

Rüdiger Follmann

Das Raspberry Pi Kompendium

2. Auflage

EBOOK INSIDE

 Springer Vieweg

Das Raspberry Pi Kompendium

Rüdiger Follmann

Das Raspberry Pi Kompendium

2., erweiterte und überarbeitete Auflage

 Springer Vieweg

Rüdiger Follmann
Vice President
IMST GmbH
Kamp-Lintfort, Deutschland

Ergänzendes Material zu diesem Buch finden Sie auf
<http://www.springer.com/978-3-662-58143-8>

ISBN 978-3-662-58143-8 ISBN 978-3-662-58144-5 (eBook)
<https://doi.org/10.1007/978-3-662-58144-5>

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Springer Vieweg

© Springer-Verlag GmbH Deutschland, ein Teil von Springer Nature 2014, 2018

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Der Verlag, die Autoren und die Herausgeber gehen davon aus, dass die Angaben und Informationen in diesem Werk zum Zeitpunkt der Veröffentlichung vollständig und korrekt sind. Weder der Verlag, noch die Autoren oder die Herausgeber übernehmen, ausdrücklich oder implizit, Gewähr für den Inhalt des Werkes, etwaige Fehler oder Äußerungen. Der Verlag bleibt im Hinblick auf geografische Zuordnungen und Gebietsbezeichnungen in veröffentlichten Karten und Institutionsadressen neutral.

Springer Vieweg ist ein Imprint der eingetragenen Gesellschaft Springer-Verlag GmbH, DE und ist ein Teil von Springer Nature
Die Anschrift der Gesellschaft ist: Heidelberger Platz 3, 14197 Berlin, Germany



Inhaltsverzeichnis

Geleitwort	XI
Einleitung	XIII
Embedded Systeme	XIII
Die Zielgruppe	XIV
Der Aufbau dieses Buches	XIV
1 Der Raspberry Pi	1
1.1 Die Qual der Wahl	1
1.1.1 Zubehör und Kosten	2
1.1.2 Darf es etwas weniger sein?	3
1.1.3 Die Übersicht behalten	3
1.1.4 Quo vadis Pi?	4
1.2 Die Raspberry Pi-Connection	5
1.2.1 Die erste Inbetriebnahme	6
1.2.2 Vorbereiten der SD-Karte	6
1.2.3 Peripherie anschließen	8
1.3 Einrichten des Systems	8
1.3.1 Netzwerkeinstellungen	12
1.4 Erstes Video und MPEG-2 Codec	13
1.5 Weniger kann mehr sein	18
1.5.1 Raspbian Lite	18
1.5.2 Systemstart von einem externen Laufwerk	18
1.5.3 Netzwerk-Boot	19
1.6 Der Betrieb	25
1.6.1 Firmware-Warnungen	25

1.6.2	Real Time Clock	25
1.6.3	Audioausgabe	26
1.6.4	Temperatur	26
2	Die UNIX-Welt	27
2.1	Updates	27
2.1.1	Updates in der Konsole	27
2.1.2	Programminstallation per Desktop	28
2.2	Die LEDs	29
2.3	Fernzugriff	29
2.3.1	SSH für Terminal-Fans	30
2.3.2	Fernzugang mit Schreibtisch	31
2.4	Netzwerk editieren	32
2.4.1	Dynamische IP	35
2.4.2	Statische IP	35
2.4.3	WLAN	36
2.5	Bluetooth	38
2.5.1	Kabellose Tastatur	38
2.5.2	Kopfhörer	39
2.5.3	Dateiübertragung	39
2.6	Init-Skripte	41
2.7	Wichtige Tipps	43
2.7.1	Missglückter Start	43
2.7.2	Partition verkleinern	44
2.7.3	config.txt	45
2.7.4	SD-Schreibschutz	47
2.8	Windows-Umsteiger	47
2.8.1	E-Mail	47
2.8.2	Dateibrowser	47
2.8.3	Verbindung mit einem NAS	48
3	Fernzugriff und mehr	51
3.1	Windows ruft „Pi“	51
3.1.1	Remote Desktop	53
3.1.2	Heimweh	54
3.1.3	Sicheres Kopieren	55
3.1.4	Fernweh	56
3.1.5	Cisco Client	56
3.1.6	Microsoft PPTP	58
3.2	Tanze Samba mit mir	61
3.3	iSCSI ist kein Apple-Device	63
3.4	Streng vertraulich!	65
3.4.1	Container erstellen	67
3.4.2	Wer soll sich das denn merken?	69

3.5	My home is my castle	70
4	Das erste Programm	73
4.1	Hallo Welt!	73
4.1.1	Auslagerungsdatei	76
4.2	Hallo Python!	77
4.3	Repositories	78
4.3.1	mercurial, svn und git	78
4.4	Betriebssystem hautnah	79
4.4.1	Die Raspberry Pi Firmware	79
4.4.2	Der eigene Kernel	80
4.5	Patches	83
5	Video Disc Recorder	85
5.1	Fernsehen	85
5.2	DVB-Stick	86
5.2.1	Neuere Modelle	91
5.3	Fernbedienung	93
5.4	VDR	99
5.5	Externe Festplatte	106
5.6	VDRAdmin-am	108
5.7	Schöner fernsehen	109
5.7.1	Stream me up!	115
6	KODI	119
6.1	KODI aus dem Repository	120
6.2	KODI aus dem Quelltext	120
6.3	Übersetzungsfehler und Neuerungen	126
6.4	LibreELEC	128
6.4.1	Einrichten der Fernbedienung	129
6.4.2	Der Wetterdienst	131
6.4.3	Tvheadend zum Fernsehen	131
6.4.4	DVD und Blu-ray	135
6.4.5	Amazon Prime	136
6.4.6	Mobile Helfer	139
6.5	AirPlay und Co.	140
6.6	Externalplayer	143
6.7	MP3	144
6.8	Ausblick	145

7	Ambilight	147
7.1	Ambilight für LibreELEC	147
7.1.1	Ambilight Hardware anschließen	148
7.1.2	Ambilight Software installieren	152
7.1.3	Ambilight Fernbedienung	162
7.1.4	Ambilight ein- und ausschalten	162
7.2	Ambilight für jede HDMI-Quelle	167
7.2.1	Hardware anschließen	169
7.2.2	Software installieren	171
7.2.3	LEDs konfigurieren	171
8	Server und Datenbanken	175
8.1	AirPrint	175
8.2	Apache	180
8.2.1	PHP	181
8.2.2	MySQL	182
8.3	WLAN-Hotspot	185
8.3.1	Hostapd	185
8.3.2	Chillispot	191
8.3.3	Zertifikate	193
8.3.4	Radius	196
8.3.5	Kein CGI-Freund?	203
8.3.6	Firewall	203
8.3.7	Toooooor!	205
8.4	OpenVPN	208
8.4.1	Der Klient	212
8.5	CMS	214
8.6	Mach' mich nicht NAS!	219
9	Erweiterungen	223
9.1	Erweiterungs-Anschlüsse	224
9.2	Servieren à la Python	227
9.2.1	Ein minimalistischer Webserver	227
9.2.2	GPIO-Ansteuerung	228
9.2.3	SPI	233
9.2.4	Von C nach Python	237
9.2.5	Ansteuerung per Webserver	241
9.2.6	Sage mir, wie Du heißt	243
9.2.7	Messtechnik	246
9.3	Zeige mir, was Du kannst	247
9.3.1	Ausgabe	249

9.4	Kamera und Anwendungen	254
9.4.1	Installation	255
9.4.2	Betrieb	256
9.4.3	Python	259
10	Hausautomatisierung	261
10.1	Busware 868 MHz Transceiver	262
10.1.1	<i>Real Time Clock</i>	262
10.2	FHEM	264
10.2.1	Absicherung	265
10.2.2	Hinzufügen von Geräten	267
10.2.3	Alternative Sender	271
10.2.4	Firmware aktualisieren	272
10.2.5	Vergleich der Systeme	273
10.2.6	Erweiterungen	274
10.3	Alarmanlage	276
10.3.1	Ruf' mich an	279
11	Schreiben mit dem Raspberry Pi	287
11.1	LibreOffice	287
11.1.1	<i>LibreWriter</i>	289
11.2	TEX	290
11.2.1	TEX-Beispiel	291
11.2.2	Bewerbungen schreiben	294
11.2.3	Rechtschreib-Prüfung	298
11.2.4	Ein Buch schreiben	299
12	Software-Perlen	301
12.1	N64-Emulator	301
12.1.1	Konfiguration und Gamepad	302
12.2	RetroPie	304
12.3	Noch mehr Spiele	306
12.4	Minecraft	307
12.5	Mathematik-Software	308
12.6	Fourier-Transformation	310
12.7	WLAN-Radio	310
	Anhang	313
	Fehlersuche	313
	Die wichtigsten LINUX-Befehle	315

Literatur	319
Bücher	319
Artikel	319
Verschiedenes	319
Abbildungsverzeichnis	321
Tabellenverzeichnis	327
Index	329



Geleitwort

Seit nunmehr vier Jahren setze ich erfolgreich den Raspberry Pi ein. Waren es anfangs nur VDRs, die ich damit realisiert habe, so kam inzwischen auch das Thema Hausautomation hinzu. Ein in einem passenden Hutschienengehäuse in der Unterverteilung montierter Raspberry Pi sorgt jetzt für die Verbindung zwischen einer Wetterstation und dem digitalSTROM-System. So werden jetzt bei Sonnenschein die Rollläden der jeweiligen Hausseite automatisch geschlossen, oder bei Wind bzw. Regen die Markisen eingefahren. Auch die Überwachung der Fensterkontakte und die Anzeige auf einem kleinen LED-Display neben der Haustüre schafft der Pi dank seiner vielen GPIO-Pins mit Leichtigkeit. Und wenn ich mal vergesse, das Gargentor zu schließen, schickt mir der Pi eine Email, um mich daran zu erinnern. Aber der Raspberry Pi eignet sich nicht nur für VDRs und Haussteuerung. Die Möglichkeiten sind schier endlos, wie allein schon ein Blick in das Inhaltsverzeichnis dieses Buches zeigt: Multimedia, Server und Datenbanken, alle möglichen Arten von Steuerungen und Regelungen, ja sogar Textverarbeitung und Spiele sind damit möglich. Und Rüdiger Follmann hält sich erfreulicherweise nicht lange mit theoretischen Abhandlungen auf, sondern es geht sehr schnell mit der praktischen Anwendung los! Halten Sie also am besten ihren Raspberry Pi griffbereit, wenn Sie anfangen, dieses Buch zu lesen, denn Sie werden sicher, genau wie ich, das Gelesene sofort in die Tat umsetzen wollen.

Taufkirchen, Juli 2018
Klaus Schmidinger



Einleitung

Embedded Systeme

Embedded Systeme sind aus unserem Leben nicht mehr wegzudenken. Sie sind Rechner oder Computer, die in einem technischen Gerät eingebunden (embedded) sind und nach außen unsichtbar die Steuerung dieser Geräte durchführen. Viele benutzen sie, ohne es überhaupt zu wissen, beispielsweise in Form eines mobilen Telefons, als DSL-Modem oder für die Hausautomatisierung. Da diese Systeme unter Umständen Tag und Nacht 24 Stunden am Tag laufen, zeichnet sie vor allem ein geringer Leistungsverbrauch aus. Die sparsamsten Systeme haben eine Leistungsaufnahme von gerade einmal 2,5 W während sie arbeiten. Bei einem Betrieb rund um die Uhr bedeutet das Kosten von weniger als 5 € pro Jahr.

Herz der Systeme bildet der Prozessor, der meist als ARM-Architektur (Advanced RISC¹ Machines) ausgelegt ist. Ihm zur Seite gestellt arbeitet die GPU (Graphics Processing Unit). Sie ist für die Grafikausgabe, also unter anderem für die beschleunigte und damit flüssige Videowiedergabe zuständig. Hier gibt es große Unterschiede nicht nur bezüglich der Leistungsfähigkeit, sondern auch bezüglich der Software-Unterstützung durch den Hersteller. Während Broadcom Beispiele für die Programmierung seiner BCM2837-Einheit zur Verfügung stellt (Raspberry Pi), müssen andere Treiber als Binärdatei aus Android-Systemen extrahiert werden, damit sie unter anderen Betriebssystemen mit mehr oder weniger großem Erfolg verwendet werden können (Cubieboard). Dies ist darin begründet, dass viele Embedded Boards als Entwickler-Boards für Mobilgerätehersteller gedacht sind und nicht als Endanwendergeräte konzipiert worden sind.

Dieses Buch beschreibt die Anschluss- und Verwendungsmöglichkeiten gängiger Embedded Systeme am Beispiel des Raspberry Pi, welcher als Ersatz für einen kleinen, sparsamen PC mit einem Linux-Betriebssystem verwendet werden kann. Ein erster Start vermittelt schnelle Erfolgserlebnisse z. B. durch das Abspielen von Videos oder Filmen. Windows-Umsteiger erhalten wertvolle Tipps zur Nutzung der Linux-Konsole oder der graphischen Oberfläche(n).

Ein weiteres Kapitel beschäftigt sich mit der Fernbedienbarkeit der Systeme, sowohl von einer Konsole als auch vom einem Windows PC aus. Bevor es dann zu den ersten Projekten geht, lernt der Leser, wie das System aktuell gehalten werden kann, wie Programmpakete installiert werden können oder wie man Quelltext „auscheckt“ und kompiliert, also in ein ausführbares Programm verwandelt. Dazu gehört auch die Installation eines eigenen Kernels.

¹Reduced Instruction Set Computer

Ein großes Kapitel ist dem Thema Multimedia gewidmet. Lernen Sie, wie man ein Embedded Board als Satellitenreceiver oder Mediacenter einrichtet und verwendet, streamen Sie das TV-Programm auf Ihr Handy oder zeigen Sie Ihre mit dem Handy gemachten Urlaubsfotos kabellos auf dem Fernseher an. Ersetzen Sie teure Audio-Streaming-Lösungen durch ein Embedded Board und hören Sie die Musik Ihres iPhones auf der Stereoanlage. Statten Sie Ihren Fernseher mit einem Ambientlight-System aus.

Auch das Thema Netzwerk kommt nicht zu kurz. In wenigen Schritten wird erklärt, wie jeder Drucker in das heimische Netzwerk integriert werden kann, um ihn dann auch vom Mobiltelefon aus zu benutzen. Sie wollen einen verschlüsselten Zugang zu Ihrem Heimnetz, um unterwegs auf Ihren PC zu Hause zugreifen zu können? Kein Problem, das `openvpn`-Tutorial zeigt Ihnen, wie es geht. Realisieren Sie einen gesicherten WLAN-Zugang für Gäste oder Kunden, verschleiern Sie Ihre Identität im Netz durch TOR (The Onion Routing). Weiterhin lernen Sie, wie man einen Webserver mit Datenbankanbindung einrichtet oder sogar Content Management Systeme auf Embedded Boards nutzt.

Das Kapitel „Erweiterungen“ zeigt, wie Zusatzhardware verwendet werden kann. Dazu zählen Displays oder Kameras, aber auch Platinen, die über serielle Protokolle programmiert werden können.

Sie möchten Ihr Zuhause automatisieren? Kein Problem. Schalten Sie Steckdosen, lesen Sie Temperaturen oder Wasserstände ab und lassen Sie das Embedded Board im Falle eines Alarms bei Ihnen anrufen. Das Kapitel „Hausautomatisierung“ zeigt, wie das mit einfachen Mitteln realisiert werden kann.

Lust auf ein Spiel? Das Kapitel „Software-Perlen“ demonstriert, was man alles aus den kleinen Embedded Boards herausholen kann. Nutzen Sie Office-Pakete oder schreiben Sie Ihre Bewerbung. Selbst auf einem Embedded Board ist das kein Problem.

Die Zielgruppe

Dieses Buch richtet sich sowohl an Anfänger als auch an fortgeschrittene Nutzer. Alle vorgestellten Programme und Skripte finden Sie im Download-Bereich dieses Buches unter <https://www.springer.com/de/book/978-3-662-58143-8>. Damit können alle besprochenen Anwendungen sofort ausprobiert werden. Schritt für Schritt wird erklärt, wie die Programme benutzt, installiert und angepasst werden können. Fortgeschrittene Benutzer können dabei die ersten Kapitel zur Inbetriebnahme und Erklärung des Linux-Betriebssystems überspringen.

Sie erhalten in den folgenden Kapiteln ausführliche Informationen über das „Auschecken“ von Quelltext oder das Übersetzen und Konfigurieren der Programme.

Der Aufbau dieses Buches

Wegen der besseren Übersichtlichkeit sind wichtige Hinweise in diesem Buch besonders markiert, ebenso wie Quelltexte. Im Buch wird dabei folgendes Schema verwendet:



Wichtige Hinweise stehen in einer Info-Box. Sie weisen den Benutzer auf mögliche Probleme hin oder geben wichtige Anhaltspunkte.



Hinweise zu Dateien oder Verzeichnissen auf dem Download-Server <https://www.springer.com/de/book/978-3-662-58143-8> befinden sich in der Download-Box. Hier finden Sie die im Buch beschriebenen Dateien, Skripte oder Programme. Alle Dateien sind mit dem Programm `tar` eingepackt und mit dem Programm `bzip2` komprimiert. Zum Entpacken der jeweiligen Datei laden Sie diese herunter und entpacken sie mit dem Befehl `tar xvfj dateiname.tar.bz2`.

Quelltext steht ebenfalls in einer eigenen Box. Die Zeilen sind nummeriert.

Code-Ausschnitt

```
1  for i:=maxint to 0 do
2    begin
3      j:=square(root(i));
4    end;
```

Am Ende jedes Kapitels gibt es eine Zusammenfassung. Diese beschreibt noch einmal die wichtigsten Themen, die in dem jeweiligen Kapitel behandelt worden sind.

Zusammenfassung Dieses Kapitel war die Einleitung zum Buch „Das Raspberry Pi Kompendium“. Ich hoffe, Sie haben beim Lesen und Ausprobieren des Buches denselben Spaß, den ich beim Basteln, Programmieren und beim Schreiben dieses Buches hatte. ■

1 — Der Raspberry Pi

1.1 Die Qual der Wahl

Embedded Systeme gibt es wie Sand am Meer [Döl14], angefangen bei Spielekonsolen wie der OUYA (<http://www.ouya.tv>) bis hin zu Mehrkernboliden wie dem Amlogic S912 (<http://www.amlogic.com>). Da mit wenigen Einschränkungen alle hier vorgestellten Projekte auf allen Boards umgesetzt werden können, habe ich mich dazu entschieden, das über 17 Millionen mal verkaufte und damit beliebteste Embedded Board zu nutzen, den Raspberry Pi. Der Raspberry Pi (Abb. 1.1) ist ein Einplatinencomputer im Kreditkartenformat, der in der aktuellen 3B+-Variante alle benötigten Schnittstellen mitbringt: 1 GB Arbeitsspeicher, 4 USB 2.0 Anschlüsse, HDMI- und FBAS-Ausgänge zum Anschluss eines Monitors oder eines Fernsehers, eine 3,5 mm Klinkenbuchse zur analogen Tonausgabe (die digitale Tonausgabe erfolgt über HDMI) oder eine 10/100/300 Mbit/s Ethernet-Buchse, die in der A-Variante nicht vorhanden ist. Die Spannungsversorgung erfolgt über eine Micro-USB Buchse (min. 5.1 V, 2.5 A).

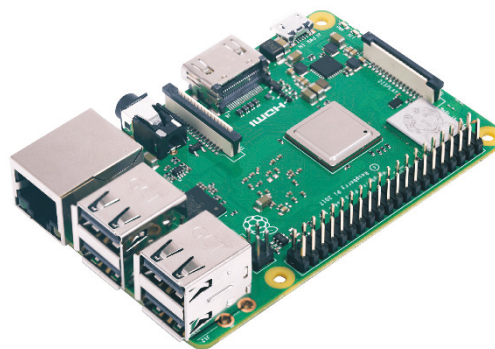


Abbildung 1.1: Der Raspberry Pi (Model 3B+) ist ein vollwertiger Rechner im Kreditkartenformat (Quelle: Reichelt).

Info

Während die USB-Anschlüsse der älteren Raspberry Pi-Modelle maximal 100 mA liefern, liefert das 3B+-Modell bis zu 1.2 A für alle USB-Ports. Das reicht auf jeden Fall, um eine Tastatur oder Maus zu betreiben, muss aber nicht ausreichend für den Betrieb einer Festplatte sein. Sollten Sie darüber hinaus weitere Geräte an den Raspberry Pi anschließen wollen, empfiehlt sich die Verwendung eines aktiven USB-Hubs oder einer externen Spannungsversorgung für das USB-Gerät. Gleichzeitig empfehle ich die Verwendung eines Netztes mit mindestens 2.5 A. Sollten Sie unerklärliche Abstürze haben, ist die häufigste Ursache ein zu schwach dimensioniertes Netzteil.

Der Raspberry Pi stellt verschiedene weitere Anschlüsse zur Verfügung, über die beispielsweise eine Kamera oder ein Display angeschlossen werden können. Diese Anschlüsse lernen Sie in den folgenden Kapiteln näher kennen.



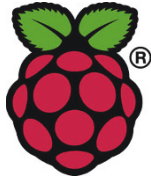


Abbildung 1.2:
Foundation

Logo (Quelle:
raspberrypi.org).

Als GPU kommt Broadcoms „VideoCore IV“ Grafikprozessor zum Einsatz, der Filme in Full-HD-Auflösung unterstützt (1080p30 H.264 high-profile). Lizenzen für das in Hardware beschleunigte Wiedergeben von MPEG-2- und VC1-kodierten Filmen können bei der Raspberry Pi Foundation für einen geringen Betrag (ca. 3 €) bestellt werden (<http://www.raspberrypi.org>). Die Installation des Codecs ist in Kapitel 1.4 beschrieben. Als Datenspeicher dient ein Micro-SD-Kartenschacht, der die Formate SDHC und SDXC nutzen kann.



Der Raspberry Pi kann nicht mit allen SD-Karten umgehen. Die Seite http://elinux.org/RPi_SD_cards gibt Auskunft darüber, welche SD-Karten geeignet sind und welche nicht.

Der Micro-SD-Karteneinschub befindet sich auf der Rückseite der Platine. Der Raspberry Pi 3B+ verfügt über 4 LEDs, von denen zwei in die Ethernetbuchse integriert sind. Die anderen beiden befinden sich auf der Hauptplatine. Sie sind wie folgt belegt:

- Zugriffe auf die SD-Karte werden durch die grüne ACT-LED angezeigt. Die LED kann durch Software gesteuert werden und flackert beispielsweise, wenn Daten von oder zu der SD-Karte übertragen werden.
- Die rote PWR-LED signalisiert eine angeschlossene 5 V Spannungsversorgung. Auch diese LED kann durch Software gesteuert werden.
- Erfolgen Zugriffe auf die Ethernet-Schnittstelle, wird das durch Blinken der orange-farbenen LNK-LED angezeigt.
- Die grüne Ethernet-LED zeigt an, ob eine hardwareseitige Verbindung besteht.

Beide Ethernet-LEDs lassen sich nicht ohne Weiteres manipulieren.

1.1.1 Zubehör und Kosten

Der Raspberry Pi kostet in der 3B+-Version 35 €. Unbedingt erforderlich ist ein Micro-USB-Netzteil, welches ca. 12 € kostet. Als SD-Karte empfiehlt sich minimal eine 16 GB Karte für 7 €. Möchte man alle Beispiele dieses Buches inklusive der zugehörigen Entwicklungsumgebungen installieren, empfiehlt sich eine 32 GB SD-Karte, die mit ca. 12 € zu Buche schlägt. Wenn man bereits USB-Tastatur und -Maus sowie einen Monitor besitzt, fehlen nur noch Netzwerkkabel, HDMI-Kabel (A-Typ) und gegebenenfalls Lautsprecher und ein 3,5 mm Klinkenkabel. Die Raspberry Pi Platine kann ohne Probleme ohne Gehäuse und Kühlkörper betrieben werden. Letzterer ist nur bei großer Übertaktung zu empfehlen, da er die Temperatur der Platine dann um ein paar Grad senkt. Vorausgesetzt, Kabel, USB-Geräte und Monitor sind bereits vorhanden, liegen die Kosten für einen Pi ohne Gehäuse bei ca. 59 €. Bei den Gehäusen kann man auf eine enorme Vielfalt zurückgreifen. Diese reicht von selbst gebauten Lego-Gehäusen bis zum gefrästen Aluminium oder gefaltetem Papier. Der Fantasie sind hier keine Grenzen gesetzt, die Preise liegen zwischen 8 € und 30 €. Das Pi 3B+-Modell passt dabei ohne Probleme in das Gehäuse des Vorgängers.

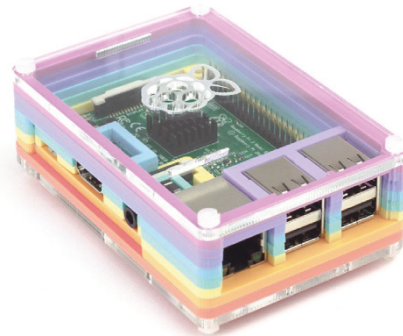


Abbildung 1.3: Das Pibow Gehäuse sieht aus wie ein Regenbogen (Quelle: Pimoroni).

1.1.2 Darf es etwas weniger sein?

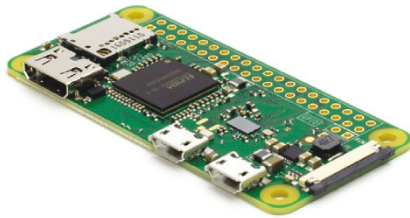


Abbildung 1.4: Der Raspberry Pi Zero W (Quelle: Pimoroni).

Wem das aktuelle Pi-Schlachtschiff zu teuer ist, der wird vielleicht mit dem Raspberry Pi Zero liebäugeln. Für nur 5 € erhält man den kleinen 1-kernigen Pi inklusive einer freien Micro-USB-Buchse, 512 MB Arbeitsspeicher und einem Mini-HDMI-Port. Die zweite Micro-USB-Buchse wird zur Spannungsversorgung benötigt. Die 40-polige Steckerleiste - die man übrigens (außer in der H-Version) selbst anlöten muss - weist volle Kompatibilität mit dem größeren Bruder auf.

Ein Kameraanschluss ist ebenfalls vorhanden. Dieser ist allerdings nicht mit dem des 3B+-Modells kompatibel. Für einen Aufpreis von weiteren 10 € gibt es den Pi Zero in der W-Variante mit WLAN und Bluetooth - ein Muss, wenn die kleine Platine eine Netzwerkver-

bindung benötigt. Für den Zero W gibt es ebenfalls eine Fülle an Zubehör. Der Energiebedarf des Winzlings liegt bei nur 0,5 bis 0,7 W. Dafür bringt er nicht ganz die Leistung des großen Bruders, weil noch der alte Broadcom SoC (*System-on-Chip*) verbaut ist.

Damit eignet sich der Pi Zero (WH) besonders für Projekte, die wenig Einbauraum zulassen und höchstens eine WLAN-Verbindung benötigen. Für viele Anwendungen werden Adapter benötigt (Micro-USB auf USB, Mini-HDMI auf HDMI, USB-Hub), so dass der Preisvorteil schnell verfliegen ist, vom Kabelsalat einmal abgesehen.

1.1.3 Die Übersicht behalten

Tabelle 1.1 fasst alle aktuellen Raspberry Pi-Modelle zusammen und gibt einen Überblick über deren Eigenschaften. Neben den Boards für den Heimbedarf gibt es auch ein sogenanntes „Compute Modul“ (Abb. 1.5). Das Modul stellt einen Standard-DDR2-SODIMM-Konnektor für die Außenwelt zur Verfügung. Dieser Konnektor wird ebenfalls dazu verwendet, Speicherbausteine in einem PC mit dem Mainboard zu verbinden. Kommen Sie aber bitte nicht auf die Idee, das Raspberry Pi Compute Modul in einem PC Mainboard zu betreiben. Das wird nicht funktionieren und mit großer Wahrscheinlichkeit werden Sie nicht nur das Modul dabei zerstören. Es gibt auch hier zum Testen passende Boards mit einem entsprechenden Sockel, der das Compute Modul aufnehmen kann. Während das Compute Modul ca. 30 € kostet, ist die Compute Modul E/A-Platine mit 120 € deutlich teurer.

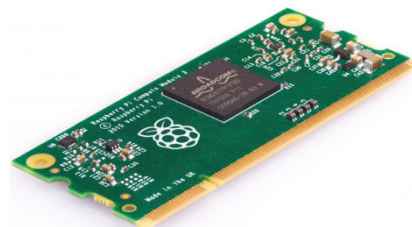


Abbildung 1.5: Das Raspberry Pi Compute Modul ist hauptsächlich für Firmen gedacht, die ihr eigenes PCB entwickeln (Quelle: raspberrypi.org).



Bitte betreiben Sie das Raspberry Pi Compute Modul auf keinen Fall im Speichersockel eines PC Mainboards. Dafür ist es nicht gedacht und Sie werden mit hoher Wahrscheinlichkeit sowohl das Mainboard als auch den Sockel dabei zerstören.

Wie aber wählt man nun den für die benötigten Einsatzzwecke optimalen Raspberry Pi aus? Für alle Anwendungsfälle, die maximale Leistung benötigen, ist der Raspberry Pi 3 B+ die erste Wahl: Er bietet die schnellste Netzverbindung und hat den höchsten CPU Takt.

Der hier verbaute WLAN-Chip unterstützt nicht nur das 2,4 GHz Band, sondern funkt auch bei 5 GHz, und das deutlich schneller als die älteren Modelle. Darüber hinaus unterstützt er PoE (*Power over Ethernet*). Die CPU ist mit 1,4 GHz getaktet und damit theoretisch 17% schneller als sein älterer Bruder. Wenn jedoch alle 4 Kerne der CPU ausgelastet werden, reduziert der Pi 3B+ aus Wärmegründen seinen Takt, so dass vom Geschwindigkeitsvorteil wenig übrig bleibt [Sto18].

Der Raspberry Pi 3B hat nicht ganz die halbe Leistungsaufnahme des 3B+ und ist daher zu empfehlen, wenn es auf hohe CPU-Leistung und geringe Leistungsaufnahme ankommt. Den Raspberry Pi 2 empfehle ich nicht mehr, da es keine Vorteile gegenüber dem Pi 3 gibt.

Der Raspberry Pi Zero (W) empfiehlt sich überall da, wo Einbauraum knapp ist und es nicht auf hohe Leistung ankommt.

	RPi 3B+	RPi 3B	RPi 2B	RPi Zero (W)
Größe	85,6 × 56 mm ²	85,6 × 56 mm ²	85,6 × 56 mm ²	65 × 30 mm ²
SOC	BCM2837	BCM2837	BCM2836	BCM2835
CPU	ARM Cortex-A53 (ARMv8-A)	ARM Cortex-A53 (ARMv8-A)	ARM Cortex-A7 (ARMv7)	ARM1176JZF-S (ARMv6)
CPU-Kerne	4	4	4	1
CPU-Takt	4 × 1400 MHz	4 × 1200 MHz	4 × 900 MHz	1000 MHz
RAM	1024 MB	1024 MB	1024 MB	512 MB
USB	4 × USB2.0	4 × USB2.0	4 × USB2.0	1 × USB2.0
Audio	HDMI (digital) 3,5 mm Klinke	HDMI (digital) 3,5 mm Klinke	HDMI (digital) 3,5 mm Klinke	HDMI (digital)
Netzwerk	10/100/1000 ¹ Mbit/s	10/100 Mbit/s	10/100 Mbit/s	–
WLAN	2,4/5 GHz WLAN ac	2,4 GHz WLAN b/g/n	nein	ja (Zero W) nein (Zero)
Bluetooth	4.2	4.1	nein	ja (Zero W) nein (Zero)
GPIO	40 Pins	40 Pins	40 Pins	optional
Leistungs- aufnahme	max. 7 W	max. 4 W	max. 4 W	0,5–0,7 W
Netzteil	5 V Micro USB min. 2,5 A	5 V Micro USB min 2,5 A	5 V Micro USB min. 2 A	5 V Micro USB min. 1 A
Preis	ca. 35 €	ca. 35 €	ca. 32 €	ca. 26 € (Set)

Tabelle 1.1: Übersicht aller aktuellen Raspberry Pi-Modelle (Quelle: datenreise.de).

1.1.4 Quo vadis Pi?

Der c't-Artikel [Sto18] ist mit „Der letzte seiner Art“ überschrieben. Eben Upton, der Gründer der Raspberry Pi Stiftung, hat dort wohl bekanntgegeben, dass 2019 ein neuer Raspberry Pi erscheinen könnte, jedoch hat er nichts über dessen Eigenschaften gesagt.

¹Da der Ethernet-Controller über USB2.0 angebunden ist, stehen effektiv nur 300 Mbit/s zur Verfügung.

Schaut man sich die zugegebenermaßen schlechter unterstützte Konkurrenz an, so wird relativ schnell deutlich, wo die Reise hingehen kann: Echtes GBit Ethernet und UHD (*Ultra High Density*) Grafikauflösung.

1.2 Die Raspberry Pi-Connection

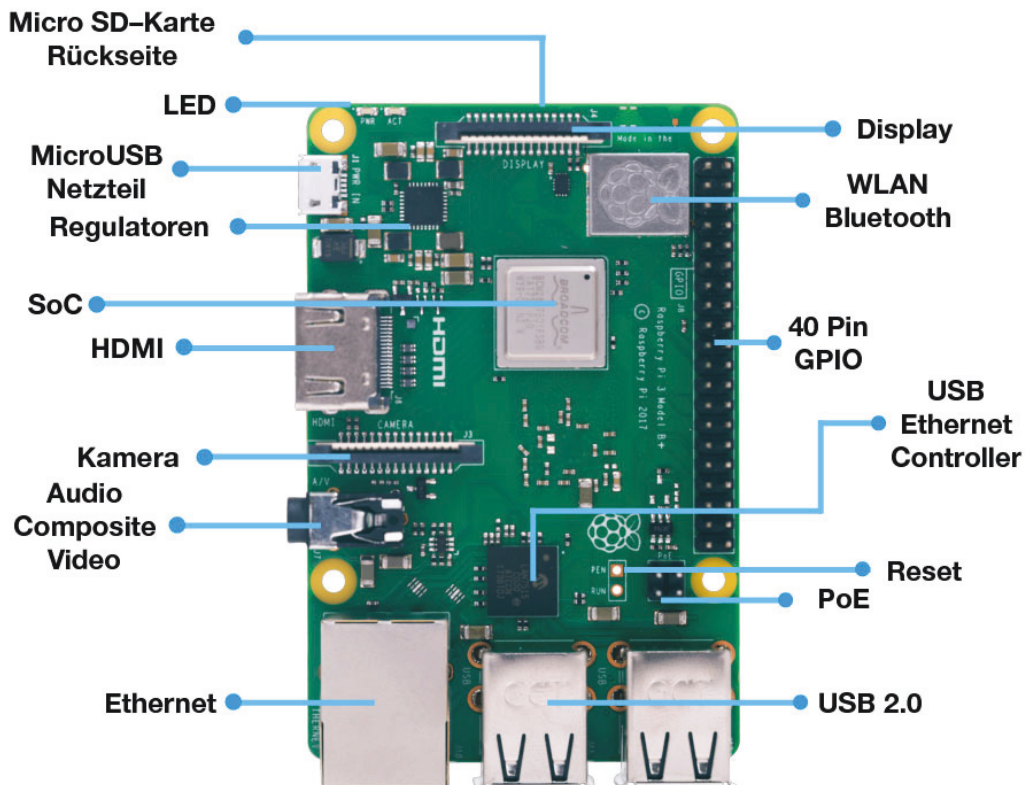


Abbildung 1.6: Die Anschlüsse des Raspberry Pi 3B+.

Abbildung 1.6 zeigt den Raspberry Pi 3B+ mit allen Anschlüssen und wichtigen Bestandteilen der Platine. Der Micro-SD-Kartenslot befindet sich auf der Rückseite der Platine. Die Micro-SD-Karte wird mit den Kontakten zur Platine hin in den Slot eingeführt. Der Raspberry Pi 3B+ hat hier im Gegensatz zum Pi 3B keinen Federkontakt. Insgesamt stehen 4 USB2.0-Ports zur Verfügung sowie eine Ethernet-Buchse. Der Ethernet-Controller ist über USB2.0 angeschlossen. Der BCM2837-SoC steckt nun unter einem Blechdeckel.

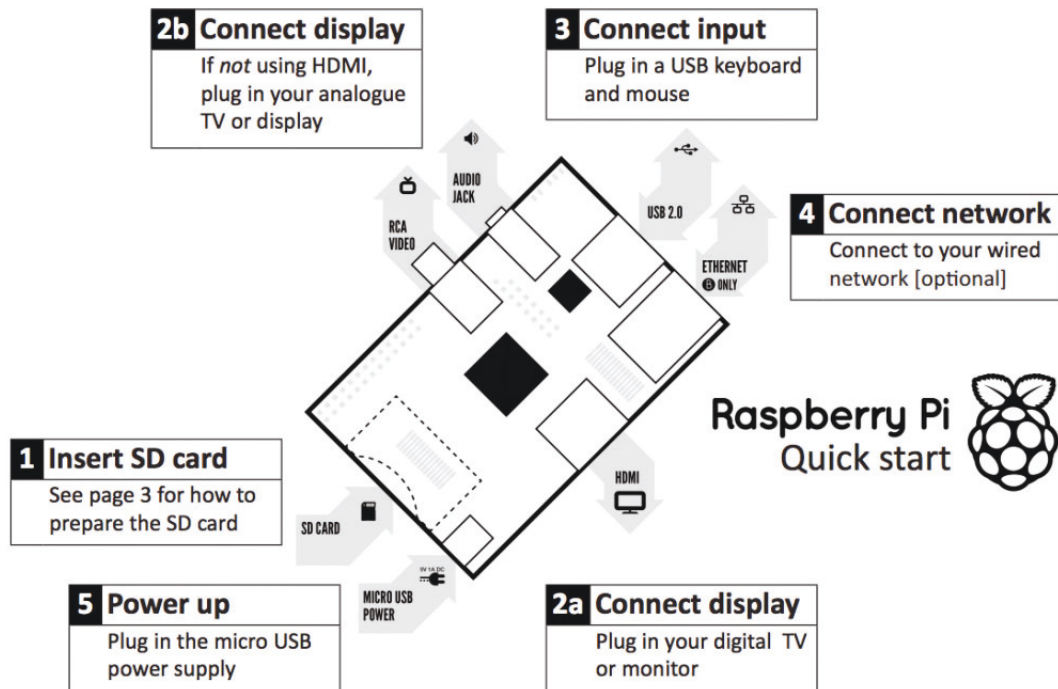
Kamera und externe Displays können mit Hilfe eines Flachbandkabels angeschlossen werden. Hierzu muss die weiße Klemme geöffnet werden, das Flachbandkabel eingesteckt und die weiße Klemme wieder geschlossen werden. Kleinere Displays können über GPIO (*General Purpose Input Output*) angeschlossen werden. Der Pi hat insgesamt 40 Pins, die beispielsweise SPI, I2C, PWM und andere Busse/Signale bereit stellen.

Der HDMI Anschluss vom Typ A erlaubt die Verbindung zu einem Fernseher oder einem Monitor. Die Micro-USB-Buchse nimmt ein Netzteil auf (minimum 5.1 V, 2.5 A). Ein PoE Modul kann alternativ verwendet werden.

Das WLAN/Bluetooth-Modul steckt ebenfalls unter einem Blechdeckel. Es gibt weiterhin vier Bohrlöcher, mit denen der Pi beispielsweise an einer VESA-Halterung befestigt werden kann. Es stehen 2 Pins zur Verfügung, um den Pi zu resetten.

1.2.1 Die erste Inbetriebnahme

Die erste Inbetriebnahme gestaltet sich einfach. Den meisten Raspberry Pi liegt eine englische Schnellstart-Anleitung bei (Abb. 1.7). Gemäß dieser Anleitung schließen wir nun unser Embedded Board an.



1

Abbildung 1.7: Die Schnellstart-Anleitung zeigt, was wo am Raspberry Pi angeschlossen werden muss (Quelle: raspberrypi.org).

1.2.2 Vorbereiten der SD-Karte

Im ersten Schritt bereiten wir die Speicherkarte für den Betrieb vor. Diese sollte mindestens eine Class 6 Karte sein. Führen Sie die SD-Karte mit den Kontakten zum Raspberry Pi hin in den Kartenslot ein.

Dieses Buch geht davon aus, dass die *Raspbian*-Desktop-Distribution auf der SD-Karte installiert wird/ist. Zur Installation sind die folgenden Schritte erforderlich:

1. Laden Sie sich die *Raspbian*-Desktop-Distribution von der Webseite <http://www.raspberrypi.org/downloads>. Das ist die wohl bekannteste Distribution für den Raspberry Pi. Mit ihr werden wir im weiteren Verlauf dieses Buches arbeiten.
2. Entpacken Sie die meist mit dem Programm zip komprimierte Datei auf Ihrem Rechner.
3. Transferieren Sie die entpackte Datei auf Ihre Speicherkarte. Unter Windows können Sie dazu das Programm *Win32DiskImager* verwenden, welches unter <http://sourceforge.net/projects/win32diskimager/> kostenlos angeboten wird.

Info Bitte achten Sie darauf, dass die SD-Karte mindestens die Speicherkapazität des ausgepackten Images hat. Fehlt auch nur ein Byte, kann die Datei nicht auf die Speicherkarte geschrieben werden. Im nächsten Kapitel erfahren Sie, wie man Images - beispielsweise nach einem Backup - verkleinern kann.

Die Oberfläche des Programms Win32 Disk Imager sehen Sie in Abbildung 1.8.

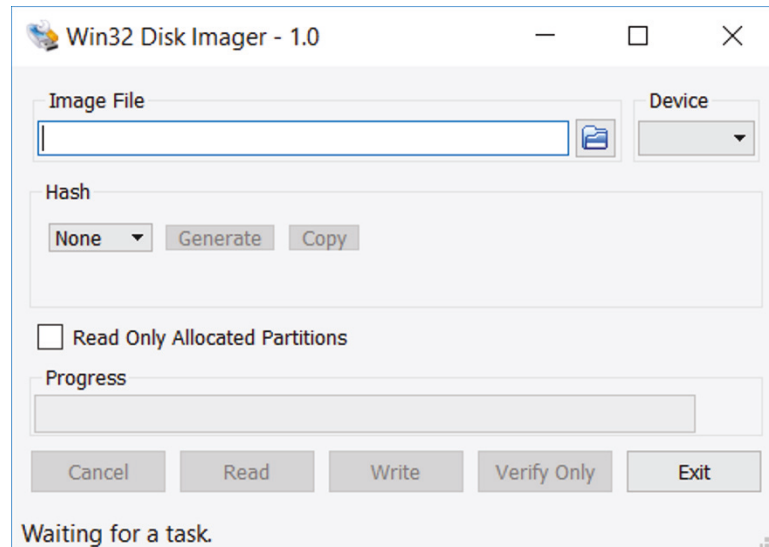


Abbildung 1.8: Win32 Disk Imager kann zum Überspielen des Betriebssystems auf die SD-Karte genutzt werden.

Info Bitte achten Sie darauf, unter *Device* den Laufwerksbuchstaben Ihrer SD-Karte anzugeben. Schreiben auf den falschen Laufwerksbuchstaben, löscht das alle Daten des Laufwerks unwiederbringlich.

Wählen Sie das Image durch einen Klick auf den Ordner aus. Bestimmen Sie danach das *Device* (also den Laufwerksbuchstaben der SD-Karte), auf das Sie das Image schreiben möchten. Starten Sie dann den Schreibvorgang durch einen Klick auf *Write*. Sie können dieses Tool auch benutzen, um Backups Ihrer SD-Karte zu erstellen. Die Image-Datei ist in diesem Fall Ihre Backup-Datei. Ein Klick auf *Read* startet den Lesevorgang und schreibt alle Daten der SD-Karte unter dem ausgewählten Image-Namen.

Unter UNIX Betriebssystemen nutzen Sie bitte die Befehle

Code-Ausschnitt 1.2.1

```
1 umount /dev/sdd1
2 dd bs=4M if=imagename.img of=/dev/sdd
```

zum Transfer der Image-Datei. Bitte beachten Sie dabei, *sdd* mit dem Namen der Partition der SD-Karte zu ersetzen. Es ist erforderlich, dass Sie als Benutzer *root* eingeloggt sind. Der Befehl *dd* (disk dump) überprüft nicht, ob der Speicherplatz auf der Karte ausreichend zum Speichern des Images ist. Er bricht ab, wenn entweder das Image übertragen wurde oder das Ende des Speicherplatzes erreicht ist. Ein inkonsistentes Dateisystem kann dann die Folge sein. Raspberry Pi Images bestehen aus zwei Partitionen. Die erste Partition ist eine FAT32-Partition. Der Bootloader des Raspberry Pi lädt die Startdatei, das sogenannte „Kernel-Image“ immer von dieser Partition. Der Rest des Betriebssystems befindet sich in einer weiteren Partition. Diese Partition ist üblicherweise eine *ext*-Partition.

Sie muss sich nicht zwangsläufig auf der SD-Karte befinden, sondern kann beispielsweise auch auf einer USB-Festplatte liegen. Die Pi Modelle 3B und 3B+ können auch von einem externen Laufwerk oder einem Netzwerk booten. Mehr hierzu finden Sie in Kapitel 1.5.2 und dem darauf folgenden.

1.2.3 Peripherie anschließen

Nachdem sich die SD-Karte im Kartenslot befindet, stellen Sie nun alle weiteren Verbindungen her:

1. Schließen Sie Ihren Monitor oder Fernseher je nach Pi-Modell mit Hilfe des HDMI- oder Cinch-Kabels an und schalten Sie das Gerät an.



Je nachdem, wie der Raspberry Pi eingestellt ist (siehe nächstes Kapitel), muss der Monitor vor dem Starten des Raspberry Pi eingeschaltet sein, damit dieser ein Bild ausgibt.

2. Schließen Sie als Nächstes die USB-Geräte an.
3. Verbinden Sie die Ethernet-Buchse mittels eines Twisted-Pair-Kabels mit Ihrem Router oder Ihrer Netzwerkdose. Die Pi-Modelle Zero, 1 und 2 beinhaltet kein WLAN-Modul. Sie können dieses bei Bedarf über die USB-Schnittstelle nachrüsten. Mehr dazu erfahren Sie im Kapitel 2.4.3.
4. Verbinden Sie als Letztes das Netzteil mit der Micro-USB-Buchse.

Der Raspberry Pi signalisiert durch die Power-LED, dass er betriebsbereit ist und beginnt nun mit dem sogenannten Boot-Vorgang: Die LED, welche die SD-Karten-Aktivität anzeigt, beginnt zu leuchten und die Boot-Meldungen erscheinen auf dem Bildschirm.

1.3 Einrichten des Systems

Nach dem ersten Starten des Raspberry Pi geht es erst einmal an das Einrichten des Systems. Dies kann entweder mit dem grafischen Setup-Tool oder in der Kommandozeile geschehen. Das grafische Tool finden Sie in der Startleiste unter Settings (Einstellungen, Abb. 1.9). Das

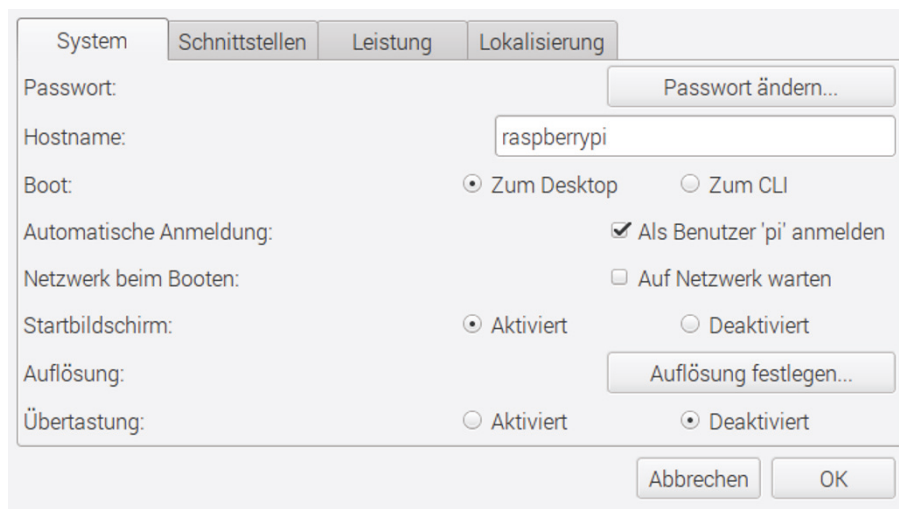


Abbildung 1.9: Das grafische Konfigurationsprogramm für den Raspberry Pi.

Konfigurationsprogramm kann mit der Maus bedient werden. Schauen Sie sich in Ruhe alle Einstellungen an. Keine Angst, sollten Sie etwas vergessen haben oder doch eine andere Einstellung wünschen, können Sie das Konfigurationsprogramm jederzeit wieder aufrufen.

Alternativ zum grafischen Konfigurationsprogramm können Sie die Einstellungen auch in einem Terminal vornehmen. Rufen Sie hierzu ein Terminal (Kapitel 1.4) auf und geben Sie den Befehl

Code-Ausschnitt 1.3.1

```
1 sudo raspi-config
```

ein. Daraufhin öffnet sich das Konfigurationsprogramm im Textmodus (Abb. 1.10). Führen Sie

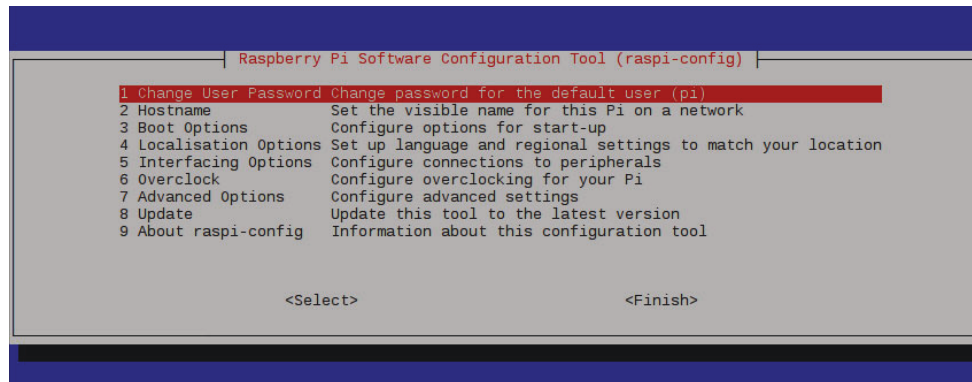


Abbildung 1.10: Der erste Startbildschirm des Raspbian-Betriebssystems und seine Einstellmöglichkeiten.

auch hier alle Einstellungen der Reihe nach durch. Sie können sich dabei mit Hilfe der Pfeiltasten \uparrow und \downarrow durch das Menü bewegen. Die *TAB*-Taste wechselt zwischen dem Menü und *Select* und *Finish*. *RETURN* wählt *Select*, *Finish* oder den entsprechenden Menüpunkt aus. Im Folgenden werden alle Einträge der Einstellungen erklärt.

1. Das Benutzerkennwort ändern (*Change User Password*):

- Geben Sie hier Ihr Benutzerkennwort ein oder ändern Sie Ihr Benutzerkennwort für den Hauptbenutzer mit dem Namen *pi* hier.



Das Benutzerkennwort sollte sicher sein, wenn Sie Ihr System - wie wir es später machen werden - ins Internet einbinden. Sichere Kennwörter sollten Groß- und Kleinbuchstaben, Sonderzeichen und Zahlen enthalten und nicht im Lexikon zu finden sein (z. B. *45!#l@1jH*).

Der Standard-Benutzer in Raspbian heißt *pi* und hat das Standard-Kennwort *raspberrypi*.

2. Netzwerk-Optionen (*Network Options*):

In diesem Untermenü können Sie den Rechnernamen des Raspebrry Pi festlegen (*hostname*), den Namen und das Kennwort einer WLAN-Verbindung (*WIFI SSID*) angeben, sowie Netzwerk-Interfaces (de)aktivieren.

- **Hostname:** Der Hostname ist der Name des Raspberry Pi. Unter diesem Namen wird Ihr Embedded Board später neben seiner IP-Adresse in Ihrem Netzwerk erreichbar sein.

3. Boot-Einstellungen (*Boot Options*):

- Diese Einstellung entscheidet darüber, ob Ihr System Sie nach dem Start zukünftig mit einer nackten Konsole oder dem X-Windows-System, einer grafischen Oberfläche, begrüßt.

Der Start in die Konsole empfiehlt sich, falls das System als Server läuft oder Sie alle Reserven (beispielsweise zum Start eines Medienzentrums) benötigen. Von der Konsole aus können Sie jederzeit die grafische Oberfläche mit dem Befehl

Code-Ausschnitt 1.3.2

```
1 startx
```

starten. Sollten Sie sich für die graphische Oberfläche als Startpunkt entscheiden, können Sie diese jederzeit durch Drücken der Tastenkombination *CTRL-ALT-BACKSPACE* beenden, sofern Sie das unter dieser Einstellung erlaubt haben. Sie sehen also: Egal, wie Sie sich entscheiden, ein Wechsel ist jederzeit möglich.

Weiterhin können Sie auswählen, ob beim Starten des Betriebssystems auf das Netzwerk gewartet wird oder nicht und ob der Plymouth Splash-Screen angezeigt wird oder nicht.

4. Lokalisierungs-Optionen (*Localisation Options*):

Das Lokalisierungs-Untermenü erlaubt Ihnen, das Tastaturlayout zu ändern, die Zeitzone und die Sprache zu ändern oder den WIFI Ländercode einzustellen. Die Default-Einstellungen stehen auf *British* bzw. auf *GB*. Ändern Sie das auf *German* bzw. *DE* oder auf ein Land und einen Ländercode Ihrer Wahl.

- Ändern der Lokale (*Change locale*): Wählen Sie für Deutschland die Einstellung *de_DE.UTF8*.
- Ändern der Zeitzone (*Change Time Zone*): Wählen Sie für Deutschland *Europe* → *Berlin*.
- Ändern des Tastatur-Layouts (*Change Keyboard Layout*): Wählen Sie für eine deutsche Tastatur *de_DE*. Unter Umständen ist ein Neustart erforderlich, bevor die Änderungen greifen.
- Ändern der WIFI-Länderkennung (*Change WiFi Country*): Diese Option erlaubt das Setzen der WIFI-Länderkennung. Wählen Sie hier *DE* für Deutschland.

5. Interface-Optionen (*Interfacing Options*):

In diesem Untermenü können Sie folgende Dinge (de)aktivieren: Kamera, ssh-Zugang, VNC-Server, I2C-Bus, serielle Konsole, 1-wire-Interface und die GPIO-Pins, beispielsweise zum Anschluss eines Infrarotempfängers.

- Kamera (*Camera*): Wenn Sie eine Raspberry Pi Kamera als Zubehör besitzen, können Sie hier das Kamera-CSI-Interface aktivieren. Näheres zum Thema Raspberry Pi und Kamera finden Sie im Kapitel 9.4.
- SSH: Aktivieren Sie bitte den Secure Shell Zugang. Hierdurch können Sie bequem von außen auf Ihren Raspberry Pi zugreifen (Kapitel 2.3.1).
- VNC: Dieser Eintrag (de)aktiviert den RealVNC Virtual Network Computing Server.
- SPI: Das Serial Peripheral Interface dient dem Anbinden externer Komponenten, wie beispielsweise eines Displays. Aktivieren Sie diesen Eintrag, wird das entsprechende Kernel-Modul geladen und damit zur Verfügung gestellt. Bitte aktivieren Sie diesen Eintrag, falls Sie einen IR-Empfänger an Ihrem Raspberry Pi nutzen möchten.
- I2C: Dieser Eintrag (de)aktiviert das I2C-Bus Kernel-Modul.
- Serial: Dieser Eintrag (de)aktiviert die Konsolen- und Kernel-Ausgabe über eine serielle Verbindung. Die Aktivierung ist beispielsweise für Debugging-Zwecke erforderlich.
- 1-wire: Dieser Eintrag (de)aktiviert das Dalles 1-wire-Interface. Dieses wird z. B. für den DS18B20 Temperatursensor benötigt.

6. Übertakten (*Overclock*):

- Die alten Raspberry Pi kommen standardmäßig mit einer mageren 700 MHz Taktfrequenz daher. Das ist auf der einen Seite sehr stromsparend, sorgt aber beispielsweise dafür, dass Programme relativ langsam ausgeführt werden. Diese Rubrik lädt zum Spielen und Experimentieren ein. Nachdem Sie den Hinweis bestätigt haben, dass eine zu große Übertaktung die Lebenserwartung Ihres Pi reduzieren kann, stehen Ihnen die Optionen *Keine Übertaktung (None)*, *Leichte Übertaktung (Modest)*, *Mittlere Übertaktung (Medium)*, *Hohe Übertaktung (High)* und *Turbo* zur Verfügung. Eine mittlere Übertaktung sollte auf jeden Fall funktionieren.

Info Bei der *Turbo*-Übertaktung habe ich häufig Lesefehler auf der SD-Karte beobachtet. Im Internet wird je nach Karte und Gerät (englische oder chinesische Produktion) von unterschiedlichen Erfolgen berichtet. Seit Ende 2013 gibt es einen neueren Kernel, der das SD-Kartenproblem nicht mehr haben soll. Die Einstellungen, die Sie hier vornehmen, können Sie später in der Datei `/boot/config.txt` für alle Einstellungsmöglichkeiten separat vornehmen. Rufen Sie hierzu das Kommando

Code-Ausschnitt 1.3.3

```
1 sudo vi /boot/config.txt
```

auf und ändern Sie die entsprechenden Einträge (Kapitel 2.7.3).

Sollte es während des Boot-Vorganges zu Problemen kommen, die auf eine zu große Übertaktung zurückzuführen sind, können Sie die *Shift*-Taste Ihrer Tastatur während des Bootens gedrückt halten. Damit wird vorübergehend die Übertaktung ausgeschaltet. Die Seite http://elinux.org/RPi_Overclocking hat dazu weitere Informationen.

7. Advanced Options

- Das Dateisystem erweitern (*Expand Filesystem*): Wählen Sie diesen Menüpunkt aus, um die volle Kapazität der Speicherkarte für das Dateisystem zur Verfügung zu stellen. Das Raspbian-Image hat eine Größe von ca. 1,7 GB und belegt damit auch auf der SD-Karte nur 1,7 GB. Auf einer 8 GB SD-Karte würden Sie damit 6,3 GB verschenken, die nicht für das Dateisystem zur Verfügung stünden. Nachdem Sie diesen Menüpunkt aufgerufen haben, vergrößert das Betriebssystem die 1,7 GB-Partition auf die volle Größe Ihrer SD-Karte. Diese steht Ihnen nach einem Neustart des Systems, dem sog. *reboot* zur Verfügung.

Info Starten Sie Ihr System auf keinen Fall durch Ziehen des Netzkabels neu. Dies kann einen Schaden auf dem Dateisystem und damit Datenverlust verursachen. Von der Konsole aus können Sie das System durch den folgenden Aufruf neu starten:

Code-Ausschnitt 1.3.4

```
1 sudo reboot
```

In Kapitel 3 lernen Sie, wie man von außen auf das System zugreifen kann. Dieser Zugriff kann ebenfalls dazu benutzt werden, das System neu zu starten.

- Overscan: Hier können Sie einen Overscan einstellen und somit den sichtbaren Bereich des Bildes ändern. Diese Option ist vor allem bei Fernsehern sinnvoll.
- Aufteilung des Speichers (*Memory Split*): Der neue Raspberry Pi besitzt 1 GB Hauptspeicher. Diesen können Sie zwischen Prozessor und GPU aufteilen. Nutzen Sie keine grafischen Anwendungen, können Sie den Speicher fast ausschließlich der CPU zuweisen.


Möchten Sie jedoch den Pi als Mediacenter benutzen, empfiehlt sich eine Aufteilung von 128-256 MB für die GPU und der Rest für die CPU. Mit dieser Einstellung funktionieren alle Projekte, die in diesem Buch vorgestellt werden.

- Audio: Hier können Sie einstellen, ob die Audioausgabe digital über HDMI oder analog (3,5 mm Buchse) erfolgen soll.
 - Auslösung (*Resolution*): Dieser Eintrag legt die Standardauflösung für HDMI/DVI fest, wenn der Raspberry Pi ohne angeschlossenen Monitor bootet. Diese Einstellung kann Einfluss auf RealVNC haben, wenn die VNC-Option aktiviert ist.
 - Verdopplung der Pixel (*Pixel Doubling*): Dieser Eintrag (de)aktiviert das 2x2 Pixel Mapping. Wenn dieses Mapping eingeschaltet ist, wird das ausgegebene Bild in doppelter Größe dargestellt. Das kann beispielsweise bei hochauflösenden Fernsehern hilfreich sein, wenn diese als Monitor genutzt werden.
 - GL Treiber (*GL Driver*): Dieser Eintrag (de)aktiviert den experimentellen OpenGL (Open Graphics Library) Desktop Grafiktreiber.
 - GL (Full KMS): Dieser Eintrag (de)aktiviert den experimentellen OpenGL Full KMS (kernel mode setting) Desktop Grafiktreiber.
 - GL (Fake KMS): Dieser Eintrag (de)aktiviert den experimentellen OpenGL Fake KMS (kernel mode setting) Desktop Grafiktreiber.
 - Legacy: Dieser Eintrag (de)aktiviert den originalen nicht OpenGL Videocore Desktop Grafiktreiber.
8. Update: Dieser Eintrag bringt das hier besprochene Konfigurations-Tool auf den neuesten Stand.
9. Über dieses Tool (*About raspi-config*): Dieser Eintrag listet lediglich einige Informationen bezüglich des hier besprochenen Konfigurations-Tools auf.

Quittieren Sie Änderungen innerhalb des *raspi-config*-Tools bitte mit Abbrechen (*Cancel*), wenn Sie nicht möchten, dass Ihre Änderungen übernommen werden. Andernfalls verlassen Sie das Programm bitte mit *OK*. Je nach ausgewählter Änderung kann ein Neustart notwendig sein.


1.3.1 Netzwerkeinstellungen

Die neuen Raspberry Pi verfügen über eine Ethernet-Buchse und über einen kombinierten WLAN/Bluetooth-Chip. Kabelgebunden nimmt der Raspberry Pi seine Netzwerk-IP per DHCP (*Dynamic Host Configuration Protocol*) entgegen. Im Kapitel 2.4.1 zeige ich Ihnen, wie Sie den Pi auf eine feste IP-Adresse einstellen können.

Möchten Sie Ihren Raspberry Pi per WLAN ins Netz hängen, muss zuvor der Wifi Ländercode eingegeben worden sein (DE). Klicken Sie zur Auswahl des WLAN-Netzes auf das Netzwerk-Symbol, das sich zwischen Bluetooth-Symbol und Lautsprecher-Symbol in der oberen Leiste des Desktops befindet. Zwei rote Kreuze  zeigen an, dass zunächst weder eine ausgehende noch eine eingehende Netzwerkverbindung existieren. Der Klick mit der Maus öffnet eine Liste mit WLAN-Netzwerken, die zur Verfügung stehen. Wählen Sie das Netzwerk aus, mit dem Sie Ihren Pi verbinden möchten und geben Sie gegebenenfalls das erforderliche Kennwort ein.



Falls Ihre Tastatur noch nicht auf Deutsch umgestellt ist, kann es sein, dass Sie Zeichen eintippen, die nicht dem gewünschten Kennwort entsprechen.

Der Raspberry Pi signalisiert eine erfolgreiche WLAN-Verbindung mit einem blauen WLAN Icon . Falls Sie eine minimale Betriebssystem-Version ohne Desktop bevorzugen, zeige ich Ihnen in Kapitel 1.5, wie Sie WLAN-Einstellungen ohne grafische Oberfläche vornehmen können.

1.4 Erstes Video und MPEG-2 Codec

Nachdem Sie alle Einstellungen vorgenommen haben und den Raspberry Pi neu gestartet haben, ist es nun an der Zeit, dem Board ein erstes Video zu entlocken um zu zeigen, welches Potenzial in der kleinen Platine steckt. Von Hause aus kann der Raspberry Pi H264-kodiertes Material ohne zusätzlichen Codec abspielen. Später beschreibe ich, wie Sie eine MPEG-2-Lizenz für ca. 3 € erwerben und einbinden können.

Aber zunächst einmal schauen wir uns die grafische Oberfläche an. Nach einem Neustart sehen Sie diese bereits (sofern Sie unter Einstellungen angegeben haben, dass der Pi in diese Oberfläche starten soll), oder Sie rufen sie mit dem *startx*-Befehl auf. Sollte der Pi die Konsole anzeigen, müssen Sie sich zuvor noch am System anmelden. Dazu geben Sie bitte den Benutzernamen *pi* ein und das Kennwort, das Sie im letzten Abschnitt dieses Buches vergeben haben. Dieser Abschnitt geht davon aus, dass Ihr Pi bereits mit einem Netzwerk verbunden ist. Die Netzwerkoptionen werden im nächsten Kapitel detailliert beschrieben.

Info Falls Sie in die Konsole gebootet haben, können Sie die Konsole mit *ALT-Fx* wechseln, wobei *x* für *1*, *2* usw. steht. Mit *ALT-F2* wechseln Sie also in die zweite Konsole, mit *ALT-F1* zurück in die erste.

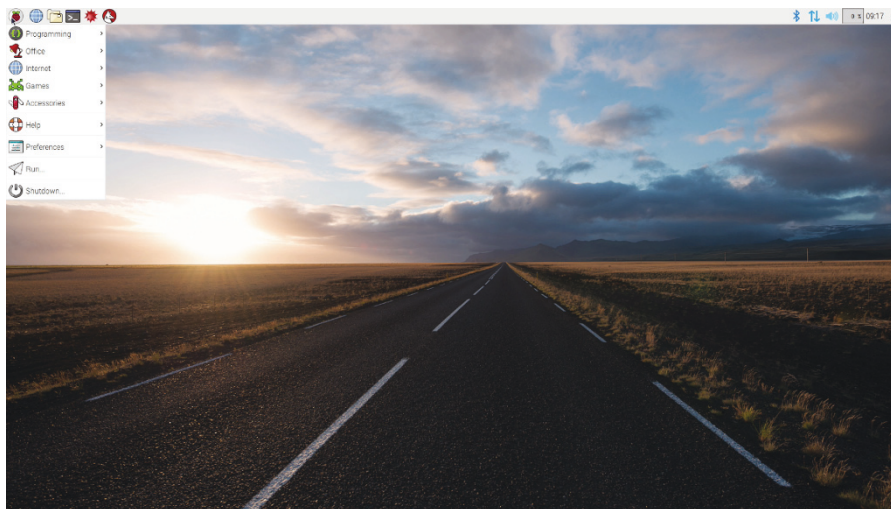


Abbildung 1.11: Die Raspbian X11-Oberfläche.

Ihr Bildschirm sollte nun so ähnlich aussehen wie in der Abb. 1.11. Der Schreibtisch „Pixel (Pi Improved Xwindow Environment Lightweight)“ (auf Englisch *Desktop*) beinhaltet am oberen linken Rand einen Programmstarter ähnlich dem von Microsoft Windows™, mit dem Sie Programme verschiedener Kategorien starten können. Weiterhin befinden sich in der oberen Leiste einige Icons einiger Programme, die durch einen einfachen Mausklick gestartet werden können. An dieser Stelle seien der Webbrowser *Chromium* und das Eingabeterminal *LXTerminal* erwähnt. Beides benötigen wir gleich, um ein Video aus dem Netz zu laden und wiederzugeben. Weitere Icons zeigen beispielsweise Uhrzeit oder CPU Aktivität und erlauben das Einstellen von Netzwerk- oder Bluetoothverbindungen.

Starten Sie nun den Chromium-Webbrowser durch einen Klick auf das blaue Webbrowser-Icon. Danach benötigen Sie ein wenig Zeit, bis sich das Browserfenster öffnet. Denken Sie daran, 700 MHz-1,4 GHz sind kein Düsenjäger. Es dauert eine Weile, bis sich das Browserfenster geöffnet hat und Sie die Websuche nach einem Video starten können. Navigieren Sie zur Seite <http://www.h264info.com/clips.html>.

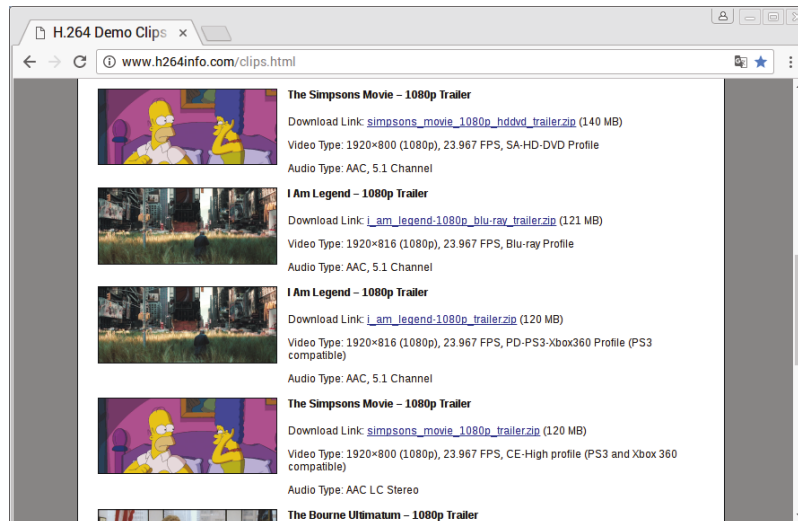


Abbildung 1.12: Mit dem Chromium-Webbrowser kann man im Internet surfen.

Sie enthält einige in H264-kodierte Full-HD Videos. Suchen Sie sich eines aus und starten sie den Download. Wie wäre es mit `i_am_legend-1080p_trailer.zip` (Abb. 1.12)? Falls Sie kein an-



Abbildung 1.13: Das LXTerminal ist eine Konsole zur Kommandoeingabe

deres Verzeichnis angeben (*Speichern unter*), wird die Datei im sogenannten *Home*-Verzeichnis des Benutzers `pi` gespeichert. Die Datei ist noch mit dem Programm `zip` gepackt. Diese Art der Dateien kennen Sie sicher von Windows-Systemen. Bevor wir den Film starten können, packen wir ihn aus. Dazu öffnen Sie bitte das *LXTerminal* mit einem Mausklick (Abb. 1.13), installieren das Programm `unzip` und packen die Datei aus.

Code-Ausschnitt 1.4.1

```
1 sudo apt-get update
2 sudo apt-get install unzip
3 unzip i_am_legend-1080p_blu-ray_trailer.zip
```

Danach befinden sich die Dateien *I Am Legend - Trailer.mp4* und *readme.txt* in Ihrem *home*-Verzeichnis.



Unter UNIX-Systemen können Sie übrigens viel Tipp-Arbeit sparen. Die **TAB**-Taste hilft Ihnen dabei. Geben Sie im oberen Beispiel einfach einmal *unzip i* ein und drücken Sie dann die **TAB**-Taste. Der Rest des Dateinamens wird automatisch ergänzt. Gibt es mehrere Möglichkeiten zur Ergänzung, zeigt ein wiederholtes Drücken der **TAB**-Taste alle Möglichkeiten an. Tippen Sie weitere Buchstaben ein, bis das Wort eindeutig ist. **TAB** wird es dann vervollständigen.

Nun wird es aber Zeit, unseren Film zu starten. Der Abspieler auf dem Raspberry Pi heißt *omxplayer* und ist bereits standardmäßig installiert. Rufen Sie ihn auf und hängen Sie den Namen des Films als Parameter an:

Code-Ausschnitt 1.4.2

```
1 omxplayer I\ Am\ Legend\ -\ Trailer.mp4
```

Erinnern Sie sich? *omxplayer I* gefolgt von der **TAB**-Taste reicht aus. Beachten Sie, dass die Leerzeichen im Dateinamen mit einem Backslash (\) maskiert sind. Bestätigen Sie mit **RETURN** - Voilà. Der Film sollte wiedergegeben werden. Beeindruckend, oder? Ruckelfreies, hochauflösendes HD-Video auf einem Mini-Board, für das auf manchem PC noch ein 2 GHz Dual-Core-Prozessor erforderlich wäre.

Sollten Sie kein Audio hören, ist es evtl. erforderlich, das Audio-Ausgabe-Device beim Aufruf festzulegen. Der Aufruf von *omxplayer -h* zeigt alle Einstellmöglichkeiten:

Code-Ausschnitt 1.4.3

```
1  -h --help           Print this help
2  -v --version       Print version info
3  -k --keys          Print key bindings
4  -n --aidx index    Audio stream index : e.g. 1
5  -o --adev device   Audio out device    : e.g. hdmi/local/both/alsa[:device]
6  -i --info          Dump stream format and exit
7  -I --with-info     dump stream format before playback
8  -s --stats         Pts and buffer stats
9  -p --passthrough  Audio passthrough
10 -d --deinterlace   Force deinterlacing
11     --nodeinterlace Force no deinterlacing
12     --nativedeinterlace let display handle interlace
13     --anaglyph type  convert 3d to anaglyph
14     --advanced[=0]   Enable/disable advanced deinterlace for HD videos (default ←
    enabled)
15 -w --hw            Hw audio decoding
16 -3 --3d mode       Switch tv into 3d mode (e.g. SBS/TB)
17 -M --allow-mvc     Allow decoding of both views of MVC stereo stream
18 -y --hdmiclocksyc Display refresh rate to match video (default)
19 -z --nohdmiclocksyc Do not adjust display refresh rate to match video
20 -t --sid index     Show subtitle with index
21 -r --refresh       Adjust framerate/resolution to video
22 -g --genlog        Generate log file
23 -l --pos n         Start position (hh:mm:ss)
24 -b --blank[=0xAARRGGBB] Set the video background color to black (or optional ARGB ←
    value)
25     --loop           Loop file. Ignored if file not seekable
26     --no-boost-on-downmix Don't boost volume when downmixing
27     --vol n          set initial volume in millibels (default 0)
28     --amp n          set initial amplification in millibels (default 0)
29     --no-osd         Do not display status information on screen
30     --no-keys        Disable keyboard input (prevents hangs for certain TTYs)
31     --subtitles path External subtitles in UTF-8 srt format
32     --font path      Default: /usr/share/fonts/truetype/freefont/FreeSans.ttf
33     --italic-font path Default: /usr/share/fonts/truetype/freefont/FreeSansOblique. ←
    ttf
34     --font-size size  Font size in 1/1000 screen height (default: 55)
35     --align left/center Subtitle alignment (default: left)
```



```

36     --no-ghost-box          No semitransparent boxes behind subtitles
37     --lines n              Number of lines in the subtitle buffer (default: 3)
38     --win 'x1 y1 x2 y2'    Set position of video window
39     --win x1,y1,x2,y2      Set position of video window
40     --crop 'x1 y1 x2 y2'   Set crop area for input video
41     --crop x1,y1,x2,y2     Set crop area for input video
42     --aspect-mode type     Letterbox, fill, stretch. Default: stretch if win is ←
                             specified, letterbox otherwise
43     --audio_fifo n         Size of audio output fifo in seconds
44     --video_fifo n         Size of video output fifo in MB
45     --audio_queue n        Size of audio input queue in MB
46     --video_queue n        Size of video input queue in MB
47     --threshold n          Amount of buffered data required to finish buffering [s]
48     --timeout n            Timeout for stalled file/network operations (default 10s)
49     --orientation n        Set orientation of video (0, 90, 180 or 270)
50     --fps n                Set fps of video where timestamps are not present
51     --live                  Set for live tv or vod type stream
52     --layout                Set output speaker layout (e.g. 5.1)
53     --dbus_name name       default: org.mpris.MediaPlayer2.omxplayer
54     --key-config <file>    Uses key bindings in <file> instead of the default
55     --alpha                 Set video transparency (0..255)
56     --layer n              Set video render layer number (higher numbers are on top)
57     --display n            Set display to output to
58     --cookie 'cookie'      Send specified cookie as part of HTTP requests
59     --user-agent 'ua'      Send specified User-Agent as part of HTTP requests
60     --lavfdopts 'opts'     Options passed to libavformat, e.g. 'probesize:250000,...'
61     --avdict 'opts'        Options passed to demuxer, e.g., 'rtsp_transport:tcp,...'

63 For example:

65     ./omxplayer -p -o hdmi test.mkv

```

Der von uns gesuchte Aufruf lautet also

Code-Ausschnitt 1.4.4

```
1 omxplayer -o local I\ Am\ Legend\ -\ Trailer.mp4
```

für die analoge Audioausgabe oder

Code-Ausschnitt 1.4.5

```
1 omxplayer -o hdmi I\ Am\ Legend\ -\ Trailer.mp4
```

für die digitale Audioausgabe über HDMI.

Als Nächstes machen wir uns daran, den Pi mit einer MPEG-2 Lizenz auszustatten. Die ist erforderlich, wenn Sie DVDs schauen möchten oder den Pi als Mediacenter einrichten, um fernzusehen. Da die CPU des alten Pi zu schwach ist, um MPEG-2 in Software zu dekodieren, stellt die Raspberry Pi Foundation für ca. 3 € einen Schlüssel zur Verfügung, der die Hardware-Beschleunigung für MPEG-2 aktiviert. Gut angelegtes Geld, wenn man den Pi als Mediacenter betreiben möchte. Für den Kauf rufen Sie bitte in einem Browser Ihrer Wahl die Seite <http://www.raspberrypi.com/mpeg-2-license-key/> auf. In das Browserfeld Seriennummer (*Serial number*) tragen Sie bitte die Seriennummer Ihres Raspberry Pi ein. Diese erhalten Sie, indem Sie ein LXTerminal öffnen und folgenden Befehl eingeben:

Code-Ausschnitt 1.4.6

```
1 cat /proc/cpuinfo
```

Dieser Befehl zeigt alles an, was auf die CPU des Raspberry Pi bezogen ist. Bei mir sieht die Ausgabe wie folgt aus:

Code-Ausschnitt 1.4.7

```
1 processor : 0
2 model name : ARMv6-compatible processor rev 7 (v6l)
3 BogoMIPS : 2.00
4 Features : swp half thumb fastmult vfp edsp java tls
5 CPU implementer : 0x41
6 CPU architecture: 7
7 CPU variant : 0x0
8 CPU part : 0xb76
9 CPU revision : 7

11 Hardware : BCM2708
12 Revision : 000d
13 Serial : 0000000062721864
```

Die Seriennummer ist der letzte Eintrag, in meinem Fall also *0000000062721864*. Der Raspberry Pi Store liefert Ihnen an Ihre Email-Adresse innerhalb von 72 Stunden einen Key in einer Email. Dieser Key sieht wie folgt aus:

Code-Ausschnitt 1.4.8

```
1 decode_MPG2=0x91f22de0
```

Diese Zeile muss in die Datei */boot/config.txt* eingetragen werden. Im Moment empfehle ich Ihnen, das unter Windows zu erledigen. Wenn Sie die SD-Karte in einen Kartenleser einschieben, lädt Windows automatisch die FAT-32 Partition und die Datei *config.txt* im *boot*-Verzeichnis ist les- und schreibbar. Im nächsten Kapitel stelle ich Ihnen einen Texteditor vor, mit dem Sie das Ändern auch bequem unter UNIX machen können. Es spielt übrigens keine Rolle, an welcher Stelle der Datei Sie die Lizenz eintragen. Nach einem Neustart, der unbedingt erforderlich ist, steht Ihnen die Lizenz zur Verfügung.



Denken Sie bitte daran, die SD-Karte nicht einfach aus dem Kartenslot zu entfernen, ohne das System anzuhalten. Dies können Sie erreichen mit

Code-Ausschnitt 1.4.9

```
1 sudo halt -p
```

Erst, wenn ausschließlich die rote Power-LED leuchtet, ist das System heruntergefahren und Sie können die SD-Karte gefahrlos entnehmen. Zum Neustart können Sie dann kurz die Stromversorgung unterbrechen und wieder zuführen. Das korrekte Funktionieren der Lizenz können Sie entweder nach einem Neustart durch Abspielen einer MPEG-2-Datei überprüfen oder durch Aufruf von

Code-Ausschnitt 1.4.10

```
1 vcgencmd codec_enabled MPG2
```

Die korrekte Antwort des Systems sollte *MPG2=enabled* lauten. Ist der Codec aktiviert, können Sie MPEG-2-kodiertes Material wiedergeben, wie beispielsweise SD-aufgelöste Fernsehprogramme oder nicht kopiergeschützte DVDs. Mehr zum Thema DVD erfahren Sie im Kapitel 6.2.

Auf dieselbe Art und Weise können Sie auch eine Lizenz für VC-1 erwerben und installieren. Dieser Codec wird für ein proprietäres Microsoft Videoformat verwendet und findet sich heute auf Bluray-Discs oder HD-DVD. Letztere gibt es heute nicht mehr.

1.5 Weniger kann mehr sein

Dieser Abschnitt gibt Ihnen Hinweise zum Raspbian Lite-Betriebssystem und erklärt, wie man den Raspberry Pi ohne Micro-SD-Karte von einem externen Laufwerk oder sogar über ein Netzwerk starten kann.

1.5.1 Raspbian Lite

Raspbian Lite ist ein minimales Debian Linux, welches 2018 in der Version „Stretch“ vorliegt. Die Lite-Version bietet keinerlei grafische Oberfläche und kommt mit erheblich weniger Speicherbedarf aus. Statt 4,6 GB werden hier in der Basis-Version lediglich 1.8 GB SD-Karten-Speicherplatz fällig. Die Lite-Version finden Sie ebenfalls zum Download auf <https://www.raspberrypi.org/downloads>.

Info Die Lite-Konsole startet standardmäßig mit einem englischen/amerikanischen Tastaturreiber. Auf dieser Tastatur sind die Buchstaben *Z* und *Y* im Vergleich zu einer deutschen Tastatur vertauscht. Wenn Sie sich also an der Konsole anmelden möchten, funktioniert zwar die Eingabe des Benutzers *pi*. Bei der Eingabe des Kennworts *raspberrry* aber übergibt der Tastaturreiber das Wort *raspberrz*, was falsch ist. Bitte geben Sie hier *raspberrz* als Kennwort ein, welches dann korrekterweise als *raspberrry* interpretiert wird und stellen Sie anschließend, wie oben beschrieben, den Tastaturreiber auf Deutsch um.

Info Sie können auch die Lite-Version zu einer Desktop-Version aufbohren durch Eingabe des Kommandos

Code-Ausschnitt 1.5.1

```
1 sudo apt-get install lightdm
```

WLAN Konfiguration lite

Das Raspbian Lite-Betriebssystem bietet keinerlei grafische Installations-Tools. Während die Ethernet-Verbindung per DHCP hier ebenfalls ohne Konfiguration funktioniert, gilt dies nicht für die WLAN-Einstellungen. Nehmen Sie die Einstellungen für das WLAN in der Konsole vor, indem Sie

Code-Ausschnitt 1.5.2

```
1 sudo raspi-config
```

aufrufen und wählen Sie unter *Network Options Wi-fi* aus. Sie können dann den Namen des WLAN-Netzwerkes eingeben und ein evtl. benötigtes Kennwort. Bitte achten Sie auch hier darauf, dass die WIFI-Kennung auf *DE* steht. Die geänderten Netzwerkeinstellungen stehen nach einem Neustart zur Verfügung.

1.5.2 Systemstart von einem externen Laufwerk

In diesem Abschnitt zeige ich Ihnen, wie Sie einen Raspberry Pi 3(B+) von einem externen Laufwerk booten können. Das kann entweder ein USB-Stick sein oder auch eine externe Festplatte. Die Raspberry Pi-Foundation bezeichnet dieses Feature selbst aber noch als experimentell, da es nicht mit allen USB-Geräten funktioniert. Grund hierfür ist der auf 32 KB begrenzte Speicherplatz für das Boot-ROM. Das hier angegebene Vorgehen funktioniert nicht mit einem Raspberry Pi 1, 2 oder Zero (W).

In einem ersten Schritt muss der USB Boot-Modus programmiert werden. Dies geschieht durch Setzen eines Bits im OTP (*One Time Programmable*) des Raspberry Pi.

Der OTP ist ein Speicherbereich, der nur einmal programmiert werden kann und sich dann nicht mehr ändern lässt. Das gesetzte Bit schaltet das Booten des Systems von einem unterstützten USB-Laufwerk frei. Bei einem kompatiblen Device ist danach zum Starten des Kleinrechners keine SD-Karte mehr erforderlich. Das OTP-Bit muss allerdings mit Hilfe einer SD-Karte programmiert werden. Diese muss eine Raspbian- (oder Raspbian Lite-) Distribution enthalten.

Bitte aktualisieren Sie in einem ersten Schritt das System, indem Sie in einem Terminal die folgenden Befehle aufrufen:

Code-Ausschnitt 1.5.3

```
1 sudo apt-get update
2 sudo apt-get upgrade
```

Anschließend muss das Setzen des OTP in der Datei `config.txt` hinterlegt werden, was mit Hilfe des Befehls

Code-Ausschnitt 1.5.4

```
1 echo program_usb_boot_mode=1 | sudo tee -a /boot/config.txt
```

erledigt werden kann. Dieser Befehl fügt die Zeile `program_usb_boot_mode=1` am Ende der Konfigurationsdatei ein.

Starten Sie anschließend den Raspberry Pi neu mit dem Kommando

Code-Ausschnitt 1.5.5

```
1 sudo reboot
```

nach dem Neustart können Sie ebenfalls in einem Terminal überprüfen, ob das Setzen des OTP erfolgreich war. Der Aufruf von

Code-Ausschnitt 1.5.6

```
1 vcgencmd otp_dump | grep 17:
2 17:3020000a
```

sollte als Ausgabe `0x3020000a` zeigen. Falls nicht, hat das Programmieren des OTP nicht funktioniert.

Sie können nun die letzte Zeile mit `program_usb_boot_mode` wieder aus der Datei `/boot/config.txt` entfernen, um nicht aus Versehen das OTP eines anderen Raspberry Pi zu programmieren, falls Sie Ihre SD-Karte anderweitig verwenden.

Das Übertragen eines Images auf das USB-Device, von dem Sie booten möchten, funktioniert übrigens genau so, wie es oben am Beispiel einer SD-Karte beschrieben wurde. Nachdem Sie das Image auf das externe USB-Laufwerk geschrieben haben, entfernen Sie es bitte vom Computer, mit dem Sie geschrieben haben und stecken es in einen der USB-Anschlüsse Ihres Raspberry Pi 3. Es kann 5 bis 10 Sekunden dauern, bis der Raspberry Pi 3 den Bootvorgang beginnt.

1.5.3 Netzwerk-Boot

Dieser Abschnitt zeigt Ihnen, wie Sie einen eigenen DHCP/TFTP-Server (*Trivial File Transfer Protocol*) aufsetzen können, der es Ihnen erlaubt, einen Raspberry Pi von einem Netzwerk zu booten. Die hierzu minimal erforderliche Version ist der Raspberry Pi 3.

Die Anleitung geht davon aus, dass ein Raspberry Pi als Server in einem vorhandenen Heim-Netzwerk benutzt wird und dass ein Raspberry Pi (3 oder 3B+) als Klient genutzt wird. Lediglich der Raspberry Pi, welcher als Server genutzt wird, benötigt eine SD-Karte. Der Klient startet über das Netzwerk.

Dieses Unterkapitel nutzt viele Begriffe, die erst im weiteren Verlauf des Buches erklärt werden. Sie können das Kapitel daher gerne überspringen und zu einem späteren Zeitpunkt wieder aufgreifen, falls Sie Ihren Pi an dieser Stelle (noch) nicht von einem Netzwerk aus booten möchten.

Einrichten des Klienten

Bevor Sie einen Raspberry Pi von einem Netzwerk booten können, muss der USB Boot-Modus aktiviert werden. Das Verfahren ist identisch mit dem Boot-Vorgang von einem externen USB-Laufwerk: Es wird ein Bit im OTP (*One Time Programmable*) Speicher gesetzt, welches das Booten über ein Netzwerk erlaubt. Hierzu ist einmalig eine SD-Karte erforderlich, die nach dem Setzen des Bits wieder entfernt werden kann. Installieren Sie also zunächst, wie oben beschrieben, eine Raspbian-Distribution auf einer SD-Karte.

Starten Sie Ihren Raspberry Pi mit dieser SD-Karte und sorgen Sie mit dem Kommando

Code-Ausschnitt 1.5.7

```
1 sudo apt-get update && sudo apt-get upgrade
```

dafür, dass das System aktuell ist. Wie oben bereits beschrieben, können Sie den USB Boot-Modus mit dem folgenden Kommando programmieren:

Code-Ausschnitt 1.5.8

```
1 echo program_usb_boot_mode=1 | sudo tee -a /boot/config.txt
```

Dieses Kommando fügt eine Zeile am Ende der Raspberry Pi Konfigurations-Datei */boot/config.txt* ein, die das USB Boot-Bit beim nächsten Booten setzt.

Starten Sie Ihren Raspberry Pi bitte neu. Dies kann entweder mit Hilfe der grafischen Oberfläche geschehen oder durch Eingabe des Kommandos

Code-Ausschnitt 1.5.9

```
1 sudo reboot
```

Nach einem erfolgreichen Neustart können Sie wieder mit dem Befehl

Code-Ausschnitt 1.5.10

```
1 vcgencmd otp_dump | grep 17:  
2 17:3020000a
```

überprüfen, ob der OTP-Programmiervorgang erfolgreich war oder nicht. Stellen Sie dabei wieder sicher, dass die Ausgabe *0x3020000a* korrekt ist. Damit ist die Einrichtung des Klienten nahezu abgeschlossen. Sie können die letzte Zeile nun wieder aus der Datei */boot/config.txt* entfernen, müssen es aber nicht. Wie bereits oben erwähnt, macht das Sinn, damit Sie nicht aus Versehen den OTP anderer Raspberry Pi beschreiben. Sie können diese Zeile mit Hilfe eines Editors entfernen, den ich Ihnen im nächsten Kapitel vorstelle.

Da sich die Datei *config.txt* aber auf der FAT32-Partition der SD-Karte befindet, können Sie auch einen WINDOWS-Editor zur Bearbeitung benutzen. Schalten Sie den Raspberry Pi danach mit dem Befehl

Code-Ausschnitt 1.5.11

```
1 sudo poweroff
```

aus.

Einrichten des Servers

Booten Sie bitte den Raspberry Pi, den Sie als Server benutzen wollen, mit Hilfe der SD-Karte. Und noch einmal: Wenn Sie das OTP dieses Pi nicht programmieren wollen, stellen Sie bitte sicher, dass die USB-Boot-Zeile aus der Datei *config.txt* gelöscht wurde.

Das Dateisystem der SD-Karte muss weiterhin mit Hilfe von *raspi-config* expandiert worden sein, so dass die gesamte SD-Karte zur Verfügung steht.

Diese Anleitung geht davon aus, dass der Raspberry Pi-Server per Ethernet mit dem Heimnetzwerk verbunden ist.

Der Klient-Raspberry Pi benötigt ein sogenanntes *root*-Dateisystem, um booten zu können. Dieses erstellen wir im Verzeichnis */nfs/client1* mit den Befehlen

Code-Ausschnitt 1.5.12

```
1 sudo mkdir -p /nfs/client1
2 sudo apt-get install rsync
3 sudo rsync -xa --progress --exclude /nfs / /nfs/client1
```

Der Befehl *mkdir -p* erstellt ein Verzeichnis und ignoriert ein eventuell schon vorhandenes Verzeichnis. In unserem Fall wird das Verzeichnis */nfs/client1* erstellt. Anschließend wird das Programm *rsync* installiert, welches im nächsten Schritt dazu genutzt wird, alle erforderlichen Dateien zu kopieren. Nun erstellen wir auf dem Server Schlüssel für eine SSH-Verbindung (Kapitel 2.3.1) mit dem Klienten. Der Befehl *chroot* steht für „change root“ und ist eine Funktion unter Unix-Systemen, um das Rootverzeichnis zu ändern. Hiermit wird eine sog. Sandbox erstellt, ein geschützter Bereich, in dem Programme gefahrlos ausgeführt werden können, ohne das Hauptdateisystem zu gefährden.

Code-Ausschnitt 1.5.13

```
1 cd /nfs/client1
2 sudo mount --bind /dev dev
3 sudo mount --bind /sys sys
4 sudo mount --bind /proc proc
5 sudo chroot .
6 rm /etc/ssh/ssh_host_*
7 dpkg-reconfigure openssh-server
8 exit
9 sudo umount dev
10 sudo umount sys
11 sudo umount proc
```

Im nächsten Schritt finden wir die Einstellungen unseres lokalen Netzwerkes heraus. Beginnen wir mit der Netzwerkadresse unseres Gateways. Das ist z. B. eine Fritzbox, die die Verbindung mit dem Internet herstellt.

Code-Ausschnitt 1.5.14

```
1 ip route | grep default | awk '{print $3}'
```

Danach rufen wir

Code-Ausschnitt 1.5.15

```
1 ip -4 addr show dev eth0 | grep inet
```

auf. Die Ausgabe sollte ähnlich aussehen wie *inet 192.168.178.66/24 brd 192.168.178.255 scope global eth0*. Die erste Adresse ist die Adresse des Raspberry Pi-Servers im Netzwerk, gefolgt von der Größe des Netzwerks, die in Ihrem Fall wahrscheinlich ebenfalls */24* ist. Schreiben Sie sich diese Adresse und auch die Broadcast-Adresse (*brd*) auf. Wir benötigen beide Adressen im weiteren Verlauf noch.

Weiterhin benötigen wir noch die Adresse des DNS-Servers, die der des Gateways entspricht. Sie können diese Adresse mit dem Kommando

Code-Ausschnitt 1.5.16

```
1 cat /etc/resolv.conf
```

herausfinden. Die IP-Adresse des Gateways befindet sich hinter dem Wort *nameserver* und lautet beispielsweise *192.168.178.1*. Um sicherzustellen, dass der Raspberry Pi, der als Server arbeitet, immer mit derselben IP-Adresse startet, ändern wir seine dynamische IP-Adresse in eine statische um. Dazu editieren wir die Datei */etc/network/interfaces* und ändern Zeile *iface eth0 inet manual*. Tragen Sie unter *address* die IP-Adresse des Servers auf dem vorletzten Kommando ein und unter *gateway* die IP-Adresse des Gateways aus dem letzten Kommando. die Netzwerkmaske *netmask* lautet *255.255.255.0*, wenn Sie nicht mehr als 256 Rechner in Ihrem Netzwerk haben. Die Datei sollte dann in etwas so aussehen:

Code-Ausschnitt 1.5.17

```
1 auto eth0
2 iface eth0 inet static
3     address 192.168.178.66
4     netmask 255.255.255.0
5     gateway 192.168.178.1
```

Deaktivieren Sie nun den DHCP-Klienten und schalten Sie dann die Standard-Netzwerkverbindungen unter Debian ein:

Code-Ausschnitt 1.5.18

```
1 sudo systemctl disable dhcpcd
2 sudo systemctl enable networking
```

Damit die Änderungen berücksichtigt werden, ist ein Neustart des Raspberry Pi-Servers erforderlich.

Code-Ausschnitt 1.5.19

```
1 sudo reboot
```

An dieser Stelle wird DNS (*Domain Name System*) nicht mehr funktionieren.

Wir müssen daher die IP-Adresse unseres Gateways in die Datei `/etc/resolv.conf` auf dem Raspberry Pi-Server eintragen, in dem wir das folgende Kommando aufrufen:

Code-Ausschnitt 1.5.20

```
1 echo "nameserver 192.168.178.1" | sudo tee -a /etc/resolv.conf
```

Bitte passen Sie die IP-Adresse entsprechend der Ihres Gateways an. Sorgen Sie nun dafür, dass die Datei `/etc/resolv.conf` nicht mehr verändert werden kann, was beispielsweise durch das Programm `dnsmasq` noch geschehen kann:

Code-Ausschnitt 1.5.21

```
1 sudo chmod +i /etc/resolv.conf
```

Installieren Sie nun die Programme `dnsmasq` und `tcpdump`, die wir im weiteren Verlauf dieser Beschreibung noch benötigen werden:

Code-Ausschnitt 1.5.22

```
1 sudo apt-get update
2 sudo apt-get install dnsmasq tcpdump
```

Da `dnsmasq` weiterhin versucht, die DNS Namensauflösung zu ändern, hindern wir es daran und starten den Server anschließend neu:

Code-Ausschnitt 1.5.23

```
1 sudo rm /etc/resolvconf/update.d/dnsmasq
2 sudo reboot
```

Im nächsten Schritt starten wir das Programm `tcpdump` auf dem Raspberry Pi-Server, damit wir nach DHCP-Paketen (Dynamic Host Configuration Protocol) des Raspberry Klienten suchen können, den wir über das Netzwerk booten wollen:

Code-Ausschnitt 1.5.24

```
1 sudo tcpdump -i eth0 port bootpc
```

Verbinden Sie nun den Raspberry Pi-Klienten per Ethernet mit dem Netzwerk und schließen Sie eine Spannungsversorgung an. Nach ungefähr 10 Sekunden sollten die LEDs am Klienten leuchten und der Server sollte Pakete mit dem Inhalt `DHCP/BOOTP, Request from ...` vom Klienten empfangen, beispielsweise `IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from b8:27:eb...` Nun ändern wir die `dnsmasq`-Konfiguration, damit DHCP auf die Klient-Anfragen antworten kann. Unterbrechen Sie bitte dazu auf dem Server das Programm `tcpdump` durch Drücken der Tastenkombination `CTRL-C` und geben dann ein:

Code-Ausschnitt 1.5.25

```
1 echo | sudo tee /etc/dnsmasq.conf
2 sudo vi /etc/dnsmasq.conf
```

Ersetzen Sie den Inhalt der Datei *dnsmasq.conf* mit:

Code-Ausschnitt 1.5.26

```
1 port=0
2 dhcp-range=192.168.178.255,proxy
3 log-dhcp
4 enable-tftp
5 tftp-root=/tftpboot
6 pxe-service=0,"Raspberry Pi Boot"
```

Für *dhcp-range* tragen Sie bitte die Broadcast-Adresse ein, die Sie sich zuvor notiert haben. Anschließend erstellen Sie bitte das Verzeichnis */tftpboot* und starten Sie *dnsmasq*:

Code-Ausschnitt 1.5.27

```
1 sudo mkdir /tftpboot
2 sudo chmod 777 /tftpboot
3 sudo systemctl enable dnsmasq.service
4 sudo systemctl restart dnsmasq.service
```

Schauen Sie sich die Log-Ausgaben von *dnsmasq* an:

Code-Ausschnitt 1.5.28

```
1 tail -F /var/log/daemon.log
```

Sie sollten so ähnlich aussehen wie *raspberrypi dnsmasq-tftp[1903]: file /tftpboot/bootcode.bin not found*. Als Nächstes kopieren wir den Bootcode in das *tftpboot*-Verzeichnis. Das sind die Dateien *bootcode.bin* und *start.elf*. Diese Dateien befinden sich im */boot*-Verzeichnis des Raspberry Pi-Servers. Da auch ein Kernel zum Booten benötigt wird, können wir das gesamte */boot*-Verzeichnis kopieren. Beenden Sie daher die Log-Ausgaben durch Drücken der Tastenkombination *CTRL-C* und rufen Sie das Kommando zum Kopieren auf:

Code-Ausschnitt 1.5.29

```
1 cp -r /boot/* /tftpboot
```

Starten Sie dann *dnsmasq* neu:

Code-Ausschnitt 1.5.30

```
1 sudo systemctl restart dnsmasq
```

Aufsetzen des NFS (Network File System) Root-Dateisystems

Die bisher durchgeführten Schritte erlauben es dem Klienten, den Kernel zu booten. Dieser möchte allerdings irgendwann das Root-Dateisystem starten, welches er (noch) nicht hat. Dieses haben wir allerdings vorher im Verzeichnis */nfs/client1* erstellt und müssen es nur noch exportieren:

Code-Ausschnitt 1.5.31

```
1 sudo apt-get install nfs-kernel-server
2 echo "/nfs/client1 *(rw, sync, no_subtree_check, no_root_squash)" | sudo tee -a /etc/exports
3 sudo systemctl enable rpcbind
4 sudo systemctl restart rpcbind
5 sudo systemctl enable nfs-kernel-server
6 sudo systemctl restart nfs-kernel-server
```

Editieren Sie anschließend die Datei `/ftpboot/cmdline.txt` und ersetzen Sie die Stelle ab `root =` mit

Code-Ausschnitt 1.5.32

```
1 root=/dev/nfs nfsroot=192.168.178.66:/nfs/client1,vers=3 rw ip=dhcp rootwait elevator=↔
   deadline
```

Ersetzen Sie dabei die IP-Adresse mit der IP-Adresse des Servers, die Sie sich zuvor aufgeschrieben haben. Editieren Sie abschließend die Datei `/nfs/client1/etc/fstab` und entfernen Sie die Einträge, die sich auf die Speicherkarte beziehen (`/dev/mmcblkp1` und `p2`). Lediglich das `proc`-Dateisystem sollte übrig bleiben. Nach diesen doch sehr umfangreichen Vorarbeiten sollten Sie in der Lage sein, einen Raspberry Pi-Klienten über das Netzwerk ohne SD-Karte von einem Raspberry Pi-Server zu booten. Gelingt das nicht beim ersten Versuch, zeigen Sie ein wenig Geduld. Es kann bis zu einer Minute dauern, bis der Bootvorgang startet.

1.6 Der Betrieb

Dieser Abschnitt gibt noch einige grundsätzliche Erklärungen zum Betrieb des Raspberry Pi.

1.6.1 Firmware-Warnungen

Die Raspberry Pi-Firmware gibt Warnungen aus, falls die Spannungsversorgung zu niedrig ist oder der SoC zu heiß wird.



Das Unterspannungs-Symbol zeigt an, dass die Spannungsversorgung des Raspberry Pi unter 4,63 V ($\pm 5\%$) abgefallen ist.



Wenn die Temperatur des SoC zwischen 80 °C und 85 °C beträgt, wird dieses Warnsymbol angezeigt. CPU und GPU werden gedrosselt.



Wenn die Temperatur des SoC mehr als 85 °C beträgt, wird dieses Warnsymbol angezeigt. CPU und GPU werden gedrosselt.

1.6.2 Real Time Clock

Der Raspberry Pi besitzt (aus Kostengründen) keine Echtzeituhr (die sog. RTC oder Real Time Clock). Er erhält seine Uhrzeit und sein Datum von einem sog. NTP-Server (*Network Time Protocol*). Dieser Server wird beim Starten des Pi automatisch aufgerufen und die Uhrzeit wird gesetzt. Das Kommando

Code-Ausschnitt 1.6.1

```
1 date
```

zeigt Ihnen die aktuelle Uhrzeit und das aktuelle Datum sowie die dazugehörige Zeitzone. Sollte der NTP-Server einmal nicht zu erreichen sein, beispielsweise weil Sie gar keine Internetverbindung haben, können Sie Uhrzeit und Datum auch manuell setzen, wie im unteren Beispiel gezeigt:

Code-Ausschnitt 1.6.2

```
1 sudo date 072012002014
```

Hier wird das Datum auf den 20.07.2014, 12:00 Uhr gesetzt. Es ist wichtig, das richtige Datum und die richtige Uhrzeit zu verwenden.

Andernfalls kann es beim Übersetzen von Programmen Probleme geben oder beim Surfen im Internet, beispielsweise wegen angeblich abgelaufener Zertifikate.

1.6.3 Audioausgabe

Sie haben bereits gelernt, wie man die analoge und digitale Audioausgabe einstellen kann (*raspi-config*). Sollte trotz aller Bemühungen der 3,5 mm-Buchse kein analoger Ton zu entlocken sein, kann ein

Code-Ausschnitt 1.6.3

```
1 amixer cset numid=3 1
```

helfen. Damit zwingen Sie die Audioausgabe auf ein bestimmtes Interface. Im oberen Fall ist dies das Interface *1*, welches für die analoge Audioausgabe steht. Verwenden Sie *2* für die HDMI-Ausgabe und *0* für eine automatische Ausgabe.

1.6.4 Temperatur

Das *vcgencmd*-Kommando haben Sie bereits beim Überprüfen der MPEG-2 Lizenz kennengelernt. Das Kommando ist aber noch mächtiger. Sie können beispielsweise nachschauen, wie warm der Raspberry Pi gerade ist. Die Ausgabe von

Code-Ausschnitt 1.6.4

```
1 vcgencmd measure_temp
```

liefert in meinem Fall *temp=49.8°C*, also knappe 50 °C. Diese Temperatur wird von einem Temperatursensor im Broadcom SoC (System-On-Chip) ausgelesen.

Zusammenfassung 1 Herzlichen Glückwunsch! Die ersten Schritte mit Ihrem Raspberry Pi sind geschafft! Sie haben gelernt, wie der Raspberry Pi erfolgreich in Betrieb genommen werden kann. Sie haben alle erforderlichen Einstellungen durchgeführt, erste Programme und Befehle kennengelernt und konnten die Video-Fähigkeiten der kleinen Platine näher unter die Lupe nehmen. Sie haben - je nach Bedarf - einen MPEG-2 Codec gekauft, erfolgreich installiert und getestet. Sie wissen nun, wie man das System anhält oder neu startet oder wie man Sicherheitskopien der SD-Karte anlegt oder zurückspielt. Weiterhin können Sie ihr System von einem externen USB-Speicher oder sogar von einem Netzwerk booten. Das ist doch schon eine ganze Menge! Jetzt geht es weiter mit einem ersten Einstieg in die UNIX-Welt, bei dem ich Ihnen Werkzeuge vorstellen möchte, die wir für die weiteren Kapitel des Buches benötigen. Lernen Sie im nächsten Kapitel, wie Sie das System aktuell halten können, Software von Fremdherstellern installieren können oder wie Sie Dateien editieren können. Richten Sie die Möglichkeit ein, von anderen Rechnern aus auf Ihren Raspberry Pi zugreifen zu können. Schreiben Sie Emails und verbinden Sie Ihre Netzlaufwerke. ■

2 — Die UNIX-Welt

In diesem Kapitel möchte ich Sie auf eine kleine Reise in die UNIX-Welt mitnehmen. Das Betriebssystem, welches wir im letzten Kapitel auf die SD-Karte gespielt haben, ist eigentlich ein LINUX-System. LINUX steht wiederum für Linux Is Not UniX, also Linux ist gar kein UNIX und ist damit ein rekursives Akronym. Es geht auf Linus Torvalds zurück und hat seinen Anfang 1991 genommen. Die *Raspbian*-Distribution ist eine auf Debian-Linux basierende Distribution, die für ihre Stabilität und ihren Reifegrad bekannt ist. Linus Torvalds hat anfangs nur seinen Kernel *Linux* genannt. Der Kernel ist dabei das Stück Software, welches den eigentlichen Programmen den Zugriff auf die Hardware ermöglicht. Der Name UNIX ist ebenfalls ein Akronym. Das Betriebssystem hieß ursprünglich UNICS und bedeutet Uniplexed Information and Computing Service.



2.1 Updates

Wie jedes Betriebssystem sollte auch ein UNIX/LINUX-System aktuell gehalten werden. Durch Updates werden Sicherheitslücken geschlossen oder Programmfehler beseitigt.

2.1.1 Updates in der Konsole

Der Vorgang des Updates gestaltet sich auf dem Pi sehr einfach. Programme werden aus einer Quelle heraus, dem sogenannten *Repository*, installiert. Das Dateiverzeichnis des Pi enthält bereits standardmäßig eine Liste mit einem Repository, welches alle Dateien des Betriebssystems beinhaltet. Dieses ist in der Datei `/etc/apt/sources.list` aufgeführt und lautet für die Debian „Stretch“ Version

Code-Ausschnitt 2.1.1

```
1 deb http://mirrordirector.raspbian.org/raspbian/ stretch main contrib non-free rpi
```

Durch den Befehl

Code-Ausschnitt 2.1.2

```
1 sudo apt-get update
```

werden alle Datei-Listen auf den neuesten Stand gebracht.

Der Befehl

Code-Ausschnitt 2.1.3

```
1 sudo apt-get upgrade
```

schaut nach, ob es aktuellere Programmpakete als die bereits installierten gibt. Falls dies der Fall ist, lädt das System die Programmpakete automatisch von den angegebenen Servern und installiert sie. Gleichzeitig wird die Version des Paketes festgehalten für ein eventuelles nächstes Update. Das vorangestellte *sudo* haben Sie bereits kennengelernt: Es führt den folgenden Befehl als Administrator (dieser heißt unter LINUX *root*) aus: Es ist eine Kennworteingabe erforderlich. Das ist auch sinnvoll, damit nicht jeder Benutzer beliebig Software einspielen oder auch löschen kann. Sollten Sie das *sudo* vergessen, wird das System Sie daran erinnern. Neue Programme können Sie mit

Code-Ausschnitt 2.1.4

```
1 sudo apt-get install programmname
```

installieren, wobei Sie natürlich *programmname* durch den Namen der Software ersetzen müssen, die Sie installieren wollen. Möchten Sie ein Programmpaket wieder entfernen, rufen Sie den Befehl

Code-Ausschnitt 2.1.5

```
1 sudo apt-get purge programmname
```

auf. Zur Suche nutze ich ganz gerne das Paket *aptitude*. Dies ist ein grafisches Installations-Tool, welches auch innerhalb eines LXTerminals funktioniert. Installieren Sie es mit

Code-Ausschnitt 2.1.6

```
1 sudo apt-get install aptitude
```

Die Suche ist nun sehr einfach:

Code-Ausschnitt 2.1.7

```
1 sudo aptitude search programmname
```

sucht alle Programmpakete, die das Wort *programmname* im Namen haben.



apt kennt auch noch die Parameter *dist-upgrade* oder *full-upgrade*. Diese Befehle sorgen beide dafür, dass auf eine neue Distribution umgestellt werden kann, beispielsweise auf die Nachfolgerversion von Debian „Stretch“, nachdem diese erschienen ist.

2.1.2 Programminstallation per Desktop

Der Raspbian Desktop ermöglicht es ebenfalls, auf komfortable Art und Weise Programme zu installieren. Das erforderliche Tool befindet sich unter *Einstellungen Add/Remove Software*. Das Tool selbst erlaubt eine Sortierung nach verschiedenen Themenbereichen, etwa nach dem Bereich *Multimedia*. Eine Suche ist ebenfalls vorhanden.

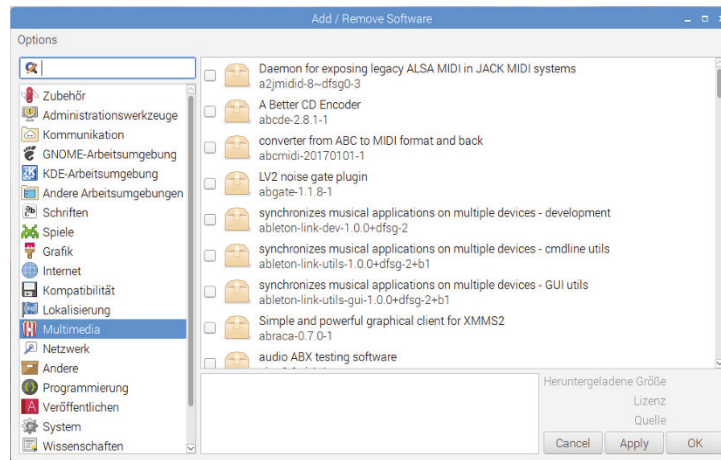


Abbildung 2.1: Das Desktop-Tool zum Hinzufügen und Entfernen von Software erlaubt eine einfache Sortierung nach Themenbereichen.

2.2 Die LEDs

In diesem Kapitel schauen wir uns noch einmal kurz die LEDs des Raspberry Pi an. Eine der LEDs - die grüne - kann per Software angesprochen werden. Die OK-LED steht unter `/sys/class/leds/led0/` zur Verfügung und kann mit den beiden folgenden Befehlen aus- oder eingeschaltet werden.

Code-Ausschnitt 2.2.1

```
1 echo 0 >/sys/class/leds/led0/brightness
2 echo 1 >/sys/class/leds/led0/brightness
```

Bitte beachten Sie, dass Sie der Hauptbenutzer (`root`) sein müssen, damit die beiden oberen Befehle funktionieren. Geben Sie hierzu den Befehl

Code-Ausschnitt 2.2.2

```
1 sudo su
```

und Ihr Kennwort ein. Die wichtigsten Konsolen-Kommandos sind in Kapitel 12.7 zusammengefasst. Es gibt auch einige Kernel-Module, welche dieser LED eine bestimmte Funktion zuweisen, z. B. einen „Ich-lebe-noch-Herzschlag“ mit Hilfe des `ledtrig_heartbeat` Moduls, welche Sie wie folgt als Benutzer `root` einbinden können:

Code-Ausschnitt 2.2.3

```
1 modprobe ledtrig_heartbeat
2 echo heartbeat >/sys/class/leds/led0/trigger
```

2.3 Fernzugriff

Manchmal ist der Raspberry Pi vielleicht hinter einem Fernseher versteckt; eine äußerst unbequeme Position, um eine Maus oder eine Tastatur anzuschließen. Manchmal fehlt vielleicht sogar ein Monitor oder Fernseher, um den Debian Desktop anzuzeigen. In diesem Abschnitt zeige ich Ihnen, wie Sie von außen auf Ihren Raspberry Pi zugreifen können, und das sogar so komfortabel, als säßen Sie davor und würden ihn mit Tastatur und Maus bedienen. Einzige Voraussetzung ist eine funktionierende Netzwerkverbindung.

2.3.1 SSH für Terminal-Fans

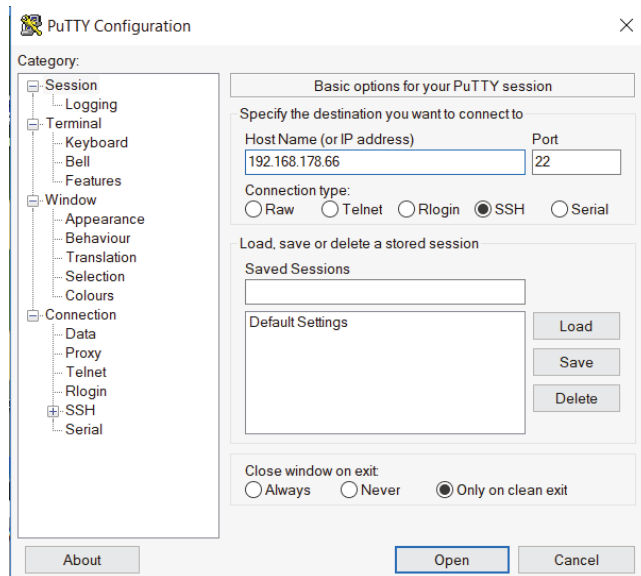


Abbildung 2.2: Der Fernzugriff ist mit Putty kinderleicht.

Falls Sie den Fernzugriff per *ssh* in der ersten Konfiguration (Kapitel 1.3) noch nicht freigeschaltet haben, holen Sie das bitte jetzt nach. Durch diese Aktion wird der *openssh*-Daemon installiert, der Ihnen einen Fernzugriff auf das System ermöglicht - ebenfalls wieder über ein Terminal. Auf einem Windows-System können Sie sich hierzu das Programm *putty.exe* herunterladen. Dieses finden Sie unter <http://www.putty.org>. Legen Sie das Programm auf Ihrem Windows Desktop ab und starten es mit einem Doppelklick. Die Programmoberfläche können Sie in Abb. 2.2 sehen. Geben Sie für einen Test unter *Hostname (or IP Address)* bitte entweder den Hostnamen, den Sie in Kapitel 1.3 vergeben haben, oder die IP-Adresse des Raspberry Pi ein. Die IP-Adresse ist eine eindeutige Nummer, unter der Ihr Rechner im Internet oder in Ihrem Heimnetz identifiziert werden kann. Die IP4-Adresse besteht aus vier Segmenten, die durch einen Punkt voneinander getrennt sind. Jedes Segment besteht aus einer Zahl, die zwischen 0 und 255 liegt. Näheres hierzu erfahren Sie im nächsten Kapitel. Die IP-Adresse Ihres Embedded Boards können Sie sich übrigens in einem LXTerminal anzeigen lassen. Rufen Sie hierzu den folgenden Befehl auf:

Code-Ausschnitt 2.3.1

```
1 ifconfig
```

Bei mir sieht die Ausgabe mit einem unter WLAN verbundenen Raspberry Pi wie folgt aus:

Code-Ausschnitt 2.3.2

```
1 wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
2   inet 192.168.178.66 netmask 255.255.255.0 broadcast 192.168.178.255
3   inet6 2001:16b8:2a:b000:8160:2e1b:f5dd:c452 prefixlen 64 scopeid 0x0 <global>
4   inet6 fe80::96d2:c5f9:4d0:59ba prefixlen 64 scopeid 0x20<link>
5   ether b8:27:eb:80:17:e6 txqueuelen 1000 (Ethernet)
6   RX packets 9674 bytes 1100001 (1.0 MiB)
7   RX errors 0 dropped 19 overruns 0 frame 0
8   TX packets 22596 bytes 25216023 (24.0 MiB)
9   TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Die WLAN-IP4-Adresse meines Raspberry Pi ist 192.168.178.66. Diese Adresse muss ich für Putty auf einem Windows-PC, der sich im internen Netz befindet, eingeben, um einen Terminal-Zugriff auf den kleinen Rechner zu erhalten. *ssh* nutzt übrigens den normierten Port 22 für diesen abgesicherten Zugriff. Nach erfolgreicher Verbindung können Sie sich mit dem Benutzernamen *pi* und Ihrem Kennwort anmelden.

Vielleicht werden Sie sich fragen, warum es neben der *inet*-Adresse (IP4) noch eine *inet6*-Adresse (IPv6) angegeben ist.

In den Anfangstagen des Internets, als es nur wenige Rechner gab, ist man davon ausgegangen, dass etwa vier Milliarden IP-Adressen ausreichen, um alle verfügbaren Rechner mit dem Internet zu verbinden und direkt anzusprechen. Vier Milliarden entspricht in etwa dem Adressraum von IPv4, nämlich 2^{32} . Durch die große Verbreitung des Internets reicht aber inzwischen die Zahl der verfügbaren IP4-Adressen nicht mehr aus. Mit IPv6 wurde daher ein besseres System entwickelt, welches ungefähr 340 Sextillionen Adressen zur Verfügung stellt. Darüber hinaus bietet IPv6 Verbesserungen im Protokollrahmen, um beispielsweise Router von Rechenaufwand zu entlasten.

Putty kommt übrigens ohne Probleme mit IPv6-Adressen zurecht. Sie können statt 192.168.178.66 (in meinem Beispiel) also auch 2001:16b8:2a:b000:8160:2e1b:f5dd:c452 als Adresse eintragen.

2.3.2 Fernzugang mit Schreibtisch

Debian „Stretch“ kommt bereits mit vorinstalliertem VNC-Server (*Virtual Network Computing*). Wenn Sie den Desktop des Raspberry Pi auf einem anderen Computer wiedergeben wollen, stellen Sie bitte sicher, dass Sie den VNC Server beim Einrichten des Raspberry Pi (Kapitel 1.3) gestartet haben. Den VNCViewer gibt es für viele Betriebssysteme. Er kann von der Webseite <https://www.realvnc.com/de/connect/download/viewer/> heruntergeladen und anschließend auf dem Gastrechner installiert werden. Es gibt sogar eine Version, die auf einem iPad läuft. Bei meiner Installation habe ich mich für einen Windows-Rechner entschieden, von dem aus ich auf den Schreibtisch des Raspberry Pi zugreifen möchte (Abb. 2.3). Der erste Start des Programms erwartet die IP-Adresse des Raspberry Pi sowie den Benutzer-

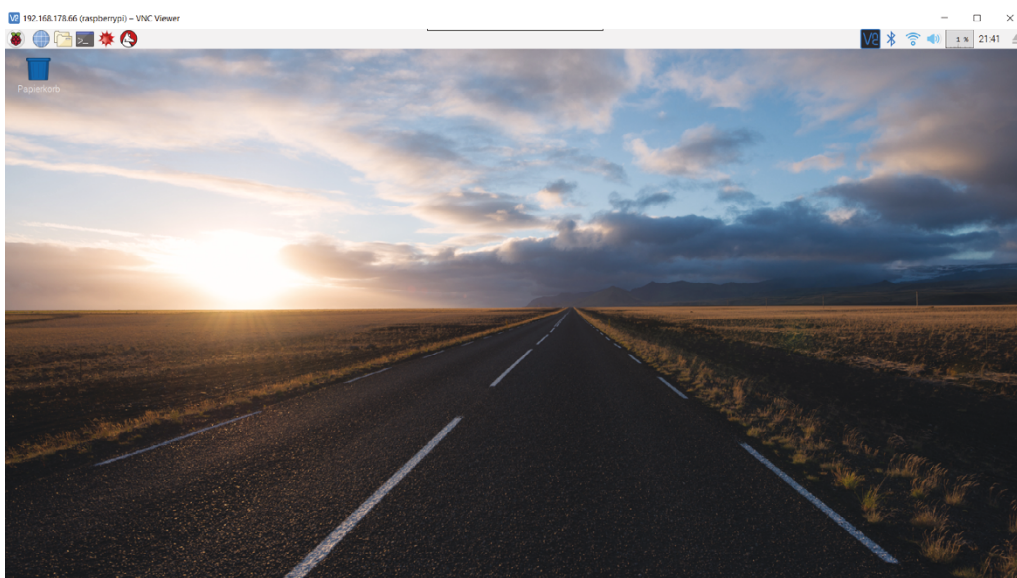


Abbildung 2.3: Der VNCViewer erlaubt es, den Desktop des Raspberry Pi auf anderen Computern/Monitoren sichtbar zu machen und zu bedienen.

namen und das Kennwort des Benutzers. Der Benutzername ist *pi*, das Kennwort haben Sie wahrscheinlich im vorangegangenen Kapitel auf ein sicheres Kennwort gesetzt.

Nach der Verbindung können Sie den Schreibtisch des Raspberry Pi so benutzen, als säßen Sie davor.

Jede Bewegung der Maus und jede Eingabe mit der Tastatur, die Sie im VNCViewer tätigen, wird genau so auf dem Raspberry Pi Desktop ausgeführt. Ist Ihr Raspberry Pi an einen Monitor angeschlossen, können Sie alle Eingaben auch auf diesem Monitor verfolgen. Konnte VNCViewer einmal erfolgreich eine Verbindung herstellen, speichert das Programm diese Verbindung zusammen mit einem Icon des Server-Desktops (Abb. 2.4). Für den nächsten Start reicht dann ein Doppelklick auf dieses Icon.

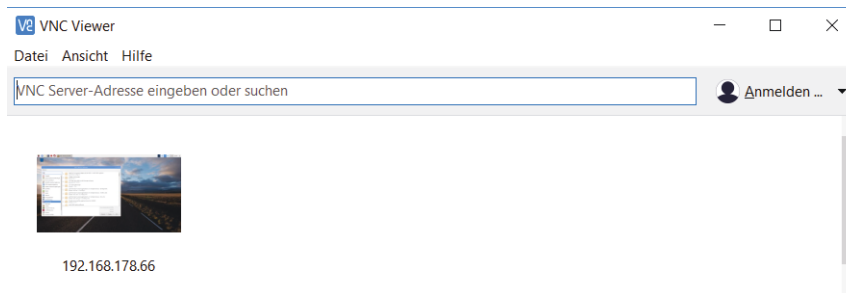


Abbildung 2.4: VNCViewer speichert erfolgreiche Verbindungen inklusive einer Vorschau.

Bisher sind wir davon ausgegangen, dass der Pi seine IP-Adresse per DHCP erhält. Im nächsten Kapitel erfahren Sie, wie man den Pi von einer dynamischen IP-Adresse auf eine feste IP-Adresse umstellen kann oder ihn per WLAN verbindet.

2.4 Netzwerk editieren

Bevor wir uns an die Einstellungen für das Netzwerk begeben, machen wir einen kleinen Ausflug in die Editor-Welt. Dabei lernen Sie zwei Editoren kennen, die man bequem von einem Terminal aus bedienen kann. Natürlich können Sie auch grafische Editoren wie beispielsweise *emacs* verwenden. Diese kann man aber nicht remote von einem Terminal aus bedienen, sondern auf dem Gast-System müsste auch ein X-Window laufen (Kapitel 3.1). Der Editor, der auf allen Linux-Systemen zu Hause ist, nennt sich *vi*. Er ist auch standardmäßig schon auf dem Pi installiert. Öffnen Sie ein LXTerminal und editieren Sie Ihre erste Datei, die wir einfach *test* nennen:

Code-Ausschnitt 2.4.1

```
1 vi test
```

Nach dem Öffnen des Editors schicken Sie diesen bitte mit dem *i*-Kommando in den Insert-Modus. Tippen Sie hierzu einfach den Buchstaben *i*. Danach können Sie die Datei *test* mit Leben füllen. Den Insert-Modus können Sie mit *ESC* verlassen. Wechseln Sie nach Fertigstellen der Datei in den Kommando-Modus und nutzen Sie das Kommando zum Speichern (*write*) und Verlassen der Datei (*quit*). Geben Sie hierzu *ESC :wq* ein. Die Datei *test* ist nun gespeichert und hat den von Ihnen eingetragenen Inhalt. Sie wollen den Inhalt überprüfen? Kein Problem. Ein

Code-Ausschnitt 2.4.2

```
1 ls -l
2 more test
```

zeigt Ihnen zunächst alle Dateien im aktuellen Verzeichnis an. Die Datei *test* sollte sich darunter befinden.

Zusätzlich werden weitere Dinge angezeigt wie der Besitzer der Datei, die Schreib- und Lese-rechte, das Datum der letzten Änderung oder die Größe der Datei. Bei mir sieht dies wie folgt aus:

Code-Ausschnitt 2.4.3

```
1 drwxrwxr-x 2 pi pi    4096 Jul 20 2012 python_games
2 -rw-r--r-- 1 pi pi    4094 Dez 4 16:59 test
```

Der Buchstabe *d* steht für *directory*, also *Verzeichnis*. Er zeigt an, dass *python_games* ein Verzeichnis ist, in dem sich weitere Dateien befinden. Sie können mit dem Befehl

Code-Ausschnitt 2.4.4

```
1 cd python_games
```

in das Verzeichnis wechseln und mit dem Befehl

Code-Ausschnitt 2.4.5

```
1 cd ..
```

wieder eine Ebene höher gelangen. Wichtig ist hierbei das Leerzeichen nach dem *cd*-Befehl (*change directory*). Die Buchstaben *r* und *w* stehen für lesen (*read*) und schreiben (*write*). Die Sektionen wiederholen sich zweimal, stehen also insgesamt dreimal da. Die erste Sektion ist dabei der Benutzer *pi*, der in der Regel alles darf. Die nächste Sektion ist die Gruppe, der der Besitzer *pi* zugeordnet ist. Diese heißt in der Raspbian-Distribution ebenfalls *pi*. Hier steht, was Benutzer derselben Gruppe mit der Datei machen dürfen. Die letzte Sektion steht für andere Benutzer und deren Rechte an der Datei. Die nächste Zahl (4096) gibt die Größe der Datei in Bytes an. Meine Datei *test* wurde zum letzten Mal am 4. Dezember editiert und zwar um 16:59 Uhr. Eine hilfreiche Referenz für den Editor *vi* finden Sie in der folgenden Tabelle. Sieht kryptisch

vi Befehl	Bedeutung
i	Fügt an der aktuellen Cursorposition ein
a	Fügt hinter der aktuellen Cursorposition ein
x	Löscht das Zeichen rechts vom Cursor
dd	Löscht eine ganze Zeile
yy	Kopiert die aktuelle Zeile und fügt sie unterhalb des Cursors ein
ESC	Verlässt den Editormodus
:w	Schreibt die Datei
:q	Verlässt den Editor
:q!	Beendet <i>vi</i> , ohne das aktuelle Dokument zu speichern
123G	Springt zur Zeile 123

Tabelle 2.1: Die wichtigsten *vi*-Befehle lernt man schnell.

aus? Ist es auch. Aber glauben Sie mir: Dadurch, dass *vi* auf jedem Kühlschrank läuft, ist es hervorragend geeignet, um schnell Änderungen an Dateien durchzuführen. Zusätzlich können Sie auch die Hilfsseite zum Editor aufrufen, die sogenannte Manual Page oder Handbuch-Seite:

Code-Ausschnitt 2.4.6

```
1 man vi
```

Hier erhalten Sie eine detaillierte Auflistung aller Parameter, die man beim Aufruf übergeben kann. Diese Hilfe ist für die meisten Programme verfügbar. Wenn Sie es nicht ganz so kryptisch mögen, kann ich Ihnen den kleinen Editor *joe* ans Herz legen. Raten Sie, was kommt? Genau, wir installieren ihn mit dem Befehl

Code-Ausschnitt 2.4.7

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

Sie können übrigens auch *aptitude* zum Installieren von Software verwenden. Der entsprechende Befehl lautet dann

Code-Ausschnitt 2.4.8

```
1 sudo aptitude install joe
```

Aptitude hat den Vorteil, dass es komplizierte Installations-Situationen besser auflösen kann als *apt-get*. Aber zurück zu unserem Editor *joe*. Rufen Sie das mit *vi* erstellte Dokument *test* auf oder erstellen Sie ein neues Dokument:

Code-Ausschnitt 2.4.9

```
1 joe test
```

Im Editor *joe* können Sie sich frei bewegen, einfügen und löschen, ganz so wie Sie es von einem Textverarbeitungsprogramm gewohnt sind. Aber das Beste kommt noch: Man muss sich keine kryptischen Befehle merken! *Joe*'s Befehle kann man sich mit dem *CTRL-h*-Kommando anzeigen lassen. Das sieht dann ungefähr so aus wie in Abb. 2.5. Der wichtigste Befehl beim Editor *joe* ist *CTRL-x*. Dieses Kommando speichert die Datei und verlässt den Editor.

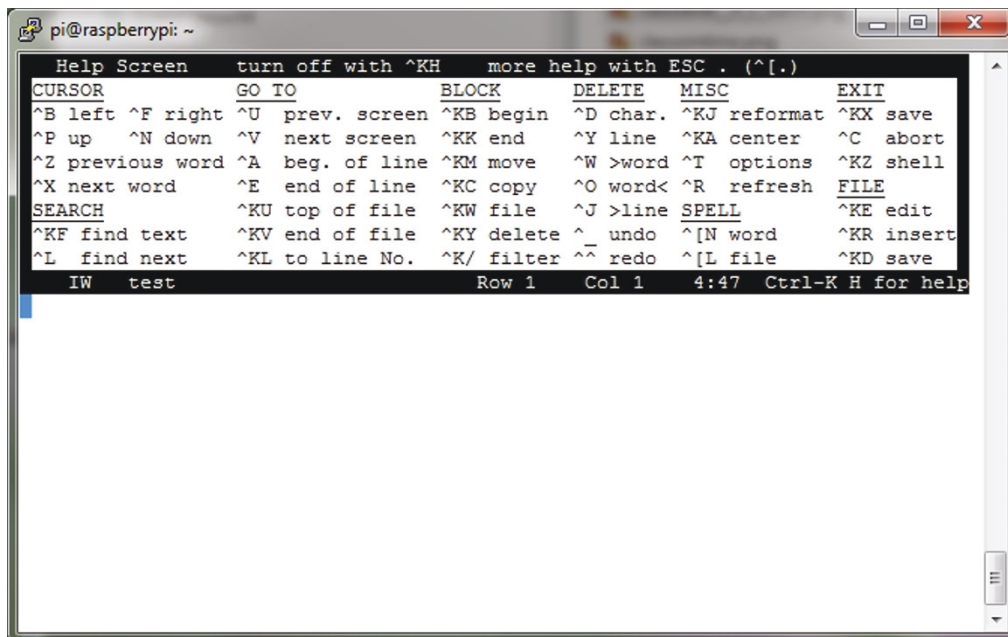


Abbildung 2.5: Die Hilfe zum Editor *Joe* kann man im Editor selbst aufrufen.

2.4.1 Dynamische IP

Standardmäßig ist der Pi so konfiguriert, dass er von einem DHCP-Server (in der Regel von Ihrem DSL-Modem) eine dynamische IP-Adresse erhält, unter der er dann in Ihrem Heimnetzwerk zu erreichen ist. Das kabelgebundene Netzwerk-Interface haben Sie bereits in Quelltext 2.3.2 kennengelernt. Interfaces unter LINUX werden dabei durchnummeriert. Das erste kabelgebundene Ethernet heißt *eth0*, das nächste *eth1* usw. Da der Pi nur einen Ethernet-Anschluss besitzt, gibt es folglich nur das *eth0*-Interface. Die Netzwerkeinstellungen sind in den Dateien */etc/network/interfaces* und */etc/resolv.conf* gesichert. Bei einer dynamisch vergebenen Adresse sehen die Dateien so aus:

Code-Ausschnitt 2.4.10

```
1 auto lo
2 iface lo inet loopback
3 iface eth0 inet dhcp
```

Entscheidend ist die Zeile 3. Sie alleine legt fest, dass das *eth0*-Interface eine dynamische IP-Adresse von einem DHCP-Server erhält, der sich im lokalen Netz befinden muss.

Code-Ausschnitt 2.4.11

```
1 domain fritz.box
2 search fritz.box
3 nameserver 192.168.178.1
```

Die Einstellungen an der oben gezeigten Datei */etc/resolv.conf* werden bei dynamischer IP-Vergabe automatisch vorgegeben. Hier hat meine Fritz!Box Domain- und Suchstring vergeben. Meine Fritz!Box ist in meinem Heimnetz unter der IP-Adresse 192.168.178.1 zu erreichen. Auf ihr läuft der Nameserver, ein Dienst, der Text-Adressen in Internet-IP-Adressen übersetzt, etwa <http://www.google.de> in 173.194.112.87. Diese Adressübersetzung können Sie übrigens mit dem Programm *nslookup* nachvollziehen. Installieren Sie *dnsutils* mit

Code-Ausschnitt 2.4.12

```
1 sudo apt-get install dnsutils
```

und geben Sie *nslookup* in ein LXTerminal ein und danach eine Webseite Ihrer Wahl. Das Programm *nslookup* fragt daraufhin den Nameserver (also in meinem Fall die Fritz!Box) nach der Auflösung des Namens und erhält o. g. Antwort. Sollte die automatische Erkennung für den Nameserver nicht funktionieren, können Sie einen beliebigen Nameserver Ihrer Wahl eintragen. Google betreibt öffentliche Nameserver, die unter den Adressen 8.8.8.8 und 8.8.4.4 erreicht werden können.

2.4.2 Statische IP

Betreiben Sie Ihr Embedded Board nicht in einem Heimnetz, sondern in einem Firmennetz, kann es sein, dass Ihr Systemadministrator Ihnen eine feste IP-Adresse zuteilt, unter der die Platine erreicht werden kann. Bitte denken Sie daran, dass der Zugriff auf die Datei */etc/network/interfaces* *root*-Rechte benötigt; stellen Sie dem Editor-Befehl Ihrer Wahl also ein *sudo* voran

Code-Ausschnitt 2.4.13

```
1 sudo joe /etc/network/interfaces
```

und ändern Sie die Datei wie folgt:

Code-Ausschnitt 2.4.14

```
1 auto lo
2 iface lo inet loopback
3 iface eth0 inet static
4 address 172.16.112.131
5 netmask 255.255.255.0
6 gateway 172.16.1.252
```

In Zeile 3 steht nun, dass wir eine statische IP-Adresse für das *eth0*-Interface nutzen wollen. Die Adresse selbst steht in Zeile 4. Bitte tragen Sie hier Ihre statische IP-Adresse ein, meine dient nur als Beispiel. Die Netzwerkmaske wird hinter dem Bezeichner *netmask* eingetragen. Sie lautet in der Regel 255.255.255.0, es sei denn, Sie haben mehr als 256 Rechner in Ihrem Netz. Über das Gateway wird die Verbindung nach draußen hergestellt. Tragen Sie die Gateway-Adresse hinter dem Bezeichner *gateway* ein. Das war auch schon alles für die Datei */etc/network/interfaces*. Tragen Sie nun bitte noch die Nameserver in die Datei */etc/resolv.conf* ein.



Mehrere Nameserver können wie folgt eingetragen werden:

Code-Ausschnitt 2.4.15

```
1 nameserver 8.8.8.8
2 nameserver 4.4.8.8
```

Nach den Änderungen an der Netzwerkeinstellung sollten Sie den Netzwerkdienst neu starten, damit die Änderungen übernommen werden. Dies geschieht durch den Aufruf

Code-Ausschnitt 2.4.16

```
1 sudo /etc/init.d/networking restart
```

Mehr zum Thema Dienste erfahren Sie in Kapitel 2.6.

2.4.3 WLAN

Während die alten Raspberry Pi Modelle noch nicht mit einem WLAN-Chip ausgestattet waren, bietet der 3B+ einen Kombi-Chip, der WLAN und Bluetooth (4.2) in einem Bauelement vereint. Der WLAN-Chip kann dabei auch als Hotspot arbeiten. Diese Funktion wird in Kapitel 8.3 noch ausführlich beschrieben. Für alle älteren Raspberry Pi Modelle kann ich Ihnen den EDIMAX EW-7811UN Wireless USB Adapter empfehlen. Auch dieser winzige USB-Adapter kann sich nicht nur mit WLAN-Netzen verbinden, sondern ebenfalls als Access Point arbeiten, also selbst ein WLAN zur Verfügung stellen. Es ist ebenfalls möglich, über den internen WLAN-Chip eine Verbindung ins Internet herzustellen und einen USB WLAN Adapter als Hotspot zu nutzen, um vielleicht Gästen einen eingeschränkten Internetzugriff zu ermöglichen. Für diesen Zweck und für ältere Pi Modelle wird daher die Installation dieses USB-Sticks beschrieben. Nach dem Einstecken in den USB-Port oder den -Hub sollte das System den Treiber automatisch laden. Das können Sie mit Hilfe der beiden folgenden Befehle überprüfen:



Abbildung 2.6: Kleiner geht es kaum: der EDIMAX USB WLAN-Stick (Quelle: EDIMAX).

Code-Ausschnitt 2.4.17

```
1 lsusb
2 dmesg | more
```


Der erste Befehl `lsusb` zeigt Ihnen alle eingesteckten USB-Geräte an, der zweite Befehl alle Kernel-Meldungen. Hier sollte am Ende stehen, dass der Treiber erfolgreich geladen wurde. Das Zeichen `/` nennt man übrigens *pipe*. Zusammen mit `more` bewirkt es, dass die Bildschirmausgabe des voranstehenden Befehls aufhört, wenn das sichtbare Terminal voll ist. Sie können dann mit der `SPACE`-Taste vorwärts blättern. Bei mir sieht die `lsusb`-Ausgabe so aus:

Code-Ausschnitt 2.4.18

```

1 Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
2 Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
3 Bus 001 Device 004: ID 2001:f103 D-Link Corp. DUB-H7 7-port USB 2.0 hub
4 Bus 001 Device 005: ID 7392:7811 Edimax Technology Co., Ltd EW-7811Un 802.11n Wireless ↵
   Adapter [Realtek RTL8188CUS]
5 Bus 001 Device 007: ID 413c:2105 Dell Computer Corp. Model L100 Keyboard
6 Bus 001 Device 008: ID 413c:3012 Dell Computer Corp. Optical Wheel Mouse
7 Bus 001 Device 009: ID 148f:5370 Ralink Technology, Corp. RT5370 Wireless Adapter

```

Die Kernel-Meldungen beim Laden eines Ralink-USB-Sticks sehen so aus:

Code-Ausschnitt 2.4.19

```

1 usb 1-1.2.7: new high-speed USB device number 9 using dwc_otg
2 usb 1-1.2.7: New USB device found, idVendor=148f, idProduct=5370
3 usb 1-1.2.7: New USB device strings: Mfr=1, Product=2, SerialNumber=3
4 usb 1-1.2.7: Product: 802.11 n WLAN
5 usb 1-1.2.7: Manufacturer: Ralink
6 usb 1-1.2.7: SerialNumber: 1.0
7 cfg80211: Calling CRDA to update world regulatory domain
8 usb 1-1.2.7: reset high-speed USB device number 9 using dwc_otg
9 ieee80211 phy0: rt2x00_set_rt: Info - RT chipset 5390, rev 0502 detected
10 ieee80211 phy0: rt2x00_set_rf: Info - RF chipset 5370 detected
11 ieee80211 phy0: Selected rate control algorithm 'minstrel_ht'
12 usbcore: registered new interface driver rt2800usb
13 ieee80211 phy0: rt2x00lib_request_firmware: Info - Loading firmware file 'rt2870.bin'
14 ieee80211 phy0: rt2x00lib_request_firmware: Info - Firmware detected - version: 0.29

```

Nachdem der WLAN-Stick erfolgreich in Betrieb genommen wurde, scannen wir nach kabellosen Netzen:

Code-Ausschnitt 2.4.20

```

1 iwlist wlan0 scan | egrep "(ESSID|IEEE)"

```

Das Programm `iwlist` bietet umfassende Information zu einem WLAN-Interface. Verfügbare Netze werden durch den Parameter `scan` angezeigt. Haben Sie mehrere WLAN-Adapter (beispielsweise den eingebauten und einen USB-Stick), ersetzen Sie bitte `wlan0` durch den Adapter Ihrer Wahl (z. B. `wlan1`). Um die Ausgabe des `iwlist`-Kommandos zu filtern, nutzen wir den Befehl `egrep`, dem wir die Ausgabe von `iwlist` weiterleiten („*pipen*“, `|`). `Egrep` steht für *Extended Global Regular Expressions Print* und listet in unserem Fall nur Zeilen mit den Begriffen `ESSID` oder `IEEE`.

Bei mir sieht die Antwort des Scans in der Konsole wie folgt aus:

Code-Ausschnitt 2.4.21

```

1 ESSID:"wireless"
2 IE: IEEE 802.11i/WPA2 Version 1

```

Die SSID meines WLAN-Netzwerkes, also der WLAN-Name lautet „`wireless`“, der Standard ist IEEE 802.11i mit einer WPA2-Verschlüsselung.

Wenn Sie das Pipe-Zeichen (`|`) und alles dahinter in dem oben gezeigten Aufruf weglassen, erhalten Sie viel zusätzliche Information über das zur Verfügung stehende WLAN wie etwa die Signalstärke.

Das Netzwerk, mit dem wir den Raspberry Pi verbinden möchten, hinterlegen wir in der Datei `wpa_supplicant.conf`. Da diese in einem geschützten Bereich steht, benötigen wir zur Bearbeitung `root`-Rechte:

Code-Ausschnitt 2.4.22

```
1 sudo vi /etc/wpa_supplicant/wpa_supplicant.conf
```

Ersetzen Sie `vi` gerne durch einen Editor Ihrer Wahl. Am Ende dieser Datei fügen wir unser Netzwerk dazu oder ändern gegebenenfalls ein vorhandenes:

Code-Ausschnitt 2.4.23

```
1 network={
2   ssid="wireless"
3   psk="kennwort"
4 }
```

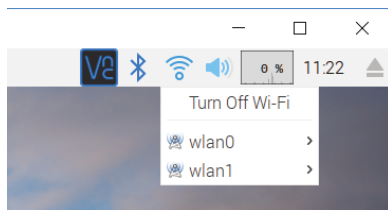


Abbildung 2.7: Grafische Oberfläche mit 2 WLAN-Adaptoren (intern und USB).

Setzen Sie bitte für `ssid` den Namen des WLAN ein, mit dem Sie verbinden möchten und ersetzen Sie `kennwort` durch das Kennwort dieses Netzwerks. Speichern Sie die Datei anschließend, damit die Änderungen wirksam werden. Der entsprechende WLAN-Adapter kann nun mit Hilfe der Befehle

Code-Ausschnitt 2.4.24

```
1 sudo ifdown wlan0
2 sudo ifup wlan0
```

neu initialisiert werden. Sie können nun mit `ifconfig` nachschauen, ob der WLAN-Adapter eine Verbindung aufbauen konnte.

Die grafische Oberfläche zeigt übrigens WLAN-Netze für alle zur Verfügung stehenden WLAN-Adapter an (Abb. 2.7).

2.5 Bluetooth

Der Raspberry Pi 3B+ versteht sich von Hause aus auf den Bluetooth-Standard 4.2. Sie können Bluetooth nutzen, um kabellose Tastaturen anzuschließen, Musik per Kopfhörer hören oder Dateien mit Ihrem Mobiltelefon austauschen.

2.5.1 Kabellose Tastatur

Das Verbinden einer Bluetooth-Tastatur gestaltet sich ganz einfach. Zunächst einmal klickt man auf das Bluetooth-Symbol (Abb. 2.7) und anschließend auf *Make Discoverable*. Das Bluetooth-Logo beginnt dann, abwechselnd grün und blau zu blinken.

Add New Device listet anschließend meine Apple-Tastatur mit der ID E8-06-88-2F-18-CD auf. Die Tastatur kann dann mit *Pair* gekoppelt werden und steht ab sofort zur Verfügung.

2.5.2 Kopfhörer

Im nächsten Schritt verbinde ich meinen Bluetooth Kopfhörer. Die Vorgehensweise ist identisch mit dem Einrichten der Tastatur. Stellen Sie den Bluetooth-Adapter wieder auf *Make Discoverable* und versetzen Sie den Kopfhörer in den Pairing-Modus. Unter dem Menüpunkt *Add Device...* taucht der Kopfhörer auf und kann hinzugefügt werden. Der Kopfhörer wird allerdings nicht automatisch verbunden, das

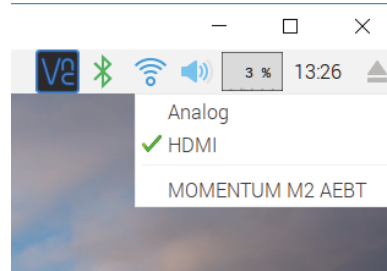


Abbildung 2.8: Per Bluetooth eingebundene Kopfhörer müssen im Audio-Menü aktiviert werden.

Logo im Bluetooth-Menü leuchtet rot und der Verbindungsversuch wird mit dem Hinweis beendet, die Audio-Einstellungen zu verändern. Das können Sie durch einen Klick mit der rechten Maustaste auf das Lautsprechersymbol, welches sich rechts neben dem WLAN-Symbol befindet. Wenn Sie hier Ihren Bluetooth-Kopfhörer auswählen, verbindet er sich und das Logo im Bluetooth-Menü leuchtet grün. Mit einem Youtube-Video im Chromium-Webbrowser können Sie schnell überprüfen, ob die Verbindung steht und funktioniert.

2.5.3 Dateiübertragung

Das Koppeln eines Mobiltelefons per Bluetooth geht ähnlich schnell wie das Koppeln einer Tastatur, einer Maus oder eines Kopfhörers. Sie müssen lediglich auf Ihrem Telefon die Koppelung bestätigen. Für die Dateiübertragung vom Telefon zum Raspberry Pi über Bluetooth kann das Programm *Obex* verwendet werden. Die Abkürzung steht für *OBject EXchange*. *Obex* ist ein Kommunikationsprotokoll für den Austausch von binären Dateien zwischen verschiedenen Geräten. Installieren wir dieses Protokoll zunächst auf dem Raspberry Pi

Code-Ausschnitt 2.5.1

```
1 sudo apt-get install obexftp obexpushd
```



Der Befehl `sudo apt-get install` kann nicht nur ein Programm installieren. Sie können mehrere Pakete durch Leerzeichen getrennt angeben, die gleichzeitig installiert werden sollen.

Damit *Obex* auf dem Raspberry Pi richtig funktioniert, musste ich im Bluetooth-Skript noch eine Änderung einfügen. In der Datei `/etc/systemd/system/dbus-org.bluez.service` muss der Bluetooth-Daemon mit dem zusätzlichen Parameter „-C“ aufgerufen werden. Dieser stellt ein veraltetes Kommandozeilen-Interface zur Verfügung, welches wir im nächsten Schritt benötigen. Editieren Sie also bitte die oben angegebene Datei

Code-Ausschnitt 2.5.2

```
1 sudo vi /etc/systemd/system/dbus-org.bluez.service
```

und ergänzen Sie bitte den Aufruf des Deemons um den o.g. Parameter:

Code-Ausschnitt 2.5.3

```
1 ExecStart=/usr/lib/bluetooth/bluetoothd -C
```

Bitte starten Sie Ihr System neu, damit die oben gemachten Änderungen zur Anwendung kommen und verbinden Sie ihr Mobiltelefon erneut. Öffnen Sie dann bitte ein Terminal und wechseln in das Verzeichnis, in dem die per Telefon übertragene Datei gespeichert werden soll. Danach wird das *Obex*-Protokoll gestartet:

Code-Ausschnitt 2.5.4

```
1 obexpushd -B -n
```

Die Parameter „-B“ und „-n“ sorgen dafür, dass der *Obex*-Dienst auf Bluetooth-Verbindungen hört und das Terminal das Programm nicht schließt. Nachdem der Dienst läuft, können Sie Dateien von Ihrem Mobiltelefon per Bluetooth zu Ihrem Raspberry Pi senden. Die Dateien werden in dem Verzeichnis gespeichert, von dem aus Sie den *Obex*-Dienst gestartet haben.

Um eine Datei vom Raspberry Pi zum Mobiltelefon zu übertragen, benötigen wir zunächst die Bluetooth-Adresse des Telefons. Der Aufruf von

Code-Ausschnitt 2.5.5

```
1 bluetoothctl
2 exit
```

ergibt bei mir die Ausgabe

Code-Ausschnitt 2.5.6

```
1 [NEW] Device 9C:D3:5B:62:71:62 Rudi (Galaxy Note Edge)
```

Die Bluetooth-Adresse benötigen gleich wieder. Finden wir als nächstes heraus, auf welchem Kanal das Mobilgerät gesendete Dateien erwartet:

Code-Ausschnitt 2.5.7

```
1 sdptool browse 9C:D3:5B:62:71:62
```

Bitte ersetzen Sie in diesem Aufruf meine Bluetooth-Adresse durch die Ihres Gerätes. Die interessanten Zeilen beinhalten die Wörter *Object Push*

Code-Ausschnitt 2.5.8

```
1 Service Name: OBEX Object Push
2 Service RecHandle: 0x1000c
3 Service Class ID List:
4 "OBEX Object Push" (0x1105)
5 Protocol Descriptor List:
6 "L2CAP" (0x0100)
7 "RFCOMM" (0x0003)
8 Channel: 12
9 "OBEX" (0x0008)
10 Profile Descriptor List:
11 "OBEX Object Push" (0x1105)
12 Version: 0x0100
```

Der Kanal (*Channel*), auf dem die gesendeten Daten erwartet werden, hat in meinem Fall die Nummer 12.

Nehmen wir an, wir möchten eine Datei namens *test.mp3* an unser Mobilfunkgerät senden, so haben wir jetzt alle Daten bereit, um das zu bewerkstelligen:

Code-Ausschnitt 2.5.9

```
1 obexftp --nopath --noconn --uuid none --bluetooth 9C:D3:5B:62:71:62 --channel 12 -p test↵  
   .mp3
```

Bitte ersetzen Sie im oben gezeigten Aufruf wieder alle Daten entsprechend Ihrem Gerät, des richtigen Kanals und Ihrer Datei. Bitte stellen Sie sicher, dass Sie entweder den gesamten Dateipfad angeben oder sich im Verzeichnis befinden, in dem die zu sendende Datei gespeichert ist.

Nachdem der Empfang auf dem Handy quittiert wurde, startet die Übertragung der Datei.

2.6 Init-Skripte

Im Abschnitt 2.4.2 haben Sie vielleicht schon einmal einen Dienst (nämlich den Netzwerkdienst) neu gestartet; LINUX kennt eine ganze Menge an Diensten. Die Dienste werden durch Skripte kontrolliert, die sich im Verzeichnis */etc/init.d* befinden. Ein Skript ist ein Kommandozeilen-Programm. Die meisten Dienste kennen die Parameter *start*, *stop*, *restart* und *status*. Fragen Sie doch einfach mal den Netzwerk-Zeitserver nach seinem Status:

Code-Ausschnitt 2.6.1

```
1 /etc/init.d/ntp status
```

Die Antwort sollte so ausfallen:

Code-Ausschnitt 2.6.2

```
1 [ ok ] NTP server is running.
```

Denken Sie bitte daran, dass Sie der Administrator *root* sein müssen, um Dienste zu stoppen oder zu starten. Stellen Sie also wie gewohnt dem Aufruf zum Starten oder Stoppen eines Dienstes das *sudo* vor. Später werden wir lernen, eigene Skripte zu erstellen und beispielsweise einen Multimediadienst automatisch zu starten. Dienste kann man mit dem Befehl

Code-Ausschnitt 2.6.3

```
1 sudo update-rc.d scriptname defaults
```

automatisch (also auch nach einem Neustart des Systems) starten lassen. *scriptname* steht dabei für das zu startende Skript. Das automatische Starten wird wieder unterbunden, nachdem der Befehl

Code-Ausschnitt 2.6.4

```
1 sudo update-rc.d -f scriptname remove
```

einggegeben wurde.

Auf neueren Raspbian-Systemen können Sie alternativ das Kommando

Code-Ausschnitt 2.6.5

```
1 sudo insserv scriptname
```

verwenden, um ein Init-Skript automatisch zu starten und

Code-Ausschnitt 2.6.6

```
1 sudo insserv -r scriptname
```

um ein Init-Skript wieder zu entfernen. Alle Init-Skripte (also die Skripte, die beim Systemstart initialisiert werden), haben immer denselben Aufbau:

Code-Ausschnitt 2.6.7

```
1  #!/bin/sh
2  # /etc/init.d/scriptname

4  ### BEGIN INIT INFO
5  # Provides:          noip
6  # Required-Start:    $remote_fs $syslog
7  # Required-Stop:     $remote_fs $syslog
8  # Default-Start:     2 3 4 5
9  # Default-Stop:      0 1 6
10 # Short-Description: Simple script to start a program at boot
11 # Description:       A simple script which will
12 # start / stop a program a boot / shutdown.
13 ### END INIT INFO

15 # If you want a command to always run, put it here

17 # Carry out specific functions when asked to by the system
18 case "$1" in
19   start)
20     echo "Starting programmname"
21     # run application you want to start
22     /usr/local/bin/programmname
23     ;;
24   stop)
25     echo "Stopping programmname"
26     # kill application you want to stop
27     killall programmname
28     ;;
29   *)
30     echo "Usage: /etc/init.d/scriptname {start|stop}"
31     exit 1
32     ;;
33 esac

35 exit 0
```

Skripte müssen ausführbar sein, damit sie gestartet werden können.

Code-Ausschnitt 2.6.8

```
1 sudo chmod 755 /etc/init.d/scriptname
```

Denken Sie bitte daran, Ihr Skript zu testen, bevor Sie es im Systemstart hinzufügen. Rufen Sie es dazu mit

Code-Ausschnitt 2.6.9

```
1 sudo /etc/init.d/scriptname start
2 sudo /etc/init.d/scriptname stop
```

auf. Doch dazu später mehr. Es sei an dieser Stelle noch erwähnt, dass Programme, die beim Systemstart automatisch starten sollen, auch in die Datei */etc/rc.local* eingetragen werden können.

2.7 Wichtige Tipps

In diesem Abschnitt möchte ich Ihnen noch einige wichtige Tipps mit auf den Weg geben, die mir auch oft geholfen haben, wenn ich eine Datei so geändert habe, dass das System nicht mehr richtig starten konnte oder wenn ich ein komplettes Backup einspielen musste.

2.7.1 Missglückter Start

Es kann vorkommen, dass man eine Datei so geändert hat, dass das System nicht mehr richtig startet. Um die Änderungen, die dazu geführt haben, rückgängig zu machen, gibt es mehrere Möglichkeiten:

1. Man kann ein Backup einspielen. Das dauert aber lange und alle Änderungen seit dem letzten Backup sind verloren. Dieser Vorgang wird im Kapitel 2.7.2 beschrieben.
2. Man kann die SD-Karte in einem Linux-Rechner einbinden und das *ext*-Dateisystem mounten. Dann können alle Änderungen an Dateien gemacht werden und der Pi kann mit den geänderten Dateien neu gestartet werden.
3. Alte Raspberry Pi (2014 und älter) konnten mit einem sogenannten Notfall-Boot wieder zum Leben erweckt werden. Dieses Feature ist aber seit vielen Jahren aus der Raspberry Pi Firmware verschwunden, so dass an dieser Stelle nicht mehr weiter darauf eingegangen wird.

Sicherlich werden Sie sich fragen, welche Änderungen denn dazu führen können, dass das System nicht mehr richtig startet. Ein kleines Beispiel ist die Übertaktung des Pi. Stellen Sie in der Datei */boot/config.txt* zu hohe Frequenzen ein, kann es durchaus sein, dass ein stabiler Betrieb nicht mehr möglich ist. Andere Beispiele sind Bildschirmausgabe-Modi, die nicht funktionieren, oder falsche Kernel-Parameter.

Die Konfigurationsdatei des Debian-Images befindet sich auf einer FAT32-Partition, die ohne Probleme von Windows-Systemen gelesen/beschrieben werden kann. Sollten Sie also versehentlich Einstellungen in dieser Datei vorgenommen haben, die einen erfolgreichen Boot-Vorgang verhindern, können Sie diese auf einem Windows-Rechner einfach korrigieren. Bitte nutzen Sie dazu einen Editor, die LINUX-kompatibel ist. Ich kann Ihnen beispielsweise *Notepad++* empfehlen.

Sollten Sie Änderungen innerhalb des *ext*-Dateisystems rückgängig machen wollen, können Sie die betroffene SD-Karte per USB-Adapter mit einem LINUX-Rechner (z. B. ein Raspberry Pi, der mit einer anderen SD-Karte bootet) reparieren. Auf die SD-Karte kann wie folgt zugegriffen werden: Binden Sie die UNIX-Partition der SD-Karte mit dem Befehl

Code-Ausschnitt 2.7.1

```
1 sudo mount /dev/mmcblk0p2 /mnt
```

ein. Dieser Befehl stellt das Dateisystem der zweiten Partition der Raspbian SD-Karte im Verzeichnis */mnt* zur Verfügung. Wechseln Sie danach in das Verzeichnis */mnt* mit dem Befehl

Code-Ausschnitt 2.7.2

```
1 cd /mnt
```

Sie können sich alle Dateien mit *ls -l* anzeigen lassen und entsprechend bearbeiten. Denken Sie bitte daran, das Dateisystem nach dem Bearbeiten mit

Code-Ausschnitt 2.7.3

```
1 sudo umount /mnt
```

wieder auszuhängen. Fahren Sie das Board - wie Sie es in Kapitel 1.3 gelernt haben - wieder herunter, bevor Sie das geänderte System wieder neu starten.

2.7.2 Partition verkleinern

Wie in Kapitel 1.2.2 bereits erwähnt, kann das Programm Win32 Disk Imager dazu verwendet werden, Backups der SD-Karte zu erstellen und auch wieder zurückzuspielen. Das geht solange gut, wie der verfügbare Speicherplatz auf der SD-Karte mindestens so groß ist wie die Backup-Datei. Hat eine neue Speicherkarte weniger verfügbaren Speicher als die Karte, von der das Backup angefertigt worden ist, scheitert das Vorhaben, das Backup mit Win32 Disk Imager zurückzuspielen. Für wichtige Backups empfehle ich daher folgendes Vorgehen:

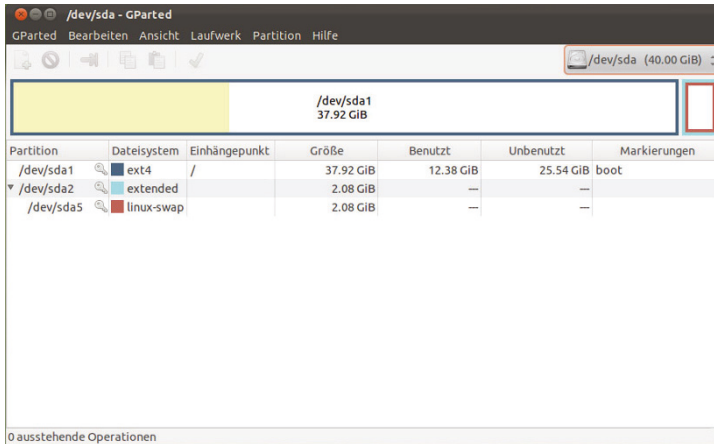


Abbildung 2.9: Mit Gparted kann man Partitionen verkleinern.

empfehle ich daher folgendes Vorgehen:

1. Das Raspbian-Betriebssystem auf der SD-Karte besteht aus zwei Partitionen, einer FAT-32 Partition, die den Kernel enthält, und einer *ext*-Partition, die den Rest des Betriebssystems und die persönlichen Daten der Benutzer enthält. Diese Partition nutzt - sofern Sie die Partition erweitert haben, wie in Kapitel 1.3 beschrieben - die volle Größe der SD-Karte aus, kann also problemlos mehrere GB groß sein. Sie muss zunächst verkleinert werden. Dies funktioniert nicht im laufenden Betrieb, während die Partition eingebunden ist, sondern muss auf einem anderen Rechner geschehen. Besorgen Sie sich hierzu eine *gparted* Live Installation, beispielsweise von <http://gparted.org/livecd.php>. Schreiben Sie diese Installation auf ein Medium (CD, USB-Stick), von dem gestartet werden kann, und starten Sie einen PC mit diesem Medium.
2. Binden Sie Ihre Raspberry Pi SD-Karte ein und verkleinern Sie die *ext*-Partition. Achten Sie darauf, dass ein kleiner Speicherplatz übrig bleibt, damit Sie das System noch starten können, um die Partition später wieder zu vergrößern. 500 MB bis 1 GB sollten ausreichen. Das Verkleinern der Partition wird je nach Füllgrad der SD-Karte bis zu 30 Minuten in Anspruch nehmen.
3. Nachdem die Partition verkleinert worden ist, sollte die SD-Karte noch einmal im Pi gestartet werden, um mögliche Fehler bei der Verkleinerung auszuschließen.
4. Fahren Sie Ihren Pi wieder herunter (`sudo halt -p`) und entnehmen Sie die SD-Karte, wenn nur noch die rote PWR-LED leuchtet.
5. Machen Sie nun ein Backup der SD-Karte mit Hilfe des in Kapitel 1.2.2 beschriebenen Programmes Win32 Disk Imager.

6. Sie können dieses Backup nun auf eine andere SD-Karte schreiben, die - wie bereits mehrfach erwähnt - mindestens den Speicherplatz der Größe des Images zur Verfügung stellen muss.
7. Starten Sie Ihren Pi mit der neuen SD-Karte, die das Backup enthält, und vergrößern Sie die Partition wieder, wie in Kapitel 1.3 beschrieben.

2.7.3 config.txt

Zum Abschluss dieses Abschnittes wird hier noch einmal die */boot/config.txt*-Datei gezeigt, die meine Einstellungen beinhaltet. Die Bedeutung aller Parameter dieser Datei erfahren Sie unter http://elinux.org/RPi_config.

Code-Ausschnitt 2.7.4

```
1 # For more options and information see
2 # http://rpf.io/configtxt
3 # Some settings may impact device functionality. See link above for details

5 # uncomment if you get no picture on HDMI for a default "safe" mode
6 #hdmi_safe=1

8 # uncomment this if your display has a black border of unused pixels visible
9 # and your display can output without overscan
10 disable_overscan=1

12 # uncomment the following to adjust overscan. Use positive numbers if console
13 # goes off screen, and negative if there is too much border
14 #overscan_left=16
15 #overscan_right=16
16 #overscan_top=16
17 #overscan_bottom=16

19 # uncomment to force a console size. By default it will be display's size minus
20 # overscan.
21 #framebuffer_width=1280
22 #framebuffer_height=720

24 # uncomment if hdmi display is not detected and composite is being output
25 #hdmi_force_hotplug=1

27 # uncomment to force a specific HDMI mode (this will force VGA)
28 #hdmi_group=1
29 #hdmi_mode=1

31 # uncomment to force a HDMI mode rather than DVI. This can make audio work in
32 # DMT (computer monitor) modes
33 #hdmi_drive=2

35 # uncomment to increase signal to HDMI, if you have interference, blanking, or
36 # no display
37 #config_hdmi_boost=4

39 # uncomment for composite PAL
40 #sdtv_mode=2

42 #uncomment to overclock the arm. 700 MHz is the default.
43 #arm_freq=800

45 # Uncomment some or all of these to enable the optional hardware interfaces
46 dtparam=i2c_arm=on
47 #dtparam=i2s=on
48 dtparam=spi=on

50 # Uncomment this to enable the lirc-rpi module
51 dtoverlay=lirc-rpi

53 # Additional overlays and parameters are documented /boot/overlays/README

55 # Enable audio (loads snd_bcm2835)
56 dtparam=audio=on
57 enable_uart=1
58 dtoverlay=w1-gpio
59 gpu_mem=144

61 # Codecs
62 decode_MPG2=0xfb694a38,0xc342d582,0x91f22de0
63 decode_WVC1=0xc4e67fbe,0x27a26e01
```



Ist Ihnen aufgefallen, dass Sie alle Codecs, die Sie gekauft haben, durch Kommata separiert in die Datei `config.txt` eintragen können? Dadurch können Sie dieselbe SD-Karte in verschiedenen Pi benutzen, ohne den Codec ändern zu müssen.

2.7.4 SD-Schreibschutz

An dieser Stelle sei darauf hingewiesen, dass der alte Raspberry Pi den Schreibschutz-Schieber an der SD-Karte (statt Mikro-SD-Karte) ignoriert. Er kann die Karte auch beschreiben, wenn der Schieber in der *Lock*-Stellung steht, der Schreibschutz also aktiviert ist.



Lassen Sie den Schreibschutz für die SD-Karte ruhig aktiviert. Der Pi kann trotzdem schreiben. Wenn Sie die Karte aber außerhalb des Pi nutzen, können Sie so versehentliches Löschen verhindern.

2.8 Windows-Umsteiger

Dieser Abschnitt soll Microsoft-WindowsTM-Umsteigern den Einstieg in die LINUX-Welt ein wenig erleichtern. Er enthält einige Informationen zu Pi Standard-Programmen wie E-Mail-Tools, Filebrowsern oder Verbindung mit externen Speichermedien. Ausführliche Information zu Textverarbeitung erhalten Sie in Kapitel 11 „Schreiben mit dem Pi“.

2.8.1 E-Mail

Als E-Mail-Client der Debian „Stretch“-Version kommt das Programm *Claws Mail* zum Einsatz. In der Desktop-Variante des Raspbian Betriebssystems ist es bereits vorinstalliert und steht unter *Startleiste* → *Internet* → *Claws Mail* zur Verfügung. Dieser E-Mail-Client kann sich mit IMAP- oder POP3-Servern verbinden.

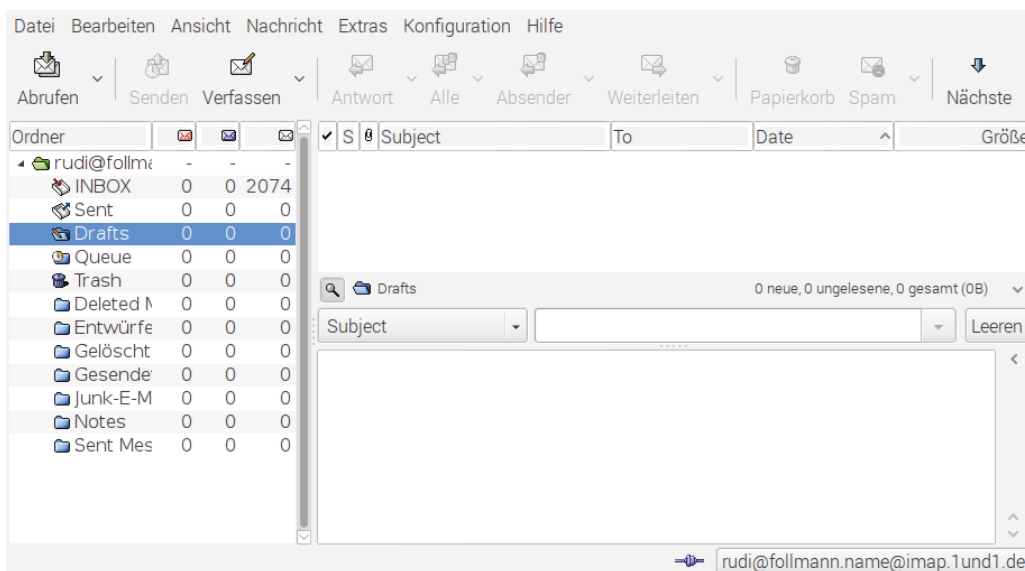


Abbildung 2.10: Der E-Mail-Client *Claws Mail* bietet IMAP- und POP3-Zugang.

2.8.2 Dateibrowser

Als Windows-Umsteiger haben Sie sich sicherlich schon einmal gefragt, wo unter UNIX die Laufwerksbuchstaben geblieben sind oder was passiert, wenn alle Buchstaben bis einschließlich Z: vergeben sind. Unter UNIX/LINUX gibt es keine Laufwerksbuchstaben.

Festplatten und externe Geräte werden in Verzeichnisse eingebunden (*ge-mounted*). Unter der X11-Oberfläche gibt es das Programm *Dateimanager*, welches Sie unter *Start* → *Zubehör* finden. Die Abb. 2.11 zeigt das Verzeichnis des Benutzers *pi*, das sogenannte *home*-Verzeichnis (daher das Haus). Das *home*-Verzeichnis eines Benutzers ist unter dem Verzeichnis */home/benutzername* angesiedelt. Mit dem Dateimanager können Sie Dateien kopieren, löschen oder

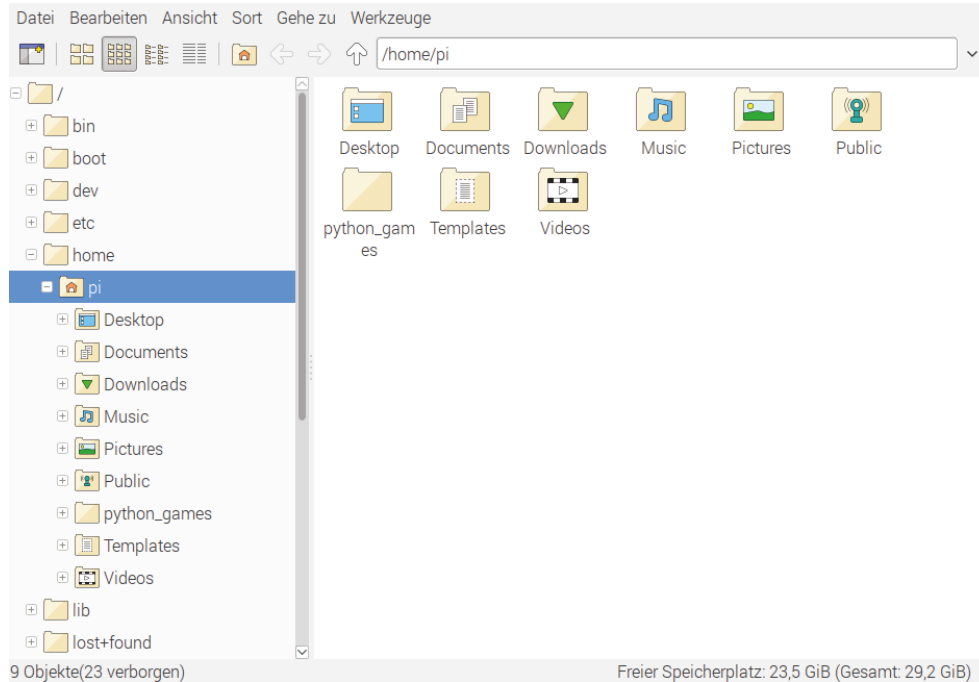


Abbildung 2.11: Mit dem Dateimanager können alle Datei-Operationen durchgeführt werden.

umbenennen. Sie können aber genauso gut die Terminal-Befehle

Code-Ausschnitt 2.8.1

```
1 cp name name_kopie
2 mv name_alt name_neu
3 rm name
```

cp (copy), *mv* (move) oder *rm* (remove) verwenden.

2.8.3 Verbindung mit einem NAS

Die Verbindung mit einem externen NAS-Laufwerk (*Network Attached Storage*) gestaltet sich sehr einfach. Navigieren Sie im Dateimanager zu *Gehe zu* → *Netzwerk*. Sollte ihr gewünschtes Gerät nicht auftauchen, verbinden Sie es bitte über das Samba-Protokoll, indem Sie in die Adressleiste des Dateimanagers `smb://name_des_laufwerks` eingeben. Der Name des Laufwerks ist dabei der Netzwerkname oder aber die IP-Adresse des zu verbindenden Laufwerks. In Kapitel 3 lernen Sie, wie man externe Laufwerke als *iSCSI*-Device permanent einbinden kann.

Zusammenfassung 2 Das war der Ausflug in die UNIX-Welt. Sie haben gelernt, wie das System aktuell gehalten werden kann und wie Sie Programme installieren und auch wieder vom System entfernen können.

Wir haben einen Fernzugriff per *ssh* eingerichtet. Weiterhin haben Sie gelernt, wie dynamische und statische IP-Adressen für den Pi vergeben werden können und wie man einen WLAN USB-Stick verwendet, um den Raspberry Pi kabellos mit dem Internet zu verbinden. Zum Editieren der Dateien habe ich Ihnen zwei minimalistische Editoren vorgestellt. Sie haben erste Erfahrungen mit dem LINUX-Dateisystem gemacht und können Systemdienste starten und stoppen.

Sie haben gelernt, wie Sie Backups durch Verkleinern von Partitionen so erstellen können, dass Sie auf andere SD-Karten passen, selbst wenn diese etwas kleiner bezüglich ihres Speicherplatzes sind.

Das Ende des Kapitels enthält eine vollständig beschriebene Konfigurationsdatei *config.txt* und einen Leitfaden für Windows-Umsteiger. ■

Windows ruff „Pi“
Tanze Samba mit mir
iSCSI ist kein Apple-Device
Streng vertraulich!
My home is my castle

3 — Fernzugriff und mehr

Nicht immer sitzt man vor dem Monitor, an dem der Raspberry Pi angeschlossen ist. Wir haben zwar im Kapitel 2 gelernt, wie man sich über ein Terminal von einem anderen Rechner in den Pi einloggen kann, ein bisschen Grafik wäre aber auch nicht schlecht.

Dieses Kapitel gibt einen Überblick über grafische Verbindungen. Sie lernen, die grafische Oberfläche des Pi auf anderen Rechnern sichtbar zu machen, Netzlaufwerke dauerhaft einzubinden oder Dateien mit beispielsweise einem Windows-Explorer auf den Pi oder zurück zu kopieren.



3.1 Windows ruff „Pi“

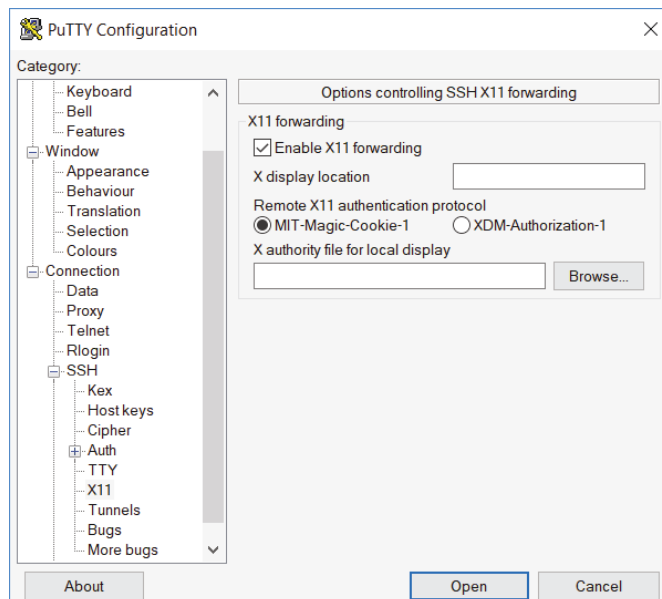


Abbildung 3.1: X-Weiterleitung mit Putty.

aus darstellen.

Im Kapitel 2 haben Sie bereits gelernt, wie man mit *putty.exe* einen sogenannten Shell-Zugriff auf ein LINUX-System erhält. Aber *putty.exe* kann noch mehr. Wie in Abb. 3.1 gezeigt, kann man *Putty* dazu veranlassen, die grafische Oberfläche des Pi auf den Windows Desktop weiterzuleiten. Aktivieren Sie bitte die Option *Enable X11 forwarding*, also Weiterleiten der X11-Oberfläche. Dann wählen Sie sich bitte auf dem Pi ein, wie in Kapitel 2 beschrieben. Wenn Sie nun ein Programm auf dem Pi aufrufen, welches die grafische Oberfläche benutzt, wird diese an Ihr Windows-System weitergeleitet. Die X-Oberfläche (also die LINUX Standard-Oberfläche) kann Windows nicht von Haus

Hierzu benötigen Sie weitere Software - einen sogenannten X-Server -, die wir nun auf Ihrem Windows PC installieren werden. Als freie Software (*freeware*) hat sich das Programm *Xming* bewährt, welches auf <http://sourceforge.net/projects/xming/> zum Download angeboten wird. Im Verzeichnis <http://sourceforge.net/projects/xming/files/Xming> fin-

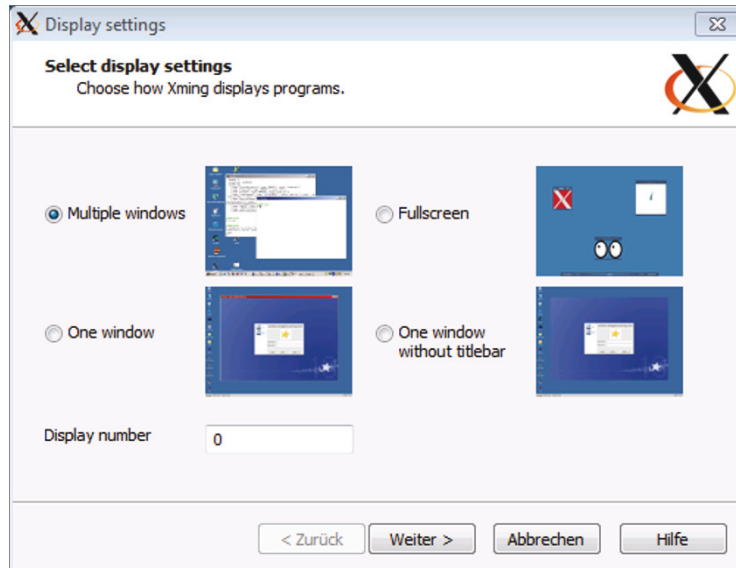


Abbildung 3.2: Die Einstellmöglichkeiten von Xlaunch.

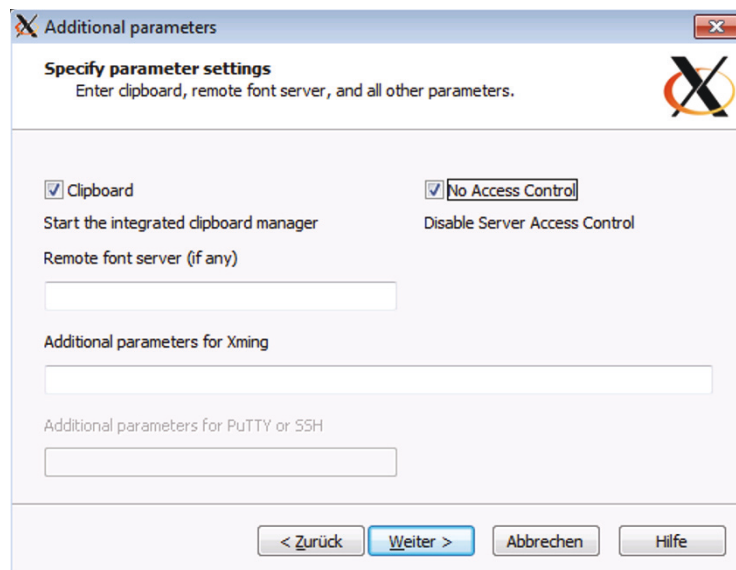


Abbildung 3.3: Die Xlaunch Zugriffskontrolle.

den Sie die aktuelle Version von *Xming* als installierbares Windows-Programm. Führen Sie die Installation durch Doppelklick auf Ihrem Windows-PC durch. Zusätzlich zu *Xming* sollten Sie die Schriftarten für *Xming*, die sogenannten *Fonts* installieren. Diese finden Sie im Verzeichnis <http://sourceforge.net/projects/xming/files/Xming-fonts>, ebenfalls als installierbare Windows-Datei. Starten Sie die Installation der Fonts auf Ihrem Windows PC ebenfalls durch Doppelklick auf den Dateinamen. Nachdem Sie sowohl *Xming* als auch die dazugehörigen Fonts installiert haben, starten Sie bitte die *Xlaunch*-Anwendung (Abb. 3.2).

Lassen Sie alle Einstellungen auf den Default-Werten, mit einer Ausnahme: Bitte erlauben Sie, dass alle Klienten auf den *Xming*-Server zugreifen dürfen (Abb. 3.3). Wenn Sie möchten, können Sie die Einstellung speichern und auf dem Windows Desktop ablegen. Dann müssen Sie beim nächsten Mal die Einstellungen nicht wieder erneut vornehmen. Sobald der X-Server auf Ihrem Windows-Betriebssystem läuft, können Sie - vorausgesetzt Sie haben im *Putty* X-forward aktiviert und sich in den Pi eingeloggt - ein Programm auf diesem aufrufen, welches dann auf Ihrem Windows Desktop erscheint. Testen Sie es einfach, indem Sie den Ihnen schon bekannten Internetbrowser *midori* in der von Ihnen gestarteten *Putty*-Session aufrufen.



Unter LINUX-Systemen (wie z. B. Raspbian, MacOS) kann man aus einem Terminal heraus mittels

Code-Ausschnitt 3.1.1

```
1 ssh name@IP-Adresse -X
```

Verbindung zu einem anderen LINUX-Rechner aufnehmen und die grafische Oberfläche an den aufrufenden Rechner weiterleiten. Dabei muss auf dem aufrufenden Rechner die X-Oberfläche laufen. *name* ist der Benutzer, der angemeldet werden soll, etwa *pi*, *IP-Adresse* ist der Name oder die IP-Adresse des Rechners, mit dem Sie Verbindung aufnehmen wollen (z. B. 192.178.178.66).

3.1.1 Remote Desktop

Vielleicht haben Sie schon einmal den Windows Remote Desktop ausprobiert, um von Ihrem Windows PC auf einen anderen Windows-PC zuzugreifen. Das Protokoll hierfür nennt sich RDP (*Remote Desktop Protokoll*). Der Zugriff ist auch auf Ihren Pi möglich. Dazu muss auf diesem lediglich das Programm *xrdp* installiert werden:

Code-Ausschnitt 3.1.2

```
1 sudo apt-get install xrdp
```

xrdp ist übrigens ein Dienst, der unter */etc/init.d/xrdp* zu Hause ist. Sollte der Dienst nach der Installation nicht laufen, kann er mit

Code-Ausschnitt 3.1.3

```
1 sudo /etc/init.d/xrdp start
```

gestartet werden. Danach können Sie von Ihrem Windows PC aus die *Remotedesktop-Verbindung* aufrufen. Diese finden Sie unter *Alle Programme* → *Zubehör*. Geben Sie einfach die IP-Adresse Ihres Pi als remote-Rechner an.

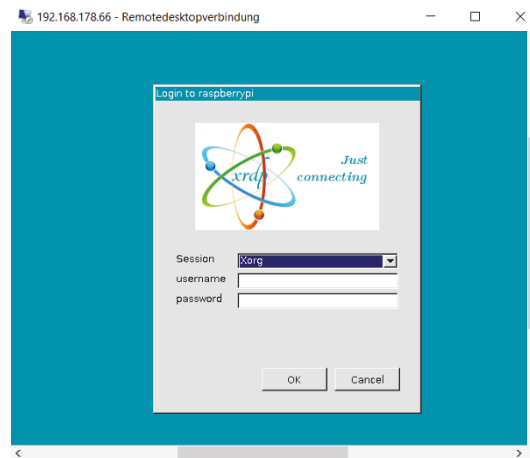


Abbildung 3.4: Fernzugriff auf xrdp.

Als Session wählen Sie bitte *Xorg*. *username* ist der Benutzername auf dem Pi, also beispielsweise *pi*, *Kennwort* das Kennwort dieses Benutzers. Nach erfolgreich hergestellter Verbindung und der Frage, ob Sie auch wirklich mit einem nicht bekannten Klienten verbinden wollen, sehen Sie Ihren Pi Desktop auf Ihrem Windows PC. Einstellungen wie Skalierung oder Bildschirmgröße kann man unter *Options* im Remotedesktop-Verbindungs-Fenster vornehmen.

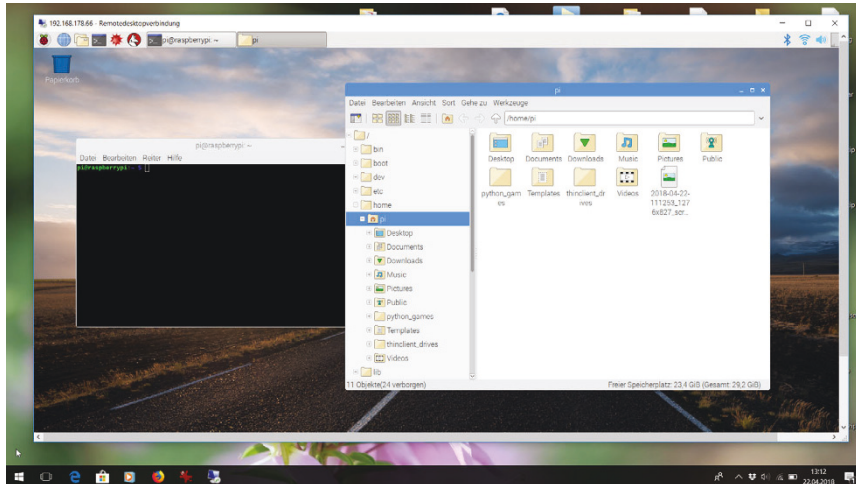


Abbildung 3.5: Der Pi Desktop wird im Windows Desktop sichtbar.

Info Unter Linux können Sie *rdesktop* verwenden. Installieren Sie es und rufen Sie es wie folgt auf:

Code-Ausschnitt 3.1.4

```
1 sudo apt-get install rdesktop
2 rdesktop -r 1024x768 IP-Adresse
```

Die Zahlen *1024x768* beschreiben dabei die Auflösung des Desktops, *IP-Adresse* ist die IP-Adresse des Rechners, mit dem Sie verbinden wollen.

3.1.2 Heimweh

Wäre es nicht schön, wenn wir den Pi Desktop auch außerhalb unseres Heimnetzes sehen könnten, beispielsweise auf dem Rechner in unserem Büro? Ein echter Fernzugriff also. Prinzipiell gibt es hierzu zwei Möglichkeiten:

1. Wir nutzen einen dynamischen Domain Service und leiten Ports für *ssh* und *rdp* in unserem DSL-Modem weiter. Diese Möglichkeit wird im Folgenden näher beschrieben.

Info Ports weiterzuleiten kann ein Sicherheitsrisiko darstellen. *ssh* nutzt zwar eine sichere Verbindung, *RDP* aber nicht und kann damit prinzipiell abgehört werden.

2. Wir richten einen VPN-Server (*Virtual Private Network*) auf unserem Raspberry Pi ein, mit dem wir uns von außen verbinden können. Diese Vorgehensweise wird in Kapitel 8.4 erklärt.

Heutige Modems werden einmal am Tag vom DSL-Provider zwangsgetrennt. Sie erhalten dann eine neue IP-Adresse. Die Zeit, in der dies geschieht, lässt sich meistens festlegen (z. B. 3:00 Uhr morgens). Da man sich nicht jeden Tag die neue IP-Adresse, unter der das DSL-Modem erreichbar ist, aufschreiben und merken kann, macht es Sinn, einen dynamischen Nameservice (*DynDNS*) in Anspruch zu nehmen.

Nach dem Wechsel der IP-Adresse teilt Ihr DSL-Modem diesem Service mit, dass sich die IP-Adresse geändert hat und wie die neue IP-Adresse lautet. Der dynamische Nameservice referenziert den Internetnamen, den Sie bei der Anmeldung am dynamischen Nameservice festgelegt haben, auf diese neue Adresse. Damit ist Ihr Modem immer unter demselben Namen im Internet erreichbar. Einen kostenlosen DynDNS-Service können Sie unter <http://freedns.no-ip.com> einrichten. Gehen wir davon aus, dass Sie die Adresse *pi.myftp.org* (die bestimmt schon vergeben ist), reserviert haben. Diese Adresse tragen Sie zusammen mit Ihren Anmelde-daten des DynDNS-Services in Ihr DSL-Modem ein. Von jetzt an ist dieses unter der Adresse *pi.myftp.org* erreichbar. In Ihrem Modem müssen Sie jetzt noch Ports für den Service weiterleiten, die Sie extern nutzen möchten. Hat also Ihr Raspberry Pi die IP-Adresse 192.168.178.66, können Sie die Ports 22 (*ssh*) und 3389 (*rdp*) auf die IP-Adresse 192.168.178.66 weiterleiten. Von außen können Sie nun ein Terminal öffnen (*putty.exe*) und als Adresse *pi.myftp.org* eingeben. Der DynDNS-Service löst den Namen *pi.myftp.org* mit der IP-Adresse Ihres Modems auf; der Port 22, den *Putty* kontaktiert, wird auf Ihren Pi umgeleitet, auf den Sie sich jetzt von außen einwählen können. Dasselbe gilt für eine *Remotedesktop*-Verbindung. Hier wird ebenfalls automatisch der Port 3389 auf den Raspberry Pi umgeleitet und Sie sehen den Pi Desktop zum Beispiel auf Ihrem Büro-Rechner. Unter Umständen hat Ihr Systemadministrator bestimmte Ports in Ihrem Büro gesperrt, die er auch nicht freigeben möchte. In diesem Fall könnte er Ihnen andere Ports, wie etwa den Port 55555, freigeben. Sie können dann in Ihrem DSL-Modem den Port 55555 auf den Port 22 Ihres Pi weiterleiten und ihn dann von außen unter *pi.myftp.org:55555* kontaktieren.

3.1.3 Sicheres Kopieren

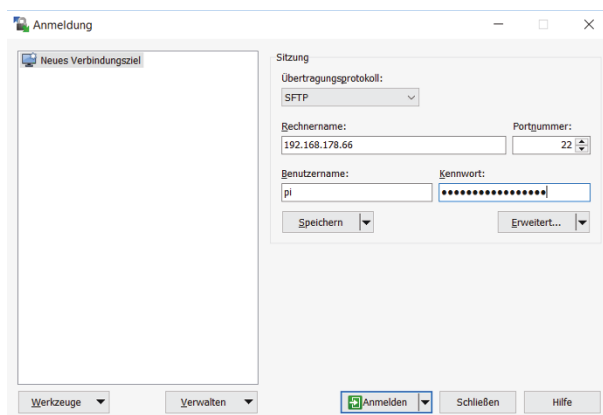


Abbildung 3.6: WinSCP erlaubt sicheres Kopieren von und zum Raspberry Pi.

Der Port bleibt auf 22 (22). Unter *Username* geben Sie bitte den Benutzernamen auf dem Pi ein, also beispielsweise *pi*. *Password* ist Ihr Benutzerkennwort. Nach dem Login (→ *Login*) erhalten Sie eine Dateiübersicht Ihres Pi-Verzeichnisses. Dateien können Sie nun einfach per Drag-and-Drop zwischen den verschiedenen Rechnern hin und her kopieren.

Alternativ kann der Linux-Aufruf *scp* verwendet werden, um Dateien von einem Rechner zum anderen zu kopieren. Die Aufrufe lauten:

Jetzt, wo wir den Zugang von extern zu unserem Pi eingerichtet haben, wäre es doch auch schön, wenn wir Dateien von und zu unserem Pi kopieren könnten. Unter Windows gibt es dazu das Programm *winscp*. *scp* steht dabei für *secure copy*, also „sicheres Kopieren“. Das kostenlose Programm erhalten Sie unter <http://winscp.net>. Nach der Installation und dem Aufrufen des Programmes werden Sie mit einem Bildschirm ähnlich dem in Abb. 3.6 begrüßt. Geben Sie unter *Host Name* die IP-Adresse Ihres Pi an. Diese kann entweder lokal oder die des DynDNS sein.

Code-Ausschnitt 3.1.5

```
1 scp Dateiname Benutzername@IP-Adresse:/Verzeichnis
2 scp Benutzername@IP-Adresse:/Verzeichnis/Datei Verzeichnis
```

je nachdem, ob man eine Datei auf einen entfernten Rechner kopieren möchte oder von einem entfernten Rechner auf den lokalen. *Benutzername* ist der Name des Benutzers auf dem Rechner (also wieder *pi*), *IP-Adresse* die IP-Adresse des jeweiligen Rechners. *Verzeichnis* und *Datei* sind selbsterklärend. *scp* kopiert immer von irgendwo *nach* irgendwo.



Sie können auch sogenannte Wildcards benutzen, um alle Dateien eines Verzeichnisses zu kopieren. Der Aufruf lautet dann:

Code-Ausschnitt 3.1.6

```
1 scp -r Verzeichnis/* Benutzername@IP-Adresse:/Verzeichnis
```

Bitte beachten Sie den Parameter *-r*, der für einen rekursiven Aufruf steht.

Weitere Hilfe zu *scp* liefert die Manual Page

Code-Ausschnitt 3.1.7

```
1 man scp
```

oder die Hilfe zu *scp*

Code-Ausschnitt 3.1.8

```
1 scp --help
```

3.1.4 Fernweh

Während wir im letzten Abschnitt gelernt haben, wie wir von außen auf den Raspberry Pi zugreifen können, drehen wir den Spieß nun um. In diesem Abschnitt stellen wir die Verbindung zwischen unserem Pi zu Hause und einem VPN-Server außerhalb, etwa im Büro, her. Nur noch einmal zur Erinnerung: VPN steht für *Virtual Private Network* und ist eine gesicherte Verbindung. Der Rechner, der sich mit einem VPN-Server verbindet, erhält eine interne IP-Adresse aus dem Netz, in dem sich auch der Server befindet. Danach kann der Pi auf die gesamte fremde Internetstruktur zugreifen, beispielsweise auf Firmenserver.

3.1.5 Cisco Client

Cisco ist ein bekanntes und häufig benutztes VPN-Protokoll. Der Cisco Client kann auf dem Pi durch das Kommando

Code-Ausschnitt 3.1.9

```
1 sudo apt-get install vpnc vpnc-scripts
```

installiert werden. Danach muss lediglich noch eine Datei angelegt werden, die sich im Verzeichnis */etc/vpnc* befinden muss. In unserem Beispiel nennen wir diese Datei *vpn_firma.conf*:

Code-Ausschnitt 3.1.10

```
1 sudo joe /etc/vpnc/vpn_firma.conf
```

In diese Datei kopieren Sie bitte folgenden Inhalt:

Code-Ausschnitt 3.1.11

```
1 IPsec gateway vpnserver
2 IPsec ID vpnbenutzer
3 IPsec secret vpnsecret
4 Xauth username benutzername
5 Xauth password benutzerkennwort
```

In der oberen Auflistung müssen folgende Ersetzungen vorgenommen werden:

- *vpnserver*: Hier muss der VPN-Server eingetragen werden. Dazu kann entweder der Servername oder die IP-Adresse verwendet werden.
- *vpnbenutzer*: Hier muss die IPsec-ID angegeben werden. Sie erhalten diese vom Systemadministrator des Cisco VPN-Servers.
- *vpnsecret*: Hier muss das IPsec-Kennwort eingetragen werden. Sie erhalten dieses ebenfalls vom Administrator des VPN-Servers.
- *benutzername*: Bitte tragen Sie hier den Benutzernamen ein, den Sie vom Administrator des VPN-Servers erhalten haben.
- *benutzerkennwort*: Bitte geben Sie hier das Kennwort ein, das Ihnen Ihr VPN-Server-Administrator gegeben hat.

Speichern Sie die Datei bitte durch die Tastenkombination *CTRL-x*. Alternativ können Sie natürlich gerne einen anderen Editor nehmen, um die Datei anzulegen (Kapitel 2.4).

Starten Sie nun die VPN-Verbindung durch Aufruf von

Code-Ausschnitt 3.1.12

```
1 sudo vpnc-connect vpn_firma
```

Die erfolgreiche Verbindung wird Ihnen angezeigt.

Code-Ausschnitt 3.1.13

```
1 Connect Banner:
2 | Connected to Firma LAN!
4 VPNC started in background (pid: 8435)...
```

PID bezeichnet dabei die *Process ID*, eine Nummer, unter welcher der Klient-VPN-Prozess identifizierbar ist.

 Der Befehl

Code-Ausschnitt 3.1.14

```
1 top
```

listet unter LINUX alle laufenden Prozesse mit und gibt neben der PID weitere wichtige Information wie Speicherverbrauch oder Laufzeit eines Programmes. Verlassen Sie die *top*-Ausgabe durch Drücken der Taste *q* für Quit. Kennen Sie zwar den Namen eines Prozesses, aber nicht die PID, können Sie diese durch Aufruf von

Code-Ausschnitt 3.1.15

```
1 ps -elf | grep name
```

herausfinden, wobei *name* den Namen des gesuchten Prozesses anzeigt. Für die *vpnc*-Verbindung könnte das Ergebnis wie folgt aussehen:

Code-Ausschnitt 3.1.16

```
1 1 S root 8435 1 0 80 0 - 1036 poll_s 21:01 ? 00:00:00 vpn-connect ↔
    vpn_firma
```

Manchmal hängt ein Prozess. Sie können diesen dann beenden, indem Sie

Code-Ausschnitt 3.1.17

```
1 sudo kill -9 pid
```

aufrufen. *pid* bezeichnet dabei die PID des zu beendenden Prozesses.

Durch einen Aufruf von

Code-Ausschnitt 3.1.18

```
1 ifconfig
```

kann man feststellen, dass es ein neues Netzwerk-Interface gibt:

Code-Ausschnitt 3.1.19

```
1 tun2      Link encap:UNSPEC Hardware Adresse
2 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
3          inet Adresse:172.16.0.11 P-z-P:172.16.0.11 Maske:255.255.255.255
4          UP PUNKTZUPUNKT RUNNING NOARP MULTICAST MTU:1412 Metrik:1
5          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
6          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
7          Kollisionen:0 Sendewarteschlangenlänge:500
8          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

In meinem Fall habe ich die IP-Adresse 172.16.0.11 zugewiesen bekommen. Nach Einwahl in das VPN-Netzwerk können Sie sich in diesem Netzwerk so bewegen, als wäre Ihr Raspberry Pi in diesem Netzwerk eingesteckt.

Die VPN-Verbindung wird durch Aufruf von

Code-Ausschnitt 3.1.20

```
1 sudo vpnconnect
```

beendet. Der Raspberry Pi bestätigt das Beenden durch

Code-Ausschnitt 3.1.21

```
1 Terminating vpn daemon (pid: 8435)
```

3.1.6 Microsoft PPTP

Neben dem Cisco-Protokoll ist auch Microsofts PPTP (*Point to Point Tunneling Protocol*) sehr beliebt. Dieser Abschnitt beschreibt die Verbindung mit einem Microsoft PPTP-Server. Zunächst installieren wir den PPTP-Klienten:

Code-Ausschnitt 3.1.22

```
1 sudo apt-get install pptp-linux
```

Danach legen wir im Verzeichnis */etc/ppp/peers* eine Datei *firma* an,

Code-Ausschnitt 3.1.23

```
1 sudo joe /etc/ppp/peers/firma
```

in die wir folgenden Inhalt kopieren:

Code-Ausschnitt 3.1.24

```
1 pty "pptp vpnserver --nolaunchpppd --debug"
2 name vpnbenutzer
3 password vpnkennwort
4 remotename PPTP
5 require-mppe-128
6 require-mschap-v2
7 refuse-eap
8 refuse-pap
9 refuse-chap
10 refuse-mschap
11 noauth
12 debug
13 persist
14 maxfail 0
15 defaultroute
16 replacedefaultroute
17 usepeerdns
```

Dabei ersetzen Sie bitte wie gehabt

- *vpnserver* mit dem Servernamen oder der IP-Adresse des VPN-Servers,
- *vpnbenutzer* mit dem Benutzernamen auf dem VPN-Server und
- *vpnkennwort* mit dem Kennwort des VPN-Benutzers.

Die Verbindung wird mit einem

Code-Ausschnitt 3.1.25

```
1 sudo pon firma
```

initiiert. Schauen Sie doch einmal in der System-Logdatei nach, wie die Verbindung aufgebaut wird:

Code-Ausschnitt 3.1.26

```
1 tail -f /var/log/syslog
```

Der Befehl *tail -f* zeigt dabei neu hinzukommende Zeilen in der Datei */var/log/syslog* an. Das *-f* steht dabei für *follow*, also „folgen“. Bei mir sieht der Verbindungsaufbau so aus:

Code-Ausschnitt 3.1.27

```

1 Jan 19 11:18:18 raspberrypi pppd[10166]: sent [IPCP TermAck id=0x0]
2 Jan 19 11:18:18 raspberrypi pppd[10166]: rcvd [CCP ConfReq id=0x0 <mppe +H -M +S +L -D -<←
  C>]
3 Jan 19 11:18:18 raspberrypi pppd[10166]: sent [CCP ConfNak id=0x0 <mppe +H -M +S -L -D -<←
  C>]
4 Jan 19 11:18:18 raspberrypi pppd[10166]: rcvd [CCP ConfNak id=0x1 <mppe +H -M +S -L -D -<←
  C>]
5 Jan 19 11:18:18 raspberrypi pppd[10166]: sent [CCP ConfReq id=0x2 <mppe +H -M +S -L -D -<←
  C>]
6 Jan 19 11:18:18 raspberrypi pppd[10166]: rcvd [CCP ConfReq id=0x1 <mppe +H -M +S -L -D -<←
  C>]
7 Jan 19 11:18:18 raspberrypi pppd[10166]: sent [CCP ConfAck id=0x1 <mppe +H -M +S -L -D -<←
  C>]
8 Jan 19 11:18:18 raspberrypi pppd[10166]: rcvd [CCP ConfAck id=0x2 <mppe +H -M +S -L -D -<←
  C>]
9 Jan 19 11:18:18 raspberrypi pppd[10166]: MPPE 128-bit stateless compression-enabled
10 Jan 19 11:18:18 raspberrypi pppd[10166]: sent [IPCP ConfReq id=0x1 <compress-VJ Of 01> <←←
  addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
11 Jan 19 11:18:21 raspberrypi pppd[10166]: rcvd [IPCP ConfReq id=0x1 <addr <←
  195.243.247.27>]
12 Jan 19 11:18:21 raspberrypi pppd[10166]: sent [IPCP ConfAck id=0x1 <addr <←
  195.243.247.27>]
13 Jan 19 11:18:21 raspberrypi pppd[10166]: sent [IPCP ConfReq id=0x1 <compress VJ Of 01> <←←
  addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
14 Jan 19 11:18:21 raspberrypi pppd[10166]: rcvd [IPCP ConfRej id=0x1 <compress VJ Of 01>]
15 Jan 19 11:18:21 raspberrypi pppd[10166]: sent [IPCP ConfReq id=0x2 <addr 0.0.0.0> <ms-←←
  dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
16 Jan 19 11:18:21 raspberrypi pppd[10166]: rcvd [IPCP ConfNak id=0x2 <addr 172.16.0.11> <←←
  ms-dns1 195.243.247.216> <ms-dns2 195.243.247.217>]
17 Jan 19 11:18:21 raspberrypi pppd[10166]: sent [IPCP ConfReq id=0x3 <addr 172.16.0.11> <←←
  ms-dns1 195.243.247.216> <ms-dns2 195.243.247.217>]
18 Jan 19 11:18:21 raspberrypi pppd[10166]: rcvd [IPCP ConfAck id=0x3 <addr 172.16.0.11> <←←
  ms-dns1 195.243.247.216> <ms-dns2 195.243.247.217>]
19 Jan 19 11:18:21 raspberrypi pppd[10166]: replacing old default route to eth0 <←
  [192.168.178.1]
20 Jan 19 11:18:21 raspberrypi pppd[10166]: local IP address 172.16.0.11
21 Jan 19 11:18:21 raspberrypi pppd[10166]: remote IP address 195.243.247.27
22 Jan 19 11:18:21 raspberrypi pppd[10166]: primary DNS address 195.243.247.216
23 Jan 19 11:18:21 raspberrypi pppd[10166]: secondary DNS address 195.243.247.217
24 Jan 19 11:18:21 raspberrypi pppd[10166]: Script /etc/ppp/ip-up started (pid 10194)
25 Jan 19 11:18:22 raspberrypi pppd[10166]: Script /etc/ppp/ip-up finished (pid 10194), <←
  status = 0x0
26 Jan 19 11:18:23 raspberrypi ntpd[3010]: Listen normally on 10 ppp0 172.16.0.11 UDP 123
27 Jan 19 11:18:23 raspberrypi ntpd[3010]: peers refreshed

```

Sie können die PPTP VPN-Verbindung mit dem Befehl

Code-Ausschnitt 3.1.28

```
1 sudo poff firma
```

wieder trennen.

3.2 Tanze Samba mit mir



Abbildung 3.7: Das Samba Logo (Quelle: www.samba.org).

Samba bezeichnet in der Computersprache keinen Tanz, sondern steht für eine freie Software, die das *Service-Message-Block*-Protokoll (auch *SMB* genannt) für UNIX-Systeme verfügbar macht. *SMB* ist ein Netzwerk-Protokoll, welches unter anderem Datei- und Druckdienste von Microsoft Windows emulieren kann. Ziel der nächsten Übung ist es also, mit einem Microsoft Datei-Explorer auf unser Raspberry Pi-Dateisystem zuzugreifen. Das *SMB*-Protokoll wird auch manchmal - unter anderem von Apple - *CIFS* (*Common Internet File System*) genannt. Zunächst installieren wir mit dem Paketmanager den Samba-Server:

Code-Ausschnitt 3.2.1

```
1 sudo apt-get install samba samba-common-bin
```

Im nächsten Schritt muss die Datei `/etc/samba/smb.conf` geändert werden.

Code-Ausschnitt 3.2.2

```
1 sudo joe /etc/samba/smb.conf
```

In dieser Datei entfernen wir den Kommentar vor `# security = user` und ändern die Zeile in

Code-Ausschnitt 3.2.3

```
1 security = user
```

Anschließend speichern wir die Datei mit `CTRL-x`. Mit dem folgenden Befehl wird für den Benutzer `pi` ein Samba-Kennwort festgelegt. Dieses sollte nicht unbedingt dem `ssh`-Kennwort entsprechen, um die Sicherheit zu erhöhen.

Code-Ausschnitt 3.2.4

```
1 sudo smbpasswd -a pi
```

Nach Änderungen an der Datei `/etc/samba/smb.conf` muss der Samba-Dienst neu gestartet werden. Das kennen Sie bereits aus dem Kapitel 2.6 über Dienste.

Code-Ausschnitt 3.2.5

```
1 sudo /etc/init.d/samba restart
```

Sie haben jetzt bereits Zugriff von Windows auf das Dateisystem des Pi. Öffnen Sie doch einmal einen Windows-Dateiexplorer und geben in der Adresszeile `\\IP-Adresse\pi` ein, wobei die *IP-Adresse* die Adresse oder der Name des Pi ist. Falls Sie das letzte `\pi` weglassen, sehen Sie alle Druckerfreigaben auf dem Pi und können sogar von Windows aus darauf zugreifen. Näheres hierzu erfahren Sie im Kapitel 8.1. Sollen nur bestimmte Verzeichnisse freigegeben werden, kann dies durch eine neue Sektion in der Datei `/etc/samba/smb.conf` erfolgen:

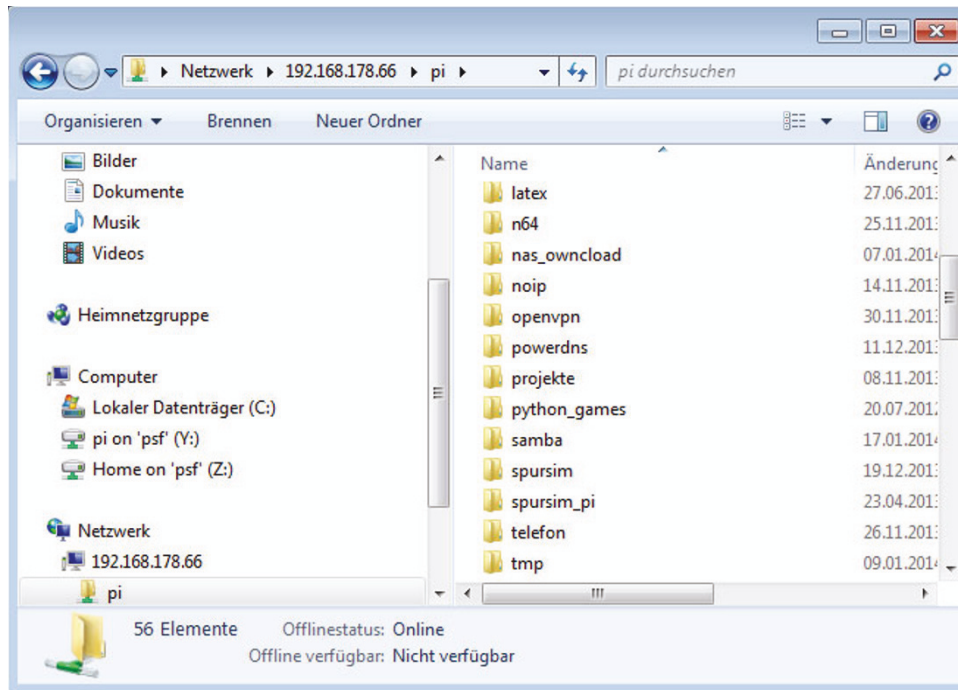


Abbildung 3.8: Samba zeigt Raspberry Pi-Dateien im Windows Explorer.

Code-Ausschnitt 3.2.6

```
1 [USBstick]
2 path = /media/usb1
3 writeable = yes
4 guest ok = no
```

Der oben gezeigte Eintrag würde einen eingebundenen USB-Stick (*/media/usb1*) per Samba zur Verfügung stellen. Weiterhin muss man sicherstellen, dass der Benutzer *pi*, den wir eben für Samba eingerichtet haben, Lese- und Schreibrechte am eingebundenen USB-Stick hat. Dies erledigen wir mit dem Kommando

Code-Ausschnitt 3.2.7

```
1 sudo chown -R pi:pi /media/usb1
```

Der *change owner (chown)*-Befehl ändert den Besitzer des Verzeichnisses */media/usb1* (also unseren USB-Stick) auf den Benutzer *pi* und die Benutzergruppe *pi*, welcher der Benutzer *pi* standardmäßig angehört. Bitte vergessen Sie nicht, den *Samba*-Dienst nach Änderungen an der Konfigurations-Datei neu zu starten. Mitunter kann es auch erforderlich sein, die sogenannte *WORKGROUP* zu ändern. Auch das können Sie bequem in der Datei */etc/samba/smb.conf* erledigen. Der Eintrag lautet üblicherweise

Code-Ausschnitt 3.2.8

```
1 workgroup = WORKGROUP
```

WORKGROUP kann aber jederzeit durch einen anderen Gruppennamen ersetzt werden. Die neue Freigabe wird unter Windows mit `\\IP-Adresse\USBStick` aufgerufen.



Externe Samba-Freigaben kann man wie folgt in den Raspberry Pi einbinden:

Code-Ausschnitt 3.2.9

```
1 sudo mount -t cifs //IP-Adresse/Freigabe /mnt -o username=Benutzername,↔  
password=Benutzerkennwort
```

Bitte ersetzen Sie dabei

- *IP-Adresse* durch die IP-Adresse oder den Namen des Gerätes, auf dem sich die Freigabe befindet,
- *Freigabe* durch den Namen der Freigabe,
- *Benutzername* durch den Namen des Benutzers und
- *Benutzerkennwort* durch das Kennwort des Benutzers.

Natürlich müssen Sie nicht das leere Verzeichnis */mnt* nehmen, um die Freigabe einzuhängen. Sie können auch ein beliebiges anderes Verzeichnis nehmen.

3.3 iSCSI ist kein Apple-Device

Kommen wir zu einer weiteren Möglichkeit, entfernte Geräte oder Verzeichnisse in unseren Raspberry Pi einzubinden. *iSCSI* klingt wie ein neues Apple-Gerät, ist es aber nicht. Der Name steht für *internet Small Computer System Interface* und ermöglicht die Nutzung des SCSI-Protokolls über ein Netzwerk, nämlich über *TCP (Transmission Control Protocol)*. Viele kennen vielleicht noch SCSI-Geräte-Controller, wie sie früher beispielsweise bei Bandlaufwerken oder den ersten CD-ROM-Laufwerken üblich waren. Einfacher ausgedrückt: Mit Hilfe des iSCSI-Protokolls kann ein DVD-Brenner, der sich in einem anderen PC befindet, so in den Raspberry Pi eingebunden werden, als wäre der Brenner direkt mit dem Pi verbunden. Neuere Raspbian-Kernel unterstützen inzwischen iSCSI. Falls Ihr Kernel das nicht unterstützen sollte, erfahren Sie im Kapitel 4.4.2, wie ein neuer Kernel kompiliert wird, in dem Sie das iSCSI-Modul aktivieren können. Zunächst installieren wir *iSCSI*:

Code-Ausschnitt 3.3.1

```
1 apt-get install open-iscsi
```

Auf dem Gerät, welches eingebunden werden soll (das sogenannte *Target* oder Ziel), muss ein iSCSI-Server laufen. Natürlich können wir auch auf dem Pi einen solchen Server installieren. Diese Vorgehensweise wird aber hier nicht beschrieben.

In meinem Heimnetz befindet sich ein LG-NAS-Laufwerk mit einem Bluray-Brenner, den ich für den Pi verfügbar machen möchte. Der Befehl

Code-Ausschnitt 3.3.2

```
1 sudo iscsiadm --mode discovery --type sendtargets --portal 192.168.178.200
```

fragt alle iSCSI-Targets der internen IP-Adresse 192.168.178.200 (mein LG-NAS) ab und sendet sie als Antwort an die Konsole:

Code-Ausschnitt 3.3.3

```
1 192.168.178.200:3260,1 iqn.2009-01.com.lge:LG-NAS.815EB4
```

Der Eintrag Ihres iSCSI-Targets sieht natürlich entsprechend anders aus. Bei mir wurde auf der IP-Adresse ein iSCSI-Target gefunden, welches den Namen *iqn.2009-01.com.lge:LG-NAS.815EB4* hat. Das iSCSI-Target wird nun wie folgt verbunden:

Code-Ausschnitt 3.3.4

```
1 sudo iscsiadm --mode node -T "iqn.2009-01.com.lge:LG-NAS.815EB4" --loginall=all
```

Den Namen Ihres iSCSI-Targets müssen Sie entsprechend anpassen. Der Raspberry Pi bestätigt die Verbindung mit

Code-Ausschnitt 3.3.5

```
1 Logging in to [iface: default, target: iqn.2009-01.com.lge:LG-NAS.815EB4,
2 portal: 192.168.178.200,3260] (multiple)
3 Login to [iface: default, target: iqn.2009-01.com.lge:LG-NAS.815EB4, portal:
4 192.168.178.200,3260] successful.
```

Nach erfolgreichem Verbinden steht das Laufwerk so zur Verfügung, als wäre es direkt am Raspberry Pi angeschlossen. Bluray-Laufwerke erscheinen unter dem Raspberry Pi Device-Zweig als `/dev/sr0` beim ersten Laufwerk. Die `0` wird dann entsprechend hochgezählt für weitere Laufwerke. Ein kleiner Test zeigt, dass das Laufwerk wirklich vorhanden ist. Die Ausgabe des Befehls `ls -l /dev/sr0` zeigt

Code-Ausschnitt 3.3.6

```
1 brw-rw---T 1 root cdrom 11, 0 Jan 19 12:44 /dev/sr0
```

Wir könnten nun mit

Code-Ausschnitt 3.3.7

```
1 sudo apt-get install k3b
```

die LINUX CD/DVD-Suite installieren und starten. Diese findet das iSCSI-Laufwerk als eigenes (Abb. 3.9). Das Programm findet sich nach der Installation auf der X-Oberfläche unter *Start* → *Systemwerkzeuge* → *K3b*.

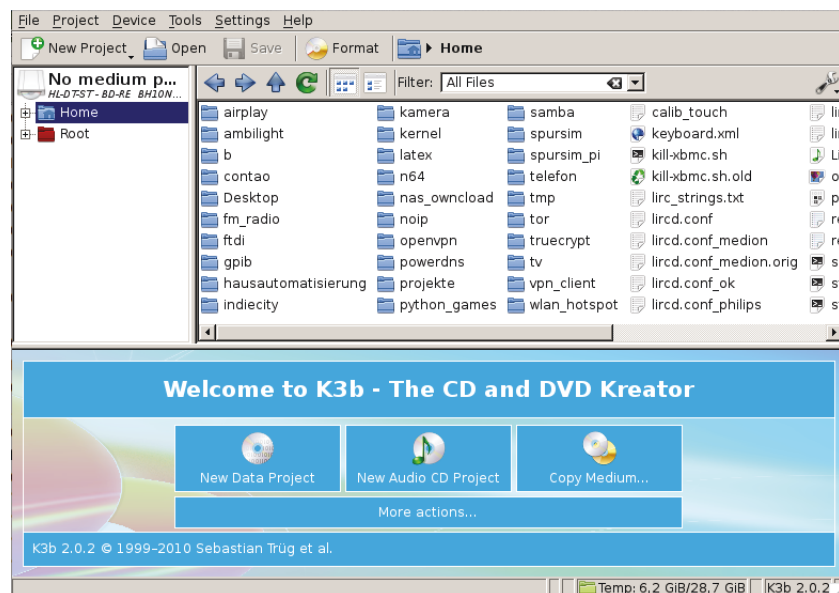


Abbildung 3.9: k3b DVD-Suite kann auf entfernte Brenner zugreifen, die mit *iSCSI* eingebunden sind.

Das *iSCSI*-Device kann mit folgendem Befehl wieder aufgehängt werden:

Code-Ausschnitt 3.3.8

```
1 sudo iscsiadm --mode node -T "iqn.2009-01.com.lge:LG-NAS.815EB4" --logout
```

3.4 Streng vertraulich!

In den vergangenen Abschnitten haben wir viel über das Einbinden von Verzeichnissen oder das Kopieren von Dateien gelernt. Was aber, wenn wir Dateien und Verzeichnisse auf unserem Pi verschlüsseln wollen, so dass nicht jeder sehen kann, was sich darauf befindet? Die Antwort heißt *veracrypt*. Das ist ein Programm, mit dessen Hilfe wir ein sogenanntes *Volume* erstellen können. Ein *Volume* ist eine Datei, die ihrerseits Verzeichnisse und Dateien enthält. In unserem Fall ist das *Volume* verschlüsselt und kann nur durch Eingabe des richtigen Kennwortes sichtbar gemacht werden. Im folgenden Abschnitt zeige ich Ihnen, wie man das Programm *veracrypt* installiert und verwendet. *veracrypt* ist aus dem Programm *truecrypt* hervorgegangen, dessen Entwicklung überraschenderweise vor einigen Jahren eingestellt worden ist. Es ist mir leider nicht gelungen, den Quelltext des Programms (Version 1.22) zu übersetzen. Der Linker steigt mit einem Fehler aus. Das ist aber nicht weiter schlimm, weil eine installierbare Version für den Raspberry Pi zur Verfügung steht.

Zunächst öffnen wir ein *LXTerminal* auf der grafischen Oberfläche und installieren ein paar Abhängigkeiten, die *veracrypt* voraussetzt.

Code-Ausschnitt 3.4.1

```
1 sudo apt-get update
2 sudo apt-get install libfuse-dev makeself libwxbase3.0-0v5 wget
```

Anschließend wechseln wir in das *home-Verzeichnis* (Kapitel 3.5) und erstellen dort ein Verzeichnis namens *veracrypt*. Wir wechseln in dieses Verzeichnis und laden die Setup-Datei für *veracrypt* mit Hilfe des Programmes *wget* herunter:

Code-Ausschnitt 3.4.2

```
1 cd ~
2 mkdir veracrypt
3 cd veracryptfiles
4 wget -L -O veracrypt-1.21-raspbian-setup.tar.bz2 https://launchpad.net/veracrypt/trunk←
   /1.21/+download/veracrypt-1.21-raspbian-setup.tar.bz2
```

Im nächsten Schritt packen wir die Dateien aus (*tar*) und machen die Installationsdatei ausführbar (*chmod +x*).

Code-Ausschnitt 3.4.3

```
1 tar xvjf ../veracrypt-1.21-raspbian-setup.tar.bz2
2 chmod +x veracrypt-1.21-setup-*
```

Der letzte Befehl macht sowohl das Installationsprogramm für die Konsole (*veracrypt-1.21-setup-console-armv7*) als auch das für die GUI (Graphical User Interface, *veracrypt-1.21-setup-gui-armv7*) ausführbar. Wir starten die Installation für das Konsolen-Programm:

Code-Ausschnitt 3.4.4

```
1 ./veracrypt-1.21-setup-console-armv7
```

Bitte führen Sie während der Installation die folgenden Schritte aus:

Code-Ausschnitt 3.4.5

```
1 VeraCrypt 1.21 Setup
2 -----
3 Installation options:
4 1) Install veracrypt_1.19_console_armv7.tar.gz
5 2) Extract package file veracrypt_1.19_console_armv7.tar.gz and place it to /tmp
7 To select, enter 1 or 2: '1'
```

Vor der eigentlichen Installation müssen die Lizenzbedingungen akzeptiert werden:

Code-Ausschnitt 3.4.6

```
1 Press Enter to display the license terms... 'Enter'
```

Die Lizenzbedingungen werden angezeigt. Mit *Enter* oder *Space* können Sie sich diese Bedingungen anschauen, durch Eingabe von *q* können Sie den Anzeigemodus verlassen.

Code-Ausschnitt 3.4.7

```
1 Press Enter or space bar to see the rest of the license.
3 VeraCrypt License
4 Software distributed under this license is distributed on an AS
5 IS BASIS WITHOUT WARRANTIES OF ANY KIND. THE AUTHORS AND
6 DISTRIBUTORS OF THE SOFTWARE DISCLAIM ANY LIABILITY. ANYONE WHO
7 USES, COPIES, MODIFIES, OR (RE)DISTRIBUTES ANY PART OF THE
8 SOFTWARE IS, BY SUCH ACTION(S), ACCEPTING AND AGREEING TO BE
9 BOUND BY ALL TERMS AND CONDITIONS OF THIS LICENSE. IF YOU DO NOT
10 ACCEPT THEM, DO NOT USE, COPY, MODIFY, NOR (RE)DISTRIBUTE THE
11 SOFTWARE, NOR ANY PART(S) THEREOF.
13 VeraCrypt is multi-licensed under Apache License 2.0 and
14 the TrueCrypt License version 3.0, a verbatim copy of both
15 licenses can be found below.
17 : 'q'
```

Bitte geben Sie Ihr Einverständnis zu der Lizenz durch Eingeben von *yes*:

Code-Ausschnitt 3.4.8

```
1 Do you accept and agree to be bound by the license terms? (yes/no): 'yes'
```

Das Installationsprogramm gibt Hinweise aus, wie etwa *veracrypt* deinstalliert werden kann und fragt dann nach dem *root*-Kennwort, damit die ausführbaren Dateien an den dafür vorgesehenen Ort kopiert werden können. Danach beginnt die Installation.

Code-Ausschnitt 3.4.9

```
1  Uninstalling VeraCrypt:
2  -----
4  To uninstall VeraCrypt, please run 'veracrypt-uninstall.sh'.
6  Installing package...
7  [sudo] password for '[your user]': '*****'
9  usr/
10 usr/share/
11 usr/share/veracrypt/
12 usr/share/veracrypt/doc/
13 usr/share/veracrypt/doc/License.txt
14 usr/share/veracrypt/doc/VeraCrypt User Guide.pdf
15 usr/share/pixmaps/
16 usr/share/pixmaps/veracrypt.xpm
17 usr/share/applications/
18 usr/share/applications/veracrypt.desktop
19 usr/bin/
20 usr/bin/veracrypt
21 usr/bin/veracrypt-uninstall.sh
23 Press Enter to exit... 'Enter'
```

Wenn Sie möchten, können Sie die Installationsdateien löschen:

Code-Ausschnitt 3.4.10

```
1  rm -r veracryptfiles
```

Die Installation des Programmes mit grafischer Oberfläche verläuft analog.

3.4.1 Container erstellen

Nach dieser Vorarbeit wird es Zeit, einen verschlüsselten Container zu erstellen. Dieser verschlüsselte Container kann auch eine komplette Partition/Festplatte sein. Das Installationsprogramm kopiert *veracrypt* in das Verzeichnis */usr/local/bin*. Alle Dateien in diesem Verzeichnis können im aktuellen Pfad des Terminals aufgerufen werden. In unserem Fall lautet dieser Aufruf einfach *veracrypt*. Rufen Sie das Programm doch mal mit der Option

Code-Ausschnitt 3.4.11

```
1  veracrypt --help
```

auf, um weitere Hilfe zu erhalten.

Jetzt erstellen wir eine Container-Datei für *veracrypt*, binden sie in ein Verzeichnis ein, erstellen eine geschützte Datei im Container und binden den Container wieder aus. Einen neuen Container erstellt man durch den Aufruf von

Code-Ausschnitt 3.4.12

```
1  veracrypt -t -c
```



Wenn Sie *truecrypt* ohne *sudo* starten, kann der Container nur im Verzeichnis des Benutzers, also in unserem Fall *pi*, erstellt werden.

Nehmen Sie die Einstellungen wie folgt vor:

Code-Ausschnitt 3.4.13

```

1  Volume type:
2   1) Normal
3   2) Hidden
4  Select [1]: 1

6  Enter volume path: /home/pi/container

8  Enter volume size (sizeK/size[M]/sizeG): 50M

10 Encryption algorithm:
11  1) AES
12  2) Serpent
13  3) Twofish
14  4) Camellia
15  5) Kuznyechik
16  6) AES(Twofish)
17  7) AES(Twofish(Serpent))
18  8) Serpent(AES)
19  9) Serpent(Twofish(AES))
20 10) Twofish(Serpent)
21 Select [1]: 1

23 Hash algorithm:
24  1) SHA-512
25  2) Whirlpool
26  3) SHA-256
27  4) Streebog
28 Select [1]: 1

30 Filesystem:
31  1) None
32  2) FAT
33  3) Linux Ext2
34  4) Linux Ext3
35  5) Linux Ext4
36  6) NTFS
37  7) exFAT
38 Select [2]: 5

40 Enter password:
41 Re-enter password:

43 Enter PIM:

45 Enter keyfile path [none]:

47 Please type at least 320 randomly chosen characters and then press Enter:

```

Das erstellt einen 50 MB-großen Linux-Dateisystem-Container (*ext4*) namens */home/pi/container* und schützt diesen mit einem Kennwort. Soll der Container auch unter Windows verwendet werden, wählen Sie FAT32 als Dateisystem. Geben Sie für *password* ein sicheres Kennwort ein. PIM steht für *Personal Iterations Multiplier*. Höhere PIM-Werte stehen für mehr Sicherheit, benötigen aber längere Zeit für das Einbinden des Containers. Für einen ersten Test können Sie an dieser Stelle getrost *RETURN* drücken. In der Rubrik *keyfile* können Sie eine Datei angeben, die Schlüssel zur Verschlüsselung enthält. Auch diesen Schritt überspringen wir mit *RETURN*. Zum Abschluss geben Sie bitte 320 Zeichen in beliebiger Reihenfolge ein.

Nachdem der Container angelegt worden ist, binden wir ihn in das Verzeichnis */home/pi/mnt* ein, das wir davor erstellen. Danach stellen wir sicher, dass der Benutzer *pi* den Container auch ohne *root*-Rechte beschreiben darf.

Code-Ausschnitt 3.4.14

```

1 mkdir /home/pi/mnt
2 veracrypt /home/pi/container /home/pi/mnt
3 sudo chown pi:pi /home/pi/mnt

```

Beim Einbinden werden sie aufgefordert, Ihr Kennwort einzugeben, welches Sie beim Einrichten des Containers festgelegt haben. Falls Sie keinen *PIM* und kein *keyfile* festgelegt haben, drücken Sie bei den entsprechenden Abfragen wieder *RETURN*. Dasselbe gilt für versteckte Container.

Code-Ausschnitt 3.4.15

```

1 Enter password for /home/pi/container:
2 Enter PIM for /home/pi/container:
3 Enter keyfile [none]:
4 Protect hidden volume (if any)? (y=Yes/n=No) [No]:

```

Sie können nun beliebige Verzeichnisse und Dateien unterhalb von */home/pi/mnt* anlegen. Nach dem *de-mounten* des Containers mittels

Code-Ausschnitt 3.4.16

```

1 veracrypt -d /home/pi/mnt

```

befinden sich alle angelegten Verzeichnisse und Dateien in der Kennwort-geschützten Container-Datei */home/pi/container*.



veracrypt kann den Container nicht aushängen, wenn Sie sich noch im Verzeichnis des Containers befinden. Verlassen Sie daher das */home/pi/mnt*-Verzeichnis zuerst (*cd ..*), bevor Sie den Container aushängen. Andernfalls erhalten Sie die Fehlermeldung

Code-Ausschnitt 3.4.17

```

1 Error: umount: /home/pi/mnt: target is busy
2         (In some cases useful info about processes that
3         use the device is found by lsof(8) or fuser(1).)

```

3.4.2 Wer soll sich das denn merken?

Die Benutzung von *veracrypt* ist schon ein wenig aufwendig. Man muss sich ja nicht nur das Kennwort merken, sondern auch noch den *PIM* oder eine Schlüsseldatei. Einfacher wäre es, wenn man eine Abkürzung (einen sog. *alias*) definieren könnte, der das Ein- und Ausbinden des Containers erleichtern würde. Das ist unter LINUX kein Problem. Abkürzungen für das LINUX-Terminal (die sog. *bash*) werden in einer versteckten Datei im *home*-Verzeichnis des Benutzers hinterlegt. Diese Datei nennt sich *.bashrc*. Der Punkt zu Beginn der Datei zeigt an, dass es sich um eine versteckte Datei handelt. Wählen wir als Abkürzung zum Einbinden des Containers *veramount* und zum Aushängen *veraunmount* und tragen diese Abkürzungen mit den verbundenen Befehlen in die Datei *.bashrc* ein.

Code-Ausschnitt 3.4.18

```

1 printf "\nalias veramount=\"veracrypt /home/pi/container /home/pi/mnt --pim=0 --keyfiles←
   = --protect-hidden=no\" >> .bashrc
2 printf "\nalias veraunmount=\"veracrypt -d /home/pi/mnt\" >> .bashrc
3 source /home/pi/.bashrc

```

Der *source*-Befehl liest die Datei *.bashrc* für das gerade geöffnete Terminal ein und stellt damit unsere beiden neuen Abkürzungen zur Verfügung. Ab jetzt können Sie mit einem einfachen *veramount* und der Eingabe des Kennworts den *container mounten* und mit *veraunmount* aushängen.

Info Wenn Sie den Befehl

Code-Ausschnitt 3.4.19

```
1 alias
```

in einer Konsole eingeben, werden Ihnen alle Abkürzungen angezeigt.

Das war Ihnen zu schnell? Kein Problem: Im nächsten Kapitel wird noch einmal ausführlich beschrieben, wie Programme kompiliert werden können, wie *Makefiles* aufgebaut sind, wie man Quelltext *auschecken* kann und Programme starten kann. Aber bevor es soweit ist, machen wir noch einen kleinen Ausflug in das Dateiverzeichnis des Pi.

3.5 My home is my castle

Zum Abschluss dieses Kapitels werfen wir noch einen Blick auf das LINUX-Dateisystem. Einige Verzeichnisse wie */home/pi* oder */mnt* haben wir uns schon näher angeschaut.

Der Anfang des Dateisystems ist das Wurzel- oder *root*-Verzeichnis, welches mit */* beginnt. Auf dieses Verzeichnis direkt hat nur der Administrator, der unter LINUX *root* heißt, Zugriff. Vom *root*-Verzeichnis ausgehend, sind alle anderen Verzeichnisse eingehängt. Die folgende Tabelle gibt einen kurzen Überblick über die Verzeichnisse und ihre Bedeutung.

Verzeichnis	Bedeutung
<i>/</i>	Der Beginn des gesamten Verzeichnisbaumes
<i>/home</i>	Hierunter sind die Verzeichnisse der Benutzer eingehängt
<i>/bin</i>	Hier befinden sich die ausführbaren Programme des Pi
<i>/usr/local/bin</i>	Hier befinden sich selbst kompilierte Programme
<i>/opt/vc</i>	Hier sind Bibliotheken des Raspberry Pi und Beispiele
<i>/tmp</i>	Hier befinden sich temporäre Dateien
<i>/var/log</i>	Hier befinden sich log-Dateien
<i>/etc</i>	Hier befinden sich Konfigurationsdateien
<i>/etc/init.d</i>	Hier befinden sich die Systemdienste

Tabelle 3.1: Ein kleiner Überblick über das LINUX-Dateisystem.

Info Eine besondere Bedeutung kommt der Tilde (*~*) zu. Sie repräsentiert das */home*-Verzeichnis des aktuellen Benutzers. Mit einem

Code-Ausschnitt 3.5.1

```
1 cd ~
```

kann man direkt in das */home*-Verzeichnis wechseln. Dasselbe erreicht man auch durch Eingabe *cd* und dem anschließenden Drücken der *RETURN*-Taste.

Werfen wir noch einen näheren Blick in das */opt/vc*-Verzeichnis, da diesem auf dem Raspberry Pi eine besondere Bedeutung zukommt.

Hier findet sich die Firmware des Raspberry Pi, die grafischen Bibliotheken und Beispiele. Die Beispiele unter `/opt/vc/src/hello_pi` zeigen, welche Fähigkeiten im Pi stecken. Sie können in die jeweiligen Unterverzeichnisse wechseln und die Beispiele durch Aufruf von `make` übersetzen. Ein Beispiel gefällig?

Code-Ausschnitt 3.5.2

```
1 sudo su
2 cd /opt/vc/src/hello_pi/hello_video
3 make
4 ./hello_video
```



Ähnlich wie bei Windows kann man unter LINUX Programme starten, indem man ihren Namen in einem Terminal aufruft. Der Bezeichner `/` vor dem Namen gibt an, dass die Datei aus dem aktuellen Verzeichnis genommen werden soll. Das aktuelle Verzeichnis wird nämlich durch einen Punkt (`.`) abgekürzt. Ein

Code-Ausschnitt 3.5.3

```
1 cd .
```

hat folglich keine Auswirkungen. Es wechselt in das aktuelle Verzeichnis, in dem man sich ohnehin befindet. Sinn macht der Punkt (`.`) wiederum beim Kopieren einer Datei, wenn man als Ziel das aktuelle Verzeichnis angeben will.

Zusammenfassung 3 Im 3. Kapitel haben wir den Raspberry Pi mit Windows PCs vernetzt, sowohl im Heimnetz als auch im Firmennetz. Wir können jetzt den Bildschirm des Pi sehen, auch wenn wir nicht vor dem Monitor sitzen, an dem der Pi angeschlossen ist. Wir haben gelernt, grafische Programme des Pi auf anderen Rechnern darzustellen.

Über das Samba- und iSCSI-Protokoll können wir auf Verzeichnisse und Geräte anderer Rechner zugreifen. Dateien können mit Hilfe des Programmes `scp` vom oder zum Raspberry Pi kopiert werden, sowohl unter Windows als auch unter LINUX-Betriebssystemen.

Abschließend haben wir das Programm `veracrypt` heruntergeladen, installiert und ausprobiert.

Darüber hinaus haben Sie die wichtigsten Verzeichnisse im LINUX-Dateibaum kennengelernt. Im weiteren Verlauf dieses Buches setzen wir unsere Reise fort. Hier lernen Sie von Grund auf, wie man Programme kompiliert, die Firmware des Raspberry Pi aktuell hält oder sogar einen eigenen Kernel installiert. Aber hierzu mehr im nächsten Kapitel. ■

4 — Das erste Programm

Nachdem wir im letzten Kapitel ein erstes Programm im Schnelldurchlauf installiert haben, gehen wir nun einen kleinen Schritt zurück. Dieses Kapitel gibt einen Überblick darüber, wie man Programme im Quelltext aus sogenannten Repositories auscheckt, übersetzt und installiert. Wir werfen dabei einen Blick auf ein C/C++-Programm und gehen kurz auf Python ein, da beides später noch für verschiedene Projekte verwendet wird.



4.1 Hallo Welt!

Beim Lernen einer Programmiersprache startet man häufig mit der Ausgabe des Textes „Hello World!“. Beginnen wir also mit der Installation der C/C++-Compiler und mit der Installation des Tools *make*.

Code-Ausschnitt 4.1.1

```
1 sudo apt-get install make gcc g++
```

Wir werden jetzt ein kleines C/C++-Programm schreiben, es übersetzen und starten. Danach zeige ich Ihnen, wie wir mit Hilfe eines *Makefiles* den Übersetzungsprozess automatisieren können.

Bitte erzeugen Sie im ersten Schritt ein neues Verzeichnis *programme* innerhalb Ihres */home/pi*-Verzeichnisses und wechseln Sie in dieses Verzeichnis.

Code-Ausschnitt 4.1.2

```
1 mkdir programme  
2 cd programme
```

Editieren Sie die Datei *hallo.c* und füllen Sie sie mit folgendem Inhalt:

Code-Ausschnitt 4.1.3

```
1 #include <stdio.h>
3 int main(void)
4 {
5     puts("Hallo Welt!");
6     return 0;
7 }
```

Speichern Sie die Datei. Was geschieht im oberen *Listing*? In Zeile 1 wird eine sogenannte *include*-Datei eingebunden. Diese Datei beinhaltet Strukturen der Funktionsaufrufe, die im Laufe des weiteren C/C++-Programmes benötigt werden. Im oben angegebenen Beispiel ist das die Struktur der Funktion *puts()*, die nichts anderes macht, als den Text „Hallo Welt!“ auszugeben. Das Programm hat keinen Rückgabewert (*void*). Es wird mit einem *return 0;* beendet. Das Programm hat nur einen Hauptteil (*main*). Machen wir uns ans Übersetzen:

Code-Ausschnitt 4.1.4

```
1 gcc hallo.c
```

Nach einer kurzen Weile ist der C-Compiler fertig und der *Linker* hat das Programm *a.out* erstellt. Dieses können Sie direkt mit

Code-Ausschnitt 4.1.5

```
1 ./a.out
```

aufrufen.



Unter Microsoft Windows™ haben ausführbare Programme die Dateierdung *.exe*, was für *executable*, also ausführbare Datei, steht. Unter LINUX ist das nicht erforderlich: Programme, die das Attribut *+x* besitzen, sind ausführbar. Man kann eine Datei also mit dem Befehl

Code-Ausschnitt 4.1.6

```
1 chmod +x dateiname
```

ausführbar machen. Für *a.out* hat der Linker das schon für uns erledigt.

a.out ist kein schöner Name für ein Programm. Starten wir das Übersetzen erneut und geben dieses Mal den Namen für unser Programm mit an:

Code-Ausschnitt 4.1.7

```
1 gcc -o hallo hallo.c
```

Unser ausführbares Programm heißt nun *hallo* und liefert beim Aufruf dasselbe Ergebnis wie *a.out*.

Je nachdem, wie viele Programmteile und Bibliotheken zusammengebunden werden, wird der Aufruf von *gcc* ganz schön lang. Um diesen Vorgang übersichtlicher zu halten, wurde das *Makefile* erfunden. Ein einfaches *Makefile* sieht wie folgt aus:

Code-Ausschnitt 4.1.8

```

1 CC = gcc
2 CFLAGS = -Wall
3 LIBS = -lm

5 all:  hello

7 hello: hello.c
8      $(CC) $(CFLAGS) hello.c -o hello $(LIBS)

10 clean:
11      rm -rf *~ *.o *.so hello

```

Das *Makefile* sieht auf den ersten Blick komplizierter aus, als es ist. Zunächst wird der Umgebungsvariablen *CC*, die den Compiler beinhaltet, mitgeteilt, welcher Compiler verwendet werden soll. Hier steht *gcc* für den C-Compiler oder *g++* für den C++-Compiler. Das *g* in beiden Ausdrücken steht für *GNU*, das Synonym für freie Software. Danach enthält die Variable *CFLAGS* alle Compiler-Einstellungen, die berücksichtigt werden sollen. In unserem Fall ist das nur *-Wall* und heißt, dass alle Warnungen, die beim Übersetzen des Programmes entstehen, angezeigt werden sollen. Unter *LIB* sind alle Bibliotheken vermerkt, die in das fertige Programm eingebunden werden sollen. Wir binden die Mathematik-Bibliothek ein (*-lm*), ohne dass wir sie benutzen. Als erste Zeile führt ein *Makefile* das aus, was hinter *all:* steht. Bei uns gibt es hier einen Verweis auf *hello*, schauen wir also nach, was hinter *hello:* steht: zunächst einmal eine Abhängigkeit, nämlich die von der Datei *hello.c*. Ändert sich eine Abhängigkeit (beispielsweise weil die Datei *hello.c* geändert wird), weiß *make*, dass das Programm neu übersetzt werden muss. Es folgt der eigentliche Aufruf des Compilers in der Zeile 6. Am Ende des *Makefiles* gibt es noch eine Rubrik namens *clean:*. Hier steht, was beim Aufruf von *make clean* geschehen soll: Es werden alle Backup-Dateien des Editors *joe* gelöscht (*rm *~*), alle Objekt-Dateien, alle Bibliotheks-Dateien (die wir hier noch nicht verwenden) sowie das Programm *hello* selbst. Der Aufruf für die Übersetzung eines Programmes mittels *Makefile* gestaltet sich sehr einfach:

Code-Ausschnitt 4.1.9

```
1 make
```

ruft automatisch die Rubrik *all* auf. Das wiederum ruft die Rubrik *hello* auf und erstellt unser Beispiel. Der Aufruf von

Code-Ausschnitt 4.1.10

```
1 make clean
```

ruft den Reinigungsstrupp auf den Plan und löscht die Dateien, die in der *clean*-Rubrik des *Makefiles* angegeben sind.



Wiederholt sich ein Befehl immer wieder, können Sie in einer Konsole (also einem Terminal) einfach die ↑-Taste drücken. Jeder weitere Druck auf diese Taste ruft den Befehl auf, den Sie davor angegeben haben. Suchen Sie einen Befehl, den Sie bereits im Terminal eingegeben haben, können Sie *CTRL-r* drücken, gefolgt von den ersten Buchstaben des Befehls. *LINUX* ergänzt den Befehl dann automatisch, sobald es einen passenden gefunden hat, den Sie zuvor eingegeben haben.

Die Hochsprachen C und C++ sind Sprachen, bei denen das ausführbare Programm zuerst übersetzt (*kompiliert*) und gebunden (*gelinkt*) werden muss. Das ist ein Nachteil.

Der Vorteil dieser Sprachen ist aber, dass der erzeugte Code, also das ausführbare Programm, mit hoher Geschwindigkeit laufen kann. Im Gegensatz dazu gibt es auch Programmiersprachen, die nicht kompiliert, sondern zur Laufzeit interpretiert werden. Eine dieser Programmiersprachen schauen wir uns im nächsten Abschnitt näher an. Es sprengt den Rahmen dieses Buches, auf die Syntax der Programmiersprachen im Detail einzugehen. Hierzu gibt es viele gute Bücher und auch viele gute Internet-Referenzen, welche die Sprachen im Detail und mit vielen Beispielen erklären. Diese kurze Einführung soll lediglich dazu dienen, Ihnen ein Grundverständnis für die tollen Projekte zu geben, die Sie bald mit Hilfe dieses Buches umsetzen können.

4.1.1 Auslagerungsdatei

Windows arbeitet, wenn der Speicher knapp wird, mit einer Auslagerungsdatei, in die der (fehlende) Arbeitsspeicher ausgelagert wird. Auch unter LINUX gibt es diesen Mechanismus, die sogenannte *swap*-Datei oder *swap*-Partition. In der Regel richtet man einen *swap* ein, der maximal so groß ist wie der Hauptspeicher selbst.

Der Pi kommt fast immer ohne *swap* aus. Übersetzt man aber große Programme (wie das *XBMC* Mediacenter im Kapitel 6), reicht der kleine Hauptspeicher des Pi nicht aus, um alle Programmbestandteile zu einem lauffähigen Programm zu binden (*linken*). Dieses Problem tritt insbesondere bei dem ersten kleinen Pi mit nur 256 MB Speicher auf (Modell A). In diesen Fällen macht es Sinn, dem Pi eine *swap*-Datei an die Seite zu stellen und so den Speicher virtuell zu vergrößern. Da nur der Systemadministrator *root* das darf, verschaffen wir uns als erstes *root*-Rechte.

Code-Ausschnitt 4.1.11

```
1 sudo su
```

Danach erzeugen wir die *swap*-Datei. Beim Raspberry Pi empfehle ich einen *swap* von 1024 MB Größe. Damit können auf einem alten Modell-B Raspberry Pi alle Beispiele dieses Buches nachgestellt werden. Zu diesem Zweck schreiben wir die *swap*-Größe in die Konfigurations-Datei */etc/dphys-swapfile* und weisen die Datei dem *swap* zu:

Code-Ausschnitt 4.1.12

```
1 echo "CONF_SWAPSIZE=1024" > /etc/dphys-swapfile
2 dphys-swapfile setup
```

Danach muss die *swap*-Datei einmalig aktiviert werden mit dem Befehl

Code-Ausschnitt 4.1.13

```
1 dphys-swapfile swapon
```

Mit

Code-Ausschnitt 4.1.14

```
1 exit
```

kann man den Systemadministrator-Status wieder abmelden. Möchte man den *swap* wieder abmelden, damit der Pi nicht zu viel auf der SD-Karte schreibt, hilft der Befehl

Code-Ausschnitt 4.1.15

```
1 sudo dphys-swapfile swapoff
```

4.2 Hallo Python!



Abbildung 4.1:
(Quelle:
python.org).

Die wohl bekannteste Interpreter-Sprache ist *Python* (Abb. 4.1). Mit ihr lassen sich sehr schnell kleine (und auch große) Projekte realisieren. Wir werden diese Sprache später noch benutzen (Kapitel 9.2), um einen kleinen Webserver aufzusetzen. Falls *Python* auf Ihrem Pi noch nicht installiert sein sollte, holen Sie das bitte nach:

Code-Ausschnitt 4.2.1

```
1 sudo apt-get install python
```

Für *Python* gibt es unzählige Erweiterungen, von mathematischen über grafische Bibliotheken bis hin zu Bibliotheken, mit denen sich Spiele einfach programmieren lassen. Aber auch hier begnügen wir uns mit einem einfachen Beispiel, damit Sie sehen, wie *Python* funktioniert. Schreiben wir also ein kleines Hallo Welt!-Beispiel.

Code-Ausschnitt 4.2.2

```
1 print "Hallo Python!"
```

Speichern Sie die Datei als *hallo.py*. Die typische Dateierdung *.py* signalisiert dabei, dass es sich bei unserem Programm um ein Python-Programm handelt. Nach dem Speichern rufen Sie bitte den Python-Interpreter auf und übergeben ihm als Parameter die eben erstellte Datei:

Code-Ausschnitt 4.2.3

```
1 python hallo.py
```

Ich vermute, Sie ahnen bereits, wie das Ergebnis aussehen wird. Da Python eine Interpreter-Sprache ist, ist sie langsam. Sie werden später noch lernen, wie man Python mit schnellen C/C++-Modulen kombinieren kann.



Die Python-Syntax kennt übrigens keine Semikolons oder Klammern als Zeilen- oder Blockende. Python regelt das über Einrückungen (*Indents*). Das kann bei großen Python-Programmen schon einmal unübersichtlich werden.

Für Python gibt es auch ein Programm, welches aus einer Python-Datei (wie *hallo.py*) eine vermeintlich ausführbare Datei macht: *PyInstaller*. Dieses in Python geschriebene Programmpaket packt den Python-Interpreter zusammen mit den Python-Dateien in einer neuen, ausführbaren Datei zusammen. Beim Aufruf (also Start des Programmes) werden Python-Dateien und der Interpreter dann in einem temporären Verzeichnis installiert und ausgeführt. In der Desktop-Version von Raspbian ist das Paketverwaltungsprogramm *pip* für Python-Pakete bereits vorinstalliert. Der Name *pip* ist ein rekursives Akronym und steht für „pip installs packages“. Mit dem Aufruf

Code-Ausschnitt 4.2.4

```
1 pip install pyinstaller
```

wird das Programm *PyInstaller* im Verzeichnis */home/pi/.local/bin/*, installiert, sofern Sie die Installation als Benutzer *pi* gestartet haben. Praktischerweise erkennt *pip* die erforderlichen Abhängigkeiten von anderen Python-Paketen und installiert diese gleich mit.

Jetzt können wir unser kleines Python-Testprogramm als lauffähige Version packen:

Code-Ausschnitt 4.2.5

```
1 /home/pi/.local/bin/pyinstaller --onefile --strip ./hallo.py
```

Mit `--onefile` weisen wir *PyInstaller* an, alles in eine Datei zu packen. Die Option `--strip` entfernt nicht benötigte Symbole aus der ausführbaren Datei und verkleinert sie damit. Die ausführbare Datei liegt im Verzeichnis *hallo/dist* und heißt einfach *hallo*. Sie kann - wie Sie es gelernt haben - mit `./hallo` gestartet werden.

4.3 Repositories

Das englische Wort *Repository* bedeutet auf Deutsch soviel wie *Lager* oder *Quelle*. Es ist ein Verzeichnis, in dem Quelltext gespeichert ist. Verschiedene Versionen des Codes werden in einer Datenbank verwaltet. Arbeiten mehrere Programmierer an einem Projekt, kann man so Unterschiede und Änderungen sehr leicht verfolgen. Hier stelle ich Ihnen die am häufigsten im Internet verwendeten Repository-Arten vor. Dabei schauen wir uns nur an, wie man Dateien auscheckt.

4.3.1 mercurial, svn und git

mercurial wird fast vollständig in Python entwickelt, welches Sie im vergangenen Kapitel kennengelernt haben. *Mercurial* hat seinen Namen vom englischen Wort für Quecksilber, *Mercury*, und wird mit *hg*, dem chemischen Element-Symbol von Quecksilber, abgekürzt. Dementsprechend beginnen alle Befehle zu *auschecken* von Software mit dem *hg*-Befehl. Installiert wird *mercurial* mit dem Befehl

Code-Ausschnitt 4.3.1

```
1 sudo apt-get install mercurial
```

Der wichtigste Befehl ist dabei der *clone*-Befehl, da er ein Verzeichnis aus dem Internet *cloned*, also lokal speichert. Die wichtigste Hilfe zum *clone* Befehl erhalten Sie mit

Code-Ausschnitt 4.3.2

```
1 hg help clone
```

Möchten Sie ein Verzeichnis *clonen*, müssen Sie nur die URL des Verzeichnisses (also den Platz im Web) angeben, wo das Verzeichnis steht, und einen Namen für das Verzeichnis, in das die Web-Dateien kopiert werden sollen. Möchten Sie *mercurial* selbst in der neuesten Version auschecken, lautet der Befehl dazu

Code-Ausschnitt 4.3.3

```
1 hg clone http://selenic.com/hg mercurial-repo
```

Der Befehl erstellt das Verzeichnis *mercurial-repo* und kopiert alle Dateien der URL `http://selenic.com/hg` dort hinein. Auf die weiteren Befehle von *hg* gehen wir dann ein, wenn wir sie benötigen. *svn* ist die Abkürzung von *subversion*. Man installiert es durch den Aufruf von

Code-Ausschnitt 4.3.4

```
1 sudo apt-get install git-svn
```

Der Aufruf, um ein Verzeichnis des Internets lokal zu speichern, ähnelt dem von *git.svn* kopiert das Internetverzeichnis durch den Aufruf von

Code-Ausschnitt 4.3.5

```
1 svn checkout http://internetverzeichnis lokales_verzeichnis
```

Ersetzen Sie bitte *internetverzeichnis* und *lokales_verzeichnis* entsprechend. *git*, das letzte in unserem Bunde, ist ebenfalls ein verteiltes Versions-Kontroll-System. Es bedeutet auf Englisch *Blödmann*. Man installiert es mit

Code-Ausschnitt 4.3.6

```
1 sudo apt-get install git-core
```

Der Befehl, mit dessen Hilfe man *git* dazu bewegen kann, ein Verzeichnis auszuchecken, lautet

Code-Ausschnitt 4.3.7

```
1 git init
2 git clone git://verzeichnis.git
```

Dabei müssen Sie *verzeichnis* mit dem Verzeichnis ersetzen, welches Sie lokal speichern möchten.

4.4 Betriebssystem hautnah

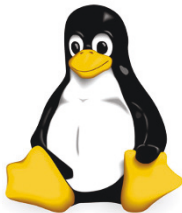


Abbildung 4.2: TUX der Pinguin (Quelle: L. Ewing).

Nach diesen grundlegenden Betrachtungen erkunden wir den Kern des Betriebssystems näher, den sogenannten Kernel, und die Firmware. Die Firmware besteht aus Bibliotheken und Include-Dateien, die von Broadcom bereitgestellt werden und unter anderem die Verbindung zur GPU herstellen. Damit unterscheidet sich der Raspberry Pi von vielen anderen Embedded Boards, die genau diesen Support und diese Unterstützung nicht erfahren. Die Firmware ist gut dokumentiert und wie ich Ihnen bereits gesagt habe, enthält der Zweig */opt/vc* neben der Firmware auch Beispiele zur GPU oder zum Audio-Dekoder. Bevor wir uns um den eigentlichen Kernel kümmern, folgt zunächst ein kleiner Abschnitt darüber, wie Sie die Firmware des Raspberry Pi aktuell halten können.

4.4.1 Die Raspberry Pi Firmware

Kennen Sie den Spruch „Never change a running system!“? Gerade im PC-Bereich enthält er viel Wahrheit. Warum sollte man ein System anfassen (also damit herumspielen, Updates einspielen), wenn es doch einwandfrei funktioniert! Hat dieser Spruch seine Berechtigung für Systeme, die einmal eingerichtet Ihren Dienst einwandfrei erledigen und keine neuen Aufgaben erfüllen müssen, so gibt es doch Gründe, die Firmware des Pi aktuell zu halten. Es existiert eine unendlich große Schar von Programmierern, die sich des Pi angenommen haben und diesen ständig verbessern. Verbessern heißt, neue Hardware wird lauffähig gemacht (wie ein Kameramodul), Fehler werden beseitigt, und neue Funktionen kommen hinzu.

Was gestern vielleicht noch nicht ganz einwandfrei funktionierte, wird morgen mit einer geänderten Firmware oder einem neuen Kernel funktionieren. Einer der Raspberry Pi-Entwickler mit dem Pseudonym *Hexxeh* hat sich der Thematik „Update der Firmware“ angenommen. In seinem *git*-Repository stellt er ein Skript zur Verfügung, welches das Update auf bequeme Art und Weise erledigt. Dieses Skript kann unter Raspbian einfach durch einen Aufruf von:

Code-Ausschnitt 4.4.1

```
1 sudo apt-get install rpi-update
```

installiert werden. Von nun an können Firmware-Updates einfach über den Befehl

Code-Ausschnitt 4.4.2

```
1 sudo rpi-update
```

eingespielt werden. In der Konsole erscheint dabei ein Protokoll über die installierten Dateien.



Das Firmware-Update installiert auch ab und an einen neuen Kernel. Bitte denken Sie daran, wenn Sie - wie im nächsten Abschnitt erklärt - einen eigenen Kernel am Start haben. Dieser kann mit einem Firmware-Update à la Hexxeh überschrieben werden. Sollte dies passieren, installieren Sie einfach Ihren eigenen Kernel neu. Wie das geht, erfahren Sie in Kapitel 4.4.2.

4.4.2 Der eigene Kernel

Colonel Hathi ist der Chef der Elefantenbande im Dschungelbuch von Rudyard Kipling. Der Chef des Raspbian-Betriebssystems ist auch ein Kernel, schreibt sich aber anders. Trotzdem hat er den Pi fest im Griff: Er stellt die Verbindung zwischen interner Hardware und externen Programmen dar. Er enthält also Treiber oder bietet Schnittstellen für externe Treiber an. Damit ergibt sich auch schon ein wichtiger Grund, einen eigenen Kernel zu übersetzen. Man kann so einen Treiber integrieren, den es noch nicht oder vielleicht auch nicht mehr gibt. Dieser Abschnitt zeigt Ihnen, wie Sie die Kernel-Quellen herunterladen und übersetzen können. Er erklärt weiterhin, wie der neue Kernel installiert und verwendet wird.

Der LINUX-Kernel ist so aufgebaut, dass er auf allen möglichen Plattformen funktioniert. Die Palette der Plattformen reicht dabei von Embedded Boards bis Hochleistungsrechnern oder Telefonen. Da nicht alle Benutzer jede Hardware haben, bietet der Kernel die Möglichkeit, nicht vorhandene Hardware nicht in den Übersetzungsprozess einzubeziehen. Das spart Platz im Speicher und auch auf der SD-Karte, setzt aber voraus, dass Sie die Hardware im Detail kennen um zu entscheiden, welche Bestandteile Sie benötigen und welche nicht. Beginnen wir damit, was wir im vorigen Abschnitt gelernt haben: Wir checken den Kernel aus.

Code-Ausschnitt 4.4.3

```
1 git init
2 git clone --depth 1 git://github.com/raspberrypi/linux.git
```

Die Option `--depth 1` kopiert dabei lediglich den aktuellen Stand des Kernels in unser lokales Verzeichnis, nicht aber die gesamte Historie. Sie können sich die aktuellen Arbeiten am Kernel auch in einem Webinterface unter <https://github.com/raspberrypi/linux> anschauen. Hier gibt es ebenfalls die Möglichkeit, den Kernel als gepacktes Archiv zu laden (→ Download ZIP). Nachdem der Kernel-Quelltext heruntergeladen wurde, müssen noch einige Pakete installiert werden, welche zum Übersetzen benötigt werden.

Code-Ausschnitt 4.4.4

```
1 sudo apt-get install gcc make bc ncurses-dev
```

Die ersten Pakete wie *gcc* oder *make* kennen Sie bereits. *bc* ist eine interaktive Algebrasprache, die zum Übersetzen des Kernels benötigt wird. Das Entwickler-Paket *ncurses* (daher der Anhang *-dev*) wird benötigt, um im Textterminal das Userinterface zur Konfiguration des Kernels darzustellen. Dies werden wir später noch beim Einbinden eines TFT-Display-Treibers benötigen. Um sicherzustellen, dass wir keine Altlasten im Kernel-Verzeichnis liegen haben (teilübersetzte Fragmente), wechseln wir zunächst mit einem *cd* in das Kernel-Verzeichnis und machen sauber. Nein, nicht *make clean*, sondern

Code-Ausschnitt 4.4.5

```
1 make mrproper
```

Die Einstellungen des Kernels, also welche Treiber beinhaltet sind, entweder als externes Modul oder als interner Treiber, werden in einer versteckten Konfigurations-Datei gespeichert.



Sie können Dateien verstecken, indem Sie dem Namen einen Punkt (.) voranstellen, z. B. *.dateiname*. Diese Dateien sind mit dem *ls -l*-Befehl nicht sichtbar. Um versteckte Dateien anzuzeigen, nutzen Sie bitte den Befehl *ls -la*. Der Parameter *a* zeigt alle Dateien an.

Die versteckte Datei, welche die Kernel-Konfiguration speichert, nennt sich *.config*. Sie ist beim Raspberry Pi als gezippte (also mit *zip* gepackte) Datei im */proc*-Verzeichnis gespeichert. Packen Sie die Datei aus und kopieren Sie sie als Kernel-Konfiguration ins aktuelle Kernel-Quelltest-Verzeichnis:

Code-Ausschnitt 4.4.6

```
1 zcat /proc/config.gz > .config
```

Alternativ können Sie auch die Datei verwenden, die sich im Verzeichnis *arch/arm/configs/bcmrpi_defconfig/.config* befindet. Im nächsten Schritt teilen wir dem neu zu erstellenden Kernel mit, dass die gerade kopierte Konfigurationsdatei verwendet werden soll. Das geschieht mit dem Aufruf

Code-Ausschnitt 4.4.7

```
1 make oldconfig
```

Möchten Sie Einstellungen ändern, d. h. neue Module einbinden oder andere ausbinden, hilft der Befehl

Code-Ausschnitt 4.4.8

```
1 make menuconfig
```

Sie können sich dann durch das sehr umfangreiche Menü des Kernels bewegen und Treiber als externe Module [*M*] einbinden oder direkt in den Kernelcode durch Markierung mit einem [*x*]. Falls Sie die Konfiguration geändert haben, vergessen Sie bitte nicht, diese zu speichern (*Save*). Dabei wird Ihre *.config*-Datei mit den neuen Einstellungen überschrieben. Jetzt machen wir uns aber daran, den Kernel zu übersetzen.

Code-Ausschnitt 4.4.9

```
1 make
2 make modules
```

make und *make modules* erstellt dabei den eigentlichen Kernel und alle externen Module.



Sicher haben Sie sich schon gefragt, welchen Sinn es hat, externe Module zu erstellen. Die Antwort ist einfach: Externe Module können bei Bedarf geladen und auch wieder entladen werden; so bleibt der eigentliche Kernel klein und damit schnell. Im Umgang mit externen Modulen helfen die Befehle

Code-Ausschnitt 4.4.10

```
1 lsmod
2 sudo modprobe modulname
3 sudo rmmod modulname
```

Der erste Befehl zeigt alle Module, die gerade in den Kernel eingebunden und/oder aktiv sind. Der zweite Befehl bindet ein Modul in den Kernel ein. Oft geschieht dies automatisch, beispielsweise beim Einstecken eines USB-Gerätes. Der Befehl *rmmod* entlädt ein Modul.

Der Übersetzungsvorgang für den Kernel dauert auf einem Raspberry Pi mehrere Stunden. Es ist also Geduld gefragt, oder der Pi muss über Nacht arbeiten. Nach erfolgter Übersetzung erstellen wir ein Verzeichnis, in welches die Module installiert werden, und lassen das Kernel-Makefile die Module dort hinein installieren:

Code-Ausschnitt 4.4.11

```
1 mkdir ~/modules
2 export MODULES_TEMP=~/.modules
3 make INSTALL_MOD_PATH=${MODULES_TEMP} modules_install
```

Der letzte Befehl erstellt im Verzeichnis */home/pi/modules* ein Verzeichnis namens *lib*, welches Sie nach */lib* kopieren müssen.

Code-Ausschnitt 4.4.12

```
1 sudo cp -r ~/modules/lib /lib
```

Im letzten Schritt kopieren wir den neuen Kernel an die Stelle, an der der Pi ihn beim Starten erwartet, und teilen der Datei */boot/config.txt* mit, dass ein neuer Kernel verwendet werden soll. Im Kernel Quellcode-Verzeichnis rufen wir

Code-Ausschnitt 4.4.13

```
1 sudo cp arch/arm/boot/Image /boot/kernel_neu.img
```

auf und fügen dann folgende Zeile in die Datei */boot/config.txt*:

Code-Ausschnitt 4.4.14

```
1 kernel=kernel_new.img
2 #kernel=kernel.img
```

Einen vorhandenen Kernel-Eintrag kommentieren wir aus. Das war's! Ein *sudo reboot* startet das System neu mit unserem frisch kompilierten Kernel.



Vergessen Sie bitte nicht, nach dem Einspielen eines neuen Kernels auch die Firmware des Raspberry Pi auf den neuesten Stand zu bringen (Kapitel 4.4.1).

4.5 Patches

Der letzte Abschnitt dieses Kapitels ist den Patches gewidmet. Patch bedeutet *Flickwerk* und das beschreibt es ziemlich genau. Ein Patch unter LINUX stellt ein eigenes Stück Quelltext dar, der in einen anderen Quelltext „geflickt“ wird. Installieren wir zunächst *patch*.

Code-Ausschnitt 4.5.1

```
1 sudo apt-get install patch
```

Im folgenden Abschnitt erstellen wir für zwei Dateien einen Patch, der den Unterschied der beiden Dateien beinhaltet. Anschließend führen wir den Patch aus, um so den Stand der ersten Datei auf den Stand der zweiten Datei zu bringen. Alles klar? Na dann los! Wechseln Sie innerhalb Ihres */home*-Verzeichnisses in ein Unterverzeichnis Ihrer Wahl und erstellen sie eine Datei *datei1.txt* mit dem Inhalt

Code-Ausschnitt 4.5.2

```
1 Text in Datei1
```

und eine zweite Datei namens *datei2.txt* mit dem Inhalt

Code-Ausschnitt 4.5.3

```
1 Text in Datei1
2 Text in Datei2
```

Die zweite Datei unterscheidet sich von der ersten Datei dadurch, dass sie eine weitere Zeile beinhaltet. Speichern Sie bitte beide Dateien. Jetzt schauen wir uns die Unterschiede zwischen beiden Dateien an:

Code-Ausschnitt 4.5.4

```
1 diff datei1.txt datei2.txt
```

Bei mir sieht das Ergebnis so aus:

Code-Ausschnitt 4.5.5

```
1 1a2
2 > Text in Datei 2
```



Wenn Sie die Ausgabe eines Befehls unter LINUX in eine Datei umleiten wollen, können Sie das *>*-Zeichen benutzen.

Nun erstellen wir einen Patch, der die Unterschiede beider Dateien beinhaltet. Den Patch können wir dann später auf *datei1.txt* anwenden und erhalten dann automatisch die Datei *datei2.txt*. Erstellen Sie den Patch mit

Code-Ausschnitt 4.5.6

```
1 diff -u -r -N datei1.txt datei2.txt > datei.diff
```

Der Parameter `-u` weist `diff` an, das Standard-Patch-Format zu nehmen. Der Parameter `-r` weist `diff` an, eventuelle Verzeichnisse, die Sie als Patch-Parameter angeben, rekursiv zu durchzusehen und in die zu erstellende Patch-Datei einzubeziehen. Der Parameter `-N` sorgt dafür, dass neue Dateien automatisch erzeugt werden. Das Ergebnis der `diff`-Operation leiten wir in die Datei `datei.diff` um:

Code-Ausschnitt 4.5.7

```
1 --- datei1.txt 2014-01-24 13:48:37.242375046 +0100
2 +++ datei2.txt 2014-01-24 13:48:44.792193204 +0100
3 @@ -1,2 +1,3 @@
4   Text in Datei 1
5 +Text in Datei 2
```

Wenden wir nun den erzeugten Patch auf die erste Datei an:

Code-Ausschnitt 4.5.8

```
1 cat datei.diff | patch
```

Der Befehl `cat` gibt den Inhalt des Patches aus. Das sogenannte Pipe-Zeichen (`|`) sorgt dafür, dass der Inhalt nicht im Terminal erscheint, sondern an das nächste Programm weitergegeben wird, also unser Patch-Programm. Das wiederum führt den in der Datei `datei.diff` beschriebenen Patch aus und ergänzt die Datei `datei1.txt` so, dass ihr Inhalt nach dem Patch dem der Datei `datei2.txt` entspricht.



Der Parameter `--dry-run` beim Patch-Kommando testet, ob der Patch ohne Probleme ausgeführt werden kann, führt ihn aber nicht aus. Das ist vor allem bei großen Patches sinnvoll, da man nicht immer weiß, ob die auch einwandfrei funktionieren.

Zusammenfassung 4 In diesem Kapitel haben wir uns angeschaut, wie man ein einfaches Programm in C/C++ oder Python schreibt und übersetzt bzw. ausführt. Dabei habe ich Ihnen auch gezeigt, wie ein Makefile das Übersetzerleben vereinfachen kann.

Sie haben gelernt, was ein Repository ist und wie man aus verschiedenen Repositories Dateien auschecken kann.

Darüber hinaus haben wir uns den Kernel des Raspberry Pi näher angeschaut. Sie wissen nun, wie man einen eigenen Kernel übersetzen und installieren kann. Das ist die Voraussetzung für das Einbinden von Treibern, die noch nicht im Kernel vorhanden sind.

Patch ist ein mächtiges Werkzeug, um Änderungen in Quelltext einzubauen. Wir werden dieses Werkzeug im Laufe dieses Buches noch das ein oder andere Mal nutzen.

Aber genug zu den Grundlagen! Nach diesem trockenen Kapitel machen wir uns endlich an die ersten Projekte mit dem Raspberry Pi. Fangen wir an mit dem großen Thema „Multimedia“.

5 — Video Disc Recorder

Nach dieser hoffentlich nicht zu trockenen Materie starten wir nun unser erstes Projekt „Video Disc Recorder“. Das Kapitel erklärt Schritt für Schritt den Aufbau eines DVB-Receivers (*Digital Video Broadcast*). Dabei spielt es keine Rolle, ob Sie Fernsehen über Satellit, über Kabel oder terrestrisch empfangen. Wir werden Klaus Schmidingers VDR (*Video Disk Recorder*) installieren, zusammen mit einer Infrarot-Fernbedienung und einem USB-DVB-Empfänger. Ich werde Ihnen zeigen, wie der Video-Disc-Recorder VDR mit Plugins erweitert werden kann, die das Aussehen verbessern oder sogar Streaming des Fernsehprogramms auf ein mobiles Gerät ermöglichen. Ein Web-Frontend erlaubt es sogar, Aufnahme-Timer aus der Ferne zu setzen.



5.1 Fernsehen



Abbildung 5.1: Das VDR-Portal ist die erste Anlaufadresse für Fragen rund um den VDR (Quelle: Jan Grell).

Der wohl bekannteste Empfänger für digitales Fernsehen ist der VDR von Klaus Schmidinger. Klaus selbst stellt ihn auf seiner Webseite <http://www.tvdr.de> vor, es gibt ein VDR-Wiki (<http://www.vdr-wiki.de>) und sogar ein sehr aktives Forum (<http://www.vdr-portal.de>) zu diesem Thema. VDR ist einfach zu bedienen. Er besitzt ein OSD (*On Screen Display*), kann mit DiSEqC (*Digital Satellite Equipment Control*) umgehen, hat eine Video-Schnittfunktion und ist netzwerkfähig. Er kann durch Plugins sehr einfach erweitert werden und sogar mit verschlüsseltem Fernsehprogramm umgehen. In diesem Kapitel richten wir eine Fernbedienung auf dem Raspberry Pi ein, installieren einen DVB-S-Stick für Satellitenfernsehempfang und

verbinden beides mit dem VDR. Für kabelgebundenen oder terrestrischen Empfang gehen Sie bitte genau so vor. Danach haben Sie einen Satellitenreceiver, der das Fernsehprogramm auf eine externe Festplatte aufzeichnen kann. Bleibt die Frage, warum wir keine fertige Distribution für den Pi nehmen, etwa die MLD (*Mini Linux Distribution*), die unter <http://www.minidvblinux.de> verfügbar ist. Wenn Sie so schnell wie möglich fernsehen möchten, ist das in Ordnung. Wenn Sie aber das System selbstständig erweitern möchten, ist dieses Kapitel genau das richtige für Sie.

5.2 DVB-Stick

Eine Liste mit DVB-Sticks, die unter LINUX funktionieren, finden Sie unter <http://www.vdr-wiki.de>. Im Prinzip gibt es mehrere Möglichkeiten, einen DVB USB-Stick unter LINUX einzubinden:

1. Sie erhalten den Treiber direkt vom Hersteller. Dieser liefert auch ein Installationsprogramm für LINUX. Ein Beispiel hierfür sind die Sundtek Sticks. Im Folgenden werden wir uns die Installation für einen Sundtek SkyTV Ultimate näher anschauen.
2. Sie kaufen einen Stick, dessen Treiber-Support bereits im Kernel enthalten ist, sie müssen aber die Firmware des Sticks installieren. Ein Beispiel hierfür ist der TeVii S2-660 Stick. Auch seine Installation wird gleich erklärt.
3. Sie kaufen einen Stick, der zwar von LINUX unterstützt wird, dessen Treiber aber noch nicht im Kernel vorhanden ist. Auch diese Installation wird beschrieben.

Wie auch immer Sie sich entscheiden, ich empfehle den Kauf eines HDTV-fähigen Sticks. Der Pi ist durchaus in der Lage, hochauflösendes Fernsehen anzuzeigen.

Sundtek

Die Firma Sundtek liefert für Ihre DVB USB-Geräte komplett eigene Treiber mit, deren Installation sich sehr einfach gestaltet. Weiterhin läuft der Treiber auf vielen Embedded Boards, wie dem Pi oder dem Pandaboard, selbst auf einigen NAS-Laufwerken. Natürlich lässt er sich auch auf einem PC installieren. Die Treiber für die Sundtek-Sticks werden unter http://www.sundtek.de/media/sundtek_netinst.sh zum Download angeboten. Da die Treiber als Systemadministrator installiert werden müssen, verschaffen wir uns zunächst *root*-Rechte. Dann wechseln wir in das temporäre Verzeichnis und laden den Treiber herunter. Beim Treiber selbst handelt es sich um ein Shell-Skript. Wir machen dieses ausführbar für alle (*chmod 777 sundtek_netinst.sh*) und rufen es dann auf. Das Skript lädt die passenden Treiberbestandteile von der Webseite des Herstellers und bindet sie so in das Betriebssystem ein, dass die Treiber selbst nach einem Neustart automatisch geladen werden.



Abbildung 5.2: Der Sundtek-Stick bringt seine eigenen Treiber mit (Quelle: Sundtek).

Code-Ausschnitt 5.2.1

```
1 sudo su
2 cd /tmp
3 wget http://www.sundtek.de/media/sundtek_netinst.sh
4 chmod 777 sundtek_netinst.sh
5 ./sundtek_netinst.sh
```

Die Treiberinstallation fordert Sie auf zu bestätigen, dass Sie fortfahren wollen. Machen Sie das. Der Treiber selbst wird im Verzeichnis */opt* installiert. Eine Liste der angeschlossenen Sticks erhalten Sie mit dem Kommando

Code-Ausschnitt 5.2.2

```
1 /opt/bin/mediaclient -e
```


Bei mir zeigt der Befehl Folgendes an:

Code-Ausschnitt 5.2.3

```
1 device 0: [Sundtek SkyTV Ultimate III (USB 2.0)] DVB-S/S2, REMOTE-CONTROL
2   [BUS]:
3     ID: 1-1.2
4   [SERIAL]:
5     ID: U120420173106
6   [DVB-S/S2]:
7     FRONTEND: /dev/dvb/adapter0/frontend0
8     DVR: /dev/dvb/adapter0/dvr0
9     DMX: /dev/dvb/adapter0/demux0
10  [REMOTECONTROL]:
11    INPUT0: /dev/mediainput0
```

Für Embedded Boards empfiehlt Sundtek, die PID (*Packet Identifier*) Hardware-Filter zu aktivieren. Dies kann mit dem Befehl

Code-Ausschnitt 5.2.4

```
1 /opt/bin/mediaclient -P on
```

erledigt werden. Tauschen sie das *on* gegen ein *off*, schalten Sie den PID Hardware-Filter wieder aus. Alle Einstellungen für die Sundtek-Sticks können in der Datei */etc/sundtek.conf* dauerhaft gespeichert werden:

Code-Ausschnitt 5.2.5

```

1 # ----- GLOBAL SECTION -----
2 #Set loglevel for logging to /var/log/mediasrv.log
3 #loglevel=[off|max] #default: min
4 #max .. little bit more debug
5 #Enable listening on network
6 #enablenetwork=[on|off] #default: off
7 #Lowest adapter number to start with, e.g. /dev/dvb/adapter5/frontend0
8 #first_adapter=5
9 #Call attach script when new device appears
10 #device_attach=[PATH_TO_SCRIPT] [PARAMETER|DEVID] #"DEVID" will
11 #automatically be replaced with the device ID
12 #Call detach script when device disappears
13 #device_detach=[PATH_TO_SCRIPT] [PARAMETER|DEVID] #"DEVID" will
14 #automatically be replaced with the device ID
15 # ----- Section for adapter with [SERIALNUMBER] -----
16 #Get adapter serial number with /opt/bin/mediaclient -e
17 #[SERIALNUMBER]
18 #Description register as dreambox tuner
19 #dreambox_support_fe1=[on|off] #default: off
20 #Infrared protocol to use
21 #ir_protocol=[RC5|NEC|RC6] #default: NEC
22 #Keymap to use, e.g. "/lib/udev/rc_keymaps/vp702x"
23 #rcmap=[PATH_TO_KEYMAP] #default: keymap which comes with
24 #the device
25 #Choose initial DVB mode for hybrid DVB-T/DVB-C devices only
26 #initial_dvb_mode=[DVBC|DVBT]
27 #Call attach script when new device appears
28 #device_attach=[PATH_TO_SCRIPT] [PARAMETER|DEVID] #"DEVID" will
29 #automatically be replaced with the device ID
30 #Call detach script when device disappears
31 #device_detach=[PATH_TO_SCRIPT] [PARAMETER|DEVID] #"DEVID" will
32 #automatically be replaced with the device ID
33 #Volume level
34 #volume=[0-127] #default: 118
35 # WSS callback (see http://en.wikipedia.org/wiki/Widescreen_signaling)
36 # trigger script when the videoformat changes between 16:9 or 4:3
37 #wss_callback=[scriptname]
38 # WSS_4_3_FULL
39 # WSS_14_9_LETTERBOX_CENTRE
40 # WSS_14_9_LETTERBOX_TOP
41 # WSS_16_9_LETTERBOX_CENTRE
42 # WSS_16_9_LETTERBOX_TOP
43 # WSS_16_9_LETTERBOX_DEEPER
44 # WSS_14_9_FULL_HEIGHT_4_3
45 # WSS_16_9_FULL_HEIGHT_16_9
46 # WSS_UNABLE_TO_DEMODULATE
47 #Timeout in ms after that WSS_UNABLE_TO_DEMODULATE will be called
48 #wss_demodulation_timeout=[MILLISECONDS]

```

Sundtek-Sticks haben einen Infrarot-Empfänger an Bord. Dieser steht unter `/dev/input/...` zur Verfügung. Wir benutzen aber die allgemeinere `lirc`-Methode, so dass Sie den Infrarot-Empfänger im Sundtek-Stick ohne Probleme ausschalten können. Das kann entweder bei der Installation durch Übergabe des Parameters `-nolirc` geschehen oder nach der Installation durch Aufruf von

Code-Ausschnitt 5.2.6

```

1 /opt/bin/mediaclient --disablirc

```

oder durch einen Eintrag in der Datei `/etc/sundtek.conf`:

Code-Ausschnitt 5.2.7

```
1 ir_disabled=1
```

Der LINUX-Treiber der Sundtek USB-Sticks unterstützt das Streamen von DVB-S/S2 über ein Netzwerk auf andere Linux- oder Windows-PCs. Um die Netzwerkunterstützung einzuschalten, geben Sie bitte den Befehl

Code-Ausschnitt 5.2.8

```
1 /opt/bin/mediaclient --enablenetwork=on
```

Tauschen Sie das *on* gegen ein *off*, um die Unterstützung wieder zu deaktivieren. Mit dem Befehl

Code-Ausschnitt 5.2.9

```
1 /opt/bin/mediaclient --mount=ip-adresse
```

stellen Sie den Stick dem Rechner mit der IP-Adresse *ip-adresse* zur Verfügung. Die Weitergabe des Sticks an einen Windows-PC ist ebenfalls möglich. Eine entsprechende Windows-Software kann man sich bei Sundtek herunterladen. Der Sundtek DVB-S2 Stick kostet übrigens 89 €. Um zu testen, ob der Stick richtig angesprochen werden kann, tunen Sie doch einfach mal auf einen Sender

Code-Ausschnitt 5.2.10

```
1 /opt/bin/mediaclient -m DVBS2 -f 1234000 -M QPSK -S 22000000 -V H -E 3/4
```

Die Antwort des Systems lautet:

Code-Ausschnitt 5.2.11

```
1 Using device: /dev/dvb/adapter0/frontend0
2 Setting DVB-S/S2 tune Parameters
3 using real frequency: 1234000
4 Modulation: QPSK (DVB-S)
5 Symbolrate: 22000000
6 Voltage: 18 Volt (Horizontal)
7 FEC: 3/4
8 Tone OFF
9 Frequency: 1234000
10 Syntax OK
11 Checking for lock:
12 .....
```

Hieran sehen wir, dass ein korrekt installierter DVB-Treiber unter LINUX neue Geräte (*devices*) angelegt hat. Das DVB-Device befindet sich unter */dev/dvb/adapter0*. Auf dieses Device greift später beispielsweise der *VDR* zu.



Der Sundtek USB-Stick hat ein eigenes Netzteil, welches die Spannung für einen Multi-switch zur Verfügung stellen kann. Der Stick selbst zieht - wie alle DVB-Sticks - aber so viel Strom, dass er an einem aktiven USB-Hub betrieben werden sollte, damit er am Pi einwandfrei funktioniert, spätestens wenn noch eine Festplatte zusätzlich angeschlossen werden soll. Alternativ können Sie auch ein USB-Y-Kabel verwenden und zusätzliche Spannung von einem USB-Netzteil zuführen. Das reduziert den Kabelsalat ein wenig.

DVBSky S960 V2



Abbildung 5.3: Die DVBSky S960 V2-Box benötigt eine Firmware-Datei (Quelle: amazon.de).

DVBSky S960 V2 (<http://dvbsky.net>) ist eine USB-Box, die ein wenig größer ist als der Sundtek-Stick. Dafür ist sie aber auch preiswerter. Die Box stellt ebenfalls ein Infrarot-Modul zur Verfügung, welches unter `(/dev/input/...)` zu finden ist. Der Treiber ist bereits im Kernel integriert und die Box wird beim Bootvorgang korrekt erkannt. Es fehlt lediglich noch die Firmware. Firmware ist eine Software, die während des Boot-Vorganges auf die Box geladen wird und deren Funktionalität sicherstellt. Dadurch hat der Hersteller die Möglichkeit, nachträgliche Verbesserungen für den Betrieb der Box in Form einer neuen Firmware herauszugeben. Die Firmware für diese Box erhalten Sie auf der Hersteller-Webseite unter <http://www.dvbsky.net/download/linux/>

`dvbsky-firmware.tar.gz`. Bitte laden Sie die Firmware herunter. Sie können dazu den `wget`-Befehl verwenden oder den Download aus einem Browser heraus starten. Nach dem Herunterladen muss die Firmware ausgepackt werden. Diese liegt im `tar.gz`-Format vor und kann mit dem Programm `tar` entpackt werden. Das Programm ist in der Raspbian-Distribution bereits vorinstalliert. Entpacken Sie die Firmware durch den Aufruf von

Code-Ausschnitt 5.2.12

```
1 tar xvfz dvbsky-firmware.tar.gz
```

Die eigentliche Firmware-Datei `dvb-usb-s660.fw` kopieren Sie bitte in das LINUX Firmware-Verzeichnis, welches sich unter `/lib/firmware` befindet. Dazu benötigen Sie `root`-Rechte.

Code-Ausschnitt 5.2.13

```
1 cd dvbsky-firmware
2 sudo ./copy-firmware.sh
```

Das kleine Skript `copy-firmware.sh` erledigt den Kopiervorgang für Sie und sorgt dafür, dass die Firmware-Dateien an ihren Bestimmungsort gelangen. Damit die Firmware-Dateien geladen werden können, ist ein Neustart erforderlich. Während des Bootens erkennen Sie die Karte in den Kernel-Logs (`dmesg`) durch folgende Einträge:

Code-Ausschnitt 5.2.14

```

1  usb 1-1.1.2: new high-speed USB device number 10 using dwc_otg
2  usb 1-1.1.2: New USB device found, idVendor=0572, idProduct=6831
3  usb 1-1.1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
4  usb 1-1.1.2: Product: S960
5  usb 1-1.1.2: Manufacturer: Bestunar
6  usb 1-1.1.2: SerialNumber: 20120511
7  usb 1-1.1.2: dvb_usb_v2: found a 'DVBSky S960/S860' in warm state
8  [usb 1-1.1.2: dvb_usb_v2: will pass the complete MPEG2 transport stream to the software ←
   demuxer
9  dvbdev: DVB: registering new adapter (DVBSky S960/S860)
10 usb 1-1.1.2: dvb_usb_v2: MAC address: 00:17:42:54:96:0c
11 i2c i2c-3: Added multiplexed i2c bus 4
12 ts2020 4-0060: Montage Technology TS2022 successfully identified
13 usb 1-1.1.2: DVB: registering adapter 0 frontend 0 (Montage Technology M88DS3103)...
14 Registered IR keymap rc-dvbsky
15 rc rc0: DVBSky S960/S860 as /devices/platform/soc/3f980000.usb/usb1/1-1/1-1.1/1-1.1.2/rc↔
   /rc0
16 input: DVBSky S960/S860 as /devices/platform/soc/3f980000.usb/usb1/1-1/1-1.1/1-1.1.2/rc/↔
   rc0/input2
17 usb 1-1.1.2: dvb_usb_v2: schedule remote query interval to 300 msec
18 usb 1-1.1.2: dvb_usb_v2: 'DVBSky S960/S860' successfully initialized and connected
19 usbcore: registered new interface driver dvb_usb_dvbsky

```

Die Bootzeiten habe ich aus Gründen der Übersichtlichkeit entfernt. Nach der erfolgreichen Initialisierung der DVBSky-Box steht auch hier das neue Device `/dev/dvb/adapter0` zur Verfügung, welches VDR zum Empfang der Kanäle benötigt.



Eventuell müssen Sie die USB DVB-Karte noch einmal entfernen und wieder neu verbinden, damit der Treiber die passende Firmware laden kann und der gewünschte Adapter danach zur Verfügung steht.

5.2.1 Neuere Modelle

Sollten Sie sich für einen USB-Empfänger entschieden haben, für den es noch keine Unterstützung im LINUX-Kernel gibt, könnten Sie mit den LINUX-DVB-Treibern Glück haben, welche Sie unter <http://www.linuxtv.org> erhalten. Die DVB-Treiber stellen einen ausgelagerten Kernel-Zweig dar. Nach ihrer Übersetzung stehen Kernel-Module neuerer Treiber zur Verfügung, die noch keinen Einzug in den offiziellen Kernel erfahren haben. Bevor das Übersetzen starten kann, müssen noch einige Pakete installiert werden.

Code-Ausschnitt 5.2.15

```

1  sudo apt-get install linux-source linux-headers make gcc git-core patch patchutils ←
   libproc-processtable-perl

```



Falls Sie bei `apt-get install` ein Paket angeben, das Sie bereits zuvor installiert haben, ist das nicht weiter schlimm. Sie erhalten dann lediglich eine Rückmeldung, dass das bestimmte Programm bereits installiert ist.

Die meisten Pakete, die zur Installation benötigt werden, haben Sie bereits im vorigen Kapitel kennengelernt. Neu hinzugekommen sind die Header- und Quellcode-Dateien des Kernels, die Patchutils (Hilfsprogramme für das Arbeiten mit `patch`) und ein Perl-Programm, um Tabellen zu prozessieren. Machen wir uns an das Auschecken und Übersetzen der Treiber. Dazu öffnen wir ein Terminal und geben die folgenden Befehle ein:

Code-Ausschnitt 5.2.16

```
1 git clone --depth=1 git://linuxtv.org/media_build.git
2 cd media\_build
3 ./build
```

Den Parameter `--depth=1` können Sie weglassen, falls die komplette Historie ausgecheckt werden soll. Das Starten der Übersetzung wird mit `./build` im Verzeichnis `media_build` initiiert.

Info Dieser Abschnitt beschreibt gängige *Makefile*-Optionen, wie sie bei fast allen Programmen zu finden sind.

- `sudo make install`: Installiert das übersetzte Programm. Default-Einstellung ist meist die Installation in `/usr/local/bin`.
- `sudo make rminstall`: Löscht die Installation wieder.
- `sudo make distclean`: Reinigt das *make*-Environment, so dass ein anschließendes *make* den Übersetzungsvorgang neu startet.
- `sudo make menuconfig`: Ruft ein Menü im Terminal auf, in dem Einstellungen zum Übersetzen vorgenommen werden können.

Wenn Sie keine Fehlermeldungen beim Übersetzen erhalten haben, installieren Sie die neu erstellten Treiber mit

Code-Ausschnitt 5.2.17

```
1 sudo make install
```

Sollte es Fehler beim Übersetzen gegeben haben, gibt Ihnen das Kapitel 12.7 einen kleinen Überblick, was man in diesem Fall machen kann. Als Letztes muss noch die Firmware des DVB-Gerätes installiert werden. Falls diese schon bekannt (und freigegeben ist), erledigt das ein

Code-Ausschnitt 5.2.18

```
1 sudo apt-get install firmware-linux
```

Falls die gewünschte Firmware nicht dabei ist, muss sie - wie im vorigen Abschnitt beschrieben - von der Webseite des Herstellers heruntergeladen und in das Verzeichnis `/lib/firmware` kopiert werden.

DL Einen vollständig übersetzten Zweig der DVB-Treiber finden Sie auch im Download-Bereich <https://www.springer.com/de/book/978-3-662-58143-8> unter dem Namen `v4l.tar.bz2`.

5.3 Fernbedienung



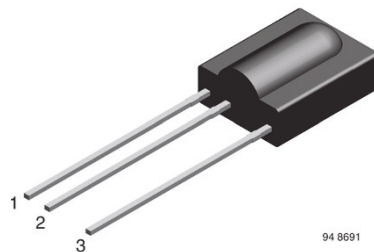
Abbildung 5.4: Philips Universalfernbedienung (Quelle: Philips).

Der erste Schritt in Richtung Fernsehempfang ist getan. Da Zappen mit der Tastatur aber keinen Spaß macht, werden wir in diesem Abschnitt einen Infrarot-Empfänger bauen, mit dem Pi verbinden und einen Treiber dafür installieren, bevor wir eine beliebige Fernbedienung anlernen. Während Sie wahrscheinlich eine Fernbedienung zu Hause herumliegen haben, die Sie ohne Probleme verwenden können, werden Sie eine kleine Infrarot-Diode kaufen müssen. Der Preis für eine solche Diode liegt bei ca. 1 €, ist also zu verschmerzen. Sollten Sie keine Fernbedienung zur Hand haben, kann ich Ihnen die Fernbedienung aus Abb. 5.4 empfehlen. Sie ist auf beliebige Marken programmierbar und bringt alle Tasten mit, die für den sinnvollen Betrieb eines *VDR* erforderlich sind. Dazu zählen unter anderem vier Farbtasten (rot, grün, gelb und blau), eine Menü- und eine OK-Taste sowie Pfeiltasten). Bitte programmieren Sie für diese Fernbedienung einen RC5-Fernbedienungscode, da dieser mit den meisten Infrarot-Dioden funktioniert, indem Sie die Fernbe-

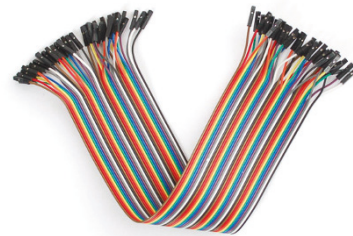
dienung z. B. auf „Medion“ programmieren. Die Fernbedienung kostet ca. 12 €. Aber bevor unsere Fernbedienung zum Einsatz kommt, benötigen wir erst einmal eine Infrarot-Diode. Ich kann Ihnen die TSOP 31236 von Vishay empfehlen. Diese kostet bei Reichelt (<http://www.reichelt.de>) ca. 0,80 €. Zur Befestigung am Pi benötigen wir noch drei kurze Kabel. Die benötigten Kabel gibt es fertig vormontiert mit Buchse, etwa bei Amazon für unter 3 €. Wir verbinden nun die drei Anschlüsse unseres Infrarot-Empfängers mit drei Kabeln, die wir dann im nächsten Schritt mit der GPIO¹-Leiste unseres Pi verbinden. Verbinden Sie nun die Anschlüsse des TSOP-Empfängers wie folgt mit dem Pi:



(a) TSOP 31236 (Quelle: Vishay Intertechnology, Inc.).



(b) 1=GND, 2= V_S , 3=OUT (Quelle: Vishay Intertechnology, Inc.).



(c) Kabel mit Buchse (Quelle: Adafruit).

Abbildung 5.5: Infrarot-Empfänger selbst gebaut: eine IR-Diode und drei Kabel reichen.

¹General Purpose Input Output (Ein- und Ausgabe-Pins zur freien Verfügung)

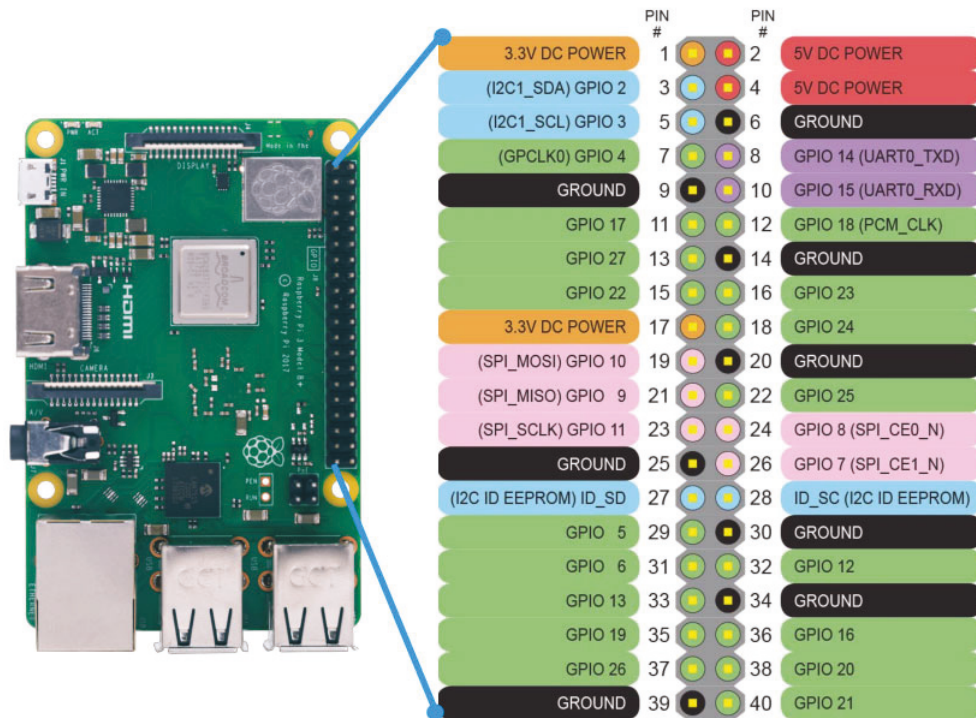


Abbildung 5.6: Die GPIO-Leiste des Raspberry Pi und ihre Möglichkeiten.

1. Den *GND*-Pin (Ground, Masse) verbinden Sie bitte mit dem *GND*-Pin des Raspberry Pi (Abb. 5.6). Dieser befindet sich in der rechten Reihe und ist der dritte Pin von oben (dunkelgrau).
2. Verbinden Sie bitte den Pin V_S (Spannungsversorgung) mit dem orange-farbenen 3,3 V Pin. Das ist der oberste Pin in der linken Reihe.
3. Verbinden Sie bitte den 3. Pin (OUT) mit dem GPIO Pin 18. Das ist der dritte Pin unter dem *GND* Pin in der rechten Reihe.

Wenn Sie eine andere Infrarot-Diode kaufen, lesen Sie bitte im Datenblatt nach, wie die Reihenfolge der drei Pins ist. Wenn Sie sie vertauschen, kann sowohl die Diode als auch der Raspberry Pi Schaden nehmen. Ersteres wäre sicher zu verschmerzen.

Info Bitte verkabeln Sie den Raspberry Pi nicht, so lange er noch an die Spannungsversorgung angeschlossen ist. Fahren Sie ihn erst herunter, trennen Sie ihn vom Netzteil und verkabeln Sie ihn dann.

Info Die meisten Infrarot-Dioden erlauben auch eine Spannung von 5,0 V. Sie können den Pin V_S dann auch mit dem 5 V-Pin des Pi verbinden. Das ist der oberste Pin in der rechten Reihe. Sie können den *OUT*-Pin theoretisch auch an einen anderen GPIO-Pin hängen (z. B. GPIO 12). Wir werden allerdings im weiteren Verlauf des Buches den Pin 18 nutzen, so dass es Sinn macht, diesen auch zu verwenden.



Der Treiber im Download-Bereich <https://www.springer.com/de/book/978-3-662-58143-8> nutzt den GPIO-Pin 18 des Raspberry Pi.

Wenn Sie den Raspberry Pi richtig verkabelt haben, sollte Ihre Verkabelung so aussehen wie in Abb. 5.7. Nachdem die Hardware nun angerichtet ist, kümmern wir uns um die Software.



Abbildung 5.7: Die fertige *lirc*-Verkabelung an einem alten Raspberry Pi. Die Verkabelung für einen neuen Raspberry Pi sieht genau so aus, wenn Sie die GPIO-Pins von rechts aus zählen.

Stellen Sie bitte zunächst sicher, dass das *lirc*-Kernel-Modul geladen wird. Editieren Sie dazu bitte die Datei `/boot/config.txt` als *root* und ändern Sie die folgende Zeile:

Code-Ausschnitt 5.3.1

```
1 dtoverlay=lirc-rpi,gpio_in_pin=18
```

Wahrscheinlich müssen Sie nur das Kommentar-Zeichen „#“ entfernen und den GPIO Input-Pin 18 deklarieren. Sollten Sie beispielsweise den Pin 12 ausgewählt haben, müssen Sie die 18 gegen die 12 ersetzen. Starten Sie danach Ihren Raspberry Pi neu. Schauen Sie nach einem Neustart nach, ob das *lirc*-Modul geladen wurde:

Code-Ausschnitt 5.3.2

```
1 lsmod | grep lirc
2 lirc_rpi          16384 0
3 lirc_dev          16384 1 lirc_rpi
```

Im nächsten Schritt installieren wir den *lirc*-Treiber:

Code-Ausschnitt 5.3.3

```
1 sudo aptitude install lirc
```

In früheren Versionen der Raspbian-Distribution wurden *lirc*-Parameter in der Datei `/etc/modules` eingetragen. Für *lirc* erfolgt das nun in der Datei `/boot/config.txt`. Für manche DVB-Sticks muss man aber Parameter angeben. Für meinen Cinergy DVB-T-Stick trage ich in der Datei `/etc/modules` ein:

Code-Ausschnitt 5.3.4

```
1 em28xx card="55"
2 em28xx_dvb
```

Ich lade also (permanent) den Treiber *em28xx* mit dem Parameter *card="55"* und anschließend das Modul *em28xx_dvb*. Gleichzeitig verbiete ich dem Betriebssystem, das Modul zu laden, damit es nur mit meinen angegebenen Parametern geladen wird. Diese Einstellung nehme ich in der Datei */etc/modprobe.d/raspi-blacklist.conf* vor:

Code-Ausschnitt 5.3.5

```
1 blacklist em28xx
```

Nun ist es aber an der Zeit, unsere Hardware einem ersten Test zu unterziehen. *lirc* ist unter LINUX ein Dienst - der Fernbedienungs-Dienst. Sicher erinnern Sie sich noch, wie Dienste unter LINUX gestartet oder gestoppt werden:

Code-Ausschnitt 5.3.6

```
1 sudo /etc/init.d/lircd start
2 sudo /etc/init.d/lircd stop
3 sudo pkill lircd
```

Den Befehl der dritten Zeile kennen Sie noch nicht. *pkill lircd* beendet alle Programme, die den String „lircd“ im Namen haben. Nach der Installation kann es sein, dass die Konfigurations-Datei für *lirc*, nämlich */etc/lirc/lircd.conf* noch nicht vorhanden ist und der Start von *lirc* mit der Fehlermeldung quitiert, dass diese Konfigurations-Datei noch nicht gefunden wird. Sie beschreibt, welche Tasten der Fernbedienung wie belegt sind. Diese Datei werden wir im weiteren Verlauf dieses Abschnitts erstellen. Aber wie versprochen, zunächst ein kleiner Test, ob wir alles richtig verkabelt haben. Stoppen Sie bitte als erstes den *lircd*, falls er läuft, und rufen Sie dann auf:

Code-Ausschnitt 5.3.7

```
1 sudo pkill lircd
2 sudo mode2 -d /dev/lirc0
```

Was passiert hier? *lirc* sucht seine Devices ebenfalls im */dev*-Verzeichnis. Der erste Empfänger ist */dev/lirc0*, der zweite wäre */dev/lirc1* usw. Dem Programm *mode2* übergeben wir das *lirc*-Device als Parameter. *mode2* zeigt die Puls- und Leerlänge von Infrarot-Impulsen an. Drücken Sie beliebige Tasten auf der Fernbedienung und schauen Sie sich das jeweilige Ergebnis im Terminal an. Brechen Sie das Programm durch Drücken der Tastenkombination *CTRL-c* ab.



Sie können laufende Programme im Terminal mit *CTRL-c* abbrechen. *CTRL-z* stoppt ein laufendes Programm. Erst ein *bg* (*background*) lässt es im Hintergrund weiterlaufen. Mit dem Befehl *fg programmname* (*fg* steht für *Foreground*) holen Sie das Programm wieder in den Vordergrund. Starten Sie ein Programm mit *programmname &*, startet das Programm im Hintergrund und das Terminal ist danach sofort wieder frei.

Wir werden uns nun im nächsten Schritt anschauen, welche Fernbedienungsschlüssel LINUX offiziell unterstützt, also beispielsweise den Namen *MENU* für die Menütaste Ihrer Fernbedienung. Das ist wichtig für Programme, die nur auf die richtigen KEYS reagieren, wie z. B. der *vomplient*, den wir uns später noch anschauen. Prinzipiell können Sie aber Namen für Fernbedienungstasten vergeben, wie es Ihnen beliebt.

Wechseln Sie in ein Verzeichnis, in dem Sie Schreibrechte haben und speichern Sie LINUX-Standschlüsselnamen in eine Datei namens *lirc_strings.txt* ab:

Code-Ausschnitt 5.3.8

```
1 irrecord --list-namespace | grep KEY &> lirc_strings.txt
```



Der Befehl *grep* sucht das Schlüsselwort, das hinter ihm steht (also im oberen Fall das Wort „KEY“) und hat als Ergebnis die Zeile(n) der Datei oder des Befehls, die das Schlüsselwort enthalten. Der Befehl *grep hallo datei* sucht in der Datei *datei* alle Zeilen, die das Wort *hallo* enthalten.

Schauen Sie sich danach die Datei *lirc_strings.txt* an:

Code-Ausschnitt 5.3.9

```
1 more lirc_strings.txt
```

Sie enthält alle Schlüsselwörter (und noch mehr), die wir gleich beim Anlernen unserer Fernbedienung verwenden werden. Starten wir also das Anlernen der Fernbedienung mit

Code-Ausschnitt 5.3.10

```
1 sudo irrecord -d /dev/lirc0 ~/lircd.conf
```

Das Programm fordert Sie zunächst auf, *RETURN* zu drücken und nach einer nochmaligen Bestätigung mit *RETURN* möglichst viele verschiedene Knöpfe auf der Fernbedienung zu drücken. Jeder Druck auf der Fernbedienung sollte dabei einen Punkt (.) in das Terminal zaubern. Nachdem das Programm (hoffentlich) den prinzipiellen Code der Fernbedienung erkannt hat, geht es an das Anlernen der Tasten. Sie werden dann aufgefordert, zunächst einen Namen für die Taste einzugeben und dann die entsprechende Taste der Fernbedienung zu drücken. Verwenden Sie bitte dabei die Namen der Datei *lirc_strings.txt*, die Sie eben gespeichert haben. Nachdem Sie alle Tasten auf diese Weise angelernt haben, beenden Sie bitte das Anlernen durch nochmaliges Drücken der *RETURN*-Taste auf Ihrer Tastatur. Sie werden dann aufgefordert, eine einzelne Taste der Fernbedienung so schnell hintereinander zu drücken wie möglich. Dadurch lernt *lirc* die Wiederholrate der Fernbedienung kennen. Nach diesem letzten Schritt ist das Anlernen der Fernbedienung abgeschlossen und eine Konfigurations-Datei namens *lircd.conf* wurde in Ihrem Home-Verzeichnis erstellt. Bevor wir die Datei dauerhaft einbinden, testen wir das Ergebnis kurz. Starten Sie bitte den *lirc*-Daemon und rufen Sie das Programm *irw* auf:

Code-Ausschnitt 5.3.11

```
1 sudo lircd -d /dev/lirc0 ~/lircd.conf
2 irw
```

Die Software *irw* (InfRared Write) reagiert auf Fernbedienungstastendrucke und schreibt den gefundenen KEY ins Terminal. Haben Sie also beispielsweise für die rote Taste der Fernbedienung den Schlüssel *KEY_RED* vergeben, erscheint diese Ausgabe, wenn Sie nun die rote Taste Ihrer Fernbedienung drücken. Wenn das alles wie gewünscht funktioniert, kopieren Sie die Datei *lircd.conf* bitte nach */etc/lircd/lircd.conf.d*, und beenden Sie den *lirc*-Daemon.

Code-Ausschnitt 5.3.12

```
1 sudo cp ~/lircd.conf /etc/lircd/lircd.conf.d
2 sudo pkill lircd
```

Bevor der *lirc*-Dienst gestartet wird, ändern Sie bitte in der Datei */etc/lirc/lirc_options.conf* die Zeilen *driver* und *device* so:

Code-Ausschnitt 5.3.13

```
1 driver=default
2 device=/dev/lirc0
```

Nennen Sie bitte noch die Datei um, die *devinput* als Treiber verwendet:

Code-Ausschnitt 5.3.14

```
1 sudo mv /etc/lirc/lircd.conf.d/devinput.lircd.conf /etc/lirc/lircd.conf.d/devinput.lircd.<-
   dist
```

Starten Sie dann den *lirc*-Dienst:

Code-Ausschnitt 5.3.15

```
1 sudo /etc/init.d/lirc start
```

Sollten Sie *lirc* ausschließlich für ein Programm benutzen, das nur auf Tastatureingaben reagiert und das Sie dennoch per Fernbedienung steuern möchten, ändern Sie bitte noch die folgende Zeile in der Datei */etc/lirc/lirc_options.conf*

Code-Ausschnitt 5.3.16

```
1 [lircmd]
2 uinput = True
```

Hierdurch wird beim Drücken der Fernbedienung gleichzeitig ein Tastatur-Event ausgelöst.

MSL Digital Solutions bietet für 20 € eine kleine Aufsteckplatine für den Raspberry Pi an, die neben einem Infrarot-Empfänger gleichzeitig auch einen beleuchteten Einschalt-Knopf bietet. Die Platine erlaubt es sogar, den Raspberry Pi mit Hilfe der Fernbedienung ein- oder auszuschalten. Ist der Raspberry Pi ausgeschaltet, leuchtet der Knopf rot, ist er eingeschaltet, leuchtet er grün. Die Platine ist mit einer eigenen Firmware ausgestattet, die es erlaubt, eine Fernbedienung zum Ein- oder Ausschalten anzulernen. Ein passendes Gehäuse liefert MSL Digital für 8 € gleich mit (Abb. 5.9). Das Gehäuse ist als durchsichtiges Gehäuse oder in schwarz erhältlich. Der Hersteller gibt die Reichweite des Infrarot-Empfängers mit 45 m an. Das MSL Digital Board nutzt übrigens ebenfalls den GPIO Pin 18, um Infrarotsignale zu empfangen.

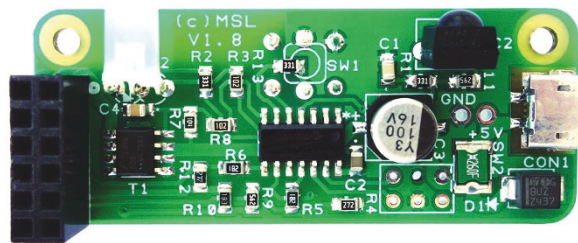
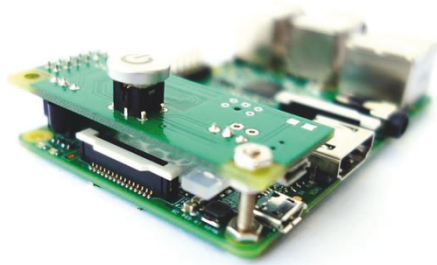
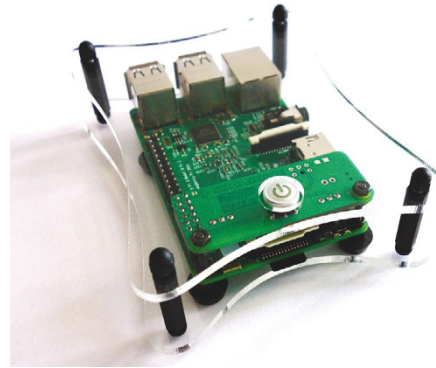


Abbildung 5.8: MSL Digital bietet eine Zusatzplatine mit *lirc*-Empfänger und Einschalt-Knopf (Quelle: msldigital.com).

Ein passendes Gehäuse liefert MSL Digital für 8 € gleich mit (Abb. 5.9). Das Gehäuse ist als durchsichtiges Gehäuse oder in schwarz erhältlich. Der Hersteller gibt die Reichweite des Infrarot-Empfängers mit 45 m an. Das MSL Digital Board nutzt übrigens ebenfalls den GPIO Pin 18, um Infrarotsignale zu empfangen.



(a) MSL Digital Board mit Raspberry Pi
(Quelle: msldigital.com).



(b) MSL Digital Board und Pi im Gehäuse
(Quelle: msldigital.com).

Abbildung 5.9: Pi mit Infrarot-Modul und Einschaltknopf mit oder ohne Gehäuse. Dank MSL Digital kommt man dabei ganz ohne Lötarbeiten aus.



Bitte denken Sie daran, Ihren Raspberry Pi zuerst von der Spannungsversorgung zu trennen, bevor Sie das MSL Digital Board aufstecken, um Kurzschlüsse zu vermeiden. Das Board belegt zwei weitere GPIO Pins, die dazu genutzt werden können, den Raspberry Pi geordnet herunterzufahren, falls das Ausschaltsignal einer Fernbedienung empfangen wurde. Ein- und Ausschaltkommando können angelernt werden, indem der Einschalter 5 Sekunden lang gedrückt wird. Danach erwartet das MSL Digital Board zunächst den Infrarot-Einschaltcode und blinkt nach Erhalt einmal grün. Abschließend wird der Ausschaltcode erwartet, der identisch zum Einschaltcode sein kann. Wenn Sie als Ausschaltcode eine Fernbedienung anlernen, die Sie sonst nie benutzen, können Sie den geordneten Ausschaltvorgang komplett in Software realisieren.

Herzlichen Glückwunsch! Sie haben erfolgreich einen Fernbedienungs-Empfänger an den Raspberry Pi angeschlossen und eine Fernbedienung angelernt. Kümmern wir uns endlich um das Fernsehen mit *VDR*.

5.4 VDR

Der Video Disk Recorder *VDR* von Klaus Schmidinger ist *die* Multimedia-Anwendung auf LINUX, mit der kaum eine andere Multimedia-Software zum Fernsehen mithalten kann. Das kostenlose Programm stellt alles zur Verfügung, was einen guten Full-HD-Receiver ausmacht. Darüber hinaus kann es beliebig individualisiert werden, und eine Fülle an Zusatzprogrammen - sogenannte Plugins - erfüllen auch die letzten Wünsche. In diesem Abschnitt werden wir den *VDR* in der neuesten Version aus dem Netz laden, kompilieren und installieren.

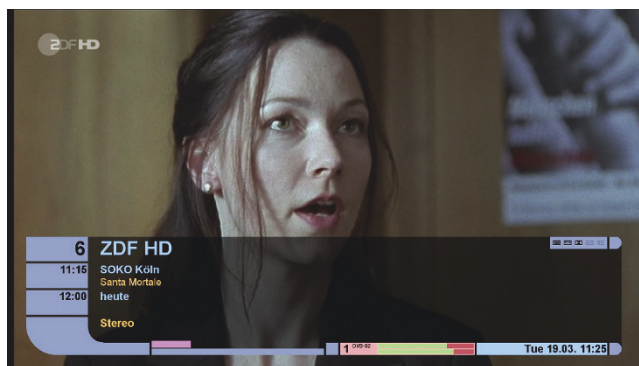


Abbildung 5.10: VDR ist die LINUX-TV-Applikation schlechthin (Quelle: tvdr.de).

Natürlich könnten wir auch ein `sudo apt-get install vdr` benutzen und uns auf die Repository-Version beschränken. Diese hängt aber meist ein paar Versionen hinterher und bringt auch nicht alle Plugins mit, die wir selbst auf dem Pi benötigen.



Den *VDR* mit allen Plugins und einem Konfigurations-Verzeichnis finden Sie unter <https://www.springer.com/de/book/978-3-662-58143-8> (*vdr_allen.tar.bz2*).

Im ersten Schritt öffnen wir ein Terminal und erstellen ein Unterverzeichnis *tv*. Wir wechseln in dieses Verzeichnis und laden die aktuellen Quellen des *VDR* herunter und packen sie aus.

Code-Ausschnitt 5.4.1

```
1 mkdir tv
2 cd tv
3 wget ftp://ftp.tvdr.de/vdr/vdr-2.4.0.tar.bz2
4 tar xvfj vdr-2.4.0.tar.bz2
5 ln -s vdr-2.4.0 VDR
```

Diese liegen auf einem *ftp*-Server (*File Transfer Protocol*). Der *tar*-Befehl mit seinen Parametern packt den *VDR*-Quellcode in einem Verzeichnis namens *vdr-2.4.0* aus (*bzip2*). Wir erstellen nun einen symbolischen Link namens *VDR*, der auf das gerade heruntergeladene und ausgepackte Verzeichnis *vdr-2.4.0* zeigt.



Der symbolische Link *ln -s* zeigt auf das Verzeichnis, das er linkt. Man kann dann in das Verzeichnis wechseln, indem man nur den Link-Namen angibt. Im oberen Beispiel reicht dann ein `cd VDR`, um in das Verzeichnis *vdr-2.4.0* zu gelangen. Sollte es eine neuere Version des *VDR* geben, kann man diese parallel installieren und ändert nur den symbolischen Link. Schon funktionieren alle Start-Skripte wieder, die wir später noch erstellen werden. Andernfalls müssten wir in allen Skripten immer den kompletten Pfad zum *VDR* mit angeben (inkl. der Versionsnummer (-2.4.0)).

Symbolische Links sehen im Listing wie folgt aus:

Code-Ausschnitt 5.4.2

```
1 ls -l
2 lrwxrwxrwx 1 pi pi 9 Mai 4 2013 VDR -> vdr-2.4.0
3 drwxr-xr-x 8 pi pi 4096 Mai 4 16:39 vdr-2.4.0
```

Der Link wird dabei durch einen Pfeil (->) symbolisiert. Links können mit `rm linkname` wieder gelöscht werden, ohne dass das Originalverzeichnis gelöscht wird. Im nächsten Schritt müssen noch ein paar Programme installiert werden, welche zum Übersetzen des *VDR* benötigt werden.

Code-Ausschnitt 5.4.3

```
1 sudo apt-get install mercurial libncurses5-dev libncursesw5-dev libproc-processtable-↔
perl libfribidi-dev libcap-dev libjpeg-dev lcdproc libssl-dev libfontconfig1-dev g++ git vim gettext
```

Wechseln wir nun ins *VDR*-Verzeichnis und starten die Übersetzung des *VDR* und der Standard-Plugins.

Code-Ausschnitt 5.4.4

```
1 cd VDR
2 make -j4
```

Das Übersetzen dauert nur ein paar Minuten, wenn wir den Parameter `-j4` verwenden. Dieser startet nämlich einen Übersetzungsvorgang für alle 4 Kerne der Raspberry Pi CPU. Das fertige Programm `vdr` liegt dann im aktuellen Verzeichnis. Bevor wir es aber starten, müssen wir noch einige Vorbereitungen treffen. `VDR` selbst benötigt nämlich noch die folgenden Verzeichnisse:

- Ein Konfigurations-Verzeichnis, in dem alle Konfigurations-Dateien stehen. Das Default-Verzeichnis lautet `/var/lib/vdr`. Dieses werden wir auch nutzen.
- Ein Video-Verzeichnis, in das der `VDR` Aufnahmen speichern kann. Für einen ersten Test erstellen wir das Verzeichnis `/video` gleich auf der SD-Karte. Es wird auch die Möglichkeit beschrieben, eine externe USB-Festplatte anzuschließen und die Aufnahmen dort zu speichern. Das Standard-Verzeichnis ist `/var/lib/video`

Öffnen wir also ein Terminal und erstellen das Video-Verzeichnis:

Code-Ausschnitt 5.4.5

```
1 sudo mkdir /video
2 sudo chown pi:pi /video
3 sudo chmod 775 /video
```

Gleichzeitig sagen wir, dass der Besitzer des Verzeichnisses `/video` der Benutzer `pi` mit der Gruppe `pi` ist und erlauben, dass der Benutzer Filme schreiben und lesen darf (`chmod 775`). Falls der Pi genügend abgesichert ist, spricht zwar nichts dagegen, den `VDR` als Benutzer `root` laufen zu lassen, das muss aber nicht sein. Es macht dann Sinn, wenn wir - wie später - Skripte benötigen, die Root-Rechte brauchen. Aber auch dann könnte man diesen Skripten Root-Rechte einräumen, so dass nicht der ganze `VDR` als Systemadministrator laufen muss.



In der Datei `/etc/sudoers` stehen Benutzer und Programme, die als Root laufen dürfen, ohne dass z. B. ein Kennwort eingegeben werden muss. Die Datei kann man mit dem Befehl `sudo visudo` ergänzen.

Kopieren wir nun alle Standard-Konfigurations-Dateien des `VDR` in das Konfigurations-Verzeichnis.

Code-Ausschnitt 5.4.6

```
1 cp /home/pi/tv/vdr/*.conf /var/lib/vdr
```

Die Konfigurations-Dateien haben die folgenden Bedeutungen:

- `channels.conf`: Das ist die Senderliste des `VDR`. Unterschiedliche Empfangsarten, wie DVB-S und DVB-T dürfen dabei gemischt werden.
- `remote.conf`: Hier stehen alle Möglichkeiten zur Fernbedienung des `VDR`. Das betrifft - Sie ahnen es - `lirc`, aber auch die Tastatur.
- `diseqc.conf`: Hier finden sich alle Diseqc-Einstellungen.
- `sources.conf`: Hier sind die Satelliten, die empfangbar sind, aufgeführt.
- `commands.conf`: Hier stehen Befehle, die der `VDR` auf Fernbedienungsdruck ausführen kann.
- `scr.conf`: Hierin steht das Satelliten-Kanalrouting.
- `keymacros.conf`: Hier können Makros gespeichert werden, die dann Fernbedienungstasten zugeordnet werden können.

Bevor wir das erste Fernsehbild auf unseren Monitor zaubern, müssen noch zwei Dinge erledigt werden. Zunächst einmal füllen wir die wichtigsten Konfigurations-Dateien mit Leben, danach installieren wir ein Ausgabe-Plugin für den Raspberry Pi, damit wir das Bild überhaupt (beschleunigt) sehen können. Fangen wir mit der `channels.conf` an:

Code-Ausschnitt 5.4.7

```

1  :-> Satelliten-Test
2  Das Erste;ARD:11837:HC34M2S0:S19.2E:27500:101=2:102=deu@3,103=mis@3;106=deu@106:104;105=↔
    deu:0:28106:1:1101:0
3  Das ErsteHD;ARD:11493:HC23M5035S1:S19.2E:22000:5101=27:5102=deu@3,5103=mis@3;5106=↔
    deu@106:5104;5105=deu:0:10301:1:1019:0
4  :-> DVB-T2-Test
5  KiKA HD;ZDFmobil:578000:B8DOG19128S1T16Y0P0:T↔
    :27500:2510=36:0;2520,2521:2530;2531:0:2005:8468:515:0

```

Das Beispiel enthält zwei Sender für den Satelliten-Empfang auf Astra 19.2° Ost und ein Beispiel für DVB-T2. Der genaue Aufbau der Datei ist unter <http://www.vdr-wiki.de/wiki/index.php/Channels.conf> beschrieben. Dort gibt es auch Beispiele für die unterschiedlichen Empfangsarten.



Voraussetzung für die Wiedergabe von SD-Fernsehmaterial ist der MPEG-2 Codec (Kapitel 1.4). HD-TV kann ohne Codec vom Pi dargestellt werden.

Der Inhalt meiner *remote.conf* sieht wie folgt aus, damit ich den VDR schon einmal mit der Tastatur bedienen kann:

Code-Ausschnitt 5.4.8

```

1  KBD.Up          00000000010C0039
2  KBD.Down       0000000001B5B42
3  KBD.Menu       00000000006D0039
4  KBD.Ok         0000000000D0039
5  KBD.Back       00000000007F0039
6  KBD.Left       00000000010E0039
7  KBD.Right      00000000010F0039
8  KBD.Red        0000000000720039
9  KBD.Green      0000000000670039
10 KBD.Yellow     0000000000000079
11 KBD.Blue       0000000000620039
12 KBD.0          0000000000300039
13 KBD.1          0000000000310039
14 KBD.2          0000000000320039
15 KBD.3          0000000000330039
16 KBD.4          0000000000340039
17 KBD.5          0000000000350039
18 KBD.6          0000000000360039
19 KBD.7          0000000000370039
20 KBD.8          0000000000380039
21 KBD.9          0000000000390039
22 KBD.Info       0000000000690039

```

Support für die Fernbedienung fügen wir gleich hinzu. Da ich Hotbird und Astra empfangen kann, sieht meine *disqc.conf* Datei so aus:

Code-Ausschnitt 5.4.9

```

1  # Full DiSEqC sequence:
2
3  S19.2E 11700 V 9750 t v W15 [EO 10 38 F0] W15 A W15 t
4  S19.2E 99999 V 10600 t v W15 [EO 10 38 F1] W15 A W15 T
5  S19.2E 11700 H 9750 t V W15 [EO 10 38 F2] W15 A W15 t
6  S19.2E 99999 H 10600 t V W15 [EO 10 38 F3] W15 A W15 T
7
8  S13.0E 11700 V 9750 t v W15 [EO 10 38 F4] W15 B W15 t
9  S13.0E 99999 V 10600 t v W15 [EO 10 38 F5] W15 B W15 T
10 S13.0E 11700 H 9750 t V W15 [EO 10 38 F6] W15 B W15 t
11 S13.0E 99999 H 10600 t V W15 [EO 10 38 F7] W15 B W15 T

```

Ich habe also die entsprechenden Kommentarzeichen (#) vor den verfügbaren Satelliten entfernt. In der *commands.conf* können Befehle eingetragen werden, die dann von *VDR* aus ausgeführt werden können, z. B.

Code-Ausschnitt 5.4.10

```
1 Reboot : sudo /sbin/reboot
```

Der Name des Kommandos steht mit einem : getrennt vom auszuführenden Befehl. Alle anderen Dateien lassen wir auf den Default-Werten stehen. Nach diesem trockenen Teil kommen wir nun zum ersten sichtbaren Ergebnis und installieren eine Ausgabemöglichkeit für den Raspberry Pi.

Das Raspberry Pi HD-Device-Plugin, welches von Thomas Reufer entwickelt wird, ist auf <http://projects.vdr-developer.org/projects/plg-rpihddevice/repository> zu Hause. Klicken Sie auf *Dateien* und laden Sie die neueste Version des Plugins herunter. Zum Zeitpunkt, als ich dieses Buch geschrieben habe, war das die Version *vdr-rpihddevice-1.0.3.tgz*. Kopieren Sie die Version bitte nach dem Herunterladen in das Source-Plugin-Verzeichnis Ihres *VDR* */home/pi/tv/vdr/PLUGINS/src*. Packen Sie das Plugin aus und setzen Sie einen symbolischen Link.

Code-Ausschnitt 5.4.11

```
1 cd /home/pi/tv/VDR/PLUGINS/src
2 tar xvfz vdr-rpihddevice-1.0.3.tgz
3 ln -s vdr-rpihddevice-1.0.3 rpihddevice
4 sudo apt-get install ffmpeg libavcodec-dev libavformat-dev
```

Der Link ist erforderlich, damit *VDR* das Plugin überhaupt übersetzt. Die Bibliotheken *ffmpeg*, *avcodec* und *avformat* sind für den Betrieb des Ausgabepugins dringend erforderlich. Sie enthalten Video- und Audio-Routinen. Alternativ können Sie die aktuelle Version des Plugins mit Hilfe von *git* auschecken:

Code-Ausschnitt 5.4.12

```
1 git clone git://projects.vdr-developer.org/vdr-plugin-rpihddevice.git rpihddevice
```

Bei der neuesten Raspbian Distribution (Stretch) musste ich zwei Bibliotheken im *Makefile* des *rpihddevice*-Plugins ändern:

Code-Ausschnitt 5.4.13

```
1 LDLIBS += -lbcm_host -lvcos -lvchiq_arm -lopenmaxil -lbrcmGLESv2 -lbrcmEGL -lpt
```

Die Bibliotheken *GLESv2* und *EGL* befinden sich im Tool *libraspberrypi0*, welches standardmäßig bereits installiert ist. Allerdings haben beide Bibliotheken den Zusatz *brcm* für *Broadcom* erhalten, was im o.g. *Makefile* noch nicht berücksichtigt ist. Nach dieser Korrektur wechseln wir in das *VDR*-Verzeichnis und rufen das Erstellen von *VDR* und der dazugehörigen Plugins erneut auf.

Code-Ausschnitt 5.4.14

```
1 cd /home/pi/tv/VDR
2 make
```

Das *rpihddevice*-Plugin, eine sogenannte *shared library*, also eine Bibliothek, die bei Bedarf dazu geladen wird, befindet sich nun im Verzeichnis */home/pi/tv/vdr/PLUGINS/src/rpihddevice* und heißt *libvdr-rpihddevice.so*. Ein Aufruf von

Code-Ausschnitt 5.4.15

```
1 sudo make install
```

installiert die ausführbare Datei für den *VDR* im Verzeichnis */usr/local/bin* und alle Plugin-Bibliotheken im Verzeichnis */usr/local/lib/vdr*. Dabei erhalten die Bibliotheken automatisch den Zusatz der *VDR*-Versionsnummer, in unserem Fall also *.2.4.0*. Bei manchen Plugins kann es erforderlich sein, die Datei händisch in das Verzeichnis für die *VDR*-Bibliotheken zu kopieren. In diesem Fall sähe der Aufruf zum Kopieren so aus:

Code-Ausschnitt 5.4.16

```
1 cp /home/pi/tv/VDR/PLUGINS/src/rpihddevice/libvdr-rpihddevice.so /usr/local/lib/vdr/↵
   libvdr-rpihddevice.so.2.4.0
```

Achten Sie bitte darauf, dass die Versionsnummer (*2.4.0*) identisch ist mit der des *VDR*. Sie erkennen das an den *VDR*-Plugins, die sich ebenfalls im o. g. Verzeichnis befinden. Bereit für einen ersten Start? Also los:

Code-Ausschnitt 5.4.17

```
1 export VDR_LANG=de_DE@euro
2 export VDR_CHARSET_OVERRIDE=ISO-8859-9
3 ./vdr --lirc -c /var/lib/vdr -v /video -L /usr/local/lib/vdr -Prpihddevice
```

Was geschieht hier? Zunächst werden zwei *VDR*-Variablen exportiert (also sichtbar gemacht und gesetzt), welche die Sprache für den *VDR* so einstellen, dass deutsche Umlaute korrekt dargestellt werden. Danach wird der *VDR* selbst mit einigen Parametern aufgerufen. Der Parameter *-c* übergibt das Konfigurations-Verzeichnis, welches wir zu Beginn erstellt haben. Der Parameter *-v* erledigt dasselbe mit dem Video-Verzeichnis. Der Parameter *-L* übergibt das Plugin-Verzeichnis. Hier nehmen wir einfach das Standard-Verzeichnis des *VDR*. Der Parameter *-P* übergibt ein Plugin. Wir nutzen im Moment nur ein einziges Plugin, nämlich das *rpihddevice*, über das wir das Bild auf unseren Monitor zaubern.



Durch den Aufruf von

Code-Ausschnitt 5.4.18

```
1 make REMOTE=LIRC
```

kann man den *VDR* auch mit *lirc*-Unterstützung übersetzen. Sollten Sie das beim Bauen auf dem Quelltext vergessen haben, können Sie den *lirc*-Support mit dem Parameter *--lirc* im *VDR*-Aufruf nachträglich einschalten.

Nach dem Start werden Sie zunächst aufgefordert, eine Taste auf Ihrer Fernbedienung zu drücken. *VDR* startet dann mit dem Zuweisen aller Fernbedienungstasten. Danach ist *VDR* nicht nur mit der Tastatur, sondern auch mit der Fernbedienung bedienbar.

Info Das *rpidhdevice*-Plugin benötigt keine grafische X-Oberfläche, damit es funktioniert. Im Gegenteil: Starten Sie den VDR mit diesem Plugin bitte ohne grafische Oberfläche. Falls diese läuft, beenden Sie sie bitte zuvor. Das spart Ressourcen auf dem kleinen Embedded Board, die dringend zur Grafikausgabe des Fernsehbildes benötigt werden. Viele Benutzer berichten davon, dass das *rpidhdevice* besser läuft, wenn vor dem *VDR*-Aufruf die HDMI-Bildausgabe auf *1080p50* gesetzt wird (full-HD, progressive, 50 Hz):

Code-Ausschnitt 5.4.19

```
1 tvservice -e "CEA 31"
```

Info Beim Beenden des *VDR* werden Sie sicher feststellen, dass der Cursor auf Ihrer Konsole verschwunden ist. Dieses Problem werden wir durch ein Start-Skript lösen.

Um den *VDR* so zu starten, dass die Konsole nach seinem Beenden wieder mit Cursor sichtbar ist, legen Sie bitte als *root* eine Datei mit dem Namen *runvdr* im Verzeichnis */usr/local/bin/* mit dem folgenden Inhalt an (*sudo vi /usr/local/bin/runvdr*):

Code-Ausschnitt 5.4.20

```
1 #!/bin/bash
2 export VDR_LANG=de_DE@euro
3 export VDR_CHARSET_OVERRIDE=ISO-8859-9
4 /usr/local/bin/vdr -c /home/pi/tv/vdrconfig -v /video -L /home/pi/tv/VDR/PLUGINS/lib -↔
   Prpidhdevice
5 # Ausgabe wieder einschalten
6 stty echo
```

Die letzte Zeile sorgt dafür, dass die Konsole nach dem Beenden des *VDR* wiederhergestellt wird. Machen Sie das Skript ausführbar und starten Sie es probierhalber.

Code-Ausschnitt 5.4.21

```
1 chmod 775 /usr/local/bin/runvdr
2 /usr/local/bin/runvdr
```

Wenn Sie die Fernbedienung nicht anlernen wollen, sondern die Tasten direkt zuweisen möchten, können Sie die *lirc*-Einträge auch direkt in der *VDR*-Konfigurationsdatei *remote.conf* vornehmen. Die Einträge sehen dann wie folgt aus:

Code-Ausschnitt 5.4.22

```
1 ...
2 LIRC.Left      KEY_LEFT
3 LIRC.Right     KEY_RIGHT
4 KBD.Up        00000000010C0039
5 KBD.Down      0000000001B5B42
6 ...
```

lirc- und Tastatur-Einträge (KDB für *keyboard*) können einfach miteinander kombiniert werden. Es würde den Rahmen dieses Buches sprengen, auf alle Einstellmöglichkeiten des *VDR* einzugehen. Probieren Sie doch einfach die Einstellungen aus, wie beispielsweise das Seitenverhältnis Ihres Bildschirms oder analoge/digitale Audioausgabe. Im weiteren Verlauf dieses Kapitels werden wir noch das eine oder andere Plugin installieren. Der Link <http://www.vdr-wiki.de/wiki/index.php/Plugins> gibt eine schöne Übersicht über verfügbare *VDR*-Plugins. Zunächst schauen wir uns aber noch kurz an, wie man eine externe Festplatte zum Aufnehmen von Fernsehsendungen verwendet.

5.5 Externe Festplatte



Abbildung 5.11: Eine per USB eingebaute 2,5 Zoll-Festplatte erweitert den Speicherplatz des Raspberry Pi. Aber Vorsicht: Für die externe Festplatte kann eine externe Spannungsversorgung erforderlich sein.

Im Dauer-Fernsehbetrieb wird der freie Speicherplatz auf der SD-Karte des Pi schnell zu klein. Ein normaler HD-Film belegt gerne schon einmal 4...5 GB Speicherplatz. Es macht also Sinn, eine externe USB-Festplatte anzuschließen. Bitte denken Sie daran, dass Festplatten einen hohen Anlaufstrom haben und auch im Dauerbetrieb mehr als die 100 mA Strom brauchen, die der kleinste Raspberry Pi an seinen USB-Anschlüssen zur Verfügung stellen kann. Sie müssen hier also auch entweder mit einem aktiven Hub oder mit einem zusätzlichen USB-Netzteil und einem Y-Kabel arbeiten. Beim Raspberry Pi 3B+ reicht es für den Betrieb einer externen Festplatte evtl. aus, den Parameter

Code-Ausschnitt 5.5.1

```
1 max_usb_current=1
```

in der Datei `/boot/config.txt` zu setzen, damit insgesamt 1,2 A Strom für die USB-Ports zur Verfügung stehen. Wenn Sie die externe Festplatte anstecken, erscheint diese als *sda*-Device. Zahlen hinter dem letzten Buchstaben kennzeichnen nicht fortlaufende Festplatten (wie z. B. beim *lirc*-Device), sondern verschiedene Partitionen (sofern vorhanden) auf der *a*-Platte. Weitere Platten erhalten dann den Buchstaben *b* statt *a*.

Das zweite Device würde also den Namen `/dev/sdb` tragen. Da wir die externe Festplatte in das `/video`-Verzeichnis einhängen (*mounten*) wollen, sollte das `/video`-Verzeichnis leer sein. Falls sich in Ihrem Video-Verzeichnis noch Dateien befinden, werden diese beim Einhängen der externen Festplatte verdeckt und erst wieder sichtbar, nachdem die externe Platte ausgehängt worden ist. Sie können den Inhalt des gesamten `/video`-Verzeichnisses so löschen:

Code-Ausschnitt 5.5.2

```
1 cd /video
2 rm -rf *
```

Info Denken Sie bitte daran: Der Befehl `rm -rf *` löscht den kompletten Inhalt eines Verzeichnisses. Seien Sie also sicher, dass Sie sich im richtigen Verzeichnis befinden, bevor Sie den Befehl aufrufen. Falls Sie sich im Verzeichnis `/` befinden, können Sie dabei zusehen, wie Ihr Betriebssystem sich in Luft auflöst. Nachdem das `/video`-Verzeichnis leer ist, binden wir die externe Festplatte darauf ein, nachdem wir sie formatiert haben. Sie können sich das Formatieren natürlich sparen, wenn die Platte bereits ein Dateisystem enthält.

Code-Ausschnitt 5.5.3

```
1 sudo fdisk /dev/sda
2 mkfs -t ext4 /dev/sda1
```

Der `fdisk`-Befehl erlaubt es Ihnen, Partitionen auf Ihrer Festplatte zu erstellen.

Der *mkfs*-Befehl (*make filesystem*) erzeugt das Dateisystem, im oberen Fall vom Typ *ext4*, auf der ersten Partition */dev/sda1*.



Sie können auch NTFS-Partitionen unter LINUX mounten, lesen und schreiben. Dafür gibt es das Programm *ntfs-3g*. Wenn Windows NTFS-Partitionen eingebunden werden sollen, kann das mit dem Aufruf

Code-Ausschnitt 5.5.4

```
1 sudo ntfs3g /dev/sda1 /mnt
```

erledigt werden. Das Beispiel bindet die erste Partition einer NTFS-Platte in das Verzeichnis */mnt* ein. Für den Betrieb mit VDR kann ich NTFS als Dateisystem nicht empfehlen, da es spürbar langsamer ist als ein *ext*-Dateisystem.

Nach diesen vorbereitenden Schritten kann die externe Festplatte im Verzeichnis */video* eingehängt werden. Der Befehl

Code-Ausschnitt 5.5.5

```
1 sudo mount /dev/sda1 /video
```

erledigt das für uns. Ab sofort werden Ihre *VDR*-Filme auf der externen Platte gespeichert und nicht mehr auf der SD-Karte. Nach einem Reboot des Systems ist der Mount-Punkt aber leider wieder verschwunden. Um ihn dauerhaft und reboot-sicher einzubinden, fügen Sie bitte in der Datei */etc/fstab* die folgende Zeile hinzu:

Code-Ausschnitt 5.5.6

```
1 /dev/sda1 /video ext4 rw,auto,users 0 0
```

Das Dateisystem *ext4* müssen Sie natürlich mit dem Dateisystem der externen Platte ersetzen.

5.6 VDRAdmin-am

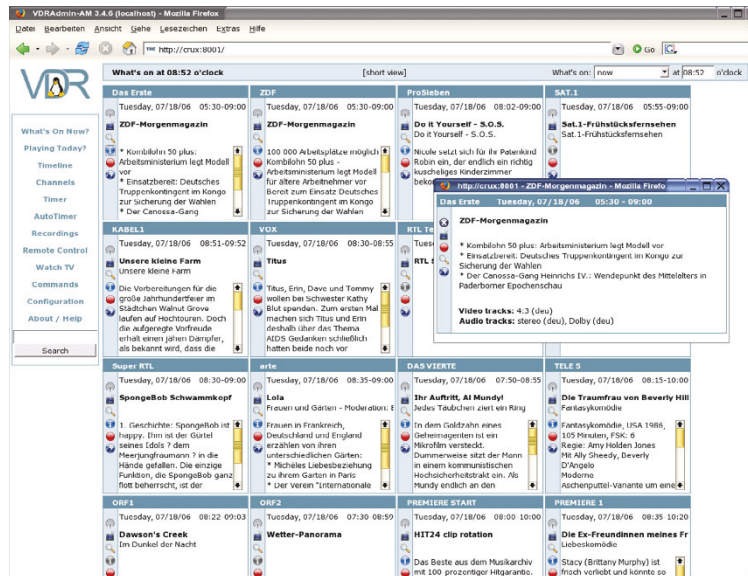


Abbildung 5.12: VDRAdmin-am erlaubt das Programmieren von Aufzeichnungen vom Internet aus (Quelle: vdr-wiki.de).

Mit VDRAdmin-am gibt es eine Software, die auf einem Webserver läuft, der mit dem VDR kommuniziert. Auf diese Art und Weise haben Sie die Möglichkeit, Timer zur Aufzeichnung von Fernsehsendungen mit Ihrem Handy zu setzen, das aktuelle Fernsehprogramm im EPG (*Electronic Program Guide*) zu sehen, Aufnahmen zu löschen und vieles mehr. VDRAdmin-am läuft zwar nicht besonders schnell auf dem Pi, mit ein wenig Geduld ist es aber dennoch vernünftig bedienbar. VDRAdmin-am setzt das *perl*-Modul voraus, das wir zusammen mit einigen weiteren Paketen im nächsten Schritt installieren.

Code-Ausschnitt 5.6.1

```
1 sudo apt-get install libauthen-sasl-perl libdigest-hmac-perl libio-socket-inet6-perl
```

Den Quelltext von VDRAdmin-am finden Sie auf <http://projects.vdr-developer.org/git/vdradmin-am.git/>. Clonen Sie die aktuelle Version mit

Code-Ausschnitt 5.6.2

```
1 git clone git://projects.vdr-developer.org/vdradmin-am.git
```

in ein Verzeichnis Ihrer Wahl. Wechseln Sie danach in das Verzeichnis *vdradmin-am.git* und führen sie die folgenden Befehle aus:

Code-Ausschnitt 5.6.3

```
1 cd vdradmin-am.git
2 sudo LANG=de_DE ./make.sh install
3 ./make.sh cvs
4 ./make.sh po
```

Die letzten beiden Zeilen erstellen einige Links und die Internationalisierung (Sprachen). Nach dem Installieren und Anlegen der Links und der Internationalisierung rufen Sie bitte den *vdradmin*-Daemon einmalig mit dem Parameter *--config* auf:

Code-Ausschnitt 5.6.4

```
1 vdradmin.pl --config
```

Nach dem Konfigurieren (Sie müssen einige Fragen beantworten) können Sie den Daemon mit

Code-Ausschnitt 5.6.5

```
1 vdradmin.pl
```

starten. Von jetzt an können Sie sich mit einem Browser unter dem Port 8001 auf dem Pi einwählen, um die Oberfläche von VDRAdmin-am im Browser zu sehen. Bitte denken Sie daran, dass der VDR laufen muss, damit VDRAdmin-am funktioniert. Der Aufruf im Browser lautet:

Code-Ausschnitt 5.6.6

```
1 http://ip-adresse:8001
```

Bitte ersetzen Sie *ip-adresse* mit der IP-Adresse Ihres Raspberry Pi.

5.7 Schöner fernsehen

Der VDR stellt zwar ein paar Standard-Skins für das OSD bereit, aber keines davon sieht besonders hübsch aus. Ein Skin, das mir besonders gut gefällt, heißt Metrix HD. Da das Aussehen eines OSD und damit auch die Skins reine Geschmacksache sind, erkläre ich Ihnen in diesem Abschnitt, wie Sie zunächst das *skindesigner*-Plugin installieren können und anschließend ein Skin Ihrer Wahl (am Beispiel von Metrix HD) installieren können. Mit dem *skindesigner* hat Louis Braun ein Plugin geschaffen, das XML-basierte (Extensible Markup Language) Skins laden und darstellen kann. Das Plugin selbst ist auf <https://projects.vdr-developer.org/projects/plg-skindesigner/wiki> zu Hause. Hier gibt es auch eine Beschreibung, wie die Skins aufgebaut sein müssen, damit *skindesigner* mit ihnen umgehen kann. Da es derzeit keine Webseite gibt, welche alle verfügbaren Skins für den VDR zusammenfasst, empfehle ich eine Google-Suche mit den Begriffen *vdr*, *skindesigner* und *skin*.

Um den *skindesigner* zu installieren, klonen wir die neueste Version auf dem Git-Repository:

Code-Ausschnitt 5.7.1

```
1 cd /home/pi/tv/VDR/PLUGINS/src
2 git clone git://projects.vdr-developer.org/vdr-plugin-skindesigner.git
3 mv vdr-plugin-skindesigner skindesigner
```



Abbildung 5.13: Metrix HD ist ein wunderschönes Skin für den VDR. Dank OpenGL läuft es auch auf einem Raspberry Pi flüssig (Quelle: vdr-portal.de).

Das Übersetzen des *skindesigner*-Plugins setzt einige Bibliotheken voraus, die wir installieren:

Code-Ausschnitt 5.7.2

```
1 sudo apt-get install libcurl4-openssl-dev libxml2-dev librsvg2-dev
```

Das *skindesigner*-Plugin kann nun übersetzt und installiert werden:

Code-Ausschnitt 5.7.3

```
1 cd /home/pi/tv/VDR
2 make
3 sudo make install
```

Ein paar Skins stellt der *skindesigner* selbst zur Verfügung, so auch Metrix HD. Ergänzen wir also zunächst das Startskript für den VDR um das *skindesigner*-Plugin. An dieser Stelle zeige ich Ihnen meine komplette Start-Datei für den VDR, die Sie leicht um weitere Einträge ergänzen können. Sie startet auch gleich den Watchdog mit (*-w 60*), der VDR neu startet, falls er einmal crashen sollte.

Code-Ausschnitt 5.7.4

```

1  #!/bin/sh
2  tvservice -o
3  tvservice -p
4  export LC_MESSAGES=de_DE.UTF-8
5  export LC_LANG=de_DE.UTF-8
6  export LANG=de_DE.UTF-8
7  export LC_ALL=de_DE.UTF-8
8  export VDR_LANG=de_DE@UTF-8
9  export VDR_CHARSET_OVERRIDE=ISO-8859-9
10 VDRPRG="/usr/local/bin/vdr"
11 VDROPTIONS="-l 0 -w 60 --lirc=/var/run/lirc/lircd -v /video -s /usr/local/bin/↵
    vdrshutdown.sh"
12 VDRPLUGINS=" -Prpихddevice \
13 -P'skindesigner -l /var/lib/vdr/plugins/skindesigner/logos -e /video/cache/epgim
14 ages -i /var/lib/vdr/plugins/skindesigner/installerskins'"

16 VDRCMD="$VDRPRG $VDROPTIONS $VDRPLUGINS $*"

18 KILL="/usr/bin/killall -q -TERM"

20 # Detect whether the DVB driver is already loaded
21 # and return 0 if it *is* loaded, 1 if not:
22 DriverLoaded()
23 {
24     return 1
25 }

27 # Load all DVB driver modules needed for your hardware:
28 LoadDriver()
29 {
30     return 0
31 }

33 # Unload all DVB driver modules loaded in LoadDriver():
34 UnloadDriver()
35 {
36     return 0
37 }

39 # Load driver if it hasn't been loaded already:
40 if ! DriverLoaded; then
41     LoadDriver
42     fi

44 while (true) do
45     eval "$VDRCMD"
46     if test $? -eq 0 -o $? -eq 2; then exit; fi
47     echo "'date' reloading DVB driver"
48     $KILL $VDRPRG
49     sleep 10
50     UnloadDriver
51     LoadDriver
52     echo "'date' restarting VDR"
53     done

```

Das Skript `/usr/local/bin/vdrshutdown.sh` sorgt dafür, dass der Raspberry Pi ausgeschaltet wird, wenn keine Aufnahme programmiert ist. Ist noch eine Aufnahme programmiert, wird mit Hilfe des `at`-Daemon ein Timer angelegt, der den `VDR` kurz vor der Aufnahme wieder startet. Danach wird der `VDR` beendet, der Raspberry Pi aber nicht ausgeschaltet. Bei mir sieht es wie folgt aus:

Code-Ausschnitt 5.7.5

```

1  #!/bin/sh
2  # VDR shutdown script
3  TIMER=$((($2-300)/60))
4  for i in $(atq | cut -f 1); do atrm $i; done
5  echo $TIMER /usr/bin/at -f /usr/local/bin/runvdr now + $TIMER min

7  sudo pkill -x runvdr
8  sleep 1
9  sudo pkill -x vdr
10 DISPLAY=:0.0 sudo -u pi xset s activate
11 if test $TIMER -lt 0
12 then
13   echo '/usr/local/bin/shutdown' | at now
14 fi
15 exit 1

```

Falls Sie dieses Skript nutzen wollen, stellen Sie bitte sicher, dass *at* installiert ist:

Code-Ausschnitt 5.7.6

```

1  sudo apt-get install at

```

Wie Sie sicherlich bemerkt haben, ruft *vdrshutdown.sh* wieder ein anderes Skript auf, nämlich */usr/local/bin/shutdown*. Da ich meinen Raspberry Pi mit dem MSL Digital Board betreibe, sorgt das o.g. Skript dafür, dass der Raspberry Pi geordnet herunterfährt:

Code-Ausschnitt 5.7.7

```

1  #!/bin/bash
2  GPIOpin=15
3  echo "$GPIOpin" > /sys/class/gpio/export
4  # execute shutdown sequence on pin
5  echo "out" > /sys/class/gpio/gpio$GPIOpin/direction
6  echo "1" > /sys/class/gpio/gpio$GPIOpin/value
7  #usleep 125000
8  sleep 0.125
9  echo "0" > /sys/class/gpio/gpio$GPIOpin/value
10 #usleep 200000
11 sleep 0.2
12 echo "1" > /sys/class/gpio/gpio$GPIOpin/value
13 #usleep 400000
14 sleep 0.4
15 echo "0" > /sys/class/gpio/gpio$GPIOpin/value
16 sudo /usr/local/bin/send 11111 3 0
17 umount /video
18 sync
19 halt -p

```

Sollten Sie kein MSL Digital Board haben, reichen in dieser Datei die letzten drei Einträge aus.



Der Raspberry Pi besitzt keinen Einschalt-Timer. Wenn er einmal von der Spannungsversorgung getrennt ist, kann er sich im Gegensatz zu PCs nicht wieder zu einer bestimmten Zeit einschalten.

Aber zurück zum *skindesigner*-Plugin. Stellen Sie bitte sicher, dass das *rpihddevice* eine beschleunigte OSD-Ausgabe nutzt. Navigieren Sie bitte dazu im VDR-Menü zu *Einstellungen* → *Plugins* → *rpihddevice* und wählen Sie im Punkt *OSD mit GPU-Unterstützung ja* aus.

Unter *Menü* → *Plugins* → *skindesigner* stehen bereits einige Skins zur Vorschau und Installation bereit. Diese nutzen wir aber nicht. Metrix wird bereits zusammen mit dem *skindesigner* vorinstalliert. Wählen Sie Metrix HD unter *Menü* → *Einstellungen* → *OSD* → *Oberfläche* aus.

Nach dem Schließen des OSD dauert es eine Weile, bis das neue Skin initialisiert ist. Beim Aufruf werden Sie feststellen, dass einige Dinge wie die Temperaturanzeige für den Raspberry Pi oder die Wettervorhersage noch nicht funktionieren. Diese Features installieren wir in Kürze. Falls Ihnen die Anordnung des Skins nicht gefällt, können Sie diese nach Belieben in den Einstellungen des *skindesigner*-Plugins ändern.



Je aufwendiger ein Skin für den VDR/*skindesigner* ist, desto mehr GPU-Speicher wird benötigt. Den GPU-Speicher können Sie in der Datei */boot/config.txt* festlegen. Bei mir stehen für die GPU 144 MB zur Verfügung:

Code-Ausschnitt 5.7.8

```
1 gpu_mem=144
```

Ist der reservierte Speicher zu klein, kann das zu Abstürzen bei aufwendigen Skins führen.

Sorgen wir aber nun dafür, dass das Skin die aktuelle Temperatur des Raspberry Pi anzeigt. Das Plugin *skindesigner* erwartet das Temperaturanzeige-Skript *temperatures* im Verzeichnis */usr/local/lib/vdr/skindesigner/scripts*:

Code-Ausschnitt 5.7.9

```
1 #!/bin/bash
3 OUTPUTFLDR="/tmp/skindesigner/"
4 mkdir -p ${OUTPUTFLDR}
6 # if the script is executed from system_information script set the locale back for "°C"
7 LANG=de_DE.UTF-8
9 # there can be 4 files, cpu, gpu, pccase, motherboard
10 rm -f ${OUTPUTFLDR}/cpu ${OUTPUTFLDR}/pccase ${OUTPUTFLDR}/gpu ${OUTPUTFLDR}/motherboard
12 CPU=$(vencmd measure_temp | tr -d "temp=" | tr -d "°" | tr -d "C")
13 echo "$CPU °C" > ${OUTPUTFLDR}/cpu
15 echo $CPU > ${OUTPUTFLDR}/pccase
16 echo $CPU > ${OUTPUTFLDR}/motherboard
18 GPU=$(vencmd measure_temp | tr -d "temp=" | tr -d "°" | tr -d "C")
19 echo "${GPU} °C" > ${OUTPUTFLDR}/gpu
```

Dasselbe gilt für die Datei *vdrstats*, die Auskunft über CPU-Verbrauch und Speicher gibt:

Code-Ausschnitt 5.7.10

```

1  #!/bin/bash
3  # please update this script to fit your needs
4  # this script is call every time the according viewelement will be drawn, so keep it ↔
   short and fast ;)
6  OUTPUTFLDR="/tmp/skindesigner/"
7  mkdir -p ${OUTPUTFLDR}
9  # there can be 2 files, vdrcpu and vdrmem
10 rm -f ${OUTPUTFLDR}/vdrcpu ${OUTPUTFLDR}/vdrmem
12 read vdr_cpu vdr_mem <<(ps -C vdr -o %cpu,%mem= | sort | tail -n1)
13 echo $vdr_cpu > ${OUTPUTFLDR}/vdrcpu
14 echo $vdr_mem > ${OUTPUTFLDR}/vdrmem

```

Im nächsten Schritt installieren wir die Wettervorhersage für Metrix HD. Diese wird auch für viele anderen Skins verwendet, die das Wetter anzeigen. Das Wettervorhersage-Plugin heißt *weaterforecast*. Wir installieren es mit den benötigten Abhängigkeiten wie folgt:

Code-Ausschnitt 5.7.11

```

1  cd /home/pi/tv/VDR/PLUGINS/src
2  git clone git://projects.vdr-developer.org/vdr-plugin-weatherforecast.git ↔
   weatherforecast
3  sudo apt-get install libcurl4-openssl-dev libjansson-dev
4  cd ../../
5  make
6  sudo make install

```



Vielleicht ist Ihnen aufgefallen, dass ich hinter dem *git clone*-Kommando einen Verzeichnisnamen *weatherforecast* angegeben habe. In diesem Fall kopiert der *git*-Befehl alle Dateien in das angegebene Verzeichnis. Wir sparen uns dann, das Verzeichnis *vdr-plugin-weatherforecast* umzubenennen.

Da das Plugin *weatherforecast* keine Setup-Parameter hat, können wir in unsere *VDR*-Startdatei */usr/local/bin/runvdr* folgende Zeile ergänzen:

Code-Ausschnitt 5.7.12

```

1  VDRPLUGINS=" -Prpindhdevice \
2             -Pweatherforecast \
3             -P'skindesigner -l /var/lib/vdr/plugins/skindesigner/logos -e /video/cache/↔
   epgimages -i /var/lib/vdr/plugins/skindesigner/installerskins'"

```

Beim Start sucht das Plugin automatisch nach einem Ort in der Nähe der eigenen IP-Adresse und stellt dessen Wetterdaten dar. Um die Abfragen beim Wetter-Server zu limitieren, cached das Plugin die Abfragen. Bitte haben Sie Geduld, falls das Wetter nicht sofort angezeigt wird. Eventuell sind die 1.000 freien Abfragen beim Wetterdienst bereits aufgebraucht.

Als Nächstes machen wir uns daran, das Metrix HD-Skin mit schönen Senderlogos auszustatten. Die Logos laden wir von <https://github.com/FrodoVDR/channellogos> im Verzeichnis */var/lib/vdr/plugins/skindesigner/logos*, das wir zuvor angelegt haben, herunter. Ich verwende die *light*-Logos.

Code-Ausschnitt 5.7.13

```

1 sudo su
2 mkdir /var/lib/vdr/plugins/skindesigner/logos
3 cd /var/lib/vdr/plugins/skindesigner/logos
4 git clone https://github.com/FrodoVDR/channellogos
5 cp channellogos/logos-light/* .

```

Nach einem Neustart des VDR sind die Senderlogos sichtbar.

5.7.1 Stream me up!

In diesem Abschnitt zeige ich Ihnen noch, wie Sie das aktuelle Fernsehprogramm auf ein mobiles Gerät (Telefon, Tablet oder Laptop) streamen können. Dazu verpassen wir dem VDR den *streamdev-server*:

Code-Ausschnitt 5.7.14

```

1 cd /home/pi/tv/VDR/PLUGINS/src
2 git clone git://projects.vdr-developer.org/vdr-plugin-streamdev.git streamdev
3 cd ../../
4 make
5 sudo make install

```

Streamdev installiert praktischerweise auch einen Klienten mit, den Sie dazu nutzen können, das TV-Programm von einem VDR auf einen anderen zu streamen. Wir nutzen aber den Server, um das Programm auf ein Tablet oder ein anderes mobiles Gerät zu übertragen. Tragen wir aber zunächst das Plugin in unser VDR-Startskript ein:

Code-Ausschnitt 5.7.15

```

1 VDRPLUGINS=" -Prpnhdddevice \
2             -Pweatherforecast \
3             -Pstreamdev-server \
4             -P'skindesigner -l /var/lib/vdr/plugins/skindesigner/logos -e /video/cache/↔
               epgimages -i /var/lib/vdr/plugins/skindesigner/installerskins'"

```



Üblicherweise spielt die Reihenfolge der Plugins keine Rolle. Sollte aber dennoch das eine oder andere Plugin nicht starten wollen, ändern Sie die Reihenfolge. Es gibt einige Plugins, die gerne an erster Stelle stehen wollen.

Nach diesen Vorbereitungen müssen wir dem Server noch mitteilen, welche Klienten denn überhaupt mit dem Server verbinden dürfen.

Code-Ausschnitt 5.7.16

```

1 cd /home/pi/tv/VDR/PLUGINS/src/streamdev/streamdev-server
2 sudo cp streamdevhosts.conf /var/lib/vdr/plugins/streamdev-server

```

Die Default-Einstellungen erlauben lediglich eine Verbindung vom selben Server (*localhost*). Das ist langweilig, denn wir möchten ja vielleicht ein Tablet verbinden, welches sich im selben Netz (z. B. WLAN) befindet. Ergänzen wir die Datei *streamdevhosts.conf* also um folgenden Eintrag:

Code-Ausschnitt 5.7.17

```
1 sudo vi /var/lib/vdr/plugins/streamdev-server
2 0.0.0.0/0 # any host on any net (DON'T DO THAT! USE AUTHENTICATION)
```



Abbildung 5.14: Das Logo des VLC Media-Players (Quelle: www.vlc.de).

Damit erlauben wir kurzfristig jede Verbindung zum Server, egal von wo. Die Datei selbst gibt Beispiele, wie Sie nur Ihr eigenes Netzwerk erlauben. Für einen kurzen Test ist diese allgemeine Freigabe aber in Ordnung. Danach starten Sie bitte den *VDR*, etwa durch Aufrufen des Skripts `/usr/local/bin/runvdr`. Zum Abspielen des Fernseh-Streams empfehle ich den *VLC* Media-Player. Der *VLC* Video-Player steht für fast jedes mobile und auch stationäre Device zum kostenlosen Download zur Verfügung und kann darüber hinaus sehr viele verschiedene Formate flüssig darstellen. Standardmäßig benutzt *streamdev* den Port 3000, um den Stream abzuliefern und das *TS*-Format (*Transport Stream*). Damit fehlt nur noch der Kanal als letzter Parameter, um die Streaming-URL perfekt zu machen. Eine typische URL ist so aufgebaut:

Code-Ausschnitt 5.7.18

```
1 http://192.168.178.66:3000/TS/1
```

Bitte ersetzen Sie die oben angegebene IP-Adresse durch die Ihres Streaming-Raspberry Pi. Ersetzen Sie weiterhin die Kanalnummer (1) durch die Nummer des Kanals, welchen Sie streamen möchten. Eine Änderung der Kanalnummer sorgt übrigens dafür, dass das Frontend automatisch auf diesen Kanal umschaltet. *TS* bezeichnet eine Datenkodierung. Folgende Kodierungen stehen zur Verfügung: *PES*, *TS*, *PS*, *ES* und *Extern*. Falls eine nicht funktioniert, probieren Sie eine andere. *Extern* greift auf die Datei `externremux.sh` zurück und wird auf dem Raspberry Pi nicht funktionieren, da die CPU des Winzlings zu schwach ist, um das *TV*-Format live in ein anderes umzuwandeln. Die Zahl hinter der Kodierung bezeichnet den ausgewählten Fernsehkanal, den man schauen möchte. Er entspricht der Eintragsnummer in der *VDR*-Kanalliste `channels.conf`.



VDR kann man übrigens auch von einem Terminal aus bedienen. Dazu verwendet man das Programm `svdrpsend` aus dem *VDR*-Verzeichnis. Dieses kopiert man nach `/usr/local/bin`, damit es überall verfügbar ist. Durch den Aufruf

Code-Ausschnitt 5.7.19

```
1 svdrpsend CHAN 2
```

schaltet man auf den zweiten Kanal der Kanalliste um. Eine komplette Hilfe zu `svdrpsend` findet sich unter <http://www.vdr-wiki.de/wiki/index.php/SVDRP>.

Zusammenfassung 5 In diesem Kapitel haben wir einen digitalen Videorekorder gebaut, der nicht nur Fernsehprogramm empfangen und darstellen, sondern auch aufzeichnen kann. Den *VDR* haben wir mit einer Fernbedienung verbunden, die wir mit Hilfe einer kleinen Infrarot-Diode an den Raspberry Pi gekoppelt haben.

Verschiedene Plugins erweitern die Videorekorder-Funktionalität und hübschen das Aussehen des *VDR* auf und erlauben das Streaming des Fernsehprogramms auf ein mobiles Gerät im Heimnetz.

Weiterhin habe ich Ihnen gezeigt, wie Sie eine externe Festplatte an Ihren Receiver anschließen und nutzen können, um Fernsehprogramm aufzuzeichnen.

Im nächsten Kapitel werfen wir einen Blick auf *KODI*, den Nachfolger des XBox-Media-Center (*XBMC*). Wir werden *KODI* nicht nur aus den Sourcen installieren, sondern uns auch die *Libreelec*-Distribution näher anschauen. ■

KODI aus dem Repository
KODI aus dem Quelltext
Übersetzungsfehler und Neuerungen
LibreELEC
AirPlay und Co.
Externalplayer
MP3
Ausblick

6 — KODI

In diesem Kapitel schauen wir uns das Mediacenter *KODI* näher an. Dabei möchte ich Ihnen drei verschiedene Wege vorstellen, um *KODI* auf den Bildschirm zu zaubern. Wir werden es per *apt-get* installieren, aus dem Quelltext übersetzen und wir werfen einen Blick auf die *Libreelec*-Distribution. Alle drei Wege haben ihre Vor- und Nachteile. Möchten Sie unter Raspbian arbeiten, weil Sie beispielsweise Compiler und andere Tools benötigen, macht es sicherlich Sinn, *KODI* aus dem Repository zu installieren. *Libreelec* führt hingegen bereits nach wenigen Minuten zum Erfolgserlebnis, allerdings gibt es hier nicht die Möglichkeit, weitere Programme zu installieren außer denen, die für *Libreelec* vorgesehen sind. Wollen Sie Ihren Raspberry Pi also als reines Multimediacenter nutzen, dann ist *Libreelec* vielleicht genau das Richtige für Sie. Weiterhin werden wir einen Blick auf wichtige Erweiterungen werfen, wie beispielsweise einem Fernseh-Backend namens *tvheadend* oder den Airplay-Streaming-Service, der es ermöglicht, Musik drahtlos von einem Apple-Device (iPhone, iPad) über den Raspberry Pi zu einer Stereoanlage zu übertragen.



KODI

KODI ist aus dem Xbox Media Center - kurz *XBMC* - hervorgegangen. *XBMC* wurde umbenannt, um rechtliche Probleme mit der Xbox zu vermeiden. Inzwischen läuft *KODI* längst nicht mehr nur auf Microsoft's Spielekonsole, sondern auf vielen Embedded Boards und PCs. Dabei kann man auf eine Vielzahl von fertigen Distributionen oder sogar Bundles (Embedded Board mit installiertem *KODI*) zurückgreifen - der ideale Einstieg für Neulinge. Zu den bekanntesten Distributionen zählen *OpenElec* [lmd14], (<http://openelec.tv>) oder *Xbian* (<http://www.xbian.org/>).

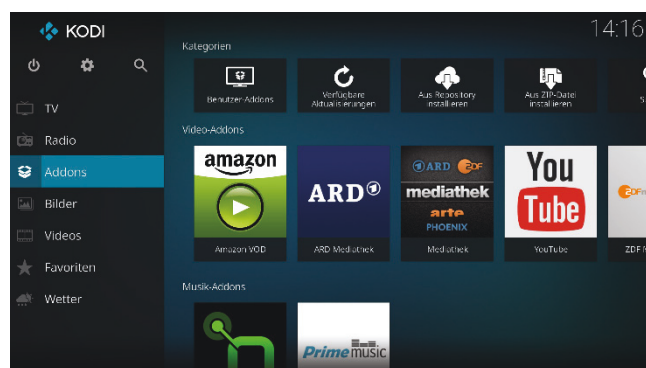


Abbildung 6.1: Das KODI Mediacenter spielt Filme, zeigt Bilder und vieles mehr auf dem Raspberry Pi.

Allerdings haben diese Distributionen alle eine gemeinsame Schwäche: Sie sind für Einsteiger gedacht und bringen keine Entwickler-Werkzeuge mit sich. Sie werden also Probleme haben, beispielsweise zwischen reinem *VDR*- und *KODI*-Betrieb umzuschalten oder *KODI* vom *VDR* aus aufzurufen. Auch für *KODI* gibt es eine Fülle an Plugins, um den *VDR* einzubinden, die aktuelle Wettervorhersage abzurufen oder um YouTube-Videos zu schauen.

6.1 *KODI* aus dem Repository

Beginnen wir damit, *KODI* aus einer Konsole heraus per *apt-get* zu installieren.

Code-Ausschnitt 6.1.1

```
1 sudo apt-get install kodi
```

Nach erfolgreicher Installation können Sie *KODI* durch die Eingabe von *kodi* in einem Terminal starten. Das Programm steht ebenfalls der Programmstartleiste in der Rubrik *Unterhaltungsmedien* zur Verfügung.



KODI benötigt mindestens 160 MB Speicher für die GPU. Für den Raspberry Pi 3 und neuer empfehle ich ein Minimum von 256 MB. Erinnern Sie sich? Den Wert des Speichers für die GPU können Sie in der Datei *config.txt* festlegen (*gpu_mem=256*).

6.2 *KODI* aus dem Quelltext

Auf der Raspberry Pi Foundation-Webseite wurde Ende 2013 angekündigt, dass der Programmierer mit dem Pseudonym *popcornmix* einen eigenen *KODI*-Zweig aufgemacht hat, der *newclock3* heißt. Dieser enthält einige Geschwindigkeitsverbesserungen auf dem Raspberry Pi im Vergleich zum normalen *KODI*-Zweig (<http://www.raspberrypi.org/archives/4986>). Den Zweig findet man unter <https://github.com/popcornmix/xbmc/commits/newclock3>. Beginnen wir damit, im Verzeichnis */home/pi/tv* ein Unterverzeichnis *xbmc* zu erstellen, in das wir den Quelltext herunterladen. Öffnen Sie einen Webbrowser und navigieren Sie zur Seite <https://github.com/popcornmix/xbmc/tree/newclock3>. Wählen Sie dort die Option *Download ZIP*, um die neueste Version von *KODI* im Zip-Format herunterzuladen. Kopieren Sie die Version nach dem Herunterladen in das oben angegebene Verzeichnis und packen Sie sie mit dem Befehl

Code-Ausschnitt 6.2.1

```
1 sudo apt-get install gzip unzip
2 gunzip xbmc-newclock3.zip
```

aus. Das dazu erforderliche Programm *gzip* installieren wir davor. *KODI* hängt von ganz vielen anderen Paketen ab, die wir alle installieren müssen, bevor das Programm übersetzt werden kann. Das ist eine beeindruckend lange Liste:

Code-Ausschnitt 6.2.2

```

1 sudo apt-get install build-essential autoconf ccache gawk gperf mesa-utils zip unzip <-
  sudo apt-get install autotools-dev comerr-dev dpkg-dev libalsaplayer-dev libapt-pkg-<-
  dev:armhf libasound2-dev:armhf libass-dev:armhf libatk1.0-dev libavahi-client-dev <-
  libavahi-common-dev libavcodec-dev libavformat-dev libavutil-dev libbison-dev:armhf <-
  libbluray-dev:armhf libboost1.49-dev libbz2-dev:armhf libc-dev-bin libc6-dev:armhf <-
  libcaca-dev libcairo2-dev libcdio-dev libcllalsadrv-dev libcrypto++-dev libcups2-dev <-
  libcurl3-gnutls-dev libdbus-1-dev libdbus-glib-1-dev libdirectfb-dev libdrm-dev <-
  libegl1-mesa-dev libelf-dev libenca-dev libept-dev libevent-dev libexpat1-dev <-
  libflac-dev:armhf libfontconfig1-dev libfreetype6-dev libfribidi-dev libgconf2-dev <-
  libgcrpt11-dev libgdk-pixbuf2.0-dev libgl1-mesa-dev libgles2-mesa-dev libglew-dev:<-
  armhf libglewmx-dev:armhf libglib2.0-dev libglu1-mesa-dev libgnome-keyring-dev <-
  libgnutls-dev libgpg-error-dev libgtk2.0-dev libhal-dev libhunspell-dev:armhf libice-<-
  -dev:armhf libicu-dev libidn11-dev libiso9660-dev libjasper-dev libjbig-dev:armhf <-
  libjconv-dev libjpeg8-dev:armhf libkrb5-dev libldap2-dev:armhf libltdl-dev:armhf <-
  liblzo2-dev libmad0-dev libmicrohttpd-dev libmodplug-dev libmp3lame-dev:armhf <-
  libmpeg2-4-dev libmysqlclient-dev libncurses5-dev libnspr4-dev libnss3-dev libogg-<-
  dev:armhf libopenal-dev:armhf libp11-kit-dev libpam0g-dev:armhf libpango1.0-dev <-
  libpcre++-dev libpcre3-dev libpixman-1-dev libpng12-dev libprotobuf-dev libpthread-<-
  stubs0-dev:armhf libpulse-dev:armhf librtp-dev libsamplerate0-dev:armhf libSDL-<-
  image1.2-dev:armhf libSDL1.2-dev libslang2-dev:armhf libsm-dev:armhf libsmclient-<-
  dev:armhf libspeex-dev:armhf libsqlite3-dev libssh-dev libssh2-1-dev libssl-dev <-
  libstdc++6-4.6-dev libtagcoll2-dev libtasn1-3-dev libtiff4-dev libtinfo-dev:armhf <-
  libtinyclang-dev libtst-dev:armhf libudev-dev libv8-dev libva-dev:armhf libvdpau-dev:<-
  armhf libvorbis-dev:armhf libvpx-dev:armhf libwebp-dev:armhf libwibble-dev libx11-<-
  dev:armhf libx11-xcb-dev libxapian-dev libxau-dev:armhf libxcb-glx0-dev:armhf libxcb-<-
  -render0-dev:armhf libxcb-shm0-dev:armhf libxcb1-dev:armhf libxcomposite-dev <-
  libxcursor-dev:armhf libxdamage-dev libxdmcp-dev:armhf libxext-dev:armhf libxfixes-<-
  dev libxft-dev libxi-dev libxinerama-dev:armhf libxml2-dev:armhf libxmu-dev:armhf <-
  libxrandr-dev libxrender-dev:armhf libxslt1-dev libxss-dev:armhf libxt-dev:armhf <-
  libxtst-dev:armhf libxxf86vm-dev libyajl-dev libzip-dev linux-libc-dev:armhf lzma-<-
  dev mesa-common-dev python-dev python2.7-dev x11proto-composite-dev x11proto-core-<-
  dev x11proto-damage-dev x11proto-dri2-dev x11proto-fixes-dev x11proto-glx-dev <-
  x11proto-input-dev x11proto-kb-dev x11proto-randr-dev x11proto-record-dev x11proto-<-
  render-dev x11proto-scrnsaver-dev x11proto-xext-dev x11proto-xf86vidmode-dev <-
  x11proto-xinerama-dev xtrans-dev zlib1g-dev:armhf

```



Sollte eines der Pakete nicht installierbar sein, da es durch eine neuere Version ersetzt worden ist, ändern Sie die Paketversion bitte einfach auf die neuere Version. *aptitude* kann Ihnen dabei helfen, Abhängigkeiten korrekt aufzulösen.

Da *KODI* in der Lage ist, DVDs abzuspielen, darf an dieser Stelle der Hinweis auf die *libdvdcss* nicht fehlen. Diese Bibliothek ist rechtlich umstritten, da sie in der Lage ist, die DVD-CSS-Verschlüsselung von Kauf-DVDs auszuhebeln. Ein finnisches Gericht hat 2007 entschieden, dass die Nutzung der *libdvdcss* legal ist. In Finnland könnte man die Bibliothek also einfach mittels

Code-Ausschnitt 6.2.3

```

1 sudo apt-get install libdvdcss2

```

installieren oder die neueste Version von *Videolan* mit den folgenden Kommandos herunterladen und installieren:

Code-Ausschnitt 6.2.4

```

1 git clone git://git.videolan.org/libdvdcss
2 cd libdvdcss
3 make
4 sudo make install

```

In Deutschland ist die rechtliche Situation jedoch nicht klar (<http://de.wikipedia.org/wiki/Libdvdcss>).

Zum Übersetzen von *KODI* wird besonders bei dem kleinen Raspberry Pi mit wenig Speicher eine Swap-Datei benötigt, die mindestens 1 GB groß ist. Falls Sie noch keine Swap-Datei angelegt und aktiviert haben, kann das mit folgenden Befehlen erledigt werden:

Code-Ausschnitt 6.2.5

```
1 dd if=/dev/zero of=/home/pi/swapdatei bs=1024 count=204800
2 sudo mkswap /home/pi/swapdatei
3 sudo chown root:root /home/pi/swapdatei
4 sudo chmod 600 /home/pi/swapdatei
5 sudo swapon /home/pi/swapdatei
```

Der Disk-Dump-Befehl *dd* erstellt eine leere Datei namens *swapdatei*, weist dieser das Attribut *swap* zu, ändert den Besitzer und die Gruppenzugehörigkeit auf *root*, stellt die Berechtigungen auf 600, d. h. *root* darf lesen und schreiben, alle anderen dürfen nichts.



Hinter dem *chmod*-Befehl stehen immer drei Zahlen. Diese sind binär zu verstehen:

- 0: Keine Rechte, binär 000
- 1: Ausführrechte, binär 001
- 2: Schreibrechte, binär 010
- 3: Schreib- und Ausführrechte, binär 011
- 4: Leserechte, binär 100
- 5: Lese- und Ausführrechte, binär 101
- 6: Lese- und Schreibrechte, binär 110
- 7: Volle Zugriffsrechte, binär 111 Die Ziffer (0..7) taucht dabei dreimal auf. Wie bereits vorher erklärt, steht die erste Ziffer für den Benutzer *root*, die zweite Ziffer für Nutzer, die der zugehörigen Gruppe angehören, und die dritte Ziffer für alle anderen Nutzer.

Abschließend wird der Swap-Speicher unter Verwendung der eingerichteten Swap-Datei eingeschaltet. Damit *KODI* fehlerfrei übersetzt werden kann, müssen einige Header-Dateien kopiert und symbolische Links angelegt werden:

Code-Ausschnitt 6.2.6

```

1  sudo cp -R /opt/vc/include/* /usr/include
2  sudo cp /opt/vc/include/interface/vcos/pthreads/* /usr/include/interface/vcos
3  sudo ln -fs /opt/vc/lib/libEGL.so /usr/lib/libEGL.so
4  sudo ln -fs /opt/vc/lib/libEGL.so /usr/lib/arm-linux-gnueabi/libEGL.so
5  sudo ln -fs /opt/vc/lib/libEGL.so /usr/lib/arm-linux-gnueabi/libEGL.so.1
6  sudo ln -fs /opt/vc/lib/libEGL_static.a /usr/lib/libEGL_static.a
7  sudo ln -fs /opt/vc/lib/libEGL_static.a /usr/lib/arm-linux-gnueabi/libEGL_static.a
8  sudo ln -fs /opt/vc/lib/libGLESv2.so /usr/lib/libGLESv2.so
9  sudo ln -fs /opt/vc/lib/libGLESv2.so /usr/lib/arm-linux-gnueabi/libGLESv2.so
10 sudo ln -fs /opt/vc/lib/libGLESv2.so /usr/lib/arm-linux-gnueabi/libGLESv2.so.2
11 sudo ln -fs /opt/vc/lib/libGLESv2_static.a /usr/lib/libGLESv2_static.a
12 sudo ln -fs /opt/vc/lib/libGLESv2_static.a /usr/lib/arm-linux-gnueabi/libGLESv2_←
    .a
13 sudo ln -fs /opt/vc/lib/libbcm_host.so /usr/lib/libbcm_host.so
14 sudo ln -fs /opt/vc/lib/libbcm_host.so /usr/lib/arm-linux-gnueabi/libbcm_host.so
15 sudo ln -fs /opt/vc/lib/libvchiq_arm.a /usr/lib/libvchiq_arm.a
16 sudo ln -fs /opt/vc/lib/libvchiq_arm.a /usr/lib/arm-linux-gnueabi/libvchiq_arm.a
17 sudo ln -fs /opt/vc/lib/libvchiq_arm.so /usr/lib/libvchiq_arm.so
18 sudo ln -fs /opt/vc/lib/libvchiq_arm.so /usr/lib/arm-linux-gnueabi/libvchiq_arm.so
19 sudo ln -fs /opt/vc/lib/libvcos.a /usr/lib/libvcos.a
20 sudo ln -fs /opt/vc/lib/libvcos.a /usr/lib/arm-linux-gnueabi/libvcos.a
21 sudo ln -fs /opt/vc/lib/libvcos.so /usr/lib/libvcos.so
22 sudo ln -fs /opt/vc/lib/libvcos.so /usr/lib/arm-linux-gnueabi/libvcos.so

```

Dies sorgt dafür, dass die Header-Dateien und Bibliotheken an den Stellen gefunden werden, an denen sie von den *KODI*-Makefiles gesucht werden. Da ein paar Fehler bezüglich des Raspberry Pi im Quelltext von *KODI* korrigiert werden müssen, installieren wir zunächst das Tool *sed*.

Code-Ausschnitt 6.2.7

```

1  sudo apt-get install sed

```

sed ist ein sog. *Stream Editor*, also ein nicht-interaktiver Texteditor, der hervorragend dazu verwendet werden kann, in Quelltexten bestimmte Passagen durch andere zu ersetzen. Und genau das erledigt der nächste Aufruf:

Code-Ausschnitt 6.2.8

```

1  sudo sed -i 's/#include "vhost_config.h"/#include"linux\vhost_config.h"/' /usr/←
    include/interface/vmcs_host/vcgencmd.h
2  sed -i 's/USE_BUILDR00T=1/USE_BUILDR00T=0/' tools/rbp/setup-sdk.sh
3  sed -i 's/TOOLCHAIN=\usr\local\bcm-gcc/TOOLCHAIN=\usr/' tools/rbp/setup-sdk.sh

```

Bevor es nun an das Konfigurieren der *KODI*-Übersetzung geht, installieren wir noch drei Bibliotheken, auf die *KODI* angewiesen ist: *taglib*, *libcec* und *libshairport*. Die *libcec* muss noch heruntergeladen werden, da die Raspbian Repository-Version zu alt ist. Los geht's:

Code-Ausschnitt 6.2.9

```

1 cd ~/tv/xbmc/xbmc-newclock3
2 make -C lib/taglib
3 sudo make -C lib/taglib install
4 cd ~/tv/xbmc
5 git clone --depth 5 https://github.com/Pulse-Eight/libcec.git
6 cd libcec
7 ./bootstrap
8 ./configure --prefix=/usr/local
9 make
10 sudo make install
11 cd ~/tv/xbmc/xbmc-newclock3
12 make -C lib/libshairport
13 sudo make -C lib/libshairport install

```

Als Nächstes müssen einige Shell-Variablen exportiert werden. Falls Sie *KODI* häufiger kompilieren, können Sie die folgenden Befehle auch in einem Skript speichern, dieses ausführbar machen und dann vor dem nächsten Kompilieren einfach das Skript aufrufen.

Code-Ausschnitt 6.2.10

```

1 cd ~/tv/xbmc/xbmc-newclock3
2 export TARGET_SUBARCH="armv6zk"
3 export TARGET_CPU="arm1176jzf-s"
4 export TARGET_FLOAT="hard"
5 export TARGET_FPU="vfp"
6 export TARGET_FPU_FLAGS="-mfloat-abi=$TARGET_FLOAT -mfpu=$TARGET_FPU"
7 export TARGET_EXTRA_FLAGS="-Wno-psabi -Wa,-mno-warn-deprecated"
8 export TARGET_COPT="-Wall -pipe -fomit-frame-pointer -O3 -fexcess-precision=fast -ffast-↵
    math -fgnu89-inline"
9 export TARGET_LOPT="-s -Wl,--as-needed"
10 export CFLAGS="-march=$TARGET_SUBARCH -mcpu=$TARGET_CPU$TARGET_FPU_FLAGS -mabi=aapcs-↵
    linux $TARGET_COPT $TARGET_EXTRA_FLAGS"
11 export CXXFLAGS="$CFLAGS"
12 export LDFLAGS="-march=$TARGET_SUBARCH -mtune=$TARGET_CPU $TARGET_LOPT"

```

Führen Sie danach bitte das Setup-Skript für den Raspberry Pi aus, bevor dann noch eine weitere *sed*-Ersetzung erfolgt und die Abhängigkeiten für den Pi übersetzt werden.

Code-Ausschnitt 6.2.11

```

1 sudo sh tools/rbp/setup-sdk.sh
2 sed -i 's/cd $(SOURCE); $(CONFIGURE)/#cd $(SOURCE); $(CONFIGURE)/' tools/rbp/depends/↵
    xbmc/Makefile
3 make -C tools/rbp/depends/xbmc

```

Jetzt wird *XBMC* konfiguriert:

Code-Ausschnitt 6.2.12

```

1 ./configure --prefix=/usr/local --build=arm-linux-gnueabi --host=arm-linux-gnueabi ↵
    --localstatedir=/var/lib --with-platform=raspberry-pi --disable-gl --enable-gles --↵
    disable-x11 --disable-sdl --enable-ccache --enable-optimizations --disable-external-↵
    libraries --disable-goom --disable-hal --disable-pulse --disable-vaapi --disable-↵
    vdpau --disable-xrandr --enable-airplay --disable-alsa --enable-avahi --enable-↵
    libbluray --enable-dvdcss --disable-debug --disable-joystick --disable-mid --enable-↵
    nfs --disable-profiling --disable-projectm --enable-rsxs --enable-rtmp --disable-↵
    vaapi --disable-viddecoder --disable-external-ffmpeg --enable-optical-drive --enable↵
    -player=omxplayer

```

Das bedeutet, dass alle gewünschten Eigenschaften angewählt (*enable*) und alle nicht benötigten Eigenschaften abgewählt (*disable*) werden.

Auch diese Befehle können Sie, wenn Sie sie häufiger benötigen, in ein eigenes Skript auslagern, damit Sie nicht jedes Mal nachschauen müssen, welche Eigenschaften für den Pi erforderlich sind und welche nicht. Wenn die Konfiguration erfolgreich durchgelaufen ist, muss ein letztes *sed*-Kommando zur Fehlerkorrektur ausgeführt werden, bevor das Kompilieren mit einem profanen *make* gestartet werden kann.

Code-Ausschnitt 6.2.13

```
1 sed -i 's/ifeq (1,1)/ifeq (0,1)/' tools/TexturePacker/Makefile
2 make
```

Das Übersetzen von *KODI* auf einem Pi der ersten Generation dauert ungefähr zwölf Stunden. Falls während der Übersetzung Fehler auftreten und Sie die Konsole nicht mehr bis zur Fehlermeldung zurückscrollen können, kann *make* auch wie folgt aufgerufen werden:

Code-Ausschnitt 6.2.14

```
1 make 2>&1 | tee logdatei.log
```

Die Kompiler-Meldungen werden dann in der Datei *logdatei.log* gespeichert, die man im Fehlerfall in Ruhe durchsehen kann.

Info Die letzte Zeile enthält gleich zwei neue Dinge: zunächst das Programm *tee* und dann auch noch eine Ausgabeumleitung (*2>&1*). Das Programm *tee* wirkt wie ein T-Stück einer Wasserleitung. Das Wasser fließt durch das *tee* durch, aber auch an der Verzweigung des *tees* nach unten. Die Ausgabe des *Makefiles* wird also quasi verdoppelt. Einmal wird sie im Terminal angezeigt (man sieht also den Kompilierfortschritt), zum Anderen wird sie über eine Ausgabeumleitung in eine Logdatei umgeleitet.

Info Unter LINUX gibt es folgende Ein- und Ausgabeumleitungen:

- *befehl < datei*: Standardeingabe, Lesen aus *datei*
- *befehl > datei*: Standardausgabe, Schreiben in *datei*
- *befehl >> datei*: Standardausgabe, Anhängen an *datei*
- *befehl 2> datei*: Standardfehlerausgabe, Schreiben der Fehlermeldungen in *datei*
- *befehl 2>> datei*: Standardfehlerausgabe, Anhängen der Fehlermeldungen in *datei*
- *befehl > datei 2>&1*: Standard- und Standardfehlerausgabe, Schreiben in *datei*

Nachdem *KODI* erfolgreich übersetzt worden ist, kann es mit einem

Code-Ausschnitt 6.2.15

```
1 sudo make install
```

installiert werden. Danach können Sie die Swap-Datei wieder ausschalten (Kapitel 4.1.1). *KODI* sollte nun bereits vollständig mit Maus und Tastatur bedienbar sein. Eine Fernbedienung fügen wir später hinzu.

Code-Ausschnitt 6.2.16

```
1 /usr/local/lib/xbmc/xbmc.bin
```

Info Falls Sie *KODI* nicht installieren wollen oder erst einmal testen wollen, können Sie es auch aus dem Verzeichnis, in dem Sie übersetzt haben, durch *.xbmc.bin* starten.

6.3 Übersetzungsfehler und Neuerungen

Nicht immer gelingt das Übersetzen von *KODI* fehlerfrei, je nach Stand des Repositories. Dieser Abschnitt gibt ein paar Anhaltspunkte, die im Fehlerfall hilfreich sein können. Wenn der Compiler mit einer Fehlermeldung aussteigt, dass ein *struct* in der Datei *EGLNativeTypeRaspberryPI.cpp* nicht definiert ist, fügen Sie bitte an den Anfang dieser Datei (hinter der letzten *#include*-Anweisung) Folgendes ein:

Code-Ausschnitt 6.3.1

```
1 typedef struct {
2     DISPMANX_ELEMENT_HANDLE_T element;
3     int width;
4     int height;
5 } EGL_DISPMANX_WINDOW_T;
```

Taucht ein weiterer Fehler in dieser Datei auf, ändern Sie bitte Zeile 135 (oder in der Nähe dieser Zeile) zu

Code-Ausschnitt 6.3.2

```
1 m_nativeWindow = (EGLNativeWindowType*)calloc(1, sizeof(EGL_DISPMANX_WINDOW_T));
```

Falls der Compiler die Include-Dateien in *tools/TexturePacker/XBTFWriter.cpp* nicht finden kann, geben Sie bitte den vollen Pfad für die Include-Dateien in dieser Datei an:

Code-Ausschnitt 6.3.3

```
1 #include "/home/pi/tv/xbmc/xbmc-newclock3/xbmc/guilib/XBTF.h"
2 #include "/home/pi/tv/xbmc/xbmc-newclock3/xbmc/utils/EndianSwap.h"
```

Achten Sie bitte darauf, dass der Anfang des Pfades dem von Ihnen gewählten Verzeichnis entspricht. Dasselbe kann für die Datei *tools/TexturePacker/XBMCTex.cpp* nötig sein

Code-Ausschnitt 6.3.4

```
1 #include "/home/pi/tv/xbmc/xbmc-newclock3/xbmc/guilib/XBTF.h"
2 #include "/home/pi/tv/xbmc/xbmc-newclock3/lib/libsquish/squish.h"
```

oder für die Datei *xbmc/utils/EndianSwap.h*

Code-Ausschnitt 6.3.5

```
1 #include "/home/pi/tv/xbmc/xbmc-newclock3/xbmc/config.h"
```

oder für die Datei *tools/TexturePacker/cmdlineargs.h*

Code-Ausschnitt 6.3.6

```
1 #include "/home/pi/tv/xbmc/xbmc-newclock3/xbmc/linux/PlatformDefs.h"
```

oder für die Datei *tools/TexturePacker/xwinapi.h*

Code-Ausschnitt 6.3.7

```
1 #include "/home/pi/tv/xbmc/xbmc-newclock3/xbmc/linux/PlatformDefs.h"
```



Die hier vorgestellte Liste kann nicht vollständig sein, da sich das *KODI*-Repository ständig in der Entwicklung befindet. Eine funktionierende *KODI*-Version finden Sie im Download-Bereich <https://www.springer.com/de/book/978-3-662-58143-8> unter dem Namen *xbmc.tar.bz2*.

Inzwischen hat sich einiges im *KODI*-Quelltext getan. Bei aktuellen *KODI*-Versionen, wie etwa dem *newclock5*-Zweig, wurde das Build-System durch *cmake* ersetzt. Erstellen Sie in diesem Fall bitte im *xbmc*-Verzeichnis in Verzeichnis mit dem *build* und rufen Sie den folgenden *cmake*-Befehl zum Übersetzen auf:

Code-Ausschnitt 6.3.8

```
1  cmake \  
2  -DVERBOSE=1 \  
3  -DCORE_SYSTEM_NAME=rbpi \  
4  -DENABLE_MMAL=ON \  
5  -DENABLE_OPENGL=OFF \  
6  -DWITH_CPU=CORTEX-A7 \  
7  -DCMAKE_PREFIX_PATH=/opt/vc \  
8  -DENABLE_OPENGL=ON \  
9  -DCMAKE_BUILD_TYPE=Release \  
10 -DCMAKE_INSTALL_PREFIX=/usr \  
11 -DENABLE_AIRTUNES=ON \  
12 -DENABLE_ALSA=ON \  
13 -DENABLE_AVAHI=ON \  
14 -DENABLE_BLURAY=ON \  
15 -DENABLE_CEC=ON \  
16 -DENABLE_DBUS=ON \  
17 -DENABLE_DVDCSS=ON \  
18 -DENABLE_EGL=ON \  
19 -DENABLE_EVENTCLIENTS=ON \  
20 -DENABLE_INTERNAL_FFmpeg=ON \  
21 -DENABLE_INTERNAL_CROSSGUID=ON \  
22 -DENABLE_MICROHTTTPD=ON \  
23 -DENABLE_MYSQLCLIENT=ON \  
24 -DENABLE_NFS=ON \  
25 -DENABLE_NONFREE=ON \  
26 -DENABLE_OPENSSL=ON \  
27 -DENABLE_OPTICAL=ON \  
28 -DENABLE_PULSEAUDIO=ON \  
29 -DENABLE_SMBCLIENT=ON \  
30 -DENABLE_SSH=ON \  
31 -DENABLE_UDEV=ON \  
32 -DENABLE_UPNP=ON \  
33 -DENABLE_VAAPI=OFF \  
34 -DENABLE_VDPAU=OFF \  
35 -DENABLE_X11=OFF \  
36 -DENABLE_XSLT=ON \  
37 -DENABLE_LIRC=ON \  
38 -DCPACK_GENERATOR=DEB \  
39 -DDEBIAN_PACKAGE_VERSION=1~ \  
40 -DDEB_PACKAGE_ARCHITECTURE=armhf \  
41 ../cmake/
```

6.4 LibreELEC

LibreELEC – Just enough OS for KODI. Mit diesem Slogan wirbt die Webseite der *LibreELEC*-Macher (<https://libreelec.tv>). Und der Slogan ist Programm. *LibreELEC* installiert lediglich soviel - oder besser gesagt so wenig - Betriebssystem, wie es zum Betrieb von *KODI* erforderlich ist. *LibreELEC* selbst ist aus dem Open-Source-Projekt *OpenELEC* hervorgegangen, nachdem verschiedene Entwickler unterschiedliche Meinungen hatten, wie die Software weiterentwickelt werden sollte. Etwas Ähnliches ist übrigens gerade bei *LibreELEC* geschehen: Einige Entwickler haben sich abgesetzt und bieten nun *CoreELEC* an (<http://www.coreelec.org>), eine *LibreELEC* für Amlogic Embedded Boards (S905- und S912-Prozessor).



Abbildung 6.2: LibreELEC – Just enough OS for KODI (Quelle: koderds.net).

In diesem Abschnitt zeige ich Ihnen, wie Sie *LibreELEC* auf einer Speicherkarte installieren, wie Sie eine Fernbedienung anlernen können und *LibreELEC* für den Fernsehempfang klar machen. Am Ende werfen wir noch einen Blick auf einige Addons, wie beispielsweise die ARD-Mediathek oder Amazon Prime Video. Das Abspielen von Blurays darf natürlich auch nicht fehlen. *LibreELEC* ist also eine perfekte Ergänzung für einen Fernseher, der (noch) nicht Multimedia-tauglich ist.

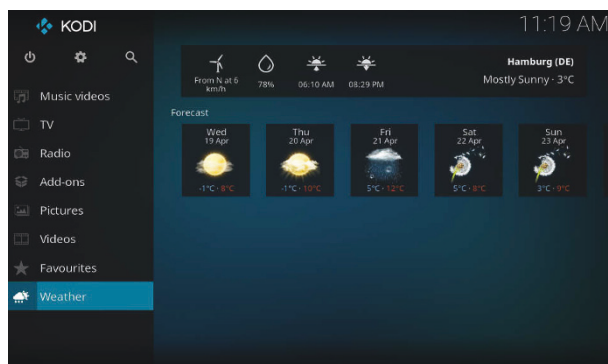


Abbildung 6.3: Estuary ist der Standard-Skin für *KODI* (Quelle: kodi.wiki).

Anschließend werden sie mit einem Startbildschirm begrüßt. Um hier weiterzukommen, benötigen Sie auf jeden Fall eine USB-Tastatur und/oder eine USB-Maus. Falls Sie beides noch nicht angeschlossen haben, holen Sie das jetzt bitte nach.

Das System zeigt Ihnen zunächst einen Startbildschirm, den Sie mit einem Klick auf *OK* wegklicken können. Anschließend wird Ihnen entweder Ihre kabelgebundene Internetverbindung angezeigt oder Sie haben Sie Möglichkeit, eine Internetverbindung per WLAN festzulegen. Wählen Sie hierzu bitte das Netzwerk aus und geben Sie gegebenenfalls das Kennwort ein. In der nun folgenden Einstellung können Sie den *ssh*-Zugang und auch *samba* aktivieren.

In der Download-Rubrik finden Sie das aktuelle Raspberry Pi 2/3-Image. Es ist ca. 130 MB groß. Bitte laden Sie es herunter und schreiben Sie es auf eine SD-Karte (Kapitel 1.2.2). Es gibt ebenfalls ein Image für den älteren Raspberry Pi 1 oder den Raspberry Pi Zero. Booten Sie anschließend Ihren Raspberry Pi mit der beschriebenen SD-Karte. Beim ersten Start wird das Dateisystem automatisch expandiert, damit die komplette SD-Karte als Speicher zur Verfügung steht. Der Raspberry Pi führt danach einen Neustart durch, der Ihnen das Logo aus der Abbildung 6.2

Bitte erlauben Sie beides, wir werden das noch im Laufe dieses Kapitels benötigen. Nach diesen Vorbereitungen begrüßt Sie *LibreELEC* zuerst einmal mit dem Skin *Estuary*, das noch auf Englisch eingestellt ist. Wir stellen *LibreELEC* nun auf Deutsch um. Ich habe die besten Erfahrungen mit einer 50 Hz Bildwiederholfrequenz gemacht, da diese Wiederholrate kompatibel mit der europäischen Fernsehnorm ist (50p, 50i). Die Änderung von 60 Hz auf 50 Hz können Sie unter *Einstellungen* (Settings) → *System* → *Video* vornehmen. Im selben Menü können Sie unter *Audio* die Option *Passthrough* wählen, falls Sie den HDMI Ausgang Ihres Raspberry Pi oder TV mit einem HDMI-Receiver verbunden haben. Stellen Sie dann bitte unter *Einstellungen* (Settings) → *Regional Language* die Sprache auf „Deutsch“ um und wählen Sie im Punkt *Standardformat für Region* „Deutschland“ aus. Für eine deutsche Tastatur benötigen Sie die Einstellung *Tastaturbelegung* „German QWERTZ“. Für die *Zeitzone* wählen Sie bitte „Germany“.

Info Mit der linken Maustaste können Sie Einstellungen bestätigen. Ein Klick auf die rechte Maustaste bringt Sie eine Menüebene zurück. Falls Ihnen die Benutzung von Maus und Tastatur zu aufwendig ist, lernen Sie bald, wie man *LibreELEC* mit einer Fernbedienung steuern kann.

Falls Sie Bluetooth verwenden möchten, etwa um Ihr Mediacenter mit einem kabellosen Kopfhörer zu verbinden, aktivieren Sie bitte den Bluetooth-Dienst in der Rubrik *Einstellungen* → *Libreelec* → *Dienste*.

Info Um sich per *ssh* mit *LibreELEC* zu verbinden, benutzen Sie bitte den Benutzernamen *root* und das Kennwort *libreelec*. Beides kann nicht verändert werden. Sollte Ihr *LibreELEC*-Rechner frei zugänglich im Internet sein, empfehle ich Ihnen, diesen Zugang nur zu aktivieren, wenn Sie Änderungen an den Einstellungen vornehmen möchten, etwas um eine Fernbedienung anzulernen.

Aktivieren Sie bitte ebenfalls den Dienst *Lirc*. Wir benötigen diesen Dienst, damit eine Fernbedienung einwandfrei funktioniert. Im späteren Verlauf dieses Kapitels werden wir noch den Fernsehempfang per Satellit für *LibreELEC* nachrüsten. Hierzu verwende ich genau wie im vorangegangenen Kapitel die DVBSKY S960 USB-Karte. Diese wird unter *LibreELEC* sofort erkannt und die benötigte Firmware steht ebenfalls zur Verfügung. Sollten Sie den Sundtek-Stick benutzen, können Sie die Treiber hierfür aus einem der *LibreELEC* Repositories nachinstallieren.

6.4.1 Einrichten der Fernbedienung

Um einen GPIO IR-Empfänger für *Lirc* zu aktivieren, sind folgende Schritte erforderlich:

Code-Ausschnitt 6.4.1

```
1 mount -o remount,rw /flash
2 vi /flash/config
3 append device_tree_overlay=gpio-ir,gpio_pin=18
4 mount -o remount,ro /flash
```

Zunächst sorgen wir dafür, dass die Partition */flash* schreibbar gemounted wird. Danach editieren wir die Datei */flash/config* und sorgen dafür, dass der Kernel den *Lirc*-Treiber lädt und dem Empfänger den korrekten GPIO-Pin zuweist (in meinem Fall den Pin 18). Anschließend wird die */flash*-Partition wieder nur lesbar gemacht.

Info Der alte Treiber *lirc-rpi* wurde durch einen neuen Treiber mit dem Namen *gpio-ir* ersetzt.

Die Aktivierung des Infrarot-Treibers erfordert einen Neustart. Nach dem Neustart können Sie mit Hilfe der Konsole überprüfen, ob das Laden des Infrarot-Moduls funktioniert hat:

Code-Ausschnitt 6.4.2

```
1 dmesg | grep gpio
```

Bei mir sieht die Ausgabe wie folgt aus:

Code-Ausschnitt 6.4.3

```
1 rc rc0: gpio_ir_recv as /devices/platform/ir-receiver/rc/rc0
2 rc rc0: lirc_dev: driver ir-lirc-codec (gpio-rc-recv) registered at minor = 0
```

Um sicher zu stellen, dass Ihr Infrarot-Empfänger richtig angeschlossen ist, können Sie den folgenden Befehl in ein Terminal eingeben und danach Tasten auf Ihrer Fernbedienung drücken. Sie sollten dann im Terminal Ausgaben bei jedem Tastendruck sehen:

Code-Ausschnitt 6.4.4

```
1 killall lircd && mode2 -d /dev/lirv0
```

Wechseln Sie nun bitte in das Verzeichnis `/storage/.config`. Bitte beachten Sie den `.` vor `config`. Hierbei handelt es sich um ein verstecktes Verzeichnis, das unter anderem die Datei enthalten kann, die Codes für die jetzt anzulernende Fernbedienung enthält. Danach starten wir das Anlernen der Fernbedienung. Dieses Prozedere kennen Sie bereits aus dem *VDR*-Kapitel.

Code-Ausschnitt 6.4.5

```
1 cd /storage/.config
2 irrecord -d /dev/lirc0
```



Bitte lernen Sie mindestens die folgenden Tasten an, um *KODI* sinnvoll bedienen zu können:

- KEY_UP
- KEY_DOWN
- KEY_LEFT
- KEY_RIGHT
- KEY_PLAY (mit dieser Taste können Sie ein Video starten und auch wieder stoppen)
- KEY_PAUSE (diese Taste ist Optional, da KEY_PLAY dieselbe Aufgabe erfüllen kann)
- KEY_STOP
- KEY_VOLUME_UP
- KEY_VOLUME_DOWN
- KEY_INFO
- KEY_OK
- KEY_EPG (mit dieser Taste können Sie später die elektronische Programmzeitschrift anzeigen)
- KEY_EXIT (das ist die „zurück“-Taste, mit der Sie eine Menüebene zurück kommen)
- KEY_MENU (mit dieser Taste können Sie verschiedene Kontext-Menüs öffnen)
- KEY_x (setzen Sie bitte für *x* die Zahlen 0...9 ein. Die Zahlen-Tasten ermöglichen es Ihnen später, ein TV-Programm direkt auszuwählen)

Nach dem Anlernen der Fernbedienung existiert im Verzeichnis `/storage/.config` eine Datei namens `name.lircd.conf`. *name* ist dabei der Name Ihrer Fernbedienung, den Sie eingegeben haben, bevor Sie die Tasten Ihrer Fernbedienung angelesen haben. Damit der Infrarot-Dienst die Daten Ihrer Fernbedienung findet, muss diese Datei nach `lircd.conf` umbenannt werden:

Code-Ausschnitt 6.4.6

```
1 mv name.lircd.conf lircd.conf
```

Bitte denken Sie daran, *name* entsprechend zu ersetzen. Nach einem Neustart sollte Ihre Fernbedienung funktionieren und Sie können Tastatur und Maus entfernen.



Bei mir hat das Programm *irrecord* zum Anlernen der Fernbedienung nicht nur den eigentlichen Tastaturcode aufgezeichnet, sondern danach noch einen weiteren Null-Code:

Code-Ausschnitt 6.4.7

```
1 ...
2 KEY_UP 0x00FB58A7 0x00000000
3 ...
```

Meine Fernbedienung funktionierte erst, nachdem ich den Code 0x00000000 hinter allen KEYS entfernt hatte.

6.4.2 Der Wetterdienst

Lassen Sie uns noch schnell ein erstes Plugin installieren, bevor wir uns gleich um Fernsehen in *LibreELEC* kümmern. Keine Sorge, in *LibreELEC* ist dazu kein aufwendiges Kompilieren erforderlich. Addons können einfach per Fernbedienung installiert werden.

Navigieren Sie im *LibreELEC*-Hauptmenü zur Rubrik „Wetter“ und bestätigen Sie Ihre Auswahl mit *OK*. Da noch kein Wetterdienst installiert ist, können Sie sich einen aussuchen (→ *mehr*). Bei mir funktioniert der *Yahoo!*-Dienst einwandfrei. Nach der Installation des erforderlichen Plugins können Sie mit der Fernbedienung den Namen der nächsten Großstadt eingeben, beispielsweise „Düsseldorf“. Das Plugin sucht daraufhin nach diesem Namen in einer Datenbank und stellt Ihnen anschließend die Wettervorhersage für diese oder auch weitere Städte zur Verfügung.

6.4.3 Tvheadend zum Fernsehen

Was der *VDR* unter LINUX leistet, das kann *Tvheadend* unter *KODI*. *Tvheadend* ist ein TV-Klient, der beispielsweise mit DVB-S2 USB-Karten umgehen kann und gleichzeitig ein komfortables Webinterface zur Konfiguration oder zum Streaming zur Verfügung stellt.

Während die Installation von *Tvheadend* schnell erledigt ist, dauert das Einrichten deutlich länger. Zur Installation des HTSP-Klienten wählen Sie bitte *Addon* → *Aus repository installieren* → *Alle Repositories* → *PVR clients* → *Tvheadend HTSP Client*. HTSP (*Home TV Streaming Protokoll*) ist ein TCP-basiertes Protokoll, welches darauf ausgelegt ist, Live-Streams (wie z. B. TV) zu übertragen. Installieren Sie bitte analog den *tvheadend*-Server, den Sie unter *Addon* → *Aus repository installieren* → *Alle Repositories* → *Dienste* finden. Damit ist die Installation von *tvheadend* auch schon erlegt und wir können uns um die Konfiguration kümmern. Diese erledigen wir bequem von einem Webbrowser Ihrer Wahl aus, der sich im selben Netzwerk befindet, in dem sich auch Ihr Raspberry Pi befindet. Der HTSP-Server steht am Port 9981 zur Verfügung. Starten Sie also bitte Ihren Browser mit der URL `http://192.168.178.66:9981` und ersetzen Sie meine oben angegebene IP-Adresse mit der Ihres Raspberry Pi. *Tvheadend* begrüßt Sie daraufhin mit einem Konfigurationsbildschirm, der in Abbildung 6.4 gezeigt wird. Ich gehe an dieser Stelle davon aus, dass Sie einen USB-Stick an Ihren Raspberry Pi angeschlossen haben, der Ihnen den Empfang von Satellitenfernsehen ermöglicht. Für kabelgebundenen (DVB-C) oder terrestrischen Empfang gilt das hier Gesagte analog.

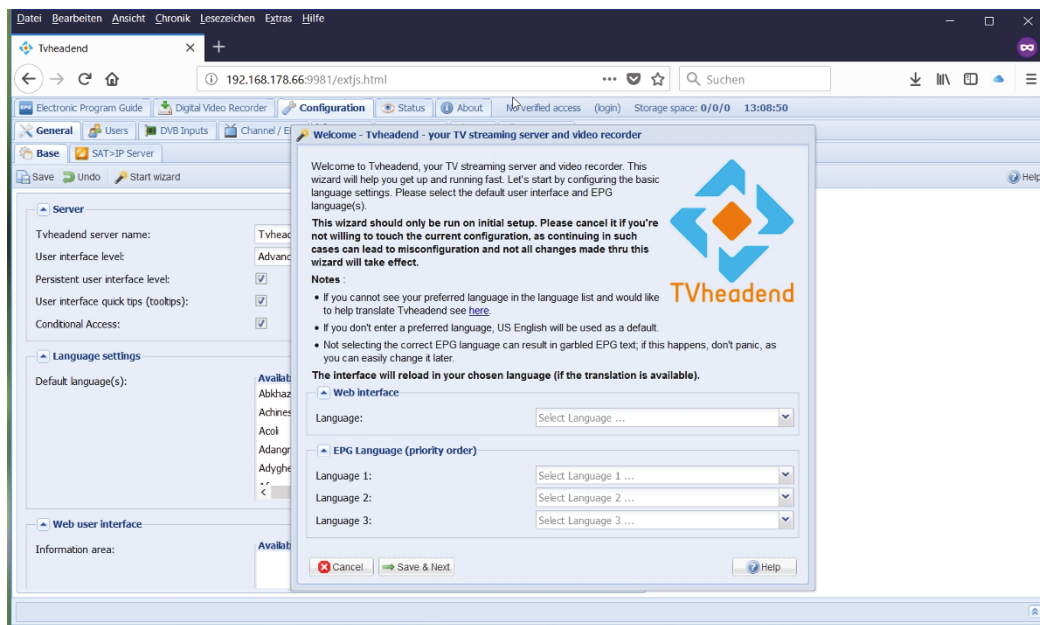


Abbildung 6.4: *tvheadend* wird komfortabel mit Hilfe eines Web-Interfaces konfiguriert.

Die Sprache ist zunächst auf Englisch voreingestellt. Ändern Sie die Sprache, indem Sie unter *Language* „German“ auswählen. Dasselbe gilt für die Sprache des elektronischen Programmführers (*EPG-Language*). Klicken Sie auf *Save and exit*.

In der nächsten Rubrik werden Sie dazu aufgefordert, Netzwerkbeschränkungen festzulegen sowie einen *Administrator-Namen* und ein *Administrator-Kennwort* festzulegen. Dasselbe gilt für einen Benutzer. Lassen Sie das Feld für die Netzwerkeinstellungen bitte frei. Damit gibt es keinerlei Restriktionen und das Risiko ist geringer, hier etwas Falsches einzustellen. Sie können diese Einstellungen jederzeit ändern, falls Ihr Raspberry Pi im Internet hängt und Sie das Risiko unberechtigten Zugriffs verringern möchten. Für alle Benutzerkonten und die Kennwörter tragen Sie bitte einen * ein. Damit ist auch die Web-Oberfläche zur Konfiguration frei zugänglich. Speichern Sie diese Änderungen ebenfalls.

Im folgenden Schritt wählen Sie bitte unter *Netzwerktyp* das DVB-S Netzwerk aus und stellen Sie als vordefinierte Muxe 19.2E: Astra ein, falls Ihr Satellit auf 19.2° Ost (Astra) ausgerichtet ist. Nach einem erneuten *Speichern und weiter* beginnt *tvheadend* damit, Sender zu scannen. Erfolgreiches Scannen können Sie daran erkennen, dass die Anzahl der gefundenen Muxe und die Anzahl der gefundenen Dienste (Services) steigt. Bitte bringen Sie Geduld mit: Ein vollständiger Scan des Astra-Satelliten kann gerne schon einmal 30 Minuten dauern.

Nach erfolgreichem Scan setzen Sie bitte alle drei Häkchen im nächsten Konfigurations-Bildschirm. Das ordnet alle Dienste (Services) zu und setzt sowohl Anbieter- als auch Netzwerk-Tags. Wenn Sie nun in *KODI* die Rubrik *TV* auswählen, werden Ihnen alle gefundenen Sender angeboten und Sie können das Fernsehprogramm mit einem Fernbedienungs-OK starten.



Bitte beachten Sie, dass für Sender mit DD5.1-Sound Audio-Passthrough in *KODI* aktiviert sein muss.



Die *KODI*-Version innerhalb der *LibreELEC*-Distribution benötigt keinen MPEG2-Codec (Kapitel 1.4) zum Abspielen von SD-Fernsehen. Ein Raspberry Pi 3B+ ist leistungsstark genug, die Umwandlung in Software durchzuführen. Das merkt man allerdings auch an der CPU-Last. Während ein HD-Sender lediglich 25% CPU-Last benötigt, sind es bei SD-Inhalt gerne schon einmal 65%. Den Versuch, UHD-TV wiederzugeben (z. B. den Sender Fashion 4K) quitierte *LibreELEC* mit einem Absturz der *KODI*-Software, gefolgt von einem Neustart.

Wenn Sie an der Wiedergabe von UHD-Inhalten interessiert sind, kann ich Ihnen die Mecool K3 Pro TV-Box empfehlen. Für knapp 100 € erhalten Sie damit ein System, welches nicht nur DVB-S2/C/T2-Tuner eingebaut hat, sondern auch noch echtes GBit-Netzwerk bietet und einen SPDIF-Ausgang zur Verfügung stellt. Die aktuelle *CoreELEC*-Distribution läuft darauf ohne Probleme.

Während wir nun schon komfortabel TV schauen, Timer setzen und Sendungen aufzeichnen können, sieht das TV-Sendermenü noch ein wenig triste aus. Dies ändern wir durch Installation von Senderlogos. Die Webseite www.picons.eu bietet aktuelle Senderlogos zum kostenlosen Download an.

Wir benötigen zwei verschiedene Dateien aus der Download-Rubrik, nämlich *srp-full400x240-370x210.light.on.transparent...hardlink.tar.xz* und das entsprechende *snp*-Pendant. Die heruntergeladenen Dateien müssen in den */storage/picons*-Ordner kopiert werden. Die *snp*-Datei muss in das Unterverzeichnis *tvh* kopiert werden, die *srp*-Datei in das Unterverzeichnis *vdr*. Sie können das z. B. bequem von einem Laptop aus unter der Verwendung von *samba* erledigen (Abbildung 6.5). Entpacken Sie die beiden Logo-Dateien in einem Terminal:

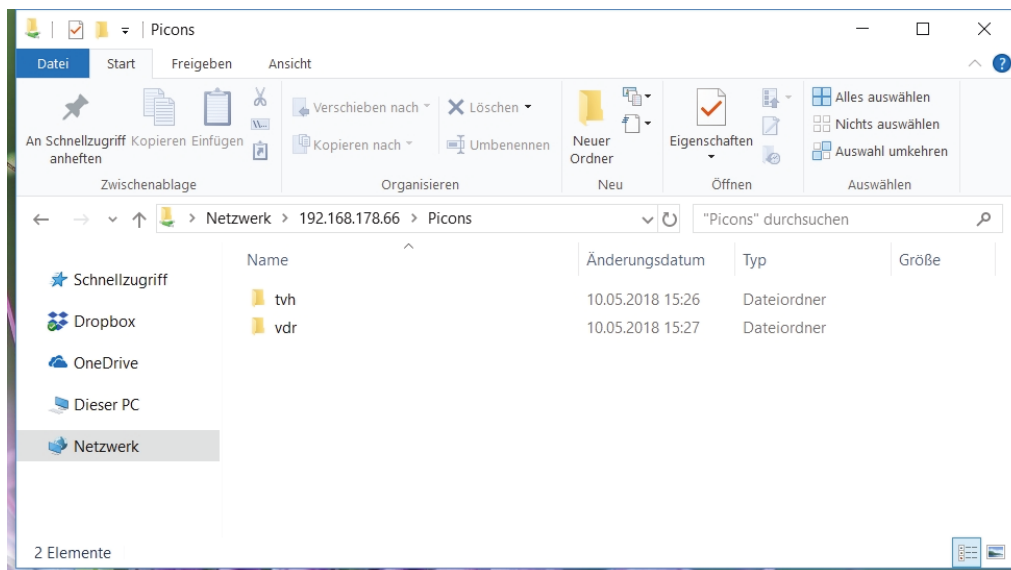


Abbildung 6.5: Mit Hilfe von *samba* können die Logo-Dateien schnell an ihren Bestimmungsort kopiert werden.

Code-Ausschnitt 6.4.8

```

1 cd /storage/picons/tvh
2 xz -d snp<TAB>
3 tar xvf snp<TAB>
4 mv snp<TAB>/* .
5 rmdir snp<TAB>
6 rm snp<TAB>
7 cd /storage/picons/vdr
8 xz -d srp<TAB>
9 tar xvf srp<TAB>
10 mv srp<TAB>/* .
11 rmdir srp<TAB>
12 rm srp<TAB>

```

Das `xz -d`-Kommando entpackt die jeweilige Logo-Datei. Der `tar`-Befehl entpackt die einzelnen Logos, allerdings in einem weiteren Unterverzeichnis. Wir bewegen (`mv`) die Dateien aus dem Unterverzeichnis wieder in das übergeordnete Verzeichnis und löschen anschließend das leere Unterverzeichnis und die gepackte Logo-Datei. Die `TAB` ergänzt den jeweiligen Dateinamen automatisch und erspart so viel Tipparbeit. Die Verzeichnisse, in welche die jeweiligen Logos kopiert werden müssen, können Sie übrigens auch vom *tvheadend* Web-Frontend einstellen (Abb. 6.6).

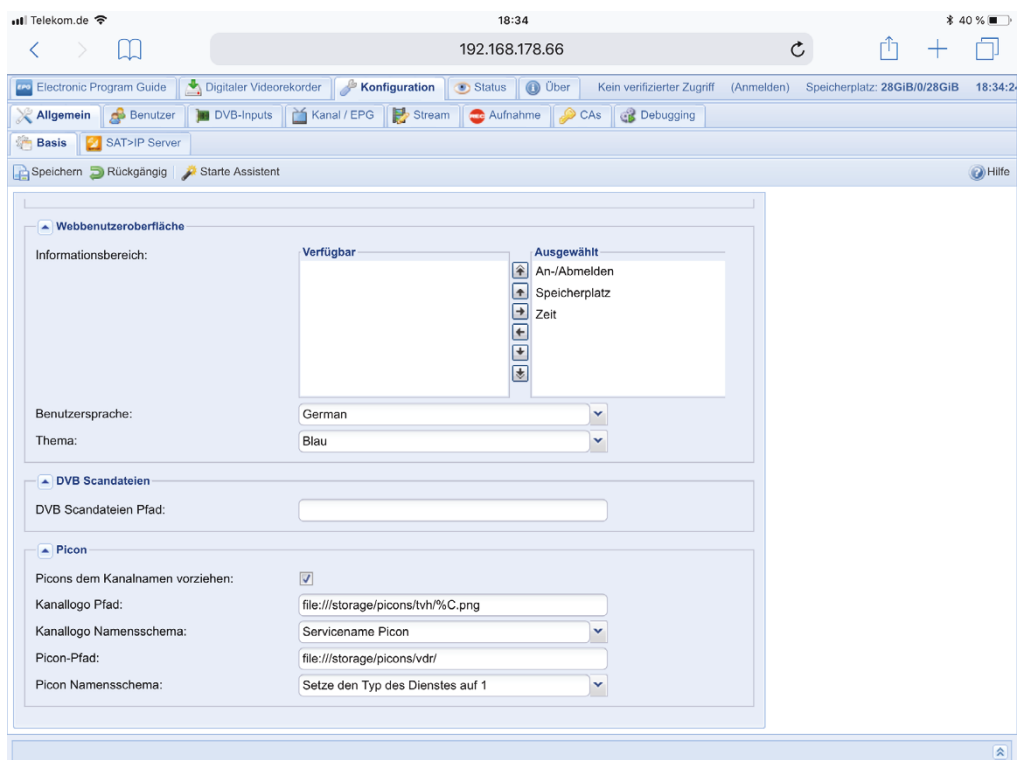


Abbildung 6.6: Die Verzeichnisse für die Senderlogos können im *tvheadend* Web-Frontend eingestellt werden.

Im letzten Schritt teilen wir *KODI* noch mit, dass wir in den o.g. Ordnern neue Kanallogos haben. Dazu wechseln wir vom *KODI*-Hauptmenü zu den Einstellungen und wählen *PVR & TV* → *Menü/OSD*. Als *Ordner mit Kanalsymbolen* geben wir das Verzeichnis `/storage/picons/tvh` an und starten anschließend *Nach fehlenden Kanalsymbolen suchen*. Danach sind die Senderlogos in den Kanälen und in der elektronischen Programmzeitschrift (EPG) enthalten.

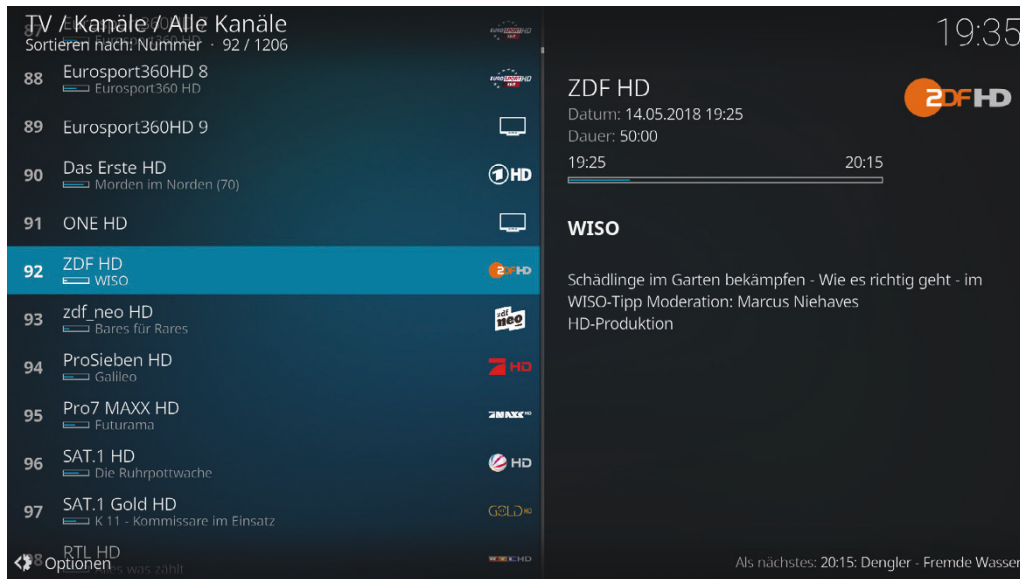


Abbildung 6.7: Mit Senderlogos sieht *tvheadend* gleich viel schöner aus.

6.4.4 DVD und Blu-ray



LibreELEC kann von Hause aus DVDs abspielen. Die *libdvdcss* ist also bereits eingebaut. Analog zur *libdvdcss* gibt es auch eine *libblueray*, für die ähnliche rechtliche Bedingungen gelten. Laut <https://wiki.libreelec.tv/blu-ray> ist die Verwendung in den meisten Ländern legal.

Der Befehl

Code-Ausschnitt 6.4.9

```
1 mkdir -p /storage/.config/aacs && curl https://web.archive.org/web/20170902140455if_/↵
  http://www.labdv.com/aacs/KEYDB.cfg -o /storage/.config/aacs/KEYDB.cfg
```

lädt eine Datenbank mit über 24.000 Blu-ray Discs auf Ihren Pi, die *KODI* danach problemlos abspielen kann.



Der oben genannte Befehl sollte von Zeit zu Zeit wiederholt werden, da ständig neue Blu-rays hinzugefügt werden.

Damit mein USB Blu-ray-Laufwerk richtig erkannt wurde, musste ich *LibreELEC* mit angestecktem Laufwerk neu starten. Danach konnte ich ohne Probleme und ruckelfrei eine Blu-ray Disc abspielen.

6.4.5 Amazon Prime

KODI bietet eine Fülle an Addons, wie verschiedene Mediatheken oder Internet-Radios. Ein sehr beliebtes ist das Amazon Prime VOD (*Video On Demand*) Addon. Amazon selbst unterstützt das Abspielen von Videos mit Hilfe von KODI nicht. Aus diesem Grund geschieht es immer wieder, dass dieses Plugin zu bestimmten Zeiten funktioniert, nach einer Änderung von Amazon aber wieder nicht. Das Plugin selbst erfordert die *libwidevine*, eine Bibliothek, die DRM-geschützten (*Digital Rights Management*) Inhalt abspielen kann. Da Amazon keine Hardware-Beschleunigung beim Abspielen der angebotenen Inhalte unterstützt, können Filme auf dem Raspberry Pi häufig nur in kleiner Auflösung abgespielt werden, damit sie ruckelfrei sind. Die Autoren von *LibreELEC* erlauben es nicht, diese Bibliothek auf Ihrem Server zu speichern oder zur Verfügung zu stellen. Das Amazon Prime Addon, welches wir gleich installieren, setzt das *Videoplayer Inputstream* Addon voraus. Installieren Sie dieses bitte zuerst, indem Sie wie folgt navigieren: *Addons* → *Aus Repository installieren* → *Alle Repositories* → *Videoplayer Inputstream Addons* → *Inputstream Adaptive*. Installieren Sie dieses Addon bitte, nachdem Sie es ausgewählt haben. Das *Amazon VOD* Addon liegt in einem Repository namens „Sandmann 79s Repository“. Dieses ist standardmäßig noch nicht aktiviert. Aktivieren Sie dieses Repository, indem sie zu *Addons* → *Aus Repository installieren* → *Alle Repositories* → *Addon Verzeichnis* → *Sandmann 79s Repository* navigieren und das Sandmann-Repository installieren. Anschließend steht Ihnen das *Amazon VOD* Addon unter *Video Addons* → *Amazon VOD* zur Verfügung.



Abbildung 6.8: Amazon VOD.

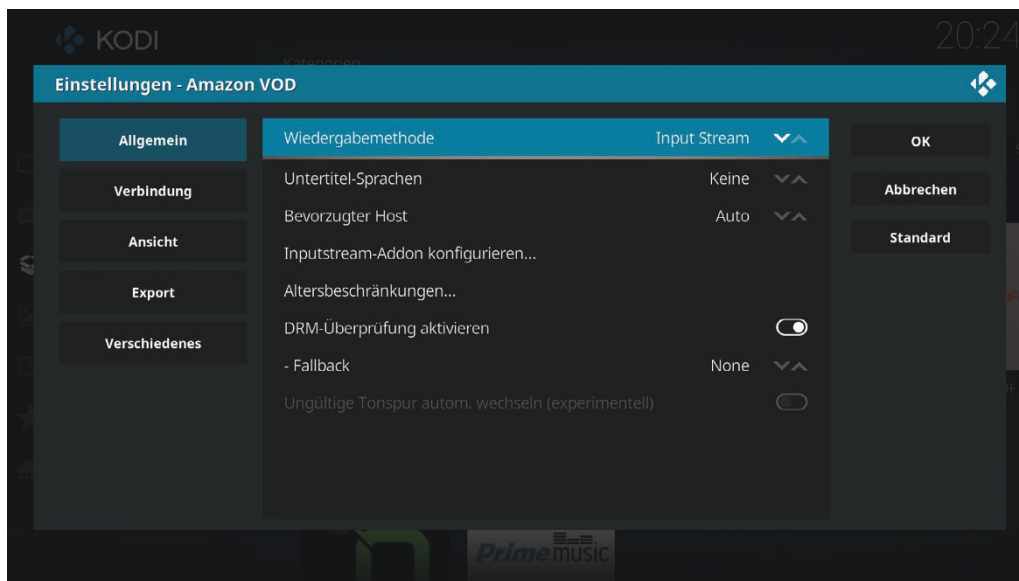


Abbildung 6.9: Als Wiedergabemethode im *Amazon VOD* Addon muss *Input Stream* gewählt werden.

Konfigurieren Sie das *Amazon VOD* Addon, indem Sie es selektieren und die *Menü*-Taste Ihrer Fernbedienung drücken. Dort können Sie *Einstellungen* auswählen. Wählen Sie als *Wiedergabemethode* *Input Stream* aus. Unter der Rubrik *Einstellungen Verbindung* können Sie unter *Anmelden* Ihren Amazon-Benutzernamen und Ihr Kennwort eingeben.

Jetzt ist der meiste Teil der Arbeit geschafft. Es fehlt nur noch das Herunterladen der *libwideovine*. Das erledigt das *Amazon VOD* Addon automatisch für uns bei der Auswahl des ersten Filmes. Starten Sie nun das Addon durch Drücken der *OK*-Taste, nachdem Sie es selektiert haben. Navigieren Sie danach zu einem beliebigen Film und starten Sie ihn ebenfalls durch Drücken der *OK*-Taste Ihrer Fernbedienung. Das Addon wird den Film nicht starten, sondern stattdessen die Meldung „Wideovine CDM ist required“ ausgeben. Bitte bestätigen Sie, dass Sie die Bibliothek installieren möchten und akzeptieren sie die Lizenzbedingungen. Das Addon lädt daraufhin ein Google Chrome Recovery Image aus dem Internet herunter, welches die *libwideovine* enthält. Dieses Recovery Image ist 1,9 GB groß, es kann also eine Weile dauern, bis es vollständig geladen wurde. Danach müssen Sie noch einmal das Entpacken des Images mit „OK“ bestätigen.

Nach diesen Schritten sollte das Abspielen von Filmen funktionieren.

Sollte das Amazon Prime Addon nicht funktionieren, kann es sich lohnen, einen Blick auf die nächste *LibreELEC*-Version 9 zu werfen. Diese können Sie einfach über eine bestehende Version 8.x installieren. Öffnen Sie bitte dazu die Einstellungen von *LibreELEC* und stellen Sie im Bereich „Aktualisierungen“ folgendes ein:

- Stellen Sie die automatischen Aktualisierungen auf „manuell“.
- Stellen Sie den Eintrag „Benutzerdefinierte Kanäle anzeigen“ auf „Ja“.
- Wählen Sie als benutzerdefinierten Kanal 1 die URL `http://milhouse.libreelec.tv/builds/master/RPi2`. Achten Sie dabei bitte unbedingt auf Groß- und Kleinschreibung.
- Wählen Sie als Update-Kanal „Milhouse-9.0“.

Unter „Verfügbare Versionen“ können Sie nun die neueste Version der Milhouse-Builds installieren. Diese sind einfach durchnummeriert. Die höchste Nummer entspricht dabei der aktuellsten Version. Nach der Auswahl wird das Update heruntergeladen und der Raspberry Pi startet mehrmals neu. Nach dieser Aktualisierung muss die neueste Version der *wideovine*-Bibliothek geladen werden. Öffnen Sie ein Terminal für Ihre *LibreELEC*-Distribution und loggen Sie sich mit dem Benutzernamen *root* und dem Kennwort *libreelec* ein. Danach geben Sie bitte folgenden Befehl ein:

Code-Ausschnitt 6.4.10

```
1 curl -Ls http://nmacleod.com/public/libreelec/getwideovine.sh | bash
```

Das Skript lädt nun eine passende Version der *libwideovine* und installiert sie im passenden Ordner.

Einen Haken gibt es allerdings mit der Version 9 der *LibreELEC*-Distribution. Die Fernbedienung funktioniert nicht mehr so wie erwartet. Bei mir äußerte sich das darin, dass der Fernbedienungs-Empfänger per *lirc* nur noch einzelne Kommandos entgegennahm, aber keine Tasten-Wiederholungen mehr registrierte. Das ist nervig, wenn man mal eben schnell in einer Kanalliste scrollen möchte.

Um dem neuen Handling für die Fernbedienung gerecht zu werden, gehen Sie bitte wie folgt vor: Benennen Sie im ersten Schritt die Konfigurations-Datei für *lirc* um und beenden Sie dann *KODI* und den *lirc*-Daemon:

Code-Ausschnitt 6.4.11

```

1 mv /storage/.config/lircd.conf /storage/.config/lircd.conf.alt
2 systemctl stop kodi
3 systemctl stop eventlircd

```

Erzeugen Sie eine leere Datei namens *my_remote* im Verzeichnis */storage/.config/rc_keymaps/*:

Code-Ausschnitt 6.4.12

```

1 vi /storage/.config/rc_keymaps/my_remote
2 systemctl stop kodi
3 systemctl stop eventlircd

```

Öffnen Sie bitte ein zweites Terminal, ohne das andere zu schließen. Wir müssen nun herausfinden, welches Protokoll die Fernbedienung verwendet, die wir benutzen wollen. Rufen Sie dazu

Code-Ausschnitt 6.4.13

```

1 ir-keytable

```

auf. Der Befehl listet alle verfügbaren Protokolle in der dritten Zeile auf:

Code-Ausschnitt 6.4.14

```

1 Found /sys/class/rc/rc0/ (/dev/input/event1) with:
2   Driver gpio-rc-recv, table rc-rc6-mce
3   Supported protocols: lirc rc-5 rc-5-sz jvc sony nec sanyo mce_kbd rc-6 sharp xmp
4   Enabled protocols: lirc nec rc-6
5   Name: gpio_ir_recv
6   bus: 25, vendor/product: 0001:0001, version: 0x0100
7   Repeat delay = 500 ms, repeat period = 125 ms

```

Um das passende Protokoll herauszufinden, wird der folgende Befehl abgesetzt:

Code-Ausschnitt 6.4.15

```

1 ir-keytable -p rc-5 -t

```

Der Eintrag hinter *-p* entscheidet über das Protokoll. Drücken Sie einige Tasten Ihrer Fernbedienung und achten Sie auf mögliche Ausgaben. Stimmt das Protokoll nicht, werden Sie keine Ausgabe erhalten. Beenden Sie in diesem Fall das Programm *ir-keytable* mit der Tastenkombination *CTRL-c* und wählen Sie ein anderes Protokoll. Haben Sie das passende Protokoll erwischt, erscheinen folgende Ausgaben:

Code-Ausschnitt 6.4.16

```

1 Protocols changed to rc-5
2 Testing events. Please, press CTRL-C to abort.
3 1503592437.660155: event type EV_MSC(0x04): scancode = 0x101a
4 1503592437.660155: event type EV_SYN(0x00).
5 1503592437.774129: event type EV_MSC(0x04): scancode = 0x101a
6 1503592437.774129: event type EV_SYN(0x00).
7 1503592437.921009: event type EV_MSC(0x04): scancode = 0x101a

```

Füllen Sie (im anderen Terminal) die Datei */storage/.config/rc_keymaps/my_remote* mit der ersten Zeile:

Code-Ausschnitt 6.4.17

```
1 # table my_custom_remote, type: rc-5
```

Bitte denken Sie daran, Ihr gefundenes Protokoll einzusetzen, das nicht *rc-5* sein muss. Im nächsten Schritt müssen wir herausfinden, welche Scancodes mit welchen Fernbedienungstasten verbunden sind. Lassen Sie den Editor, den Sie gerade mit einer Protokollzeile gefüllt haben, geöffnet und geben Sie in Ihrem zweiten Terminal den Befehl

Code-Ausschnitt 6.4.18

```
1 ir-keytable -t
```

ein. Drücken Sie nun beliebige Tasten auf Ihrer Fernbedienung und tragen Sie den Scancode und eine passende *KEY*-Bezeichnung in den immer noch geöffneten Editor, beispielsweise so:

Code-Ausschnitt 6.4.19

```
1 # table my_custom_remote, type: rc-5
3 0x101a KEY_UP
4 0x101b KEY_DOWN
5 0x1013 KEY_LEFT
6 0x1014 KEY_RIGHT
7 0x1015 KEY_OK
```

Scancode und *KEY*-Name müssen dabei durch ein einzelnes Leerzeichen voneinander getrennt sein. Noch einmal zur Erinnerung: Eine Liste gültiger *KEY*-Bezeichnungen erhalten Sie durch Eingabe des Befehls

Code-Ausschnitt 6.4.20

```
1 irrecord -l | grep ^KEY
```

Diese Keys werden in der Datei */usr/share/kodi/system/Lircmap.xml* verwendet. Sollten Sie andere *KEY*-Namen verwenden, müssen Sie die entsprechende Rubrik *<remote device="devinput">* durch eine eigene *Lircmap.xml*-Datei abändern.

Nachdem Sie alle Scancodes herausgefunden und eingetragen haben, können Sie die Datei */storage/.config/rc_keymaps/my_remote* speichern. Abschließend erstellen Sie bitte eine Datei namens */storage/.config/rc_maps.cfg* mit dem folgenden Inhalt:

Code-Ausschnitt 6.4.21

```
1 * * my_remote
```

Nach einem Neustart sollte Ihre Fernbedienung dann wie gewohnt auch unter der neuen *LibreELEC*-Version 9 funktionieren.

6.4.6 Mobile Helfer

In diesem Abschnitt möchte ich Ihnen noch einige mobile Helfer vorstellen, die den Umgang mit *LibreELEC* unter Verwendung eines Mobiltelefons oder eines Tablet-Computers deutlich vereinfachen können. Diese Apps erlauben es, *KODI* mit dem Telefon zu bedienen, Fernsehsendungen zum Tablet zu streamen oder TV-Aufnahmen von unterwegs zu programmieren.

TvhClient

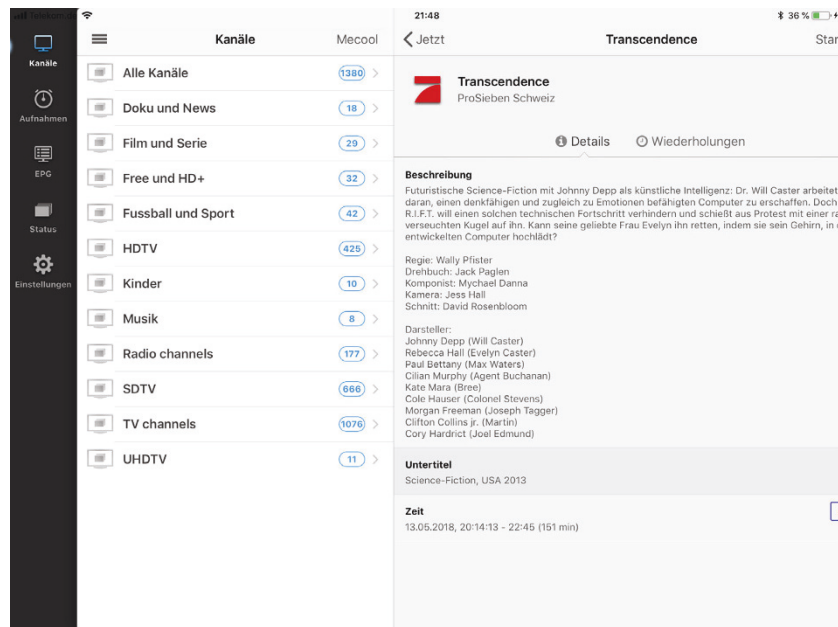


Abbildung 6.10: Die Verzeichnisse für die Senderlogos können im *tvheadend* Web-Frontend eingestellt werden.

Zu meinen Lieblings-Apps im Zusammenhang mit *KODI* und *tvheadend* zählt die iPad-APP *TvhClient*. *TvhClient* zeigt das EPG einzelner Sendungen oder das EPG über einen Zeitraum, es streamt das Fernsehprogramm oder Aufnahmen auf das iPad. Der Benutzer kann Timer einstellen, editieren und löschen. Für Android-Geräte empfehle ich Ihnen die kostenlose App *Kore*. Diese App ist eine perfekte Fernbedienung für *KODI*, welche Sie von Ihrem Mobilfunkgerät aus verwenden können. Dabei gibt es nicht nur die typischen Fernbedienungsgtasten, sondern auch noch weitere Information über aufgezeichnete Filme wie eine Inhaltsangabe oder Schauspieler. *LibreELEC* stellt auch einen AirPlay-Server zur Verfügung, so dass Sie Musik von Ihrem Apple-Gerät zu *KODI* streamen können, um diese beispielsweise über Ihre Stereo-Anlage abspielen zu können. Das nächste Kapitel zeigt Ihnen, wie Sie diesen Dienst auch ohne *LibreELEC* unter Ihrer Raspian-Distribution installieren können.

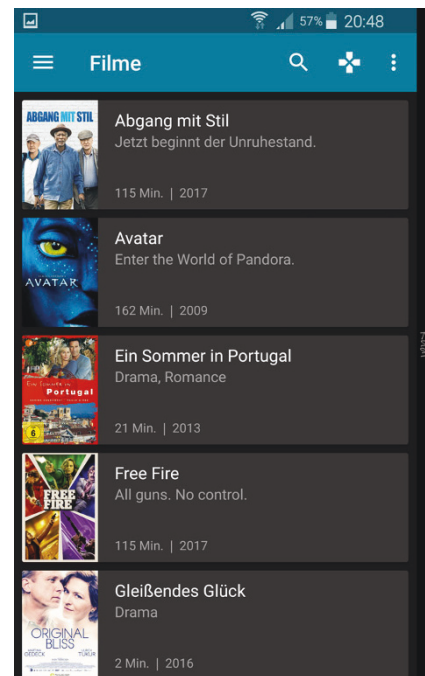


Abbildung 6.11: Die *Kore* App unter Android.

6.5 AirPlay und Co.

Ein großer Computerkonzern aus Cupertino wirbt auf der Webseite damit, dass seine Geräte *AirPlay* beherrschen, das Abspielen von Medieninhalten über die WLAN-Luft-Schnittstelle. Haben Sie das schon bei *KODI* versucht? Unter den Einstellungen können Sie *AirPlay* erlauben und so multimediale Inhalte von Ihrem Telefon oder Tablet auf dem Fernseher anzeigen -

vorausgesetzt, *KODI* läuft¹. Das Mediacenter erkennt automatisch, dass ein *AirPlay*-Inhalt angefordert wird und stellt diesen dar - unabhängig davon, ob es sich um Urlaubsfotos und -videos oder um einen Film handelt. Eine ähnliche *AirPlay*-Schnittstelle gibt es für die Audio-Ausgabe. Allerdings macht es keinen Sinn, für ein paar Songs den Fernseher einzuschalten. Und wäre es nicht schön, wenn wir Webradio über unsere Stereoanlage hören könnten, vielleicht sogar noch in digitaler Qualität? Kein Problem! In diesem Abschnitt werden wir einen *AirPlay*-Server installieren und ihn mit unserer Stereoanlage verbinden. Da die analoge Audioausgabe des Raspberry Pi nicht als audiophil bezeichnet werden kann, zeige ich Ihnen, wie Sie mit Hilfe von externer Zusatz-Hardware, etwa einer kleinen USB-Soundkarte für 12 € (Delock 61961), einen vollwertigen, digitalen Toslink-Anschluss realisieren können. Alternativ können Sie natürlich den Ton über HDMI abgreifen, wenn Ihr Audio-Receiver damit umgehen kann. Beginnen wir mit der Installation des *AirPlay*-Servers. Sicher kennen Sie bereits den ersten Schritt: die Installation weiterer Programmpakete.

Code-Ausschnitt 6.5.1

```
1 sudo apt-get install autoconf automake avahi-daemon build-essential git libasound2-dev ↵  
libavahi-client-dev libconfig-dev libdaemon-dev libpopt-dev libssl-dev libtool ↵  
xsltoman
```

Die aktuelle Version des *AirPlay*-Servers nennt sich *shairport-sync* und hat ein eigenes Github-Repository. Bitte wechseln Sie in ein Verzeichnis Ihrer Wahl (z. B. */home/pi*) und clonen Sie den *AirPlay*-Server so:

Code-Ausschnitt 6.5.2

```
1 git clone https://github.com/mikebrady/shairport-sync.git
```

Wechseln Sie anschließend in das *shairport-sync*-Verzeichnis

Code-Ausschnitt 6.5.3

```
1 cd shairport-sync
```

und starten Sie den Übersetzungsvorgang für den Server.

Code-Ausschnitt 6.5.4

```
1 autoreconf -i -f  
2 ./configure --with-alsa --with-avahi --with-ssl=openssl --with-systemd --with-metadata
```

In den beiden oberen Befehlen wird zunächst *autoreconf* ausgeführt. Dieser Befehl erstellt ein Konfigurationsskript, welches danach aufgerufen wird und das Makefile erstellt, das dann ausgeführt wird, um unseren *AirPlay*-Server zu übersetzen. Der Vorteil dieser Methode ist, dass die Konfiguration für verschiedene Systeme durchgeführt werden kann. *shairport-sync* wird mit Unterstützung für das ALSA-Soundsystem übersetzt. Weiterhin unterstützt unser Server Avahi und OpenSSL-Verschlüsselung.

Avahi ist eine freie Implementierung einer Technik zur Vernetzung von Geräten in einem lokalen Netzwerk, ohne dass diese manuell konfiguriert werden müssen. Der Parameter *--with-systemd* erlaubt den automatischen Start des *ShairPort*-Servers nach dem Bootvorgang.

¹Videos können nur bis iOS Version 8 übertragen werden. Audio-Übertragungen sind für alle iOS-Versionen möglich.

Der Parameter `--with-metadata` erlaubt es dem *ShairPort*-Server, Metadaten an kompatible Applikationen zu „pipen“. Das Übersetzen des Quelltextes und das Installieren der Applikation ist schnell erledigt:

Code-Ausschnitt 6.5.5

```
1 make
2 sudo make install
```

Damit *Shairport-Sync* nach dem Booten automatisch startet, ist ein letzter Befehl notwendig, der das Startup-Skript nach jedem Booten des Raspberry Pi automatisch ausführt.

Code-Ausschnitt 6.5.6

```
1 sudo systemctl enable shairport-sync
```

Nach einem Neustart steht der *AirPlay*-Server zur Verfügung. Möchten Sie den Raspberry Pi nicht neu starten, können Sie auch mit dem Befehl

Code-Ausschnitt 6.5.7

```
1 sudo service shairport-sync start
```

dafür sorgen, dass der Server gestartet wird. Wenn Sie nun Musik mit Ihrem Apple-Device abspielen, taucht der *AirPlay*-Server unter dem Namen „Raspberrypi“ auf.

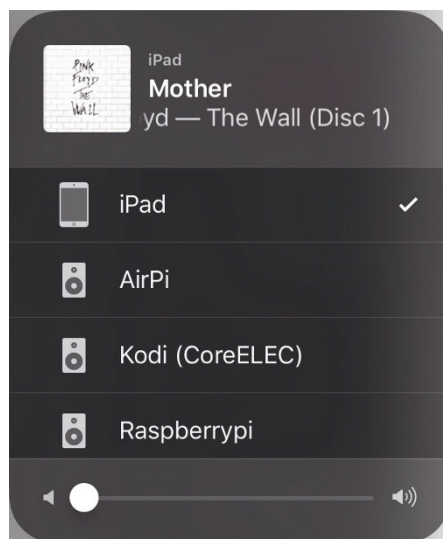


Abbildung 6.12: Auf einem Apple-Gerät taucht *shairport-sync* unter dem Namen „Raspberrypi“ auf.



Erinnern Sie sich noch daran, dass Sie die Audioausgabe mit `amixer cset numid=3 device` umschalten können?



Abbildung 6.13: Delock 61961 ist eine preiswerte USB-Soundkarte für den Raspberry Pi (Quelle: Tragant).

Während die digitale HDMI-Audioausgabe eine hervorragende Qualität hat, kann man das für den 3,5 mm analogen Klinkenausgang nicht behaupten. Abhilfe schafft eine externe Soundkarte, wie die Delock 61961, die über USB eingebunden werden kann, oder das HD-Soundsystem von Busware (http://busware.de/tiki-browse_image.php?imageId=403). Der nur 12 € teure Delock-Stick kann ohne zusätzlichen USB-Hub direkt am Raspberry Pi betrieben werden. Delock stellt einen analogen und einen optischen digitalen Ausgang zur Verfügung. Das Einbinden des Sticks ist denkbar einfach. Fügen Sie in die Datei `/etc/modules` folgende zwei Zeilen an beliebiger Stelle ein:

Code-Ausschnitt 6.5.8

```
1 snd-usb-audio
2 snd-bcm2835
```

Nach einem Neustart des Raspberry Pi lädt der Kernel zwei Treiber: den prinzipiellen USB-Audiotreiber und den `bcm2825` Audiotreiber. Die Datei `/etc/modprobe.d/alsa-base.conf`, die ebenfalls angepasst werden muss, räumt dem USB-Treiber aber Vorrang vor dem Broadcom-Treiber ein.

Code-Ausschnitt 6.5.9

```
1 #options snd-usb-audio index=-2
2 options snd-usb-audio index=0
```

Nun sollte einem ungetrübten Musikgenuss nichts mehr im Wege stehen. Auf den *i*Geräten können Sie *PiPlay* als *AirPlay*-Plattform auswählen und der Ton wird über Ihren Pi an die Stereoanlage oder ein paar Lautsprecherboxen weitergeleitet. Beide Geräte (also Tablet/Telefon und Raspberry Pi) müssen sich dabei im selben Netzwerk befinden.

6.6 Externalplayer

In diesem Abschnitt möchte ich Ihnen noch ein Plugin vorstellen, welches es auf einfache Art und Weise erlaubt, zwischen *VDR* und *KODI* hin- und herzuschalten. Das Plugin heißt *externalplayer*. Installieren Sie den Quelltext bitte im Verzeichnis `/home/pi/VDR/PLUGINS/src`, indem Sie es von Github clonen:

Code-Ausschnitt 6.6.1

```
1 cd /home/pi/VDR/PLUGINS/src
2 git clone http://git.uli-eckhardt.de/vdr-plugin-externalplayer.git externalplayer
```

Bitte übersetzen und installieren Sie das Plugin so

Code-Ausschnitt 6.6.2

```
1 cd /home/pi/VDR
2 make
3 sudo make install
```

und fügen Sie den Aufruf für dieses Plugin in der Datei `/usr/local/bin/runvdr` dazu:

Code-Ausschnitt 6.6.3

```

1 VDRPLUGINS=" -Prpvhdevice \
2           -Pweatherforecast \
3           -Pstreamdev-server \
4           -Pexternalplayer \
5           -P'skindesigner -l /var/lib/vdr/plugins/skindesigner/logos -e /video/cache/↔
           epgimages -i /var/lib/vdr/plugins/skindesigner/installerskins'"

```

Bitte legen Sie im nächsten Schritt als *root* eine Datei namens */var/lib/vdr/plugins/external-player.conf* an und füllen Sie sie mit folgendem Inhalt:

Code-Ausschnitt 6.6.4

```

1 {
2   MenuEntry = "KODI";
3   Command = "/usr/local/bin/starte_xbmc.sh";
4   OutputMode = extern;
5   InputMode = normal; # XBMC should be configured for LIRC.
6 }

```

Der Inhalt des Skripts */usr/local/bin/starte_xbmc.sh* lautet:

Code-Ausschnitt 6.6.5

```

1 #!/bin/sh
2
3 /usr/local/bin/svdrpsend REMO off
4 sudo -u pi DISPLAY=:0.0 /usr/local/src/xbmc/kodi.bin -fs --standalone
5 #sudo -u pi DISPLAY=:0.0 /usr/local/src/xbmc/kodi.bin -fs --standalone
6 #sudo -u pi DISPLAY=:0.0 /usr/lib/kodi/kodi.bin -fs --standalone
7 /usr/local/bin/svdrpsend REMO on
8 sleep 1
9 /usr/local/bin/svdrpsend REMO on

```

Bitte passen Sie gegebenenfalls die Verzeichnisse an, in denen sich Ihre *KODI*-Installation befindet. Der Befehl

Code-Ausschnitt 6.6.6

```

1 whereis kodi

```

hilft Ihnen dabei, das korrekte Verzeichnis Ihrer *KODI*-Installation zu finden. Nun können Sie innerhalb der Raspbian-Distribution komfortabel zwischen *VDR* und *KODI* hin- und herschalten.

6.7 MP3

Eines darf in einem Multimedia-Kapitel sicher nicht fehlen: das Abspielen von mp3-Dateien. Unter Raspbian ist dieses Thema schnell erledigt:

Code-Ausschnitt 6.7.1

```

1 sudo apt-get install mpg123

```

Das Programm *mpg123* wird einfach mit dem Dateinamen des abzuspielenden Liedes als Parameter aufgerufen - fertig!

6.8 Ausblick

Die Raspberry Pi Foundation sagte einmal: „Wir dachten, wir würden am Anfang 1.000 Stück verkaufen“. Inzwischen sind weltweit über 17 Millionen Raspberry Pi verkauft worden. Viele davon verrichten sicher ihren Dienst in Kombination mit einer Multimedia-Distribution wie *LibreELEC*. Dieses Kapitel hat Ihnen einen Einblick in die multimedialen Möglichkeiten des Raspberry Pi gegeben. Das nächste Kapitel beinhaltet nicht nur Software, sondern auch Hardware. Wir schauen uns den Nachbau eines Ambilight für Fernseher an. Dieses von Philips erfundene System leuchtet die Wand hinter einem Fernseher in den Farben des aktuellen Fernsehbildes aus und vergrößert damit virtuell die eigentliche Bildschirmgröße.

Zusammenfassung 6 Das war das Projektkapitel *KODI*. Der Raspberry Pi ist, obwohl er nie dafür konzipiert wurde, eine ideale Multimedia-Plattform. Sie können damit fernsehen, hochauflösende Videos schauen oder Medieninhalte streamen.

Probieren Sie einfach weitere *KODI*-Plugins aus. Machen Sie beispielsweise Telefonnummern von Anrufern mit dem Fritz!Box-Addon auf Ihrem Fernsehgerät sichtbar oder installieren sie *radio.de*, ein Webradio. Schließen Sie ein externes USB-DVD-Laufwerk an und schauen Sie Ihre DVDs mit *KODI*.

Im nächsten Kapitel zeige ich Ihnen, wie Sie das multimediale Erlebnis mit Hilfe eines Ambilight weiter steigern können. ■

7 — Ambilight

Der Begriff „Ambilight“ setzt sich zusammen aus „ambi“ (Umgebung) und „light“ (Licht). Ambilight bedeutet also so viel wie Umgebungslicht. Diese Technologie wurde von Philips für Fernseher entwickelt und patentiert. Ambilight vergrößert das TV-Gesichtsfeld in der Wahrnehmung des Zuschauers dadurch, dass der TV-Hintergrund in den Farben des aktuellen TV-Bildes angestrahlt wird. Was einst entwickelt wurde, um die Augen beim Fernsehen zu schonen, erfüllt nicht nur diesen Zweck, sondern sieht darüber hinaus auch noch gut aus. Ambilight ist kontext-sensitiv: Ändern sich die Farben des TV-Bildes, ändern sich simultan die Farben dieser Hintergrundbeleuchtung. Dieses Kapitel stellt Ihnen zwei verschiedene Wege vor, mit denen Sie Ihren „normalen“ Fernseher in einen Ambilight-TV verwandeln können. Der minimalistische Weg nutzt das Mediacenter *KODI* und einen Micro-Controller für 5 €. Der universelle Weg nutzt einen Raspberry Pi und stellt ein Ambilight für jede HDMI-Quelle zur Verfügung. Falls Sie einen Ambilight-TV von Philips Ihr eigen nennen, können Sie dieses Kapitel getrost überspringen. Falls nicht, zeige ich Ihnen, wie sie Ihren Fernseher mit Ambilight nachrüsten können, so dass dieser dem Original in nichts nachsteht.

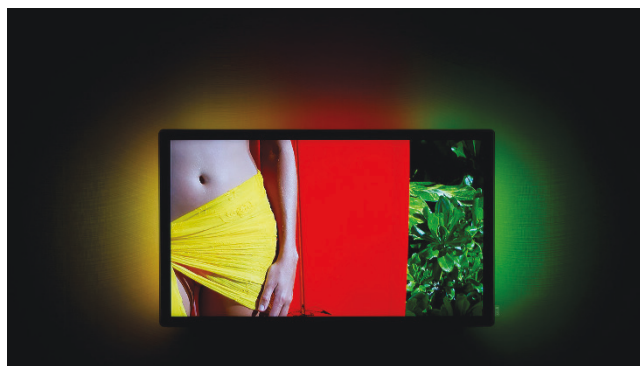


Abbildung 7.1: Ambilight verlängert das Fernsehbild über die Ränder des TVs hinaus (Quelle: Stephan Legachev, wikipedia.de).



7.1 Ambilight für LibreELEC

Der in diesem Abschnitt beschriebene Ambilight-Ansatz ist minimalistisch: Es werden nur sehr wenige Komponenten benötigt, um ein Ambilight zu verwirklichen:

- Ein Micro-Controller (z. B. Arduino Nano) für ca. 5 €
- Ein LED-Streifen (5 m für einen 55 Zoll Fernseher), beispielsweise APA102 mit 30 LEDs/m für ca. 50 €
- Ein Netzteil (5 V, ca. 10 A je nach Anzahl der LEDs) für 15 €
- Ein USB nach Mini-USB-Kabel zum Anschluss des Micro-Controllers für 1 €
- Litze bzw. Jumperkabel Buchse-Buchse für ca. 2 €
- Klebehaken zur Montage des LED-Streifens hinter dem Fernseher für ca. 2 €

In Summe macht das also ca. 75 €, um Ihren Fernseher bei der Verwendung von *LibreELEC* mit Ambilight auszustatten.

7.1.1 Ambilight Hardware anschließen

Dreh- und Angelpunkt des Ambilights sind die LED-Streifen, die hinter das Fernsehgerät montiert werden müssen. Hier gibt es verschiedene Hersteller und verschiedene Bestückungsdichten. Ich habe mit dem APA102-Streifen (ca. 30 LEDs pro Meter) sehr gute Erfahrungen gemacht. Für einen 55 Zoll Fernseher werden ca. 5 m LED-Streifen benötigt. Der APA102-Streifen ist



Abbildung 7.2: Der APA102(C) LED-Streifen eignet sich hervorragend, um ein Ambilight nachzubauen (Quelle: amazon.de).

beispielsweise bei Amazon oder Ebay erhältlich und das sowohl auf weißem wie auf schwarzem PCB (*Printed Circuit Board*). Wie in Abbildung 7.2 gut zu erkennen ist, stellt der APA102 LED-Streifen 4 Verbindungskabel zur Verfügung: Eine Spannungsversorgung (5 V, z. B. rot), eine Masseverbindung (GND, z. B. schwarz), eine Datenverbindung (DO/I, z. B. gelb) in beliebiger Farbe sowie eine Clock-Verbindung (CO/I, z. B. blau) in beliebiger Farbe. Die Buchstaben „I“ und „O“ stehen dabei für „Input“ und „Output“. Alle LEDs sind seriell geschaltet: Der Ausgang der einen LED ist mit dem Eingang der nächsten LED verbunden.

Verbinden Sie bitte die 5 V-Leitung (rot) mit dem 5 V-Ausgang Ihres Netzteils. Dasselbe machen Sie mit der Masse-Verbindung.

Info Werden längere LED-Streifen nur von einer Seite mit einer Spannung gespeist (z. B. 5 V), leuchten die LEDs am Ende der Kette dunkler als die LEDs am Anfang der Kette. Daher empfehle ich Ihnen, sowohl den Anfang als auch das Ende der LED-Kette an die Spannungsversorgung anzuschließen (5 V und Masse).

Info Bei manchen vorkonfigurierten APA102-Streifen ist die Masse mit einem roten Kabel verbunden und die 5 V-Leitung mit einem schwarzen Kabel. Schauen Sie hier bitte ganz genau hin und stellen Sie sicher, dass Masse auch mit Masse verbunden wird und 5 V mit 5 V des Netzteils. Vertauschen Sie beide Anschlüsse, kann das sowohl das Netzteil, aber in erster Linie auch den LED-Streifen zerstören.

Den DATA-in-Anschluss des LED-Streifens und den Clock-Anschluss des LED-Streifens verbinden wir mit dem Micro-Controller. Als Micro-Controller empfehle ich Ihnen einen Arduino Nano oder einen entsprechenden Nachbau (Abbildung 7.3).

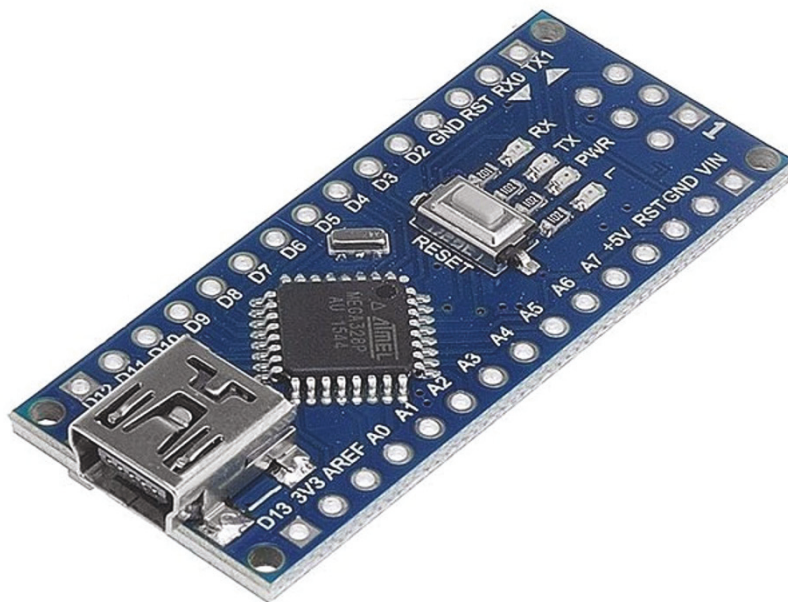


Abbildung 7.3: Einen Arduino-Nano-Nachbau gibt es günstig im Dreierpack. Der Elgoo Nano V3 mit CH340 USB-Chip wird von *LibreELEC* ohne Probleme erkannt (Quelle: amazon.de).

Info Einige Arduino (Nachbauten) werden als Set ausgeliefert, das man selbst zusammenlöten muss. Dies bezieht sich nur auf die Stiftleisten. Trauen Sie sich diese einfachen Lötarbeiten nicht zu, können Sie auch fertig konfektionierte Micro-Controller kaufen.

Der Masse-Anschluss des LED-Streifens muss mit einem Masse-Anschluss des Micro-Controllers verbunden werden.

Info Verbinden Sie auf keinen Fall die 5 V-Leitung des LED-Streifens oder des LED-Netzteils mit dem Arduino Micro-Controller.

Schließen Sie den LED-Streifen wie folgt an den Micro-Controller an:

Verbinden Sie bitte die Masse-Leitung des LED-Streifens mit einem GND-Pin des Arduino-Nano, z. B. dem 4. Pin von oben links (Abbildung 7.4). Verbinden Sie die Clock-Leitung (CLK) mit dem Pin D2. Verbinden Sie die Data-In-Leitung mit dem Pin D4.

NANO PINOUT

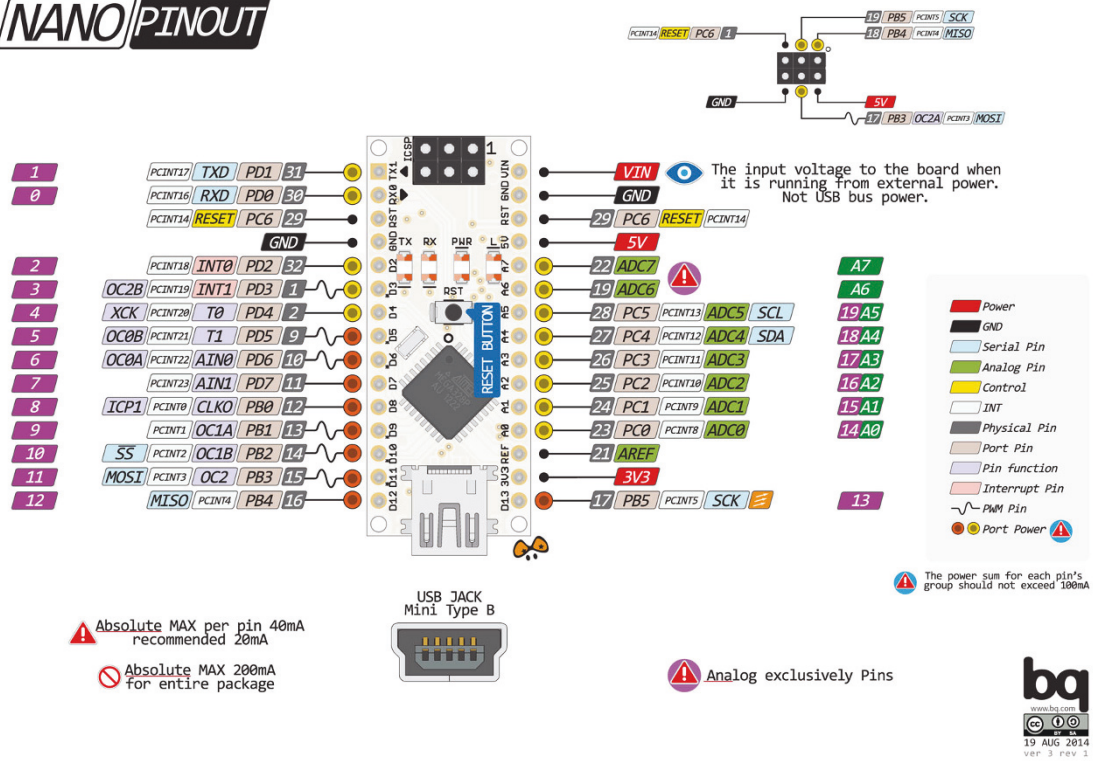


Abbildung 7.4: Die Pins des Arduino Nano (Quelle: Arduino Forum, user pighixx).

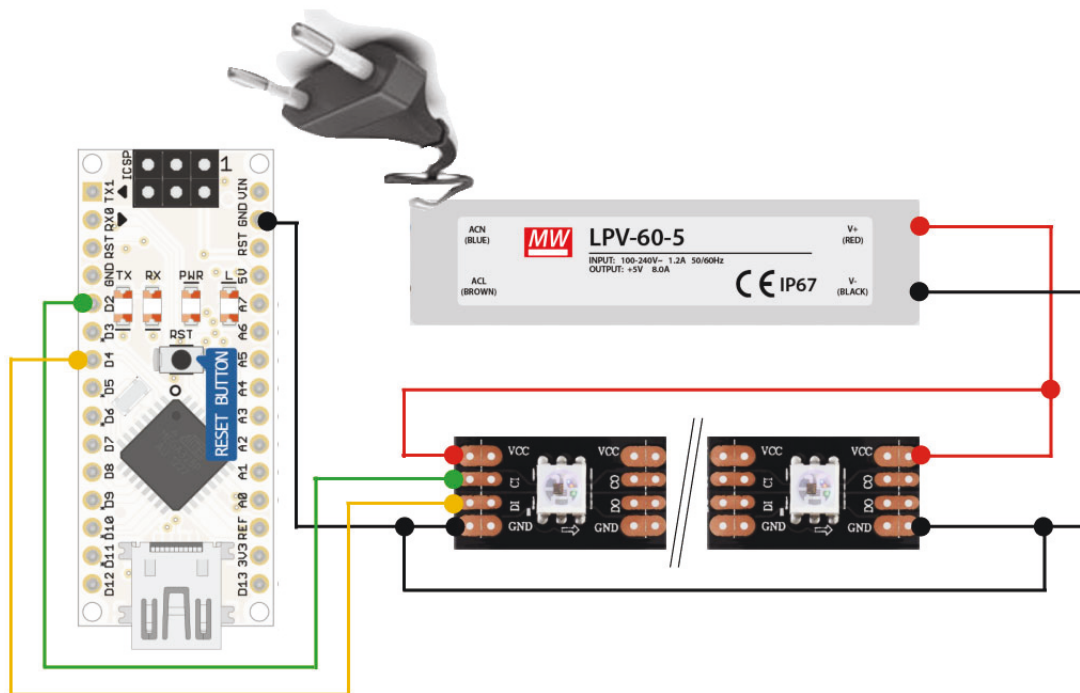


Abbildung 7.5: So Verkabeln Sie Ihr Ambilight richtig. Bitte achten Sie auf die Pfeile des LED-Streifens. DATA- und CLK-Eingänge müssen sich am Anfang des LED-Streifens befinden.



Sollten Sie eine LED-Kette mit Pegelwandler (3.3 V auf 5 V gekauft haben, können Sie die 3.3 V Spannungsversorgung des Pegelwandlers an den Pin 3V3 des Arduino Nano anschließen (2. Pin von unten rechts in Abbildung 7.4).

Verbinden Sie bitte den Micro-Controller mit Hilfe eines USB-nach-Mini-USB-Kabels mit Ihrem Raspberry Pi.

Den LED-Streifen können Sie entweder direkt auf die Rückseite Ihres Fernsehers kleben oder mit Hilfe von selbstklebenden Haltern befestigen.

Die LED-Streifen können Sie an jeder Stelle kürzen, indem Sie einfach mit einer Schere ein Stück zwischen den Kontakten entfernen. Wahrscheinlich wird Ihr LED-Streifen auch ein wenig zu lang sein. Das ist nicht weiter schlimm. Wir werden später die nicht benötigten LEDs per Software ausschalten.

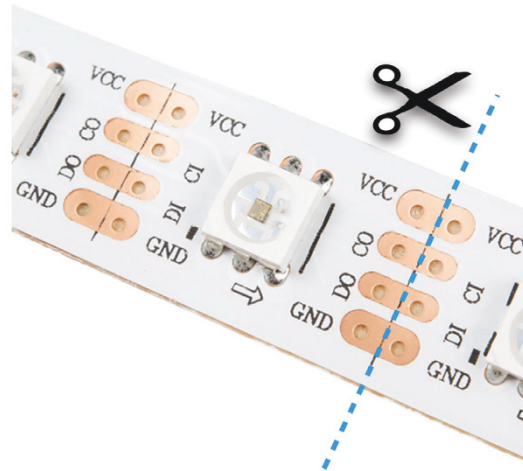


Abbildung 7.6: So schneiden Sie den LED-Streifen richtig.



Abbildung 7.7: Der LED-Streifen kann mit Hilfe von Klebehaken und Kabelbindern hinter dem Fernseher befestigt werden. Für einen 55 Zoll-Fernseher werden ca. 16 Klebehaken benötigt. Die LEDs strahlen auch seitlich. Kleben Sie den Streifen direkt auf den Fernseher, müssen Sie evtl. die Ecken schneiden und neu löten. Das Ambilight macht selbst bei curved-Fernsehern eine gute Figur.

7.1.2 Ambilight Software installieren

Damit unser Ambilight funktioniert, werden wir dreimal Software installieren: Zunächst einmal benötigt der Micro-Controller Software, einen sogenannten „Sketch“. Danach benötigen wir *Hyperion* für *LibreELEC* und zum Schluß müssen wir uns noch eine Konfigurationsdatei erzeugen, welche wir dann in ein Verzeichnis der *LibreELEC*-Distribution kopieren.

Beginnen wir mit dem Code für den Micro-Controller. Diesen benötigen wir, damit die LEDs unserer Lichterkette über USB mit Hilfe des Micro-Controllers angesprochen werden können. Die Entwicklungsumgebung für einen Arduino Nano und andere Derivate finden Sie unter <http://www.arduino.cc>. Diese Internetseite stellt Software für MacOS, Windows und LINUX bereit, mit deren Hilfe Sie den Micro-Controller flashen können. Die Software selbst ist Open-Source-Software. Bei meinen Versuchen habe ich die Windows-Version genutzt, aber ich bin sicher, dass die anderen Versionen dieser Software genau so gut funktionieren.

Laden Sie also bitte die Version Ihrer Wahl herunter. Um den Sketch (also das Micro-Controller-Programm), das ich Ihnen gleich vorstellen werden, zu flashen, benötigen wir ebenfalls die Bibliothek namens *FastLED*. Diese können Sie von <http://fastled.io/> herunterladen.



Die Arduino-Software-IDE kann mit gepackten (ge-zippten) Bibliotheks-Dateien umgehen. Sie müssen also die Bibliothek *FastLED* nicht entpacken, sondern können der IDE den Speicherort der gepackten Bibliothek mitteilen (Sketch → Bibliothek einbinden).

Für den APA102 LED-Streifen habe ich den folgenden Sketch benutzt:

Code-Ausschnitt 7.1.1

```

1  #include "FastLED.h"

3  #define ANALOG_MODE_AVERAGE 0
4  #define ANALOG_MODE_LAST_LED 1

6  /*****
7   S E T U P

9   set following values to your needs
10  *****/

12 #define INITIAL_LED_TEST_ENABLED true
13 #define INITIAL_LED_TEST_BRIGHTNESS 32 // 0..255
14 #define INITIAL_LED_TEST_TIME_MS 2000 // 10..

16 // Number of leds in your strip. set to "1" and ANALOG_OUTPUT_ENABLED to "true" to ↔
    activate analog only
17 // As of 26/1/2017:
18 // 582 leaves ZERO bytes free and this
19 // 410 is ok
20 // tested with 500 leds and is fine (despite the warning)
21 #define MAX_LEDS 160

23 // type of your led controller, possible values, see below
24 #define LED_TYPE APA102

26 // 3 wire (pwm): NEOPIXEL BTM1829 TM1812 TM1809 TM1804 TM1803 UCS1903 UCS1903B UCS1904 ↔
    UCS2903 WS2812 WS2852
27 //
    S2812B SK6812 SK6822 APA106 PL9823 WS2811 WS2813 APA104 WS2811_40 GW6205 ↔
    GW6205_40 LPD1886 LPD1886_8BIT
28 // 4 wire (spi): LPD8806 WS2801 WS2803 SM16716 P9813 APA102 SK9822 DOTSTAR

30 // For 3 wire led stripes line Neopixel/Ws2812, which have a data line, ground, and ↔
    power, you just need to define DATA_PIN.
31 // For led chipsets that are SPI based (four wires - data, clock, ground, and power), ↔
    both defines DATA_PIN and CLOCK_PIN are needed

33 // DATA_PIN, or DATA_PIN, CLOCK_PIN
34 // #define LED_PINS 6 // 3 wire leds
35 // #define LED_PINS 6, 8 // 4 wire leds
36 #define LED_PINS 4, 2 // 4 wire leds

39 #define COLOR_ORDER GRB // colororder of the stripe, set RGB in hyperion

41 #define OFF_TIMEOUT 15000 // ms to switch off after no data was received, set 0 to ↔
    deactivate

43 // analog rgb uni color led stripe - using of hyperion smoothing is recommended
44 // ATTENTION this pin config is default for atmega328 based arduinos, others might work ↔
    to
45 // if you have flickering analog leds this might be caused by unsynced pwm ↔
    signals
46 // try other pins is more or less the only thing that helps
47 #define ANALOG_OUTPUT_ENABLED false
48 #define ANALOG_MODE ANALOG_MODE_LAST_LED // use ANALOG_MODE_AVERAGE or ↔
    ANALOG_MODE_LAST_LED
49 #define ANALOG_GROUND_PIN 8 // additional ground pin to make wiring a ↔
    bit easier
50 #define ANALOG_RED_PIN 9
51 #define ANALOG_GREEN_PIN 10
52 #define ANALOG_BLUE_PIN 11

54 // overall color adjustments
55 #define ANALOG_BRIGHTNESS_RED 255 // maximum brightness for analog 0-255
56 #define ANALOG_BRIGHTNESS_GREEN 255 // maximum brightness for analog 0-255
57 #define ANALOG_BRIGHTNESS_BLUE 255 // maximum brightness for analog 0-255

```



```

59 #define BRIGHTNESS 255 // maximum brightness 0-255
60 #define DITHER_MODE BINARY_DITHER // BINARY_DITHER or DISABLE_DITHER
61 #define COLOR_TEMPERATURE CRGB(255,255,255) // RGB value describing the color ←
    temperature
62 #define COLOR_CORRECTION TypicalLEDStrip // predefined fastled color correction
63 //#define COLOR_CORRECTION CRGB(255,255,255) // or RGB value describing the color ←
    correction

65 // Baudrate, higher rate allows faster refresh rate and more LEDs
66 //#define serialRate 460800 // use 115200 for ftdi based boards
67 #define serialRate 115200 // use 115200 for ftdi based boards
68 #define serialRate 500000 // use 115200 for ftdi based boards

71 /*****
72  A D A L I G H T C O D E

74  no user changes needed
75  *****/

77 // Adalight sends a "Magic Word" (defined in /etc/boblight.conf) before sending the ←
    pixel data
78 uint8_t prefix[] = {'A', 'd', 'a'}, hi, lo, chk, i;

80 unsigned long endTime;

82 // Define the array of leds
83 CRGB leds[MAX_LEDS];

85 // set rgb to analog led stripe
86 void showAnalogRGB(const CRGB& led) {
87     if (ANALOG_OUTPUT_ENABLED) {
88         byte r = map(led.r, 0,255,0,ANALOG_BRIGHTNESS_RED);
89         byte g = map(led.g, 0,255,0,ANALOG_BRIGHTNESS_GREEN);
90         byte b = map(led.b, 0,255,0,ANALOG_BRIGHTNESS_BLUE);
91         analogWrite(ANALOG_RED_PIN , r);
92         analogWrite(ANALOG_GREEN_PIN, g);
93         analogWrite(ANALOG_BLUE_PIN , b);
94     }
95 }

97 // set color to all leds
98 void showColor(const CRGB& led) {
99     #if MAX_LEDS > 1 || ANALOG_OUTPUT_ENABLED == false
100     LEDs.showColor(led);
101     #endif
102     showAnalogRGB(led);
103 }

105 // switch of digital and analog leds
106 void switchOff() {
107     #if MAX_LEDS > 1 || ANALOG_OUTPUT_ENABLED == false
108     memset(leds, 0, MAX_LEDS * sizeof(struct CRGB));
109     FastLED.show();
110     #endif
111     showAnalogRGB(leds[0]);
112 }

114 // function to check if serial data is available
115 // if timeout occurred leds switch of, if configured
116 bool checkIncommingData() {
117     boolean dataAvailable = true;
118     while (!Serial.available()) {
119         if ( OFF_TIMEOUT > 0 && endTime < millis()) {
120             switchOff();
121             dataAvailable = false;
122             endTime = millis() + OFF_TIMEOUT;
123         }

```



```
124     }
126     return dataAvailable;
127 }

129 // main function that setups and runs the code
130 void setup() {
131     Serial.begin(serialRate);

133     // analog output
134     if (ANALOG_OUTPUT_ENABLED) {
135         // additional ground pin to make wiring a bit easier
136         pinMode(ANALOG_GROUND_PIN, OUTPUT);
137         digitalWrite(ANALOG_GROUND_PIN, LOW);
138         pinMode(ANALOG_BLUE_PIN, OUTPUT);
139         pinMode(ANALOG_RED_PIN, OUTPUT);
140         pinMode(ANALOG_GREEN_PIN, OUTPUT);
141     }

143     int ledCount = MAX_LEDS;
144     if (ANALOG_MODE == ANALOG_MODE_LAST_LED) {
145         ledCount--;
146     }

148     #if MAX_LEDS > 1 || ANALOG_OUTPUT_ENABLED == false
149         FastLED.addLeds<LED_TYPE, LED_PINS, COLOR_ORDER>(leds, ledCount);
150     #endif

152     // color adjustments
153     FastLED.setBrightness ( BRIGHTNESS );
154     FastLED.setTemperature( COLOR_TEMPERATURE );
155     FastLED.setCorrection ( COLOR_CORRECTION );
156     FastLED.setDither ( DITHER_MODE );

158     // initial RGB flash
159     #if INITIAL_LED_TEST_ENABLED == true
160     for (int v=0;v<INITIAL_LED_TEST_BRIGHTNESS;v++)
161     {
162         showColor(CRGB(v,v,v));
163         delay(INITIAL_LED_TEST_TIME_MS/2/INITIAL_LED_TEST_BRIGHTNESS);
164     }

166     for (int v=0;v<INITIAL_LED_TEST_BRIGHTNESS;v++)
167     {
168         showColor(CRGB(v,v,v));
169         delay(INITIAL_LED_TEST_TIME_MS/2/INITIAL_LED_TEST_BRIGHTNESS);
170     }
171     #endif
172     showColor(CRGB(0, 0, 0));

174     Serial.print("Ada\n"); // Send "Magic Word" string to host

177     boolean transmissionSuccess;
178     unsigned long sum_r, sum_g, sum_b;

180     // loop() is avoided as even that small bit of function overhead
181     // has a measurable impact on this code's overall throughput.
182     for(;;) {
183         // wait for first byte of Magic Word
184         for (i = 0; i < sizeof prefix; ++i) {
185             // If next byte is not in Magic Word, the start over
186             if (!checkIncommingData() || prefix[i] != Serial.read()) {
187                 i = 0;
188             }
189         }

191         // Hi, Lo, Checksum
192         if (!checkIncommingData()) continue;
```

```

193 hi = Serial.read();
194 if (!checkIncommingData()) continue;
195 lo = Serial.read();
196 if (!checkIncommingData()) continue;
197 chk = Serial.read();

199 // if checksum does not match go back to wait
200 if (chk != (hi ^ lo ^ 0x55)) continue;

202 memset(leds, 0, MAX_LEDS * sizeof(struct CRGB));
203 transmissionSuccess = true;
204 sum_r = 0;
205 sum_g = 0;
206 sum_b = 0;

208 int num_leds = min ( MAX_LEDS, (hi<<8) + lo + 1 );

210 // read the transmission data and set LED values
211 for (int idx = 0; idx < num_leds; idx++) {
212     byte r, g, b;
213     if (!checkIncommingData()) {
214         transmissionSuccess = false;
215         break;
216     }
217     r = Serial.read();
218     if (!checkIncommingData()) {
219         transmissionSuccess = false;
220         break;
221     }
222     g = Serial.read();
223     if (!checkIncommingData()) {
224         transmissionSuccess = false;
225         break;
226     }
227     b = Serial.read();
228     leds[idx].r = r;
229     leds[idx].g = g;
230     leds[idx].b = b;
231     #if ANALOG_OUTPUT_ENABLED == true && ANALOG_MODE == ANALOG_MODE_AVERAGE
232         sum_r += r;
233         sum_g += g;
234         sum_b += b;
235     #endif
236 }

238 // shows new values
239 if (transmissionSuccess) {
240     endTime = millis() + OFF_TIMEOUT;
241     #if MAX_LEDS > 1 || ANALOG_OUTPUT_ENABLED == false
242     FastLED.show();
243     #endif

244     #if ANALOG_OUTPUT_ENABLED == true
245     #if ANALOG_MODE == ANALOG_MODE_LAST_LED
246         showAnalogRGB(leds[MAX_LEDS-1]);
247     #else
248         showAnalogRGB(CRGB(sum_r/MAX_LEDS, sum_g/MAX_LEDS, sum_b/MAX_LEDS));
249     #endif
250     #endif
251 }
252 }
253 }
254 } // end of setup

256 void loop() {
257     // Not used. See note in setup() function.
258 }

```

Bitte ändern Sie die entsprechenden Zeilen am Anfang des Sketches, um evtl. andere LED-Streifen zu verwenden.

Weiterhin sollten Sie den Wert von `MAX_LEDS` auf die Anzahl Ihrer LEDs anpassen. Dasselbe gilt für Zeile 36 des Sketches, den `DATA-Pin` und den `CLOCK-Pin`.

Für den von mir empfohlenen Arduino Nano-Nachbau habe ich die Arduino-IDE-Einstellung Arduino/Genuino Uno gewählt, um den Sketch auf den Micro-Controller hochzuladen. Kopieren Sie den Sketch in ein leeres Projekt und laden Sie ihn dann auf den Micro-Controller hoch. Nachdem der Arduino startet - der USB-Anschluss dient dabei gleichzeitig als Spannungsversorgung - sollten die LEDs bereits ein wenig leuchten.

Bitte starten Sie für den nächsten Schritt Ihre LibreELEC-Distribution und installieren Sie das Hyperion-Addon. Dieses Addon hat keinerlei Einstellungsmöglichkeiten. Nachdem es einmal geladen wurde, wird das Ambilight dauernd an sein. Ich stelle Ihnen im weiteren Verlauf dieses Kapitels noch ein eigenes Plugin vor, welches es Ihnen erlaubt, das Ambilight mit der Fernbedienung ein- und auszuschalten. Sie finden das Hyperion-Addon unter *Alle Addons* → *Dienste*. Hyperion erwartet noch eine Konfigurationsdatei, damit das Ambilight funktioniert.

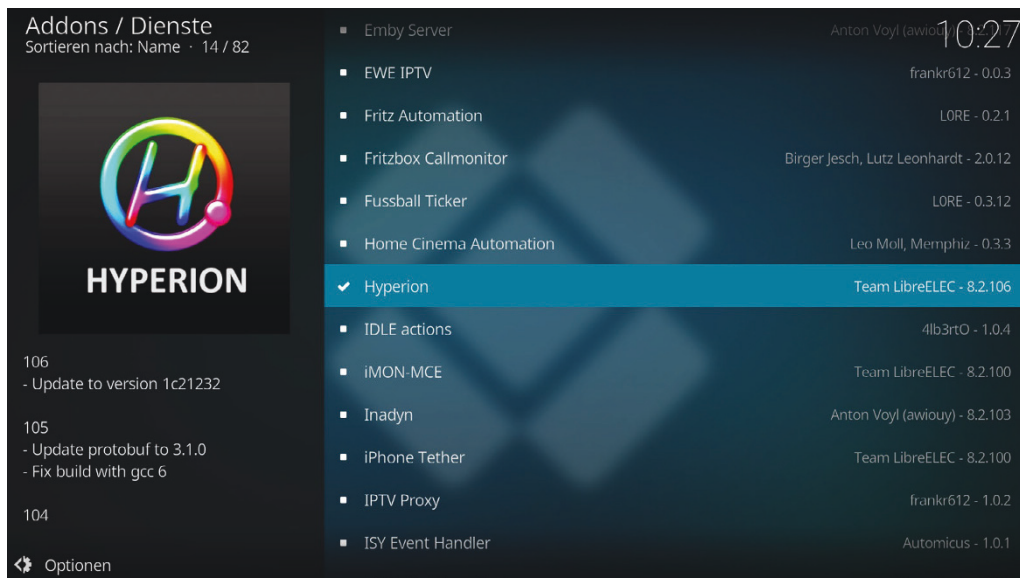


Abbildung 7.8: LibreELEC stellt das Hyperion-Addon unter Dienste bereit.

Diese Datei heißt `hyperion.config.json` und muss in das LibreELEC-Verzeichnis `.kodi/userdata/addon_data/service.hyperion` kopiert werden. Es würde den Rahmen dieses Buches sprengen, meine komplette Konfigurationsdatei an dieser Stelle abzdrukken. Daher erwähne ich an dieser Stelle nur die wichtigsten Einstellungen.

Code-Ausschnitt 7.1.2

```
1  {
2      // DEVICE CONFIGURATION
3      "device" :
4      {
5          "name"      : "MyHyperionConfig",
6          "type"      : "adalight",
7          "output"    : "/dev/ttyUSB0",
8          "rate"      : 500000,
9          "delayAfterConnect" : 0,
10         "colorOrder" : "gbr"
11     },
12     // COLOR CALIBRATION CONFIG
13     "color" :
14     {
15         "channelAdjustment" :
16         [
17             {
18                 "id" : "default",
19                 "leds" : "0-107",
20                 "pureRed" :
21                 {
22                     "redChannel" : 255,
23                     "greenChannel" : 0,
24                     "blueChannel" : 0
25                 },
26                 "pureGreen" :
27                 {
28                     "redChannel" : 0,
29                     "greenChannel" : 255,
30                     "blueChannel" : 0
31                 },
32                 "pureBlue" :
33                 {
34                     "redChannel" : 0,
35                     "greenChannel" : 0,
36                     "blueChannel" : 255
37                 }
38             }
39         ],
40         "temperature" :
41         [
42             {
43                 "id" : "default",
44                 "leds" : "0-107",
45                 "correctionValues" :
46                 {
47                     "red" : 255,
48                     "green" : 255,
49                     "blue" : 255
50                 }
51             }
52         ],
53         "transform" :
54         [
55             {
56                 "id" : "default",
57                 "leds" : "0-107",
58                 "hsl" :
59                 {
60                     "saturationGain" : 1.0500,
61                     "luminanceGain" : 1.0500,
62                     "luminanceMinimum" : 0.0000
63                 },
64                 "red" :
65                 {
66                     "threshold" : 0.0800,
67                     "gamma" : 1.4900
```

```
68         },
69         "green" :
70         {
71         "threshold" : 0.0800,
72         "gamma" : 1.4900
73         },
74         "blue" :
75         {
76         "threshold" : 0.0800,
77         "gamma" : 1.4500
78         }
79     },
80     ],
81     // SMOOTHING CONFIG
82     "smoothing" :
83     {
84         "type" : "linear",
85         "time_ms" : 200,
86         "updateFrequency" : 25.0000,
87         "updateDelay" : 0
88     }
89 },
90 "framegrabber" :
91 {
92     "width" : 64,
93     "height" : 64,
94     "frequency_Hz" : 50.0,
95     "priority" : 890
96 },
97 // BLACKBORDER CONFIG
98 "blackborderdetector" :
99 {
100     "enable" : true,
101     "threshold" : 0.05,
102     "unknownFrameCnt" : 600,
103     "borderFrameCnt" : 50,
104     "maxInconsistentCnt" : 10,
105     "blurRemoveCnt" : 1,
106     "mode" : "default"
107 },
108 // KODI CHECK CONFIG
109 "xbmcVideoChecker" :
110 {
111     "xbmcAddress" : "127.0.0.1",
112     "xbmcTcpPort" : 9090,
113     "grabVideo" : true,
114     "grabPictures" : true,
115     "grabAudio" : true,
116     "grabMenu" : false,
117     "grabPause" : false,
118     "grabScreensaver" : true,
119     "enable3DDetection" : true
120 },
121 // BOOTEFFECT CONFIG
122 "bootsequence" :
123 {
124     "color" : [0,0,0],
125     "effect" : "Rainbow swirl fast",
126     "duration_ms" : 7000,
127     "priority" : 700
128 },
129 // JSON SERVER CONFIG
130 "jsonServer" :
131 {
132     "port" : 19444
133 },
134 // PROTO SERVER CONFIG
135 "protoServer" :
136 {
```

```

137         "port" : 19445
138     },
139     // WEBCONFIG SERVER
140     "webConfig" :
141     {
142         "document_root" : "/storage/.kodi/addons/service.hyperion/webconfig",
143         "port" : 8099
144     },
145     // EFFECT PATH
146     "effects" :
147     {
148         "paths" :
149         [
150             "/storage/.kodi/addons/service.hyperion/effects",
151             "/usr/share/hyperion/effects"
152         ]
153     },
154     // LED CONFIGURATION
155     "leds" :
156     [
157         {
158             "index" : 0,
159             "hscan" : { "minimum" : 0.0000, "maximum" : 0.0500 },
160             "vscan" : { "minimum" : 0.9474, "maximum" : 1.0000 }
161         },
162         ...
163     ],
164     "endOfJson" : "endOfJson"
165 }

```

Die genauen LED-Einstellungen ab Zeile 163 habe ich weggelassen.



Die komplette Datei *hyperion.config.json* für meine LED-Konfiguration steht im Downloadbereich dieses Buches auf <https://www.springer.com/de/book/978-3-662-58143-8> zur Verfügung. Selbst, wenn meine LED-Konfiguration nicht für Ihre passt, sollte die LED-Anzeige eines Regenbogens bei Ihrem Ambilight auch mit meiner Konfigurationsdatei funktionieren.

Um die LEDs Ihres Ambilight zu konfigurieren, gibt es das kostenlose Tool *Hypercon* für *LibreELEC*. Das Tool steht zum Download unter <https://wiki.libreelec.tv/hypercon> in einer LINUX- und in einer Windows-Version bereit. Da das Tool eine Java-Applikation ist, setzt es ein installiertes Java voraus. Unter LINUX können Sie Java so installieren:

Code-Ausschnitt 7.1.3

```
1 sudo apt-get install openjdk-7-jre
```

und das Programm anschließend mit

Code-Ausschnitt 7.1.4

```
1 java -jar ./hypercon-LE.jar
```

aufrufen. Unter Windows müssen Sie zunächst ein Java von <http://www.java.com/en/> installieren, bevor Sie das Tool mit einem Doppelklick öffnen können.

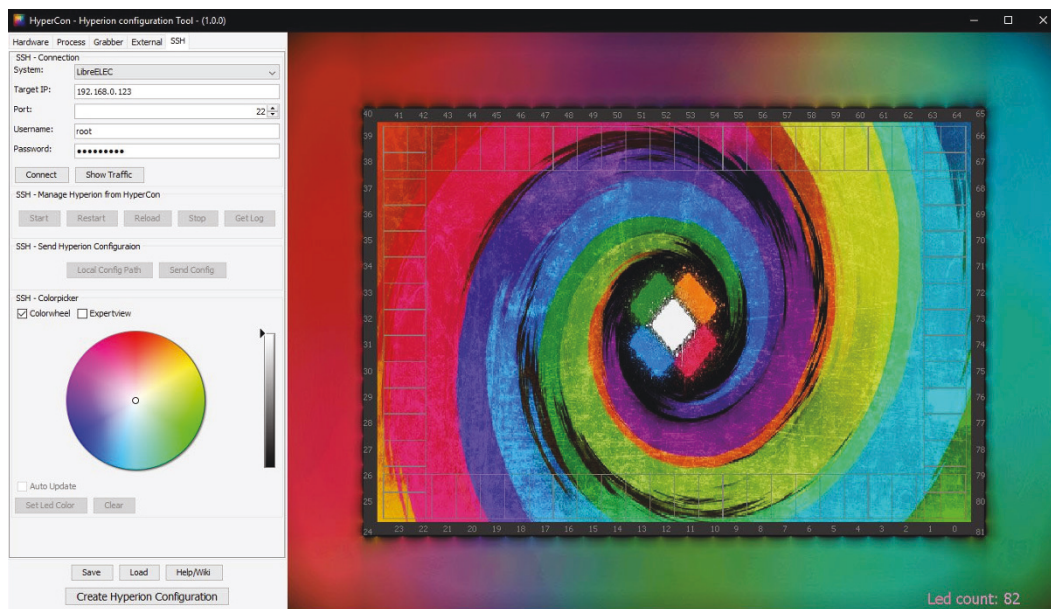


Abbildung 7.9: Mit dem Java-Tool *Hypercon* erstellt man schnell und komfortabel eine Konfigurationsdatei *hyperion.config.json* für *Hyperion* (Quelle: wiki.libreelec.tv).



Abbildung 7.10: Ambilight sieht selbst hinter einem curved-Fernseher gut aus.

Tragen Sie unter TargetIP die IP-Adresse Ihres Raspberry Pi ein. Der *Username* lautet „root“ und das *Password* „libreelec“.

Die Daten für den Grabber können Sie aus meinem Listing (Quelltext 7.1.2) übernehmen. Die meisten Änderungen werden Sie im Abschnitt *Hardware* vornehmen müssen und in Bezug auf die Anordnung und Anzahl Ihrer LEDs.



Bei meinem Ambilight-Aufbau habe ich die erste LED unten links hinter dem Fernseher platziert und dann den LED-Streifen nach oben (nicht nach rechts) weiter befestigt. Sie können aber - falls Sie sich für eine andere Reihenfolge/Anordnung entschieden haben - Ihre eigenen Daten in *Hypercon* eintragen.

Ein Klick auf den Button *Create Hyperion Config* speichert die Datei *hyperion.config.json*, die Sie dann nur noch in das entsprechende *LibreELEC*-Verzeichnis kopieren müssen. Selbst auf meinem curved-TV macht das Ambilight Marke „Eigenbau“ eine gute Figur.

7.1.3 Ambilight Fernbedienung

Auch für das Ambilight gibt es eine Fernbedienung, die auf einem mobilen Telefon oder einem Tablet läuft. Die App nennt sich „Hyperion“. Es gibt sie sowohl für Apple-Geräte als auch für Android-Geräte kostenlos.

Mit Hilfe dieser App kann das Ambilight durch Antippen des Farbkreises auf eine bestimmte Farbe eingestellt werden. Weiterhin kann die Helligkeit des Ambilight eingestellt werden.

Darüber hinaus bietet die App verschiedene Sondereffekte wie beispielsweise den „Knight Rider-Effekt“ oder den „Regenbogen-Effekt“. Die App sucht automatisch nach einer *LibreELEC/Hyperion*-Installation im Heimnetz. Der benötigte Port ist dabei schon eingetragen.

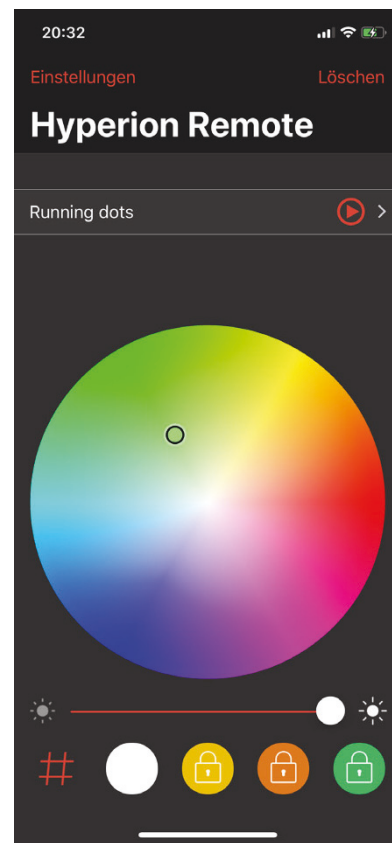


Abbildung 7.11: Hyperion remote.

7.1.4 Ambilight ein- und ausschalten

Wenn *Hyperion* einmal in *LibreELEC* aktiviert wurde, läuft es. Die einzige Möglichkeit, es abzuschalten wäre, es wieder zu deinstallieren. Da das keine Option ist, zeige ich Ihnen in diesem Abschnitt, wie Sie ein einfaches Addon für *LibreELEC* programmieren und installieren können, das genau diese Aufgabe übernimmt. Das Addon erlaubt es, den *Hyperion*-Dienst per Fernbedienung an- bzw. auszuschalten. Wir können es sogar noch erweitern und parallel das Ein- bzw. Ausschalten einer Funksteckdose ermöglichen, um das Netzteil für den LED-Streifen vom Netz zu trennen, um so Strom zu sparen. Doch dazu später mehr.

Damit wir das Rad nicht komplett neu erfinden müssen, benutzen wir als Vorlage ein „Hello World“-Beispiel. Dieses Beispiel steht unter https://kodi.wiki/view/HOW-TO:HelloWorld_addon zum Download bereit. Laden Sie bitte die Zip-Datei auf Ihren Rechner herunter (<https://github.com/zag2me/script.hello.world/archive/master.zip>).

Das Addon könnte man installieren, indem man im Addon-Browser den Eintrag *Aus Zip-Datei installieren* wählt. Bitte installieren Sie dieses Plugin aber noch nicht, da wir noch ein paar kleine Änderungen vornehmen werden.



Falls Sie ein *KODI*-Addon installieren und es danach ändern, müssen Sie eventuell die *KODI*-Datenbank löschen, bevor beispielsweise geänderte Icons eines Plugins angezeigt werden. *KODI* speichert bei der Installation die Icons des Plugins in einer Datenbank und ruft sie danach aus der Datenbank ab. Änderungen werden dadurch erst nach Löschen der Datenbank sichtbar.

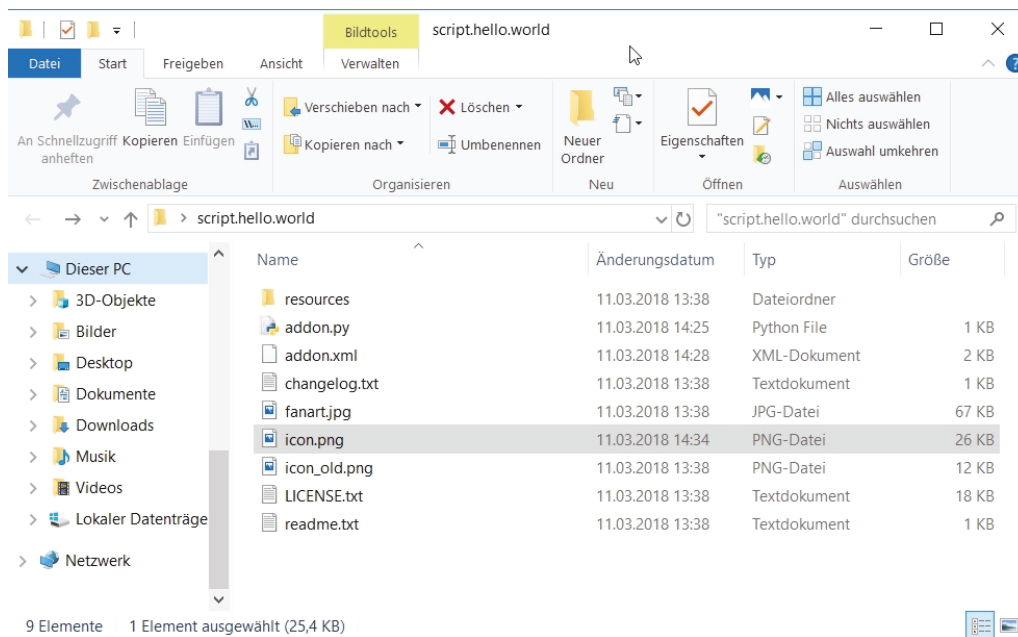


Abbildung 7.12: Die Struktur eines *KODI*-Addons.

Die wichtigsten Bestandteile des Addons sind:

- **addon.py**: *KODI*-Addons sind in Python programmiert. Diese Datei enthält den Python-Quelltext des Plugins.
- **addon.xml**: Diese Datei enthält die Metadaten des Addons. Die Daten liegen im XML-Format (*Extensible Markup Language*) vor und sind überwiegend selbsterklärend.
- **changelog.txt**: Diese Datei enthält Änderungsinformationen zum Addon und wird üblicherweise von Update zu Update fortgeschrieben.
- **icon.png**: Diese Datei ist das Icon des Plugins. Es kann entweder 256×256 oder 512×512 Pixel groß sein.
- **LICENSE.txt**: Diese Datei ist eine Textdatei, welche die Lizenzinformationen des Addons beinhaltet.

Darüber hinaus kann es noch weitere Verzeichnisse geben (z. B. *ressources*, welche weitere Hintergrundbilder beinhalten können). Suchen Sie sich zunächst aus dem Internet ein schönes Icon für die Datei *icon.png* aus oder entwerfen Sie ein eigenes Icon. Eine *Google*-Suche nach „Ambilight Icon“ ist dabei sehr hilfreich.

Den eigentlichen Quelltext des Addons kopieren wir in die Datei *addon.py*, nachdem wir zuvor deren Inhalt gelöscht haben:

Code-Ausschnitt 7.1.5

```

1  import xbmcaddon
2  import xbmcgui
3  import os
4  import subprocess

6  def get_process_id(name):
7      child = subprocess.Popen(['pgrep', '-f', name], stdout=subprocess.PIPE, shell=False)
8      response = child.communicate()[0]
9      return [int(pid) for pid in response.split()]

11 addon      = xbmcaddon.Addon()
12 addonname = addon.getAddonInfo('name')

14 try:
15     pid = get_process_id("hyperion")[0]
16     os.system('systemctl stop service.hyperion.service 2>/dev/null')
17     os.system('echo "0" > /dev/ttyUSB1')
18     line1 = "Hyperion was running. Stopping..."
19 except:
20     os.system('systemctl start service.hyperion.service 2>/dev/null')
21     os.system('echo "1" > /dev/ttyUSB1')
22     line1 = "Hyperion was not running. Starting..."

24 xbmcgui.Dialog().ok(addonname, line1)
25 %

```

Zunächst werden weitere Bibliotheken importiert, welche vom Addon benötigt werden. Nach der Definition der Funktion `get_process_id` und der Initialisierung des Addons stellt dieses fest, ob der `hyperion`-Dienst läuft oder nicht. Läuft er, wird er abgeschaltet, läuft er nicht, wird er eingeschaltet. Das Addon toggelt also zwischen „Ambilight an“ und „Ambilight aus“. Anschließend öffnet das Addon ein Fenster in *KODI* und zeigt den aktuellen Status des Addons an.

Vielleicht sind Ihnen die beiden Zeilen aufgefallen, in denen eine „0“ bzw. „1“ auf das USB-Gerät `/dev/ttyUSB1` geschrieben werden. Diese beiden Zeilen schalten eine Funksteckdose ein- bzw. aus, die bei mir von einem zweiten Arduino-Nano mit einem kleinen Funksender gesteuert wird. Diesen Aufbau erkläre ich Ihnen aber, nachdem wir das Addon im *KODI* installiert haben.

Nachdem wir die Datei `addon.py` mit dem oben gezeigten Inhalt gefüllt haben, „zippen“ wir wieder alle Dateien unseres Plugins. Sollten Sie die Dateien unter Windows bearbeitet haben, können Sie alle Dateien markieren und mit Hilfe der rechten Maustaste zu einem „zip“-Archiv hinzufügen.



Da Windows andere Zeilenumbrüche nutzt als LINUX, empfehle ich Ihnen unter Windows den freien Editor Notepad++, der kompatibel mit LINUX ist.

Unter LINUX können Sie Datei mit folgendem Befehl auspacken:

Code-Ausschnitt 7.1.6

```
1  zip -r ambilight.zip *
```

Die „gezippte“ Datei heißt `ambilight.zip` und enthält alle Dateien (*) des aktuellen Ordners. Kopieren Sie diese Datei dann bitte mit Hilfe von Samba in das `storage`-Verzeichnis der *LibreELEC*-Distribution. Sie können dieses Plugin jetzt installieren, indem Sie unter Addons *Aus zip-Datei installieren* wählen. Das Addon erscheint dann unter der Rubrik *Addons* im Hauptmenü.

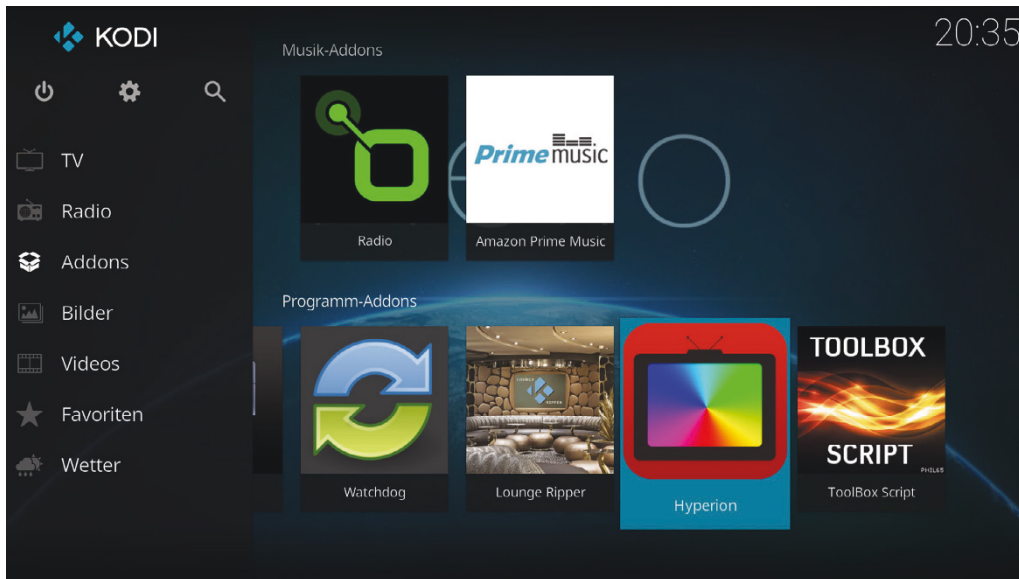


Abbildung 7.13: Das eigene *Hyperion*-Addon. Im Hintergrund ist noch das Titelbild des *Hello*-Addons zu erkennen. Das Icon wurde getauscht.

Info Wenn Sie viele Addons installiert haben, kann das schon einmal unübersichtlich werden. In diesem Fall können Sie das Ambilight-Addon unter *Favoriten* speichern, um schneller Zugriff darauf zu haben.

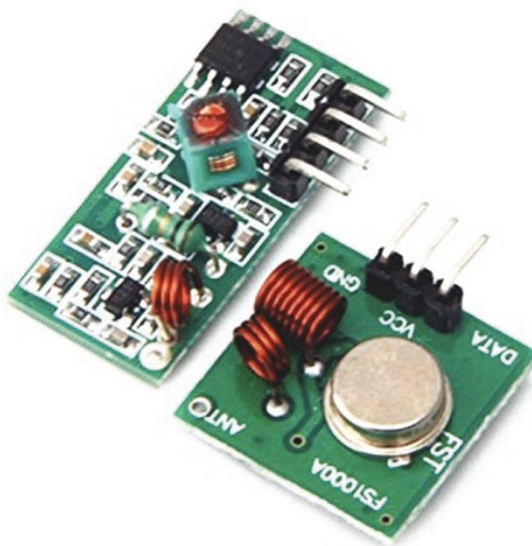


Abbildung 7.14: 433 MHz Sender (rechts) und Empfänger (links) (Quelle: amazon.de).

DATA-Pin nutze ich den Pin 10. Das ist der dritte Pin von unten links. Für die Verbindungen von Sender und Arduino Nano bieten sich Jumperkabel an. Der Sketch, den ich zum Ausschalten meiner Funksteckdose nutze, sieht wie folgt aus:

Das Addon gibt Auskunft darüber, ob der *Hyperion*-Dienst läuft oder nicht und schaltet ihn dementsprechend ein oder aus. Falls Sie Strom sparen wollen, empfehle ich Ihnen die Verwendung einer Funksteckdose, die Sie zusammen mit dem Ambilight ein- bzw. ausschalten können. Auch das können Sie mit Hilfe eines Arduino Nano erledigen, an den Sie einen 433 MHz stecken. Verbinden Sie dafür bitte den Ground-Pin (GND) des Senders mit einer GND-Verbindung auf dem Arduino Nano, z. B. den 2. Pin von oben rechts (Abbildung 7.4). Den VCC-Pin verbinden Sie bitte mit einem 5 V-Pin des Arduino Nano. Der befindet sich 2 Pins unterhalb des oben genannten GND-Pins. Als

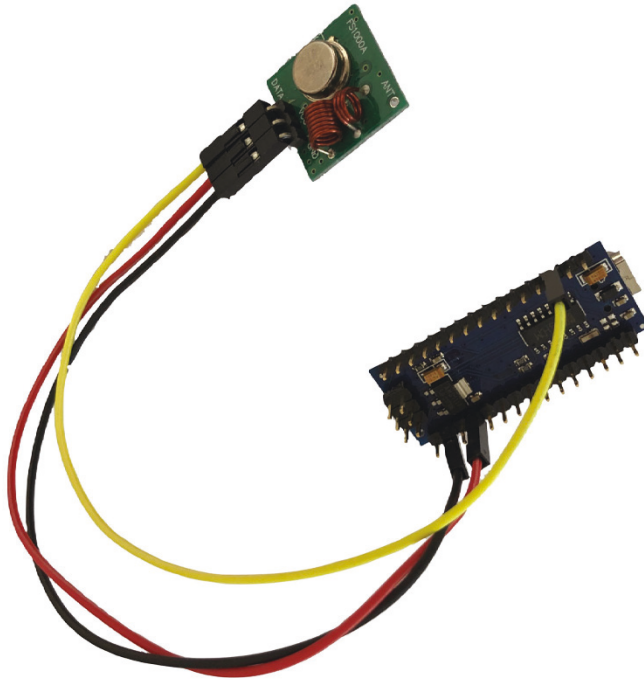


Abbildung 7.15: Der 433 MHz-Sender kann mit Jumperkabeln schnell mit dem Arduino Nano verbunden werden. Bitte beachten Sie, dass der Arduino Nano hier auf dem Kopf liegt, also mit dem USB-Anschluss nach unten (im Gegensatz zur Abbildung 7.3).

Code-Ausschnitt 7.1.7

```

1  #include <RCSwitch.h>
2  int incomingByte = 0; // for incoming serial data
3  RCSwitch mySwitch = RCSwitch();

4
5  void setup() {
6    Serial.begin(9600);
7    // Requires Libreelec screen /dev/ttyUSB1 9600
8    // Transmitter is connected to Arduino Pin #10
9    mySwitch.enableTransmit(10);
10   // Optional set protocol (default is 1, will work for most outlets)
11   mySwitch.setProtocol(1);
12   // Optional set pulse length.
13   mySwitch.setPulseLength(308);
14   // Optional set number of transmission repetitions.
15   mySwitch.setRepeatTransmit(15);
16 }

17
18 void loop() {
19   // send data only when you receive data:
20   if (Serial.available() > 0) {
21     // read the incoming byte:
22     incomingByte = Serial.read();

23     if (incomingByte == 48)
24     {
25       mySwitch.sendTriState("0000FFFOFFO");
26       delay(1000);
27     }
28     if (incomingByte == 49)
29     {
30       mySwitch.sendTriState("0000FFFOFOF");
31       delay(1000);
32     }
33   }
34 }
35 }

```


Das Skript zum Ein- und Ausschalten des Ambilights geht davon aus, dass der Arduino Nano, der mit der LED-Kette verbunden ist, am USB-Anschluss 0 steckt und der Sender, der die Funksteckdose schaltet, am USB-Anschluss 1. Gegebenenfalls müssen Sie die beiden USB-Ports tauschen. Für die Kommunikation mit dem 433 MHz-Sender muss die Baudrate für den USB-Anschluss, an dem der Arduino Nano mit dem Funksender steckt, auf 9600 Baud eingestellt werden. Üblicherweise nutzt man hierzu das Programm *setserial*, das aber unglücklicherweise unter *LibreELEC* nicht existiert. Die einzige Möglichkeit, die ich gefunden habe, ist der Befehl *screen*, den man auch dazu missbrauchen kann, die Baudrate eines USB-Anschlusses einzustellen. Bitte ergänzen Sie die folgende Zeilen in der Datei *storage/.config/autostart.sh*

Code-Ausschnitt 7.1.8

```
1 systemctl stop service.hyperion.service
2 sleep 5
3 screen -dmS myscreen /dev/ttyUSB1 9600
```

Die erste Zeile sorgt dafür, dass der Ambilight-Dienst *Hyperion* nach dem Start von *LibreELEC* ausgeschaltet ist. Möchten Sie, dass er eingeschaltet ist, löschen Sie diese Zeile bitte. Ein *sleep 5* stellt sicher, dass das System auch hochgefahren wurde, bevor der *screen*-Befehl dann aufgerufen wird, um die Übertragungsgeschwindigkeit für den USB-Port mit Arduino Nano und Funksender zu setzen. Der Name des Screens ist dabei vollkommen willkürlich (*myscreen*).

Wir kommen in Kapitel 10 noch einmal auf den Funksender zurück, den wir dann direkt mit dem Raspberry Pi verbinden. Dort finden Sie auch die Beschreibung einer passenden Funksteckdose.

7.2 Ambilight für jede HDMI-Quelle

Im vorangegangenen Kapitel haben wir eine *LibreELEC*-Distribution genutzt und diese mit einem Ambilight ergänzt. Das funktioniert ganz hervorragend und ist relativ preiswert. Der Nachteil dieser Lösung ist allerdings, dass Sie keine weiteren Quellen, wie beispielsweise eine Spielekonsole oder einen externen TV-Receiver mit Ambilight versehen können. Das ändern wir jetzt. Mit Hilfe des Raspberry Pi bauen wir ein Ambilight, das jede HDMI-Quelle als Eingang benutzen kann. Dazu ist allerdings ein wenig zusätzliche Hardware erforderlich:

- Ein HDMI-Splitter, der ein HDMI-Eingangssignal in zwei HDMI-Ausgangssignale mit demselben Inhalt umwandelt. Der Eingang des Splitters wird mit der Quelle verbunden, die Sie mit einem Ambilight versehen möchten, beispielsweise eine Spielekonsole. Ein Ausgang wird dann direkt mit dem Fernseher verbunden, am anderen Ausgang wird das Ambilight abgegriffen. Mit einer 4k-Auflösung kostet dieser Splitter ca. 20 €.



Abbildung 7.16: HDMI-Splitter (Quelle: amazon.de).



Abbildung 7.17: HDMI nach CVBS Konverter (Quelle: amazon.de).

- Ein HDMI nach CVBS Video-Konverter. Diesen Video-Konverter gibt es ebenfalls in einer 4k-fähigen Version. Der Konverter nimmt ein HDMI-Signal auf und wandelt es in ein CVBS-Signal (*Color Video Blanking Sync*) um. Das ist ein analoges Signal, welches von einem Grabber weiterverarbeitet werden kann. Der Konverter kostet ungefähr 25 € und ist etwas preiswerter, wenn man auf eine 4k-fähige Version verzichten kann. Genau wie der HDMI-Splitter benötigt auch der Konverter eine aktive Spannungsversorgung.

- Der Grabber nimmt das Bild vom HDMI-Konverter auf und stellt es dem Raspberry Pi zur Verfügung. Achten Sie bitte darauf, dass der Grabber mit dem Fusica UTV007 Chipsatz ausgestattet ist, da dieser problemlos unter LINUX funktioniert. Der Grabber kostet ca. 12 €. Wir benötigen nur den gelben Videoausgang des Grabbers.



Abbildung 7.18: CVBS Grabber (Quelle: amazon.de).

Weiterhin benötigen wir noch einige Kleinteile. Dazu zählen ein HDMI-A-Stecker auf HDMI-A für ca. 5 € sowie ein Chinch-Stecker auf Chinch-Stecker für ca. 2 €. Der Chinch-Stecker verbindet den gelben Video-Ausgang des HDMI-Konverters mit dem gelben Chinch-Eingang des Grabbers. Der HDMI-Verbindungsstecker verbindet einen Ausgang des HDMI-Splitters mit dem Eingang des HDMI-Konverters. Schließlich werden noch zwei HDMI-A-Kabel benötigt. Eines davon schließt die Quelle an unser Ambilight an, das andere dient dazu, die Quelle hinter dem Splitter an den Fernseher anzuschließen. Ein HDMI-Kabel ist für ca. 3 € erhältlich. Damit liegen die zusätzlichen Kosten für dieses Ambilight in einer Größenordnung von 70 €.



(a) HDMI-Kabel (A)
(Quelle: amazon.de).

(b) Chinch Steckverbindung
(Quelle: amazon.de).

(c) HDMI Steckverbindung
(Quelle: amazon.de).

Abbildung 7.19: Das Ambilight für jede HDMI-Quelle benötigt noch einige Kleinteile.

7.2.1 Hardware anschließen

Bitte verbinden Sie die LED-Hardware gemäß Abbildung 7.20.

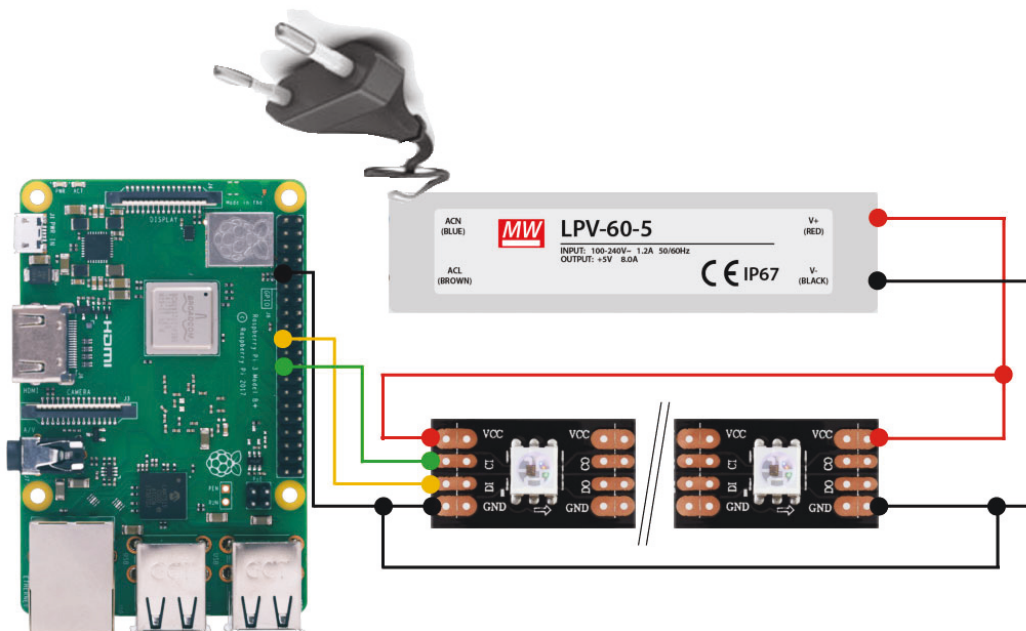


Abbildung 7.20: Verkabelung der LED-Kette mit Netzteil und Raspberry Pi.

Die Masseverbindung (GROUND, GND) schließen Sie bitte an den Ground-Pin (9) des Raspberry Pi an. Die DATA-Verbindung wird mit dem GPIO-Pin 19 des Raspberry Pi verbunden. Die Clock-Leitung (CLK, CLOCK) der LED-Kette wird mit dem Pin 23 des Raspberry Pi verbunden.

Info Bitte achten Sie auch hier auf die Pfeile der LED-Kette. Der DATA- und der CLK-Anschluss müssen am Anfang der LED-Kette eingespeist werden.

Die GPIO-Pins des Raspberry Pi sind für eine Spannung von 3.3 V ausgelegt. Viele LED-Ketten benötigen eine Spannung von 5 V für den DATA- und den CLK-Eingang. In diesem Fall empfehle ich Ihnen die Verwendung eines Level-Konverters, der die 3.3 V der verwendeten GPIOs auf 5 V hochsetzt. Die Firma lightberry.eu bietet einen solchen Konverter für ca. 23 € an.



Abbildung 7.21: Level-Konverter 3.3 V auf 5 V (Quelle: lightberry.eu).

Darüber hinaus bietet der oben genannte Laden auch komplette Lösungen mit verschiedenen LED-Streifen an, die bereits fertig vorkonfektioniert sind. Gleichzeitig können Sie auch HDMI-Splitter, HDMI-Konverter und Grabber in einem Gerät erwerben.



Abbildung 7.22: Verkabeln Sie den HDMI-Splitter, den Konverter und den Grabber wie in dieser Abbildung gezeigt. Die jeweiligen Netzteile habe ich wegen der Übersichtlichkeit nicht angeschlossen.

Info Häufig bieten die HDMI-Konverter zwei Modi: *PAL* und *NTSC*, die amerikanische Fernseh-Norm. Stellen Sie den Konverter bitte auf *PAL* ein.

7.2.2 Software installieren

Da wir einiges an USB-Geräten an den Raspberry Pi anschließen werden, stellen Sie bitte sicher, dass dieser den maximalen USB-Strom liefert, indem Sie in der Datei `/boot/config.txt` den folgenden Eintrag hinzufügen:

Code-Ausschnitt 7.2.1

```
1 max_usb_current=1
```

Weiterhin empfehle ich Ihnen, den HDMI-Ausgang des Raspberry Pi zu aktivieren, auch wenn kein HDMI-Gerät angeschlossen ist. Das ist z. B. sinnvoll, wenn Sie den Raspberry Pi ausschließlich für ein Ambilight nutzen, ihn aber nicht an einen Monitor oder Fernseher angeschlossen haben. Auch das erledigen Sie in der Datei `config.txt` durch Hinzufügen des Eintrags

Code-Ausschnitt 7.2.2

```
1 hdmi_force_hotplug=1
```

Der LED-Streifen selbst wird über ein sogenanntes SPI (*Serial Peripheral Interface*) gesteuert. Damit der Raspberry Pi dieses Interface aktiviert, muss es ebenfalls in der Datei `config.txt` aktiviert werden:

Code-Ausschnitt 7.2.3

```
1 dtparam=spi=on
```

Das SPI erwartet die DATA- und CLK-Leitung des LED-Streifens genau an den Pins, an denen wir sie zuvor angeschlossen haben (19 und 23). Speichern Sie Ihre Änderungen an der Datei `config.txt` und starten Sie den Raspberry Pi neu. Der Grabber sollte ohne Angabe weiterer Parameter gefunden werden.

Um das Ambilight zu steuern, nutzen wir wieder das Programm *Hyperion*. Wechseln Sie zunächst auf Ihrem Raspberry Pi in ein Verzeichnis Ihrer Wahl, beispielsweise das `/home`-Verzeichnis und laden dann das Installations-Skript herunter:

Code-Ausschnitt 7.2.4

```
1 cd /home/pi
2 curl -L --output install_hyperion.sh --get https://raw.githubusercontent.com/tvdzwan/↔
   hyperion/master/bin/install_hyperion.sh
3 chmod 755 ./install_hyperion.sh
4 sudo su
5 sh ./install_hyperion.sh
```

Das `chmod`-Kommando macht die Installationsdatei ausführbar. Zur Installation benötigen wir *root*-Rechte.

Nach einem Neustart läuft der *hyperion*-Dienst.

7.2.3 LEDs konfigurieren

Ähnlich wie bereits im *LibreELEC*-Kapitel beschrieben, müssen wir auch hier die LED-Kette konfigurieren. Das erledigen wir komfortabel mit dem Programm *HyperCon*, welches von der Webseite <https://sourceforge.net/projects/hyperion-project/files/hypercon/HyperCon.jar> heruntergeladen werden kann. Um meine LEDs zu konfigurieren, habe ich die Java-Applikation *HyperCon* unter Windows gestartet.

Das Programm funktioniert genauso gut unter LINUX. Bitte denken Sie daran, dass Java installiert sein muss, bevor Sie das Konfigurations-Tool starten.



Noch einmal zur Erinnerung: Unter Windows können Sie Java hier beziehen: <https://www.java.com/de/download/>.

Im ersten Schritt kümmern wir uns um die allgemeinen Einstellungen des *HyperCon*-Tools. Wählen Sie unter *Typ* bitte *SPI/APA102* für den APA102 LED-Streifen aus oder einen anderen Eintrag entsprechend Ihrem LED-Streifen. Die Ausgabe mit der oben gezeigten Verkabelung sollte auf */dev/spidev0.0* erfolgen. Stellen Sie die Baudrate auf 1.000.000. Sollte das nicht funktionieren, können Sie auch einen kleineren Wert eintragen. Bei meinen Experimenten habe ich bis 500.000 einen stabiles, flackerfreies Gesamtbild erhalten. Der APA102 LED-Streifen erwartet die RGB Byte-Reihenfolge als *BGR*. Nun ist LED-Zählen angesagt. Wie bereits erwähnt, beginnt meine LED-Kette unten links (von vorne gesehen) und bewegt sich dann im Uhrzeigersinn um den Fernseher. Die schwarze Randerkennung sollte aktiviert sein, da das Ambilight sonst nicht bei Filmen mit schwarzen Streifen funktioniert. Ein Schwellwert von 1% sollte ausreichend sein. Die Werte unterhalb des Reiters *Verarbeitung* können Sie zunächst so stehen

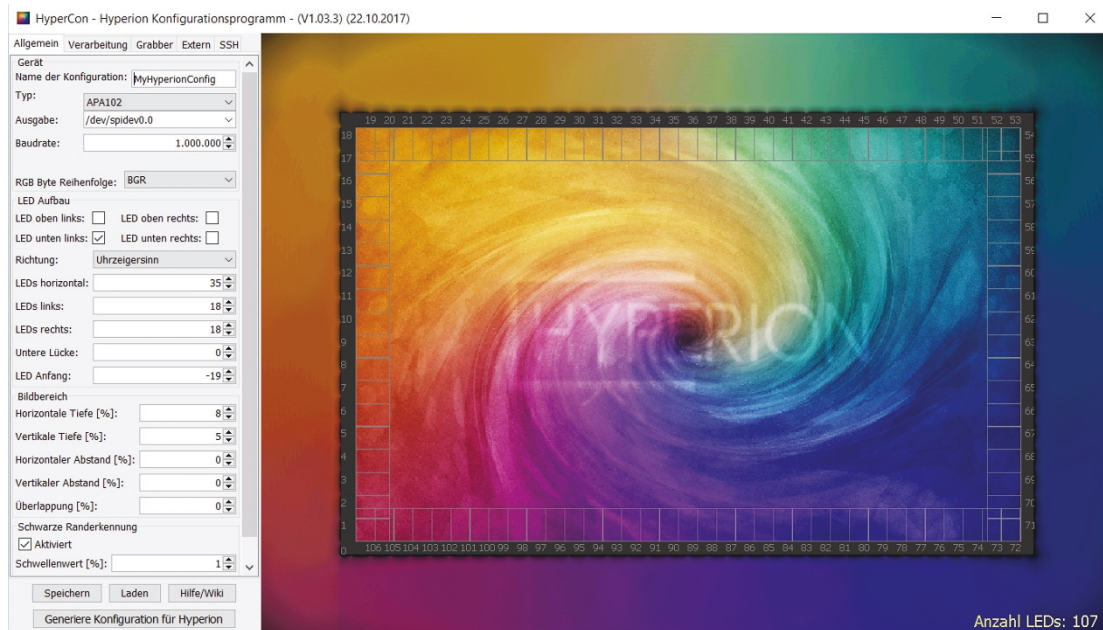


Abbildung 7.23: *Hypercon*-Konfiguration 1. Teil.

lassen. Sollten Sie später Fein-Tuning betreiben wollen, können Sie die hier genannten Einstellungen wie Weißwert, Gamma, Temperatur oder Schwelle nach Ihrem persönlichen Geschmack einstellen. Manche empfinden das Bild als „flüssiger“, wenn die Zeit der Glättung auf 100 ms und die Aktualisierungsfrequenz auf 25 Hz eingestellt werden.

In der Rubrik *Grabber* können Sie ebenfalls alle Einstellungen auf den Standardeinstellungen stehen lassen.

Die externen Einstellungen *Extern* erlauben es Ihnen, beispielsweise den Starteffekt für das Ambilight auszuwählen. Ich mag den Regenbogeneffekt gerne, der bereits voreingestellt ist.

In der Rubrik *SSH* können Sie sich über *ssh* mit Ihrem Raspberry Pi verbinden und die aktuelle Konfiguration senden bzw. fest eingestellte Farben für Ihr Ambilight ausprobieren.

Ein Druck auf den Knopf „Generiere Konfiguration für *Hyperion*“ generiert und speichert die Datei *hyperion.config.json*, die dann noch auf Ihren Raspberry Pi übertragen werden muss. Der Raspberry Pi erwartet diese Datei im Verzeichnis */etc/hyperion*. Mit Hilfe des „Speichern“-Knopfes können Sie alle Einstellungen von *HyperCon* speichern. Das ist praktisch, damit Sie nicht bei jedem neuen Einstellungsversuch wieder LEDs zählen müssen.

Mit dem Raspberry Pi können Sie auch den Video4Linux2-Grabber (V4L2) verwenden. Meine Sektion für den Grabber in der Konfigurationsdatei sieht wie folgt aus:

Code-Ausschnitt 7.2.5

```
1     "grabber-v4l2" :
2     {
3         "device" : "/dev/video0",
4         "input" : 0,
5         "standard" : "PAL",
6         "width" : 720,
7         "height" : 576,
8         "frameDecimation" : 2,
9         "sizeDecimation" : 8,
10        "priority" : 800,
11        "mode" : "2D",
12        "cropLeft" : 102,
13        "cropRight" : 98,
14        "cropTop" : 5,
15        "cropBottom" : 7,
16        "redSignalThreshold" : 0.1,
17        "greenSignalThreshold" : 0.1,
18        "blueSignalThreshold" : 0.1
19    },
```

Hier können Sie auch den 2D/3D-Mode umschalten. Bei mir hat das leider nicht funktioniert. Darum habe ich mir so geholfen, dass ich den Wert von „cropRight“ einfach so groß gemacht habe, dass die zweite Bildhälfte in einem Side-by-Side-Bild komplett weggeblendet wurde (z. B. einen Wert von 358). Der Crop-Wert gibt dabei einen Bildbereich an, den der Grabber einfach wegschneidet und damit nicht berücksichtigt, um das Ambilight anzusteuern. In der Regel wird dies ein schwarzer Bildbereich sein, den der Grabber nicht auswerten soll.

Um die richtigen Crop-Bereiche zu finden, gibt es mehrere Möglichkeiten. Sie können in *HyperCon* mit einem Klick der rechten Maustaste auf dem Hintergrundbild ein Bild vom Grabber holen und dabei den Crop-Wert so lange verkleinern, bis die schwarzen Ränder verschwunden sind. Damit das funktioniert, muss man per *SSH* mit dem *hyperion*-Server verbunden sein (*SSH*-Tab).

Eine weitere Möglichkeit besteht darin, den Screenshot in einem Terminal zu veranlassen. Dazu führen Sie bitte den Befehl

Code-Ausschnitt 7.2.6

```
1 sudo hyperion-v4l2 --width 320 -- height 180 --screenshot
```

aus. Schauen Sie sich anschließend das Bild an und vergrößern Sie die Crop-Werte so lange, bis die schwarzen Ränder an allen Seiten komplett verschwunden sind. Dem oben gezeigten Befehl *hyperion-v4l* können Sie übrigens auch direkt die Crop-Parameter mitgeben.

Diese lauten `--crop-right` `--crop-left` `--crop-bottom` oder `--crop-top`. Hinter den jeweiligen Befehl müssen Sie noch die Anzahl der Pixel angeben, die Sie abschneiden möchten, also beispielsweise `--crop-right 12`. Den *Hyperion*-Dienst können Sie jederzeit mit

Code-Ausschnitt 7.2.7

```
1 sudo service hyperion stop
```

ausschalten und mit mit

Code-Ausschnitt 7.2.8

```
1 sudo service hyperion start
```

wieder einschalten.

Zusammenfassung 7 In diesem Kapitel haben wir uns angeschaut, wie man ein Fernsehgerät mit einem Ambilight versehen kann. Dabei habe ich Ihnen zwei verschiedene Möglichkeiten vorgestellt, die LED-Kette anzusteuern: mit oder ohne einem Micro-Controller. Die Version mit Micro-Controller ist nach meinen Erfahrungen deutlich unempfindlicher gegen Flackern einzelner LEDs. In Verbindung mit *LibreELEC* können Sie sich so ein Multimedia-Center bauen, das diesen Namen wirklich verdient hat.

Im nächsten Kapitel schauen wir uns Datenbanken an und nutzen sie, um eine Benutzerauthentifizierung für einen Hotspot zu erstellen. Diesen Hotspot können wir sogar dazu verwenden, uns (fast) anonym im Internet zu bewegen. Doch dazu mehr im nächsten Kapitel.

■

8 — Server und Datenbanken

Im den letzten vorigen haben Sie einen Eindruck von den multimedialen Fähigkeiten des Raspberry Pi erhalten und dabei den speziellen Grafikchip genutzt. In der kleinen Platine steckt aber noch mehr. In diesem Kapitel möchte ich Ihnen das Thema Server und Datenbanken näher bringen. Das hier Gesagte gilt für jedes beliebige LINUX-System und kann sofort auf einen LINUX-PC übertragen werden. Warum also der Raspberry Pi? Ganz einfach! Er ist klein und kostet im Dauerbetrieb sehr wenig. Damit ist er prädestiniert für den Einsatz als Webserver oder Hotspot. Beides werden wir in diesem Kapitel mit einer Datenbank verbinden. Lernen Sie, wie man einen WLAN-Hotspot für Gäste einrichtet und den Zugang mit einem Web-Interface und einer Datenbank beschränkt. Drucken Sie Ihre Dokumente von einem Telefon oder Tablet aus. Verschleiern Sie Ihre Identität im Netz oder richten Sie einen VPN-Server ein, mit dem Sie jederzeit von außen Kontakt aufnehmen können, um so auch im Hotel oder im Büro gesichert auf Ihr Heimnetz zugreifen zu können. Kennen Sie bereits Content Management-Systeme? Mit Ihnen lassen sich sehr einfach Webseiten generieren und warten. Ich werde Ihnen ein CMS vorstellen und wir werden es auf dem Raspberry Pi installieren.



8.1 AirPrint

Genau wie das weiter oben beschriebene *AirPlay*, ist auch *AirPrint* eine Erfindung eines im Silicon Valley ansässigen Computerkonzerns. Die Idee ist simpel und gut: kabellos drucken vom mobilen Gerät aus auf dem heimischen Drucker. *AirPrint* ist dabei nicht auf mobile Geräte beschränkt. Aber cool ist es schon, oder? Handy zücken, Email checken und den Anhang mal eben schnell ausdrucken. Dieser Abschnitt erklärt alle Schritte, die auf dem Raspberry Pi erforderlich sind, damit dieser als *AirPrint*-Druckerserver arbeitet. Und das Beste dabei: Sie müssen nicht einmal einen *AirPrint*-fähigen Drucker kaufen, sondern benötigen lediglich einen von LINUX unterstützten Drucker; und das sind inzwischen die meisten. Wie so oft beginnen wir mit der Installation einiger erforderlicher Programmpakete. Das eine oder andere haben Sie sicher schon installiert.

Code-Ausschnitt 8.1.1

```
1 sudo apt-get install avahi-daemon cups cups-pdf python-cups
```



CUPS steht für *Common Unix Printing System*. *CUPS* ist ein frei nutzbares Drucksystem für UNIX-Systeme (sagte ich nicht einmal „LINUX is not UNIX“?). Es wurde von Michael Sweet entwickelt, der inzwischen bei Apple angestellt ist. Damit hat Apple offiziell die Pflege und Weiterentwicklung von *CUPS* übernommen. *CUPS* selbst ist eine Server-Client-Struktur: Der Client sendet das zu druckende Dokument an den *CUPS*-Server. Dieser wiederum kümmert sich um den Ausdruck an dem Rechner, an dem der Drucker angeschlossen ist. Das bedeutet, Ihr Drucker muss selbst nicht am Pi angeschlossen sein, sondern kann sich an einem beliebigen Rechner in Ihrem Heimnetz (etwa an ein NAS angeschlossen) befinden. *CUPS* kann übrigens in einem Web-Browser konfiguriert werden. Nachdem wir *CUPS* installiert haben, müssen noch einige Änderungen an der Konfigurationsdatei vorgenommen werden, damit *CUPS* auch wirklich Drucker auf beliebigen Rechnern im Heimnetz adressieren kann. Öffnen Sie bitte die Datei `/etc/cups/cupsd.conf` in einem Editor Ihrer Wahl mit *root*-Rechten. Entfernen Sie ein mögliches `#`-Zeichen vor dem Port 631 und kommentieren sie *localhost* aus, so dass die entsprechende Zeile in der o. g. Datei so aussieht:

Code-Ausschnitt 8.1.2

```
1 # Only listen for connections from the local machine.
2 # Listen localhost:631
3 Port 631
```

Ändern Sie die *Location*-Sektion wie folgt:

Code-Ausschnitt 8.1.3

```
1 <Location />
2 # Allow shared printing...
3 Order allow,deny
4 Allow @LOCAL
5 Allow 10.*
6 </Location>
7 <Location /admin>
8 Order allow,deny
9 Allow @Local
10 Allow 10.*
11 </Location>
12 <Location /admin/conf>
13 AuthType Default
14 Require user @SYSTEM
15 Order allow,deny
16 Allow @Local
17 Allow 10.*
18 </Location>
```

Die Zeile *Allow Local* erlaubt dem lokalen Netzwerk den Zugriff auf den Drucker-Daemon. Die nächste Zeile *Allow 10.** erlaubt IP-Adressen, die mit 10 beginnen. Das wird Sie zu diesem Zeitpunkt noch verwundern. Wir werden aber später einen VPN-Server auf dem Pi installieren, der einen Zugriff von außen ermöglicht. Dieser Server vergibt dann IP-Adressen aus dem Adress-Raum 10.x.x.x. Durch den oben gezeigten Eintrag erlauben wir, die Heimdrucker aus der Ferne konfigurieren zu können. Da *CUPS* nach der Installation bereits gestartet wurde, muss der Dienst jetzt neu gestartet werden, damit unsere Änderungen an der Konfigurationsdatei Wirkung zeigen.

Code-Ausschnitt 8.1.4

```
1 sudo service cups restart
```

CUPS selbst läuft unter der Benutzergruppe *lpadmin*, was für *line printer administrator* steht, also Drucker-Administrator. Der Benutzer *pi*, unter dem Sie gerade arbeiten, gehört dieser Gruppe nicht an. Damit *pi* Einstellungen an den Druckern über *CUPS* ändern darf, muss er der Gruppe *lpadmin* angehören. Fügen wir *pi* also zur Gruppe *lpadmin* hinzu:

Code-Ausschnitt 8.1.5

```
1 sudo adduser pi lpadmin
```

Das System bestätigt die erweiterte Gruppenzugehörigkeit mit

Code-Ausschnitt 8.1.6

```
1 Füge Benutzer "pi" der Gruppe "lpadmin" hinzu ...
2 Benutzer pi wird zur Gruppe lpadmin hinzugefügt.
3 Fertig.
```

Die Konfiguration des Druckers nehmen wir über ein Web-Interface vor. Dazu rufen wir im Browser die IP-Adresse des Raspberry Pi für den Port 631 auf. Auf diesem Port lauscht nämlich *CUPS*.

Code-Ausschnitt 8.1.7

```
1 https://ip-adresse:631
```

Auf meinem iPad sieht die Raspberry Pi *CUPS*-Oberfläche so aus wie in Abb. 8.1. Wählen Sie

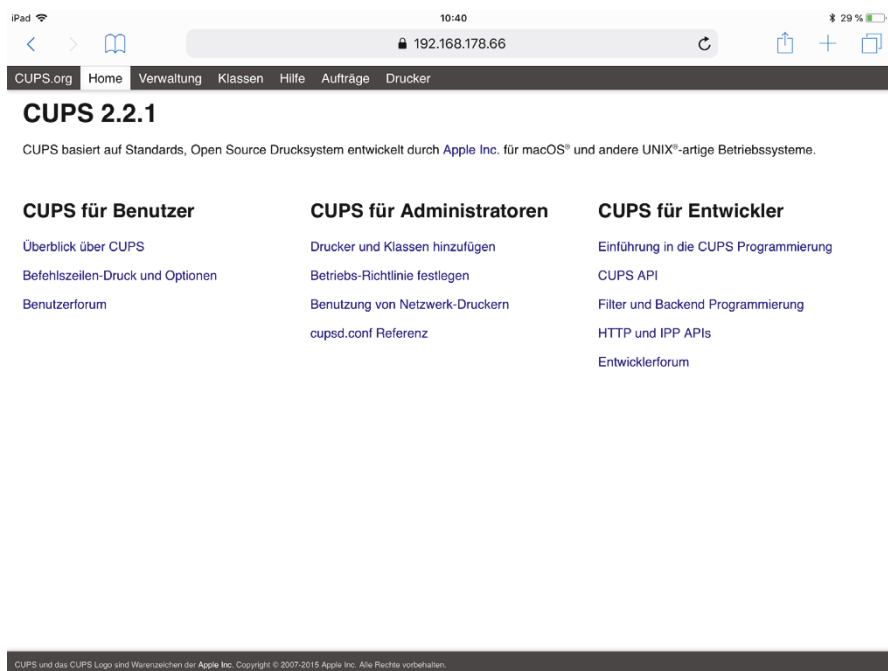


Abbildung 8.1: Der *CUPS*-Startbildschirm im Web-Browser.

im Menü bitte den Punkt *Verwaltung* aus. Unter grundlegenden Servereinstellungen wählen Sie bitte die folgenden Punkte an:

- Mit diesem System verbundene Drucker freigeben
- Fernwartung zulassen

Entscheiden Sie selbst, ob Sie das Drucken vom Internet aus erlauben möchten oder nicht. Wenn alle Einstellungen vorgenommen sind, klicken Sie bitte auf *Einstellungen ändern*. Der CUPS-Server wird neu gestartet, wenn Sie die Anmeldeaufforderung mit dem Benutzernamen *pi* und Ihrem Kennwort quittiert haben (Abb. 8.2). Sie haben nun zwei Möglichkeiten, fortzufahren:

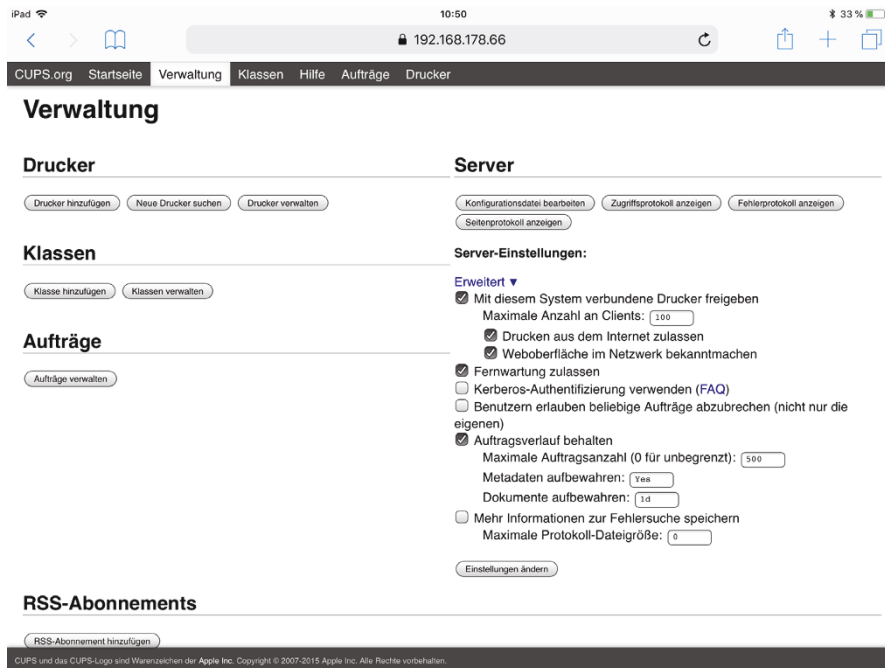


Abbildung 8.2: Die CUPS-Verwaltung im Web-Browser.

1. Der Drucker steckt in einem USB-Anschluss des Raspberry Pi. Schalten Sie in diesem Fall bitte den Drucker ein.
2. Der Drucker befindet sich in Ihrem Netzwerk und ist an einen anderen Rechner angeschlossen. Schalten Sie in diesem Fall bitte den anderen Rechner und den Drucker ein.

Wählen Sie im Verwaltungs-Dialog (Abb. 8.1) *Verfügbare Drucker auflisten*. CUPS zeigt Ihnen nun alle Drucker, die gefunden werden. Klicken Sie bitte links neben dem Druckernamen auf *Diesen Drucker hinzufügen*, passen Sie die Beschreibung für *Ort* an und aktivieren Sie *Diesen Drucker freigeben*. Danach klicken Sie bitte auf *Weiter*. Daraufhin müssen Sie den Druckerhersteller und das Modell angeben. Der Drucker ist fertig eingerichtet. Das können Sie durch Drucken einer Testseite überprüfen.



Sollte der gewünschte Drucker nicht unter *Verfügbare Drucker* auftauchen, klicken Sie im Verwaltungs-Menü auf *Drucker hinzufügen* (Abb. 8.2). Sie können dann in einem weiteren Menü den Netzwerkdrucker und sein Protokoll selbst eintragen.

Nachdem alle Drucker im Drucker-Server CUPS hinzugefügt worden sind, kümmern wir uns jetzt um *AirPrint*. Wir installieren *AirPrint* im */opt*-Verzeichnis.

Code-Ausschnitt 8.1.8

```

1 sudo mkdir /opt/airprint
2 cd /opt/airprint
3 sudo wget -O airprint-generate.py --no-check-certificate https://raw.githubusercontent.com/tjfontaine/airprint-generate/master/airprint-generate.py
4 sudo chmod +x airprint-generate.py
5 sudo ./airprint-generate.py -d /etc/avahi/services
6 sudo service cups restart
7 sudo service avahi-daemon restart

```

Nach der Installation müssen *cups* und *avahi* neu gestartet werden.



Mit Hilfe von *avahi* können Geräte im lokalen Netzwerk miteinander vernetzt werden, ohne dass diese manuell konfiguriert werden müssen.

Laut Autor der *AirPrint*-Software kann es vorkommen, dass bei den oben angegebenen Befehlen eine Fehlermeldung auftaucht:

Code-Ausschnitt 8.1.9

```

1 image/urf is not in mime types, [PRINTER_NAME] may not be available on ios6 (see https://github.com/tjfontaine/airprint-generate/issues/5)

```

In diesem Fall gehen Sie bitte so vor: Erzeugen Sie als Benutzer *root* die Datei */usr/share/cups/mime/airprint.types* mit dem folgenden Inhalt:

Code-Ausschnitt 8.1.10

```

1 #
2 # "$Id: $"
3 #
4 # AirPrint type
5 image/urf urf string(0,UNIRAST<00>)
6 #
7 # End of "$Id: $".
8 #

```

Erzeugen Sie weiterhin die Datei */usr/share/cups/mime/airprint.convs*, ebenfalls als Benutzer *root* - denken Sie also daran, dem Editor Ihrer Wahl ein *sudo* voran zu stellen. In diese Datei kopieren Sie bitte folgenden Inhalt:

Code-Ausschnitt 8.1.11

```

1 # "$Id: $"
2 #
3 # AirPrint
4 # Updated list with minimal set 25 Sept
5 image/urf application/pdf 100 pdftoraster
6 #
7 # End of "$Id: $".
8 #

```

Danach müssen das *AirPrint*-Skript erneut ausgeführt werden und *cups* und *avahi* neu gestartet werden:

Code-Ausschnitt 8.1.12

```

1 sudo ./airprint-generate.py -d /etc/avahi/services
2 sudo service cups restart
3 sudo service avahi-daemon restart

```

8.2 Apache

„LINUX is like an Indian tent: No Windows, no Gates and an Apache inside“. Übersetzen könnte man das mit „LINUX ist wie ein Indianerzelt: keine Fenster, keine Tore und ein Apache sitzt darin“. Man kann diesen Satz aber auch auf ein Betriebssystem und den Firmengründer des Microsoft-Imperiums beziehen. Den „Apache“ schauen wir uns in diesem Abschnitt genauer an. Die Installation ist schnell erledigt.



Code-Ausschnitt 8.2.1

```
1 sudo apt-get install apache2
```

Info Der *Apache*-Webserver ist ebenfalls ein Dienst, welcher unter `/etc/init.d/apache2` zu Hause ist. Nach der Installation wird *apache2* automatisch gestartet.

Es gibt einige Alternativen zu *apache2*, etwa *lighthttp* oder *LiteSpeed*. Die erste Alternative ist in Maik Schmidts Buch ausführlich beschrieben [Sch13b]. Ich habe mich für *Apache* entschieden, weil wir im späteren Verlauf eine komplette Datenbankanbindung und einen geschützten Zugang über *https* realisieren und beides mit *apache2* perfekt harmoniert. Das betrifft auch den *Radius* Zugangs-Server oder das Content Management-System *Contao*. Aber dazu später mehr. Jetzt unterziehen wir den *Apachen* einem ersten Test. Rufen Sie in einem Browser, der Zugriff auf ihr Netzwerk hat, die IP-Adresse des Raspberry Pi auf (`http://ip-adresse`). Der Browser sollte nun die *Apache* Standard-Seite anzeigen (Abb. 8.3).

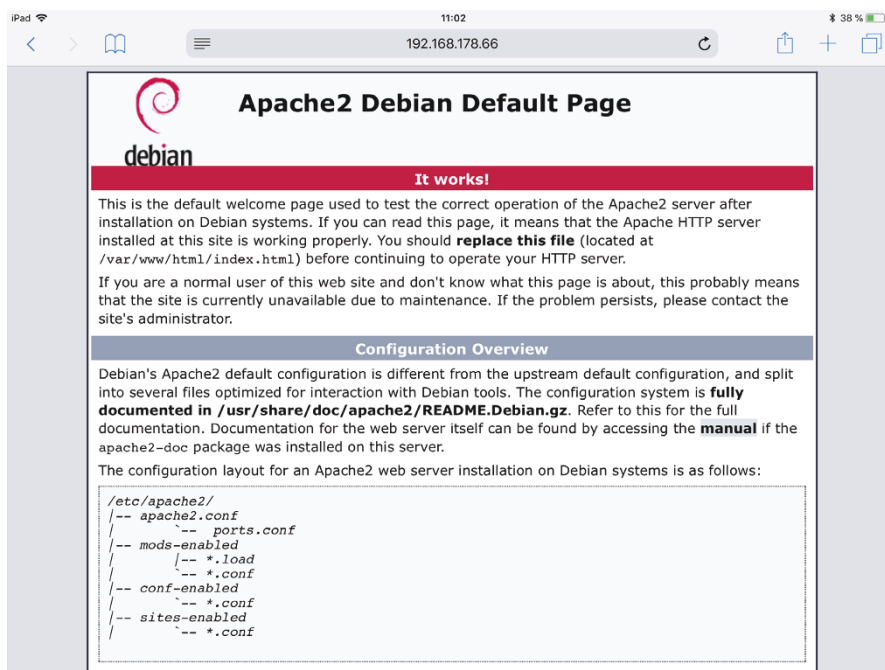


Abbildung 8.3: Die *Apache* Standard-Seite sagt „Hallo!“.

Info Wenn Sie den Browser auf dem Raspberry Pi selbst starten, können Sie über *localhost* auf den eigenen Rechner zugreifen. Das ersetzt dann die lokale IP-Adresse.

Der *Apache* schaut im Verzeichnis `/var/www/html` nach vorhandenen Webseiten. Dort liegt nach der Installation auch die Datei `index.html`, welche den in Abb. 8.3 angezeigten Text liefert.



Die Endung `.html` bezeichnet eine *HTML*-Datei (*HyperText Markup Language*). Hierbei handelt es sich um eine Beschreibungssprache, die festlegt, wie ein im Internet-Browser dargestellter Inhalt aussieht.

Möchte man weitere Webseiten erstellen, kann man das unterhalb von `/var/www` machen. Das Verzeichnis dieser Webseite müssen Sie dann hinter der IP-Adresse angeben. Beispiel: Sie erstellen einen Webauftritt im Verzeichnis `/var/www/beispiel`. Dieser ist dann unter der URL `http://ip-adresse/beispiel` erreichbar. Bitte beachten Sie, dass sowohl der Benutzer als auch die Gruppe der Web-Inhalte (also des Verzeichnisses `/var/www` und der dort angesiedelten Unterverzeichnisse) `www-data` heißen. Wenn Sie also als Benutzer `pi` mit der Gruppenzugehörigkeit `pi` mit einem `sudo`-Befehl neue Internet-Inhalte unterhalb der Verzeichnisstruktur `/var/www` erstellen, müssen Sie Besitzer und Gruppe aller Dateien auf `www-data` umstellen. Bleiben wir beim Unterverzeichnis `/var/www/beispiel`, können Sie das mit einem

Code-Ausschnitt 8.2.2

```
1 sudo chown -R www-data:www-data /var/www/beispiel
```

erledigen. Der Parameter `-R` bedeutet wieder *Rekursiv*. Alle Dateien und Unterverzeichnisse von `beispiel` erhalten nach dem Aufruf den neuen Besitzer und die neue Gruppe.

8.2.1 PHP

Viele Webseiten werden mit Hilfe von *PHP* erstellt.



Die Endung `.php` bezeichnet eine *PHP*-Datei (*PHP Hypertext Preprocessor*). Hierbei handelt es sich um eine Skript-Programmiersprache, die sehr häufig beim Erstellen von Internetseiten verwendet wird. Die Sprache wird auf dem Server abgearbeitet und liefert dann z. B. die Darstellung eines Internet-Auftritts.

Da wir *PHP* im weiteren Verlauf noch benötigen, installieren wir es in der aktuellen Version 7.

Code-Ausschnitt 8.2.3

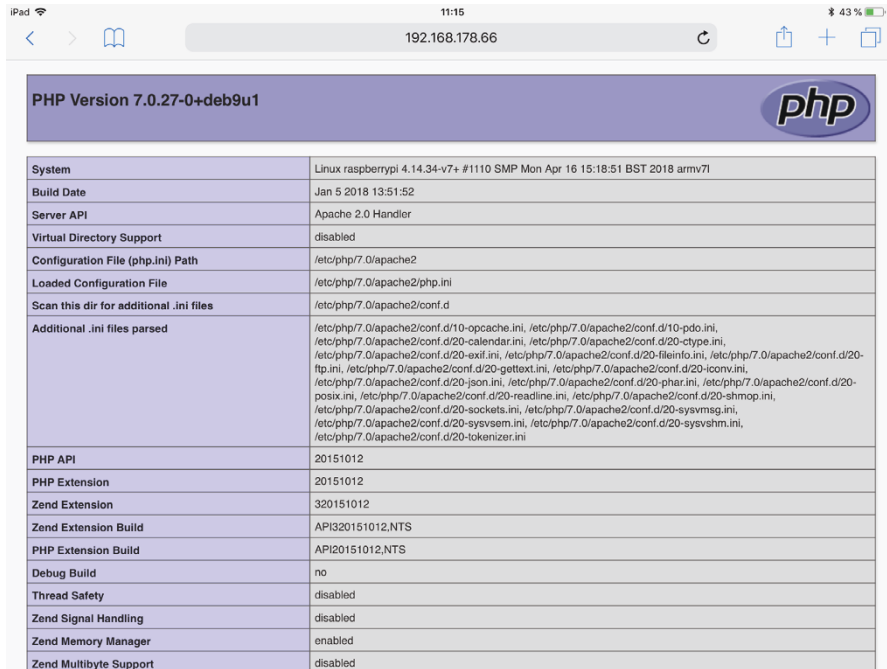
```
1 sudo apt-get install php7.0 libapache2-mod-php7.0
```

Kleines Beispiel gefällig? Editieren Sie bitte eine Datei `/var/www/html/phpinfo.php` unter Verwendung eines vorangestellten `sudo` mit folgendem Inhalt:

Code-Ausschnitt 8.2.4

```
1 <?php
2 phpinfo();
3 ?>
```

Rufen Sie dann in einem Browser die URL `http:ip-adresse/phpinfo.php` auf. Als Ergebnis sehen Sie alle Informationen über die aktuelle *PHP*-Version.



PHP Version 7.0.27-0+deb9u1	
System	Linux raspberrypi 4.14.34-v7+ #1110 SMP Mon Apr 16 15:18:51 BST 2018 armv7l
Build Date	Jan 5 2018 13:51:52
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-pear.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysmsg.ini, /etc/php/7.0/apache2/conf.d/20-syssem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012.NTS
PHP Extension Build	API20151012.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled

Abbildung 8.4: Die *PHP*-Infoseite listet alle Standard-Einstellungen auf.

8.2.2 MySQL

Noch sind wir nicht am Ende unserer Installationen angelangt. Ein wichtiger Baustein für unsere nächsten Projekte fehlt noch: die Datenbank. In der Datenbank werden wir später speichern, wer wie lange auf unseren Hotspot zugreifen kann. Weiterhin benötigen wir die Datenbank für das Content Management-System (CMS) *contao*, welches wir im weiteren Verlauf dieses Buches noch in Betrieb nehmen werden. Viele bezeichnen die Open Source-Datenbank *MySQL* als die populärste der Welt, welche die Grundlage unzähliger Webauftritte darstellt. Ähnlich wie LEGO beinhaltet der Name *MySQL* den Vornamen der Tochter eines Gründers (*My*) und die Abkürzung *SQL*, die für *Structured Query Language* (Strukturierte Anfrage Sprache) steht. Die Übersetzung trifft die Daseins-Berechtigung von *MySQL* genau. In der Datenbank stehen Einträge. Anfragen an die Datenbank werden mit entsprechenden Einträgen (oder auch keinen, wenn sie nicht vorhanden sind) beantwortet. Auf zur Installation!



Abbildung 8.5:
(Quelle: mysql.de)

Code-Ausschnitt 8.2.5

```
1 apt-get install mysql-server mysql-client php7.0-mysql
```

In der Raspberry Pi Debian-Stretch-Distribution wurde *MySQL* inzwischen durch *MariaDB* ersetzt. *MariaDB* ist aus einem Zweig von *MySQL* hervorgegangen und sollte vollständig kompatibel sein.



MySQL ist auch ein Dienst. Sie wissen schon... `/etc/init.d/mysql`.



Abbildung 8.6:

(Quelle: phpmyadmin.net).

Das Programm *myphpadmin* ist eine komfortable Web-Oberfläche zur Verwaltung der *MySQL*-Datenbank. Hier können Sie neue Datenbanken anlegen, Rechte vergeben oder Einträge erstellen und löschen. Ich habe es schon oft als Hilfsmittel gebraucht, wenn ich schnell in einer Datenbank etwas nachschauen wollte und keine Lust auf Konsolenbefehle hatte. Auch auf die Gefahr hin, dass ich Sie langweile: Wir installieren *myphpadmin* und die dazugehörigen *apache*- und *php*-Module.

Code-Ausschnitt 8.2.6

```
1 sudo apt-get install libapache2-mod-auth-pgsql php7.0-mysql phpmyadmin
```

Während der Installation werden Sie nach einem *root*-Kennwort gefragt. Wählen Sie ein sicheres Kennwort und wiederholen Sie die Eingabe.



Bitte merken Sie sich das Kennwort. Wir werden es im Laufe weiterer Projekte häufiger benötigen. Schreiben Sie es sich notfalls irgendwo auf. Ohne Kennwort haben Sie keinen Zugriff mehr auf die Datenbank.

Darüber hinaus werden Sie mit weiteren Fragen konfrontiert. Wählen Sie bitte als Frontend für *phpmyadmin apache2* aus (und nicht *lighthttp*). Weiterhin werden Sie gefragt, ob Datenbanken automatisch erstellt werden sollen. Bejahen Sie das (*YES*). Um Verbindung zur Datenbank zu erhalten, fragt *phpmyadmin* nach dem Datenbank-Kennwort, welches Sie weiter oben vergeben haben (ich habe Ihnen ja gesagt, schreiben Sie es sich auf!). Die letzte Abfrage verlangt ein Kennwort für die *myphpadmin*-Oberfläche. Hier können Sie Ihrer Kreativität wieder freien Lauf lassen und das sogar noch einmal bestätigen. Um *phpmyadmin* mit *apache* zu verknüpfen, muss eine Änderung in der Datei `/etc/php/7.0/apache2/php.ini` vorgenommen werden. Fügen Sie bitte die folgende Zeile ein, sofern sie noch nicht vorhanden ist:

Code-Ausschnitt 8.2.7

```
1 extension=mysql.so
```

Seit *MySQL 5.7* hat sich das Sicherheitsmodell der Datenbank geändert: Der *MySQL*-Root-Zugang per Terminal erfordert jetzt ein vorangestelltes *sudo*. Das bedeutet, dass *phpMyAdmin* den Benutzer *root* nicht erlaubt. Das Problem kann einfach gelöst werden, indem man einen Benutzer-Zugang zur Datenbank erstellt und diesen mit den erforderlichen Privilegien ausstattet. Dazu verbinden wir uns zunächst in einem Terminal mit der Datenbank:

Code-Ausschnitt 8.2.8

```
1 sudo mysql --user=root mysql
```

Nun erstellen wir einen Benutzer für *phpMyAdmin* mit dem Namen *phpmyadmin* und einem sicheren Kennwort.

Code-Ausschnitt 8.2.9

```
1 CREATE USER 'phpmyadmin'@'localhost' IDENTIFIED BY 'kennwort';
2 GRANT ALL PRIVILEGES ON *.* TO 'phpmyadmin'@'localhost' WITH GRANT OPTION;
3 FLUSH PRIVILEGES;
```

Bitte ersetzen Sie *kennwort* im oben gezeigten Quelltext durch ein sicheres Kennwort. Wenn Sie die Datenbank mit Hilfe von *phpMyAdmin* nur auf dem lokalen Rechner verbinden möchten, reichen diese Befehle aus.

Möchten Sie darüber hinaus Remote-Verbindungen zulassen, rufen Sie bitte noch den folgenden Befehl auf:

Code-Ausschnitt 8.2.10

```
1 CREATE USER 'phpmyadmin'@'%' IDENTIFIED BY 'some_pass';
2 GRANT ALL PRIVILEGES ON *.* TO 'phpmyadmin'@'%' WITH GRANT OPTION;
3 FLUSH PRIVILEGES;
```

Denken aber bitte daran, dass ein solcher Zugang von außen ein Sicherheitsrisiko darstellen könnten. Danach können Sie sich mit einem

Code-Ausschnitt 8.2.11

```
1 quit
```

von der Datenbank abmelden. Im letzten Schritt editieren Sie bitte die Datei `/etc/dbconfig-common/phpmyadmin.conf` und fügen Sie den Datenbankbenutzer `phpmyadmin` und das oben vergebene Kennwort (*kennwort*) hinzu, sofern diese Einträge noch nicht vorhanden sind:

Code-Ausschnitt 8.2.12

```
1 # dbc_dbuser: database user
2 #     the name of the user who we will use to connect to the database.
3 dbc_dbuser='phpmyadmin'
4 # dbc_dbpass: database user password
5 #     the password to use with the above username when connecting
6 #     to a database, if one is required
7 dbc_dbpass='kennwort'
```

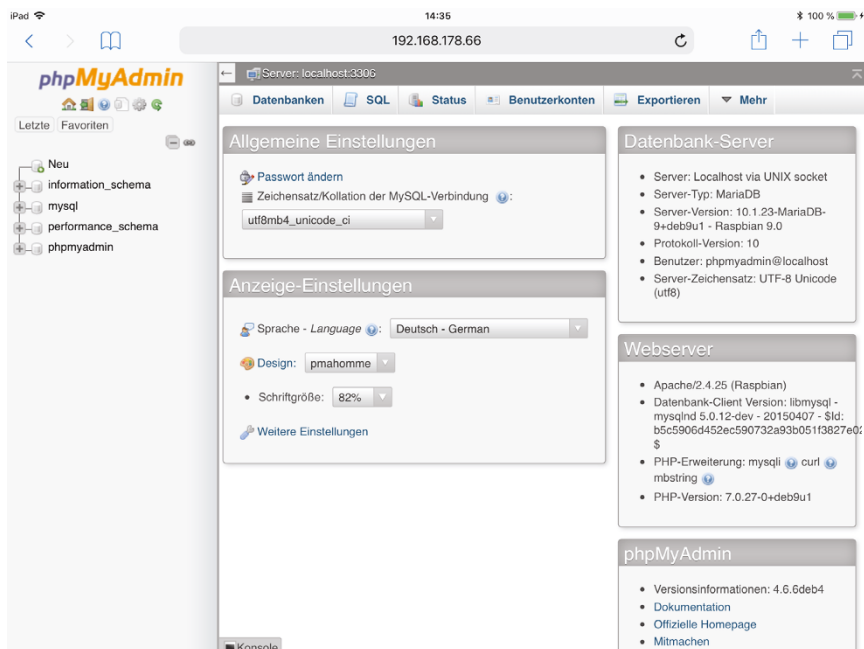


Abbildung 8.7: Die *phpmyadmin*-Oberfläche verwaltet die *MySQL*-Datenbank.

Um die fertige Installation zu testen, rufen Sie bitte `http://ip-adresse/phpmyadmin` in Ihrem Lieblings-Browser auf. Wählen Sie sich mit dem Benutzernamen *phpmyadmin* und dem eben vergebenen Kennwort ein. Daraufhin sollten Sie die *phpmyadmin*-Oberfläche sehen (Abb. 8.7).



Sollte Ihr Browser die *phpmyadmin*-Oberfläche nicht anzeigen, legen Sie bitte den folgenden symbolischen Link an:

Code-Ausschnitt 8.2.13

```
1 sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-enabled/↵
   phpmyadmin.conf
```

Danach muss der *apache*-Dienst neu gestartet werden.

8.3 WLAN-Hotspot

Jeder von uns hat sicher schon einmal einen Hotspot benutzt, um sich per WLAN mit dem Internet zu verbinden. Vielleicht hatten Sie aber auch schon einmal das Problem, dass Ihr Besuch sich per WLAN mit dem Internet verbinden wollte, Sie ihm aber nicht die Zugangsdaten geben wollten, weil Sie damit auch ihr Netzwerk öffnen? Zwar gibt es bereits DSL-Modems, die ein sogenanntes Gäste-WLAN zur Verfügung stellen, aber dennoch wäre es schön, wenn wir schwarz auf weiß speichern könnten, welcher unserer Gäste wann was im Internet getan hat. Der nachfolgende Abschnitt ist für alle interessant, die ihren Raspberry Pi in einen Hotspot mit Protokoll-Funktion und Freigabe verwandeln wollen. Sie lernen, wie Sie Benutzerkennwörter erstellen und vergeben können, die Dauer des Zugangs oder das Download-Volumen begrenzen können und vieles mehr. Der Abschnitt ist in folgende Teile aufgeteilt:

1. Wir erstellen einen Hotspot, den wir mit einem WPA-Schlüssel absichern.
2. Wir schützen den Hotspot weiter, indem wir auf eine Login-Seite weiterleiten und dort die Zugangsdaten des Benutzers abfragen.
3. Wir setzen einen Authentifizierungsserver auf, der eine Eingabe von Benutzernamen und Kennwort mit den Daten einer Datenbank vergleicht und nur erlaubten Personen für die erlaubte Zeit das Netzwerk frei schaltet.
4. Wir programmieren ein einfaches Netzwerk-Interface, über das wir neue Benutzer zulassen oder alte löschen können.
5. Wir benutzen den Hotspot mit *TOR*, einer Methode, um unsere Identität im Netz zu verschleiern.

Der Raspberry Pi wird bei diesem Projekt über ein Ethernetkabel mit dem DSL-Modem verbunden.

8.3.1 Hostapd

Nicht jeder WLAN USB-Stick kann für unser Vorhaben verwendet werden. Während alle am Markt verfügbaren WLAN-Sticks sich als Klient mit einem Hotspot verbinden können, beherrschen nicht alle Sticks den sogenannten *access point*-Modus, den der LINUX-Dienst *hostapd* unterstützt. Jetzt raten Sie, was *hostapd* heißt? *Host* ist der „Gast“. Das *ap* steht für *access point* und *d* sagt, dass es sich um einen Daemon handelt, also einen Dienst, der im Hintergrund läuft.



Abbildung 8.8: Der TP-Link TL-WN722N hat eine große Reichweite dank externer Antenne (Quelle:TP-Link).

Der EDIMAX-Stick, den ich in Kapitel 2.4.3 vorgestellt habe, beherrscht diesen Modus. Eine Alternative mit Antenne ist der *TP-Link TL-WN722N High Gain Wireless-LAN USB*, den der große Internetversender mit dem A im Namen für 14 € anbietet. Dieser Stick eignet sich hervorragend, wenn Sie z. B. eine Ferienwohnung versorgen möchten. Natürlich können Sie für den Hotspot auch den Raspberry Pi-eigenen WLAN-Adapter als Hotspot umfunktionieren.

Für unseren Hotspot bauen wir eine Netzwerkbrücke vom kabelgebundenen Ethernet zum Hotspot. Die dafür nötigen Tools installieren wir wie immer mit:

Code-Ausschnitt 8.3.1

```
1 sudo apt-get install bridge-utils hostapd
```

Früher war das installierte *hostapd* ein wenig veraltet, so dass manche Chipsätze nicht unterstützt wurden. Dazu zählte unter anderem der RTL8188CUS des Edimax. Falls der Chip-satz Ihres WLAN-Sticks unterstützt wird, können Sie sich den nächsten Schritt sparen. In diesem laden wir den Realtek-Treiber herunter und installieren diesen. Den Treiber finden Sie auf <https://github.com/jenssegers/RTL8188-hostapd>. Wir erstellen ein Verzeichnis namens *wlan_hotspot* und den Treiber:

Code-Ausschnitt 8.3.2

```
1 mkdir wlan_hotspot
2 cd wlan_hotspot
3 git clone https://github.com/jenssegers/RTL8188-hostapd
```

Im Verzeichnis *RTL8188-hostapd* finden Sie das Verzeichnis *hostapd*. Wechseln Sie in das angegebene Verzeichnis und rufen Sie *make* auf.

Code-Ausschnitt 8.3.3

```
1 cd RTL8188-hostapd
2 make
```

Das erstellt den *hostapd* mit Unterstützung für den Edimax-Stick. Da unsere *hostapd*-Installation komplett funktioniert, mit Ausnahme der Datei *hostapd*, ersetzen wir die Original-Datei mit unserer.

Code-Ausschnitt 8.3.4

```
1 sudo mv /usr/sbin/hostapd /usr/sbin/hostapd.orig
2 sudo cp hostapd /usr/sbin/hostapd.edimax
3 sudo ln -sf /usr/sbin/hostapd.edimax /usr/sbin/hostapd
4 sudo chown root.root /usr/sbin/hostapd
5 sudo chmod 755 /usr/sbin/hostapd
```

Der Name der Original-Datei wird zunächst geändert in *hostapd.orig*. Dann wird die neue *hostapd*-Datei in das Verzeichnis */usr/sbin* kopiert, allerdings unter dem Namen *hostapd.edimax*. Damit erinnern wir uns daran, dass es ja nicht die Original-Datei, sondern unsere eigene ist. Damit diese Datei aber unter dem Namen *hostapd* gefunden werden kann, setzen wir noch einen symbolischen Link darauf mit diesem Namen. Anschließend ändern wir Besitzer und Gruppe in *root* und statten den Link mit Ausführ-Rechten aus.



Sollten Sie die Datei *hostapd* ausversehen überschrieben haben, können Sie *hostpad* mit dem Befehl

Code-Ausschnitt 8.3.5

```
1 sudo apt-get install hostapd --reinstall
```

wieder neu installieren.

Die nächsten Schritte sind wieder für alle WLAN-Sticks gleich. Erinnern Sie sich noch an das Kapitel, in dem wir statische und dynamische IP-Adressen eingestellt haben? Richtig, wir haben die Datei */etc/network/interfaces* editiert. Das machen wir nun wieder und richten die Netzwerkbrücke ein.

Code-Ausschnitt 8.3.6

```
1 auto br0
2 iface br0 inet dhcp
3 bridge_ports eth0 wlan0
```



Wir werden die Brücke später wieder entfernen, wenn wir einen Login programmieren. Aber für einen ersten Hotspot-Test lassen wir sie stehen.

Jetzt müssen wir noch den *hostapd* konfigurieren. Das erledigen wir in der Datei */etc/hostapd/hostapd.conf*:

Code-Ausschnitt 8.3.7

```
1 #WPA2
2 interface=wlan0
3 driver=rtl871xdrv
4 ssid=pi
5 channel=5
6 #hw_mode=g
7 ieee80211n=1
8 wmm_enabled=0
9 wpa=2
10 wpa_passphrase=raspberry
11 wpa_key_mgmt=WPA-PSK
12 wpa_pairwise=TKIP
13 rsn_pairwise=CCMP
14 auth_algs=1
15 macaddr_acl=0
```

In der *ssid* steht der Name, unter dem der Hotspot sichtbar ist. Sie können hier einen Namen Ihrer Wahl einsetzen. Mit *wpa_passphrase* legen Sie das WPA2-Kennwort fest. Mit dem Eintrag *ieee80211n=1* schalten wir den Stick in den schnellsten Übertragungs-Modus.

Nutzen Sie den Raspberry Pi-eigenen WLAN-Adapter, so füllen Sie die Datei *hostapd* bitte mit folgendem Inhalt:

Code-Ausschnitt 8.3.8

```

1 interface=wlan0
2 # Treiber für Raspberry Pi
3 driver=nl80211
4 ssid=pi
5 # 2.4 GHz band benutzen
6 hw_mode=g
7 # Kanal 6 benutzen
8 channel=6
9 # 802.11n einschalten
10 ieee80211n=1
11 # WMM einschalten
12 wmm_enabled=1
13 # 40 MHz Kanäle mit 20 ns Guard-Interval
14 ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
15 # Alle MAC-Adressen akzeptieren
16 macaddr_acl=0
17 # WPA authentication
18 auth_algs=1
19 # Klienten müssen das Netzwerk nicht kennen
20 ignore_broadcast_ssid=0
21 # WPA2 benutzen
22 wpa=2
23 # Pre-shared Schlüssel benutzen
24 wpa_key_mgmt=WPA-PSK
25 wpa_passphrase=raspberry
26 # AES benutzen und nicht TKIP
27 rsn_pairwise=CCMP

```

Wenn Sie möchten, können Sie den Hotspot jetzt testen. Ein

Code-Ausschnitt 8.3.9

```
1 sudo hostapd -dd /etc/hostapd/hostapd.conf
```

startet den *hostapd* in der Konsole. Der *hostapd* startet nicht allein während des Boot-Vorgangs, ohne dass sein DEFAULT auf *enable* gesetzt wird. Entfernen Sie zu diesem Zweck in der Datei */etc/default/hostapd* das Kommentarzeichen vor der folgenden Zeile:

Code-Ausschnitt 8.3.10

```
1 DAEMON_CONF="/etc/hostapd/hostapd.conf"
```



Wenn Sie einen offenen Hotspot einrichten wollen, ändern sie die Datei */etc/hostapd/hostapd.conf* so:

Code-Ausschnitt 8.3.11

```

1 interface=wlan0
2 driver=rtl871xdrv
3 ssid=pi
4 channel=5
5 #hw_mode=g
6 ieee80211n=1
7 auth_algs=1
8 wmm_enabled=0

```

Den Kanal, auf dem Sie den Hotspot bereitstellen, können Sie hinter dem Eintrag *channel* eintragen. Andere Chipsätze nutzen andere Treiber. Die Seite <http://elinux.org/RPI-Wireless-Hotspot> hält weitere Informationen darüber bereit.

Wir entfernen die Netzwerkbrücke aber jetzt wieder und werden dynamische IP-Adressen per *DHCP* (*Dynamic Host Configuration Protocol*) vergeben, da wir dieses Verfahren später noch für unseren Authentifizierungs-Server benötigen. Dazu installieren wir zunächst *dhcp*.

Code-Ausschnitt 8.3.12

```
1 sudo apt-get install udhcpd
```

Die Konfigurationsdatei für *udhcpd* */etc/udhcpd.conf* ändern wir so:

Code-Ausschnitt 8.3.13

```
1 start 192.168.179.2 # Start und Ende des IP-Bereichs, den der Hotspot für Klienten ←
   vergibt
2 end 192.168.179.20 # Wir können also 19 Klienten (2-20) gleichzeitig zulassen
3 interface wlan0 # Der DHCP-Server gilt für das WLAN-Interface
4 remaining yes
5 opt dns 8.8.8.8 4.2.2.2 # Wir nutzen Googles öffentliche Nameserver (DNS)
6 opt subnet 255.255.255.0 # Die Subnetz-Maske. Wir haben nicht mehr als 256 Rechner
7 opt router 192.168.179.1 # Die WLAN-Adresse des Raspberry Pi, die wir gleich setzen
8 opt lease 864000 # 10 day DHCP lease time in seconds
```

Wir binden den *DHCP*-Dienst dauerhaft ein, indem wir in der Datei */etc/default/udhcpd* folgende Änderung machen:

Code-Ausschnitt 8.3.14

```
1 #DHCPD_ENABLED="no"
```

Wir kommentieren aus, dass der *DHCP*-Dienst nicht läuft - er wird damit also laufen. Nun müssen wir dem Netzwerk-Interface *wlan0* noch mitteilen, dass es ab sofort eine statische IP-Adresse hat. Das erledigen wir in der Datei */etc/network/interfaces*:

Code-Ausschnitt 8.3.15

```
1 auto lo
3 iface lo inet loopback
4 iface eth0 inet dhcp
6 iface wlan0 inet static
7   address 192.168.179.1
8   netmask 255.255.255.0
```

Falls die Zeilen

Code-Ausschnitt 8.3.16

```
1 allow-hotplug wlan0
2 wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
3 iface default inet manual
```

noch vorhanden sind, kommentieren Sie sie bitte aus:

Code-Ausschnitt 8.3.17

```
1 #allow-hotplug wlan0
2 #wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
3 #iface default inet manual
```

Damit der gesamte Netzverkehr unserer Hotspot-Klienten über die Netzwerk-Verbindung unseres Raspberry Pi geroutet wird, müssen wir *NAT (Network Address Translation)* einrichten, um das zu ermöglichen. Hierzu teilen wir dem Kernel mit, dass die IP-Weiterleitung eingeschaltet werden soll.

Code-Ausschnitt 8.3.18

```
1 sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

Damit die Weiterleitung auch nach einem Neustart erhalten bleibt, schreiben wir an das Ende der Datei */etc/sysctl.conf*

Code-Ausschnitt 8.3.19

```
1 net.ipv4.ip_forward=1
```

Um *NAT* zu aktivieren, helfen die folgenden Befehle:

Code-Ausschnitt 8.3.20

```
1 sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
2 sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
3 sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

Wir speichern diese Einstellungen in der Datei */etc/iptables.ipv4.nat*

Code-Ausschnitt 8.3.21

```
1 sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

und machen auch diese Änderung *reboot*-fest, indem wir an das Ende der Datei */etc/network/interfaces* schreiben:

Code-Ausschnitt 8.3.22

```
1 up iptables-restore < /etc/iptables.ipv4.nat
```

Testen Sie die Änderungen nach einem

Code-Ausschnitt 8.3.23

```
1 sudo service hostapd start
2 sudo service udhcpd start
```

Soll der Hotspot nach einem Neustart automatisch gestartet werden, binden Sie den *hostapd*- und den *udhcpd*-Dienst in den Systemstart ein.

Code-Ausschnitt 8.3.24

```
1 sudo update-rc.d hostapd enable
2 sudo update-rc.d udhcpd enable
```


8.3.2 Chillispot

Chillispot ist eine Software, welche ein sogenanntes *captive portal* zur Verfügung stellt. Hierunter versteht man eine Seite, auf die Besucher Ihres Hotspots umgeleitet werden, um sich zu authentifizieren. Dieses Portal kann bestimmte Internetseiten freigeben, ohne dass man eingeloggt sein muss. Vielleicht kennen Sie das von Flughäfen oder anderen Hotspots. In diesem Abschnitt werden wir das Portal installieren und dann mit einem Authentifizierungs-Server verbinden. *Chillispot* gehört leider nicht zum Installationsumfang der Raspbian-Distribution. Das macht aber nichts, wir besorgen es uns von der *Chillispot*-Webseite www.chillispot.org. Sicher werden Sie merken, dass die letzte Version des Programmes aus dem Jahre 2006 stammt. Trotzdem spielt sie hervorragend mit einem Kernel aus dem Jahr 2014 zusammen. Der Quelltext ist unter <http://www.chillispot.org/download/chillispot-1.1.0.tar.gz> verfügbar. Erstellen wir also ein Verzeichnis, laden den Quelltext herunter und packen ihn aus.



Abbildung 8.9:
(Quelle:
chillispot.org)

Code-Ausschnitt 8.3.25

```
1 cd ~/wlan_hotspot
2 mkdir chillispot
3 wget http://www.chillispot.org/download/chillispot-1.1.0.tar.gz
4 tar xvzf chillispot-1.1.0.tar.gz
```

Bevor es an das Übersetzen geht, müssen noch zwei Zeilen in der Datei `src/chilli.c` geändert werden. Diese Änderungen sind erforderlich, damit wir in einem späteren Schritt MAC-Adressen festlegen können, die sich ohne Authentifizierung am Hotspot anmelden dürfen.

Info *MAC* steht für *Media Access Control* und bezeichnet die Hardware-Adresse eines Netzwerk-Adapters, die als eindeutige Identifizierung eines Netzwerk-Gerätes im Netz gilt. Wir werden die *MAC*-Adressen der Besucher unseres Hotspots später in einer Datenbank speichern.

Fügen Sie bitte in Zeile 527 ein `return 0;` vor Ende der Funktion ein:

Code-Ausschnitt 8.3.26

```
1     else {
2         p1 = NULL;
3     }
4 }
5 free(p3);
6 return 0;
```

Ändern Sie bitte Zeile 2991 so:

Code-Ausschnitt 8.3.27

```
1     if (!radius_getattr(pack, &stateattr, RADIUS_ATTR_SERVICE_TYPE, 0, 0, 0))
2     {
3         /* if(ntohl(attr->v.i) == RADIUS_SERVICE_TYPE_CHILLISPOT_AUTHORIZE_ONLY) { */
4         if(ntohl(stateattr->v.i) == RADIUS_SERVICE_TYPE_CHILLISPOT_AUTHORIZE_ONLY) {
```

Danach kann das Übersetzen beginnen:

Code-Ausschnitt 8.3.28

```

1 ./bootstrap
2 ./configure
3 make
4 sudo make install

```

Chillispot bringt eine Beispieldatei *hotspotlogin.cgi* für eine Startseite mit Authentifizierung mit. Diese Datei kopieren wir nach */usr/lib/cgi-bin* und machen sie ausführbar.

Code-Ausschnitt 8.3.29

```

1 sudo cp doc/hotspotlogin.cgi /usr/lib/cgi-bin
2 sudo chmod 755 /usr/lib/cgi-bin/hotspotlogin.cgi

```

In dieser Datei stellen wir ein neues Kennwort ein, welches der *chillispot*-Server später verwendet. Das ursprüngliche Kennwort kommentieren wir aus:

Code-Ausschnitt 8.3.30

```

1 # Shared secret used to encrypt challenge with. Prevents dictionary attacks.
2 # You should change this to your own shared secret.
3 # $uamsecret = "ht2eb8ej6s4et3rg1ulp";
4 $uamsecret = "chillikennwort";

```

Ich habe das Kennwort *chillikennwort* genannt, bitte denken Sie sich etwas Komplizierteres aus. Als Nächstes passen wir die Konfigurationsdatei für *chillispot* an. Diese nennt sich */etc/chilli.conf*. Nachfolgend habe ich lediglich meine Änderungen aufgeschrieben. Alles andere ist auskommentiert. Wir werden diese Datei später noch einmal anfassen, um die Daten für den Authentifizierungs-Server anzupassen.

Code-Ausschnitt 8.3.31

```

1 dhcpif wlan0
2 uamserver https://192.168.179.1/cgi-bin/hotspotlogin.cgi
3 uamsecret chillikennwort

```

Tragen Sie unter *uamsecret* dasselbe Kennwort ein, welches Sie auch in */etc/chilli.conf* vergeben haben. Wir konfigurieren weiterhin, dass unser WLAN-Interface ein DHCP-Interface (*Dynamic Host Configuration Protocol*) ist (*dhcpif wlan0*), wir also dynamische IP-Adressen für die Gäste unseres Hotspots vergeben werden. Den IP-Adressraum haben wir bereits im vorigen Abschnitt für das WLAN-Interface eingestellt. Zum Start des *chillispot*-Servers legen wir eine Datei */etc/default/chilli* mit folgendem Inhalt an:

Code-Ausschnitt 8.3.32

```

1 START_CHILLI=1
2 CONFFILE="/etc/chilli.conf"
3 HS_USER="chilli"

```

Danach richten wir den Dienst unter */etc/init.d/chillispot* ein.

Code-Ausschnitt 8.3.33

```

1  #! /bin/sh
2  ### BEGIN INIT INFO
3  # Provides:          chilli
4  # Default-Start:    2 3 4 5
5  # Default-Stop:     1
6  # Short-Description: chillisport server
7  ### END INIT INFO
8  PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin
9  DAEMON=/usr/local/sbin/chilli
10 NAME=chilli
11 DESC=chilli
12 test -f $DAEMON || exit 0
13 set -e
14 case "$1" in
15     start)
16         echo -n "Starting $DESC: "
17         start-stop-daemon --start --quiet --pidfile /var/run/$NAME.pid --exec $DAEMON
18         echo "$NAME."
19         ;;
20     stop)
21         echo -n "Stopping $DESC: "
22         start-stop-daemon --oknodo --stop --quiet --pidfile /var/run/$NAME.pid --exec $DAEMON
23         echo "$NAME."
24         ;;
25     restart|force-reload)
26         echo -n "Restarting $DESC: "
27         start-stop-daemon --stop --quiet --pidfile /var/run/$NAME.pid --exec $DAEMON
28         sleep 1
29         start-stop-daemon --start --quiet --pidfile /var/run/$NAME.pid --exec $DAEMON
30         echo "$NAME."
31         ;;
32     *)
33         N=/etc/init.d/$NAME
34         echo "Usage: $N {start|stop|restart|force-reload}" >&2
35         exit 1
36         ;;
37     esac
38     exit 0

```

Der Kontakt zu *hotspotlogin.cgi* setzt eine *https*-Verbindung voraus. Die werden wir nun einrichten, bevor wir *chillispot* das erste Mal aufrufen.

8.3.3 Zertifikate

Normaler Internetverkehr wird unverschlüsselt übertragen. Das bedeutet, dass jeder, der über entsprechende Programme verfügt, diesen Verkehr mitlesen kann. Spätestens bei der Übertragung persönlicher Daten wie Kontonummern o. Ä. ist das kein Spaß mehr. Der sog. *Secure Socket Layer (SSL)* verschlüsselt den Datenverkehr zwischen Web-Server und dem Web-Browser. *SSL* nutzt asymmetrische Kryptographie. Dabei werden zwei Schlüssel generiert, ein öffentlicher und ein privater. Alles, was mit einem der Schlüssel verschlüsselt worden ist, kann nur mit dem jeweils anderen Schlüssel entschlüsselt werden. Wenn Daten Ihres Bank-Servers mit dem privaten Schlüssel der Bank verschlüsselt worden sind, können sie nur mit dem passenden öffentlichen Schlüssel entschlüsselt werden. Das stellt sicher, dass die Daten auch wirklich vom richtigen Server - nämlich dem Bank-Server - kommen und nicht von einem Server, der versucht, Ihre Daten abzugreifen. Bleibt die Frage, warum ein Zertifikat erforderlich ist, wenn der Internetverkehr doch durch die beiden Schlüssel abgesichert werden kann? Aus technischer Sicht ist ein Zertifikat nicht wirklich notwendig, da die Daten aufgrund der zwei Schlüssel sicher sind und nicht von einem Angreifer einfach dekodiert werden können. Dennoch spielen Zertifikate eine wichtige Rolle in Kommunikationsprozessen. Signierte Zertifikate sollen zusätzlich sicherstellen, dass der Inhaber des Zertifikates auch der ist, für den er sich ausgibt.

Das gibt eine zusätzliche Sicherheit, dass auch wirklich derjenige am anderen Ende der Leitung ist, den Sie erwarten - etwa Ihre Bank. Installieren Sie also das Programm, mit dem Zertifikate ausgestellt werden können.

Code-Ausschnitt 8.3.34

```
1 sudo apt-get install ssl-cert
```

Wir erstellen im Konfigurations-Verzeichnis */etc/apache2* ein Unterverzeichnis für *ssl*-Zertifikate (*Secure Socket Layer*). *SSL* ist ein Protokoll zur sicheren Übertragung von Daten (z. B. über *https*).

Code-Ausschnitt 8.3.35

```
1 sudo mkdir /etc/apache2/ssl
2 sudo make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/apache2/ssl/apache.pem
```

Beim Erstellen des Zertifikates mit *make-ssl-cert* müssen Sie den Rechnernamen sowie alternative Namen eingeben. Sie können den Namen auf *localhost* stehen lassen und die alternativen Namen freilassen. Aktivieren Sie zuerst den *ssl*-Port 443 für *apache* und danach *ssl*. Ist das fertig, muss *apache* neu gestartet werden. Kommentieren Sie also in der Datei */etc/apach2/ports.conf* alle Zeilen mit *Listen 443* aus.

Code-Ausschnitt 8.3.36

```
1 NameVirtualHost *:80
2 Listen 80
3 <IfModule mod_ssl.c>
4     # If you add NameVirtualHost *:443 here, you will also have to change
5     # the VirtualHost statement in /etc/apache2/sites-available/default-ssl
6     # to <VirtualHost *:443>
7     # Server Name Indication for SSL named virtual hosts is currently not
8     # supported by MSIE on Windows XP.
9     Listen 443
10 </IfModule>
11
12 <IfModule mod_gnutls.c>
13     Listen 443
14 </IfModule>
```

Die Aktivierung des *ssl*-Moduls erreichen Sie mit

Code-Ausschnitt 8.3.37

```
1 sudo a2enmod ssl
2 systemctl restart apache2
```



Statt

Code-Ausschnitt 8.3.38

```
1 /etc/init.d/apache2 restart
```

können Sie auch

Code-Ausschnitt 8.3.39

```
1 systemctl restart apache2
```

ausführen.

Im nächsten Schritt müssen wir einen sicheren *Host* erstellen. Ein *Host* ist dasselbe wie ein Server - also ein System, welches über das Netzwerk Dienste bereit stellt. Dazu kopieren wir einfach die Standard Host-Konfiguration unter einem neuen Namen (*secure*).

Code-Ausschnitt 8.3.40

```
1 cp /etc/apache2/sites-available/default-ssl /etc/apache2/sites-available/secure-ssl
```

In dieser gerade erstellten Datei müssen wir noch unser Zertifikat von oben mitteilen. Ändern Sie dazu bitte die *SSLCertificate*-Sektion so:

Code-Ausschnitt 8.3.41

```
1 #SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
2 SSLCertificateFile /etc/apache2/ssl/apache.pem
3 #SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
```

Starten Sie dann *apache* neu mit dem Kommando */etc/init.d/apache2 restart*. Zu diesem Zeitpunkt können Sie *chillispot* bereits testen. Starten Sie ihn mit

Code-Ausschnitt 8.3.42

```
1 sudo /etc/init.d/chillispot start
```



Abbildung 8.10: *Chillispot*-Login auf einem Telefon.

Wählen Sie mit einem WLAN-fähigen Gerät unter der Liste der verfügbaren WLAN-Netze Ihren Hotspot aus. Sie haben dafür einen Namen (*ssid*) vergeben, evtl. *pi*. Falls Sie eine WPA2-Verschlüsselung angegeben haben, werden Sie aufgefordert, das Kennwort dafür einzugeben.

Danach werden Sie automatisch auf die neue *chillispot* Login-Seite weitergeleitet (Abb. 8.10). Der Login selbst läuft vorerst ins Leere, da wir noch die Authentifizierung einarbeiten müssen.

8.3.4 Radius

Nach so viel Vorarbeit richten wir im letzten Schritt den Authentifizierungs-Server ein. Der wird in einer Datenbank nachschauen, ob der Gast unseres Hotspots überhaupt eine Berechtigung hat, diesen zu benutzen oder nicht. Der Server, der das übernimmt, heißt *freeradius*. Wir installieren ihn:

Code-Ausschnitt 8.3.43

```
1 sudo apt-get install freeradius freeradius-mysql
```

Als Nächstes legen wir unsere erste MySQL-Datenbank im Terminal an. *freeradius* wird sich aus ihr alle Datensätze besorgen, die zur Zugangskontrolle benötigt werden. Wir melden uns bei MySQL an.

Code-Ausschnitt 8.3.44

```
1 mysql -u root -p
```

Geben Sie Ihr Kennwort ein, welches Sie beim Einrichten von MySQL vergeben haben. Dann legen Sie eine neue Datenbank namens *radius* an und verlassen MySQL wieder.

Code-Ausschnitt 8.3.45

```
1 CREATE DATABASE radius;  
2 quit
```

Wir melden uns erneut bei MySQL an, diesmal aber direkt bei der neu eingerichteten Datenbank *radius*.

Code-Ausschnitt 8.3.46

```
1 mysql -u root -p radius
```

Wir räumen dem Benutzer *radius* alle Rechte zum Bearbeiten der Datenbank *radius* ein. Danach verlassen wir die Datenbank *radius* wieder.

Code-Ausschnitt 8.3.47

```
1 GRANT ALL PRIVILEGES ON radius.* TO 'radius'@'localhost' IDENTIFIED BY 'radiuskennwort';  
2 FLUSH PRIVILEGES;  
3 quit
```

Vergeben Sie dabei ein neues Kennwort, mit dem der Benutzer *radius* auf die Datenbank mit demselben Namen zugreifen darf. Der Authentifizierungs-Server *freeradius* bringt Vorlagen mit, die wir nun in die Datenbank importieren.

Code-Ausschnitt 8.3.48

```
1 mysql -u root -p radius < /etc/freeradius/sql/mysql/schema.sql  
2 mysql -u root -p radius < /etc/freeradius/sql/mysql/nas.sql
```

Geben Sie bitte nach Aufforderung das Kennwort für den Benutzer *radius* ein, welches Sie eben festgelegt haben.



Wenn Sie wollen, öffnen Sie das Web-Frontend *phpmyadmin* und werfen Sie einen Blick in die Datenbank *radius*, die wir gerade angelegt haben.

Im nächsten Schritt passen wir einige Einstellungen in der Datei */etc/freeradius/radiusd.conf* an. Besorgen Sie sich dazu *root*-Rechte, um Zugriff zu erhalten.

Code-Ausschnitt 8.3.49

```
1 sudo su
```

Kommentieren Sie *sql.conf* und *counter.conf* ein und fügen Sie in der *instantiate*-Sektion die Begriffe *expiration* und *logintime* dazu. Aktivieren Sie auch die virtuellen Server. Hiermit werden wir später unsere Zugangskontrolle gestalten.

Code-Ausschnitt 8.3.50

```
1 ...
2 $INCLUDE sql.conf
3 ...
4 $INCLUDE sql/mysql/counter.conf
5 ...
6 instantiate {
7     exec
8     expr
9     expiration
10    logintime
11 }
12 ...
13 $INCLUDE sites-enabled/
```

In der Datei */etc/freeradius/sql.conf* tragen wir die Verbindungseinstellungen zur Datenbank ein.

Code-Ausschnitt 8.3.51

```
1 sql {
2     database = "mysql"
3     driver = "rlm_sql_${database}"
4     server = "localhost"
5     #port = 3306
6     login = "radius"
7     password = "radiuskennwort"
8     radius_db = "radius"
9     ...
```

Bitte öffnen Sie im nächsten Schritt die Datei */etc/freeradius/clients.conf* und stellen Sie sicher, dass sie die folgenden Inhalte enthält. Sie müssen sich dafür wieder mit *sudo su* Administratorrechte besorgen.

Code-Ausschnitt 8.3.52

```
1 client localhost {
2     ipaddr = 127.0.0.1
3     secret = radiuskennwort
4     require_message_authenticator = no
5     nastype = other
6 }
```

In dieser Datei steht das Kennwort des Radius-Servers. Die IP-Adresse habe ich auf 127.0.0.1 gesetzt.



Unter LINUX ist die IP-Adresse 127.0.0.1 immer der lokale Host, also unser Raspberry Pi, auf dem wir den Hotspot installieren möchten. 127.0.0.1 und *localhost* sind gleichbedeutend.

Jetzt ist es an der Zeit, einen Testbenutzer zuzulassen, der sich in den Hotspot einwählen kann. Die Datenbankeinträge erledigen wir danach. Öffnen Sie zu diesem Zweck wieder als Administrator *sudo su* die Datei */etc/freeradius/users* und kommentieren Sie die folgende Zeile ein:

Code-Ausschnitt 8.3.53

```
1  steve  Auth-Type := Local, User-Password == "testing"
```

Das ermöglicht Ihnen, sich mit dem Benutzer *steve* und dem Kennwort *testing* am Hotspot anzumelden. Starten Sie für den Test den Radius-Server und *chillispot* neu:

Code-Ausschnitt 8.3.54

```
1  sudo /etc/init.d/chillispot restart
2  sudo /etc/init.d/freeradius restart
```



Die Wahrscheinlichkeit, bei der Einrichtung des Radius-Severs einen Fehler zu machen, ist recht hoch. Bevor Sie den Dienst *freeradius* starten, können Sie den Radius-Server manuell aufrufen:

Code-Ausschnitt 8.3.55

```
1  sudo freeradius -XXX
```

Erscheint die Meldung

Code-Ausschnitt 8.3.56

```
1  Info: Ready to process requests.
```

können Sie diese mit *CTRL-c* abbrechen und den Dienst wie gewohnt starten. Der o. g. Aufruf hat den Vorteil, dass mögliche Fehlermeldungen im Terminal angezeigt werden. Erscheint eine Fehlermeldung, können Sie *freeradius* jederzeit durch den Aufruf von

Code-Ausschnitt 8.3.57

```
1  sudo killall freeradius
```

beenden.

Nun binden wir die Datenbank ein. Kommentieren Sie dazu den Benutzer *steve* wieder ein, damit hier keine Sicherheitslücke entsteht. Öffnen Sie bitte die Datei */etc/freeradius/sites-available/default* und passen Sie sie so an, wie auf der nächsten Seite beschrieben wird. Wichtig dabei ist, den Kommentar bei *sql* zu entfernen. Zusätzlich müssen noch die Zeilen mit *sql* unter der Sektion *accounting* sowie *session* einkommentiert (also von den Kommentarzeichen befreit) werden. Am Ende werden einige *Zählvariablen* definiert. Der *expire_on_login*-Zähler setzt beispielsweise den Zugang des Benutzers (nach dem ersten Login) nach einer definierten Zeit zurück.

Code-Ausschnitt 8.3.58

```
1  authorize {
2      preprocess
3      chap
4      mschap
5      digest
6      suffix
7      eap {
8          ok = return
9      }
10     files
11     sql
12     expiration
13     logintime
14     pap
15     expire_on_login # Rücksetzen nach einer vorgegebenen Zeit nach Login
16     timelimit
17 }
18 authenticate {
19     Auth-Type PAP {
20         pap
21     }
22     Auth-Type CHAP {
23         chap
24     }
25     Auth-Type MS-CHAP {
26         mschap
27     }
28     digest
29     unix
30     eap
31 }
32 preacct {
33     preprocess
34     acct_unique
35     files
36 }
37 accounting {
38     unix
39     radutmp
40     sql
41     exec
42 }
43 session {
44     radutmp
45     sql
46 }
47 post-auth {
48     exec
49     Post-Auth-Type REJECT {
50         attr_filter.access_reject
51     }
52 }
53 pre-proxy {
54 }
55 post-proxy {
56     eap
57 }
```

Selbst definierte Zähler haben ihr Zuhause im Verzeichnis `/etc/freeradius/modules/`. Der Dateiname beginnt mit `sqlcounter_`, gefolgt vom Namen des Zählers. Wir benutzen hier zwei Zähler: `sqlcounter_expire_on_login` und `sqlcounter_timelimit`. Die Datei `sqlcounter_timelimit` sieht so aus:

Code-Ausschnitt 8.3.59

```

1  sqlcounter timelimit {
2  counter-name = Max-All-Session-Time
3  check-name = Max-All-Session
4  sqlmod-inst = sql
5  key = User-Name
6  reset = never
7  query = "SELECT SUM(AcctSessionTime) FROM radacct where UserName='%{<
8  }

```

Die Datei `sqlcounter_expire_on_login` habe ich so angepasst:

Code-Ausschnitt 8.3.60

```

1  sqlcounter expire_on_login {
2  counter-name = Expire-After-Initial-Login
3  check-name = Expire-After
4  counter-attribute = Acct-Status-Type
5  sqlmod-inst = sql
6  key = User-Name
7  reset = never
8  query = "SELECT TIME_TO_SEC(TIMEDIFF(NOW(), acctstarttime)) FROM radacct WHERE UserName←
          ='%{<
9  }

```

Beide Dateien enthalten MySQL Datenbank-Abfragen. `timelimit` beschränkt den Nutzerzugang auf eine maximale Gesamtzeit (z. B. acht Stunden in Summe, unabhängig davon, wann der Benutzer sich einwählt). `expire_on_login` legt einen Zeitraum fest, innerhalb dessen der Benutzer den Hotspot überhaupt benutzen darf (z. B. eine Gültigkeit für drei Tage ab einem bestimmten Tag). Die Datei `counter.conf`, die wir eben einkommentiert haben, enthält einen Zähler namens `noresetcounter`, den wir nun für einen ersten Test für die Datenbank-Authentifizierung benutzen werden. Dieser Zähler läuft sofort nach dem Anlegen eines Benutzers ab. Für unseren Test stellen wir den Zähler auf fünf Minuten ein. Innerhalb der ersten fünf Minuten nach Anlegen darf sich der Benutzer in den Hotspot einwählen, danach wird er nicht mehr zugelassen. Ist der Benutzer während der ersten fünf Minuten im Hotspot eingewählt, wird er sogar nach Ablauf der Zeit getrennt. Bevor wir gleich ein Web-basiertes Benutzer-Interface realisieren, bemühen wir hier die Konsole zum Anlegen des Benutzers. Wir wählen uns (mit Eingabe des Kennwortes) in die `radius`-Datenbank ein. Dann legen wir einen Benutzer namens `pi` an und geben ihm das Kennwort `raspberrry`. Anschließend setzen wir die maximal erlaubte Zeit (`noresetcounter` fragt `Max-All-Session` ab) auf 300 Sekunden, also fünf Minuten.

Code-Ausschnitt 8.3.61

```

1  mysql -u root -p radius
2  INSERT INTO radcheck VALUES ('','pi','Password',':','=','raspberrry');
3  INSERT INTO radcheck VALUES ('','pi','Max-All-Session',':','=','300');
4  quit

```



Würden Sie die dritte Zeile (`Max-All-Session`) weglassen, hätten Sie einen dauerhaften Zugang für den Benutzer `pi` geschaffen. Eine andere Möglichkeit, einen dauerhaften Zugang zu schaffen, besteht darin, die MAC-Adresse des erlaubten Gerätes zu speichern. Öffnen Sie dazu die Datei `/etc/freeradius/users` als `root` und machen Sie am Ende der Datei folgenden Eintrag:

Code-Ausschnitt 8.3.62

```

1 # Erlaubte MAC-Adresse
2 "58-55-CA-F8-65-27" Cleartext-Password := "password"
3 Service-Type = Framed-User

```

Weiterhin müssen Sie dieselbe IP-Adresse dem *chillispot*-Server bekannt machen. Ändern Sie in der Datei */etc/chilli.conf* folgende Einträge in der Rubrik *MAC authentication*:

Code-Ausschnitt 8.3.63

```

1 macallowed 58-55-CA-F8-65-27
2 macpasswd password

```

Der Chillispot-Server weiß dann, dass die Beispiel-MAC-Adresse (58-55-CA-F8-65-27) sich mit einem Kennwort anmelden darf. Radius setzt, wenn die MAC-Adresse 58-55-CA-F8-65-27 auftaucht, das Kennwort auf „password“. Dieses Kennwort ist in der Datenbank als *erlaubt* hinterlegt. Mehrere MAC-Adressen können in der Datei *chilli.conf* mit einem Komma getrennt hintereinander angegeben werden. In der *users*-Datei benötigen verschiedene MAC-Adressen jeweils separate Einträge.

DL Im Download-Bereich ist der Hotspot eingerichtet, wie hier beschrieben. *hostapd* ist für den EDIMAX WLAN-Stick ausgelegt, kann aber auf andere Sticks angepasst werden. Die Radius-Datenbank ist ebenfalls enthalten (*sql*). Importieren Sie diese Datei in eine Datenbank namens *radius* (Kapitel 12.7). Alle Benutzer und Kennworte sind auf den Namen *pi* mit dem Kennwort *raspberry* ausgelegt.

Vielleicht haben Sie in einem Hotel schon einmal miterlebt, wie die nette Dame am Empfang Ihnen einen Zettel mit den WLAN-Zugangsdaten ausgedruckt hat. Auf diesem Zettel steht dann meistens ein Benutzername und ein Kennwort, welches dann für die Dauer Ihres Aufenthaltes gültig ist. Ihre Daten werden dabei in eine Datenbank übernommen - *Radius* eben. In diesem Abschnitt stelle ich Ihnen ein Web-basiertes Interface vor, mit dem Sie Benutzer Ihres Hotspots anlegen, löschen und verwalten können. Sogar das Erzeugen eines pdf-Dokumentes mit den Zugangsdaten ist möglich. Die hier beschriebene Vorgehensweise entspricht in weiten Teilen dem unter <http://www.imrazor.de/wordpress/?p=29> beschriebenen Vorgang. Zum Web-Interface haben maßgeblich Michael Krahn und Andreas Reichert beigetragen. Alle Skripte können unter http://www.imrazor.de/wordpress/?attachment_id=257 heruntergeladen werden. Diese Skripte gehen davon aus, dass die *Radius*-Datenbank zunächst um einen Eintrag erweitert wird, nämlich um die *Firma* unseres Hotspot-Besuchers. Wählen Sie sich also in die *Radius*-Datenbank ein und ergänzen Sie die Tabelle *firma*:

Code-Ausschnitt 8.3.64

```

1 mysql -u root -p radius
2 ALTER TABLE firma
3 ADD FOREIGN KEY (id)
4 REFERENCES radcheck(id);
5 quit

```

Die Tabellen *radcheck* und *firma* werden dabei über die *id* miteinander verbunden. Ich habe eine Web-Oberfläche wie in Abb. 8.11 umgesetzt. Der Autor der Skripte erstellt mit einem Zufallsgenerator einen Benutzernamen und ein Kennwort, bevor die Daten in die *radius*-Datenbank eingetragen werden. Im Anschluss daran wird eine pdf-Datei generiert und geöffnet, die alle Zugangsdaten enthält. Der Zufallsgenerator erzeugt sowohl einen kurzen Benutzernamen als auch ein kurzes Kennwort, damit die Eingabe nicht unnötig erschwert wird.

<p>LOGIN FÜR PI-WLAN ANLEGEN</p> <p>Bitte füllen Sie die unteren Felder aus.</p> <p>Firma: <input type="text"/></p> <p>Name: <input type="text"/></p> <p>Vorname: <input type="text"/></p> <p>Tag(e) Gültigkeit: <input type="text"/></p> <p>USER ANLEGEN/PDF GENERIEREN</p>	<p>LISTE ALLER ACCOUNTS</p> <p>Eine Liste aller Accounts finden Sie → hier</p>
--	---

Abbildung 8.11: Ein Web-Interface zum Anlegen von Hotspot-Nutzern.

Sie können den Generator natürlich nach eigenem Gusto verändern. Um eine MySQL-Injection mit böartigem Code zu verhindern, stellt PHP eine Funktion bereit, die alle Felder, welche Strings enthalten, unschädlich macht. Die Funktion heißt *mysql_real_escape_string*. Die Funktion *mysql_query* kümmert sich dann um die Datenbank. Für die Erstellung des PDF-Dokumentes wird die PHP-Klasse *tcpdf* verwendet, die Sie von <http://www.tcpdf.org> herunterladen können.

DL Es würde den Rahmen dieses Buches sprengen, alle Skripte abzudrucken. Meinen angepassten Code finden Sie im Download-Bereich <https://www.springer.com/de/book/978-3-662-58143-8> unter dem Namen *www_p.tar.bz2*. Kopieren Sie das Verzeichnis nach */var/www* und rufen Sie das Interface mit <https://ip-adresse/p> auf. Ich habe dem Interface ebenfalls ein Login vorangestellt, welches die Datenbank abfragt. Schauen Sie sich im Quelltext um, wie das funktioniert. Die Datenbank-Dateien sind ebenfalls enthalten.

Ein fertiges PDF-Dokument sieht so aus wie in Abb. 8.12:

WLAN-Gastzugang

Zur Benutzung des Gäste WLANs bitte die nachfolgende Punkte durchführen:

1. Verbinden sie sich mit dem WLAN der SSID **pi**.
2. Rufen sie mit dem Browser eine beliebige Internetseite auf, sie werden dann zu einem Login-Fenster umgeleitet. Als Login benutzen sie nachfolgende Daten:

Benutzername: **ysn3o**
Passwort: **idx51bMy**
3. Ihr Zugang ist für folgenden Zeitraum gültig: **8 Tag(e)**

DISCLAIMER

Eine Verfügbarkeit oder Störungsfreiheit kann nicht garantiert werden!
Der Benutzer verpflichtet sich mit dem Login, gültige Gesetze zu beachten.
Die Benutzerinformationen werden für einen Zeitraum von 3 Monaten gespeichert.

Abbildung 8.12: Ein PDF-Dokument zum Hotspot-Zugang.

Wenn Sie eine Übersicht der Verbindungen eines bestimmten Benutzers benötigen, hilft Ihnen die Eingabe von

Code-Ausschnitt 8.3.65

```
1 mysql -uradiusbenutzer -pradiuskennwort radius -e 'select AcctUniqueId,acctstarttime,↵
    acctstoptime,callingstationid,FramedIPAddress from radacct where username="↵
    benutzername"'
```

Ersetzen Sie bitte *radiusbenutzer* und *radiuskennwort* mit dem Benutzer der *radius*-Datenbank und dessen Kennwort. Ersetzen Sie bitte *benutzername* durch den Namen des Benutzers, dessen Internetaufenthalte Sie sich anschauen möchten. Mit den folgenden beiden Befehlen können Sie übrigens bequem *MySQL*-Datenbanken exportieren und importieren:

Code-Ausschnitt 8.3.66

```
1 mysqldump -ubenutzer -pkennwort databank
2 mysql -ubenutzer -pkennwort -hhostname < databank.sql
```

Ersetzen Sie bitte *benutzer* mit dem Benutzernamen für die Datenbank, *kennwort* mit dem Kennwort des Benutzers, *databank* mit Ihrer Datenbank und *hostname* mit dem Host-Namen des Rechners, auf dem sich die Datenbank befindet.

8.3.5 Kein CGI-Freund?

Ich gebe es zu: *CGI*-Skript (*Common Gateway Interface*) zu lesen, fällt mir deutlich schwerer als *PHP*-Code. *CGI* ist ein Standard für den Datenaustausch zwischen einem Webserver und anderer Software, die Anfragen bearbeitet. Warum ich darauf komme? Die Datei *hotspotlogin.cgi*, die die Startseite unseres *chillispot*-Servers darstellt, ist eine *CGI*-Datei, die wir in */etc/chilli/chilli.conf* angegeben haben. Für alle Freunde des *PHP*-Skripts habe ich eine alternative *hotspotlogin*-Datei, die ausschließlich *PHP*-Code enthält.



Die Datei *hotspotlogin.php* finden Sie im Download-Bereich <https://www.springer.com/de/book/978-3-662-58143-8> unter dem Namen *hotspotlogin.php.tar.bz2*.

Wenn Sie die *PHP*-Datei anstelle der *CGI*-Datei verwenden wollen, ändern Sie bitte den Eintrag in der Datei */etc/chilli/chilli.conf* wie folgt:

Code-Ausschnitt 8.3.67

```
1 uamserver https://192.168.179.1/cgi-bin/hotspotlogin.php
```

8.3.6 Firewall

Mit Hilfe des *iptables*-Befehls kann man unter *LINUX* eine *Firewall* realisieren. Eine Firewall filtert den Netzverkehr, lässt nur bestimmte Pakete durch oder sperrt bestimmte Ports. Der folgende Quelltext zeigt ein Skript, welches bestimmte Firewall-Regeln für *Chillispot* setzt. Speichern Sie es unter einem Namen Ihrer Wahl, machen Sie das Skript ausführbar und starten Sie es, nachdem *Chillispot* komplett läuft.



Mit Hilfe von *iptables* können Sie sich hervorragend aus Ihrem Rechner aussperren (z. B. den *ssh*-Zugang kappen). Sie müssen sich dann meist vor Ihren Raspberry Pi setzen, der an einen Monitor angeschlossen ist, und diesen von da aus neu starten. Aber keine Angst: Sie können nichts kaputt machen. Nach einem Neustart ist das Verhalten wieder so, wie es vorher war.

Firewall-Skripte können Sie automatisch von *chillispot* aufrufen lassen.

Kopieren Sie das Skript unter dem Namen *ipup.sh* in das Verzeichnis */etc/chilli*. Machen Sie es ausführbar und tragen Sie in die Datei */etc/chilli/chilli.conf* folgende Zeile ein:

Code-Ausschnitt 8.3.68

```

1 # TAG: ipup
2 # Script executed after network interface has been brought up.
3 # Executed with the following parameters: <devicename> <ip address>
4 # <mask>
5 # Normally you do not need to uncomment this tag.
6 ipup /etc/chilli/ipup.sh

```

Chillispot wird dann automatisch die Firewall beim Start laden. Analog dazu können Sie beim Beenden von *Chillispot* ein Skript angeben - *ipdown* -, welches anlässlich des Beendens ausgeführt wird. Damit können Sie die alten Firewall-Regeln wieder herstellen. Das Skript wird dann ebenfalls an entsprechender Stelle in der Datei */etc/chilli/chilli.conf* eingetragen. Mit Hilfe der *iptables* können Sie auch die Verbindung auf andere Rechner Ihres Ethernets unterbinden.

Code-Ausschnitt 8.3.69

```

1 #!/bin/sh
2 IPTABLES="/sbin/iptables"
3 EXTIF="eth0"
4 INTIF="wlan0"
5 $IPTABLES -P INPUT DROP
6 $IPTABLES -P FORWARD ACCEPT
7 $IPTABLES -P OUTPUT ACCEPT
8 #Allow related and established on all interfaces (input)
9 $IPTABLES -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
10 #Allow related, established and ssh on $EXTIF. Reject everything else.
11 $IPTABLES -A INPUT -i $EXTIF -p tcp -m tcp --dport 22 --syn -j ACCEPT
12 $IPTABLES -A INPUT -i $EXTIF -j REJECT
13 #Allow related and established from $INTIF. Drop everything else.
14 $IPTABLES -A INPUT -i $INTIF -j DROP
15 #Allow http and https on other interfaces (input).
16 #This is only needed if authentication server is on same server as chilli
17 $IPTABLES -A INPUT -p tcp -m tcp --dport 80 --syn -j ACCEPT
18 $IPTABLES -A INPUT -p tcp -m tcp --dport 443 --syn -j ACCEPT
19 #Allow 3990 on other interfaces (input).
20 $IPTABLES -A INPUT -p tcp -m tcp --dport 3990 --syn -j ACCEPT
21 #Allow everything on loopback interface.
22 $IPTABLES -A INPUT -i lo -j ACCEPT
23 # Drop everything to and from $INTIF (forward)
24 # This means that access points can only be managed from ChilliSpot
25 $IPTABLES -A FORWARD -i $INTIF -j DROP
26 $IPTABLES -A FORWARD -o $INTIF -j DROP
27 #Enable NAT on output device
28 $IPTABLES -t nat -A POSTROUTING -o $EXTIF -j MASQUERADE

```



Sie können in die Datei */etc/chilli/chilli.conf* auch Seiten eintragen, die der Gast Ihres Hotspots besuchen darf, ohne angemeldet zu sein. Diese Seiten werden in der Rubrik *uamallowed* aufgeführt.

8.3.7 Tooor!



Abbildung 8.13:
(Quelle:
torproject.org).

... schreie ich gerne, wenn mein Lieblings-Fußballverein Borussia Mönchengladbach eines schießt. Hier geht es aber nicht um Fußballtore, sondern um *TOR - The Onion Router*. *Onion* ist Englisch und bedeutet Zwiebel. Das Innere der Zwiebel, unsere Netzidentität, verschleiern wir in diesem Abschnitt, indem wir viele Schichten darüber fügen, so dass der innere Kern (hoffentlich) verborgen bleibt. Wie funktioniert das? Die ersten Ideen für das *TOR*-Netzwerk stammen aus dem Jahr 2000, als sich eine englische Universität mit verschiedenen Verteidigungsorganisationen zusammengeschlossen hat, um Internet-Verbindungsdaten zu anonymisieren. Paradoxerweise wird das Projekt heute zu 60 % von der US-Regierung und zu 40 % von Spenden finanziert. Ein Schelm, wer Böses dabei denkt! Die Funktionsweise ist folgende: Der Klient des Nutzers verbindet sich mit dem *TOR*-Netzwerk und lädt eine Liste mit vorhandenen *TOR*-Servern herunter. Die Liste der Server ist verschlüsselt, der öffentliche Schlüssel wird mit dem *TOR*-Quelltext ausgeliefert. Nach Empfang der Liste wählt der Zwiebel-Proxy eine zufällige Route über die *TOR*-Server. Mit dem ersten *TOR*-Server wird dabei eine verschlüsselte Verbindung hergestellt. Wenn diese steht, kommen zwei weitere *TOR*-Server hinzu, so dass der Netzverkehr immer über drei *TOR*-Server läuft. Der letzte *TOR*-Server ist dabei der Server, der die Verbindung zum eigentlichen Ziel aufnimmt. Er wird auch als *Exitnode* (als Endknoten) bezeichnet. Der beschriebene Verbindungsaufbau wird regelmäßig wiederholt, wobei die Verbindungsstrecken selbst alle zehn Minuten gewechselt werden. Einzige Schwachstelle dieses Systems ist der *TOR*-Server selbst. Hier kann der gesamte Datenverkehr mitgelesen werden. Aus diesem Grunde empfehle ich, ausschließlich vertrauenswürdige *TOR*-Server zu verwenden - doch dazu später mehr. *TOR* kann auch dazu verwendet werden, Internetsperren oder -zensur zu umgehen. Daher sperren viele chinesische Provider oft auch *TOR*-Zugänge. Ich möchte an dieser Stelle noch einmal darauf hinweisen, dass *TOR* keine hundertprozentige Anonymität gewährleisten kann. Setzen Sie *TOR* darum mit Bedacht ein. In diesem Abschnitt gehen wir davon aus, dass Sie einen WLAN-Hotspot aufgesetzt haben wie oben beschrieben. Wir werden den WLAN-Hotspot mit Hilfe von *TOR* anonymisieren. *TOR* ist in der Raspbian-Distribution bereits vorhanden. Installieren wir es auf die übliche Art und Weise:

Code-Ausschnitt 8.3.70

```
1 sudo apt-get install tor
```

Nach der Installation müssen einige Anpassungen in der Datei `/etc/tor/torrc` vorgenommen werden. Meine Datei sieht so aus (die Kommentare habe ich aus Gründen der Übersichtlichkeit weggelassen):

Code-Ausschnitt 8.3.71

```
1 Log notice file /var/log/tor/tor.log
2 VirtualAddrNetwork 10.192.0.0/10
3 AutomapHostsSuffixes .onion,.exit
4 AutomapHostsOnResolve 1
5 TransPort 9040
6 TransListenAddress 192.168.179.1
7 DNSPort 53
8 DNSListenAddress 192.168.179.1
9 # Nur vertrauenswürdige exit nodes verwenden
10 StrictNodes 1
11 ExitNodes $CA1CF70F4E6AF9172E6E743AC5F1E918FFE2B476
12 ExitNodes $944224E9413705EEAFCBAC98BF57C475EB1960C5
13 ExitNodes $1041213B53CCF586093BB65D9CC4BC0B9656EF17
```

Zur Erklärung: *TOR* legt eine *log*-Datei unter */var/log/tor/tor.log* an, anhand derer man den Verbindungsaufbau verfolgen kann. Als Nächstes tragen wir eine willkürliche virtuelle Netzwerkadresse ein. *TOR* selbst soll unsere IP-Adresse 192.168.179.1 verschlüsseln: Das ist die Hotspot-Adresse, die wir in den letzten Abschnitten angelegt haben.

Info Möchten Sie ein anderes Netzwerk-Interface verschlüsseln, z. B. den gesamten Datenverkehr über *eth0*, müssen Sie die IP-Adresse dieses Interfaces angeben.

Unter *ExitNodes* legen wir drei als vertrauenswürdig eingestufte Exitnodes fest [Sch13a]. Bevor wir *TOR* starten, müssen wir noch die Log-Datei initiieren und dem richtigen Besitzer zuweisen.

Code-Ausschnitt 8.3.72

```
1 sudo touch /var/log/tor/tor.log
2 sudo chown debian-tor /var/log/tor/tor.log
3 sudo chmod 644 /var/log/tor/tor.log
```

Info Den Befehl *touch* kennen Sie noch nicht. Er macht das, wofür seine Übersetzung steht: „anfassen“, und zwar die Datei, die als Parameter angegeben ist. Existiert diese Datei bereits, wird die letzte Zugriffszeit aktualisiert. Existiert sie noch nicht, wird eine 0 Byte große leere Datei angelegt.

Info Der *TOR*-Dienst erwartet den Benutzer *debian-tor*, den wir der Log-Datei zuweisen. *TOR* benötigt einen freien DNS-Port mit der Nummer 53. Das ist, wie gesagt, der Port, auf dem normalerweise der dynamische Namensdienst läuft (vielleicht erinnern Sie sich an den Befehl *nslookup*, den ich Ihnen zu Beginn dieses Buches vorgestellt habe). Ist dieser Port belegt, wird *TOR* nicht starten. Dieser Port ist dann belegt, wenn Sie einen eigenen DNS-Server auf dem Raspberry Pi einrichten. Das werden wir später noch mit *powerdns* machen. Sollten Sie also *TOR* starten wollen und *powerdns* läuft, muss der *powerdns*-Dienst zuerst mit dem üblichen Kommando beendet werden.

Wir sind noch nicht ganz am Ziel, da wir noch einige Routing-Tabellen umstellen müssen:

Code-Ausschnitt 8.3.73

```
1 # Entfernen Sie die alten Regeln der IP NAT-Tabelle
2 sudo iptables -F
3 sudo iptables -t nat -F

5 # Fügen Sie eine Ausnahme für den Port 22 (ssh) hinzu
6 sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 22 -j REDIRECT --to-ports 22

8 # Leiten Sie alle DNS-Anfrage (UDP Port 53) von wlan0 auf den internen Port 53
9 sudo iptables -t nat -A PREROUTING -i wlan0 -p udp --dport 53 -j REDIRECT --to-ports 53

11 # Leiten Sie den gesamten TCP-Verkehr von wlan0 über den Port 9040 (Proxy für TOR)
12 sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp --syn -j REDIRECT --to-ports 9040

14 # save it to our old NAT
15 # sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Die letzten Zeilen habe ich einkommentiert - dazu später mehr. Speichern Sie den oberen Quelltext, beispielsweise im Verzeichnis */home/pi/tor/iptables.sh* und machen Sie die Datei ausführbar. Danach rufen Sie sie auf und starten den *TOR*-Dienst.

Code-Ausschnitt 8.3.74

```
1 chmod 775 ~/tor/iptables.sh
2 cd ~/tor
3 sudo ./iptables.sh
4 sudo /etc/init.d/tor start
```

Ihr gesamter Hotspot-Verkehr sollte nun über *TOR* geleitet werden. Das werden wir gleich überprüfen. Zurück zum Thema Auskommentieren: Möchten Sie die Anonymisierung über *TOR* dauerhaft einbinden, entfernen Sie den letzten Kommentar. Das speichert die aktuellen Tabellen und unser Eintrag in */etc/network/interfaces* lädt die aktuellen Tabellen beim Start.

Code-Ausschnitt 8.3.75

```
1 up iptables-restore < /etc/iptables.ipv4.nat
```

Die aktuellen IP-Tabellen können Sie sich mit dem Befehl

Code-Ausschnitt 8.3.76

```
1 sudo iptables -t nat -L
```

anzeigen lassen. Jetzt ist es aber Zeit, unsere Identität im Netzwerk zu überprüfen. Wählen Sie sich in den Hotspot ein und rufen Sie die Seite <http://www.ip-adresse-ermitteln.de/> in Ihrem Browser auf. Bei mir sieht das Ergebnis wie in Abb. 8.14 aus. Ich wohne jetzt in der Schweiz!

The screenshot shows a web browser window with the URL www.ip-adresse-ermitteln.de. The page title is "Fremde IP-Adressen lokalisieren sowie eigene IP-Adresse ermitteln". The main content area displays "Jetzt kostenlos IP-Adresse lokalisieren" and "Ihre aktuelle IP-Adresse: 77.109.138.42". Below this, there is a search input field containing "77.109.138.42" and a blue button labeled "→ IP lokalisieren". The page footer includes navigation links: "Startseite / Providerübersicht / FAQ / Speedtest / Praktikum IT / Lexikon". A prominent advertisement for "Netzwerk-Port-Scanner" is visible, with the URL www.gfisoftware.de/LanGuard_2014 and the text "Spüren Sie offene Ports auf mit GFI LanGuard®. Jetzt gratis testen!". Below the ad, there is a section titled "Informationen zur gesuchten IP-Adresse 77.109.138.42" with the following details:

IP Adresse:	77.109.138.42
Land:	Switzerland
Stadt:	
Breitengrad:	47
Längengrad:	8
Host:	spfor4e1.privacyfoundation.ch

To the right of this information is a map showing the location of the IP address. A pop-up window on the map is titled "Position der IP Adresse" and displays "77.109.138.42 / CH". The map shows the location in Switzerland, near the town of Luthern.

Abbildung 8.14: IP-Ermittlung im Internet: *TOR* funktioniert wie erwartet.

8.4 OpenVPN

Vielleicht betreibt die Firma, in der Sie arbeiten, einen VPN-Server (*Virtual Private Network*), so dass Sie sich von zu Hause aus in Ihr Firmennetz einwählen können, um Daten vom Server herunterzuladen, die Sie in einen Bericht einbauen müssen. In diesem Abschnitt zeige ich Ihnen den umgekehrten Weg. Wir richten auf dem Raspberry Pi einen VPN-Server ein, so dass Sie sich von außen in Ihr Heimnetz einwählen können. *OpenVPN* ist ein freier VPN-Server, der auf vielen Rechnerplattformen läuft. Dasselbe gilt für die Klienten. Auch hier bleiben keine Wünsche offen. Selbst Telefone werden unterstützt. Nachdem die VPN-Verbindung zu Ihrem Raspberry Pi hergestellt worden ist, wird der gesamte Netzverkehr über den Raspberry Pi geroutet. Sie befinden sich „virtuell“ in Ihrem privaten Netz - daher der Name VPN. Beginnen wir mit der Installation, indem wir *openvpn* und das freie SSL-Paket *openssl* installieren.



Abbildung 8.15:
(Quelle:
openvpn.net).

Code-Ausschnitt 8.4.1

```
1 sudo apt-get install openvpn openssl easy-rsa
```

Aus der Installation von *openvpn* installieren wir den Ordner *easy-rsa* in das *openvpn*-Konfigurations-Verzeichnis.

Code-Ausschnitt 8.4.2

```
1 cd /etc/openvpn
2 sudo cp -r /usr/share/easy-rsa ./easy-rsa
```

Wechseln Sie dann bitte in den Ordner *easy-rsa* und öffnen die Datei *vars* in einem Texteditor Ihrer Wahl als Benutzer *root*, z. B. mit *sudo vi /etc/openvpn/easy-rsa/vars*. In dieser Datei ersetzen Sie bitte

Code-Ausschnitt 8.4.3

```
1 export EASY_RSA="/etc/openvpn/easy-rsa"
```

durch

Code-Ausschnitt 8.4.4

```
1 export EASY_RSA="/etc/openvpn/easy-rsa"
```

Im nächsten Schritt vervollständigen wir die *openvpn*-Installation, indem wir im *easy-rsa*-Verzeichnis als Administrator die folgenden Befehle aufrufen:

Code-Ausschnitt 8.4.5

```
1 cd easy-rsa
2 sudo su
3 source vars
4 ./clean-all
5 ./pkitoool --initca
6 ln -s openssl-1.0.0.cnf openssl.cnf
```

Jetzt ist es an der Zeit, die Schlüssel für die *openvpn*-Verbindung zu erstellen. Dazu rufen Sie bitte die folgenden Befehle auf:

Code-Ausschnitt 8.4.6

```
1 ./build-ca OpenVPN
2 ./build-key-server server
3 ./build-key client1
```

Die Befehle erstellen einen privaten Schlüssel für den Server (*server*) und einen öffentlichen Schlüssel für einen Klienten (*client1*). Auf dem Weg dorthin werden Ihnen einige Fragen gestellt, die ich so beantwortet habe:

Code-Ausschnitt 8.4.7

```
1 Country Name (2 letter code) [GB]:DE
2 State or Province Name (full name) [Berkshire]:NRW
3 Locality Name (eg, city) [Newbury]:Mettmann
4 Organization Name (eg, company) [My Company Ltd]:Ready4us UG
5 Organizational Unit Name (eg, section) []:Webservice
6 Common Name (eg, your name or your server's hostname) []:R. Follmann
7 Email Address []:rudi@follmann.name
8 Please enter the following 'extra' attributes
9 to be sent with your certificate request
10 A challenge password []:
11 An optional company name []:
```

Die komplette Installation der Schlüssel erfolgt nach dem nächsten Aufruf:

Code-Ausschnitt 8.4.8

```
1 ./build-dh
2 exit
```

Der *exit*-Befehl meldet uns wieder als Administrator von der Konsole ab.



Geben Sie als normaler Benutzer den Befehl *exit* in einer Konsole ein, schließt LINUX die Konsole. Sind Sie als ein anderer Benutzer eingewählt - beispielsweise als *root* -, schließt ein *exit* diese Konsole nicht, sondern meldet lediglich den Benutzer ab, unter dem Sie eingewählt waren.

Bitte wechseln Sie nun in das Verzeichnis */etc/openvpn* und erstellen Sie dort eine Datei namens *openvpn.conf* als Benutzer *root*, z. B. mit *sudo vi /etc/openvpn/openvpn.conf*. In diese Datei kopieren Sie bitte folgenden Inhalt:

Code-Ausschnitt 8.4.9

```

1 dev tun
2 proto udp
3 port 1194
4 ca /etc/openvpn/easy-rsa/keys/ca.crt
5 cert /etc/openvpn/easy-rsa/keys/server.crt
6 key /etc/openvpn/easy-rsa/keys/server.key
7 dh /etc/openvpn/easy-rsa/keys/dh1024.pem
8 user nobody
9 group nogroup
10 server 10.8.0.0 255.255.255.0
11 persist-key
12 persist-tun
13 status /var/log/openvpn-status.log
14 verb 3
15 client-to-client
16 push "redirect-gateway def1"
17 # DNS Server setzen
18 push "dhcp-option DNS 8.8.8.8"
19 push "dhcp-option DNS 8.8.4.4"
20 log-append /var/log/openvpn
21 comp-lzo
22 duplicate-cn
23 keepalive 10 120
24 push "explicit-exit-notify 3"

```

Was geschieht hier? Wir teilen *openvpn* mit, wo unsere Schlüssel liegen oder welchen Server-IP-Adress-Bereich wir verwenden.



Erinnern Sie sich noch daran, dass wir bei *CUPS* den 10.*-Bereich in die Konfigurationsdatei eingetragen haben. Dadurch können wir später unsere Heimnetz-Drucker von extern warten, sofern wir uns mit VPN in das Heimnetz eingewählt haben.

Weiterhin definieren wir eine Log-Datei und geben wieder Google's öffentliche Nameserver als DNS an. Damit der Rechner, der später per VPN mit unserem Raspberry Pi verbindet, auch eine Internetverbindung erhält, muss diese eingerichtet werden. Wahrscheinlich wird der eine oder die andere jetzt denken „Bitte keine *iptables* mehr“, aber genau darauf wird es hinauslaufen. Wir aktivieren das IP-Forwarding. Das ist die Bedingung dafür, dass wir überhaupt von einem IP-Adressraum in den anderen weiterleiten dürfen. Anschließend leiten wir alle Pakete auf dem 10.*-Interface (also dem Adressraum unseres VPN-Servers) auf das *eth0*-Interface (nämlich unsere kabelgebundene Internetverbindung) um.

Code-Ausschnitt 8.4.10

```

1 sudo sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
2 sudo iptables -t nat -A POSTROUTING -s 10.0.0.0/8 ! -d 10.0.0.0/8 -o eth0 -j MASQUERADE

```

Um das IP-Fowarding dauerhaft zu aktivieren, muss in der Datei */etc/sysctl.conf* der Eintrag

Code-Ausschnitt 8.4.11

```

1 net.ipv4.ip_forward=1

```

vorhanden sein. Um die Einstellungen *reboot*-sicher zu machen, möchte ich Ihnen noch die zeitgesteuerten Dienste und Befehle in LINUX näher bringen.



Der *Cron*-Daemon unter LINUX ist ein Dienst, der Skripte und Programme zu einer vorgegebenen Zeit oder einem vorgegebenen Event startet. Der Dienst liest aus einer Tabelle mit dem Namen *crontab* die auszuführenden Befehle und die Zeiten, zu denen sie ausgeführt werden sollen. Die Seite <http://wiki.ubuntuusers.de/Cron> hält weitere Information über den Aufbau dieser Tabelle bereit. Den Editor, den *crontab -e* verwendet, kann man über die Systemvariable *EDITOR* festlegen. Ein *export EDITOR=/usr/bin/joe* vor dem Aufruf von *crontab -e* stellt den Editor *joe* zum Editieren der *Cron*-Tabelle ein. Nach Eingabe des Befehls *update-alternatives --config editor* kann man sich sogar einen Editor aus einer Liste der installierten Editoren aussuchen.

Rufen Sie bitte

Code-Ausschnitt 8.4.12

```
1 sudo crontab -e
```

auf und tragen Sie folgende Information ein:

Code-Ausschnitt 8.4.13

```
1 @reboot sudo /home/pi/tor/iptables.sh
```

Das Skript *iptables.sh* enthält alle o. a. *iptables*-Befehle. Die Anweisung *reboot* sagt der *Cron*-Tabelle, dass der darauf folgende Befehl nach jedem Reboot einmal ausgeführt wird. Nach so viel Vorarbeit ist unser *OpenVPN*-Server schon fast lauffähig. Wir müssen nur noch das Schlüsselpaket für den Klienten erzeugen, bevor wir den Verbindungsaufbau starten können. Nehmen Sie hierzu wieder Administratorrechte an, wechseln Sie in das Verzeichnis */etc/openvpn/easy-rsa/keys* und editieren Sie eine Datei namens *raspberrypi.ovpn*.

Code-Ausschnitt 8.4.14

```
1 sudo su
2 cd /etc/openvpn/easy-rsa/keys
3 vi raspberrypi.ovpn
```

Füllen Sie diese Datei bitte mit folgendem Inhalt:

Code-Ausschnitt 8.4.15

```
1 dev tun
2 client
3 proto udp
4 remote ip-adresse 1194
5 resolv-retry infinite
6 nobind
7 persist-key
8 persist-tun
9 ca ca.crt
10 cert client1.crt
11 key client1.key
12 comp-lzo
13 verb 3
```

Für das Wort *ip-adresse* setzen Sie bitte die *eth0*-IP-Adresse Ihres Raspberry Pi ein. Ein *ifconfig* hilft Ihnen dabei, diese ausfindig zu machen. Bevor wir uns wieder als Administrator mit einem *exit* abmelden, erstellen wir ein Paket, welches alle Zertifikate und Schlüssel enthält, die unser Klient benötigt. Dieses Paket packen wir mit dem *tar*-Befehl zusammen und verschieben es ins */home*-Verzeichnis des Raspberry Pi. Nachdem wir den Besitzer des Pakets von *root* auf *pi* gesetzt haben (inklusive der richtigen Gruppenzugehörigkeit), darf der Standard-Benutzer *pi* auf dieses Paket zugreifen und es z. B. auf den Klienten kopieren.

Code-Ausschnitt 8.4.16

```
1 tar czf openvpn-keys.tgz ca.crt client1.crt client1.csr client1.key
2 raspberrypi.ovpn
3 mv openvpn-keys.tgz /home/pi
4 chown pi:pi /home/pi/openvpn-keys.tgz
5 exit
```

Durch ein Neustarten des *OpenVPN*-Servers machen wir dem Server alle bisher durchgeführten Änderungen bekannt.

Code-Ausschnitt 8.4.17

```
1 sudo /etc/init.d/openvpn restart
```

Möchten Sie Schlüssel für andere Klienten erzeugen, können Sie das - analog zum Erzeugen der Schlüssel für *client1* - wie folgt:

Code-Ausschnitt 8.4.18

```
1 cd /etc/openvpn/easy-rsa/
2 sudo su
3 source vars
4 ./build-key client2
```

Bei der Erstellung der Datei *raspberrypi.ovpn* müssen Sie dann darauf achten, hier *client2* oder entsprechend höhere Nummern zu berücksichtigen. Dies gilt natürlich auch für die abschließende Schlüsselpaket-Erstellung:

Code-Ausschnitt 8.4.19

```
1 cd /etc/openvpn/easy-rsa/keys
2 tar czf openvpn-keys-client2.tgz ca.crt client2.crt client2.csr client2.key
3 raspberrypi.ovpn
4 mv openvpn-keys-client2.tgz /home/pi
5 chown pi:pi /home/pi/openvpn-keys-client2.tgz
6 exit
```

Den eben erstellten Schlüssel kopieren wir auf den Klienten, damit dieser sich mit unserem *OpenVPN*-Server verbinden kann.

8.4.1 Der Klient

Den Klienten gibt es nicht nur in Form von John Grisham's Roman, sondern auch als *OpenVPN*-Verbindungssoftware auf unzähligen Betriebssystemen. Unter Windows können Sie ihn von <http://openvpn.se> herunterladen. Unter MacOS gibt es den *Tunnelblick* (<https://tunnelblick.net>) und unter LINUX selbst steht das Paket *openvpn* zur Verfügung. Selbst für iPhone und Co. gibt es offizielle *OpenVPN*-Klienten. Werfen wir noch kurz einen Blick auf den Windows- und iPhone-Klienten. Der Download für Windows gestaltet sich unspektakulär. Einen Installer für die letzte stabile Version erhalten Sie unter <http://swupdate.openvpn.org/privatetunnel/client/privatetunnel-win-2.8.exe>.

Info Beachten Sie bitte, dass der Windows *OpenVPN*-Klient mit Administrator-Rechten laufen muss. Andernfalls kann die Verbindung nicht hergestellt werden.

Info *tar*-Archive können Sie unter Windows mit dem Programm *WINRAR* auspacken.

Möchten Sie *OpenVPN* auf dem iPhone nutzen, können Sie eine neue *.ovpn*-Datei erstellen, die zusätzlich zu den oben in *raspberrypi.ovpn* enthaltenen Zeilen die folgenden Zeilen enthält:

Code-Ausschnitt 8.4.20

```

1  ...
2  <cert>
3  ...
4  </cert>
5  ...
6  <key>
7  ...
8  </key>

```

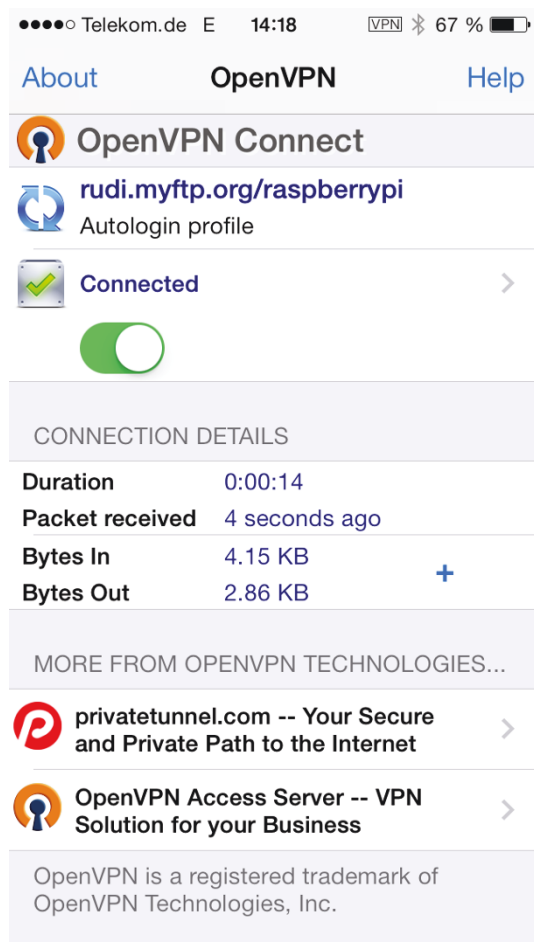


Abbildung 8.16: *OpenVPN* auf dem iPhone.

Anstelle der Punkte tragen Sie bitte Ihr Zertifikat (Inhalt der *.cert*-Datei) und Ihren Schlüssel (Inhalt der *.key*-Datei) aus Ihrem Schlüsselpaket ein. Die so geänderte Datei können Sie sich nach der Installation von *OpenVPN* als Email zusenden. Beim Öffnen der Email können Sie auf den Anhang (*.ovpn*-Datei) klicken. Ihr Telefon verbindet die Datei dann automatisch mit der *OpenVPN*-Software und installiert den Schlüssel. Um von außen auf den VPN-Server zugreifen zu können, fehlt noch eine Portweiterleitung in Ihrem DSL-Modem. Ich gehe davon aus, dass Sie - wie in Kapitel 3.1.2 beschrieben - bereits einen DynDNS-Service nutzen. Wir haben ihn damals *pi.myftp.org* genannt. Diesen Namen müssen Sie von extern nutzen, wenn Sie sich mit Ihrem *OpenVPN*-Server verbinden wollen. Gleichzeitig müssen Sie in Ihrem DSL-Modem den Port 1194 für *TCP* und *UDP* auf die IP-Adresse Ihres Raspberry Pi umleiten (und zwar ebenfalls an die jeweiligen 1194-Ports). Der Verbindungsaufbau auf meinem iPhone sieht so aus wie in Abb. 8.16. Der hier installierte *OpenVPN*-Server wird uns noch im Kapitel 10 (Hausautomatisierung) von großem Nutzen sein. Er ermöglicht uns nämlich eine sichere Verbindung zum Schalten von Hausautomatisierungs-Geräten aus der Ferne.

8.5 CMS

Content Management-Systeme *CMS* sind ein Paradebeispiel für die Verbindung eines Webservers mit einer Datenbank. Ein bekanntes Open-Source CMS ist *Contao*. Es wird maßgeblich vom Wuppertaler Programmierer und Server-Administrator Leo Feyer programmiert und liegt inzwischen in der Version 4 vor. *Contao* funktioniert dabei nach folgendem Prinzip: Eine Datenbank enthält die Struktur der Webseite. Das betrifft sowohl den Inhalt als auch das Aussehen der Webseite. Ruft ein Benutzer diese Webseite auf, baut ein *php*-Skript *html*-Code zur Darstellung der Seite zusammen, indem es in der Datenbank nachschaut, was dem Benutzer präsentiert werden soll und wie es ihm präsentiert werden soll. So kann man beispielsweise sehr einfach zwischen Gästen und angemeldeten Mitgliedern einer Seite unterscheiden, die jeweils andere Zugangsberechtigungen haben und daher auch anderen Inhalt (*content*) sehen. Der Inhalt der Webseite ist ebenfalls in der Datenbank gespeichert. Während das *Frontend* - also das, was der Benutzer einer Webseite sieht - diesen Inhalt darstellt, erlaubt der Zugang über ein *Backend*, den Inhalt zu modifizieren. Modifizierter Inhalt ist dann sofort als neuer Webinhalt sichtbar; Inhalt und Form werden getrennt. Der Raspberry Pi eignet sich durchaus als Testplattform für ein *CMS*. Allerdings braucht *Contao* einige Ressourcen. Wundern Sie sich also nicht, wenn der Aufbau einer Webseite einige Sekunden Zeit in Anspruch nimmt. Trotzdem ist der Raspberry Pi eine ideale Versuchsplattform und ich lade Sie in diesem Abschnitt ein, gemeinsam mit mir die Demo-Installation von *Contao* gemeinsam vorzunehmen. *Contao* erwartet, dass einige Programmpakete installiert sind. Erledigen wir dies also zuerst.



Abbildung 8.17:
(Quelle: Leo Feyer).

Code-Ausschnitt 8.5.1

```
1 sudo apt-get install php7.0 mysql-server libgd2-noxpm-dev php-soap libapache2-mod-php7.0↵
   php7.0-mcrypt apache2-bin php7.0-mcrypt php7.0-intl
```

Die Namen der Programmpakete haben folgende Bedeutungen:

- *GDlib* wird für die Bildbearbeitung benutzt.
- *SOAP* wird für das Extension-Repository benutzt. Dieses stellt *Contao*-Erweiterungen zur Verfügung. Die meisten *Contao*-Erweiterungen können bequem vom Web-Backend aus installiert werden.
- *mcrypt* stellt Verschlüsselung zur Verfügung.
- *intl* stellt verschiedene Sprachpakete zur Verfügung.

Einige der Programmpakete haben Sie wahrscheinlich schon längst installiert. *Contao* selbst gibt es auf der Webseite <http://www.contao.org>. Da wir aber möglichst schnell zum Ziel kommen wollen und dieses Buch kein Leitfaden für die Installation und Benutzung von *Contao* ist, installieren wir eine *Contao*-Demo-Webseite unter Verwendung des PHP-Composers. Dieses Programm laden wir im *home* Verzeichnis des Raspberry Pi herunter

Code-Ausschnitt 8.5.2

```
1 cd /home/pi
2 wget https://raw.githubusercontent.com/composer/getcomposer.org/1↵
   b137f8bf6db3e79a38a5bc45324414a6b1f9df2/web/installer -0 - -q | php -- --quiet
3 sudo cp composer.phar /usr/local/bin/composer
```

Contao stellt einen eigenen Check zur Verfügung, mit dem Sie überprüfen können, ob alle Voraussetzungen zum Betrieb von *Contao* erfüllt sind. Das Programm erhalten Sie unter <https://github.com/contao/check/zipball/master>.

Die Benutzung wird ausführlich unter dem Link <http://de.contaowiki.org/Systemdiagnosetool> beschrieben, so dass ich an dieser Stelle darauf verzichte, dieses Programm zu erklären. Wir gehen einfach davon aus, dass unsere Installation korrekt durchgeführt worden ist. Damit *Contao* mit *MySQL* kommunizieren kann, erstellen wir zunächst eine neue, frische Datenbank. Wir nennen sie *db_contaodemo*. In den bisherigen Beispielen haben wir immer das Terminal bemüht, um eine Datenbank zu erstellen. Zur Abwechslung starten wir jetzt *phpmyadmin* und nutzen das Web-Interface zum Anlegen einer neuen Datenbank und eines Benutzers, der Zugriff auf diese Datenbank erhält. Rufen Sie also entweder auf dem Raspberry Pi einen Browser auf oder verbinden Sie sich per OpenVPN mit dem Raspberry Pi und rufen auf dem Klienten einen Browser auf mit der Adresse <http://ip-adresse/phpmyadmin>. Das Wort *ip-adresse* ersetzen Sie bitte wieder durch die IP-Adresse Ihres Raspberry Pi. Geben Sie bitte den Benutzernamen *phpmyadmin* und das Kennwort ein. Beides haben Sie bei der Installation von *phpmadmin* vergeben.

Wählen Sie bitte den Reiter *Datenbanken* aus und geben Sie in das Feld *Neue Datenbank anlegen db_contaodemo* ein. Das Feld *Kollation* lassen Sie bitte unangetastet. Nach einem Klick auf *Anlegen* erscheint die Demo-Datenbank in der linken Spalte. Klicken Sie in dieser Spalte auf die neu erstellte Datenbank und wählen Sie sie damit aus. Selektieren Sie nun im oberen Menü die Rubrik *Rechte*. Auf der neuen Seite erscheint der Link *Neuen Benutzer hinzufügen*. Klicken Sie darauf. Es öffnet sich ein neues Fenster, in dem Sie die Anmelde-Information für einen neuen Nutzer hinterlegen können. Füllen Sie alles wie folgt aus:

- *Benutzername*: Wählen Sie den Namen *user_contaodemo*.
- *Host*: Stellen Sie das Feld auf *Lokal*. Es erscheint dann *localhost* im Feld nebenan.
- *Passwort*: Vergeben Sie ein Kennwort, z. B. 123democontao321.
- *Wiederholen*: Geben Sie das Kennwort erneut ein.
- Unter der Rubrik *Datenbank für Benutzer* ist bereits die Datenbank *db_contaodemo* ausgewählt. Lassen Sie diese Auswahl stehen.
- Wählen Sie unter *Globale Rechte* *Alle Auswählen*.
- Bestätigen Sie alle ausgewählten Fenster durch einen Klick auf den *OK*-Button am Ende der Seite.

Der Benutzer wird dadurch hinzugefügt. Das System zeigt Ihnen sogar die *MySQL*-Befehle an, die Sie stattdessen in einem Terminal hätten eingeben können. Wir nutzen nun den Composer, um die Demo zu laden.

Code-Ausschnitt 8.5.3

```
1 cd /home/pi
2 composer create-project bytehead/contao-4-demo contao-4-demo
```

Der oben genannte Befehl erstellt einen Order namens *contao-4-demo* und lädt alle für die Demo-Webseite erforderlichen Erweiterungen und Abhängigkeiten. Während der Installation müssen Sie ein paar Daten angeben:

Code-Ausschnitt 8.5.4

```

1 database_host (localhost):
2 database_port (3306):
3 database_user (null): user_contaodemo
4 database_password (null): 123democontao321
5 database_name (null): db_contaodemo
6 mailer_transport (mail):
7 mailer_host (127.0.0.1):
8 mailer_user (null):
9 mailer_password (null):
10 mailer_port (25):
11 mailer_encryption (null):
12 prepend_locale (false):

```

Danach kann der Webserver gestartet werden:

Code-Ausschnitt 8.5.5

```

1 cd contaο-4-demo
2 php app/console server:start

```

Der Webserver, der im Demo-Verzeichnis beinhaltet ist, wird anschließend dazu benutzt, um *Contao* zu installieren. Dazu rufen wir einen lokalen Web-Browser (etwas *chromium-browser*) mit der URL `http://127.0.0.1:8000/install.php` auf. Auf einem Remote-Rechner können Sie *VNC* für diese Verbindung benutzen. Scrollen Sie zunächst an das Ende der angezeigten Seite und akzeptieren Sie die Lizenz. *Contao* fordert Sie dann auf, ein Kennwort für die Installation zu vergeben. Bitte notieren Sie sich auch dieses Kennwort. Sie benötigen es jedes Mal, wenn Sie auf die Installation zugreifen wollen. Nach Eingabe des Kennworts müssen Sie im

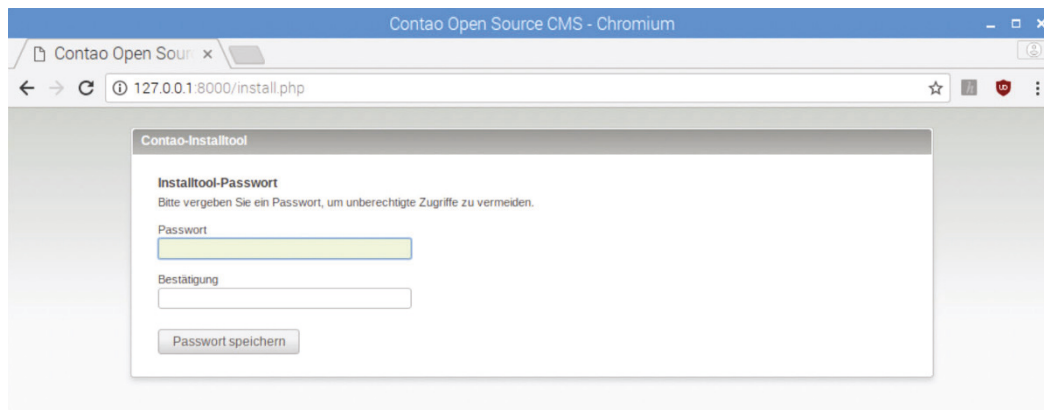


Abbildung 8.18: Der *Contao* Installations-Screen verlangt die Eingabe eines Installations-Kennwortes.

nächsten Schritt die Datenbank aktualisieren. Die Datenbank, die wir gerade noch aktualisiert haben, werden wir nun mit den Demo-Daten überschreiben. Diese befinden sich im *contaο-4-demo*-Verzeichnis und heißen *dump.sql*. Die Daten liegen übrigens in lesbarer Form (ASC-II-Text) vor. Es gibt nun zwei Möglichkeiten, diese Daten in die Datenbank *db_democontao* zu bekommen. Ein Weg besteht in einem Terminal-Aufruf:

Code-Ausschnitt 8.5.6

```

1 mysql -u user_democontao -p 123democontao321 db_democontao < dump.sql

```

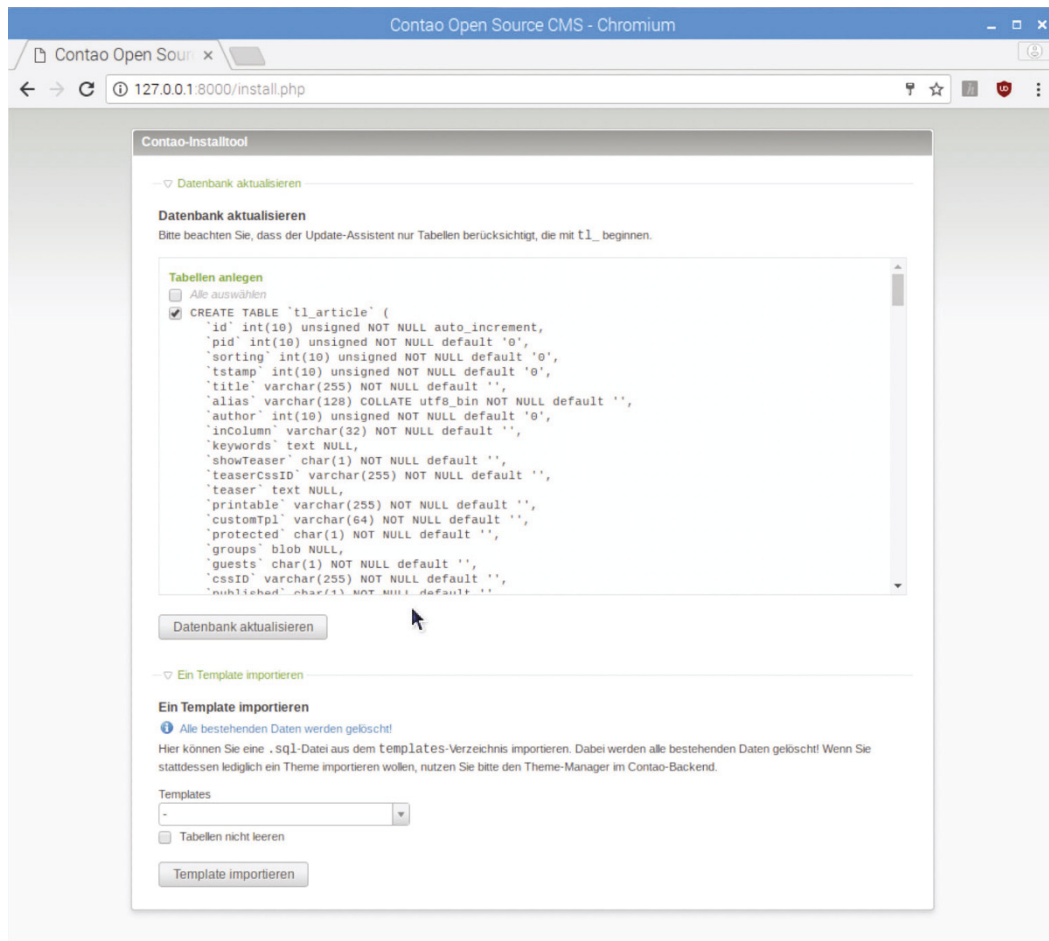


Abbildung 8.19: Die Aktualisierung der Datenbank.

Die andere Möglichkeit besteht im Importieren der sql-Daten in *phpmyadmin*. Selektieren Sie bei diesem Weg die Datenbank *db_democontao*, wählen Sie dann „Importieren“ und laden Sie die Daten *dump.sql*. Sie können sich nun unter der URL <http://127.0.0.1:8000/contao> im Backend einloggen. Der vordefinierte Benutzername ist „k.jones“ und das dazugehörige Kennwort „kevinjones“. Aktualisieren Sie dann bitte den Cache (*Build the cache*).

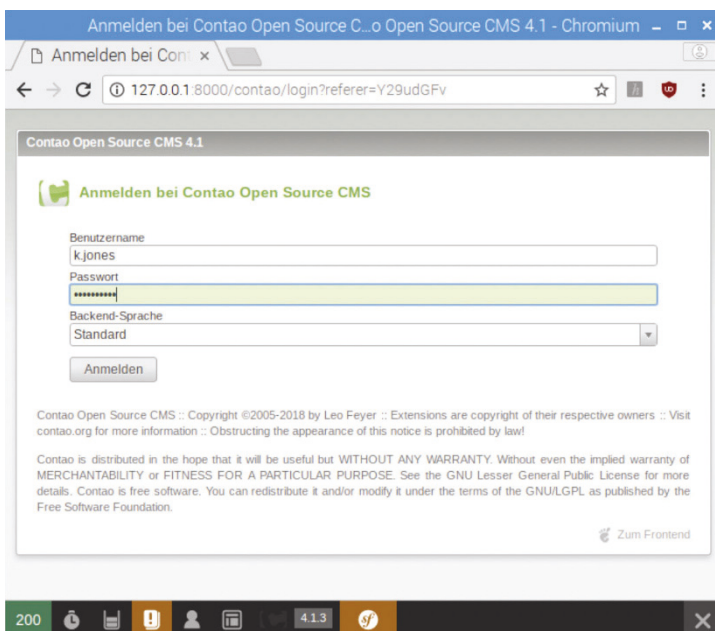


Abbildung 8.20: Login-Seite für das das Contao-Backend.

Danach können Sie sich vom Backend abmelden und die fertigen Webseiten unter `http://127.0.0.1:8000` bestaunen.

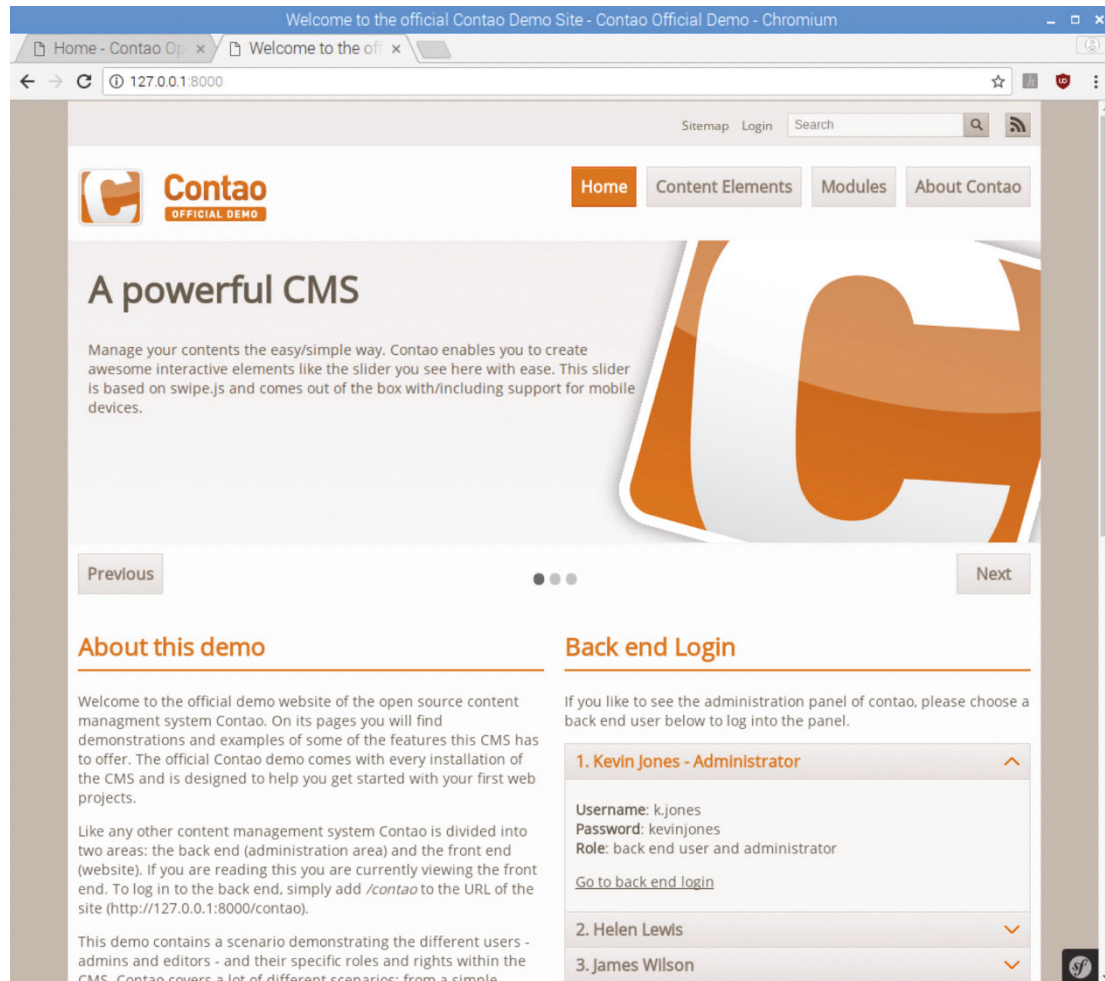


Abbildung 8.21: Geschafft: Das *Contao*-Beispiel *Demo* ist installiert.

Wer sich weitergehend mit *Contao* beschäftigen will, dem kann ich das Buch [Mül13] von Peter Müller empfehlen. Anfänger erhalten hier einen leicht verständlichen und umfassenden Überblick über dieses *CMS*. Der Raspberry Pi serviert *Contao*-Seiten zwar nicht besonders schnell, aber für einen ersten Einstieg ist er perfekt. Leider hat Peter Müller auf seiner Webseite angekündigt, kein Buch mehr für die Version 4 von *Contao* zu veröffentlichen. Die Gründe hierfür liegen im geänderten Fokus der Software, die mit der Version 4 mehr auf Agenturen als auf andere Benutzer zielt. *Contao* ist aber bei Weitem nicht das einzige *CMS*. Wer ein einfach zu benutzendes Tool sucht, ist sicherlich auch bei *Wordpress* gut aufgehoben.

Info Wenn Sie von einem anderen Rechner aus über die IP-Adresse auf den *Contao*-Server zugreifen möchten, dann rufen Sie den Start des Servers bitte so auf:

Code-Ausschnitt 8.5.7

```
1 php app/console server:start 192.168.178.66
```

8.6 Mach' mich nicht NAS!

NAS steht für *Network Attached Storage*. Das ist ein Speicher, der per Netzwerk zur Verfügung steht. Auf diesen Speicher können Sie zugreifen, um Backups zu sichern oder auf Backups zurückzugreifen. Oder Sie können Urlaubsfotos speichern und im Netzwerk verfügbar machen. Da der Raspberry Pi aber maximal nur über eine 300 Mbit/s Ethernet-Verbindung verfügt, gibt es gerade für größere Datenmengen sinnvollere *NAS*-Hardware, die mit einem 1 GB/s-Port ausgestattet ist, wie den *Cubietruck*. Das vorliegende Kapitel hat Ihnen bereits einige Möglichkeiten aufgezeigt, den Raspberry Pi in ein *NAS* zu verwandeln. Sie haben gelernt, wie eine externe USB-2.0-Festplatte eingebunden werden kann. Sie haben bereits vorher *iSCSI* und *Samba* kennengelernt. Damit können Sie schon jetzt komfortabel von anderen Betriebssystemen aus auf Ihren Raspberry Pi zugreifen. Wie das abgesichert funktioniert, habe ich Ihnen bereits in Kapitel 3 gezeigt. Sie können per *Samba* aber auch einen frei zugänglichen Speicherplatz auf Ihrem Datenträger einrichten. Wie das funktioniert, zeigt dieser Abschnitt. Erstellen Sie zunächst ein Verzeichnis, welches frei zugänglich sein soll. In dieses Verzeichnis werden wir später unsere externe Festplatte einhängen. In meinem Beispiel habe ich das Verzeichnis `/home/nas` genannt. Unterhalb dieses Verzeichnisses erstellen wir ein öffentliches Verzeichnis, welches wir *public* nennen.

Info An dieser Stelle gehe ich davon aus, dass eine externe USB-Festplatte in das Verzeichnis `/home/nas` eingehängt ist (Kapitel 5.5) und dass das Verzeichnis *nas* direkt auf diese Festplatte zeigt. Die entsprechenden Einträge sind in der Datei `/etc/fstab` vorzunehmen, indem Sie z. B. das im Kapitel 5.1 vorgestellte Verzeichnis `/video` durch `/home/nas` ersetzen. In der Wahl des Dateisystems sind Sie frei.

Info Unter LINUX steht Ihnen der Befehl *fdisk* zur Verfügung, wenn Sie Partitionen anlegen, löschen oder ändern wollen. Rufen Sie ihn mit dem Device auf, auf das Sie zugreifen möchten, z. B. `sudo fdisk /dev/sda`. Der Befehl *mkfs* (*make filesystem*) kann zum Formatieren verwendet werden.

Code-Ausschnitt 8.6.1

```
1 sudo mkdir /home/nas
2 sudo mkdir /home/nas/public
3 sudo chown -R root:users /home/nas/public
4 sudo chmod -R ug=rwx,o=rx /home/nas/public
```

Nach dem Anlegen der Verzeichnisse weisen wir den Benutzer *root* mit der Gruppe *users* dem Verzeichnis *public* zu und erlauben Schreib-, Lese- und Ausführ-Rechte für Nutzer (*u*) und Gruppe (*g*) sowie Lese- und Ausführ-Rechte für alle anderen (*o=others*). In der *Samba*-Konfigurationsdatei (`sudo vi /etc/samba/smb.conf`) fügen wir folgende Ergänzungen ein:

Code-Ausschnitt 8.6.2

```
1 [public]
2 comment = Public Storage
3 path = /home/nas/public
4 valid users = @users
5 force group = users
6 create mask = 0660
7 directory mask = 0771
8 read only = no
```

Anschließend speichern wir diese Datei und starten den *Samba*-Dienst neu.

Code-Ausschnitt 8.6.3

```
1 sudo /etc/init.d/samba restart
```

Danach steht das öffentliche Verzeichnis *public* zur Verfügung und jeder darf nach Herzenslust darauf Daten ablegen bzw. von dort Daten lesen und Programme ausführen. Auf ähnliche Art und Weise können wir auf dem Raspberry Pi einen *ftp*-Server einrichten. Dabei steht *ftp* für *File Transfer Protocol*, also ein Protokoll, das die Übertragung von Dateien erlaubt, und zwar Netz-übergreifend. Die Installation des *ftp*-Servers gestaltet sich gewohnt einfach.

Code-Ausschnitt 8.6.4

```
1 apt-get install proftpd
```

Die Installation fragt, ob *proftpd* als Dienst (*Inetd*) oder als eigener Server laufen soll. Wenn Sie *ftp* nur ab und an benötigen, empfehle ich das Einrichten als Dienst. Der Dienst kann dann jederzeit mit den bekannten Kommandos gestartet und angehalten werden. Soll der Raspberry Pi aber permanent ausschließlich als *NAS* arbeiten, bietet sich der Server-Modus an. Ändert man die Konfiguration von *proftpd* nicht, hat ausschließlich der Benutzer *pi* Zugriff auf sein *home*-Verzeichnis */home/pi*. Wir sehen hier aber ein Verzeichnis vor, welches wieder auf der externen Festplatte liegt, die wir wie oben beschrieben, eingebunden haben. Erstellen wir zuerst einen neuen Benutzer mit einem neuen Kennwort, der *ftp* benutzen kann.

Code-Ausschnitt 8.6.5

```
1 sudo addgroup ftpgruppe
2 sudo adduser ftpbenutzer --ingroup ftpgruppe --shell /bin/false
```



Und wieder lernen Sie neue LINUX-Befehle kennen. Der Befehl *addgroup* erstellt eine neue Benutzergruppe. Der Befehl *adduser* erstellt einen neuen Benutzer. In unserem Beispiel weisen wir ihn direkt der Gruppe *ftpgruppe* zu und sorgen dafür, dass er sich nicht lokal am System anmelden kann (*--shell /bin/false*).

Nach Eingabe des letzten Befehls müssen Sie ein Kennwort für den *ftp*-Benutzer vergeben. Mit diesem Kennwort kann der Benutzer sich später von extern für den Transfer von Dateien einwählen. Erstellen wir nun ein *ftp*-Verzeichnis auf der extern eingebundenen Festplatte, über das wir unseren Dateiverkehr abwickeln wollen.

Code-Ausschnitt 8.6.6

```
1 mkdir /home/nas/ftp
2 chown root:ftpgruppe -R /home/nas/ftp
3 chmod 775 -R /home/nas/ftp
```

Das gerade erstellte Verzeichnis müssen wir noch für *proftpd* freigeben. Dazu editieren Sie bitte die Datei */etc/proftpd/proftpd.conf* mit vorangestelltem *sudo* und ändern sie wie folgt:

Code-Ausschnitt 8.6.7

```
1 <Directory /home/nas/ftp>
2     UserOwner root
3     GroupOwner ftpgruppe
4 </Directory>

6 # Include other custom configuration files
7 Include /etc/proftpd/conf.d/
```


Die beiden Einträge *UserOwner* und *GroupOwner* stellen sicher, dass alle hochgeladenen Daten dem Benutzer *root* und der Gruppe *ftpgruppe* gehören. In derselben Datei müssen Sie das Kommentarzeichen vor *RequireValidShell* entfernen.

Code-Ausschnitt 8.6.8

```
1 RequireValidShell off
```

Damit stellen wir sicher, dass unser Benutzer, der sich ja nicht direkt am System anmelden darf, den *ftp*-Server benutzen kann. Nach diesen Änderungen muss *proftpd* neu gestartet werden. Wenn Sie möchten, fügen Sie einen automatischen Start des Servers nach einem Systemneustart hinzu.

Code-Ausschnitt 8.6.9

```
1 service proftpd restart
2 update-rc.d proftpd defaults
```



Wenn Sie außerhalb Ihres Netzwerkes auf den *FTP*-Server zugreifen wollen, müssen Sie den Port 21 an den Raspberry Pi weiterleiten.

Unter Windows können Sie den Explorer zum Zugriff auf den *FTP*-Server benutzen. Geben Sie dazu `ftp://ip-adresse` ein. Sie werden dann automatisch nach dem Benutzer und dem Kennwort gefragt.



Sie können sich die Benutzer- und Kennwort-Abfrage sparen, wenn Sie beides im *ftp*-Aufruf unterbringen: `ftp://benutzer:kennwort@ip-adresse`.

Zusammenfassung 8 Das war das Kapitel „Server und Datenbanken“, das Ihnen gezeigt hat, welche Netzwerkdienste der Raspberry Pi zur Verfügung stellen kann. Ob als Drucker-server oder zum Datenaustausch, die Palette der Möglichkeiten ist vielfältig. Sie haben gelernt, wie man einen eigenen Hotspot einrichten kann, der eine komplette Benutzer-Authentifizierung ermöglicht. Den kompletten Hotspot haben wir dann Schritt für Schritt mit Hilfe von *TOR* anonymisiert.

Ich habe Ihnen erklärt, welche Rolle Datenbanken dabei spielen. Wir haben sie zur Authentifizierung, aber auch im Rahmen von *Content Management-Systemen* kennengelernt. Eines dieser Systeme (*Contao*) haben wir installiert. Vielleicht haben Sie ja Lust bekommen, das eine oder andere Web-Projekt mit einem *CMS* umzusetzen?

Wem der *apache*-Webserver zu viel Ballast mitbringt, wird im nächsten Kapitel fündig. Hier werde ich Ihnen unter anderem den kleinsten Webserver der Welt vorstellen, der mit wenigen Zeilen Python-Code auskommt. Wir werden uns die *GPIO*-Pins des Raspberry Pi genauer anschauen und lernen, wie man mit ihrer Hilfe zusätzliche Hardware wie z. B. ein Display oder programmierbare Elektronik-Bausteine steuern kann. ■

9 — Erweiterungen



Abbildung 9.1: Busware macht's möglich: Raspberry Pi mit Display, 868 MHz Transceiver, RTC (*Real Time Clock*) und IR-Diode (Quelle: Busware).

In diesem Kapitel möchte ich Ihnen Erweiterungsmöglichkeiten für den Raspberry Pi näher bringen. Wir starten mit einem minimalistischem Webserver, der in Python programmiert ist. In ihn binden wir ein Modul ein, welches die Ansteuerung der GPIO-Pins ermöglicht. Aus Geschwindigkeitsgründen habe ich dabei den Umweg über C/C++ gewählt. Mit diesem Server kann man andere Geräte steuern. Weiterhin schließen wir ein Display mit Touchscreen an den Raspberry Pi an. Ich zeige Ihnen, wie man die Konsole, die X-Oberfläche oder sogar das Fernsehprogramm auf dem Display darauf darstellen kann. Aber der Pi kann noch mehr. Das Kapitel über GPIB zeigt Ihnen, wie Messgeräte angeschlossen und bedient werden können. Die Raspberry Pi Kamera-Erweiterung rundet dieses Kapitel ab. Lernen Sie, wie man die Kamera anschließt und bedient oder wie man das Kamerabild in einem Webserver sichtbar macht.

Das ist bereits die erste Komponente, die wir für den Bau einer Hausautomatisierung oder Alarmanlage benötigen. Aber darum geht es ja erst im nächsten Kapitel.

9.1 Erweiterungs-Anschlüsse

Die Stiftleiste des Raspberry Pi haben Sie bereits im Kapitel 5.3 kennengelernt. Wir schauen uns nun alle Erweiterungs-Anschlüsse (Abb. 9.2) noch einmal im Detail an. Einige davon werden wir im Laufe dieses Kapitels benutzen. Lassen Sie mich zunächst die Bedeutung der einzelnen Begriffe erklären:

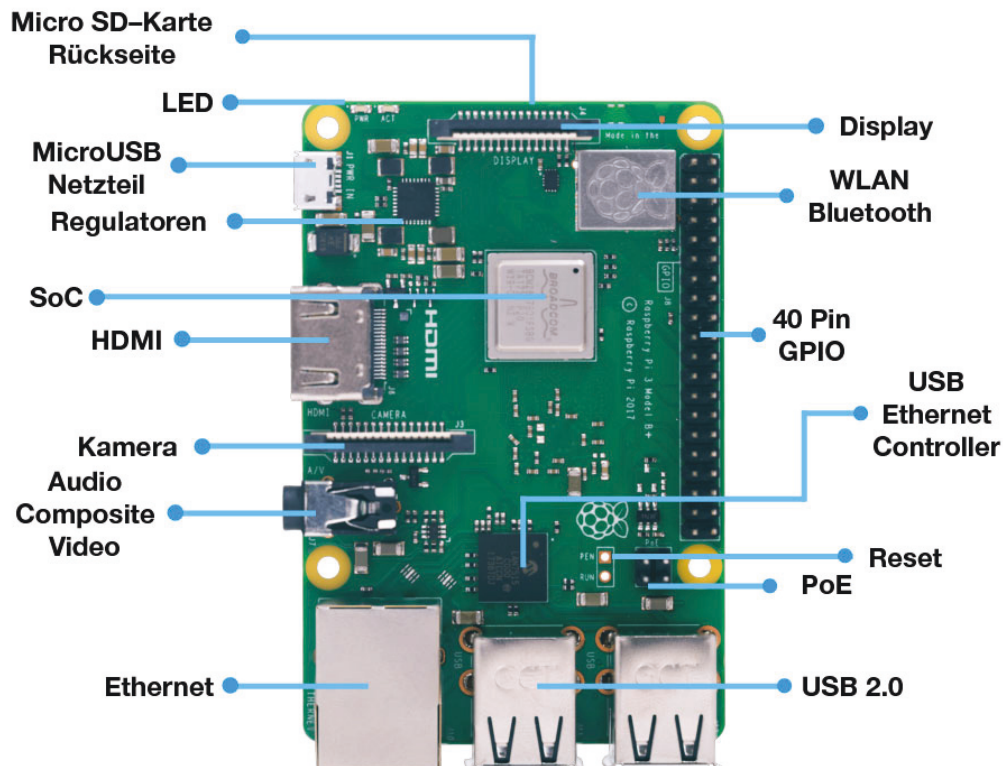


Abbildung 9.2: Alle Erweiterungs-Anschlüsse des Raspberry Pi auf einen Blick.

- *Display Serial Interface DSI*. Der Raspberry Pi verfügt über einen Display-Anschluss, an den DSI-Bildschirme angeschlossen werden können.
- *Camera Serial Interface CSI*. An diesem Port kann die Raspberry Pi-Kamera angeschlossen werden. Diese werden wir uns ebenfalls in diesem Kapitel näher anschauen.
- *General Purpose Input Output GPIO*. Die größte Stiftleiste des Raspberry Pi bietet gleich mehrere freie Schnittstellen, über die LEDs, Sensoren, Displays und andere Geräte angesteuert werden können. Insgesamt besteht diese Leiste aus 40 Pins. Zwei davon stellen eine Spannung von 5 V bereit, zwei weitere eine Spannung von 3,3 V. Acht Pins dienen als Masse-Verbindung. Die verbleibenden Pins sind frei programmierbar. Einige davon können besondere Funktionen übernehmen: Fünf Pins können als SPI-Schnittstelle verwendet werden. Alle Schnittstellen werden gleich separat erklärt. Vier dieser Pins können als I2C-Schnittstelle verwendet werden. Zwei Pins können als UART-Schnittstelle verwendet werden.

- *I2C*. Der I2C-Bus wurde von Philips (heute NXP Semiconductors) entwickelt und ist ein serieller Datenbus. Er wird heute hauptsächlich für die Kommunikation verschiedener Schaltungsteile verwendet und war ursprünglich konzipiert, um verschiedene Chips in Fernsehgeräten steuern zu können.
- *Serial Peripheral Interface SPI*. SPI wurde von Motorola entwickelt und bildet ebenfalls einen seriellen Datenbus. Es gibt 3-Draht SPI und 4-Draht SPI, je nachdem, ob nur Lesen oder auch Schreiben auf dem Bus realisiert wird.
- *Universal Asynchronous Receiver Transmitter UART*. Auch dieses System dient der Realisierung einer seriellen Schnittstelle. Dabei kann es sich sowohl um ein eigenständiges elektronisches Bauelement als auch um einen Funktionsblock eines höherintegrierten Bauteils handeln. Diese Schnittstelle ist im industriellen Umfeld sehr verbreitet, z. B. als RS232-Interface.
- *Puls-Weiten-Modulation PWM*. Bei der Puls-Weiten-Modulation wechselt eine technische Größe (z. B. eine Spannung) zwischen zwei Werten. Bei einer konstanten Frequenz wird der Tastgrad eines Rechteckimpulses moduliert.

Um die Erweiterungs-Anschlüsse des Raspberry Pi zu nutzen, müssen die Kernel-Module für SPI und I2C aktiviert werden. Das können Sie entweder in der Datei `/boot/config.txt` erledigen

Code-Ausschnitt 9.1.1

```
1 dtparam=i2c_arm=on
2 dtparam=spi=on
```

oder mit Hilfe des Tools `raspi-config`. Wenn Sie den I2C-Bus nutzen wollen, erweitern Sie bitte gleichzeitig die Datei `/etc/modules` um folgenden Eintrag:

Code-Ausschnitt 9.1.2

```
1 i2c-dev
```

und installieren Sie die notwendigen I2C-Werkzeuge mit

Code-Ausschnitt 9.1.3

```
1 sudo apt-get install i2c-tools
```

Die I2C-Treiber und -Programme gehen davon aus, dass die Gruppe `i2c` den Bus direkt ansprechen darf. Möchten Sie das als Benutzer `pi`, ohne jedes Mal ein `sudo` vor den Befehl zu stellen, fügen Sie `pi` zur Gruppe `i2c` hinzu.

Code-Ausschnitt 9.1.4

```
1 sudo adduser pi i2c
```

Schauen wir uns ein erstes kleines Projekt an, bei dem wir ohne zusätzliche Hardware-Kosten (mit Ausnahme eines Stückchens Draht) auskommen. Wir benutzen die PWM-Erweiterungspins, um einen Radiosender zu „bauen“. Der Betrieb ist allerdings in Deutschland verboten, wenn der Sender eine Leistung von mehr als 50 nW hat. Das reicht üblicherweise, um eine Entfernung von bis zu zehn Metern zu überbrücken. Würde man einen kleinen Draht an einen der GPIO-Pins klemmen (GPIO Pin 4), könnte man mit dem Raspberry Pi eine Reichweite von bis zu 100 Metern überbrücken (http://www.icrobotics.co.uk/wiki/index.php/Turning_the_Raspberry_Pi_Into_an_FM_Transmitter).

Wir schauen uns das Ganze - zumindest in der Theorie - einmal näher an. Der Quelltext zum Radiosender stammt vom Imperial College (Robotik Society) in London und wurde im Rahmen eines #pihack-Wochenendes für ein Schulprojekt entwickelt. Der Sender kann auf UKW-Frequenzen senden (und darüber hinaus). Das frequenzmodulierte Signal selbst wird durch eine Modulation der GPIO-Clock erzeugt. In der aktuellen Softwareversion kann der Sender sogar in Stereo senden. Hifi-Qualität darf man aber nicht erwarten. Laden wir im ersten Schritt den Quelltext der Applikation im Verzeichnis `/home/pi/pifm_radio` herunter und packen ihn aus.

Code-Ausschnitt 9.1.5

```
1 mkdir ~/fm_radio
2 cd fm_radio
3 wget http://omattos.com/pifm.tar.gz
4 tar xvzf pifm.tar.gz
```

Die ausführbare Applikation `pifm` liegt bereits nach dem Auspacken vor. Wer selber kompilieren will, kann das mit dem Befehl

Code-Ausschnitt 9.1.6

```
1 g++ -O3 -o pifm pifm.c
```

Der Parameter `-O3` bedeutet das Einschalten vieler Optimierungen. Die Installation des `g++`-Compilers setze ich hier voraus (Kapitel 4). Der Sender unterstützt sogar Stereo-Ausgabe. Bereit für einen ersten Test? Dann los:

Code-Ausschnitt 9.1.7

```
1 sudo ./pifm left_right.wav 103.3 22050 stereo
```

Das Kommando spielt die Datei `left_right.wav` auf einer Frequenz von 103.3 MHz ab. Die Abtastrate der `wav`-Datei ist 22050 KHz, die Ausgabe erfolgt in `stereo`. Die Abspielfrequenz darf übrigens zwischen 1 MHz und 250 MHz liegen. Die o. g. Webseite erklärt weitere Möglichkeiten, etwa das Abspielen einer `mp3`-Datei mittels

Code-Ausschnitt 9.1.8

```
1 ffmpeg -i input.mp3 -f s16le -ar 22.05k -ac 1 - | sudo ./pifm -
```

oder Senden von Sprache via USB-Mikrofon mittels

Code-Ausschnitt 9.1.9

```
1 arecord -d0 -c2 -f S16_LE -r 22050 -twav -D copy | sudo ./pifm -
```

Bitte beachten Sie, dass für die oberen beiden Aufrufe `ffmpeg` und `arecord` installiert sein müssen. Erledigen Sie das, falls noch nicht geschehen, mit

Code-Ausschnitt 9.1.10

```
1 sudo apt-get install ffmpeg alsa-utils
```



Sie können das Programm *mplayer* dazu benutzen, Sample-Raten von Audiodateien zu ändern. Nach der Installation via `sudo apt-get install mplayer` hilft Ihnen vielleicht das unten angehängte Beispiel.

Code-Ausschnitt 9.1.11

```
1 mplayer -quiet -vo null -vc dummy -af resample=22050,channels=1 -ao pcm:↔
   waveheader:file=file_out_22kHz.wav file_in.wav
```

Das FM-Radio bringt auch ein Python-Skript namens *PiFm.py* mit. Dieses macht aber nichts anderes, als das kompilierte C++-Programm aufzurufen.

9.2 Servieren à la Python

Darf ich Ihnen den kürzesten Webserver-Code der Welt vorstellen?

Code-Ausschnitt 9.2.1

```
1 import cherrypy
2 class HelloWorld(object):
3     def index(self):
4         return "Hello World!"
5 index.exposed = True
7 cherrypy.quickstart(HelloWorld())
```



(Quelle: Rolando Murillo)

Das glauben Sie nicht? Wir probieren es gleich aus. In diesem Abschnitt werden wir zunächst einen kleinen, minimalistischen Webserver mit Python aufsetzen. Danach schauen wir uns ein C-Programm an, mit dessen Hilfe wir das *SPI* programmieren können. Ich zeige Ihnen, wie Sie mit wenigen Schritten aus dem C-Programm ein Modul für Python erstellen können. Dieses Python-Modul werden wir in unseren Webserver einbauen, den wir dann über einen WLAN-Hotspot zur Verfügung stellen. So können wir mit unserem Handy externe Geräte steuern.

9.2.1 Ein minimalistischer Webserver

Ich bin Ihnen den Beweis schuldig. Installieren Sie bitte die *CherryPy*-Erweiterung für Python.

Code-Ausschnitt 9.2.2

```
1 sudo apt-get install python-cherrypy3 python-dev
```

Gleichzeitig benötigen wir die Python-Entwicklungsumgebung (*python-dev*), die wir später noch verwenden werden, um unser C-Modul einzubinden. Erstellen Sie ein Verzeichnis `/home/pi/python`. Wechseln Sie in dieses Verzeichnis und editieren Sie eine Datei namens *server.py*, in die Sie den oben gezeigten minimalistischen Webserver-Quelltext hinein kopieren. Speichern Sie die Datei und starten Sie sie mit *python*.

Code-Ausschnitt 9.2.3

```
1 cd /home/pi/python
2 python ./server.py
```

Bei mir erscheint daraufhin die folgende Ausgabe:

Code-Ausschnitt 9.2.4

```

1 [15/Jun/2018:11:13:12] ENGINE Listening for SIGHUP.
2 [15/Jun/2018:11:13:12] ENGINE Listening for SIGTERM.
3 [15/Jun/2018:11:13:12] ENGINE Listening for SIGUSR1.
4 [15/Jun/2018:11:13:12] ENGINE Bus STARTING
5 CherryPy Checker:
6 The Application mounted at '' has an empty config.
8 [15/Jun/2018:11:13:12] ENGINE Started monitor thread '_TimeoutMonitor'.
9 [15/Jun/2018:11:13:12] ENGINE Started monitor thread 'Autoreloader'.
10 [15/Jun/2018:11:13:12] ENGINE Serving on 127.0.0.1:8080
11 [15/Jun/2018:11:13:12] ENGINE Bus STARTED

```

Rufen Sie einen Browser auf dem Raspberry Pi auf und geben Sie dort bitte die Adresse `http://localhost:8080` ein. 8080 ist der Port, auf dem unser Minimal-Server läuft. Mein Midori (`sudo apt-get install midori`) liefert das Bild, das ich Ihnen in der nächsten Abb. zeige. Sogar eine Kirsche als Favicon wird angezeigt. Wenn Sie den Server von einem anderen Rechner im

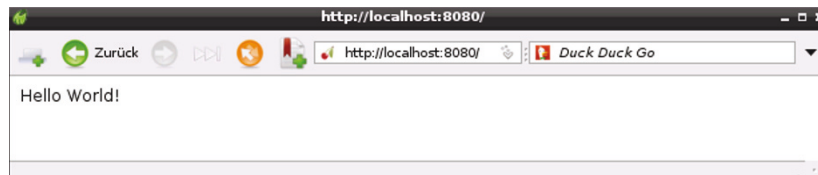


Abbildung 9.3: So sieht der minimalistische CherryPy-Server auf Midori aus.

selben Netz aufrufen, wird dieser vorerst nicht funktionieren, da er an die Adresse 127.0.0.1, also den *localhost*, gebunden ist und nicht an die IP-Adresse des Raspberry Pi. Das werden wir zu einem späteren Zeitpunkt noch ändern. Aber wie Sie in Abb. 9.3 sehen, liefert der Server die Ausgabe, die er liefern soll: *Hello World!* Und dafür haben wir nur sechs Zeilen Programm-Quelltext verwendet. Glauben Sie es jetzt?

Der folgende Abschnitt wird ein wenig komplizierter, weil wir dort ein C-Programm schreiben werden, mit dem wir das *SPI*-Protokoll innerhalb der GPIO-Leiste verwenden können.

9.2.2 GPIO-Ansteuerung

Bevor wir mit C in die Programmierung einsteigen, möchte ich Ihnen noch ein paar nützliche Tipps und Befehle an die Hand geben, mit denen man auf die Schnelle GPIO-Pins setzen kann.

Setzen per Terminal

Am schnellsten kann man GPIO-Pins mit Hilfe von Terminal-Befehlen setzen. Da diese GPIO-Pins Logik-Pins sind, bedeutet das, dass sie entweder *an* oder *aus* sind. In der Digitalwelt nennt man diese Signal-Pegel auch *high* oder *low*. Sie entsprechen dabei bestimmten Spannungen, die Sie mit einem Multimeter an den Pins des Raspberry Pi nachmessen können. Die Raspberry Pi-Pins sind 3,3 V-Pins. Ein logisches *high* entspricht also einer Spannung von 3,3 V, ein logisches *low* einer Spannung von 0 V. Das ist schon das ganze Geheimnis der GPIO-Pins: Man kann die Spannung an- oder ausschalten oder man kann abfragen, ob eine Spannung anliegt oder nicht. Üblicherweise nimmt man Toleranzen der Logik-Pegel von 20 % an (*low* bis 0,66 V, *high* ab 2,64 V).

Info Bitte seien Sie vorsichtig beim Anlegen von Spannungen an die GPIO-Pins. Es gibt Logik-Spannungspegel von 5 V. Legen Sie eine Spannung von 5 V an einen Pin, der nur für 3,3 V maximale Spannung ausgelegt ist, können Sie zuschauen, wie die Magie aus Ihrem Raspberry Pi raucht. Für diese Zwecke gibt es spezielle Pegel-Konverter (Spannungswandler), die die Spannung von 5 V oder mehr auf 3.3 V reduzieren.

Info Noch ein wichtiger Hinweis an alle, die sofort „Vorwiderstand“ schreien, um 5 V auf 3,3 V zu reduzieren: Alle Lösungen mit Vorwiderständen reduzieren die Flankensteilheit der Signale. Dies kann bei Takt- und Zählengängen zu unerwünschten Schwingungen und damit Fehlzählungen führen. Die internen Pull-Up- und Pull-Down-Widerstände des Raspberry Pi haben einen Widerstand von ca. 50 ... 60 kΩ.

Info Eine letzte Bemerkung zum Thema Spannungen und Ströme: Aus dem 3,3 V Pin der GPIO-Leiste darf maximal 50 mA Strom gezogen werden, aus dem 5 V Pin maximal 1 A minus dem Strom, den der Raspberry Pi selbst verbraucht. In der alten Pi1 B-Variante sind das 700 mA. Damit ist der maximale Strom, den man in der B-Variante (das ist der Raspberry Pi mit Ethernet-Buchse) aus dem 5 V GPIO-Pin ziehen darf, 300 mA.

Die ganze Digital-Magie besteht nun darin, An- und Auszustände in einer bestimmten Reihenfolge an bestimmten Pins anzulegen. Bei einem SPI (*Serial Peripheral Interface*) haben die Pins meist bestimmte Namen (dazu später mehr). Immer gibt es eine sogenannte *clock*, die nichts anderes macht, als ein rhythmisches An- und Aus-Signal mit einer bestimmten Frequenz (also Wiederholrate) zu erzeugen. Ein Rechtecksignal mit einer Frequenz von 100 MHz hat eine Periodendauer von 10 ns. Diese Relation können Sie mit der folgenden Gleichung berechnen:

$$t = \frac{1}{f}, \quad (9.1)$$

wobei t die Periodendauer und f die Frequenz des Signals darstellt. Die Periodendauer ist also der Kehrwert der Frequenz und umgekehrt. Schauen wir uns jetzt die Terminal-Befehle an, mit denen wir GPIO-Pins setzen können: Bevor man einen Pin ansteuern kann, muss dieser nach jedem Neustart erst einmal freigeschaltet werden. Das Arbeiten mit den Pins setzt außerdem Root-Rechte voraus, sofern man die Berechtigungen nicht ändern will. Wir beginnen also mit einem `sudo su` und nehmen uns den GPIO-Pin 17 vor.

Code-Ausschnitt 9.2.5

```
1 sudo su
2 echo "17" > /sys/class/gpio/export
3 echo "out" > /sys/class/gpio/gpio17/direction
4 echo "1" > /sys/class/gpio/gpio17/value
5 echo "17" > /sys/class/gpio/unexport
```

Was bedeuten diese Zeilen? Die erste ist klar, werden Sie antworten. In der zweiten Zeile teilen wir dem System mit, dass wir den Pin 17 benutzen wollen. In der dritten Zeile sagen wir dem Raspberry Pi, dass er die Spannung am Pin 17 setzen soll. Das Gegenteil von *out* ist *in*. Dann würden wir einlesen wollen, ob am Pin 17 eine Spannung anliegt oder nicht. In der vierten Zeile setzen wir Pin 17 auf ein logisches *high*, also auf „1“ und schalten damit die Spannung für diesen Pin ein. Ein entsprechender Aufruf mit einer „0“ würde die Spannung (wieder) ausschalten. Die letzte Zeile setzt alle bisherigen Aufrufe wieder zurück. Mit diesen Befehlen und ein wenig Skript-Programmierung könnten Sie also schon eine LED blinken lassen, was in vielen Büchern als Projekt beschrieben ist. Keine Sorge, ich werde es gleich auch kurz erwähnen. Zum Einstieg ist das sicherlich toll. Aber wir wollen mehr. Unser Ziel ist es, ein SPI zu programmieren, welches dann mit externen Bausteinen kommunizieren kann.

WiringPi

WiringPi ist ein hilfreiches Programm, mit dem man auf die GPIO-Pins des Raspberry Pi zugreifen kann. Es liefert Bibliotheken für viele Programmiersprachen, darunter C/C++, Python und PHP. WiringPi hat inzwischen eine eigene Webseite (<http://wiringpi.com>). Wir laden das Programmpaket per *git* herunter, übersetzen und installieren es. Dazu verwenden wir wieder das Verzeichnis *python*.

Code-Ausschnitt 9.2.6

```
1 cd ~/python
2 git clone git://git.drogon.net/wiringPi
3 cd WiringPi
4 git pull origin
5 ./build
```

Der Befehl *git pull origin* checkt den *origin*-Zweig aus, der letzte Updates enthält. Das *build*-Skript wird Sie an einer Stelle nach Ihrem Kennwort fragen, bevor nach der Übersetzung die Installation beginnt. Nach erfolgter Installation können Sie die Bibliothek mit dem Programm *gpio* testen.

Code-Ausschnitt 9.2.7

```
1 gpio -v
2 gpio readall
```

Der erste Befehl zeigt die Version des Programmes an, der zweite liest alle GPIO-Register aus. Überträgt man die Terminal-Befehle von oben auf das *GPIO*-Programm, sehen die Befehle sehr einfach aus:

Code-Ausschnitt 9.2.8

```
1 gpio export 17 out
2 gpio -g write 17 1
3 gpio -g read 17
```

Der Parameter *read* liest im oberen Beispiel Pin 17 aus. Die Antwort ist *1* für *high* und *0* für *low*.



Es ist wichtig zu erwähnen, dass WiringPi im Vergleich zur Abb. 5.6 eine andere GPIO-Belegung benutzt. Die Option *-g* weist WiringPi an, die normale Raspberry Pi Belegungs-nomenklatur zu verwenden.

Setzen mit C/C++

Jetzt schauen wir uns kurz das LED-Beispiel (*Light Emitting Diode*) an (Abb. 9.4). Wir versorgen eine LED über Pin 11 mit Spannung. Der 11. Pin entspricht dem GPIO-Pin 17 bzw. dem WiringPi-Pin 0. Über Pin 14 wird die LED über einen 270 Ω -Widerstand mit Masse verbunden. Der Widerstand ist erforderlich, damit die Diode nicht durchbrennt. Wir lassen sie nun mit einer Frequenz von 1/s per C-Programm blinken. Dabei greifen wir auf die *wiringPi.h*-Bibliothek zurück. Erstellen Sie in einem Verzeichnis Ihrer Wahl die Datei *led.cpp* mit dem folgenden Inhalt:

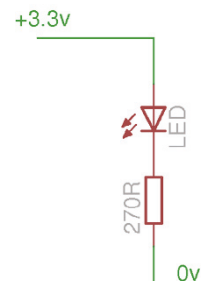


Abbildung 9.4: LED-Verdrahtung.

Code-Ausschnitt 9.2.9

```

1 // Hier binden wir WiringPi ein sowie die C-Standard Ein-/Ausgabe
2 #include <wiringPi.h>
3 #include <stdio.h>

4
5 int main() {

6
7     // Initialisieren der WiringPi API
8     if (wiringPiSetup() == -1)
9         return 1;

10
11    // Setze GPIO 17 (=WiringPi Pin 0) als Ausgangspin
12    pinMode(0, OUTPUT);

13
14    // Start einer Endlosschleife
15    while(1) {
16        // Spannung einschalten, LED leuchtet.
17        digitalWrite(0, 1);

18
19        // 1 s warten. Wartezeit wird in ms angegeben.
20        delay(1000);

21
22        // Spannung ausschalten. LED geht aus.
23        digitalWrite(0, 0);

24
25        // Wieder 1 s warten
26        delay(1000);
27    }
28 }

```

Übersetzen Sie das Programm mit

Code-Ausschnitt 9.2.10

```
1 g++ led.cpp -o led -lwiringPi
```

Der Parameter *-lwiringPi* weist den Linker (das ist der Compiler-Teil, der ein lauffähiges Programm zusammenbindet) an, die WiringPi-Bibliothek einzubinden, die u. a. den verwendeten *digitalWrite*-Befehl beinhaltet. Um auf die GPIO-Pins zugreifen zu können, führen Sie das Programm bitte mit Root-Rechten aus.

Code-Ausschnitt 9.2.11

```
1 sudo ./led
```

Falls Sie keine LED zur Hand haben (auf richtige Polung achten!), können Sie auch mit einem Multimeter die Spannung am betreffenden Pin nachmessen. Diese sollte im 1-Sekunden-Rhythmus von 3,3 V auf 0 V wechseln und umgekehrt. Eine komplette Beschreibung der WiringPi API (*Application Programming Interface*) erhalten Sie unter <https://projects.drogon.net/raspberry-pi/wiringpi/functions/>. Mit WiringPi können Sie auch Pins auslesen. Die entsprechenden Befehle lauten dann:

Code-Ausschnitt 9.2.12

```
1 pinMode(5, INPUT);
2 unsigned int result = digitalRead(5);
```

Der erste Befehl setzt WiringPi Pin 5 als *INPUT* Pin, der zweite Befehl fragt diesen Pin ab und speichert das Ergebnis in der Variablen *result* ab. Das Ergebnis ist entweder *1* oder *0*, je nachdem, ob eine Spannung am Pin anliegt oder nicht.

Mit dieser Funktion könnten Sie z. B. einen simplen Wassermelder bauen. Ändert sich der Widerstand Ihres Sensors, schaltet dieser einen GPIO-Pin auf Masse. Der Raspberry Pi fragt diesen Pin ab. Sobald der Wert von 1 auf 0 fällt, wissen Sie, dass der Melder Alarm geschlagen hat. Eine Bauanleitung dazu gibt es unter <http://fritzing.org/projects/raspberry-pi-water-sensor>. Wir kümmern uns im nächsten Kapitel um das Thema Hausautomatisierung. In diesem Zusammenhang werde ich Ihnen eine professionelle Lösung mit dem Raspberry Pi präsentieren.

Setzen mit Python

Für die Skriptsprache Python gibt es ebenfalls eine GPIO-Bibliothek, die sehr einfach zu verwenden ist. Sie wird mit

Code-Ausschnitt 9.2.13

```
1 sudo apt-get install python-rpi.gpio
```

installiert. Hier möchte ich Ihnen noch ein Python-Programm vorstellen, welches eine LED blinken lässt (gemäß des oberen Beispiels), sobald Pin 18 (GPIO 24) mit Pin 14 (Masse) verbunden wird. Die Nummerierung der Python-Bibliothek entspricht der Pin-Nummerierung des Raspberry Pi. Editieren Sie bitte in einem Verzeichnis Ihrer Wahl die Datei *led.py*, die so aussieht:

Code-Ausschnitt 9.2.14

```
1 import time
2 import RPi.GPIO as GPIO

4 # Nummerierung wie Raspberry Pi verwenden
5 GPIO.setmode(GPIO.BOARD)

7 # Pin 18 (GPIO 24) auf Input setzen
8 GPIO.setup(18, GPIO.IN)

10 # Pin 11 (GPIO 17) auf Output setzen
11 GPIO.setup(11, GPIO.OUT)

13 # Endlosschleife starten
14 while 1:
15     # LED immer ausmachen
16     GPIO.output(11, GPIO.LOW)

18     # GPIO lesen
19     if GPIO.input(18) == GPIO.HIGH:
20         # LED an
21         GPIO.output(11, GPIO.HIGH)

23     # Warte 1 s
24     time.sleep(1.0)

26     # LED aus
27     GPIO.output(11, GPIO.LOW)

29     # Warte 1 s
30     time.sleep(1)
```

Starten Sie das Programm mit

Code-Ausschnitt 9.2.15

```
1 sudo python gpio.py
```

9.2.3 SPI

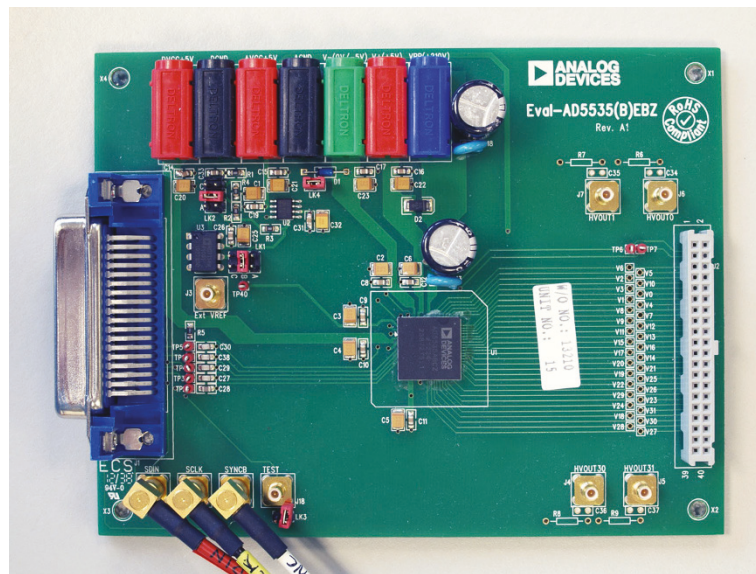


Abbildung 9.5: Der AD5535 ist ein 14-Bit Gleichstrom-DAC.

Das Serial Peripheral Interface *SPI* nutzt ein serielles Protokoll zur Programmierung vieler Elektronikbausteine. Ich möchte Ihnen dieses Protokoll mit einer kompletten Webserver-Anwendung am Beispiel des Analog Devices Board *AD5535* aus Abb. 9.5 näher bringen. Dieses Board beinhaltet 32 14-Bit DACs (*Digital Analog Converter*) und kann auf 32 Kanälen eine Spannung zwischen 0 V und 200 V ausgeben. Das Protokoll ist sehr einfach und kann leicht auf beliebige andere Bauteile übertragen werden, die Sie dann ebenfalls mit dem Raspberry Pi ansteuern und programmieren können. Werfen wir zunächst einen Blick auf das *Timing*-Diagramm. Dieses sagt uns, wie das serielle Protokoll im Detail aussieht (Abb.9.6). Die Timing-Diagramme fin-

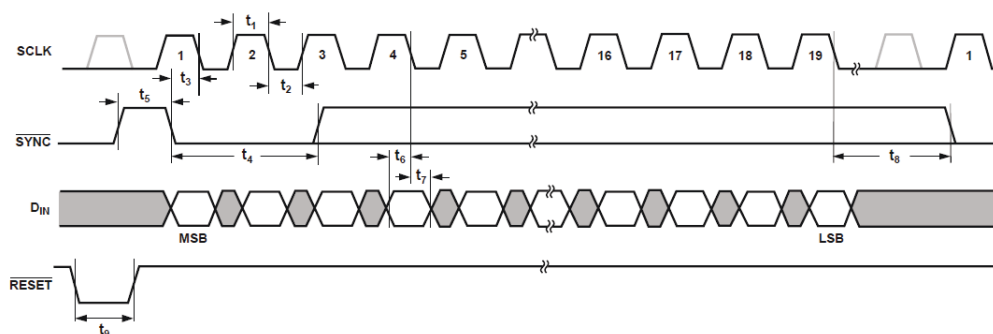


Abbildung 9.6: Das AD5535 Timing-Diagramm aus dem Datenblatt.

den Sie üblicherweise im Datenblatt des Herstellers. Auf den ersten Blick sieht das Diagramm vielleicht verwirrend aus. Schauen wir genauer hin. Das SPI-Protokoll nutzt drei Leitungen:

- *SCLK*: Das ist die *Serial Clock*, unser Taktgeber. Laut Datenblatt darf diese Clock maximal 30 MHz schnell sein. Mit Hilfe dieses Signals werden die Daten in ein Gerät übertragen oder aus einem Gerät gelesen.

- *SYNC*: Das ist die Synchronisations-Leitung. Nimmt sie einen bestimmten Wert an, weiß das zu programmierende Gerät, dass Daten gesendet werden oder Daten empfangen werden sollen. Es gibt zwei Möglichkeiten: Bei *low*-aktiven Geräten muss dieses Signal von *high* auf *low* gehen, damit Daten ab diesem Zeitpunkt übernommen werden, bei *high*-aktiven Geräten ist es genau anderes herum. Der AD5535 ist ein *low*-aktives Gerät. Geht also das *Sync*-Signal von *high*=an auf *low*=aus, werden Daten, die am D_{in} -Pin (Data in) anliegen, ins das Gerät gelesen.
- D_{in} : Auf dieser Leitung werden die Daten gesendet, die in das Gerät übertragen werden sollen. Bei *low*-aktiven Geräten startet die Übertragung, wenn das *SYNC*-Signal von *high* auf *low* gegangen ist.

Sind Sie noch bei mir? Wir haben drei Leitungen: Eine gibt den Takt vor, die andere sagt, wann Daten gesendet oder empfangen werden sollen, die dritte beinhaltet die Daten selbst. Alle Leitungen können an oder aus sein. Der Wechselrhythmus zwischen an und aus entscheidet, wann und was gesendet oder empfangen wird. Der hier verwendete AD5535-Chip kann übrigens keine Daten senden. Wir gestalten die folgenden Programme aber so kompatibel, dass sie das Empfangen gesendeter Daten selbst implementieren können, wenn Ihr Gerät das unterstützt. Üblicherweise haben diese Geräte dann eine vierte Leitung, die sich (analog zu D_{in}) D_{out} für ausgehende Daten nennt. Der Raspberry Pi bringt eine Bibliothek mit, die man hervorragend für die SPI-Programmierung verwenden kann. Diese nennt sich *bcm2835*. Wir installieren sie in einem neuen Verzeichnis, welches wir z. B. *spi* nennen. Die Bibliothek erhalten Sie auf <http://www.airspayce.com/mikem/bcm2835/>. Laden Sie die letzte Version herunter und kopieren sie in das gerade erstellte *spi*-Verzeichnis. Dann packen wir sie aus. Zu dem Zeitpunkt, als ich dieses Buch geschrieben habe, war die Version 1.56 aktuell.

Code-Ausschnitt 9.2.16

```
1 mkdir ~/spi
2 cd spi
3 tar xvfz bcm2835-1.56.tar.gz
4 cd bcm2835-1.56
```

Die Bibliothek enthält leider noch einen kleinen Fehler, den wir vor der Übersetzung korrigieren: Das *SYNC*-Signal wird nicht zurückgesetzt, wenn die Datenübertragung zu Ende ist. Ändern Sie bitte im Verzeichnis *src* die Datei *bcm2835.c* so, dass der folgende Aufruf ersetzt wird:

Code-Ausschnitt 9.2.17

```
1 //bcm2835_peri_set_bits(paddr, 0, BCM2835_SPI0_CS_TA);
2 bcm2835_peri_set_bits(paddr, BCM2835_SPI0_CS_TA, BCM2835_SPI0_CS_TA);
```

Nach der Korrektur können Sie die Bibliothek übersetzen und installieren.

Code-Ausschnitt 9.2.18

```
1 ./configure
2 make
3 sudo make check
4 sudo make install
```

Schreiben wir zuerst ein kleines C-Programm, welches das SPI gemäß des AD5535-Datenblattes programmiert. Bitte verbinden Sie dabei den Chip, den Sie programmieren möchten, gemäß Abb. 5.6 (SPI). Die Data-in-Leitung ist dabei MISO (*Master In Slave Out*). Bei einem 4-Draht SPI entspricht die Data-out-Leitung MOSI (*Master Out Slave In*). Die drei Leitungen, die den AD5535 mit dem Raspberry Pi verbinden, habe ich selbst gebaut und mit Steckern versehen.

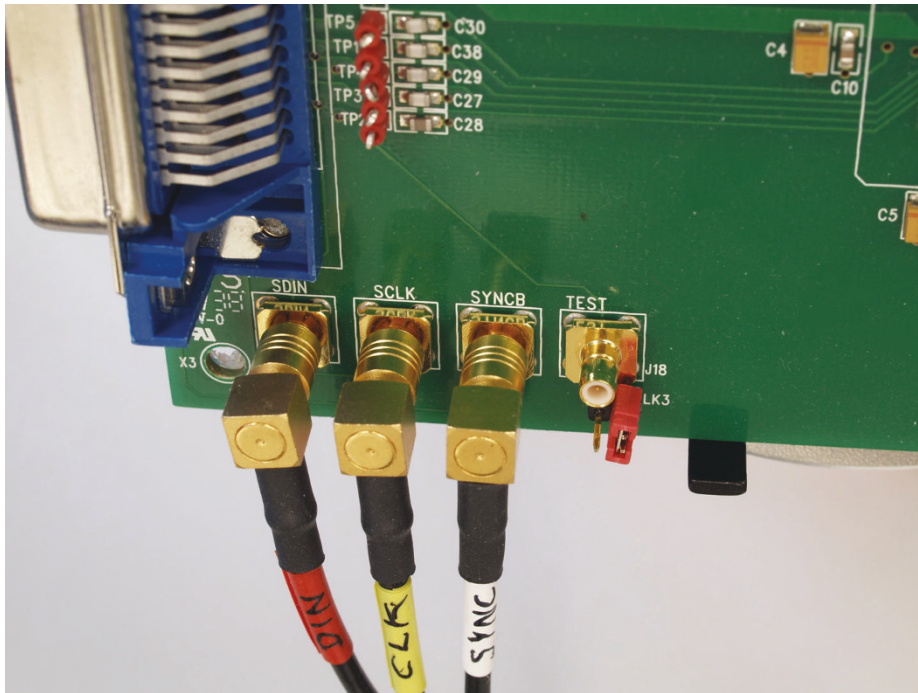


Abbildung 9.7: Der AD5535 benötigt drei Leitungen: Data-in, CLK und Chip-enable.

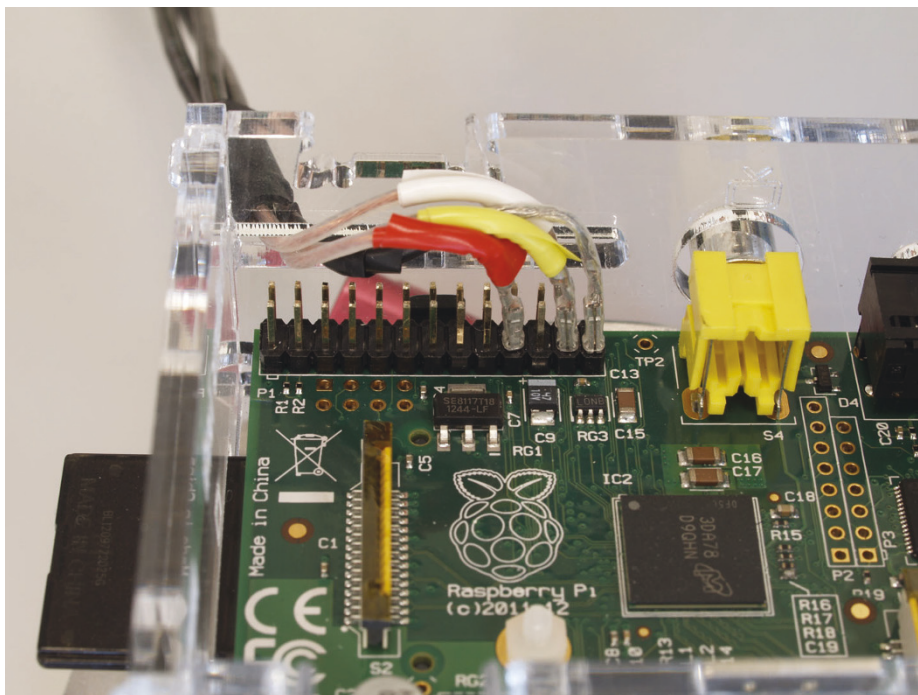


Abbildung 9.8: Die Massen aller drei SPI-Leitungen sind am Raspberry Pi zusammengeführt und auf Masse gesteckt. Das Bild zeigt einen Raspberry Pi der ersten Generation. Beim neuen Modell verwenden Sie die gleichen Pins von links oben aus gezählt.

Für einen Programmierertest bei kleiner Geschwindigkeit kann man normales, geschirmtes Kabel verwenden. Die Schirmung jedes Kabels habe ich am Raspberry Pi zusammengeführt und dann auf Masse gesteckt (PIN 25, Abb. 9.8).



Sie finden alle hier erklärten Programme inklusive des Quelltextes im Download-Bereich <https://www.springer.com/de/book/978-3-662-58143-8> unter dem Namen *spi.tar.bz2*.

Dabei gehen wir - gemäß Datenblatt - von folgenden Voraussetzungen aus: Die Daten stellen ein 19 Bit breites Wort dar. Die ersten 5 Bit (A4 ... A0) repräsentieren den zu programmierenden DAC-Kanal (0 ... 31). Die verbleibenden 14 Bit (D13 ... D0) stellen den DAC-Wert für den vorher ausgewählten Kanal ein. Das hochwertigste Bit (MSB, *Most Significant Bit*) wird zuerst gesendet. Da die *bcm2835*-Bibliothek nur mit Vielfachen von Bytes umgehen kann, füllen wir die oben genannten 19 Bit um 5 weitere auf, die wir nicht verwenden. Wir kommen dann auf 24 Bit. Das sind genau 3 Byte.

Code-Ausschnitt 9.2.19

```

1 // AD5535 SPI interface for Raspberry Pi
2 #include <bcm2835.h>
3 #include <stdio.h>
4 #include <cstdlib>
5 #include <math.h>
6 #include <string.h>
7 #include <unistd.h>
8 #include <stdint.h>
9 #include <fcntl.h>
10 #include <sys/ioctl.h>
11 #include <linux/spi/spidev.h>
12 #include <stdio.h>
13 #include <errno.h>
14 #include <stdlib.h>
15 #include <string>
16 #include <iostream>

17 inline unsigned int bintoint(char *binstr)
18 { // Binärzahl in Integer-Wert umwandeln
19     unsigned int zahl=0;
20     zahl|=(binstr[7]&1)<<(0);
21     zahl|=(binstr[6]&1)<<(1);
22     zahl|=(binstr[5]&1)<<(2);
23     zahl|=(binstr[4]&1)<<(3);
24     zahl|=(binstr[3]&1)<<(4);
25     zahl|=(binstr[2]&1)<<(5);
26     zahl|=(binstr[1]&1)<<(6);
27     zahl|=(binstr[0]&1)<<(7);
28     return zahl;
29 }

30

31
32 int main(int argc, char **argv)
33 {
34     if (!bcm2835_init())
35     {
36         printf("Fehler: Konnte BCM2835 nicht initialisieren");
37         return 1;
38     }
39     bcm2835_spi_begin();
40     bcm2835_spi_setBitOrder(BCM2835_SPI_BIT_ORDER_MSBFIRST); //MSB zuerst
41     bcm2835_spi_setDataMode(BCM2835_SPI_MODE1); //Default Modus
42     // AD5535 maximal 30 MHz. Wir verwenden 3.9 MHz
43     bcm2835_spi_setClockDivider(BCM2835_SPI_CLOCK_DIVIDER_64);
44     bcm2835_spi_chipSelect(BCM2835_SPI_CS0); // The default
45     bcm2835_spi_setChipSelectPolarity(BCM2835_SPI_CS0, LOW); //SYNC ist low-enable

46
47     double vmax = 210; // 210 V Maximalspannung
48     int channel = 3; // DAC Kanal zum Testen (0...31)
49     double vout = 9.3; // Testspannung auf Kanal 3 soll 9.3V sein

50
51 // 14bit DAC bedeutet 2**14=16384 Zustände
52 int val = (int)(vout/vmax * ((1<<14)-1)); // DAC-Wert berechnen

```

```

54 char data_string[]="000000000000000000000000"; // 19bit Wort (Datenblatt) plus 5 dummy ←
    bits = 3 Byte
55 for (int x=4; x>=0; x--) if (channel & 1<<x) data_string[4-x]='1';
56 for (int x=13; x>=0; x--) if (val & 1<<x) data_string[18-x]='1';

58 char *sub1, *sub2, *sub3;
59 sub1 = strdup(data_string, 8);
60 sub2 = strdup(data_string+8, 8);
61 sub3 = strdup(data_string+16, 8); // 3Byte-Wert zusammenbauen

63 uint32_t length = 3;
64 char data[3];
65 data[0] = bintoint(sub1);
66 data[1] = bintoint(sub2);
67 data[2] = bintoint(sub3); //3Byte-Wert zusammenfassen

69 bcm2835_spi_writenb(data, length); // SPI schreiben

71 bcm2835_spi_end(); // SPI schliessen und beenden
72 bcm2835_close();
73 return 0;
74 }

```

Übersetzen Sie das Programm mit

Code-Ausschnitt 9.2.20

```
1 g++ -fpermissive -Wall spi.c -o spi -lbcm2835
```

Nach der Übersetzung können Sie das Programm durch den Aufruf von `./spi` starten. In unserem Beispiel würde auf Kanal 3 des DACs ein Wert von 9,3 V bei einer maximalen Spannung von 210 V ausgegeben werden. Die wichtigsten Programmteile sind:

- Die *include*-Anweisungen laden Header der Bibliotheken ein, deren Befehle im Laufe unseres Programmes benutzt werden.
- Die Funktion *bintoint* wandelt eine Binärzahl in einen Integer-Wert um. In den darauf folgenden Befehlen, die mit *bcm2835_spi_* beginnen, initialisieren wir den SPI-Treiber, stellen seine *clock*-Frequenz ein und legen die Bit-Folge *MSB* fest.
- Nachdem die zu übertragenden Bytes anhand der Registertabelle des Datenblattes berechnet wurden, werden diese zusammengefasst (*data*) und über die GPIO-Pins des Raspberry Pi ausgesendet.
- Abschließend wird das SPI-Interface wieder geschlossen.

Die Konstante `BCM2835_SPI_CLOCK_DIVIDER_64` verdient noch unsere Aufmerksamkeit. Die letzte Zahl (in unserem Beispiel 64) gibt den Teiler zu 250 MHz an. Wir stellen hier also eine Clock-Frequenz $250 \text{ MHz} / 64 = 3,9 \text{ MHz}$ ein. Die Zahlen zum Einstellen der Clock-Frequenzen müssen Potenzen von 2 sein.

9.2.4 Von C nach Python

Nachdem unser C-Programm einwandfrei funktioniert, möchte ich Ihnen in diesem Abschnitt zeigen, wie wir das C-Programm in Python einbinden. Das ist sehr hilfreich, wenn es innerhalb von Python auf Geschwindigkeit ankommt. Beginnen wir mit dem Quelltext, auf dessen Besonderheiten ich gleich eingehe. Wir speichern ihn im *spi*-Verzeichnis unter dem Namen *spi.cc*. Auslassungspunkte (...) habe ich da eingeführt, wo sich der neue Quelltext nicht von dem alten unterscheidet. Das verbessert die Lesbarkeit des Quelltextes (nächste Seite). Um einen C-Quelltext als Python-Modul zur Verfügung zu stellen, binden wir die Include-Datei *Python.h* ein.

Weiterhin definieren wir eine Funktion *STANDARD_ERROR*, die im Fehlerfall eine Fehlermeldung ausgibt. Objekte, die wir später von Python aus aufrufen wollen, werden als *PyObject* definiert und über *PyMethodDef* zur Verfügung gestellt. Der Export der neuen Bibliothek erfolgt in der *DL_EXPORT*-Sektion, die mit *Py_InitModule(„spi“, moduleMethods)* ein Modul namens *spi* mit den unter *moduleMethods* aufgelisteten Funktionen für Python bereitstellt. Ich weiß, das war ein Schnelldurchgang. Es gibt aber sehr gute Internet-Literatur zu diesem Thema (<http://docs.python.org/2/extending/extending.html>).

Code-Ausschnitt 9.2.21

```

1  ...
2  #include <Python.h> //
3  ...
4  #define STANDARD_ERROR(s) { PyErr_SetString(PyExc_StandardError,s); return #NULL; }
5  #define VMAX 210.0 // Maximum voltage of AD5535 device
6  inline unsigned int bintoint(char *binstr)
7  {
8  ...
9  }
10 double round(double number)
11 {
12     return number < 0.0 ? ceil(number - 0.5) : floor(number + 0.5);
13 }
14 double get_double(PyObject *a,int b)
15 {
16     PyObject *c = PySequence_GetItem(a,b);
17     if(!c) return 0.0;
18     return PyFloat_AsDouble(c);
19 }
20 int get_byte(PyObject *a, int b)
21 {
22     PyObject *c = PySequence_GetItem(a, b);
23     if(!c) return 0;
24     return PyLong_AsLong(c);
25 }
26 static PyObject *spi_hello(PyObject *self, PyObject *args) {
27     printf("***** AD5535 SPI-Interface.\n");
28     return Py_BuildValue("i", 0);
29 }
30 static PyObject *spi_write(PyObject *self, PyObject *args) {
31     // get two values: channel (int) and voltage (double)
32     int channel;
33     double vout;
34     if (!PyArg_ParseTuple(args, "id", &channel, &vout))
35         STANDARD_ERROR("Parsing Args");
36     if (!bcm2835_init())
37         ...
38     bcm2835_spi_setChipSelectPolarity(BCM2835_SPI_CS0, LOW); // The default
39     int val = (int) (vout/VMAX * ((1<<14)-1));
40     char data_string[]="0000000000000000000000";
41     for (int x=4; x>=0; x--) if (channel & 1<<x) data_string[4-x]='1';
42     for (int x=13; x>=0; x--) if (val & 1<<x) data_string[18-x]='1';
43     char *sub1, *sub2, *sub3;
44     ...
45     uint32_t length = 3;
46     char data[3];
47     ...
48     bcm2835_spi_writenb(data, length);
49     bcm2835_spi_end();
50     bcm2835_close();
51     return Py_BuildValue("i", 0);
52 }
53 static PyMethodDef moduleMethods[] = {
54     {"spi_hello", spi_hello, METH_VARARGS, 0},
55     {"spi_write", spi_write, METH_VARARGS, 0},
56     {0}
57 };
58 extern "C" DL_EXPORT(void) initspi(void) {
59     PyObject *m;
60     m = Py_InitModule("spi",moduleMethods);
61 };

```

Im nächsten Schritt werden wir den Quelltext übersetzen, um die neue Python-Bibliothek zu erzeugen, und wir werden diese kurz testen. Im Anschluss werden wir diese Bibliothek dann in einen CherryPy-Server einbauen, damit wir externe Geräte vom Tablet aus programmieren und damit bedienen können. Beginnen wir mit der Übersetzung der C-Bibliothek für Python.

Hier gibt es leider keinen einfachen Übersetzungsbefehl, den wir sofort eintippen könnten. Wir müssen eine Anweisungsdatei für Python generieren, die dann selbst den eigentlichen Übersetzungsvorgang initiiert. Wir nennen diese Datei *setup.py*, editieren sie im *spi*-Verzeichnis und füllen sie mit folgendem Inhalt:

Code-Ausschnitt 9.2.22

```

1  from distutils.core import setup,Extension
2  import sys,os,string,glob

4  sys.argv = ["","build_ext","--inplace"]

6  spi = Extension('spi',
7                  define_macros = [('MAJOR_VERSION', '1'),
8                                    ('MINOR_VERSION', '0')],
9                  include_dirs = ['/usr/include'],
10                 libraries = ['m', 'bcm2835'],
11                 library_dirs = ['/usr/lib'],
12                 sources = ['spi.cc'])

14  setup (name = 'Raspberry Pi SPI Wrapper',
15         version = '1.0',
16         description = 'SPI Wrapper for AD5535',
17         author = 'R. Follmann',
18         author_email = 'rudi@follmann.name',
19         url = 'www.ready4us.de',
20         long_description = '''
21  This is Raspberry Pi SPI wrapper for AD5535.
22  ''' ,
23         ext_modules = [spi])

```

In dieser Datei teilen wir Python mit, dass wir eine Bibliothek übersetzen möchten. Wir wählen den Quelltext *spi.cc* und geben auch die Bibliotheken an, die zum Übersetzen benötigt werden (*bcm2835*). Eine kurze Beschreibung des Codes darf auch dabei sein. Stoßen wir das Übersetzen an mit

Code-Ausschnitt 9.2.23

```

1  python setup.py

```

Der Übersetzungsvorgang erstellt die Datei *spi.so*, ein sog. *shared object*. Das ist eine Bibliothek, die Python einladen kann, um dann damit arbeiten zu können. Testen wir es: Schreiben Sie ein kleines Python-Test-Programm *spi_test.py* mit dem folgenden Inhalt:

Code-Ausschnitt 9.2.24

```

1  import spi as ad5535
2  if name=='__main__':
3      ad5535.spi_hello()
4      ad5535.spi_write(3, 10.0)

```

Kurz, oder? Wir binden unsere gerade übersetzte *spi*-Bibliothek unter dem Namen *ad5535* ein. Dann tätigen wir zwei Aufrufe: Aus der Bibliothek (die jetzt *ad5535* heißt, weil wir sie unter diesem Namen importiert haben) rufen wir zunächst die Funktion *spi_hello()* auf. Diese macht nichts anderes, als ***** *AD5535 SPI-Interface*. auf den Bildschirm zu schreiben. Danach schreiben wir in unseren Baustein auf Kanal 3 den Wert 10.0, setzen also den 3. DAC auf 10 V. Damit können wir nun sehr einfach einen Webserver programmieren, der unser Gerät bedienen kann.

9.2.5 Ansteuerung per Webserver

Speichern Sie den unten angegebenen Quelltext im Verzeichnis `/home/pi/spi` unter dem Namen `server_simple.py`.

Code-Ausschnitt 9.2.25

```

1  #!/usr/bin/env python

3  import os.path, sys, spi, cgi, cherrypy
4  from cherrypy.lib.static import serve_file
5  from cherrypy import wsgiserver

7  global form_body, v0
8  v0 = 0.0 # Anfangswert

10 # Server Default-Adresse und Port
11 IP_ADDRESS = "192.168.178.66"
12 PORT      = 8080
13 SERVER_DIR = os.path.dirname(os.path.abspath(__file__))

15 class Root(object):
16     def template(self, body):
17         return ("""
18             <html><head><title>Pi Web AD5535</title>
19             <link rel="stylesheet" href="styles/style.css" type="text/css"></head><←
                body>%s</body></html>
20             """ % body)

22     def index(self):
23         global v0
24         form_body = ("""
25         <font color="#DE002E"><h2>Pi AD5535 web interface</font></h2></font><form action←
                ="/confirmation">
26         """)
27         form_body += ("""<p>
28             V0: <input type="text" name="V0" style="width:50px" value=%s><br><br><←
                p />
29             <input type="submit" name="_action" value="Senden" style="height:28px;←
                font-size:13pt"><p
30             """)%(float(v0))
31         return self.template(form_body)
32     index.exposed = True

34     def confirmation(self, V0, _action, **kwargs):
35         global v0, form_body
36         action = cgi.escape(_action)
37         try:
38             v0_new = float(cgi.escape(V0));
39         except:
40             v0_new = 0.0;

42         spi.spi_write(0, v0_new);
43         print >> sys.stderr, "***** Writing successful\n"
44         return self.index()
45     confirmation.exposed = True
46 if __name__ == '__main__':
47     spi.spi_hello()
48     root = Root()
49     IMAGE_DIR = SERVER_DIR+"/images"
50     STYLE_DIR = SERVER_DIR+"/styles"
51     INI_FILE = SERVER_DIR+"/server.ini"
52     conf = {
53         "/": {
54             "tools.digest_auth.on": True,
55             "tools.digest_auth.realm": 'Raspi web server',
56             "tools.digest_auth.users": {
57                 "pi": "pi",
58             },

```

```

59     },
60     'global': {'request.show_tracebacks': False, # suppress python
61     traceback
62         'engine.autoreload.on': False}, # make it work with bbfreeze
63     '/images': {'tools.staticdir.on': True,
64         'tools.staticdir.dir': IMAGE_DIR},
65     '/styles': {'tools.staticdir.on': True,
66         'tools.staticdir.dir': STYLE_DIR},
67     '/favicon.ico': {'tools.staticfile.on': True,
68         'tools.staticfile.filename': '/home/pi/spi/tux.png'},
69 }
70 cherrypy.config.update({'server.socket_host': IP_ADDRESS,
71     'server.socket_port': PORT})
72 cherrypy.quickstart(root, config = conf)

```

Zum Starten benötigt der Server Root-Rechte, damit er über unser *spi*-Modul auf die GPIO-Pins des Raspberry Pi zugreifen darf. Starten Sie den Server mit

Code-Ausschnitt 9.2.26

```
1 sudo python server_simple.py
```

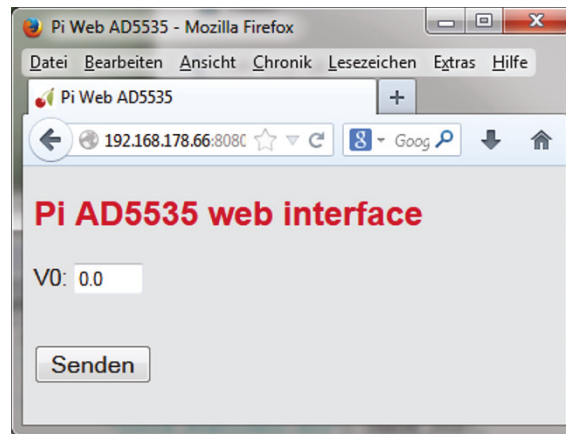


Abbildung 9.9: Der vereinfachte SPI-Webserver für den AD5535 Chip.

Das Ergebnis sollte so ähnlich aussehen wie das in Abb. 9.9. Ich möchte Sie noch mit einigen Besonderheiten dieses kleinen Webservers vertraut machen. Die hier abgedruckte Version ist eine Minimal-Version, die lediglich die Eingabe eines Wertes erlaubt und diesen über unser *spi*-Modul an das zu programmierende Gerät übergibt. Das Protokoll (*Timing Diagramm*) können Sie mit einem Oszilloskop mitlesen, falls Sie eines besitzen. Der abgebildete Server-Quelltext erlaubt weiterhin das Setzen einer IP-Adresse und eines Ports, auf dem er läuft. Bitte ersetzen Sie meine IP-Adresse mit Ihrer. Weitere Zeilen zeigen, wie man eigene Favicons einbinden kann oder *CSS (Cascading Style Sheets)* Stylesheets. Diese legen das prinzipielle Aussehen der Webseite fest.

DL Der Download-Bereich <https://www.springer.com/de/book/978-3-662-58143-8> enthält alle abgebildeten Quelltexte und einen vollwertigen SPI-Server (Lesen, Schreiben, Speichern) unter dem Namen *spi_all.tar.bz2*.

Info Sie können den Python-Webserver durch Eingabe von *CTRL-c* stoppen.

Ich möchte Ihnen zum Abschluss dieses Abschnitts noch ein kleines Programm vorstellen, mit dem man schöne Terminal-Installationsprogramme generieren kann. Mit *makeself* kann man selbst-extrahierende Archive erstellen.

Das Programm wird mit

Code-Ausschnitt 9.2.27

```
1 sudo apt-get install makeself
```

installiert. Rufen Sie *makeself --help* für eine ausführliche Hilfe auf. Sie können mit diesem Programm mehrere Dateien zu einer zusammenpacken. Beim Aufruf der neuen Datei werden die Dateien im Archiv ausgepackt und nach Bedarf sogar gestartet. Sie können auf diese Art und Weise z. B. mittels *PyInstaller* (Kapitel 4.2) den Webserver mit allen Initialisierungsdateien einpacken und jemandem als installierbares Programmpaket weitergeben.

9.2.6 Sage mir, wie Du heißt

Wenn ich für meine Kunden einen Webserver wie den gerade vorgestellten programmiere, wäre es für sie lästig, sich IP-Adressen merken zu müssen, um auf den Server zuzugreifen. Schöner wäre es, dem Kind einen Namen zu geben. Sie geben doch auch lieber *fritz.box* als *192.168.178.1* ein, oder? In diesem Abschnitt zeige ich Ihnen, wie man das mit einem sogenannten *Nameserver* erledigen kann. Dafür verwende ich den *stärksten* unter ihnen, *powerdns*. Wir installieren ihn zusammen mit einigen kleinen Hilfsprogrammen, die wir benötigen werden, um die Installation des *Nameservers* zu testen.

Code-Ausschnitt 9.2.28

```
1 sudo apt-get install pdns-server dnsutils
```

Beginnen wir damit, unseren eigenen *Nameserver* zu konfigurieren, indem wir zunächst eine Sicherung einbauen: Namen, die wir nicht auflösen können, weil wir sie nicht gespeichert haben, leiten wir einen anderen *Nameserver* weiter und bitten diesen, die Namen in IP-Adressen aufzulösen. Dafür bietet sich wieder der öffentliche *Google-Nameserver* mit der IP-Adresse *8.8.8.8* an. Für die durchzuführenden Änderungen an der Konfigurationsdatei */etc/powerdns/pdns.conf* greifen wir auf *sed* zurück.

Code-Ausschnitt 9.2.29

```
1 sudo sed -i 's/# recursor=/recursor=8.8.8.8/g' /etc/powerdns/pdns.conf
2 sudo sed -i 's/allow-recursion=127.0.0.1/allow-recursion=127.0.0.1,192.168.0.0\24/g' /etc/powerdns/pdns.conf
```

In der zweiten Zeile teilen wir der Konfiguration mit, welcher IP-Adressbereich bei Google (*8.8.8.8*) anfragen darf, nämlich *192.168.*.**. Sollten Sie ein anderes privates Netz haben, passen Sie den Eintrag entsprechend an. Nachdem wir diese Änderungen durchgeführt haben, starten wie den *Nameserver*-Dienst neu.

Code-Ausschnitt 9.2.30

```
1 sudo service pdns restart
```

Zeit für einen ersten Test: Fragen Sie den jetzt lokal laufenden *Nameserver* doch einmal, wie die IP-Adresse von *google.de* lautet:

Code-Ausschnitt 9.2.31

```
1 nslookup google.de localhost
```

Wenn Sie eine IP-Adresse erhalten, ist alles korrekt installiert und der *Nameserver* läuft lokal (*localhost*) auf dem Raspberry Pi.

Einstellen der Zone

Der *Nameserver* funktioniert zwar, kennt aber noch nicht unsere eigenen „Namen“. Das ändern wir nun, indem wir eine neue *Zone* konfigurieren. In der Datei `/etc/powerdns/bindbackend.conf` geben wir die neu zu erstellende *Zone* an. Das beinhaltet auch deren Aufenthaltsort im Verzeichnisbaum. Ich habe die *Zone* `pi.home` genannt.

Code-Ausschnitt 9.2.32

```
1 zone "pi.home" {
2     type master;
3     file "/etc/powerdns/bind/pi.home.zone";
4     allow-update { none; };
5 };
```

Die Datei `pi.home.zone` erstellen wir im nächsten Schritt. Auf diese Art und Weise können Sie beliebig viele Zonen definieren und einbinden. Um die neue Zonen-Datei anzulegen, müssen wir zunächst das *bind*-Verzeichnis erstellen, in dem die Zonen-Dateien liegen sollen.

Code-Ausschnitt 9.2.33

```
1 sudo mkdir /etc/powerdns/bind
```

In diesem Verzeichnis editieren wir danach die Datei `pi.home.zone` mit folgendem Inhalt:

Code-Ausschnitt 9.2.34

```
1 $ORIGIN pi.home ; base for unqualified names
2 $TTL 1h ; default time-to-live
3 @ IN SOA ns.pi.home hostmaster.pi.home (
4     1; serial
5     1d; refresh
6     2h; retry
7     4w; expire
8     1h; minimum time-to-live
9 )
10 IN NS ns
11 IN A 192.168.178.66
12 ns IN A 192.168.178.66
```

In dieser Datei habe ich folgende Einstellungen vorgenommen:

- 192.168.178.66 ist die IP-Adresse meines Raspberry Pi. Ersetzen Sie diese Adresse bitte mit der Ihres Raspberry Pi.
- Der erste Eintrag ist der *SOA*-Eintrag (*Start Of Authority*). Er teilt dem *Nameserver* die primäre Datenquelle für die entsprechende *Zone* mit.
- *IN* steht für Internet und ist die Zonenklasse.
- *serial* ist die Seriennummer. Sie wird bei jeder Änderung inkrementiert und dient als Hinweis darauf, wann die *Zone* zuletzt aktualisiert worden ist,
- *Refresh* ist der Sekundenabstand, in dem die *Slaves* anfragen, ob sich etwas geändert hat.
- *Retry* ist der Sekundenabstand, in denen ein *Slave* den Kontakt wiederholt, falls sein *Master* nicht antwortet.
- *Expire*: Wenn der *Master* auf einen Zonentransfer-Request nicht reagiert, deaktiviert ein *Slave* nach dieser Zeitspanne in Sekunden die *Zone*.

Am Ende setzen wir einen *Nameserver* (*NS*)-Eintrag. Er zeigt auf die IP-Adresse unseres gerade eingerichteten *Nameservers*. Es folgt die IP-Adresse der *Zone* selbst und ein Alias auf *ns*. Dieses ist mit meiner IP-Adresse verbunden.

Nach diesen Änderungen muss der *Nameserver*, wie oben angegeben, neu gestartet werden. Lust auf einen kleinen Test? Suchen wir nach unserer neuen Zone *pi.home* auf dem lokalen Raspberry Pi:

Code-Ausschnitt 9.2.35

```
1 nslookup pi.home localhost
```

Bei mir lautet die Antwort:

Code-Ausschnitt 9.2.36

```
1 Server:          localhost
2 Address:         127.0.0.1#53

4 Name:   pi.home
5 Address: 192.168.178.66
```

Das Programm *nslookup* teilt mir mit, dass es den lokalen (127.0.0.1) *Nameserver* nutzt, der auf Port-Nummer 53 läuft. Es übersetzt die Anfrage nach *pi.home* mit der korrekten IP-Adresse (192.168.178.66).



An dieser Stelle noch einmal der Hinweis: Wenn Sie *TOR* benutzen, darf auf Port 53 kein *Nameserver* laufen. Beenden Sie diesen bitte, bevor Sie *TOR* starten.



Sie können das Programm *netstat* nutzen, um nachzuschauen, welche Netzwerk-Dienste auf welchen Ports laufen. Rufen Sie im Terminal *netstat -nlp* auf und schauen Sie, welche Ports belegt sind.

Weitere Einträge

Einen weiteren Eintrag für eine Zone kann man in der Datei */etc/powerdns/bind/pi.home.zone* so hinzufügen:

Code-Ausschnitt 9.2.37

```
1 name           IN      A       192.168.178.67
2 namealias      IN      CNAME   webservers
```

Ersetzen Sie *name* durch den Namen, mit dem Sie die IP-Adresse (in meinem Beispiel 192.168.178.67) finden wollen. Prinzipiell können Sie dabei zwei verschiedene Einträge vornehmen: Den, der in Zeile 1 des Quelltextes abgebildet ist, oder einen Alias, den Sie in Zeile 2 finden. Verwenden Sie beim Alias das Schlüsselwort *CNAME*. Der Vorteil in der Verwendung eines Alias ist der: Sollte sich eine IP-Adresse ändern, muss diese nur einmal in der Zonen-Datei neu eingetragen werden. Die Aliase haben dann automatisch auch die geänderte IP-Adresse. Bitte denken Sie daran, *pdns* nach jeder Änderung neu zu starten, damit alle Änderungen übernommen werden.



Mit dem Befehl *ping* kann man unter LINUX nachschauen, ob sich ein Rechner im Netz befindet und in welcher Zeit er antwortet. Versuchen Sie das mit einem Aufruf von *ping ip-adresse*, wobei Sie eine beliebige IP-Adresse oder einen Namen verwenden können. Falls Sie sich wundern, dass *ping pi.home* ins Leere läuft: Geben Sie in der Datei */etc/resolv.conf* als *Nameserver* die IP Ihres Raspberry Pi an, nachdem Sie *pdns* installiert haben. Nach einem Neustart des Netzwerkdienstes steht Ihnen die neu eingerichtete Zone dann komplett zur Verfügung. Dann kann auch der Parameter *localhost* bei *nslookup* entfallen.

Wenn Sie den *Nameserver* richtig installiert haben, können Sie einen Webserver, der z. B. auf Port 8080 läuft, durch Eingabe von `http://pi.home:8080` erreichen.

9.2.7 Messtechnik

Viele Geräte, vor allem Messgeräte, werden nicht über *SPI*, sondern über *USB* oder *GPIB* gesteuert. Der *GPIB*-Bus (*General Purpose Interface Bus*), der auch *IEC-625-Bus* heißt, ist ein paralleler Bus zur Ansteuerung von Messgeräten. Er wurde von Hewlett Packard entwickelt. Die maximale Übertragungsrate liegt bei 1 Mbit/s. Die meisten modernen Messgeräte nutzen inzwischen *USB* zur Kommunikation mit dem Rechner. Ältere Messgeräte können mit einem *USB-GPIB*-Adapter (Abb. 9.10) angeschlossen werden. *LINUX* bietet *GPIB*-Treiber auf



(a) *GPIB*-Messgerät mit Buchse und Interface.



(b) *GPIB*-Kabel mit Interface.

Abbildung 9.10: *GPIB*-Kabel können übereinander gesteckt werden, um so mehrere Geräte zu vernetzen.

<http://linux-gpib.sourceforge.net/> an. Es würde den Rahmen dieses Buches sprengen, auf alle Details und Feinheiten des Treibers einzugehen. Daher werden an dieser Stelle nur die Schritte aufgelistet, die man bis zu einer erfolgreichen Inbetriebnahme von *GPIB* mit dem Raspberry Pi gehen muss.

1. Laden Sie sich den *GPIB*-Treiber von der oben genannten Adresse herunter.
2. Laden Sie sich für Ihren Adapter die passende Firmware von <http://linux-gpib.sourceforge.net/firmware/> herunter.
3. Packen Sie den Treiber aus und übersetzen Sie ihn mit den folgenden Schritten:

Code-Ausschnitt 9.2.38

```
1 cd treiber_verzeichnis
2 ./bootstrap
3 ./configure
4 make
5 sudo make install
```

4. Verbinden Sie Ihren *USB-GPIB*-Adapter mit dem Raspberry Pi über einen aktiven Hub.
5. Laden Sie die Kernel-Module, indem Sie zuerst das allgemeine *GPIB*-Modul und anschließend das Herstellermodul laden. Bei einem Agilent *USB*-Adapter sieht das so aus:

Code-Ausschnitt 9.2.39

```
1 sudo modprobe gpib_common
2 sudo modprobe agilent_82357a
```

- Rufen Sie den Befehl `lsusb` auf, der Ihnen alle angeschlossenen USB-Geräte zeigt. Merken Sie sich den *USB*-Bus (001) und die Device-Nummer (020).
- Installieren Sie das Programm `fxload`, mit dem die Firmware in das *GPIB*-Gerät übertragen werden kann:

Code-Ausschnitt 9.2.40

```
1 sudo apt-get install fxload
```

- Laden Sie die Hersteller-Firmware in Ihr *GPIB*-Gerät:

Code-Ausschnitt 9.2.41

```
1 sudo fxload -D /dev/bus/usb/001/020 -I /firmware_verzeichnis/agilent_82357a/82357a_
a_fw.hex
```

- Nehmen Sie mögliche Einstellungen in der Datei `/etc/gpib.conf` vor und laden Sie die Konfiguration:

Code-Ausschnitt 9.2.42

```
1 sudo gpib_config --minor 0
```

Mit dem Programm `ibtest` kann man das Interface ansprechen und testen. Bezüglich der *GPIB*-Kommandos erwartet das Programm dabei eine hexadezimale Eingabe (z. B. 0x3f).



Der Download-Bereich <https://www.springer.com/de/book/978-3-662-58143-8> enthält fertig kompilierte *GPIB*-Treiber unter dem Namen `gpib.tar.bz2`. Dieses Archiv enthält auch einige Beispielprogramme, die in Python programmiert wurden.

9.3 Zeige mir, was Du kannst

Es gibt viele kleine Bildschirme, die zusammen mit dem Raspberry Pi betrieben werden können. Die Auswahl reicht von kleinen, zwei-zeiligen Matrix-Displays bis hin zum 7-Zoll TFT (*Thin-Film-Transistor*) Touchscreen. Je nach Anwendungsfall macht es sicher Sinn, sich für das eine oder das andere Display zu entscheiden. Vor einigen Jahren wurden noch nicht alle TFT-Displays am Raspberry Pi unterstützt. Damals mussten Treiber von <https://github.com/notro/fbtt> heruntergeladen und in den Kernel integriert werden. Das ist heute nicht mehr erforderlich. Die Framebuffer-TFT-Treiber sind seit Januar 2015 Bestandteil des LINUX-Kernels und in der Raspbian-Distribution bereits enthalten.



Abbildung 9.11: Adafruit hat viele Displays im Programm (Quelle: Adafruit).

Sie müssen lediglich mit den richtigen Parametern aufgerufen werden, die sich von Display zu Display unterscheiden. Ich beschreibe im Folgenden die Installation des Busware-Modules (<http://busware.de/tiki-index.php?page=CCD>), welches für 89 € nicht nur ein Display, sondern auch noch eine RTC und einen 868 MHz-Transceiver bereitstellt. Darauf aufbauend werden wir im nächsten Kapitel den 868 MHz-Transceiver in Betrieb nehmen und unser Haus damit automatisieren. Ein Gehäuse für diese Platine schlägt mit weiteren 20 € zu Buche. Zum Betrieb des Displays ist es erforderlich, die folgenden Parameter in die Datei `/boot/config.txt` aufzunehmen:

Code-Ausschnitt 9.3.1

```
1 dtparam=spi=on
2 dtparam=i2c=on
3 dtoverlay=ads7846:cs=1,speed=2Mhz,penirq=25,penirq_pull=0,xmin=200,ymin=200,xmax=3900,↵
   ymax=3900,pmin=0,pmax=255,xohms=60,swapxy=0
```

Damit werden das SPI- und das I2C-Interface geladen und der TFT-Treiber mit den richtigen Parametern initialisiert. Fügen Sie bitte folgende Zeilen in die Datei `/etc/modules` ein:

Code-Ausschnitt 9.3.2

```
1 spi-bcm2708
2 i2c-bcm2708
3 i2c_dev
4 fbtft_device
5 ads7846
```

Die Einträge sorgen dafür, dass alle weiteren benötigten Kernel-Module (also Treiber) geladen werden. Erstellen Sie nun bitte eine Datei namens `/etc/modprobe.d/fbtft.conf` und füllen Sie sie mit folgendem Inhalt:

Code-Ausschnitt 9.3.3

```
1 options fbtft_device custom name=fb_ili9341 dma=1 buswidth=8 speed=48000000 gpios=reset↵
   :23,dc:24,led:22 rotate=90 bgr=1 cs=0
```

Damit weiß der Display-Treiber beispielsweise, mit welcher Geschwindigkeit er arbeiten kann und welche Rotation das Display hat. Starten Sie danach ihr System neu.

Um die RTC in Betrieb zu nehmen, sind folgende Schritte erforderlich. Fügen Sie in die Datei `/etc/rc.local` folgende Zeile an

Code-Ausschnitt 9.3.4

```
1 echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
2 sleep 1
3 hwclock -s
```

und entfernen Sie die sogenannte fake-RTC (also falsche RTC):

Code-Ausschnitt 9.3.5

```
1 apt-get remove fake-hwclock
2 rm /etc/cron.hourly/fake-hwclock
3 update-rc.d -f fake-hwclock remove
4 rm /etc/init.d/fake-hwclock
5 update-rc.d hwclock.sh enable
```

Der Lirc-Empfänger dieser Platine ist mit dem GPIO-Port 4 verbunden.

Die *dmesg*-Ausgabe zeigt Ihnen, wie die Treiber für das Display initialisiert und eingebunden werden.

Code-Ausschnitt 9.3.6

```

1  bcm2708_i2c_init_pinmode(0,0)
2  bcm2708_i2c_init_pinmode(0,1)
3  bcm2708_i2c bcm2708_i2c.0: BSC0 Controller at 0x20205000 (irq 79) (baudrate 100k)
4  bcm2708_spi bcm2708_spi.0: master is unqueued, this is deprecated
5  bcm2708_spi bcm2708_spi.0: SPI Controller at 0x20204000 (irq 80)
6  bcm2708_i2c_init_pinmode(1,2)
7  bcm2708_i2c_init_pinmode(1,3)
8  bcm2708_i2c bcm2708_i2c.1: BSC1 Controller at 0x20804000 (irq 79) (baudrate 100k)
9  ...
10 fbtft_device: SPI devices registered:
11 fbtft_device: 'fb' Platform devices registered:
12 fbtft_device:   bcm2708_fb id=-1 pdata? no
13 fbtft_device: GPIOs used by 'ili9341fb':
14 fbtft_device:   'reset' = GPIO23
15 fbtft_device:   'led' = GPIO24
16 fbtft_device: SPI devices registered:
17 fbtft_device:   ili9341fb spi0.0 32000kHz 8 bits mode=0x00
18 graphics fb1: ili9341fb frame buffer, 320x240, 150 KiB video memory, 16 KiB buffer ↔
   memory, fps=20, spi0.0 at 32 MHz
19 lirc_dev: IR Remote Control driver registered, major 247
20 lirc_rpi: module is from the staging directory, the quality is unknown, you have been ↔
   warned.
21 lirc_rpi: auto-detected active low receiver on GPIO pin 18
22 lirc_rpi lirc_rpi.0: lirc_dev: driver lirc_rpi registered at minor = 0
23 lirc_rpi: driver registered!
24 ...
25 ads7846_device: ads7846_device_init()
26 ads7846_device: SPI devices registered:
27 ads7846_device:   ili9341fb spi0.0 32000kHz 9 bits mode=0x00
28 ...
29 input: ADS7846 Touchscreen as /devices/platform/bcm2708_spi.0/spi_master/spi0/spi0.1/↔
   input/input0
30 ads7846_device: SPI devices registered:
31 ads7846_device:   ili9341fb spi0.0 32000kHz 9 bits mode=0x00
32 ads7846_device:   ads7846 spi0.1 500kHz 8 bits mode=0x00

```

Die Zeitstempel habe ich wieder aus Gründen der Übersichtlichkeit entfernt. Nach dem Neustart des Rechners sollte die LED-Hintergrundbeleuchtung des Displays leuchten. Sie kann mit dem Kommando

Code-Ausschnitt 9.3.7

```

1  sudo su
2  echo 0 > /sys/class/backlight/ili9341fb/bl_power
3  echo 1 > /sys/class/backlight/ili9341fb/bl_power

```

ein- und ausgeschaltet werden.

9.3.1 Ausgabe

Ist der TFT-Bildschirm erst einmal eingebunden, eröffnen sich vielfältige Möglichkeiten zur Ausgabe. Testen Sie das Display, bevor der Touchscreen justiert wird.



Sie werden sicherlich merken, dass der Touchscreen zwar auf Eingaben reagiert, wenn Sie z. B. die X-Oberfläche darauf ausgeben. Der Cursor-Pfeil erscheint aber noch nicht da, wo er sollte. Wir werden den Touchscreen später justieren.

Für einen ersten Test können Sie die X-Oberfläche nach dem Beenden mit folgendem Kommando aufrufen:

Code-Ausschnitt 9.3.8

```
1 sudo /etc/init.d/lightdm stop
2 FRAMEBUFFER=/dev/fb1 startx
```

Der *Framebuffer* `/dev/fb1` ist das angeschlossene TFT LCD. Um die Orientierung der Achsen für den Touchscreen zu korrigieren, installieren Sie bitte das Programm *xinput*.

Code-Ausschnitt 9.3.9

```
1 sudo apt-get install xinput
```

Dann ändern Sie bitte die Datei `/etc/X11/xinit/xinitrc` so:

Code-Ausschnitt 9.3.10

```
1 %
2 #!/bin/sh

4 # /etc/X11/xinit/xinitrc
5 #
6 # global xinitrc file, used by all X sessions started by xinit (startx)

8 # Touchscreen
9 DISPLAY=:0 xinput --set-prop 'ADS7846 Touchscreen' 'Evdev Axis Inversion' 0 1

11 # invoke global X session script
12 . /etc/X11/Xsession
```

Jetzt kalibrieren wir den Touchscreen. Das dazu erforderliche Programm befindet sich im Paket *tslib*.

Code-Ausschnitt 9.3.11

```
1 sudo apt-get install tslib
```



Da das Programm *ts_calibrate* nicht mehr Bestandteil der *tslib* ist, müssen wir *tslib* aus dem Quelltext übersetzen:

Code-Ausschnitt 9.3.12

```
1 apt-get install libtool automake build-essential git
2 git clone git://github.com/kergoth/tslib.git
3 cd tslib
4 ./autogen.sh
5 ./configure
6 make
7 make install
```

Für das Busware-Display editieren Sie bitte die Datei `/etc/ts.conf` wie folgt:

Code-Ausschnitt 9.3.13

```
1 module_raw input
2 module pthres pmin=1
3 module variance delta=30
4 module dejitter delta=100
5 module linear
```

Jetzt müssen wir herausfinden, mit welchem *Event* der Touchscreen verbunden ist.

Code-Ausschnitt 9.3.14

```
1 ls -l /dev/input/by-path/
```

Wende ich den Befehl bei mir an, erhalte ich:

Code-Ausschnitt 9.3.15

```
1 lrwxrwxrwx 1 root root 15 Jul 3 20:04 platform-bcm2708_spi.0-event -> ../event0
```

Bei mir ist der Touchscreen also mit `/dev/input/event0` verknüpft.



Events unter LINUX sind auslösende Ereignisse, die eine bestimmte Reaktion, beispielsweise im Kernel, hervorrufen. Der Druck auf den Touchscreen löst ein solches Event aus.

Alternativ können Sie auch den Befehl

Code-Ausschnitt 9.3.16

```
1 cat /proc/bus/input/devices
```

aufrufen. Das passende *Event* für den Touchscreen wird dann ebenfalls angezeigt:

Code-Ausschnitt 9.3.17

```
1 I: Bus=0000 Vendor=0000 Product=0000 Version=0000
2 N: Name="ADS7846 Touchscreen"
3 P: Phys=spi0.1/input0
4 S: Sysfs=/devices/platform/bcm2708_spi.0/spi_master/spi0/spi0.1/input/input0
5 U: Uniq=
6 H: Handlers=mouse0 event0
7 B: PROP=0
8 B: EV=b
9 B: KEY=400 0 0 0 0 0 0 0 0 0
10 B: ABS=1000003
```

Nachdem wir das passende *Event* herausgefunden haben, rufen wir das Programm zum Kalibrieren des Touchscreens auf.

Code-Ausschnitt 9.3.18

```
1 sudo TSLIB_TSDEVICE=/dev/input/event0 ts_calibrate
```

Das Programm `ts_calibrate` liefert alle Daten für die Kalibrier-Datei `/etc/pointercal`. Bei mir enthält die Datei die folgenden Werte:

Code-Ausschnitt 9.3.19

```
1 5976 -133 -698500 -79 -4407 16963592 65536
```

Damit ist die Kalibrierung des Touchscreens abgeschlossen. Sie können diese mit dem Programm `ts_test` (*Touchscreen Test*) überprüfen. Rufen Sie dieses so auf:

Code-Ausschnitt 9.3.20

```
1 sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_CALIBFILE=/etc/pointercal TSLIB_TSDEVICE=/dev/input/↔
event2 ts_test
```




Sie können auch die oben aufgeführten Umgebungsvariablen exportieren, bevor Sie das Programm `ts_test` starten:

Code-Ausschnitt 9.3.21

```
1 export TSLIB_FBDEVICE=/dev/fb1
2 export TSLIB_CALIBFILE=/etc/pointercal
3 export TSLIB_TSDEVICE=/dev/input/event2
4 sudo ts_test
```

Zum Ende dieses Abschnitts möchte ich Ihnen noch zeigen, wie Sie Ihren Raspberry Pi direkt im LCD booten können und welche besonderen Programme es gibt, die das LCD nutzen. So viel verrate ich vorher: Man kann sogar mit dem LCD fernsehen. Die Einstellung, dass der Raspberry Pi direkt im LCD *booten* soll, nimmt man in der Datei `/boot/cmdline.txt` vor. In dieser Datei werden Parameter für den LINUX-Kernel eingestellt.

Code-Ausschnitt 9.3.22

```
1 dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 ro elevator=↔
   deadline rootwait fbcon=map:10 fbcon=font:ProFont6x11
```

Die letzten beiden `fbcon`-Einträge sorgen zum einen dafür, dass der Kernel mit dem LCD als Bildschirmausgabe bootet, und zum anderen, dass die Schriftgröße (der Font) kleiner wird, damit der Text im LCD dargestellt werden kann.



Falls Sie in der Datei `/boot/cmdline.txt` falsche Einstellungen vornehmen, kann es sein, dass der Kernel nicht mehr bootet. In diesem Fall können Sie die SD-Karte in einem Windows-System einlesen und die Datei wieder reparieren.

Ich möchte Ihnen noch einige Programme zeigen, die mit einem kleinen LCD hervorragend zusammenspielen.

LCD-Browser

Ein minimaler Browser, der mit LCDs harmoniert, ist `links2`. Installieren Sie das Programm wie immer:

Code-Ausschnitt 9.3.23

```
1 sudo apt-get install links2
```

Anschließend muss noch eine passende Auflösung in die Datei `/etc/fb.modes` eingetragen werden:

Code-Ausschnitt 9.3.24

```
1 #TFT Busware
2 mode "320x240"
3   geometry 320 240 320 240 16
4   timings 0 0 0 0 0 0
5   nonstd 1
6   rgba 5/11,6/5,5/0,0/0
7   endmode
```

Der Browser wird dann so aufgerufen:

Code-Ausschnitt 9.3.25

```
1 sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_CALIBFILE=/etc/pointercal TSLIB_TSDEVICE=/dev /input/↵
   event2 link2 -g -mode 320x240x16k www.raspberrypi.org
```



Diesen Browser können Sie hervorragend für eine Hausautomatisierung benutzen, bei der Sie mögliche Steuerungen darstellen und auf Tastendruck Aktionen ausführen. Mit einem solchen Server werden wir uns im nächsten Kapitel ausführlich beschäftigen.

Bildbetrachter

Sicherlich kennen Sie die digitalen Bilderrahmen, die Slideshows von Urlaubsfotos abspielen. Der Raspberry Pi bringt ebenfalls einen Bildbetrachter mit, der im Framebuffer des LCD läuft. Installieren Sie den Bildbetrachter und rufen ihn wie folgt auf:

Code-Ausschnitt 9.3.26

```
1 sudo apt-get install fbi
2 sudo fbi -noverbose -T 1 -a -d /dev/fb1 bild.jpg
```

Die Datei *bild.jpg* ersetzen Sie bitte durch das Bild, das dargestellt werden soll.

Videos im LCD

Auch der Multimedia-Abspieler *mplayer* ermöglicht eine Framebuffer-Videoausgabe im LCD. Er wird so installiert und aufgerufen:

Code-Ausschnitt 9.3.27

```
1 sudo apt-get install mplayer
2 sudo mplayer -vo fbdev2:/dev/fb1 -x 128 -y 160 -zoom dateiname
```

Bildschirm clonen

Eine Framebuffer-Perle ist das Programm *rpi-fbcp*. *fbcp* steht für *framebuffer copy*, und genau das macht das Programm: Es kopiert den Inhalt der Monitorausgabe auf den kleinen LCD. Sie sehen also alles, was Sie auf dem großen Monitor sehen, eins zu eins auf dem TFT Display. Dazu muss das Programm lediglich heruntergeladen und übersetzt werden, bevor es dann im Hintergrund (&) gestartet wird. Für den Übersetzungsvorgang wird das Programm *cmake* benötigt.

Code-Ausschnitt 9.3.28

```
1 sudo apt-get install cmake
2 git clone https://github.com/tasanakorn/rpi-fbcp.git
3 cd rpi-fbcp
4 mkdir build
5 cd build
6 cmake ..
7 make
8 ./fbcp &
```

Beenden Sie das Programm durch einen Aufruf von

Code-Ausschnitt 9.3.29

```
1 sudo pkill fbcp
```

Starten Sie dieses, bevor Sie, wie in Kapitel 5 beschrieben, den *VDR* oder *KODI* aufrufen. Sie erhalten die Bildausgabe dann sowohl auf dem Monitor als auch auf dem LCD. Sie können so das Fernsehprogramm oder Ihr Lieblingsvideo auf dem kleinen TFT-Bildschirm darstellen (Abb. 9.12).



Abbildung 9.12: Mit *fbcp* wird das Fernsehprogramm des *VDR* im LCD angezeigt (Quelle: Busware).

Konsole umschalten

Das Programmpaket *fbset* beinhaltet eine Software, mit der Sie die Konsole auf das LCD und zurückschalten können. Installieren Sie es so:

Code-Ausschnitt 9.3.30

```
1 sudo apt-get install fbset
```

Umschalten können Sie dann mit dem Befehl *con2fbmap*.

Code-Ausschnitt 9.3.31

```
1 sudo con2fbmap 1 1
2 sudo con2fbmap 1 0
```

9.4 Kamera und Anwendungen

Eine tolle Erweiterung für den Raspberry Pi ist die *Raspicam*. Die Daten der knapp 30 € teuren Kamera sind rundum erneuert worden. In der aktuellen Version 2 ist ein 8 Megapixel-Sensor der Firma Sony verbaut worden (IMX219). Die Linse ist auf eine feste Fokussierung eingestellt. Die Kamera liefert Bilder in einer Auflösung von bis zu 3280 × 2464 Pixeln und unterstützt Videoaufnahmen in den Formaten 1080p30, 720p60 und 640x480p90. Die höchste Auflösung entspricht also Full-HD, allerdings nur mit 30 Bildern pro Sekunde in einer progressiv-Darstellung.

Die Kamera ist auch als Infrarot-Kamera verfügbar. Sie trägt dann den französischen Namen *Noir*, was *schwarz*, *Finsternis* bedeutet.



(a) Raspicam am Raspberry Pi (Quelle: Adafruit).

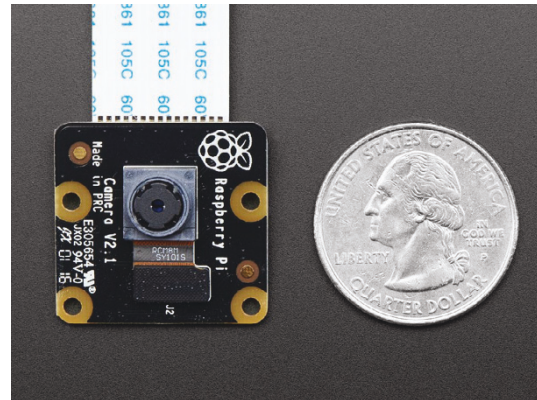
(b) Raspicam *Noir* (Quelle: Adafruit).

Abbildung 9.13: Für den Raspberry Pi gibt es eine eigene Kamera, die einfach am CSI-Adapter angeschlossen wird.

Info Die Kamerakabel für den Raspberry Pi 3 und für den Raspberry Pi Zero sind unterschiedlich. Bitte beachten Sie das bei Ihrer Bestellung.

Die Kamera-Erweiterung lädt zum Spielen ein. Zeitraffer-Videos, Raumüberwachung, Alarmanlage; der Fantasie sind auch hier keine Grenzen gesetzt. Ich zeige Ihnen in diesem Unterkapitel, wie Sie die Kamera installieren und in Betrieb nehmen können und stelle Ihnen die wichtigsten Kamera-Softwarepakete vor.

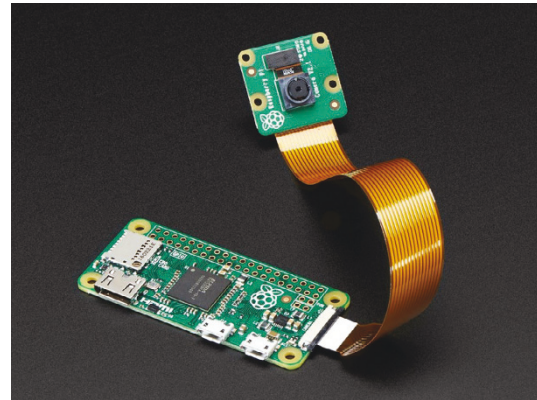


Abbildung 9.14: Pi 3 und Pi Zero nutzen unterschiedliche Kabel, um die Kamera zu befestigen (Quelle: Adafruit).

9.4.1 Installation

Bevor wir mit der Installation beginnen, möchte ich Ihnen einen Sicherheitshinweis an die Hand geben.

Info Bitte installieren Sie Erweiterungen wie die Raspberry Pi Kamera nur, wenn der Raspberry Pi vom Stromnetz getrennt ist. Die Gefahr, ihn andernfalls zu beschädigen (z. B. durch einen Kurzschluss), ist sehr hoch.

Die mechanische Befestigung des Kamerakabels gestaltet sich einfach: Ziehen Sie die Abdeckung der CSI-Buchse (Abb. 9.13) nach oben und setzen Sie das 15-polige Flachbandkabel mit der blauen Markierung in Richtung Ethernet-Anschluss ein. Danach drücken Sie den Verschluss wieder nach unten und klemmen das Kabel damit fest.

Info Vergessen Sie nicht, nach der mechanischen Installation die Schutzfolie von der Linse der Kamera zu entfernen.

Info Vermeiden Sie es, das Kabel zu oft hin und her zu biegen, da dies einen Kabelbruch mit sich bringen kann. Sollte das trotz aller Vorsicht geschehen, können Sie im Internet ein Ersatzkabel bestellen.

Wenn Sie die Kamera noch nicht aktiviert haben, holen Sie das jetzt bitte nach: Booten Sie Ihren Raspberry Pi. Rufen Sie danach das Konfigurationstool mit `sudo raspi-config` auf und aktivieren Sie im Punkt 5 des Auswahlmenüs die Kamera. Nach erfolgter Aktivierung muss der Raspberry Pi wieder neu gestartet werden (`sudo reboot`). Ab jetzt kann die Kamera benutzt werden. Auf das Kamera-Board ist eine rote LED gelötet. Falls diese Sie stört, kann sie durch einen Eintrag in der Datei `/boot/config.txt` dauerhaft deaktiviert werden.

Code-Ausschnitt 9.4.1

```
1 disable_camera_led=1
```

So können Sie rote Reflexionen an Glasscheiben vermeiden oder die Kamera betreiben, ohne dass der Betrieb sofort offensichtlich ist. Damit Änderungen an der Datei `/boot/config.txt` wirksam sind, muss der Raspberry Pi neu gestartet werden. Schauen wir uns an, was die kleine Kamera leistet.

9.4.2 Betrieb

Die Raspbian-Distribution hat bereits zwei Terminal-Programme an Bord, mit denen die Kamera in Betrieb genommen werden kann: `raspistill` und `raspivid`. Das erste Programm dient dazu, einzelne Bilder aufzunehmen. Mit dem zweiten steht einer Karriere als Videofilmer nichts mehr im Wege. `raspivid` nimmt dabei ausschließlich im H264-Format auf. Das Format kann aber ohne Probleme in ein anderes konvertiert werden.

Bilder

Um ein einzelnes Bild mit der Kamera aufzunehmen, geben Sie bitte den Befehl

Code-Ausschnitt 9.4.2

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

ein. Das Bild wird dann im aktuellen Verzeichnis unter dem Namen `bild.jpg` gespeichert. Der Parameter `-o` steht für *output*, also *Ausgabe*. Auf der Seite <http://www.raspberrypi.org> gibt es beeindruckende Zeitraffer-Aufnahmen. Bei diesem Verfahren wird in bestimmten Zeitabständen ein Foto erstellt. Nachdem alle Fotos gemacht worden sind, werden diese nacheinander in einem Video zusammengefasst. Der Befehl für eine Zeitrafferaufnahme lautet:

Code-Ausschnitt 9.4.3

```
1 raspistill -o bild_%04d.jpg -t1 5000 -t 1800000
```

Die Fotosoftware schießt so eine halbe Stunde lang (1.800.000 Millisekunden) alle fünf Sekunden (5000 ms) ein Bild. Die Bilder werden wieder in eine Datei namens `bild_0000.jpg` gespeichert, wobei mit dem Kürzel `%04d` eine fortlaufende vierstellige Zahlenreihe generiert wird. Die Bilder werden also hochgezählt. Anschließend können die einzelnen Bilder mit dem Audio- und Videokonvertierer `ffmpeg` in einen Film umgewandelt werden. Falls `ffmpeg` noch nicht installiert sein sollte, holen Sie das bitte nach.

Code-Ausschnitt 9.4.4

```
1 sudo apt-get install ffmpeg
```


Erzeugen Sie den Film so:

Code-Ausschnitt 9.4.5

```
1 ffmpeg -qscale 5 -r 4 -b 9600 -i bild_%04d.jpg video.mp4
```

Videos

Videos können mit dem Programm *raspivid* „gedreht“ werden. Der Aufruf von

Code-Ausschnitt 9.4.6

```
1 raspivid -o video.h264
```

liefert einen fünf-Sekunden-Schnipsel in voller 1920x1080 Pixel-Auflösung. Möchte man längere Videos aufnehmen, hilft der Parameter *-t*. Hinter ihm kann man wieder eine Zeit in Millisekunden angeben. Mit den Parametern *w* und *h* können *Weite* und *Höhe* des Videos eingestellt werden. Der Aufruf

Code-Ausschnitt 9.4.7

```
1 raspivid -o video.h264 -w 640 -h 480 -t 60000
```

erzeugt also ein einminütiges Video in der VGA-Auflösung 640x480 Pixel. Das Programmpaket *gpac* installiert die Software *MP4Box*, mit deren Hilfe man das H264-Video in ein MPEG4-Video umwandeln kann. Installieren Sie es und rufen Sie es so auf:

Code-Ausschnitt 9.4.8

```
1 sudo apt-get install gpac
2 MP4Box -fps 30 -add video.h264 video.mp4
```

Motion detection

Motion detection bedeutet *Bewegungserkennung*. Sie kann für Überwachungszwecke genutzt werden, wenn Bilder oder Videos nur aufgenommen werden sollen, falls „Bewegung“ im Bild ist. So könnte man beispielsweise einen Einbrecher auf frischer Tat filmen. Unter Raspbian erledigt das Programmpaket *motion* die Erkennung.

Motion erwartet einen funktionierenden Video4Linux-Treiber, welcher die Kamera unter (*/dev/videoX*) zur Verfügung stellt. Hierzu müssen die Kernel-Module *v4l2_common* und *bcm2835-v4l2* geladen werden:

Code-Ausschnitt 9.4.9

```
1 sudo modprobe v4l2_common
2 sudo modprobe bcm2835-v4l2
```

Damit die beiden Module auch nach einem Neustart noch aktiv sind, tragen Sie sie bitte in die Datei */etc/modules* ein. Danach steht das *video*-Device zur Verfügung. Die Installation ist denkbar einfach:

Code-Ausschnitt 9.4.10

```
1 sudo apt-get install motion
```

Wenn Sie möchten, dass *motion* nach dem Hochfahren des Raspberry Pi automatisch startet, ersetzen Sie bitte in der Datei */etc/default/motion* das „no“ durch ein „yes“.

Motion benötigt einen Ordner, in dem es Bilder speichern kann, sowie eine Konfigurationsdatei, die Sie unter */etc/motion* finden.

Wir erstellen einen Ordner für *motion* und versehen ihn mit den nötigen Schreibrechten.

Code-Ausschnitt 9.4.11

```
1 mkdir ~/motion
2 sudo chgrp motion /home/pi/motion
3 chmod g+rwX /home/pi/motion
```

Editieren Sie bitte als *root* die Steuerdatei */etc/motion/motion.conf* und ändern Sie die folgenden Einstellungen:

Code-Ausschnitt 9.4.12

```
1 daemon on
2 target_dir target_dir /home/pi/motion
3 stream_localhost off
```

Motion bietet viele weitere Parameter, die Sie nach eigenem Geschmack einstellen und ändern können:

- **width 640**: Hier können Sie die Breite des Videos in Pixeln festlegen.
- **height 480**: Hier können Sie die Höhe des Videos in Pixeln festlegen.
- **framerate 10**: Dieser Eintrag legt die empfangenen Bilder pro Sekunde fest.
- **threshold 5000**: Dieser Eintrag legt die Empfindlichkeit der Bewegungserkennung in Anzahl von Pixeln fest.
- **ffmpeg_output_movies off**: Bei erkannter Bewegung werden keine Videos gespeichert.
- **output_pictures off**: Bei erkannter Bewegung werden keine Bilder gespeichert.
- **stream_port 8081**: Dieser Eintrag legt den Port fest, über den der Livestream in einem Server erreichbar ist.
- **stream_quality 75**: Dieser Eintrag legt die Bildqualität des Livestreams in Prozent fest.
- **stream_motion on**: Ohne Erkennung einer Bewegung wird im Livestream nur 1 Bild pro Sekunde gezeigt.
- **stream_maxrate 10**: Dieser Eintrag gibt die maximale Anzahl von Bildern pro Sekunde im Livestream an.

Falls Sie das Bild der Kamera im Internet zur Verfügung stellen wollen, sollten Sie diesen unbedingt mit einem Kennwort schützen, damit nicht jeder sehen kann, was gerade in Ihrer Wohnung vor sich geht. Benutzernamen und Kennwort können ebenfalls in der Datei */etc/motion/motion.conf* festgelegt werden:

- **stream_auth_method 1**: Dieser Eintrag aktiviert den Kennwortschutz.
- **stream_authentication benutzername:kennwort**: Dieser Eintrag legt einen Benutzernamen und ein Kennwort fest. Bitte ersetzen Sie *benutzername* mit einem Benutzernamen und *kennwort* mit einem sicheren Kennwort.

Motion selbst ist ein Dienst und kann nun so gestartet werden:

Code-Ausschnitt 9.4.13

```
1 sudo service motion start
```

Nach der Installation und Einrichtung von *motion* kann der Livestream der Kamera getestet werden. *Motion* stellt diesen am Port 8081 bereit.

Rufen Sie also lokal in einem Browser Ihrer Wahl die URL `192.168.178.66:8081` auf. Ersetzen Sie meine IP-Adresse dabei bitte durch die Ihres Raspberry Pi. Sollten Sie den Stream von außerhalb anschauen wollen, können Sie sich per *OpenVPN* (Kapitel 8.4) zuerst in ihr Heimnetz einwählen, bevor Sie dann die oben gezeigte Adresse in Ihren Browser eingeben. Sollten Sie, wie weiter oben beschrieben, *PowerDNS* installiert haben, können Sie statt der IP-Adresse auch den Namen, den Sie für den Raspberry Pi vergeben haben, im Browser-Aufruf verwenden.

Damit die Speicherkarte des Raspberry Pi nicht mit Bildern voll läuft, können Sie alle Bilder, die älter als 10 Tage sind z. B. durch einen *cronjob* so

Code-Ausschnitt 9.4.14

```
1 find /home/pi/motion -type f -name '*.jpg' -mtime +10 -delete
```

automatisch löschen lassen.

9.4.3 Python

Es gibt ein eigenes Kamera-Modul für Python, welches man mit

Code-Ausschnitt 9.4.15

```
1 sudo apt-get install python-picamera
```

installieren kann. Ein Python-Beispiel, wie es auf `www.raspberrypi.org` zu finden ist, sehen Sie im unteren Quelltext.

Code-Ausschnitt 9.4.16

```
1 import picamera
2 from time import sleep

4 camera = picamera.PiCamera()
5 camera.capture('image.jpg')

7 camera.start_preview()
8 camera.vflip = True
9 camera.hflip = True
10 camera.brightness = 60

12 camera.start_recording('video.h264')
13 sleep(5)
14 camera.stop_recording()
```

Zunächst wird das zuvor installierte Python-Modul *picamera* eingebunden. Damit erstellt das Programm ein Bild mit dem Namen *image.jpg*, bevor dann die vertikale und horizontale Ansicht des Bildes gedreht wird (*flip*), um ein fünf-Sekunden-H.264-Video aufzunehmen (*video.h264*). Helligkeits-Level der Kamera können mit der Python-Bibliothek *picamera* ebenfalls sehr einfach variiert werden. Das letzte Beispiel dieses Kapitels zeigt dies in einer Python-Schleife von 1 bis 100:

Code-Ausschnitt 9.4.17

```
1 for i in range(100):
2     camera.brightness = i
3     sleep(0.1)
```

Die Python-Bibliothek bietet weitere Einstellmöglichkeiten, von der Variation des Kontrastes bis hin zur Bild-Manipulation.

Eine komplette Dokumentation erhalten Sie unter <http://picamera.readthedocs.org/en/release-1.2/>.

Zusammenfassung 9 Sie haben das Kapitel zum Thema „Erweiterungen“ gelesen. Ich habe Ihnen gezeigt, wie man einen Webserver benutzen kann, um andere Geräte zu steuern, die über einen seriellen Bus mit dem Raspberry Pi verbunden sind. Die GPIO-Pins des Raspberry Pi wurden dabei mit Hilfe verschiedener Computersprachen (Python, C und Terminal-Befehle) angesprochen und ausgelesen. Wir haben ein TFT-Display in Betrieb genommen und die dafür erforderlichen Parameter beim LINUX-Start übergeben. Viele kleine Programme helfen mit, verschiedenen Inhalt auf einem kleinen Touchscreen darzustellen. Das funktioniert sogar mit dem Fernsehprogramm.

Ich habe Ihnen gezeigt, wie man die Raspberry Pi-Kamera-Erweiterung anschließt, um damit Fotos zu machen oder Videos zu drehen. Diese können über einen eigenen Webserver als Stream zur Verfügung gestellt werden. Der Webserver selbst kann seinen Namen von einem eigenen DNS erhalten.

Das nächste Kapitel beschäftigt sich mit dem Thema Hausautomatisierung, Alarmanlagen, Sprachsynthese und VOIP-Telefonie. ■

10 — Hausautomatisierung

Das Thema Hausautomatisierung hat in den letzten Jahren immer mehr an Bedeutung gewonnen. Vor vielen Jahren gab es bereits Steckdosen, die man mit einer Fernbedienung an- und ausschalten konnte. Heute werden Garagentore über das Handy geöffnet, Rollläden heruntergelassen oder das Licht per Tablet-PC an- oder ausgeschaltet. Eine Hausautomatisierung spart Strom und Heizenergie und erlaubt, abhängig von äußeren Faktoren wie Klima, individuelle Aktionen oder Zutrittskontrollen. Herzstück einer Hausautomatisierung ist ein *Transceiver*. *Transceiver* ist ein Kunstwort und besteht aus *Transmitter* und *Receiver*, was soviel wie Sender und Empfänger bedeutet. Bei einem *Transceiver* handelt es sich also um ein Gerät, das sowohl senden als auch empfangen kann. Hausautomatisierungs-*Transceiver* senden und empfangen auf freigegebenen *ISM*-Band-Frequenzen (*Industrial, Scientific and Medical band*). Die bekanntesten Frequenzbänder sind 433 MHz und 868 MHz. Wir schauen uns in diesem Kapitel einen 868 MHz *Transceiver* an, den wir bereits im vorigen Kapitel zusammen mit dem Busware-CCD installiert haben. Ich gebe aber auch einen Ausblick auf preiswertere Möglichkeiten. Zunächst werden wir den *Transceiver* in Betrieb nehmen. Anschließend werden wir einen Hausautomatisierungs-Server installieren. Wir werden eine Funksteckdose, die wir mit Hilfe eines Mobiltelefons an- und ausschalten können, mit diesem Server verbinden. Ich werde Ihnen dabei weitere Sensoren und Aktoren vorstellen, die Sie mit dem Hausautomatisierungs-Server verbinden können. Im Verlauf dieses Kapitels schauen wir uns an, wie wir einen Alarm - etwa von einem Wassersensor - per Anruf an unser Telefon weiterleiten können. Im Falle eines Wasserschadens kann so der Nachbar informiert werden und notfalls die Wasserversorgung unseres Hauses abstellen.



Abbildung 10.1: Das Telekom-Haus in Berlin (2005-2006) war ein Paradebeispiel für Hausautomatisierung (Quelle: http://commons.wikimedia.org/wiki/File:T-Corn-Haus_Berlin_%282005%29.jpg).



Der Raspberry Pi bietet das alles zu einem unschlagbar günstigen Preis. Während einige Hausautomatisierungs-Zentralen bis zu 800 € kosten, liegen die Kosten für einen Raspberry Pi samt Zubehör deutlich darunter.

10.1 Busware 868 MHz Transceiver

Wir haben uns das Busware CCD-Modul bereits im vorigen Kapitel bezüglich des TFT LCD angeschaut. In dieser Raspberry Pi-Zusatzplatine steckt aber noch mehr:

- Sie besitzt eine *Real Time Clock (RTC)* mit der Bezeichnung DS1338. Der Chip wird von Maxim hergestellt und ist über den I2C-Bus mit dem Raspberry Pi verbunden. Die *RTC* wird von einem Kondensator gespeist, der seine Ladung extrem lange halten kann.
- Sie führt den I2C-Bus der Raspberry Pi GPIO-Pins auf eine Buchse.
- Sie ist mit einem Texas Instruments CC1101 *Transceiver* (<http://www.ti.com/lit/ds/symlink/cc1101.pdf>) und mit einem Atmel ATmega M1284P Co-Prozessor (<http://www.atmel.com/images/doc8059.pdf>) bestückt, der eine *CULW/Contiki* Firmware enthält. Ein *ISP-Header* ist optional. *CULW* ist eine Firmware, die unter der *GNU Public License, GPL* steht. *CUL* ist ein Gerät, welches mit einem PC (oder einem Embedded System) verbunden wird und Hochfrequenz-Signale auf einer Frequenz von 868 MHz senden und empfangen kann. Der Name *CUL* ist die Abkürzung von *CC1101 USB Lite*. Die Firmware, welche das Hochfrequenzprotokoll auf einem *CUL*-Gerät implementiert, nennt man *CULW (CC1101 USB Lite FirmWare)*. Sie erlaubt, dass das Gerät unter dem Hausautomatisierungs-Server *fhem* verwendet werden kann. *FHEM* bedeutet *Freundliche Hausautomatisierung und Energie-Messung*. *ISP* ist eine Schnittstelle, die das Programmieren einer logischen Schaltung direkt in einer Einsatzanwendung ermöglicht. Dazu wird meist eine serielle Verbindung wie *JTAG* oder *SPI* verwendet. Der Vorteil einer Programmierung im laufenden System ist, dass der zu programmierende Schaltkreis nicht mehr aus dem Zielsystem entfernt werden muss. Er wird also nicht mechanisch belastet und der gesamte Programmiervorgang kann sehr schnell durchgeführt werden.
- Die Platine besitzt einen I2C-Anschluss ähnlich eines FTDI UMFT200XD-Moduls. Dieses Modul stellt einen I2C-Bus per USB zur Verfügung.

10.1.1 Real Time Clock

Binden wir zuerst die *Real Time Clock* des Busware-Moduls ein:

Code-Ausschnitt 10.1.1

```
1 echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
2 sleep 1
3 hwclock -s
```

Anschließend entfernen wir die fake-RTC:

Code-Ausschnitt 10.1.2

```
1 apt-get remove fake-hwclock
2 rm /etc/cron.hourly/fake-hwclock
3 update-rc.d -f fake-hwclock remove
4 rm /etc/init.d/fake-hwclock
5 update-rc.d hwclock.sh enable
```

In der Datei */ect/modules* muss das Modul *i2c-bcm2708* geladen werden.

Code-Ausschnitt 10.1.3

```
1 i2c-bcm2708
```

Wenn Sie die Änderungen permanent machen wollen, initialisieren Sie die RTC in der Datei `/etc/rc.local`:

Code-Ausschnitt 10.1.4

```
1 echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
2 sleep 1
3 hwclock -s
```

Nach diesen Einstellungen liest der Raspberry Pi nach dem Booten die aktuelle Zeit aus der *Real Time Clock* und stellt die interne Raspberry Pi Uhr entsprechend.



Eine korrekte Uhrzeit und ein korrektes Datum sind für den Betrieb eines Hausautomatisierungs-Servers unerlässlich. Sollten Sie kein *RTC*-Modul für den Raspberry Pi besitzen, denken Sie bitte daran, die korrekte Zeit per *ntp* zu synchronisieren (Kapitel 2.6).

Der Raspberry Pi ist so konfiguriert, dass er während des Boot-Vorganges Konsolen-Ein- und -Ausgaben zum seriellen Port sendet. Das ist zwar sehr nützlich, wenn man sich über den seriellen Port in den Raspberry Pi einwählt, hindert aber eigene Programme, den seriellen Port überhaupt zu benutzen. Um den Port für das Busware-Interface (oder auch für ein Arduino-Interface) freizugeben, muss der *Login* für diesen Port deaktiviert werden. Sie werden ohnehin meist einen *ssh*-Zugang für die Einwahl benutzen. Kommentieren Sie bitte in der Datei `/etc/inittab` die folgende Zeile aus:

Code-Ausschnitt 10.1.5

```
1 #T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Dasselbe gilt für die Kernel-Parameter-Datei `/boot/cmdline.txt`. Aus dieser Datei müssen alle Referenzen auf `ttyAMA0` entfernt werden. Bei mir sieht diese Datei so aus:

Code-Ausschnitt 10.1.6

```
1 dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 ro elevator=↵
  deadline rootwait
```

Nach den Änderungen ist ein Neustart des Raspberry Pi erforderlich. Das Busware-Interface hält das Transceiver-Interface per Default im Reset-Zustand. Der GPIO-Pin 17 wird dazu benutzt, das Sende- und Empfangs-Interface zu initialisieren. Das können Sie wie folgt erledigen:

Code-Ausschnitt 10.1.7

```
1 sudo su
2 if test ! -d /sys/class/gpio/gpio17; then echo 17 > /sys/class/gpio/export; fi
3 echo out > /sys/class/gpio/gpio17/direction
4 echo 1 > /sys/class/gpio/gpio17/value
```

Diese Befehle kennen Sie bereits aus dem letzten Kapitel. In Zeile 1 prüfen wir, ob das Interface überhaupt vorhanden ist und „exportieren“ den Pin 17. Die Richtung steht in Zeile 2, nämlich *out*. Der Wert selbst (1) wird dann in Zeile 3 gesetzt. Nach dem Absetzen dieser Befehle blinkt die orangefarbene LED im Busware-Interface (CCD-Chip).



Busware liefert das Interface mit einer installierten und getesteten *CULW*-Firmware. Der GPIO-Pin 22 wird benutzt, um den AVR109-Bootloader aufzurufen. Der Abschnitt 10.2.4 gibt Auskunft darüber, wie die Firmware aktuell gehalten werden kann.

10.2 FHEM

Die *Freundliche Hausautomatisierung und Energiemessung (FHEM)* ist ein Server, der von Rudolf Koenig in der Programmiersprache *Perl* geschrieben wurde. Mit *FHEM* können Komponenten der Hausautomatisierung, die über Funk oder Kabel angebunden sind, gesteuert werden. *FHEM* bietet schöne Web-Oberflächen, mit denen Hausautomatisierungs-Geräte angesprochen werden können oder mit denen Eingaben von Hausautomatisierungs-Komponenten, etwa die Temperatur eines Sensors, grafisch angezeigt werden können. Wir installieren zunächst den *FHEM*-Server, bevor wir ihn absichern und dann beispielhaft ein Gerät anlernen. Da der Server in *Perl* programmiert ist, müssen zunächst einige *Perl*-Pakete installiert werden, bevor es dann an die Installation von *FHEM* selbst geht.



Abbildung 10.2:
(Quelle: Rudolf König).

Code-Ausschnitt 10.2.1

```
1 sudo apt-get -f install && sudo apt-get install perl libdevice-serialport-perl && sudo ↵
   apt-get install libio-socket-ssl-perl && sudo apt-get install libwww-perl
```

FHEM steht als Debian-Package (*deb*-Datei) auf der *FHEM*-Webseite zum Download bereit. Laden Sie es herunter und rufen Sie zum Installieren *dpkg -i* auf.

Code-Ausschnitt 10.2.2

```
1 sudo wget http://fhem.de/fhem-5.8.deb
2 sudo dpkg -i fhem-5.8.deb
```

Als ich dieses Buch geschrieben haben, war die *FHEM*-Version 5.8 aktuell. Schauen Sie bitte auf der Webseite nach der jeweils aktuellen Version und passen Sie die Versionsnummer beim Herunterladen und Installieren entsprechend an. Im nächsten Schritt wechseln wir in das Installationsverzeichnis (*FHEM* liegt im Verzeichnis */opt*) und passen die Benutzer-Rechte wie folgt an:

Code-Ausschnitt 10.2.3

```
1 cd /opt
2 sudo chmod -R a+w fhem && sudo usermod -a -G tty pi && sudo usermod -a -G tty fhem
```



Der Befehl *usermod* wird unter LINUX dazu verwendet, einen bestehenden Benutzer zu bearbeiten. Der Befehl *usermod --help* zeigt alle Möglichkeiten dazu auf.

FHEM wird als Dienst gestartet. Die Dienste-Datei heißt */etc/init.d/fhem*. Nach der Installation sollte *FHEM* bereits laufen und kann über den Port 8083 erreicht werden. Für einen ersten Test können Sie also in Ihrem Browser die folgende Adresse aufrufen:

Code-Ausschnitt 10.2.4

```
1 http://ip-adresse:8083/fhem
```

Bitte ersetzen Sie dabei *ip-adresse* durch die IP-Adresse Ihres Raspberry Pi.

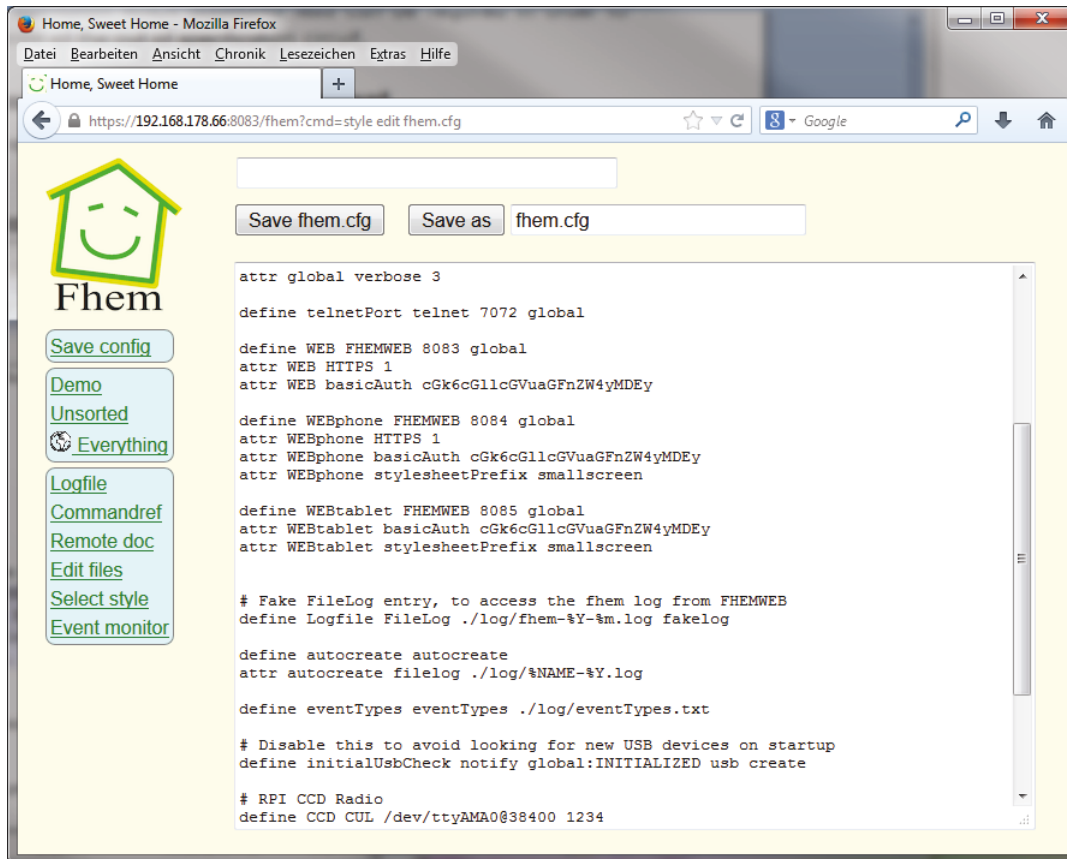


Abbildung 10.3: Die *FHEM*-Konfigurationsdatei kann komfortabel über ein Web-Interface editiert werden.

Nachdem die oben genannten Änderungen durchgeführt worden sind, muss die Konfigurationsdatei gespeichert und der Hausautomatisierungs-Server neu gestartet werden. Klicken Sie den *Save*-Knopf und geben Sie dann in die *FHEM*-Befehlseingabezeile der Web-Oberfläche den nächsten Befehl ein:

Code-Ausschnitt 10.2.9

```
1 shutdown restart
```

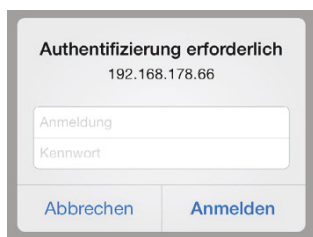


Abbildung 10.4: Authentifizierung per Telefon.

Dadurch wird der *FHEM*-Server neu gestartet und übernimmt dabei die eben durchgeführten Änderungen. Wenn Sie ihn nun in einem Browser aufrufen, verlangt der die korrekte Eingabe des von Ihnen kodierte Benutzernamens und des richtigen Kennwortes (Abb. 10.4), bevor die Web-Oberfläche angezeigt wird. Der Text, der über der Eingabeaufforderung für Benutzernamen und Kennwort steht, kann durch den Eintrag *attr WEB basicAuthMsg Text* angepasst werden.

https

Bisher werden der Benutzername und das dazugehörige Kennwort im Klartext (*Base64*-kodierte) übertragen. Damit niemand diese Kennung mitlesen kann, richten wir in diesem Abschnitt eine über *https*-gesicherte Verbindung zum *FHEM*-Server ein. Erstellen Sie zu diesem Zweck das Zertifikatsverzeichnis *certs* unterhalb von */opt/fhem*.

Zur Generierung von Zertifikatsdateien verwenden wir *openssl*. Installieren Sie dieses Programm ebenfalls, falls Sie das nicht schon getan haben.

Code-Ausschnitt 10.2.10

```
1 sudo mkdir /opt/fhem/certs
2 sudo apt-get install openssl
```

Die Server-Key- und Server-Zertifikats-Datei werden mit folgendem Aufruf erstellt:

Code-Ausschnitt 10.2.11

```
1 cd /opt/fhem/certs
2 sudo openssl req -new -x509 -nodes -out server-cert.pem -days 3.650 -keyout server-key.↵
   pem
```

Im oberen Beispiel sind die Zertifikate 3650 Tage lang gültig. Sie können natürlich eine kürzere Gültigkeitsdauer vergeben. Fügen Sie, wie oben beschrieben, in die Datei *fhem.cfg* die *HTTPS*-Attribute in den entsprechenden Sektionen hinzu:

Code-Ausschnitt 10.2.12

```
1 attr WEB HTTPS 1
2 attr WEBphone HTTPS 1
3 attr WEBtablet HTTPS 1
```

Nach dem Speichern der Datei und einem Neustart des *FHEM*-Servers mittels *shutdown restart* serviert *FHEM* die Webseite gesichert. Der Aufruf erfolgt dann so:

Code-Ausschnitt 10.2.13

```
1 https://ip-adresse:8083/fhem
```

10.2.2 Hinzufügen von Geräten

Bei Hausautomatisierungs-Geräten unterscheidet man prinzipiell die *Transceiver*-Frequenzen 433 MHz und 868 MHz. Dabei sind folgende Systeme weit verbreitet:

- *Homematic*: Bei diesem System handelt es sich um ein 868 MHz-System, das für bidirektionale Verbindungen ausgelegt ist. Der Empfang wird quittiert. Erfolgt die Quittierung nicht, sendet der Sender erneut. Damit ist dieses System sehr zuverlässig, aber auch teuer. Die Übertragung erfolgt verschlüsselt. Das System wird von der ELV-/eQ-3-Gruppe vertrieben.
- *FS20*: Bei diesem System handelt es sich ebenfalls um ein 868 MHz-System, welches nur senden oder nur empfangen kann. Im Gegensatz zum *Homematic*-System weiß der Sender nicht, ob das Signal beim Empfänger angekommen ist. *FS-20*-Systeme sind ebenfalls verschlüsselt und werden von ELV und Conrad Elektronik vertrieben.
- *Max* ist ebenfalls ein verschlüsselt 868 MHz-System der eQ-3-Gruppe, welches hauptsächlich zur Steuerung einer Heizungsanlage verwendet wird.
- *Intertechno* bietet ein Hausautomatisierungs-System auf der Basis von 433 MHz-Funk an.

Eine Mischung verschiedener Systeme ist möglich. Allerdings wird meistens ein *CUL* pro System benötigt. Da *FS-20*-Komponenten relativ preiswert sind, habe ich mir für einen ersten Hausautomatisierungs-Versuch mit dem Raspberry Pi eine schaltbare *FS-20*-Steckdose besorgt. (Abb. 10.5). Die Schaltleistung der Steckdose beträgt 16 A bei 230 V Netzspannung (3,68 kW). Die Steckdose kann am Gerät selbst oder mit Hilfe eines *FS-20*-Senders eingeschaltet werden.



Abbildung 10.5: Die ELV FS-20-Steckdose bietet einen preiswerten Einstieg in die Hausautomatisierungstechnik für unter 30 € (Quelle: ELV AG).

Sie besitzt einen Abschalt-Timer, der die Steckdose nach einer vorgegebenen Zeit wieder automatisch ausschalten kann.



Es existieren verschiedene Revisionen von FS-20-Komponenten, beispielsweise ST-2 oder ST-3. In ihrer Funktion sind die Geräte identisch und können ohne Probleme gemischt betrieben werden.

Die Firmware des Busware CC1101-Systems beherrscht das FS-20-Protokoll im Auslieferungszustand. Zunächst teilen wir FHEM mit, welches Device unser Transceiver ist. Tragen Sie dazu über das FHEM Web-Interface, wie oben beschrieben, folgende Zeilen in die Datei *fhem.cfg* ein:

Code-Ausschnitt 10.2.14

```
1 # RPI CCD Radio
2 define CCD CUL /dev/ttyAMA0@38400 1234
```

Die Zahlenfolge 1234 können Sie dabei beliebig wählen (maximal 8 Ziffern mit den Zahlen von 1 bis 4). Sie bilden die Identifikationsnummer des gesamten Systems, den sogenannten „Hauscode“. Nachdem das CUL eingetragen worden ist, muss FHEM neu gestartet werden, damit das CUL registriert wird. FHEM zeigt das CUL dann unter der Rubrik „Everything“ als *opened* an. Nachdem das CUL korrekt angezeigt wird, teilen Sie FHEM in der Eingabe-Befehlszeile mit, dass Sie eine FS-20-Komponente, nämlich eine Funksteckdose, steuern möchten.

Code-Ausschnitt 10.2.15

```
1 define steckdose1 FS20 1234 56
2 attr steckdose1 model fs20st
```

Wir nennen die Steckdose *steckdose1* und teilen ihr mit, dass sie auf den Hauscode (1234) reagieren soll. Gleichzeitig erhält sie noch eine Gerätenummer, in unserem Fall die 56. Das Anlernen der Steckdose ist einfach: Versetzen Sie diese durch 5-sekündiges Drücken der Ein-/Aus-Taste in den Lernmodus. Setzen Sie dann in der FHEM Kommando-Eingabe das Kommando zum Einschalten der Steckdose ab.

Code-Ausschnitt 10.2.16

```
1 set steckdose1 on
```

Die Funksteckdose koppelt sich nun mit dem *CUL* und signalisiert diesen Vorgang durch das Ausschalten der LED. Sollte der Vorgang nicht beim ersten Mal funktionieren, wiederholen Sie ihn einfach. Weisen wir unserer Steckdose noch einen Raum zu, in dem sie sich befindet. Ich nenne ihn *Demo*. Als Icon weise ich der Steckdose das *FHEM*-Icon mit dem Namen *black_Steckdose.off* zu.

Code-Ausschnitt 10.2.17

```
1 attr steckdose1 room Demo
2 attr steckdose1 icon black_Steckdose.off
```

Das Icon können Sie später durch einen Klick auf dasselbe ändern. Bei mir sieht der Raum *Demo* so aus wie in Abb. 10.6. Die Steckdose kann nun entweder durch Absetzen der Befehle

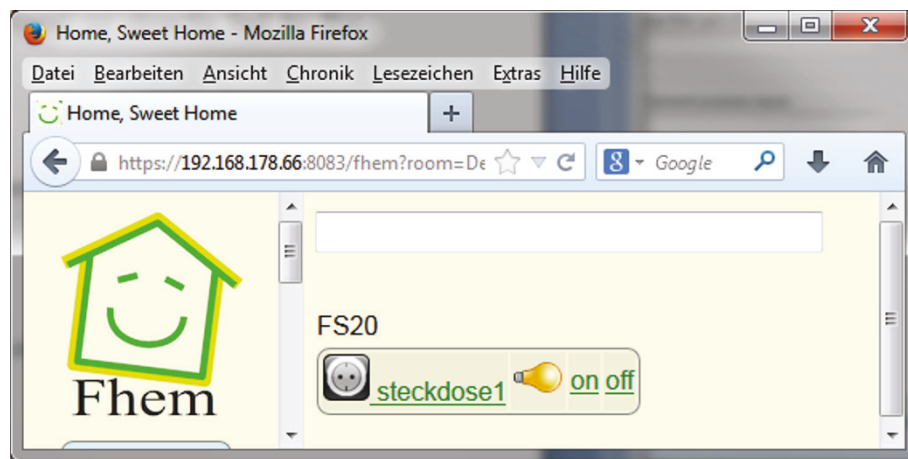


Abbildung 10.6: Die Kurzübersicht des *Demo*-Raumes erlaubt das Ein- und Ausschalten der Steckdose.

set steckdose1 on und *set steckdose1 off* an- und ausgeschaltet werden oder durch Klicks auf *on* und *off*. Eine weitere Alternative ist ein Klick auf die Glühbirne selbst, die im Aus-Zustand der Steckdose durch ein Icon einer Glühbirne, die nicht leuchtet, ersetzt wird. Durch einen Klick auf das Steckdosen-Icon werden alle Einstellungen für das *FS-20*-Gerät sichtbar (Abb. 10.7). Falls Sie das Blinken des Busware-*CUL* stört, können Sie es in der *FHEM*-Konfigurationsdatei ausschalten.

Code-Ausschnitt 10.2.18

```
1 #Blinken LED ausschalten
2 set CCD raw 100
```

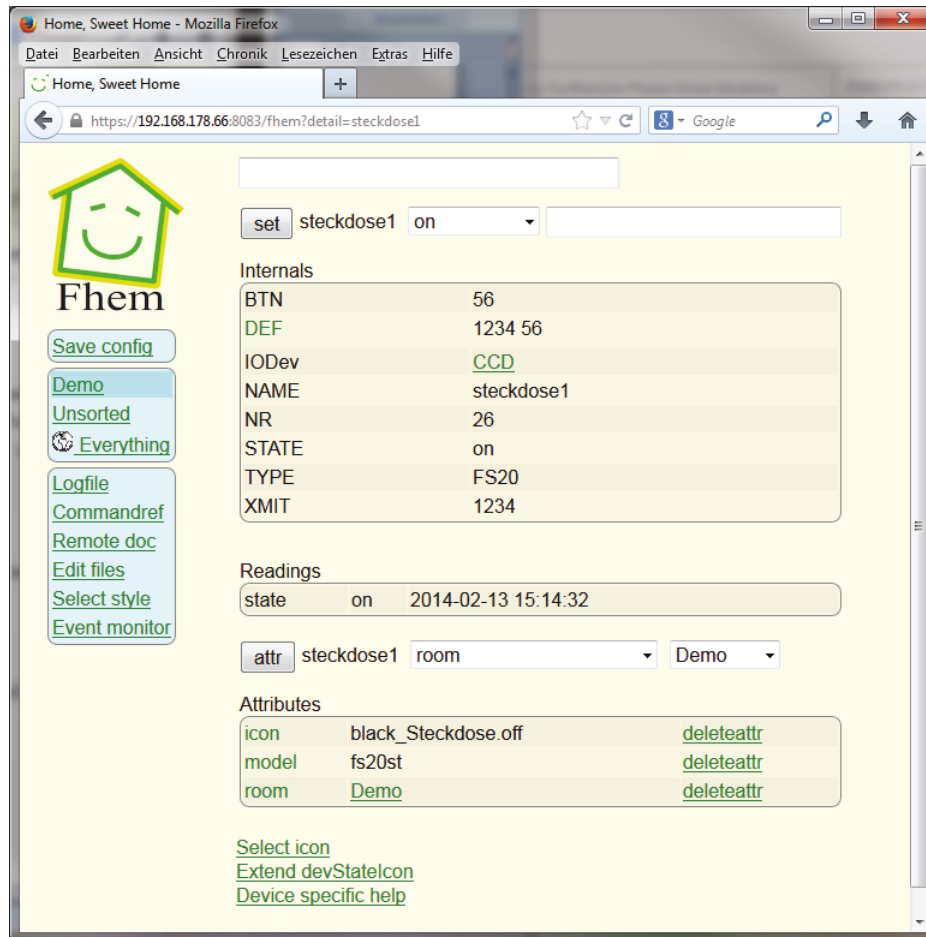



Abbildung 10.7: Die Einstellungs-Seite für das *FS-20*-Gerät lässt keine Wünsche offen.

Die *FHEM*-Nomenklatur verwendet die folgenden Begriffe:

<i>FHEM</i> -Nomenklatur	Bedeutung
<i>define</i> und <i>attr</i>	Geräte
<i>structure</i>	Strukturen
<i>room</i>	Räume
<i>group</i>	Gruppen
<i>timer</i>	Zeitsteuerung
<i>notify</i>	Notifikationen
<i>dummy</i> und <i>notify</i>	Aktionen
<i>sequence</i>	Sequenzen

Tabelle 10.1: Die wichtigsten Zuordnungen in der *FHEM*-Nomenklatur.



Wenn man plant, ein ganzes Haus oder auch nur mehrere Räume mit *FHEM* zu automatisieren, bietet es sich an, die Raum- und Hausstruktur auch auf Dateiebene abzubilden. Das kann durch *Include*-Dateien geschehen, welche in die *FHEM*-Konfigurations-Datei *fhem.cfg* eingeladen werden.

10.2.3 Alternative Sender

Wem der hier vorgestellte Transceiver zu teuer ist, der kann zu preiswerteren Lösungen greifen. Busware bietet eine Reihe von weiteren Interfaces an. Diese werden entweder an den USB-Port oder an die Raspberry Pi GPIO-Leiste angeschlossen. Das Busware COC-Interface entspricht

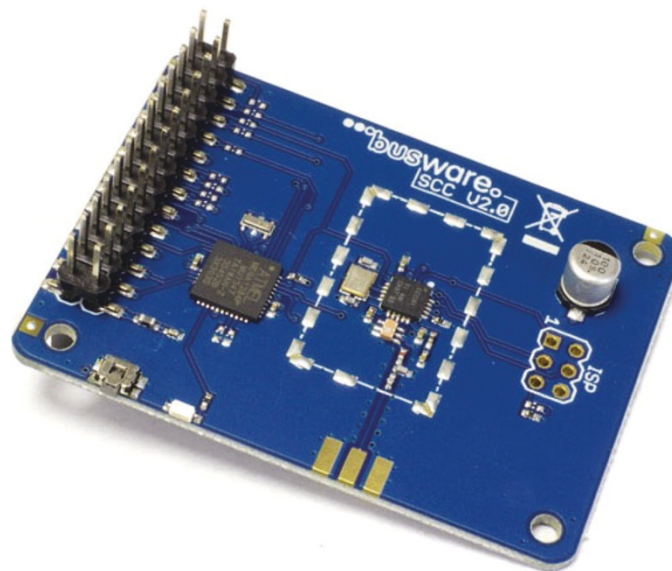


Abbildung 10.8: Das Busware SCC-Interface ist stapelbar. So können verschiedene Protokolle auf einem Raspberry Pi gefahren werden. Das passende Gehäuse gibt es gleich dazu (Quelle: Busware).

weitgehend dem CCD-Interface, mit Ausnahme des TFT LCDs, und wechselt zu einem Preis von 69 € den Besitzer. Auch dieses Interface nutzt den CC1101-Chip.



Abbildung 10.9: 433 MHz Sender zum kleinen Preis (Quelle: Watterott).

Wer noch weniger Geld für einen 433 MHz Sender (ohne Empfänger) ausgeben will, wird für unter 5 € bei Händlern wie Watterott oder Amazon fündig. Dieser abgebildete Sender hat die folgenden Anschlüsse (von links nach rechts): *GND* (Masse), *Data in*, *V_{CC}* (Versorgungsspannung) und *Antenne*. An den Antennen-Anschluss können Sie ein Kabel von ca. 17 cm Länge anschließen. Die Masse schließt man in Pin 6 des Raspberry Pi an, die Versorgungsspannung *V_{CC}* an den 5 V Pin 2 und den Dateneingang an den Pin 11 (GPIO 17). Als Funksteckdosen können Sie Modelle benutzen, die sich mit Hilfe von 10 DIP-Schaltern konfigurieren lassen. Diese kosten etwa 10 € und werden z. B. von Pollin angeboten. Zur Ansteuerung der einfachen Steckdosen kann das Programm *Rcswitch-Pi* verwendet werden. Dieses Programm setzt auf *WiringPi* auf, das wir bereits im Kapitel 9.2.2 installiert haben. Diese Installation setze ich hier voraus. *Rcswitch-Pi* installieren Sie so:

Code-Ausschnitt 10.2.19

```

1 cd /home/pi
2 git clone https://github.com/r10r/rcswitch-pi.git
3 cd rcswitch-pi
4 make

```

`make` übersetzt den Quelltext zur ausführbaren Datei `send`. Diese Datei erwartet drei Argumente als Parameter: den Hauscode, den Gerätecode und den Schaltbefehl. Die Funksteckdosen besitzen 10 DIP-Schalter. Die ersten fünf kodieren die Hausadresse, die Schalter *A* bis *E* die Geräteadresse. Der Hauscode ist eine Binärzahl. Es können also maximal $2^5 = 32$ verschiedene Hauscodes programmiert werden. Bei den Gerätecodes steht ein Schalter für ein Gerät. Es gibt damit also maximal fünf verschiedene Geräte. In Summe lassen sich also 160 verschiedene Steckdosen auf diese Art und Weise schalten. Nehmen wir an, die Steckdosen DIP-Schalter stehen auf `0001101000`. Das ist der Hauscode `00011` und der Gerätecode `2`. Sie können die Steckdose dann mit dem Befehl

Code-Ausschnitt 10.2.20

```

1 ./send 00011 2 1

```

einschalten und mit dem Befehl

Code-Ausschnitt 10.2.21

```

1 ./send 00011 2 0

```

wieder ausschalten. Die letzte Ziffer dieses Befehls bedeutet also *1* für einschalten und *0* für ausschalten. Mit Hilfe dieses Buches könnten Sie jetzt sehr einfach einen kleinen Webserver mit *CherryPy* aufsetzen, mit dessen Hilfe Sie die Steckdosen dann vom einem Handy aus schalten könnten, ohne *FHEM* zu bemühen.

10.2.4 Firmware aktualisieren

In diesem Abschnitt zeige ich Ihnen, wie Sie die Firmware der Busware-Module aktuell halten können. Die Firmware selbst sowie ein `makefile` zum Flashen der Firmware finden Sie unter <https://sourceforge.net/p/culfw/code/HEAD/tree/trunk/culfw/Devices/COC>. Damit die Firmware geflasht werden kann, ist das Programm `avrdude` erforderlich, das wie folgt installiert wird:

Code-Ausschnitt 10.2.22

```

1 sudo apt-get install make avrdude

```

Laden Sie danach den Snapshot von der oben angegebenen Webseite herunter und entpacken Sie ihn in einem Verzeichnis Ihrer Wahl. Wechseln Sie danach in das Verzeichnis, das die Firmware beinhaltet:

Code-Ausschnitt 10.2.23

```

1 cd /home/pi
2 wget https://sourceforge.net/code-snapshots/svn/c/cu/culfw/code/culfw-code-r566-trunk-↔
   culfw-Devices-COC.zip
3 unzip culfw-code-r566-trunk-culfw-Devices-COC.zip
4 cd culfw-code-r566-trunk-culfw-Devices-COC

```

Zum Flashen der Firmware ist es erforderlich, dass der serielle Port des Raspberry Pi nicht anderweitig benutzt wird.

Stellen Sie daher bitte sicher, dass der serielle Port bei älteren Betriebssystemen weder in der Datei `/etc/inittab` verwendet wird, noch in der Datei `/boot/cmdline.txt`, beispielsweise, um das Debuggen zu nutzen. Die Firmware kann nun durch einen einfachen Aufruf von

Code-Ausschnitt 10.2.24

```
1 make
```

auf das Busware-Device gespielt werden. In Abhängigkeit von Ihrer Busware-Platine rufen Sie danach

Code-Ausschnitt 10.2.25

```
1 make program_full
```

bzw.

Code-Ausschnitt 10.2.26

```
1 make program_radio_only
```

auf. Starten Sie Ihren Raspberry Pi bitte neu, nachdem Sie eine neue Firmware für das CUL aufgespielt haben.



Die Webseite <http://www.fhemwiki.de/wiki/CUL> gibt einen Überblick über alle Funktionen, welche die *CULW*-Firmware zur Verfügung stellt.

10.2.5 Vergleich der Systeme

In der Präsentation [Kra] werden noch einmal verschiedene Hausautomatisierungs-Systeme mit ihren jeweiligen Möglichkeiten dargestellt (Tab. 10.2).

System	Preis	Installation	Zentrale	Routing	Birektional
Telefunken Joonior (ENOCEAN basiert)	sehr hoch	einfach bis Profi	Zentrale mit GSM	nur Repeater, kein Routing	nein (nur über Zentrale)
Siemens Synco Living	sehr hoch	einfach	eigene Zentrale	mit Router Modul	teilweise
XComfort	sehr hoch	Profi	24h PC erforderlich	mit Profiprogrammierung	ja
ENOCEAN	hoch	einfach bis Profi	24h PC erforderlich	mit Router Modul	aktuell nein, später ja
HomeMatic	mittel	einfach	eigene Zentrale	kein Routing	ja
FS20	mittel	einfach	24h PC erforderlich	nur Schaltbefehle mit Router	nein (nur über Zentrale)

Tabelle 10.2: Überblick über die verschiedenen Hausautomatisierungs-Systeme.

10.2.6 Erweiterungen

Nach der ersten Inbetriebnahme des *FHEM*-Servers beschreibt dieser Abschnitt noch einige Erweiterungsmöglichkeiten.

Timer

In *FHEM* können auf einfache Art und Weise *Timer* gesetzt werden. Soll Ihre Steckdose z. B. nur für 20 Sekunden eingeschaltet werden, erledigt das der folgende Befehl in *FHEM*:

Code-Ausschnitt 10.2.27

```
1 set steckdose1 on-for-timer 20
```

Nach 20 Sekunden wird die Steckdose *steckdose1* wieder automatisch ausgeschaltet. Der Timer selbst wird innerhalb des Aktors (also der Steckdose) abgearbeitet. Da der letzte abgesetzte Befehl jedoch ein *on* war, leuchtet das Glühbirnen-Symbol im Frontend dauerhaft, auch nachdem die Steckdose wieder ausgeschaltet worden ist. Dies kann man mit dem Attribut *follow-on-for-timer* verhindern, welches so eingestellt wird:

Code-Ausschnitt 10.2.28

```
1 attr steckdose1 follow-on-for-timer 1
```

Zeitschaltuhr

Wahrscheinlich werden Sie häufiger eine *Zeitschaltuhr* benötigen. Ihre Steckdose soll zu einem bestimmten Zeitpunkt ein- oder ausgeschaltet werden. *FHEM* stellt hierzu den *define*-Befehl zur Verfügung.

Code-Ausschnitt 10.2.29

```
1 define steckdosean at *20:00:00 set steckdose1 on
```

Wir definieren ein Event *steckdosean*, welches um 20:00 Uhr beginnt und täglich (mit einem * gekennzeichnet) wiederholt wird. Geplante Ereignisse erscheinen unter *Everything* im Abschnitt *at*. Weitere Möglichkeiten beschreibt das Dokument <http://fhem.de/Heimautomatisierung-mit-fhem.pdf>.

Sprachsteuerung

Eine interessante Erweiterung ist eine Sprachsteuerung. Mit dieser können sie „Licht an!“ sagen und der Raspberry Pi schaltet Ihre Funksteckdose ein. Zur Sprachsteuerung hat c't Hacks [Hac] ein Skript veröffentlicht, welche die Sprachsteuerung auf einem Raspberry Pi implementiert.

Code-Ausschnitt 10.2.30

```
1 #!/bin/bash
3 # c't Hardware Hacks - Spracherkennung für den Raspberry Pi, GPL-Lizenz
5 count=1
6 lastsize=0
7 rec=0
8 first=1
10 # Der Soundchip des RPI erzeugt vor und nach der Wiedergabe ein Knacken. Deutlich ↔
    bessere Ergebnisse liefert eine USB-Soundkarte, wie man sie bereits für rund fünf ↔
    Euro bekommt. Damit mplayer die USB-Soundkarte benutzt, ändert man den Parameter "-↔
    ao alsa:device=hw=0.0" in "-ao alsa:device=hw=1.0".
```

```

12 function say {
13 mplayer -ao alsa:device=hw=0.0 -really-quiet -http-header-fields "User-Agent:Mozilla/5.0↵
    (Windows NT 6.2; WOW64) AppleWebKit/537.22 (KHTML, like Gecko) Chrome/25.0.1364.172↵
    Safari/537.22m" "http://translate.google.com/translate_tts?tl=de&q=$1";
14 }

16 sox -t alsa hw:1,0 test.wav silence 1 0 0.5% -1 1.0 1% &
17 sox_pid=$!

19 while [ $count -le 9 ]
20 do

22 size=$(stat --printf="%s" test.wav)

24 if [ $size -gt $lastsize ]
25 then
26     if [ $first -eq 0 ]
27     then
28         echo "Aufnahme!"
29         rec=1
30     else
31         first=0
32     fi
33 else
34     if [ $rec -eq 1 ]
35     then
36         echo "Abschicken"
37         kill $sox_pid
38         ffmpeg -loglevel panic -y -i test.wav -ar 16000 -acodec flac file.flac
39         wget -q -U "Mozilla/5.0" --post-file file.flac --header "Content-Type: audio/x-↵
            flac; rate=16000" -O - "http://www.google.com/speech-api/v1/recognize?lang=↵
            de-de&client=chromium" | cut -d\" -f12 >stt.txt
40         cat stt.txt
41         say "$(cat stt.txt)"

43         if [[ $(cat stt.txt) =~ "Befehl" ]]
44         then
45             echo "Sprachbefehl erkannt!"
46             say "Sprachbefehl erkannt! Ich könnte jetzt einen beliebigen Shell-Befehl ausfü↵
                hren!"

47             # mach was
48         elif [[ $(cat stt.txt) =~ "Wetterbericht" ]]
49         then
50             echo "Wetterbericht erkannt!"
51             say "Ich würde Dir jetzt den Wetterbericht vorlesen! Bring es mir bei!"
52             # mach was
53         else
54             echo "Kein Kommando erkannt..."
55         fi

57         sleep 1
58         bash ctvoice.sh
59     else
60         echo "Stille..."
61     fi
62     rec=0
63 fi

65 lastsize=$size

67 sleep 1

69 done

```

Das Skript setzt die Installation der folgenden Pakete voraus:

Code-Ausschnitt 10.2.31

```
1 sudo apt-get install sox mplayer ffmpeg
```

Es erwartet eine passende Sound-Hardware, beispielsweise ein USB-Mikrofon. Speichern Sie das Skript im Verzeichnis `/opt/fhem/contrib` unter dem Namen `ctvoice.sh` und machen Sie es ausführbar unter Verwendung des `chmod`-Befehls. Damit das Skript mit unserer Steckdose funktioniert, muss es an den entsprechenden Stellen angepasst werden, z. B. so:

Code-Ausschnitt 10.2.32

```
1 /opt/fhem/fhem.pl 7072 "set steckdose1 on"
```

Info Vielleicht ist Ihnen die Zahl `7072` im oberen *FHEM*-Aufruf aufgefallen. Sie verbindet mit *FHEM* über den *telnet*-Port. Dieser ist in der Datei `fhem.cfg` so definiert: `define telnetPort telnet 7072 global`. *Telnet* selbst steht für *Telecommunication Network* und ist ein Netzwerkprotokoll. Installieren Sie es mit `sudo apt-get install telnet`.

Info Über die *Telnet*-Schnittstelle können Sie ebenfalls mit *FHEM* kommunizieren. Melden Sie sich per *telnet* an *FHEM* an und lassen Sie sich beispielsweise alle Timer anzeigen:

Code-Ausschnitt 10.2.33

```
1 telnet ip-adresse 7072
2 inform timer
```

FHEM zeigt eine Warnmeldung, dass der *Telnet*-Zugang nicht gesichert ist. Sichern Sie den Zugang durch einen Eintrag in der Datei `fhem.cfg`:

Code-Ausschnitt 10.2.34

```
1 attr telnetPort password kennwort
```

Den Namen *kennwort* ersetzen Sie dabei mit einem Kennwort Ihrer Wahl.

Die entsprechende Zeile für das Ausschalten der Steckdose muss analog angepasst werden.

10.3 Alarmanlage

Ist Ihnen der Keller schon einmal voller Wasser gelaufen, als Sie nicht zu Hause waren? Jeder, der schon einmal einen Wasserschaden hatte, wäre sicher froh gewesen, wenn er sofort benachrichtigt worden wäre und ein Nachbar vielleicht die Möglichkeit gehabt hätte, das Wasser abzustellen. In diesem Kapitel stelle ich Ihnen eine solche Möglichkeit vor. Wir verbinden einen Sensor mit unserem Hausautomatisierungs-System und lassen uns im Falle eines Falles benachrichtigen. Die Benachrichtigung erfolgt per E-Mail oder per Anruf. Es gibt übrigens nicht nur Sensoren, die Feuchtigkeit oder Wasser melden. Die Palette reicht von Erschütterungssensoren über Temperatursensoren bis hin zu Licht- und Klingelsensoren. Der *HMS100WD* von ELV, den wir uns nun näher anschauen, ist ein Feuchtigkeitssensor, der Alarm schlägt, wenn er Feuchtigkeit detektiert. Er wird einfach auf die zu überwachende Fläche gestellt. Die hier besprochene Vorgehensweise funktioniert auch mit anderen Komponenten. Es muss dann später nur das *FHEM* Notify-Event ausgelesen werden. Im ersten Schritt binden wir den Sensor in *FHEM* ein. Danach kümmern wir uns um das Versenden der E-Mail und den Telefonanruf.



Abbildung 10.10: Der ELV Wassersensor HMS100WD benachrichtigt Sie über ungewollte Feuchtigkeit und kann so Schlimmeres verhindern (Quelle: ELV AG).

Wir beginnen damit, ein Dummy-Device in *FHEM* zu erstellen. Dieses Device nennen wir *WASSER.dummy*. Die Befehle können Sie in der *FHEM* Web-Oberfläche oder in der Datei *fhem.cfg* eingeben.

Code-Ausschnitt 10.3.1

```
1 # Wassermelder Dummy anlegen und ausschalten
2 define WASSER.dummy dummy
3 set WASSER.dummy off
```

Nach der Definition des Dummys *WASSER.dummy* schalten wir diesen aus. Nun wird der Sensor definiert. Ich nenne ihn *WDSensor*. Gleichzeitig setze ich einen Alias auf den Sensor, damit ich weiß, dass dieser den Keller überwacht.

Code-Ausschnitt 10.3.2

```
1 # HMS100WD Wassersensor einbinden
2 define WD.Sensor HMS 1002
3 attr WD.Sensor alias Sensor Keller
4 attr WD.Sensor model hms100-w
```

Nun wird das eigentliche *Notify* definiert. Das Device *WD.Sensor* wird auf das Event *.*Water.*Detect.*on.** überwacht.

Code-Ausschnitt 10.3.3

```
1 # HMS100-W Wassersensor Notify
2 define WD.not.01 notify WD.Sensor:.*Water.*Detect:.*on.* {\
3   if (ReadingsVal("WATER.dum","state","off") eq "off") {\
4     fhem("set WASSER.dummy on");\
5     fhem("define at.WASSER.dummy.off at +00:30:00 set WASSER.dummy off");\
6     '/usr/local/bin/fhem2mail SMS WASSER "Keller"';\
7     Log 3, "HMS @ Water detection %, send notification via eMail/SMS";\
8   }\
9 }
```

Wenn der Wassersensor Alarm schlägt, wird das oben definierte Dummy-Device *WASSER.dummy* auf *on* (also *an*) gesetzt. Das wird aber nur gemacht, wenn der aktuelle Status - also der vor dem Setzen - *aus* (*off*) war.

Gleichzeitig wird ein neuer *Job* definiert, der *WASSER.dummy* nach 30 Minuten automatisch wieder ausschaltet. Der Job heißt *at.WASSER.dummy.off*. Über ein Skript wird dabei eine Nachricht versendet und ein Log-Eintrag gemacht. Durch diesen Job ist sichergestellt, dass nur alle 30 Minuten eine E-Mail versendet wird und nicht bei jedem neuen in der Zwischenzeit eintretenden Event. Das Skript `/usr/local/bin/fhem2mail` wird von Martin Fischer auf seiner Webseite <http://www.fischer-net.de> zur Verfügung gestellt. *FHEM* erwartet es im Verzeichnis `/usr/local/bin`. Editieren Sie es mit *root*-Rechten, speichern Sie es und machen Sie es ausführbar.

Code-Ausschnitt 10.3.4

```

1  #!/bin/bash
2  BASENAME="$(basename $0)"
3  DST=$1
4  TYP=$2
5  ARG=$3
6  FRM="absender@mailadresse.de"
7  case $DST in
8    SMS)
9      RCP="mail2sms@empfänger.de"
10     ;;
11    MAIL)
12      RCP="mail@empfänger.de"
13     ;;
14    *)
15     ;;
16  esac
17  case $TYP in
18    BATTERY)
19      SUB="FHEM: ${ARG}"
20      TXT="Batterie schwach: ${ARG}"
21     ;;
22    FIRE)
23      SUB="FHEM: FEUERALARME ${ARG}"
24      TXT="Feueralarm wurde ausgelöst!"
25     ;;
26    GAS)
27      SUB="FHEM: GASALARME ${ARG}"
28      TXT="Gasalarm wurde ausgelöst!"
29     ;;
30    MOTION)
31      SUB="FHEM: BEWEGUNG"
32      TXT="Bewegungsmelder wurde ausgelöst!"
33     ;;
34    WATER)
35      SUB="FHEM: WASSERALARME ${ARG}"
36      TXT="Wasseralarm wurde ausgelöst!"
37     ;;
38    *)
39      SUB="FHEM: ${ARG}"
40      TXT="${ARG}"
41     ;;
42  esac
43  /usr/bin/sendemail -f ${FRM} -t ${RCP} -u ${SUB} -m ${TXT} -q -o message-content-type=↵↵
    text
44  exit 0

```

Das Skript selbst greift auf eine universelle E-Mail-Lösung von Brandon Zehm zurück. Diese kann man auf seiner Webseite <http://caspian.dotconf.net/menu/Software/SendEmail/> herunterladen.



Im Download-Bereich unter <https://www.springer.com/de/book/978-3-662-58143-8> finden Sie die E-Mail-Lösung von Brandon Zehm unter dem Namen *sendEmail-v1.XX.tar.gz*. Zur Installation sind folgende Schritte erforderlich:

Code-Ausschnitt 10.3.5

```
1 tar xvfz sendEmail-v1.XX.tar.gz
2 sudo cp -a sendEmail-v1.XX/sendEmail /usr/local/bin
3 sudo chmod +x /usr/local/bin/sendEmail
```

Das *tar*-Archiv wird ausgepackt und die einzig benötigte Datei in das Verzeichnis */usr/local/bin* kopiert und ausführbar gemacht. Bitte ersetzen Sie die Buchstaben *XX* durch die gerade aktuelle Versionsnummer. Rufen Sie das Programm mit dem Parameter *--help* auf, um eine Hilfe-Übersicht zu erhalten.

Bitte ersetzen Sie im oberen Skript die E-Mail-Adressen durch Ihre eigenen.

10.3.1 Ruf' mich an

E-Mails machen sich in der Regel nicht so laut bemerkbar wie ein Telefonanruf. Daher möchte ich Ihnen in diesem Abschnitt noch diese Möglichkeit des Alarms vorstellen. Wir beginnen mit der Sprachsynthese, die wir entweder durch Software auf dem Raspberry Pi oder im Internet erledigen können. Die dann synthetisierte Datei spielen wir in einen *Voice Over IP*-Server *VOIP* ein, der unsere Telefonnummer anruft, um uns zu sagen, dass der Keller gerade mit Wasser vollläuft. Beginnen wir mit der Sprachsynthese.

Sprachsoftware

Es gibt verschiedene Software-Lösungen auf dem Raspberry Pi, die Text in Sprache umsetzen. Dazu zählen unter anderem

- *Cepstral Text to Speech*. Diese Lösung ist keine Open-Source-Lösung und muss bezahlt werden. Die Qualität der Lösung soll besser sein als die von Lösungen mit offenem Quelltext. Informationen hierzu findet man unter <https://www.cepstral.com/raspberrypi>.
- *Festival Text to Speech*. Diese Lösung kann mit `sudo apt-get install festival` installiert werden. Ein typischer Aufruf sieht so aus:

Code-Ausschnitt 10.3.6

```
1 echo "Hello. My Name is Raspberry Pi" | festival --tts
```

- *Espeak Text to Speech*. Diese Software soll moderner sein als *Cepstral Text to Speech*. Man installiert sie mit `sudo apt-get install espeak`. Ein typischer Aufruf sieht so aus:

Code-Ausschnitt 10.3.7

```
1 espeak -ven+f3 -k5 -s150 "Hello. My name is Raspberry Pi."
```

Bei *espeak* kann der gesprochene Text durch die Angabe des Parameters *-w dateiname.wav* in einer *wav*-Datei gespeichert werden. Die Qualität der Sprachsynthese ist aber nicht sonderlich gut. Daher empfehle ich, die Sprachsynthese im nächsten Abschnitt von Google erledigen zu lassen.

Google-Synthese

Google bietet eine schöne Möglichkeit, Text in Sprache umzuwandeln. Dan Fountain hat dazu das folgende Skript veröffentlicht. Das Skript setzt das Programm *mpg123* voraus, das so installiert werden kann:

Code-Ausschnitt 10.3.8

```
1 sudo apt-get install mpg123
```

Code-Ausschnitt 10.3.9

```
1 #!/bin/bash
2 #####
3 # Speech Script by Dan Fountain #
4 # TalkToDanF@gmail.com #
5 #####
6 INPUT=$*
7 STRINGNUM=0
8 ary=($INPUT)
9 echo "-----"
10 echo "Speech Script by Dan Fountain"
11 echo "TalkToDanF@gmail.com"
12 echo "-----"
13 for key in "${!ary[@]}"
14 do
15 SHORTTMP[$STRINGNUM]="${SHORTTMP[$STRINGNUM]} ${ary[$key]}"
16 LENGTH=$(echo ${#SHORTTMP[$STRINGNUM]})
17 #echo "word:$key, ${ary[$key]}"
18 #echo "adding to: $STRINGNUM"
19 if [[ "$LENGTH" -lt "100" ]]; then
20 #echo starting new line
21 SHORT[$STRINGNUM]=$(SHORTTMP[$STRINGNUM])
22 else
23 STRINGNUM=$((STRINGNUM+1))
24 SHORTTMP[$STRINGNUM]="${ary[$key]}"
25 SHORT[$STRINGNUM]="${ary[$key]}"
26 fi
27 done
28 for key in "${!SHORT[@]}"
29 do
30 #echo "line: $key is: ${SHORT[$key]}"
31 echo "Playing line: $((key+1)) of $((STRINGNUM+1))"
32 mpg123 -w "/test.wav" -q "http://translate.google.com/translate_tts?tl=en&q=${S
33 HORT[$key]}"
34 done
```

Ich habe es so abgewandelt, dass die Ausgabe der Synthese in der Datei *test.wav* im aktuellen Verzeichnis gespeichert wird. Speichern Sie das Skript in einem Verzeichnis Ihrer Wahl unter dem Namen *sprache.sh* und machen Sie es ausführbar. Dann rufen Sie es so auf:

Code-Ausschnitt 10.3.10

```
1 ./sprache.sh "Hier ist der FHEM-Server. Der Wassermelder hat Alarm geschlagen."
```

Das Ergebnis der Synthese können Sie sich mit dem Programm *aplay* anhören.

Code-Ausschnitt 10.3.11

```
1 aplay test.wav
```

Das Programm *aplay* selbst ist Bestandteil des *alsa-utils*-Paketes und kann so installiert werden:

Code-Ausschnitt 10.3.12

```
1 sudo apt-get install alsa-utils
```

Nachdem wir nun unsere Alarmdatei (*test.wav*) erzeugt haben, richten wir *VOIP* ein. Diese Einrichtung beinhaltet zwei Schritte: Wir benötigen einen *VOIP*-Server, mit dem wir verbinden können, und wir benötigen einen *VOIP*-Klienten.

VOIP-Server

Als *VOIP*-Server nutze ich meine *Fritz!Box*. Im Menü der *Fritz!Box* können Sie unter *Telefoniergeräte* den Knopf *Neues Gerät einrichten* klicken. Wählen Sie dann *Telefon (mit und ohne Anrufbeantworter)* und *LAN/WLAN (IP-Telefon)*. Die *Fritz!Box* fordert Sie dann zur Eingabe eines Kennwortes und zur Auswahl der Nummer auf, mit der ein Gespräch geführt werden soll. Nach erfolgreichem Anlegen des IP-Telefons können Sie den Namen editieren. Bei mir habe ich dieses Telefon *Raspiphone* genannt. Da ich mit diesem Telefon nur im Alarmfall telefonieren möchte, aber keine Anrufe empfangen will, habe ich die Rubrik „nur auf folgende Rufnummern reagieren“ selektiert und die Auswahl selbst freigelassen. Sollte Ihr DSL-Modem kein *VOIP* unterstützen, können Sie sich im Internet einen *VOIP*-Provider suchen und diesen im nächsten Schritt verwenden.

VOIP-Klient

Als Klienten nutze ich auf dem Raspberry Pi das Programm-Paket von *linphone*. Installieren Sie es so:

Code-Ausschnitt 10.3.13

```
1 sudo apt-get install linphone linphone-nogtk linphone-common
```

Nach der Installation von *linphone* muss der *VOIP*-Server eingetragen werden. Starten Sie zu diesem Zweck die grafische Oberfläche, öffnen Sie ein Terminal und rufen Sie *linphone* auf.

Code-Ausschnitt 10.3.14

```
1 linphone
```

Unter *Optionen* finden Sie das Menü *Einstellungen*. Mit Hilfe des Wizards können Sie Ihren *VOIP*-Server hinzufügen. Nach der Einrichtung des *VOIP*-Klienten *linphone* benötigen wir die grafische Oberfläche nicht mehr. Wir führen nun einen ersten Test durch, indem wir *linphone* von der Konsole (Terminal) aus starten:

Code-Ausschnitt 10.3.15

```
1 linphonec
```

linphonec ist die Konsolen-Version von *linphone*, die sich nach dem Start so meldet:

Code-Ausschnitt 10.3.16

```
1 Warning: video is disabled in linphonec, use -V or -C or -D to enable.
2 linphonec> Registration on sip:623@fritz.box successful.
3 linphonec>
```

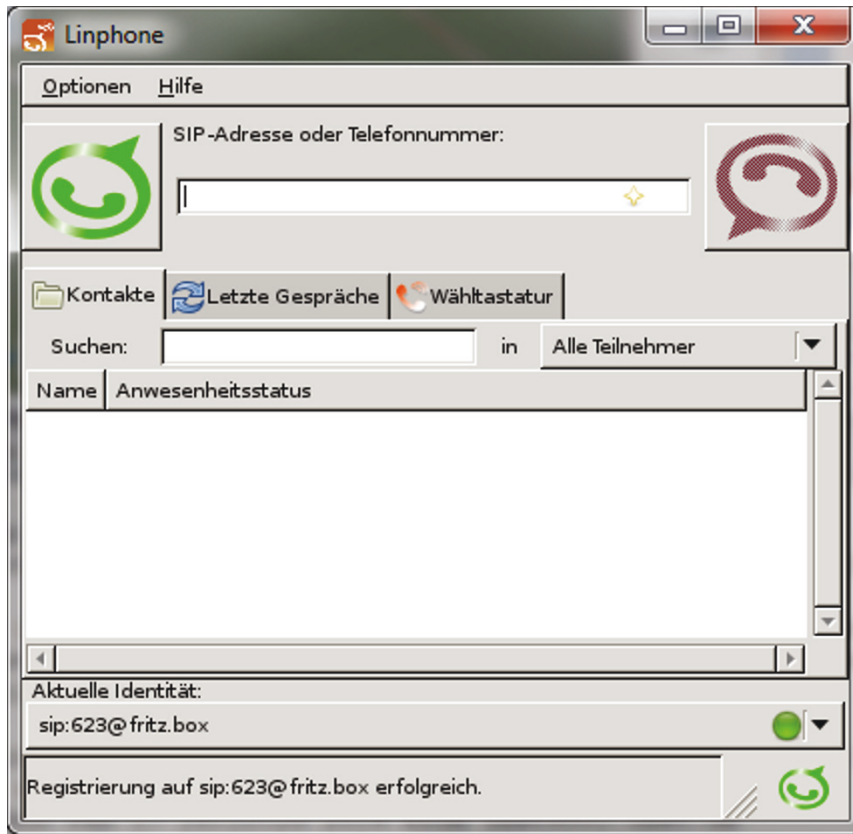



Abbildung 10.11: *linphone* ist ein VOIP-Klient für den Raspberry Pi.

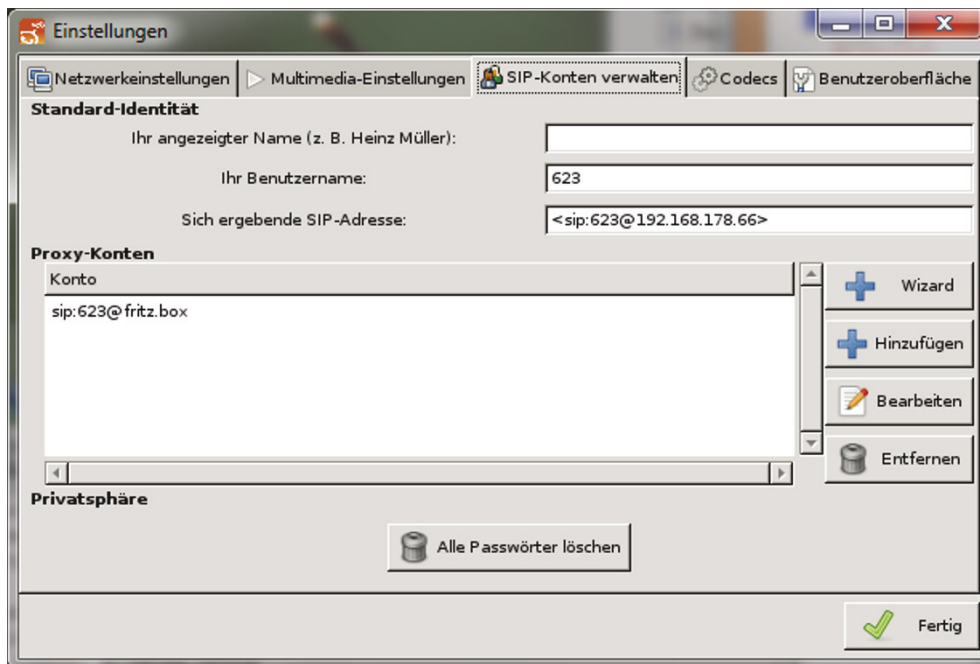


Abbildung 10.12: *linphone* bietet einen Wizard, um VOIP-Server hinzuzufügen.

Ein Aufruf von

Code-Ausschnitt 10.3.17

```
1 help
```

gibt eine Übersicht über alle Befehle. Möchten Sie eine dedizierte Hilfe zu einem bestimmten Thema, z. B. zum Thema „Soundkarte“, geben Sie Folgendes ein:

Code-Ausschnitt 10.3.18

```
1 help soundcard
```

Bei mir sieht die Ausgabe dann so aus:

Code-Ausschnitt 10.3.19

```
1 linphonec> help soundcard
2 'soundcard list' : list all sound devices.
3 'soundcard show' : show current sound devices configuration.
4 'soundcard use <index>' : select a sound device.
5 'soundcard use files' : use .wav files instead of soundcard
7 linphonec>
```

Zeile 5 ist die, die wir für einen Test brauchen. Mit dem *linphone*-Befehl „*soundcard use files*“ teilen wir *linphone* mit, dass wir eine Audio-Datei als Sprache benutzen möchten. Setzen Sie im nächsten Schritt die Audio-Ausgabe auf „Soundkarte“, wählen Sie eine Nummer Ihrer Wahl (z. B. Ihr Handy oder ein weiteres Telefon, das Sie zu Hause haben) und spielen Sie die weiter oben erzeugte Alarm-Datei ab.

Code-Ausschnitt 10.3.20

```
1 soundcard use files
2 dial 0123454321
3 play /home/pi/test.wav
4 terminate
```

Der *dial*-Befehl ruft die Telefonnummer an, die als Parameter übergeben wurde. Bitte ersetzen Sie *0123454321* mit der Nummer, die Sie anrufen möchten. Nachdem die Verbindung zustande gekommen ist, können Sie die Datei *test.wav* abspielen. Der Befehl *play* erwartet eine komplette Pfadangabe vor dem Dateinamen. Der Befehl *terminate* beendet den Anruf. Sie können *linphonec* mit der Tastatureingabe *CTRL-D* wieder schließen. Das *linphone*-Paket stellt ein Konsolen-Tool zur Verfügung, mit dessen Hilfe Anrufe per Skript getätigt werden können. Das hierzu erforderliche Programm nennt sich *linphonecsh*. Der erste Schritt im Arbeiten mit *linphonecsh* besteht darin, einen Daemon zu erzeugen, mit dessen Hilfe dann die weiteren Telefonie-Befehle ausgeführt werden können.

Code-Ausschnitt 10.3.21

```
1 linphonecsh init
```

Nach der Initialisierung muss die Registrierung am *VOIP*-Server durchgeführt werden. Erledigen Sie das mit einem Aufruf von

Code-Ausschnitt 10.3.22

```
1 linphonecsh register --host ip-adresse --username benutzername --password kennwort
```

Ersetzen Sie dabei bitte *ip-adresse* mit der IP-Adresse Ihres *VOIP*-Servers. Dasselbe gilt für *benutzername* und *kennwort*. Unter Verwendung des Schlüsselwortes *generic* versteht *linphonecsh* alle Kommandos, die *linphonec* auch versteht. So können Sie das mögliche Abspielen von Audiodateien sowie folgtaktivieren:

Code-Ausschnitt 10.3.23

```
1 linphonecsh generic "soundcard use files"
```

Der eigentliche Rufaufbau erfolgt danach mit

Code-Ausschnitt 10.3.24

```
1 linphonecsh dial 0123454321
```

Dabei müssen Sie die Zahlenfolge *0123454321* wieder durch die Nummer ersetzen, die Sie wählen möchten. Eine Audiodatei können Sie dann mit dem Befehl

Code-Ausschnitt 10.3.25

```
1 linphonecsh generic "play /home/pi/test.wav"
```

abspielen. Beenden Sie den Anruf mit

Code-Ausschnitt 10.3.26

```
1 linphonecsh hangup
```

Beenden Sie den *linphone*-Daemon mit

Code-Ausschnitt 10.3.27

```
1 linphonecsh exit
```

Einen Überblick über laufende Anrufe erhalten Sie mit

Code-Ausschnitt 10.3.28

```
1 linphonecsh generic "calls"
```

Martin Verges stellt auf seiner Internetseite <http://martinverges.blogspot.de> ein Skript zur Verfügung, welches Anrufe mit *linphonecsh* automatisiert.

Code-Ausschnitt 10.3.29

```
1 #!/bin/bash
2 #####
3
4 SIP_HOST='192.168.178.1'
5 SIP_USER='benutzername'
6 SIP_PASS='kennwort'
7 CALL=${1}
8
9 #####
10 get_phone_state () {
11     linphonecsh status register 2>/dev/null 1>&2
12     if [ $? -eq $1 ]; then
13         true
14     else
15         false
16     fi
17 }
```

```

17 }
18 get_call_state () {
19     linphonecsh status hook 2>/dev/null 1>&2
20     if [ $? -eq $1 ]; then
21         true
22     else
23         false
24     fi
25 }

27 if [ $# -ne 1 ]; then
28     echo "Usage: 'basename $0' <number_to_dial>"
29     exit 255
30 fi

32 # ret code 255 = no daemon started
33 if get_phone_state 255; then
34     echo "Starting Phone client ..."
35     linphonecsh init
36     while get_phone_state 255; do sleep 0.2; done
37 else echo "Phone client already started ..."
38 fi

40 # ret code 0 = unregistered
41 # ret code 1 = registered
42 if get_phone_state 0; then
43     echo "Register Client on SIP Server ..."
44     linphonecsh register --host ${SIP_HOST} --username ${SIP_USER} --password ${SIP_PASS}
45 else echo "Client already registered ..."
46 fi
47 while ! get_phone_state 1; do sleep 0.2; done

49 # Änderung 1: Spiele wav-Datei
50 linphonecsh generic "soundcard use files"
51     echo "Using soundcard play files..."

53     echo "Calling Number: ${CALL} ..."
54     linphonecsh dial ${CALL}
55     while get_call_state 4; do sleep 0.2; done
56     echo "Ringing on remote site ..." # what the heck,.. no hook=ringing or stuff like that

58 # Änderung 2: Verbindung wurde hergestellt. Wav-Datei abspielen
59 linphonecsh generic "play /home/pi/test.wav"

62     echo -n "Press any key to terminate the call ... "
63     ABORT=false
64     while [ "`linphonecsh status hook`" == "" ] && ! ${ABORT}; do
65         read -n1 -s -t 1; ABORT=$?
66         if [ ${ABORT} -eq 0 ]; then ABORT=true; else ABORT=false; fi
67     done
68     if get_call_state 32; then echo -n "call established"; fi
69     while get_call_state 32 && ! ${ABORT}; do
70         read -n1 -s -t 1; ABORT=$?
71         if [ ${ABORT} -eq 0 ]; then ABORT=true; else ABORT=false; fi
72     done

74     echo
75     echo "Cleaning up ..."
76     linphonecsh generic terminate 2>&1 1>/dev/null
77     echo "Exiting Daemon ..."
78     linphonecsh exit
79     echo

81 </number_to_dial>

```

Alle Änderungen im o. g. Skript habe ich markiert. Bitte denken Sie daran, die IP-Adresse Ihres VOIP-Servers korrekt einzutragen, ebenso wie den Benutzernamen und das Kennwort.

Dasselbe gilt für die abzuspielende *wav*-Datei. Speichern Sie dieses Skript im Verzeichnis */usr/local/bin*, etwa unter dem Namen *voip_call.sh*. Testen Sie das Skript beispielsweise so:

Code-Ausschnitt 10.3.30

```
1 /usr/local/bin/voip_call.sh 0123454321
```

Ersetzen Sie dabei wieder die Nummer *0123454321* mit der Telefonnummer, die Sie anrufen möchten. Dieses Skript können Sie jetzt von *FHEM* aufrufen lassen, um sich im Falle eines Alarms anrufen zu lassen.

Zusammenfassung 10 Das war das Kapitel Hausautomatisierung. Wir haben uns eine *Real Time Clock* angeschaut, die für einen Hausautomatisierungs-Server (*FHEM*) wichtig ist. Letzteren haben wir installiert und über eine *ssl*-Verschlüsselung abgesichert. Ich habe Ihnen am Beispiel der *Busware*-Hardware gezeigt, wie Sie eine Funksteckdose in den Hausautomatisierungs-Server einbinden und steuern können. Wir haben einen kurzen Blick auf alternative Sender geworfen, bevor wir eine Alarmanlage gebaut haben, die im Falle eines Wasserschadens anruft, um einen vorher abgelegten Text vorzulesen. Wer sich weitergehend mit diesem Thema beschäftigen will, kann z. B. Komponenten einer Stereoanlage steuern oder grafische Pläne für eine Hausautomatisierung hinterlegen.

Das nächste Kapitel handelt vom Schreiben mit dem Raspberry Pi. Ich werde Ihnen ein freies Office-Paket vorstellen sowie das Satz-System \LaTeX . Dabei zeige ich Ihnen u. a. eine Vorlage zum Schreiben von Bewerbungen. ■

11 — Schreiben mit dem Raspberry Pi

Jeder von uns hat sicher schon einmal mit dem Computer ein Schriftstück verfasst. Das wohl bekannteste Office-Paket für den PC stammt von Microsoft aus Redmond. In der Zwischenzeit gibt es aber durchaus brauchbare, kostenlose Alternativen wie *LibreOffice*. Wer allerdings schon einmal ein großes Werk wie ein Buch oder eine Dissertation verfasst hat, hat vielleicht auch die Vorzüge von L^AT_EX kennen- und schätzen gelernt. Hier werden Form und Inhalt strikt getrennt. Die Eingabe beispielsweise von Gleichungen mag zwar am Anfang etwas kryptisch erscheinen, das Ergebnis kann sich aber sehen lassen. Haben Sie vielleicht schon einmal bei *WORD* einen anderen Drucker eingestellt und sich gewundert, dass das Dokument danach anders aussieht? Genau das passiert bei L^AT_EX nicht. Dieses Kapitel gibt einen kleinen Überblick über das offene Office-Paket *LibreOffice* und über die Einsatz-Möglichkeiten von L^AT_EX. Letztere werden wir uns am Beispiel einer Bewerbung und eines Buches genauer anschauen.



Abbildung 11.1: *LibreOffice* stellt ein komplettes Office-Paket kostenlos bereit (Quelle: libreoffice.org).

11.1 *LibreOffice*

LibreOffice umfasst ein Programm zur Textverarbeitung, eine Tabellenkalkulation, ein Präsentationsprogramm und ein Programm zum Erstellen von Zeichnungen. Ein Formel-Editor ist genau so vorhanden wie ein Datenbank-Managementsystem. *LibreOffice* ist aus dem *OpenOffice*-Projekt entstanden. Grund für die Abspaltung war die Unzufriedenheit über die Firma *Oracle*, die das *OpenOffice*-Paket verantwortlich übernommen hat. Inzwischen hat *Oracle* diese Verantwortung wieder an die *Apache Software Foundation* abgegeben. *LibreOffice* ist ein aus zwei Wörtern zusammengesetztes Wort. Im spanischen bedeutet *libre* „frei“, im Englischen bedeutet *office* soviel wie „Büro“. Zweck von *LibreOffice* ist es also, eine freie Büro-Software zur Verfügung zu stellen. In der Raspbian-Distribution liegt *LibreOffice* als installierbares Programmpaket bereit. Installieren Sie es mit

Code-Ausschnitt 11.1.1

```
1 sudo aptitude install libreoffice
```

Wer möchte, kann direkt noch das deutsche Sprachpaket und die deutsche Hilfe installieren.

Code-Ausschnitt 11.1.2

```
1 sudo apt-get install libreoffice-help-de libreoffice-l10n-de
```

Es gibt noch viele andere Pakete für *LibreOffice* wie z. B. eine Grammatik-Überprüfung oder Clip-Art-Bibliotheken. Ein

Code-Ausschnitt 11.1.3

```
1 sudo aptitude search libreoffice
```

zeigt Ihnen alle installierbaren *LibreOffice*-Pakete an. *LibreOffice* wird nach der Installation von der X-Oberfläche aus gestartet. Falls Sie also in die Konsole gebootet haben, starten Sie zunächst die X-Oberfläche mit

Code-Ausschnitt 11.1.4

```
1 startx
```

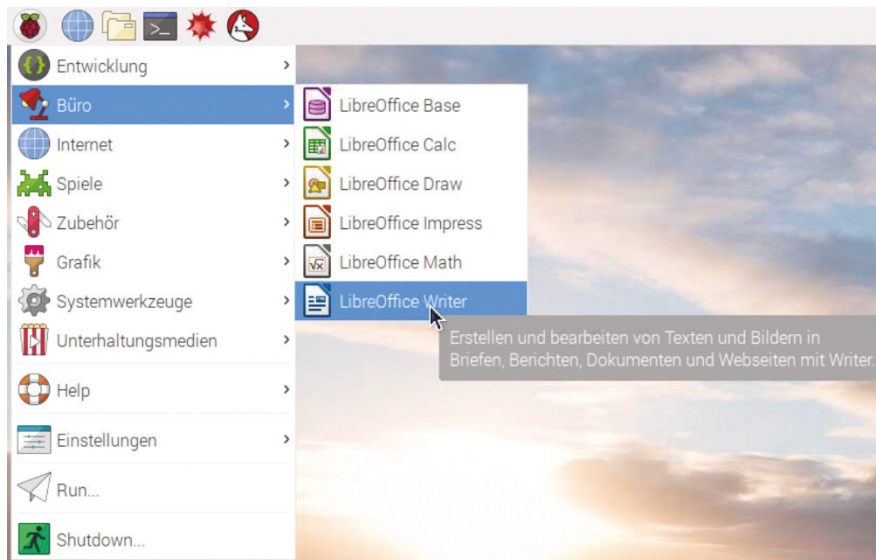


Abbildung 11.2: Nach der Installation steht *LibreOffice* unter dem Menüpunkt „Büro“ zur Verfügung.

LibreOffice steht Ihnen unter den Programmen im Menüpunkt „Büro“ zur Verfügung. Beim Start des Programmes benötigen Sie ein wenig Geduld: Der Raspberry Pi ist nicht der schnellste Computer. Nach dem Start wird Ihnen ein Auswahl-Menü präsentiert, aus dem Sie die Büro-Applikation auswählen können, mit der Sie arbeiten wollen.

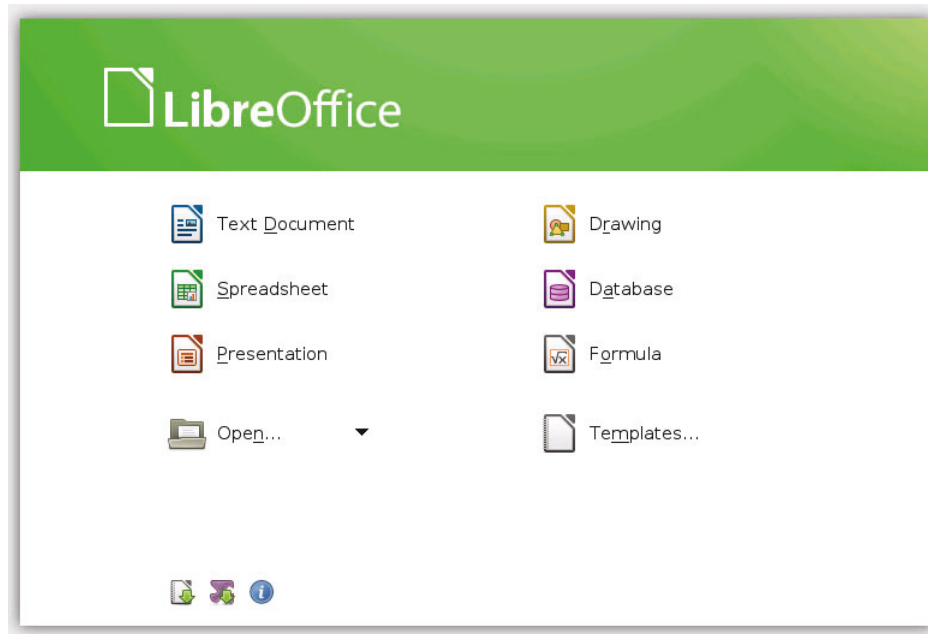


Abbildung 11.3: Das Startmenü von *LibreOffice* erlaubt die einfache Auswahl der zu startenden Büro-Applikation.

11.1.1 *LibreWriter*

Wir werfen an dieser Stelle einen kurzen Blick auf das *WORD*-Pendant names *LibreWriter*, das Sie durch einen Klick auf *Text Document* starten können.

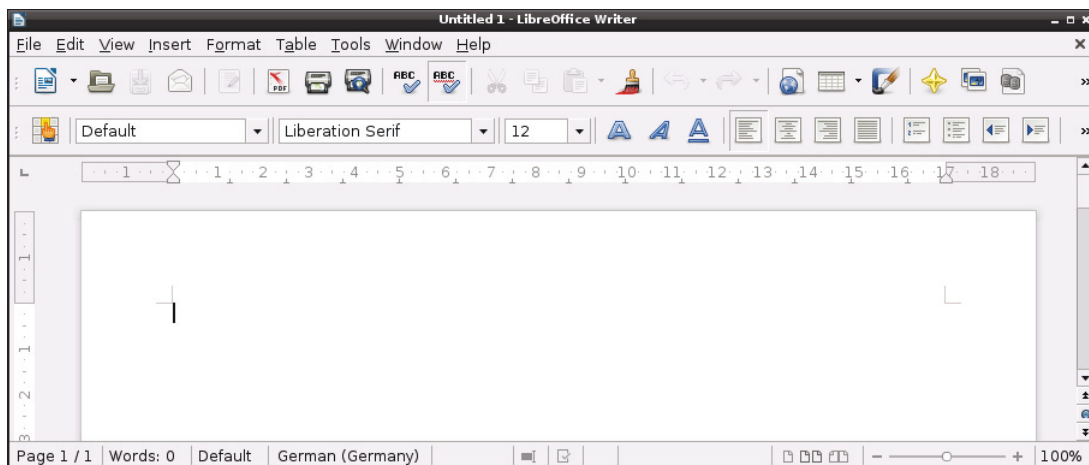


Abbildung 11.4: *LibreWriter* erlaubt das Erstellen von Texten ähnlich wie *WORD*.



Wenn Sie ein geschriebenes Dokument ausdrucken möchten, können Sie auf Drucker zurückgreifen, die Sie, wie in Kapitel 8.1 angegeben, mit Hilfe von *AirPrint* und/oder *CUPS* installiert haben.

LibreWriter erlaubt das Einlesen und Speichern von Microsoft *WORD*-Dokumenten in den Formaten *doc* und *docx*. Bei aufwendig formatierten Texten ist es mir allerdings häufiger passiert, dass die Formatierungen in *LibreOffice* anders aussahen als in Microsoft *Office*.

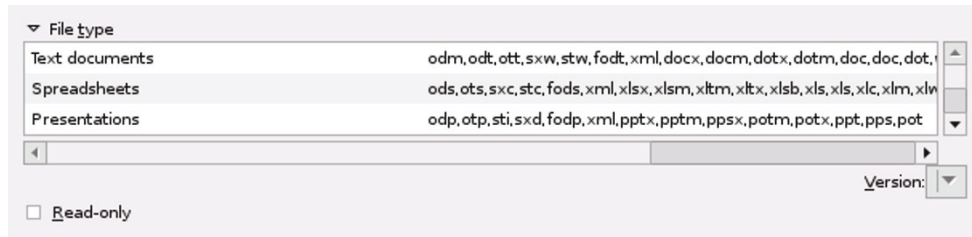


Abbildung 11.5: LibreWriter kann WORD-Dateien lesen und speichern.

Arbeitet man mit mehreren Kollegen und verschiedenen *Office*-Programmen am selben Dokument, kann das schon einmal zu Stress führen. Weitere Informationen über *LibreOffice* erhalten Sie unter <http://de.libreoffice.org/>. Obwohl der Raspberry Pi ein langsamer Rechner ist, kann man *LibreOffice* durchaus für das Schreiben kürzerer Dokumente verwenden. Wer größere Schreib-Projekte umsetzen möchte, dem empfehle ich einen Blick in Abschnitt (11.2).

11.2 L^AT_EX

L^AT_EX ist ein Programmpaket, welches das Setzen von Text mit Hilfe des Textsatzsystems T_EX unter Verwendung von *Makros* vereinfacht.



Ein *Makro* in der Computer-Sprache bedeutet die Zusammenfassung mehrerer Anweisungen zu einer kürzeren Anweisung.

L^AT_EX ist kein *WYSIWYG* (*What you see is what you get*)-System. Texte werden mit Editoren wie *vi* oder *joe* geschrieben. Inhalt und Form sind getrennt und können sogar in getrennten Dateien verwaltet werden. Erst nach einem Übersetzungslauf werden beide zusammengeführt. Formatierungen werden mit einfachem Text gekennzeichnet. So weist die Anweisung

Code-Ausschnitt 11.2.1

```
1 \chapter{Kapitelname}
```

den L^AT_EX-Übersetzer an, ein neues Kapitel mit dem Namen *Kapitelname* zu erstellen. Das sieht im ersten Moment zwar nach viel Arbeit aus, da viele sicher gewohnt sind, mit der Maus den geschriebenen Text zu markieren und dann auf *Überschrift 1* zu klicken. L^AT_EX nimmt dem Benutzer dafür an anderer Stelle sehr viel Arbeit ab: Kapitel eines Buches beginnen automatisch auf der rechten Seite, entsprechende linke Leerseiten werden automatisch eingefügt. Kapitel werden automatisch nummeriert und der definierte Absatz wird verwendet. Diese Vorgehensweise ist auch bei der Eingabe von Formeln vorteilhaft: Diese werden nicht nur in der korrekten Größe dargestellt und richtig eingerückt, sondern auch gleich noch passend nummeriert. L^AT_EX läuft auf einer Vielzahl von Betriebssystemen, darunter *LINUX*, *Windows* oder *MacOS*. Dadurch, dass Texte nicht in einem binären Format gespeichert werden (wie z. B. *docx*), sind sie für alle Rechner- und Betriebssystem-Typen kompatibel: Ein Buch, welches auf einem Raspberry Pi geschrieben wurde, kann in Sekunden auf einem Mac-Computer sichtbar gemacht werden - vorausgesetzt, L^AT_EX ist auf diesem Rechner installiert. Apropos: Wer die Minimal-Version installieren will, geht so vor:

Code-Ausschnitt 11.2.2

```
1 sudo apt-get install texlive texlive-lang-german
2 sudo apt-get install texlive-generic-extra texlive-latex-extra texlive-fonts-extra
```

Die *extra*-Pakete können - je nach Bedarf - auch weggelassen werden. Die komplette Installation erhält man nach Aufruf von

Code-Ausschnitt 11.2.3

```
1 sudo apt-get install texlive-full
```

L^AT_EX bietet eine Reihe nützlicher Zusatzprogramme an:

- *BibTeX* automatisiert das Erstellen von Literaturverzeichnissen.
- *pdfLaTeX* erzeugt aus T_EX-Dateien *PDF*-Dateien.
- *MakeIndex* erzeugt Stichwortverzeichnisse und
- *MusiXTeX* kann zum Setzen von Noten verwendet werden.

Die entsprechenden Pakete installieren Sie so:

Code-Ausschnitt 11.2.4

```
1 sudo apt-get install biber texlive-music
```

11.2.1 L^AT_EX-Beispiel

Die Wikipedia-Seite <http://de.wikipedia.org/wiki/LaTeX> stellt ein schönes L^AT_EX-Beispiel zur Verfügung.

Code-Ausschnitt 11.2.5

```

1  %% Erläuterungen zu den Befehlen erfolgen unter
2  %% diesem Beispiel.
3  \documentclass{scrartcl}

5  \usepackage[utf8]{inputenc}
6  \usepackage[T1]{fontenc}
7  \usepackage{lmodern}
8  \usepackage[ngerman]{babel}
9  \usepackage{amsmath}

11 \title{Ein Testdokument}
12 \author{Otto Normalverbraucher}
13 \date{05. Januar 2004}
14 \begin{document}

16 \maketitle
17 \tableofcontents
18 \section{Einleitung}

20 Hier kommt die Einleitung. Ihre Überschrift kommt
21 automatisch in das Inhaltsverzeichnis.

23 \subsection{Formeln}

25 \LaTeX{} ist auch ohne Formeln sehr nützlich und
26 einfach zu verwenden. Grafiken, Tabellen,
27 Querverweise aller Art, Literatur- und
28 Stichwortverzeichnis sind kein Problem.

30 Formeln sind etwas schwieriger, dennoch hier ein
31 einfaches Beispiel. Zwei von Einsteins
32 berühmtesten Formeln lauten:
33 \begin{align}
34 E &= mc^2 && \\\
35 m &= \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}} && \\
36 \end{align}
37 Aber wer keine Formeln schreibt, braucht sich
38 damit auch nicht zu beschäftigen.
39 \end{document}

```

Speichern Sie diesen Text für eine erste Begegnung mit \LaTeX unter `/home/pi/wiki.tex` und rufen Sie danach \LaTeX so auf:

Code-Ausschnitt 11.2.6

```

1  pdflatex wiki.tex

```

Der Aufruf erstellt eine Datei mit dem Namen `wiki.pdf`. Das Ergebnis ist in Abb. 11.6 zu sehen. Da es sehr viele und gute \LaTeX -Referenzen gibt, verzichte ich an dieser Stelle darauf, dieses Satzsystem zu erklären. Stattdessen stelle ich Ihnen im nächsten Abschnitt eine schöne Möglichkeit vor, Bewerbungen mit dem Raspberry Pi zu schreiben.

Ein Testdokument

Otto Normalverbraucher

05. Januar 2004

Inhaltsverzeichnis

1	Einleitung	1
1.1	Formeln	1

1 Einleitung

Hier kommt die Einleitung. Ihre Überschrift kommt automatisch in das Inhaltsverzeichnis.

1.1 Formeln

L^AT_EX ist auch ohne Formeln sehr nützlich und einfach zu verwenden. Grafiken, Tabellen, Querverweise aller Art, Literatur- und Stichwortverzeichnis sind kein Problem.

Formeln sind etwas schwieriger, dennoch hier ein einfaches Beispiel. Zwei von Einsteins berühmtesten Formeln lauten:

$$E = mc^2 \tag{1}$$

$$m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{2}$$

Aber wer keine Formeln schreibt, braucht sich damit auch nicht zu beschäftigen.

11.2.2 Bewerbungen schreiben

Mit \LaTeX lassen sich hervorragend Bewerbungen verfassen. Dabei kann man sehr einfach das Anschreiben, den Lebenslauf und die Zeugnisse zu einem elektronischen Dokument zusammenfassen, welches man dann per E-Mail an die Personalabteilung versenden kann. Im Internet finden sich viele Vorlagen für \LaTeX -Bewerbungen. Allerdings hat mir persönlich keine so gut gefallen wie die von *stefanqn*. Er stellt seine Vorlage freundlicherweise unter <https://github.com/Stefanqn/Bewerbung> zur Verfügung. Erstellen Sie sich ein Verzeichnis `/home/pi/bewerbung` und laden Sie dann die Vorlage herunter.

Code-Ausschnitt 11.2.7

```
1 cd ~
2 mkdir bewerbung
3 cd bewerbung
4 svn co https://github.com/Stefanqn/Bewerbung.git
```

Sie erhalten dann ein Verzeichnis namens *Bewerbung.git*, in dem alle benötigten Dateien enthalten sind. Die Vorlage selbst ist DIN5008-konform (http://de.wikipedia.org/wiki/DIN_5008).



Ich habe ein paar Änderungen an den Vorlagen durchgeführt. Sie können meine geänderten Vorlagen mit einem kompletten Beispiel im Download-Bereich <https://www.springer.com/de/book/978-3-662-58143-8> herunterladen.

Nach dem Download befinden sich die folgenden Unterverzeichnisse im Verzeichnis *bewerbung*:

- *Bsp-Bin*,
- *Bsp-Latex*,
- *DIN5008*,
- *Fertiges-PDF* und
- *Vorlage*.

Bsp-Bin

Das Verzeichnis *Bsp-Bin* enthält alle Dateien, die im Binär-Format vorliegen. Dazu zählen z. B. gescannte Zeugnisse im *PDF*-Format oder eine eingescannte Unterschrift im *png*-Format.

Bsp-Latex

Das Verzeichnis *Bsp-Latex* enthält drei wichtige Dateien:

1. *Inhalt-Anlagen.tex*: Diese Datei enthält alle *PDF*-Anlagen, die der Bewerbung beigelegt werden sollen.
2. *Inhalt-Anschreiben.tex*: Diese Datei enthält das Anschreiben der Bewerbung.
3. *Inhalt-CV.tex*: Diese Datei enthält den kompletten Lebenslauf.
4. *mm.tex*: Diese Datei enthält gemeinsame Daten, die sowohl im Anschreiben als auch im Lebenslauf verwendet werden. Dazu zählen z. B. der Name des Bewerbers oder der Wohnort.

Die Datei *Inhalt-Anlagen.tex* hat den folgenden Inhalt (Beispiel):

Code-Ausschnitt 11.2.8

```
1 % Weitere Anlagen
2 \newcommand* \pics {\priv Anhang/}

4 \bookmark [page=\thepage,level=0]{Arbeitszeugnisse}
5 \bookmark [page=\thepage,level=1]{Zeugnis 1}
6 \includepdf [pages=-] {\pub zeugnis1.pdf}
7 \bookmark [page=\thepage,level=1]{Zeugnis 2}
8 \includepdf [pages=-] {\pub zeugnis2.pdf}
9 ...
```

Unter der Überschrift *Arbeitszeugnisse* werden alle gescannten Zeugnisse (*PDF*-Dateien) an die Bewerbung angehängt. Gleichzeitig wird ein Inhaltsverzeichnis-Eintrag für das Zeugnis angelegt. Die Datei *Inhalt-Anschreiben.tex* stellt das komplette Anschreiben der Bewerbung im Brief-Format dar. Ein Beispiel sieht so aus:

Code-Ausschnitt 11.2.9

```
1 \setkomavar{date}{1. Januar 2014}
2 \emergencystretch=20pt\tolerance=1200\hyphenpenalty=1000% Gewichte für Trennung
3 \begin{letter}{%
4 Firma\\
5 z. Hd. Frau/Herrn XYZ\\
6 Stra{\ss}e 123\\
7 12345 Ort\\
8 }
9 \setkomavar{subject}[Betreff]{Bewerbung}
10 \opening{Sehr geehrte(r) Herr/Frau XYZ,}
11 hier steht der Bewerbungstext.
12 \newline
13 Mit freundlichen Grüßen\\sig%
14 \encl{}
15 \end{letter}
```

Die Datei *Inhalt-CV.tex* enthält den Lebenslauf. Eine typische Datei sieht so aus:

Code-Ausschnitt 11.2.10

```

1  \laengsteStadt{Hamburg}
2  %\toggletrue{keinFoto}
3  %\toggletrue{lebenslaufTitelZentriert}
4  %\thead{\anschriftKopfZeile}

6  \emergencystretch=25pt\tolerance=800\hyphenpenalty=1000\hbadness=10000\linepenalty↔
    =5000%\linepenalty
7  =40\hyphenpenalty=80%\exhyphenpenalty=3000%
8  \titel{\vspace*{5mm}\emph{Lebenslauf}\vspace*{2mm}}

10 \section{Persönliche Daten}
11 \eintragK{Name} {\nachname}
12 \eintragK{Vorname} {\vorname}
13 \eintragK{Geburtsdatum} {1. Januar 2014}
14 \eintragK{Geburtsort} {Ort}
15 \eintragK{Nationalität} {deutsch}
16 \eintragK{Familienstand} {ledig}
17 \eintragK{Anschrift} {\strasse}
18 \eintragK{} {\PLZ \ wohnort}
19 \eintragK{Telefon} {\mobilNr}
20 \eintragK{E-Mail} {\myEmail}

22 \section{Berufserfahrung}
23 \subsection{\href{http://url.de}{Firmenname}}

25 \eintragL{\von[01.2014]bis{\small heute}}{Position}{Ort}{%
26     \begin{itemize}
27     \item Stichwort 1
28     \item Stichwort 2
29     \end{itemize}
30 }

32 \eintragL{\zeit[01.2013-01.2014]}{Position}{Abteilung}{Ort}{\vskip 3mm%
33     \begin{itemize}
34     \item Stichwort 1
35     \end{itemize}
36 }

38 \section{Referenzen}
39 \eintragKb{Firma}{Name (email@firma.de)}

41 \unterschriftLL

```

In der Datei *mm.tex* stehen folgende Angaben:

Code-Ausschnitt 11.2.11

```

1  \RequirePackage{GemeinsameDaten}
2  \newcommand* \buildDir {../Vorlage/build/}
3  \meinVorname{Vorname}
4  \meinNachname{Nachname}
5  \meineStrasse{Straße Nr.}
6  \meinePLZ{PLZ}
7  \meinWohnort{Ort}
8  \meineMobilNr{Mobilnummer}
9  \meineFestnetzNr{Festnetznummer}
10 %\meineFaxNr{Faxnummer}
11 \meineEmail{email@privat.de}
12 \unterschrift[0.65][\hspace*{-30mm}]{../Bsp-Bin/Unterschrift.png}
13 \foto[4cm][0.4pt]{../Bsp-Bin/picture}

```

DIN5008

Das Verzeichnis *DIN5008* enthält lediglich die Datei *din5008frame.tex*, die sich um die DIN-gemäße Formatierung der Bewerbung kümmert.

Fertiges-PDF

Dieses Verzeichnis enthält nach Fertigstellung die kompletten Bewerbungsunterlagen im *PDF*-Format.

Vorlagen

Das Verzeichnis *Vorlagen* enthält alle T_EX-Steuerungs-Dateien. Die Datei *anschreiben.tex* bindet den Inhalt des Anschreibens im Brief-Format ein:

Code-Ausschnitt 11.2.12

```

1 \documentclass [DINmtext, draft=false]{scr1ltr2}
2 \usepackage [utf8x]{inputenc}
3 \usepackage{Vorlage_Anschreiben}
4 \input{../Bsp-Latex/mm.tex}
5 \input{../DIN5008/din5008frame.tex}
6 \farbe{blue}
7 \begin{document}
8 \input{../Bsp-Latex/Inhalt-Anschreiben.tex}
9 \end{document}

```

Die Datei *cv.tex* bindet den Lebenslauf analog ein:

Code-Ausschnitt 11.2.13

```

1 \PassOptionsToPackage{dvipsnames}{xcolor}
2 \documentclass [12pt, a4paper, sans]{moderncv} % ***test letter*** % font size ('10pt', '11←
   pt' and '12p
3 t'), paper size ('a4paper', 'letterpaper', 'a5paper', 'legalpaper', 'executivepaper' and←
   'landscape')
4 and font family ('sans' and 'roman')
5 \usepackage{Vorlage_Lebenslauf}
6 \usepackage [utf8x]{inputenc}
7 \PrerenderUnicode{ßÄäÜüÖö}
8 \input{../Bsp-Latex/mm.tex}
9 \begin{document}
10 \input{../Bsp-Latex/Inhalt-CV.tex}
11 \end{document}

```

Die entsprechenden Dateien mit dem Zusatz *Gen* generieren die dazugehörigen *PDF*-Dateien. Die Dateien mit dem Präfix *Vorlage_*.sty* bestimmen das Aussehen von Anschreiben, Lebenslauf und der gesamten Bewerbung. Das Erstellen der kompletten Bewerbungsunterlagen funktioniert nun wie folgt:

1. Füllen Sie die Datei *mm.tex* mit Ihren persönlichen Daten.
2. Schreiben Sie das Anschreiben und den Lebenslauf.
3. Passen Sie das jeweilige Datum an.
4. Rufen Sie L^AT_EX so auf:

Code-Ausschnitt 11.2.14

```

1 cd Vorlage
2 pdflatex anschreibenGen.tex
3 pdflatex cvGen.tex
4 cp *Gen.pdf build

```

Die *PDF*-Dateien für das Anschreiben und den Lebenslauf müssen in das *build*-Verzeichnis kopiert werden.

5. Erstellen Sie die komplette Bewerbungsmappe mit

Code-Ausschnitt 11.2.15

```
1 pdflatex Bewerbung_Komplett.tex
```

6. Die fertigen Bewerbungs-Unterlagen stehen danach als *Bewerbung_Komplett.pdf* im Verzeichnis *Vorlage* zur Verfügung und können in das Verzeichnis *Fertiges-PDF* zur Archivierung kopiert werden.

Ein Bewerbungs-Beispiel gibt es unter https://github.com/Stefanqn/Bewerbung/blob/master/Fertiges-PDF/Bewerbung_Komplett.pdf oder im Download für dieses Kapitel.

11.2.3 Rechtschreib-Prüfung

Mit ein wenig Übung kann man $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ hervorragend in einem kleinen Texteditor schreiben. Allerdings fehlt hier die rote Unterstreichung von Wörtern, die entweder falsch geschrieben sind oder nicht im Lexikon auftauchen. Abhilfe schafft hier das Programm *aspell*. Installieren Sie das Programm zusammen mit dem deutschen Sprachpaket so:

Code-Ausschnitt 11.2.16

```
1 sudo apt-get install aspell aspell-de
```

Nach der Installation können Sie die Rechtschreibung im Terminal durch Aufruf des Befehls

Code-Ausschnitt 11.2.17

```
1 aspell check dateiname
```

überprüfen. Bitte ersetzen Sie dabei *dateiname* mit dem Namen der Datei, die Sie bezüglich der Rechtschreibung überprüfen möchten.

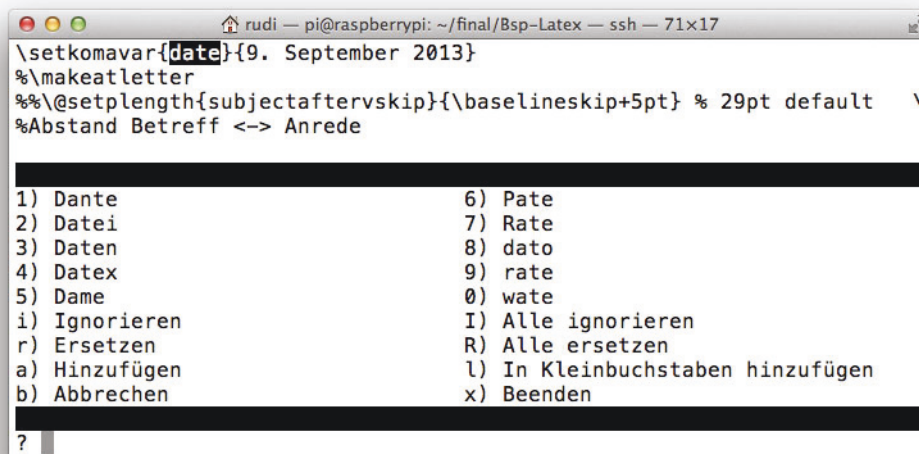
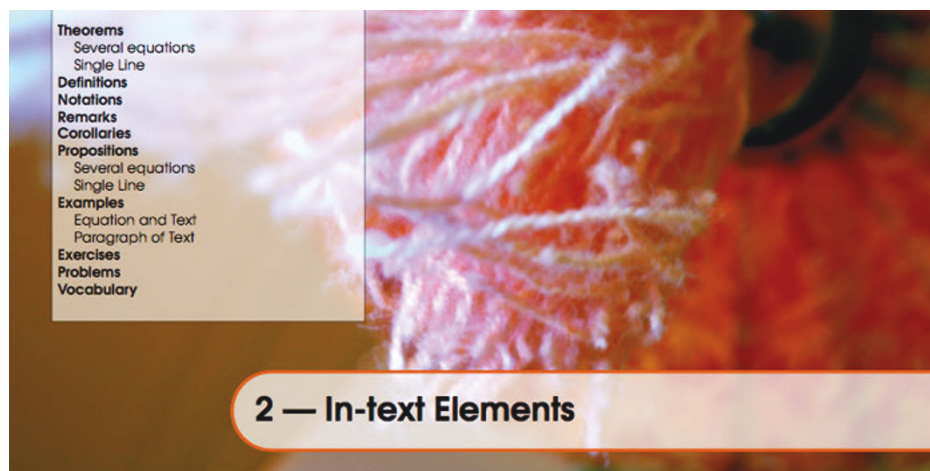


Abbildung 11.7: Mit *aspell* gibt es auf dem Raspberry Pi die Möglichkeit, die Rechtschreibung einer Datei zu überprüfen.

Während der Überprüfung können Sie entscheiden, ob ein nicht erkanntes Wort mit einem bekannten ersetzt werden soll, ob der Fehler einmal oder immer ignoriert werden soll oder ob ein nicht erkanntes Wort ins Lexikon übernommen werden soll.

11.2.4 Ein Buch schreiben

L^AT_EX eignet sich ebenfalls hervorragend, um ein Buch zu schreiben. Dass das nicht langweilig aussehen muss, beweisen die kostenlosen *Templates* von <http://www.latextemplates.com>. Unter <http://www.latextemplates.com/cat/books> stehen mehrere Buchvorlagen zum Download bereit. Das *The Legrand Orange Book-Template* ist ein Beispiel für professionelles Aussehen. Alle Templates, die auf <http://www.latextemplates.com> verfügbar sind, dürfen sogar für kommerzielle Zwecke verwendet werden, vorausgesetzt, das Projekt gibt einen Verweis auf die o. g. Webseite.



2.1 Theorems

This is an example of theorems.

2.1.1 Several equations

Theorem 2.1 In $E = \mathbb{R}^n$ all norms are equivalent. It has the properties:

$$\left| \|x\| - \|y\| \right| \leq \|x - y\| \quad (2.1)$$

$$\left\| \sum_{i=1}^n x_i \right\| \leq \sum_{i=1}^n \|x_i\| \quad \text{where } n \text{ is a finite integer} \quad (2.2)$$

2.1.2 Single Line

Theorem 2.2 A set $\mathcal{D}(G)$ is dense in $L^2(G)$, $|\cdot|_0$.

2.2 Definitions

This is an example of a definition. A definition could be mathematical or it could define a concept.

Definition 2.1 — Definition name. Given a vector space E , a norm on E is an application, denoted $\|\cdot\|$, E in $\mathbb{R}^+ = [0, +\infty[$ such that:

$$\|x\| = 0 \Rightarrow x = 0 \quad (2.3)$$

$$\|\lambda x\| = |\lambda| \cdot \|x\| \quad (2.4)$$

$$\|x + y\| \leq \|x\| + \|y\| \quad (2.5)$$

Abbildung 11.8: Das *The Legrand Orange Book-Template* ist ein professionelles L^AT_EX-Template zum Schreiben von Büchern. Quelle: www.latextemplates.com.



Das Buch-Template, welches <http://www.latextemplates.com> zur Verfügung stellt, basiert auf englischen Sprachregeln. Ich habe aus dieser Version eine Version erstellt, welche auf deutschen Regeln basiert. Die Änderungen stehen im Download-Bereich <https://www.springer.com/de/book/978-3-662-58143-8> zur Verfügung.

Zusammenfassung 11 Das war ein kurzer Ausflug in die Welt des Schreibens auf dem Raspberry Pi. Obwohl große und umfangreiche Office-Pakete auf dem Raspberry Pi installiert werden können und auch funktionieren, empfehle ich doch eher die Verwendung einer schlanken \LaTeX -Distribution. So kann man selbst auf dem Raspberry Pi ein Buch in einem kleinen Texteditor schreiben, welches höchsten Ansprüchen an Qualität und Text-Satz genügt.

Nach soviel anstrengenden Kapiteln gönne ich Ihnen im letzten Kapitel eine Erholung. Ich stelle Ihnen einige Software-Perlen vor. Lust auf ein Spiel? Ein Nintendo N64-Emulator bringt den italienischen Klempner Mario auf den Bildschirm. Das Paket *Mathematika* hilft bei der Lösung selbst komplexer mathematischer Fragestellungen. Aber dazu mehr im letzten Kapitel. ■

12 — Software-Perlen

1996 erblickte Nintendos Spielekonsole *N64* das Licht der Welt. Der Name leitet sich vom 64-Bit-Hauptprozessor der Konsole ab. Ein äußerst beliebtes Spiel für diese Konsole ist Super Mario. Während dieser italienische Klempner heute längst auf moderneren Konsolen über die Wohnzimmerbildschirme flimmert, machten vor einigen Jahren Emulatoren für den PC von sich reden, welche Nintendos Spielekonsole emulieren konnten. Seit kurzem steht ein solcher Emulator auch für den Raspberry Pi zur Verfügung. Dieses Kapitel beschreibt die Installation und die Inbetriebnahme des Emulators ebenso wie weitere Software-Perlen. Darunter finden sich Größen wie das kostenlose *Mathematika*-Paket von Wolfram Research (<http://www.wolfram.com/raspberry-pi/?fp=middle>) oder ein einfaches WLAN-Radio.



12.1 N64-Emulator

Der N64-Emulator ist ein Derivat des LINUX-Emulators *Mupen64Plus* und wird von *ricrpi* unter <https://github.com/ricrpi/mupen64plus-rpi> zur Verfügung gestellt. Das *git*-Repository beherbergt zwei Zweige: einen Entwicklerzweig (*ric_dev*) und einen stabilen Zweig (*master*). Letzten Endes bleibt es Ihnen überlassen, welchen Zweig Sie ausprobieren möchten. Erstellen Sie das Verzeichnis *n64* und laden Sie den Emulator herunter.

Code-Ausschnitt 12.1.1

```
1 cd ~
2 mkdir n64
3 cd n64
4 sudo apt-get install hgsvn
5 svn co https://github.com/ricrpi/mupen64plus-rpi.git
```

Je nachdem, welche Version Sie ausprobieren möchten, wechseln Sie in den Entwickler- oder in den stabilen Zweig.

Code-Ausschnitt 12.1.2

```
1 cd ~/n64/mupen64plus-rpi.git/branches/ric_dev
2 cd ~/n64/mupen64plus-rpi.git/trunk
```

Die nachfolgenden Schritte sind für beide Zweige identisch. Ich wähle daher für meine Betrachtungen den stabilen (*trunk*) Zweig. Das Übersetzen des Quelltextes gestaltet sich einfach. Ein Aufruf von

Code-Ausschnitt 12.1.3

```
1 ./m64p_build.sh
```

genügt und alle zum Betrieb erforderlichen Dateien landen im Verzeichnis *test*. In diesem Verzeichnis befinden sich unter anderem verschiedene Plugin-Bibliotheken (Endung *.so*) und das ausführbare Programm (*mupen64plus*). Die Konfigurations-Datei *mupen64plus.cfg* kopieren wir aus dem *build*-Verzeichnis in das *test*-Verzeichnis.

Code-Ausschnitt 12.1.4

```
1 cp mupen64plus.cfg test
```

Im *test*-Verzeichnis befindet sich ein Nintendo Test-ROM. Unter *ROM* (*Read Only Memory*) versteht man im Zusammenhang mit der Nintendo-Konsole das Speicherabbild eines Spieles. Das Test-ROM nennt sich *m64p_test_rom.v64*.



Im Internet finden sich viele N64-ROMs. Während es legal ist, den Emulator zu betreiben, ist es nicht legal, ROMs aus dem Internet herunterzuladen.

Sie können den Emulator bereits jetzt für einen ersten Probelauf starten.

Code-Ausschnitt 12.1.5

```
1 cd test
2 ./mupen64plus --configdir /home/pi/n64/mupen64plus-rpi.git/trunk/test/ m64p_test_rom.v64
```

Der Parameter *--configdir* teilt dem Simulator mit, in welchem Verzeichnis sich die Konfigurationsdatei *mupen64plus.cfg* befindet.

12.1.1 Konfiguration und Gamepad

Die Konfigurations-Einstellungen können Sie in der Datei *mupen64plus.cfg* vornehmen. Die meisten Einstellungen sind gut kommentiert. Weitere Hilfe gibt es im Englisch-sprachigen Forum <http://www.raspberrypi.org/phpBB3/viewtopic.php?f=78&t=58395>. Dort werden die folgenden Einstellungen empfohlen:

Code-Ausschnitt 12.1.6

```
1 [Video-General]
2 # Width of output window or fullscreen width
3 ScreenWidth = (Default: 640)
4 # Height of output window or fullscreen height
5 ScreenHeight = (Default: 480)

7 [Video-Rice]
8 # Control when the screen will be updated (0=ROM default, 1=VI origin update, 2=VI ↔
   origin change, 3=CI change, 4=first CI change, 5=first primitive draw, 6=before ↔
   screen clear, 7=after screen drawn)
9 ScreenUpdateSetting = (Default: 6)

11 # Force to use texture filtering or not (0=auto: n64 choose, 1=force no filtering, 2=↔
   force filtering)
12 ForceTextureFilter = (Default: 1)

14 # Force to use normal alpha blender
15 NormalAlphaBlender = (Default: False)

17 # If this option is enabled, the plugin will skip every other frame
18 SkipFrame = (Default: True)

20 Set ScreenUpdateSetting=1, SkipFrame = False
```

Info Der Programmierer empfiehlt, ältere Raspberry Pi zu übertakten (Kapitel 1.3), damit der Emulator flüssiger läuft.

DL Meine Konfigurationsdatei steht im Download-Bereich unter <https://www.springer.com/de/book/978-3-662-58143-8> zum Herunterladen bereit.

Der Emulator hat noch einige Probleme im Umgang mit der Steuerung per Tastatur, kann aber mit einem Gamepad umgehen, welches beispielsweise im USB-Anschluss steckt.



Abbildung 12.1: Mit einem Gamepad macht das Spielen am Raspberry Pi mehr Spaß (Quelle: Logitech).

12.2 RetroPie

Der N64-Emulator ist nicht der einzige Emulator für den Raspberry Pi. Freunde von C64, Amiga, ZX Spectrum und Co. kommen ebenfalls auf ihre Kosten. *RetroPie* heißt das Skript, welches viele der altgedienten (<http://blog.petrockblock.com/retropie/>) Spielekonsolen und Computer unter dem Raspberry Pi wieder zu neuem Leben erweckt. Das *RetroPie*-Skript erlaubt eine einfache Auswahl und Konfiguration der gewünschten Emulatoren. Der N64-Emulator, den ich Ihnen im vorigen Kapitel vorgestellt habe, befindet sich übrigens auch in der langen Liste der unterstützten Geräte. Mit Hilfe von zusätzlicher Hardware (RetroPie GPIO-Adapter) ist es sogar möglich, die alten Spiele-Controller zu verwenden. Eine schöne Bauanleitung für diesen Adapter findet sich unter <http://blog.petrockblock.com/2013/07/09/getting-started-with-the-retropie-gpio-adapter/>. Um *RetroPie* auf der Raspbian-Distribution zu installieren, werden zwei Programm-Pakete benutzt:



Abbildung 12.2: *RetroPie* zaubert alte Spiele-Gefühle auf den Raspberry Pi (Quelle: <http://blog.petrockblock.com/retropie/>).

Code-Ausschnitt 12.2.1

```
1 sudo apt-get update
2 sudo apt-get install git dialog
```

Das *git*-Paket kennen Sie bereits und haben es sicher schon installiert. Das *dialog*-Paket stellt Methoden zur Anzeige unterschiedlicher Arten von Dialogfeldern aus Shell-Skripten heraus zur Verfügung, die im weiteren Verlauf der Installation benötigt werden. Erstellen Sie zunächst ein Verzeichnis, in welches das *RetroPie*-Skript heruntergeladen wird.

Code-Ausschnitt 12.2.2

```
1 cd ~
2 mkdir retropie
3 cd retropie
4 git clone git://github.com/petrockblog/RetroPie-Setup.git
5 cd RetroPie-Setup
6 sudo ./retropie_setup.sh
```

Nachdem das Skript heruntergeladen wurde, starten wir es als Administrator. Das nach dem Aufruf erscheinende Auswahlmenü bietet zwei Möglichkeiten: die Installation fertig übersetzter Programme (Binaries-based installation) oder die Installation vom Quelltext ausgehend (Source-code based installation). Beide Methoden haben ihre Vor- und Nachteile. Während die Binär-Dateien eventuell veraltet sind, nimmt die Übersetzung der Quelltext-Dateien 16 bis 20 Stunden Zeit in Anspruch. Entscheiden Sie also selbst: Wenn Sie Wert auf die neuesten Versionen legen, übersetzen Sie den Quelltext. Wollen Sie nur einmal schnell schauen, was möglich ist, installieren Sie die binären Quellen. Befolgen Sie die Bildschirmanweisungen und warten Sie - je nach Auswahl - geduldig, bis die entsprechenden Programme installiert sind. Spiele-ROMS können wieder aus dem Internet besorgt werden.



Einige Hersteller stellen inzwischen offiziell *ROMs* ihrer Spiele zur Verfügung, andere nicht. Rechtlich bewegt man sich hier in einer Grauzone: Die *ROMs* bzw. die Originale gehören zwar den jeweiligen Herstellern, allerdings haben diese in der Regel kein Interesse mehr daran, illegale Downloads zu verfolgen.

Die *ROMs* müssen in das Verzeichnis `/RetroPie/roms/` kopiert werden. Dies kann beispielsweise mit Hilfe von *Winscp* (Kapitel 3.1.3) erledigt werden. *RetroPie* benutzt eine eigene grafische Benutzeroberfläche, die sich *EmulationStation* nennt. Diese darf nicht von der X-Oberfläche, sollte von einer Konsole aus gestartet werden. Sollte Ihre X-Oberfläche also laufen, beenden Sie sie (`sudo /etc/init.d/lightdm stop`). Die *RetroPie*-Oberfläche wird durch Eingabe von

Code-Ausschnitt 12.2.3

```
1 emulationstation
```

gestartet und zeigt alle Spiele an, die zuvor in das Verzeichnis `/RetroPie/roms/` kopiert worden sind. Zum Abschluss muss noch der USB-Controller (Gamepad) konfiguriert werden, sofern er vorhanden ist. Die Tasten des Controllers können so festgelegt und gespeichert werden:

Code-Ausschnitt 12.2.4

```
1 cd RetroPie/emulators/RetroArch/tools/  
2 sudo retroarch-joyconfig -o /etc/retroarch.cfg
```

Nach dem Aufruf müssen alle Knöpfe des Gamepads nacheinander gedrückt werden, um die korrekte Belegung in der Datei `/etc/retroarch.cfg` zu speichern. Die Liste der unterstützten Konsolen und Rechner kann sich sehen lassen:

- Amiga (UAE4All)
- Apple II (Basilisk II)
- Arcade (PiFBA, Mame4All-RPi)
- Atari 800
- Atari 2600 (RetroArch)
- Atari ST/STE/TT/Falcon
- C64 (VICE)
- CaveStory (NXEngine)
- Doom (RetroArch)
- Duke Nukem 3D
- Final Burn Alpha (RetroArch)
- Game Boy Advance (gpSP)
- Game Boy Color (RetroArch)
- Game Gear (Osmose)
- Intellivision (RetroArch)
- MAME (RetroArch)
- MAME (AdvMAME)
- NeoGeo (GnGeo)
- NeoGeo (Genesis-GX, RetroArch)
- Sega Master System (Osmose)
- Sega Megadrive/Genesis (DGEN, Picodrive)
- Nintendo Entertainment System (RetroArch)
- N64 (Mupen64Plus-RPi)
- PC Engine / Turbo Grafx 16 (RetroArch)

- Playstation 1 (RetroArch)
- ScummVM
- Super Nintendo Entertainment System (RetroArch, PiSNES, SNES-Rpi)
- Sinclair ZX Spectrum (Fuse)
- PC / x86 (rpix86)
- Z Machine emulator (Frotz)

12.3 Noch mehr Spiele

Es gibt eine ganze Reihe an LINUX-Spielen, die auf dem Raspberry Pi problemlos laufen. Wie wäre es mit dem Lemminge-Clone *Pingus*? Installieren Sie das Spiel so:

Code-Ausschnitt 12.3.1

```
1 sudo apt-get install pingus
```

Rufen Sie es durch Eingabe von

Code-Ausschnitt 12.3.2

```
1 pingus
```

in einem Terminal auf.



Abbildung 12.3: *Pingus* ist ein Lemminge-Clone auf dem Raspberry Pi (Quelle: pingus.seul.org).

Auf diese Art und Weise können Sie eine ganze Reihe von Spielen installieren, ganz nach Ihrem Geschmack.

12.4 Minecraft

Minecraft ist ein Spiel, welches weltweit bereits über 144 Millionen mal verkauft worden ist - nicht nur auf dem Raspberry Pi, sondern alle Betriebssysteme zusammen gezählt. Bei dem Spiel handelt es sich um ein sogenanntes *Open-World-Spiel*. Das ist ein Spiel, bei dem es sehr viele verschiedene Freiheiten gibt, es zu spielen. Der Spieler entscheidet selbst, was er wie zu welchem Zeitpunkt erkundet. Es gibt keine vorgeschriebene Handlung oder eine Spiel-Reihenfolge, an die sich der Spieler halten muss, um ein bestimmtes Level zu erreichen. In *Minecraft*, welches im Februar 2013 für den Raspberry Pi in einer abgespeckten Version erschien, baut der Spieler eine dreidimensionale Welt, die er gegen andere verteidigt. Die Welten sind mit Höhlen durchzogen, man kann Rohstoffe abbauen und weiterverarbeiten. Auf dem Raspberry Pi kann



Abbildung 12.4: Das Indie-Spiel *Minecraft* gibt es seit Februar 2013 für den Raspberry Pi. Im Vergleich zum PC-Spiel ist die *Minecraft*-Edition nicht ganz so umfangreich (Quelle: <http://minecraft-de.gamepedia.com>).

Minecraft wie folgt installiert werden:

Code-Ausschnitt 12.4.1

```
1 sudo apt-get install minecraft-pi
```

Minecraft kann so gestartet werden:

Code-Ausschnitt 12.4.2

```
1 minecraft-pi
```

Minecraft funktioniert nur auf dem Raspberry Pi direkt und nicht über eine Netzwerk-Verbindung wie *VNC*. *Minecraft* ist zwar unter Spiele → verfügbar, ich zeige Ihnen aber noch, wie Sie ein Icon auf dem Desktop anlegen können, um *Minecraft* durch einen Doppelklick auf das Icon zu starten. Wechseln Sie dazu in das *Desktop*-Verzeichnis:

Code-Ausschnitt 12.4.3

```
1 cd ~/Desktop
```

Erstellen Sie dann bitte eine Datei namens *minecraft.desktop* mit folgendem Inhalt:

Code-Ausschnitt 12.4.4

```
1 [Desktop Entry]
2 Encoding=UTF-8
3 Version=1.0
4 Type=Application
5 Exec=/usr/bin/minecraft-pi
6 Icon=
7 Terminal=false
8 Name=MCPI
9 Comment=The Minecraft Game for Pi
10 Categories=Application;Games
```

Dadurch wird das Icon auf dem Desktop erstellt, mit dem *Minecraft* gestartet werden kann.

12.5 Mathematik-Software

Im November 2013 verkündete Ebon Upton, der Gründer der Raspberry Pi Foundation, dass Wolfram Research, Entwickler des Programmes *Mathematica* dieses für den Raspberry Pi kostenlos zur Verfügung stellt. Einzige Voraussetzung ist die nicht-kommerzielle Nutzung des Tools (<http://www.wolfram.com/raspberry-pi/?fp=middle>). Das Programm selbst ist schnell installiert:

Code-Ausschnitt 12.5.1

```
1 sudo apt-get install wolfram-engine
```

Es läuft unter der X-Oberfläche und findet sich im Menü *Entwicklung*. Unter der Raspberry Pi Stretch-Distribution ist es bereits vorinstalliert. Das Programm löst Gleichungssysteme, stellt Diagramme grafisch dar und kostete in der günstigsten Studentenversion bisher 150 €. Selbst auf dem Raspberry Pi geht das erstaunlich flüssig von statten. Aber Vorsicht: Die Installation des Programmes setzt einen freien Speicherbereich von ca. 600 MB voraus.

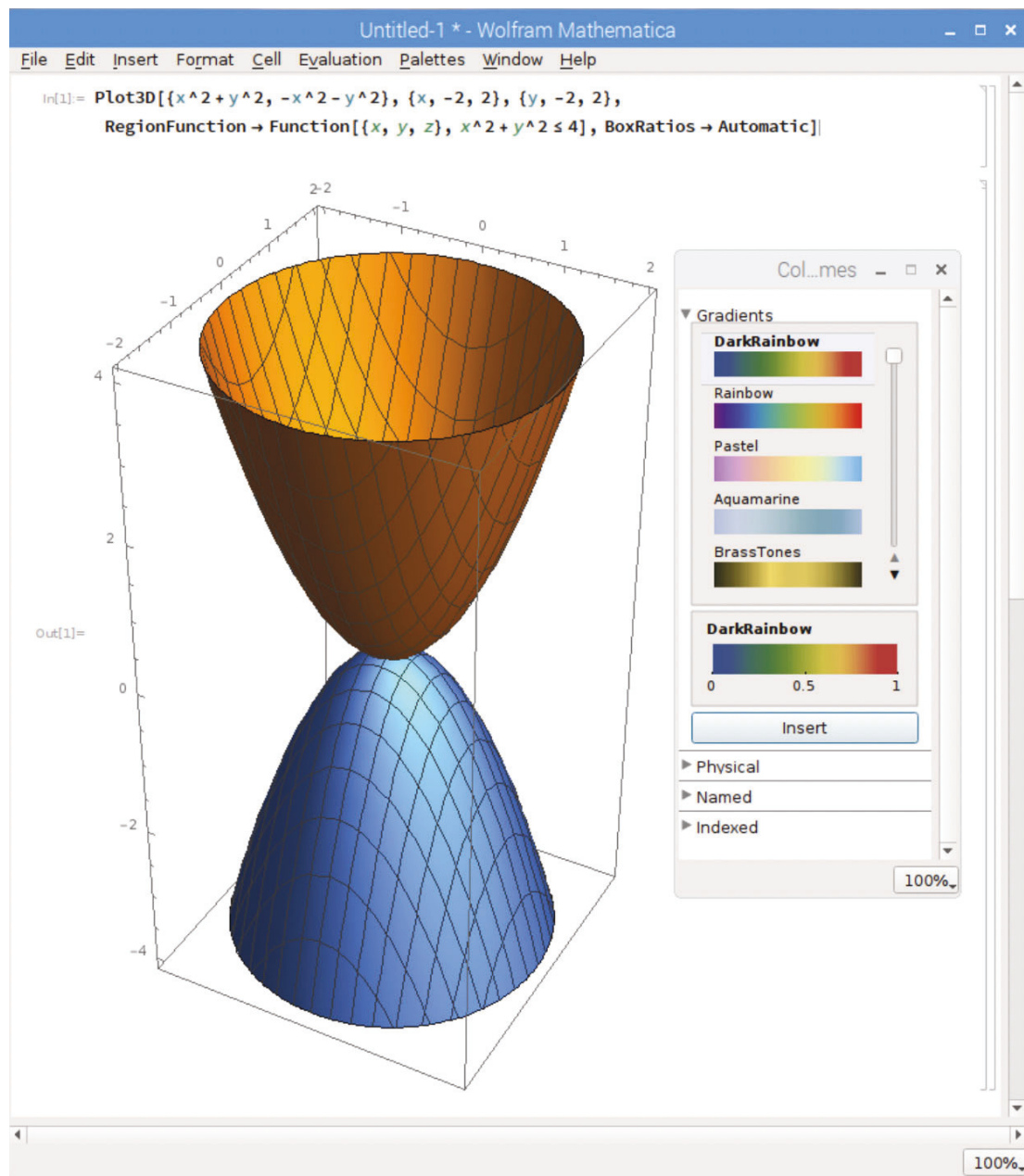


Abbildung 12.5: Mit *Mathematica* wirft Wolfram Research ein Schwergewicht in den Raspberry Pi-Ring.

Nach dem Aufruf von *Mathematica* können Sie z. B. mit folgendem Befehl eine Sinus-Funktion ausgeben lassen:

Code-Ausschnitt 12.5.2

```
1 Plot[Sin[x], {x,0,2}]
```

12.6 Fourier-Transformation

Wissenschaftlicher und Mathematiker nutzen ab und an die *Fourier-Transformation*, mit deren Hilfe ein Zeitsignal in ein Spektrum (also in den Frequenzbereich) transformiert werden kann. Dazu sind aufwendige Berechnungen erforderlich, die viel Zeit benötigen, insbesondere auf einem Raspberry Pi. Ihren Einsatz findet eine *Fourier-Transformation* beispielsweise im Lösen von partiellen Differentialgleichungen. Um den Rechen-Zeitbedarf zu reduzieren, hat Andrew Holme eine Bibliothek für den Raspberry Pi entwickelt, welche die *GPU (Graphical Processing Unit)* des Raspberry Pi nutzt, um die erforderlichen Rechnungen einer *FFT (Fast Fourier Transformation)* durchzuführen. Das steigert die Rechenleistung um den Faktor zehn. Die Beschleunigung von Grafikkarten machen sich unter anderem auch 3D-Feldsimulatoren zu Nutze. Andrews Bibliothek hat ihr zu Hause bei den Raspberry Pi-Grafikbeispielen im Verzeichnis `/opt/vc/src/hello_pi/hello_fft` gefunden und kann so übersetzt werden:

Code-Ausschnitt 12.6.1

```
1 cd /opt/vc/src/hello_pi/hello_fft
2 make
```

Das Beispielprogramm wird so aufgerufen:

Code-Ausschnitt 12.6.2

```
1 sudo mknod char_dev c 100 0
2 sudo ./hello_fft.bin
```

Eine kleine Dokumentation findet sich ebenfalls im o. g. Verzeichnis.



Falls das Verzeichnis nicht in Ihrer Installation zu finden sein sollte, helfen die folgenden Schritte:

Code-Ausschnitt 12.6.3

```
1 sudo rpi-update && sudo reboot
```

Denken Sie bitte daran, dass *rpi-update* Ihren eigenen Kernel, der eventuell installiert ist, überschreibt. Die Zeichen `&&` führen den zweiten Befehl in der oberen Zeile dann aus, wenn der erste erfolgreich durchgeführt werden konnte. Der Raspberry Pi wird also nur neu gestartet, wenn das Update erfolgreich installiert werden konnte oder überhaupt ein Update erforderlich war.

12.7 WLAN-Radio

Adafruit (<http://learn.adafruit.com/assets/7372>) hat eine schöne Anleitung zum Bau eines WLAN-Radios veröffentlicht. Wer es einfacher mag, ist mit *Radio-Tray* gut bedient. *Radio-Tray* wird so installiert:

Code-Ausschnitt 12.7.1

```
1 sudo apt-get install radiotray
```

Nach der Installation ist *Radio-Tray* im Startmenü unter *Unterhaltungsmedien* zu finden. Nach einem Klick auf das Icon in der unteren Leiste öffnet sich ein Kontextmenü, welches u. a. die Auswahl des Radiosenders erlaubt.



Abbildung 12.6: Adafruit bietet alle Teile an, die zum Bau eines WLAN-Radios erforderlich sind. Dazu gehören u. a. ein Display und diverse Taster (Quelle: Adafruit).



Abbildung 12.7: *Radio-Tray* ist ein Web-Radio für den Raspberry Pi. Nach der Installation ruft man das Programm auf und findet es in der oberen Leiste auf der X-Oberfläche.

Zusammenfassung 12 Das war das letzte Kapitel des Raspberry Pi-Kompendiums. Es hat Ihnen einen kleinen Überblick über Spiele und andere Software gegeben. Ob Retro-Spiele oder moderne Mathematik-Software: Der Raspberry Pi bietet unzählige Möglichkeiten. Stöbern Sie doch einfach weiter im Internet und in den verschiedenen Repositories. Bauen Sie z. B. Ihre eigene Cloud mit *Owncloud*. Nach den Erfahrungen, die Sie durch Lesen dieses Buches gewonnen haben, sollte das kein Problem sein.

Software ist mitunter sehr kurzlebig, was bestimmte Einstellungen oder Abhängigkeiten von anderer Software anbelangt. Sollten Sie beim Übersetzen eines Programmpaketes mit Problemen konfrontiert werden, hilft Ihnen vielleicht der kleine Abschnitt des Anhangs, der einige Informationen zur Fehlerbehebung bei der Übersetzung von Programmen bereithält.

■



Anhang

Fehlersuche

Die meisten Programme, die Sie übersetzen werden, sind sicherlich C/C++-Programme. Die ersten Fehler im Übersetzungsprozess zeigen sich schon beim Aufruf der Konfiguration mit

Code-Ausschnitt 12.7.2

```
1 ./configure
```

Meist wird ein Paket nicht gefunden, welches für das Übersetzen des Programmes erforderlich ist. In der Regel wird die Entwicklungs-Version, das sogenannte *Developer*-Paket benötigt. Die erkennt man im Repository meist am Zusatz *-dev*. Für die *Alsa-Sound-Bibliothek* wäre der Name also

Code-Ausschnitt 12.7.3

```
1 libasound2-dev
```

./configure gibt oftmals die Namen der fehlenden Pakete an. Sollte dies nicht der Fall sein, empfiehlt sich eine Internetsuche nach dem Namen des fehlenden Paketes. Wenn ein Compiler beim Übersetzen eines Programmes einen Fehler meldet, kann dies viele Ursachen haben:

- Es fehlt eine Header-Datei (*include*). Entweder ist das entsprechende *Developer*-Paket dann nicht installiert, oder der Pfad zur Header-Datei ist nicht richtig angegeben. Suchen Sie in diesem Fall nach der Header-Datei und schauen Sie, wo sich diese befindet. Header-Dateien liegen meist in den Verzeichnissen */usr/include* oder */usr/include*. Möchten Sie dort nach einer Datei namens *header.h* suchen, können Sie das so machen:

Code-Ausschnitt 12.7.4

```
1 cd /usr/include
2 find . -name "header.h" -print
```



Der *find*-Befehl findet Dateien, im oberen Fall ab dem aktuellen Verzeichnis (*.*), die *header.h* heißen, und gibt den Namen der Datei inklusive des Pfades an. Diesen Pfad können Sie dann im Fehlerfall korrekt in das C/C++-Programm eintragen.

- Steigt der C-Compiler mit einer anderen Fehlermeldung aus, hilft es oft, den Fehler im Internet zu suchen. Eventuell haben andere schon eine Lösung für den Fehler gefunden und stellen diese bereit.
- Manchmal meckert der Compiler ein *permissive* an. Damit weist er eigentlich auf un-saubere Programmierung hin, die manche Compiler akzeptieren. Der Compiler-Schalter *-fpermissive* erlaubt nicht-konformen Code und das Programm lässt sich eventuell damit übersetzen. Bei meinen Übersetzungsversuchen war dieser Schalter z. B. beim *vomplclient* erforderlich.
- Beim Linken eines Programmes fehlt oftmals eine Bibliothek. Diese kann mit der Option *-lbibname* beim Linken (z. B. im Makefile) angegeben werden. Ersetzen Sie bitte *bibname* durch den Namen der Bibliothek.
- Oftmals verschwindet der eigentliche Fehler in den Makefile-Ausgaben. Leiten Sie in diesem Fall einfach die Ausgabe in eine Datei um, die Sie dann in aller Ruhe inspizieren können.

Code-Ausschnitt 12.7.5

```
1 make >log 2>&1
```

Die Datei *log* enthält dann mögliche Fehler.

- Ein *make install* scheitert oft an den Root-Rechten. Ändern Sie ihn dann so: *sudo make install*.
- Sollte sich der C/C++-Compiler oder der Linker mit einem *internal error* verabschieden, haben Sie wahrscheinlich zu wenig *cache*-Speicher eingerichtet. Ändern Sie die Einstellung in diesem Fall wie in Kapitel 4.1.1 beschrieben und wiederholen Sie den Übersetzungsvorgang.
- Oftmals sucht man Text in Dateien, gerade bei der Fehleranalyse. Hier ist der *grep*-Befehl eine große Hilfe.

Code-Ausschnitt 12.7.6

```
1 grep wort *
```

sucht das Wort *wort* in allen (*) Dateien im aktuellen Verzeichnis.



Das *-Zeichen steht für alle Dateien im aktuellen Verzeichnis. Möchten Sie beispielsweise alle Backup-Dateien des Editors *joe* löschen, können Sie das mit dem Befehl

Code-Ausschnitt 12.7.7

```
1 rm *~
```

- Oftmals ändern sich Funktionsaufrufe, weil sich Parameter ändern oder neue Parameter dazugekommen sind. In diesem Fall müssen die Aufrufe mit den neuen Parametern angepasst werden.
- Viele Repositories ändern sich täglich. Hier hilft es, eine bestimmte Version auszuchecken. Mit *git* kann eine bestimmte Version so ausgecheckt werden:

Code-Ausschnitt 12.7.8

```
1 git clone git://repository/project.git
2 cd project
3 git checkout version
```

Die wichtigsten LINUX-Befehle

Tabelle 12.1 fasst die wichtigsten LINUX-Befehle zusammen.

LINUX-Befehl	Bedeutung
<i>bzip2 datei</i>	Komprimiert eine Datei
<i>cat /proc/cpuinfo</i>	Zeigt Information über die CPU an
<i>cd ~</i>	Wechselt in das Benutzer-Home-Verzeichnis
<i>cd verzeichnis</i>	Wechselt in ein Verzeichnis
<i>CTRL-c</i>	Beendet den aktuellen Vorgang im Terminal
<i>cp name1 name2</i>	Kopiert eine Datei
<i>date</i>	Zeigt Datum und Zeit an oder setzt diese
<i>df -hk</i>	Zeigt den freien Speicherplatz an
<i>dmesg</i>	Zeigt die Kernel-Meldungen an
<i>exit</i>	Beendet ein Terminal oder meldet Administrator ab
<i>fdisk</i>	Partitioniert einen Datenträger
<i>find wo name</i>	Sucht nach Dateinamen
<i>free</i>	Zeigt freien Arbeitsspeicher an
<i>grep was datei</i>	Sucht in einer Datei
<i>ifconfig</i>	Zeigt Netzwerk-Informationen an
<i>joe name</i>	Editiert eine Datei
<i>kill -9 pid</i>	Beendet einen Prozess
<i>ls -la</i>	Zeigt alle Dateien und Verzeichnisse im aktuellen Verzeichnis an
<i>man name</i>	Ruft die Hilfeseite zu einem Programm auf
<i>mkdir name</i>	Erzeugt ein neues Verzeichnis
<i>modprobe</i>	Lädt ein Kernel-Modul
<i>more name</i>	Zeigt den Inhalt einer Datei seitenweise an
<i>mount</i>	Bindet Laufwerke in Verzeichnisse ein
<i>mv name1 name2</i>	Benennt eine Datei um oder verschiebt sie
<i>ping ip-adresse</i>	Testet die Verbindung zu einem Rechner
<i>patch < datei</i>	Patched ein Programm mit einer Patchdatei
<i>ps -elf</i>	Zeigt alle laufenden Prozesse an
<i>ps -elf grep name</i>	Zeigt Prozess-ID eines Prozesses an
<i>pwd</i>	Zeigt das aktuelle Verzeichnis an
<i>rm name</i>	Löscht eine Datei
<i>rmdir name</i>	Löscht ein Verzeichnis
<i>rmmod</i>	Entlädt ein Kernel-Modul
<i>startx</i>	Startet die X-Oberfläche
<i>sudo halt -p</i>	Führt das System herunter
<i>sudo su</i>	Meldet den Systemadministrator an
<i>tail -f /var/log/syslog</i>	Zeigt Systemlog permanent an
<i>tar cvf datei.tar *</i>	Packt alles aus dem aktuellen Verzeichnis in ein Archiv
<i>tar xvjf datei</i>	Packt eine <i>bzip2</i> -komprimierte Datei aus
<i>top</i>	Zeigt alle laufenden Prozesse und die Auslastung an
<i>uname -a</i>	Zeigt die aktuelle Kernel-Version an
<i>vi name</i>	Editiert eine Datei
<i>which name</i>	Zeigt den Pfad zu einem Programm an
<i>who</i>	Zeigt, welche Benutzer im System eingewählt sind
<i>whoami</i>	Zeigt aktuell eingewählten Benutzer an

Tabelle 12.1: Die wichtigsten LINUX-Befehle.

Danksagung

Computer sind meine Leidenschaft. Das waren sie schon, als ich noch ein kleiner Junge war und meine ersten Geh-Versuche mit einem BASIC-Interpreter und Assembler für den ZX-Spectrum unternommen habe. Dieser Rechner kostete damals über 500 DM. Für einen Bruchteil dieses Preises gibt es heute den Raspberry Pi - zusammen mit dem kostenlosen *LINUX*-Betriebssystem. Mit dem Raspberry Pi habe ich viele Stunden, Tage und Wochen verbracht, Software installiert, programmiert, getestet. Das hat sehr viel Spaß gemacht, aber auch viel Zeit gekostet. Zeit, in der meine Familie auf mich verzichten musste. Lea und Jonas, ihr seid die Besten! Darum gilt mein Dank zuerst meiner Frau, die mich ermutigt hat, dieses Buch zu schreiben, um meine Erfahrungen zu teilen. Obwohl sie immer noch wenig Ahnung von Computern hat, hat sie das Manuskript der Neuauflage komplett gelesen, korrigiert und mir wertvolle Hinweise gegeben. Ich danke meinem Vater, dem ich meine Technik-Verbundenheit zu verdanken habe, für das Lesen des Manuskriptes und die wertvollen Hinweise zur besseren Verständlichkeit.

Ich danke meiner Mutter, die auch in schwierigen Situationen immer zu mir gehalten hat.

Ein großer Dank geht an Klaus Schmidinger für das Aufspüren von Fehlern und das Einbringen von Korrekturen, das Schreiben des Geleitwortes und vor allem für den VDR, der für viele der Hauptgrund ist, sich überhaupt mit LINUX, PCs und Embedded Boards zu beschäftigen.

Ich bedanke mich bei meinem verstorbenen Kollegen Dr. Dietmar Köther für die vielen anregenden Diskussionen und bei allen, die mich ganz spontan mit Foto-Material für dieses Buch versorgt haben.

Dieses Buch wäre nicht möglich gewesen ohne die vielen Programmierer, die dazu beitragen, freie Software zu liefern und am Leben zu halten. Ich hoffe, dass das Buch Ihnen einen Einblick in diese Welt und in die Welt der Embedded Boards gegeben hat und dazu anregt, eigene Projekte, Wünsche und Vorstellungen umzusetzen.

Unter <https://www.springer.com/de/book/978-3-662-58143-8> finden Sie alle Downloads zum Buch. Außerdem gibt es dort die Errata und eine Datei, die alle Code-Ausschnitte enthält.

Mettmann, im August 2018
Rüdiger Follmann



Literatur

Bücher

- [Mül13] Peter Müller. *Websites erstellen mit Contao 3*. 1. Auflage. Band 3. 2. Galileo Computing, 2013. ISBN: 978-3-8362-2010-1 (siehe Seite 218).
- [Sch13b] Maik Schmidt. *Raspberry Pi: Einstieg - Optimierung - Projekte*. 1. Auflage. Band 1. Heidelberg: dpunkt-Verlag, 2013 (siehe Seite 180).

Artikel

- [Döl14] Mirko Dölle. “Rechenzwerge”. In: *c’t magazin für computer technik* 3 (2014), Seiten 84–89 (siehe Seite 1).
- [lmd14] lmd. “Multimedia-Zwerg”. In: *c’t magazin für computer technik* 3 (Jan. 2014), Seiten 90–93 (siehe Seite 119).
- [Sch13a] Jürgen Schmidt. “Eigen-Tor, Gefahren der Tor-Nutzung im Alltag”. In: *c’t magazin für computer technik* 20 (2013), Seite 102 (siehe Seite 206).
- [Sto18] Ingo T. Storm. “Der letzte seiner Art”. In: *c’t magazin für computer technik* 7 (2018), Seiten 44–45 (siehe Seite 4).

Verschiedenes

- [Hac] C’t Hacks. *ctvoice.sh*. <http://www.heise.de/hardware-hacks/links/1302064> (siehe Seite 274).
- [Kra] Dr. Dennis Krannich. *Hausautomation mit dem RaspberryPi + FHEM*. Internet (siehe Seite 273).



Abbildungsverzeichnis

1.1	Der Raspberry Pi (Model 3B+) ist ein vollwertiger Rechner im Kreditkartenformat (Quelle: Reichelt).	1
1.2	Foundation Logo (Quelle: raspberrypi.org).	2
1.3	Das Pibow Gehäuse sieht aus wie ein Regenbogen (Quelle: Pimoroni).	2
1.4	Der Raspberry Pi Zero W (Quelle: Pimoroni).	3
1.5	Das Raspberry Pi Compute Modul ist hauptsächlich für Firmen gedacht, die ihr eigenes PCB entwickeln (Quelle: raspberrypi.org).	3
1.6	Die Anschlüsse des Raspberry Pi 3B+.	5
1.7	Die Schnellstart-Anleitung zeigt, was wo am Raspberry Pi angeschlossen werden muss (Quelle: raspberrypi.org).	6
1.8	Win32 Disk Imager kann zum Überspielen des Betriebssystems auf die SD-Karte genutzt werden.	7
1.9	Das grafische Konfigurationsprogramm für den Rasperry Pi.	8
1.10	Der erste Startbildschirm des Raspbian-Betriebssystems und seine Einstellmöglichkeiten.	9
1.11	Die Raspbian X11-Oberfläche.	13
1.12	Mit dem Chromium-Webbrowser kann man im Internet surfen.	14
1.13	Das LXTerminal ist eine Konsole zur Kommandoingabe	14
2.1	Das Desktop-Tool zum Hinzufügen und Entfernen von Software erlaubt eine einfache Sortierung nach Themenbereichen.	29
2.2	Der Fernzugriff ist mit Putty kinderleicht.	30
2.3	Der VNCViewer erlaubt es, den Desktop des Raspberry Pi auf anderen Computern/Monitoren sichtbar zu machen und zu bedienen.	31
2.4	VNCViewer speichert erfolgreiche Verbindungen inklusive einer Vorschau.	32
2.5	Die Hilfe zum Editor <i>Joe</i> kann man im Editor selbst aufrufen.	34
2.6	Kleiner geht es kaum: der EDIMAX USB WLAN-Stick (Quelle: EDIMAX).	36
2.7	Grafische Oberfläche mit 2 WLAN-Adapttern (intern und USB).	38
2.8	Per Bluetooth eingebundene Kopfhörer müssen im Audio-Menü aktiviert werden.	39
2.9	Mit Gparted kann man Partitionen verkleinern.	44
2.10	Der E-Mail-Client <i>Claws Mail</i> bietet IMAP- und POP3-Zugang.	47

2.11	Mit dem Dateimanager können alle Datei-Operationen durchgeführt werden. . . .	48
3.1	X-Weiterleitung mit Putty.	51
3.2	Die Einstellmöglichkeiten von Xlaunch.	52
3.3	Die Xlaunch Zugriffskontrolle.	52
3.4	Fernzugriff auf xrdp.	53
3.5	Der Pi Desktop wird im Windows Desktop sichtbar.	54
3.6	Winscp erlaubt sicheres Kopieren von und zum Raspberry Pi.	55
3.7	Das Samba Logo (Quelle: www.samba.org).	61
3.8	Samba zeigt Raspberry Pi-Dateien im Windows Explorer.	62
3.9	k3b DVD-Suite kann auf entfernte Brenner zugreifen, die mit <i>iSCSI</i> eingebunden sind.	64
4.1	Python-Logo (Quelle: python.org)	77
4.2	TUX der Pinguin (Quelle: L. Ewing).	79
5.1	Das VDR-Portal ist die erste Anlaufadresse für Fragen rund um den <i>VDR</i> (Quelle: Jan Grell).	85
5.2	Der Sundtek-Stick bringt seine eigenen Treiber mit (Quelle: Sundtek).	86
5.3	Die DVBSky S960 V2-Box benötigt eine Firmware-Datei (Quelle: amazon.de). . . .	90
5.4	Philips Universalfernbedienung (Quelle: Philips).	93
5.5	Infrarot-Empfänger selbst gebaut: eine IR-Diode und drei Kabel reichen.	93
5.6	Die GPIO-Leiste des Raspberry Pi und ihre Möglichkeiten.	94
5.7	Die fertige <i>lirc</i> -Verkabelung an einem alten Raspberry Pi. Die Verkabelung für einen neuen Raspberry Pi sieht genau so aus, wenn Sie die GPIO-Pins von rechts aus zählen.	95
5.8	MSL Digital bietet eine Zusatzplatine mit <i>lirc</i> -Empfänger und Einschalt-Knopf (Quelle: msldigital.com).	98
5.9	Pi mit Infrarot-Modul und Einschaltknopf mit oder ohne Gehäuse. Dank MSL Digital kommt man dabei ganz ohne Lötarbeiten aus.	99
5.10	VDR ist die LINUX-TV-Applikation schlechthin (Quelle: tvdr.de).	99
5.11	Eine per USB eingebundene 2,5 Zoll-Festplatte erweitert den Speicherplatz des Raspberry Pi. Aber Vorsicht: Für die externe Festplatte kann eine externe Spannungsversorgung erforderlich sein.	106
5.12	VDRAdmin-am erlaubt das Programmieren von Aufzeichnungen vom Internet aus (Quelle: vdr-wiki.de).	108
5.13	Metrix HD ist ein wunderschönes Skin für den <i>VDR</i> . Dank OpenGL läuft es auch auf einem Raspberry Pi flüssig (Quelle: vdr-portal.de).	109
5.14	Das Logo des VLC Media-Players (Quelle: www.vlc.de).	116
6.1	Das KODI Mediacenter spielt Filme, zeigt Bilder und vieles mehr auf dem Raspberry Pi.	119
6.2	LibreELEC – Just enough OS for KODI (Quelle: kodinerds.net).	128
6.3	Estuary ist der Standard-Skin für <i>KODI</i> (Quelle: kodi.wiki).	128
6.4	<i>tvheadend</i> wird komfortabel mit Hilfe eines Web-Interfaces konfiguriert.	132
6.5	Mit Hilfe von <i>samba</i> können die Logo-Dateien schnell an ihren Bestimmungsort kopiert werden.	133
6.6	Die Verzeichnisse für die Senderlogos können im <i>tvheadend</i> Web-Frontend eingestellt werden.	134
6.7	Mit Senderlogos sieht <i>tvheadend</i> gleich viel schöner aus.	135

6.8	Amazon VOD.	136
6.9	Als Wiedergabemethode im <i>Amazon VOD</i> Addon muss <i>Input Stream</i> gewählt werden.	136
6.10	Die Verzeichnisse für die Senderlogos können im <i>tvheadend</i> Web-Frontend eingestellt werden.	140
6.11	Die <i>Kore</i> App unter Android.	140
6.12	Auf einem Apple-Gerät taucht <i>shairport-sync</i> unter dem Namen „Raspberrypi“ auf.	142
6.13	Delock 61961 ist eine preiswerte USB-Soundkarte für den Raspberry Pi (Quelle: Tragant).	143
7.1	Ambilight verlängert das Fernsehbild über die Ränder des TVs hinaus (Quelle: Stephan Legachev, wikipedia.de).	147
7.2	Der APA102(C) LED-Streifen eignet sich hervorragend, um ein Ambilight nachzubauen (Quelle: amazon.de).	148
7.3	Einen Arduino-Nano-Nachbau gibt es günstig im Dreierpack. Der Elgoo Nano V3 mit CH340 USB-Chip wird von <i>LibreELEC</i> ohne Probleme erkannt (Quelle: amazon.de).	149
7.4	Die Pins des Arduino Nano (Quelle: Arduino Forum, user pighixxx).	150
7.5	So verkabeln Sie Ihr Ambilight richtig	150
7.6	So schneiden Sie den LED-Streifen richtig.	151
7.7	Der LED-Streifen kann mit Hilfe von Klebehaken und Kabelbindern hinter dem Fernseher befestigt werden.	151
7.8	<i>LibreELEC</i> stellt das <i>Hyperion</i> -Addon unter <i>Dienste</i> bereit.	157
7.9	Mit dem Java-Tool <i>Hypercon</i> erstellt man schnell und komfortabel eine Konfigurationsdatei <i>hyperion.config.json</i> für <i>Hyperion</i> (Quelle: wiki.libreelec.tv).	161
7.10	Ambilight sieht selbst hinter einem curved-Fernseher gut aus.	161
7.11	<i>Hyperion remote</i>	162
7.12	Die Struktur eines <i>KODI</i> -Addons.	163
7.13	Das eigene <i>Hyperion</i> -Addon.	165
7.14	433 MHz Sender (rechts) und Empfänger (links) (Quelle: amazon.de).	165
7.15	Der 433 MHz-Sender kann mit Jumperkabeln schnell mit dem Arduino Nano verbunden werden.	166
7.16	HDMI-Splitter (Quelle: amazon.de).	168
7.17	HDMI nach CVBS Konverter (Quelle: amazon.de).	168
7.18	CVBS Grabber (Quelle: amazon.de).	168
7.19	Das Ambilight für jede HDMI-Quelle benötigt noch einige Kleinteile.	169
7.20	Verkabelung der LED-Kette mit Netzteil und Raspberry Pi.	169
7.21	Level-Konverter 3.3 V auf 5 V (Quelle: lightberry.eu).	170
7.22	Verkabeln Sie den HDMI-Splitter, den Konverter und den Grabber wie in dieser Abbildung gezeigt. Die jeweiligen Netzteile habe ich wegen der Übersichtlichkeit nicht angeschlossen.	170
7.23	<i>Hypercon</i> -Konfiguration 1. Teil.	172
8.1	Der <i>CUPS</i> -Startbildschirm im Web-Browser.	177
8.2	Die <i>CUPS</i> -Verwaltung im Web-Browser.	178
8.3	Die <i>Apache</i> Standard-Seite sagt „Hallo!“.	180
8.4	Die <i>PHP</i> -Infoseite listet alle Standard-Einstellungen auf.	182
8.5	(Quelle: mysql.de)	182

8.6	(Quelle: phpmyadmin.net).	183
8.7	Die <i>phpmyadmin</i> -Oberfläche verwaltet die <i>MySQL</i> -Datenbank.	184
8.8	Der TP-Link TL-WN722N hat eine große Reichweite dank externer Antenne.	185
8.9	(Quelle: chillispot.org)	191
8.10	<i>Chillispot</i> -Login auf einem Telefon.	195
8.11	Ein Web-Interface zum Anlegen von Hotspot-Nutzern.	202
8.12	Ein PDF-Dokument zum Hotspot-Zugang.	202
8.13	(Quelle: torproject.org).	205
8.14	IP-Ermittlung im Internet: <i>TOR</i> funktioniert wie erwartet.	207
8.15	(Quelle: openvpn.net).	208
8.16	<i>OpenVPN</i> auf dem iPhone.	213
8.17	(Quelle: Leo Feyer).	214
8.18	Der <i>Contao</i> Installations-Screen verlangt die Eingabe eines Installations-Kennwortes.	216
8.19	Die Aktualisierung der Datenbank.	217
8.20	Login-Seite für das <i>Contao</i> -Backend.	217
8.21	Geschafft: Das <i>Contao</i> -Beispiel <i>Demo</i> ist installiert.	218
9.1	Busware macht's möglich: Raspberry Pi mit Display, 868 MHz Transceiver, RTC (<i>Real Time Clock</i>) und IR-Diode (Quelle: Busware).	223
9.2	Alle Erweiterungs-Anschlüsse des Raspberry Pi auf einen Blick.	224
9.3	So sieht der minimalistische CherryPy-Server auf Midori aus.	228
9.4	LED-Verdrahtung.	230
9.5	Der <i>AD5535</i> ist ein 14-Bit Gleichstrom-DAC.	233
9.6	Das <i>AD5535</i> Timing-Diagramm aus dem Datenblatt.	233
9.7	Der <i>AD5535</i> benötigt drei Leitungen: Data-in, CLK und Chip-enable.	235
9.8	Die Massen aller drei SPI-Leitungen sind am Raspberry Pi zusammengeführt und auf Masse gesteckt. Das Bild zeigt einen Raspberry Pi der ersten Generation. Beim neuen Modell verwenden Sie die gleichen Pins von links oben aus gezählt.	235
9.9	Der vereinfachte SPI-Webserver für den <i>AD5535</i> Chip.	242
9.10	<i>GPIB-Kabel</i> können übereinander gesteckt werden, um so mehrere Geräte zu ver- netzen.	246
9.11	Adafruit hat viele Displays im Programm (Quelle: Adafruit).	247
9.12	Mit <i>fbcp</i> wird das Fernsehprogramm des <i>VDR</i> im LCD angezeigt (Quelle: Busware).	254
9.13	Für den Raspberry Pi gibt es eine eigene Kamera, die einfach am <i>CSI</i> -Adapter angeschlossen wird.	255
9.14	Pi 3 und Pi Zero nutzen unterschiedliche Kabel, um die Kamera zu befestigen (Quelle: Adafruit).	255
10.1	Das Telekom-Haus in Berlin (2005-2006) war ein Paradebeispiel für Hausautoma- tisierung.	261
10.2	(Quelle: Rudolf König).	264
10.3	Die <i>FHEM</i> -Konfigurationsdatei kann komfortabel über ein Web-Interface editiert werden.	266

10.4	Authentifizierung per Telefon.	266
10.5	Die ELV <i>FS-20</i> -Steckdose bietet einen preiswerten Einstieg in die Hausautomatisierungs-Technik für unter 30 € (Quelle: ELV AG).	268
10.6	Die Kurzübersicht des <i>Demo</i> -Raumes erlaubt das Ein- und Ausschalten der Steckdose.	269
10.7	Die Einstellungs-Seite für das <i>FS-20</i> -Gerät lässt keine Wünsche offen.	270
10.8	Das Busware SCC-Interface ist stapelbar. So können verschiedene Protokolle auf einem Raspberry Pi gefahren werden. Das passende Gehäuse gibt es gleich dazu (Quelle: Busware).	271
10.9	433 MHz Sender zum kleinen Preis (Quelle: Watterott).	271
10.10	Der ELV Wassersensor HMS100WD benachrichtigt Sie über ungewollte Feuchtigkeit und kann so Schlimmeres verhindern (Quelle: ELV AG).	277
10.11	<i>linphone</i> ist ein <i>VOIP</i> -Klient für den Raspberry Pi.	282
10.12	<i>linphone</i> bietet einen Wizard, um <i>VOIP</i> -Server hinzuzufügen.	282
11.1	Das kostenlose Office-Paket <i>LibreOffice</i>	287
11.2	Der Aufruf von <i>LibreOffice</i> im Menü.	288
11.3	Das Startmenü von <i>LibreOffice</i> erlaubt die einfache Auswahl der zu startenden Büro-Applikation.	289
11.4	<i>LibreWriter</i> erlaubt das Erstellen von Texten ähnlich wie <i>WORD</i>	289
11.5	<i>LibreWriter</i> kann <i>WORD</i> -Dateien lesen und speichern.	290
11.6	\LaTeX kann sich sehen lassen: Das einfache Wikipedia-Dokument macht einen guten Eindruck (Quelle: http://de.wikipedia.org/wiki/LaTeX).	293
11.7	Mit <i>aspell</i> gibt es auf dem Raspberry Pi die Möglichkeit, die Rechtschreibung einer Datei zu überprüfen.	298
11.8	Das <i>The Legrand Orange Book-Template</i> ist ein professionelles \LaTeX -Template zum Schreiben von Büchern. Quelle: www.latextemplates.com	299
12.1	Mit einem Gamepad macht das Spielen am Raspberry Pi mehr Spaß (Quelle: Logitech).	303
12.2	<i>RetroPie</i> zaubert alte Spiele-Gefühle auf den Raspberry Pi.	304
12.3	<i>Pingus</i> ist ein Lemming-Clone auf dem Raspberry Pi (Quelle: pingus.seul.org).	306
12.4	Das Indie-Spiel <i>Minecraft</i> gibt es seit Februar 2013 für den Raspberry Pi. Im Vergleich zum PC-Spiel ist die <i>Minecraft</i> -Edition nicht ganz so umfangreich (Quelle: http://minecraft-de.gamepedia.com).	307
12.5	Mit <i>Mathematica</i> wirft Wolfram Research ein Schwergewicht in den Raspberry Pi-Ring.	309
12.6	Adafruit bietet alle Teile an, die zum Bau eines WLAN-Radios erforderlich sind. Dazu gehören u. a. ein Display und diverse Taster (Quelle: Adafruit).	311
12.7	<i>Radio-Tray</i> ist ein Web-Radio für den Raspberry Pi. Nach der Installation ruft man das Programm auf und findet es in der oberen Leiste auf der X-Oberfläche.	311



Tabellenverzeichnis

1.1	Übersicht aller aktuellen Raspberry Pi-Modelle (Quelle: datenreise.de).	4
2.1	Die wichtigsten vi-Befehle lernt man schnell.	33
3.1	Ein kleiner Überblick über das LINUX-Dateisystem.	70
10.1	Die wichtigsten Zuordnungen in der <i>FHEM</i> -Nomenklatur.	270
10.2	Überblick über die verschiedenen Hausautomatisierungs-Systeme.	273
12.1	Die wichtigsten LINUX-Befehle.	316

Index

Symbole

L ^A T _E X	290
Bewerbungen	294
Buchvorlage	299
Rechtschreibung	298
433 MHz-Sender	167

A

AirPlay	140
AirPrint	175
Alarmanlage	276
Ambilight	147
2D/3D	173
APA102(C)	148
App, Fernbedienung	162
Für jede HDMI-Quelle	167
Funksteckdose	165
Klebehaken	148
KODI Addon	162
KODI App	162
LibreELEC	147
Netzteil	148
Pegelwandler	151
Software	152
Apache	180
Arduino Nano	148
Audioausgabe	26

B

Baudrate	172
Boot-Probleme	43
Busware CCD	263

C

C/C++	73
CC1101	271
Chillispot	191
Chinchstecker	169
Chromium	13
CMS	214
CSI	224
CUL	262
CULW	262
CVBS	168

D

Dateibrowser	47
Dateisystem	70
Datenbank	175
DHCP	189
DSI	224
DVB-Sticks	86
DynDNS	54

E

E-Mail	47
Editor	
joe	34
vi	33
Embedded Systeme	XIII
Erweiterungen	223
Anschlüsse	224
Externalplayer	143
Externe Festplatte	106

F

Fehlersuche	313
Fernbedienung	93
Anlernen	97
Lirc	94
Fernsehen	85
Fernzugriff	51
FFT	310
FHEM	264
Absicherung	265
Email	279
Funksteckdose	267
https	266
Nomenklatur	270
Sprachsteuerung	274
Timer	274
Vergleich	273
Zeitschaltuhr	274
Firewall	203
Firmware	79
Aktualisieren	272
DVB	92
Framebuffer	250
FS20	267

G

Gamepad	302
GPIB	246
GPIO	228
GPIO-Pins	170
Grabber	168
Crop	173
Screenshot	173
V4L2	173

H

Hausautomatisierung	261
HDMI-Konverter	168
HDMI-Splitter	168
Homematic	267
Hostapd	185
Hotspot	185
Webinterface	201
HyperCon	171

I

I2C	224
Init-Skripte	41
Intertechno	267
IP-Adresse	30
Dynamische	35
Statische	35
iSCSI	63

J

Java	172
JTAG	262

K

Kamera	254
Motion detection	257
Kernel	79
KODI	119
Eigenes Addon	162
Fehler	126
Fernbedienung	129

L

LCD	247
LED	2, 29, 231
LibreELEC	128
LibreOffice	287
LibreWriter	289
linphone	281
LINUX	27, 315
LINUX-Befehle	315

Log-Datei 59

M

Makefile 75
 Manual Page 56
 Mathematica 308
 Messtechnik 246
 Micro-Controller 148
 MP3 144
 MPEG-2 Codec 13
 MySQL 182

N

N64 301
 NAS 48, 219
 NAT 190
 Netzwerk 32
 NTFS 107
 NTSC 170

O

Office-Paket 287
 OMXPlayer 15
 OpenVPN 208
 Klient 212

P

PAL 170
 Partition verkleinern 44
 Patch 83
 PHP 181
 Portweiterleitung 54
 PowerDNS 243
 Zone 244
 PWM 225
 Python 73, 77
 C-Modul 237
 Kamera 259
 Webserver 227

R

Radiosender 225
 Radius-Server 196
 Raspberry Pi 1
 Anschlüsse 8
 Kosten 2
 Modell 3B+ 1
 Netzteil 1
 SD-Karten 2
 Zero 3
 Rcswitch-Pi 271
 Real Time Clock 262
 Remote Desktop 53
 Repository 78
 RetroPie 304
 ROM 305
 RTC 25, 262

S

Samba 61, 219
 SD-Karte 6
 Schreibschutz 47
 Secure Copy 55
 Sender 271
 433 MHz 271
 Seriennummer 17
 Server 175
 Software 301
 SPI 171, 225, 233
 Spiele 306
 Sprachsynthese 279
 SSH 30
 Swap-Datei 76
 System einrichten 8

T

Taktrate 11
 Temperatur 26
 texlive 291
 TOR 205
 Touchscreen 250
 Transceiver
 868 MHz 262

U

UART	225
UNIX	27
Updates	27
USB-Port	
Initialisierung	167

V

VDR	99
Fernbedienung	105
Kanalliste	101
Plugins	109
VDRAdmin-am	108
Verschlüsselung	65
Video Disk Rekorder	85
VNCViewer	31
VOIP	279
Klient	281

Server	281
VPN	54, 56
Cisco	56
PPTP	58

W

Win32 Disk Imager	7
WiringPi	230
WLAN	36
WLAN-Radio	310

X

XMING	51
-------------	----

Z

Zertifikate	193
-------------------	-----