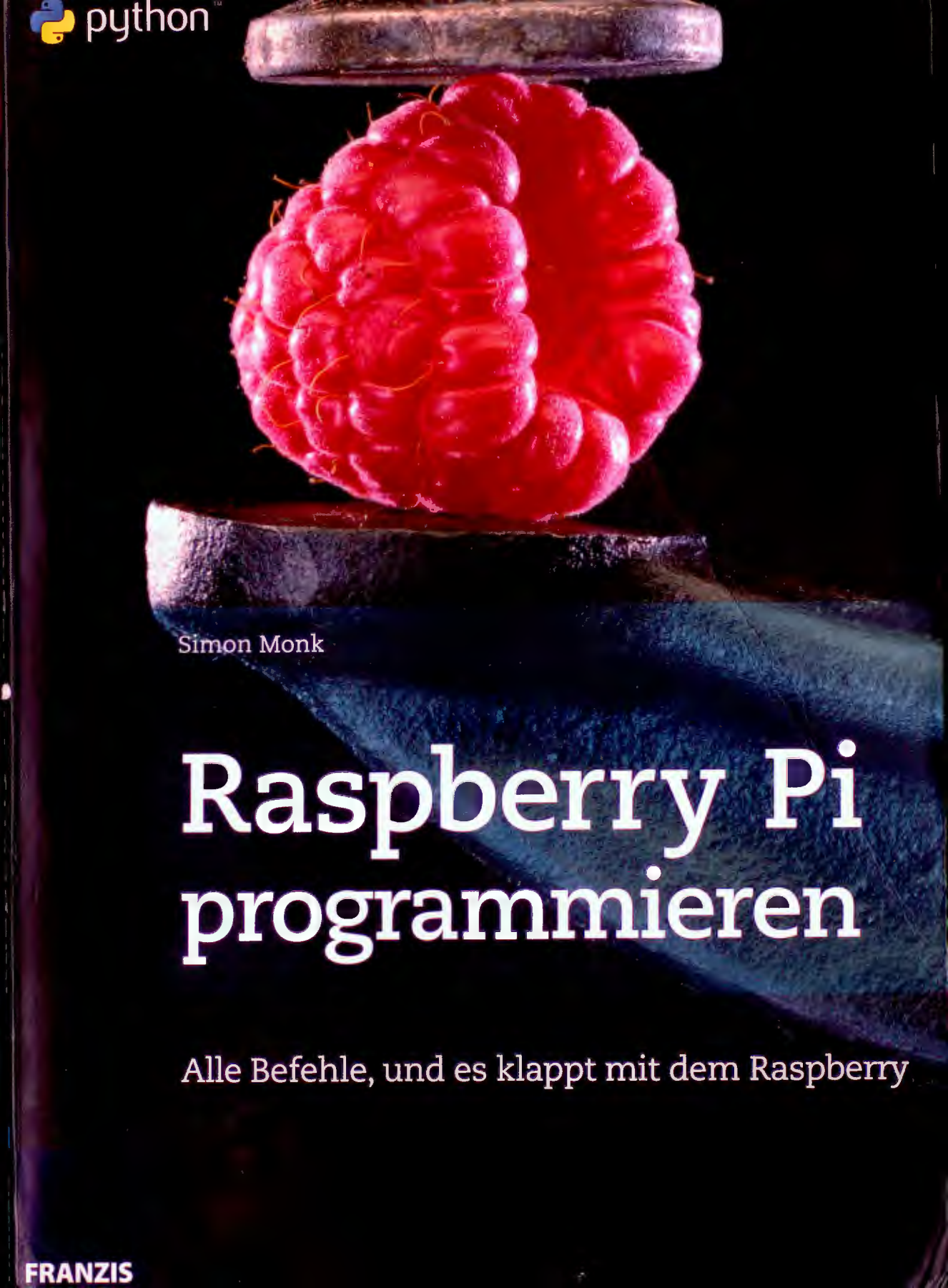
python



Simon Monk

Raspberry Pi programmieren

Alle Befehle, und es klappt mit dem Raspberry

FRANZIS

Simon Monk

Raspberry Pi programmieren

Alle Befehle, und es klappt mit dem Raspberry

Bibliografische Information der Deutschen Bibliothek

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

Titel der englischen Originalausgabe: Programming the Raspberry Pi: Getting Started with Python

Verleger der englischen Originalausgabe: McGraw-Hill Companies, Inc.

Programmleitung: Dr. Markus Stäuble
Übersetzung: G&U Language & Publishing Services GmbH
Satz: DTP-Satz A. Kugge, München
art & design: www.ideehoch2.de
Druck: C.H. Beck, Nördlingen
Printed in Germany

ISBN 978-3-645-60261-7

Über den Autor

Dr. Simon Monk (Preston, UK) hat einen Abschluss in Kybernetik und Informatik und einen Ph.D. in Softwareengineering. Simon hat mehrere Jahre in der Forschung verbracht, bevor er in die Industrie zurückwechselte und als Mitgründer das Unternehmen Momote Ltd. für mobile Software aufbaute. Simon ist Vollzeitautor und hat für die McGraw-Hill-Serie Evil Genius drei Bücher geschrieben. Er ist außerdem Autor von **Programming Arduino** und hat Bücher über IOIO und .NET Gadgeteer veröffentlicht.

Folgen Sie Simon auf Twitter: *@simonmonk2.*

Danksagung

Auch diesmal danke ich Linda für ihre Geduld und Unterstützung.

Außerdem danke ich Andrew Robinson und meinem Sohn, Matthew Monk, für die technische Durchsicht vieler Teile dieses Buchs. Sehen Sie sich auch Andrews Raspberry-Pi-Projektbuch an. Ich bin sicher, es wird Ihnen ebenfalls gefallen.

Ich möchte außerdem Roger Stewart, meinem geduldigen und gründlichen Lektor bei TAB/McGraw-Hill, danken sowie Vastavikta Sharma und Patty Mon für ihr exzellentes Projektmanagement. Es war mir ein großes Vergnügen, mit einem so hervorragenden Team zusammenzuarbeiten.

Inhaltsverzeichnis

Einleitung

Der Raspberry Pi wurde binnen kürzester Zeit zu einem weltweiten Phänomen. Die Menschen sind begeistert, für nur ca. 35 € einen Computer zu bekommen, mit dem sie alles Mögliche anstellen können: vom Einsatz als Desktoprechner oder Mediencenter bis zur Hausautomatisierung.

Dieses Buch erläutert den Raspberry Pi in einfachen Worten und zeigt, wie man Programme für dieses Gerät in der beliebten Programmiersprache Python entwickelt – für Programmierer und Nichtprogrammierer. Es vermittelt Ihnen die Grundlagen zu grafischen Benutzeroberflächen und einfachen Spiele mit dem Modul `pygame`.

Die Programme in diesem Buch verwenden meist Python 3 und in Ausnahmefällen Python 2 (wenn das aufgrund der Modulverfügbarkeit notwendig ist). Die von der Raspberry Pi Foundation empfohlene Distribution Raspbian Wheezy wird in diesem Buch durchgehend verwendet.

Das Buch beginnt mit einer Einführung in den Raspberry Pi, beschreibt das notwendige Zubehör und zeigt, wie Sie Ihren Pi einrichten. Sie erhalten eine Einführung in die Programmierung, während Sie sich Schritt für Schritt durch die folgenden Kapitel bewegen. Programmierkonzepte werden anhand einfacher Anwendungen vorgestellt, mit denen Sie schnell in die Programmierung Ihres Pi einsteigen können.

Drei Kapitel widmen sich der Programmierung des GPIO-Anschlusses des Raspberry Pi, mit dem Sie externe Geräte anschließen können. In diesen Kapiteln behandeln wir zwei Beispielprojekte – eine LED-Uhr und eine Raspberry-Pi-Robotersteuerung, einschließlich Ultraschall-Entfernungsmesser.

Die in diesem Buch behandelten Themen sind:

- Python-Zahlen, Variablen und weitere Grundkonzepte
- Strings, Listen, Dictionaries und andere Python-Datenstrukturen
- Module und Objektorientierung
- Dateien und das Internet
- Grafische Benutzerschnittstellen mit Tkinter
- Spieleprogrammierung mit Pygame
- Hardware über den GPIO-Anschluss steuern
- Einfache Hardwareprojekte

Alle Codebeispiele dieses Buchs gibt es auf der Buchwebsite unter *http://www.buch.cd* zum Download. Dort finden Sie auch nützliche Zusatzmaterialien und eine Errata-Liste.

1 Einführung

Der Verkauf des Raspberry Pi startete Ende Februar 2012 und brachte die Websites der Anbieter durch den riesigen Andrang von Interessenten beinahe zum Absturz. Was war an jenem kleinen Gerät so besonders, dass es dieses große Interesse hervorrief?

1.1 Was ist der Raspberry Pi?

Der Raspberry Pi, gezeigt in Bild 1.1, ist ein Computer, der unter Linux läuft. Er besitzt USB-Anschlüsse für Tastatur und Maus und einen HDMI-Ausgang (High Definition Multimedia Interface) für einen Monitor oder einen Fernseher. Viele Monitore haben nur einen VGA-Anschluss, mit dem der Raspberry Pi nicht funktioniert. Wenn Ihr Monitor jedoch einen DVI-Anschluss hat, können Sie einen günstigen HDMI-zu-DVI-Adapter verwenden.

Wenn der Raspberry Pi hochfährt, sehen Sie die Linux-Oberfläche, wie in Bild 1.2 gezeigt. Es handelt sich tatsächlich um einen richtigen Computer – mit Office-Programmen, Media-Player, Spielen und vielem mehr. Es ist kein Microsoft Windows, sondern sein Rivale, das Open-Source-Betriebssystem Linux (genauer Debian Linux), und die Fensteroberfläche heißt LXDE.

Er ist klein (etwa so groß wie eine Kreditkarte) und sehr günstig (ab 25 € aufwärts). Ein Grund für den geringen Preis ist, dass einige Komponenten nicht mit dem Board mitgeliefert werden, sondern Extras darstellen. So fehlt z. B. ein schützendes Gehäuse – es handelt sich um die nackte Platine. Es wird auch keine Stromversorgung geliefert, sodass Sie ein eigenes 5-V-Micro-USB-Netzteil besorgen müssen, wie es z. B. zum Laden eines Telefons verwendet wird (jedoch mit höherer Leistung). Hierzu dienen häufig USB- oder Micro-USB-Netzteile.

Bild 1.1: Raspberry Pi

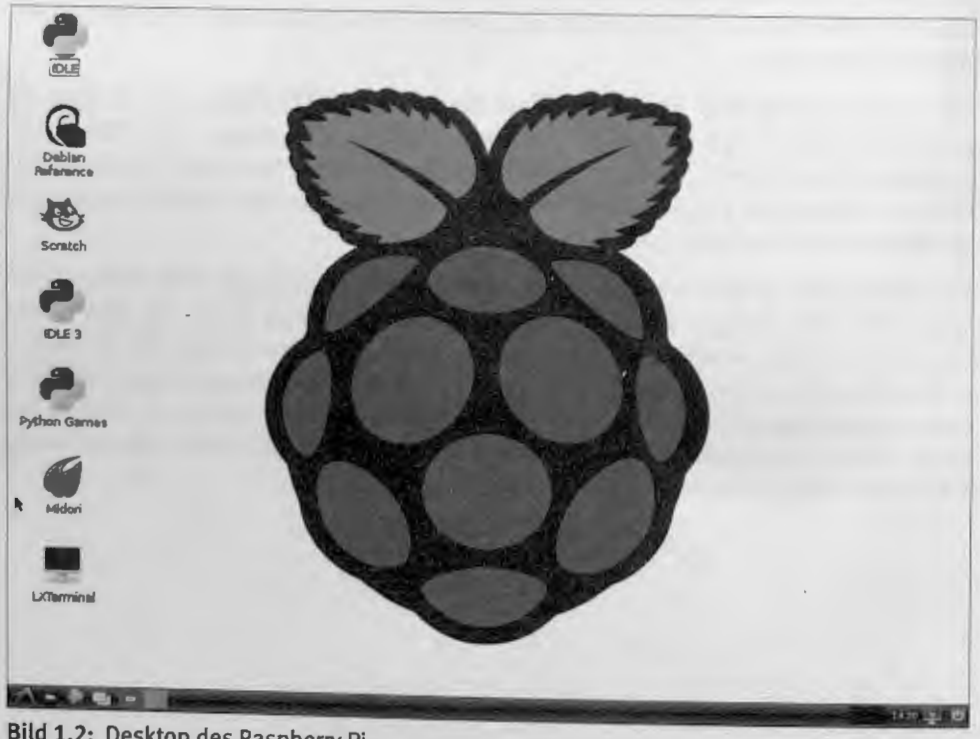


Bild 1.2: Desktop des Raspberry Pi

1.2 Was können Sie mit dem Raspberry Pi anstellen?

Sie können mit dem Raspberry Pi fast alles machen, was Sie auch mit einem Linux-Desktopcomputer tun können. Der Raspberry Pi nutzt statt einer Festplatte eine SD-Karte. Sie können aber auch eine USB-Festplatte anschließen. Sie können Office-Dokumente bearbeiten, im Internet surfen und Spiele spielen (sogar solche mit aufwendiger Grafik, wie beispielsweise `Quake`).

Der günstige Preis des Pi macht ihn außerdem zum idealen Kandidaten für ein Mediencenter. Er kann Video abspielen, und häufig können Sie ihn über den USB-Anschluss vieler Fernseher mit Strom versorgen.

1.3 Der Raspberry Pi im Einzelnen

Bild 1.3 zeigt Ihnen die Bestandteile des Raspberry Pi. Die Abbildung erläutert die verschiedenen Teile des Modells B, das sich von Modell A im Wesentlichen darin unterscheidet, dass es den RJ-45-Netzwerkanschluss zur Verfügung stellt, mit dem Sie ihn mit dem Netzwerk verbinden können.

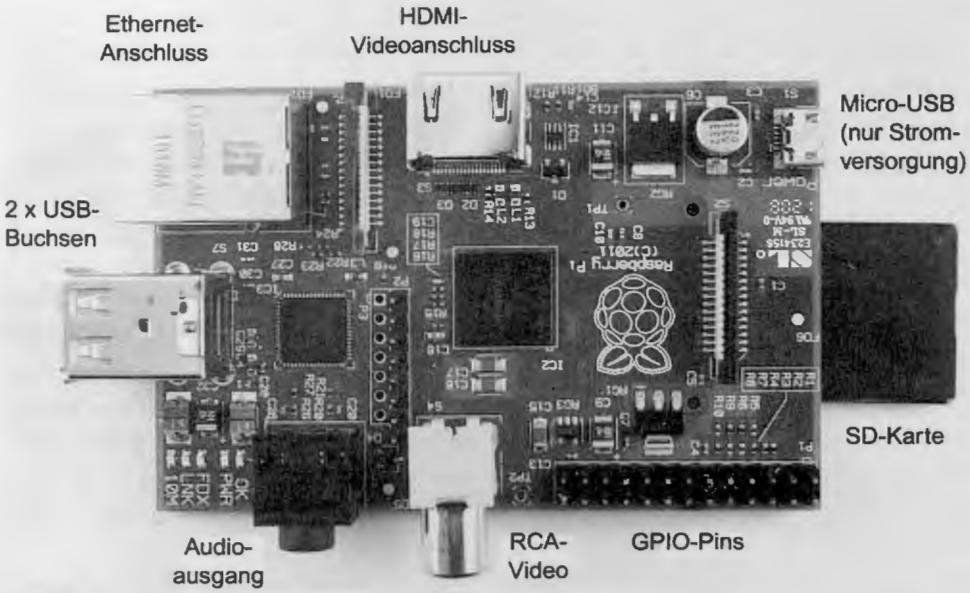


Bild 1.3: Aufbau des Raspberry Pi

Der RJ-45-Ethernet-Anschluss ist oben links im Bild. Wenn sich Ihr Hub in der Nähe befindet, können Sie Ihren Raspberry Pi direkt mit dem lokalen Netz verbinden. Da wir gerade dabei sind: Der Raspberry Pi hat kein integriertes WLAN. Eine drahtlose Netz-

werkverbindung können Sie mit einem USB-Adapter herstellen. Dafür müssen Sie vielleicht auch Treiber installieren.

Direkt unter dem Ethernet-Anschluss finden Sie zwei USB-Anschlüsse, die übereinandergestapelt sind. Sie können eine Tastatur, die Maus oder externe Festplatten an das Board anschließen, doch sind die Anschlüsse schnell belegt. Daher setzen viele Anwender einen USB-Hub ein, der den Anschluss weiterer Geräte ermöglicht.

Unten links im Bild sehen Sie einen Audioanschluss, der ein analoges Stereosignal für Kopfhörer oder Aktivlautsprecher liefert. Der HDMI-Anschluss ist ebenfalls soundfähig.

Neben dem Audioanschluss folgt die RCA-Videoschnittstelle. Diesen Anschluss benötigen Sie fast nie, es sei denn, Sie betreiben den Raspberry Pi an einem alten Fernseher. Vermutlich werden Sie eher den HDMI-Anschluss verwenden, der sich direkt darüber befindet, wie oben in der Abbildung gezeigt. HDMI bietet eine bessere Qualität, Audioübertragung und kann über einen günstigen Adapter auch an DVI-Monitore angeschlossen werden.

Rechts neben dem RCA-Anschluss befinden sich zwei Kontaktleisten. Sie werden GPIO-Pins (General Purpose Input/Output) genannt und ermöglichen es, den Raspberry Pi an andere Schaltungen anzuschließen. Wenn Sie Arduino oder andere Mikrocontroller-Boards kennen, sind Ihnen die GPIO-Pins bereits geläufig. Weiter unten in Kapitel 11 verwenden wir diese Pins, um aus unserem Raspberry Pi das »Gehirn« eines kleinen Roboters zu machen, das die Motoren steuert. In Kapitel 10 verwandeln wir den Raspberry Pi in eine LED-Uhr.

Unter dem Board befindet sich ein SD-Kartenslot. Die SD-Karte muss mindestens 2 GB groß sein. Sie enthält das Betriebssystem des Computers und ein Dateisystem, in dem Sie Ihre Dokumente ablegen können. Die SD-Karte ist beim Kauf des Raspberry Pi als Extra erhältlich. Ihre eigene SD-Karte vorzubereiten ist aufwendig, und Anbieter wie SK Pang, Farnell und RS Components bieten vorbereitete SD-Karten zum Kauf an. Da sich im Raspberry Pi keine Festplatte befindet, stellt die Karte einen wesentlichen Teil Ihres Computers dar. Sie können sie herausnehmen, in einen anderen Raspberry Pi legen, und all Ihre Daten sind vorhanden.

Über der SD-Karte befindet sich ein Micro-USB-Anschluss. Er dient nur der Stromversorgung des Raspberry Pi. Sie benötigen daher ein Netzteil mit einem Micro-USB-Stecker. Dieser Anschluss wird auch von vielen Handys und Android-Smartphones verwendet. Achten Sie darauf, dass der Adapter mindestens 700 mA liefert, sonst verhält sich Ihr Raspberry Pi unvorhersagbar.

Für die technisch Interessierten: Der große, quadratische Chip in der Mitte des Boards ist das Herz des Geräts. Es ist ein Einchipsystem von Broadcom und enthält neben dem Grafik- und einem allgemeinen Prozessor, der den Raspberry Pi steuert, auch 256 MB Speicher.

Vermutlich haben Sie die Anschlüsse neben der SD-Karte und zwischen Ethernet- und HDMI-Anschluss bemerkt. Sie sind für LCD-Displays und eine Kamera gedacht. In der nahen Zukunft werden Kamera- und LCD-Module für den Pi erhältlich sein.

1.4 Ihren Raspberry Pi einrichten

Sie können sich das Leben erheblich vereinfachen, wenn Sie zusammen mit Ihrem Raspberry Pi eine vorbereitete SD-Karte und eine passende Stromversorgung kaufen sowie eine USB-Tastatur und -Maus (wenn Sie nicht beides irgendwo herumliegen haben). Beginnen wir jetzt mit der Einrichtung und was wir dazu alles benötigen.

1.4.1 Die benötigten Teile kaufen

Tabelle 1.1 zeigt, was Sie für ein voll funktionsfähiges Raspberry-Pi-System benötigen. Zur Zeit der Drucklegung dieses Buchs wird der Raspberry Pi über zwei weltweit anbietende Distributoren in Großbritannien verkauft: Farnell (und die dazugehörige US-Gesellschaft Newark) sowie RS Components, die aber nichts mit RadioShack zu tun haben.

Tabelle 1.1: Teile für einen Raspberry-Pi-Bausatz

Komponente	*Bezugsquelle*	*Zusatzinformationen*
Raspberry Pi, Modell A oder B	Farnell (*de.farnell.com*)	
RS Components (*de.rs-online.com*)	Der Unterschied zwischen beiden Modellen besteht in der Ethernet-Schnittstelle von Modell B und der Hauptspeichergröße.	
USB-Netzteil (mit Eurostecker)	Beliebiger Elektronikversender/Computerladen	5-V-USB-Netzteil. Sollte mindestens 700 mA liefern, 1 A ist jedoch besser.
Micro-USB-Kabel	Elektronikversender/Computerladen	
Tastatur und Maus	Elektronikversender/Computerladen	Jede USB-Tastatur funktioniert. Auch drahtlose Tastaturen und Mäuse mit USB-Empfänger sind möglich.
Fernseher/Monitor mit HDMI	Elektronikversender/Computerladen	
HDMI-Kabel	Elektronikversender/Computerladen	
SD-Karte	Elektronikversender/Computerladen	

Komponente	Bezugsquelle	Zusatzinformationen
WLAN-Adapter*	*http://elinux.org/RPi_VerifiedPeripherals#USB_WiFi_Adapters*	Elinux bietet eine aktuelle Liste kompatibler WLAN-Adapter.
USB-Hub*	Elektronikversender/Computerladen	
HDMI-zu-DVI-Adapter*	Elektronikversender/Computerladen	
Ethernet-Patchkabel*	Elektronikversender/Computerladen	
Gehäuse*	Amazon.de	
* Diese Komponenten sind optional.		

1.4.2 Stromversorgung

Bild 1.4 zeigt eine typische USB-Stromversorgung und ein USB-A-auf-Micro-USB-Kabel.

Bild 1.4: USB-Stromversorgung

Sie können das Netzteil eines alten MP3-Players verwenden, solange es 5 V und genügend Strom liefert. Es ist wichtig, das Netzteil nicht zu überlasten, da es heiß werden und Schaden nehmen (oder vielleicht sogar in Flammen aufgehen) kann. Daher sollte das Netzteil mindestens 700 mA Strom liefern. Mit 1 A sind Sie auf der sicheren Seite und können dann auch Geräte versorgen, die am USB-Anschluss des Raspberry Pi angeschlossen sind.

Die Spezifikationen auf dem Netzteil werden Ihnen verraten, ob es genügend Leistung hat. Manchmal wird die Leistung auch in Watt (W) angegeben. In diesem Fall sollten es mindestens 3 W sein. 5 W entsprechen 1 A.

Tastatur und Maus

Der Raspberry Pi funktioniert mit fast jeder USB-Tastatur und -Maus. Sie können auch die meisten drahtlosen USB-Tastaturen und -Mäuse verwenden, wenn diese ihren eigenen Adapter für den USB-Anschluss haben. Besonders im Set ist diese Möglichkeit sinnvoll. So belegen Sie nur einen der USB-Anschlüsse. Auch für Kapitel 10 ist eine drahtlose Tastatur nützlich, wenn wir damit einen Rapsberry Pi-Roboter steuern.

Bildschirm

Der RCA-Anschluss am Raspberry Pi ist etwas befremdlich, denn die meisten Personen werden den moderneren HDMI-Anschluss verwenden. Ein kostengünstiger 22-Zöller ist für den Pi ideal geeignet. Wenn Sie sich jedoch dafür entscheiden, Ihren Fernseher zu verwenden, müssen Sie den Pi einfach nur bei Bedarf daran anschließen.

Wenn Sie einen Monitor mit nur einem VGA-Anschluss haben, können Sie ihn nicht ohne teuren Konverter verwenden. Hat Ihr Monitor aber einen DVI-Anschluss, ist es möglich, einen günstigen Adapter einzusetzen.

SD-Karte

Sie können Ihre eigene SD-Karte für den Raspberry Pi nutzen, aber sie muss mit einem Betriebssystem-Image versehen werden. Das ist aufwendig, sodass Sie besser bedient sind, wenn Sie für einen oder zwei Euro mehr eine vorbereitete SD-Karte kaufen.

Bei Treffen von Raspberry-Pi-Anwendern finden Sie vielleicht jemanden, der Ihnen beim Vorbereiten einer solchen Karte hilft. Die von Farnell und RS Components angebotenen Fertigkarten sind überteuert. Suchen Sie im Internet nach Anbietern (wie SK Pang), die auf vorbereiteten Karten ein aktuelles Betriebssystem anbieten, und das günstiger als im lokalen Handel. Wenn Sie jedoch unbedingt Ihre eigene SD-Karte einrichten möchten, finden Sie die Anweisungen dazu unter *www.raspberrypi.org/downloads*.

Um Ihre eigene SD-Karte einzurichten, benötigen Sie einen Computer mit einem SD-Kartenleser. Der Vorgang unterscheidet sich abhängig davon, ob Ihr Computer unter Windows, Mac OS oder Linux läuft. Es wurden jedoch diverse Tools entwickelt, die den Vorgang so weit wie möglich automatisieren.

Wenn Sie Ihre eigene Karte vorbereiten, folgen Sie den Anweisungen genau. Mit manchen Tools könnten Sie versehentlich eine an Ihren Computer angeschlossene Festplatte formatieren, wenn Sie sie mit der Karte verwechseln. Erfreulicherweise wird die Sache mit jedem neuen Tool einfacher.

Ein großer Vorteil beim Einrichten Ihrer eigenen SD-Karte ist, dass Sie aus einer Reihe von Betriebssystemen auswählen können. In Tabelle 1.2 sehen Sie die bei Drucklegung

dieses Buchs beliebtesten. Auf der Website der Raspberry Pi Foundation finden Sie die neuesten Distributionen.

Es hindert Sie natürlich nichts daran, mehrere SD-Karten zu kaufen, die verschiedenen Distributionen auszuprobieren und Ihren Favoriten zu finden. Wenn Sie jedoch Linux-Anfänger sind, sollten Sie bei der Standarddistribution Wheezy bleiben.

Gehäuse

Der Raspberry Pi wird ohne Gehäuse geliefert. Dadurch ist der Preis niedrig, eine Beschädigung jedoch auch leichter möglich. Daher sollten Sie sich so schnell wie möglich ein Gehäuse zulegen. In Bild 1.5 sehen Sie einige der erhältlichen Fertiggehäuse.

Tabelle 1.2: Linux-Distributionen für den Raspberry Pi

Distribution	*Hinweise*
Raspbian Wheezy	Ist das Standard-Raspberry-Pi-Betriebssystem und wird für alle Beispiele in diesem Buch verwendet. Nutzt den LXDE-Desktop.
Arch Linux ARM	Eine Distribution mehr für Linux-Experten.
QtonPi	Diese Distribution ist für Entwickler von aufwendigen grafischen Schnittstellen mit dem Qt5 Graphics Framework gedacht.
Occidentalis	Eine Distribution von Adafruit auf Basis von Raspbian Wheezy mit Verbesserungen für Hardwarehacker.

Die hier gezeigten Gehäuse stammen von Adafruit (*www.adafruit.com*), SK Pang (*www.skpang.co.uk*) und ModMyPi (*www.modmypi.com*). Der Gehäusetyp hängt davon ab, was Sie mit Ihrem Raspberry Pi machen wollen. Haben Sie die Möglichkeit, einen 3-D-Drucker zu benutzen, können Sie auch eines der Open-Source-Designs verwenden:

- *www.thingiverse.com/thing:23446*
- *www.thingiverse.com/thing:24721*

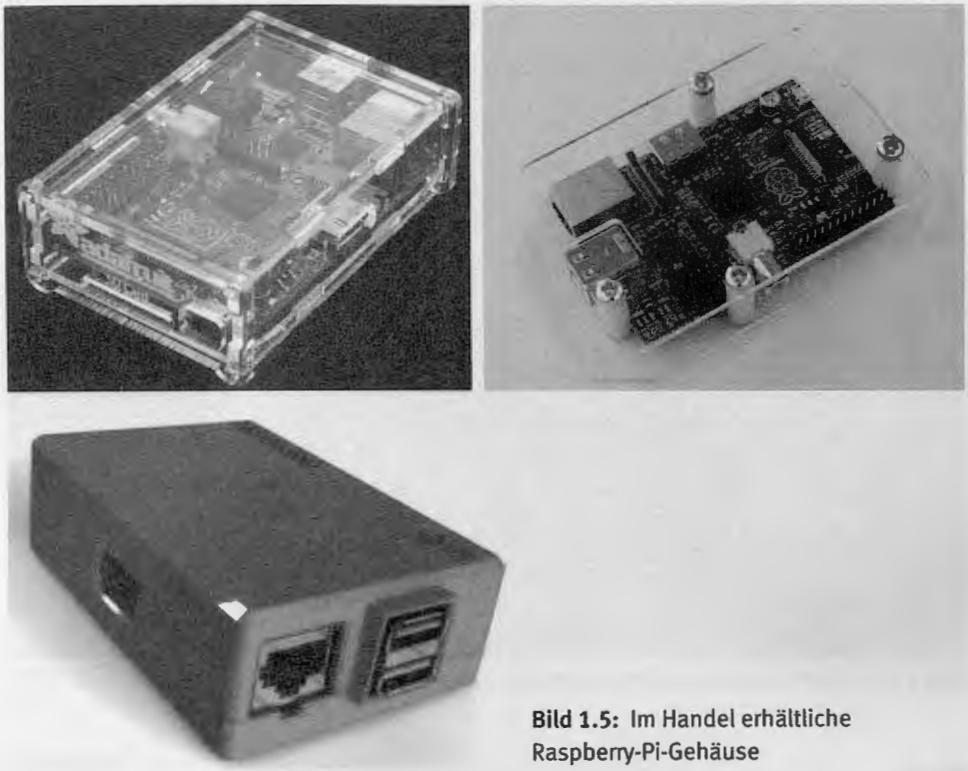


Bild 1.5: Im Handel erhältliche Raspberry-Pi-Gehäuse

Sie finden auf *www.raspberrypi.org/archives/1310* außerdem eine Faltanleitung namens Raspberry Punnet.

Viele Menschen haben Spaß daran, ihren Raspberry Pi in einem vorhandenen Gehäuse unterzubringen, wie einem alten Computer oder Spielkonsolen. Sogar mit Legosteinen lässt sich ein Gehäuse bauen. Mein erstes Raspberry-Pi-Gehäuse bestand aus einem Plastikbehälter für Visitenkarten, in den ich Löcher geschnitten habe (siehe Bild 1.6).

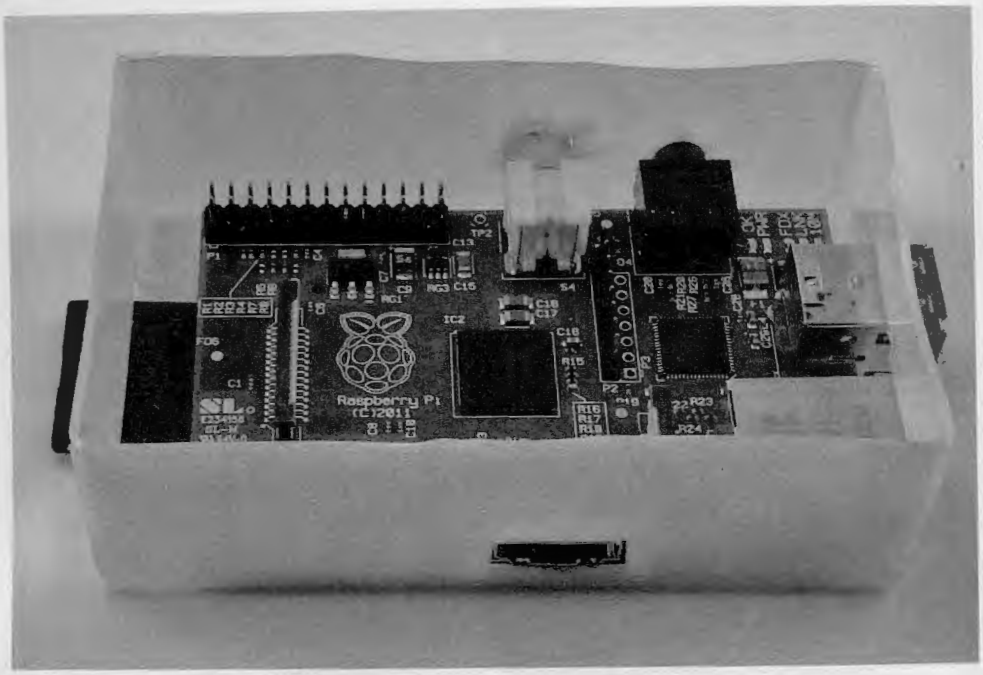


Bild 1.6: Ein selbst gebautes Raspberry-Pi-Gehäuse

WLAN

Keines der Raspberry-Pi-Modelle bietet Unterstützung für WLAN. Um Ihren Raspberry Pi an ein solches Netzwerk anzuschließen, haben Sie nur zwei Möglichkeiten: Zum einen können Sie einen USB-WLAN-Adapter verwenden, den Sie in einen der USB-Anschlüsse stecken (Bild 1.7). Mit etwas Glück erkennt Linux den Adapter und ermöglicht Ihnen den Anschluss sofort (oder zeigt Ihnen, was Sie dafür tun müssen).

Die WLAN-Adapter aus der Liste in Tabelle 1.1 sollten mit dem Raspberry Pi zusammenarbeiten. Manchmal gibt es jedoch Probleme mit WLAN-Treibern, sehen Sie also im Raspberry Pi-Forum und in der Wiki nach, um für das Gerät aktuelle Informationen zu finden.

Die zweite Möglichkeit besteht darin, bei Modell B eine WLAN-Bridge einzusetzen. Diese Geräte werden normalerweise über USB mit Strom versorgt und mit dem Ethernet-Anschluss an den Raspberry Pi angeschlossen. Sie dienen oft Besitzern von Spielkonsolen mit Ethernet-Schnittstelle zum Anschluss der Konsole ans WLAN. Der Vorteil liegt darin, dass der Raspberry Pi dabei keine besondere Konfiguration benötigt.

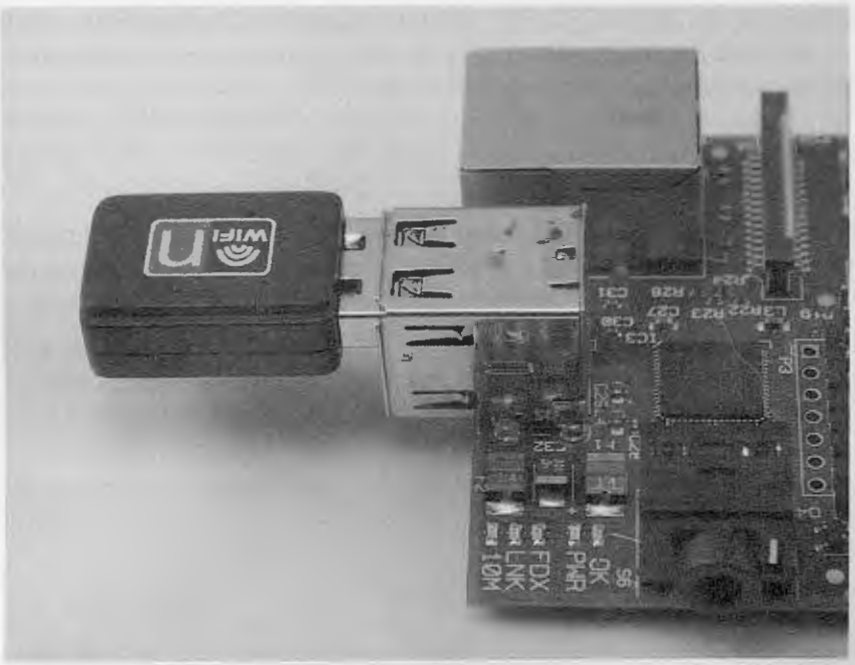


Bild 1.7: Eine WLAN-Karte

USB-Hub

Da der Raspberry nur zwei USB-Anschlüsse besitzt, werden sie Ihnen schnell ausgehen. Abhilfe schafft ein USB-Hub (siehe Bild 1.8).

Bild 1.8: Eine USB-Hub

Diese Hubs sind mit drei oder bis zu acht Anschlüssen erhältlich. Das Hub sollte USB 2 unterstützen. Es ist auch sinnvoll, einen USB-Hub mit Stromversorgung zu verwenden, sodass der Raspberry Pi nicht so viel Strom liefern muss.

1.4.3 Zusammenbau

Wenn Sie alle Teile zusammenhaben, können Sie sie miteinander verbinden und Ihren Raspberry Pi zum ersten Mal starten. Bild 1.9 zeigt, wie Sie die Einzelteile aneinander anschließen müssen.

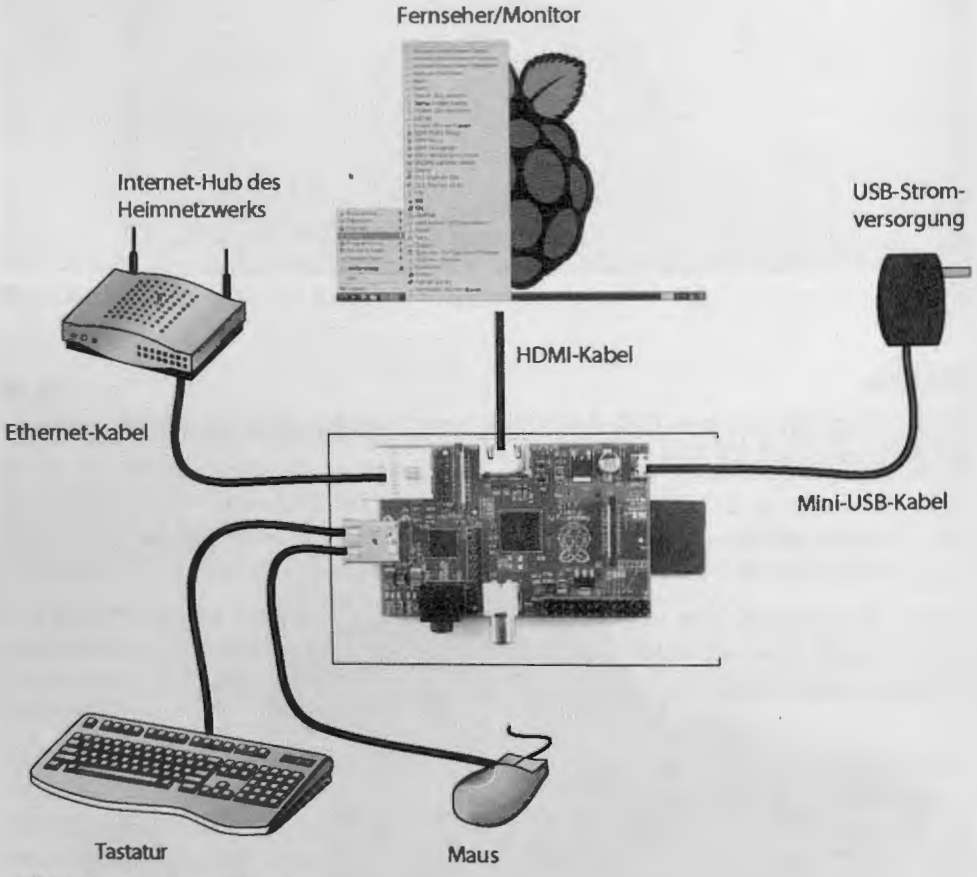


Bild 1.9: Raspberry-Pi-System

Legen Sie die SD-Karte ein, schließen Sie Tastatur, Maus und Monitor an den Pi an, verbinden Sie ihn mit dem Netzteil – und los geht's.

1.5 Der Systemstart

Wenn Sie Ihren Raspberry Pi das erste Mal hochfahren, wird Ihnen nicht die z. B. bei Windows übliche grafische Oberfläche angezeigt. Stattdessen gelangen Sie in eine Erstkonfiguration (siehe Bild 1.10). Es ist sinnvoll, dabei die hier gezeigten Konfigurationsänderungen vorzunehmen.

Wenn Ihre SD-Karte größer als 2 GB ist, nutzt der Raspberry Pi davon nur die ersten 2 GB, wenn Sie nicht die Option `expand_rootfs` zum Erweitern des Dateisystems nutzen. Wählen Sie die Option mit den Pfeiltasten und bestätigen Sie mit Enter.

Eine andere sinnvolle Änderung ist die Option `boot_behaviour`. Ist sie nicht auf »Boot Straight to Desktop« eingestellt, müssen Sie sich einloggen und die Benutzeroberfläche jedes Mal manuell starten (siehe Bild 1.11).

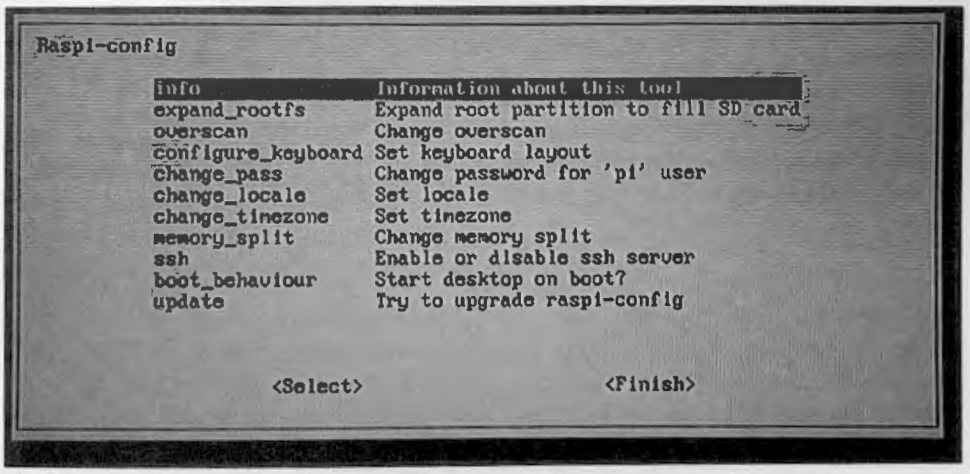


Bild 1.10: Konfigurationsbildschirm

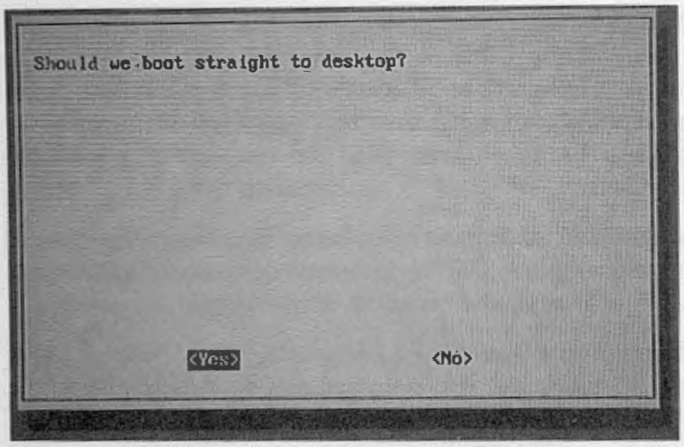


Bild 1.11: Option zum Starten im Desktopmodus

1.6 Zusammenfassung

Nachdem Sie Ihren Raspberry Pi eingerichtet haben und benutzen können, erforschen wir seine Funktionen und lernen einige Linux-Grundlagen.

2 Erste Schritte

Der Raspberry Pi verwendet als Betriebssystem Linux. In diesem Kapitel lernen Sie, wie Sie den Desktop und die Kommandozeile verwenden.

2.1 Linux

Linux ist ein Open-Source-Betriebssystem. Die Software wurde als Gemeinschaftsprojekt für diejenigen entwickelt, die eine Alternative zu den beiden Betriebssystemen Microsoft Windows und Apple OS X wollten. Es ist ein komplettes Betriebssystem auf Basis von UNIX aus den frühen Tagen der Computerei. Es hat loyale und hilfsbereite Anhänger und ist heute leistungsfähig und einfach einzusetzen.

Auch wenn das Betriebssystem Linux genannt wird, gibt es viele unterschiedliche Fassungen, die Distributionen genannt werden. Sie beinhalten das gleiche Grundbetriebssystem, liefern aber unterschiedliche Anwendungen oder Benutzeroberflächen mit. Auch wenn es viele verschiedene Distributionen gibt, wird für den Raspberry Pi die mit dem Namen Raspbian Wheezy empfohlen.

Wenn Sie an Microsoft Windows gewöhnt sind, wird Ihnen das neue Betriebssystem zuerst etwas ungewohnt vorkommen. In Linux funktionieren manche Dinge unterschiedlich. Fast alles unter Linux lässt sich nach Ihren Wünschen anpassen. Das System ist offen und von Ihnen vollständig steuerbar. Wie jedoch schon aus Spiderman bekannt, geht mit großer Freiheit auch große Verantwortung einher. Wenn Sie also nicht vorsichtig sind, können Sie Ihr Betriebssystem beschädigen.

2.2 Der Desktop

Am Ende von Kapitel 1 hatten wir unseren Raspberry Pi gerade gestartet, uns eingeloggt und die Fensteroberfläche gestartet. Bild 2.1 zeigt, wie der Desktop des Raspberry Pi aussieht.

Wenn Sie sonst Windows- oder Mac-Systeme verwenden, kennen Sie bereits das Konzept des Desktops als Ordner innerhalb des Dateisystems, der als Basis für alle Arbeiten am Computer dient.

Auf der linken Seite des Desktops sehen Sie ein paar Icons zum Starten von Anwendungen. Wenn Sie auf das Icon ganz links unten am Bildschirm klicken, werden uns einige der auf dem Raspberry Pi installierten Anwendungen und Tools angezeigt (ähnlich dem

Startmenü unter Windows). Wir beginnen mit dem Datei-Manager, der sich unter den Accessories befindet.

Der Datei-Manager entspricht dem Datei-Explorer unter Windows oder dem Finder auf dem Mac. Er ermöglicht Ihnen, das Dateisystem zu durchstreifen, Dateien zu kopieren und zu verschieben und ausführbare Dateien (Anwendungen) zu starten.

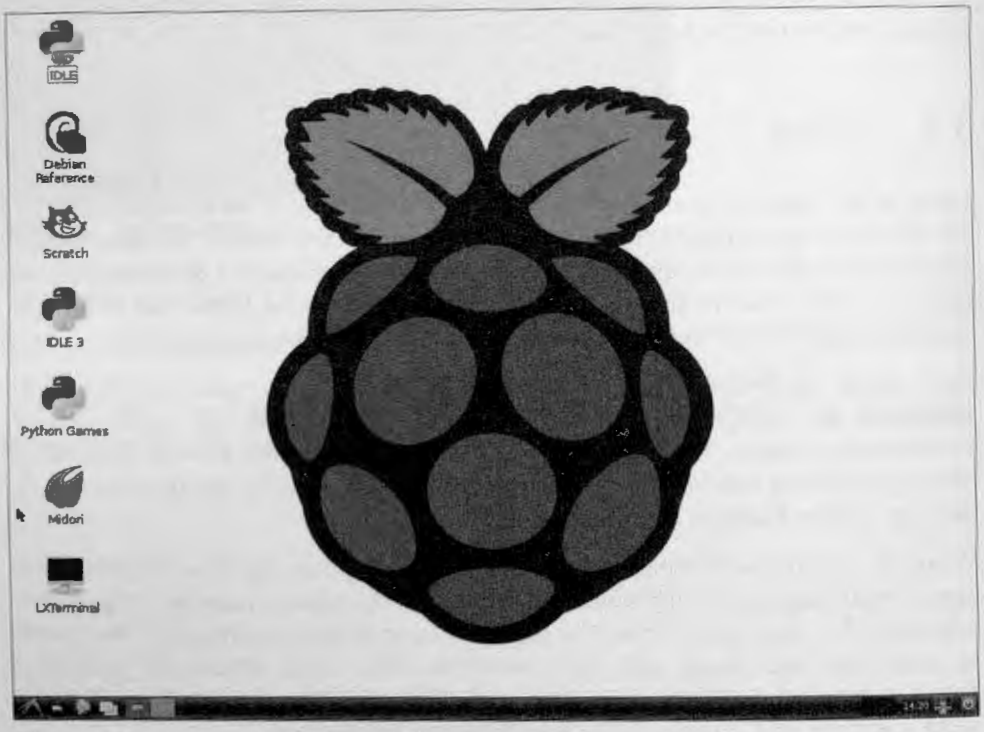


Bild 2.1: Der Desktop des Raspberry Pi

Beim Start zeigt der Datei-Manager den Inhalt Ihres Stammverzeichnisses an. Sie erinnern sich, dass Sie beim Einloggen Ihren Namen als `pi` angegeben haben. Der Pfad zu Ihrem Stammverzeichnis lautet **/home/pi**. Beachten Sie, dass Linux wie Mac OS X Schrägstriche (/) als Trennzeichen für den Pfadnamen verwendet. Daher wird / als Stammverzeichnis bezeichnet, und **/home/** ist ein Verzeichnis, das weitere Verzeichnisse enthält, für jeden Anwender eines. Unser Raspberry Pi hat nur einen Anwender (namens `pi`), sodass dieses Verzeichnis lediglich ein weiteres namens **pi** enthält. Das aktuelle Verzeichnis wird in der Adresszeile oben angezeigt, und Sie können darin Eingaben vornehmen, um das angezeigte Verzeichnis zu verändern, oder die Navigationsleiste an der Seite verwenden. Der Inhalt des Verzeichnisses **/home/pi** besteht nur aus den Verzeichnissen **Desktop** und **python_games**.

Ein Doppelklick auf **Desktop** öffnet das Verzeichnis **Desktop**, das jedoch nicht besonders interessant ist, da es nur die Kürzel links auf dem Desktop enthält. Wenn Sie **python_games** öffnen, sehen Sie einige Spiele, die Sie, wie in Bild 2.2 gezeigt, ausprobieren können.

Bild 2.2: Der Inhalt von **python_games** im Datei-Manager

Sie werden das Dateisystem außerhalb Ihres Stammverzeichnisses nicht oft verwenden müssen. Alle Dokumente, Musikdateien usw. sollten Sie innerhalb von Verzeichnissen in Ihrem Stammverzeichnis ablegen oder auf einem externen USB-Flash-Laufwerk.

2.3 Das Internet

Wenn Sie zu Hause ein Netzwerk haben, an das Sie Internetgeräte mit einem Ethernet-Kabel anschließen können, dürfte es kein Problem sein, mit dem Raspberry Pi online zu gehen. Ihr Netzwerk sollte dem Raspberry Pi automatisch eine IP-Adresse zuweisen und eine Verbindung ins Internet ermöglichen.

Der Raspberry Pi wird mit einem Webbrowser namens Midori geliefert, den Sie im Abschnitt **Internet** in Ihrem Startmenü finden. Sie können prüfen, ob Ihre Internetverbindung funktioniert, indem Sie Midori starten und eine Website Ihrer Wahl öffnen, wie in Bild 2.3 gezeigt.

Bild 2.3: Der Webbrowser Midori

2.4 Die Kommandozeile

Wenn Sie ein Windows- oder Mac-Anwender sind, haben Sie die Kommandozeile möglicherweise noch nicht benutzt. Als Linux-Anwender ist sie Ihnen aber vermutlich schon vertraut. Sind Sie also Linux-Anwender, haben Sie sicher schon bemerkt, dass Sie dieses Kapitel überspringen können, weil Sie den Inhalt bereits kennen.

Obwohl Sie ein Linux-System komplett über die grafische Oberfläche steuern können, müssen Sie manchmal Befehle über die Kommandozeile eingeben. Dies machen Sie z. B. bei der Installation neuer Anwendungen und um den Raspberry Pi zu konfigurieren.

Starten Sie LXTerminal an der Launcher-Schaltfläche, wie in Bild 2.4 gezeigt.

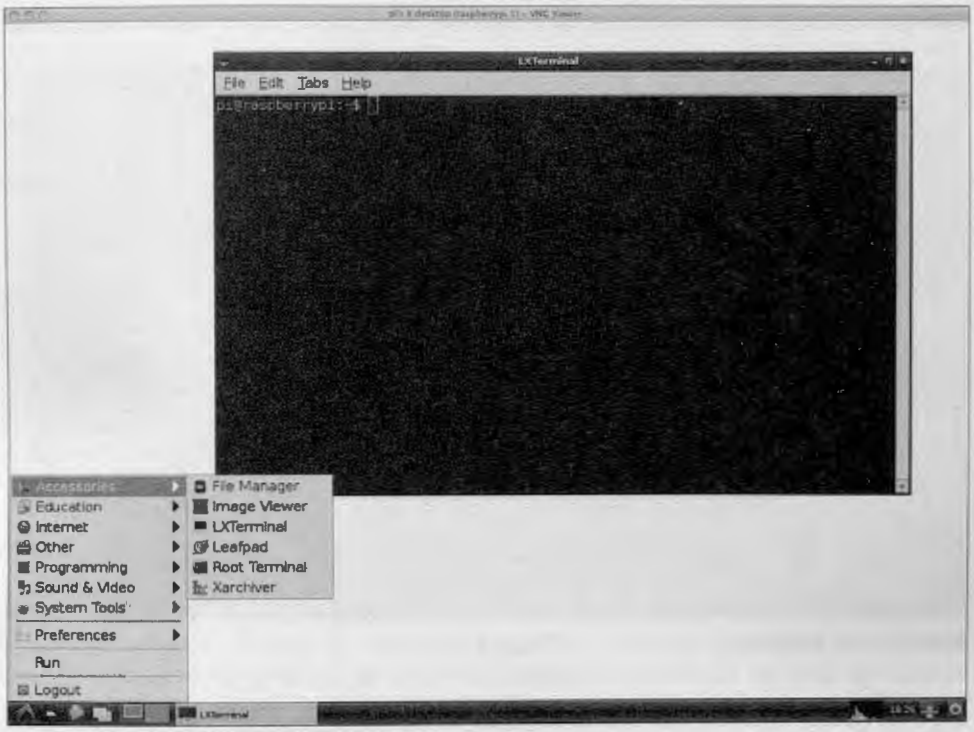


Bild 2.4: Die Kommandozeile LXTerminal

2.4.1 Mit dem Terminal navigieren

Drei Befehle werden für Sie an der Kommandozeile besonders nützlich sein. Der erste Befehl lautet `pwd`, was für **print working directory** steht. Er zeigt Ihnen, in welchem Verzeichnis Sie sich gerade befinden. Wenn also ein `$`-Zeichen im Terminalfenster erscheint, geben Sie `pwd` ein und drücken `Enter`, wie in Bild 2.5 gezeigt.

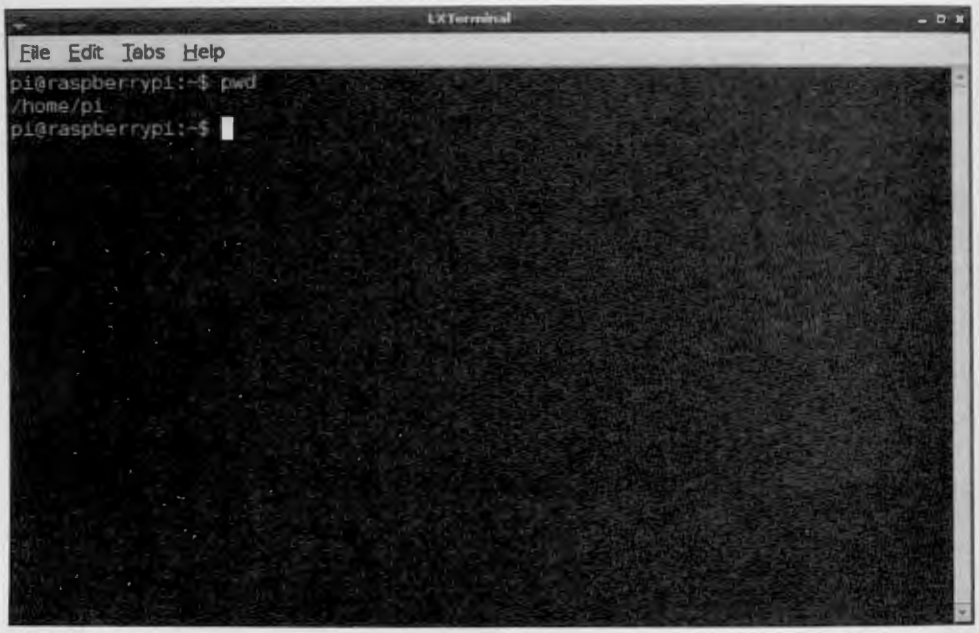


Bild 2.5: Der Befehl `pwd`

Wie Sie sehen, befinden wir uns momentan in **/home/pi**. Statt für jede Eingabe einen Screenshot zu zeigen, werde ich alle Eingaben mit `$` beginnen, so wie hier:

```
$pwd
```

Alles, was Sie als Antwort sehen, beginnt nicht mit einem `$`. Die Ausführung des Befehls `pwd` sieht also wie folgt aus:

```
$pwd
/home/pi
```

Der nächste, häufig genutzte Befehl lautet `ls`, die Abkürzung für **list**, der uns eine Liste der Dateien und Verzeichnisse im aktuellen Verzeichnis ausgibt. Geben Sie Folgendes ein:

```
$ls
Desktop
```

Das sagt uns, dass sich im Verzeichnis **/home/pi** nur das Verzeichnis **Desktop** befindet.

Der letzte für die Navigation notwendige Befehl lautet `cd` (was für **change directory** steht). Dieser Befehl ändert das aktuelle Arbeitsverzeichnis. Er wechselt das Verzeichnis relativ entweder zum aktuellen Arbeitsverzeichnis oder zu einem komplett anderen Ver-

zeichnis, wenn Sie den Pfad mit / beginnen. Der folgende Befehl ändert das aktuelle Arbeitsverzeichnis auf **/home/pi/Desktop**.

```
$pwd
/home/pi
$cd Desktop
```

Sie erreichen dasselbe, indem Sie Folgendes eingeben:

```
cd /home/pi/Desktop
```

Wenn Sie ein Verzeichnis oder einen Dateinamen angeben, müssen Sie den Namen nicht ausschreiben. Stattdessen können Sie jederzeit während Ihrer Eingabe die Tabulatortaste drücken. Ist der Dateiname ab diesem Punkt eindeutig, wird er automatisch für Sie komplettiert.

2.4.2 sudo

Ein weiterer nützlicher Befehl ist `sudo` (für **super user do**). Dadurch werden alle von Ihnen eingegebenen Befehle als Superuser ausgeführt. Vielleicht fragen Sie sich, warum Ihre Befehle nicht automatisch als Superuser ausgeführt werden, da Sie doch der einzige Anwender dieses Systems sind. Der Grund liegt darin, dass Ihr reguläres Konto (Benutzername: `pi`, Passwort: `raspberry`) keine Berechtigungen hat, die es Ihnen z. B. ermöglichen würden, an wichtigen Stellen des Betriebssystems Dateien zu löschen. Um ein solches Chaos zu verhindern, müssen Sie solche Befehle mit dem Präfix `sudo` beginnen. Dadurch wird Unfällen dieser Art ein wenig vorgebeugt.

Für die bisher besprochenen Befehle ist ein solches Präfix jedoch nicht notwendig. Geben Sie jedoch rein interessehalber Folgendes ein:

```
sudo ls
```

Das funktioniert genau wie `ls` allein – Sie sind immer noch im gleichen Arbeitsverzeichnis. Der einzige Unterschied besteht darin, dass Sie beim ersten Aufruf von `sudo` nach Ihrem Passwort gefragt werden.

2.5 Anwendungen

Die Raspbian-Wheezy-Distribution für den Raspberry Pi ist eher schlicht ausgestattet. Es können jedoch Mengen von Anwendungen installiert werden. Für die Installation neuer Anwendungen ist wieder die Kommandozeile erforderlich. Der Befehl `apt-get` wird zum Installieren und Deinstallieren von Anwendungen verwendet. Da die Installation einer Anwendung häufig Superuser-Rechte erfordert, sollten Sie `apt-get`-Befehlen `sudo` voranstellen.

Der Befehl `apt-get` nutzt eine Datenbank mit verfügbaren Paketen, die über das Internet aktualisiert wird, sodass der erste `apt-get`-Befehl, den Sie verwenden sollten, folgender ist:

```
sudo apt-get update
```

Damit wird die Paketdatenbank aktualisiert. Damit er funktioniert, müssen Sie mit dem Internet verbunden sein.

Um ein bestimmtes Paket zu installieren, müssen Sie seinen Namen im Paket-Manager kennen. Um z. B. die Textverarbeitung Abiword zu installieren, brauchen Sie nur Folgendes einzugeben:

```
sudo apt-get install abiword
```

Es dauert etwas, bis alle notwendigen Komponenten heruntergeladen und installiert sind, aber schließlich haben Sie einen neuen Ordner namens **Office** in Ihrem Startmenü, in dem sich die Anwendung Abiword befindet (siehe Bild 2.6).

Sie können sehen, dass das Textdokument in Abiword Teil dieses Kapitels ist. Da ich es gerade schreibe, befindet sich der Text nahe an dieser Stelle des Kapitels. (Ich merke gerade, wie ich mich am Rande einer Endlosrekursion befinde. Gleich verschwinde ich in einer Logik-Wolke.)

Abiword ist eine hervorragende Textverarbeitung. Wäre ich nicht so verliebt in meinen Mac, würde ich das ganze Buch auf meinem Raspberry Pi schreiben.

Da wir gerade über Office-Programme sprechen: Der Kollege von Abiword, der für die Tabellenkalkulation zuständig ist, heißt Gnumeric. Um ihn zu installieren, geben Sie Folgendes ein:

```
sudo apt-get install gnumeric
```

Wenn die Anwendung installiert ist, erscheint eine weitere Option in Ihrem Office-Menü, dieses Mal für Gnumeric.

Um andere Packages zu finden, die Sie vielleicht installieren möchten, suchen Sie im Internet nach Empfehlungen, z. B. im Raspberry-Pi-Forum (*www.raspberrypi.org/phpBB3*). Sie können sich auch die Liste der für Raspbian Wheezy verfügbaren Pakete unter *http://packages.debian.org/stable/* anschauen.

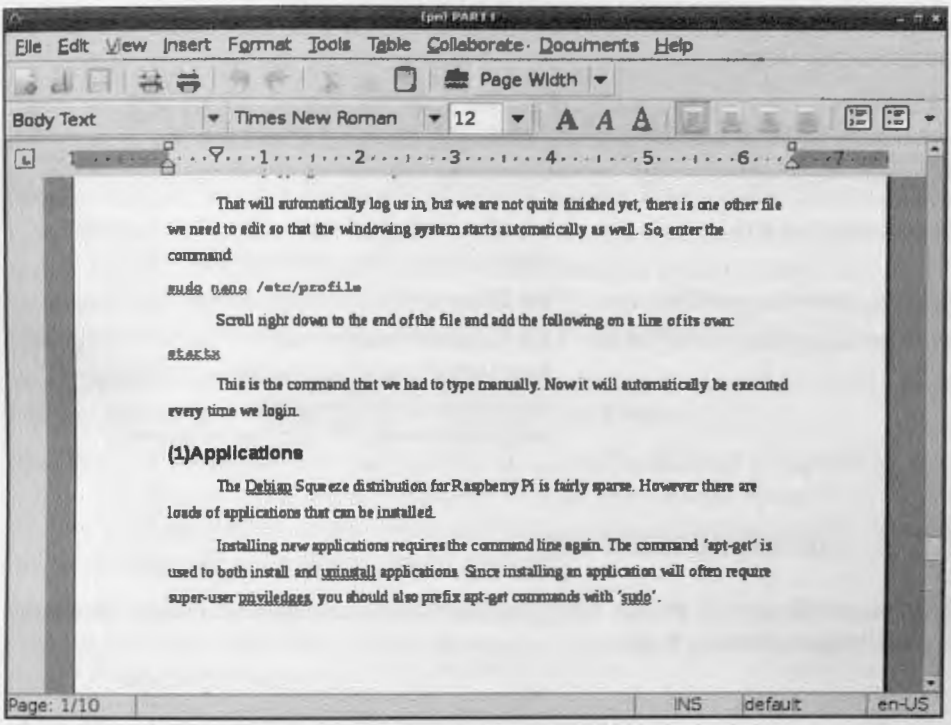


Bild 2.6: Der Bildschirm von Abiword

Nicht alle diese Pakete sind lauffähig, da Raspberry nicht über viel Arbeits- und Massenspeicher verfügt, viele aber laufen dennoch.

Mit dem folgenden Befehl können Sie Pakete entfernen:

```
sudo apt-get remove --auto-remove --purge packagename
```

Dadurch werden das Paket sowie alle davon abhängigen Pakete entfernt, die nicht von einer anderen Anwendung verwendet werden. Behalten Sie die untere rechte Ecke des Datei-Managers im Auge: Dort sehen Sie, wie viel Speicher noch zur Verfügung steht.

2.6 Quellen im Internet

Nachdem Sie den Raspberry Pi programmiert haben, haben Sie nun einen funktionierenden Computer, den Sie näher erforschen möchten. Um Ihnen dabei zu helfen, gibt es viele nützliche Websites, auf denen Sie Tipps und Tricks finden, um das Beste aus Ihrem Raspberry Pi herauszuholen.

Tabelle 2.1 zeigt einige nützliche Seiten für den Raspberry Pi. Mit Ihrer Suchmaschine finden Sie viele weitere.

Tabelle 2.1: Quellen im Internet für den Raspberry Pi

Website	*Beschreibung*
www.raspberrypi.org	Die offizielle Seite der Raspberry Pi Foundation. Sehen Sie sich das Forum und die FAQs an.
www.raspberrypi-spy.co.uk	Ein Blog mit nützlichen How-to-Informationen.
http://elinux.org/RaspberryPiBoard	Die Raspberry Pi-Wiki: viele Informationen über den Raspberry Pi und eine besonders nützliche Liste kompatibler Peripheriegeräte (*http://elinux.org/RPi_VerifiedPeripherals*).

2.7 Zusammenfassung

Nachdem der Raspberry Pi jetzt fertig eingerichtet und lauffähig ist, sollten wir mit der Python-Programmierung beginnen.

3 Python-Grundlagen

Es ist so weit: Entwickeln wir unsere eigenen Programme für den Raspberry Pi. Dazu verwenden wir die Programmiersprache Python. Einer der großen Vorteile von Python ist die leichte Erlernbarkeit, obwohl sie gleichzeitig leistungsfähig genug ist, um spannende Programme zu entwickeln – einfache Spiele sowie auch solche mit Grafik.

Wie bei den meisten Dingen im Leben müssen Sie zuerst gehen lernen, bevor Sie rennen können. Wir beginnen daher mit den Grundlagen von Python.

Eine Programmiersprache ist eine Sprache, in der Programme für Computer geschrieben werden können. Aber warum müssen wir dazu eine spezielle Sprache verwenden? Können wir nicht einfach unsere Muttersprache einsetzen? Wie interpretiert der Computer die Dinge, die wir in dieser Sprache schreiben?

Der Grund dafür, dass wir keine menschliche Sprache nutzen, liegt darin, dass sie vage und unpräzise ist. Computersprachen verwenden englische Begriffe und Symbole, aber in sehr strukturierter Form.

3.1 IDLE

Die beste Möglichkeit, eine neue Sprache zu lernen, besteht darin, sie einfach zu verwenden. Beginnen wir daher mit einem Programm, das uns hilft, Python zu lernen. Dieses Programm heißt IDLE, und Sie finden es im Programmierabschnitt Ihres Startmenüs. Sie finden sogar mehrere Einträge für IDLE. Wählen Sie den mit der Beschriftung IDLE 3. Bild 3.1 zeigt IDLE und die Python-Shell.

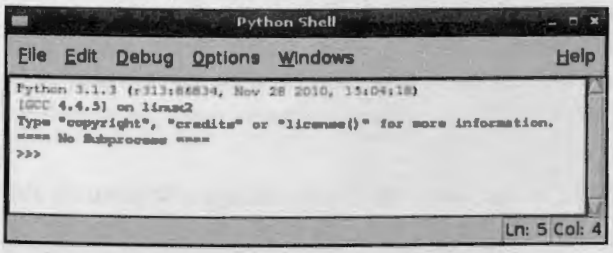


Bild 3.1: IDLE und die Python-Shell

3.1.1 Python-Versionen

Python 3 hatte gegenüber Python 2 größere Änderungen. Dieses Buch basiert auf Python 3.1, aber je weiter Sie mit Python kommen, desto eher stoßen Sie auf Module, die Sie benutzen möchten, die aber nicht für Python 3 verfügbar sind.

3.1.2 Die Python-Shell

In Bild 3.1 sehen Sie die Python-Shell. In dieses Fenster geben Sie Python-Befehle ein und sehen, was passiert. Es ist für kleine Experimente sehr nützlich, besonders wenn Sie Python lernen.

Ähnlich wie an der Kommandozeile können Sie an der Eingabeaufforderung (in diesem Fall >>>) Befehle eingeben, und die Python-Konsole zeigt Ihnen in der darunterliegenden Zeile, was sie gemacht hat.

Arithmetik ist Bestandteil jeder Programmiersprache, und Python ist dabei keine Ausnahme. Geben Sie an der Aufforderung einmal 2+2 ein, und Sie sollten das Ergebnis (4) in der Zeile darunter sehen, wie in Bild 3.2 gezeigt.

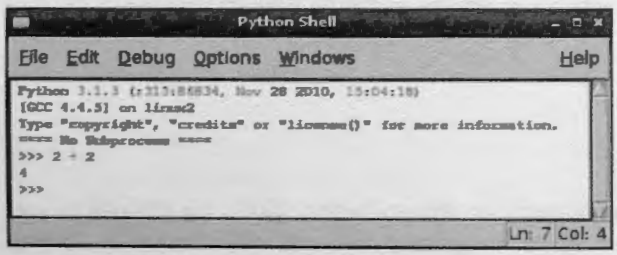


Bild 3.2: Arithmetik in der Python-Shell

3.1.3 Der Editor

Die Python-Shell ist nützlich für Experimente, Sie sollten darin aber keine Programme schreiben. Python-Programme werden in Dateien abgelegt, sodass Sie sie nicht immer neu eingeben müssen. Eine Datei kann eine lange Liste von Befehlen der Programmiersprache enthalten, und wenn Sie diese Befehle ausführen möchten, rufen Sie einfach die Datei auf.

Die Menüzeile über IDLE ermöglicht es uns, eine neue Datei anzulegen. Wählen Sie also **File** und dann **New Window**. Bild 3.3 zeigt den IDLE-Editor in einem neuen Fenster.

Geben Sie die beiden folgenden Zeilen in IDLE ein:

```
print('Hello')
print('World')
```

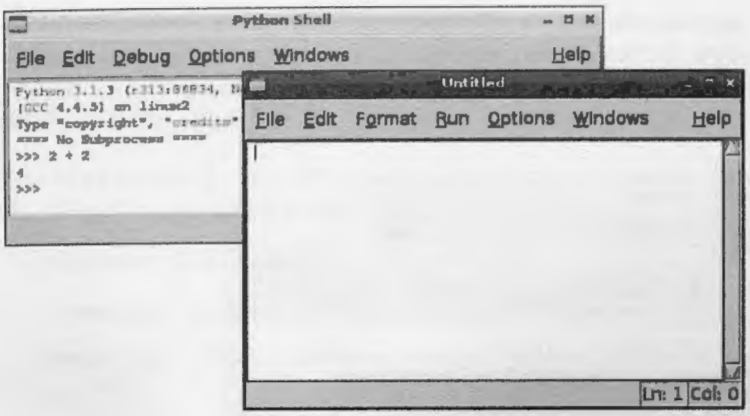


Bild 3.3: Der IDLE-Editor

Sie werden sehen, dass der Editor keine >>>-Aufforderung hat, denn was wir hier eingeben, wird nicht sofort ausgeführt. Stattdessen wird es in einer Datei gespeichert, bis wir diese ausführen. Wenn Sie möchten, können Sie Nano oder einen anderen Texteditor verwenden, um die Datei zu schreiben, aber der IDLE-Editor arbeitet gut mit Python zusammen. Er kennt außerdem die Programmiersprache Python ein wenig und kann als Gedächtnisstütze dienen, wenn Sie Programme eingeben.

Wir benötigen einen sicheren Ort, um die Python-Programme zu speichern, die wir schreiben werden. Öffnen Sie also den Dateibrowser aus dem Startmenü (er befindet sich unter **Accessories**).

Klicken Sie mit rechts auf den Hauptbereich und wählen Sie **New** und dann **Folder** im erscheinenden Pop-up-Menü (siehe Bild 3.4). Geben Sie dem Ordner den Namen **Python** und drücken Sie Enter.

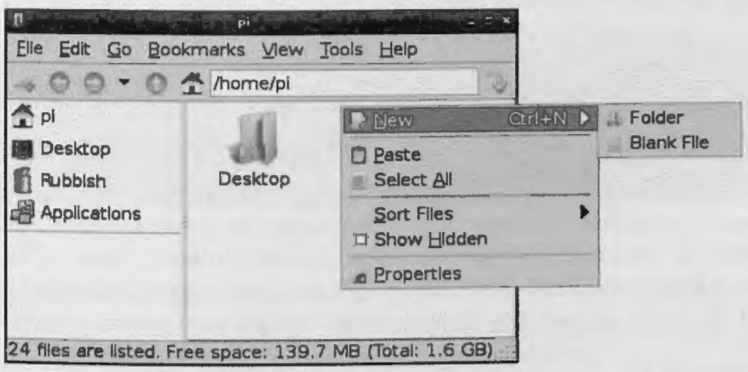


Bild 3.4: Einen Python-Ordner anlegen

Nun müssen wir wieder zum Editorfenster zurückschalten und die Datei über das Menü **File** speichern. Navigieren Sie in das neue Python-Verzeichnis und nennen Sie die Datei wie in Bild 3.5 **hello.py**.

Um das Programm auszuführen und zu sehen, was es macht, gehen Sie ins Menü **Run** und wählen **Run Module**. Sie können in der Python-Shell sehen, was nach dem Ausführen des Programms passiert. Es ist keine große Überraschung, dass das Programm die beiden Wörter `Hello` und `World` jeweils in einer eigenen Zeile ausgibt.

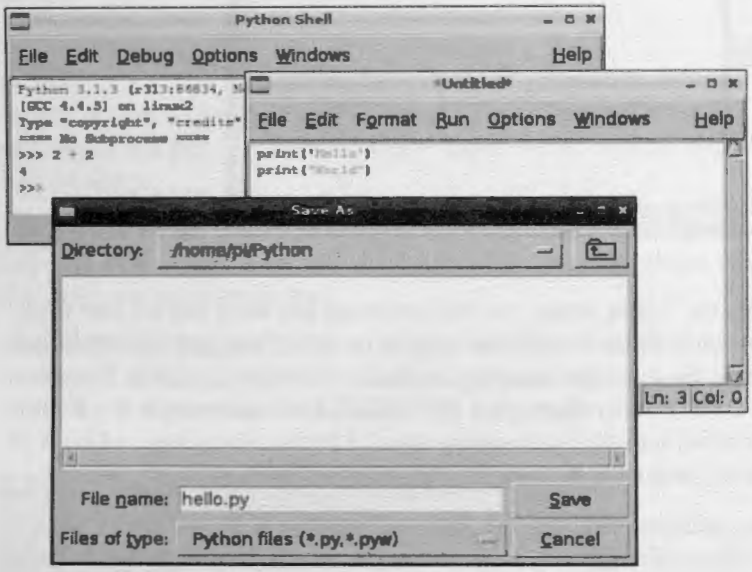


Bild 3.5: Das Programm speichern

Was Sie in der Python-Shell eingeben, wird nicht gespeichert. Wenn Sie IDLE verlassen und später wieder starten, ist alles, was Sie dort eingegeben haben, verschwunden. Da wir unsere Editordatei aber gespeichert haben, können wir sie jederzeit über das File-Menü wieder laden.

Hinweis
Damit dieses Buch keine Ansammlung von Screenshots wird, verwende ich von nun an drei spitze Klammern (>>>), wenn ich etwas in die Python-Shell eingebe. Das Ergebnis erscheint dann in den Zeilen darunter.

3.2 Zahlen

Zahlen sind grundlegend für die Programmierung, und Arithmetik beherrschen Computer sehr gut. Wir beginnen mit einem kleinen Zahlenexperiment, und der beste Ort dazu ist die Python-Shell.

Geben Sie in der Python-Shell Folgendes ein:

```
>>> 20 * 9 / 5 + 32
68.0
```

Das führt zwar nicht weit über das 2+2-Experiment von vorhin hinaus, jedoch zeigt uns das Beispiel einige andere Dinge:

- * bedeutet Multiplikation.
- / bedeutet Division.
- Python führt die Multiplikation vor der Division und die Division vor der Addition aus.

Wenn Sie möchten, können Sie auch Klammern verwenden, um sicherzugehen, dass die Operationen in der richtigen Reihenfolge ausgeführt werden:

```
>>> (20 * 9 / 5) + 32
68.0
```

In diesem Beispiel handelt es sich um ganze Zahlen (oder den Typ **Integer**, wie er von Programmierern genannt wird). Wir können auch Zahlen mit Dezimalpunkt verwenden, wenn wir sie benötigen. In der Programmierung werden diese Zahlen **Float** genannt, was die Kurzform für **floating point** (Fließkommazahl) ist.

3.3 Variablen

Bleiben wir einen Moment bei Zahlen und wenden wir uns den Variablen zu. Sie können sich Variablen als etwas mit einem Wert vorstellen. Sie ähneln Buchstaben als Platzhalter für Zahlen in der Algebra. Versuchen Sie Folgendes:

```
>>> k = 9.0 / 5.0
```

Das Gleichheitszeichen weist einer Variablen einen Wert zu. Die Variable muss auf der linken Seite stehen und aus einem einzelnen Wort bestehen (ohne Leerzeichen). Dieses Wort kann beliebig lang sein und Zahlen sowie Unterstriche (_) enthalten. Es können Klein- oder Großbuchstaben verwendet werden. Das sind die Regeln zur Variablenbenennung. Es gibt jedoch auch Konventionen. Der Unterschied besteht darin, dass eine Regelverletzung von Python moniert wird. Ein Verstoß gegen die Konventionen führt dagegen zu Irritationen bei anderen Programmierern.

Die Konventionen für Variablen bestehen darin, dass sie mit einem Kleinbuchstaben beginnen sollen und als Worttrennzeichen Unterstriche (z. B. `anzahl_der_hennen`)

benutzt werden. Die Beispiele in Tabelle 3.1 geben Ihnen einen Überblick über zulässige und konventionskonforme Benennungen.

Viele andere Sprachen verwenden für Variablennamen andere Konventionen, wie z. B. CamelCase (dt. Binnenmajuskel), um den Beginn einzelner Wörter darin zu kennzeichnen. Ein Beispiel dafür ist `anzahlDerHennen`. Auch diese Schreibweise können Sie im Python-Quellcode finden. Da der Code primär für Sie gedacht ist, spielt die Variablenschreibweise daher eigentlich keine Rolle. Trotzdem ist es sinnvoll, sich an die Konventionen zu halten.

Tabelle 3.1: Variablennamen

Variable	*Zulässig*	*Konvention*
`x`	Ja	Ja
`X`	Ja	Nein
`anzahl_der_hennen`	Ja	Ja
`anzahl der hennen`	Nein	Nein
`anzahlDerHennen`	Ja	Nein
`AnzahlDerHennen`	Ja	Nein
`2beOrNot2b`	Nein	Nein
`toBeOrNot2b`	Ja	Nein

Wenn Sie sich an die Konventionen halten, werden Ihre Programme für andere Programmierer viel besser lesbar.

Sollten Sie etwas tun, das Python nicht möchte oder nicht versteht, erhalten Sie eine Fehlermeldung. Geben Sie Folgendes ein:

```
>>> 2beOrNot2b = 1
SyntaxError: invalid syntax
```

Dies stellt einen Fehler dar, weil Sie versuchen, eine Variable zu definieren, die mit einer Ziffer beginnt, was nicht zulässig ist.

Etwas weiter vorn haben wir der Variablen einen Wert `k` zugewiesen. Wir können ihren Wert auslesen, indem wir einfach `k` eingeben:

```
>>> k
1.8
```

Python hat sich an den Wert von k erinnert, sodass wir es auch in anderen Ausdrücken verwenden können. Wir könnten jetzt also beispielsweise Folgendes formulieren:

```
>>> 20 * k + 32
68.0
```

3.4 for-Schleifen

Arithmetik ist eine tolle Sache, reicht aber nicht für ein tolles Programm aus. Deshalb beschäftigen wir uns in diesem Kapitel mit Schleifen, in denen wir Python anweisen, eine bestimmte Tätigkeit mehrmals und nicht nur einmal auszuführen. Im folgenden Beispiel werden Sie mehr als eine Zeile Python-Code eingeben. Wenn Sie Enter drücken und zur zweiten Zeile gehen, sehen Sie, dass Python wartet. Ihre Eingabe wurde nicht sofort umgesetzt, denn Python weiß, dass Sie noch nicht fertig sind. Das :-Zeichen am Ende der Zeile bedeutet, dass es noch mehr zu tun gibt.

Diese Zusatzaufgaben müssen in die eingezogene Zeile geschrieben werden. Im folgenden Programm drücken Sie daher einmal die Tabulatortaste und geben dann `print(x)` ein. Damit das zweizeilige Programm ausgeführt wird, drücken Sie nach Eingabe der zweiten Zeile zweimal die Taste Enter.

```
>>> for x in range(1, 10):
    print(x)
1
2
3
4
5
6
7
8
9
>>>
```

Das Programm hat die Zahlen zwischen 1 und 9 ausgegeben statt von 1 bis 10. Der Befehl `range` hat einen Endpunkt, der nicht eingeschlossen ist. Der erste Punkt ist jedoch enthalten.

Sie können das überprüfen, indem Sie den `range`-Teil des Programms herausnehmen und sich seine Werte als Liste ausgeben lassen, wie hier:

```
>>> list(range(1, 10))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Die Verwendung der Kommata ist etwas erklärungsbedürftig. Die Klammern enthalten etwas, das wir **Parameter** nennen. In diesem Fall hat `range` zwei Parameter, `from(1)` und `to(10)`, getrennt durch ein Komma.

Der Befehl `for in` besteht aus zwei Teilen. Nach dem Wort `for` muss ein Variablenname folgen. Dieser Variablen wird bei jedem Schleifendurchlauf ein neuer Wert zugewiesen. Beim ersten Mal ist er `1`, beim zweiten `2` usw. Nach dem Wort `in` erwartet Python etwas, das als Objektliste fungiert. In diesem Fall ist es eine Liste der Zahlen zwischen 1 und 9.

Der Befehl `print` nimmt ein Argument und zeigt es in der Python-Shell an. Bei jedem Schleifendurchlauf wird der Wert von `x` ausgegeben.

3.5 Eine Würfelsimulation

Mit unseren bisherigen Kenntnissen über Schleifen können wir ein Programm schreiben, das zehn Würfe mit einem Würfel simuliert.

Um das zu tun, müssen Sie noch wissen, wie Sie eine Zufallszahl generieren. Schauen wir uns das einmal genauer an. Wenn Sie dieses Buch nicht hätten, könnten Sie herausfinden, wie Sie Zufallszahlen erzeugen, indem Sie **Zufallszahlen Python** in Ihre Suchmaschine eingeben und nach Codefragmenten suchen, die Sie in die Python-Shell eingeben können. Da Sie aber dieses Buch haben, geben Sie einfach dies ein:

```
>>> import random
>>> random.randint(1,6)
2
```

Geben Sie die zweite Zeile mehrfach ein, und Sie sehen, dass Sie jedes Mal eine andere Zufallszahl erhalten.

Die erste Zeile importiert eine Bibliothek, die Python verrät, wie es Zufallszahlen generiert. Weiter unten in diesem Buch lernen Sie mehr über Bibliotheken. Im Moment reicht es uns, zu wissen, dass wir diesen Befehl verwenden müssen, bevor wir die Anweisung `randint` geben können, um eine Zufallszahl zu erhalten.

> **Hinweis**
> Ich verwende das Wort Befehl bzw. Anweisung in diesem Buch recht liberal. Genau genommen sind Objekte wie `randint` Funktionen und keine Befehle, aber damit beschäftigen wir uns später.

Nachdem Sie eine einzelne Zufallszahl erzeugen können, müssen Sie dieses Wissen mit dem über Schleifen kombinieren, um nacheinander zehn Zufallszahlen auszugeben. So

umfangreiche Eingaben können wir in der Python-Shell nicht vornehmen, deshalb verwenden wir den IDLE-Editor.

Sie können die Beispiele selbst eingeben oder sich alle Python-Codes dieses Buchs von der Website herunterladen (*www.buch.de*). Jedes Programmierbeispiel hat eine Nummer. Dieses Programm ist in der Datei **3_1_dice.py** enthalten, die in den IDLE-Editor geladen werden kann.

Zu diesem Zeitpunkt ist es sinnvoll, die Beispiele abzutippen, damit sie sich festigen. Öffnen Sie ein neues Fenster in IDLE, geben Sie Folgendes ein und speichern Sie es:

```
#3_1_dice
import random
for x in range(1, 11):
    random_number = random.randint(1, 6)
    print(random_number)
```

Die erste Zeile beginnt mit einem #. Das bedeutet, dass die gesamte Zeile kein Programmcode ist, sondern ein Kommentar für diejenigen, die sich dieses Programm anschauen. Kommentare wie dieser helfen dabei, anderen Informationen über ein Programm zukommen zu lassen, ohne den Programmablauf zu beeinträchtigen. Mit anderen Worten: Python ignoriert alle Zeilen, die mit einem # beginnen.

Wählen Sie im Menü **Run** jetzt **Run Module**. Das Ergebnis sollte wie das in Bild 3.6 aussehen, in dem Sie die Ausgabe der Python-Shell hinter dem Editorfenster sehen können.

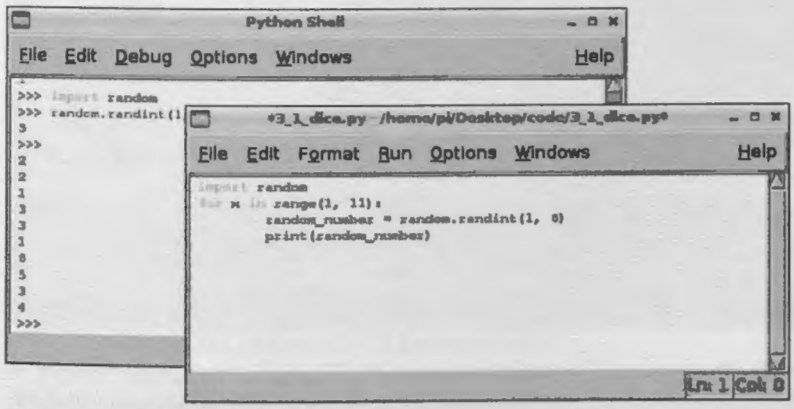


Bild 3.6: Die Würfelsimulation

3.6 if

Jetzt wird es Zeit, das Würfelprogramm so aufzupeppen, dass zwei Würfel geworfen werden, und wenn wir auf eine Punktzahl von 7 oder 11 oder einen Pasch kommen, geben wir nach dem Wurf eine Nachricht aus. Tippen Sie folgendes Programm ein oder laden Sie es in den Editor:

```
#3_2_double_dice
import random
for x in range(1, 11):
    throw_1 = random.randint(1, 6)
    throw_2 = random.randint(1, 6)
    total = throw_1 + throw_2
    print(total)
    if total == 7:
        print('Seven Thrown!')
    if total == 11:
        print('Eleven Thrown!')
    if throw_1 == throw_2:
        print('Double Thrown!')
```

Wenn Sie dieses Programm ausführen, sollten Sie folgende Ausgabe sehen:

```
6
7
Seven Thrown!
9
8
Double Thrown!
4
4
8
10
Double Thrown!
8
8
Double Thrown!
```

Der erste Schritt des Programms besteht darin, zwei Zufallszahlen zwischen 1 und 6 zu erzeugen, für jeden Würfel eine. Der Summe beider Würfe wird eine neue Variable namens `total` zugewiesen.

Jetzt wird es interessant, nämlich in Form des Befehls `if`. Dem Befehl `if` folgt unmittelbar eine Bedingung (im ersten Fall `total == 7`). Dann folgt ein Doppelpunkt (`:`), und die folgenden Zeilen werden nur dann von Python ausgeführt, wenn die Bedingung wahr ist. Auf den ersten Blick mag das wie ein Fehler aussehen, denn wir verwenden ==

statt nur eines =. Das doppelte Gleichheitszeichen wird verwendet, um zwei Objekte miteinander zu vergleichen, während ein einzelnes der Zuweisung von Werten zu einer Variablen dient.

Das zweite `if` ist nicht eingezogen, sodass es unabhängig davon ausgeführt wird, ob das erste `if` wahr ist. Das zweite `if` verhält sich genau wie das erste, nur dass wir hier nach der Summe `11` suchen. Das letzte `if` ist ein wenig anders, da es zwei Variablen vergleicht (`throw_1` und `throw_2`), um zu prüfen, ob sie gleich sind, was auf einen Pasch hindeutet.

Wenn Sie das nächste Mal Monopoly spielen und die Würfel weg sind, wissen Sie, was zu tun ist. Starten Sie Ihren Raspberry Pi und schreiben Sie ein kleines Programm.

3.6.1 Vergleiche

Um zu prüfen, ob zwei Werte gleich sind, verwenden wir ==. Wir nennen == auch Vergleichsoperator. Die Vergleichsoperatoren, die wir benutzen können, werden in Tabelle 3.2 aufgeführt.

Tabelle 3.2: Vergleichsoperatoren

Operator	*Bedeutung*	*Beispiel*
==	Gleich	`total == 11`
!=	Ungleich	`total != 11`
>	Größer als	`total > 10`
<	Kleiner als	`total < 3`
>=	Größer oder gleich	`total >=11`
<=	Kleiner oder gleich	`total <=2`

Sie können mit diesen Vergleichsoperatoren in der Python-Shell ein wenig experimentieren. Hier ist ein Beispiel:

```
>>> k = 10 > 9
True
```

In diesem Fall haben wir Python gefragt: »Ist 10 größer als 9?« Python hat geantwortet: `True`. Jetzt fragen wir Python, ob 10 kleiner ist als 9:

```
>>> 10 < 9
False
```

3.6.2 Die Logik

Logik kann nicht getäuscht werden. Wenn Python uns `True` oder `False` mitteilt, zeigt es uns nicht nur eine Nachricht an. `True` und `False` sind besondere Variablen, die Logikvariablen genannt werden. Jede Bedingung, die wir bei einer `if`-Anweisung verwenden, kann von Python in einen logischen Wert umgewandelt werden, wenn es entscheidet, ob die nächste Zeile ausgeführt werden soll oder nicht.

Diese Logikwerte können wie in arithmetischen Operationen auch miteinander kombiniert werden. Es ist nicht sinnvoll, `True` und `True` zu kombinieren, aber sehr wohl, `True and True` zu schreiben.

Wenn wir zum Beispiel jedes Mal eine Nachricht anzeigen möchten, wenn die Summe unserer Würfel zwischen 5 und 9 liegt, können wir Folgendes schreiben:

```
if total >= 5 and total <= 9: print('not bad')
```

Alternativ können wir auch `or` verwenden. Wir können außerdem `True` mittels `not` in `False` verwandeln und umgekehrt, wie hier gezeigt:

```
>>> not True False
```

Eine andere Möglichkeit, dieselbe Aussage zu treffen, ist:

```
if not (total < 5 or total > 9):
print('not bad')
```

3.6.3 Übung

Versuchen Sie, den vorher beschriebenen Test in das Würfelprogramm zu integrieren. Fügen Sie dabei zwei weitere `if`-Anweisungen hinzu: eine, die »Good Throw!« ausgibt, wenn der Wurf höher als 10 ist, und eine, die »Unlucky!« ausgibt, wenn der Wurf weniger als 4 ergibt. Probieren Sie Ihr Programm aus. Wenn Sie Schwierigkeiten dabei haben, können Sie sich die Lösung in der Datei **3_3_double_dice_solution.py** ansehen.

3.6.4 else

Im vorherigen Beispiel haben Sie gesehen, dass einige der möglichen Würfe von mehr als einer Nachricht begleitet werden. Jede der `if`-Zeilen kann eine Nachricht produzieren, wenn die Bedingung wahr ist. Manchmal benötigen Sie eine etwas andere Logik: Wenn die Bedingung erfüllt ist, soll eine Sache geschehen, wenn nicht, eine andere.

In Python erreichen Sie das mit `else`:

```
>>> a = 7
>>> if a > 7:
    print('a is big')
else:
    print('a is small')
a is small
>>>
```

In diesem Fall wird nur eine der beiden Nachrichten ausgegeben.

Eine weitere Variation ist `elif`, die Kurzform für `else if`: Wir können unser Beispiel so erweitern, dass es drei sich ausschließende Klauseln wie hier gibt:

```
>>> a = 7
>>> if a > 9:
    print('a is very big')
elif a > 7:
    print('a is fairly big')
else:
    print('a is small')
a is small
>>>
```

3.7 while

Ein anderer Befehl für Schleifen lautet `while` und funktioniert etwas anders als `for`. Der Befehl `while` ähnelt dem Befehl `if`, da die Bedingung genau danach angegeben wird. In diesem Fall handelt es sich um die Bedingung, um in der Schleife zu verbleiben. Mit anderen Worten: Der Code innerhalb der Schleife wird so lange ausgeführt, bis die Bedingung nicht mehr wahr ist. Das bedeutet, dass Sie dafür sorgen müssen, dass die Bedingung irgendwann falsch wird. Ansonsten läuft die Schleife immer weiter, und Ihr Programm verhält sich wie bei einem Absturz.

Um die Verwendung von `while` zu veranschaulichen, habe ich das Würfelprogramm so verändert, dass es nur so lange würfelt, bis eine 6 kommt:

```
#3_4_double_dice_while
import random
throw_1 = random.randint(1, 6)
throw_2 = random.randint(1, 6)
```

```
while not (throw_1 == 6 and throw_2 == 6):
    total = throw_1 + throw_2
    print(total)
    throw_1 = random.randint(1, 6)
    throw_2 = random.randint(1, 6)
print('Double Six thrown!')
```

Dieses Programm funktioniert. Probieren Sie es aus. Es ist allerdings etwas länger als notwendig. Wir müssen die folgenden Zeilen zwei Mal wiederholen: einmal vor Schleifenbeginn und einmal innerhalb der Schleife:

```
throw_1 = random.randint(1, 6)
throw_2 = random.randint(1, 6)
```

Eine bekannter Grundsatz bei der Programmierung lautet DRY (Don't Repeat Yourself – wiederholen Sie sich nicht). Auch wenn es bei einem so kleinen Programm nicht so wichtig ist, sollten Sie sich wiederholenden Code vermeiden, weil Programme dadurch deutlich komplexer werden können. Wird identischer Code in mehreren Programmteilen verwendet, ist er schwieriger zu pflegen.

Mit dem Befehl `break` können wir den Code verkürzen und damit den Grundsatz befolgen. Stößt Python auf den Befehl `break`, unterbricht es die Schleife. Hier zeige ich das Programm erneut, diesmal mit dem Befehl `break`:

```
#3_5_double_dice_while_break
import random
while True:
    throw_1 = random.randint(1, 6)
    throw_2 = random.randint(1, 6)
    total = throw_1 + throw_2
    print(total)
    if throw_1 == 6 and throw_2 == 6:
        break
print('Double Six thrown!')
```

Die Bedingung, in der Schleife zu bleiben, ist permanent auf wahr gesetzt. Die Schleife läuft weiter, bis der Befehl `break` erscheint, der aber nur nach Werfen zweier Sechsen kommt.

3.8 Zusammenfassung

Sie sollten sich nun mit IDLE vertraut fühlen und auch mit den Experimenten an der Python-Shell. Ich empfehle Ihnen dringend, einige der Beispiele dieses Kapitels zu modifizieren, um herauszufinden, wie das die Funktion der Programme ändert.

Im nächsten Kapitel gehen wir über Zahlen hinaus und sehen uns einige andere Datentypen an, die Sie in Python verwenden können.

4 Strings, Listen und Dictionaries

Der Name dieses Kapitels hätte auch noch »und Funktionen« enthalten können, aber er war auch so schon lang genug. In diesem Kapitel experimentieren Sie zuerst mit den verschiedenen Möglichkeiten, Daten darzustellen und Ihren Python-Programmen mehr Struktur zu verleihen. Sie fügen dann das Gelernte zusammen und schreiben das Spiel »Hangman«, bei dem Sie ein zufällig ausgewähltes Wort anhand von Buchstabenvorschlägen erraten müssen.

Das Kapitel endet mit einem Referenzabschnitt, der Ihnen alles über die nützlichsten integrierten Funktionen für Berechnungen, Strings, Listen und Dictionaries verrät.

4.1 String-Theorie

Nein, ich meine hier nicht die bekannte Physiktheorie. In der Programmierung stellt ein String eine Folge von Zeichen dar, die Sie in Ihren Programmen verwenden können. Um in Python eine Variable zu erzeugen, die einen String enthält, verwenden Sie einfach den Zuweisungsoperator = und weisen statt einer Zahl eine Reihe von Zeichen in einfachen Anführungszeichen zu:

```
>>> book_name = 'Programming Raspberry Pi'
```

Wenn Sie den Inhalt einer Variablen sehen möchten, geben Sie entweder den Variablennamen in die Python-Shell ein oder verwenden den Befehl `print`, so wie wir es bei Zahlenvariablen getan haben:

```
>>> book_name
'Programming Raspberry Pi'
>>> print(book_name)
Programming Raspberry Pi
```

Es gibt einen kleinen Unterschied zwischen beiden Methoden. Wenn Sie nur den Variablennamen eingeben, setzt Python einfache Anführungszeichen darum, um anzuzeigen, dass es sich um einen String handelt. Verwenden Sie jedoch `print`, gibt Python den Wert aus.

Hinweis
Sie können auch doppelte Anführungszeichen verwenden, die Konvention besteht aber in einfachen, es sei denn, Sie haben einen besonderen Grund (z. B. wenn der zu erzeugende String ein Apostroph enthält).

Sie erfahren, wie viele Zeichen ein String enthält, indem Sie Folgendes eingeben:

```
>>> len(book_name)
24
```

So finden Sie die Position eines bestimmten Zeichens im String:

```
>>> book_name[1]
'r'
```

Dabei gibt zwei Dinge zu beachten: Erstens ist die Verwendung von eckigen Klammern gebräuchlicher als die runder, und zweitens beginnen die Positionen bei 0 und nicht bei 1. Um den ersten Buchstaben im String zu finden, müssen Sie Folgendes tun:

```
>>> book_name[0]
'P'
```

Wenn Sie eine Zahl eingeben, die zu groß für die Länge des Strings ist, erscheint folgende Ausgabe:

```
>>> book_name[100]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>>
```

Es handelt sich dabei um einen Fehler, und Python teilt uns auf diese Weise mit, dass wir etwas falsch gemacht haben. Genauer: Der Teil `string index out of range` teilt uns mit, dass wir auf etwas zugreifen wollen, was es nicht gibt. In diesem Fall ist es Element `100` in einem String, der nur 24 Zeichen lang ist.

Sie können einen großen String auch in kleinere zerlegen, so wie hier:

```
>>> book_name[0:11]
'Programming'
```

Die erste Zahl in den Klammern gibt die Startposition des Strings an, den wir extrahieren wollen. Die zweite ist nicht, wie Sie vielleicht gedacht haben, die Position des letzten Zeichens, sondern die des letzten Zeichens +1.

Versuchen Sie einfach einmal, das Wort `raspberry` aus dem Titel zu extrahieren. Geben Sie die zweite Zahl nicht an, wird standardmäßig das Ende des Strings genommen.

```
>>> book_name[12]
'Raspberry Pi'
```

Geben Sie die erste Zahl nicht an, ist der Standardwert 0.

Schließlich können Sie mit dem Operator + zwei Strings miteinander verbinden. Hier ist ein Beispiel:

```
book_name + ' by Simon Monk'
'Programming Raspberry Pi by Simon Monk'
```

4.2 Listen

Weiter oben in diesem Buch haben wir mit Zahlen experimentiert, wobei eine Variable nur eine Zahl enthalten konnte. Manchmal ist es jedoch nützlich, in einer Variablen eine Liste von Zahlen oder Strings ablegen zu können oder eine Mischung – oder sogar Listen mit Listen. In Bild 4.1 sehen Sie, was passiert, wenn eine Variable zu einer Liste wird.

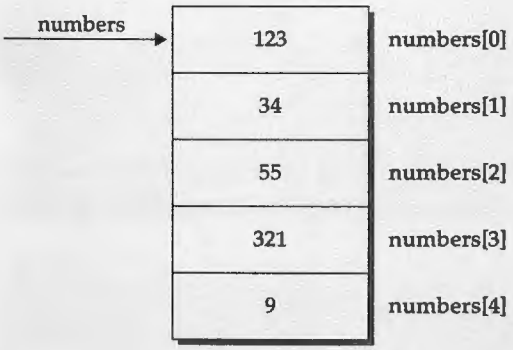


Bild 4.1: Ein Array

Listen verhalten sich ähnlich wie Strings. Letztlich ist ein String auch nur eine Liste mit Zeichen. Im folgenden Beispiel sehen Sie, wie Sie eine Liste erzeugen. `len` funktioniert bei Listen genauso wie bei Strings.

```
>>> numbers = [123, 34, 55, 321, 9]
>>> len(numbers)
5
```

Eckige Klammern dienen der Kennzeichnung von Listen, und wie bei Strings können wir mit eckigen Klammern individuelle Elemente einer Liste finden oder kleinere Listen aus größeren erzeugen:

```
>>> numbers[0]
123
>>> numbers[1:3]
[34, 55]
```

Zusätzlich können wir mit = einem Element in der Liste einen neuen Wert zuweisen, so wie hier:

```
>>> numbers[0] = 1
>>> numbers
[1, 34, 55, 321, 9]
```

Dadurch ändern wir das erste Element der Liste (Element `0`) von 123 in 1. Wie bei Strings können Sie mit dem Operator + Listen zusammenfügen.

```
>>> more_numbers = [5, 66, 44]
>>> numbers + more_numbers
[1, 34, 55, 321, 9, 5, 66, 44]
```

Wenn Sie die Liste sortieren möchten, geht das so:

```
>>> numbers.sort()
>>> numbers
[1, 9, 34, 55, 321]
```

Um ein Element aus einer Liste zu entfernen, verwenden Sie den Befehl `pop`, wie hier gezeigt. Wenn Sie `pop` kein Argument mitgeben, wird nur das letzte Element der Liste entfernt und übergeben.

```
>>> numbers
[1, 9, 34, 55, 321]
>>> numbers.pop()
321
>>> numbers
[1, 9, 34, 55]
```

Übergeben Sie `pop` eine Zahl als Argument, wird sie als Position des zu entfernenden Elements interpretiert. Hier ist ein Beispiel:

```
>>> numbers
[1, 9, 34, 55]
>>> numbers.pop(1)
```

```
9
>>> numbers
[1, 34, 55]
```

Ebenso wie Sie Elemente aus einer Liste entfernen können, können Sie auch welche an bestimmten Positionen einfügen. Die Funktion `insert` verwendet zwei Argumente: Das erste ist die Position, vor der eingefügt werden soll, und das zweite das einzufügende Objekt.

```
>>> numbers
[1, 34, 55]
>>> numbers.insert(1, 66)
>>> numbers
[1, 66, 34, 55]
```

Wenn Sie herausfinden wollen, wie lang eine Liste ist, verwenden Sie `len(numbers)`, aber wenn Sie die Liste sortieren oder ein Element daraus entfernen wollen, setzen Sie hinter die Listenvariable einen Punkt und geben den Befehl so wie hier:

```
numbers.sort()
```

Diese beiden unterschiedlichen Schreibweisen sind der **Objektorientierung** geschuldet, die wir im nächsten Kapitel erläutern.

Listen können in komplexe Strukturen überführt werden, die andere Listen und eine Mischung aus verschiedenen Typen wie Zahlen, Strings und Logikvariablen enthalten. Bild 4.2 zeigt die Listenstruktur, die sich aus folgender Codezeile ergibt:

```
>>> big_list = [123, 'hello', ['inner list', 2, True]]
>>> big_list
 123, 'hello', ['inner list', 2, True]]
```

Sie können Ihr Wissen über Listen mit `for`-Schleifen kombinieren und ein kurzes Programm schreiben, das eine Liste erzeugt und jedes Element darin in einer eigenen Zeile ausgibt.

```
#4_1_list_and_for
list = [1, 'one', 2, True] for item in list:
print(item)
```

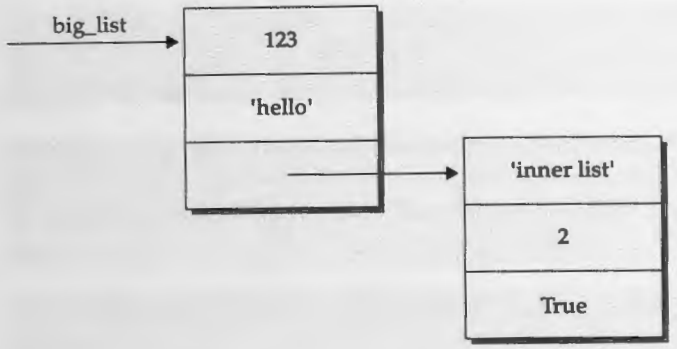


Bild 4.2:
Eine komplexe Liste

Hier folgt die Ausgabe des Programms:

```
1
one
2

True
```

4.3 Funktionen

Wenn Sie kleine Programme schreiben wie diejenigen, die wir bereits erstellt haben, führen sie nur eine Funktion aus. Wir sollten sie ein wenig aufbohren. Was wir erreichen möchten, ist eigentlich recht einfach. Sobald Programme größer werden, werden sie auch komplizierter, sodass es notwendig wird, sie in sogenannte Funktionen zu unterteilen. Wenn wir weiter in die Programmierung einsteigen, werden wir bessere Möglichkeiten finden, sie mittels Klassen und Modulen zu strukturieren.

Viele der von mir als Befehle bezeichneten Konstrukte sind eigentlich Funktionen, die in Python integriert sind. Beispiele dafür sind `range` und `print`.

Die größte Schwierigkeit bei der Softwareentwicklung besteht in der Verwaltung ihrer Komplexität. Die besten Programmierer schreiben Software, die einfach lesbar und zu verstehen ist und mit wenigen Erklärungen auskommt. Funktionen sind der Schlüssel für einfache, verständliche Programme, die ohne das Risiko, dass am Ende nichts mehr geht, verändert werden können.

Eine Funktion ist fast wie ein Programm in einem Programm. Wir können sie verwenden, um eine Befehlssequenz einzuwickeln. Eine von uns definierte Funktion kann von überall in unserem Programm aufgerufen werden und enthält ihre eigenen Variablen und ihre eigene Befehlsliste. Wenn die Befehle ausgeführt worden sind, gelangen wir wieder an die Stelle im Code zurück, von der aus wir die Funktion aufgerufen haben.

Lassen Sie uns als Beispiel eine Funktion erstellen, die einen String als Argument übernimmt und am Ende das Wort »please« anfügt. Laden Sie die folgende Datei – oder geben Sie sie besser ins Editorfenster ein –, lassen Sie sie laufen und sehen Sie, was passiert:

```
#4_2_polite_function
def make_polite(sentence):
    polite_sentence = sentence + ' please'
    return polite_sentence
print(make_polite('Pass the salt'))
```

Die Funktion beginnt mit dem Schlüsselwort `def`. Es folgt der Name der Funktion, der denselben Benennungskonventionen folgt wie Variablen. Danach kommen die Parameter in Klammern und getrennt durch Kommata, wenn es mehrere sind. Die erste Zeile muss mit einem Doppelpunkt enden.

In der Funktion verwenden wir eine neue Variable `polite_sentence`, die den der Funktion übergebenen Parameter nimmt und `please` daran anfügt (natürlich mit führendem Leerzeichen). Diese Variable kann nur innerhalb der Funktion verwendet werden.

Die letzte Zeile der Funktion ist ein Rückkehrbefehl. Damit wird festgelegt, welchen Wert die Funktion an den aufrufenden Code zurückgibt. Sie ähnelt trigonometrischen Funktionen wie Sinus, denen Sie einen Winkel übergeben und eine Zahl zurückerhalten. In diesem Fall wird der Wert in der Variablen `polite_sentence` zurückgegeben.

Um die Funktion zu verwenden, geben wir einfach ihren Namen an und übergeben ihr die passenden Argumente. Ein Rückgabewert ist nicht notwendig, und manche Funktionen machen noch weniger, als einen Wert zurückzugeben. Zum Beispiel können wir eine wenig nützliche Funktion schreiben, die »Hello« eine bestimmte Anzahl von Malen ausgibt:

```
#4_3_hello_n
def say_hello(n):
    for x in range(0, n):
        print('Hello')
say_hello(5)
```

Dies sind die Grundlagen, die wir für unser Spiel Hangman benötigen. Auch wenn Sie noch weitere Dinge lernen müssen, kommen wir hierauf zurück.

4.4 Hangman

Hangman ist ein Wortratespiel, das normalerweise mit Stift und Papier gespielt wird. Ein Spieler wählt ein Wort, zeichnet für jeden Buchstaben darin einen Strich, und der andere muss das Wort erraten. Es werden jeweils einzelne Buchstaben geraten. Wenn

der geratene Buchstabe nicht im Wort enthalten ist, verliert der Spieler ein Leben, und ein Teil des Galgens wird gezeichnet. Ist der Buchstabe im Wort enthalten, werden alle Stellen auf die entsprechenden Striche geschrieben.

Wir lassen Python sich ein Wort ausdenken, das wir dann raten müssen. Statt einen Galgen zu zeichnen, wird Python uns sagen, wie viele Leben noch übrig sind.

Wir beginnen damit, für Python eine Wortliste anzulegen. Das ist eine Aufgabe für eine Liste von Strings:

```
words = ['chicken', 'dog', 'cat', 'mouse', 'frog']
```

Nun muss das Programm eines der Wörter zufällig auswählen. Wir können eine Funktion dafür schreiben und sie selbst testen:

```
#4_4_hangman_words
import random

words = ['chicken', 'dog', 'cat', 'mouse', 'frog']

def pick_a_word():
   word_position = random.randint(0, len(words) - 1)
   return words[word_position]
print(pick_a_word())
```

Lassen Sie das Programm ein paarmal laufen und prüfen Sie, ob es verschiedene Wörter aus der Liste wählt.

Dies ist ein guter Anfang, er muss aber in die Struktur des Spiels passen. Nun definieren wir eine Variable namens `lives_remaining`. Es handelt sich um einen Integer, der bei 14 beginnt und jedes Mal um 1 verringert wird, wenn falsch geraten wurde. Dieser Variablentyp wird globale Variable genannt, denn anders als bei in Funktionen definierten Variablen können wir von überall im Programm darauf zugreifen.

Neben der neuen Variablen schreiben wir auch eine Funktion namens `play`, die das Spiel steuert. `play` verwendet die globale Variable `lives_remaining`, die mit `livesRemaining = 14` initialisiert wird. Wir wissen, was `play` machen sollte, nur die Details kennen wir noch nicht alle. Deshalb schreiben wir einfach die Funktion `play` und erstellen dann Funktionen, die sie aufrufen, wie `get_guess` und `process_guess`, so wie schon bei der Funktion `pick_a_word`. Hier ist sie:

```
def play():
   word = pick_a_word()
   while True:
      guess = get_guess(word)
      if process_guess(guess, word):
```

```
        print('You win! Well done!')
        break
    if lives_remaining == 0:
        print('You are hung!')
        print('The word was: ' + word)
        break
```

Hangman beginnt mit der Auswahl eines Worts. Dann folgt eine Schleife, die fortgesetzt wird, bis das Wort erraten wurde (`process_guess` ergibt `True`) oder `lives_remaining` null enthält. Bei jedem Schleifendurchlauf lassen wir den Anwender raten.

Noch können wir das Programm nicht laufen lassen, denn die Funktionen `get_guess` und `process_guess` fehlen noch. Wir können jedoch Platzhalterfunktionen für sie schreiben, damit wir die Funktion `play` zumindest testen können. Platzhalter sind einfache Versionen von Funktionen, die nur wenig tun. Sie werden später durch die vollwertige Funktion ersetzt.

```
def get_guess(word):
    return 'a'
def process_guess(guess, word):
    global lives_remaining
    lives_remaining = lives_remaining - 1
    return False
```

Der Platzhalter für `get_guess` simuliert einen Spieler, der immer »a« rät, und der für `process_guess` nimmt immer an, dass sich der Spieler geirrt hat, verringert daher den Lebenszähler um 1 und gibt `False` zurück, das Zeichen dafür, dass der Spieler nicht gewonnen hat.

Der Platzhalter für `process_guess` ist etwas komplizierter. Die erste Zeile teilt Python mit, dass die verbleibenden Leben eine globale Variable sind.

Ohne diese Zeile nimmt Python an, dass es sich um eine neue lokale Variable einer Funktion handelt. Der Platzhalter verringert die verbleibenden Leben um 1 und gibt `False` zurück, um anzuzeigen, dass der Anwender nicht gewonnen hat. Schließlich bauen wir Prüfungen ein, um zu sehen, ob der Spieler alle Buchstaben im Wort erraten hat.

Öffnen Sie die Datei **4_5_hangman_play.py** und starten Sie sie. Sie erhalten eine Ausgabe wie diese:

```
You are hung!
The word was: dog
```

Was passierte, ist, dass wir durch alle 14 Rateversuche gefallen sind und Python uns am Ende verraten hat, welches das gesuchte Wort war (und wir verloren haben).

Um das Programm fertigzustellen, müssen wir nur noch die Platzhalter mit Leben füllen, was wir mit `get_guess` beginnen:

```
def get_guess(word):
    print_word_with_blanks(word)
    print('Lives Remaining: ' + str(lives_remaining))
    guess = input(' Guess a letter or whole word?')
    return guess
```

Zuerst teilt `get_guess` dem Spieler den aktuellen Spielstand mit der Funktion `print_word` mit (wie etwa `c--c--cn`). Fürs Erste ist das ein weiterer Platzhalter. Dann erfährt der Spieler, wie viele Leben er noch übrig hat. Da wir dem String `Lives Remaining:` eine Zahl anfügen wollen (`lives_remaining`), muss sie mit der Funktion `str` in einen String konvertiert werden.

Diese integrierte Funktion `input` gibt ihren Parameter als Eingabeaufforderung aus und liefert zurück, was der Anwender eingibt. In Python 2 wurde diese Funktion `raw_input` genannt. Wenn Sie also entscheiden, Python 2 zu verwenden, ändern Sie den Namen in `raw_input`.

Schließlich gibt die Funktion `get_guess` zurück, was der Spieler eingegeben hat.

Der Platzhalter `print_word_with_blanks` erinnert uns einfach daran, dass wir später noch etwas programmieren müssen.

```
def print_word_with_blanks(word):   print('print_word_with_blanks:not done
yet')
```

Öffnen Sie die Datei **4_6_hangman_get.py** und starten Sie sie. Sie erhalten eine Ausgabe wie diese:

```
not done yet
Lives Remaining: 14
 Guess a letter or whole word?x
not done yet
Lives Remaining: 13
 Guess a letter or whole word?y
not done yet
Lives Remaining: 12
 Guess a letter or whole word?
```

Raten Sie, bis alle Ihre Leben verbraucht sind, um zu prüfen, ob die Meldung, dass Sie verloren haben, angezeigt wird.

Nun erstellen wir die richtige Version von `print_word_with_blanks`. Diese Funktion muss etwas anzeigen wie `c--c--n`, sie muss also wissen, welche Buchstaben der Spieler geraten hat und welche nicht. Um das zu tun, nutzt sie eine neue globale Variable

(dieses Mal einen String), der alle geratenen Buchstaben enthält. Jeder richtig geratene Buchstabe wird an den String angefügt.

```
guessed_letters = ''
```

Das ist die eigentliche Funktion:

```
def print_word_with_blanks(word):
   display_word = ''
   for letter in word:
      if guessed_letters.find(letter) > -1:
         # letter found
         display_word = display_word + letter
      else:
         # letter not found
         display_word = display_word + '-'
   print display_word
```

Diese Funktion beginnt mit einem leeren String und durchläuft jeden Buchstaben des Worts. Wenn ein Buchstabe bereits vom Spieler geraten wurde, wird er an `display_word` angefügt, ansonsten ein Strich (`-`). Die eingebaute Funktion `find` wird verwendet, um zu prüfen, ob sich ein Buchstabe unter den geratenen befindet. Die Funktion `find` gibt `-1` zurück, wenn der Buchstabe nicht vorhanden ist, ansonsten seine Position. Was uns eigentlich interessiert, ist, ob der Buchstabe vorhanden ist oder nicht, daher prüfen wir nur, ob das Ergebnis `-1` ist. Dann wird das Wort ausgegeben.

Wenn `process_guess` aufgerufen wird, passiert nichts mit dem Geratenen, weil die Funktion noch ein Platzhalter ist. Wir können sie ein wenig umbauen, sodass sie schon einmal den geratenen Buchstaben an `guessed_letters` anfügt:

```
def process_guess(guess, word):
   global lives_remaining
   global guessed_letters
   lives_remaining = lives_remaining - 1
   guessed_letters = guessed_letters + guess
   return False
```

Öffnen Sie die Datei **4_7_hangman_print_word.py** und starten Sie sie. Sie erhalten eine Ausgabe wie diese:

```
Lives Remaining: 14
 Guess a letter or whole word?c
c--c---
Lives Remaining: 13
 Guess a letter or whole word?h
```

```
ch-c---
Lives Remaining: 12
 Guess a letter or whole word?
```

Nun sieht das Ganze schon eher nach einem richtigen Spiel aus. Allerdings müssen wir noch den Platzhalter für `process_guess` füllen. Das tun wir jetzt:

```
def process_guess(guess, word):
   if len(guess) > 1:
      return whole_word_guess(guess, word)
   else:
      return single_letter_guess(guess, word)
```

Wenn der Spieler rät, haben wir zwei Möglichkeiten: Er kann entweder einen einzelnen Buchstaben raten oder das ganze Wort. Mit dieser Methode entscheiden wir, um welche Art es sich handelt, und rufen entweder `whole_word_guess` oder `single_letter_guess` auf. Da beide Funktionen recht einfach sind, implementieren wir sie direkt ohne Platzhalter.

```
def single_letter_guess(guess, word):
   global guessed_letters
   global lives_remaining
   if word.find(guess) == -1:
      # word guess was incorrect
      lives_remaining = lives_remaining - 1
   guessed_letters = guessed_letters + guess
   if all_letters_guessed(word):
      return True

def all_letters_guessed(word):
   for letter in word:
      if guessed_letters.find(letter) == -1:
         return False
   return True
```

Die Funktion `whole_word_guess` ist einfacher als `single_letter_guess`:

```
def whole_word_guess(guess, word):
   global lives_remaining
   if guess.lower() == word.lower():
      return True
    else:
      lives_remaining = lives_remaining - 1
      return False
```

Wir müssen das geratene nur mit dem tatsächlichen Wort vergleichen und sehen, ob sie nach einer Konvertierung in Kleinbuchstaben gleich sind. Sind sie verschieden, kostet das ein Leben. Die Funktion gibt `True` zurück, wenn der Tipp richtig war, ansonsten `False`.

Das ist schon alles. Öffnen Sie **4_8_hangman_full.py** im Editor und starten Sie die Datei. Hier folgt noch einmal das komplette Listing:

```
#04_08_hangman_full
import random

words = ['chicken', 'dog', 'cat', 'mouse', 'frog']
lives_remaining = 14
guessed_letters = ''

def play():
   word = pick_a_word()
   while True:
      guess = get_guess(word)
      if process_guess(guess, word):
         print('You win! Well done!')
         break
      if lives_remaining == 0:
         print('You are hung!')
         print('The word was: ' + word)
         break

def pick_a_word():
   word_position = random.randint(0, len(words) - 1)
   return words[word_position]

def get_guess(word):
   print_word_with_blanks(word)
   print('Lives Remaining: ' + str(lives_remaining))
   guess = input(' Guess a letter or whole word?')
   return guess

def print_word_with_blanks(word):
   display_word = ''
   for letter in word:
      if guessed_letters.find(letter) > -1:
         # letter found
         display_word = display_word + letter
      else:
         # letter not found
         display_word = display_word + '-'
```

```
    print(display_word)

def process_guess(guess, word):
    if len(guess) > 1:
        return whole_word_guess(guess, word)
    else:
        return single_letter_guess(guess, word)

def whole_word_guess(guess, word):
    global lives_remaining
    if guess == word:
        return True
    else:
        lives_remaining = lives_remaining - 1
        return False

def single_letter_guess(guess, word):
    global guessed_letters
    global lives_remaining
    if word.find(guess) == -1:
        # letter guess was incorrect
        lives_remaining = lives_remaining - 1
    guessed_letters = guessed_letters + guess
    if all_letters_guessed(word):
        return True
    return False

def all_letters_guessed(word):
    for letter in word:
        if guessed_letters.find(letter) == -1:
            return False
    return True
play()
```

In diesem Stadium hat das Spiel noch einige Einschränkungen. Sie müssen Ihren Tipp in Kleinbuchstaben eingeben, so wie die Wörter auch im Array stehen. Wenn Sie versehentlich `aa` statt `a` als Tipp eingeben, wird das als ganzes Wort behandelt. Das Spiel sollte das erkennen und nur Tipps in der Länge des geheimen Worts als ganzes Wort interpretieren.

Übungshalber sollten Sie versuchen, diese Änderungen umzusetzen. Tipp: Um das Schreibweisenproblem zu lösen, versuchen Sie es mit der Funktion `lower`. Sie finden die angepasste Programmversion in der Datei **4_8_hangman_full_solution.py**.

4.5 Dictionaries

Listen sind prima, wenn Sie von Anfang an auf Ihre Daten zugreifen und sich hindurcharbeiten möchten, aber sie sind langsam und ineffizient, wenn sie lang werden und Sie viele Daten durchkämmen müssen (z. B. nach einem bestimmten Eintrag). Sie ähneln ein wenig einem Buch ohne Inhaltsverzeichnis und Index. Um das Gesuchte zu finden, müssen Sie das Buch komplett durchlesen.

Wie Sie vielleicht schon erraten haben, bieten Dictionaries einen effizienteren Weg, um auf ein bestimmtes Element in einer Datenstruktur zuzugreifen. Bei einem Dictionary versehen Sie jeden Wert mit einem Schlüssel. Wenn Sie den Wert abrufen möchten, benutzen Sie dazu den Schlüssel. Das ähnelt zwar einem Variablennamen, dem ein Wert zugewiesen ist, der Unterschied besteht aber darin, dass ein Dictionary aufgebaut werden kann, während das Programm bereits läuft.

Hier ist ein Beispiel:

```
>>> eggs_per_week = {'Penny': 7, 'Amy': 6, 'Bernadette': 0}
>>> eggs_per_week['Penny'] 7
>>> eggs_per_week['Penny'] = 5
>>> eggs_per_week {'Amy': 6, 'Bernadette': 0, 'Penny': 5}
>>>
```

Dieses Beispiel dient zum Verzeichnen der von meinen Hennen gelegten Eier. Jeder Henne ist die Anzahl der pro Woche gelegten Eier zugeordnet. Wenn wir den Wert für eine der Hennen abrufen wollen (z. B. Penny), verwenden wir wie bei einer Liste statt eines Index ihren Namen in eckigen Klammern. Mit derselben Syntax können wir auch einzelne Werte verändern.

Wenn Bernadette beispielsweise ein Ei gelegt hat, können wir die Aufzeichnung wie folgt aktualisieren:

```
eggs_per_week['Bernadette'] = 1
```

Vielleicht haben Sie bemerkt, dass beim Ausgeben des Dictionary die Reihenfolge nicht der bei der Definition entsprach. Ein Dictionary behält die Reihenfolge der definierten Elemente nicht bei. Und obwohl wir einen String als Schlüssel und eine Zahl als Wert definiert haben, könnte der Schlüssel ein String, eine Zahl oder ein Tupel sein (siehe nächster Abschnitt). Der Wert könnte alles enthalten, sogar eine Liste oder ein anderes Dictionary.

4.6 Tupel

Zuerst scheinen Tupel genau dasselbe wie Listen zu sein, nur ohne eckige Klammern. Wir können also ein Tupel wie folgt definieren und darauf zugreifen:

```
>>> tuple = 1, 2, 3
>>> tuple (1, 2, 3)
>>> tuple[0] 1
```

Wenn wir aber versuchen, ein Element eines Tupels zu ändern, bekommen wir eine Fehlermeldung wie diese:

```
>>> tuple[0] = 6
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Der Grund für diese Fehlermeldung ist, dass Tupel unveränderlich sind. Das gilt auch für Strings und Zahlen. Obwohl Sie eine Variable ändern können, um auf andere Strings, Zahlen oder Tupel zu verweisen, können Sie die Zahl selbst nicht verändern. Ist die Variablenreferenz aber eine Liste, können Sie sie ändern, indem Sie Elemente hinzufügen, entfernen oder ändern.

Wenn ein Tupel also eine Liste darstellt, mit der Sie nicht viel anstellen können, fragen Sie sich vielleicht, warum Sie sie verwenden sollten. Der Grund besteht darin, dass Tupel gut für temporäre Datensammlungen geeignet sind. In Python können Sie einige Tricks auf Tupel anwenden, wie in den folgenden beiden Abschnitten beschrieben.

4.6.1 Mehrfachzuweisung

Um einer Variablen einen Wert zuzuweisen, verwenden Sie den Operator =, wie hier:

```
a = 1
```

Sie können in Python auch mehrere Zuweisungen in einer Zeile vornehmen:

```
>>> a, b, c = 1, 2, 3
>>> a
1
>>> b
2
>>> c
3
```

4.6.2 Mehrere Rückgabewerte

Manchmal möchten Sie in einer Funktion mehrere Werte gleichzeitig zurückgeben: Stellen Sie sich zum Beispiel eine Funktion vor, die aus einer Liste mit Zahlen den kleinsten und größten Wert zurückgibt. Dies ist ein solches Beispiel:

```
#04_09_stats
def stats(numbers):
    numbers.sort()
    return (numbers[0], numbers[-1])

list = [5, 45, 12, 1, 78]
min, max = stats(list)
print(min)
print(max)
```

Diese Methode zum Ermitteln von Minimum und Maximum ist nicht sonderlich effizient, aber ein gutes Beispiel. Die Liste wird sortiert, und wir verwenden dann die erste und die letzte Zahl. Beachten Sie, dass `numbers[-1]` die letzte Zahl zurückgibt, wenn Sie einem Array oder String einen negativen Wert zuweisen. Python zählt vom Ende der Liste oder des Strings rückwärts. Die Position `-1` zeigt also das letzte Element an, `-2` das vorletzte usw.

4.7 Ausnahmen

Python verwendet Ausnahmen, um anzuzeigen, dass etwas in Ihrem Programm schiefgelaufen ist. Während des Programmlaufs können die unterschiedlichsten Fehler auftauchen. Ein Beispiel dafür ist der Versuch, auf ein Element außerhalb des zulässigen Bereichs einer Liste oder eines Strings zuzugreifen, wie bereits erwähnt. Hier ist ein Beispiel:

```
>>> list = [1, 2, 3, 4]
>>> list[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

Wenn jemand, der Ihr Programm verwendet, eine Fehlermeldung wie diese erhält, ist das sehr verwirrend. Daher bietet Python eine Möglichkeit, solche Probleme abzufangen und sie selbst zu lösen.

```
try:
    list = [1, 2, 3, 4]
    list[4]
```

```
except IndexError:
   print('Oops')
```

Wir behandeln Ausnahmen erneut im folgenden Kapitel, in dem Sie die Hierarchie der unterschiedlichen Fehlertypen, die auftreten können, kennenlernen.

4.8 Zusammenfassung der Funktionen

Dieses Kapitel diente dazu, Ihnen die wichtigsten Python-Funktionen schnellstmöglich darzulegen. Dabei haben wir einige Dinge recht knapp behandelt und einige andere weggelassen. Aus diesem Grund enthält der folgende Abschnitt eine Übersicht über wichtige Funktionen für die meisten Typen, die wir besprochen haben. Nutzen Sie diese Übersicht auch, wenn Sie dieses Buch weiter durchlesen, und probieren Sie einige der Funktionen aus, um zu sehen, wie sie funktionieren. Sie müssen nicht jedes Detail dieses Abschnitts nachvollziehen – Sie sollten nur wissen, dass Sie hier die Informationen finden. Vergessen Sie nicht: Die Python-Shell ist Ihr Freund.

Einzelheiten für fast alles in Python finden Sie auch unter *http://docs.python.org/py3k*.

4.8.1 Zahlen

Tabelle 4.1 zeigt einige der Funktionen, die Sie für Zahlen verwenden können.

Tabelle 4.1: Zahlenfunktionen

Funktion	*Beschreibung*	*Beispiel*
`abs(x)`	Gibt den absoluten Wert zurück (entfernt das Vorzeichen).	`>>>abs(-12.3)` `12.3`
`bin(x)`	Konvertiert einen Binärstring.	`>>> bin(23)` `'0b10111'`
`complex(r,i)`	Erzeugt eine komplexe Zahl mit Real- und Imaginäranteil. Für Wissenschaft und Ingenieurwesen.	`>>> complex(2,3)` `(2+3j)`
`hex(x)`	Konvertiert einen Hexadezimalstring.	`>>> hex(255)` `'0xff'`
`oct(x)`	Konvertiert einen Oktalstring.	`>>> oct(9)` `'0o11'`
`round(x, n)`	Rundet x auf n Dezimalstellen.	`>>> round(1.111111, 2)` `1.11`

Funktion	Beschreibung	Beispiel
`math. factorial(n)`	Fakultätsfunktion (wie 4x3x2x1).	`>>> math.factorial (4)24`
`math.log(x)`	Natürlicher Logarithmus.	`>>> math.log(10) 2.302585092994046`
`math.pow(x, y)`	Erhebt x zur Potenz von y (alternativ `x ** y`).	`>>> math.pow(2, 8) 256.0`
`math.sqrt(x)`	Quadratwurzel.	`>>> math.sqrt(16) 4.0`
`math.sin, cos, tan, asin, acos, atan`	Trigonometrische Funktionen (Bogenmaß).	`>>> math.sin(math.pi / 2) 1.0`

4.8.2 Strings

Stringkonstanten können in einzelne (meistens) oder doppelte Anführungszeichen eingeschlossen werden. Doppelte Anführungszeichen sind nützlich, wenn sich im String bereits einfache Anführungszeichen befinden, wie hier:

```
s = "Its 3 o'clock"
```

Manchmal möchten Sie Sonderzeichen wie Zeilenende oder Tabulator in einen String einfügen. Dazu verwenden Sie sogenannte Escape-Zeichen, die mit einem Backslash (\) beginnen. Die folgenden werden Sie am häufigsten benötigen:

- `\t`: *Tabulator*
- `\n`: *Zeilenende*

Tabelle 4.2 zeigt einige der Stringfunktionen.

Tabelle 4.2: Stringfunktionen

Funktion	Beschreibung	Beispiel
`s.capitalize()`	Schreibt den ersten Buchstaben groß und den Rest klein.	`>>> 'aBc'.capitalize() 'Abc'`
`s.center(width)`	Füllt den String mit Leerzeichen auf und zentriert ihn. Ein optionaler Parameter ist das Füllzeichen.	`>>> 'abc'.center(10, '-') '---abc----'`

Funktion	Beschreibung	Beispiel
`s.endswith(str)`	Gibt `True` zurück, wenn das Ende des Strings übereinstimmt.	`>>> 'abcdef' .endswith('def')` `True`
`s.find(str)`	Gibt die Position eines Teilstrings zurück. Optionale Suchargumente für die Start- und Endposition können den Bereich einschränken.	`>>> 'abcdef'.find('de')` `3`
`s.format(args)`	Formatiert einen String mit Templatemarkern mittels `{}`.	`>>> "Its {0} pm".format('12')` `"Its 12 pm"`
`s.isalnum()`	Gibt `True` zurück, wenn alle Zeichen im String Buchstaben oder Ziffern sind.	`>>> '123abc'.isalnum()` `True`
`s.isalpha()`	Gibt `True` zurück, wenn alle Zeichen alphabetisch sind.	`>>> '123abc'.isalpha()` `False`
`s.isspace()`	Gibt `True` zurück, wenn das Zeichen ein Leerzeichen, ein Tabulator oder ein anderes Abstandszeichen ist.	`>>> ' \t'.isspace()` `True`
`s.ljust(width)`	Wie `center()`, nur linksbündig.	`>>> 'abc'.ljust(10, '-')` `'abc-------'`
`s.lower()`	Konvertiert einen String in Kleinbuchstaben.	`>>> 'AbCdE'.lower()` `'abcde'`
`s.replace(old, new)`	Ersetzt alle Vorkommen von `old` durch `new`.	`>>> 'hello world' .replace('world', 'there')` `'hello there'`
`s.split()`	Gibt eine durch Leerzeichen getrennte Liste aller Wörter eines Strings zurück. Ein optionaler Parameter kann als Ersatztrennzeichen angegeben werden. Gern wird dafür ein Zeilenendezeichen (`\n`) verwendet.	`>>> 'abc def'.split()` `['abc', 'def']`
`s.splitlines()`	Teilt den String am Zeilenendezeichen.	

Funktion	Beschreibung	Beispiel
`s.strip()`	Entfernt Leerraum an beiden Enden des Strings.	`>>> '  a b  'strip()` `'a b'`
`s.upper()`	Wie `lower()` weiter oben in dieser Tabelle.	

4.8.3 Listen

Die meisten Funktionen für Listen haben wir uns bereits angesehen. Tabelle 4.3 fasst diese Funktionen zusammen.

Tabelle 4.3: Listenfunktionen

Funktion	Beschreibung	Beispiel

`del(a[i:j])`	Löscht die Elemente von `i` bis `j-1` aus dem Array.	`>>> a = ['a', 'b', 'c']`
		`>>> del(a[1:2])`
		`>>> a`
		`['a', 'c']`

`a.append(x)`	Fügt am Ende der Liste ein Element an.	`>>> a = ['a', 'b', 'c']`
		`>>> a.append('d')`
		`>>> a`
		`['a', 'b', 'c', 'd']`

`a.count(x)`	Zählt die Vorkommen eines bestimmten Elements.	`>>> a = ['a', 'b', 'a']`
		`>>> a.count('a')`
		`2`

`a.index(x)`	Gibt die Indexposition des ersten Vorkommens von `x` in `a` an. Optionale Parameter können für den Start- und Endindex verwendet werden.	`>>> a = ['a', 'b', 'c']`
		`>>> a.index('b')`
		`1`

`a.insert (i, x)`	Fügt `x` an Position `i` in die Liste ein.	`>>> a = ['a', 'c']`
		`>>> a.insert(1, 'b')`
		`>>> a`
		`['a', 'b', 'c']`

`a.pop()`	Gibt das letzte Element der Liste zurück und entfernt es. Ein optionaler Parameter ermöglicht die Angabe eines anderen Index zum Entfernen.	`>>> ['a', 'b', 'c']`
		`>>> a.pop(1)  'b'`
		`>>> a`
		`['a', 'c']`

Funktion	Beschreibung	Beispiel

`a.remove(x)`	Entfernt das angegebene Element.	`>>> a = ['a', 'b', 'c']`
		`>>> a.remove('c')`
		`>>> a`
		`['a', 'b']`

`a.reverse()`	Kehrt die Liste um.	`>>> a = ['a', 'b', 'c']`
		`>>> a.reverse()`
		`>>> a`
		`['c', 'b', 'a']`

`a.sort()`	Sortiert die Liste. Für die Sortierung von Objektlisten sind weitere Optionen verfügbar. Details finden Sie im nächsten Kapitel.	

4.8.4 Dictionaries

Tabelle 4.4 zeigt ein paar Informationen über Dictionaries, die Sie kennen sollten.

Tabelle 4.4: Dictionaryfunktionen

Funktion	Beschreibung	Beispiel

`len(d)`	Gibt die Anzahl der Objekte im Dictionary zurück.	`>>> d = {'a':1, 'b':2}`
		`>>> len(d)`
		`2`

`del(d[key])`	Löscht ein Objekt aus dem Dictionary.	`>>> d = {'a':1, 'b':2}`
		`>>> del(d['a']) >>> d`
		`{'b': 2}`

`key in d`	Gibt `True` zurück, wenn Dictionary `(d)` den Schlüssel enthält.	`>>> d = {'a':1, 'b':2}`
		`>>> 'a' in d`
		`True`

`d.clear()`	Entfernt alle Objekte aus dem Dictionary.	`>>> d = {'a':1, 'b':2}`
		`>>> d.clear()`
		`>>> d`
		`{}`

`get(key, default)`	Gibt den Wert für den Schlüssel oder die Vorgabe zurück, wenn dort kein Schlüssel ist.	`>>> d = {'a':1, 'b':2}`
		`>>> d.get('c', 'c')`
		`'c'`

4.8.5 Typumwandlungen

Wir haben bereits die Situation besprochen, in der wir eine Zahl in einen String umwandeln wollen, sodass wir sie an einen anderen String anfügen können. Python enthält einige integrierte Funktionen, um Objekte verschiedener Typen ineinander zu konvertieren, wie in Tabelle 4.5 beschrieben.

Tabelle 4.5: Typumwandlungen

Funktion	*Beschreibung*	*Beispiel*

`float(x)`	Konvertiert x in eine Fließkommazahl.	`>>> float('12.34')`
		`12.34`
		`>>> float(12)`
		`12.0`

`int(x)`	Optionales Argument zur Angabe der Zahlenbasis.	`>>> int(12.34)`
		`12`
		`>>> int('FF', 16)`
		`255`

`list(x)`	Konvertiert x in eine Liste. Ein einfacher Weg, um eine Liste mit Dictionary-Schlüsseln zu bekommen.	`>>> list('abc')`
		`['a', 'b', 'c']`
		`>>> d = {'a':1, 'b':2}`
		`>>> list(d)`
		`['a', 'b']`

4.9 Zusammenfassung

Viele Dinge in Python entdecken Sie stückweise. Verzweifeln Sie also nicht bei dem Gedanken, sie alle auswendig lernen zu müssen. Das ist wirklich nicht notwendig, da Sie immer nach Python-Befehlen suchen oder sie nachschlagen können.

Im nächsten Kapitel gehen wir einen Schritt weiter und sehen uns an, wie Python mit Objektorientierung umgeht.

5 Module, Klassen und Methoden

In diesem Kapitel besprechen wir, wie wir unsere eigenen Module erstellen, so wie das Modul `random` aus Kapitel 3. Wir sehen uns auch an, wie Python Objektorientierung implementiert, die es uns ermöglicht, Programme durch Klassen zu strukturieren, die ihr jeweils eigenes Verhalten haben. Das hilft uns dabei, die Komplexität unserer Programme im Blick zu behalten, und erleichtert uns die Verwaltung. Die Hauptmechanismen dafür sind Klassen und Methoden. Sie haben bereits integrierte Klassen und Methoden in den früheren Kapiteln verwendet, ohne es zu wissen.

5.1 Module

Die meisten Programmiersprachen haben ein Konzept wie Klassen, die es Ihnen ermöglichen, eine Gruppe von Funktionen zu erstellen, die für andere leicht zu verwenden sind – und auch für Sie selbst.

Python bietet diese Gruppierung von Funktionen auf einfache und elegante Weise. Grundsätzlich kann jede Datei mit Python-Code als Modul mit dem Namen der Datei gelten. Bevor wir jedoch unsere eigenen Module schreiben, sehen wir uns an, wie wir die bereits in Python integrierten Module benutzen.

5.1.1 Module verwenden

Als wir das Modul `random` zuletzt benutzten, machten wir das wie folgt:

```
>>> import random
>>> random.randint(1, 6)
6
```

Zuerst haben wir Python mitgeteilt, dass wir das Modul `random` verwenden möchten, indem wir es mit dem Befehl `import` importiert haben. Irgendwo in der Python-Installation befindet sich eine Datei namens **random.py**, in der unter anderem die Funktion `randint` enthalten ist.

Bei so vielen verfügbaren Modulen ist die Gefahr groß, dass unterschiedliche Module Funktionen mit dem gleichen Namen haben. Woher weiß Python in diesem Fall, welche

benutzt werden soll? Glücklicherweise müssen wir uns darum nicht kümmern, da wir das Modul importiert haben und keine der Funktionen darin sichtbar ist, bis wir den Modulnamen schreiben und einen Punkt vor den Funktionsnamen setzen. Lassen Sie den Modulnamen einfach einmal weg, so wie hier:

```
>>> import random
>>> randint(1, 6)
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
NameError: name 'randint' is not defined
```

Den Modulnamen jedes Mal bei einem Funktionsaufruf voranstellen zu müssen, kann etwas lästig werden. Glücklicherweise wird das etwas einfacher, indem wir dem Befehl `import` etwas hinzufügen:

```
>>> import random as r
>>> r.randint(1,6)
2
```

Dadurch erhält das Modul innerhalb unseres Programms den lokalen Namen `r`, was uns viel Tipperei erspart.

Wenn Sie sicher sind, dass der Name einer Funktion aus einer Bibliothek in Ihrem Programm keine Konflikte verursacht, können Sie auch einen Schritt weiter gehen:

```
>>> from random import randint
>>> randint(1, 6)
5
```

Wollen Sie die Sache weiter beschleunigen, können Sie alles aus dem Modul in einem Rutsch importieren. Wenn Sie nicht genau wissen, was sich im Modul befindet, ist das zwar nicht sinnvoll, aber möglich. So geht es:

```
>>> from random import *
>>> randint(1, 6)
2
```

In diesem Fall bedeutet der Stern (*) »alles«.

5.1.2 Nützliche Python-Bibliotheken

Bislang haben wir das Modul `random` verwendet, es gibt in Python aber noch andere. Diese Module werden oft als Standardbibliothek von Python bezeichnet. Es gibt zu viele dieser Module, um sie hier alle aufzulisten. Sie finden die vollständige Liste aller Python-

Module online unter *http://docs.python.org/release/3.1.5/library/index.html*. Die hier genannten Module sollten Sie sich einmal genauer ansehen:

- `string` String-Utilities.
- `datetime` Zum Arbeiten mit Datum und Uhrzeit.
- `math` Mathematische Funktionen (Sinus, Kosinus usw.).
- `pickle` Zum Speichern und Wiederherstellen von Datenstrukturen in Dateien (siehe Kapitel 6).
- `urlib.request` Zum Lesen von Webseiten (siehe Kapitel 6).
- `tkinter` Für spezielle grafische Benutzeroberflächen (siehe Kapitel 7).

5.1.3 Neue Module installieren

Zusätzlich zu den Modulen der Standardbibliothek gibt es Tausende Module aus der Python-Community. Ein sehr beliebtes Modul heißt `pygame`, das in Kapitel 8 verwendet wird. Oft ist es als binäres Paket erhältlich, sodass Sie es durch folgende Eingabe installieren können:

```
sudo apt-get install python-pygame
```

Auf viele Module trifft das aber nicht zu, und Sie müssen etwas mehr tun, um sie zu installieren.

Jedes nützliche Modul wird nach einem standardisierten Verfahren gepackt. Um es zu installieren, müssen Sie eine komprimierte Datei herunterladen, in der sich ein Verzeichnis für das Modul befindet. Lassen Sie uns das `RPi.GPIO`-Modul nehmen, das wir in Kapitel 10 als Beispiel verwenden. Um dieses Modul zu installieren, gehen Sie zuerst zur Website des Moduls, suchen den Downloadabschnitt und laden die Archivdatei herunter. Das wird in Bild 5.1 gezeigt. Dann wird die Datei in einem Verzeichnis abgelegt (zum Beispiel im Python-Verzeichnis, das wir in Kapitel 3 erstellt haben).

Wenn die Datei gespeichert ist, öffnen Sie LXTerminal und verwenden `cd`, um zum Python-Verzeichnis zu gelangen:

```
pi@raspberrypi:~/Python$ ls
RPi.GPIO-0.3.1a.tar.gz
```

Jetzt müssen Sie mit folgendem Befehl aus dem Archiv das Verzeichnis extrahieren:

```
pi@raspberrypi:~/Python$ tar -xzf RPi.GPIO-0.3.1a.tar.gz
pi@raspberrypi:~/Python$ ls
RPi.GPIO-0.3.1a  RPi.GPIO-0.3.1a.tar.gz
```

Sie haben jetzt einen neuen Ordner mit dem Modul, wechseln mittels `cd` dorthin und installieren es mit `install`. Es ist immer sinnvoll, nach Anweisungen zu suchen, die Ihnen sagen, ob Sie mehr tun müssen. Um die Anweisungen zu sehen, geben Sie `more INSTALL.txt` ein.

Gut, dass Sie einmal nachgesehen haben! Die Anweisungen besagen, dass Sie noch Folgendes tun müssen:

```
sudo apt-get install python3-dev
```

Jetzt sind Sie so weit, den Modul-Installer verwenden zu können.

```
pi@raspberrypi:~/Python$ cd RPi.GPIO-0.3.1a
pi@raspberrypi:~/Python/RPi.GPIO-0.3.1a$ sudo python3
setup.py install
```

Wenn das Modul installiert ist, können Sie es in der Python-Shell installieren.

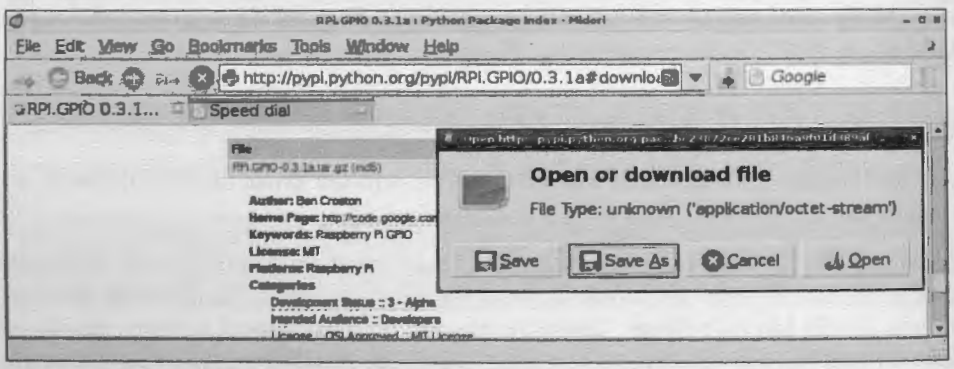


Bild 5.1: Das Modul `RPi.GPIO` herunterladen

5.2 Objektorientierung

Objektorientierung hat viel mit Modulen zu tun. Sie hat ebenfalls das Ziel, zusammengehörende Objekte beieinanderzuhalten, sodass sie einfach verwaltet und gefunden werden können. Wir geben daher ein:

```
>>> 'abc'.upper()
```

Damit sagen wir dem String `abc`, dass wir eine Kopie von ihm haben möchten, aber in Großbuchstaben. Objektorientiert ausgedrückt, bedeutet das, dass `abc` eine Instanz der integrierten Klasse `str` ist und `upper` eine Methode für die Klasse `str`.

Wir finden die Klasse eines Objekts wie hier gezeigt heraus (beachten Sie den doppelten Unterstrich vor und nach dem Wort `class`):

```
>>> 'abc'.__class__
<class 'str'>
>>> [1].__class__
<class 'list'>
>>> 12.34.__class__
<class 'float'>
```

5.3 Klassen definieren

Das ist genug zu fremden Klassen, erstellen wir jetzt unsere eigene. Wir beginnen mit der Erstellung einer Klasse, die die Umrechnung einer Einheit in eine andere übernimmt, indem sie einen Wert mit einem Skalierungsfaktor multipliziert.

Wir geben der Klasse den treffenden Namen `ScaleConverter`. Es folgt das Listing für die gesamte Klasse sowie ein paar Zeilen Code zum Testen.

```
#05_01_converter
class ScaleConverter:
   def __init__(self, units_from, units_to, factor):
      self.units_from = units_from
      self.units_to = units_to
      self.factor = factor

   def description(self):
      return 'Convert ' + self.units_from + ' to ' + self.units_to

   def convert(self, value):
      return value * self.factor

c1 = ScaleConverter('inches', 'mm', 25)
print(c1.description())
print('converting 2 inches')
print(str(c1.convert(2)) + c1.units_to)
```

Ein wenig Erläuterung ist vermutlich notwendig. Die erste Zeile ist recht klar: Sie deutet den Definitionsbeginn einer Klasse namens `ScaleConverter` an. Der Doppelpunkt (`:`) am Ende zeigt an, dass alle folgenden Zeilen Teil der Klassendefinition sind, bis wir wieder am linken Zeilenbeginn ankommen.

Innerhalb von `ScaleConverter` sehen wir drei verschiedene Funktionen. Diese Funktionen gehören zur Klasse und können nur über eine Instanz dieser Klasse verwendet werden. Diese Art von Funktionen, die zu einer Klasse gehören, heißen Methoden.

Die erste Methode, `__init__`, sieht etwas merkwürdig aus. Sie hat auf beiden Seiten zwei Unterstriche. Wenn Python eine neue Instanz einer Klasse erstellt, wird automatisch die Methode `__init__` aufgerufen. Die Anzahl der Parameter, die `__init__` haben sollte, hängt davon ab, welche Parameter bei der Erstellung der Instanz der Klasse angegeben werden. Um das herauszufinden, schauen wir uns folgende Zeile am Ende der Datei an:

```
c1 = ScaleConverter('inches', 'mm', 25)
```

Diese Zeile erzeugt eine neue Instanz von `ScaleConverter` und gibt an, welche Einheiten in welche Richtung konvertiert werden, sowie einen Skalierungsfaktor. Die Methode `__init__` muss diese Parameter enthalten sowie einen Parameter `self` als ersten Parameter.

```
def __init__(self, units_from, units_to, factor):
```

Der Parameter `self` bezieht sich auf das Objekt selbst. Wenn wir uns den Körper der Methode `__init__` ansehen, benötigen wir einige Zuweisungen:

```
self.units_from = units_from
self.units_to = units_to
self.factor = factor
```

Jede dieser Zuweisungen erzeugt eine Variable, die zum Objekt gehört und ihren Initialwert von an `__init__` übergebenen Parametern bekommt.

Wenn wir also einen neuen `ScaleConverter` durch folgende Eingabe erzeugen

```
c1 = ScaleConverter('inches', 'mm', 25)
```

erzeugt Python eine neue Instanz von `ScaleConverter` und weist den drei Variablen die Werte `'inches'`, `'mm'` und `25` zu.

Der Begriff Kapselung taucht häufig in Abhandlungen zu Klassen auf. Es ist Aufgabe einer Klasse, alles, was sie durchführt, zu kapseln. Das bedeutet, dass Daten (wie die drei Variablen) und andere Dinge, die Sie mit den Daten anstellen möchten, in Form der Methoden `description` und `convert` gespeichert werden.

Die erste (`description`) übernimmt die Informationen, die der Konverter über Einheiten kennt, und erzeugt einen String, der sie beschreibt. Wie bei `__init__` müssen alle Methoden einen ersten Parameter `self` besitzen. Die Methode benötigt ihn wahrscheinlich, um auf die Daten der dazugehörigen Klasse zuzugreifen.

Versuchen Sie es selbst, indem Sie das Programm **05_01_converter.py** ausführen und Folgendes in die Python-Shell eingeben:

```
>>> silly_converter = ScaleConverter('apples', 'grapes', 74)
>>> silly_converter.description()
'Convert apples to grapes'
```

Die Methode `convert` hat zwei Parameter: den notwendigen Parameter `self` und einen Parameter namens `value`. Die Methode gibt einfach das Ergebnis der Multiplikation des übergebenen Werts in `self.factor` zurück:

```
>>> silly_converter.convert(3)
222
```

5.4 Vererbung

Der `ScaleConverter` ist sinnvoll bei der Umwandlung von Längen und Ähnlichem. Er funktioniert jedoch nicht bei der Umwandlung von z. B. Celsius (C) in Fahrenheit (F). Die Formel dafür lautet F = C * 1,8 + 32.

Es gibt also einen Skalierungsfaktor (1,8) und einen Offset (32).

Lassen Sie uns einen Konverter namens `ScaleAndOffsetConverter` erstellen, der wie `ScaleConverter` funktioniert, aber auch einen Offset unterstützt. Eine Möglichkeit besteht darin, einfach den Code von `ScaleConverter` zu kopieren und ihm eine Extravariable hinzuzufügen. Das Ganze könnte dann wie folgt aussehen:

```
#05_02_converter_offset_bad
   class ScaleAndOffsetConverter:
      def __init__(self, units_from, units_to, factor, offset):
         self.units_from = units_from
         self.units_to = units_to
         self.factor = factor
         self.offset = offset

      def description(self):
         return 'Convert ' + self.units_from + ' to ' + self.units_to

      def convert(self, value):
         return value * self.factor + self.offset

c2 = ScaleAndOffsetConverter('C', 'F', 1.8, 32)
print(c2.description())
```

```
print('converting 20C')
print(str(c2.convert(20)) + c2.units_to)
```

Nehmen wir an, wir möchten beide Konvertertypen in dem Programm haben, das wir schreiben, dann ist diese Vorgehensweise nicht zu empfehlen, denn wir wiederholen dabei Code. Wir sollten also eine andere Lösung finden.Die Methode `description` ist bereits vorhanden, und `__init__` ist ebenfalls fast gleich. Es ist wesentlich sinnvoller, **Vererbung** einzusetzen.

Bei der Vererbung von Klassen geht es darum, dass Sie bei einer speziellen Variante einer bestehenden Klasse alle Variablen und Methoden der Elternklasse vererben können und neue einfach hinzufügen oder überschreiben. Bild 5.2 zeigt ein Klassendiagramm für die beiden Klassen und wie `ScaleAndOffsetConverter` von `ScaleConverter` erbt, eine neue Variable (`offset`) einführt und die Methode `convert` überschreibt (da sie etwas anders funktioniert).

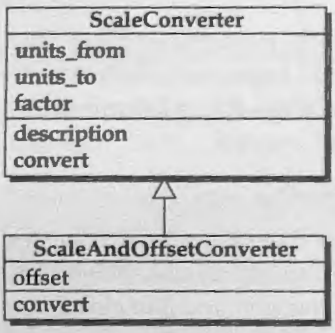


Bild 5.2: Ein Beispiel für Vererbung

Dies ist die Klassendefinition für `ScaleAndOffsetConverter` mit Vererbung:

```
class ScaleAndOffsetConverter(ScaleConverter):

    def __init__(self, units_from, units_to, factor, offset):
        ScaleConverter.__init__(self, units_from, units_to, factor)
        self.offset = offset

    def convert(self, value):
        return value * self.factor + self.offset
```

Zuerst ist zu beachten, dass der Klassendefinition für `ScaleAndOffsetConverter` die von `ScaleConverter` in Klammern direkt folgt. So geben Sie die Elternklasse für eine Klasse an.

Die Methode `__init__` für die neue Unterklasse von `ScaleConverter` ruft die `__init__`-Methode von `ScaleConverter` auf, bevor die neue Variable `offset` definiert wird. Die Methode `convert` überschreibt die Methode `convert` in der Elternklasse, da wir für diese

Art Konverter einen Offset hinzufügen müssen. Sie können die beiden Klassen zusammen ausführen und experimentieren, indem Sie **05_03_converters_final.py** starten.

```
>>> c1 = ScaleConverter('inches', 'mm', 25)
>>> print(c1.description())
Convert inches to mm
>>> print('converting 2 inches')
converting 2 inches
>>> print(str(c1.convert(2)) + c1.units_to)
50mm
>>> c2 = ScaleAndOffsetConverter('C', 'F', 1.8, 32)
>>> print(c2.description())
Convert C to F
>>> print('converting 20C')
converting 20C
>>> print(str(c2.convert(20)) + c2.units_to)
68.0F
```

Es ist leicht, diese beiden Klassen in ein Modul zu konvertieren, das wir in anderen Programmen verwenden können. Auf dieses Modul werden wir in Kapitel 7 zurückkommen, wenn wir es mit einer grafischen Benutzerschnittstelle versehen.

Um diese Datei in ein Modul zu konvertieren, sollten wir den Code an ihrem Ende prüfen und der Datei dann einen passenderen Namen geben. Nennen wir sie **converters.py**. Sie finden die Datei unter den Downloads für dieses Buch. Das Modul muss sich im selben Verzeichnis befinden wie das Programm, das es nutzen will.

Um das Modul zu verwenden, machen Sie Folgendes:

```
>>> import converters
>>> c1 = converters.ScaleConverter('inches', 'mm', 25)
>>> print(c1.description())
Convert inches to mm
>>> print('converting 2 inches')
converting 2 inches
>>> print(str(c1.convert(2)) + c1.units_to)
50mm
```

5.5 Zusammenfassung

Es gibt für Python viele Module und einige davon speziell für den Raspberry Pi, wie z. B. `RPi.GPIO library` zum Steuern der GPIO-Pins. Beim Durcharbeiten dieses Buchs lernen Sie weitere Module kennen. Wenn die von Ihnen geschriebenen Programme komplexer

werden, werden Sie den objektorientierten Ansatz für Design und Codierung zu schätzen wissen, da er Ihre Projekte übersichtlicher gestaltet.

Im nächsten Kapitel sehen wir uns Dateien und das Internet an.

6 Dateien und das Internet

Python macht Ihren Programmen die Verwendung von Dateien und Internet leicht. Sie können Daten aus Dateien lesen und in sie hineinschreiben sowie Inhalte aus dem Internet holen. Sie können sogar Ihre E-Mails abholen und twittern – alles aus Ihrem Programm heraus.

6.1 Dateien

Wenn Sie ein Python-Programm starten, gehen alle Werte in Variablen verloren. Mit Dateien können Sie Daten permanenter bereithalten.

6.1.1 Dateien lesen

Python macht es sehr einfach, den Inhalt aus Dateien zu lesen. Zum Beispiel können wir das Programm Hangman aus Kapitel 4 so umschreiben, dass die Wortliste aus einer Datei gelesen wird, statt fest ins Programm integriert zu sein.

Zuerst erstellen wir in IDLE eine neue Datei und schreiben ein paar Wörter hinein, jeweils eines pro Zeile. Dann speichern Sie die Datei unter dem Namen **hangman_words.txt** im selben Verzeichnis wie das Hangman-Programm aus Kapitel 4 (**04_08_hangman_full.py**). Achten Sie darauf, im Speichern-Dialog den Dateityp auf **.txt** zu ändern (siehe Bild 6.1).

Bevor wir das Hangman-Programm verändern, können wir damit experimentieren und versuchen, eine Datei an der Python-Konsole zu lesen. Geben Sie an der Konsole Folgendes ein:

```
>>> f = open('Python/hangman_words.txt')
```

Beachten Sie, dass die Python-Konsole als aktuelles Verzeichnis **/home/pi** verwendet, sodass das Verzeichnis **Python** (oder wo Sie die Datei sonst gespeichert haben) angegeben werden muss.

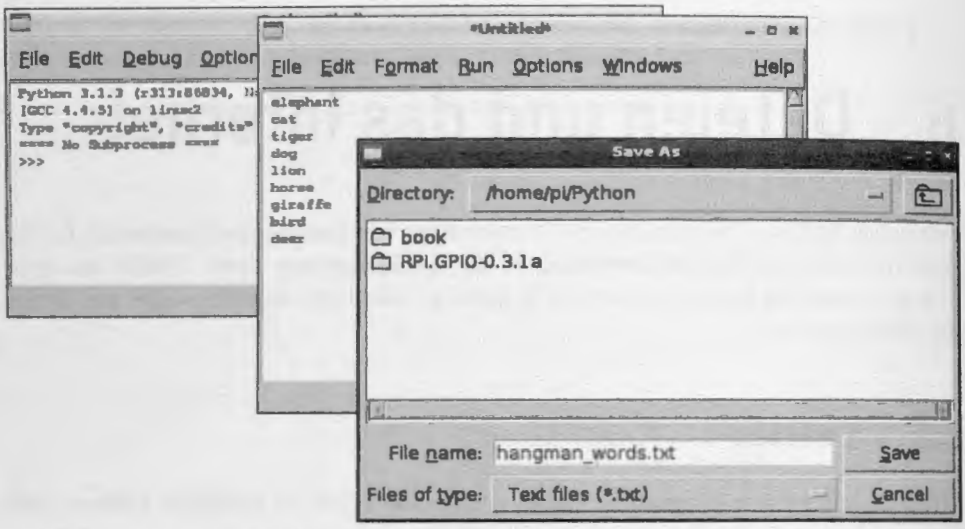


Bild 6.1: In IDLE eine Textdatei erstellen und speichern

Jetzt geben Sie Folgendes an der Python-Konsole ein:

```
>>> words = f.read()
>>> words 'elephant\ncat\ntiger\ndog\nlion\nhorse\ngiraffe\nbird\ndeer\n'
>>> words.splitlines()
['elephant', 'cat', 'tiger', 'dog', 'lion', 'horse', 'giraffe', 'bird',
'deer']
>>>
```

Ich habe Ihnen doch gesagt, dass es einfach ist! Alles, was wir im Hangman-Programm tun müssen, ist, die Zeile

```
words = ['chicken', 'dog', 'cat', 'mouse', 'frog']
```

durch folgende zu ersetzen:

```
f = open('hangman_words.txt') words = f.read().splitlines() f.close()
```

Die Zeile `f.close()` wurde hinzugefügt. Sie sollten den Befehl `close` immer aufrufen, wenn Sie mit einer Datei fertig sind, um Ressourcen des Betriebssystems freizugeben. Eine Datei geöffnet zu lassen, kann zu Problemen führen.

Das gesamte Programm ist in der Datei **06_01_hangman._file.py** enthalten, und eine passende Liste mit Tiernamen finden Sie in der Datei **hangman_words.txt**.

Dieses Programm macht nichts anderes, als zu prüfen, ob die Datei vorhanden ist, bevor sie gelesen wird. Ist die Datei nicht vorhanden, bekommen wir eine Fehlermeldung wie diese:

```
Traceback (most recent call last):
  File "06_01_hangman_file.py", line 4, in <module>
    f = open('hangman_words.txt')
IOError: [Errno 2] No such file or directory: 'hangman_words.txt'
```

Um die Sache etwas benutzerfreundlicher zu machen, verwendet der Code zum Lesen der Datei eine `try`-Anweisung:

```
try:
  f = open('hangman_words.txt')
  words = f.read().splitlines()
  f.close()
except IOError:
  print("Cannot find file 'hangman_words.txt'")
  exit()
```

Python versucht, die Datei zu öffnen, da diese aber fehlt, ist das nicht möglich. Daher wird der Programmteil `except` ausgeführt, und die benutzerfreundliche Meldung wird ausgegeben. Da wir nichts ohne die Wortliste machen können, ist es nicht sinnvoll, weiterzumachen, und wir beenden das Programm mit `exit`.

Beim Ausgeben der Fehlermeldung haben wir den Namen der Datei doppelt angegeben. Wollen wir uns streng an die Vorgabe halten, uns nicht zu wiederholen (DRY), sollte der Dateiname in einer Variablen abgelegt werden, so wie hier. Wenn wir uns später entscheiden, eine andere Datei zu verwenden, müssen wir den Code nur an einer Stelle verändern:

```
words_file = 'hangman_words.txt'
try:
  f = open(words_file)
  words = f.read().splitlines()
  f.close()
except IOError:
  print("Cannot find file: " + words_file)
  exit()
```

Eine modifizierte Fassung von Hangman mit diesem Code finden Sie in der Datei **06_02_hangman_file_try.py**.

6.1.2 Sehr große Dateien lesen

Das im vorigen Abschnitt gezeigte Verfahren eignet sich gut für kleine Dateien mit einigen Wörtern. Wenn wir jedoch eine große Datei (mit mehreren Megabytes) lesen, können zwei Dinge passieren. Es würde sehr lange dauern, bis Python alle Daten gelesen hat. Da außerdem alle Daten in einem Zug gelesen werden, würde Arbeitsspeicher in der Größe der Datei verbraucht, und bei sehr großen Dateien könnte Python der Speicher ausgehen.

Wenn Sie beim Lesen einer großen Datei in diese Situation kommen, müssen Sie eine andere Strategie verfolgen. Suchen Sie in der Datei nach einem bestimmten String, können Sie sie zeilenweise lesen, wie hier:

```
#06_03_file_readline
words_file = 'hangman_words.txt'
try:
   f = open(words_file)
   line = f.readline()
   while line != '':
      if line == 'elephant\n':
         print('There is an elephant in the file')
         break
      line = f.readline()
   f.close()
except IOError:
   print("Cannot find file: + words_file)
```

Wenn die Funktion `readline` zur letzten Zeile der Datei kommt, gibt sie einen leeren String zurück. Wenn nicht, gibt sie die Zeile zurück, abgeschlossen mit dem Zeilenendezeichen (`\n`). Wird eine leere Zeile gelesen, die nur eine Lücke darstellt und nicht das Ende der Datei, gibt sie nur das Zeilenendezeichen (`\n`) zurück. Indem das Programm immer nur eine Zeile liest, wird auch immer nur so viel Arbeitsspeicher verwendet, wie die Zeile lang ist.

Ist die Datei nicht in bequeme Zeilen aufgeteilt, können Sie als Argument angeben, wie viele Zeichen gelesen werden sollen. Der folgende Befehl liest z. B. nur die ersten 20 Zeichen einer Datei:

```
>>> f = open('hangman_words.txt')
>>> f.read(20)
'elephant\ncat\ntiger\nd'
>>> f.close()
```

6.1.3 Dateien schreiben

Dateien zu schreiben, ist fast genauso leicht. Wenn eine Datei geöffnet wird, können Sie neben dem Dateinamen auch den Modus angeben, in dem sie geöffnet werden soll. Der Modus wird durch ein Zeichen angegeben, und wenn es fehlt, wird die Datei im Lesemodus (`r`) geöffnet. Es gibt folgende Modi:

- `r` (read/lesen).
- `w` (write/schreiben): Ersetzt den Inhalt einer bestehenden Datei dieses Namens.
- `a` (append/anhängen): Fügt Informationen ans Ende einer bestehenden Datei an.
- `r+`: Öffnet die Datei zum Lesen und Schreiben (nicht sehr gebräuchlich).

Um in eine Datei zu schreiben, öffnen Sie sie mit einem zweiten Parameter `w`, `a` oder `r+`. Hier ist ein Beispiel:

```
>>> f = open('test.txt', 'w')
>>> f.write('This file is not empty')
>>> f.close()
```

6.1.4 Das Dateisystem

Manchmal werden Sie Dateisystemoperationen mit Dateien durchführen (verschieben, kopieren usw.). Python nutzt Linux, um diese Aufgaben auszuführen, verpackt das aber in freundlichen Python-Anweisungen. Viele dieser Funktionen befinden sich im Paket `shutil` (Shell Utilities). Es gibt ein paar subtile Unterschiede zwischen den grundlegenden Kopier- und Verschiebebefehlen, die mit Zugriffsrechten und Metadaten zu tun haben. In diesem Abschnitt beschäftigen wir uns mit den grundlegenden Operationen. Sie finden in der offiziellen Python-Dokumentation weitere Informationen über die Funktionen (*http://docs.python.org/release/3.1.5/library*).

So kopieren Sie eine Datei:

```
>>> import shutil
>>> shutil.copy('test.txt', 'test_copy.txt')
```

So ändern Sie den Namen einer Datei oder verschieben sie in ein anderes Verzeichnis:

```
shutil.move('test_copy.txt', 'test_dup.txt')
```

Das funktioniert mit Verzeichnissen wie mit Dateien. Wenn Sie einen kompletten Ordner kopieren möchten – einschließlich seines Inhalts und der Unterordner –, können Sie die Funktion `copytree` verwenden. Die recht gefährliche Funktion `rmtree` dagegen durchläuft einen Ordner rekursiv und löscht alle Inhalte. Seien Sie damit sehr vorsichtig!

Am einfachsten lässt sich der Inhalt eines Ordners mit `globbing` herausfinden. Das Paket `global` ermöglicht es, eine Liste von Dateien in einem Ordner mittels Wildcard (*) anzugeben. Hier ist ein Beispiel:

```
>>> import glob
glob.glob('*.txt')
['hangman_words.txt', 'test.txt', 'test_dup.txt']
```

Wenn Sie alle Dateien im Ordner haben möchten, verwenden Sie Folgendes:

```
glob.glob('*')
```

6.2 Pickling

Pickling bedeutet, den Inhalt einer Variablen so in einer Datei zu speichern, dass er später wieder geladen werden kann. Das geschieht dann, wenn Programmdaten zwischen auseinanderliegenden Programmläufen behalten werden sollen. Wir können z. B. eine komplexe Liste erstellen, die eine weitere Liste und verschiedene andere Datenobjekte enthält, und diese in eine Datei namens Pickle schreiben:

```
>>> mylist = ['a', 123, [4, 5, True]]
>>> mylist
['a', 123, [4, 5, True]]
>>> import pickle
>>> f = open('mylist.pickle', 'w')
>>> pickle.dump(mylist, f)
>>> f.close()
```

Wenn Sie die Datei suchen und sie in einem Editor öffnen, sehen Sie kryptische Dinge wie diese:

```
(lp0
S'a'
p1
aI123
a(lp2
I4
aI5
aI01
aa.
```

Das war zu erwarten: Es ist zwar Text, aber nicht in einer für Menschen lesbaren Form. Um eine Pickle-Datei wieder in ein Objekt zu verwandeln, tun Sie Folgendes:

```
>>> f = open('mylist.pickle')
>>> other_array = pickle.load(f)
>>> f.close()
>>> other_array ['a', 123, [4, 5, True]]
```

6.3 Internet

Die meisten Anwendungen nutzen irgendwie das Internet, selbst dann, wenn sie nur prüfen, ob es eine neue Version der Anwendung gibt, über die der Anwender informiert werden soll. Sie interagieren mit einem Webserver, indem sie HTTP-Anfragen (Hypertext Transfer Protocol) austauschen. Der Webserver sendet als Antwort einen Textstrom zurück. Dieser Text besteht aus HTML (Hypertext Markup Language), der Sprache, mit der wir Webseiten erzeugen.

Jetzt geben Sie folgenden Code an der Python-Konsole ein:

```
>>> import urllib.request
>>> u = 'http://www.amazon.com/s/ref=nb_sb_noss?field-keywords=raspberry+pi'
>>> f = urllib.request.urlopen(u)
>>> contents = f.read()
... hier folgt viel HTML-Code
>>> f.close()
```

Sie müssen die Zeile `read` so früh wie möglich ausführen, nachdem Sie die URL geöffnet haben. Sie haben hier eine Anfrage an *www.amazon.com* gesendet und eine Suche nach »raspberry pi« durchgeführt. Danach wird von der Amazon-Webseite HTML-Code zurückgesendet (als hätten Sie einen Webbrowser verwendet).

Wenn Sie sich die Struktur der Webseite genau ansehen, können Sie sehen, dass Sie sie verwenden können, um eine Liste von durch Amazon gefundenen Raspberry-Pi-Objekten anzulegen. Wenn Sie durch den Text scrollen, finden Sie Zeilen wie diese:

```
<div class="productTitle">
<a href="http://www.amazon.com/Raspberry-User-Guide-Gareth-
Halfacree/dp/111846446X">Raspberry Pi User Guide</a>
<span class="ptBrand">by <a href="/Gareth-Halfacree/e /B0088CA5ZM">Gareth
Halfacree</a> and Eben Upton</span>
<span class="binding"> (<span class ="format">Paperback</span> - Nov. 13,
2012)</span>
</div>
```

Die Schlüsselinformationen liegen in `<div class="productTitle">`. Es gibt davon vor jedem Ergebnis eine Instanz. (Es ist hilfreich, die betreffende Seite im Browser zum Vergleich geöffnet zu haben.) Sie sollten jetzt den Text des richtigen Titels herauskopieren. Das können Sie machen, indem Sie die Position des Texts `productTitle`

suchen, zwei >-Zeichen abzählen und dann den Text von dieser Position bis zum nächsten < verwenden, so wie hier:

```
#06_04_amazon_scraping
import urllib.request

u = 'http://www.amazon.com/s/ref=nb_sb_noss?fieldkeywords=raspberry+pi'
f = urllib.request.urlopen(u)
contents = str(f.read()) f.close()
i = 0
while True:
   i = contents.find('productTitle', i)
   if i == -1:
      break
   # Find the next two '>' after 'productTitle'
   i = contents.find('>', i+1)
   i = contents.find('>', i+1)
   # Find the first '<' after the two '>'
   j = contents.find('<', i+1)
   title = contents[i+2:j]
   print(title)
```

Damit bekommen Sie Ihre Produktliste. Wenn Sie sich wirklich mit solchen Dingen auseinandersetzen möchten, empfehle ich Ihnen, im Internet nach »Reguläre Ausdrücke in Python« zu suchen. Reguläre Ausdrücke sind beinahe eine eigene Sprache. Man verwendet sie, um komplexe Suchen und Textprüfungen durchzuführen. Sie zu erlernen ist nicht einfach, aber sie können Aufgaben wie diese extrem erleichtern.

Was wir hier getan haben, wird Web-Scraping genannt und ist aus verschiedenen Gründen nicht ideal. Zuerst einmal mögen es Unternehmen nicht, wenn andere ihre Webseiten mit automatisierten Programmen durchforsten. Sie könnten eine Verwarnung bekommen oder sogar vom Besuch der Seite ausgeschlossen werden.

Zweitens ist die Abfrage immer von der Struktur der Website abhängig. Eine kleine Änderung auf der Website kann dazu führen, dass nichts mehr geht. Ein besserer Ansatz besteht darin, einen offiziellen Webdienst zu finden, der die Daten bereitstellt. Statt die Daten als HTML zurückzugeben, bieten diese Dienste vorverarbeitete Daten, oft im Format XML oder JSON.

Möchten Sie Näheres darüber lernen, suchen Sie im Internet nach »Web Services in Python«.

6.4 Zusammenfassung

Dieses Kapitel hat Ihnen die Grundlagen zum Umgang mit Dateien und zum Zugreifen auf Webseiten mit Python gezeigt. Es gibt wesentlich mehr Dinge, die Python mit dem Internet anstellen kann: Zugriff auf E-Mails und andere Internetprotokolle. Weitere Informationen darüber finden Sie in der Python-Dokumentation unter *http://docs.python.org/release/3.1.5/library/internet.html*.

7 Grafische Benutzerschnittstellen

Alles, was wir bis jetzt getan haben, war textgestützt. Unser Hangman-Spiel hätte auch auf einem Homecomputer der 80er-Jahre nicht anders ausgesehen. In diesem Kapitel erfahren Sie jedoch, wie Sie Anwendungen mit einer grafischen Benutzeroberfläche (Graphical User Interface, GUI) schreiben können.

7.1 Tkinter

Tkinter ist die Python-Schnittstelle zum GUI-System Tk, für das es auch Schnittstellen in vielen anderen Sprachen gibt und das auf so ziemlich jedem Betriebssystem läuft, darunter Linux. Tkinter ist im Lieferumfang von Python enthalten, Sie müssen es also nicht eigens installieren. Dies ist der am häufigsten genutzte Weg, in Python eine grafische Benutzeroberfläche zu erstellen.

7.2 Hello World

Die Tradition verlangt, dass das erste Programm, das Sie in einer neuen Sprache oder mit einem neuen System schreiben, irgendetwas Triviales tun soll, nur um zu zeigen, dass es funktioniert. Dazu gibt das Programm meistens die Meldung »Hello World!« aus. In Kapitel 3 haben wir das bereits für Python getan, deshalb beginne ich nun ohne großes Hin und Her mit dem folgenden Programm:

```
#07_01_hello.py

from tkinter import *
root = Tk()
Label(root, text='Hello World').pack()
root.mainloop()
```

Bild 7.1: Hello World in Tkinter

Bild 7.1 zeigt die nicht sehr eindrucksvolle Anwendung.

Wie das alles genau funktioniert, muss Sie jetzt nicht kümmern. Sie müssen allerdings wissen, dass Sie dem Objekt `Tk` eine Variable zuweisen müssen. Hier nennen wir sie `root`, was allgemein üblich ist. Anschließend erstellen wir eine Instanz der Klasse `Label` mit `root` als erstem Argument. Dadurch erfährt Tkinter, dass das Label zu ihm gehört. Das zweite Argument nennt den Text, der in dem Label angezeigt werden soll. Abschließend wird die Methode `pack` für das Label aufgerufen, die das Label anweist, sich selbst in den verfügbaren Platz einzupassen. Die Methode `pack` legt das Layout der Elemente im Fenster fest. Später werden wir eine andere Art von Layout für die Elemente in einem Raster verwenden.

7.3 Ein Temperaturumrechner

Um erste Erfahrungen mit Tkinter zu sammeln, werden wir nach und nach eine einfache Anwendung schreiben, die eine grafische Benutzerschnittstelle für die Umrechnung von Temperaturwerten bereitstellt (siehe Bild 7.2). Für die Berechnungen greift diese Anwendung auf das Modul `converter` aus Kapitel 5 zurück.

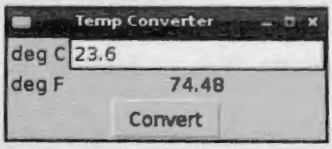


Bild 7.2: Eine Anwendung zur Umrechnung von Temperaturwerten

Unsere Hello-World-Anwendung ist nicht nur einfach, sondern auch schlecht strukturiert und damit als Grundlage für ein anspruchsvolleres Beispiel ungeeignet. Beim Erstellen von grafischen Benutzeroberflächen mit Tkinter ist es üblich, eine Klasse für jedes Anwendungsfenster anzulegen. Unser erster Schritt besteht also darin, ein Framework zu bilden, in das wir die Anwendung einhängen können, wobei wir mit einem Fenster namens »Temp Converter« und einem einzigen Label beginnen:

```
#07_02_temp_framework.py

from tkinter import *

class App:

    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        Label(frame, text='deg C').grid(row=0, column=0)
        button = Button(frame, text='Convert', command=self.convert)
        button.grid(row=1)
```

```
    def convert(self):
        print('Not implemented')

root = Tk()
root.wm_title('Temp Converter')
app = App(root)
root.mainloop()
```

Wir haben dem Programm die Klasse `App` hinzugefügt, deren Methode `__init__` verwendet wird, wenn wir in der folgenden Zeile eine neue Instanz von `App` bilden:

```
app = App(root)
```

Damit übergeben wir das `Tk`-Objekt `root` an `__init__`, wo die Benutzerschnittstelle erstellt wird.

Wie im Hello-World-Beispiel verwenden wir ein Label, das wir diesmal aber nicht dem `Tk`-Objekt `root` hinzufügen, sondern einem `Frame`-Objekt, das das Label und die anderen Elemente enthält, aus denen das Fenster unserer Anwendung besteht. Die Struktur der Benutzerschnittstelle sehen Sie in Bild 7.3. Am Ende wird sie alle dargestellten Elemente enthalten.

Der Frame wird in `root` untergebracht, aber dieses Mal verwenden wir zum Hinzufügen des Labels `grid` statt `pack`, denn dadurch können wir ein Rasterlayout für die einzelnen Teile unserer Benutzeroberfläche verwenden. Das Label nimmt die Position `0, 0` im Raster ein, das Schaltflächenobjekt, das in der folgenden Zeile erstellt wird, gelangt in die zweite Zeile (Zeile 1). Die Definition der Schaltfläche enthält auch einen »Befehl«, der bei einem Klick auf die Schaltfläche ausgeführt wird. Zurzeit ist dies jedoch nur ein Stub, der die Meldung »Not implemented« ausgibt.

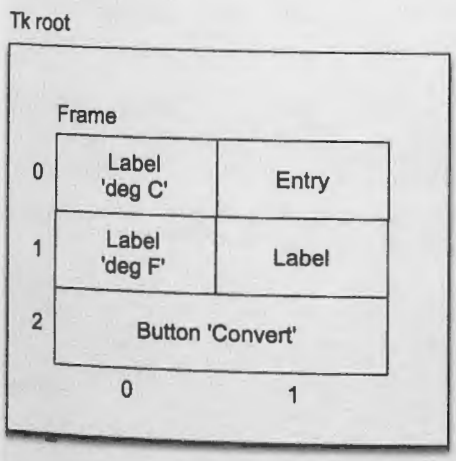


Bild 7.3: Struktur der Benutzeroberfläche

Die Funktion `wm_title` legt den Fenstertitel fest. Bild 7.4 zeigt, wie die grundlegende Benutzeroberfläche jetzt aussieht.

Bild 7.4: Die grundlegende Benutzeroberfläche für den Temperaturumrechner

Der nächste Schritt besteht darin, den Rest der Benutzeroberfläche auszufüllen. Wir brauchen ein Eingabefeld (`Entry`) für die Temperaturwerte in Celsius sowie zwei weitere Labels. Das eine davon zeigt unveränderlich `deg F` an, das andere, das sich rechts daneben befindet, die in Fahrenheit umgerechnete Temperatur.

Zur Verknüpfung der Felder auf der Benutzeroberfläche mit Werten hat Tkinter eine besondere Vorgehensweise. Wenn wir die im Eingabefeld eingegebenen Werte abrufen oder die auf dem Label angezeigten Werte festlegen wollen, müssen wir eine Instanz eines besonderen Variablenobjekts erstellen. Davon gibt es verschiedene Spielarten, von denen `StringVar` am gebräuchlichsten ist. Da wir jedoch Zahlen eingeben und anzeigen, verwenden wir `DoubleVar`. **Double** steht hier für eine Fließkommazahl mit doppelter Genauigkeit. Sie verhält sich wie ein Fließkommawert, ist aber präziser.

Nachdem wir damit den Rest der Steuerelemente für die Benutzeroberfläche und der Variablen für die Interaktion hinzugefügt haben, sieht das Programm wie folgt aus:

```
#07_03_temp_ui.py

from tkinter import *

class App:

    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        Label(frame, text='deg C').grid(row=0, column=0)
        self.c_var = DoubleVar()
        Entry(frame, textvariable=self.c_var).grid(row=0, column=1)
        Label(frame, text='deg F').grid(row=1, column=0)
        self.result_var = DoubleVar()
        Label(frame, textvariable=self.result_var).grid(row=1, column=1)
        button = Button(frame, text='Convert', command=self.convert)
        button.grid(row=2, columnspan=2)

    def convert(self):
        print('Not implemented')
```

```
root = Tk()
root.wm_title('Temp Converter')
app = App(root)
root.mainloop()
```

Das erste `DoubleVar`-Objekt (`c_var`) wird dem Eingabefeld als `textvariable`-Eigenschaft zugewiesen. Das bedeutet, dass das Eingabefeld das anzeigt, was in dem `DoubleVar` enthalten ist. Wenn sich der Wert des `DoubleVar` ändert, wird das Feld automatisch aktualisiert, um den neuen Wert darzustellen. Der Wert von `DoubleVar` ändert sich, wenn der Benutzer etwas in das Eingabefeld schreibt. Beachten Sie auch, dass wir jetzt das neue Label `deg F` hinzugefügt haben.

Das zweite `DoubleVar`-Objekt ist mit einem anderen Label verknüpft, auf dem das Ergebnis der Berechnung angezeigt wird. Zu dem Befehl `grid` haben wir einen weiteren Befehl hinzugefügt, der das Layout für die Schaltfläche ändert: Durch die Angabe `columnspan=2` erstreckt sich die Schaltfläche über beide Spalten.

Wenn Sie das Programm ausführen, zeigt es zwar die endgültige Benutzeroberfläche an, doch wenn Sie auf **Convert** klicken, wird lediglich die Meldung `Not implemented` auf der Python-Konsole ausgegeben.

Der letzte Schritt besteht also darin, die Stub-Methode `convert` durch eine echte Methode zu ersetzen, die das Modul `converters` aus Kapitel 5 nutzt. Dazu müssen wir das Modul importieren. Um uns Schreibarbeit zu ersparen, importieren wir alles wie folgt:

```
from converters import *
```

Es ist wirtschaftlicher, nur einen einzigen »Konverter« in `__init__` zu erstellen und ihn dann bei jedem Klick auf die Schaltfläche zu verwenden. Daher legen wir die Variable `self.t_conv` an, um auf den Konverter zu verweisen. Damit erhalten wir die Methode `convert`:

```
def convert(self):
    c = self.c_var.get()
    self.result_var.set(self.t_conv.convert(c))
```

Das vollständige Programm sieht wie folgt aus:

```
#07_04_temp_final.py

from tkinter import *
from converters import *

class App:
```

```
    def __init__(self, master):
        self.t_conv = ScaleAndOffsetConverter('C', 'F', 1.8, 32)
        frame = Frame(master)
        frame.pack()
        Label(frame, text='deg C').grid(row=0, column=0)
        self.c_var = DoubleVar()
        Entry(frame, textvariable=self.c_var).grid(row=0, column=1)
        Label(frame, text='deg F').grid(row=1, column=0)
        self.result_var = DoubleVar()
        Label(frame, textvariable=self.result_var).grid(row=1, column=1)
        button = Button(frame, text='Convert', command=self.convert)
        button.grid(row=2, columnspan=2)

    def convert(self):
        c = self.c_var.get()
        self.result_var.set(self.t_conv.convert(c))

root = Tk()
root.wm_title('Temp Converter')
app = App(root)
root.mainloop()
```

7.4 Weitere GUI-Widgets

Für den Temperaturumrechner haben wir nur feste Textfelder (Klasse `Entry`) und Labels (Klasse `Label`) verwendet. Sie können jedoch auch viele andere Arten von Steuerelementen in Ihre Anwendungen einbauen. Bild 7.5 zeigt den Hauptbildschirm einer Sammelsurium-Anwendung (»Kitchen Sink«, weil sich darin wie in einer Küchenspüle alles sammelt), der die meisten in Tkinter verfügbaren Steuerelemente vorführt. Das Programm finden Sie in **07_05_kitchen_sink.py**.

Bild 7.5: Eine Sammelsurium-Anwendung

7.4.1 Kontrollkästchen (Checkbutton)

Das Widget für Kontrollkästchen (in Bild 7.5 das zweite von oben auf der linken Seite) wird wie folgt erstellt:

```
Checkbutton(frame, text='Checkbutton')
```

Diese Codezeile legt einfach nur ein Kontrollkästchen mit einer Beschriftung an. Wenn wir uns aber schon die Mühe machen, ein Kontrollkästchen in ein Fenster einzubauen, wollen wir natürlich auch herausfinden können, ob es angeklickt wurde oder nicht.

Dazu verwenden wir wie in dem Beispiel mit dem Temperaturumwandler eine besondere Variable. Hier nehmen wir eine Stringvariable (`StringVar`), aber wenn die Werte von `onvalue` und `offvalue` Zahlen wären, würden wir stattdessen eine Integervariable (`IntVar`) verwenden.

```
check_var = StringVar()
check = Checkbutton(frame, text='Checkbutton',
                    variable=check_var, onvalue='Y', offvalue='N')
check.grid(row=1, column=0)
```

7.4.2 Listenfeld (Listbox)

Um eine Liste von Einträgen anzuzeigen, aus denen der Benutzer einen oder mehrere auswählen kann, brauchen wir ein Listenfeld (in Bild 7.5 in der Mitte dargestellt). Betrachten Sie dazu das folgende Beispiel:

```
listbox = Listbox(frame, height=3, selectmode=BROWSE)
for item in ['red', 'green', 'blue', 'yellow', 'pink']:
  listbox.insert(END, item)
listbox.grid(row=1, column=1)
```

Hiermit wird einfach eine Liste von Farben angezeigt. Sie müssen jeden String einzeln der Liste hinzufügen. Mit `END` legen Sie fest, dass der jeweilige Eintrag ans Ende der Liste rücken soll.

Welche Möglichkeiten die Benutzer haben, um in dem Listenfeld eine Auswahl zu treffen, können Sie mit der Eigenschaft `selectmode` steuern, die folgende Werte annehmen kann:

- `SINGLE` Es kann immer nur ein Eintrag auf einmal ausgewählt werden.
- `BROWSE` Bietet die gleichen Möglichkeiten wie `SINGLE`, erlaubt aber die Auswahl mithilfe der Maus. In Tkinter auf dem Pi ist diese Einstellung nicht von `SINGLE` zu unterscheiden.

- `MULTIPLE` Der Benutzer kann bei gedrückter [Umschalt]-Taste mehr als ein Element auswählen.
- `EXTENDED` Bietet die Möglichkeiten von `MULTIPLE` und lässt darüber hinaus zu, dass der Benutzer einen Bereich auswählt, indem er bei gedrückter [Strg]- und [Umschalt]-Taste klickt.

Anders als bei anderen Widgets, bei denen Werte mithilfe von `StringVar` oder ähnlichen Sondervariablen eingegeben und abgerufen werden, müssen Sie bei einem Listenfeld die Methode `curselection` verwenden, um herauszufinden, welche Elemente ausgewählt wurden. Diese Methode gibt eine Sammlung von Auswahlindizes zurück. Wurden beispielsweise der erste, zweite und vierte Eintrag der Liste ausgewählt, erhalten Sie folgendes Ergebnis:

```
[0, 1, 3]
```

Haben Sie `selectmode` auf `SINGLE` eingestellt, wird ebenfalls eine Liste zurückgegeben, die allerdings nur einen Wert enthält.

7.4.3 Drehfeld (Spinbox)

Drehfelder bieten eine weitere Möglichkeit, über die der Benutzer einen einzelnen Eintrag aus einer Liste auswählen kann.

```
Spinbox(frame, values=('a','b','c')).grid(row=3)
```

Die Methode `get` gibt das zurzeit in der Liste angezeigte Element zurück, nicht dessen Auswahlindex.

7.4.4 Layouts

Eine der kniffligsten Aufgaben bei der Gestaltung einer grafischen Benutzeroberfläche besteht darin, die einzelnen Bestandteile so anzuordnen, dass sich auch nach einer Veränderung der Fenstergröße ein ausgewogenes Erscheinungsbild ergibt.

Sehr häufig werden Sie Layouts ineinander verschachteln. So weist beispielsweise die Sammelsurium-Anwendung ein 3-x-3-Raster auf, in dem sich aber wiederum ein Frame für die beiden Optionsschalter befindet:

```
radio_frame = Frame(frame)
radio_selection = StringVar()

b1 = Radiobutton(radio_frame, text='portrait', variable=radio_selection,
                 value='P')
b1.pack(side=LEFT)
b2 = Radiobutton(radio_frame, text='landscape', variable=radio_selection,
```

```
                value='L')
b2.pack(side=LEFT)
radio_frame.grid(row=1, column=2)
```

Das ist eine übliche Vorgehensweise. Es ist hilfreich, die Anordnung der Steuerelemente auf Papier zu skizzieren, bevor Sie anfangen, den Code zu schreiben.

Ein besonderes Problem, dem Sie sich bei der Gestaltung von grafischen Benutzeroberflächen gegenübersehen, besteht darin, zu steuern, was bei einer Größenänderung des Fensters geschieht. Dafür müssen meistens einige Widgets in unveränderter Größe an Ort und Stelle verbleiben, während andere ausgedehnt werden. Betrachten Sie als Beispiel das einfache Fenster in Bild 7.6: Das Listenfeld auf der linken Seite behält seine Größe bei, während sich der Meldungsbereich auf der rechten Seite an Änderungen der Fenstergröße anpasst und entsprechend zusammenschrumpft oder sich ausdehnt.

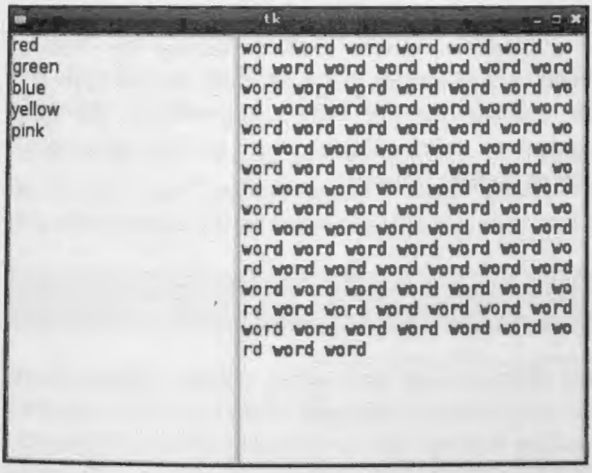


Bild 7.6: Beispiel für die Größenveränderung eines Fensters

Der Code dafür lautet wie folgt:

```
#07_06_resizing.py

from tkinter import *

class App:

   def __init__(self, master):
      frame = Frame(master)
      frame.pack(fill=BOTH, expand=1)
      #Listbox
      listbox = Listbox(frame)
      for item in ['red', 'green', 'blue', 'yellow', 'pink']:
```

```
            listbox.insert(END, item)
        listbox.grid(row=0, column=0, sticky=W+E+N+S)

        #Message
        text = Text(frame, relief=SUNKEN)
        text.grid(row=0, column=1, sticky=W+E+N+S)
        text.insert(END, 'word ' * 100)
        frame.columnconfigure(1, weight=1)
        frame.rowconfigure(0, weight=1)
root = Tk()
app = App(root)
root.geometry("400x300+0+0")
root.mainloop()
```

Dreh- und Angelpunkt solcher Layouts ist das Attribut `sticky`, mit dem Sie festlegen, welche Seiten einer Komponente an der Rasterzelle »kleben« bleiben. Um zu bestimmen, welche Spalten und Zeilen bei einer Größenänderung des Fensters ausgedehnt oder geschrumpft werden, verwenden Sie die Befehle `columnconfigure` und `rowconfigure`. Bild 7.7 zeigt die Anordnung der GUI-Komponenten, die unser Beispielfenster bilden. Die Linien geben an, welche Kanten einer solchen Komponente am Rand der umgebenden Zelle hängen bleiben müssen.

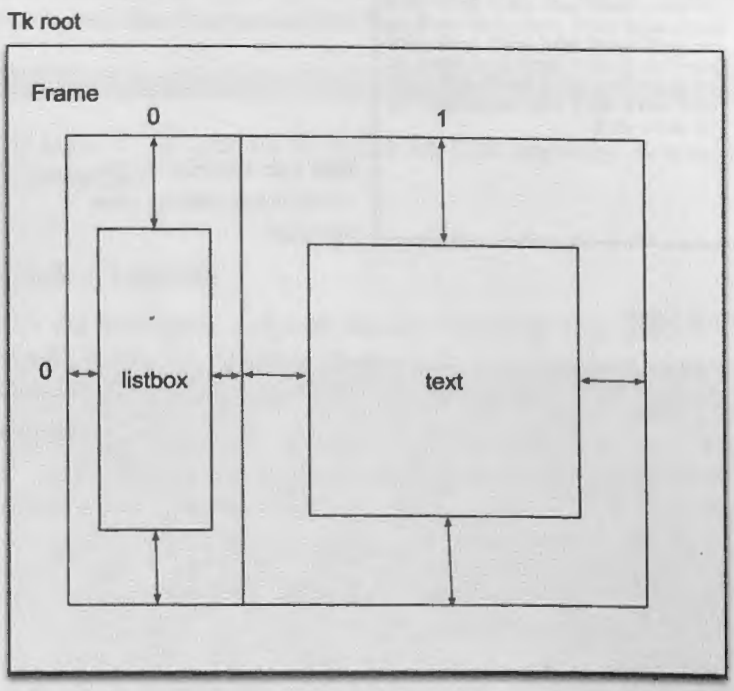


Bild 7.7: Layout für das Beispielfenster

Wir wollen diesen Beispielcode Schritt für Schritt durchgehen, um deutlich zu machen, was hier geschieht. Als Erstes betrachten wir die folgende Zeile:

```
frame.pack(fill=BOTH, expand=1)
```

Sie stellt sicher, dass der Frame das umgebende `root`-Fenster ausfüllt, sodass er jeder Größenänderung dieses Fensters folgt.

Nachdem wir das Listenfeld erstellt haben, fügen wir es mit der folgenden Zeile zum Rasterlayout des Frames hinzu:

```
listbox.grid(row=0, column=0, sticky=W+E+N+S)
```

Damit legen wir fest, dass das Listenfeld in Zeile 0, Spalte 0 gesetzt werden soll, wobei das Attribut `sticky` angibt, dass seine West-, Ost-, Nord- und Südseite am umgebenden Raster hängen bleiben soll. `W`, `E`, `N` und `S` sind numerische Konstanten, die Sie in beliebiger Reihenfolge angeben können. Auf die gleiche Weise fügen wir das Widget `Text` dem Frame des Rasters hinzu, wobei wir den Inhalt initialisieren, indem wir das Wort `word` 100-mal wiederholen lassen.

Die letzte Aufgabe besteht jetzt darin, dafür zu sorgen, dass sich bei einer Vergrößerung des Fensters der Textbereich nach rechts ausdehnt, das Listenfeld aber nicht. Dazu verwenden wir die Funktionen `columnconfigure` und `rowconfigure`:

```
frame.columnconfigure(1, weight=1)
frame.rowconfigure(0, weight=1)
```

Standardmäßig werden Zeilen und Spalten nicht ausgedehnt, wenn die umgebenden Elemente der Schnittstelle vergrößert werden. Da das genau das ist, was wir für das Listenfeld brauchen, müssen wir für Spalte 0 nichts tun. Allerdings soll sich Spalte 1 nach rechts ausdehnen können und Zeile 0 (die einzige Zeile) nach unten. Dazu geben wir ihnen mithilfe der Methoden `columnconfigure` und `rowconfigure` ein »Gewicht«. Wenn sich mehrere Spalten gleichermaßen ausdehnen sollen, müssen wir ihnen auch das gleiche Gewicht verleihen (üblicherweise `1`). Soll sich jedoch eine Spalte doppelt so weit ausdehnen wie eine andere, muss sie auch das doppelte Gewicht bekommen. Da wir in unserem Beispiel nur eine Spalte und eine Zeile haben, die sich ausdehnen sollen, können wir beiden das Gewicht `1` geben.

7.4.5 Rollbalken (Scrollbar)

Wenn Sie das Fenster aus dem Programm **07_06_resizing.py** verkleinern, werden Sie feststellen, dass keine Rollbalken angezeigt werden, um auf den verborgenen Text zugreifen zu können. Sie können diesen Text zwar immer noch erreichen, allerdings würde ein Rollbalken die Sache deutlich vereinfachen.

Rollbalken sind eigenständige Widgets. Um einen Rollbalken zusammen mit einem Text-, Meldungs- oder Listenfeld-Widget einsetzen zu können, müssen Sie beide nebeneinander anordnen und dann verknüpfen.

Bild 7.8 zeigt ein Text-Widget mit Rollbalken.

Der Code dafür lautet wie folgt:

```
#07_07_scrolling.py

from tkinter import *

class App:

   def __init__(self, master):
      scrollbar = Scrollbar(master)
      scrollbar.pack(side=RIGHT, fill=Y)
      text = Text(master, yscrollcommand=scrollbar.set)
      text.pack(side=LEFT, fill=BOTH)
      text.insert(END, 'word ' * 1000)
      scrollbar.config(command=text.yview)

root = Tk()
root.wm_title('Scrolling')
app = App(root)
root.mainloop()
```

Bild 7.8: Ein Text-Widget mit Rollbalken

In diesem Beispiel platzieren wir den Rollbalken rechts und das Textfeld links im Layout `pack`. Das Attribut `fill` gibt an, dass das Text-Widget den gesamten freien Platz sowohl in x- als auch in y-Richtung nutzen darf.

Um den Rollbalken mit dem Text-Widget zu verknüpfen, setzen wir die Eigenschaft `yscrollcommand` des Textfelds auf die `set`-Methode des Rollbalkens und das Attribut `command` des Rollbalkens auf `text.yview`.

7.5 Dialogfelder

In manchen Situationen ist es angebracht, ein kleines Fenster mit einer Meldung einzublenden, in dem der Benutzer auf OK klicken muss, bevor er irgendetwas anderes tun kann (siehe Bild 7.9). Solche Fenster werden als **modale Dialogfelder** bezeichnet. Im Paket `tkinter.messagebox` bietet Tkinter einige Varianten davon an.

Bild 7.9: Eine eingeblendete Meldung

Das folgende Beispiel zeigt, wie Sie eine solche Meldung einblenden. Neben `showinfo` umfasst `tkinter.messagebox` auch die Funktionen `showwarning` und `showerror`, die genauso funktionieren, aber jeweils ein anderes Symbol im Fenster anzeigen.

```
#07_08_gen_dialogs.py

from tkinter import *
import tkinter.messagebox as mb

class App:

   def __init__(self, master):
      b=Button(master, text='Press Me', command=self.info).pack()

   def info(self):
      mb.showinfo('Information', "Please don't press that button again!")

root = Tk()
app = App(root)
root.mainloop()
```

In den Paketen `tkinter.colorchooser` und `tkinter.filedialog` finden Sie weitere Arten von Dialogfeldern.

7.5.1 Farbwähler

Der Farbwähler gibt eine Farbe in Form ihrer einzelnen RGB-Komponenten und eines hexadezimalen Farbstrings zurück (siehe Bild 7.10).

```
#07_09_color_chooser.py

from tkinter import *
import tkinter.colorchooser as cc

class App:

   def __init__(self, master):
      b=Button(master, text='Color..', command=self.ask_color).pack()

   def ask_color(self):
      (rgb, hx) = cc.askcolor()
      print("rgb=" + str(rgb) + " hx=" + hx)

root = Tk()
app = App(root)
root.mainloop()
```

Dieser Code gibt ein Ergebnis wie das folgende zurück:

```
rgb=(255.99609375, 92.359375, 116.453125) hx=#ff5c74
```

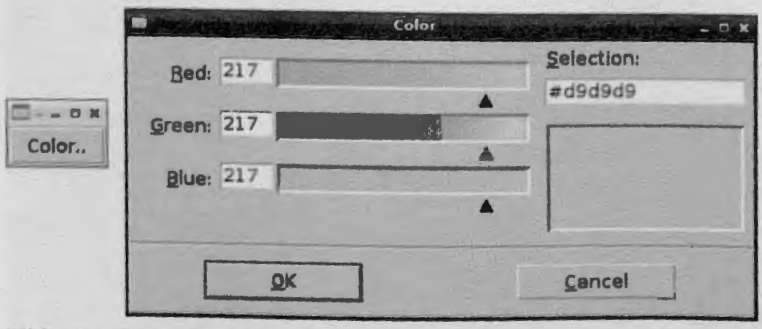


Bild 7.10: Der Farbwähler

7.5.2 Dateiwähler

Dateiwähler finden Sie im Paket `tkinter.filedialog`. Ihre Verwendung folgt dem gleichen Muster wie die der anderen betrachteten Dialogfelder.

7.6 Menüs

Ihre Anwendungen können Sie auch mit Menüs versehen. Als Beispiel sehen wir uns hier eine sehr einfache Anwendung mit einem Eingabefeld und zwei Menüs an (siehe Bild 7.11).

```
#07_10_menus.py

from tkinter import *

class App:

   def __init__(self, master):
      self.entry_text = StringVar()
      Entry(master, textvariable=self.entry_text).pack()

      menubar = Menu(root)

      filemenu = Menu(menubar, tearoff=0)
      filemenu.add_command(label='Quit', command=exit)
      menubar.add_cascade(label='File', menu=filemenu)

      editmenu = Menu(menubar, tearoff=0)
      editmenu.add_command(label='Fill', command=self.fill)
      menubar.add_cascade(label='Edit', menu=editmenu)

      master.config(menu=menubar)

   def fill(self):
      self.entry_text.set('abc')

root = Tk()
app = App(root)
root.mainloop()
```

Bild 7.11: Menüs

Der erste Schritt besteht darin, das Root-Menü zu erstellen, das alle Menüs enthält (in unserem Fall **File** und **Edit** sowie die Menüoptionen).

```
menubar = Menu(root)
```

Um das Menü **File** zu erzeugen (das nur den Menüpunkt **Quit** enthält), erstellen wir zunächst eine weitere Instanz von `Menu`, fügen ihr den Befehl `Quit` hinzu und platzieren das dann im Root-Menü:

```
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label='Quit', command=exit)
menubar.add_cascade(label='File', menu=filemenu)
```

Das Menü **Edit** wird auf die gleiche Weise erstellt. Damit die Menüs im Fenster angezeigt werden, müssen wir folgenden Befehl verwenden:

```
master.config(menu=menubar)
```

7.7 Die Zeichenfläche (Canvas)

Im nächsten Kapitel erhalten Sie eine kurze Einführung in die Spieleprogrammierung mit Pygame, womit Sie alle möglichen Arten von grafischen Effekten hervorrufen können. Wenn Sie aber nur einige Grafiken erstellen wollen, um beispielsweise Flächen oder Linien zu zeichnen, können Sie auch die Tkinter-Schnittstelle `Canvas` verwenden (siehe Bild 7.12).

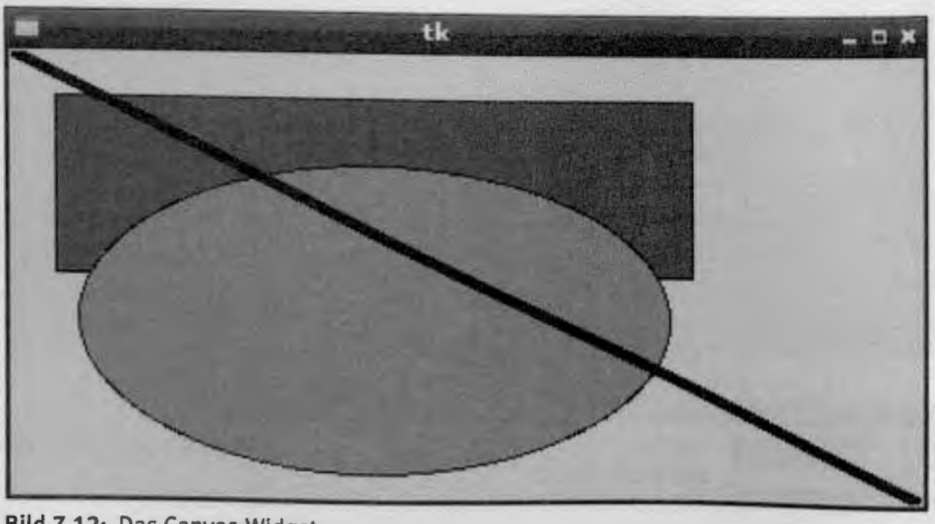


Bild 7.12: Das Canvas-Widget

`Canvas` verhält sich wie jedes andere Widget, das Sie einem Fenster hinzufügen. Das folgende Beispiel zeigt, wie Sie Rechtecke, Ovale und Linien zeichnen:

```
#07_11_canvas.py

from tkinter import *

class App:

   def __init__(self, master):
      canvas = Canvas(master, width=400, height=200)
      canvas.pack()
      canvas.create_rectangle(20, 20, 300, 100, fill='blue')
      canvas.create_oval(30, 50, 290, 190, fill='#ff2277')
      canvas.create_line(0, 0, 400, 200, fill='black', width=5)

root = Tk()
app = App(root)
root.mainloop()
```

Kurven, Bilder, Polygone und Text können Sie auf ähnliche Weise erstellen. Informationen darüber erhalten Sie online in Quellen über Tkinter wie *http://infohost.nmt.edu/tcc/help/pubs/tkinter/*.

Hinweis
Der Ursprung des Koordinatensystems ist die obere linke Ecke des Fensters, die Einheiten sind Pixel.

7.8 Zusammenfassung

In einem Buch dieses Umfangs ist es manchmal nur möglich, ein Thema anzureißen, um Ihnen einen Ausgangspunkt zu geben. Wenn Sie alle Beispiele in diesem Buch nachvollzogen, ausgeführt, verändert und analysiert haben, werden Sie sicherlich noch mehr wissen wollen. Sie müssen dann nicht mehr an die Hand genommen werden, sondern entwickeln Ihre eigenen Ideen. Kein Buch kann Ihnen genau erklären, wie Sie das Projekt, das Ihnen im Kopf herumspukt, in die Tat umsetzen können. Hier kommt dann das Internet ins Spiel. Online finden Sie unter anderem unter folgenden Adressen gute Quellen, um Ihr bisheriges Wissen zu erweitern:

- *www.pythonware.com/library/tkinter/introduction/*
- *http://infohost.nmt.edu/tcc/help/pubs/tkinter/*

8 Spieleprogrammierung

Ein einziges Kapitel reicht natürlich nicht aus, um Sie zu einem Experten in Sachen Spieleprogrammierung zu machen. Es gibt eine Reihe guter Bücher, die eigens der Spieleprogrammierung in Python gewidmet sind, etwa **Beginning Game Development with Python and Pygame** von Will McGugan. In diesem Kapitel lernen Sie die äußerst praktische Bibliothek Pygame kennen und erstellen ein einfaches Spiel.

8.1 Was ist Pygame?

Pygame ist eine Bibliothek, die es Ihnen erleichtert, Spiele für den Raspberry Pi zu schreiben – oder ganz allgemein für jeden Computer, auf dem Python läuft. Die Verwendung einer Bibliothek ist sinnvoll, da die meisten Spiele bestimmte gemeinsame Elemente aufweisen, die nicht so einfach zu schreiben sind. Eine Bibliothek wie Pygame erleichtert die Sache, da Leute, die in Python und Spieleprogrammierung sehr bewandert sind, hübsche kleine Pakete vorgefertigt haben, die wir nutzen können. Vor allem hilft uns Pygame in den folgenden Bereichen:

- Wir können Grafiken zeichnen, die nicht flackern.
- Wir können die Animation steuern, sodass sie stets mit derselben Geschwindigkeit abläuft, ob wir sie nun auf einem Raspberry Pi oder auf einem Spiele-PC der Spitzenklasse ausführen.
- Wir können Tastatur- und Mausereignisse abfangen, um den Spielverlauf zu steuern.

Im Lieferumfang der Raspbian-Wheezy-Distribution sind zwei Versionen von Python enthalten, Python 2 und 3. Darum sehen Sie auf dem Desktop auch zwei Verknüpfungen zu IDLE. In diesem Buch haben wir bisher immer IDLE 3 und damit auch Python 3 verwendet. Allerdings ist in der Python-3-Installation von Raspbian Wheezy Pygame nicht enthalten, während sie bei Python 2 vorinstalliert ist.

Anstatt Pygame in Python 3 zu installieren (was ein bisschen kompliziert ist), verwenden wir in diesem Kapitel daher Python 2. Machen Sie sich keine Sorgen, wenn Sie lieber mit Python 3 arbeiten (oder wenn Sie eine jüngere Distribution haben, in der die Bibliothek enthalten ist) – der gesamte Code, den wir schreiben, funktioniert auch in Python 3.

8.2 Hello Pygame

Möglicherweise befindet sich auch auf Ihrem Desktop die Verknüpfung **Python Games**. Damit rufen Sie ein Startprogramm auf, mit dem Sie einige Python-Spiele ausführen können. Auf jeden Fall aber gibt es im File Explorer das Verzeichnis **python_games** innerhalb des Wurzelverzeichnisses. Darin sind die .py-Dateien der Spiele untergebracht. Sie können sie in IDLE öffnen, um sich anzusehen, wie andere ihre Spiele geschrieben haben.

Bild 8.1 zeigt eine Hello-World-artige Anwendung in Pygame. Der Code dafür lautet wie folgt:

```
#08_01_hello_pygame.py

import pygame

pygame.init()

screen = pygame.display.set_mode((200, 200))
screen.fill((255, 255, 255))
pygame.display.set_caption('Hello Pygame')

ball = pygame.image.load('raspberry.jpg').convert()
screen.blit(ball, (100, 100))

pygame.display.update()
```

Bild 8.1: Hello Pygame

Das ist ein sehr primitives Beispiel, in dem es nicht einmal eine Möglichkeit gibt, das Programm auf elegante Weise zu beenden. Wenn Sie die Python-Konsole, in der Sie es gestartet haben, schließen, wird es nach einigen Sekunden abgebrochen.

Sehen Sie sich den Code dieses Beispiels genauer an, können Sie erkennen, dass wir als Erstes `pygame` importieren. Anschließend wird die Methode `init` (für initialisieren) ausgeführt, um Pygame einzurichten und betriebsbereit zu machen. Danach weisen wir die Variable `screen` wie folgt zu:

```
screen = pygame.display.set_mode((200, 200))
```

Dadurch wird ein neues Fenster von 200 x 200 Pixeln Größe erstellt. In der nächsten Zeile füllen wir es mit Weiß aus (mit der Farbe `255, 255, 255`) und legen dann den Fenstertitel »Hello Pygame« fest.

In Spielen werden Grafiken eingesetzt, und das bedeutet normalerweise, dass wir Bilddateien verwenden müssen. In diesem Beispiel binden wir in Pygame wie folgt eine Bilddatei ein:

```
raspberry = pygame.image.load('raspberry.jpg').convert()
```

Hier verwenden wir die Bilddatei **raspberry.jpg** (mit dem Motiv einer Himbeere), die zusammen mit allen Programmen aus diesem Buch im Downloadbereich der Begleitwebsite zu finden ist. Der Aufruf von `convert()` am Ende der Zeile ist wichtig, da er das Bild in eine interne Darstellung umwandelt, die sich sehr schnell zeichnen lässt. Das ist besonders dann wichtig, wenn wir das Bild im Fenster verschieben wollen.

Als Nächstes zeichnen wir das Bild der Himbeere mit dem Befehl `blit` an den Koordinaten `100, 100` auf den Bildschirm. Wie auf der Tkinter-Zeichenfläche aus dem vorherigen Kapitel liegt der Koordinatenursprung `0, 0` auch hier in der oberen linken Bildschirmecke.

Der letzte Befehl weist Pygame an, die Anzeige zu aktualisieren, damit das Bild zu sehen ist.

8.3 Das Himbeerspiel

Um Ihnen zu zeigen, wie Sie mit Pygame einfache Spiele schreiben können, werden wir nach und nach ein Spiel neu erstellen, in dem Sie fallende Himbeeren mit einem Löffel auffangen müssen. Die Himbeeren regnen mit unterschiedlicher Geschwindigkeit herab und müssen mit dem richtigen Ende des Löffels erfasst werden, bevor sie den unteren Bildschirmrand erreichen. Bild 8.2 zeigt das fertige Spiel in Aktion. Es ist schlicht, funktioniert aber, und ich hoffe, dass Sie es als Ausgangspunkt nehmen und weiter verfeinern.

Bild 8.2: Das Himbeerspiel

8.3.1 Der Mausbewegung folgen

Um mit der Entwicklung dieses Spiels zu beginnen, erstellen wir zuerst den Hauptbildschirm mit dem Löffel, der die Bewegungen der Maus nach rechts und links nachvollzieht. Laden Sie das folgende Programm in IDLE:

```
#08_02_rasp_game_mouse

import pygame
from pygame.locals import *
from sys import exit

spoon_x = 300
spoon_y = 300

pygame.init()

screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption('Raspberry Catching')

spoon = pygame.image.load('spoon.jpg').convert()
```

```
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()

    screen.fill((255, 255, 255))
    spoon_x, ignore = pygame.mouse.get_pos()
    screen.blit(spoon, (spoon_x, spoon_y))

    pygame.display.update()
```

Die Grundstruktur unseres Hello-World-Programms ist noch vorhanden, aber jetzt gibt es viele neue Elemente. Zunächst werden zusätzliche Importvorgänge durchgeführt. Der Import von `pygame.local` bietet uns Zugriff auf praktische Konstanten wie `QUIT`, mit der wir bestimmen, wann das Spiel beendet werden soll, und der Import von `exit` aus `sys` erlaubt es uns, das Programm elegant zu beenden.

Zur Speicherung der Löffelposition haben wir die beiden Variablen `spoon_x` und `spoon_y` hinzugefügt. Da sich der Löffel nur nach links und rechts bewegt, ändert sich `spoon_y` nie.

Am Ende des Programms befindet sich eine `while`-Schleife. Bei jedem Durchlauf prüfen wir, ob ein `QUIT`-Ereignis vom Pygame-System eintritt. Ereignisse treten auf, wenn der Spieler die Maus bewegt, eine Taste drückt oder loslässt. In diesem Fall sind wir nur an dem Ereignis `QUIT` interessiert, das ausgelöst wird, wenn der Spieler auf das Schließen-Symbol in der oberen rechten Ecke des Spielfensters klickt. Anstatt das Programm sofort zu beenden, könnten wir den Spieler hier auch fragen, ob er die Anwendung wirklich schließen möchte. In der nächsten Zeile wird der Bildschirm mit der Farbe Weiß gefüllt und dadurch geleert.

Anschließend folgt eine Zuweisung, in der wir `spoon_x` auf den x-Wert der Mausposition setzen. Dies ist zwar eine Doppelzuweisung, doch da uns die y-Position der Maus nicht interessiert, weisen wir den zweiten Rückgabewert der Variablen `ignore` zu, die wir dann einfach ignorieren. Anschließend zeichnen wir den Löffel auf den Bildschirm und aktualisieren die Anzeigen.

Führen Sie das Programm aus. Der Löffel sollte jetzt der Mausbewegung folgen.

8.3.2 Die erste Himbeere

Der nächste Schritt besteht darin, eine Himbeere hinzuzufügen. Später werden wir das Spiel erweitern, sodass drei Himbeeren gleichzeitig herunterfallen, aber es ist einfacher, mit einer einzigen zu beginnen. Den Code hierfür finden Sie in der Datei **08_03_rasp_game_one.py**.

Gegenüber der vorherigen Version ändert sich Folgendes:

- Es kommen globale Variablen für die Position der Himbeere hinzu (`raspberry_x` und `raspberry_y`).
- Das Bild **raspberry.jpg** wird geladen und konvertiert.
- Die Aktualisierung des Löffels wird in eine eigene Funktion ausgelagert.
- Es gibt die neue Funktion `update_raspberry`.
- Die Hauptschleife wird umgeschrieben, sodass sie die neuen Funktionen nutzt.

Mit den ersten beiden Punkten dieser Liste sind Sie bereits vertraut, weshalb wir gleich mit den neuen Funktionen loslegen:

```
def update_spoon():
    global spoon_x
    global spoon_y
    spoon_x, ignore = pygame.mouse.get_pos()
    screen.blit(spoon, (spoon_x, spoon_y))
```

Die Funktion `update_spoon` enthält lediglich den Code der Hauptschleife von `08_02_raps_game_mouse`. Dadurch bleibt die Hauptschleife klein und übersichtlich.

```
def update_raspberry():
    global raspberry_x
    global raspberry_y
    raspberry_y += 5
    if raspberry_y > spoon_y:
        raspberry_y = 0
        raspberry_x = random.randint(10, screen_width)
    raspberry_x += random.randint(-5, 5)
    if raspberry_x < 10:
        raspberry_x = 10

    if raspberry_x > screen_width - 20:
        raspberry_x = screen_width - 20
    screen.blit(raspberry, (raspberry_x, raspberry_y))
```

Die Funktion `update_raspberry` ändert die Werte von `raspberry_x` und `raspberry_y`. Damit die Himbeere nach unten fällt, wird 5 zur y-Position addiert, während der x-Wert um eine Zufallszahl zwischen –5 und +5 verändert wird. Dadurch schwanken die Himbeeren beim Abstieg auf unvorhersehbare Weise hin und her. Irgendwann sinken sie unter den unteren Bildschirmrand. Sobald der y-Wert also größer ist als der y-Wert des Löffels, stellt die Funktion sie wieder auf den oberen Rand zurück, wobei sie eine zufällige x-Position erhalten.

Es besteht die Gefahr, dass die Himbeeren jenseits des linken oder rechten Bildschirmrands verschwinden. Daher wird mit zwei weiteren Tests geprüft, ob sie sich bereits sehr nah am Rand befinden. Wenn ja, wird die Weiterbewegung in diese Richtung unterbunden.

Die neue Hauptschleife, in der diese neuen Funktionen aufgerufen werden, sieht folgendermaßen aus:

```
while True:
   for event in pygame.event.get():
      if event.type == QUIT:
         exit()

   screen.fill((255, 255, 255))
   update_raspberry()
   update_spoon()
   pygame.display.update()
```

Probieren Sie `08_03_rasp_game_one` aus. Was Sie sehen, ist ein sehr einfaches, funktionierendes Programm, das so wirkt, als könnten Sie das Spiel schon spielen. Allerdings geschieht nichts, wenn Sie eine Himbeere auffangen.

8.3.3 Erfolgreiches Auffangen erkennen und Punkte zählen

Als Nächstes fügen wir einen Meldungsbereich hinzu, in dem der Punktestand (also die Anzahl der aufgefangenen Himbeeren) angezeigt wird. Dazu müssen wir jedoch erkennen können, dass eine Himbeere aufgefangen wird. Das erweiterte Programm, das dazu in der Lage ist, finden Sie in **08_04_rasp_py_game_scoring.py**.

Die wichtigsten Änderungen in dieser Version sind die beiden neuen Funktion `check_for_catch` und `display`.

```
def check_for_catch():
   global score

if raspberry_y >= spoon_y and raspberry_x >= spoon_x and \
   raspberry_x < spoon_x + 50:
    score += 1
display("Score: " + str(score))
```

Da die `if`-Bedingung so lang ist, haben wir die Zeile hier mit dem Fortsetzungszeichen \ in zwei Zeilen aufgeteilt.

Die Funktion `check_for_catch` addiert 1 zum Punktestand (`score`), wenn die Himbeere auf Löffelhöhe heruntergefallen ist (`raspberry_y >= spoon_y`) und ihre x-Position

zwischen der x-Position des Löffels (also dessen linkem Rand) und der x-Position des Löffels plus 50 liegt (da 50 ungefähr der Länge des breiten Löffelendes entspricht).

Unabhängig davon, ob Sie die Himbeere auffangen oder nicht, wird der Punktestand mithilfe der Funktion `display` angezeigt. Die Funktion `check_for_catch` wird zur Hauptschleife hinzugefügt und bildet damit einen weiteren Aspekt, den wir bei jedem Durchlauf überprüfen müssen.

Die Funktion `display` ist dafür zuständig, eine Nachricht auf dem Bildschirm anzuzeigen:

```
def display(message):
    font = pygame.font.Font(None, 36)
    text = font.render(message, 1, (10, 10, 10))
    screen.blit(text, (0, 0))
```

Um in Pygame Text auf dem Bildschirm auszugeben, legen Sie zunächst die Schrift (`font`) fest (in diesem Fall keine besondere Schriftfamilie, sondern nur eine Größe von 36 Punkt) und erstellen dann das Objekt `text`, indem Sie den Inhalt des `message`-Strings mit dieser Schrift darstellen. Der Wert `(10, 10, 10)` ist die Textfarbe. Das Ergebnis in der Variablen `text` wird dann wie üblich mit `blit` auf den Bildschirm übertragen.

8.3.4 Zeitliche Abstimmung

Vielleicht ist Ihnen schon aufgefallen, dass es in diesem Programm keine Möglichkeit gibt, zu regeln, wie schnell die Himbeeren vom Himmel fallen. Glücklicherweise geschieht das auf dem Raspberry Pi gerade mit der richtigen Geschwindigkeit. Wenn wir das Spiel jedoch auf einem schnelleren Computer ausführen, rauschen die Früchte wahrscheinlich viel zu schnell vorbei, um sie noch auffangen zu können.

Um die Geschwindigkeit zu regeln, gibt es in Pygame eine eingebaute Uhr, mit der Sie die Hauptschleife um den geeigneten Betrag an Aktualisierungen pro Sekunde verlangsamen können. Eine Beschleunigung der Hauptschleife ist jedoch leider nicht möglich. Die Uhr lässt sich einfach verwenden. Stellen Sie lediglich die folgende Zeile irgendwo vor die Hauptschleife:

```
clock = pygame.time.Clock()
```

Dadurch wird eine Instanz der Uhr erstellt. Um die Hauptschleife um den erforderlichen Wert abzubremsen, nehmen Sie die folgende Zeile in die Schleife auf (am besten am Ende):

```
clock.tick(30)
```

Der Wert 30 steht für 30 Einzelbilder pro Sekunde. Sie können auch einen anderen Wert angeben, aber das menschliche Auge (und das Gehirn) kann bei mehr als etwa 30 Bildern pro Sekunde keine weitere Verbesserung der Qualität feststellen.

8.3.5 Viele, viele Himbeeren

Unser Programm sieht schon ein wenig verwickelter aus. Wenn wir nun die Möglichkeit hinzufügen, dass mehr als eine Himbeere herabfällt, wird es noch schwieriger, den Überblick zu behalten. Daher führen wir ein **Refactoring** durch, das heißt, wir bearbeiten ein Programm, an dem es eigentlich nichts auszusetzen gibt, und bauen seine Struktur um, ohne seine Funktion zu ändern oder irgendwelche neuen Eigenschaften hinzuzufügen. Dazu erstellen wir die Klasse `Raspberry`, die all das tut, was eine Himbeere tun soll. Das funktioniert auch im Fall einer einzigen Himbeere, erleichtert aber später vor allem den Umgang mit mehreren. Den Code dafür finden Sie in der Datei **08_05_rasp_game_refactored.py**. Die Klassendefinition sieht wie folgt aus:

```
class Raspberry:
    x = 0
    y = 0

    def __init__(self):
        self.x = random.randint(10, screen_width)
        self.y = 0

    def update(self):
        self.y += 5
        if self.y > spoon_y:
            self.y = 0
            self.x = random.randint(10, screen_width)
        self.x += random.randint(-5, 5)
        if self.x < 10:
            self.x = 10
        if self.x > screen_width - 20:
            self.x = screen_width - 20
        screen.blit(raspberry_image, (self.x, self.y))

    def is_caught(self):
        return self.y >= spoon_y and self.x >= spoon_x and \
               self.x < spoon_x + 50
```

Die Variablen `raspberry_x` und `raspberry_y` werden dabei einfach zu Variablen der neuen Klasse `Raspberry`. Wird eine neue Instanz von `Raspberry` erstellt, wird deren x-Position zufällig festgelegt. Die alte Funktion `update_raspberry` ist zu einer Methode

von `Raspberry` geworden und heißt jetzt schlicht `update`. Die Funktion `check_for_catch` fragt nun einfach die `Raspberry`-Instanz, ob sie gefangen wurde.

Nachdem wir die Klasse für Himbeeren definiert haben, können wir wie folgt eine Instanz davon erstellen:

```
r = Raspberry()
```

Um nachzuprüfen, ob die Himbeere gefangen wurde, fragt `check_for_catch` wie folgt ihre Instanz ab:

```
def check_for_catch():
    global score
    if r.is_caught():
        score += 1
```

Der Aufruf zur Anzeige des Punktestands wurde aus `check_for_catch` heraus- und in die Hauptschleife aufgenommen. Alles funktioniert noch genauso wie zuvor, aber wir haben jetzt die Gelegenheit, weitere Himbeeren hinzuzufügen. Die endgültige Version des Spiels finden Sie in der Datei **08_06_rasp_game_final.py**, den vollständigen Code zeigt das folgende Listing:

```
#08_06_rasp_game_final

import pygame
from pygame.locals import *
from sys import exit
import random

score = 0

screen_width = 600
screen_height = 400

spoon_x = 300
spoon_y = screen_height - 100

class Raspberry:
    x = 0
    y = 0
    dy = 0

    def __init__(self):
        self.x = random.randint(10, screen_width)
        self.y = 0
```

```
        self.dy = random.randint(3, 10)

    def update(self):
        self.y += self.dy
        if self.y > spoon_y:
            self.y = 0
            self.x = random.randint(10, screen_width)
        self.x += random.randint(-5, 5)
        if self.x < 10:
            self.x = 10
        if self.x > screen_width - 20:
            self.x = screen_width - 20
        screen.blit(raspberry_image, (self.x, self.y))

    def is_caught(self):
        return self.y >= spoon_y and self.x >= spoon_x
                and self.x < spoon_x + 50

clock = pygame.time.Clock()
rasps = [Raspberry(), Raspberry(), Raspberry()]

pygame.init()

screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption('Raspberry Catching')

spoon = pygame.image.load('spoon.jpg').convert()
raspberry_image = pygame.image.load('raspberry.jpg').convert()

def update_spoon():
    global spoon_x
    global spoon_y
    spoon_x, ignore = pygame.mouse.get_pos()
    screen.blit(spoon, (spoon_x, spoon_y))

def check_for_catch():
    global score
    for r in rasps:
        if r.is_caught():
            score += 1

def display(message):
    font = pygame.font.Font(None, 36)
    text = font.render(message, 1, (10, 10, 10))
    screen.blit(text, (0, 0))
```

von `Raspberry` geworden und heißt jetzt schlicht `update`. Die Funktion `check_for_catch` fragt nun einfach die `Raspberry`-Instanz, ob sie gefangen wurde.

Nachdem wir die Klasse für Himbeeren definiert haben, können wir wie folgt eine Instanz davon erstellen:

```
r = Raspberry()
```

Um nachzuprüfen, ob die Himbeere gefangen wurde, fragt `check_for_catch` wie folgt ihre Instanz ab:

```
def check_for_catch():
    global score
    if r.is_caught():
        score += 1
```

Der Aufruf zur Anzeige des Punktestands wurde aus `check_for_catch` heraus- und in die Hauptschleife aufgenommen. Alles funktioniert noch genauso wie zuvor, aber wir haben jetzt die Gelegenheit, weitere Himbeeren hinzuzufügen. Die endgültige Version des Spiels finden Sie in der Datei **08_06_rasp_game_final.py**, den vollständigen Code zeigt das folgende Listing:

```
#08_06_rasp_game_final

import pygame
from pygame.locals import *
from sys import exit
import random

score = 0

screen_width = 600
screen_height = 400

spoon_x = 300
spoon_y = screen_height - 100

class Raspberry:
    x = 0
    y = 0
    dy = 0

    def __init__(self):
        self.x = random.randint(10, screen_width)
        self.y = 0
```

```
        self.dy = random.randint(3, 10)

    def update(self):
        self.y += self.dy
        if self.y > spoon_y:
            self.y = 0
            self.x = random.randint(10, screen_width)
        self.x += random.randint(-5, 5)
        if self.x < 10:
            self.x = 10
        if self.x > screen_width - 20:
            self.x = screen_width - 20
        screen.blit(raspberry_image, (self.x, self.y))

    def is_caught(self):
        return self.y >= spoon_y and self.x >= spoon_x
                and self.x < spoon_x + 50

clock = pygame.time.Clock()
rasps = [Raspberry(), Raspberry(), Raspberry()]

pygame.init()

screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption('Raspberry Catching')

spoon = pygame.image.load('spoon.jpg').convert()
raspberry_image = pygame.image.load('raspberry.jpg').convert()

def update_spoon():
    global spoon_x
    global spoon_y
    spoon_x, ignore = pygame.mouse.get_pos()
    screen.blit(spoon, (spoon_x, spoon_y))

def check_for_catch():
    global score
    for r in rasps:
        if r.is_caught():
            score += 1

def display(message):
    font = pygame.font.Font(None, 36)
    text = font.render(message, 1, (10, 10, 10))
    screen.blit(text, (0, 0))
```

```
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()

    screen.fill((255, 255, 255))
    for r in rasps:
        r.update()
    update_spoon()
    check_for_catch()
    display("Score: " + str(score))
    pygame.display.update()
    clock.tick(30)
```

Um mehrere Himbeeren zu erstellen, wurde die einzelne Variable `r` durch die Sammlung `rasps` ersetzt:

```
rasps = [Raspberry(), Raspberry(), Raspberry()]
```

Dadurch entstehen drei Himbeeren. Wir könnten dies dynamisch ändern, während das Programm läuft, um neue Himbeeren hinzuzufügen (oder auch um einige zu entfernen).

Wenn wir es mit mehr als einer Himbeere zu tun haben, müssen wir noch zwei weitere kleine Änderungen vornehmen. Erstens müssen wir uns in der Funktion `check_for_catch` alle Himbeeren ansehen und jede daraufhin überprüfen, ob sie aufgefangen wurde. Zweitens müssen wir in der Hauptschleife alle Himbeeren anzeigen, indem wir sie eine nach der anderen durchgehen und aktualisieren.

8.4 Zusammenfassung

Es gibt noch viel mehr über Pygame zu lernen! Die offizielle Website auf *www.pygame.org* bietet diverse Quellen und Musterspiele, die Sie spielen und ändern können.

9 Hardware anschließen

An der einen Seite verfügt der Raspberry Pi über eine Doppelreihe von Kontaktstiften (Pins), die als GPIO-Anschluss bezeichnet werden (General Purpose Input/Output, also »Allzweckein-/-ausgabe«). Damit können Sie als Alternative zur USB-Buchse Elektronikhardware mit dem Pi verbinden.

Die Bastler- und die Pädagogenszene haben bereits Platinen hergestellt, die zur Erweiterung des Pi und zur Konstruktion von Prototypen dienen können. Sie können sie an Ihren Pi anschließen, um elektronische Bauteile hinzuzufügen – von einfachen Temperatursensoren bis zu Relais. Es ist damit sogar möglich, den Pi zur Steuereinheit eines Roboters zu machen.

In diesem Kapitel sehen wir uns die verschiedenen Möglichkeiten an, die Sie haben, um über den GPIO andere elektronische Geräte an den Pi anzuschließen. Dabei verwenden wir einige der ersten Produkte, die es für diesen Zweck gab. Dieser Bereich unterliegt einem raschen Wandel, daher sind seit dem Zeitpunkt, zu dem ich dieses Kapitel geschrieben habe, sicherlich schon neue Produkte auf den Markt gekommen. Schauen Sie deshalb im Internet nach, um sich über die aktuelle Palette zu informieren. Ich habe mich bemüht, eine repräsentative Auswahl der verschiedenen Möglichkeiten zum Anschließen von Hardware zu treffen. Sollte eines der besprochenen Teile in genau dieser Form nicht mehr erhältlich sein, gibt Ihnen dieses Kapitel trotzdem einen guten Eindruck davon, was es alles gibt und wie Sie es einsetzen können.

Die Produkte, die es für den Anschluss von elektronischen Bauteilen an Ihren Pi gibt, fallen entweder in die Kategorie der Erweiterungsplatinen oder der Hilfsmittel für die Konstruktion von Prototypen. Bevor wir uns diese Produkte ansehen, wollen wir uns aber genauer mit den Möglichkeiten beschäftigen, die der GPIO-Anschluss bietet.

9.1 Verbindungen mit den GPIO-Pins

Bild 9.1 zeigt die Verbindungen, die am GPIO-Anschluss des Raspberry Pi zur Verfügung stehen. Die mit GPIO markierten Pins dienen als allgemeine Ein- und Ausgabestifte, d. h., Sie können für jeden einzelnen festlegen, ob er als Eingang oder als Ausgang verwendet werden soll. Wenn Sie einen Pin zum Eingang machen, können Sie prüfen, ob er auf `1` gesetzt ist (über 1,7 V) oder auf `0` (unter 1,7 V). Beachten Sie, dass alle GPIO-Pins für 3,3 V ausgelegt sind. Wenn Sie eine höhere Spannung daran anlegen, kann das Ihren Raspberry Pi beschädigen.

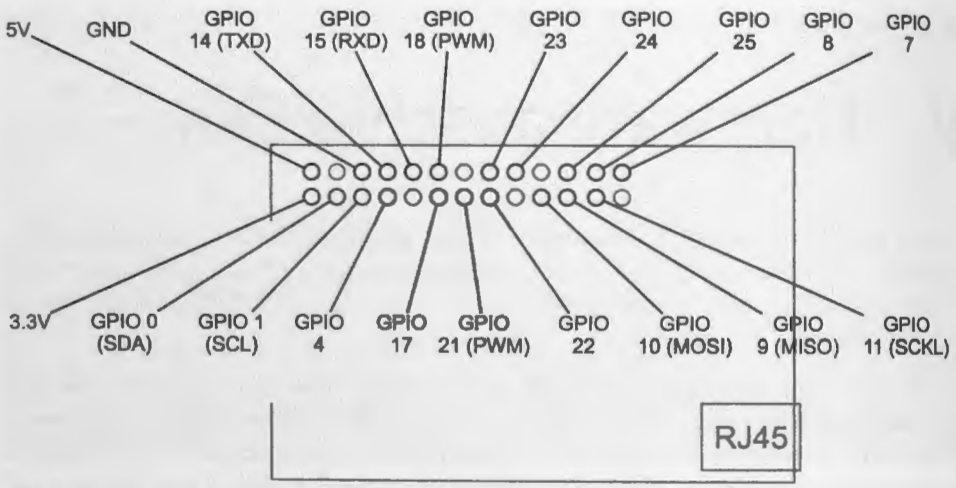


Bild 9.1: Der GPIO-Anschluss

An einem als Ausgang definierten Pin liegen entweder 0 V (logische 0) oder 3,3 V (logische 1) an. Pins können als Stromquelle oder -senke dienen, aber nur jeweils für eine geringe Stromstärke (mit 5 mA sind Sie auf der sicheren Seite). Damit können Sie gerade mal eine LED zum Leuchten bringen, wenn Sie einen hohen Widerstand (ca. 1 kΩ) verwenden. Bei einigen der GPIO-Pins stehen hinter der Nummer noch Buchstaben in Klammern. Diese Stifte dienen einem besonderen Zweck. So tragen beispielsweise GPIO 0 und 1 die zusätzlichen Bezeichnungen SDA und SCL, denn sie fungieren auch als Daten- und Uhrzeitleitungen für den I2C-Bus, der gern für die Kommunikation mit Peripheriegeräten wie Temperatursensoren, LCD-Anzeigen usw. verwendet wird. Dieser Bus wird auch von den Produkten Pi Face und Slice of PI/O genutzt, die wir in den folgenden Abschnitten noch besprechen werden.

Die GPIO-Pins 14 und 15 dienen auch als Rx- und Tx-Kontakte (Receive/Transmit, also Empfangen/Übertragen) für den seriellen Port des Raspberry Pi. Serielle Kommunikation ist auch über GPIO 9 bis 11 möglich (MISO, MOSI und SCLK). Diese Art von serieller Schnittstelle wird SPI genannt. Schließlich gibt es noch die Pins GPIO 18 und GPIO 21 mit der Bezeichnung PWM, was bedeutet, dass sie zur Pulsbreitenmodulation fähig sind. Mit dieser Technik können Sie die Leistungsabgabe an Motoren, LEDs usw. steuern, indem Sie die Breite des generierten Pulses ändern.

9.2 Direkter Anschluss an die GPIO-Pins

Wenn Sie die gebotene Vorsicht walten lassen, können Sie einfache elektronische Bauteile wie LEDs auch direkt an die GPIO-Pins anschließen. Das sollten Sie aber nur dann machen, wenn Sie wirklich genau wissen, was Sie da tun, denn es besteht die Gefahr, den Raspberry Pi zu beschädigen. Im Grunde genommen ist es aber das, was wir weiter unten im Abschnitt 9.4 »Platinen zur Prototypentwicklung« durchführen werden.

9.3 Erweiterungsplatinen

Erweiterungsplatinen weisen gewöhnlich Schraubklemmen und eine gewisse Menge bereits vormontierter elektronischer Bauteile auf. Dadurch eignen sie sich sehr gut für Unterrichtszwecke, aber auch für diejenigen, die nicht allzu tief in die elektronische Bastelei einsteigen möchten. Im Allgemeinen müssen Sie bei diesen Platinen nichts mehr löten. Gewöhnlich »puffern« sie auch alle Verbindungen zum Raspberry Pi, sodass der Pi vor schädlichen Vorgängen auf der Platine geschützt ist. Ein Kurzschluss an einem Ausgang beispielsweise kann die Erweiterungsplatine beschädigen, Ihr kostbarer Pi aber wird dadurch nicht angekratzt.

In den folgenden Abschnitten finden Sie nähere Angaben zu einigen der bekannteren Platinen, ihren Merkmalen und den Einsatzmöglichkeiten. Eine davon (das RaspiRobotBoard) verwenden wir in Kapitel 11, um einen einfachen Roboter zu bauen.

9.3.1 Pi Face

Das Pi Face aus Bild 9.2 wurde von der Universität Manchester entwickelt und ist hauptsächlich für Unterrichtszwecke gedacht. Es stellt nicht nur eine brauchbare Hardwareplattform dar, sondern bietet auch eine leicht zu nutzende Python-Bibliothek und die Integration in die Programmierumgebung Scratch.

Das Pi Face wird oben auf den Raspberry Pi aufgesetzt und bietet praktische Schraubklemmen für den Anschluss von Geräten. Die Platine nutzt die GPIO-Pins nicht direkt, sondern kommuniziert über einen MCP23S17-Porterweiterungschip mit der seriellen I2C-Schnittstelle. Damit stehen auf der Erweiterungsplatine acht Eingänge und acht Ausgänge zur Verfügung, während auf dem Raspberry Pi nur die beiden I2C-Pins des GPIO-Anschlusses belegt werden. Die Ausgänge sind über einen Darlington-Treiber-IC mit einer Stromverstärkung ausgestattet und können jeweils bis zu 500 mA liefern, was mehr als genug ist, um ein Relais oder eine 1-W-LED direkt anzutreiben.

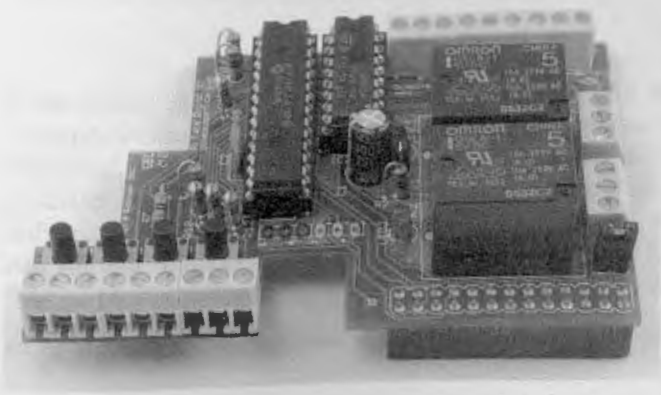


Bild 9.2:
Die Erweiterungsplatine Pi Face

Zu den Ausgabegeräten auf der Platine gehören zwei Relais, mit denen Hochlastströme geschaltet werden können. Jedes Relais verfügt auch über eine LED, die bei Bestromung des Relais aufleuchtet. Außerdem gibt es zwei LEDs, die unabhängig voneinander angesteuert werden können. Vier der Eingänge sind mit Drucktasten ausgestattet.

Für das Pi Face gibt es ein eigenes Python-Modul, das die Verwendung der Platine erleichtert. Im folgenden Beispiel sehen Sie den Code, den Sie in die Python-Konsole eingeben müssen, um den digitalen Eingang 2 auszulesen:

```
>>> import piface.pfio as pfio
>>> pfio.init()
>>> pfio.digital_read(2)
0
```

Um den digitalen Ausgang 2 einzuschalten, müssen Sie Folgendes schreiben:

```
>>> pfio.digital_write(2, 1)
```

Für die LEDs und Relais gibt es jeweils eigene Steuerfunktionen. Der folgende Beispielcode schaltet LED 1 ein, dann wieder aus und schaltet schließlich Relais 1 ein:

```
>>> led1 = pfio.LED(1)
>>> led1.turn_on()
>>> led1.turn_off()
>>> relay = pfio.Relay(1)
>>> relay.turn_on()
```

Die Bibliothek müssen Sie herunterladen und installieren. Downloads, eine Dokumentation sowie einige einfache Projekte finden Sie auf der Projektcodeseite unter *https://github.com/thomasmacpherson/piface*. Weitere Informationen über das Projekt erhalten Sie auf *http://pi.cs.man.ac.uk/interface.htm*.

9.3.2 Slice of PI/O

Das Slice of PI/O, das Sie in Bild 9.3 sehen, ist eine kleine, preisgünstige Platine mit je acht gepufferten Ein- und Ausgängen und dem gleichen MCP23S17-Porterweiterungschip, der auch auf dem Pi Face verwendet wird. Diese Platine verfügt jedoch nicht über einen Darlington-Treiber und kann daher keine hohen Lasten antreiben. Die maximale Last, die Sie direkt vom MCP23S17 beziehen können, beträgt 25 mA, was immerhin ausreicht, um eine LED mit einem geeigneten Widerstand zu versorgen, aber für den direkten Antrieb eines Relais zu wenig ist.

Alle E/A-Pins dieser Platine sind zu Anschlüssen am Rand geführt, wobei Sie jeden davon entweder als Eingang oder als Ausgang festlegen können.

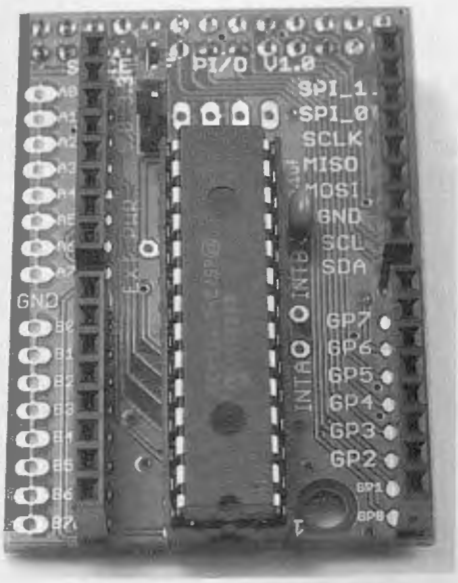


Bild 9.3: Das Slice of PI/O

Die Platine weist folgende Hauptmerkmale auf:

- 16 bidirektionale, gepufferte E/A-Verbindungen.
- Betrieb mit 3,3 V oder 5 V über Jumper einstellbar.
- Separate Verbindungen zum seriellen I2C- und SPI-Anschluss des Raspberry Pi (Vorsicht: ungepuffert!).
- Separate Verbindungen zu den GPIO-Pins 0 bis 7 des Raspberry Pi (Vorsicht: ungepuffert!).

Zurzeit gibt es für die Platine noch kein Python-Modul, der Hersteller oder die Raspberry-Pi-Community werden das aber bald ändern.

9.3.3 RaspiRobotBoard

Ich muss vorausschicken, dass ich dem RaspiRobotBoard aus Bild 9.4 nicht neutral gegenüberstehe, da ich es selbst entwickelt habe. Der Zweck dieser Platine besteht darin, den Raspberry Pi als Steuereinheit für einen Roboter zu verwenden. Daher verfügt sie über eine Motorsteuereinheit, mit der Sie die Laufrichtung der beiden Motoren (die gewöhnlich mit Rädern verbunden werden) regeln können.

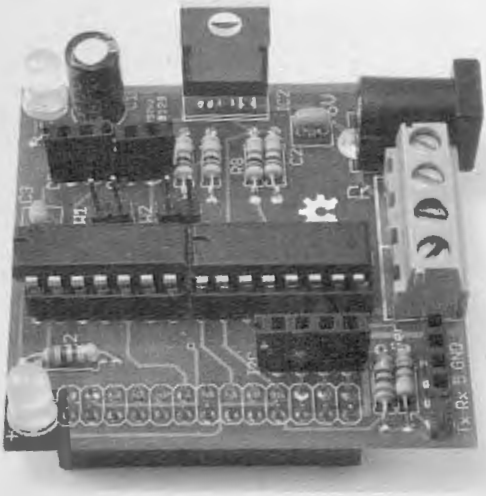


Bild 9.4: Das RaspiRobotBoard

Ein weiteres Merkmal, das diese Platine als Plattform für den Roboterbau geeignet macht, ist der Spannungsregler, über den der Raspberry Pi von jeder Stromquelle versorgt werden kann, die eine Spannung zwischen 6 und 9 V aufweist, also beispielsweise durch vier AA-Batterien. Das RaspiRobotBoard verfügt außerdem über Anschlüsse für zwei Arten von seriellen Schnittstellen, von denen die eine zur Aufnahme einer Adapterplatine für einen Ultraschall-Entfernungsmesser gedacht ist. Darüber hinaus weist die Platine ein Paar Schaltereingänge, zwei LEDs und ein Paar gepufferte Ausgänge auf, mit denen Sie andere LEDs oder Lasten mit geringer Stromstärke antreiben können. In Kapitel 11 bauen wir mithilfe dieser Platine ein kleines Roboterfahrzeug.

9.3.4 Gertboard

Das Gertboard wurde vom Broadcom-Mitarbeiter Gert van Loo entworfen und ist daher die »offiziellste« Raspberry-Pi-Erweiterungsplatine (siehe Bild 9.5).

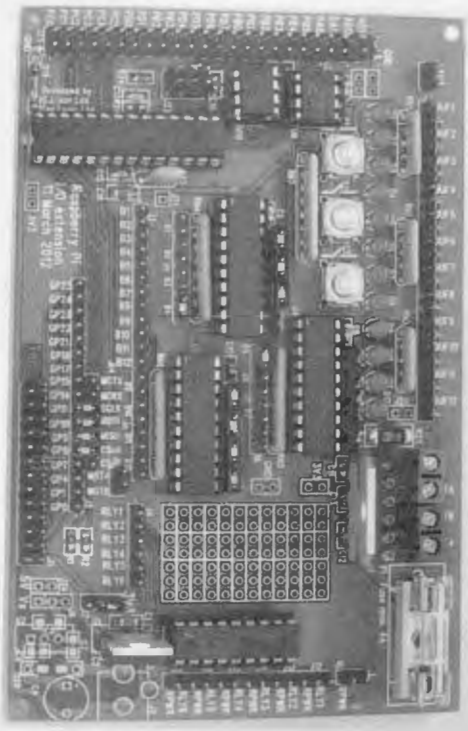


Bild 9.5: Die Gertboard-Erweiterungsplatine für den Raspberry Pi

Das Gertboard ist die Eier legende Wollmilchsau unter den Erweiterungsplatinen und weist folgende Hauptmerkmale auf:

- Einen Kopplungsbereich, in dem die GPIO-Pins mit verschiedenen Modulen verbunden werden können.
- ATmega-Mikrocontroller (wie beim Arduino).
- Analog/Digital- und Digital/Analog-SPI-Konverter.
- Motorsteuereinheit (wie auf dem RaspiRobotBoard).
- Offenen 500-mA-Ausgang (wie auf dem Pi Face).
- Zwölf LEDs und drei Drucktasten.

9.4 Platinen zur Prototypentwicklung

Anders als bei Erweiterungsplatinen brauchen Sie bei Platinen zur Prototypentwicklung meistens auch einen Lötkolben und gewisse Kenntnisse in Elektronik. Außerdem werden sie direkt an den Hauptchip des Raspberry Pi angeschlossen, was bedeutet, dass Sie bei einem Fehler Ihren Pi beschädigen können. Diese Platinen sind nur etwas für

erfahrene Hobbyelektroniker – oder für sehr vorsichtige oder für sehr rücksichtslose Naturen (die keine Angst davor haben, ihren Raspberry Pi möglicherweise zu schlachten).

Eine der Prototypplatinen, der »Cobbler«, ist in Wirklichkeit gar keine Verbindung, sondern ein Stecker, mit dem Sie die GPIO-Pins mit einer Steckplatine verbinden können, auf der Sie Ihre eigene Elektronik aufbauen. Diese Vorgehensweise sehen wir uns im nächsten Kapitel genauer an.

9.4.1 Pi Cobbler

Der Pi Cobbler von Adafruit (*www.adafruit.com/products/914*) wird als Bausatz geliefert, den Sie erst zusammenlöten müssen. Das ist aber recht einfach, und sobald Sie alles zusammengebaut haben, verfügen Sie über eine »Platine« mit 26 Pins an der Unterseite, die Sie an eine Steckplatine anschließen können (siehe Bild 9.6). Oben auf der Platine befindet sich ein 26-Pin-Sockel, über den Sie den Cobbler mithilfe des mitgelieferten Bandkabels mit dem GPIO-Anschluss des Raspberry Pi verbinden können.

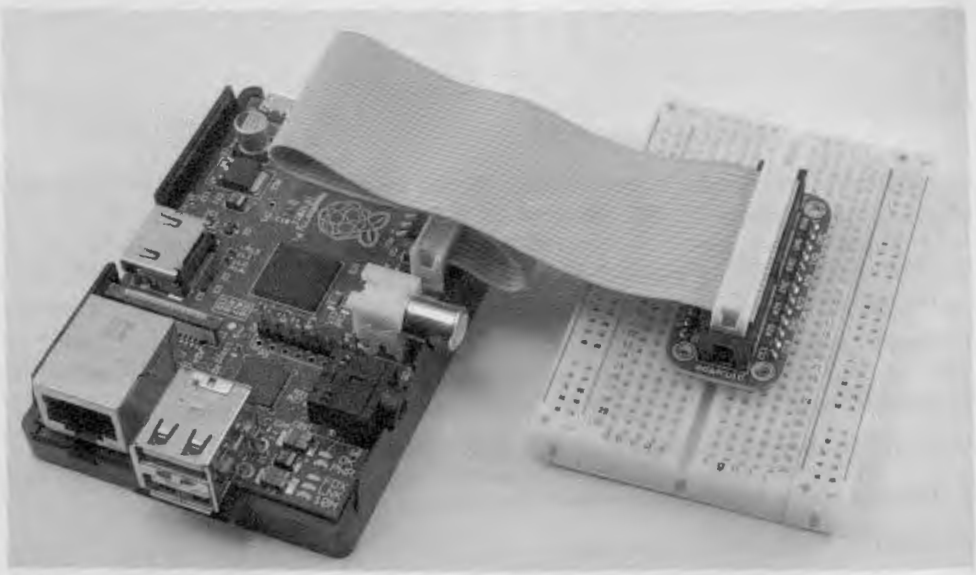


Bild 9.6: Der Pi Cobbler von Adafruit

9.4.2 Pi Plate

Die Pi Plate aus Bild 9.7 ist ein weiteres Adafruit-Produkt (*https://www.adafruit.com/products/801*). Es handelt sich um eine Prototypplatine mit einer großen freien Fläche in

der Mitte, auf der Sie die Komponenten Ihres Projekts auflöten können. An allen Rändern sind Schraubklemmen angebracht, sodass Sie Kabel zu externen Komponenten anschließen können, die nicht auf die Platine passen, beispielsweise Motoren. In einer Ecke befindet sich eine vorbereitete Fläche, um einen IC zur Oberflächenmontage aufzulöten. Die Pins daneben führen zu den nur schwer nutzbaren Pins des IC.

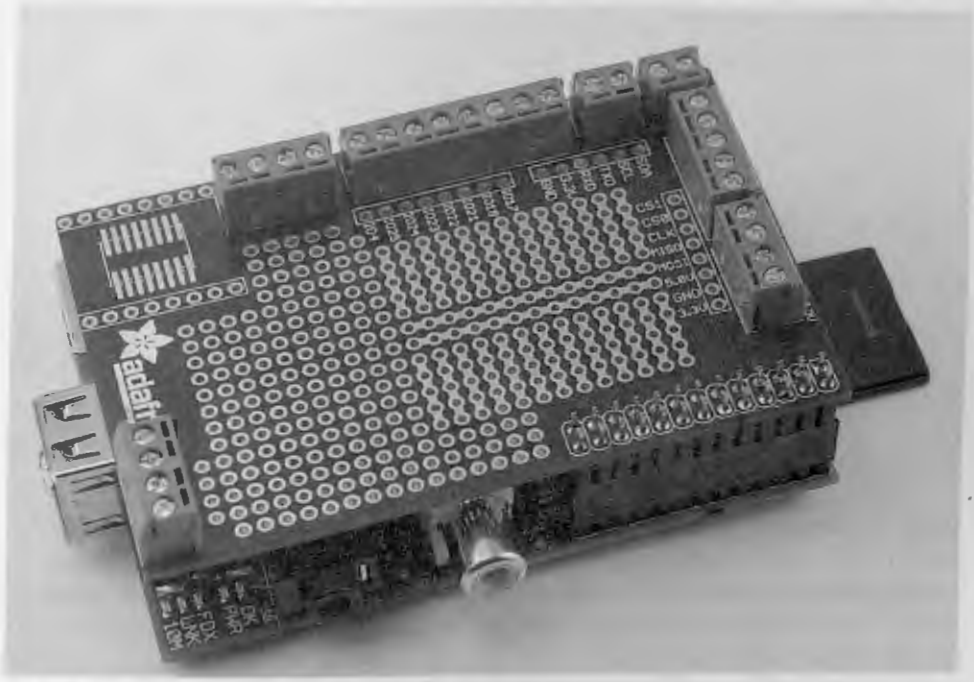


Bild 9.7: Die Pi Plate von Adafruit

9.4.3 Humble Pi

Der Humble Pi, den Sie in Bild 9.8 sehen, ist der Pi Plate sehr ähnlich, ihm fehlt aber der Bereich zur Oberflächenmontage eines IC. Dafür verfügt er über eine Fläche, auf der Sie einen eigenen Spannungsregler und eine Strombuchse anbringen können, wodurch Sie den Pi mit 5 V durch Batterien oder eine externe Stromquelle versorgen können. Ein Spannungsregler und die zugehörigen Widerstände sind im Lieferumfang nicht enthalten, allerdings bietet Ciseco einen Satz von Bauteilen an.

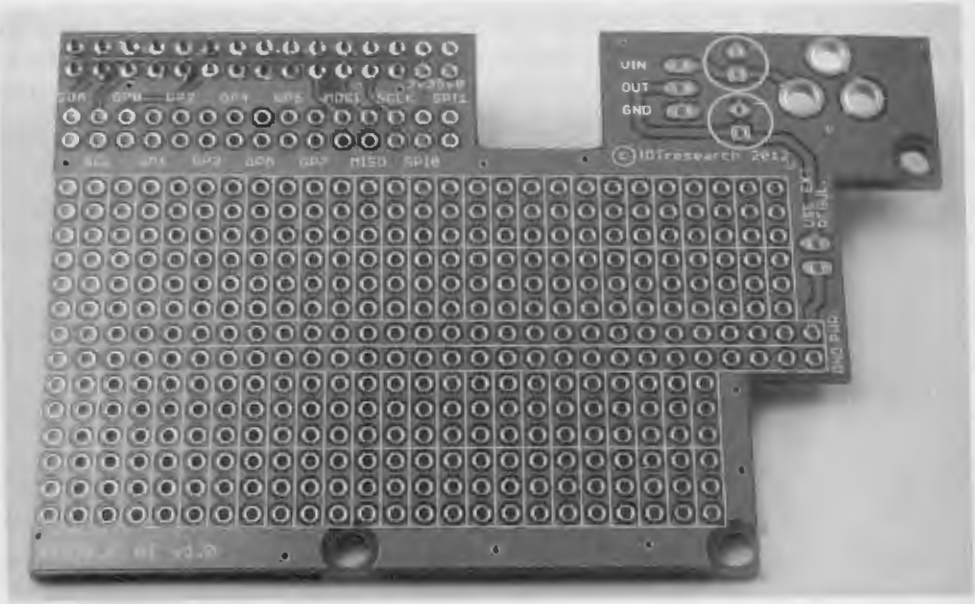


Bild 9.8: Der Humble Pi

9.5 Arduino und der Pi

Der Raspberry Pi kann zwar als Mikrocontroller zur Steuerung von Motoren und ähnlichen Geräten eingesetzt werden, aber das ist nicht das, wofür er ursprünglich entwickelt wurde. Daher liefern die GPIO-Pins auch keine ausreichend hohe Stromstärke und sind ziemlich empfindlich für Missbrauch. Aus diesem Grund wurden die im vorherigen Abschnitt beschriebenen Erweiterungsplatinen entwickelt.

Dagegen sind Arduino-Platinen weit robuster und für die Steuerung elektronischer Geräte ausgelegt. Vor allem aber verfügen sie über analoge Eingänge, die beispielsweise die von einem Temperatursensor gelieferte Spannung messen können.

Arduino-Platinen sind dafür vorgesehen, über USB mit einem Hostcomputer zu kommunizieren, und nichts hindert Sie daran, einen Raspberry Pi als Hostcomputer zu verwenden. Dabei kümmert sich der Arduino um die gesamte Elektronik, während der Raspberry Pi Befehle sendet und auf Anforderungen des Arduino wartet.

Wenn Sie einen Arduino haben, können Sie das folgende einfache Beispiel ausprobieren, in dem der Raspberry Pi eine Nachricht an den Arduino sendet, um seine eingebauten LEDs zum Blinken zu bringen, während er gleichzeitig eingehende Meldungen vom Arduino empfängt. Wenn Sie erst einmal wissen, wie das geht, können Sie entweder den Arduino-Sketch oder das Python-Programm auf dem Raspberry Pi ändern, um anspruchsvollere Aufgaben in Angriff zu nehmen.

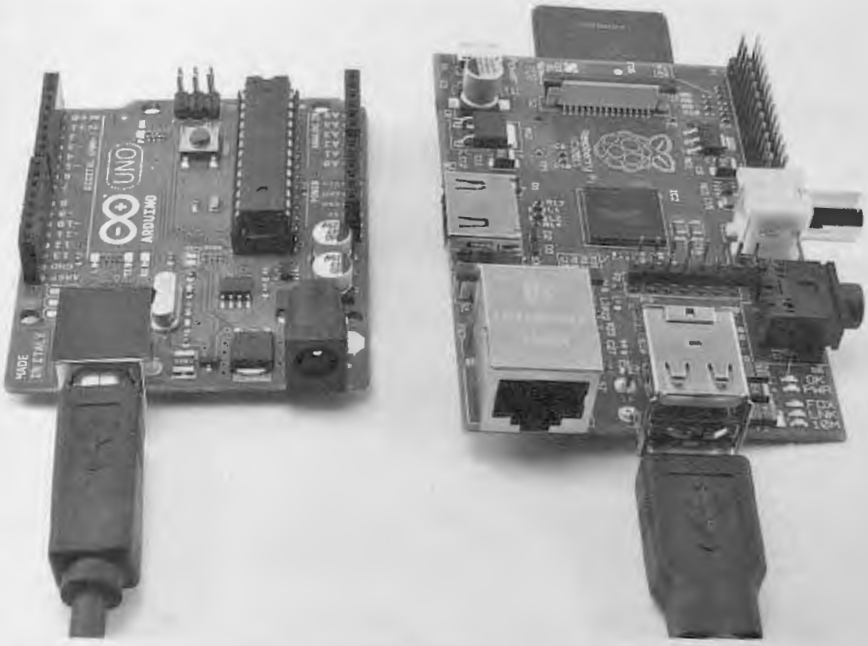


Bild 9.9: Eine Arduino-Platine an einem Raspberry Pi

Bei diesem Beispiel setzen wir voraus, dass Sie mit dem Arduino vertraut sind. Wenn nicht, sollten Sie eines meiner Bücher über den Arduino lesen, etwa **Programming Arduino: Getting Started with Sketches** oder **30 Arduino Projects for the Evil Genius.**

9.5.1 Kommunikation zwischen Arduino und Pi

Damit der Arduino und der Raspberry Pi miteinander reden können, müssen wir sie über den USB-Anschluss am Pi miteinander verbinden. Da der Arduino nur etwa 50 mA Strom braucht und in diesem Fall keine zusätzliche Elektronik an ihn angeschlossen ist, kann er vom Pi mit Strom versorgt werden.

9.5.2 Die Arduino-Software

Sie brauchen nicht mehr zu tun, als den folgenden Arduino-Sketch auf den Arduino zu laden. Dazu verwenden Sie am besten Ihren regulären Computer, da für den Raspberry Pi zurzeit nur eine sehr alte Version der Arduino-Software zur Verfügung steht. Den folgenden Sketch finden Sie unter dem Namen PiTest.ino im Downloadpaket:

```
// Pi und Arduino

const int ledPin = 13;
```

```
void setup()
{
   pinMode(ledPin, OUTPUT);
   Serial.begin(9600);
}

void loop()
{
   Serial.println("Hello Pi");
   if (Serial.available())
   {
      flash(Serial.read() - '0');
   }
   delay(1000);
}

void flash(int n)
{
   for (int i = 0; i < n; i++)
   {
      digitalWrite(ledPin, HIGH);
      delay(100);
      digitalWrite(ledPin, LOW);
      delay(100);
   }
}
```

Dieser sehr einfache Sketch enthält lediglich drei Funktionen. Die Funktion `setup` initialisiert die serielle Kommunikation und legt Pin 13 an der LED des Arduino als Ausgang fest. Die Funktion `loop` wird wiederholt aufgerufen, bis der Arduino abgeschaltet wird. Als Erstes sendet sie die Meldung `Hello Pi` an den Raspberry Pi, dann schaut sie nach, ob Nachrichten vom Pi eingegangen sind. Wenn ja (erwartet wird eine einzige Ziffer), lässt sie die LED mithilfe der Funktion `flash` entsprechend oft aufblinken.

9.5.3 Die Software für den Raspberry Pi

Der Python-Code zur Kommunikation mit dem Arduino ist sogar noch einfacher und kann leicht in die Python-Konsole eingegeben werden. Zunächst jedoch müssen Sie das Paket `PySerial` installieren, um die Kommunikation möglich zu machen. Dazu gehen Sie wie bei allen anderen Paketen vor, die wir uns bereits angesehen haben. Laden Sie als Erstes die gepackte tar-Datei von *http://sourceforge.net/projects/pyserial/files/latest/download?source=files* herunter.

Entpacken Sie dann das Verzeichnis, das sich in dem Archiv befindet, mithilfe des folgenden Befehls:

```
tar -xzf pyserial-2.5.tar.gz
```

Damit haben Sie einen neuen Ordner für das Modul. Wechseln Sie mit `cd` dorthin und führen Sie den Befehl `install` aus (zuvor sollten Sie allerdings in der Anleitung nachsehen, ob Sie irgendwelche vorbereitenden Schritte ausführen müssen):

```
cd pyserial-2.5
sudo python setup.py install
```

Nachdem Sie das Modul installiert haben, können Sie es von der Python-Shell importieren. Wechseln Sie vom Linux-Terminal zu einer Python-Konsole und geben Sie Folgendes ein:

```
import serial
ser = serial.Serial('/dev/ttyACM0', 9600)
```

Dadurch wird die serielle USB-Verbindung zum Arduino mit 9600 Baud geöffnet. Jetzt können Sie eine Schleife starten, die auf Meldungen vom Arduino wartet:

```
while 1 :
   ser.readline()
```

Nachdem Sie die zweite Zeile eingegeben haben, müssen Sie zweimal [Enter] drücken. Daraufhin sollten die Meldungen angezeigt werden. Dabei sehen Sie in blauer Schrift die Nachrichten, die der Arduino an den Pi sendet. Wenn Sie [Strg]+[C] drücken, um die Meldungen vom Arduino abzubrechen, wird eine Fehlermeldung angezeigt.

Geben Sie in der Python-Konsole jetzt Folgendes ein:

```
ser.write('5')
```

Daraufhin blinkt die LED fünfmal.

9.6 Zusammenfassung

In diesem Kapitel haben wir uns einige der vielen Möglichkeiten angesehen, um den Raspberry-Pi-Projekten elektronische Bauteile hinzuzufügen. In den nächsten beiden Kapiteln führen wir zwei Projekte mit unterschiedlichen Vorgehensweisen durch: Im ersten verwenden wir den Cobbler von Adafruit und eine Steckplatine, und im zweiten bauen wir auf der Grundlage des RaspiRobotBoard ein kleines Roboterfahrzeug.

10 Das Prototypprojekt (Uhr)

In diesem Kapitel bauen wir etwas, was man eigentlich nur als eine absurd überkonstruierte LED-Digitaluhr beschreiben kann. Wir verwenden dazu einen Raspberry Pi, das Bandkabel von Adafruit, eine Steckplatine und eine vierstellige LED-Anzeige (siehe Bild 10.1).

Im ersten Bauabschnitt zeigt das Projekt nur die Zeit an, doch im zweiten erweitern wir es um eine Taste, mit der Sie den Anzeigemodus zwischen der Uhrzeit in Stunden und Minuten, der Sekundenzählung und dem Datum umschalten können.

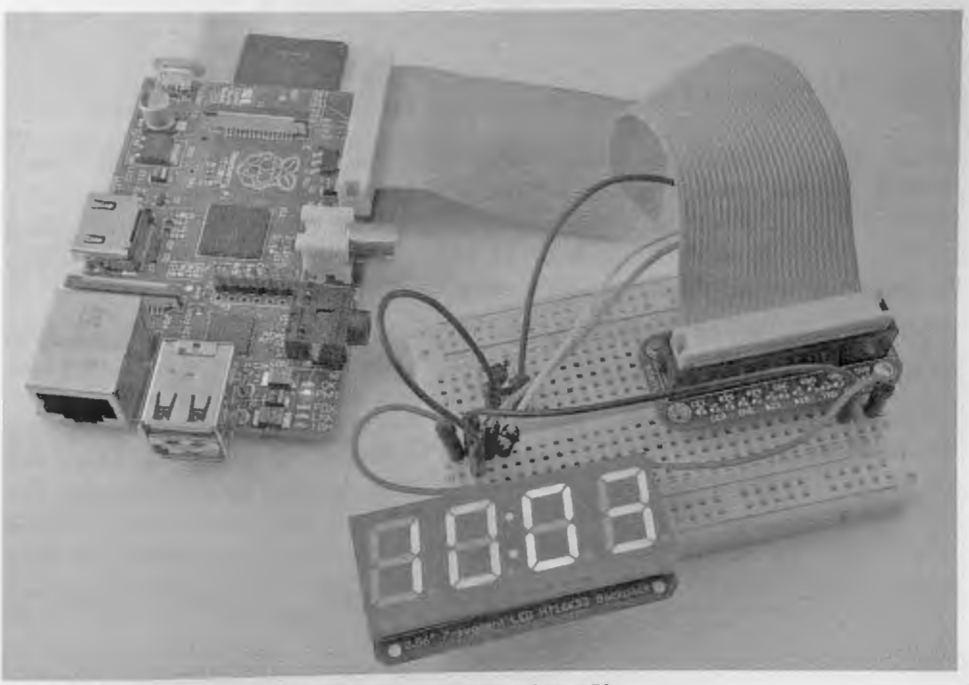


Bild 10.1: Eine LED-Uhr auf der Grundlage des Raspberry Pi

10.1 Benötigtes Material

Um das Projekt zu bauen, benötigen Sie die folgenden Teile. Im Internet können Sie Bezugsquellen finden.

Bauteil	*Richtpreis*
Raspberry Pi Modell B	35 €
Pi Cobbler (Bandkabelsatz)	16,95 €
Vierstellige Sieben-Segment-Anzeige (I2C)	5 €
Breadboard	7 €
Jumperkabel (männlich/männlich) oder einadrige Drähte	10 €
Drucktaster, für Leiterplatte geeignet*	3 €
* Optional, nur für Bauabschnitt 2 erforderlich.	

10.2 Hardwaremontage

Sowohl der Cobbler als auch das Anzeigemodul von Adafruit werden als Bausatz geliefert, der erst zusammengelötet werden muss, bevor Sie ihn einsetzen können. Der Lötvorgang ist in beiden Fällen sehr einfach, und auf der Website von Adafruit gibt es eine ausführliche Schrittanleitung dafür. Die Module verfügen über Kontaktstifte (Pins), die einfach in die Steckplatine gesteckt werden.

Die LED-Anzeige hat vier solcher Pins (VCC, GND, SDA und SCL). Platzieren Sie sie so auf der Steckplatine, dass VCC in Zeile 1 sitzt.

Der Cobbler verfügt über 26 Pins, von denen wir aber nur wenige verwenden. Platzieren Sie ihn am anderen Ende der Steckplatine oder zumindest weit genug weg, sodass sich keiner der Pins in derselben Zeile befindet wie einer der Pins der LED-Anzeige. Der Cobbler ist an einer Stelle ausgeschnitten, sodass das Kabel nur auf eine Weise eingesteckt werden kann. Dieser Ausschnitt muss zur oberen Seite der Steckplatine zeigen, wie Sie in Bild 10.2 sehen.

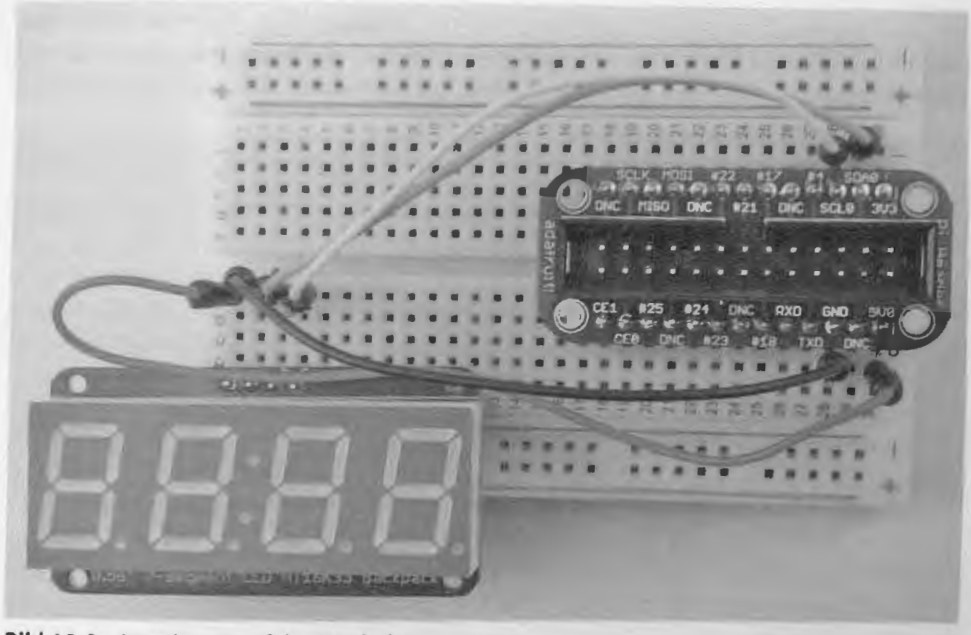


Bild 10.2: Anordnung auf der Steckplatine

Unterhalb der Löcher in der Steckplatine verlaufen elektrische Leiter, die die fünf Löcher einer Zeile miteinander verbinden. Da die Platine auf der Seite liegt, verlaufen die Zeilen in Bild 10.2 in senkrechter Richtung.

Ebenfalls in Bild 10.2 sehen Sie die Steckplatine mit der LED-Anzeige am linken Ende und dem Cobbler am rechten. Wenn Sie die Anleitung in diesem Kapitel nachvollziehen wollen, ist es hilfreich, die Module auf die gleiche Weise einzustecken wie in dieser Abbildung.

Hinweis
Die Überbrückungsdrähte lassen sich viel einfacher auf der Steckplatine anbringen, wenn das Bandkabel noch nicht angeschlossen ist.

Stellen Sie folgende Verbindungen her:

Farbe des Drahts	*Von*	*Zu*
Schwarz	GND am Cobbler	GND an der LED-Anzeige (zweiter Pin von links)
Rot	5V am Cobbler	VCC an der LED-Anzeige (Pin ganz links)
Orange	SDA0 am Cobbler	SDA an der LED-Anzeige (dritter Pin von links)
Gelb	SCL0 am Cobbler	SCL an der LED-Anzeige (Pin ganz rechts)

Die in dieser Tabelle angegebenen Farben sind nur ein Vorschlag, aber es ist üblich, Rot für die positive Stromversorgung zu verwenden und Schwarz oder Blau für die Masseverbindung.

Vorsicht
In diesem Projekt schließen wir ein 5-V-Anzeigemodul an den Raspberry Pi an, der gewöhnlich 3,3 V verwendet. Das können wir nur deshalb gefahrlos tun, da das Anzeigemodul als Folgegerät dient und nur auf den Leitungen SDA und SCL auf Signale lauscht. Andere I2C-Geräte jedoch können als Hauptgerät fungieren, und wenn sie mit 5 V arbeiten, besteht die Gefahr, dass sie den Pi beschädigen. Bevor Sie ein I2C-Gerät an Ihren Raspberry Pi anschließen, müssen Sie genau wissen, was Sie tun.

Wir können den Cobbler jetzt über das mitgelieferte Bandkabel mit dem Raspberry Pi verbinden. Dabei sollte der Pi ausgeschaltet sein. Das Kabel passt nur auf eine Weise in den Cobbler, aber die Buchse am Pi weist keine solche Sicherheitsvorkehrung auf. Achten Sie daher genau darauf, dass die rote Markierung am Kabel zur Außenseite des Raspberry Pi zeigt.

Schalten Sie den Raspberry Pi ein. Wenn die LEDs nicht wie gewöhnlich aufleuchten, schalten Sie den Pi sofort wieder aus und überprüfen die Verkabelung.

10.3 Die Software

Alles ist angeschlossen, der Raspberry Pi ist gestartet – und doch bleibt die Anzeige dunkel, da wir schließlich noch keine Software zu ihrer Verwendung geschrieben haben! Als erste Aufgabe wollen wir eine einfache Uhrenfunktion einrichten, die lediglich die Systemzeit des Raspberry Pi anzeigt. Der Raspberry Pi hat keine Echtzeituhr, die ihm mitteilt, wie spät es ist, sondern bezieht die Uhrzeitangabe von einem Zeitserver im Internet (sofern ein entsprechender Anschluss vorhanden ist).

Die Uhrzeit zeigt der Raspberry Pi unten rechts auf dem Bildschirm an. Wenn er nicht mit dem Internet verbunden ist, können Sie die Zeit mit dem folgenden Befehl manuell angeben:

```
sudo date -s "Aug 24 12:15"
```

Das müssen Sie jedoch bei jedem Neustart tun. Daher ist es besser, den Raspberry Pi mit dem Internet zu verbinden.

Bei der Nutzung der Zeitangaben aus dem Internet kann es sein, dass die Minuten zwar korrekt angezeigt werden, aber die Stunden nicht. Das liegt höchstwahrscheinlich daran,

dass der Raspberry Pi nicht weiß, in welcher Zeitzone er sich befindet. Dieses Problem können Sie mit dem folgenden Befehl lösen, der ein Fenster öffnet, in dem Sie Ihren Kontinent und dann eine Stadt aus Ihrer Zeitzone auswählen können:

```
sudo dpkg-reconfigure tzdata
```

Damit unser Python-Programm den I2C-Bus nutzen kann, den die LED-Anzeige verwendet, muss die Raspbian-Wheezy-Distribution zurzeit noch einige besondere Befehle ausgeben. In späteren Versionen von Raspbian (und anderen Distributionen) wird der Port wahrscheinlich schon vorkonfiguriert sein, sodass diese Befehle nicht mehr nötig sind. Noch aber müssen wir die folgenden Befehle verwenden:

```
sudo apt-get install python-smbus
sudo modprobe i2c-dev
sudo modprobe i2c-bcm2708
```

Hinweis
Möglicherweise müssen Sie die beiden letzten Befehle bei jedem Neustart des Raspberry Pi geben.

Da der Raspberry Pi jetzt die richtige Uhrzeit kennt und der I2C-Bus zur Verfügung steht, können wir ein Python-Programm schreiben, das die Uhrzeit an die LED-Anzeige sendet. Um die Sache einfacher zu machen, habe ich eigens für diese Art von Anzeige ein Python-Bibliotheksmodul vorbereitet, das Sie von *http://code.google.com/p/i2c7segment/downloads/list* herunterladen können.

Wie bei den zuvor installierten Modulen müssen Sie die Datei herunterladen, an einem gut erreichbaren Speicherort entpacken (mit `tar -xzf`) und dann den folgenden Befehl zur Installation unter Python 2 geben:

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

Das Uhrenprogramm selbst ist in dem Dateipaket zu diesem Buch enthalten (siehe *www.buch.cd*). Es heißt **10_01_clock.py** und sieht wie folgt aus:

```
import i2c7segment as display
import time

disp = display.Adafruit7Segment()

while True:
    h = time.localtime().tm_hour
    m = time.localtime().tm_min
```

```
    disp.print_int(h * 100 + m)
    disp.draw_colon(True)
    disp.write_display()
    time.sleep(0.5)
    disp.draw_colon(False)
    disp.write_display()
    time.sleep(0.5)
```

Das Programm ist hübsch und einfach. Die Schleife wird endlos durchlaufen, ruft die aktuelle Stunde und Minute ab und zeigt sie an den richtigen Stellen des LED-Displays an. Dazu wird der Stundenwert mit 100 multipliziert, um ihn an die Stelle ganz links zu verschieben, und dann der rechts daneben erscheinende Minutenwert addiert.

Die meiste Arbeit erledigt die Bibliothek `i2c7segment`. Sie wird verwendet, um mit `print_int` oder `draw_colon` festzulegen, was angezeigt wird, und die Anzeige mit `write_display` zu aktualisieren.

Der Doppelpunkt wird zum Blinken gebracht, indem er eine halbe Sekunde lang dargestellt und dann wieder entfernt wird.

Der Zugriff auf den I2C-Port steht nur Superusern zur Verfügung, deshalb müssen Sie das Programm mit dem folgenden Befehl als Superuser ausführen:

```
sudo python 10_01_clock.py
```

Wenn alles richtig funktioniert, zeigt das Display die Uhrzeit an.

10.4 Zweiter Bauabschnitt

Nachdem das einfache Display jetzt funktioniert, wollen wir sowohl die Hardware als auch die Software erweitern, indem wir eine Taste zum Umschalten des Anzeigemodus hinzufügen, um nacheinander zwischen der Anzeige der Stunden und Minuten, der Sekunden und des Datums und wieder zurück wechseln zu können. Bild 10.3 zeigt die Steckplatine mit der Taste sowie zwei neuen Kabeln. Beachten Sie, dass wir die Anordnung des ersten Bauabschnitts durch diese Teile nur erweitern, an der ursprünglichen Konfiguration aber nichts ändern.

Hinweis
Fahren Sie Ihren Raspberry Pi herunter und schalten Sie ihn ab, bevor Sie irgendwelche Änderungen an der Steckplatine vornehmen.

Die Taste verfügt über vier Drähte, die in der richtigen Stellung platziert werden müssen, da der Schalter ansonsten stets geschlossen ist. Die Drähte müssen von den

Seiten ausgehen, die in Bild 10.3 oben und unten liegen. Wenn Sie die Taste falsch herum eingebaut haben, ist das jedoch kein Beinbruch, denn dadurch wird keines der Bauteile beschädigt. Sie werden jedoch feststellen, dass der Anzeigemodus ständig von selbst wechselt, ohne dass die Taste betätigt wird.

Um die Taste anzuschließen, brauchen Sie zwei neue Kabel. Beide gehen von Kontakten des Schalters aus, wobei das eine zum GND-Anschluss der LED-Anzeige, das andere zum Anschluss Nr. 17 des Cobblers verläuft (siehe Bild 10.3). Das hat zur Folge, dass bei jeder Betätigung der Taste der Pin GPIO 17 des Raspberry Pi mit Masse verbunden wird.

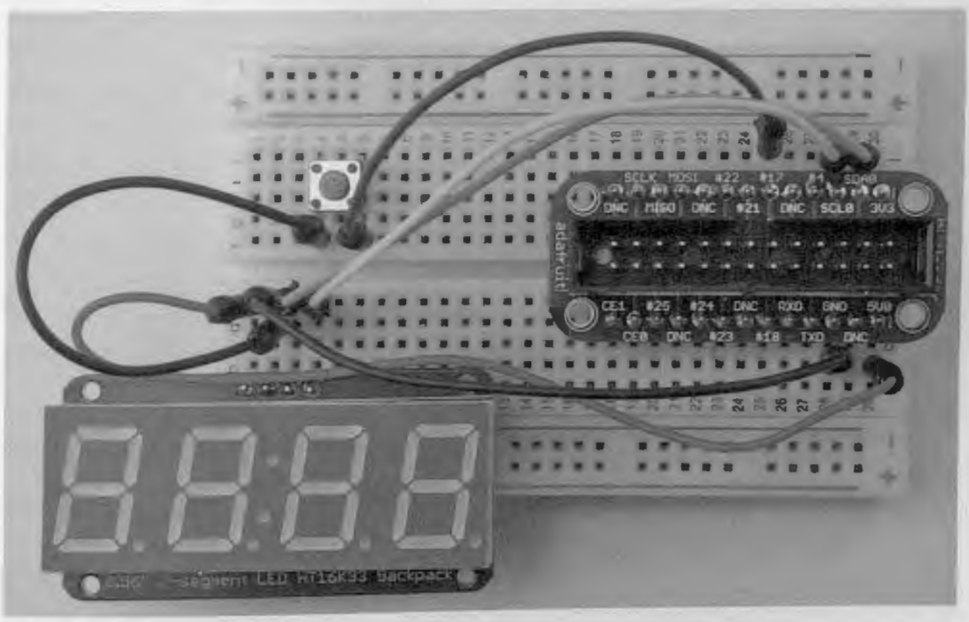


Bild 10.3: Eine Taste hinzufügen

Die aktualisierte Software befindet sich in der Datei **10_02_fancy_clock.py** und sieht wie folgt aus:

```
import i2c7segment as display
import time
import RPi.GPIO as io

switch_pin = 17
io.setmode(io.BCM)
io.setup(switch_pin, io.IN, pull_up_down=io.PUD_UP)
disp = display.Adafruit7Segment()

time_mode, seconds_mode, date_mode = range(3)
```

```
disp_mode = time_mode

def display_time():
    h = time.localtime().tm_hour
    m = time.localtime().tm_min
    disp.print_int(h * 100 + m)
    disp.draw_colon(True)
    disp.write_display()
    time.sleep(0.5)
    disp.draw_colon(False)
    disp.write_display()
    time.sleep(0.5)


def disply_date():
    d = time.localtime().tm_mday
    m = time.localtime().tm_mon
    #disp.print_int(d * 100 + m) # Weltformat
    disp.print_int(m * 100 + d)  # US-Format
    disp.draw_colon(True)
    disp.write_display()
    time.sleep(0.5)

def display_seconds():
    s = time.localtime().tm_sec
    disp.print_str('----')
    disp.print_int(s)
    disp.draw_colon(True)
    disp.write_display()
    time.sleep(0.5)

while True:
    key_pressed = not io.input(switch_pin)
    if key_pressed:
        disp_mode = disp_mode + 1
        if disp_mode > date_mode:
            disp_mode = time_mode
    if disp_mode == time_mode:
        display_time()
    elif disp_mode == seconds_mode:
        display_seconds()
    elif disp_mode == date_mode:
        disply_date()
```

Da wir Zugriff auf den GPIO-Pin 17 benötigen, um erkennen zu können, ob die Taste gedrückt wurde, müssen wir die Bibliothek `RPi.GPIO` verwenden, die wir bereits in

Kapitel 5 als ein Beispiel zur Installation von Modulen kennengelernt haben. Sollten Sie `RPi.GPIO` nicht installiert haben, holen Sie das jetzt anhand der Anleitung aus Kapitel 5 nach.

Mit dem folgenden Befehl legen wir das Signal vom Tasten-Pin als Eingabe fest:

```
io.setup(switch_pin, io.IN, pull_up_down=io.PUD_UP)
```

Dadurch wird auch ein interner Pull-up-Widerstand eingeschaltet, der dafür sorgt, dass die Eingabe stets 3,3 V beträgt (»high«), sofern die Taste nicht gedrückt wird. Bei Betätigung der Taste wird der Widerstand überbrückt und die Spannung gesenkt.

Der größte Teil des Schleifeninhalts wurde in die Funktion `display_time` ausgelagert. Außerdem haben wir jetzt die beiden neuen Funktionen `display_seconds` und `display_date`, die selbsterklärend sein sollten.

Besonders beachtenswert ist die Tatsache, dass `display_date` das Datum im US-Format anzeigt. Wenn Sie das auf das international übliche Format ändern wollen, bei dem erst der Tag und dann der Monat genannt werden, ändern Sie die Zeile, die mit `disp.print_int` beginnt, entsprechend ab (siehe die Kommentare im Code).

Um verfolgen zu können, in welchem Modus wir uns gerade befinden, haben wir in den folgenden Zeilen einige neue Variablen hinzugefügt:

```
time_mode, seconds_mode, date_mode = range(3)
disp_mode = time_mode
```

In der ersten dieser Zeilen erhalten die drei Variablen jeweils eine eigene Nummer. Die zweite setzt `disp_mode` auf den Wert der Variablen `time_mode`, die wir später in der Hauptschleife brauchen.

Die Hauptschleife wurde geändert, um ermitteln zu können, ob die Taste gedrückt wurde oder nicht. Wenn ja, wird `1` zu `disp_mode` addiert, um den Anzeigemodus weiterzuschalten. Ist der letzte Anzeigemodus erreicht, wird wieder auf `time_mode` zurückgeschaltet.

Die folgenden `if`-Blöcke wählen die für den vorliegenden Modus geeignete Anzeigefunktion aus und rufen sie auf.

10.5 Zusammenfassung

Die Hardware dieses Projekts können Sie sehr einfach für andere Verwendungszwecke anpassen. Beispielsweise können Sie auf dem Display alle möglichen Arten von Informationen anzeigen lassen, indem Sie das Programm ändern. Unter anderem ist Folgendes möglich:

- Aktuelle Internetbandbreite (Geschwindigkeit).
- Anzahl der E-Mails im Posteingang.
- Countdown der Tage bis Silvester.
- Anzahl der Besucher einer Website.

Im nächsten Kapitel bauen wir ein weiteres Hardwareprojekt, nämlich ein Roboterfahrzeug mit dem Raspberry Pi als Gehirn.

11 Der RaspiRobot

In diesem Kapitel sehen wir uns an, wie Sie den Raspberry Pi als Gehirn des einfachen Roboterfahrzeugs aus Bild 11.1 verwenden. Der Pi nimmt Befehle von einer drahtlosen USB-Tastatur entgegen und steuert die Motoren, die am Chassis befestigt sind. Optional kann der Roboter auch mit einem Ultraschall-Entfernungsmesser ausgestattet werden, der ihm den Abstand von Hindernissen mitteilt, und mit einem LCD-Bildschirm zur Anzeige der Informationen des Entfernungsmessers. Wie das Projekt im vorigen Kapitel ist auch dieses in zwei Abschnitte aufgeteilt. Als Erstes bauen wir einen einfachen Rover, den Sie über eine drahtlose Tastatur steuern können, danach fügen wir den Entfernungsmesser und den Bildschirm hinzu.

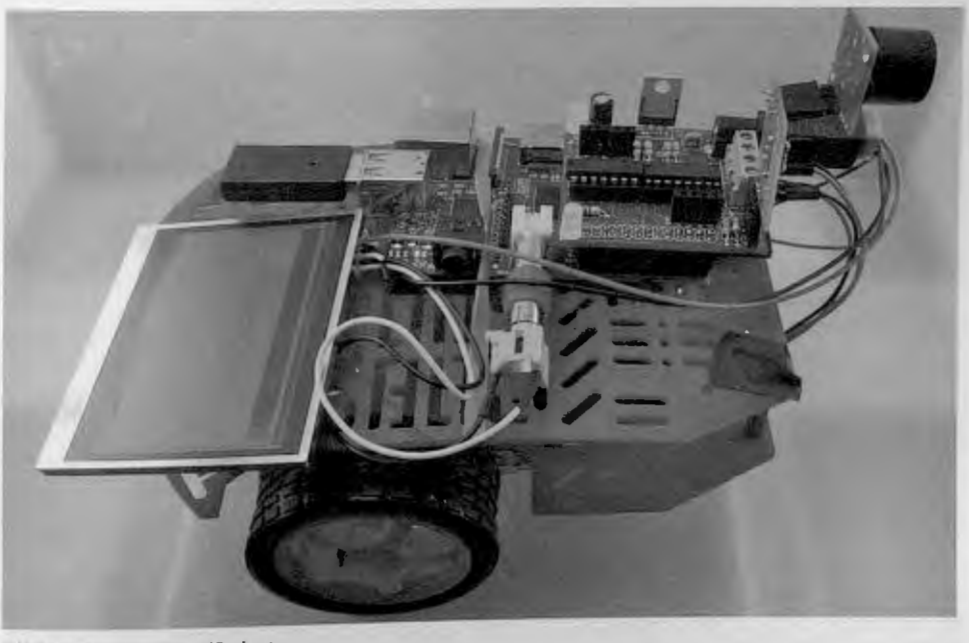


Bild 11.1: Der RaspiRobot

Warnung
Wenn Sie an das RaspiRobotBoard Batterien anschließen, versorgen diese auch den Raspberry Pi mit Strom. Treiben Sie den Raspberry Pi unter keinen Umständen gleichzeitig über sein eigenes Netzteil **und** das RaspiRobotBoard an! Sie können das RaspiRobotBoard am Raspberry Pi belassen, wenn Sie das Netzteil verwenden, dürfen dann aber nicht die Motoren oder die Batterie anschließen.

11.1 Benötigtes Material

Um das Projekt zu bauen, benötigen Sie die folgenden Teile. Im Internet können Sie Bezugsquellen finden.

Bauteil	*Richtpreis*
Raspberry Pi	35 €
RaspiRobotBoard	25 €
Serieller Adapter für Entfernungsmesser*	5 €
Entfernungsmesser Maxbotix LV-EZ1 für seriellen Anschluss*	25 €
3,5"-LCD-Bildschirm	50 €
Cinchadapter männlich/männlich für den Bildschirm*	2 €
Magician-Chassis	12 €
2,1-mm-Schraubklemme (Stromanschluss männlich)**	2 €
Halter für sechs AA-Batterien	2 €
PP3-Batterieklemme**	2 €
Sechs AA-Batterien (Alkali oder wiederaufladbar)	
Drahtlose USB-Tastatur	10 €
* Erst für den zweiten Bauabschnitt.	
** Dies betrifft eine andere Batteriefachkonstruktion als diejenige, die ich verwendet habe. Sie wird mit einem 2,1-mm-Stecker angeschlossen, weshalb die PP3-Batterieklemme nicht notwendig ist. Wenn Sie nur den ersten Bauabschnitt realisieren wollen und den Batteriekasten von Adafruit verwenden, brauchen Sie weder die 2,1-mm-Schraubklemmen noch die PP3-Batterieklemme. PP3 ist übrigens der Anschluss einer 9-V-Batterie.	

Hinweis
Mache Bauteile sind nur im Ausland erhältlich, daher ist mit höheren Versandkosten zu rechnen.

11.2 Erster Bauabschnitt: Der einfache Rover

Bild 11.2 zeigt den einfachen Rover, der auf dem Chassis-Satz von Magician aufgebaut ist. Dieser nützliche Satz besteht aus einem Kunststoff-Chassis, Getriebemotoren, Rädern und allen erforderlichen Montageteilen. Außerdem ist ein Batteriefach für vier AA-Batterien enthalten, das wir in diesem Projekt jedoch durch ein Fach für sechs Batterien ersetzen.

11.2.1 Hardwaremontage

Dieses Projekt wird aus verschiedenen Teilesätzen zusammengebaut. Wenn Sie nach dem RaspiRobotBoard und dem seriellen Adapter für den Entfernungsmesser suchen, werden Ihnen möglicherweise auch vorgefertigte Elemente angeboten, sodass Sie das ganze Projekt ohne jegliche Lötarbeiten (oder sogar ohne Zuhilfenahme irgendwelcher Werkzeuge, die komplizierter sind als ein Schraubendreher) zusammenbauen können.

Schritt 1: Das Chassis zusammenbauen

Das Magician-Chassis wird als Bausatz geliefert, den Sie erst zusammenfügen müssen. Eine ausführliche Bauanleitung ist beigefügt. Bei der Montage müssen Sie den zugehörigen Batteriekasten (für vier AA-Batterien) durch die Version für sechs Batterien ersetzen (siehe Bild 11.3). Wenn Sie einen Batteriekasten haben, bei dem die Batterien in zwei Lagen zu je drei angeordnet sind, können Sie ihn an der oberen Platte des Magician-Chassis festklemmen. Er sitzt sehr fest und federt ein wenig, deshalb müssen Sie den Mittelträger nicht montieren.

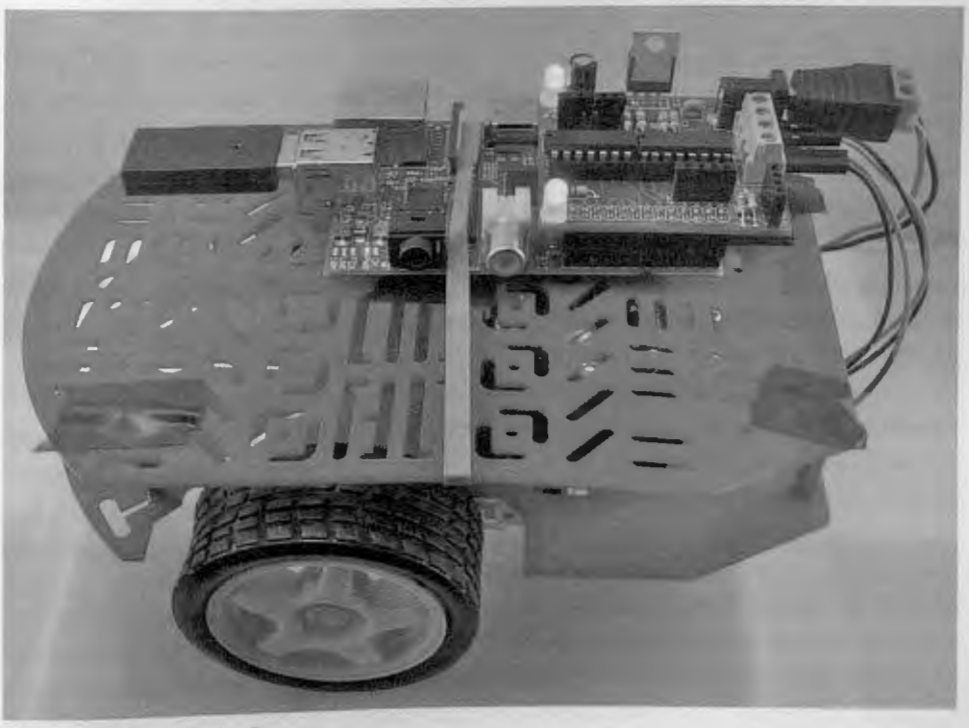


Bild 11.2: Der einfache Rover

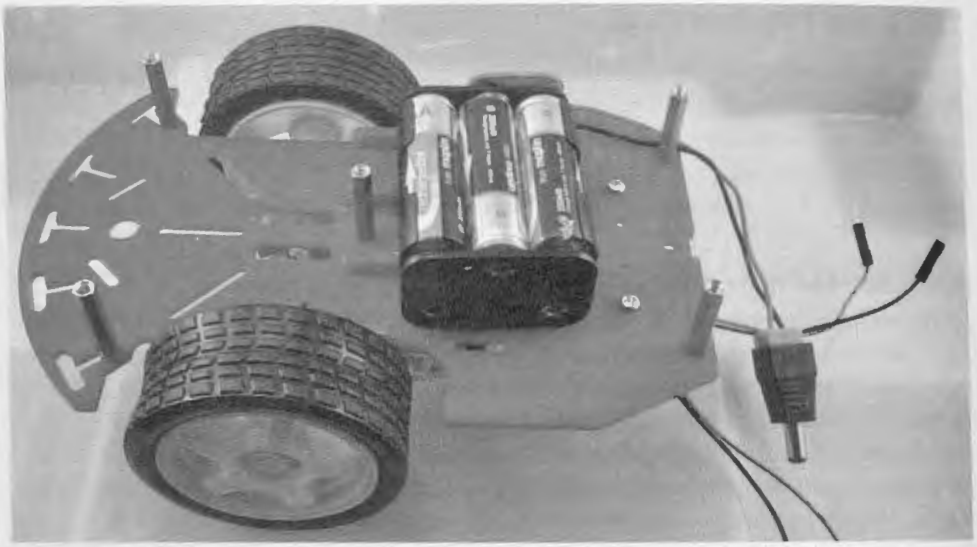


Bild 11.3: Das Batteriefach ersetzen

Sind die Batterien dagegen in einer Lage angeordnet, müssen Sie Ihr Batteriefach mithilfe der Schrauben, die für das ursprüngliche Batteriefach mitgeliefert wurden, am Chassis befestigen.

Bringen Sie die Batterieklemme am Batteriefach und die davon ausgehenden Drähte an den Schraubklemmen an, um den Netzstecker anschließen zu können. Achten Sie dabei genau auf die Polung (Rot an Plus)!

Bevor Sie die obere Platte des Magician-Chassis aufsetzen, ziehen Sie ein Gummiband darüber, um damit den Raspberry Pi festzuhalten (siehe Bild 11.2).

Schritt 2: Montage des RaspiRobotBoard

Zurzeit ist noch unklar, ob das RaspiRobotBoard fertig montiert oder nur als Bausatz erhältlich sein wird. In letzterem Fall müssen Sie der mitgelieferten Bauanleitung folgen. Im fertigen Zustand sieht die Platine aus wie die in Bild 11.4.

Beachten Sie, dass die folgenden Anweisungen nur für Version 1 der Platine gelten, denn bei späteren Ausgaben mag sich die Lage der Anschlüsse ändern. Weitere Informationen erhalten Sie auf der Website zu diesem Buch (*www.buch.cd*). Alle Anschlüsse, die für uns von Interesse sind, befinden sich in Bild 11.4 auf der rechten Seite. Oben befindet sich der Stromanschluss, darunter die Schraubklemmen für den linken und den rechten Motor.

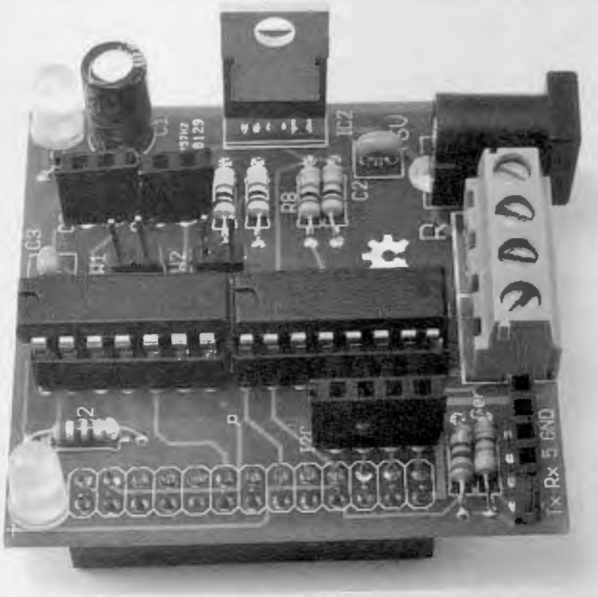


Bild 11.4: Das RaspiRobotBoard

Schritt 3: Installieren der Software auf dem Raspberry Pi

Zur Steuerung des Roboters schreiben wir ein Programm in Python, das Tastenbetätigungen erkennt und dementsprechend die Leistungsabgabe der Motoren am Roboter regelt. Dazu verwenden wir Pygame, was uns eine hilfreiche Möglichkeit an die Hand gibt, um herauszufinden, ob eine Taste gedrückt wurde oder nicht.

Es ist sehr viel einfacher, das Programm einzurichten, bevor wir den Raspberry Pi am Chassis befestigen. Bringen Sie daher das RaspiRobotBoard am Raspberry Pi an, schließen Sie Motor und Batterie aber noch nicht an, sondern versorgen Sie den Pi über das normale USB-Netzteil mit Strom.

Das RaspiRobotBoard verfügt über seine eigene Python-Bibliothek, stützt sich aber auch auf einige andere Bibliotheken, die zusätzlich installiert werden müssen. Als Erstes benötigt es die Bibliothek `RPi.GPIO`, die Ihnen bereits in Kapitel 5 und Kapitel 10 begegnet ist. Wenn Sie sie noch nicht installiert haben, holen Sie das jetzt nach. Außerdem müssen Sie die Bibliothek `PySerial` installieren. Anweisungen dazu erhalten Sie im Arduino-Abschnitt am Ende von Kapitel 9.

Die RaspiRobotBoard-Bibliothek können Sie von der Website *http://code.google.com/p/raspirobotboard/downloads/list* herunterladen und installieren.

Die Installation läuft wie bei allen anderen Python-Paketen ab. Da wir in diesem Projekt Python 2 verwenden, müssen Sie die Bibliothek mit dem folgenden Befehl installieren:

```
tar -xzf raspirobotboard-1.0.tar.gz
cd raspirobotboard-1.0
sudo python setup.py install
```

Das eigentliche Python-Programm für diese Version des Roboters befindet sich in der Datei **11_01_rover_basic.py**, die Sie als Superuser ausführen müssen. Wechseln Sie zum Ausprobieren (solange die Motoren noch nicht angeschlossen sind!) in das Verzeichnis mit dem Code und geben Sie im Terminal folgenden Befehl ein:

```
sudo python 11_01_rover_basic.py
```

Es erscheint ein leeres Pygame-Fenster, und zwei der LEDs erlöschen. Da das Programm neben den Motoren auch die LEDs steuert, können wir es auch ohne Motoren testen. Wenn Sie die Taste mit dem aufwärts weisenden Pfeil drücken, leuchten beide LEDs wieder auf. Versuchen Sie es dann mit den Pfeiltasten rechts und links. Jetzt leuchtet jeweils die LED auf, die der betätigten Taste entspricht.

Da wir keine Kabelverbindung zwischen dem Rover und einem Bildschirm und einer Maus haben, richten wir das Programm so ein, dass es nach dem Start des Raspberry Pi automatisch ausgeführt wird. Dazu müssen wir die Datei **raspirobot_basic.desktop** (die im Codeverzeichnis enthalten ist) in das Verzeichnis **/home/pi/.config/autostart** legen, was Sie mit dem Datei-Manager erledigen können. Geben Sie in der Adressleiste oben auf dem Bildschirm einfach `/home/pi/.config` ein. Verzeichnisse, deren Namen mit einem Punkt beginnen, sind versteckt, deshalb können Sie sich im Datei-Manager nicht einfach dorthin vorklicken.

Gibt es in **.config** kein Verzeichnis namens **autostart**, legen Sie eins an und kopieren **raspirobot_basic.desktop** dort hinein. Um zu überprüfen, ob der automatische Start funktioniert, führen Sie einen Neustart des Pi durch. Das Pygame-Fenster sollte automatisch geöffnet werden.

Den Code für dieses Projekt sehen wir uns später noch genauer an. Jetzt aber wollen wir zunächst dafür sorgen, dass alle Teile laufen.

Schritt 4: Die Motoren anschließen

Fahren Sie den Raspberry Pi herunter und trennen Sie ihn vom Netzteil. Legen Sie es beiseite, sodass Sie es nicht versehentlich gleichzeitig mit den Batterien anschließen. Füllen Sie das Batteriefach und setzen Sie die obere Platte des Chassis auf. Decken Sie die Metallschrauben mit kleinen Stücken Isolier- oder Klebeband ab, um versehentliche Kurzschlüsse mit dem Raspberry Pi zu verhindern, und schieben Sie den Pi unter das Gummiband. Schließen Sie als Nächstes die Motoren am Klemmenblock an, wie Bild 11.5 zeigt.

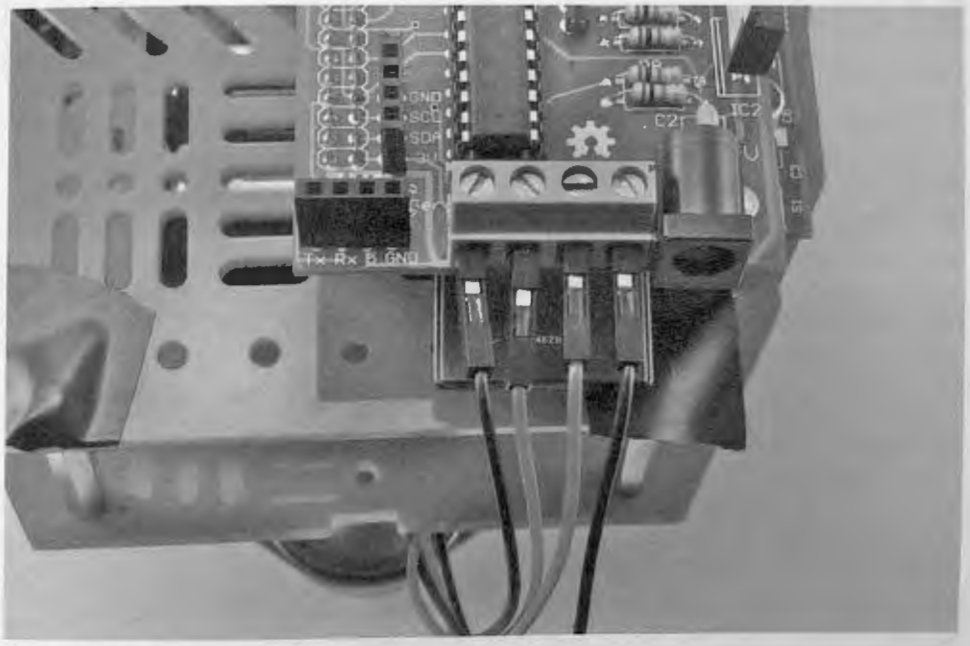


Bild 11.5: Die Motoren anschließen

Jeder Motor hat ein rotes und ein schwarzes Kabel. Das schwarze Kabel des linken Motors gehört an die Schraubklemme außen links, das rote Kabel dieses Motors an die Klemme rechts daneben. Das rote Kabel des rechten Motors kommt in die dritte Schraubklemme von links, das schwarze in die verbleibende rechts außen (siehe Bild 11.5).

Schritt 5: Ausprobieren

Das war's schon – jetzt können Sie loslegen! Schließen Sie den USB-Dongle der drahtlosen Tastatur am Pi und den Stecker des Batteriekabels an der Buchse auf dem RaspiRobotBoard an. Die LEDs auf dem Raspberry Pi flackern wie bei einem Startvorgang. Sollte das nicht geschehen, trennen Sie sofort die Verbindung zu den Batterien und überprüfen das Fahrzeug.

Zu Anfang leuchten beide LEDs auf dem RaspiRobotBoard, wenn aber das Python-Programm ausgeführt wird, erlöschen sie. Warten Sie eine oder zwei Sekunden, bis das Programm sauber gestartet ist, und versuchen Sie dann, die Pfeiltasten oder die Leertaste auf Ihrer Tastatur zu drücken. Und siehe da, der RaspiRobot bewegt sich!

11.2.2 Die Software

Die Software für den ersten Bauabschnitt sieht wie folgt aus:

```
from raspirobotboard import *
import pygame
import sys
from pygame.locals import *
```

```
rr = RaspiRobot()

pygame.init()
screen = pygame.display.set_mode((640, 480))
pygame.display.set_caption('RaspiRobot')
pygame.mouse.set_visible(0)

while True:
   for event in pygame.event.get():
      if event.type == QUIT:
         sys.exit()
      if event.type == KEYDOWN:
         if event.key == K_UP:
            rr.forward()
            rr.set_led1(True)
            rr.set_led2(True)
         elif event.key == K_DOWN:
            rr.set_led1(True)
            rr.set_led2(True)
            rr.reverse()
         elif event.key == K_RIGHT:
            rr.set_led1(False)
            rr.set_led2(True)
            rr.right()
         elif event.key == K_LEFT:
            rr.set_led1(True)
            rr.set_led2(False)
            rr.left()
         elif event.key == K_SPACE:
            rr.stop()
            rr.set_led1(False)
            rr.set_led2(False)
```

Hinweis
Wenn Sie Kapitel 8 über Pygame übersprungen haben, sollten Sie es jetzt lesen.

Zu Anfang importiert das Programm die Bibliotheksmodule, die es braucht. Dann erstellt es eine Instanz der Klasse `RaspiRobot` und weist sie der Variablen `rr` zu. Die Hauptschleife sucht als Erstes nach einem `QUIT`-Ereignis. Wenn sie fündig wird, beendet sie das Programm. Im Rest der Schleife geht es darum, alle Tasten zu überprüfen und den entsprechenden Befehl zu geben, wenn eine Taste betätigt wurde. Wird beispielsweise die Taste mit dem aufwärts weisenden Pfeil gedrückt (`K_UP`), wird dem Roboter der Befehl `forward` gesendet, der dafür sorgt, dass beide Motoren vorwärts laufen und beide LEDs aufleuchten.

11.3 Zweiter Bauabschnitt: Entfernungsmesser und Bildschirm hinzufügen

Am Ende des zweiten Bauabschnitts sieht Ihr RaspiRobot wie in Bild 11.1 aus. Trennen Sie die Batterien vom RaspiRobotBoard, um die notwendigen Änderungen vorzunehmen.

11.3.1 Schritt 1: Den seriellen Adapter des Entfernungsmessers zusammenbauen

Das serielle Entfernungsmessermodul aus Bild 11.6 gibt ein invertiertes Signal aus. Wir benötigen also eine kleine Platine mit einem Transistor und einem Widerstand, um es wieder richtig herum zu drehen. Eine vollständige Anleitung für den Zusammenbau dieser Adapterplatine erhalten Sie auf der Website zu diesem Buch (*www.buch.cd*). Das Entfernungsmessermodul wird oben auf den Adapter aufgesteckt, und der untere Teil des Adapters wiederum passt in eine serielle Buchse, wie Sie in Bild 11.7 sehen.

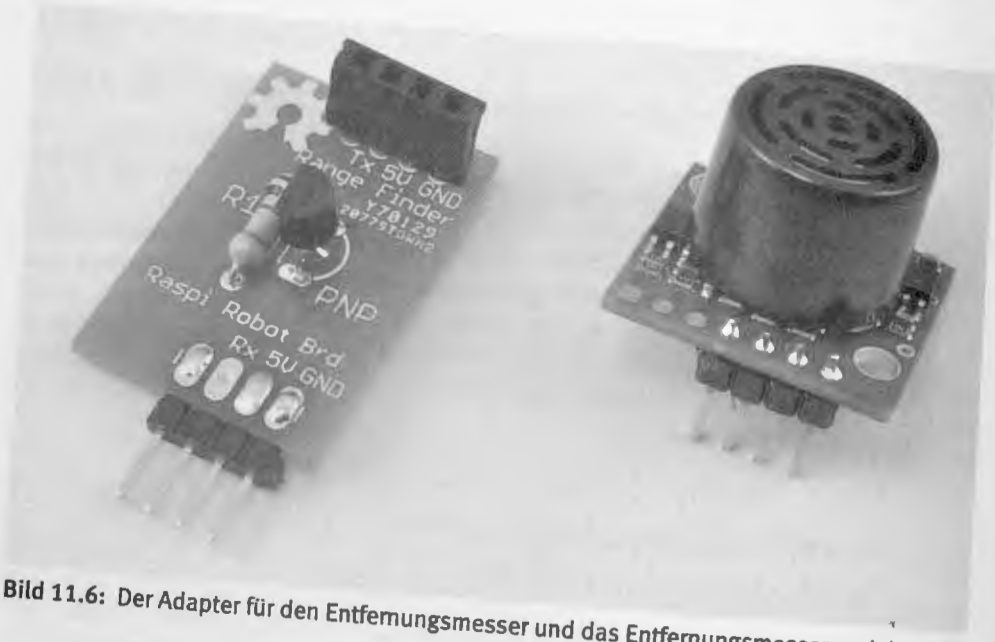


Bild 11.6: Der Adapter für den Entfernungsmesser und das Entfernungsmessermodul

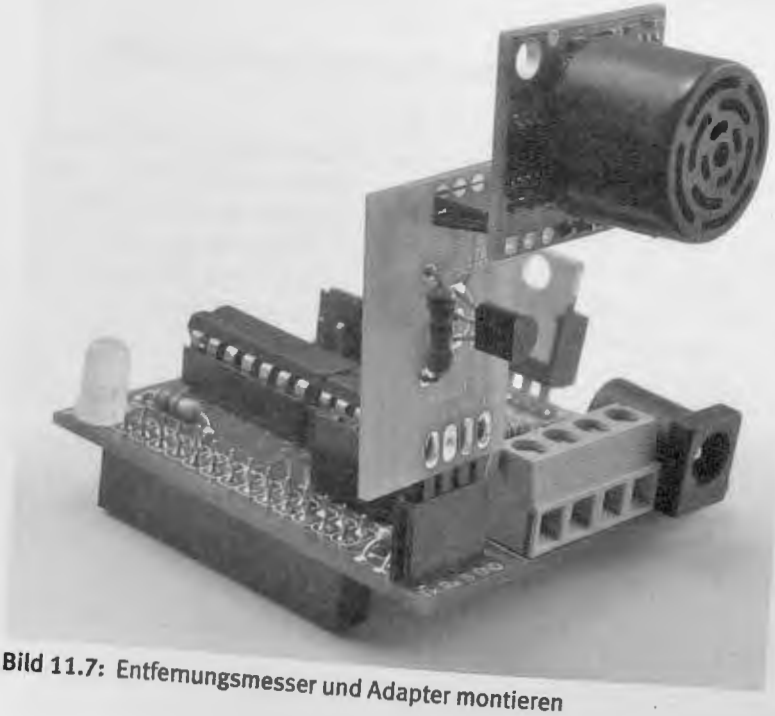


Bild 11.7: Entfernungsmesser und Adapter montieren

11.3.2 Schritt 2: Den Bildschirm anbringen

Der LCD-Bildschirm wird in zwei Teilen geliefert: Neben dem eigentlichen Bildschirm gibt es auch noch eine Treiberplatine. Die beiden Teile sind über ein ziemlich empfindliches Flachbandkabel verbunden. Ich habe sie mit Klebefilz aneinander befestigt. Seien Sie bei der Montage des Bildschirms sehr vorsichtig und behandeln Sie den Monitor wie ein rohes Ei!

Der Bildschirm wird mit Stromkabeln (in Rot und Schwarz) sowie mit zwei Cinchsteckern geliefert. Um unnötigen Kabelsalat zu vermeiden, habe ich die mittleren Kabel mit dem weißen Stecker abgeschnitten, wie Sie in Bild 11.8 erkennen können. Wenn Ihnen diese Maßnahme zu drastisch erscheint, können Sie sie auch einfach mithilfe eines Kabelbinders irgendwo befestigen, wo sie nicht im Weg sind.

An den verbliebenen Stecker habe ich den Cinchadapter »männlich/männlich« angeschlossen. Anschließend habe ich die Stromkabel mit den jeweils gleichfarbigen Stromkabeln von der Batterieklemme verdrillt und in den Schraubklemmen des Steckeradapters befestigt. Sollte der Batterieeinsatz mit einem Stecker enden, können Sie diesen abschneiden und das Kabelende abisolieren. Anschließend können Sie die frei liegenden Drähte wie die Drähte von einer Batterieklemme benutzen. In jedem Fall sieht die Verkabelung für das Projekt aus, wie in Bild 11.9 gezeigt.

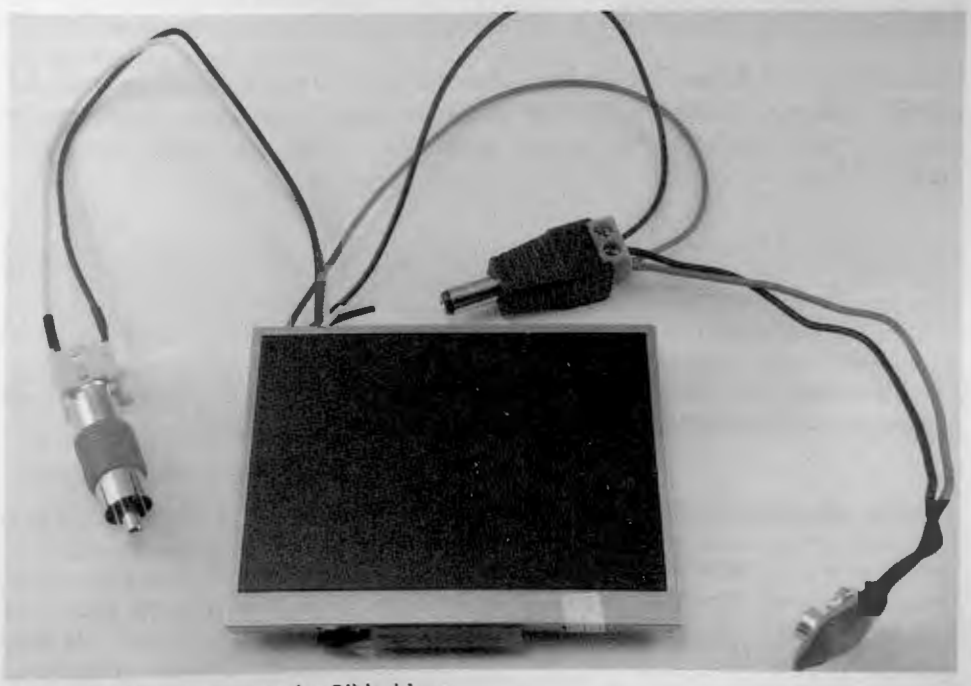


Bild 11.8: Die Verkabelung des Bildschirms

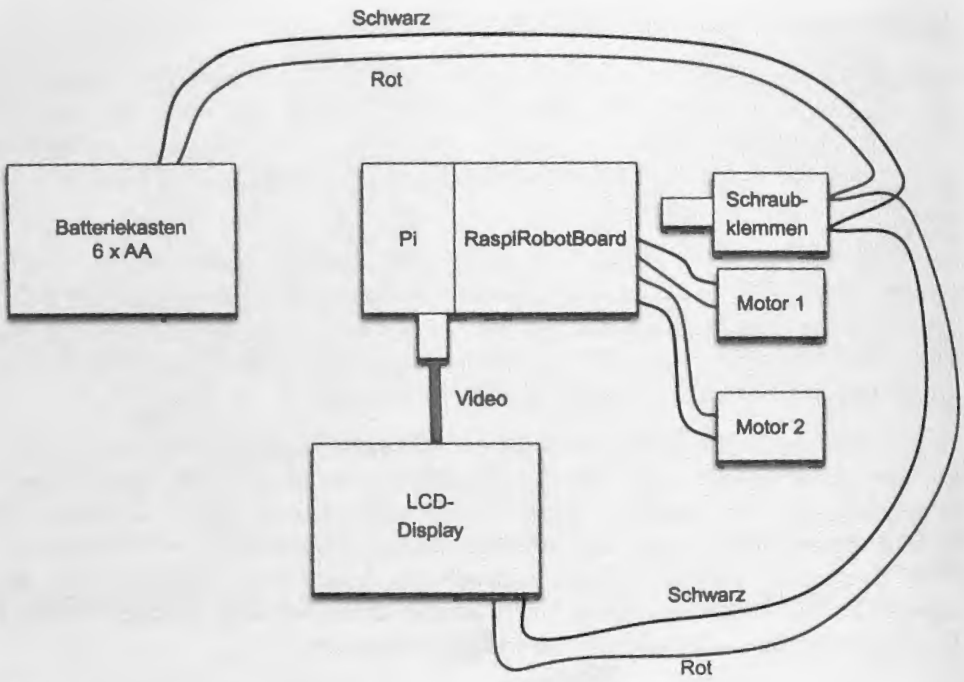


Bild 11.9: Verkabelungsdiagramm

Diese Art der Verkabelung hat zur Folge, dass der Bildschirm auch dann noch mit der Batterie verbunden ist, wenn Sie das RaspiRobotBoard von der Stromversorgung trennen. Daher müssen Sie zum Ein- und Ausschalten des Roboters die Batterieklemme am Batteriekasten verwenden.

Hinweis
Anspruchsvollere Leser können sich auch den Luxus eines Ein-/Ausschalters leisten.

Der Bildschirm wird mit Kitt am Chassis befestigt, was auf Dauer jedoch keine gute Lösung darstellt. Irgendeine Art von Kunststoffhalterung eignet sich besser.

11.3.3 Schritt 3: Die Software aktualisieren

Zur Steuerung der erweiterten Hardware ist auch eine aktualisierte Software vonnöten. Das entsprechende Programm finden Sie in der Datei **11_02_rover_plus.py**. Außerdem müssen Sie dafür sorgen, dass dieses Programm gestartet wird und nicht die alte, einfache Version. Dazu kopieren Sie die Datei **raspirobot_plus.desktop** in das Verzeichnis **/home/pi/.config/autostart** und entfernen die Datei **raspirobot_basic.desktop** aus diesem Ordner, da ansonsten beide Programme gestartet würden.

Da der Raspberry Pi in dieser Projektphase über einen Bildschirm und eine Tastatur verfügt (wenn auch nur sehr klein), können Sie die beschriebenen Änderungen auch direkt hier vornehmen, wobei Sie mit dem winzigen Bildschirm vorlieb nehmen müssen. Sollte das zu schwierig sein, trennen Sie die Batterie und die Motoren, versorgen den Raspberry Pi wie zuvor über sein USB-Netzteil mit Strom und arbeiten mit dem regulären Monitor, der regulären Tastatur und der Maus.

11.3.4 Schritt 4: Ausführen

Damit ist das Projekt betriebsbereit! Wenn die LEDs am Raspberry Pi nicht sofort aufleuchten, müssen Sie wie immer die Batterieverbindung trennen und nach dem Problem suchen. Der Raspberry Pi ist für ein batteriebetriebenes Gerät ein ziemlicher Stromfresser, und auch der Bildschirm ist alles andere als sparsam. Um die Batterien nicht zu oft aufladen zu müssen, lösen Sie daher ihre Verbindung, wenn sie nicht gebraucht werden.

11.3.5 Die veränderte Software

Das neue Programm ist umfangreicher als das alte, deshalb geben wir es hier nicht komplett wieder. Sie können es sich jedoch in IDLE ansehen. Die Hauptunterschiede liegen, wie zu erwarten ist, in der Distanzmessung und der Anzeige. Im Folgenden sehen Sie die Funktion `get_range`:

```
def get_range():
   try:
      dist = rr.get_range_inch()
   except:
      dist = 0
   return dist
```

Bei dieser Funktion handelt es sich nur um einen dünnen Wrapper, um `get_range_inch` aus dem RaspiRobot-Modul aufzurufen. Die Ausnahmebehandlung ist vorhanden, da eine Ausnahme geworfen wird, wenn der Entfernungsmesser aus irgendeinem Grund nichts finden kann (etwa wenn er nicht angeschlossen ist). Diese Funktion fängt solche Ausnahmen ab und gibt dann die Entfernung `0` zurück.

Die Funktion `update_display` ruft die Entfernung ab und zeigt sie zusammen mit einer grafischen Darstellung des Abstands zu irgendwelchen Hindernissen an, wie Bild 11.10 zeigt.

Der Code sieht wie folgt aus:

```
def update_distance():
   dist = get_range()
```

```
if dist == 0:
    return
message = 'Distance: ' + str(dist) + ' in'
text_surface = font.render(message, True, (127, 127, 127))
screen.fill((255, 255, 255))
screen.blit(text_surface, (100, 100))

w = screen.get_width() - 20
proximity = ((100 - dist) / 100.0) * w
if proximity < 0:
    proximity = 0
pygame.draw.rect(screen, (0, 255, 0), Rect((10, 10),(w, 50)))
pygame.draw.rect(screen, (255, 0, 0), Rect((10, 10),(proximity, 50)))
pygame.display.update()
```

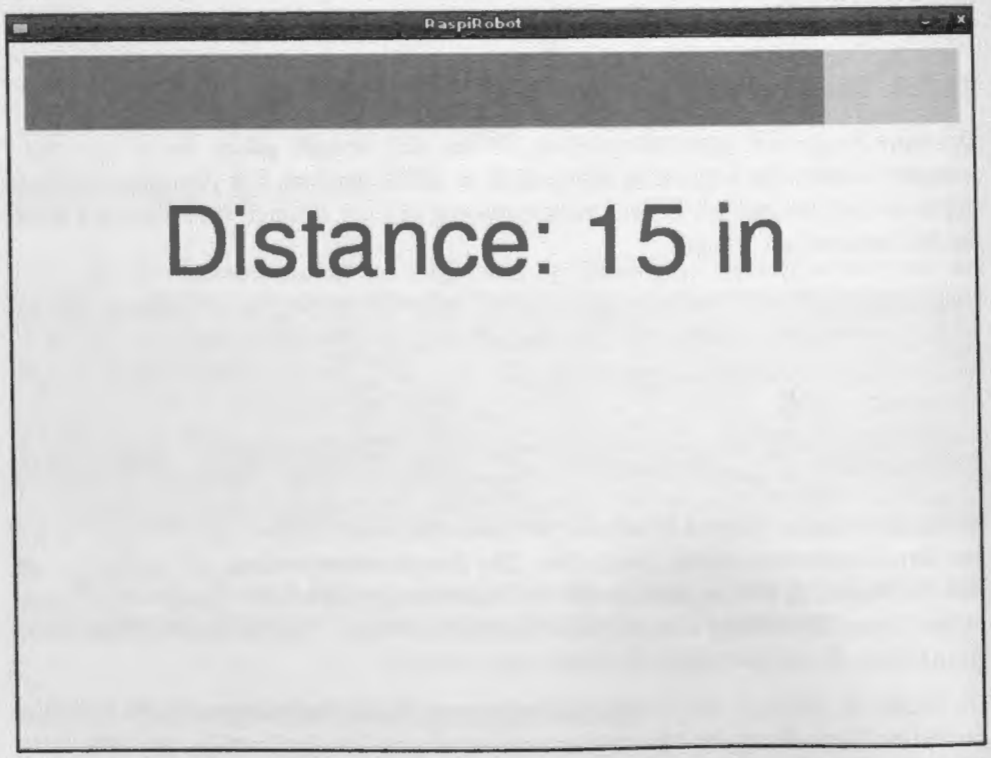


Bild 11.10: Die Anzeige von RaspiRobot

Nach der Messung des Abstands wird eine Meldung auf einer Oberfläche konstruiert, die dann an das Display übertragen wird. Zur grafischen Darstellung wird ein grünes

Rechteck fester Größe gezeichnet und darüber ein rotes Rechteck, dessen Breite von der gemessenen Entfernung abhängt.

11.4 Zusammenfassung

Dieses Projekt können Sie als Grundlage für eigene Roboterprojekte nutzen. Das RaspiRobotBoard verfügt über zwei zusätzliche Ausgänge, mit denen Sie einen Summer oder andere elektronische Geräte steuern können. Eine interessante Möglichkeit zur Erweiterung des Projekts besteht darin, Software zu schreiben, mit der der Roboter auf dem Teller drehen kann, oder mit dem Entfernungsmesser eine Sonarkarte des umgebenden Raums aufzuzeichnen. Mit einem Raspberry-Pi-Kameramodul und einem WLAN-Dongle stehen Ihnen alle Möglichkeiten für mobile Fernüberwachungsgeräte offen!

Im letzten Kapitel dieses Buchs sehen wir uns an, was Sie als Nächstes mit Ihrem Raspberry Pi tun können. Außerdem lernen Sie einige wichtige Quellen zum Raspberry Pi kennen.

Distance: 15 in

12 Die nächsten Schritte

Der Raspberry Pi ist ein Gerät von erstaunlicher Flexibilität, das Sie in allen möglichen Situationen einsetzen können – als Ersatz für einen Desktopcomputer, als Mediencenter oder in Form eines eingebetteten Computers als Regelsystem.

In diesem Kapitel erhalten Sie einige Hinweise auf die verschiedenen Nutzungsmöglichkeiten für den Raspberry Pi und lernen einige der verfügbaren Quellen kennen, in denen erläutert wird, wie Sie den Raspberry Pi programmieren und auf viele interessante Weisen bei sich zu Hause einsetzen können.

12.1 Quellen zu Linux

Der Raspberry Pi gehört zu den vielen Computern, die Linux verwenden. In den meisten Büchern über Linux finden Sie nützliche Informationen. Halten Sie aber vor allem nach Büchern zu der von Ihnen verwendeten Distribution Ausschau. Für Raspbian ist das Debian.

Neben Informationen über den Datei-Manager und Anwendungen, die genauere Erklärungen erfordern, benötigen Sie vor allem Hinweise zur Verwendung des Terminals und zur Konfiguration von Linux. Ein nützliches Buch zu diesem Thema ist **The Linux Command Line: A Complete Introduction** von William E. Shotts, Jr. Im Internet finden Sie viele weitere gute Quellen, um mehr über Linux zu lernen. Nutzen Sie die Suchmaschinen!

12.1.1 Quellen zu Python

Python ist nicht auf den Raspberry Pi beschränkt. Es gibt viele Bücher und Internetquellen zu diesem Thema. Als leichte Einführung in Python können Sie **Python: Visual QuickStart Guide** von Toby Donaldson nehmen. Es ist in einem ähnlichen Stil abgefasst wie dieses Buch, bietet aber eine andere Perspektive. Außerdem ist es sehr entgegenkommend und aufmunternd geschrieben. Wenn Sie etwas Handfesteres wollen, das aber immer noch für Anfänger geeignet ist, greifen Sie zu **Python Programming: An Introduction to Computer Science** von John Zelle.

Wollen Sie mehr über Pygame wissen, erweist sich **Beginning Game Development with Python and Pygame** von Will McGugan als sehr hilfreich.

Auch im Web finden Sie einige gute Quellen zu Python, die Sie wahrscheinlich sogar in die Favoritenliste Ihres Browsers aufnehmen werden:

- *http://docs.python.org/py3k/* ist die offizielle Python-Website, komplett mit hilfreichen Tutorials und Referenzmaterial.
- *www.pythonware.com/library/tkinter/introduction/* ist eine nützliche Referenz zu Tkinter.
- *http://zetcode.com/gui/tkinter/layout/*: Dieses Tutorial wirft dringend benötigtes Licht auf das Widget-Layout mit Tkinter.
- *www.pygame.org* ist die offizielle Pygame-Website. Sie enthält Nachrichten, Tutorials, Referenzmaterial und Beispielcode.

12.2 Quellen zum Raspberry Pi

Die offizielle Website der Raspberry Pi Foundation ist *www.raspberrypi.org*. Dort finden Sie reichhaltige nützliche Informationen und erfahren, was in der Welt des Raspberry Pi vor sich geht.

Wenn Sie nach der Lösung eines kniffligen Problems forschen, sind vor allem die Foren sehr hilfreich. Sie können im Forum nach den Erfahrungen anderer Benutzer suchen, die dasselbe zu tun versucht haben wie Sie, Sie können Fragen stellen oder einfach vorführen, was Sie machen.

Auch wenn Sie das Image Ihrer Raspberry-Pi-Distribution aktualisieren wollen, sind Sie hier an der richtigen Stelle. Die Downloadseite führt die aktuell genutzten Distributionen auf.

Es gibt sogar ein eigenes Onlinemagazin für den Raspberry Pi mit dem launigen Titel **The MagPi** (was man einerseits als »das Magazin für den Pi« lesen kann, aber auch als »the magpie«, also »die Elster«). Sie können es kostenlos als PDF herunterladen (*www.themagpi.com*). Es enthält eine gute Mischung aus Artikeln und Anleitungen, die Sie dazu anregen, großartige Dinge mit Ihrem Pi anzustellen.

Weitere Informationen über die Hardware des Raspberry Pi erhalten Sie unter den folgenden Links:

- *http://elinux.org/RPi_VerifiedPeripherals* zeigt eine Liste der Peripheriegeräte, deren Funktionstüchtigkeit im Zusammenhang mit dem Raspberry Pi erwiesen ist.
- *http://elinux.org/RPi_Low-level_peripherals* zeigt eine Liste von Peripheriegeräten für den GPIO-Anschluss.
- *www.element14.com/community/docs/DOC-43016/* enthält ein Datenblatt für den Broadcom-Chip, der das Herz des Raspberry Pi bildet. (Nichts für Feiglinge!)

Wenn Sie Zusatzhardware und Komponenten für Ihren Raspberry Pi kaufen wollen, wenden Sie sich an die Firma Adafruit, die dem Raspberry Pi eine eigene Kategorie widmet. Auch SparkFun verkauft Zusatzplatinen und Module für den Raspberry Pi.

12.3 Andere Programmiersprachen

In diesem Buch haben wir uns die Programmierung des Raspberry Pi ausschließlich in Python angesehen, und das nicht ohne Grund, denn Python ist eine weit verbreitete Sprache, die einen guten Kompromiss zwischen leichter Benutzbarkeit und Leistungsumfang bietet. Allerdings ist Python bei Weitem nicht die einzige Möglichkeit für die Programmierung des Raspberry Pi. Die Raspbian-Wheezy-Distribution umfasst auch mehrere andere Sprachen.

12.3.1 Scratch

Scratch ist eine grafische Programmiersprache, die vom MIT entwickelt wurde. Sie ist unter Pädagogen sehr beliebt, um junge Menschen dazu zu ermuntern, das Programmieren zu erlernen. Scratch bringt seine eigene Entwicklungsumgebung mit (vergleichbar mit IDLE für Python), die Programmierung erfolgt aber dadurch, dass Programmstrukturen mit der Maus verschoben werden, nicht durch Texteingabe.

Bild 12.1 zeigt einen Ausschnitt aus einem Beispielprogramm für das Spiel Pong, das zusammen mit Scratch ausgeliefert wird. Der Ball wird hier mit einem Paddel geschlagen.

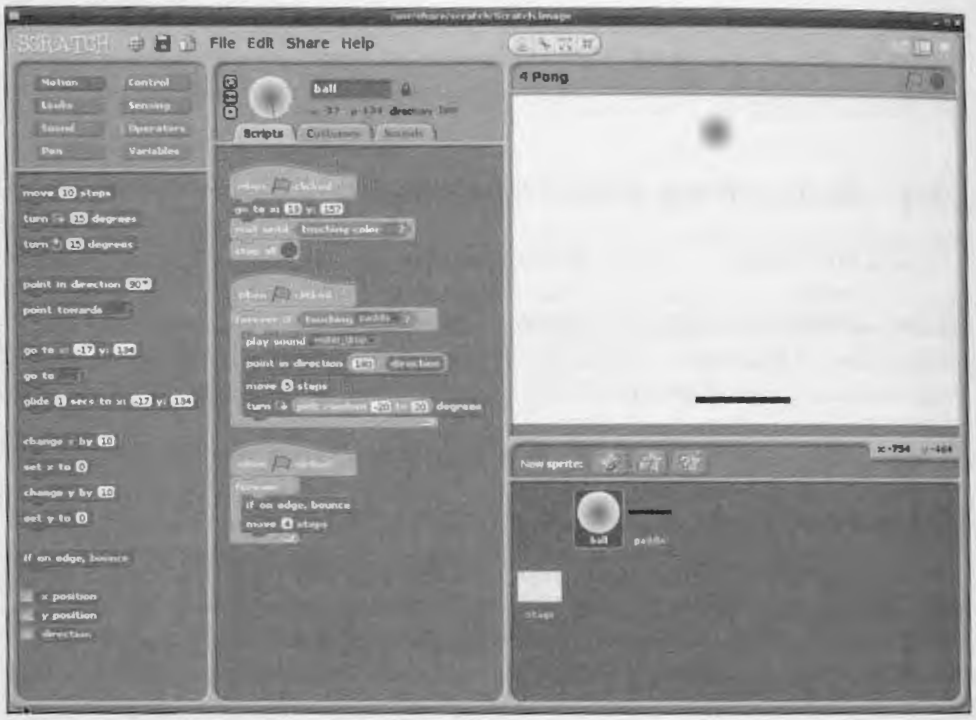


Bild 12.1: Ein Programm in Scratch bearbeiten

12.3.2 C

C ist die Programmiersprache, in der Linux geschrieben wurde, und in der Raspbian-Wheezy-Distribution ist der GNU-C-Compiler enthalten.

Um ein kleines Hello-World-Programm in C auszuprobieren, erstellen Sie mithilfe von IDLE eine Datei mit dem folgenden Inhalt:

```
#include<stdio.h>
main()
{
   printf("\n\nHello World\n\n");
}
```

Speichern Sie die Datei, nennen Sie sie `hello.c` und geben Sie in demselben Verzeichnis im Terminal folgenden Befehl ein:

```
gcc hello.c -o hello
```

Dadurch wird der C-Computer `gcc` ausgeführt, der `hello.c` in das ausführbare Programm mit dem Namen `hello` umwandelt. Dieses Programm können Sie dann wie folgt an der Kommandozeile ausführen:

```
./hello
```

Das Fenster des IDLE-Editors und die Kommandozeile mit der Ausgabe sehen Sie in Bild 12.2. Die `\n`-Zeichen sorgen dafür, dass die Meldung von Leerzeilen umgeben ist.

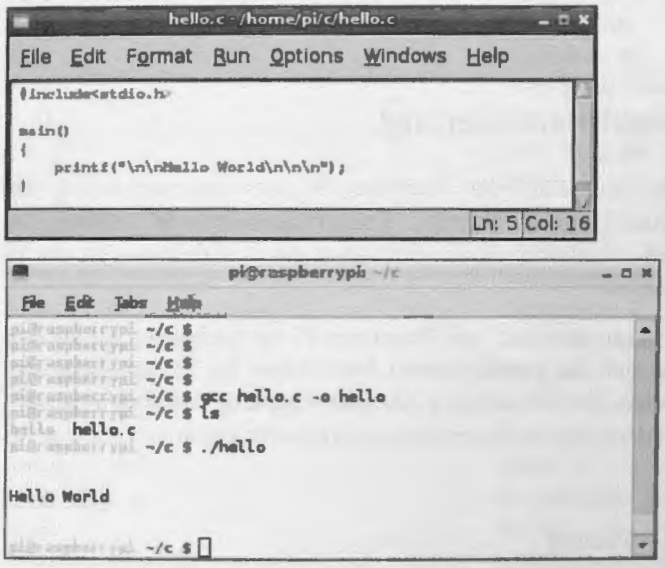


Bild 12.2: Ein C-Programm kompilieren

12.4 Anwendungen und Projekte

Jede neue Technologie, wie sie auch der Raspberry Pi darstellt, lockt innovative Enthusiasten an, die entschlossen nach interessanten Verwendungszwecken für diese Novität suchen. Zurzeit laufen unter anderem die folgenden bemerkenswerten Projekte.

12.4.1 Mediencenter (Raspbmc)

Raspbmc ist eine Distribution, die den Raspberry Pi in ein Mediencenter verwandelt, sodass Sie damit Filme und Audiodateien von einem angeschlossenen USB-Medium abspielen oder Audio- und Videodaten als Stream von anderen Geräten übertragen können, die an Ihr Heimnetzwerk angeschlossen sind, beispielsweise iPads. Raspbmc basiert auf dem erfolgreichen Projekt XBMC, das als eine Möglichkeit begann, die

Microsoft Xbox als Mediencenter zu nutzen, mittlerweile aber für viele Platt erhältlich ist.

Da der Raspberry Pi nur wenig kostet, ist damit zu rechnen, dass viele in kleine eingebaut und neben dem Fernseher aufgestellt werden, vor allem, da viele Fe geräte heute über einen USB-Anschluss verfügen, über den der Pi mit Strom v werden kann.

Auf *www.raspbmc.com/about/* erfahren Sie mehr über Raspbmc, auf *www.xbmc.or* über das Projekt XBMC. In allen Fällen handelt es sich natürlich um Open-S Software.

12.5 Haustechnikautomatisierung

Es gibt viele kleine Projekte, in denen der Raspberry Pi zur Automatisierun Haustechnik genutzt wird (auch **Domotik** genannt). Da sich Sensoren und Stellmo sehr leicht anschließen lassen – entweder direkt oder über einen Arduino –, ist sehr gut als Steuerzentrale geeignet.

Bei den meisten Vorgehensweisen wird auf dem Raspberry Pi ein Webserver im lo Netzwerk ausgeführt, sodass sich die verschiedenen Funktionen im Haus – beis weise das Ein- und Ausschalten der Beleuchtung oder die Regelung des Thermos über einen Browser steuern lassen, der sich irgendwo im Netzwerk befindet.

12.6 Zusammenfassung

Der Raspberry Pi ist ein äußerst flexibles und billiges Gerät, für das wir mit Siche viele nützliche Verwendungszwecke finden werden. Selbst als einfacher Computer Surfen auf dem Fernsehschirm ist er ideal geeignet (und viel billiger als alle and Möglichkeiten). Wenn Sie anfangen, bei sich zu Hause Projekte zu verwirklic werden Sie wahrscheinlich zusätzliche Raspberry Pis kaufen.

Nutzen Sie auch die Website zu diesem Buch (*www.buch.cd*). Dort finden Sie Softw die zum Download bereitsteht. Möchten Sie den Autor kontaktieren, können Sie über seine Webseite (*www.raspberrybook.com*) tun. Dort finden Sie auch die Errat diesem Buch.

Stichwortverzeichnis

Microsoft Xbox als Mediencenter zu nutzen, mittlerweile aber für viele Plattformen erhältlich ist.

Da der Raspberry Pi nur wenig kostet, ist damit zu rechnen, dass viele in kleine Kästen eingebaut und neben dem Fernseher aufgestellt werden, vor allem, da viele Fernsehgeräte heute über einen USB-Anschluss verfügen, über den der Pi mit Strom versorgt werden kann.

Auf *www.raspbmc.com/about/* erfahren Sie mehr über Raspbmc, auf *www.xbmc.org* mehr über das Projekt XBMC. In allen Fällen handelt es sich natürlich um Open-Source-Software.

12.5 Haustechnikautomatisierung

Es gibt viele kleine Projekte, in denen der Raspberry Pi zur Automatisierung der Haustechnik genutzt wird (auch **Domotik** genannt). Da sich Sensoren und Stellmotoren sehr leicht anschließen lassen – entweder direkt oder über einen Arduino –, ist der Pi sehr gut als Steuerzentrale geeignet.

Bei den meisten Vorgehensweisen wird auf dem Raspberry Pi ein Webserver im lokalen Netzwerk ausgeführt, sodass sich die verschiedenen Funktionen im Haus – beispielsweise das Ein- und Ausschalten der Beleuchtung oder die Regelung des Thermostats – über einen Browser steuern lassen, der sich irgendwo im Netzwerk befindet.

12.6 Zusammenfassung

Der Raspberry Pi ist ein äußerst flexibles und billiges Gerät, für das wir mit Sicherheit viele nützliche Verwendungszwecke finden werden. Selbst als einfacher Computer zum Surfen auf dem Fernsehschirm ist er ideal geeignet (und viel billiger als alle anderen Möglichkeiten). Wenn Sie anfangen, bei sich zu Hause Projekte zu verwirklichen, werden Sie wahrscheinlich zusätzliche Raspberry Pis kaufen.

Nutzen Sie auch die Website zu diesem Buch (*www.buch.cd*). Dort finden Sie Software, die zum Download bereitsteht. Möchten Sie den Autor kontaktieren, können Sie das über seine Webseite (*www.raspberrybook.com*) tun. Dort finden Sie auch die Errata zu diesem Buch.

Stichwortverzeichnis

Simon Monk

python

Raspberry Pi programmieren

Raspberry Pi verfügt mit Linux über ein ausgewachsenes Betriebssystem mit allen seinen Möglichkeiten. Für die Verwirklichung eigener Projekte, ob mit oder ohne Elektronik, kommt man um den Einsatz einer Programmiersprache nicht herum. Für Einsteiger, aber auch Profis bietet sich zu diesem Zweck die leicht erlernbare und weit verbreitete Skriptsprache Python an. Entwickler, die bereits eine andere Programmiersprache kennen, finden schnell Zugang zu dieser Sprache. Neulingen macht Python durch seine verständlichen Konstrukte den Einstieg leicht.

Raspberry-Pi-Einstieg

Einsteigern in die Welt des Minicomputers Raspberry Pi zeigen die ersten beiden Kapitel, worauf es beim Raspberry Pi wirklich ankommt. Neben dem Aufbau der Platine und den wichtigsten Anschlüssen werden die Grundlagen des Betriebssystems dargestellt.

Programmierung mit Python

Ausführlich werden alle essenziellen Bestandteile der Programmierung mit der Skriptsprache Python aufgezeigt – von Kontrollstrukturen wie if- oder for-/while-Schleifen bis hin zur Arbeit mit Strings und Listen.
Aufbauend auf diesen Grundlagen werden erweiterte Themen wie Module und Objektorientierung behandelt.

Praxisprojekte mit dem Raspberry Pi

Das aufgebaute Wissen zur Programmierung mit Python wird anhand von Praxisbeispielen gefestigt. Ein Kapitel widmet sich komplett der Entwicklung von Spielen mit Python. Nach einer Einführung in die GPIO-Schnittstelle werden zwei Hardwareprojekte anhand von Programmcode vorgestellt: eine LED-Uhr und ein Roboter auf Basis von Raspberry Pi. Alle Hardwareprojekte werden mit der Skriptsprache Python durchgeführt.

Aus dem Inhalt:

Über den Autor:

Dr. Simon Monk (Preston, UK) hat einen Abschluss in Kybernetik und Informatik und einen Ph.D. in Software Engineering. Monk hat mehrere Jahre in der Forschung verbracht, bevor er in die Industrie zurückwechselte und als Mitgründer das Unternehmen Momote Ltd. für mobile Software aufbaute.

Auf www.buch.cd

Der komplette Quellcode des Buchs

9 783645 602617

30,– EUR [D] / **30,90 EUR** [A]
ISBN 978-3-645-60261-7

Besuchen Sie unsere Website
www.franzis.de

FRANZIS