an:

https://github.com/raspberrypi/hats

In der Elektronik nennt man das Beschreiben von digitalen Speicherbausteinen *flashen*. Bei diesem Vorgang wird eine entsprechende Datei auf das EEPROM geladen und dort gespeichert.

Jedes HAT-EEPROM verfügt über eine I^2C-Schnittstelle. Diese wird jedoch nicht an die bekannten I^2C-Pins 3 und 5 angeschlossen, sondern an den I^2C-Bus 0. Dieser befindet sich an den Pins 27 und 28 und soll auch *nur* für die EEPROM-HATs verwendet werden. Um auf den I^2C-Bus 0 zugreifen zu können, bedarf es einer kleinen Änderung in der `/boot/config.txt` in Form eines Device-Tree-Overlays. Öffnen Sie dazu die oben genannte Datei:

```
sudo nano /boot/config.txt
```

Fügen Sie im unteren Teil der Datei, jedoch vor dem ersten `dt`-Parameter, die folgende Anweisung in einer eigenen Zeile ein: `dtparam=i2c0=on`. Die resultierende Datei sieht dann beispielsweise so aus:


```
# Datei /boot/config.txt
# Additional overlays and parameters are documented
# in /boot/overlays/README
start_x=0
dtparam=i2c0=on
dtparam=spi=on
dtparam=i2c_arm=on
```

Nach dem Speichern der Datei sollten Sie den Raspberry Pi neu starten, um die Änderungen wirksam zu machen.

Nun laden Sie vom offiziellen HAT-Repository auf GitHub die *EEPROM-Utils* herunter. Dies sind drei kleine Programme, die das Beschreiben und Auslesen des EEPROMs ermöglichen.

```
sudo git clone https://github.com/raspberrypi/hats/tree/master/eepromutils
```

Wechseln Sie nun in das heruntergeladene Verzeichnis, und installieren Sie die EEPROM-Utils:

```
cd hats/eepromutils
make
chmod +x eepflash.sh
```

Im nächsten Schritt schauen wir uns den zukünftigen Inhalt des EEPROMs etwas genauer an. Eine Beispieldatei, die wir für unsere Zwecke abändern werden, haben Sie mit dem vorangegangenen `git-clone`-Befehl bereits auf Ihren Raspberry Pi geladen. Öffnen Sie die Datei `eeprom_settings.txt` im Ordner `eepromutils`:

```
sudo nano eeprom_settings.txt
```

Der obere Teil der Datei trägt den Titel *Vendor Info* und beinhaltet Informationen, die ein Hersteller eines HATs seinem Produkt mitgeben kann. Das sind z. B. eine einzigartige Produkt-ID sowie der Produktname und der Herstellername. Wir passen vier Zeilen dieses Bereichs an unser Vorhaben an:

```
# 16-bit product id
product_id 0x0001

# 16-bit product version
product_ver 0x0002

# ASCII vendor string  (max 255 characters)
vendor "Max Mustermann"

# ASCII product string (max 255 characters)
product "Mein erster Hut"
```

Der darauffolgende Abschnitt beinhaltet nun die für uns wichtigen Informationen. Hier können Sie Einstellungen zum GPIO-Port vornehmen.

- **gpio_drive:** Dieser Parameter kann Werte von 0 bis 8 beinhalten und bestimmt die Strombegrenzung der frei verwendbaren GPIO-Pins. 0 behält die Standardwerte bei, 1–8 entsprechen 2, 4, 6, 8, 10, 12, 14, 16 mA.
- **gpio_slew:** Die Slew-Rate bestimmt die Anstiegs- oder Abfallgeschwindigkeit einer Ausgangsspannung. Kurz gesagt: Sie bestimmt die Zeit, die Ihr GPIO-Pin von Low auf High benötigt. In der Regel ist diese Zeit zu vernachlässigen. Im Extremfall jedoch, wenn Sie sehr viele GPIOs auf einmal einschalten, können kurzzeitig sehr große Ströme fließen, die zum Einbruch der Versorgungsspannung führen können. Um die Anstiegszeit zu begrenzen und so einen *sanfteren* Wechsel der Zustände zu ermöglichen, können Sie mit diesem Parameter die Slew-Rate begrenzen. Der Parameter versteht die Werte 0 = Standard, 1 = Ein (Slew-Rate-Limitierung) 2 = Aus (Keine Slew-Rate-Limitierung).
- **gpio_hysteresis:** Der Wechsel eines Eingangs-Pins vom Zustand Low zu High findet bei einem minimal anderen Spannungspegel statt als der Wechsel von High zu Low. Durch dieses Phänomen gibt es einen kleinen Spannungsbereich, der einen undefinierten Zustand erzeugt. Durch die *Schalthysterese* kann dieses Problem vermieden werden. Mögliche Werte sind:
  - 0 = Standard
  - 1 = Hysterese
  - 2 = Hysterese Aus

  Die Slew-Rate und die Schalthysterese erfordern in der Regel nur Änderungen bei speziellen Sensoren. Ob und wie Sie die Werte anpassen müssen, entnehmen Sie dem Datenblatt Ihres Sensors. Beide Werte stehen im Zusammenhang mit dem sogenannten Schmitt-Trigger. Um zu verstehen, worauf es hier ankommt, sollten Sie sich mit dem Schmitt-Trigger-Prinzip vertraut machen:

  https://de.wikipedia.org/wiki/Schmitt-Trigger
- **back_power** legt fest, ob Ihr HAT das Backpowering nutzt, den Raspberry Pi also über die GPIO-Leiste mit 5 V versorgt. Ein Wert von 0 sollte eingetragen werden, falls Sie den Raspberry Pi ganz normal über die Micro-USB-Buchse versorgen. Eine 1 besagt, dass Ihr Board mindestens 1,3 A liefern kann. Der Wert 3 sollte genutzt werden, wenn Ihr HAT mindestens 2 A liefert.

Der letzte Parameterblock bestimmt nun die Funktion der genutzten GPIO-Pins. Der BCM-Pin 5 (physischer Pin 29) kann nun mit der folgenden Zeile als Ausgang definiert werden:

```
setgpio  5     OUTPUT     DEFAULT
```


Der erste Parameter dieser Zeile beschreibt den BCM-Pin, der zweite Wert die Funktion (`INPUT`, `OUTPUT`, `ALT0-ALT5`). Im dritten Parameter können Sie die internen Pull-up-/Pull-down-Widerstände hinzuschalten. Hier werden `UP`, `DOWN`, `NONE` sowie `DEFAULT` erwartet. `Default` lässt die Einstellung unverändert. Die folgende Zeile definiert den BCM-Pin 27 als Eingang mit aktiviertem Pull-up-Widerstand:

```
setgpio 27    INPUT    UP
```

Die beiden oben genannten Zeilen haben wir nun angepasst, und danach speichern wir die Datei. Achten Sie darauf, dass Sie in der Beispieldatei das #-Zeichen vor der entsprechenden Zeile entfernen. Alle anderen Zeilen können Sie unverändert in der Datei belassen.

Im nächsten Schritt erzeugen Sie aus der Textdatei eine `.eep`-Datei, die vom EEPROM verarbeitet werden kann. Hierzu nutzen Sie das Programm `eepmake` aus den EEPROM-Utils:

```
sudo ./eepmake eeprom_settings.txt eeprom_settings.eep
```

Im Ordner sollte nun die Datei `eeprom_settings.eep` vorhanden sein. Diese kann jetzt in das EEPROM geflasht werden:

```
sudo ./eepflash.sh -w -t=24c32 -f=eeprom_settings.eep
```

Bestätigen Sie die folgende Sicherheitsabfrage mit `yes`. Nach wenigen Sekunden gibt das kleine Programm `Done.` zurück und der Flash-Vorgang ist erfolgreich beendet. Starten Sie nun den Raspberry Pi neu, so sind die beiden Pins bereits im zuvor eingestellten Modus.

Sie können auch jederzeit den EEPROM-Inhalt aus dem Baustein auslesen. Nutzen Sie dafür das Programm *eepflash* mit geänderten Parametern:

```
sudo ./eepflash.sh -r -t=24c32 -f=backup.eep
```

Mit diesem Befehl wird das EEPROM ausgelesen und die Datei `backup.eep` erstellt. Sie ist in diesem Zustand allerdings noch nicht lesbar. Mithilfe von *eepdump* erzeugen Sie aus den Informationen die lesbare Textdatei `backup.txt`:

```
sudo ./eepdump backup.eep backup.txt
```

Eigener Device-Tree-Eintrag

Um einen eigenen Device-Tree-Eintrag zu erstellen und somit auch Treiber für Ihr Board zu laden (falls notwendig), müssen Sie sich sehr gut mit dem Betriebssystem Linux auskennen. Möchten Sie es dennoch versuchen, so finden Sie in dieser Anleitung aus dem englischen Raspberry-Pi-Forum einige erste Schritte:

https://www.raspberrypi.org/forums/viewtopic.php?f=29&t=108134

16.10 Adafruit PWM/Servo-HAT

Die Entwickler von Adafruit haben mit dem PWM/Servo-HAT ein sehr nützliches HAT entwickelt, das Hobby-Bastlern bei der Ansteuerung mehrerer Servomotoren hilft. Zudem kann das Board genutzt werden, um 16 unabhängige PWM-Signale zu erzeugen. Falls die zwei Hardware-PWM-Pins (Pins 12 und 33) des Raspberry Pi 3 für Ihr Projekt nicht ausreichen, so ist dieses Erweiterungsboard eine sehr gute Alternative. Das HAT kommuniziert über I^2C mit dem Raspberry Pi. Der auf dem Board verbaute Controller empfängt lediglich die Einstellparameter und erzeugt dann, unabhängig vom Raspberry Pi, die gewünschten Pulsweitensignale.

Das Board finden Sie direkt bei Adafruit:

https://www.adafruit.com/products/2327

Nachdem Sie es ausgepackt haben, müssen Sie nur noch die Steckerleisten in die Leiterplatte löten. Das ist zwar ein wenig Fleißarbeit, geht aber schnell von der Hand. Nach dem Aufstecken auf den Raspberry Pi ist das Board betriebsbereit (siehe Abbildung 16.32).

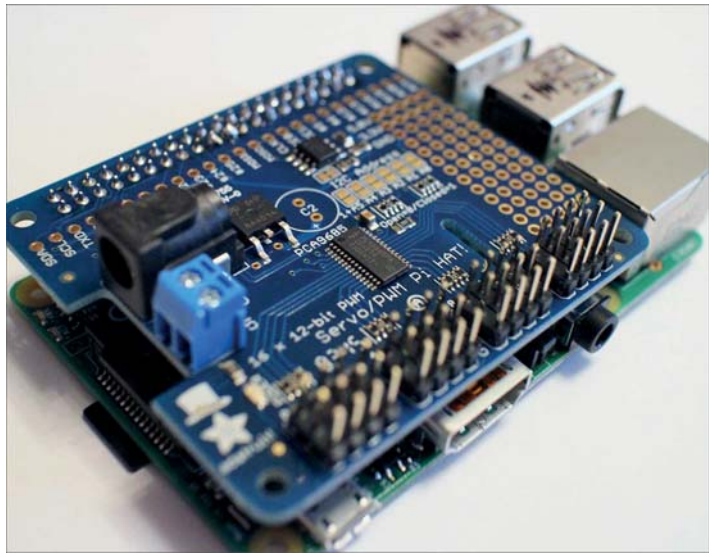


Abbildung 16.32 Das PWM/Servo-HAT auf dem Rücken des Raspberry Pi 3

PWM-Signale erzeugen

Prüfen Sie zuerst, ob das HAT erfolgreich am I^2C-Bus erkannt wird. Nutzen Sie dafür die *i2c-tools*. Das Board sollte an der Adresse `0x40` sichtbar sein. Stellen Sie vorab sicher, dass Sie die I^2C-Schnittstelle korrekt eingerichtet haben (siehe Abschnitt 14.4, »I^2C«).

Die Ausgabe von `i2cdetect -y 1` sollte Ihnen nun das Board an `0x40` anzeigen (siehe Abbildung 16.33).

```
pi@pi ~ $ 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: 70 -- -- -- -- -- --
```

Abbildung 16.33 Das PWM/Servo-HAT wurde an Adresse 0x40 erkannt.

Das Generieren von PWM-Signalen ist mit der Adafruit-Python-Bibliothek ein Kinderspiel. Laden Sie sich die Code-Sammlung auf Ihren Raspberry Pi:

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

Nun wechseln Sie in den Ordner für das PWM/Servo-HAT:

```
cd Adafruit-Raspberry-Pi-Python-Code
cd Adafruit_PWM_Servo_Driver
```

Dort finden Sie die Dateien `Adafruit_I2C.py` sowie `Adafruit_PWM_Servo_Driver.py`. Um ein lauffähiges Python-Script zu schreiben, das die Adafruit-Bibliothek nutzt, ist es notwendig, dass Sie die beiden Dateien in denselben Ordner kopieren, in dem Ihr zukünftiges Python-Script abgelegt wird.

Leider sind die Bibliotheken noch nicht an das neue Python 3 angepasst. Daher nutzen wir hier Python 2. Mit ein wenig Fleißarbeit können Sie allerdings die beiden oben genannten Dateien an die Python-3-Syntax anpassen.

Erzeugen Sie nun die Datei `pwm-hat.py`, und füllen Sie diese mit folgendem Inhalt:

```
#!/usr/bin/python
# Datei pwm-hat.py
from Adafruit_PWM_Servo_Driver import PWM
import time

pwm = PWM(0x40)
pwm.setPWMFreq(1000) # 1000 Hz
pwm.setPWM(15, 0, 410) # 10 % Duty Cycle
pwm.setPWM(8, 0, 1024) # 25 % Duty Cycle
pwm.setPWM(4, 0, 2048) # 50 % Duty Cycle
time.sleep(2)
pwm.setAllPWM(0, 0)
```

In diesem kleinen Programm werden die Kanäle 4, 8 und 15 des PWM/Servo-HATs angesprochen. Auf jedem Kanal wird ein unterschiedliches Signal erzeugt. Nach zwei Sekunden werden alle Signale gestoppt.

Die gesamte Bibliothek bietet drei Funktionen, die Ihnen die Möglichkeit geben, das Board in vollem Umfang zu nutzen:

- **setPWMFreq(freq):** Führen Sie diese Funktion aus, bevor Sie den Duty Cycle einstellen, denn Sie erzeugen mit ihr eine Frequenz, auf die sich der Duty Cycle bezieht. Hier sind Werte von 40 bis 1000 erlaubt, wobei dieser Wert direkt für die Frequenz in Hertz steht.
- **pwm.setPWM(Kanal, An, Aus):** Diese Funktion startet ein PWM-Signal mit dem eingestellten Duty Cycle. Dieser wird in den Parametern `An` und `Aus` definiert. Sie geben für `An` den Startpunkt und für `Aus` den Endzeitpunkt des High-Pegels an.

  Der integrierte Controller bietet eine Auflösung von 12 Bit. Aus diesem Grund können Sie den Maximalwert von 4096 angeben. Möchten Sie nun einen Duty Cycle von 25 % einstellen, so geben Sie für `An` den Wert 0 ein, für `Aus` den Wert 2048. Mit einem einfachen Dreisatz können Sie nun jeden beliebigen Duty Cycle wählen. Außerdem müssen Sie noch den Parameter `Kanal` füllen. Dieser gibt den PWM-Kanal des Boards an, auf dem das Signal erzeugt werden soll, und erwartet einen Wert zwischen 0 und 15 (16 Kanäle).
- **pwm.setAllPWM(An, Aus):** Mit dieser Funktion stellen Sie an allen 16 Kanälen das gewünschte Signal ein. Handhaben Sie diese Funktion genau wie die vorige, jedoch ohne Angabe des Kanals.

Dauerbetrieb?

Durch das Ausführen der obigen Funktionen wird ein Befehl über den Bus an den Controller auf dem HAT geschickt. Dieser nimmt sofort seine Arbeit auf und setzt sie danach endlos fort. Nutzen Sie daher die Funktionen `setPWM(Kanal, 0, 0)` beziehungsweise `setAllPWM(0,0)`, um die Signalerzeugung wieder zu beenden.

Servomotoren ansteuern

Der zweite Verwendungszweck dieses Boards ist das Ansteuern von Servomotoren. Das Prinzip dieser Motoren und die Ansteuerung ohne ein Erweiterungsboard haben wir Ihnen bereits im Abschnitt zu den Motoren gezeigt (siehe Abschnitt 13.5, »Servomotoren«).

Das PWM/Servo-HAT bietet eine Hohlbuchse, an die ein externes Netzteil angeschlossen werden kann. Dort kann eine Spannung von 5 bis 6 V genutzt werden, um die Motoren zu versorgen. Prinzipiell kann ein an das Board angeschlossener Servomotor auch ohne Netzteil funktionieren. Er wird dann über die 5 V des Raspberry Pi versorgt. Da Motoren allerdings in der Regel sehr viel Strom benötigen, empfehlen wir, die externe Versorgung vorzuziehen.


```
#!/usr/bin/python
# Datei servo-hat.py
from Adafruit_PWM_Servo_Driver import PWM
import time
pwm = PWM(0x40)
servoMin = 150
servoMax = 600
pwm.setPWMFreq(60)

while (True):
  pwm.setPWM(5, 0, servoMin)  # Minimale Endlage
  time.sleep(1)
  pwm.setPWM(5, 0, servoMax)  # Maximale Endlage
  time.sleep(1)
```

16.11 BrickPi

Die Erweiterung *BrickPi* stellt eine Verbindung zwischen dem Raspberry Pi und dem *LEGO Mindstorms*-System her. Damit können Sie vier NXT- und EV3-Motoren sowie fünf digitale sowie analoge LEGO-Sensoren anschließen und vom Raspberry Pi aus steuern bzw. auslesen. Das Steuern bzw. Automatisieren von LEGO-Fahrzeugen oder -Robotern wird damit besonders einfach.

Die Installation erfolgt auch hier wieder durch das Aufstecken der BrickPi-Platine auf den Raspberry Pi. Die Stromversorgung von BrickPi ist so entworfen worden, dass mehrere Methoden möglich sind. So kann z. B. ein Batterie-Pack verwendet werden, um die selbst gebauten Roboter mobil zu halten. Alternativ ist die Versorgung via Micro-USB möglich.

Kompatibilitätsproblem mit EV3-Sensoren

BrickPi funktioniert tadellos mit Motoren des LEGO-Mindstorms-EV3-Systems. Lediglich die Sensoren sind momentan noch nicht mit BrickPi kompatibel. Der Grund dafür ist ein spezielles Protokoll der EV3-Sensoren, das von den Entwicklern von BrickPi noch nicht implementiert wurde.

Das Team und die Community arbeiten an einigen Beta-Versionen der Software. Daher lohnt sich ein Blick in das betreffende Forenthema, um hier auf einem aktuellen Stand zu bleiben:

http://www.dexterindustries.com/forum/?topic=ev3-and-brickpi

BrickPi gibt es in unterschiedlichen Versionen zu kaufen. So gibt es Pakete, die den Raspberry Pi bereits enthalten. Des Weiteren ist ein Gehäuse verfügbar, das direkt auf LEGO- bzw. LEGO-Technics-Bausteine aufgesteckt werden kann. So kann die Platine nahtlos in den LEGO-Aufbau integriert werden (siehe Abbildung 16.34 bis Abbildung 16.36).

Abbildung 16.34 BrickPi samt Gehäuse für die nahtlose Adaption von LEGO-Bausteinen (Quelle: Dexter Industries)

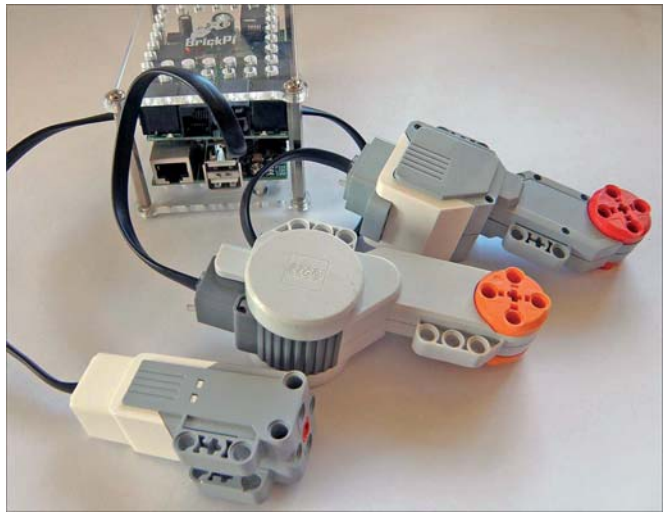


Abbildung 16.35 Die Anbindung der LEGO-Motoren an BrickPi erfolgt durch die Standardleitungen. (Quelle: Dexter Industries)

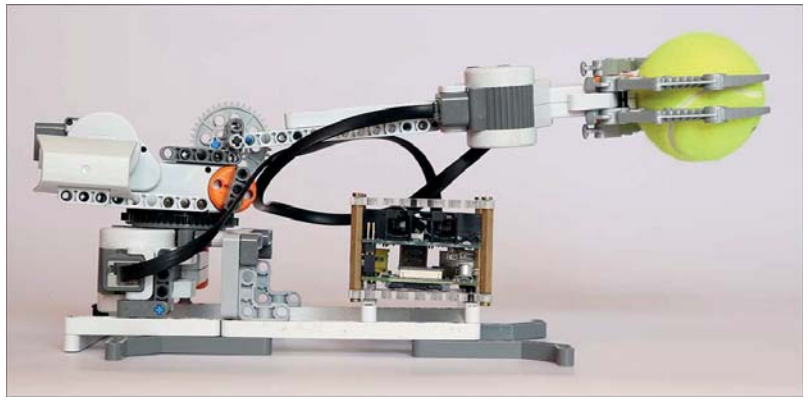


Abbildung 16.36 LEGO-Greifarm für einen Tennisball (Quelle: Dexter Industries)

16.12 GrovePi

GrovePi führt die Idee des *Plug & Play* noch einen Schritt weiter und ermöglicht es, Sensoren und Aktoren mithilfe von genormten Steckern durch simples Anstecken an die GrovePi-Platine zu nutzen (siehe Abbildung 16.37).

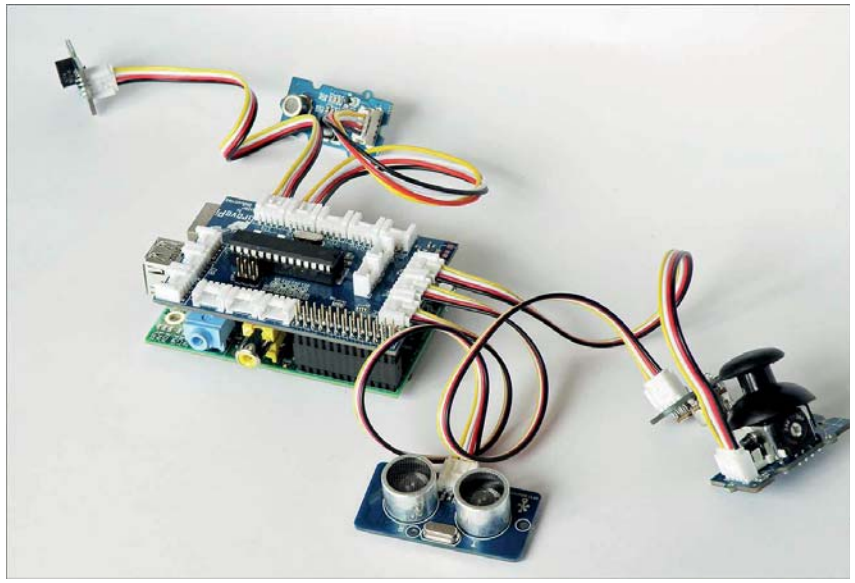


Abbildung 16.37 Das GrovePi-System samt angeschlossener Module (Quelle: Dexter Industries)

GrovePi ist nicht nur ein Erweiterungsboard, sondern ein komplettes System. Das System entstand aus dem von *Seeedstudio* entwickelten System *Grove*. Dieses stellt mehr

als 100 Module zur Verfügung, die durch Plug & Play an den Arduino angeschlossen werden können (siehe Abbildung 16.38).

Dexter Industries führte dieses Konzept weiter und entwickelte eine Methode, um die vorhandenen Komponenten auch dem Raspberry Pi zugänglich zu machen. Das eigentliche Board stellt die Steckverbindungen zur Verfügung und sorgt mit einem ATmega 328P für die Kompatibilität zwischen Raspberry-Pi- und Arduino-Modulen. Möchten Sie nun die Einfachheit dieses Systems nutzen, so benötigen Sie die Sensoren und Aktoren aus der Grove-Reihe. Die Auswahl ist mittlerweile recht vielfältig und umfasst unter anderem:

- Taster
- LEDs
- Schalter
- Relais
- Temperatursensoren
- Geräuschsensoren
- Displays
- Berührungssensoren
- Alkoholsensoren

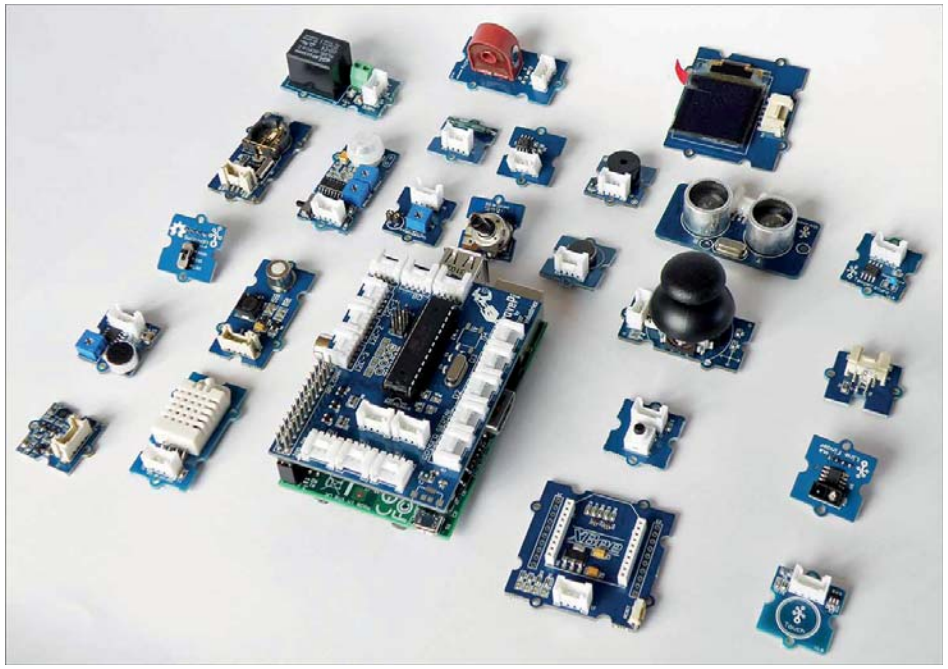


Abbildung 16.38 Die Auswahl an unterstützten Fertigmodulen ist riesig. (Quelle: Dexter Industries)

Die komplette Auswahl aller unterstützten Komponenten finden Sie bei Dexter Industries:

http://www.dexterindustries.com/GrovePi/sensors/supported-sensors

GrovePi wurde für den Raspberry Pi B entworfen, ist allerdings auch mit dem Raspberry Pi 3 kompatibel:

http://www.dexterindustries.com/GrovePi/get-started-with-the-grovepi/raspberry-pi-model-b-grovepi

Kapitel 22

C-Programmierung

Die Programmiersprache C ist eine Low-Level-Programmiersprache. In der Linux-Welt wird C von vielen Kernel- und Treiberprogrammierern eingesetzt – vor allem deswegen, weil C-Code oft viel effizienter ausgeführt wird als vergleichbarer Code anderer Programmiersprachen. Leider ist die Entwicklung von C-Programmen vergleichsweise mühsam und fehleranfällig.

Da es für die eingebauten Funktionen des Raspberry Pi und für nahezu alle Erweiterungskomponenten bereits fertige Treiber und vielfach sogar Python-Module gibt, ist der Einsatz von C für die meisten Raspberry-Pi-Projekte überflüssig. Das gilt auch für alle in diesem Buch vorgestellten Raspberry-Pi-Erweiterungen und -Projekte! Das erklärt auch die Kürze dieses Kapitels. Es stellt Ihnen den GNU-C-Compiler, das Programm `make` sowie die Bibliotheken *WiringPi* und *bcm2835* vor.

22.1 Hello World!

Der GNU-C-Compiler

Als Beispiel zum Kennenlernen des C-Compilers dient wie in den anderen Programmierkapiteln »Hello World!«. Die einzige Aufgabe dieses minimalistischen Programms besteht darin, eine Zeichenkette auszugeben. Den Code geben Sie in einem beliebigen Editor ein:

```
/* Datei hellow.c */
#include <stdio.h>
int main(void) {
  printf("Hello World!\n");
}
```

Das Programm müssen Sie nun mit dem GNU-C-Compiler (`gcc`) kompilieren. Die Option `-o` gibt dabei an, in welche Datei der ausführbare Code geschrieben werden soll. Das resultierende Programm `hellow` wird automatisch als ausführbar markiert. Sie können es in der Schreibweise `./name` ausführen, die Sie bereits aus den vorangegangenen Kapiteln kennen.

```
gcc -o hellow hellow.c
./hellow
  Hello World!
```

make

Größere C-Programme bestehen aus vielen `*.c`- und `*.h`-Dateien, wobei die `*.c`-Dateien den eigentlichen Code enthalten und die `*.h`-Dateien (Header-Dateien) öffentliche Funktionsdefinitionen und gemeinsame Konstanten beinhalten.

Das Kompilieren erfolgt in zwei Schritten: Zuerst wird jede `*.c`-Datei für sich zu einer Objektdatei mit der Dateikennung `*.o` kompiliert, und danach werden alle `*.o`-Dateien zu einem Programm verbunden (gelinkt). Diese Vorgehensweise hat insbesondere den Vorteil, dass bei einer Änderung in *einer* Datei nicht alle Dateien neu kompiliert werden müssen, sondern nur die geänderten Dateien.

Nehmen wir an, Ihr Projekt besteht aus den folgenden fünf Dateien:

```
/* Datei func1.h */
#define THENUMBER 42
int func1(void);

/* Datei func1.c */
#include "func1.h"
int func1(void) {
  return(THENUMBER);
}

/* Datei func2.h */
int func2(int a, int b);

/* Datei func2.c */
int func2(int a, int b) {
  return(a+b);
}

/* Datei main.c */
#include <stdio.h>
#include "func1.h"
#include "func2.h"
int main(void) {
  int x, y;
  x = func1();
  y = func2(x, 7);
  printf("Ergebnis %d\n", y);
}
```


Wenn Sie immer alles neu kompilieren möchten, wie es bei kleinen Projekten oft der Fall ist, reicht weiterhin ein `gcc`-Aufruf. Dabei gibt `-I` den Pfad zu eigenen Header-Dateien an. Das ist insbesondere dann erforderlich, wenn diese Dateien in einem eigenen Verzeichnis gespeichert werden.

```
gcc -I . -o myprog *.c
```

Sie können aber auch jede `*.c`-Datei für sich zu einer Objektdatei kompilieren und dann alle Objektdateien zusammenfügen:

```
gcc -I . -c func1.c
gcc -I . -c func2.c
gcc -I . -c main.c
gcc -o myprog *.o
```

Um diesen Vorgang so bequem und effizient wie möglich zu gestalten, können Sie `make` zu Hilfe nehmen. Dieses Programm wertet die Datei `makefile` aus. `makefile` enthält wiederum Regeln, wie Ihr Programm kompiliert werden soll. `make` kümmert sich dann darum, alle geänderten Dateien neu zu kompilieren und das ausführbare Programm zusammenzusetzen.

Die `makefile`-Syntax ist ziemlich komplex, und es wurden schon ganze Bücher geschrieben, um ihre Logik begreiflich zu machen. Wir beschränken uns hier auf ein Beispiel:

```
# Datei makefile
CC     = gcc
CFLAGS = -I .
INCL   = func1.h func2.h
OBJ    = func1.o func2.o main.o

myprog: $(OBJ)
        $(CC) $(CFLAGS) -o myprog $(OBJ)

%.o: %.c $(INCL)
        $(CC) $(CFLAGS) -c -o $@ $<

clean:
        rm *.o myprog
```

In den ersten Zeilen werden Konstanten definiert: `CC` gibt an, welcher Compiler eingesetzt werden soll. `CFLAGS` enthält alle Optionen, die an den Compiler übergeben werden sollen. `INCL` zählt alle Header-Dateien des Projekts auf, `OBJ` alle Objektdateien.

Nun folgen die `makefile`-Regeln, die in der Form `ziel: abhängigkeiten` angegeben werden. Der Ausdruck vor dem Doppelpunkt gibt also an, was kompiliert werden soll, während der Ausdruck dahinter festlegt, welche Dateien dabei berücksichtigt werden

müssen. In den weiteren Zeilen folgen dann die Kommandos, die ausgeführt werden müssen, wenn sich abhängige Dateien seit der letzten `make`-Ausführung verändert haben. Die erste Regel besagt also, dass das Programm `myprog` aus den Objektdateien zusammengesetzt werden soll.

Woher aber stammen die Objektdateien? Dafür ist die zweite Regel zuständig. Sie besagt, dass jede Objektdatei `name.o` aus der zugehörigen Datei `name.c` resultiert. Außerdem müssen Objektdateien neu erzeugt werden, wenn sich eine Include-Datei ändert. Die Kompilieranweisung enthält zwei `makefile`-Besonderheiten: `$@` bezeichnet den linken Regelausdruck. Wird der Regelausdruck `%.o` also auf die Datei `func1.o` angewendet, enthält `$@` den Dateinamen `func1.o`. Der Ausdruck `$<` wird wiederum durch den ersten Namen des rechten Regelausdrucks ersetzt, in diesem Fall also durch `func1.c`.

Die erste Regel von `makefile` gilt als Defaultregel, d. h., `make` ohne weitere Parameter verarbeitet die Regel `myprog`. Dabei werden Abhängigkeiten automatisch berücksichtigt, sodass auch die zweite Regel zum Einsatz kommt. Darüber hinaus können Sie Zusatzregeln wie die `clean`-Regel definieren. Um die Anweisungen dieser Regel auszuführen, müssen Sie an `make` den Regelnamen übergeben, also `make clean`.

Jetzt können Sie das `makefile` ausprobieren. `make clean` löscht alle Objektdateien und das Kompilat. `make` kompiliert das Programm nun von Grund auf neu. Wenn Sie anschließend die Datei `func2.c` verändern, erzeugt `make` die neue Objektdatei `func2.o` und linkt das Programm neu zusammen, verzichtet aber auf das neuerliche Kompilieren von `func1.c` und `main.c`.

```
make clean
  rm *.o myprog
make
  gcc -I . -c -o func1.o func1.c
  gcc -I . -c -o func2.o func2.c
  gcc -I . -c -o main.o main.c
  gcc -I . -o myprog func1.o func2.o main.o
touch func2.c
make
  gcc -I . -c -o func2.o func2.c
  gcc -I . -o myprog func1.o func2.o main.o
```

make erfordert Tabulatoren!

Beim Verfassen eines `makefile`s müssen Sie darauf achten, dass Sie die Regelanweisungen durch echte Tabulatoren einrücken, nicht durch Leerzeichen!

22.2 GPIOs steuern

Zur Steuerung der GPIOs durch ein C-Programm bestehen verschiedene Möglichkeiten. Die folgende Aufzählung fasst die drei wichtigsten Varianten kurz zusammen:

- **sysfs-Dateisystem:** Wenn Sie ganz ohne externe Bibliotheken auskommen möchten, können Sie in Ihrem C-Programm Dateien, die den GPIOs zugeordnet sind, im Verzeichnis `/sys/class/gpio` lesen bzw. schreiben. Diese Vorgehensweise ist ziemlich umständlich, weswegen wir Ihnen davon abraten. Alle grundlegenden Informationen sowie Beispielcode finden Sie auf diesen Webseiten:

  https://www.kernel.org/doc/Documentation/gpio/sysfs.txt
  http://elinux.org/RPi_Low-level_peripherals

- **WiringPi-Bibliothek:** Die WiringPi-Bibliothek ist eine speziell für den Raspberry Pi entwickelte C-Bibliothek, die neben allen GPIO-Grundfunktionen inklusive I^2C, SPI und UART auch diverse externe Hardware-Komponenten unterstützt, die oft in Raspberry-Pi-Projekten eingesetzt werden.

- **bcm2835-Bibliothek:** Die bcm2835-Bibliothek hat einen ähnlichen Funktionsumfang wie die WiringPi-Bibliothek. Sie ist für alle Embedded-Systeme geeignet, die wie der Raspberry Pi eine BCM2835-, BCM3836- oder BCM2837-CPU enthalten.

GPIO-Steuerung mit der WiringPi-Bibliothek

In Kapitel 21, »bash-Programmierung«, haben wir Ihnen bereits das Kommando `gpio` des WiringPi-Projekts vorgestellt. Es ermöglicht eine unkomplizierte GPIO-Steuerung im Terminal bzw. in bash-Scripts. Teil des WiringPi-Projekts ist aber auch die gleichnamige Bibliothek zur C-Programmierung, die im Mittelpunkt dieses Abschnitts steht. Dabei setzen wir voraus, dass Sie die Installation der Bibliothek bereits durchgeführt haben, die in Abschnitt 21.4, »WiringPi«, beschrieben ist. Da die WiringPi-Bibliothek der *Lesser General Public License* (LGPL) untersteht, kann sie bedenkenlos auch in kommerziellen Projekten eingesetzt werden.

Bei der Anwendung der WiringPi-Funktionen müssen Sie beachten, dass die WiringPi-Bibliothek eine eigene Pin-Nummerierung verwendet, die weder mit den Pin-Nummern noch mit der Nummerierung in der Dokumentation der BCM2835-Bibliothek übereinstimmt:

http://wiringpi.com/pins

Wenn man von diesem Detail einmal absieht, ist der Einsatz der Bibliothek denkbar unkompliziert. Das im Folgenden abgedruckte Programm schaltet eine mit dem Pin 26 des P1- bzw. J8-Headers verbundene Leuchtdiode zehnmal ein und aus. Die Funktion `wiringPiSetup` ist für die Initialisierung verantwortlich. `pinMode` steuert, wie der

GPIO-Pin verwendet werden soll – in diesem Beispiel als Signalausgang. `digitalWrite` stellt den Ausgang anschließend auf High bzw. Low.

```
// Datei ledonoff.c
#include <wiringPi.h>

// Die Leuchtdiode ist mit Pin 26 des P1/J8-Headers verbunden.
// Die Wiring-Nummer dieses Pins lautet 11.
#define LEDPIN 11

int main(void) {
  if(wiringPiSetup() == -1)
    return 1;

  // GPIO-Pin als Signalausgang verwenden
  pinMode(LEDPIN, OUTPUT);

  // LED zehnmal ein- und ausschalten
  int i;
  for (i=0; i<10; i++) {
    digitalWrite(LEDPIN, 1);
    delay(500);               // 1/2 Sekunden
    digitalWrite(LEDPIN, 0);
    delay(500);
  }
  return 0;
}
```

Beim Kompilieren müssen Sie mit der `gcc`-Option `-l` angeben, dass das Programm mit der WiringPi-Bibliothek verbunden werden soll. Die Bibliothek aus dem Paket `wiringpi` ist in Raspbian Jessie standardmäßig installiert.

```
gcc -I . -o ledonoff ledonoff.c -l wiringPi
```

Das resultierende Programm funktioniert nur, wenn es mit root-Rechten ausgeführt wird:

```
sudo ./ledonoff
```

Die WiringPi-Bibliothek kann auch dazu verwendet werden, um die Bussysteme SPI und I^2C zu steuern. Außerdem gibt es diverse Erweiterungsmodule zur Steuerung von Hardware-Komponenten, die oft in Raspberry-Pi-Projekten zum Einsatz kommen. Die vollständige Dokumentation sowie eine Menge Beispielprogramme finden Sie hier:

http://wiringpi.com

GPIO-Steuerung mit der bcm2835-Bibliothek

Die bcm2835-Bibliothek ist nach der CPU benannt, die im Raspberry Pi 1 eingebaut ist. Die Bibliothek ist aber natürlich auch zu den neueren CPUs der Raspberry-Pi-Versionen 2 und 3 kompatibel. Sie hilft bei der Programmierung vieler Hardware-Funktionen. Die Bibliothek unterliegt der General Public License (GPL) und kann somit in den meisten Fällen kostenlos verwendet werden, also zur privaten Nutzung sowie in Open-Source-Projekten. Kommerzielle Projekte, bei denen das resultierende Programm ohne Codeweitergabe verkauft werden soll, erfordern hingegen eine spezielle Lizenz des Autors der Bibliothek.

http://www.airspayce.com/mikem/bcm2835

Bevor Sie die bcm2835-Bibliothek verwenden können, müssen Sie den Quellcode entsprechend den folgenden Anweisungen herunterladen, kompilieren und die resultierenden Dateien in die Verzeichnisse `/usr/local/lib` und `/usr/local/include` installieren. Vermutlich wird es bereits eine neuere Version geben, wenn Sie dieses Buch lesen. Aktuelle Download-Links finden Sie auf der Webseite des Projekts.

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.49.tar.gz
tar xzf bcm2835-*.tar.gz
cd bcm2835-*
./configure
make
sudo make check
sudo make install
```

An die `bcm2835`-Funktionen werden keine Pin-Nummern übergeben, sondern die GPIO-Nummern gemäß der BCM2835-Nomenklatur. Pin 26 des P1- bzw. J8-Headers entspricht dem GPIO 7, dementsprechend erwarten die bcm2835-Funktionen im ersten Parameter den Wert 7. Wenn Sie – wie viele Raspberry-Pi-Anwender – in Pin-Nummern denken, verwenden Sie am besten die vordefinierten Konstanten `RPI_V2_GPIO_P1_nn`, wobei `nn` die Pin-Nummer ist. `RPI_V2_GPIO_P1_26` enthält somit den Wert 7. Werfen Sie gegebenenfalls auch einen Blick in die Include-Datei `/usr/local/include/bcm2835.h`!

Das folgende C-Programm schaltet die mit dem GPIO-Pin 26 des P1- bzw. J8-Headers verbundene Leuchtdiode zehnmal ein und aus. Der Code sollte auf Anhieb verständlich sein: `bcm2835_init` initialisiert die Bibliothek. `bcm2835_gpio_fsel` steuert, wie ein GPIO-Pin verwendet werden soll – hier als Ausgang. `bcm2835_gpio_write` stellt den Signalausgang auf Low oder High.

```
// Datei ledonoff.c
// die Leuchtdiode ist mit Pin 26 des P1/J8-Headers verbunden
#define LEDPIN RPI_V2_GPIO_P1_26

int main(void) {
  // bcm2835_set_debug(1);
  if (!bcm2835_init())
    return 1;

  // den Pin als Signalausgang verwenden
  bcm2835_gpio_fsel(LEDPIN, BCM2835_GPIO_FSEL_OUTP);

  // zehnmal ein- und ausschalten
  int i;
  for(i=0; i<10; i++) {
    bcm2835_gpio_write(LEDPIN, HIGH);
    delay(500);
    bcm2835_gpio_write(LEDPIN, LOW);
    delay(500);
  }
  return 0;
}
```

Mit dem folgenden Kommando kompilieren Sie das Programm. Die `gcc`-Option `-l` gibt an, welche Bibliothek mit dem Programm verbunden werden soll.

```
gcc -I . -o ledonoff ledonoff.c -l bcm2835
```

Wie beim WiringPi-Beispiel funktioniert auch hier das resultierende Programm nur, wenn es mit root-Rechten ausgeführt wird:

```
sudo ./ledonoff
```

Sie können die bcm2835-Bibliothek auch dazu verwenden, um die PWM-Funktionen des Raspberry Pi sowie die Bussysteme I^2C und SPI zu steuern. Beispielprogramme finden Sie hier:

http://www.airspayce.com/mikem/bcm2835/examples.html

Kapitel 26

Der Raspberry Pi im Vogelhaus

Der Winter war schon fast vorbei, als die Idee aufkam, ein Vogelhäuschen im Garten aufzuhängen. Ein eher kleines sollte es sein, geeignet für Meisen. Ob es vielleicht möglich wäre, mithilfe eines Raspberry Pi und einer Kamera die Vögel beim Brüten zu beobachten, ohne sie zu stören? »Versuch macht kluch!« Dabei ist das Vogelhaus natürlich nur der Aufhänger, um Ihnen die Möglichkeiten des Raspberry-Pi-Kameramoduls näherzubringen: In diesem Kapitel lernen Sie, wie Sie Einzelbilder, Videos und Zeitrafferaufnahmen anfertigen, und realisieren zum Schluss eine rein software-gesteuerte Bewegungserkennung. Diese können Sie nicht nur zur Beobachtung von Tieren, sondern auch als Alarmanlage nutzen.

26.1 Einbau des Raspberry Pi samt Kameramodul in ein Vogelhaus

Als Grundlage für unser Projekt diente ein Vogelhaus aus dem Gartenmarkt (siehe Abbildung 26.1); handwerklich geschickte Menschen greifen sicher lieber selbst zur Säge. Das Vogelhaus, auf das die Wahl fiel, ist ein Mehrfamilienhaus: Es hat drei separate Brutkammern, jede mit einem eigenen Eingang.

Von den drei Kammern soll nur die mittlere für brütende Meisen (auf deren Größe sind die Einfluglöcher bemessen) zur Verfügung stehen. Die beiden äußeren Kammern sollen die Technik aufnehmen (siehe Abbildung 26.2). Zuerst wird der Raspberry Pi installiert und die Kamera ausgerichtet. Die Befestigung erfolgte mit kurzen Stahlnägeln, die durch die unteren Belüftungslöcher des Kunststoffgehäuses geführt wurden.

Vom Raspberry-Pi-Kameramodul gibt es zwei Varianten: eine »normale« und das sogenannte »PiNoIR«-Modul. Wir wählten hier die »PiNoIR«-Variante, weil sie auch bei dämmrigem Licht noch ansehnliche Bilder liefert. Der Name (NoIR = *No Infrared*, außerdem ist *noir* das französische Wort für *schwarz*) deutet darauf hin, dass diesem Kameramodul der sonst übliche Filter für infrarotes Licht fehlt. Das führt am Tag zu verfälschten Farben. Bei Dunkelheit oder bei Beleuchtung mit einem Infrarotscheinwerfer liefert die PiNoIR-Variante aber auch dann noch Bilder, wenn das normale Kameramodul schon lange aufgegeben hat.

Abbildung 26.1 Das Vogelhaus, frisch aus dem Gartenmarkt

Das Modul löst fünf Megapixel auf, liefert also Bilder mit 2592×1944 Pixeln. Kleinere Formate lassen sich per Software einstellen. Im Video-Betrieb liefert das Modul maximal 1080p (Full HD) bei 30 Bildern pro Sekunde. Es misst 25×20×9 Millimeter und wiegt 3 Gramm.

Im Vogelhaus reicht das durch die Einflugöffnung scheinende Mondlicht aus, um der Kamera brauchbare Aufnahmen zu ermöglichen (siehe Abbildung 26.3).

Flachbandkabel zu kurz?

Wenn das Flachbandkabel der Kamera zu kurz ist, können Sie inzwischen im Fachhandel eine längere Ausführung erwerben. Auch verschiedene Gehäuse für das Kameramodul (das in der Regel ohne Gehäuse verkauft wird) sind erhältlich.

Greifen Sie beim Stromanschluss unbedingt auf eine Steckdose zurück, die für den Betrieb im Außenbereich ausgelegt ist. In unser Vogelhaus hat es zwar nie hineingeregnet, aber Kondensfeuchte kann natürlich trotzdem entstehen. Achten Sie auch auf eine Zugentlastung der Zuleitung. Die Stromversorgung für den Raspberry Pi ist durch die Brutkammer geführt (siehe Abbildung 26.2). Bei Meisen funktioniert das, denn sie zerknabbern keine Kabel.

Abbildung 26.2 Eine Kammer für die Stromversorgung, eine für den Raspberry Pi, eine für die Mieter

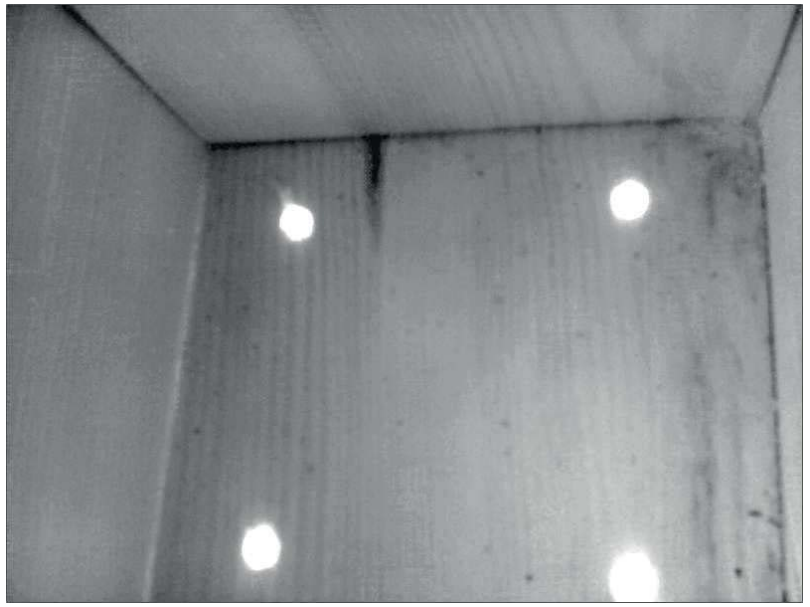


Abbildung 26.3 Die bezugsfertige Kammer, von innen fotografiert durch den Raspberry Pi bei Mondlicht ohne künstliche Beleuchtung

Wenn Sie ein Vogelhaus wählen, das auch größeren Vögeln Platz bietet, sollten Sie das Kabel entweder anders führen oder es durch ein stabiles Rohr schützen. Auch für das Kameragehäuse wählen Sie in diesem Fall eine möglichst robuste Ausführung.

Zum Schluss schadet es nicht, das aufklappbare Dach der Vogelhauses zusätzlich mit etwas Dichtband abzudichten. Die Öffnung, aus der das Stromkabel austritt, kleben Sie mit wetterfestem Band, etwa Gewebeband, ab.

26.2 Kamerapraxis

Wir haben das Kameramodul für den Raspberry Pi ja bereits in Abschnitt 15.8, »Raspberry Pi Camera Board und PiNoIR«, näher vorgestellt. Dieser Abschnitt fasst nochmals zusammen, wie Sie das Modul in Betrieb nehmen und wie Sie Bild- und Filmaufnahmen erstellen. Außerdem erfahren Sie hier, wie Sie aus Einzelbildern einen Zeitrafferfilm machen, und lernen Möglichkeiten zur Bildoptimierung kennen.

Das Kameramodul betriebsbereit machen

Nachdem Sie die Kamera angeschlossen haben, müssen Sie diese software-seitig aktivieren. Am einfachsten gelingt dies mit Einstellungen • Raspberry Pi Configuration (siehe Abbildung 15.20). Sollten Sie Raspbian Lite ohne Grafiksystem verwenden, rufen Sie stattdessen `sudo raspi-config` auf (siehe Abbildung 26.4). Wählen Sie den Menüpunkt `Enable Camera` und danach `Finish`. Sie werden nun aufgefordert, den Raspberry Pi einmal neu zu starten. Danach ist das Kameramodul betriebsbereit.

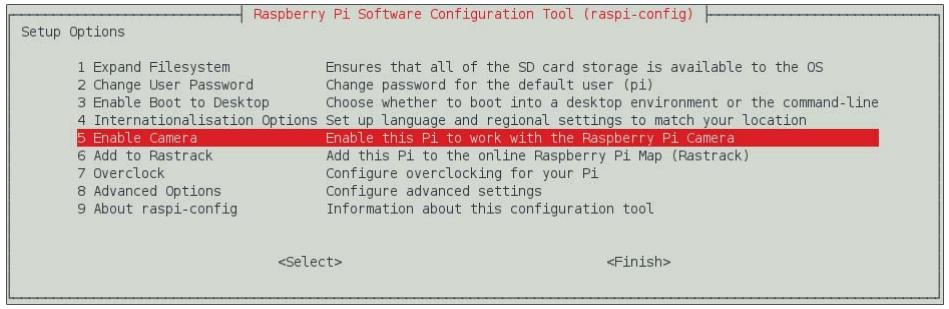


Abbildung 26.4 Aktivierung des Raspberry-Pi-Kameramoduls in raspi-config

Kamera-LED deaktivieren

Immer wenn die Kamera aktiv ist, also ein Bild oder Video aufzeichnet, leuchtet eine rote Leuchtdiode (LED) an der Vorderseite des Moduls. Das ist im Vogelhaus nicht erwünscht, denn auf so kleinem Raum ist die LED sehr hell und würde die Vögel mit Sicherheit verschrecken.

Sie können die Aktivierung der LED zum Glück unterdrücken. Editieren Sie dazu die Datei `/boot/config.txt`, und hängen Sie die folgende Zeile an das Ende der Datei an:

```
disable_camera_led=1
```

Auch diese Einstellung wird erst nach einem Neustart wirksam.

Standbilder mit raspistill aufnehmen

Bereits vorinstalliert sind die Programme `raspistill` und `raspivid`. Das eine ist für Bilder zuständig (engl. *still* = Standbild), das andere für Videos. Mit `raspistill` können Sie nicht nur einfach Bilder anfertigen, sondern diese auch in vielfältiger Weise manipulieren und nachbearbeiten. Es gibt Korrekturmöglichkeiten für die meisten gängigen Bildfehler, und Sie können auf Belichtung, Schärfe, Sättigung und vieles Weitere Einfluss nehmen. Sogar Reihen- und Zeitrafferaufnahmen sind möglich.

Im einfachsten Fall entlocken Sie Ihrer Kamera mit diesem Kommando ein Standbild:

```
raspistill -o bild.jpg
```

Das Ergebnis `bild.jpg` wird im aktuellen Verzeichnis abgelegt. Es wird in voller Auflösung erzeugt (2595 breit, 1944 Pixel hoch) und im JPEG-Format gespeichert. Obwohl das JPEG-Format eine Kompression beinhaltet, ist die Qualitätsstufe standardmäßig so gewählt, dass in der Regel keine Bildstörungen sichtbar sind. Falls Sie direkt auf dem Raspberry Pi arbeiten, also Monitor und Tastatur angeschlossen haben, ist Ihnen sicher aufgefallen, dass das aufgenommene Bild auf dem Monitor eingeblendet wird. Das ist die *Preview*-Funktion (Preview = Vorschau). Sie können diese Funktion mit dem Parameter `-n` abschalten:

```
raspistill -o bild.jpg -n
```

Auch die anderen Bildeigenschaften können Sie durch weitere Parameter beeinflussen. Wollen Sie etwa ein kleineres Bild haben, können Sie Höhe und Breite angeben:

```
raspistill -o bild.jpg -w 640 -h 480
```

Sind Ihnen die Bilddateien zu groß, können Sie die Qualität herunterschrauben und erhalten so kleinere Dateien. Hier wird die JPEG-Qualität auf 60 Prozent reduziert:

```
raspistill -o bild.jpg -q 60
```

Möchten Sie Ihre Bilder nicht im JPEG-Format bekommen, so stehen Ihnen auch noch die Ausgabeformate GIF, PNG und BMP zur Verfügung. Um die Ausgabe im PNG-Format zu wählen, geben Sie Folgendes ein:

```
raspistill -o bild.png -e png
```

Möglichkeiten zur Bildkorrektur

Wenn die Standardwerte des Kameramoduls keine zufriedenstellenden Ergebnisse liefern, gibt es einige Stellschrauben, an denen Sie drehen können. Ist Ihr Bild chronisch unterbelichtet, ohne dass Sie das Problem durch eine Änderung der Beleuchtungssituation beheben können, kommt Ihnen vielleicht der Parameter `-ev` (Exposure Value, Belichtungskorrektur) zur Hilfe. Das Bild wird, wenn Sie hier einen positiven Wert angeben, künstlich aufgehellt. Dieser und weitere Parameter zu Bildkorrektur werden in Abschnitt 15.8, »Raspberry Pi Camera Board und PiNoIR«, erläutert.

ISO-Einstellung

Wenn heute ein Kindergartenkind gebeten wird, Filmdöschen zum Basteln mitzubringen, guckt es in der Regel ein wenig verwirrt aus der Wäsche. Wir Älteren wissen aber noch, dass es Filmrollen in verschiedenen Empfindlichkeitsstufen gibt, die an der ISO-Zahl erkennbar sind und von Anhängern der analogen Fotografie noch heute gern benutzt werden. Je höher der ISO-Wert ist, umso empfindlicher ist der Film, das heißt, umso weniger Licht muss auf ihn fallen, um ein ansehnliches Bild zu produzieren.

Die Sensoren heutiger Kameras können ebenfalls in ihrer Empfindlichkeit eingestellt werden. Höhere ISO-Werte erhöhen auch hier die Empfindlichkeit und ermöglichen Aufnahmen bei schlechten Lichtverhältnissen. Diesen Vorteil erkauft man sich in der Regel mit zunehmendem Bildrauschen.

Auch den ISO-Wert der Raspberry-Pi-Kamera können Sie einstellen. Standardmäßig kommt eine Automatik zum Einsatz, die selbstständig versucht, die richtige ISO-Einstellung zu wählen. Bei schwierigen Lichtverhältnissen kann die Automatik aber versagen und über- oder unterbelichtete Bilder liefern. In diesem Fall stellen Sie den Wert manuell ein:

```
raspistill -o bild.jpg -ISO 800
```

Im Vogelhaus wurde die Kamera auf die höchste Empfindlichkeitsstufe eingestellt. Die Skala reicht bei der Raspberry-Kamera von 100 bis 800 in Hunderterschritten. Den ISO-Parameter können Sie auch für Video-Aufnahmen mit `raspivid` nutzen.

Zeitverzögerung und Zeitrafferfilme

Sie können die Kamera eine Zeitlang warten lassen, bevor das Bild erzeugt wird. Die Länge der Pause geben Sie in Millisekunden an, für fünf Sekunden Verzögerung also 5000:

```
raspistill -o bild.jpg -t 5000
```


Eine Zeitrafferaufnahme erstellen Sie, indem Sie den zusätzlichen Parameter `-tl` (*tl = timelapse*, Zeitraffer) hinzunehmen. Das folgende Kommando erstellt alle fünf Sekunden ein Bild, insgesamt sechzig Sekunden lang. Das `%03d` im Dateinamen führt dazu, dass `raspistill` die Bilder mit einer fortlaufenden dreistelligen Nummer versieht, also `bild-001.jpg`, `bild-002.jpg` und so weiter.

```
raspistill -o bild-%03d.jpg -t 60000 -tl 5000
```

Jetzt haben Sie eine Reihe von Einzelaufnahmen, die Sie zu einem Zeitraffer-Video zusammensetzen können. Das gelingt mit `avconv` (*Audio Video Converter*). Sollte `avconv` auf Ihrem Raspberry Pi noch nicht installiert sein, können Sie das schnell mit `sudo apt-get -fym install libav-tools` nachholen. Das folgende Kommando erstellt aus Ihren Einzelbildern ein Video im MP4-Format mit fünf Bildern pro Sekunde:

```
avconv -r5 -f image2 -i bild-%03d.jpg zeitraffer.mp4
```

Dieser Vorgang ist sehr rechenintensiv und dauert auf dem Raspberry Pi eine ganze Weile. Falls Sie noch einen weiteren, schnelleren Linux-Rechner zur Verfügung haben, ist es eine gute Idee, die Einzelbilder auf diesen zu kopieren und das Zeitraffer-Video dort erstellen zu lassen.

Videos aufzeichnen mit raspivid

Das Aufzeichnen von Videos mit `raspivid` ist genau so einfach wie das Anfertigen von Standbildern, und viele Parameter sind ebenfalls gleich oder ähnlich. Das folgende Kommando nimmt ein Video von 10 Sekunden Länge auf (auch hier wieder in Millisekunden angegeben). Die Größe ist dabei auf 640×480 Pixel reduziert.

```
raspivid -o video.h264 -w 640 -h 480 -t 10000
```

Das Video-Format `H.264`, das standardmäßig verwendet wird, können die meisten Abspielprogramme problemlos verarbeiten. Sollten Sie doch einmal Probleme haben, können Sie das Video mit `avconv` konvertieren, das Sie bei den Zeitrafferaufnahmen schon kennengelernt haben. Das folgende Kommando rechnet Ihr Video in das MP4-Format um:

```
avconv -i video.h264 -vcodec copy video.mp4
```

Hier gilt wie beim Zeitraffer: Es dauert auf dem Raspberry Pi recht lange. Sie können tricksen, indem Sie die Bildwiederholrate reduzieren, etwa auf 15 Bilder pro Sekunde:

```
avconv -i video.h264 -r 15 -vcodec copy video.mp4
```

Das geht natürlich zulasten der Bildqualität. Bei 15 Bildern pro Sekunde nimmt das menschliche Auge schon ein störendes Ruckeln wahr. Besser ist es, Sie nehmen die Konvertierung auf einem anderen, schnelleren Rechner vor.

26.3 Bewegungserkennung mit motion

Zeitrafferaufnahmen und Videos sind gut und schön, aber wenn sich vor der Linse nichts tut, sind sie genauso langweilig wie ein Standbild. Daher wäre es sinnvoll, eine Bewegungserkennung zu haben, die die Kamera nur dann zu einer Aufnahme veranlasst, wenn tatsächlich etwas passiert. Das ist nicht nur für unser Vogelhaus sinnvoll, sondern eignet sich auch gut als Alarmanlage während des Urlaubs oder zur Beobachtung schreckhafter Tiere.

Das Paket `motion` ermöglicht es Ihnen, diese Idee mit dem Raspberry-Pi-Kameramodul und natürlich auch mit anderen Webcams umzusetzen. Mit den folgenden Kommandos installieren Sie `motion` und das Paket `v4l-utils`. Dieses Paket enthält einen v4l-Treiber (*v4l = video for linux*), der das Raspberry-Pi-Kameramodul unter der Bezeichnung `/dev/video0` für `motion` sicht- und nutzbar macht.

```
apt-get update
apt-get upgrade
apt-get -fym install v4l-utils motion
```

Das folgende Kommando lädt das Treibermodul. Nachdem Sie es ausgeführt haben, existiert die Datei `/dev/video0`.

```
modprobe bcm2835-v4l2
```

Dass das Modul korrekt geladen und die Kamera erkannt wurde, sehen Sie auch im System-Logfile. Schauen Sie sich die letzten Zeilen mit `tail -n 20 /var/log/syslog` einmal an:

```
[  864.023270] Linux video capture interface: v2.00
[  864.068272] bcm2835-v4l2: scene mode selected 0, was 0
[  864.074260] bcm2835-v4l2: V4L2 device registered as
               video0 - stills mode > 1280x720
[  864.079525] bcm2835-v4l2: Broadcom 2835 MMAL video capture
               ver 0.0.2 loaded.
```

Das Treibermodul wird noch weiterentwickelt, deshalb können bei Ihnen andere Versionsnummern auftauchen. Wichtig ist `registered as video0`, denn das bedeutet, dass Ihre Kamera startklar ist.

Motion konfigurieren

Das Paket `motion` bringt bei der Installation eine Konfigurationsdatei mit, die `/etc/motion/motion.conf` heißt. Lassen Sie sich nicht von der Größe der Datei abschrecken! Man kann sehr viel einstellen, aber fast alle Werte haben sinnvolle Voreinstellungen und müssen nicht geändert werden. Um mit `motion` loslegen zu können, reichen ganz wenige Modifikationen, die wir nun Schritt für Schritt erläutern. Trotzdem ist es immer eine gute Idee, die unveränderte Konfigurationsdatei unter einem anderen Namen zu sichern, etwa so:


```
sudo cp /etc/motion/motion.conf /etc/motion.motion.conf.sicher
```

Jetzt kann es losgehen. Öffnen Sie die `motion.conf`, und suchen Sie diese Zeilen:

```
# Datei /etc/motion/motion.conf
# Image width (pixels).
# Valid range: Camera dependent, default: 352
width 320

# Image height (pixels).
# Valid range: Camera dependent, default: 288
height 240
```

Hier können Sie die Bildgröße einstellen, die `motion` aufzeichnen wird. 320×240 Pixel ist arg klein, diese Werte können Sie getrost verdoppeln.

Danach stellen Sie die Empfindlichkeit ein, mit der `motion` auf Änderungen im Bild reagiert. Die Bewegungserkennung funktioniert so, dass `motion` nacheinander aufgenommene Bilder miteinander vergleicht und prüft, wie viele Bildpunkte sich von einem zum anderen Bild geändert haben. Ist eine gewisse Schwelle überschritten, startet `motion` die Aufnahme und stoppt sie wieder, wenn das Bild sich beruhigt. Diese Schwelle stellen Sie an folgender Stelle der Konfigurationsdatei ein:

```
# Datei /etc/motion/motion.conf
# Threshold for number of changed pixels in an image that
# triggers motion detection (default: 1500)
threshold 1500
```

Standardmäßig müssen sich also 1500 Pixel zwischen zwei Bildern ändern, damit `motion` dies als Bewegung interpretiert und reagiert. Für Ihre ersten Experimente können Sie diesen Wert niedrig ansetzen. Später finden Sie den richtigen Wert eigentlich nur durch ein wenig Experimentieren heraus, denn er hängt natürlich auch ganz wesentlich davon ab, was Sie beobachten oder überwachen möchten. Eine Beobachtungskamera in einem Vogelhaus benötigt hier natürlich andere Werte, als wenn Sie `motion` nachts als Einbruchsüberwachung in einer Lagerhalle einsetzen.

Nun kommen wir zum Ausgabeformat. Per Default speichert `motion` alle Videos im Shockwave-Flash-Format mit der Dateiendung `.swf`:

```
# Datei /etc/motion/motion.conf
ffmpeg_video_codec swf
```

Dafür benötigen Sie jedoch eine proprietäre Abspielsoftware, was unschön ist. Um stattdessen Videos im MP4-Format zu erhalten, ändern Sie die Zeile so ab:

```
# Datei /etc/motion/motion.conf
ffmpeg_video_codec mpeg4
```

Nicht notwendig, aber eine nette Spielerei ist die `locate`-Funktion. Wenn `motion` irgendwo im Bild eine Bewegung erkannt hat, kann es diesen Bildbereich mit einem rechteckigen Rahmen kennzeichnen. Diese Funktion ist zunächst deaktiviert (`off`). Setzen Sie sie auf `locate on`, wenn Sie diese Funktion nutzen möchten.

```
# Datei /etc/motion/motion.conf
locate off
```

Auf Port 8081 stellt `motion` einen Mini-Webserver zur Verfügung, auf dem Sie das aktuelle Bild der Kamera live verfolgen können. Es gibt aber zwei Haken: Erstens ist `motion` zunächst so konfiguriert, dass sich das Live-Bild nur bei einer erkannten Bewegung aktualisiert. Zweitens ist dieser Webserver nicht von anderen Rechnern im gleichen Netz erreichbar, denn er ist nur an das lokale Loopback-Interface gebunden. Glücklicherweise lässt sich beides leicht ändern. Finden Sie die folgenden Zeilen:

```
# Datei /etc/motion/motion.conf
# rate given by webcam_maxrate when motion is detected
# (default: off)
webcam_motion off
...
# Restrict webcam connections to localhost only
# (default: on)
webcam_localhost on
```

Ändern Sie nun `off` bzw. `on` in das jeweilige Gegenteil:

```
# Datei /etc/motion/motion.conf
webcam_motion on
...
webcam_localhost off
```

Jetzt stellt der Livestream ein Bild pro Sekunde dar, auch wenn nichts passiert (ein Video speichert `motion` natürlich trotzdem nur dann, wenn eine Bewegung erkannt wird). Außerdem ist der Livestream jetzt auch von den Netznachbarn Ihres Raspberry Pi zu bewundern. Das können Sie auch gleich einmal ausprobieren, denn die grundlegende Konfiguration ist damit abgeschlossen.

Sie starten `motion` mit dem gleichnamigen Kommando im Terminal. Es sucht nach der Konfigurationsdatei `/etc/motion/motion.conf` und liest sie ein. Jetzt können Sie auf einem Browser die Adresse Ihres Raspberry Pi, gefolgt von `:8081`, eingeben. Hat der Raspberry Pi also zum Beispiel die IP-Adresse 192.168.2.10, so geben Sie in die Adresszeile des Browsers `192.168.2.10:8081` ein. Jetzt sehen Sie ein Bild pro Sekunde live aus der Raspberry-Pi-Kamera. Wenn eine Bewegung im Bild erkannt wird, umrahmt `motion` den Bereich (siehe Abbildung 26.5) und startet gleichzeitig die Aufnahme.

Abbildung 26.5 motion erkennt eine Bewegung.

Der fehlende Infrarotfilter des PiNoIR-Kameramoduls verfälscht die Farben, wenn man es bei Tageslicht einsetzt. Die Aufnahmen speichert `motion` im Verzeichnis `/tmp/motion`. Es ist sinnvoll, alte Dateien regelmäßig aus diesem Verzeichnis zu löschen, denn je nach Aktivität kann es dort bald recht eng zugehen. Es empfiehlt sich, der in Linux eingebauten Zeitsteuerung `Cron` diese Aufgabe zu überlassen. `Cron` hat eine Art To-do-Liste, die `crontab`. Sie editieren sie mit dem Kommando `sudo crontab -e`. Fügen Sie diese Zeile hinzu:

```
0 0 * * * find /tmp/motion/ -iname "*" -mtime +7 -delete
```

Verlassen Sie nun den Editor. Jetzt werden täglich um Mitternacht alle Dateien aus `/tmp/motion` gelöscht, die älter als 7 Tage sind. Mehr zu Cron finden Sie in Abschnitt 4.10, »Programme regelmäßig ausführen (Cron)«.

26.4 Das Vogelhaus im praktischen Einsatz

Nach so vielen Tipps zur optimalen Verwendung der Kamera sollen Sie zum Abschluss noch erfahren, welche der bisher dargestellten Möglichkeiten wir tatsächlich im Vogelhaus genutzt haben: Die Bewegungserkennung mit `motion`, die ein

durchaus breites Einsatzspektrum hat, haben wir nicht genutzt. Der Grund: Wären tatsächlich Meisen in das Haus eingezogen, so wäre dort permanent Bewegung gewesen, und `motion` hätte praktisch unablässig gefilmt – jedenfalls, solange das Licht ausreicht.

Stattdessen wurde mit `raspistill` alle 60 Sekunden ein Einzelbild in der Auflösung 1024×768 Pixel geschossen. Es bietet sich an, das von `cron` erledigen zu lassen (siehe Abschnitt 4.10). Der folgende `crontab`-Eintrag ist hier nur aus Platzgründen über zwei Zeilen verteilt. Geben Sie das gesamte Kommando ohne \ in einer Zeile an!

```
* * * * * raspistill -o /var/www/html/birdpi.jpg -w 1024 -h 768 \
                     -ex night -ifx denoise -sh 50
```

Der Parameter `-ex night` schaltet die Kamera dabei in eine Art Nachtmodus. Dieser bewirkt hauptsächlich, dass die Kamera hohe ISO-Werte nutzt, die sonst nicht zum Einsatz kämen. Mit `-ifx denoise` wird eine Nachbearbeitung vorgenommen, die das Bildrauschen reduzieren soll, das durch die hohen ISO-Werte entsteht. Dadurch wird das Bild aber recht stark »gebügelt«, und es besteht die Gefahr, dass Details verloren gehen. Deshalb wird zum Schluss noch mit `-sh 50` ein wenig nachgeschärft.

Abgelegt wird das Bild unter `/var/www/html`. Das ist das Standardverzeichnis des Webservers Apache, der ebenfalls auf dem Raspberry Pi installiert ist. Im gleichen Verzeichnis liegt eine sehr einfach gestrickte HTML-Datei, die nichts weiter macht, als dieses Bild anzuzeigen:

```
<html>
  <head>
    <title>BirdPi</title>
  </head>
  <body>
    <img src="./birdpi.jpg" alt="Vogelhaus">
  </body>
</html>
```

Durch Eingabe der IP-Adresse des Vogelhaus-Raspberry-Pi wird nun das Bild aus der Brutkammer angezeigt.

Lichtverhältnisse und Bildqualität

Wir waren uns nicht sicher, wie Meisen auf zusätzliches Infrarotlicht in der Brutkammer reagieren, und haben es daher nicht eingesetzt. Das hat zur Folge, dass für einige Stunden in der Mitte der Nacht ein rein schwarzes Bild entsteht. Allerdings ist die PiNoIR-Variante der Raspberry-Kamera ausreichend lichtstark, um auch bei sehr geringem Umgebungslicht, etwa bei Mondschein oder in der Morgen- und Abenddämmerung, schon erkennbare Bilder zu liefern. Falls doch irgendwann ein Hilfslicht

zum Einsatz kommen wird, werden wir zu einer einzelnen Infrarot-Diode greifen – alles andere wäre für den Einsatz auf so kleinem Raum völlig übertrieben.

Die Bildqualität haben wir getestet, indem wir kleine Gegenstände in das Vogelhaus gelegt haben, etwa ein Spielzeugauto oder eine Tomate. Die entstandenen Bilder waren brauchbar, wenn auch nicht hundertprozentig scharf. Das liegt daran, dass man bei dieser Entfernung an der Naheinstellgrenze der Kamera kratzt. Sie kennen das von Ihren eigenen Augen – was Sie sich direkt vor die Pupille halten, können Ihre Augen nicht scharf abbilden, ein gewisser Mindestabstand muss sein. Trotzdem war das Bild hinreichend gut, um von weiteren Modifikationen abzusehen.

Wo ist nun die brütende Meise?

Gern hätten wir Ihnen an dieser Stelle noch ein Foto von brütenden Meisen gezeigt, aber unser Vogelhaus wurde leider nur temporär bezogen. Im Winter übernachtete dort regelmäßig eine Kohlmeise (siehe Abbildung 26.6), zum Nestbau kam es aber nicht. Wir versuchen es weiter und halten Sie im Blog zu diesem Buch unter der Adresse *https://pi-buch.info* auf dem Laufenden!

Abbildung 26.6 Eine Kohlmeise im Vogelhaus