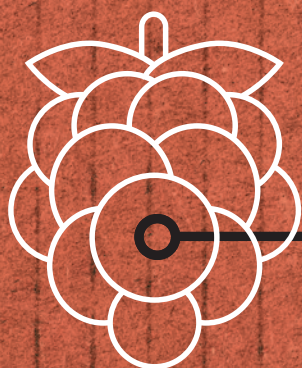
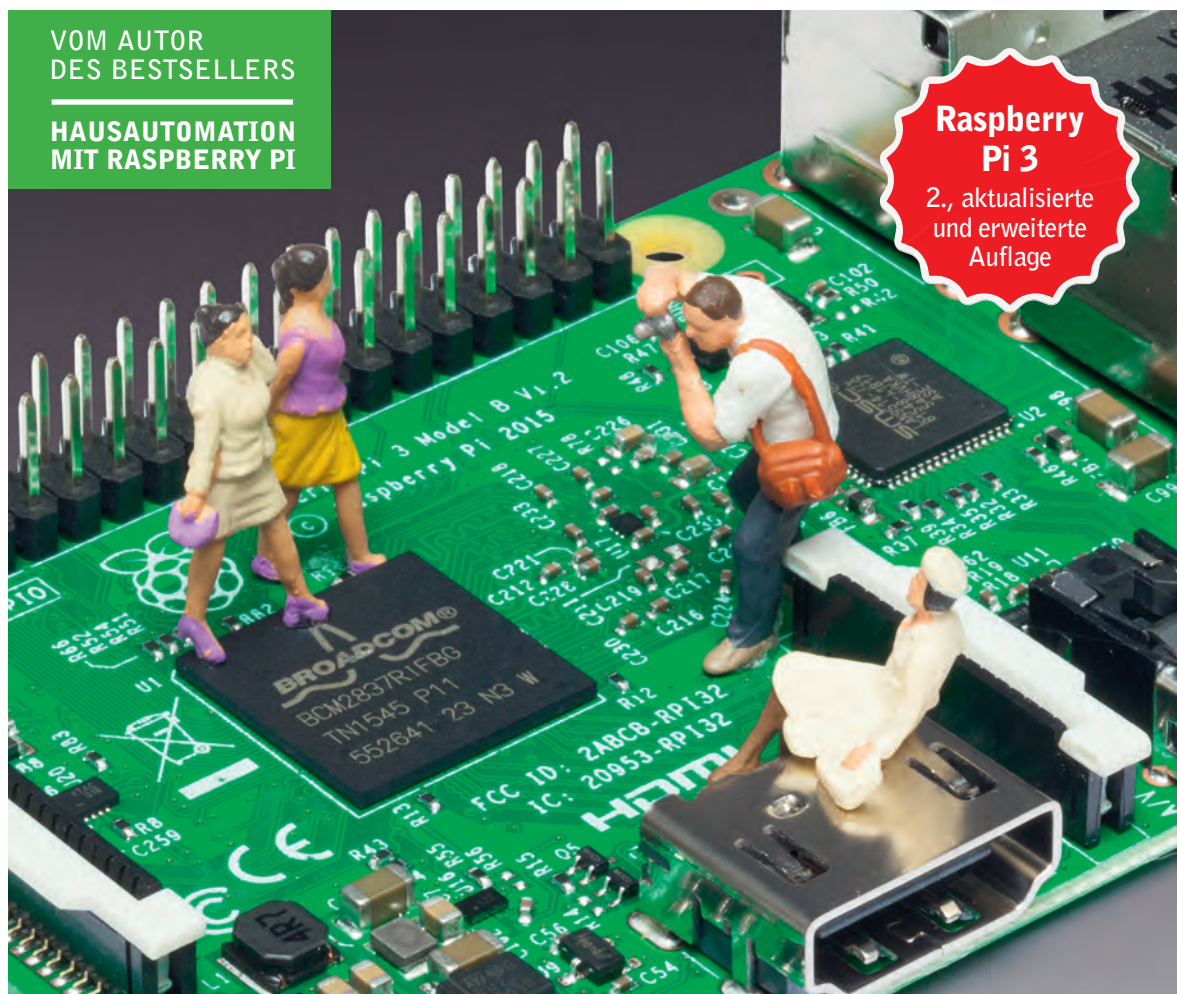


VOM AUTOR
DES BESTSELLERS

HAUSAUTOMATION
MIT RASPBERRY PI

**Raspberry
Pi 3**

2., aktualisierte
und erweiterte
Auflage



**FRUIT UP
YOUR
FANTASY**

E. F. ENGELHARDT

SENSOREN AM RASPBERRY PI

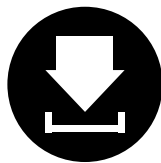
Der Raspberry Pi erfasst alles, analog oder digital: Temperatur, Abstand, Infrarotlicht, Bilder, Bewegung, Stromstärke, Gas, Neigung und mehr. 25 Sensoren, und Sie haben Ihre Umgebung im Griff.

FRANZIS

E. F. Engelhardt

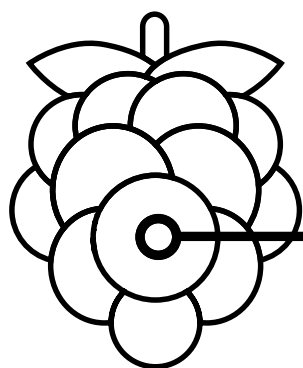
Sensoren am Raspberry Pi

2., aktualisierte und erweiterte Auflage 2016



Zusatzinformationen zum Produkt

Zu diesem Produkt, wie zu vielen anderen Produkten des Franzis Verlags, finden Sie unter www.buch.cd Zusatzmaterial zum Download.



FRUIT UP
YOUR
FANTASY

E. F. ENGELHARDT

SENSOREN AM RASPBERRY PI

Der Raspberry Pi erfasst alles, analog oder digital: Temperatur, Abstand, Infrarotlicht, Bilder, Bewegung, Stromstärke, Gas, Neigung und mehr. 25 Sensoren, und Sie haben Ihre Umgebung im Griff.

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Alle Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag und der Autor sehen sich deshalb gezwungen, darauf hinzuweisen, dass sie weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen können. Für die Mitteilung etwaiger Fehler sind Verlag und Autor jederzeit dankbar. Internetadressen oder Versionsnummern stellen den bei Redaktionsschluss verfügbaren Informationsstand dar. Verlag und Autor übernehmen keinerlei Verantwortung oder Haftung für Veränderungen, die sich aus nicht von ihnen zu vertretenden Umständen ergeben. Evtl. beigefügte oder zum Download angebotene Dateien und Informationen dienen ausschließlich der nicht gewerblichen Nutzung. Eine gewerbliche Nutzung ist nur mit Zustimmung des Lizenzinhabers möglich.

© 2016 Franzis Verlag GmbH, 85540 Haar bei München

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Das Erstellen und Verbreiten von Kopien auf Papier, auf Datenträgern oder im Internet, insbesondere als PDF, ist nur mit ausdrücklicher Genehmigung des Verlags gestattet und wird widrigenfalls strafrechtlich verfolgt.

Die meisten Produktbezeichnungen von Hard- und Software sowie Firmennamen und Firmenlogos, die in diesem Werk genannt werden, sind in der Regel gleichzeitig auch eingetragene Warenzeichen und sollten als solche betrachtet werden. Der Verlag folgt bei den Produktbezeichnungen im Wesentlichen den Schreibweisen der Hersteller.

Autor: E. F. Engelhardt

Programmleitung: Dr. Markus Stäuble

Lektorat: Ulrich Dorn und Dr. Markus Stäuble

Satz: DTP-Satz A. Kugge, München

art & design: www.ideehoch2.de

Druck: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Printed in Germany

2., aktualisierte und erweiterte Auflage 2016

ISBN 978-3-645-60490-1

Vorwort

Das vorliegende Buch ist kein einsteigerfreundliches Buch für jene, die gerade erst dabei sind, das Thema Raspberry Pi neu zu entdecken. Entsprechendes Grundlagenwissen für die Installation des Raspberry Pi wird zwingend vorausgesetzt – ebenso sollten Sie wissen, wie herum ein Lötkolben zu halten ist, und der Einsatz von Bash-Skripten und Python sollte auch keine schlaflosen Nächte bereiten. Denn das Buch beginnt dort, wo andere aufhören: Kauf und Anschluss des Sensors sind das eine, das andere ist bei vielen Sensoren die Schwierigkeit, sie auch im Rahmen der Möglichkeiten mit dem Raspberry Pi ordnungsgemäß in Betrieb zu nehmen.

Manche Sensoren brauchen aufgrund ihrer Bauweise Nachhilfe in Form der Auswahl einer geeigneten Schnittstelle, sei es ein gewöhnlicher Analog-digital-Wandler oder eine Erweiterungsplatine, die auf den GPIO-Pfosten des Raspberry Pi gesteckt wird. Aber auch die Datenausgabe der Messwerte ist oftmals eleganter, wenn diese – beispielsweise die Temperatur des Wasserkessels – direkt von einem kleinen LC-Display abgelesen werden kann. Für diese Schwierigkeiten bietet das Buch ebenfalls praxisnahe und dank Quellcode sofort umsetzbare Lösungen an, mit denen Sie auch fertige Erweiterungen wie das Gertboard, das GertDuino- oder das Embedded-Pi-Erweiterungsboard auf Arduino-Basis in Betrieb nehmen können.

Nicht nur im Auto sorgen die elektronischen Helferlein in Form verbauter Sensoren für mehr Sicherheit und zusätzlichen Komfort, auch zu Hause und im Haushalt jenseits der Hausautomation sind sie mehr oder weniger sichtbar im Einsatz: Egal ob der berührungslose Seifenspender von Sagrotan, der Airwick-Luftaromatisierer für die Gästetoilette oder ein elektronischer Schwangerschaftstest – in sehr vielen Haushaltsartikeln sind Sensoren mittlerweile kaum mehr wegzudenken.

Der Raspberry Pi bietet dank der zahlreichen GPIO-Anschlüsse und der beiden Spannungspins 3,3 V und 5 V (V = Volt) viele Möglichkeiten, solche Sensoren aus dem Haushalt zu zweckentfremden – aber auch die breite Palette an Aktoren und Sensoren aus der Mikrocontroller-Welt und aus dem Fachhandel für Kommunikations- und Steuerungssysteme lässt sich am Raspberry Pi nutzen. Die meisten in diesem Buch beschriebenen Sensormodule und Aktoren für den Raspberry Pi finden Sie für kleines Geld auf den einschlägigen Auktions- und Kaufhausseiten im Netz. Beachten Sie hier, dass die Lieferanten solcher Arduino- und Raspberry-Pi-tauglichen Billigmodule meist in China oder Hongkong sitzen, was drei Wochen oder mehr Lieferzeit bedeuten kann. Als Dankeschön für die Wartezeit winken günstige Preise – soll es schneller gehen, bieten hiesige Anbieter die gleichen Module mit einem Aufschlag an und sind damit im Vergleich deutlich teurer.

Mit den in diesem Buch vorgestellten Sensoren, Schaltplänen, Techniken und Algorithmen lässt sich nahezu jeder denkbare Einsatzzweck abdecken. Angefangen bei

Akustik- und Bewegungssensor über die Messung der Gas- und Luftqualität sowie die Farb- und Lichtmessung, die Sensoren für die Feuchte-, Temperatur- und Strom- bzw. Stromstärkemessung bis hin zu Druck-, Höhen- und Lagesensoren sowie dem Einsatz der Raspberry-Pi-Kamera als Sensor und zu guter Letzt den Infrarot- und Ultraschallsensoren werden alle in diesem Buch am Raspberry Pi betrieben.

Wir wünschen Ihnen viel Spaß mit und vor allem viel Nutzen von diesem Buch!

Autor und Verlag

Sie haben Anregungen, Fragen, Lob oder Kritik zu diesem Buch? Sie erreichen den Autor per E-Mail unter ef.engelhardt@gmx.de.

Inhaltsverzeichnis

1	Raspberry Pi: Schnittstellen und Erweiterungen.....	9
1.1	Raspberry Pi für Hardwareprojekte konfigurieren.....	10
1.2	Programmierung der GPIO-Pinleiste	19
	GPIO-Leiste - Unterschiede: BCM-, WiringPi- und Pin-Zählung.....	19
	GPIO-Funktionen nutzen - Pinbelegung entschlüsselt.....	21
	Jenseits von Pin 26: GPIO-Anschlüsse des Raspberry Pi 1 B+	24
1.3	Python-Zugriff mit der RPi.GPIO-API	32
	LED-Praxis mit der RPi.GPIO-Bibliothek	34
	PIR-Praxis mit der RPi.GPIO-Bibliothek	35
1.4	WiringPi-API: schnell auf der Shell.....	36
	PIR-Modul am Raspberry Pi Zero	38
	Shell-Skript für PIR-Bewegungsmelder	41
1.5	gpiozero-Bibliothek im Einsatz	42
	LED-Praxis mit der gpiozero-Bibliothek	43
	PIR-Bewegungsmelder mit der gpiozero-Bibliothek	44
	Viele Klassen und Funktionen.....	45
1.6	I ² C-Protokoll - neue Spielregeln	46
	LCD-Bildschirm am I ² C-Bus	51
	LCD-I ² C-Adapter mit PCF8574 im Eigenbau	60
1.7	Analog-digital-Wandler MCP3008 nachrüsten	68
	MCP3008 auf dem Steckboard nutzen	70
	Programmierung des MCP3008 mit Python	73
1.8	GPIO-Porterweiterung mit MCP23017 und I ² C	78
	Anschluss und Adressierung des MCP23017.....	79
	LED-/Schalter-Projekt mit dem MCP23017.....	81
	MCP23017-Register - Kontrolle und Adressierung.....	82
	Schalten der zusätzlichen GPIO-Ausgänge	85
	MCP23017 am I ² C-Bus mit Python.....	85
1.9	Erweiterungsplatinen für den Raspberry Pi	88
2	Hören, sehen und fühlen mit Sensoren.....	121
2.1	Aktive vs. passive Sensoren	123
2.2	Licht- und Farbsensoren im Raspberry-Pi-Einsatz.....	124
	LDR-Lichtsensorschaltung auf dem Steckboard	124
	TCS34725-Farbsensor installieren und einsetzen.....	128
	CCT (Farbtemperatur) und CIE-Werte bestimmen.....	132

APDS-9002-Lichtsensoren mit MCP3008 nutzen	136
SPI-Schnittstelle aktivieren	137
Mehrere Analogsensoren über py-spidev verarbeiten	141
2.3 Temperaturmessung mit LM35 und MCP3008	144
LM35D-Temperatursensor und MCP3008-IC koppeln	145
LM35D mit Python und py-spidev auslesen	146
2.4 Temperaturmessung mit dem DS18B20-Sensor	148
Temperatursensor in Betrieb nehmen	152
Temperaturmessung mit Python	156
2.5 Schallali, Schallala – Ultraschallsensor ist da!	158
Abstandssensor mit Python-Skript in Betrieb nehmen	161
2.6 Infrarotabstandssensor im Einsatz	164
Infrarotabstandssensor mit Python	164
Ultraschall- und IR-Abstandssensoren kombinieren	166
2.7 Freie Auswahl – Sharp-Abstandssensor	169
Messwertbestimmung der Sharp-Abstandssensoren	171
2.8 Bewegungssensor mit Infrarotmodul	176
Shell-Skript für PIR-Bewegungsmelder	180
2.9 Raspberry-Pi-Kameramodul als Kamerasensor	181
OpenCV für die Kamera	192
Kamerasensor für die Gesichtserkennung	193
2.10 Infrarotsensor – Schwarz und Weiß auf der Linie	195
QTR-8RC-Sensor mit Raspberry Pi und GertDuino nutzen	197
QTR-8RC-Sensor am Analog-digital-Wandler MCP3008	202
2.11 Touch- und Drucksensor – Dateneingabe via I ² C-Bus	206
2.12 Akustiksensoren – Tanzbär mit dem Raspberry Pi	210
2.13 Höhenbestimmung mit dem BMP085-Luftdrucksensor	215
2.14 Lage- und Neigungssensor SW-520D	222
Schaltung für den Neigungssensor	223
2.15 Gyrometer-Experimente mit dem Raspberry Pi	227
Sensoren im I ² C-Einsatz auf dem Raspberry Pi	227
Gyrosensoren – Begriffe und Unterschiede	229
Sensorexperimente mit dem MPU-6050 und Python	232
Gyroskop mit Druckmesser – Pololu AltIMU-10	236
Die Gyrosensoren AltIMU-10 und MiniIMU-9 v2 mit Python nutzen	238
2.16 Hygrometer als Feuchtesensor im Blumentopf	241
2.17 Stromstärkemessung mit Linear-Hall-Effekt-Sensor	244
2.18 Gas- und Rauchsensor – Alarmanlage mit dem Raspberry Pi	250
Stichwortverzeichnis	255



Raspberry Pi: Schnittstellen und Erweiterungen

Die zentrale Schnittstelle für das Messen, Steuern und Regeln von angeschlossenen Schaltern, Sensoren und Aktoren ist die GPIO-Schnittstelle (*General Purpose Input/Output*) des Raspberry Pi. Mit dieser Schnittstelle sind Sie für sämtliche Dinge in diesem Buch gerüstet – hier erweitern Sie den Raspberry Pi mit Schaltungen und Funktionen auf dem Steckboard, die später per Lötkolben in ein »festes« Platinendesign überführt werden können. Neben den Sensor- und Schaltungslösungen »Marke Eigenbau« können Sie auch auf die mittlerweile zahlreichen zusätzlich zu erwerbenden Steck- und Erweiterungsboards zurückgreifen, wie beispielsweise das Embedded-Pi- und das GertDuino-Board oder das umfangreich bestückte Erweiterungsboard Gertboard, und sie für eigene Sensorikzwecke nutzen.

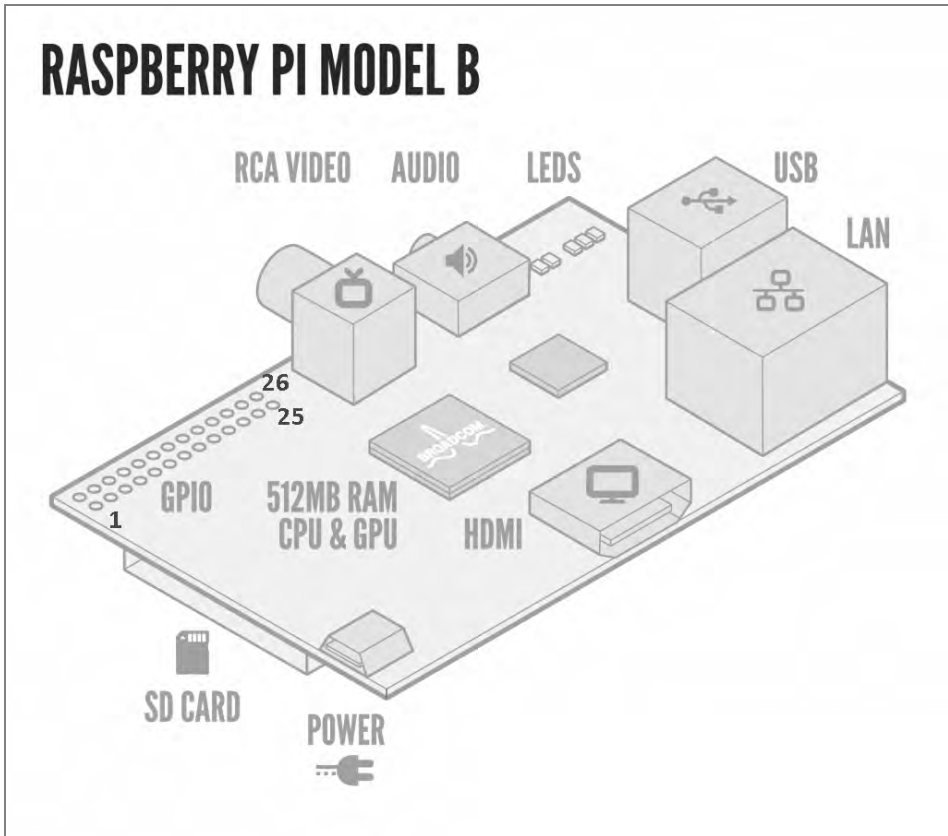


Bild 1.1: Grundsätzlicher Aufbau des Raspberry Pi und der GPIO-Pinleiste Model B, Revision 2.
(Grafik: raspberrypi.org)

1.1 Raspberry Pi für Hardwareprojekte konfigurieren

Egal ob Sie selbst für die vielen verschiedenen Sensoren am Markt eine passende Schaltung im Eigenbau entwickeln oder eine Hilfsplatine wie das PiFace oder das Gertboard nutzen möchten, das A und O ist der Zugriff per Software auf die Schnittstelle bzw. die Funktionen der einzelnen GPIO-Pins. Dafür stehen zahlreiche Möglichkeiten, viel Zubehör und Erweiterungsboards zur Verfügung, die in den nachfolgenden Kapiteln Schritt für Schritt erklärt und eingesetzt werden. Doch diese technischen Hilfsmittel sind oft wertlos, wenn das Grundsystem nicht stimmt – zunächst müssen Sie für ein aktuelles Betriebssystem und Treiber sowie für die Installation der notwendigen Tools und Hilfsmittel sorgen, damit der Umgang mit den Sensoren aller Art auf dem Raspberry Pi von Anfang an erfolgreich ist.

Alles für ein frisches und knackiges Raspbian

Unabhängig davon, ob es sich um einen »normalen« Computer mit Windows, Linux oder Mac OS X oder um einen kleinen Raspberry Pi handelt, ein topaktuelles System ist immer empfehlenswert. Der Vorteil eines frisch aktualisierten Systems besteht vor allem darin, dass Sie hier etwaigen Fehlern schon im Vorfeld aus dem Weg gehen. Dank der Prüfung möglicher Abhängigkeiten bleiben Pakete aktuell, auch wenn sie nicht explizit zum Aktualisieren ausgewählt werden.

Wenn Sie unsicher sind, welcher Kernel beispielsweise im Moment im Einsatz ist, nutzen Sie dafür das `uname`-Kommando:

```
uname -a
```

Einen alten bzw. veralteten Kernel können Sie »upgraden« und brauchen nicht von vorne zu starten, indem Sie die Raspbian-Imagedatei installieren. Stattdessen nutzen Sie bei einem bestehenden System den Befehl:

```
sudo apt-get update
```

Erst nach einem `update` ist das `upgrade` wirklich sinnvoll, da erst mit dem `apt-get update`-Kommando die lokale Paketdatenbank auf den aktuellsten Stand gebracht wird. Mit dem Befehl

```
sudo apt-get upgrade
```

aktualisieren Sie das installierte Raspbian, bei einem notwendigen Kernel-Upgrade verwenden Sie nach einem etwaigen Neustart per `sudo reboot` anschließend dieses dazugehörige Kommando:

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

Während sich das `apt-get upgrade` primär um Anwendungen und Treiber kümmert, sorgt `apt-get dist-upgrade` für den aktuellen Kernel und installiert dessen Updates. Hat das Installationsprogramm hier Änderungen durchgeführt, sollten Sie den Raspberry Pi neu starten und den Neustart live mitverfolgen. Nur dann haben Sie die Gewissheit, dass bei der Kernel-Aktualisierung alles gut gegangen ist und alle Dienste wieder starten.

Ungenutzte Tools und Pakete entrümpeln

Je nach Einsatzzweck des Raspberry Pi ist es oftmals, und nicht nur aus Speicherplatzgründen, sinnvoll, nicht benötigte Tools und Pakete zu deinstallieren. Wer etwa den Raspberry Pi ausschließlich als Überwachungsserver für diverse Sensoren im Heimnetz betreiben möchte, benötigt Dinge wie beispielsweise den grafischen Desktop samt dazugehöriger Tools nicht. Entfernen Sie diese, schaffen Sie zudem auch Platz auf der (Micro-)SD-Karte.

```
apt-get purge --auto-remove scratch debian-reference-en dillo idle3  
python3-tk idle python-pygame python-tk lightdm gnome-themes-standard
```

```
gnome-icon-theme raspberrypi-artwork gvfs-backends gvfs-fuse desktop-base  
lxpolkit netsurf-gtk zenity xdg-utils mupdf gtk2-engines alsa-utils lxde  
lxtask menu-xdg gksu midori xserver-xorg xinit xserver-xorg-video-fbdev  
libraspberrypi-dev libraspberrypi-doc dbus-x11 libx11-6 libx11-data libx11-  
xcb1 x11-common x11-utils lxde-icon-theme gconf-service gconf2-common  
iceweasel
```

Das Entfernen obiger Pakete hat in diesem Beispiel mehrere Hundert MB Speicherplatz freigeräumt, was sich gerade beim Einsatz einer klein dimensionierten (Micro-) SD-Karte auszahlt.

```
apt-get autoremove  
sudo apt-get update  
sudo apt-get upgrade  
sudo reboot
```

Nach dem Entrümpeln sollten Sie sicherheitshalber die Abhängigkeiten prüfen und anschließend das System auf den aktuellen Stand bringen, bevor Sie mit dem **reboot**-Kommando den Raspberry Pi neu starten.

Fernzugriff und Remotedesktop

Wenn Sie den Raspberry Pi in das Heimnetz und ins Internet bringen möchten, muss er über ein Kabel an den Verteiler, also den Router, angeschlossen werden. Sie können auch eine Netzwerkverbindung per Funk anlegen. Dazu benötigen Sie nur einen passenden WLAN-Adapter für den Raspberry Pi – bei dem Raspberry Pi 3 ist bereits ein WLAN-Anschluss standardmäßig mit dabei. Egal welche Netzwerkschnittstelle Sie im Endeffekt nutzen, standardmäßig ist auf dem Raspberry Pi ein DHCP-Client aktiv, der seine Netzwerkparameter vom DHCP-Server (*Dynamic Host Configuration Protocol*) in Ihrem Heimnetz bezieht. Bekanntlich liefert DHCP nicht nur die IP-Adresse, sondern es lassen sich auch Einstellungen zum DNS-Server, dem Gateway, der Netzmaske, der Domäne und weiterem mithilfe von Optionen automatisch tätigen. Die IP-Adresse des Raspberry Pi kann statisch, aber in Abhängigkeit von der MAC-Adresse des Rechners, oder dynamisch zugewiesen werden. Kurzum: Der Raspberry Pi bekommt »seine« IP-Adresse und die dazugehörigen Netzwerkeinstellungen automatisch zugewiesen. Wer nicht immer direkt vor dem Raspberry Pi sitzen, sondern diesen aus der Ferne steuern möchte, der nutzt auf der Kommandozeile den SSH-Zugriff.

Einmal eingerichtet, können Sie benutzerabhängig nahezu nach Belieben auf die System- und Nutzerdaten auf dem Zielcomputer zugreifen, Daten hin- und herkopieren und vieles mehr. Manche Lösungen in diesem Buch, wie beispielsweise die Nutzung der Arduino-Software für das Gertboard, GertDuino oder das Embedded-Pi-Board, benötigen die grafische Benutzeroberfläche **lxde**. Dafür brauchen Sie im Heimnetz je nach Computer nicht nur ein Programm wie Remotedesktop, VNC etc. und eventuell noch zusätzliche Tools, sondern auch die sichere SSH-Verbindung muss entsprechend eingerichtet werden.

Kommandozeilenadministration mit der Allzweckwaffe SSH

Wer den Raspberry Pi mit installiertem Linux ohne Tastatur, Maus und eigenen Bildschirm betreiben möchte, ihn aber dennoch vom Schreibtisch oder der Elektronikwerkstatt aus bequem steuern und programmieren will, der benötigt einen Fern- bzw. Remotezugriff auf das Gerät. Damit nutzen Sie die entfernte Kommandozeile auf dem lokalen Rechner, egal ob es sich bei Ihrem Arbeitsplatz-PC um einen Linux, Mac- oder Windows-Rechner handelt.

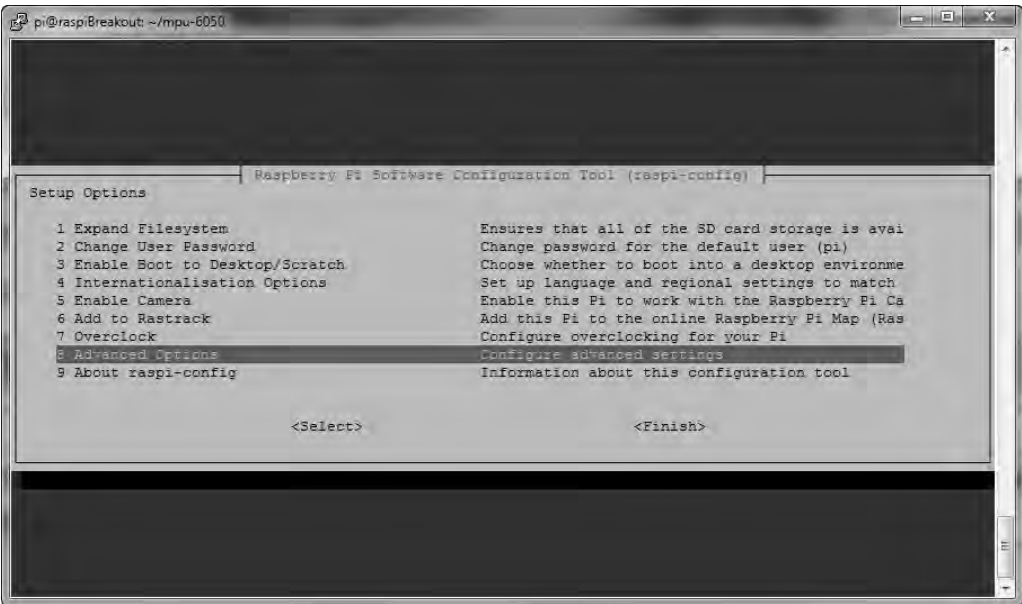


Bild 1.2: Grundvoraussetzung für den SSH-Zugriff ist selbstverständlich ein installierter SSH-Client auf dem Computer sowie ein installierter und konfigurierter SSH-Server auf dem Raspberry Pi. Bei den neueren **raspi-config**-Versionen ist der Konfigurationsdialog von SSH in das Menü *Advanced Options* verschoben worden.

Ist der Raspberry Pi frisch mit dem Raspbian-Betriebssystem installiert, richten Sie diesen mit dem Kommando **raspi-config** ein, wählen dort im Hauptmenü den *Advanced Options*-Eintrag und anschließend den *SSH*-Menüeintrag aus. Bestätigen Sie die Frage *Would you like the SSH server enabled or disabled?* des Assistenten mit der **[Enter]**-Taste, nachdem Sie mit den Pfeiltasten auf die *Enable*-Schaltfläche gewechselt sind. Damit aktivieren Sie den automatischen Start des eingebauten SSH-Servers.

Je nach verwendeter Version erscheint dieser Konfigurationsdialog, in dem sich der Start des SSH-Servers festzurren lässt, damit dieser nach jedem Einschalten zur Verfügung steht, beim erstmaligen Einschalten des Raspberry Pi automatisch. Anschließend können Sie den Raspberry Pi ohne Bildschirm, Maus und Tastatur betreiben und sich über das Netzwerk mit jedem beliebigen Client über das sichere SSH-Protokoll mit dem Raspberry Pi verbinden.

IP-Adresse des Raspberry Pi im Heimnetz herausfinden

Ist das Netzkabel in den Raspberry Pi eingesteckt und ist dieser gestartet, ist es anfangs noch nicht ganz klar, ob bzw. mit welcher IP-Adresse der Raspberry Pi im Heimnetz überhaupt erreichbar ist. Hier behelfen Sie sich am schnellsten mit dem Start des Konfigurationsmenüs des heimischen WLAN/LAN-Routers und prüfen dort im Bereich *Netzwerk* oder *DHCP-Server* bei den Namen und IP-Adressen, ob dort ein neuer Hostname *raspberrypi* vorhanden ist oder nicht.

Standardmäßig bezieht der Raspberry Pi seine IP-Adresse automatisch per DHCP, sofern im Heimnetz ein DHCP-Server zur Verfügung steht. Haben Sie im Heimnetz bereits einen Linux/Unix-Computer im Einsatz, können Sie die IP-Adresse auch über den *nmap*-Terminalbefehl herausfinden, der allerdings erst installiert werden muss. Dies holen Sie gegebenenfalls mit dem Kommando

```
sudo apt-get install nmap
```

nach. Anschließend scannen Sie mit dem *nmap*-Kommando das heimische Netzwerk nach verfügbaren Computern. Dafür benötigen Sie den Adressbereich des Heimnetzes (hier *192.168.123.0*), der in Ihrem Fall jedoch ganz anders lauten kann. Diesen finden Sie per Eingabe des *ifconfig*/*ipconfig*-Kommandos in ein Terminalfenster an Ihrem Computer heraus. Wenn Sie anschließend das *nmap*-Kommando

```
nmap 192.168.123.0/24
```

starten, kann es je nach Netzwerk etwas länger dauern, bis das komplette Netzwerk durchsucht ist und *nmap* dann auf die Suche gehen kann.



Bild 1.3: Mit dem *nmap*-Kommando durchsuchen Sie das heimische Netzwerk nach IP-Adressen, die auch einen geöffneten SSH-Port 22 zur Verfügung stellen.

Besser ist es, die Suche von vornherein einzuschränken und sich nur Computer anzeigen zu lassen, die auch einen geöffneten Port 22 (für SSH nötig) im Heimnetz anbieten.

```
nmap -p 22 --open -sV 192.168.123.0/24
```

Alternativ können Sie das Ergebnis in eine Datei (hier `ssh-homenet.txt`) schreiben:

```
nmap -p 22 --open -sV 192.168.123.0/24 >> ssh-homenet.txt
```

Ist nun die IP-Adresse bekannt, verbinden Sie sich mit dem Raspberry Pi – unter Windows steht Ihnen das kostenlose PuTTY-Werkzeug (<http://bit.ly/1jsQjnt>) zur Verfügung, unter Linux und Mac OS nutzen Sie das eingebaute Terminal.

Einrichten einer statischen IP-Adresse

Wie bereits erwähnt, sind die Netzwerkeinstellungen des Raspberry Pi auf DHCP konfiguriert. Soll der Raspberry Pi hingegen immer eine feste, statische IP-Adresse verwenden, tragen Sie diese in die Konfigurationsdatei `/etc/network/interfaces` ein. Dafür ändern Sie sie wie folgt:

```
sudo /etc/network/interfaces
```

und ändern die Zeile mit den Standardeinstellungen von:

```
iface eth0 inet dhcp
```

in:

```
iface eth0 inet static
    address 192.168.123.45
    network 192.168.123.0
    netmask 255.255.255.0
    broadcast 192.168.123.255
    gateway 192.168.123.199
```

Die Parameter müssen natürlich zu Ihrem eigenen Netzwerk passen. In diesem Fall verwendet das Netzwerk den Bereich `192.168.123.x`. Erst nach dem Speichern der Datei `/etc/network/interfaces` und dem notwendigen Neustart per `sudo reboot`-Kommando werden die Änderungen wirksam.

Remotedesktop für den entfernten Zugriff einrichten

Nicht jeder hat an dem Raspberry Pi einen Bildschirm, eine Maus und eine Tastatur angeschlossen. Manche Programme benötigen auf dem Raspberry Pi jedoch eine grafische Benutzeroberfläche, damit das Programm überhaupt eingerichtet und genutzt werden kann. Die Rettung für das genannte Problem ist die Installation des *Remote Desktop Protocol* (RDP), mit dem Sie nicht nur von Mac OS X und Linux, sondern auch von Windows-PCs aus direkt auf die grafische Oberfläche des Raspberry Pi zugreifen können. Voraussetzung dafür ist jedoch eine aktive grafische

Benutzeroberfläche auf dem Raspberry Pi, die Sie mit dem Systemwerkzeug `sudo raspi-config` einschalten.

```

pi@raspiBreakout: ~$ sudo -i
root@raspiBreakout:~# apt-get install arduino
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libblas3gf liblapack3gf
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  arduino-core avr-libc avrdude binutils-avr ca-certificates-java default-jre default-jre-headless
  extra-xdg-menus gcc-avr icedtea-6-jre-cacao icedtea-6-jre-jamvm icedtea-netx icedtea-netx-common java-common
  libatk-wrapper-java libatk-wrapper-java-jni libjna-java libnspr4 libnss3 libnss3-1d librtx-java
  openjdk-6-jre openjdk-6-jre-headless openjdk-6-jre-lib ttf-dejavu-extra tzdata-java
Suggested packages:
  arduino-mk avrdude-doc task-c-devel gcc-doc gcc-4.2 equivs libjna-java-doc icedtea-plugin sun-java6-fonts
  fonts-ipafont-gothic fonts-ipafont-mincho ttf-wqy-microhei ttf-wqy-zenhei ttf-indic-fonts
The following NEW packages will be installed:
  arduino arduino-core avr-libc avrdude binutils-avr ca-certificates-java default-jre default-jre-headless
  extra-xdg-menus gcc-avr icedtea-6-jre-cacao icedtea-6-jre-jamvm icedtea-netx icedtea-netx-common java-common
  libatk-wrapper-java libatk-wrapper-java-jni libjna-java libnspr4 libnss3 libnss3-1d librtx-java
  openjdk-6-jre openjdk-6-jre-headless openjdk-6-jre-lib ttf-dejavu-extra tzdata-java
0 upgraded, 27 newly installed, 0 to remove and 1 not upgraded.
Need to get 59.6 MB of archives.
After this operation, 175 MB of additional disk space will be used.
Do you want to continue [Y/n]?

```

Bild 1.4: Mit dem Systemwerkzeug `raspi-config` initiieren Sie den Start der grafischen Benutzeroberfläche über den Eintrag **Enable Boot to Desktop**.

Anschließend bestätigen Sie die Nachfrage bei *Should we boot straight to desktop?* mit Yes und starten das System neu. Nach dem Neustart installieren Sie noch das `xrdp`-Paket, das die Remotedesktopfunktionen auf dem X-Server des Raspberry Pi zur Verfügung stellt:

```

sudo -i
apt-get install xrdp

```

Nach einem kurzen Moment ist das Paket installiert und der entsprechende Dienst gestartet. Nun starten Sie auf dem Windows-Computer das Programm *Remotedesktop-Verbindung* und tragen im Verbindungsfenster die IP-Adresse des Raspberry Pi ein.

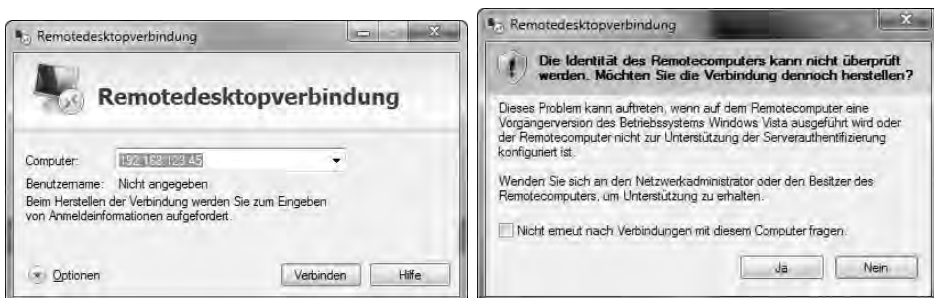


Bild 1.5: Nach dem Start des unter Windows standardmäßig verfügbaren Remotedesktop klicken Sie den Sicherheitshinweis zunächst weg.

Dann erscheint der Log-in-Bildschirm der grafischen Benutzeroberfläche des Raspberry Pi auf dem Windows-Bildschirm. Hier nutzen Sie unter *username* den Benutzer *pi* samt dem dazugehörigen Passwort, um sich einzuloggen.



Bild 1.6: Etwas Geduld bitte: Bis die Daten geladen sind und entsprechend dargestellt werden, dauert es wenige Sekunden.

Erscheint hingegen nach dem Start von *xrdp* und dem Log-in-Versuch eine Fehlermeldung mit dem Text

```
connecting to 127.0.0.1 5910  
error - problem connecting
```

ist der installierte notwendige *tightvncserver*-Dienst möglicherweise defekt. Dann deinstallieren Sie zunächst das *xrdp*- und das *tightvncserver*-Programm und installieren dann beide erneut.

```
sudo apt-get remove xrdp -y  
sudo apt-get remove tightvncserver -y  
sudo apt-get install tightvncserver -y  
sudo apt-get install xrdp -y
```

Was jetzt noch fehlt, ist die eigentliche Software – installieren Sie diese auf dem gewohnten Weg über `sudo apt-get PAKETNAME install` nach.

Remotedesktop-Alternative: xming über SSH nutzen

Wer hingegen Mac OS X oder Linux im Einsatz hat, installiert entweder die Remote-Desktop-Verbindung für dieses Betriebssystem nach oder verwendet statt des Remote Desktop Protocol (RDP) das kostenlose Xming (<http://bit.ly/1iVhjvl>), das mit dem verbreiteten VNC-Protokoll arbeitet.

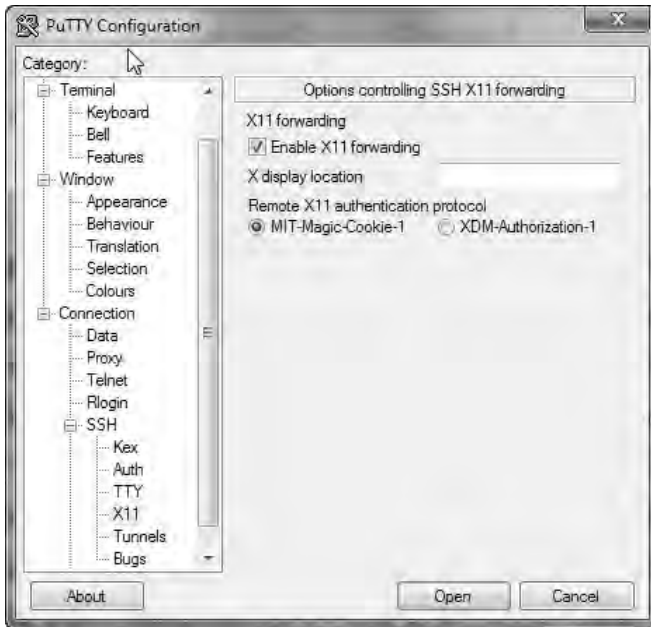


Bild 1.7: Bei dem gespeicherten PuTTY-Profil für den Raspberry Pi müssen erst die *X11 forwarding*-Einstellungen konfiguriert werden.

Bekanntlich verwendet der Raspberry Pi eine X-11-Benutzeroberfläche zur grafischen Darstellung der Fenster. Mit dem sicheren SSH-Protokoll verbinden Sie sich per Konsole mit dem Raspberry Pi und passen die SSH-Verbindungsparameter entsprechend an. Dann kann sogar per SSH mit dem Werkzeug `xming` auf einem Windows-PC auch die X-Server-Benutzeroberfläche angezeigt werden.

Prüfen Sie zunächst auf dem Raspberry Pi in der Datei `/etc/ssh/sshd_config`, ob dort X-11-Forwarding aktiviert ist – falls nicht, schalten Sie es ein. Nach der Installation des X-Servers `xming` von SourceForge (<http://bit.ly/1iViBq5>) auf dem Windows-PC können Sie nach einem kleinen Eingriff in das PuTTY-Profil der SSH-Verbindung zum Raspberry Pi die grafische Oberfläche auf den Windows-PC zaubern.

Dann laden Sie unter PuTTY das Verbindungsprofil und setzen in der Kategorie *Connection* unter *SSH* bei *X11* das Häkchen bei *Enable X11 forwarding*. Nun bauen Sie eine Verbindung über `ssh` auf, loggen sich ein und geben das Kommando `startlxde` ein, um die grafische Oberfläche zu starten. Soll das PuTTY-Konsolenfenster weiterhin zur Verfügung stehen, während die grafische Oberfläche `lxde` läuft, fügen Sie dem Kommando das `&`-Zeichen hinzu.

```
startlxde &
```

Wie auch immer: Welche Lösung Sie einsetzen, um die grafische Oberfläche des Raspberry Pi auf den PC-Desktop zu transportieren, ist letztendlich Geschmacksache. Für den Hausgebrauch können Sie auch einen Mini-LCD-Bildschirm direkt am Raspberry Pi betreiben, der gerade für kleine Statusmeldungen, Temperaturanzeigen und dergleichen ganz praktisch sein kann.

Wie von Linux gewohnt, haben Sie die Möglichkeit, auf der Shell die geläufigen Linux-Basiswerkzeuge wie `grep`, `cut` und Konsorten auf die Bildschirmausgabe anzuwenden und die Rückgabe weiter auszuwerten.

1.3 Python-Zugriff mit der RPi.GPIO-API

Python ist standardmäßig bei jedem Raspberry Pi-Image mit an Bord. Für den einfachen Zugriff auf die GPIO-Pinreihe sowie die Steuerung und die Überwachung der einzelnen Pins erleichtert eine passende Bibliothek einiges, da diese jede Menge grundsätzlicher Vorarbeiten abnimmt. Von jeher kam dafür im Python-Umfeld die Raspberry Pi-RPi.GPIO-Bibliothek zum Einsatz, die in der aktuellsten Version kostenlos auf <http://pypi.python.org/pypi/RPi.GPIO> zum Download bereitsteht, sowie gegebenenfalls `python-dev`, das grundlegende Pakete für die Python-Programmierung mitbringt.

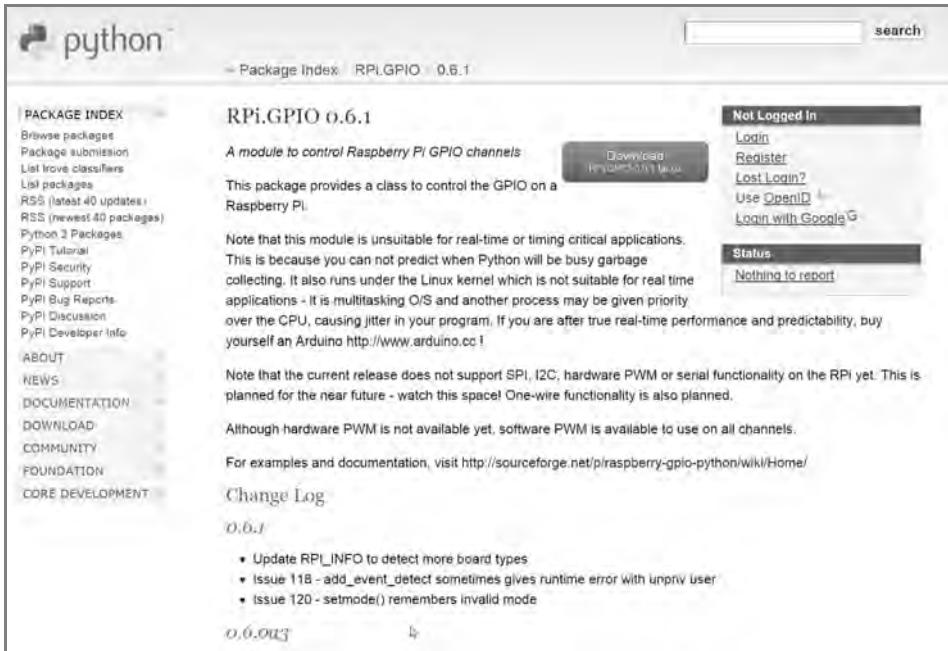


Bild 1.11: Download der Raspberry Pi-RPi.GPIO-Bibliothek unter <http://pypi.python.org/pypi/RPi.GPIO>.

In diesem Beispiel wird exemplarisch Version 0.6.1 installiert, möglicherweise steht aktuell im Internet bereits eine neuere Version zur Verfügung. In dem Fall passen Sie die nachfolgenden Kommandos einfach an diese Versionsnummer an.

Erscheint eine Zertifikatsfehlermeldung, nutzen Sie zusätzlich die Option `--no-check-certificate`, um die `tar.gz` dennoch auf den Raspberry Pi zu laden.

```
cd ~
mkdir RPi.GPIO
cd RPi.GPIO
wget --no-check-certificate
https://pypi.python.org/packages/source/R/RPi.GPIO/RPi.GPIO-0.6.1.tar.gz
tar xzf RPi.GPIO-0.6.1.tar.gz
cd RPi.GPIO-0.6.1
```

Nach dem Download entpacken Sie die `gz-tarball`-Datei ins `/home/pi`-Verzeichnis und navigieren per `cd`-Kommando in das Verzeichnis mit dem Inhalt. Anschließend starten Sie mit dem Kommando

```
sudo python setup.py install
```

die Installation von RPi.GPIO. Das Installationsskript `setup.py` sorgt für sämtliche Installationsarbeiten. Allerdings ist ein installiertes `python-dev`-Kernpaket Voraussetzung. Erscheint beispielsweise also eine Fehlermeldung mit dem Text `fatal error: Python.h: No such file or directory`, sorgt das nächste Kommando für die vorherige Grundinstallation von Python auf dem Raspberry Pi:

```
sudo apt-get install python-dev
```

```

$ sudo apt-get install python-dev
Reading package lists... Done
Building dependency tree... Done
The following packages will be installed:
  libpython2.7-dev python-dev python2.7-dev
Suggested packages:
  python2.7-doc
The following NEW packages will be installed:
  libpython2.7-dev python-dev python2.7-dev
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 14.2 MB of archives.
After this operation, 25.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get: 1 http://mirrors.ubuntu.com/mirrors.edges.ubuntu.com/ubuntu/ precise/main python2.7-dev amd64 2.7.3-2 [12,250 B]
Get: 2 http://mirrors.ubuntu.com/mirrors.edges.ubuntu.com/ubuntu/ precise/main libpython2.7-dev amd64 2.7.3-2 [17,8 MB]
Get: 3 http://mirrors.ubuntu.com/mirrors.edges.ubuntu.com/ubuntu/ precise/main python-dev amd64 2.7.3-2 [19,6 kB]
Get: 4 http://mirrors.ubuntu.com/mirrors.edges.ubuntu.com/ubuntu/ precise/main python2.7-dev amd64 2.7.3-2 [229 kB]
Fetched 14.2 MB in 10s (1,427 kB/s)
Prepared to unpack libpython2.7-dev:amd64 (2.7.3-2) ...
Prepared to unpack python-dev:amd64 (2.7.3-2) ...
Prepared to unpack python2.7-dev:amd64 (2.7.3-2) ...
Unpacking libpython2.7-dev:amd64 (2.7.3-2) ...
Unpacking python-dev:amd64 (2.7.3-2) ...
Unpacking python2.7-dev:amd64 (2.7.3-2) ...
Setting up libpython2.7-dev:amd64 (2.7.3-2) ...
Setting up python-dev:amd64 (2.7.3-2) ...
Setting up python2.7-dev:amd64 (2.7.3-2) ...
Processing triggers for libc-bin (2.15-0ubuntu2) ...

```

Bild 1.12: Installation erfolgreich: Das Nachziehen des Python-Grundpakets `python-dev` brachte auf dem Raspberry Pi letztlich die Installation der RPi.GPIO-Bibliothek zustande.

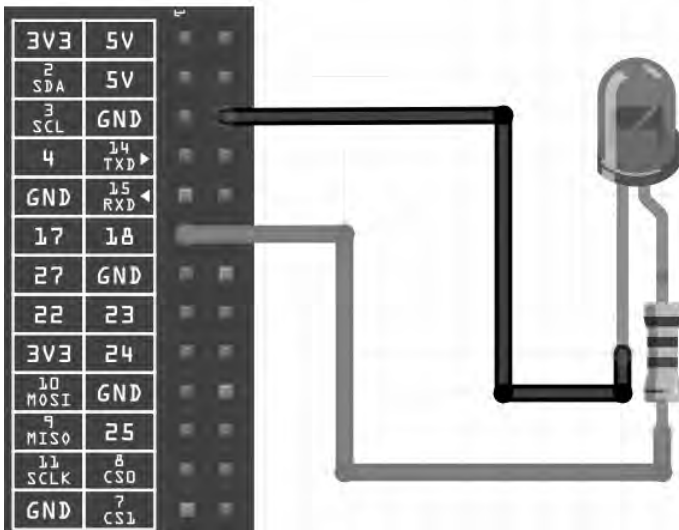
Nun können Sie die `RPi.GPIO`-Bibliothek mit Python verwenden – für den Start empfiehlt sich zunächst eine einfache Schaltung, die sich in wenigen Minuten auf einem Steckboard umsetzen lässt. Die klassische LED (den passenden Vorwiderstand nicht vergessen) wird über einen GPIO-Pin ein- und dann wieder ausgeschaltet.

LED-Praxis mit der `RPi.GPIO`-Bibliothek

Um die Funktion der `RPi.GPIO`-Bibliothek zu prüfen, reichen nur wenige Zeilen Code, um einen GPIO-Pin zu konfigurieren, zu initialisieren und zu verwenden. Mit den systemnahen Dingen müssen Sie sich nicht mehr beschäftigen, das übernimmt die `RPi.GPIO`-Bibliothek, die am Anfang des Python-Codes mit dem `import`-Befehl in Ihr Python-Programm eingebunden wird.

```
#!/usr/bin/python
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM) # GPIO Mode
GPIO_LED = 17 # GPIO17 (Ausgang) / Pin 11 / wiring Pi: 0
import time
import RPi.GPIO as GPIO
waitLED = 10
# RPi.GPIO BCM Layout verwenden
GPIO.setmode(GPIO.BCM)
# Pin 11 (GPIO 17) auf Output setzen
GPIO.setup(GPIO_LED, GPIO.OUT)
GPIO.output(GPIO_LED, GPIO.HIGH)
time.sleep(waitLED)
GPIO.output(GPIO_LED, GPIO.LOW)
```

Speichern Sie den Quellcode in eine Datei mit der Bezeichnung `rpigpioled.py` im Home-Verzeichnis (`/home/pi`) des Benutzers `pi`. Dafür nutzen Sie auf der Kommandozeile den `vi`-, Einsteiger besser den `nano`-Editor mit dem Kommando `nano rpigpioled.py`.

**Bild 1.13:**

Die Schaltung für die LED ist auf dem Steckboard schnell umgesetzt.

Natürlich können Sie in der GUI auch den grafischen Editor `gedit` verwenden – das Ergebnis ist dasselbe. Ist die Datei mit dem obigen Quellcode befüllt und auf der Speicherkarte gesichert, starten Sie das Programm mit dem Python-Interpreter:

```
sudo python rpigpioled.py
```

Damit führen Sie das obige Python-Beispiel mit der `RPi.GPIO`-Bibliothek auf dem Terminal aus – die auf dem Steckboard angeschlossene LED sollte nun leuchten und nach einem kurzen Augenblick wieder gelöscht werden.

PIR-Praxis mit der RPi.GPIO-Bibliothek

Analog zum obigen Shell-Skript mit der LED lässt sich der PIR-Sensor auch mit Python mithilfe der `RPi.GPIO`-Bibliothek unproblematisch einsetzen. Die Schaltung ist die gleiche – der Datenpin bleibt an GPIO7 gesteckt.

Ähnlich wie eine LED lässt sich auch ein Sensor mit der `RPi.GPIO`-Bibliothek verwenden. Da der Sensor vom Raspberry Pi überwacht werden soll, muss der GPIO-Pin als Eingang (vom Sensor kommend) definiert werden, was in der nachstehenden Zeile erledigt wird:

```
GPIO.setup(piri, GPIO.IN)
```

Im nächsten Schritt läuft das Python-Programm in eine »Endlosschleife«, die mit der Tastenkombination `[Strg]+[C]` unterbrochen werden kann. In dieser Schleife prüft das `if`-Konstrukt, ob am `piri`-Datenpin ein Signal eingetroffen ist oder nicht.

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

```
GPIO.setmode(GPIO.BCM)
piri = 7
waitpiri = 10

GPIO.setup(piri, GPIO.IN)

try:
    print "PIRI Modul im Einsatz (Beenden mit: CTRL-C)"
    time.sleep(waitpiri)
    print "Ready"
    while True:
        if GPIO.input(piri):
            print "Bewegung"
            time.sleep(1)
except KeyboardInterrupt:
    print "Exit!"
    GPIO.cleanup()
```

Wird eine Bewegung erkannt, wird auf dem Terminal der schlichte Text Bewegung ausgegeben – als Standardausgabe erscheint anderenfalls alle zehn Sekunden der Ready-Hinweis.



Bild 1.14: PIR-Sensorüberwachung mit der RPi .GPIO-Bibliothek.

Nicht nur über Python, auch direkt auf der Shell lassen sich solche Prüfungen und kleinere Schaltungen einfach überwachen und steuern. Ein sehr beliebtes Werkzeug in der Praxis mit dem Raspberry Pi ist die WiringPi-API, die verschiedene und oftmals notwendige Dinge im Umgang mit den GPIO-Anschlüssen auf der Shell in einfachen Kommandos zusammenfasst.

1.4 WiringPi-API: schnell auf der Shell

Wer jenseits der Programmiersprachen auf der Shell mal schnell einen oder mehrere GPIO-Pins konfigurieren und verwenden möchte, musste sich jenseits des `raspi-gpio`-Kommandos bisher umständlich mit den Linux-Basiswerkzeugen behelfen.

2

Hören, sehen und fühlen mit Sensoren

Der Raspberry Pi bietet dank der zahlreichen GPIO-Anschlüsse und den beiden Spannungspins 3.3V und 5V bekanntlich auch die Möglichkeit, Aktoren und Sensoren aus der Arduino-Ecke oder aus dem Fachhandel für Kommunikations- und Steuerungssysteme zu nutzen. Die meisten Sensormodule und Aktoren für den Raspberry Pi finden Sie für kleines Geld auf den einschlägigen Auktions- und Kaufhausseiten im Netz. Meist sitzen die Lieferanten solcher Arduino- und Raspberry-Pi-tauglichen Billigmodule in China oder Hongkong, was in Sachen Lieferzeit bis zu drei Wochen und manchmal länger bedeuten kann. Als Dankeschön für die Wartezeit winken günstige Preise – soll es schneller gehen, bieten hiesige Anbieter die gleichen Module mit einem Aufschlag an und sind damit im Vergleich deutlich teurer.

Sensor	Bemerkung/Modellbezeichnung	Bezugsquelle(n)
Abstands-sensor	Abstandsmessung mit HC-SR04 (Ultraschallsensor) oder QTR-1A (Pololu Abstandssensor)	Diverse China-/Hongkong-Händler

Sensor	Bemerkung/Modellbezeichnung	Bezugsquelle(n)
Akustiksensord	Akustiksensord	Diverse China-/Hongkong-Händler
Barometer-sensord	BMP085, Suchmaschine: SKU: SKU039532	Diverse China-/Hongkong-Händler
Bewegungs-sensord	PIR	Diverse China-/Hongkong-Händler
Feuchtig-keitssensord	Unterschiedliche auf Basis des LM393-IC	Diverse China-/Hongkong-Händler
Gassensord	MQ-2 Gas/Luftsensord	Diverse China-/Hongkong-Händler, Fachhandel
Touch-/Drucksensord	MPR121 Capacitive Touch Keypad	Diverse China-/Hongkong-Händler, Fachhandel
Temperatur-sensord	LM35 (analog) DS18B20 (digital)	www.farnell.com , Fachhandel
Gyroskop (Kreiselinstrument)	MPU-6050 (Gyro, Beschleunigungssensord)	Fachhandel
Gyroskop (KreiselInstrument)	AltIMU-10 (Gyro, Beschleunigungssensord, Compass und Altimeter (L3GD20, LSM303DLHC, LPS331AP Carrier)	www.pololu.com/catalog/product/1269 , Fachhandel
Farbsensord	TCS34725	www.adafruit.com/products/1334
Infrarot-sensord	QTR-8RC-Reflektorsensord von Pololu	www.pololu.com/product/961
Kamerasensord	Raspberry-Pi-Kamera oder beliebige USB-Webcam	www.farnell.com , Fachhandel
Lichtsensord	LDR	www.farnell.com , Fachhandel
Lichtsensord	APDS-9002, 1142	Hersteller, Fachhandel
Motortreiber	Pololu A4988	www.pololu.com/product/1183
Tilt-/Neigungssensord	SW-520D	China-/Hongkong-Händler

Auf den nachfolgenden Seiten werden die wichtigsten praxisnahen Sensoren im Raspberry-Pi-Umfeld und ihre Inbetriebnahme beschrieben, damit Sie diese direkt mit dem Steckboard oder einer Platine bzw. direkt mit der GPIO-Leiste des Raspberry Pi koppeln und nutzen können.

2.1 Aktive vs. passive Sensoren

Grundsätzlich ist in den Datenblättern auch von analogen und digitalen Sensoren die Rede – hier ist weniger der Sensor selbst als vielmehr die Ausgabe gemeint, die entweder als Analog- oder als Digitalwert übermittelt werden kann. Analogwerte sind typischerweise Spannung bzw. Spannungsunterschiede, mit denen die Zustandsänderung an einem Sensor gemessen wird. Digitalsensoren liefern hingegen Nullen oder Einsen – stellt beispielsweise ein Abstandssensor oder Bewegungsmelder ein Signal fest, wird der Wert **1** gemeldet, ansonsten bleibt er auf **0**.

Bei vielen Sensoren kommt das LM393-IC-Modul zum Einsatz. Der LM393 ist eine Komparatorschaltung, um das analoge Messsignal – beispielsweise von einem Sensor – zu digitalisieren, also in einen Wert **0** oder **1** zu überführen. Paart man dies mit einer Bedingung – beispielsweise wird nur dann der Wert **1** geschrieben, wenn der Sensor auch eine Bewegung erkannt hat oder die Temperatur über dem festgelegten Schwellenwert liegt –, haben Sie einen digitalen Sensor im Einsatz.

Analoge Sensoren auf Basis des LM393-IC sind im Heimelektronikbereich häufig anzutreffen: Eine typische Anwendung ist ein Akustiksensord, der auf seiner Platine mit einem Mikrofon zusammenarbeitet. Ist dieses aktiviert, informiert das digitale »Messergebnis« anhand des Werts **0** oder **1**, ob ein Geräusch erkannt wurde oder nicht. In der Regel ist in Kombination mit dem LM393-IC auch ein Potenziometer verbaut, um die Empfindlichkeit des angeschlossenen Sensors – also die Eingangsspannung in das LM393 – zu regeln.

Egal ob Sie jetzt einen Wasserstandssensor, einen Bewegungsmelder, einen Farb-/Fototransistor oder ähnliche Sensoren verwenden – das grundsätzliche Prinzip ist immer dasselbe.

In der Literatur wird manchmal auch zwischen aktiven und passiven Sensoren unterschieden: Einfach ausgedrückt, sind Sensoren immer genau dann »aktiv«, wenn sie durch Anlegen einer Versorgungsspannung eingeschaltet werden, interne verstärkende oder signalformende Bauelemente (beispielsweise Potenziometer etc.) verwenden und erst dann auch ein analoger oder digitaler Messwert am Ausgang generiert wird.

Ein Sensor ist hingegen »passiv«, wenn er ohne zusätzliche Versorgungsspannung eine Zustandsänderung feststellen kann. Zum Auslesen des Messwerts eines passiven Sensors ist wieder eine Versorgungsspannung notwendig. Im weiteren Verlauf dieses Buchs wird nicht mehr explizit auf den Unterschied aktiv/passiv eingegangen, da ohnehin für die Verarbeitung der Messwerte eine Versorgungsspannung anliegen muss.

2.2 Licht- und Farbsensoren im Raspberry-Pi-Einsatz

Gegenüber Licht- und Farbsensoren ist das menschliche Auge in Sachen Lichtintensität zwar weniger empfindlich, doch es nimmt ein breiteres Spektrum wahr. Das menschliche Auge passt sich vom Sternenlicht bei 50 Mikrolux bis hin zu 100.000 Lux bei starkem Sonnenschein automatisch an. Günstige Licht-/Farbsensoren arbeiten je nach Bauweise im Spektrum 1 bis 1000 Lux. Je nach verwendetem Sensor beruht die Messung der Lichtstärke im einfachsten Fall auf folgender Vorgehensweise: Zunächst wird die Beleuchtung/LED eingeschaltet, und eine Messung wird durchgeführt. Nach dem Ausschalten der Beleuchtung/LED wird nochmals eine Messung durchgeführt, die anschließend vom ersten Messwert abgezogen wird. Die berechnete Differenz stellt somit den Rohwert des Sensors dar. Kombinieren Sie diese Vorgehensweise mit einem Zeitstempel und messen Sie die Unterschiede in einer festen Zeiteinheit. Durch dieses feste Intervall stellt das Ergebnis dann eine bestimmte Frequenz dar, die Sie elektronisch auswerten können.

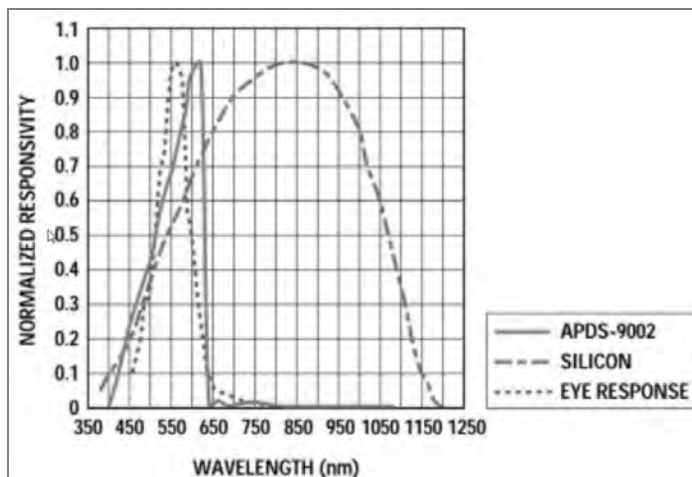


Bild 2.1: Infrarot bzw. Lichtwellenlänge in nm vs. Lichterkennung des menschlichen Auges. (Abbildung: Datenblatt APDS-9002 light sensor)

LDR-Lichtsensorschaltung auf dem Steckboard

Die einfachste Messung – der einfachste Sensor – zum Feststellen einer Helligkeitsänderung besteht im Einsatz eines lichtabhängigen Widerstands (LDR, *Light Dependent Resistor*). Grundsätzlich wandeln Lichtsensoren Licht in eine Spannung, Strom oder eine Frequenzänderung um, die anschließend über den Raspberry Pi weiterverarbeitet werden kann. Für das grundsätzliche Verständnis eines solchen LDR reicht hier eine einfache Steckboardschaltung mit einem Elko (*Elektrolyt-Kondensator*) und LDR-Lichtsensor aus.

LDR-Licht-sensor	1 uF Elko	2,2-kΩ-Wi-derstand	Bemerkung	Raspberry-Pi-Bezeichnung	Raspberr-y-Pi-Pin	Wiring Pi
A1	–	–	3.3V	3.3V	1	–
	C–	–	Masse	GND	6	–
A2	C+	R1	–	–	–	–
–	–	R1	GPIO-18	GPIO-18	12	1

Zunächst verbinden Sie den Masse-Anschluss und die 3,3-V-Spannungsversorgung des Raspberry Pi mit dem Steckboard. Die Masse-Leitung wird mit dem negativen C-Anschluss des Elkos gekoppelt. Das andere Beinchen des Elkos (C+) wird mit einem Anschluss des LDR-Widerstands und dem GPIO-Anschluss verbunden. Der zweite Anschluss des LDR-Widerstands wird anschließend an den 2,2-kΩ-Widerstand und von dort aus an die 3,3-V-Spannungsversorgung geführt.

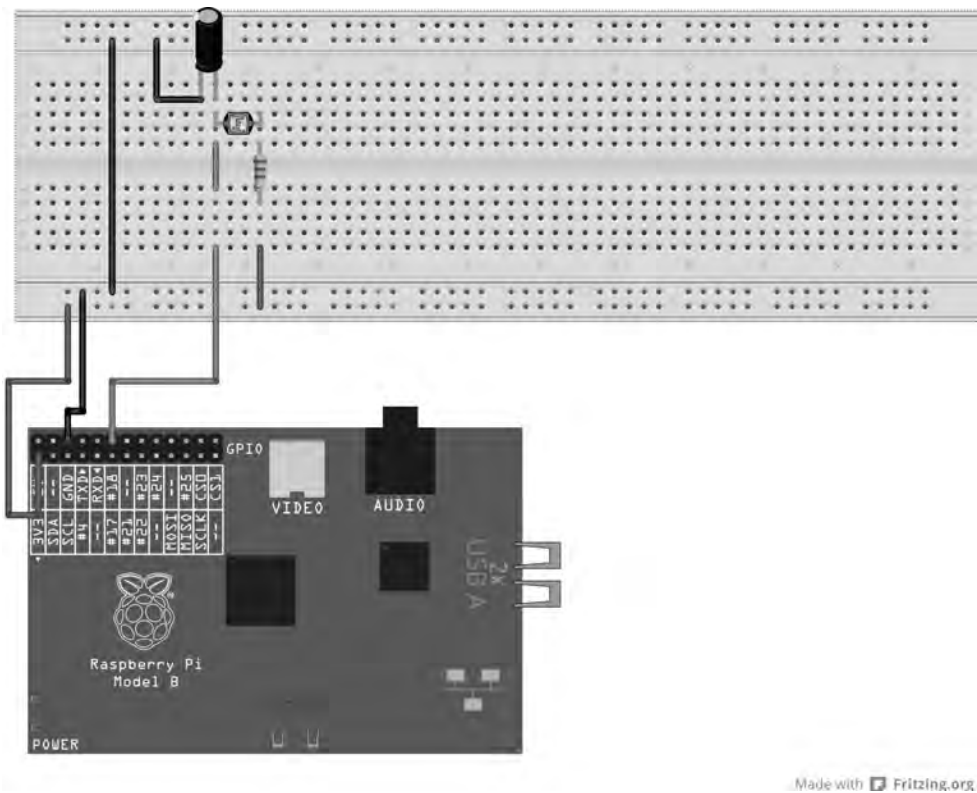


Bild 2.2: Beim Anschluss des 1-uF-Elektrolytkondensators achten Sie darauf, dass das mit einem Minuszeichen bezeichnete Beinchen mit dem Masse-Anschluss (GND) auf dem Steckboard verbunden wird.

Grundsätzlich beruht der Kniff in der Schaltung darauf, festzustellen, wie lange die vorliegende Schaltung benötigt wird, bis am Anschluss GPIO18 ein HIGH-Signalpegel anliegt. Der Schwellenwert dafür liegt bei etwas über 1,9 V, was ca. 58 % der Ausgangsspannung von 3,3 V entspricht. Verwenden Sie für die Berechnung der benötigten Zeit diese Formel:

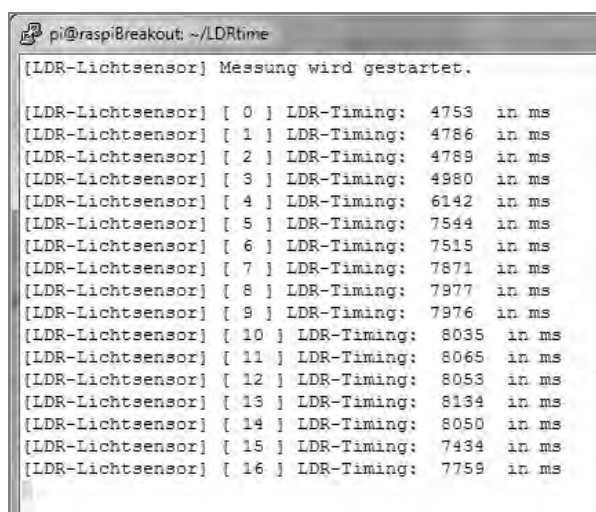
$$t = R * C$$

Hier entspricht R dem Widerstandswert in Ω (Ohm) und C der verwendeten Kapazität des Elkos in Farad – die Messeinheit des Kondensators. Die Zeit ist also analog zu dem Messwert, den die Schaltung benötigt, wenn der GPIO-Eingang vom LOW- in den HIGH-Zustand übergeht. Im nächsten Schritt nehmen Sie die Steckboardschaltung mit dem Raspberry Pi und einem kleinen Python-Programm in Betrieb.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# -----
# Sensoren am Raspberry Pi
# E.F.Engelhardt, Franzis Verlag, 2016
# -----
# ldr-lichtsensor.py
# Pfad: /LDRtime
#
import RPi.GPIO as GPIO
import os
import time
LDR_GPIO = 18
# -----
def init():
    GPIO.setmode(GPIO.BCM)
    os.system('clear')
    print "[LDR-Lichtsensord] Messung wird gestartet.\n"
# -----
def main():
    init()
    i = 0
    try:
        while True:
            # LDR-Timing mit GPIO-Pin 18
            print "[LDR-Lichtsensord] [" + str(i) + "] LDR-Timing: " + str(LDRtime(LDR_GPIO)) + "
in ms"
            i = i + 1
    except KeyboardInterrupt:
        # Strg-C gedrueckt
        # Reset GPIO
        print "[LDR-Lichtsensord] Messung abgebrochen."
        GPIO.cleanup() # Aufräumen !
```

```
# -----
def LDRtime(GPIOpin):
    ldrtime = 0
    GPIO.setup(LDR_GPIO, GPIO.OUT)
    GPIO.output(GPIOpin, GPIO.LOW) # Elko entladen
    time.sleep(0.1)
    GPIO.setup(GPIOpin, GPIO.IN) # Strom wird eingeschaltet
    # -> Messung: Je Durchlauf ca. 1ms .> proportional zu Widerstand des LDR
    while (GPIO.input(GPIOpin) == GPIO.LOW):
        ldrtime += 1
    return ldrtime
# -----
if __name__ == '__main__':
    main()
sys.exit(0)
# -----
```

Für eine einfache und schnelle Schaltung, um beispielsweise Helligkeitsunterschiede festzustellen, reicht obige Beispielschaltung sowie der dargestellte Python-Code völlig aus.



```
pi@raspiBreakout: ~/LDRtime
[LDR-Lichtsensordr] Messung wird gestartet.

[LDR-Lichtsensordr] [ 0 ] LDR-Timing: 4753 in ms
[LDR-Lichtsensordr] [ 1 ] LDR-Timing: 4786 in ms
[LDR-Lichtsensordr] [ 2 ] LDR-Timing: 4789 in ms
[LDR-Lichtsensordr] [ 3 ] LDR-Timing: 4980 in ms
[LDR-Lichtsensordr] [ 4 ] LDR-Timing: 6142 in ms
[LDR-Lichtsensordr] [ 5 ] LDR-Timing: 7544 in ms
[LDR-Lichtsensordr] [ 6 ] LDR-Timing: 7515 in ms
[LDR-Lichtsensordr] [ 7 ] LDR-Timing: 7671 in ms
[LDR-Lichtsensordr] [ 8 ] LDR-Timing: 7977 in ms
[LDR-Lichtsensordr] [ 9 ] LDR-Timing: 7976 in ms
[LDR-Lichtsensordr] [ 10 ] LDR-Timing: 8035 in ms
[LDR-Lichtsensordr] [ 11 ] LDR-Timing: 8065 in ms
[LDR-Lichtsensordr] [ 12 ] LDR-Timing: 8053 in ms
[LDR-Lichtsensordr] [ 13 ] LDR-Timing: 8134 in ms
[LDR-Lichtsensordr] [ 14 ] LDR-Timing: 8050 in ms
[LDR-Lichtsensordr] [ 15 ] LDR-Timing: 7434 in ms
[LDR-Lichtsensordr] [ 16 ] LDR-Timing: 7759 in ms
```

Bild 2.3: LDR-Sensor im Einsatz: Je dunkler die Umgebung, desto größer ist der angezeigte Timing-Wert.

Soll der Sensor hingegen Farben oder die Farbintensität messen, nutzen Sie eigens für diesen Zweck gebaute Sensoren, wie beispielsweise den nachfolgend vorgestellten TCS34725-Farbsensor.

TCS34725-Farbsensor installieren und einsetzen

Grundsätzlich erfassen Farbsensoren die Farbe einer Oberfläche. Je nach Bauweise des Sensors wirft dieser Licht (Rot, Blau, Grün) auf das Objekt und berechnet aus der reflektierten Strahlung die Farbwertanteile. Sind Referenzfarbwerte gespeichert, können diese mit der Messung verglichen werden, und entsprechend wird dann das Ergebnis der Messung ausgegeben. In diesem Projekt kommt der für den Hausgebrauch völlig ausreichende TCS34725-Farbsensor zum Einsatz, der sich dank der verfügbaren I²C-Schnittstelle unkompliziert mit dem Raspberry Pi verbinden lässt.

Zunächst ist das Studieren des Datenblatts des TCS34725-Farbsensors (erhältlich unter www.ams.com/eng/content/download/319364/1117183/287875) notwendig, um die Arbeitsweise des Sensors in etwa nachzuvollziehen und die Belegung der Pins für den Anschluss an den Raspberry Pi herauszufinden. Die sieben verfügbaren Pins des TCS34725-Farbsensors lassen sich wie in der nachstehenden Tabelle zusammengefasst mit dem Raspberry Pi koppeln – optional ist der Anschluss des LED-Pins, der in diesem Projekt nicht beschaltet wurde.

Dafür lässt sich Pin 1 auf dem TAOS TCS34725 – dem LED-Anschluss des Sensors – mit einem GPIO-Pin des Raspberry Pi verbinden. In diesem Fall wird der GPIO-Pin als Ausgang genutzt, damit die LED auf dem Sensor vom Raspberry Pi geschaltet werden kann.

TAOS TCS34725- Pin	TAOS TCS34725	Bemerkung	Pi 1 (Rev. 1)	Pi 1 (Rev. 2), Pi 2, Pi 3, Zero	Rasp- berry- Pi-Pin	Wiring Pi
1	LED		–	–	–	–
2	INT	Inter- rupt	–	–	–	–
3	SDA		GPIO 0 (SDA)	GPIO 2 (SDA)	3	8
4	SCL		GPIO 1 (SCL)	GPIO 3 (SCL)	5	9
5	V _{DD}	Spannung	3.3V	3.3V	1	–
6	GND	Masse	Masse	Masse	6	–
7	VIN		–	–	–	–

Nach dem Verkabeln der vier Anschlüsse und dem Einschalten des Raspberry Pi leuchtet die auf der Platine vorhandene LED auf. Egal ob Sie nun die LED steuern möchten oder nicht – Sie legen unabhängig davon ein Projektverzeichnis (hier `\color_tcs34725`) sowie die Programmdatei `color_tcs34725-step1.py` an, um damit auf die Messwerte des Sensors bzw. die Registerinhalte zuzugreifen und diese auszugeben.

```
mkdir color_tcs34725
cd color_tcs34725
nano color_tcs34725-step1.py
```

Die Messwerte der Grundfarben Rot, Grün und Blau werden wie auch die Klarlichtdaten jeweils als 16-Bit-Wert in den dazugehörigen Registern des Sensors abgelegt. Dafür ist über die I²C-Schnittstelle ein Zwei-Byte-Lesezugriff gepaart mit dem Setzen des Protokollbits im Befehlsregister notwendig, damit die beiden Bytes (Lower- und Upper-Byteanteil) korrekt ausgelesen und einander zugeordnet werden können. Damit steht für jede Farbe ein Low-Byte und High-Byte und jeweils ein Register zur Verfügung, wie im Datenblatt auf Seite 13 beschrieben. Im nachstehend dargestellten Quellcode sind die Register bzw. die Registerbezeichnung des Datenblatts als Konstanten definiert, was für eine einfachere Zuordnung der Register und ein besseres Verständnis des Codes sorgt.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# -----
# Sensoren am Raspberry Pi
# E.F.Engelhardt, Franzis Verlag, 2016
# -----
# color_tcs34725-step1.py in der Schleife
# Pfad: /color_tcs34725
import time
import smbus
import os
# -----
# I2C Adresse vorher pruefen . hier: 0x29
I2C_TCS34725_ADDRESS = 0x29
i2cbus = smbus.SMBus(1) # Raspberry Pi Revision 2 !!
#
# Definitionen aus Datenblatt
# http://www.ams.com/eng/content/download/319364/1117183/287875
# data sheet page 13
TAOS_COMMAND_BIT = 0x80
TAOS_ENABLE = 0x00 # Enables states and interrupts
TAOS_ETIME = 0x01 # A time 0xFF
TAOS_WTIME = 0x03 # W time 0xFF
TAOS_AILT = 0x04 # interrupt low threshold
TAOS_AILTH = 0x05 # interrupt low threshold
TAOS_AIHTL = 0x06 # interrupt high threshold
TAOS_AIHTH = 0x07 # interrupt high threshold
TAOS_PERS = 0x0C # persistence filter 0x00
TAOS_CONFIG = 0x0D # 0x00
TAOS_CONTROL = 0x0F # 0x00
TAOS_VERSION = 0x12 # ID -> 0x44 = TCS34721/TCS34725, 0x4D =
TCS34723/TCS34727
```



```

TAOS_STATUS = 0x13 # status 0x00
TAOS_CDATAL = 0x14 # data low byte clear data
TAOS_CDATAH = 0x15 # data high byte
TAOS_RDATAL = 0x16 # data low byte red data
TAOS_RDATAH = 0x17 # data high byte
TAOS_GDATAL = 0x18 # data low byte green data
TAOS_GDATAH = 0x19 # data high byte
TAOS_BDATAL = 0x1A # data low byte blue data
TAOS_BDATAH = 0x1B # data high byte
# data sheet page 15 !
TAOS_REGISTER_ADDRESS = 0x80
TAOS_POWER_ON = 0x01
TOS_RGB_ENABLE = 0x02 # RGBC enable. This bit activates the two-channel ADC.
                        # Writing a 1 (binaer) activates the RGBC. Writing a 0
disables the RGBC
TOS_WEN = 0x08 # wait enable 1000 0x00 disable
#
# -----
def main ():
    # Register TAOS_VERSION 0x12 has device version
    # Register addresses must be OR'ed with 0x80 (TAOS_COMMAND_BIT)
    i2cbus.write_byte(I2C_TCS34725_ADDRESS, TAOS_COMMAND_BIT | TAOS_VERSION) #
ID
    ver = i2cbus.read_byte(I2C_TCS34725_ADDRESS)
    # version # should be 0x44
    os.system('clear')
    if ver == 0x44: # = TCS34725
        print "[TCS34725] Farbsensor angeschlossen\n"
        i2cbus.write_byte(I2C_TCS34725_ADDRESS, TAOS_COMMAND_BIT | TAOS_ENABLE) #
0x00 = ENABLE register
        i2cbus.write_byte(I2C_TCS34725_ADDRESS, TAOS_ATIME | TOS_RGB_ENABLE) #
0x01 = Power on, 0x02 RGB sensors enabled
        i2cbus.write_byte(I2C_TCS34725_ADDRESS, TAOS_COMMAND_BIT | TAOS_CDATAL) #
Reading results start register 14, LSB then MSB
        while True:
            # read values of register
            data = i2cbus.read_i2c_block_data(I2C_TCS34725_ADDRESS, 0)
            clear = data[1] << 8 | data[0]
            red = data[3] << 8 | data[2]
            green = data[5] << 8 | data[4]
            blue = data[7] << 8 | data[6]
            #ohne Zuordnung: crgb = "RAW: Clear: %s, Red: %s, Green: %s, Blue: %s
\n" % (clear, red, green, blue)
#-----
            if((red>blue) and (red>green)):
                #print "[TCS34725] Farbe: Rot erkannt."
                col= "[TCS34725] Farbe: Rot erkannt."

```

```
elif((green>blue) and (green>red)):
    #print "[TCS34725] Farbe: Gruen erkannt."
    col="[TCS34725] Farbe: Gruen erkannt."
elif((blue>red) and (blue>green)):
    #print "[TCS34725] Farbe: Blau erkannt."
    col="[TCS34725] Farbe: Blau erkannt."
else:
    #print "[TCS34725] Farbe konnte nicht spezifiziert werden."
    col="[TCS34725] Farbe konnte nicht spezifiziert werden."
#-----
    crgb = "RAW: Clear: %s, Red: %s, Green: %s, Blue: %s | %s\n" % (clear,
red, green, blue, col)
    print crgb
    time.sleep(1)
else:
    print "[TCS34725] Kein TCS34725-Geraet gefunden!\n"
    return

if __name__ == '__main__':
    main()
sys.exit(0)
# -----
```

Die Farbumterscheidung bzw. die Farbzuzuordnung im dargestellten `if`-Konstrukt erfolgt einfach durch den Größenvergleich der Registerwerte. Damit kann das Programm nun eine Aussage darüber treffen, ob der Rot-, Grün- oder Blauanteil überwiegt. Zur Laufzeit wirft das dargestellte Programm die Rohwerte der Farbregister aus.

Stichwortverzeichnis

1

1-Wire-Sensoren 149

A

Abstandssensor,
 Schaltungsaufbau 159
Advanced Options 13
Aktive Sensoren 123
Aktoren 121
Akustiksensord 210
 Schaltungsaufbau 211
 Shell-Skript 212
AltIMU-10 238
Analog-digital-Wandler
 nachrüsten 68
Analoge Lichtsensormessung
 144
APDS-9002-Lichtsensord 136
Arduino 92, 96
 Arduino Shields 104
 Arduino-GUI, GertDuino 116
 Arduino-IDE 92, 99, 113, 114
 Arduino-Pin 96
 arduPi 106
 arduPi-Library 106
 ATmega 90
 atmega.sh 115
 ATmega328P 92, 113
 ATmega-IC, Arduino-IDE 94
 AVR 90
 avrdude 93
 avrsetup 95, 115

B

Benutzeroberfläche Ixde 12
Betriebssystem auffrischen 183
Bewegungsmelder 42, 180
 Shell-Skript 41
Bewegungsmelder, Shell-Skript
 180, 212
Bewegungssensord 176
bmp085.py 217

C

CAN 104
cd 37
checkbmp085-step1.py 217
checkbmp085-step2.py 222

chmod 93
CIE 132
color_tcs34725-step1.py 128
config.txt 189
CPP-Datei 107
CSI-Anschluss 183

D

DHCP 12
distance.py 161
Drucksensord 206

E

Embedded Pi 104, 105
 LED-Schaltung 107
 Motorsteuerung 109
Embedded-Pi-Board 9, 12
Enable Camera 186
Erweiterungsboards 9
Erweiterungsplatinen 88

F

Farbsensoren 124, 128
Farbtemperatur 132
Farbzuordnung 131
Feuchtesensord 241
Feuchtesensord, Python-Routine
 242
Firmware aktualisieren 183
Fotografieren 187
Funktion, tempsensord.py 156

G

Gassensord 250
Gert van Loo 90
Gertboard 9, 12, 88, 90
 GertDuino 111
 Motorsteuerung 102
GertDuino 111
GertDuino 12
GertDuino, Inbetriebnahme 111
GertDuino-Board 9
Gesichtserkennung 193
getAnalogData 74
git pull origin 38
GitHub 190
GIT-Versionsverwaltung 37
Gordon Henderson 92

GPIO-Bibliothek 32
GPIO-Pins 10
GPIO-Porterweiterung 78
GPIO-Schnittstelle 9
GPU 186
Gyrometer 227
Gyrometer-Experimente 227
Gyrosensoren 229
Gyroskop 51
Gyroskop mit Druckmesser 236

H

H.264 188
halt 115
HD44780 58
Höhenbestimmung 215
Höhenmessung_Python-Skript
 217
Hostname, raspberrypi 14
Hygrometer 241

I

i2c_lib.py 53
I²C-Betrieb 52
I²C-Bibliothek 53
I²C-Bus 221
i2cdetect 81
i2cget 81
I²C-Platine 51
i2cset 81
IMU 229
Inertialeinheit 229
Infrarotabstandssensord 164
Infrarotmodul 176
Infrarotsensord 195
IP-Adresse 12
 herausfinden 14
 statische 15
ipconfig 14

K

Kamerasensord 181, 182
Kommandozeile 187
Kommandozeilenadministration
 13
Konsole 42, 180

L

Lagesensor 222
 LCD-Bildschirm-Projekt 53
 LCD-I²C-Adapter 60
 LDR-Lichtsensorschaltung 124
 LED abschalten 189
 LED-Schaltung 107
 Lichtsensoren 124
 Lichtsensoren, APDS-9002 136
 LM35D 146
 LM393-IC-Modul 123
 lm393-step1.py 164
 lm393-step2.py 166
 Luftdruckmessung 215
 Luftdrucksensor 215
 Lux 136
 lxd 12

M

make 37
 MCP23017, I²C-Bus 85
 MCP23017-IC 82
 mcp23017-step1.py 85, 86
 MCP3002 89
 MCP3008-IC 69
 mcp3008-step1.py 74
 mcp3008-step4.py 146
 MCP4801 89
 messung-sharp-step1.py 176
 minicom 118
 MiniIMU-9 v2 238
 mjkdz.py 65, 67
 MOTA 102
 MOTB 102
 motor.cpp 109
 Motorcontroller 89, 103
 MPU-6050, Inbetriebnahme 230
 MPU-6050, Sensorexperimente 232
 Myra Van Inwegen 90

N

Neigungssensor 222
 Neigungssensor, Schaltung 223
 nmap 14

O

OpenCV-Bibliothek 192

P

Passive Sensoren 123
 PCF8574 60
 picam-Modul 190
 piri.sh 180
 PIR-Modul 38, 177
 pitouch-step01.py 208
 Pololu AltIMU-10 236
 Pololu-QTR-8RC-Sensorarray 195
 Potenziometer 58

print 59
 Pull-up-Schaltung 223
 PuTTY-Profil 18
 PWM 90
 py-spidev-Modul installieren 139
 Python, Sensorexperimente 232

Q

QTR-8RC-Sensor 197
 qtr8-step01.py 205
 Quadrocopter 215

R

Raspberry Pi
 Aufbau 10
 entrümpeln 11
 Erweiterungsplatinen 88
 im Heimnetz 14
 update 11
 upgrade 11
 Raspberry-Pi-Kamera 181, 182
 Programmierung 189
 Raspbian 186
 raspi-config 13, 186
 raspistill 187, 190
 raspivid 187, 190
 Rauchsensor 250
 RDP 15, 99
 reboot 11, 12
 Remote Desktop Protocol 15, 17
 Remotedesktop, Arduino-
 Zugriff 98
 Remotedesktopverbindung 16
 Repository-Verwaltung 190

S

Schnittstelle, zentrale 9
 SD-Karte, Platz schaffen 11
 Sensoren 121
 Sensoren, analoge 141
 sensor-face-detect.py 193
 sensorobo_Example_
 ReadRealData.py 240
 setup.py 33
 Sharp-Abstandssensor 169
 Sharp-Abstandssensor,
 Messwertbestimmung 171
 Skript
 bmp085.py 217
 checkbmp085-step1.py 217
 checkbmp085-step2.py 222
 color_tcs34725-step1.py 128
 i2c_lib.py 53
 lm393-step1.py 164
 lm393-step2.py 166
 mcp23017-step1.py 85, 86
 mcp3008-step1.py 74
 mcp3008-step4.py 146
 messung-sharp-step1.py 176

mjkdz.py 67
 motor.cpp 109
 noise-step1.sh 212
 piri.sh 41, 180
 pitouch-step01.py 208
 qtr8-step01.py 205
 sensor-face-detect.py 193
 sensorobo_Example_
 ReadRealData.py 240
 setup.py 33
 text2lcd.py 55, 58, 67
 sleep-Funktion 87
 SourceForge 190
 Spannungsteiler 245
 SPI-Schnittstelle aktivieren 137
 SSH-Zugriff 12, 13
 Statische IP-Adresse 15
 Steckboard 9
 Steckboards 9
 Stromsensorschaltung 247
 Stromstärke bestimmen 250
 Stromstärkemessung 244
 sudo raspi-config 16

T

Temperaturmessung 145
 DS18B20-Sensor 149
 mit Python 156
 Steckboard 149
 Temperatursensor 152
 text2lcd.py 55, 58, 67
 tigntvncserver-Dienst 17
 Touchsensor 206
 Türspion 189

U

ULN2803A 89
 Ultraschallsensor 158
 Ultrasonic-Sensor 166
 uname 11, 185
 update 11
 upgrade 11

V

Versionsstand 185
 VNC 17

W

wget 37

X

X11 18
 XMing 17
 xrdp 17
 xrdp-Paket 16

Z

Zentrale Schnittstelle 9



E. F. ENGELHARDT

SENSOREN AM RASPBERRY PI

Der Raspberry Pi erfasst alles, analog oder digital: Temperatur, Abstand, Infrarotlicht, Bilder, Bewegung, Stromstärke, Gas, Neigung und mehr. 25 Sensoren, und Sie haben Ihre Umgebung im Griff.

Wie waren die ersten Wochen mit Ihrem Raspberry Pi? Mit Sicherheit hatten Sie viel Freude, aber auch hin und wieder Frust. Warum Frust? Weil Linux und Elektronik doch ihre Tücken haben und Sie sicherlich mehr wollen, als ein paar LEDs zum Blinken zu bringen. Es wäre doch toll, wenn Ihr Raspberry auf die Umgebung reagieren könnte, um z. B. den Wecker früher klingeln zu lassen, falls Schnee geschippt werden muss.

Ob Temperatur, Licht, Bewegung oder Schall – so gut wie jeder Einfluss aus der Umgebung lässt sich mit einem elektronischen Sensor erfassen. Nur wie kommt diese Information in das Python-Skript des Raspberry Pi? Genau, über die richtige Schaltung an der GPIO-Schnittstelle. Da nicht alle Sensoren gleich anzuschalten sind, werden 25 davon in diesem Buch vorgestellt. Es zeigt Ihnen, wie sie angeschlossen und ausgelesen werden. Dieses Wissen können Sie nutzen, um jeden anderen Sensor an Ihren Pi anzuschließen.

Viele Sensoren liefern analoge Werte, aber der Raspberry Pi kann diese nicht direkt verarbeiten. Zu diesem Zweck muss ein A/D-Wandler an die GPIO-Schnittstelle angeschlossen werden. Auch das zeigt dieses Buch. Für große Schaltungen gibt es nicht genug nutzbare GPIO-Eingänge – diese können aber mit einem elektronischen Bauteil erweitert werden. In einer praxisnahen Erklärung erfahren Sie, wie das funktioniert.

Neben den elektronischen Aufbauten liefert das Buch auch den Quellcode zur Ansteuerung der Sensoren mit. Damit können Sie direkt mit eigenen Projekten loslegen und sparen sich viele Stunden des Ausprobierens.

Der komplette Quellcode aus dem Buch auf www.buch.cd

Aus dem Inhalt:

- GPIO für eigene Projekte nutzen
- I²C-Bus und SPI im Projekteinsatz
- Analog-digital-Wandler nachrüsten
- GPIO-Anschlüsse erweitern
- Gertboard, GertDuino und Embedded Pi
- Licht- und Farbsensoren
- Temperatur messen
- Abstands- und Infrarotsensor
- Bewegungssensor
- Kameramodul als Kamerasensor
- Gesichtserkennung mit OpenCV
- Touch- und Drucksensor
- Akustiksensor im Betrieb
- Höhenbestimmung mit Luftdrucksensor
- Lage- und Neigungssensor
- Gyrometer-Experimente
- Feuchtigkeit im Blumentopf
- Linear-Hall-Sensor zur Stromstärkenmessung
- Gas- und Rauchsensor

Über den Autor:

E.F. Engelhardt, Jahrgang 1975, hat bereits über 40 Computerbücher veröffentlicht - und keines dieser Bücher ist wie andere Computerbücher: Der Autor beginnt direkt mit Praxis, ohne langatmige Technikerläuterungen. E.F. Engelhardt ist Autor des Bestsellers „Hausautomation mit Raspberry Pi“. Nun zeigt er seinem Raspberry Pi die Welt und schließt analoge und digitale Sensoren an. Wie in jedem seiner Bücher hat Engelhardt die Projekte komplett selbst entwickelt.



9 783645 604901

30,- EUR [D] / 30,90 EUR [A]
ISBN 978-3-645-60490-1

Besuchen Sie
unsere Website
www.franzis.de

FRANZIS