Kapitel 1
Kauf und Inbetriebnahme

Unter dem Namen »Raspberry Pi« hat die Raspberry Pi Foundation in den letzten knapp 10 Jahren eine bunte Familie von Geräten veröffentlicht. Das Hauptgerät, das im Mittelpunkt dieses Buchs steht, hat aktuell die Bezeichnung *Raspberry Pi 4B*. Wenn wir also in diesem Buch einfach vom Raspberry Pi schreiben, meinen wir dieses Modell bzw. damit kompatible Vorgänger und Nachfolger. Natürlich behandeln wir in diesem Buch auch diverse Varianten, z. B. das Zero-Modell, das *Compute Module*, den Desktop-Rechner *Raspberry Pi 400* sowie den ganz neuen Micro-Controller *Raspberry Pi Pico*.

Das Standardmodell des 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 RAM-Ausstattung etwa zwischen 40 € und 80 €. Zusätzlich brauchen Sie in der Regel ein Netzteil, ein Gehäuse, eine Micro-SD-Speicherkarte und eventuell ein paar Kabel. Bei vielen Händlern gibt es entsprechende Set-Angebote.

Dafür erhalten Sie einen vollwertigen, Linux-basierten Computer mit einer ARM-CPU, den Sie zur Steuerung elektrischer Geräte, für Versuchsaufbauten, als Miniserver oder als kleines Multimedia-Center einsetzen können. Wenn Sie sich nicht an Details stören (es gibt z. B. keinen Ein/Aus-Schalter), kann der Raspberry Pi sogar als Ersatz für einen gewöhnlichen PC verwendet werden. Die Rechenleistung des Raspberry Pi 4 ist vergleichbar mit einem einige Jahre alten Mittelklasse-Notebook.

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

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 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 Elektronikhändlern wie Conrad oder Pollin gibt es auch eine Menge kleinere Webshops, die sich auf Elektronikbastler und die sogenannte Maker-Szene spezialisiert haben.

Beachten Sie beim Einkauf immer den jeweiligen Firmenstandort! Manche Angebote werden aus Großbritannien oder asiatischen Ländern versandt. Das kann nicht nur lange dauern, sondern auch zu zusätzlichen Kosten (Zoll!) führen.

Raspberry-Pi-Modelle

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

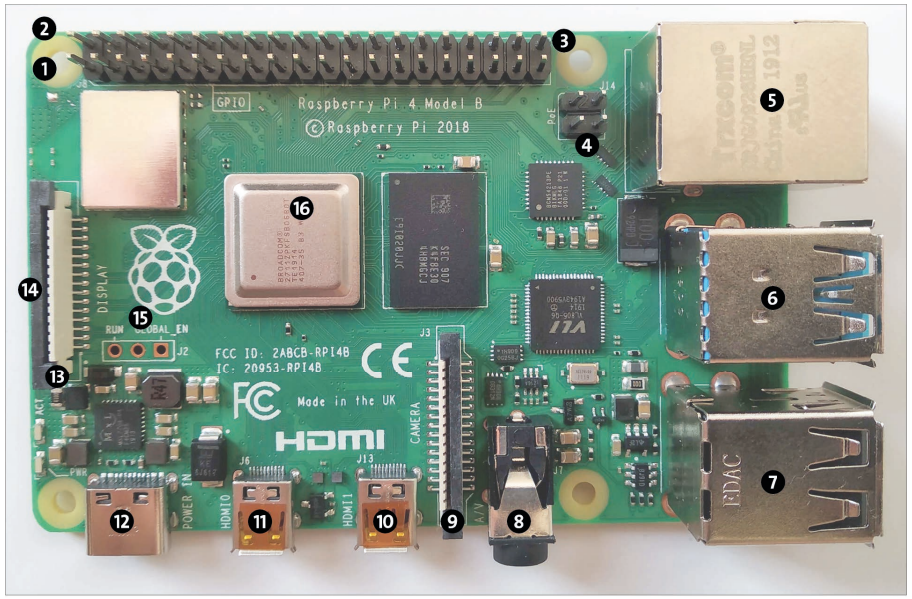
- **Raspberry Pi 4, Modell B:** Dieses seit Sommer 2019 verfügbare Modell ist der zurzeit leistungsfähigste Raspberry Pi (siehe Abbildung 1.1). Eine moderne 64-Bit-CPU mit einer Taktfrequenz von 1,5 GHz machen diesen Raspberry Pi deutlich schneller als das Vorgängermodell.

  Der Rechner verfügt über je zwei USB-2.0- und USB-3.0-Anschlüsse, zwei Micro-HDMI-Ausgänge (4K), einen echten GBit-Ethernet-Adapter, je einen WLAN- und Bluetooth-Adapter sowie über eine 40-Pin-Steckerleiste mit GPIOs (General Purpose Input/Output). Im Vergleich zu den Vorgängermodellen sind die USB- und die Netzwerkschnittstellen vollständig voneinander getrennt und nehmen sich nicht gegenseitig Bandbreite weg. Auch die Übertragungsraten von der/zur SD-Karte haben sich deutlich verbessert.

  Die eigentliche Rechenleistung stellt ein Broadcom-BCM2711-SoC (System-on-a-Chip) zur Verfügung: Es enthält vier CPU-Cores auf Basis der Cortex-A72-Architektur sowie ein Broadcom-Video-Core IV mit H.264- und H.265-Decoder. Aktuell gibt es das Gerät in drei Varianten mit 2, 4 oder 8 GByte zu kaufen. (Fallweise werden auch noch Restbestände des ursprünglichen 1-GByte-Modells abverkauft.)

  Je nach Benchmarktest ist das Modell 4B zwischen 30 und 100 Prozent schneller als sein Vorgänger. Allerdings ist auch die Leistungsaufnahme gestiegen: Ohne

Peripheriegeräte beträgt sie je nach CPU-Auslastung zwischen 4 und 8 Watt. Die Stromversorgung kann wahlweise über ein USB-C-Kabel oder über ein Netzwerkkabel erfolgen (Power-over-Ethernet, erfordert eine ca. 20 € teure Zusatzplatine).



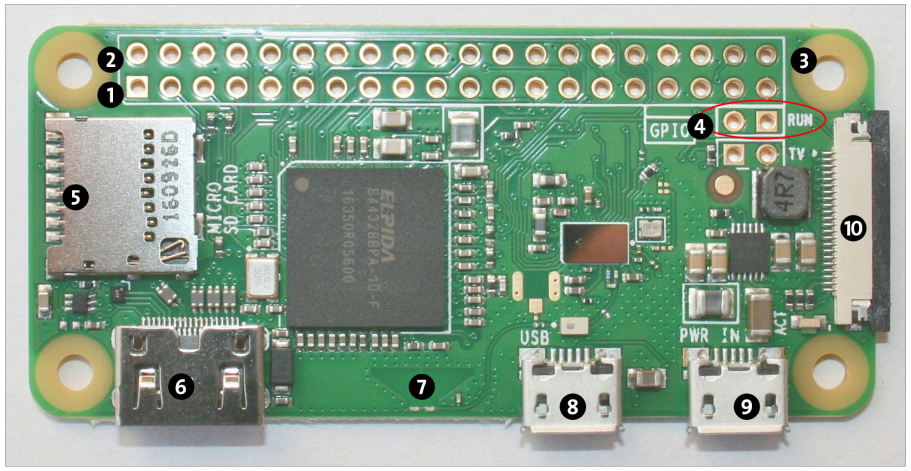
1. GPIO-Header Pin 1
2. GPIO-Header Pin 2
3. GPIO-Header Pin 40
4. POE-Header
5. RJ45-GBit-Ethernet
6. 2x USB 3
7. 2x USB 2
8. Audio
9. CSI-Anschluss für Kamera
10. HDMI-Ausgang 2
11. HDMI-Ausgang 1
12. Stromversorgung USB-C
13. DSI-Anschluss für Display
14. Micro-SD-Kartenslot auf der Platinenunterseite
15. Run-Header
16. SoC BCM 2711

Abbildung 1.1 Der Raspberry Pi 4B

- **Raspberry Pi Zero W und Zero WH:** Die Platine des Zero-Modells ist wesentlich kleiner als die des Standardmodells (siehe Abbildung 1.2): Anstelle einer normalen HDMI-Buchse gibt es ihre Minivariante. Zwei Micro-USB-Buchsen dienen zur Stromversorgung und zur Datenübertragung. Weitere USB-Buchsen wurden ebenso eliminiert wie die Ethernet-Buchse und der analoge Audioausgang. Dank eines Chips mit WLAN- und Bluetooth-Funktionen sind die Zero-Modelle W und WH aber netzwerkfähig.

  Beim Modell W wurde die GPIO-Steckerleiste durch 40 Lötpunkte ersetzt. Die im Februar 2018 vorgestellte Variante Zero WH hat exakt dieselben Daten wie das W-Modell, allerdings ist eine Steckerleiste aufgelötet. Das erleichtert Bastelprojekte (kein Löten erforderlich), macht den Pi Zero WH aber deutlich voluminöser.

  Der Kameraanschluss nutzt den besonders kleinen FPC-Anschluss. Deswegen müssen Sie ein spezielles Kabel verwenden, das teilweise in Raspberry-Zero-Sets mitgeliefert wird.



❶ Pin 1
❷ Pin 2
❸ Pin 40
❹ Run-Header (Reset)
❺ Micro-SD
❻ Mini-HDMI
❼ Integrierte WLAN-Antenne
❽ Micro-USB (Daten)
❾ Micro-USB (Stromversorgung)
❿ FPC-Kamera-anschluss

Abbildung 1.2 Der Raspberry Pi Zero W

Im Vergleich zum Modell 4B verwenden die Zero-Modelle W und WH ein viel älteres SoC: Der BCM2835 bietet nur ein CPU-Core mit ARMv6-Architektur bei einer Taktfrequenz von 1 GHz. Außerdem steht nur ein Arbeitsspeicher von 512 MByte zur Verfügung.

Diesen Nachteilen steht ein großer Vorteil gegenüber: Die Leistungsaufnahme beträgt nicht einmal 1 Watt! Ein weiterer Pluspunkt, wenn auch selten das Entscheidungskriterium, ist der etwas geringere Preis.

- **Raspberry Pi Compute Module 4 (CM4):** Bei dieser Raspberry-Pi-Variante wurde das gesamte Innenleben des Raspberry Pi 4 im Format eines DDR4-SODIMM-Speicherriegels verpackt.

  Das Compute Module enthält standardmäßig einen Flash-Speicher zwischen 8 und 32 GByte und macht mehr Steuerungs-Pins des BCM2711 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 für die industrielle Nutzung gedacht, z. B. wenn der Raspberry Pi zur Steuerung eines in hohen Stückzahlen produzierten Geräts verwendet werden soll.

Abbildung 1.3 gibt einen tabellarischen Überblick über die Raspberry-Pi-Modelle, die zwischen 2012 und 2021 vorgestellt wurden. Weitere technische Details können Sie in der Wikipedia nachlesen:

https://de.wikipedia.org/wiki/Raspberry_Pi#Hardware

	Modell A/A+ 65 mm × 56 mm	**Modell B/B+** 86 mm × 56 mm	**Zero** 65 mm × 30 mm	**Compute Module** 68 mm × 31 mm bzw. 55 mm × 40 mm
Anschlüsse	USB, HDMI	2/4 × USB, HDMI	Micro-USB, Mini-HDMI	
Netzwerk/ Bluetooth	kein Ethernet, Modell 3A+ mit WLAN/ Bluetooth	Ethernet, Version 3: plus WLAN/ Bluetooth	Modelle W und WH: mit WLAN/ Bluetooth	Modell CM4: mit Ethernet
Version 1 **BCM 2835** 1 Core, ARMv6	**April 2012 (A)** **Nov. 2014 (A+)** 700 MHz, 256 MByte	**April 2012 (B)** **Juli 2014 (B+)** 700 MHz, 512 MByte	**Nov. 2015** **Mai 2016** **Feb. 2017** **Feb. 2018** 1 GHz, 512 MByte	**Juni 2014** 700 MHz, 512 MByte, 4 GByte eMMC, LPDDR2-SDRAM
Version 2 **BCM 2836** 4 Cores, ARMv7		**Feb. 2015** 900 MHz, 1 GByte		
Version 3 **BCM 2837** 4 Cores, ARMv8/ Cortex-A53	**Nov. 2018 (A+)** 1,4 GHz, 512 MByte	**Feb. 2016 (3B)** **März 2018 (3B+)** 1,2/1,4 GHz, 1 GByte		**Jan. 2017** 1,2 GHz, 1 GByte
Version 4 **BCM 2711** 4 Cores, ARMv8/ Cortex-A72		**Juni 2019** **Mai 2020** 1,5 GHz, 1/2/4 GByte RAM, je 2 × USB 2/3, 2 × Micro-HDMI (4k), USB-C-Stromversorgung		**Oktober 2020** 1,5 GHz, 1/2/4/8 GByte RAM, 8/16/32 GByte eMMC, GBit-Ethernet, LPDDR4-SDRAM

Abbildung 1.3 Überblick über die bis Mitte 2021 vorgestellten Raspberry-Pi-Modelle, jeweils mit Taktfrequenz und RAM-Größe

Raspberry Pi 400

Seit November 2020 ist der Raspberry Pi 400 auch auf dem Desktop-Markt präsent. Das Innenleben des *Raspberry Pi 400* entspricht dem Modell 4B mit 4 GByte RAM. Das Gerät ist aber in ein kompaktes Komplettsystem samt Tastatur verpackt, mit großen Ähnlichkeiten mit den in den 90er-Jahren so populären Home-Computern. Dementsprechend versucht die Raspberry Pi Foundation, mit dem Gerät Anwender anzusprechen, die auf der Suche nach einem kostengünstigen Komplettsystem sind. Dass das Gerät zum Höhepunkt der Corona-Epidemie ausgeliefert wurde, ist natürlich kein Zufall: Selten war der Bedarf nach Computern für Kids im Heimunterricht so groß wie jetzt.

Was die technischen Daten und Schnittstellen betrifft, entspricht der Raspberry Pi 400 fast dem Modell 4B – aber nicht ganz:

- Das SoC (BMC2711) läuft mit bis zu 1,8 GHz. (Das Modell 4B taktet mit maximal 1,4 GHz.)
- Es gibt nur drei USB-Buchsen. (Der vierte USB-Anschluss wird intern für die Tastatur benötigt.)
- Es gibt keinen eigenen Audioausgang (also die 3,5-mm-Buchse).
- Es gibt keinen Kameraanschluss.

Immerhin sind alle GPIOs zugänglich. Über ein 40-poliges Kabel können die Pins mit einem Breadboard zum Experimentieren verbunden werden.

Abbildung 1.4 Der Raspberry Pi 400

Trotz des konkurrenzlosen Preises – ein Set bestehend aus Computer, SD-Karte, Netzteil, Maus und allen erforderlichen Kabeln kostet nur ca. 100 € – bleibt die schwierige Frage: Für wen ist dieser Computer eigentlich gedacht?

Bastler werden damit nicht glücklich werden; der »normale« Raspberry Pi 4B ist in diesem Fall vorzuziehen. Home-Office- oder Home-Schooling-Anwender werden aber

auch nicht laut »Hurra!« schreien, wenn sie ernsthaft auf diesem Gerät arbeiten sollen – für Videokonferenzen mit Microsoft Teams und Co., aber auch für typische Büroanwendungen ist die Geschwindigkeit zu gering. Am ehesten kann man sich den Computer in einem IT-Labor für Kinder vorstellen: Sei es, um dort erste Schritte auf einem Computer zu machen, sei es, um mit schon etwas älteren Schülern einen Programmierkurs durchzuführen.

Abbildung 1.5 Die Rückseite mit den Schnittstellen des Raspberry Pi 400: Von links nach rechts: GPIO, SD-Karte, 2× HDMI, USB-C, 2× USB 3.0, USB 2.0 (Maus), Ethernet.

Sonderfall Raspberry Pi Pico

Der im Januar 2021 vorgestellte Raspberry Pi Pico ist trotz seines Namens *kein* richtiges Mitglied der bisherigen Raspberry-Pi-Familie. Alle bisherigen Modelle (also A, B, 2B, 3A, 3B, 4B, Zero, Compute Model, 400) sind »richtige« Computer, auf denen als Betriebssystem Linux läuft. Der Raspberry Pi Pico, den wir in diesem Buch in der Regel einfach verkürzt *Pico* nennen, ist dagegen ein Microcontroller. Wir befassen uns in diesem Buch in Teil VI, »Raspberry Pi Pico«, eingehend mit diesem Modell.

Kaufempfehlung

Als Grundlage für die Projekte in diesem Buch empfehlen wir Ihnen den Kauf des aktuellen Raspberry Pi 4B. Dabei haben Sie die Wahl zwischen Modellen mit 1 bis 8 GByte RAM. Für Bastelprojekte und den Multimedia-Einsatz sind 2 GByte absolut ausreichend.

Es gibt freilich keinen Grund, alte Raspberry-Pi-Modelle der zweiten oder dritten Generation wegzuwerfen: Gerade für elektronische Experimente oder einfache Hardware-Projekte reicht ihre Rechenleistung vollkommen aus.

Die preisgünstigen Zero-Modelle sind aufgrund der limitierten Anschlussmöglichkeiten für Raspberry-Pi-Einsteiger nicht optimal geeignet. Ihr Einsatz bietet sich an, wenn Sie Platzprobleme haben oder eine möglichst stromsparende Lösung suchen.

Die Anschlüsse des Raspberry Pi 4B

Der Raspberry Pi 4B bietet eine vielfältige Palette von Anschlussmöglichkeiten (siehe Abbildung 1.6).

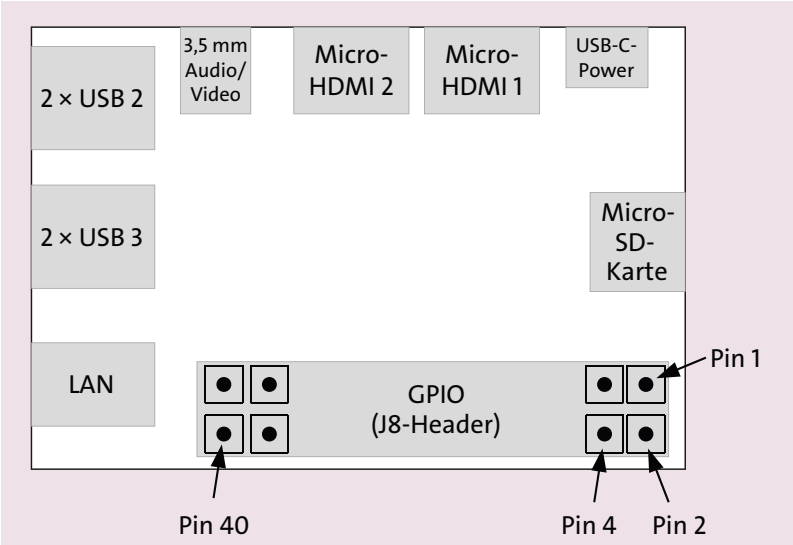


Abbildung 1.6 Schematische Darstellung der wichtigsten Raspberry-Pi-Anschlüsse (gilt für den Raspberry Pi 4B, Sicht von oben)

- Einen USB-C-Anschluss zur Stromversorgung (5 V, 2,5 A bis 3 A, entspricht 12,5 bis 15 W). Der tatsächliche Stromverbrauch ist zumeist deutlich geringer. Er hängt stark von der CPU-Auslastung und dem Leistungsbedarf der USB-Geräte ab.
- Je zwei USB-3.0- und zwei USB-2.0-Anschlüsse (alle im USB-A-Format) für USB-Sticks, Festplatten, SSDs, Tastatur, Maus und andere USB-Geräte. Der Minicomputer kann über alle vier USB-Anschlüsse insgesamt ca. 1.200 mA weitergeben.
- Zwei Micro-HDMI-Ausgänge für Bild und Ton, Auflösung bis zu 3.840 × 2.160 Pixel (also 4K). Ein Monitor kann selbst bei voller Auflösung mit 60 Hz angesteuert werden. Wenn zwei 4K-Monitore zugleich angeschlossen sind, sinkt die Bildfrequenz auf 30 Hz.
- Einen kombinierten Audio-Video-Ausgang für einen vierpoligen 3,5-mm-Klinkenstecker. Wenn das Videosignal nicht genutzt werden soll, kann das Audiosignal auch mit jedem dreipoligen 3,5-mm-Klinkenstecker abgegriffen werden.
- einen Micro-SD-Slot (SDHC)
- einen Ethernet-Anschluss (GBit)

- 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«.
- eine vierpolige Steckerleiste für den Anschluss einer Power-over-Ethernet-Erweiterung (»PoE HAT«)
- Einen integrierten WLAN-Adapter (2,4 GHz und 5,0 GHz, IEEE 802.11ac). Leider gibt es keine Anschlussmöglichkeit für eine externe Antenne.
- einen integrierten Bluetooth-Adapter (Version 5, BLE)

Die Anschlüsse des Raspberry Pi 3B+

Der Raspberry Pi 3B+ bietet ähnliche Anschlussmöglichkeiten wie das Modell Pi 4B. Im Folgenden sind nur die Unterschiede zusammengefasst:

- Alle vier USB-Anschlüsse entsprechen dem USB-2-Standard.
- Es gibt nur einen HDMI-Ausgang mit der Standardbuchse. Die maximale Auflösung beträgt 1.920 × 1.200 Pixel.
- Die Stromversorgung erfolgt über einen Micro-USB-Anschluss (5 V, 2 bis 2,5 A, entspricht 10 bis 12,5 W).
- Der Ethernet-Anschluss unterstützt nur eine Geschwindigkeit von maximal ca. 300 MBit/s.

Netzteil

Das Netzteil ist entscheidend dafür, dass der Raspberry Pi stabil und zuverlässig funktioniert. Die Versuchung ist groß, auf ein ausrangiertes Smartphone-Netzteil aus dem hintersten Winkel der Schublade zurückzugreifen. Das geht aber nicht immer gut: Je nachdem, ob die CPU gerade ausgelastet ist und wie viele Zusatzkomponenten (Maus, Tastatur, Kameramodul usw.) angeschlossen sind, benötigen die Raspberry-Pi-Modelle 3B+ bzw. 4B bis zu 12,5 bzw. 15 W Leistung. Bei einer Spannung von 5 V entspricht das einer Stromstärke von 2.500 bis 3.000 mA. (Ohne externe Geräte und im Leerlauf reichen ca. 5 W bzw. 1.000 mA aus. Aber die Stromversorgung muss so ausgelegt sein, dass der Minicomputer auch unter Last stabil läuft!)

Achten Sie darauf, dass es je nach Raspberry-Pi-Modell unterschiedliche USB-Buchsen zur Stromversorgung gibt: Bis zur 3er-Serie waren Micro-USB-Buchsen üblich. Der Raspberry Pi 4B hat hingegen eine USB-C-Buchse.

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 € pro Jahr.

USB-C-Ärger

Bei der ersten Serie der Raspberry-Pi-4B-Modelle ist die USB-C-Schnittstelle fehlerhaft implementiert. Deswegen funktioniert die Stromversorgung nicht mit jedem Kabel bzw. Netzteil:

https://pi-buch.info/usb-c-aerger-mit-dem-raspberry-pi-4

Das Problem wurde wenige Monate später behoben und betrifft aktuell ausgelieferte Modelle nicht mehr.

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 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 20.4, »Reset/Shutdown-Taste«.

Mehr Komfort bietet diesbezüglich der Raspberry Pi 400: Mit der Tastenkombination Fn + F10 können Sie das Gerät ein- und ausschalten. (Zum Ausschalten müssen Sie beide Tasten vier Sekunden lang drücken.)

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:

- **Micro-SD-Karte:** 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.
- **Standard-SD-Karte:** Nur die allerersten Raspberry-Pi-Modelle (Raspberry Pi 1, Modelle A und B) erwarteten die 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 funktionieren auch derartige SD-Karten wunderbar, sofern Sie nicht das veraltete NOOBS-Installationssystem einsetzen.

Probleme mit SD-Karten

Den Raspberry-Pi-Diskussionsforen zufolge sind defekte SD-Karten die häufigste Fehlerursache auf dem Raspberry Pi. Das hat sich leider auch bei unseren Tests immer wieder bestätigt. Kümmern Sie sich regelmäßig um Backups Ihrer Daten, und halten Sie für den Notfall eine SD-Reservekarte bereit.

Bleibt noch, die optimale Größe der SD-Karte zu klären: Wenn Sie Raspberry Pi OS einsetzen möchten, also das gängigste Linux-System für den Raspberry Pi, dann brauchen Sie eine SD-Karte mit 16 oder 32 GByte. Bei der Lite-Variante von Raspberry Pi OS sowie bei einigen anderen Distributionen, z. B. für den Multimedia-Einsatz, reichen aber auch SD-Karten mit 4 GByte aus.

Gehäuse

Für Versuchsaufbauten auf Ihrem Schreibtisch können Sie auf ein Gehäuse verzichten. Sollten Sie aber vorhaben, Ihren Raspberry Pi im Rahmen eines Projekts dauerhaft 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 aktive Kühlung. Beim Modell 4B ist ein Kühlkörper aber empfehlenswert.

Tastatur und Maus

Nahezu jede handelsübliche USB-Tastatur und -Maus eignet sich als Eingabegerät für den Raspberry Pi. Wir haben 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. Tipps zur Bluetooth-Konfiguration folgen in Abschnitt 2.5, »Bluetooth-Konfiguration«.

Monitor

Als Monitor für den Raspberry Pi eignet sich jeder Monitor oder jedes TV-Gerät mit HDMI-Eingang. Die Modelle 4B und 400 unterstützen 4K-Monitore. Ein normaler HD-Monitor reicht aber vollkommen aus. Achten Sie aber darauf, dass Sie das richtige Kabel besorgen. Aktuelle Raspberry-Pi-Modelle sind aus Platzgründen mit winzigen Micro-HDMI-Buchsen ausgestattet. Normale HDMI-Kabel passen nicht. Abhilfe schafft entweder ein Micro-HDMI-zu-HDMI-Adapter in Kombination mit einem herkömmlichen HDMI-Kabel oder (besser) ein Micro-HDMI-zu-HDMI-Kabel.

Verwenden Sie die richtige HDMI-Buchse!

Beim Raspberry Pi 4B sowie beim Raspberry Pi 400 gibt es *zwei* HDMI-Ausgänge. Solange Sie nur einen Monitor anschließen, müssen Sie die Buchse verwenden, die näher bei der USB-C-Buchse zur Stromversorgung liegt.

USB-Hub

Die Zero-Modelle haben nur einen einzigen USB-Anschluss. Aber nicht nur die Anzahl der USB-Anschlüsse ist limitiert, sondern auch der Strom, den der Raspberry Pi den USB-Geräten liefern kann. Relativ großzügig sind diesbezüglich die Modelle 3B+ und 4B dimensioniert, die in Summe bis zu 6 W Leistung an USB-Geräte weitergeben können. Bei älteren Modellen beträgt das Limit aber 0,5 W pro USB-Buchse. Für viele USB-Geräte ist das zu wenig!

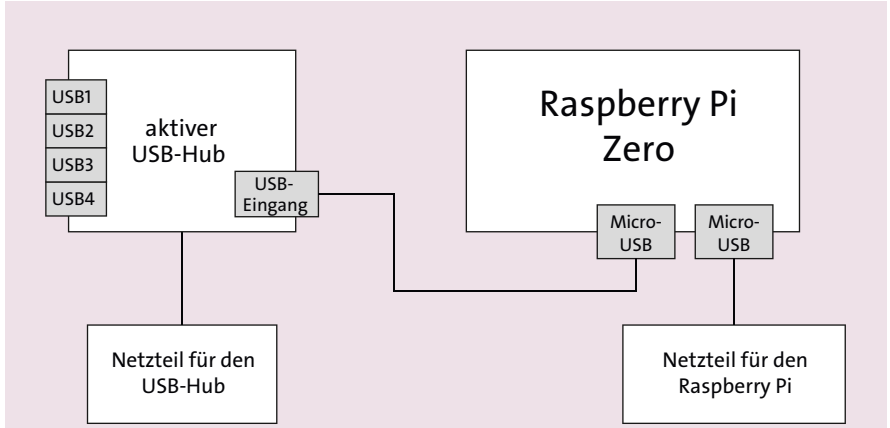


Abbildung 1.7 Raspberry Pi Zero mit aktivem USB-Hub zur gleichzeitigen Verwendung von vier USB-Geräten

Wenn Sie viele energiehungrige USB-Geräte gleichzeitig verwenden möchten, führt an einem aktiven USB-Hub kein Weg vorbei (siehe Abbildung 1.7). *Aktiv* bedeutet in diesem Fall, dass der USB-Hub über eine eigene Stromversorgung verfügt. Die Abbildung macht aber schon klar, dass die Verwendung eines USB-Hubs unweigerlich in einem Kabelsalat endet.

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 Imagedatei 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 Elektronikshop 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 komfortabler als das direkte Arbeiten auf dem Raspberry Pi.

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 und fünften Teil dieses Buchs.

1.2 Raspberry-Pi-Distributionen

Der Raspberry Pi wird ohne Betriebssystem geliefert. 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:

- **Raspberry Pi OS:** Raspberry Pi OS (ehemals Raspbian) ist die populärste Linux-Distribution für den Raspberry Pi. Raspberry Pi OS basiert auf Debian, hat sich von dieser Basis aber im Laufe der Jahre relativ weit weg entwickelt.

  Fast alle Kapitel dieses Buchs beziehen sich auf Raspberry Pi OS. Auch im Internet setzen die meisten Anleitungen und Tipps voraus, dass Sie diese Distribution oder ihren Vorgänger Raspbian verwenden. Diverse Zusatzpakete stehen ausschließlich für Raspberry Pi OS zur Verfügung (z. B. Mathematica) bzw. müssen beim Einsatz anderer Distributionen extra kompiliert werden.

  Neben der Vollversion gibt es auch die Lite-Version, bei der der grafische Desktop sowie alle Desktop-Anwendungen fehlen. Raspberry Pi OS Lite läuft zwar nur im Textmodus, ist dafür aber besonders klein und vor allem für Anwendungen geeignet, bei denen der Raspberry Pi nicht mit einem Bildschirm verbunden wird.

- **Ubuntu:** Recht wechselhaft ist die Entwicklung von Ubuntu für den Raspberry Pi verlaufen. Anfänglich konnte nur Ubuntu Mate auf ausgewählten Raspberry-Pi-Modellen installiert werden. Später wurden auch die Ubuntu-Vollversion sowie Ubuntu Server Pi-kompatibel. Die aktuelle Desktop-Version von Ubuntu unterstützt allerdings nur die Modelle 4B und 400.

  Während sich Ubuntu in der Elektronik- und Bastlerszene nicht etablieren konnte, ist Ubuntu für Desktop-Anwender mit dem Raspberry Pi 400 eine interessante Alternative zu Raspberry Pi OS. Störend ist die spürbar geringere Geschwindigkeit aufgrund des hohen Overheads durch das grafische Benutzeroberfläche Gnome.

- **Windows IoT:** Etwas überraschend ist auch Microsoft auf den Raspberry-Pi-Zug aufgesprungen und bietet die kostenlose Windows-Version »Windows 10 IoT Core« an. Die Modelle 2B und 3B wurden gut unterstützt. Danach ist das Projekt eingeschlafen: Für die Modelle 3B+ und 4B gibt es aktuell (Mitte 2021) keine kompatible Windows-Version.

- **Volumio und Pi Musicbox:** Diese Distributionen machen aus Ihrem Raspberry Pi einen Audioplayer für Ihre Stereoanlage. Beide Distributionen werden über einen Webbrowser bedient, z. B. auf dem Smartphone im WLAN zu Hause. Eine Beschreibung finden Sie in Kapitel 8, »Audioplayer mit Smartphone-Fernbedienung«.

- **LibreELEC, OSMC und RasPlex:** Diese Distributionen sind speziell dazu gedacht, aus Ihrem Raspberry Pi ein Multimedia-Center zu machen. LibreELEC beschreiben wir im Detail in Kapitel 9, »Multimedia-Center mit Kodi und LibreELEC«.

- **Lakka, RecalboxOS und RetroPie:** Diese Distributionen verwandeln Ihren Raspberry Pi in eine Retro-Spielkonsole, auf der Sie diverse alte Videospiele ausführen können. Allerdings brauchen Sie außer einem Monitor und einem USB-Gamecontroller auch ROM-Dateien mit den Spielen. Diese Dateien werden aus Copyright-Gründen nicht mitgeliefert. Geeignete Dateien lassen sich zwar leicht im Internet finden, ihr Download ist aber illegal.

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

https://elinux.org/RPi_Distributions

Wartungsprobleme

Nicht jede der auf der obigen Webseite aufgeführten Distributionen ist so ausgereift wie Raspberry Pi OS. Manche Distributionen laufen nur auf alten Raspberry-Pi-Modellen oder werden nicht mehr gewartet. Wenn Sie sich für eine solche Distribution entscheiden, bekommen Sie weder Sicherheits-Updates noch Bugfixes.

1.3 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 installieren Sie auf Ihrem PC das kostenlose Programm *Raspberry Pi Imager*, laden damit ein Raspberry-Pi-Image herunter und übertragen es dann auf die SD-Karte. Das gelingt mit drei Mausklicks.

Eine Option bestünde darin, dass Sie eine fertige SD-Karte kaufen, auf der Raspbian bereits vorinstalliert ist. Mitunter sind derartige SD-Karten Teil von Komplett-Sets, die neben dem Raspberry Pi auch diverses Zubehör enthalten. Diese Variante ist allerdings vergleichsweise teuer. Zudem müssen Sie früher oder später auf jeden Fall lernen, Ihre SD-Karten selbst zu initialisieren. (Das ist nicht schwierig!)

Wo ist NOOBS?

Das in der Vergangenheit sehr beliebte Installationssystem *NOOBS* (für *New Out Of Box Software*) wurde 2020 eingestellt.

USB-SD-Card-Reader

Viele Notebooks besitzen einen Slot für SD-Karten in Standardgröße. Zum Beschreiben von Micro-SD-Karten müssen Sie daher einen SD-Kartenadapter verwenden, der bei vielen Micro-SD-Karten gleich mitgeliefert wird.

Sollten Sie Ihre SD-Karte auf einem Rechner formatieren oder 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 1.8)

Abbildung 1.8 Eine SD-Karte in Standardgröße, eine Micro-SD-Karte, ein SD-Karten-Adapter sowie ein winziger USB-Adapter für Micro-SD-Karten

Raspberry Pi Imager

Den seit 2020 verfügbaren *Raspberry Pi Imager* laden Sie für Windows, macOS oder Linux kostenlos von der folgenden Website herunter:

https://www.raspberrypi.org/software

Nach der Installation starten Sie das Programm. Im ersten Schritt wählen Sie die zu installierende Raspberry-Pi-Distribution aus – in der Regel einfach Raspberry Pi OS (siehe Abbildung 1.9). Zur Auswahl stehen auch Raspberry Pi OS Lite (ohne Grafiksystem), Raspberry Pi OS Full (mit diversen Zusatzprogrammen, die Sie aber auch später installieren können), Ubuntu, LibreElec und einige weitere Distributionen.

Im zweiten Schritt legen Sie fest, auf welchen Datenträger die Distribution gespeichert werden soll. Passen Sie auf, dass Sie hier wirklich Ihre SD-Karte auswählen und

nicht einen anderen externen Datenträger (z. B. Ihre Backup-Festplatte)! Schreiben startet den Download und parallel dazu den Schreibprozess.

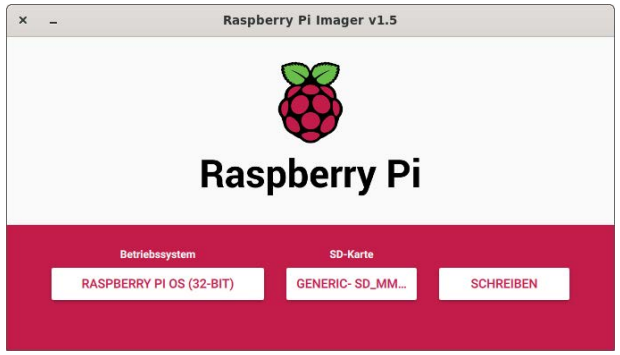


Abbildung 1.9 Der Raspberry Pi Imager

Wenige Minuten später ist Ihre SD-Karte bereit für den Einsatz im Raspberry Pi. Tipps zum ersten Start des Raspberry Pi und zu den dabei fälligen Konfigurationsarbeiten folgen im nächsten Kapitel.

Der Raspberry Pi Imager speichert die heruntergeladenen Images lokal:

```
C:\users\your-username\AppData\Local     (Windows)
Library/Caches oder AppData              (macOS)
.cache/Raspberry Pi/Imager/              (Linux)
```

Wenn Sie später eine zweite SD-Karte beschreiben wollen und es seit dem letzten Mal kein Update Ihrer Distribution gab, erspart Ihnen der Raspberry Pi Imager einen neuerlichen Download.

Zusatzfunktionen

Der Raspberry Pi Imager verfügt über einige verborgene Zusatzfunktionen. So gibt es in der Liste zur Auswahl des Betriebssystem ganz unten den Eintrag Use Custom. Damit können Sie eine selbst heruntergeladene Image-Datei auf die SD-Karte übertragen. Die Datei darf auch komprimiert sein – der Raspberry Pi Imager entpackt das Image während des Schreibprozesses.

Sehr praktisch ist auch die Möglichkeit, bei einer Installation von Raspberry Pi OS vorweg die SSH- und WLAN-Konfiguration durchzuführen. Im Detail beschreiben wir die Vorgehensweise in Abschnitt 4.2.

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).

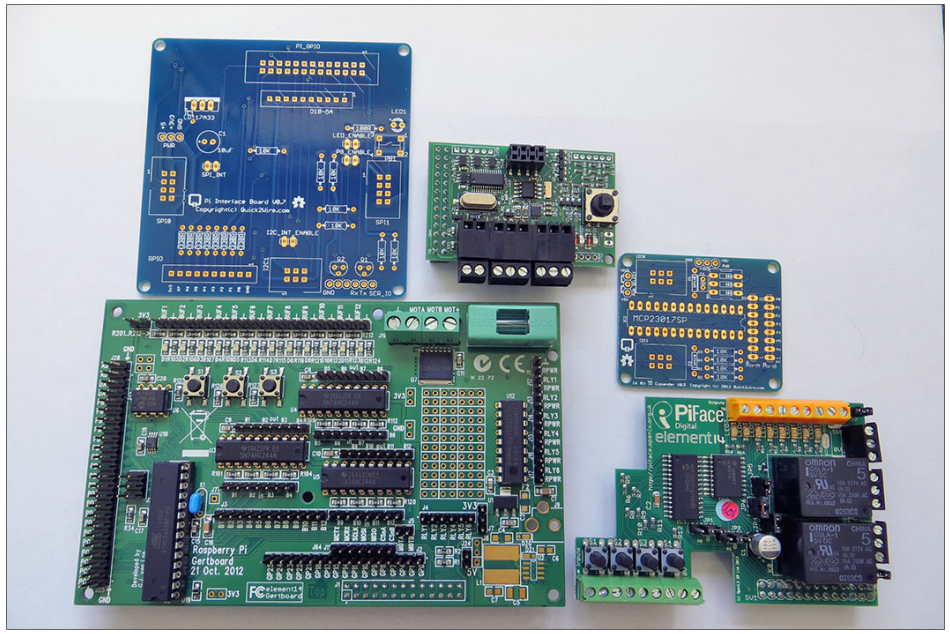


Abbildung 16.1 Eine Auswahl einiger verfügbarer Erweiterungsboards

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 Erweiterungsboards 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.
- Der *Strom Pi* gibt Ihnen die Möglichkeit, eine unterbrechungsfreie Stromversorgung nachzurüsten.
- 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 *PiFace Digital 2* bietet unter anderem fertig bestückte Relais.

Kompatibilität

Manche in diesem Kapitel beschriebenen Boards wurden für den Raspberry Pi 2 oder für noch ältere Modelle konzipiert. Einige Erweiterungen sind direkt mit den Modellen 3B, 3B+ und 4B 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:

https://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 keine Version des Gertboard, die an die neue 40-polige Steckerleiste angepasst ist. Da die ersten 26 Pins der Steckerleiste jedoch denen der alten Modelle entsprechen, können Sie das Gertboard auch weiterhin mit dem Raspberry Pi 4 perfekt zum Experimentieren verwenden.

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
- einem 8-Bit-D/A-Wandler
- einem 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. Stecken Sie den Steckverbinder einfach auf die ersten 26 Pins der 40-poligen Steckerleiste (siehe Abbildung 16.3).

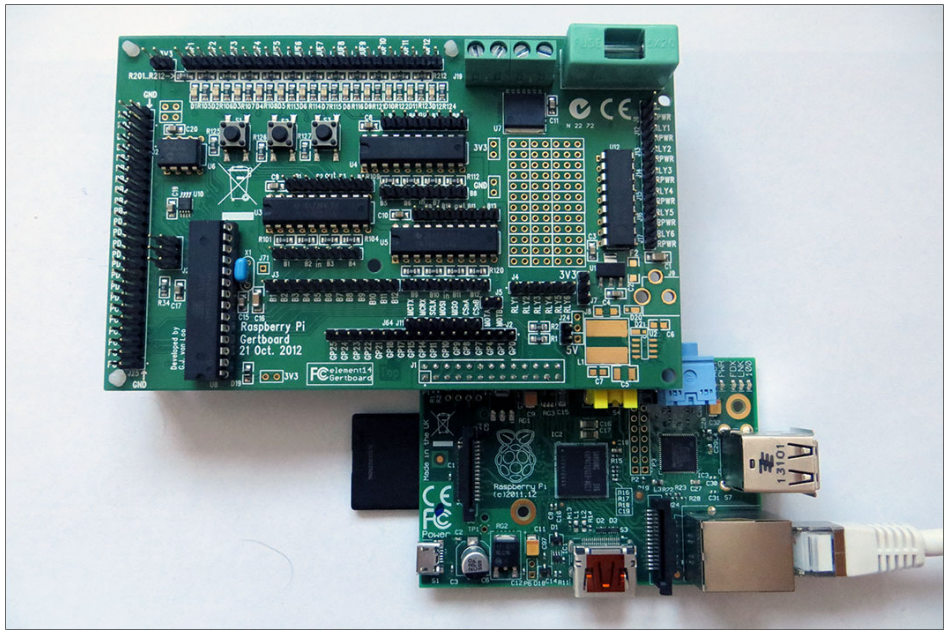


Abbildung 16.2 Auf dem Raspberry Pi montiertes Gertboard

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.

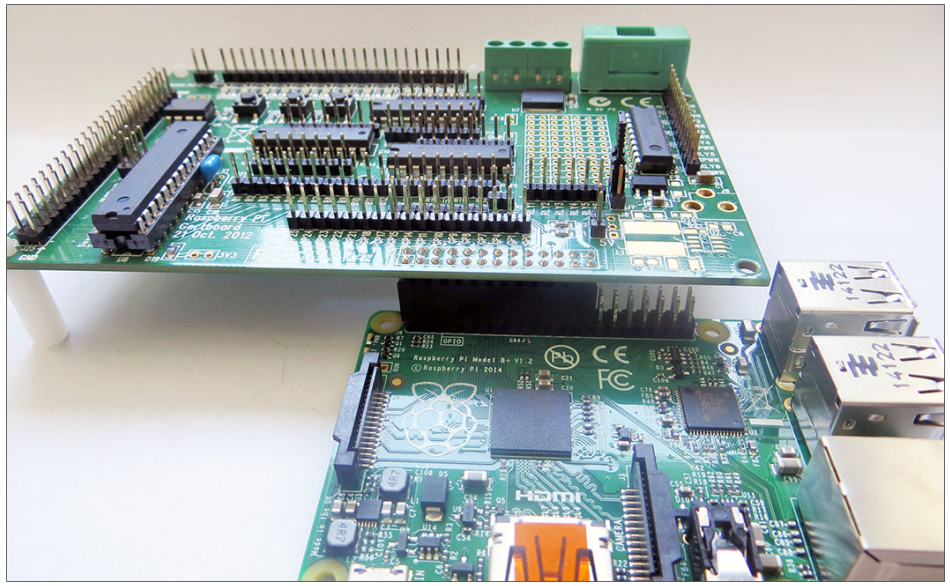


Abbildung 16.3 Das Gertboard kann problemlos auf den Raspberry Pi 2, 3 oder 4 gesteckt werden.

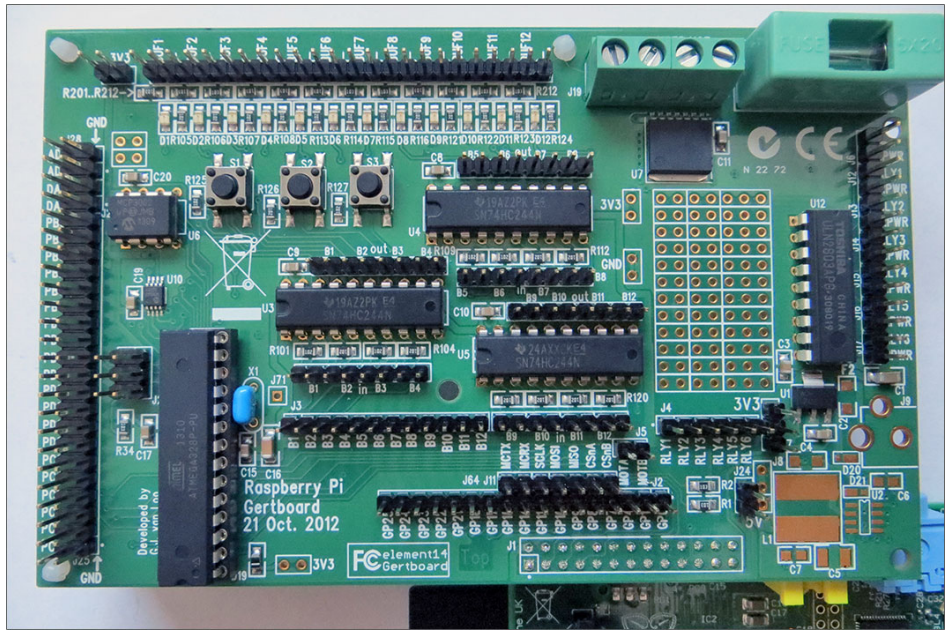


Abbildung 16.4 Das Gertboard in der Detailansicht

Auf den folgenden Seiten zeigen wir Ihnen, wie Sie die einzelnen Blöcke in Betrieb nehmen und deren Funktion auf dem Raspberry Pi testen. Ein umfangreiches Hand-

buch sowie die Belegung und das Layout des Gertboards finden Sie in der offiziellen PDF-Datei:

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

Kaufen können Sie das Gertboard u. a. bei Pollin:

https://www.pollin.de/p/raspberry-pi-zusatzplatine-gertboard-701813

Die drei Taster

Das Gertboard enthält drei Druckknöpfe, die bei Betätigung den entsprechenden GPIO-Pin über einen 1-kΩ-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 Pins auf den Gertboard sind aber in der BCM-Variante beschriftet. In diesem Beispiel möchten wir die drei Taster den physischen Pins 11 (GPIO 17), 13 (GPIO 27) und 15 (GPIO 22) zuweisen. 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 weiterhin mit GP21 beschriftet!

Vorsicht, irreführende Pin-Bezeichnungen!

Auch die Beschriftung der Pins GP0 und GP1 bezieht sich auf die kaum mehr gebräuchliche Revision 1 des Modells B. Im Modell 4 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 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* enthält. Stecken Sie je einen Jumper über die Pins B1 bis B3 (siehe Abbildung 16.5).

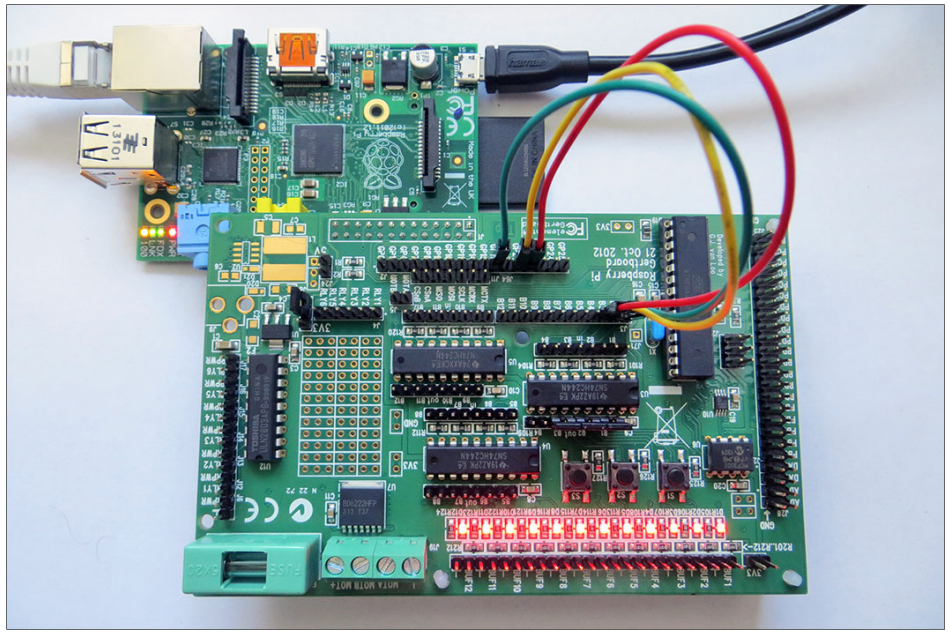


Abbildung 16.5 Verdrahtung des Gertboards zur Inbetriebnahme der Taster

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 Sie die Taster des Gertboards in Python verwenden 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 legen Sie mittels Jumpern auf dem Gertboard die Ports als Ein- oder Ausgang fest. 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 GPIO 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 Rasp-

berry 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 den Pin 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).

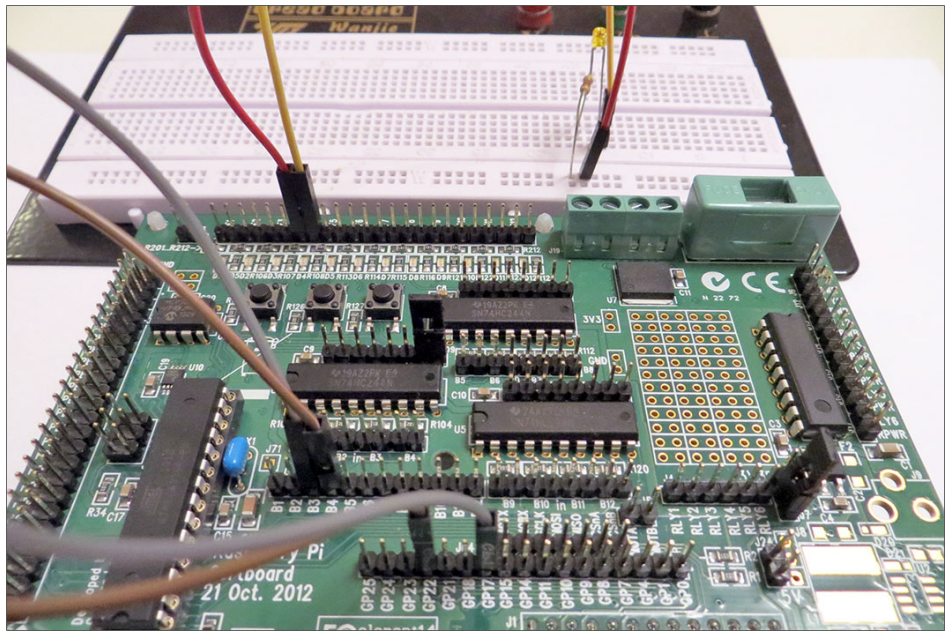


Abbildung 16.6 Die gesamte Verdrahtung zum Testen der digitalen Ein-/Ausgänge

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 den 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 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«!

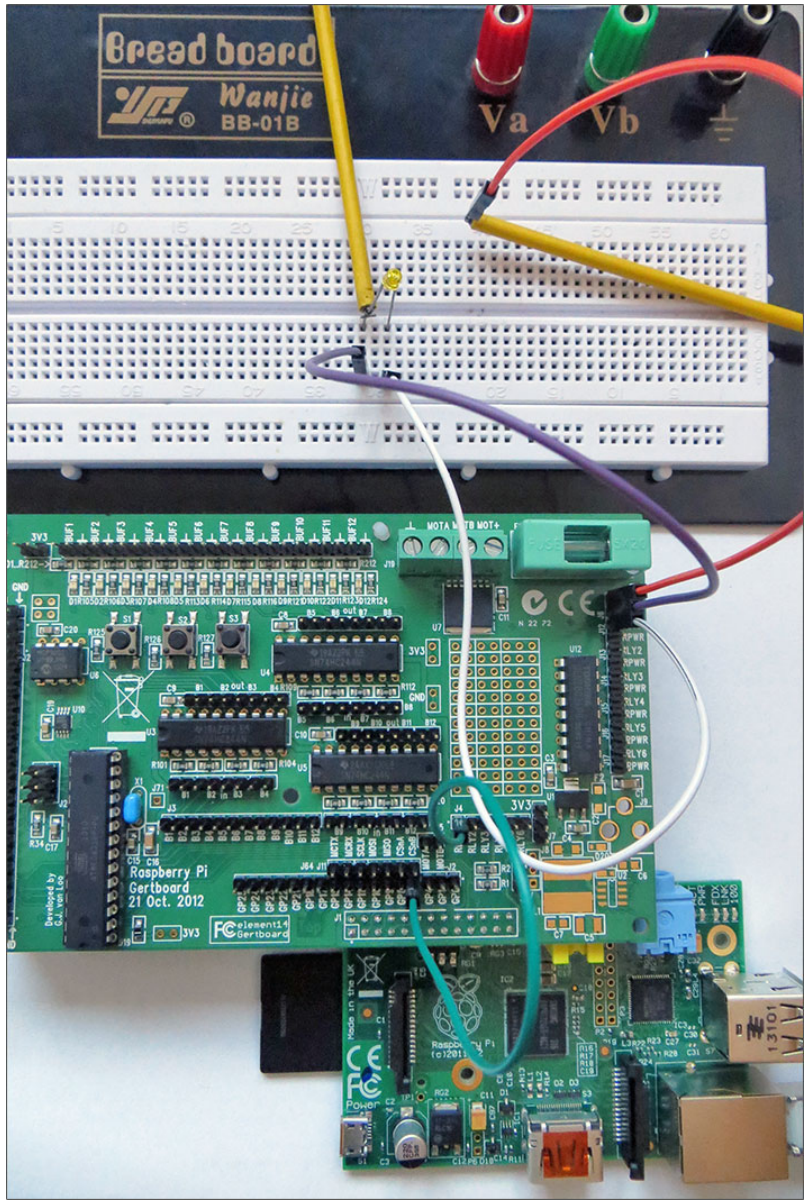


Abbildung 16.7 Eine LED wird über ein externes Netzteil am ULN2803A des Gertboards betrieben.

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

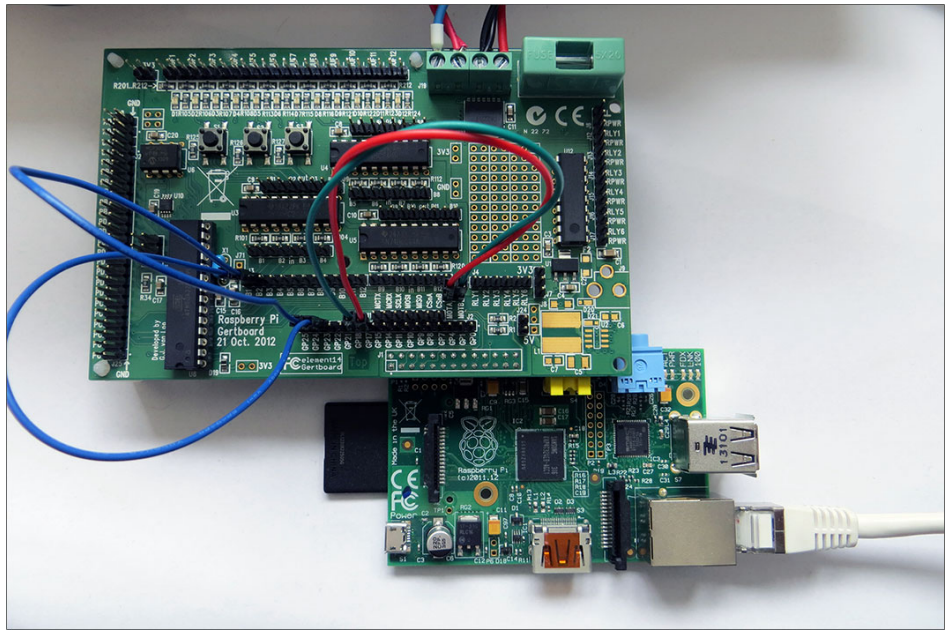


Abbildung 16.8 Verkabelung des Gertboards zur Nutzung des Motortreibers und der Taster

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:

https://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 sie 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, werden zwei SPI-Kanäle genutzt. 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)
```

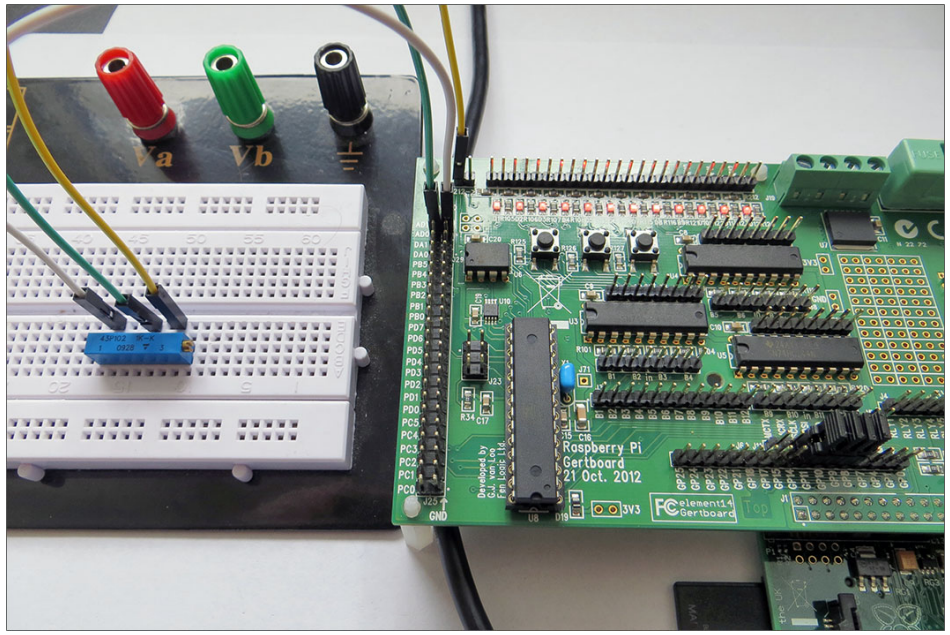


Abbildung 16.9 Jumper-Positionen und Poti zur Nutzung des A/D-Wandlers

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, das Sie der Funktion `xfer` folgende Parameter mitgeben 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.

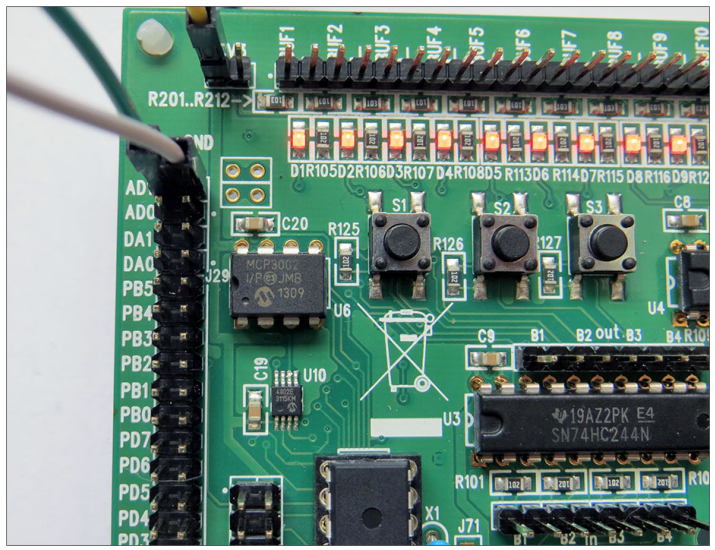


Abbildung 16.10 Auf unserem Gertboard ist ein MCP4802 verbaut. Das ist an der Beschriftung des Bausteins im Gebiet U10 zu erkennen.

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://www.mouser.com/ds/2/268/22249A-14224.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 / 1.024 = 2 mV
	2 ×	4,096 V / 1.024 = 4 mV
MCP4822 (n=12)	1 ×	2,048 V / 4.096 = 0,5 mV
	2 ×	4,096 V / 4.096 = 1 mV

Tabelle 16.1 Schrittweiten der MCP48X2-Familie. Der MCP4802 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 Microcontrollers. Das Gertboard enthält dazu einen vormontierten ATmega168- oder ATmega328-Microcontroller. Dadurch können Sie Arduino-Programme auf den Microcontroller 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 16.11).

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

http://ww1.microchip.com/downloads/en/DeviceDoc/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-Microcontroller. 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
- Port-Erweiterung
- allgemeine Logikverarbeitung

Für unser Beispielprojekt beginnen wir mit der Verkabelung. Diese ist nur einmalig notwendig, um das Microcontroller-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).

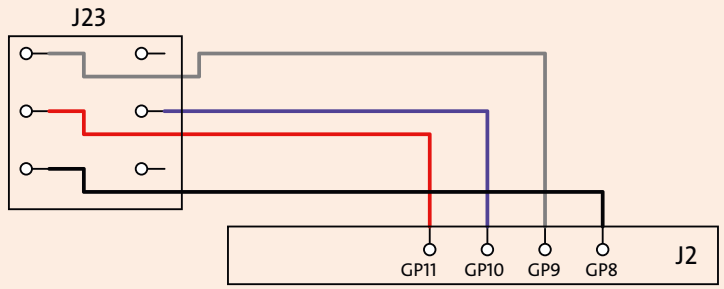


Abbildung 16.11 Schematische Darstellung der erforderlichen Gertboard-Verbindungen zur Programmierung des ATmega

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 Microcontroller

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!

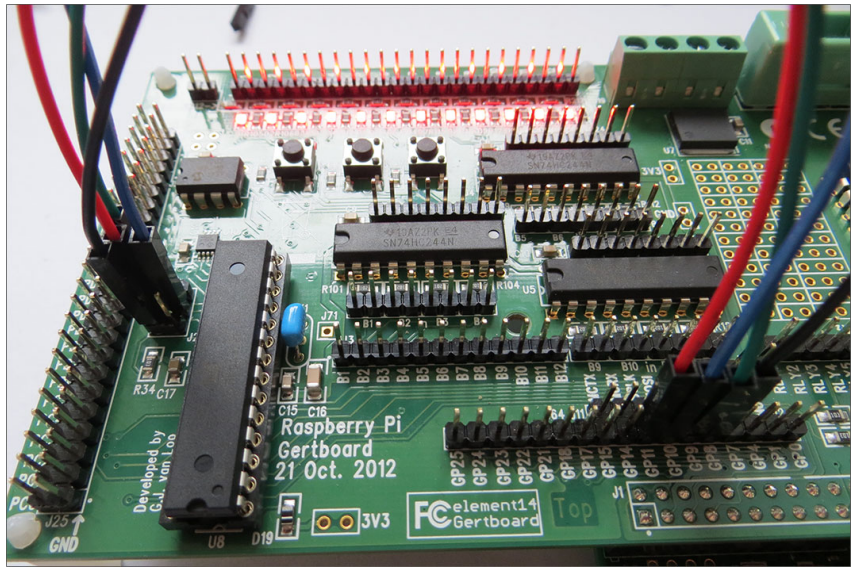


Abbildung 16.12 Die vier nötigen Leitungen von der SPI-Schnittstelle des Raspberry Pi zum ATmega. Der Microcontroller ist ganz links im Bild zu sehen.

avrdude

Zur Übertragung von Microcontroller-Code vom Raspberry Pi auf den Microcontroller des Gertboards ist das Programm `arvdude` notwendig. Es wurde von Gordon Henderson entwickelt, dem die Raspberry-Pi-Community auch die (mittlerweile veraltete) 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 nehmen Sie einige Einstellungen am System vor und passen ein paar Gertboard-spezifische Parameter im Arduino-IDE an. 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 den ATmega328P oder [2] für den ATmega168. Dem Microcontroller 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 Microcontrollers geglückt, und Sie können mit der Programmierung beginnen.

Die Arduino-IDE

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

```
sudo apt 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 ihn 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 Microcontroller 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. Wenn das nicht der Fall ist, prüfen Sie, ob Sie JS29-PB5 mit der LED verbunden haben (BUF1).

Werfen Sie einen Blick in das englische Handbuch des Gertboards! Darin 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 Microcontroller-Programmierung auf dem Gertboard:

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

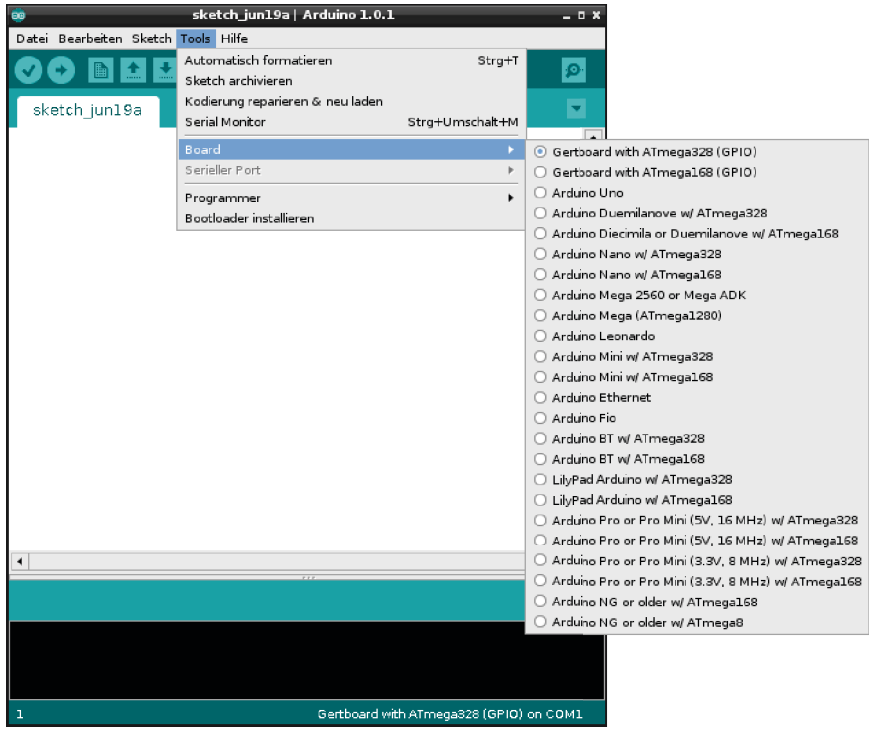


Abbildung 16.13 Auswahl des verwendeten Microcontrollers

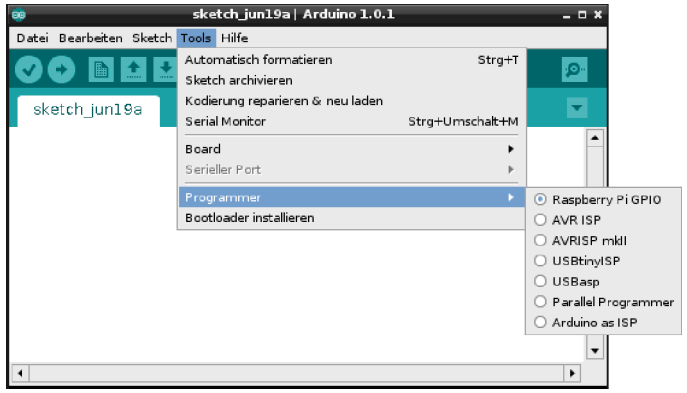


Abbildung 16.14 Wahl der Programmierschnittstelle

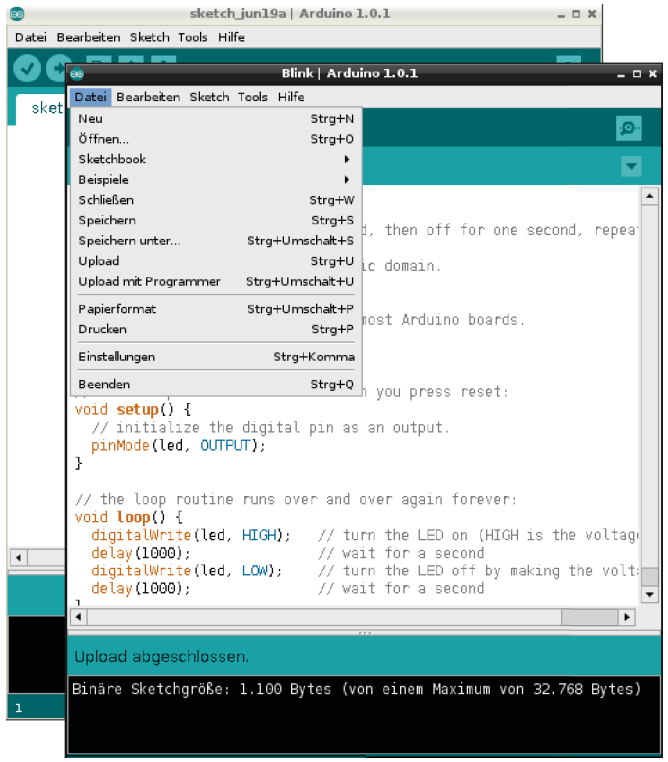


Abbildung 16.15 Upload des Programms auf den ATmega

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. (TTL steht für Transistor-Transistor-Logik.) 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. Er 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).

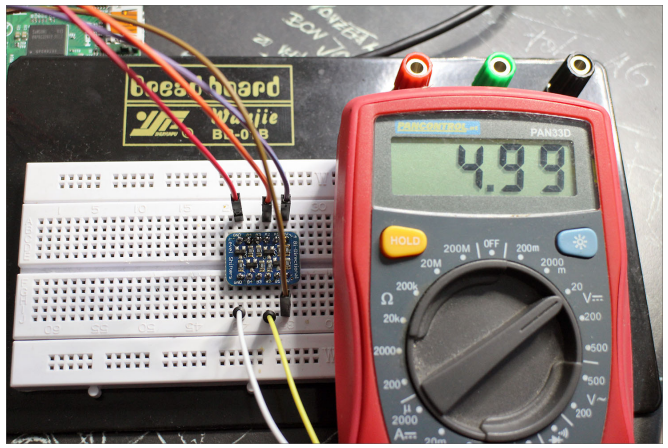


Abbildung 16.16 Ein Logic-Level-Converter wandelt die 3,3 V eines GPIO-Pins in 5 V um.

16.4 PiFace Digital 2

Das *PiFace Digital 2* ist ein kleines Erweiterungsboard für digitale Ein- und Ausgänge (siehe Abbildung 16.17). 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 an das Layout des aktuellen Modells 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 es auch noch auf dem Raspberry Pi 3B/3B+/4B 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.

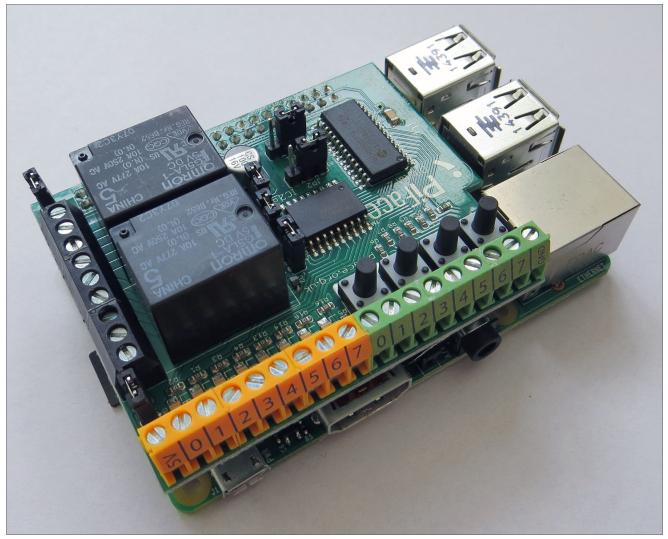


Abbildung 16.17 Das PiFace Digital 2 auf dem Raspberry Pi

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. Sie enthält Python-Bibliothek, die den Umgang mit den Ein- und Ausgängen sehr komfortabel gestaltet:


```
sudo pip3 install pifacecommon
sudo pip3 install pifacedigitalio
```

Die PiFace-Bibliothek

Mit den zuvor installierten Paketen wurde eine bedienerfreundliche Python-Library installiert. Da im PiFace Digital 2 mit dem MCP23S17 ein Port-Expander 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 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 entsprechenden acht Eingänge. Die Ausgabe der Funktion ist bei anliegendem Signal `1`, bei keinem Signal `0`. Dies können Sie auch einfach kontrollieren:

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

Die vier Onboard-Taster liegen an den Eingängen 0 bis 3 parallel. Damit können auch 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. Die Adresse jedes einzelnen Boards kann über die Jumper *JP1* und *JP2* in der Mitte des PiFace eingestellt werden (siehe Tabelle 16.2).

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

Tabelle 16.2 Adressierung mehrerer PiFace-Boards durch Jumper

Auch in Python können Sie mit der Bibliothek gezielt jedes Board ansprechen:

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

Weitere Infos zum PiFace Rack finden Sie auf der Herstellerseite:

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

Auf dem PiFace Digital 2 befinden sich 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.

16.5 StromPi 2 – USV und Wide-Range-Spannungsversorgung

Die Erweiterungsplatine StromPi 2 von Joy-IT behebt zwei Schwachstellen des Raspberry Pi: Zum einen erlaubt sie den Anschluss von Spannungsquellen von bis zu 61 V, zum anderen dient der StromPi 2 als unterbrechungsfreie Stromversorgung (USV). Der StromPi 2 ist die Platine der Wahl, wenn Sie Ihren Raspberry Pi an einer Autobatterie (12 V) oder im LKW (24 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. Das Board ist zu allen aktuellen Raspberry-Pi-Modellen (inklusive 4B) kompatibel.

Installation und Modusauswahl

Um den StromPi 2 mit dem Raspberry Pi zu verbinden, stecken Sie die Platine auf die GPIO-Leiste J8 (siehe Abbildung 16.18). Sie können nun durch die Position der beiden Modus-Jumper-Brücken zwischen dem Wide-Range- und dem USV-Modus wählen:

- Im Wide-Range-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. Der StromPi 2 besitzt einen Spannungsregler, der die Wide-Range-Spannung zuverlässig auf 5 V herunterregelt.

  Sie können ebenfalls ein Micro-USB-Kabel in die mit *In* beschriftete Buchse des StromPi 2 stecken. In diesem Fall wird die Spannungsquelle am Micro-USB-Anschluss bevorzugt. Fällt sie 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 16.19). 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.

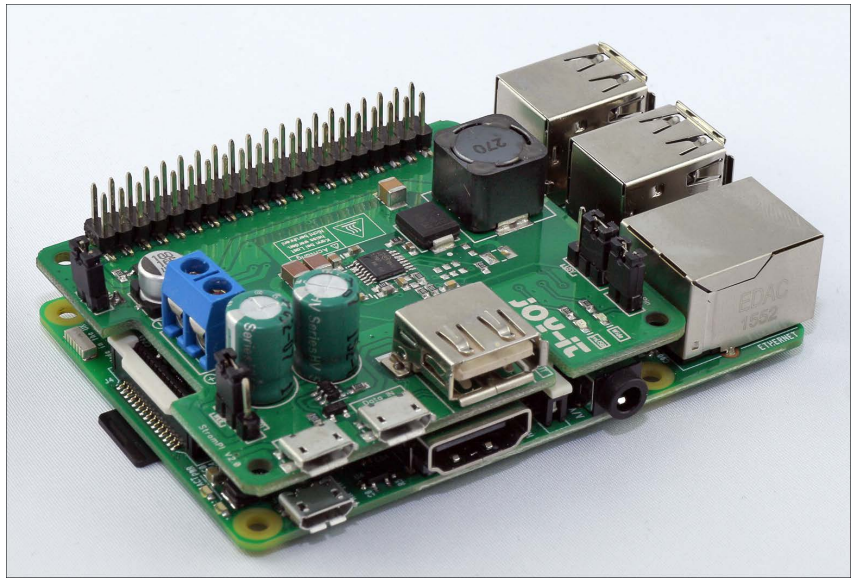
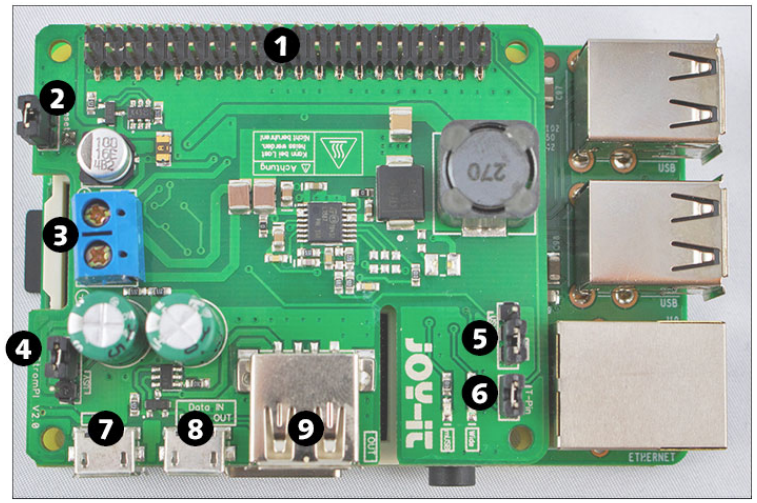


Abbildung 16.18 Der StromPi 2 passt perfekt auf das B-Modell des Raspberry Pi.



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 16.19 Der 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* des StromPi 2 mit einer der USB-Buchsen des Raspberry Pi verbinden (siehe Abbildung 16.20). Die Voraussetzung hierfür ist natürlich, dass Ihre gewählte Primärspannungsquelle in der Lage ist, den benötigten Strom zu liefern.

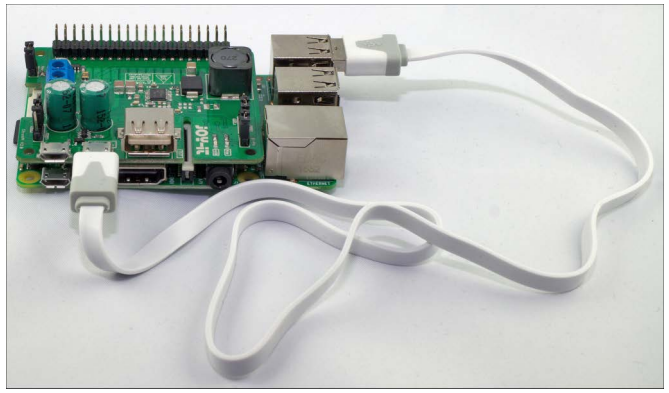


Abbildung 16.20 Durch diese Verbindung liefern auch die USB-Ports des Raspberry Pi bis zu 3 A Strom.

Der StromPi 2 kann den Raspberry Pi wieder neu starten, wenn eine unterbrochene Micro-USB-Spannung wiederhergestellt wurde. Dazu muss der Reset-Jumper auf dem StromPi 2 installiert sein (siehe Abbildung 16.19). Eine denkbare Anwendung ist das schonende Herunterfahren nach Ausfall der Hauptspannung und ein erneutes Hochfahren des Raspberry Pi, sobald die Spannung wiederhergestellt wurde. Dieses Szenario können Sie mit der Software realisieren, die wir im folgenden Abschnitt beschreiben.

Die Software

Über die Website des Herstellers Joy-IT können Sie eine Software beziehen, die weitere Überwachungseinstellungen zulässt:

https://strompi.joy-it.net/downloads

Nach dem Download der Software erhalten Sie zwei kleine Python-Programme: `power-alarm.py` und `powershutdown.py`. Ersteres sendet eine E-Mail, sobald der 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 des StromPi 2 und GPIO-Pin 21 des Raspberry Pi her. Wenn Sie diese Funktion nicht benötigen, GPIO-Pin 21 anderweitig nutzen möchten oder einen eigenen Alarm-Pin verwenden wollen, so entfernen Sie den Jumper und nutzen gegebenenfalls einen 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`.

Bezugsquellen für den StromPi 2 finden Sie hier:

https://joy-it.net/de/supply

Strom Pi 3

Es gibt mittlerweile auch das Nachfolgemodell StromPi 3 über den gleichen Onlineshop zu kaufen. Neue Features des StromPi 3 sind:

- eine aufsteckbare Batterieeinheit als wiederaufladbare Notstromquelle
- Möglichkeit zur Programmierung von Szenarien (z. B. Zeitpläne zum Hoch- und Herunterfahren)
- serielle Schnittstelle zum Auslesen von Zustandsdaten (Ladezustand, Status der Ausgänge etc.)
- Real-Time-Clock

16.6 Pimoroni Zero LiPo

Das kleine Board *Pimoroni Zero LiPo* ist ein winzig kleiner Spannungswandler, der die 3,7 V eines LiPo-Akkus in 5 V konvertiert. Das kleine Board wird einfach auf die ersten acht Pins der GPIO-Stiftleiste gesteckt. Der Raspberry Pi startet, sobald Sie den LiPo-Akku in den passenden Steckverbinder gedrückt haben.

Ein schwacher Akku (ab 3,4 V) wird mit einem Low-Signal an Pin 7 (GPIO 4) angezeigt. Mit dieser Information können Sie ein kleines Programm schreiben, das den Raspberry Pi gegebenenfalls sicher herunterfährt.

Pimoroni Zero LiPo kann bis zu 1,5 A Strom liefern und sollte für die meisten Anwendungsfälle ausreichen.

Als Bezugsquelle haben wir nur Adafruit oder Pimoroni ausfindig machen können. Beide Anbieter sitzen nicht in Deutschland:

https://shop.pimoroni.com/products/zero-lipo
https://adafruit.com/product/3196

Leider muss der LiPo-Akku zum Laden immer wieder abgestöpselt und über ein LiPo-taugliches Ladegerät geladen werden. Eine Alternative ist der *PowerBoost 1000 Charger*. Dieses kleine Board können Sie zwar nicht auf den Raspberry Pi stecken, jedoch kann es LiPos laden und gleichzeitig den Raspberry Pi versorgen. Weitere Details können Sie hier nachlesen:

https://www.adafruit.com/product/2465

Eine mögliche Bezugsquelle ist EXP-Tech:

https://exp-tech.de/adafruit-powerboost-1000-charger

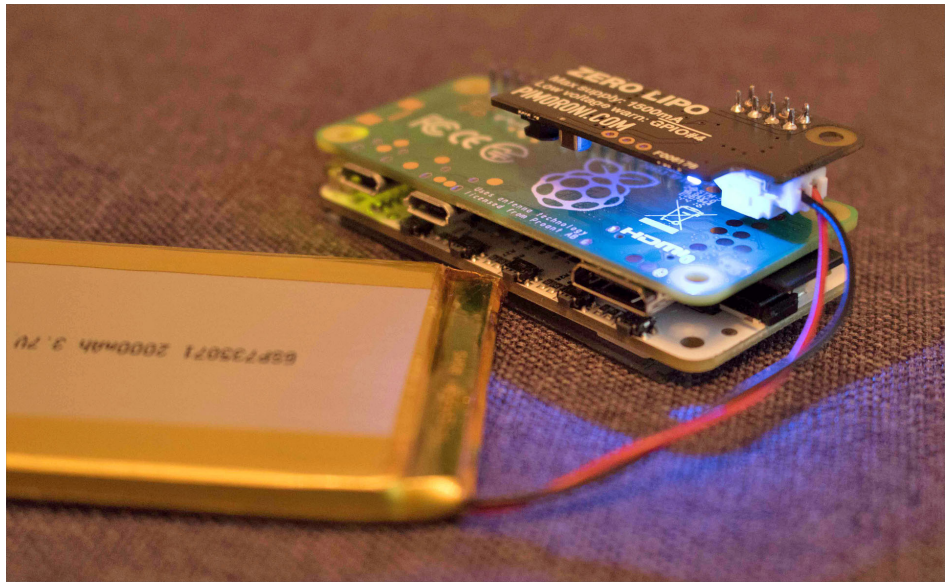


Abbildung 16.21 Der Pimoroni Zero LiPo auf dem Raspberry Pi Zero (unten PaPiRus-Display)

16.7 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 16.22). 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.

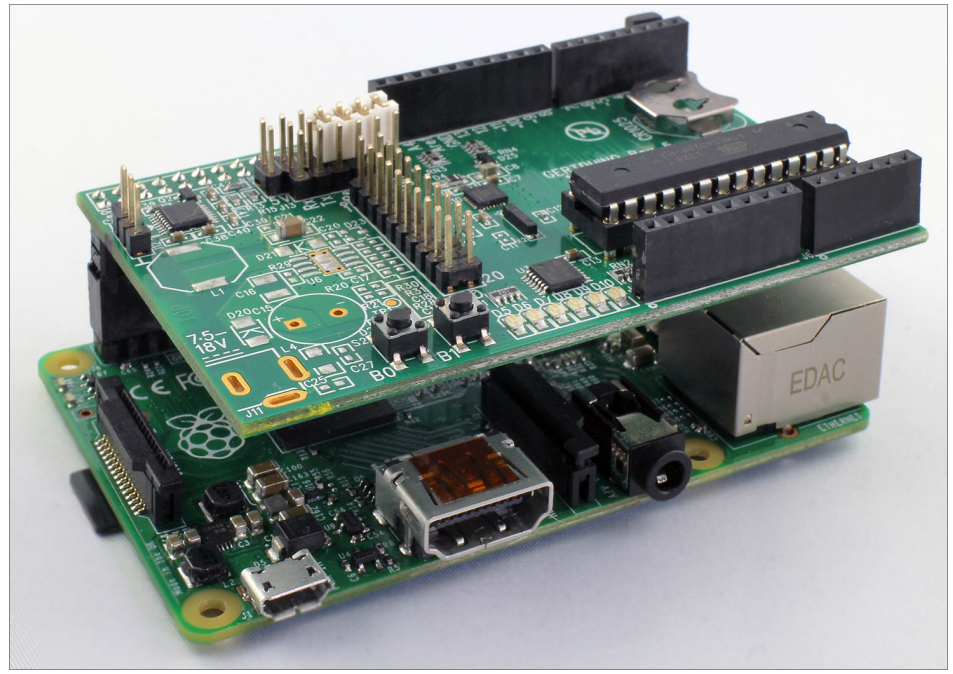


Abbildung 16.22 Das GertDuino auf dem Raspberry Pi

Zusätzlich zum verbauten ATmega-328 befindet sich ein ATmega-48 auf dem Board. Beide Microcontroller 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 mit allen Arduino-Uno-Shields.

Das GertDuino-Board finden Sie u. a. in folgendem Onlineshop:

http://www.designtools.at/Raspberry-Pi/Zusatz-Module/Gertduino-RPI-Atmega-Board.html

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 zeitkritische Aufgaben präziser erledigen.

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. Zuallererst legen wir Ihnen die offizielle Anleitung ans Herz, die die Grundfunktionen und den Aufbau der Platine beschreibt:

http://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. Er passt sowohl auf die alten Raspberry-Pi-Modelle als auch auf die aktuellen Modelle 3B/3B+/4B. 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 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 dem Raspberry Pi und dem ATmega328 her (siehe Abbildung 16.23).

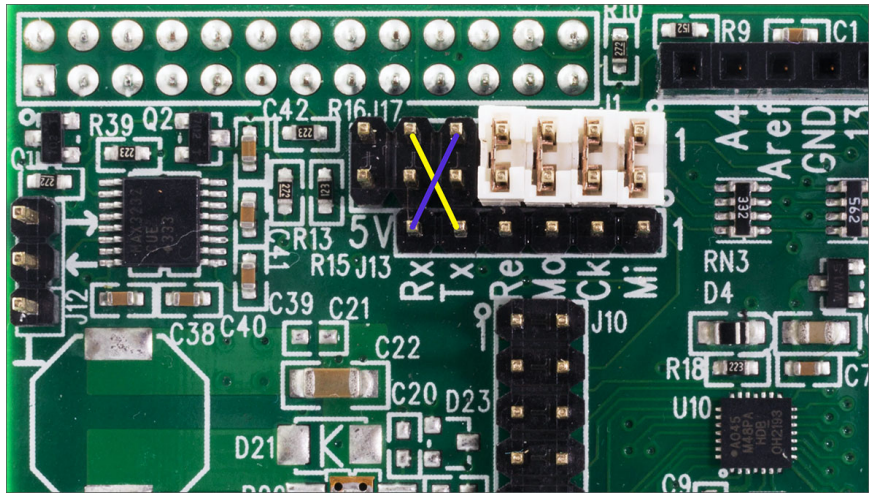


Abbildung 16.23 Alle vier Programmier-Jumper und in Gelb/Blau die benötigte UART-Verbindung

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 Raspberry-Pi-OS-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, ihn in eine Spannung umrechnet und über UART wieder an den Raspberry Pi schickt.

```
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:

https://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 wecken Sie also den ATmega aus dem Reset-Schlaf. Dadurch beginnt er 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 16.24).

```
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 16.24 Konsolenausgabe des Python-Programms

16.8 Raspberry-Pi-HATs

Etwa gleichzeitig mit dem Erscheinen des Raspberry Pi B+ und der damit verbundenen Layoutänderung der Platine hat die Raspberry Pi Foundation eine Richtlinie zur Standardisierung von Erweiterungsboards veröffentlicht. 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.

Zwar 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 eine richtige HAT wird?

- **Mechanische Spezifikationen:** Der HAT-Standard gibt die Geometrie der Erweiterungsplatine vor. So benötigt eine 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^2C-Schnittstelle sowie zur Möglichkeit des Backpowerings 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. Er soll in Zukunft einzigartige Informationen zu jedem HAT enthalten. 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-up-Widerstände), und es werden Fehler vermieden, wie z. B. die Belegung von 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 decken (siehe Abbildung 16.25). Vor dem Kauf von Prototyping-HATs achten Sie auf die Produktbeschreibung. Generell sind die HATs mit allen Raspberry-Pi-Modellen mit 40 GPIO-Pins kompatibel. Sie sollten jedoch nicht erwarten, dass aktuelle HATs auch direkt mit den alten A- und B-Modellen mit 26 GPIO-Pins kompatibel sind.

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. Produktinformationen zu zwei derartigen HATs finden Sie unter den folgenden Links:

https://distrelec.de/de/p/30133607
https://adafruit.com/products/2314

Ebenso finden Sie Prototyping-HATs für den Raspberry Pi Zero, so z. B. das *ProtoZero Board*:

https://thepihut.com/products/protozero

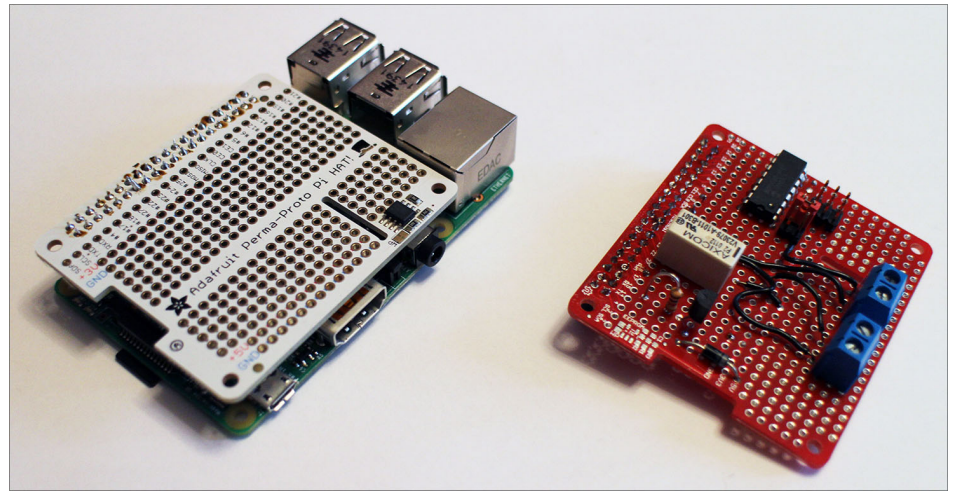


Abbildung 16.25 Zwei Prototyping-HATs: links auf dem Raspberry Pi, rechts bereits mit einer kleinen Schaltung bestückt

Beide Boards werden, sofern gewünscht, bereits mit EEPROM bestückt, jedoch ohne verlöteten Steckverbinder verkauft. Die vierzig Lötpunkte sind aber 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.

Unter dem Suchbegriff »raspberry pi zero prototyping hat« finden Sie auch bei deutschen Händlern jede Menge Auswahl. Sicherlich werden Sie bei Ihrer Suche nach einem Raspberry-Pi-Zero-Hat über den Begriff *pHAT* stolpern. Dies ist die Abkürzung für *partial HAT* und beschreibt oft die HAT-Boards für den Raspberry Pi Zero.

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 Rasp-

berry 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 enthält 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 enthält nun die für uns wichtigen Informationen. Hier können Sie Einstellungen zum GPIO-Port vornehmen:

- **gpio_drive:** Dieser Parameter kann Werte von `0` bis `8` enthalten und bestimmt die Strombegrenzung der frei verwendbaren GPIO-Pins. `0` behält die Standardwerte bei, `1`–`8` entsprechen 2, 4, 6, 8, 10, 12, 14 bzw. 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) und `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. Sie sollten einen Wert von `0` eintragen, 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. Den Wert `3` nutzen Sie, wenn Ihr HAT mindestens 2 A liefert.

Der letzte Parameterblock bestimmt nun die Funktion der genutzten GPIO-Pins. Der Pin 29 (GPIO 5) 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 GPIO 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 der Datei können Sie unverändert lassen.

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 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

16.9 Sense HAT – das Multitalent

Das *Sense HAT* (siehe Abbildung 16.26) ist ein offizielles Erweiterungsboard für alle Raspberry-Pi-B-Modelle. Das Board wurde 2015 entwickelt und wurde damals im Rahmen der Astro-Pi-Aktion zur Raumstation ISS geschickt. Auf dem Weg dorthin konnte der Raspberry Pi mit dem Sense HAT viele Umgebungsbedingungen messen und auswerten. Nach dem Ausflug in den Weltraum ist das Sense HAT nun für die Öffentlichkeit zugänglich und kann bei vielen bekannten deutschen Onlineshops gekauft werden.

Das Sense HAT ist mit folgenden Features ausgestattet:

- 8 × 8-RGB-LED-Matrix
- 5-Wege-Joystick
- Beschleunigungssensor

- Gyroskop
- Magnetfeldsensor
- Temperatursensor
- Luftfeuchtigkeitssensor

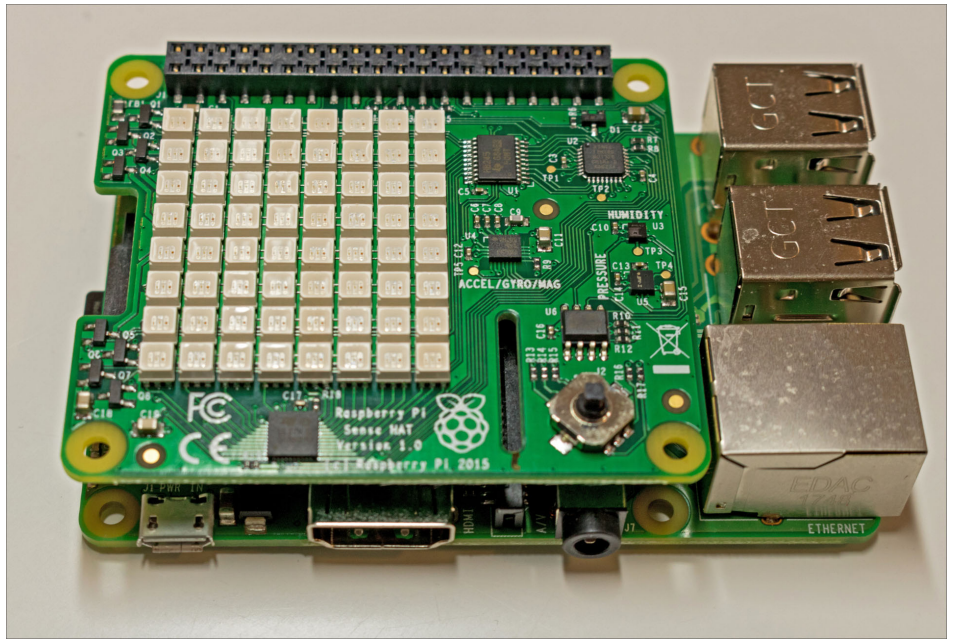


Abbildung 16.26 Das Sense HAT auf dem Raspberry Pi

Mit dieser Vielzahl an Sensoren lassen sich unzählige Projekte realisieren. Die Raspberry Pi Foundation hat hier ganze Arbeit geleistet und bietet exzellente Unterstützung im Umgang mit dem Sense HAT in Form von detaillierten und verständlichen Anleitungen sowie einem Sense-HAT-Simulator, der es erlaubt, Programme zu simulieren, ohne dass das Sense HAT vorhanden sein muss. Durch die grafische Oberfläche des Simulators sehen Sie auch direkt, welche Ausgaben die LED-Matrix anzeigen würde. Zu guter Letzt gibt es eine vorbildlich einfache Python-3-Bibliothek, die den Umgang mit dem Sense HAT so einfach wie möglich macht. Die Links zu den erwähnten Tools finden Sie am Ende dieses Abschnitts.

Wir möchten Ihnen, trotz der vielen offiziellen Informationen, einen kurzen Einblick in den Umgang mit dem Sense HAT geben, um Ihre Fantasie für eigene Projekte anzuregen.

Installation

Die Installation des Sense HATs kann nicht einfacher sein: Das Board wird einfach auf die GPIO-Leiste gesteckt. Hierzu liegt ein Erweiterungs-Header bei, der die Länge der GPIO-Pins etwas erhöht.

Das Sense HAT passt perfekt auf die B-Modelle des Raspberry Pi. Selbstverständlich können Sie das Sense HAT auch auf die Zero-Modelle stecken, dort passt es sich aber nicht so optimal in das Gesamtbild ein.

Als kleines Manko müssen wir erwähnen, dass nach dem Aufsetzen des Sense HATs alle GPIO-Pins belegt sind. Durch die flache Steckbuchse auf dem Board ist kein Zugang zu den Pins mehr möglich.

Software

Wir beginnen mit einem Python-Programm, das Ihnen fast alle grundlegenden Funktionen des Sense HATs nahebringen wird. Sofern Sie die neueste Version von Raspberry Pi OS installiert haben, ist die Sense-HAT-Python-Bibliothek bereits vorinstalliert. Anderenfalls können Sie diese einfach nachinstallieren:

```
sudo apt update
sudo apt install sense-hat
sudo reboot
```

Sie können nun direkt mit dem Schreiben Ihres Python-Programms starten. Unser Beispielprogramm hat folgenden Inhalt:

```
#!/usr/bin/python3
# coding=utf-8
# Datei sensehat.py
from sense_hat import SenseHat, ACTION_PRESSED, ACTION_HELD,
                      ACTION_RELEASED
from time import sleep

sense = SenseHat()
sense.clear()

black= (0,0,0)
red = (255,0,0)
green = (0,255,0)
white = (255,255,255)
```

```
def measure_pressure (event):
     if event.action != ACTION_RELEASED:
        pressure = sense.get_pressure()
        pressure=round(pressure, 1)
        sense.show_message(str(pressure), scroll_speed=0.1,
                           text_colour=black, back_colour=white)
        print(pressure)

def measure_temp(event):
    if event.action != ACTION_RELEASED:
        temp = sense.get_temperature()
        temp=round(temp, 1)
        sense.show_message(str(temp), scroll_speed=0.1,
                           text_colour=red, back_colour=white)
        print(temp)

def measure_humidity(event):
    if event.action != ACTION_RELEASED:
        humidity = sense.get_humidity()
        humidity=round(humidity, 1)
        sense.show_message(str(humidity), scroll_speed=0.1,
                           text_colour=green, back_colour=white)
        print(humidity)

sense.stick.direction_up = measure_pressure
sense.stick.direction_left = measure_temp
sense.stick.direction_right = measure_humidity
sense.stick.direction_middle = sense.clear

while True:
    sleep(0.1)
```

Wenn Sie das Programm starten, können Sie über den Joystick die Funktionen auswählen. Sobald Sie den Joystick nach oben drücken, wird der Luftdruck gemessen und in Laufschrift auf der LED-Matrix angezeigt. Drücken Sie den Joystick nach links, sehen Sie die Temperatur (siehe Abbildung 16.27). Bewegen Sie den Stick nach rechts, erhalten Sie die Luftfeuchtigkeit. Ein zentraler Druck auf den Joystick löscht das Display.

Zu Beginn definieren wir die Farben, die wir nutzen möchten. Das geschieht in Form von RGB-Codes. Auf dieser Webseite können Sie sehr einfach die RGB-Codes zu jeder Farbe generieren:

https://www.w3schools.com/colors/colors_rgb.asp

Danach definieren wir die Funktionen zum Messen der drei Werte. Jeder Wert wird in einer anderen der zuvor definierten Farben dargestellt.

Im letzten Teil des Programms sehen Sie, dass wir auf die Eingaben des Joysticks warten, um daraufhin die entsprechend benannte Funktion zu starten. In den Funktionen erkennen Sie, dass wir explizit darauf achten, den Moment des Loslassens (`ACTION_RELEASED`) zu ignorieren.

Das hat den Hintergrund, dass die Methoden `sense.stick.direction_XXX` die Events `PRESSED`, `RELEASED` und `HELD` wahrnehmen. Ohne diesen expliziten Ausschluss würde beim Drücken des Joysticks in eine Richtung die entsprechende Funktion zweimal ausgeführt: einmal für das Drücken (`PRESSED`) und einmal für das Loslassen (`RELEASED`).

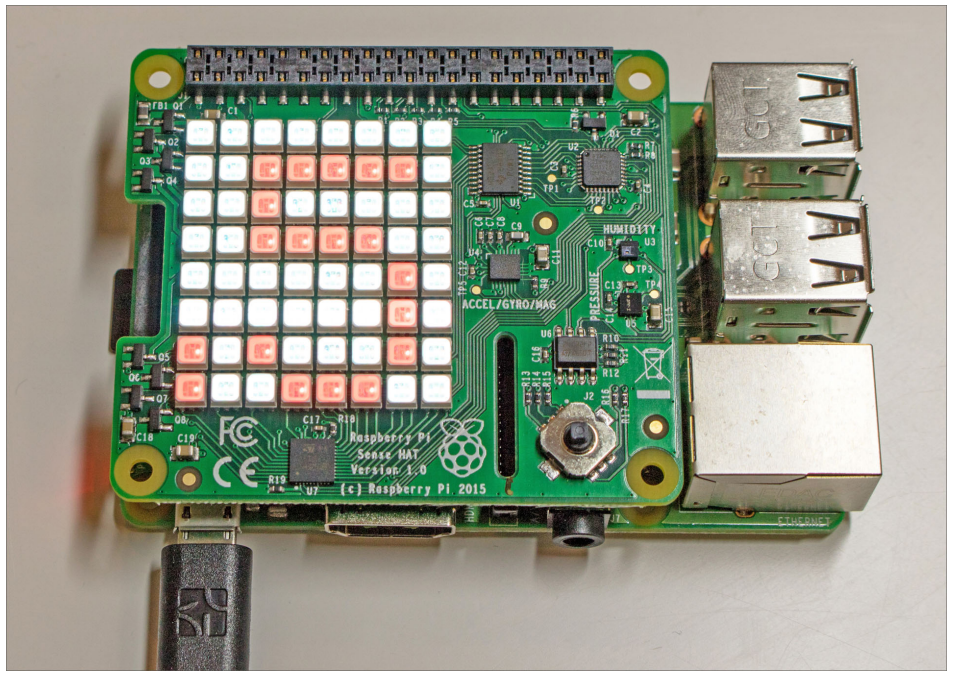


Abbildung 16.27 Das Sense HAT zeigt die Temperatur in Laufschrift an.

Neben den von uns verwendeten Funktionen haben Sie eine Vielzahl an weiteren Möglichkeiten. Alle Funktionen und Sensoren des Sense HATs finden Sie in der offiziellen Anleitung:

https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat

Zudem finden Sie unter den folgenden Links den Simulator und die Python-Bibliothek auf GitHub:

https://trinket.io/sense-hat https://github.com/RPi-Distro/python-sense-hat

Sie werden auf den Seiten viele Projekte finden, z. B. Spiele, Wetterstationen oder Infodisplays. Weil das Sense HAT so viele Ein- und Ausgabemöglichkeiten bietet, können Sie nun eigene umfangreiche Projekte verwirklichen. Wie wäre es mit einem eigenen Wetterballon, der mithilfe des Sense HAT diverse Messwerte während des Fluges mit den Sensoren für die Nachwelt festhält?

16.10 Adafruit PWM/Servo-HAT

Die Entwickler von Adafruit haben mit dem PWM/Servo-HAT ein sehr nützliches HAT entwickelt, das Hobbybastlern 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 für Ihr Projekt nicht ausreichen, so ist dieses Erweiterungsboard eine sehr gute Alternative. Das HAT kommuniziert über I^{2}C mit dem Raspberry Pi und ist mit jedem Raspberry-Pi-Modell kompatibel, das über den 40-Pin-GPIO-Header verfügt.

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://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 16.28).

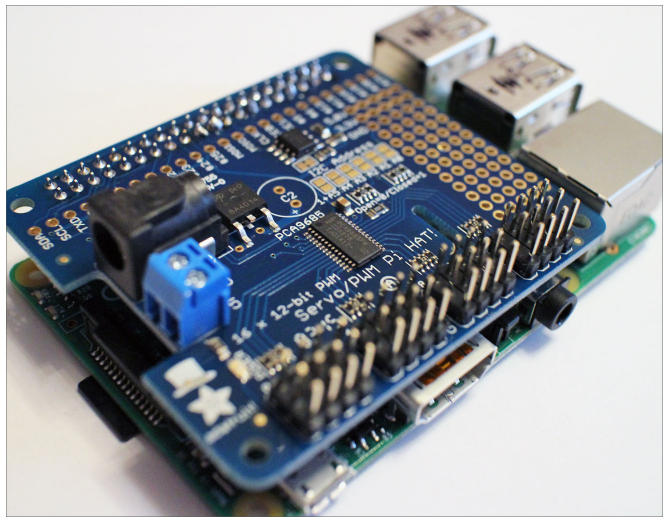


Abbildung 16.28 Das PWM/Servo-HAT auf dem Rücken des Raspberry Pi

PWM-Signale erzeugen

Prüfen Sie zuerst, ob das HAT erfolgreich am I^{2}C-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^{2}C-Schnittstelle korrekt eingerichtet haben (siehe Abschnitt 14.4, »I^{2}C«).

Die Ausgabe von `i2cdetect -y 1` sollte Ihnen nun das Board an 0x40 anzeigen (siehe Abbildung 16.29).

```
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 16.29 Das PWM/Servo-HAT wurde an Adresse 0x40 erkannt.

Das Generieren von PWM-Signalen ist mit der Adafruit-Python-Bibliothek ein Kinderspiel. Installieren Sie die passende Bibliothek:

```
sudo pip3 install adafruit-pca9685
```

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

```
#!/usr/bin/python3
# Datei pwm-hat.py
import Adafruit_PCA9685
import time

pwm = Adafruit_PCA9685.PCA9685()
pwm.set_pwm_freq(60) # 60 Hz
pwm.set_pwm(15, 0, 410) # 10 % Duty Cycle
pwm.set_pwm(8, 0, 1024) # 25 % Duty Cycle
pwm.set_pwm(4, 0, 2048) # 50 % Duty Cycle
time.sleep(2)
pwm.set_all_pwm(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:

- **set_pwm_freq(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.set_pwm(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.set_all_pwm(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/python3
# Datei servo-hat.py

import Adafruit_PCA9685
import time

pwm = Adafruit_PCA9685.PCA9685()
servoMin = 150
servoMax = 600
pwm.set_pwm_freq(60)

while (True):
  pwm.set_pwm(5, 0, servoMin)  # Minimale Endlage
  time.sleep(1)
  pwm.set_pwm(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 des BrickPi ist so entworfen worden, dass mehrere Methoden möglich sind. So können Sie z. B. einen Batterie-Pack verwenden, um die selbst gebauten Roboter mobil zu halten. Alternativ ist die Versorgung via Micro-USB möglich.

Kompatibilitätsproblem mit EV3-Sensoren

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

Den 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 16.30 bis Abbildung 16.31).

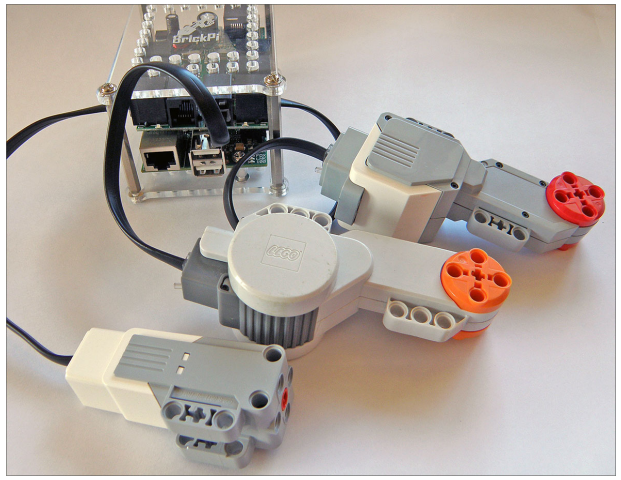


Abbildung 16.30 Die Anbindung der LEGO-Motoren an den BrickPi erfolgt durch die Standardleitungen. (Quelle: Dexter Industries)

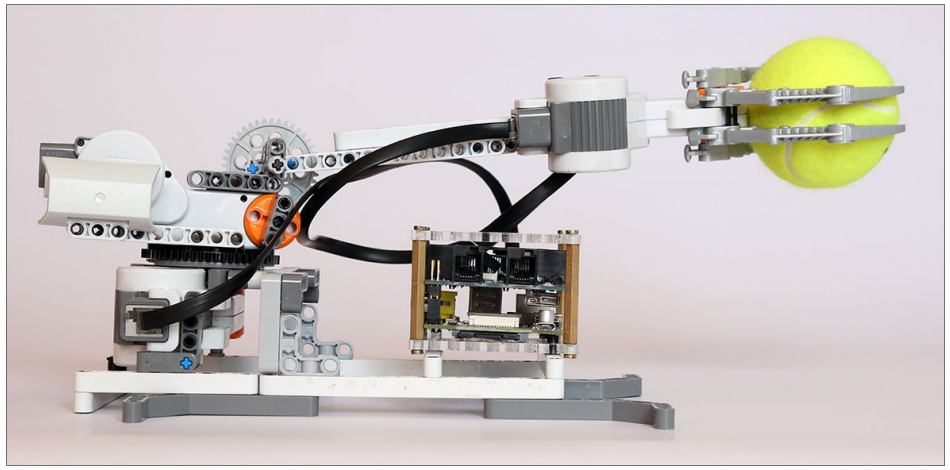


Abbildung 16.31 LEGO-Greifarm für einen Tennisball (Quelle: Dexter Industries)

16.12 GrovePi

Der *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 16.32).

Der 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.

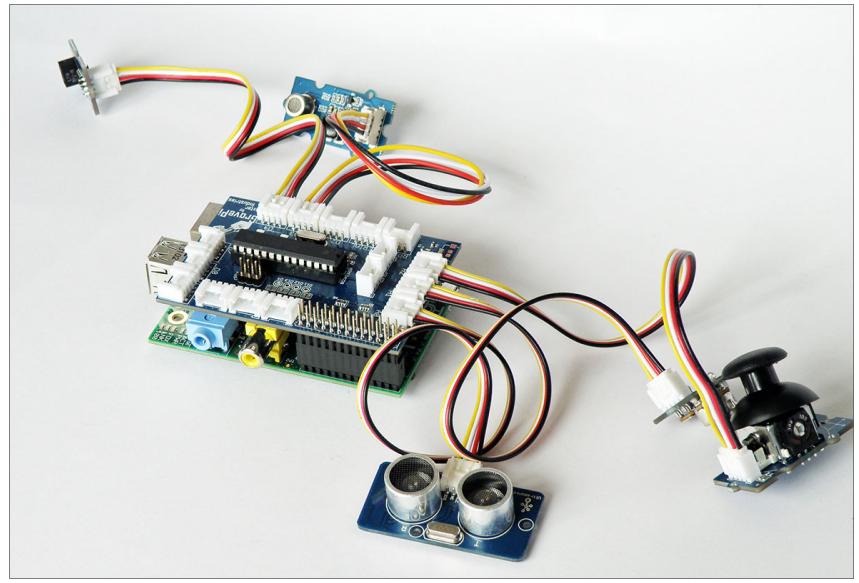


Abbildung 16.32 Das GrovePi-System samt angeschlossener Module (Quelle: Dexter Industries)

Dexter Industries führte dieses Konzept weiter und entwickelte eine Methode, 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 und Schalter
- LEDs
- Relais
- Temperatur-, Geräusch-, Berührungs- und Alkoholsensoren
- Displays

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

https://dexterindustries.com/GrovePi/sensors/supported-sensors

Der GrovePi wurde für den Raspberry Pi 1B entworfen, ist allerdings auch mit den aktuellen Modellen kompatibel:

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

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, das aus der Ausbildungstischlerei eines Gefängnisses in Nordrhein-Westfalen stammt (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 zwei separate Brutkammern, jede mit einem eigenen Eingang. Wie Sie auf dem Bild erkennen können, lassen sich Teile des Vogelhauses herausziehen, um die Reinigung zu vereinfachen.

Von den beiden Kammern soll nur die untere für brütende Meisen (auf deren Größe sind die Einfluglöcher bemessen) zur Verfügung stehen. Die obere Kammer soll den Raspberry Pi und seine Stromversorgung aufnehmen. Die Kamera, die mit einem schlanken Flachbandkabel mit dem Raspberry Pi verbunden ist, kommt an die Decke der unteren Kammer.

Vom Raspberry-Pi-Kameramodul gibt es zwei Varianten: eine »normale« und das sogenannte PiNoIR-Modul. Wir haben es zunächst mit der Standardkamera versucht, wählten dann aber schnell 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.

Das Modul löst acht Megapixel auf, liefert also Bilder mit 3.280 × 2.464 Pixeln. Kleinere Formate lassen sich per Software einstellen. Im Videobetrieb liefert das Modul maximal 1.080p (Full HD) bei 30 Bildern pro Sekunde. Es misst 25 × 20 × 9 Millimeter und wiegt 3 Gramm.

Abbildung 26.1 Das Vogelhaus vor dem Einbau der Technik

Im Vogelhaus reicht das durch die Einflugöffnung scheinende Mondlicht aus, um der Kamera brauchbare Aufnahmen zu ermöglichen (siehe Abbildung 26.2).

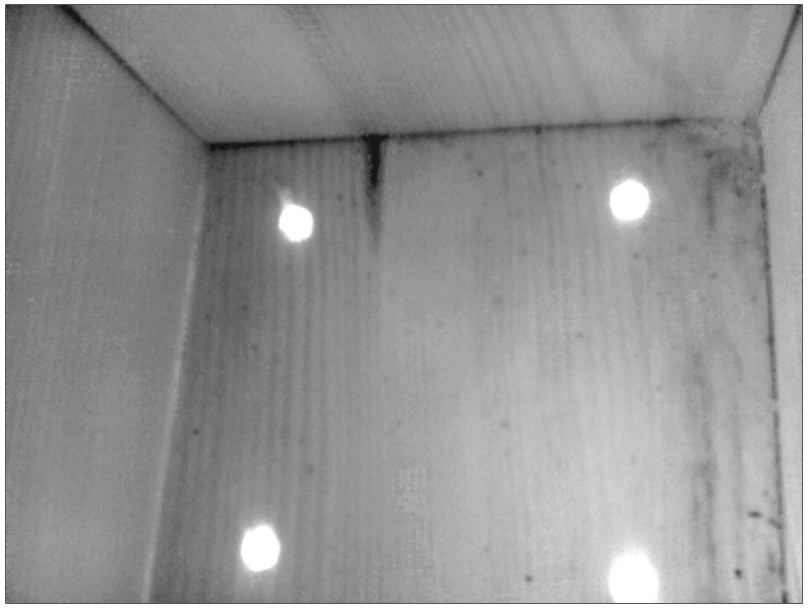


Abbildung 26.2 Die bezugsfertige Kammer, von innen fotografiert durch den Raspberry Pi bei Mondlicht ohne künstliche Beleuchtung

Flachbandkabel zu kurz?

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

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.

Wenn Sie ein Vogelhaus wählen, das auch größeren Vögeln Platz bietet, sollten Sie das Kamerakabel 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 des 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 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 sie softwareseitig aktivieren. Am einfachsten gelingt dies mit EINSTELLUNGEN • RASPBERRY PI CONFIGURATION. Sollten Sie Raspberry Pi OS Lite ohne Grafiksystem verwenden, rufen Sie stattdessen `sudo raspi-config` auf. 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.

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 (siehe auch Abschnitt 15.8, »Raspberry Pi Camera Board und PiNoIR«). Mit `raspistill` können Sie nicht nur einfach Bilder anfertigen, sondern sie 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 und im JPEG-Format gespeichert. Obwohl das JPEG-Format eine Kompression enthält, 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 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, sollten Sie versuchen, den Parameter `-ev` (Exposure Value, Belichtungskorrektur) zu ändern. Das Bild wird, wenn Sie hier einen positiven Wert angeben, 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 Videoaufnahmen 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 -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 ihn 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 Videoformat `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.

```
sudo apt update
sudo apt full-upgrade
sudo apt -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 1.500 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 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 Abspiel-Software, 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 Livebild 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.

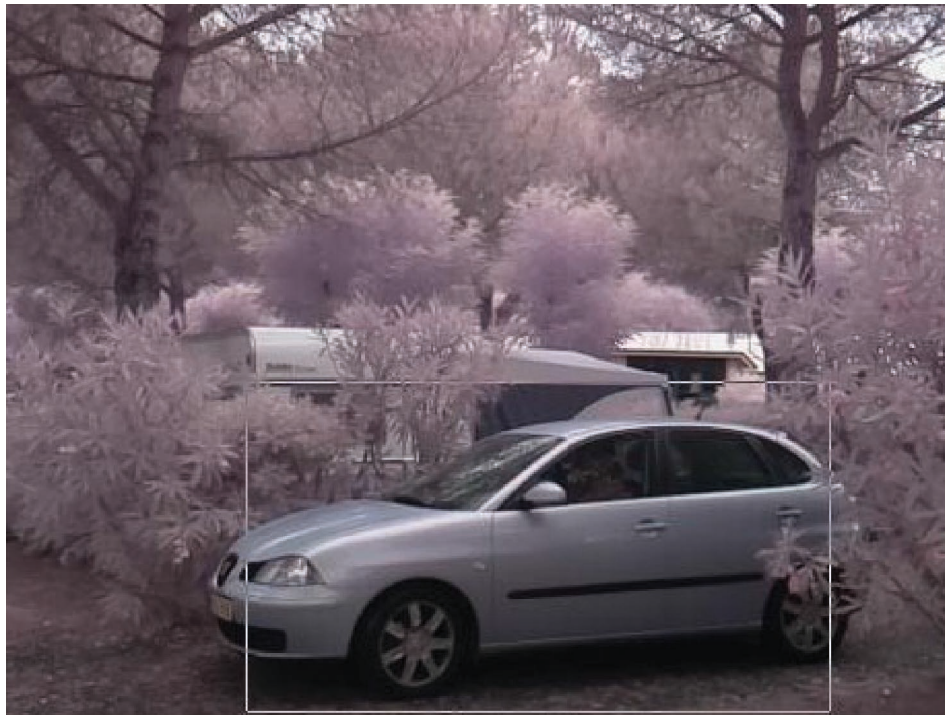


Abbildung 26.3 motion erkennt eine Bewegung.

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 Adress-

zeile 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 26.3) und startet gleichzeitig die Aufnahme.

Der fehlende Infrarotfilter des PiNoIR-Kameramoduls verfälscht die Farben, wenn man es bei Tageslicht einsetzt.

`motion` speichert die Aufnahmen 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. Die Konfigurationsdatei `crontab` 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 5.10, »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 eingesetzt. 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 1.024 × 768 Pixel geschossen. Es bietet sich an, das von `cron` erledigen zu lassen (siehe Abschnitt 5.10). 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 Ver-

zeichnis 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.

Abbildung 26.4 Eine Kohlmeise im Vogelhaus

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-Pi-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. Wenn Sie dennoch das letzte Quäntchen Schärfe herausholen möchten, finden Sie in unserem Blog eine Anleitung, wie Sie die Naheinstellgrenze der Kamera verändern:

https://pi-buch.info/naheinstellgrenze-der-raspberry-pi-kamera-veraendern

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 26.4), 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!