# Raspberry Pi
A Quick-Start Guide

Maik Schmidt

Edited by Jacquelyn Carter

Early Praise for Raspberry Pi

The Raspberry Pi is bringing back the golden days of experimenting with home computers and Maik's book is an ideal starting point. The included projects are perfect for Raspberry Pi users of any age or level of experience.

➤ **Tony Williamitis, Senior Embedded Systems Engineer**

Schmidt takes a quick dip into many of the things you can do with a Raspberry Pi straight out of the box. I found it very useful for understanding exactly what I can use my Pi for, and it's given me some ideas for what I can do next!

➤ **Stephen Orr, Technical Enthusiast and Web Developer**

This is the owner's manual all Raspberry Pi buyers should get before they start diving in. It's clear, comprehensive and succinct. I couldn't ask for more.

➤ **Thomas Lockney, Professional Geek**
  DorkbotPDX

A wonderfully clear, concise, and useful introduction to the Raspberry Pi.

➤ **Michael Hunter**

Raspberry Pi

A Quick-Start Guide

Maik Schmidt

The Pragmatic Bookshelf

Pragmatic Bookshelf

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

The team that produced this book includes:

Jacquelyn Carter (editor)
Kim Wimpsett (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Contents

Acknowledgments

Whenever I tell people that I am an author, they look at me dreamily for a few seconds. Obviously, many people think that writing is about sitting at an old wooden desk, staring outside the window on a stormy day, and enjoying a good glass of red wine. For me this has rarely been the case, but still most of the time I have a lot of fun while writing books.

I had a lot of fun writing this book, too—mainly because of the invaluable support of my editor, Jacquelyn Carter. She cheered me up on countless occasions, and her thoughtful advice made most of my problems disappear immediately. Thank you very much, Jackie!

As always, the whole team at the Pragmatic Bookshelf has been tremendously helpful and agile. Without you, this book would have been impossible!

This book deals with electronics, and I have created all the circuit diagrams with Fritzing.[1] I am deeply grateful that the Fritzing team has made such a great tool available for free. Also, I have to thank Gordon Henderson for WiringPi.[2] It makes working with the Raspberry Pi's GPIO pins a piece of cake, and it saved me countless hours of debugging low-level code.

Simon Quernhorst kindly gave me permission to use screenshots of his great game *A-VCS-tec Challenge* in this book.

I cannot thank my reviewers enough: Daniel Bachfeld, Gordon Haggart, Michael Hunter, Thomas Lockney, Angus Neil, Stephen Orr, Mike Riley, Sam Rose, Mike Williamitis, and Tony Williamitis. Your comments and suggestions made this book so much better.

Finally, I have to thank Tanja and Mika for being so patient and understanding. I am so glad I have you!

---

1. http://fritzing.org/
2. https://projects.drogon.net/raspberry-pi/wiringpi/

Preface

Over the past decades computers have gotten cheaper and cheaper, so today you can find them not only beneath your desk but in nearly every consumer electronics device such as smartphones or DVD players. Still, computers aren't so cheap that you spontaneously buy one when shopping for your groceries. Usually, you carefully plan your next PC, because you have to use it for a couple of years.

Computers like the Raspberry Pi will change the situation completely in the near future. The Raspberry Pi, or Pi for short, is a full-blown desktop PC that costs only $35. You can directly connect it to the Internet, and it is able to display high-definition videos. Also, it runs Linux, so you do not have to pay for an operating system. This makes the Pi probably the first throwaway computer in history.

Originally, the Raspberry Foundation[1] built the Pi to teach children how to program, so it comes as no surprise that the Pi is an excellent device for exactly this purpose. On top of that, you can use the Pi for many other exciting things. For example, you can turn it into a multimedia center, use it as a cheap but powerful web server, or play some classic games.

The Pi is also a great machine for experimenting with electronics. In contrast to many popular microcontroller boards like the Arduino, the Pi runs a full-blown operating system, and you can choose from a wide range of programming languages to implement your projects.

With cheap and small devices like the Raspberry Pi, a new era of ubiquitous computing has begun, and you can be part of it. This book helps you get up to speed quickly.

---

1. http://www.raspberrypi.org/

Who Should Read This Book?

This book is for everyone who wants to get started with the Raspberry Pi. Even if you have some experience with other computers, you'll quickly see that the Pi is different in many regards, and this book helps you avoid the most common pitfalls.

You can choose from a variety of operating systems for the Pi, but this book's focus is on Debian Linux (Raspbian), because it is the most convenient choice for beginners. If you've never worked with Linux before, you should start with Appendix 1, *A Linux Primer*, on page 101. Even if you've worked with Linux before, you still might learn a few things, because running Linux on the Pi is different in some ways.

Of course, you'll get the most out of this book if you have a Raspberry Pi and follow all the book's examples closely.

What's in This Book?

The Raspberry Pi does not come with a user guide, but in this book you'll learn step-by-step how to get the most out of your mini-computer quickly. You'll learn not only how the Pi's hardware works in principle but also how to run different operating systems and use the Pi for special purposes such as turning it into a multimedia center.

Here's a list of all the things you're going to learn:

• The book starts with an introduction to the Raspberry Pi's hardware. You'll learn what the Pi's connectors are for and which additional hardware you need to start the Pi for the first time.

• After you've connected all necessary devices to your Pi, you need an operating system. Although the Pi is a fairly young project, you can already choose from several, and you'll learn what their pros and cons are.

• Installing an operating system on the Pi is quite different from installing an operating system on a regular PC. So, you'll learn how to get Debian Linux up and running on the Pi.

• Debian Linux runs fine out of the box on the Pi, but to get the most out of it, you have to tweak a few configuration parameters. For example, it's beneficial to set the right layout for your keyboard. In addition, you'll learn how to install, update, and remove new software.

- The Pi's hardware, especially its graphics hardware, is special in many regards. Depending on the display you're using, you have to adjust some low-level settings for the Pi's firmware. You'll learn what settings are available and how to solve the most common firmware problems.

- To see what can be achieved with the Pi with a minimum of effort, you'll turn it into a kiosk system. It will be able to display a set of static slides as well as live information from the Internet.

- Until this point, you've used the Pi more or less in isolation, but now you'll learn how to integrate it with networks. You'll use the Pi for everyday tasks such as browsing the Web, you'll make it accessible via Secure Shell, and you'll even turn it into a full-blown web server. Also, you'll learn how to share your Pi's desktop with a PC, and vice versa.

- With the XBMC project, you can turn your Raspberry Pi into a multimedia center with ease. Not only can you show your photos collections to your friends in your living room, but you can also play music in all popular formats, and you can watch your favorite movies and TV shows in high definition.

- The Raspberry team originally built the Pi for educational purposes, but you can easily use it to play some entertaining games. Even though it's possible to run some first-person shooters, you might prefer some classic genres such as interactive fiction and point-and-click adventures.

- One of the greatest advantages the Pi has over regular PCs is its GPIO pins. In the book's final chapter, you'll learn how to easily use them to attach your own electronics projects to the Pi.

- The appendix contains a short introduction to Linux. If you've never worked with Linux before, you should read the appendix before you start with Chapter 3, *Configure Raspbian,* on page 21.

Where Can I Get a Raspberry Pi and Additional Hardware?

At the time of this writing, only two distributors produce and sell the Raspberry Pi. To buy a Pi, visit the web shops of Farnell[2] or RS Components.[3] These shops also sell many accessories such as power supplies, keyboards, mice, and so on, for the Pi. Adafruit[4] sells useful accessories, too.

---

2. http://www.farnell.com/
3. http://www.rs-online.com/
4. http://adafruit.com/category/105

You can find a growing list of compatible hardware on the project's wiki,[5] but when in doubt, it's better to buy hardware from one of the shops mentioned here.

Debian Linux

The most popular operating system for the Pi is Linux. Several Linux distributions are available for the Pi, and we chose Debian. Recently the Debian team has frozen the latest version named *wheezy*, and because of the great efforts of the Raspbian team,[6] it is available for the Pi already. Raspbian supersedes Debian squeeze, which has been the reference operating system for the Pi for a long time.

The Raspbian distribution has many advantages over all its predecessors. It is much faster, it has more recent software, and it will soon be more stable. Also, it is the preferred solution of the Raspberry team, so this book's focus is on Raspbian.

Code Examples and Conventions

In this book you'll find a few code examples written in PHP, in HTML, and in the programming language of the Bash shell. They are all very short, and if you've done some programming before, you'll have no problems understanding them. If you haven't developed software before, you'll still be able to copy the code to the Pi and make it run.

Online Resources

This book has its own web page at http://pragprog.com/titles/msraspi where you can download the code for all examples, or you can click the file name above each code example to download the source file directly. On the web page, you can also participate in a discussion forum and meet other readers and me. If you find bugs, typos, or other annoyances, please let me and the world know about them on the book's errata page.

Now it's time to unbox your Raspberry Pi and have some real fun!

---

5.   http://elinux.org/RPi_VerifiedPeripherals
6.   http://www.raspbian.org/

Meet the Raspberry Pi

Before you start the Raspberry Pi for the first time, you should make yourself familiar with its connectors and its capabilities. This will help you decide what kind of projects you can use the Pi for, and it will help you understand what kind of additional hardware you'll need. For example, you'll need a power supply, a keyboard, a mouse, and a display. In this chapter, you'll learn which devices work best.

1.1 Get to Know the Hardware

Unboxing a new Pi is exciting, but it certainly is not comparable to unboxing a new Apple product. Usually, the Pi comes in a plain cardboard box with one or two sheets of paper containing the usual safety hints for electronic devices and a quick-start guide.

The first version of the Pi looks attractive only to the real geeks. It is a single-board computer without a case, and it's the size of a credit card. Somehow it resembles the innards of the many electronic devices you might have opened when you were a child. Later versions of the Pi might have a case, but until then, we have to focus on its inner values, and that's what counts, isn't it?

What's on the Pi

The Pi will be available in two flavors named Model A and Model B. Model A is a bit cheaper and does not have as many connectors as Model B. I'll explain their differences in detail in the following text, but because at the time of this writing Model A is still not available, I'll cover only Model B in the rest of this book. You can see it in .
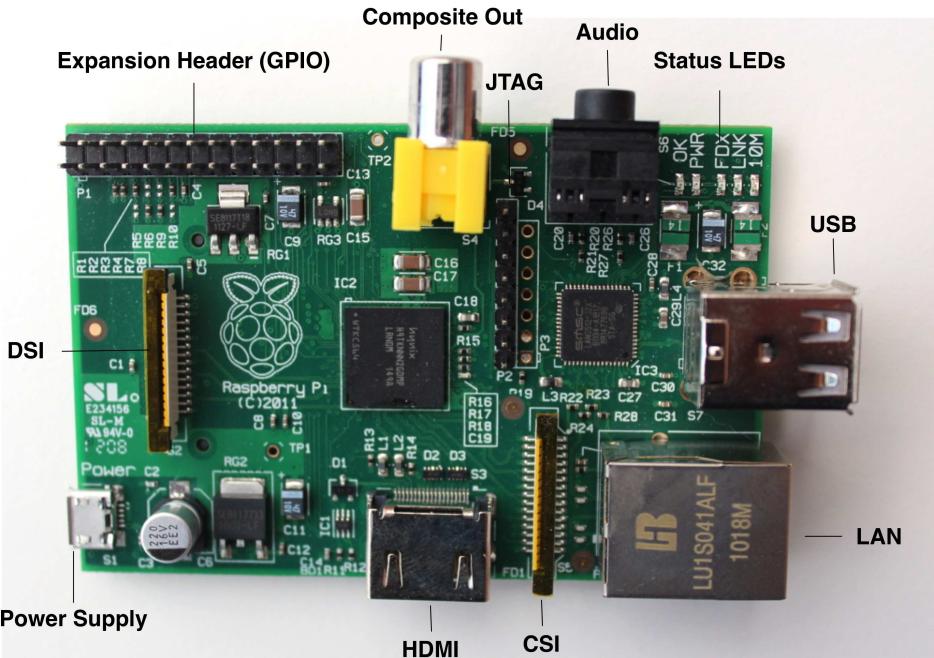
Figure 1—The front side of a Model B

All Raspberry Pi models have the same heart and brain: a system on a chip (SoC) named BCM2835[1] that you can find in many mobile phones. It's cheap, it's powerful, and it does not consume a lot of power. These characteristics made it a perfect choice for the Raspberry team.

In contrast to a typical PC architecture, a SoC integrates a processor (CPU), a graphics processing unit (GPU), and some memory into a single unit. The BCM2835 contains an ARM1176JZ-F processor running at 700MHz, 256MB of RAM, and a GPU named VideoCore IV. For purists, this GPU is a bit problematic because its design and its graphics drivers are proprietary; that is, their source code is not publicly available. This will probably not affect you in your daily work with the Pi, but it really is a problem for some strong proponents of free software.

The Pi has many connectors, and most of them look familiar. On a Model B board, you can find two regular-sized USB ports that you can use to connect a keyboard and a mouse, for example. You'll also find a micro-USB port, but

---

1.   http://www.broadcom.com/products/BCM2835

you'll need it to power the Pi, and you cannot use it to connect more devices. If you need to connect more devices, you have to connect them to a USB hub. The Model A board has only a single USB port, so you'll probably always need a USB hub.

You can connect the Model B to a network directly using its Ethernet (LAN) port. Model A does not have an Ethernet port, but you can add one by attaching a USB-to-Ethernet converter. Interestingly, Model B uses its internal USB hardware for networking, too, so there'll be no difference in networking performance between a Model B and a Model A with a USB-to-Ethernet adapter.

To connect the Pi to a display or a TV set, you have two options: the Pi has ports for connecting both HDMI and composite video. The digital HDMI standard is way more powerful than its much older brother, the analog composite standard. With HDMI, you can transmit high-definition video in crystal-clear quality, while the composite output is limited to what the older geeks know as "the childhood TV." Using composite video, you cannot display high-definition graphics, and the output usually flickers a bit. Its biggest advantage is that you can still find many TV sets that have a composite connector, but HDMI is gaining ground quickly. By the way, the Raspberry team did not add a VGA connector because it thinks that VGA is at the end of its life. Of course, you can use an adapter to connect the Pi's HDMI output to a DVI or VGA display.

With HDMI you can also transmit both video and sound, but if you're using composite video, you'll need a separate connector for sound output. That's what the audio jack is for—you can connect it to headphones, to speakers, or to your audio receiver using a standard 3.5mm plug.
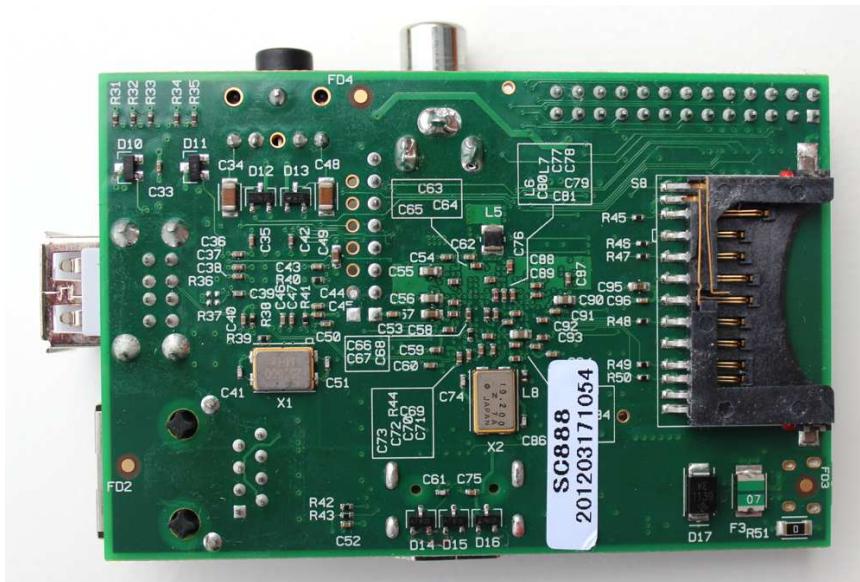
To the left of the composite video connector, you can see an expansion header that consists of two rows of pins. Most of these pins are general-purpose input/output pins (GPIOs), and you can use them to connect the Pi to other electronic devices. As you might have guessed from their name, they do not have a special purpose, so you can do a lot of different things with them. For example, you can use them to connect your good ol' Atari VCS 2600 game controllers to the Pi so you can run your favorite 8-bit games in an emulator. In Chapter 9, *Tinker with the GPIO Pins*, on page 87, you'll learn how to use the expansion header, and you'll build a small hardware project.

On the board you can find several other connectors. The CSI connector[2] is meant for connecting a camera to the Pi. With the DSI connector,[3] you can connect a display, and the JTAG headers[4] help you debug your hardware projects.

The board also has five status LEDs that have the following meanings:

- The *OK* LED indicates SD card access; it blinks whenever the Pi tries to access the SD card. You can control this LED by software, so it's not completely accurate.

- As soon as you connect a power supply to the Pi, the *PWR* LED turns on.

- The *FDX* LED shows whether your LAN is running full duplex.

- At every LAN activity, the *LNK* LED blinks.

- The *10M* LED indicates whether the Pi's Ethernet link is running at 10Mbit/s or 100Mbit/s. When this LED is on, the Pi runs at 100Mbit/s.

In the following graphic, you can see the back side of a Pi, and you can also see a slot for an SD card on the right side.

The Pi has no persistent internal memory, so you have to boot it from an SD card. You might have worked with SD cards before, because they are very

---

2. http://en.wikipedia.org/wiki/Camera_interface
3. http://en.wikipedia.org/wiki/Display_Serial_Interface
4. http://en.wikipedia.org/wiki/Jtag

popular as storage media in cameras, cell phones, and portable game consoles. They are available in different sizes and with different capacities, usually ranging from 1GB to 64GB (see the following graphic).

What the Pi Does Not Have

Taking its cheap price into account, the Pi comes with a lot of nice things already, but it also lacks some useful features. For example, the Pi does not have a real-time clock (RTC) with a backup battery, and it does not have a Basic Input Output System (BIOS).[5] You can easily work around the missing clock using a network time server, and most operating systems do this automatically, but the lack of a BIOS is a bit more severe.

Simply put, a BIOS is a program stored in read-only memory (ROM) that runs on a PC at startup. Among other things, it's responsible for configuring new devices and for determining the boot order. For example, using the BIOS, you can specify whether you'd like to boot from your hard drive or from a DVD. The Pi has no BIOS, so it always boots from an SD card. Even if you have a perfectly valid installation of an operating system on a USB stick or an external hard drive, you cannot boot it. Of course, you can still use external storage devices, but you cannot use them to boot the Pi.

The Pi does not support Bluetooth or WiFi out of the box, but you can add support for both of them using USB dongles. Unfortunately, most Linux distributions are still a bit picky about their hardware, so you should first check whether your flavor of Linux supports your particular device. (See Section 3, *Where Can I Get a Raspberry Pi and Additional Hardware?*, on page xi for some advice about where to get compatible hardware.) All this is true for

---

5. http://en.wikipedia.org/wiki/BIOS

other types of hardware such as microphones or webcams. As long as your operating system and your applications support your devices, you'll be fine. Otherwise, you'd better look for an alternative that is known to work on your operating system.

You now know what all the connectors on the Pi are for, and in the next section, you'll learn what devices you can actually connect to the Pi.

1.2 What Else You Need

After unboxing the Pi for the first time, you'll quickly realize that the Raspberry team obeys the BYOP mantra.[6] The box contains nothing but the board; you'll need a couple of other things to get it up and running. Most of them you'll probably have at home already.

Choose a Power Supply

First you need a power supply with a Micro USB connector, because currently the Pi does not ship with one. According to the Pi's specification, both models need a power supply that outputs 5V. The power supply should source 300mA for a Model A and 700mA for a Model B. Depending on the devices you connect to the Pi, it might have to source even more.

Many cell-phone chargers meet the Pi's requirements, and this is not a coincidence. The Raspberry team wanted the Pi to work with cell-phone chargers because of their ubiquity. I've used the charger of a Samsung Galaxy S II for a couple of days, and it worked well for my first experiments. When I started to add more devices, it was no longer sufficient, and I replaced it with a wall charger from Belkin (see Figure 2, *A USB wall charger*, on page 7). It outputs 1A and worked better, but for some hardware setups, you still need more power.

The Pi's biggest limitation regarding the power supply is that no external device should draw more than 100mA from any of its USB ports. So, as long as your keyboard and your mouse need 100mA each, everything works fine. Usually, you can find a small sticker with the power characteristics on the back of a device. If one device draws more than 100mA, sooner or later you'll observe strange effects.[7] To be on the safe side use a power supply that delivers 1A to 1.2A for the Model B. For Model A it should be between 500mA and 700mA.

_____

6. Bring Your Own Peripherals
7. http://elinux.org/RPi_Hardware#Power

You can unburden the Pi with a powered USB hub, but it doesn't work with every product. So, before you buy something for your Pi, it's best to take a look at the project's wiki.[8]

Figure 2—A USB wall charger

Choose an SD Card

Even with a perfect power supply, a Pi will not do much when you start it, because it needs an SD card with an operating system. You can buy preloaded SD cards,[9] but you can also start with an empty card and prepare it yourself (see Section 2.2, *Prepare a Bootable SD Card*, on page 14 for how to do this). Usually, this is the better approach, because it makes sure that you get the latest and greatest software for your Pi. For example, at the moment of this writing, all preloaded SD cards still contain Debian squeeze, which has been superseded by Debian wheezy (Raspbian) already.

---

8. http://elinux.org/RPi_VerifiedPeripherals
9. http://uk.farnell.com/raspberry-pi-accessories#operatingsystem or http://uk.rs-online.com/web/p/flash-memory/7631030/?

Some users have reported problems with incompatible SD cards, so when in doubt, you should take a look at Section 3, *Where Can I Get a Raspberry Pi and Additional Hardware?, on page xi*. In principle, you can use a card of any size. Of course, the minimum size depends on your operating system, on the applications you're going to install, and on the data you're going to create on the Pi later. As often in life, bigger is better, and you should use a card with a capacity of at least 4GB for convenient Pi experience.

Connect a Keyboard and a Mouse

Unless you're planning to use the Pi as a headless system,[10] you'll need a keyboard and a mouse. Probably you have a spare keyboard and a spare mouse at home, and as long as they have a USB connector, they'll probably work with the Pi. Sometimes keyboards with an internal USB hub cause problems, because they steal some current from the Pi that it might need for other things. If you experience strange effects such as an unresponsive keyboard or infinite repetitions of keystrokes, try another keyboard first or connect it to the Pi using a powered USB hub. It's best if your keyboard and your mouse consume only 100mA each.

Some wireless keyboards and mice also will not work properly, because Linux does not support them all. In the beginning, you'd better be conservative and use wired equipment until everything works as expected. Then start to replace components one by one, and in case of problems, check to see whether your operating system supports your particular keyboard or mouse.

Often you'll need even more than two USB devices (or one, if you have a Model A), so you'll have to connect them using a USB hub to the Pi. Make sure the hub delivers enough current to power all connected devices. In nearly all cases, you'll need a hub that has its own power supply.

Choose a Display

Depending on the display you're going to use, you need an HDMI cable or a composite-video cable. If you're using HDMI and your display also has audio output, you're done. Otherwise, you have to connect the Pi's audio jack to your sound system using a cable with a standard 3.5mm TRS connector. It's the same connector you can find at the end of your iPod's headphones, and of course you can use these, too.

---

10. http://en.wikipedia.org/wiki/Headless_system

Choose the Right Network Equipment

If you want to connect a Model B to a network, you need only an Ethernet cable. Model A does not have an Ethernet port, so to connect a Model A to a network, you need a USB-to-Ethernet converter.

Add a Case

Future releases of the Pi might come with a case, but until then, you have to protect it yourself. Like every electronics device, the Pi is sensitive to dust and conductive surfaces, so sooner or later you should hide it in a case.

The Pi community is very creative, and already people have created cases using Lego bricks[11] and even paper.[12] One of the biggest problems with most self-made cases is that they usually do not offer a convenient access to the Pi's connectors. So, the best solution often is to buy a professional case, for example from Adafruit[13] or from ModMyPi.[14]

In addition to all the devices mentioned, you need a separate PC for some tasks such as copying an image to an SD card or cross-compiling applications. So, all in all, setting up a Pi is not as cheap as it sounds in the beginning.

A typical Pi setup looks quite messy on your desk after you've connected all cables (see below). Despite its look, the hardware is ready for a first test run!

---

11. http://www.raspberrypi.org/archives/1515
12. http://squareitround.co.uk/Resources/Punnet_net_Alpha3.pdf
13. https://www.adafruit.com/products/859
14. http://modmypi.com

1.3   Next Steps

In this chapter, you learned what all the connectors on the Pi are for, and you learned what additional devices you need and how to choose the right ones. In principle, you could start the Pi for the first time, but it will not do much without an operating system. In the next chapter, you'll learn what your options are and how to install a full-blown Linux system.

Install an Operating System

Like every computer, the Raspberry Pi needs an operating system, and the preferred one for the Pi is Linux. That's partly because it's free, but mainly it's because it runs on the Pi's ARM processor while most other operating systems work only on the Intel architecture. Still, not every Linux distribution will run on the Pi, because some do not support the Pi's particular type of ARM processor. For example, you cannot install Ubuntu Linux on a Pi. So, in this chapter, you'll first learn what your options are.

Choosing an operating system is only a first step, because you also have to install it. The installation procedure on the Pi is quite different from what you're probably used to, but it's not difficult: you need to install the operating system on an SD card. In this chapter, we're going to install the latest Debian Linux distribution, but the process is the same for all operating systems. You can actually create several SD cards, each with a different operating system, so at the end you'll have a pretty versatile system that you can turn into completely different machines by simply replacing the card.

2.1 See What's Available
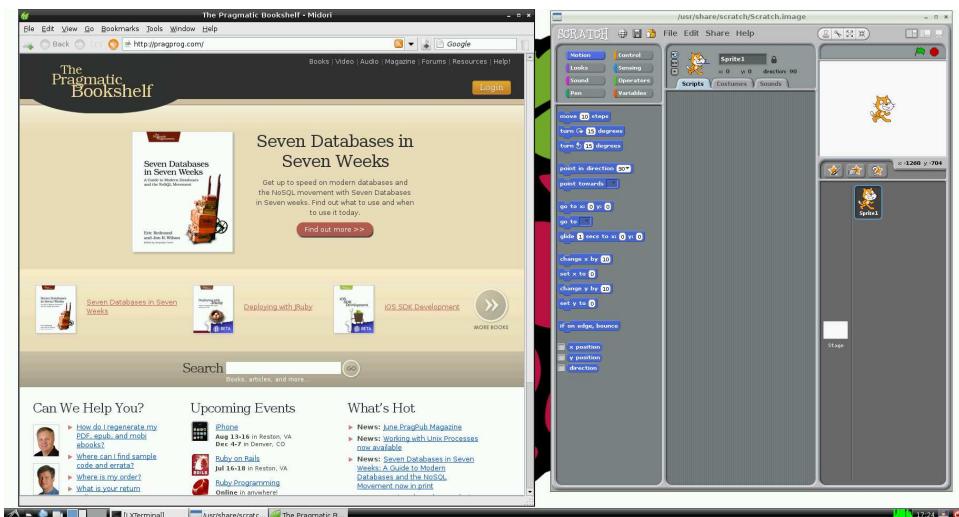
Linux is still the most popular choice for an operating system on the Pi, and it helps you to get the most out of the Pi. Also, many people are already familiar with Linux, while the other operating systems running on the Pi are a bit more exotic.

Even if a Linux distribution runs on the Pi, it will often look and behave different from its "regular desktop PC" equivalent, because it might use a windows manager that does not need a lot of resources. Also, you won't find all of the applications you're used to such as many popular web browsers or office products.

In addition to all that, there are some limitations around installing the operating system. Modern operating systems are fairly big, and they ship on DVD or are available as ISO image downloads. These images and DVDs contain the full installation process for the operating system; they start a program that detects your computer's hardware, and then they copy all files needed to the hard drive. Unfortunately, you can't insert a DVD into the Pi and install it, and because the Pi has no BIOS (see *What the Pi Does Not Have*, on page 5), you cannot boot from an external USB drive either. You also cannot copy an ISO image of a DVD to an SD card. Instead, we need a snapshot of a system that has already been installed and that we can boot from.

So, you have to create or find an image of a Linux distribution that you can copy to an SD card, and it has to be compatible with the Pi. The easiest way to get such an image is to visit the download page of the Raspberry project.[1] At the time of this writing, you can find images for Raspbian (Debian wheezy), Arch Linux ARM, and Qton Pi. More operating systems will certainly appear in the future; at the least, Bodhi Linux[2] is already available today. Also, some clever folks are currently trying to port Google's Chrome OS.[3]

At the moment, the best choice for your first steps with the Pi is Raspbian (Debian wheezy). It fully supports the Pi's hardware, it comes with a full-blown desktop (see the image below), and it contains some useful applications such as a web browser.

---

1. http://www.raspberrypi.org/downloads
2. http://jeffhoogland.blogspot.co.uk/2012/06/bodhi-linux-arm-alpha-release-for.html
3. http://www.cnx-software.com/2012/04/19/building-chromium-os-for-raspberry-pi-armv6/

On top of that, it has a powerful package manager that makes it very easy to install more software. We'll use Debian in the rest of this book, and in the next section you'll learn how to install it. Note that we'll use the names Raspbian and Debian interchangeably.

The other distributions are very interesting, too, but they target a different audience. Still, I'll briefly describe them in the following sections.

Arch Linux ARM

Arch Linux[4] is very minimalistic and assumes that you have a fair amount of Linux knowledge already. Arch Linux does not use many resources, and it also has a nice package manager, so it's a good choice when you want to use the Pi as a server. For a desktop system, Debian is more convenient, though, because by default Arch Linux does not ship with a desktop environment. You have to install and configure it yourself.

Qton Pi

Qton Pi is a distribution with a special focus on Qt,[5] an application development framework that makes it easy to create applications with a nice and rich graphical user interface. Qt comes with a powerful SDK that includes a full-blown IDE named *Qt Creator*. The framework is powerful enough even for creating games, and it's cross-platform, so you can run your Qt applications wherever the Qt framework is available.

When the Qt team heard about the Raspberry Pi, they immediately decided to create the Qton Pi Linux distribution. It comes with everything related to Qt and is a perfect environment for developing Qt applications. But it's a cross-compiling environment—you have to install the development environment on your PC. You use the Qt Creator to build applications on the PC, and then you can transfer the results automatically to the Pi using SSH.[6] So, if you're not interested in developing Qt applications, Qton Pi will not be the right distribution for you.

Special-purpose distributions like Qton are common in the Linux world. In Chapter 7, *Turn the Pi into a Multimedia Center*, on page 69, you'll get to know Raspbmc, a Linux distribution that will turn your Pi into a multimedia center.

---

4. http://www.archlinux.org/
5. http://qt-project.org/
6. See http://qt-project.org/wiki/Create for details.

Try Something Exotic

But the Pi doesn't only run Linux; it also runs RISC OS,[7] for example. This comes as no surprise, because RISC OS was one of the first operating systems designed for the ARM architecture. It still has a lot of fans and is definitely worth a look.

Even though you cannot change the Pi's hardware easily, you can still turn it into many different machines within a second: simply insert an SD card containing another operating system. In the next section, you'll learn how to prepare such an SD card.

2.2 Prepare a Bootable SD Card

As we saw in Chapter 1, *Meet the Raspberry Pi,* on page 1, the Raspberry Pi doesn't have a BIOS or internal persistent storage. It has only an SD card slot. You use a separate computer to install the Pi's operating system on an SD card that you use to boot the Pi. Fortunately, some people have done this already for several operating systems, and they've kindly made available the content of such SD cards for free on the Internet. In this chapter, you'll learn how to transfer an SD card image to an SD card.

You will need a PC with a card reader (which is quite a misnomer, because you can use it for writing, too) to modify the SD card. Some PCs have built-in readers, but you can also get USB readers for a few dollars. In principle, it doesn't matter which operating system you use, and we're going to look at how to create the SD card on all major platforms. If you have access to a Windows box, I strongly suggest you use it, because it's easier and more convenient than Mac OS X or Linux for this particular purpose. Preparing an SD card on Mac OS X or Linux isn't rocket science, but you have to invoke a fairly dangerous command, and you can easily delete some important files. Also, on Windows you'll get more feedback while copying the card image. A better Raspbian installer is in the works, but until then, you have to copy the SD card image yourself.

No matter what operating system you're going to use for the installation process, you have to download the Debian image from the official download site.[8] You can download it using HTTP or via Torrent. After the download has finished, you should have a file named 2012-07-15-wheezy-raspbian.zip on your local hard drive (the filename might vary, if a new version has been released).

---

7. http://en.wikipedia.org/wiki/RISC_OS
8. http://www.raspberrypi.org/downloads

The procedures described in the following sections will be the same for images of all operating systems compatible with the Pi. You have to replace only the name of the image file.

Prepare an SD Card on Windows

Preparing the SD card on a Windows box is the most convenient alternative, because of Win32DiskImager.[9] This small application is free, has a nice user interface, and has a single purpose: writing images to SD cards. You do not even have to install it; it's sufficient to download the ZIP file from the project's website and unzip it to a directory of your choice. Double-click Win32DiskImager.exe, and you're ready to go.
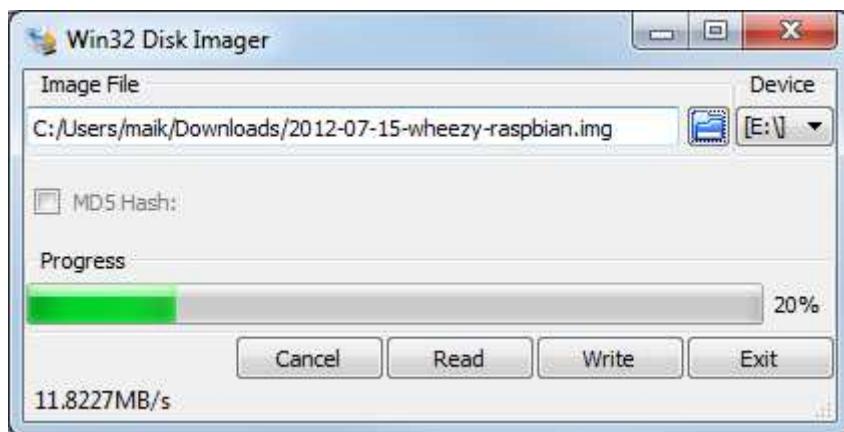
Figure 3—Win32DiskImager in action

Before you write the SD card image to an SD card, you can optionally check whether the image is valid. Therefore, you have to calculate the ZIP file's SHA1 checksum. To do this, you have to install the fciv command; Microsoft's support site has a lot of information about it.[10] After you've installed fciv, you can use it as follows:

```
C:>fciv 2012-07-15-wheezy-raspbian.zip -sha1
//
// File Checksum Integrity Verifier version 2.05.
//
3947412babbf63f9f022f1b0b22ea6a308bb630c 2012-07-15-wheezy-raspbian.zip
```

9.　http://www.softpedia.com/get/CD-DVD-Tools/Data-CD-DVD-Burning/Win32-Disk-Imager.shtml

10.　http://support.microsoft.com/kb/841290

If the long hexadecimal number is the same as on the download page, the ZIP file has not been compromised, and you can safely proceed. Otherwise, download the image from another location.

After the application has started, you have to select the Debian image and the drive letter of your card reader. Be very careful, and make sure you do not choose the wrong drive! Otherwise, you risk losing important data! Then click the Write button, and you should see something like . Writing the image will take a few minutes, but then you'll have an SD card you can use to boot the Pi.

Prepare an SD Card on Linux

Preparing an SD card for the Pi on a modern Linux system is not too difficult, but you have to be very careful when performing the following steps, because you can easily destroy important data! Do *not* insert the SD card into your card reader right now. You'll do it later in the process to determine the device name of your reader.

Download the ZIP file containing the Debian image from the official download site, open a terminal, and change to the directory containing the ZIP file you've just downloaded. Although it's not necessary, it doesn't hurt to check the integrity of the file you've downloaded.

```
maik> sha1sum 2012-07-15-wheezy-raspbian.zip
3947412babbf63f9f022f1b0b22ea6a308bb630c  2012-07-15-wheezy-raspbian.zip
```

If the long hexadecimal number is the same as on the download page, the ZIP file has not been compromised, and you can safely proceed. Otherwise, download the image from another location.

The following command unzips the image file to the current directory:

```
maik> unzip 2012-07-15-wheezy-raspbian.zip
Archive:  2012-07-15-wheezy-raspbian.zip
  inflating: 2012-07-15-wheezy-raspbian.img
```

Next you have to determine the location of your card reader. Run the following command to get a list of all storage devices currently connected to your computer:

```
maik> df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        63G   15G   46G  24% /
udev            494M  4.0K  494M   1% /dev
tmpfs           201M  740K  200M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            501M  124K  501M   1% /run/shm
```

Insert the SD card into your reader and run the command again.

```
maik> df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        63G   15G   46G  24% /
udev            494M  4.0K  494M   1% /dev
tmpfs           201M  772K  200M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            501M  124K  501M   1% /run/shm
/dev/sdc2       1.6G  1.2G  298M  81% /media/18c27e44-ad29-4264-9506-c93bb7083f47
/dev/sdc1        75M   29M   47M  39% /media/95F5-0D7A
```

As you can see on my system, the SD card is named sdc, and it has two partitions, named sdc1 and sdc2. Of course, this will vary on your system; that is, you might have more or fewer partitions, and your SD card might be named sdd, for example. Before you proceed, you have to unmount all partitions, so in this case, you'd have to invoke the following commands:

```
maik> umount /dev/sdc1
maik> umount /dev/sdc2
```

As a final step, you have to copy the image to the SD card. You have to run the following command with root privileges and make sure you're using the right device name for the of option.

```
maik> sudo dd bs=1M if=2012-07-15-wheezy-raspbian.img of=/dev/sdc
[sudo] password for maik:
1850+1 records in
1850+1 records out
1939865600 bytes (1.9 GB) copied, 160.427 s, 12.1 MB/s
```

Copying the image will take a few minutes, but if everything went fine, you have a bootable SD card that will bring Debian to your Pi!

Prepare an SD Card on Mac OS X

Preparing an SD card containing Debian on a Mac is very similar to preparing one on Linux, but there are a few important differences. You have to run only a few commands, but you have to be focused.

Do *not* insert an SD card into your card reader right now. You'll do it later to determine the device name of your reader. Download the latest ZIP file containing the Debian image from the official download page. Open a terminal, and change to the folder you've saved the ZIP file to. Then generate the file's fingerprint using the following command (this step is optional if you trust your download source or if you got the ZIP file from another trusted source):

```
maik> shasum 2012-07-15-wheezy-raspbian.zip
3947412babbf63f9f022f1b0b22ea6a308bb630c  2012-07-15-wheezy-raspbian.zip
```

If the hexadecimal number printed to the terminal is not the same as the number on the download page, the ZIP file might have been compromised, and you should download it from another location. Otherwise, you can safely proceed. Unzip the file to the current directory.

```
maik> unzip 2012-07-15-wheezy-raspbian.zip
Archive:  2012-07-15-wheezy-raspbian.zip
  inflating: 2012-07-15-wheezy-raspbian.img
```

Now you have to identify the name of your card reader. Run the following command to see all file systems that are currently mounted on your Mac:

```
maik> df -h
Filesystem      Size   Used  Avail Capacity  Mounted on
/dev/disk0s2   465Gi  383Gi   81Gi    83%    /
devfs          189Ki  189Ki    0Bi   100%    /dev
map -hosts       0Bi    0Bi    0Bi   100%    /net
map auto_home    0Bi    0Bi    0Bi   100%    /home
/dev/disk1s1   466Gi  460Gi  6.1Gi    99%    /Volumes/macback
```

The output on your system will vary, but you need it only to identify your SD card. Insert the card into your card reader now, and after a few seconds, run the command again.

```
maik> df -h
Filesystem      Size   Used  Avail Capacity  Mounted on
/dev/disk0s2   465Gi  383Gi   81Gi    83%    /
devfs          191Ki  191Ki    0Bi   100%    /dev
map -hosts       0Bi    0Bi    0Bi   100%    /net
map auto_home    0Bi    0Bi    0Bi   100%    /home
/dev/disk1s1   466Gi  460Gi  6.1Gi    99%    /Volumes/macback
/dev/disk3s1    15Gi  1.1Mi   15Gi     1%    /Volumes/SD
```

As you can see on my Mac, the SD card can be found at /dev/disk3s1. On your Mac, it might be at a different location, and it does not have to be mounted on /Volumes/SD. So, in the following command, you have to replace /dev/disk3s1 with the location of your SD card:

```
maik> diskutil unmount /dev/disk3s1
Volume SD on disk3s1 unmounted
```

This unmounts the SD card, and you can finally copy the Debian image to it. For this operation, you need the name of the SD card's raw device. You can derive it from the SD card's location by deleting *s1* and putting an *r* in front of *disk*. So, on my system, it's /dev/rdisk3.

WARNING: the following command will copy the Debian image to the device you specify with the of option. If you specify the wrong device, for example your Mac's main hard drive or an external USB drive containing your most

precious photos, all data will be lost. If you're absolutely sure that you've chosen the right target, run the following command:

```
maik> sudo dd if=2012-07-15-wheezy-raspbian.img of=/dev/rdisk3 bs=1m
1850+0 records in
1850+0 records out
1939865600 bytes transferred in 150.830724 secs (12861210 bytes/sec)
```

The command will run silently, and it will not emit any progress messages. As you can see in the previous output, it took more than two minutes to copy the image to the card, so be patient.

When you create the SD card, there's one thing to watch out for: some people have experienced read/write errors or unrecognized cards with SDHC cards on recent MacBooks and MacBook Pros with internal card readers. Using an external card reader solved these problems.

Finally, you can eject the card.

```
maik> diskutil eject /dev/rdisk3
Disk /dev/rdisk3 ejected
```

That's it! You've created a bootable SD card containing Debian on your Mac.

2.3   Next Steps

No matter what operating system you've used, you should now have a bootable SD card containing Debian Linux. You also know how to transfer the image of every operating system that is compatible with the Pi to a bootable SD card. In the next chapter, you'll learn how to start Debian on the Pi for the first time.

Configure Raspbian

No operating system or hardware will fit everybody's needs out of the box. This is especially true for the version of Debian that runs on the Pi, because it comes as an image, which means you cannot choose all the configuration parameters that you usually enter at installation time. For example, the image comes with a fixed keyboard layout and locale. In this chapter, you'll boot the Pi for the first time and take a look around. You'll learn how to configure a lot of basics such as your password and the time zone.

3.1 Boot the Pi for the First Time

Preparing the hardware and installing an operating system are important, but it's way more fun to actually boot the Raspberry Pi and see what it's capable of. So, insert the SD card you prepared in the previous chapter, and plug in the power supply.

If you've worked with Linux before, you'll recognize most of the messages pouring onto the screen. This comes as no surprise, because even if the Pi is an unusual computer, Raspbian still is an ordinary Linux distribution.

When you boot Raspbian for the first time, it starts a configuration program named Raspi-config. It helps you configure the most important aspects of the Linux system. You can see its main menu in Figure 4, *Raspi-config makes most configuration tasks a breeze,* on page 22.

You're probably used to controlling user interfaces with your mouse, but you have to control Raspi-config with your keyboard. Use the cursor-down key to move to the next menu item, and use the cursor-up key to move to the preceding one. To select a menu item, press the Tab key or the cursor-right key. This will highlight the Select button at the bottom. Press the spacebar or the Return key to select the menu item.
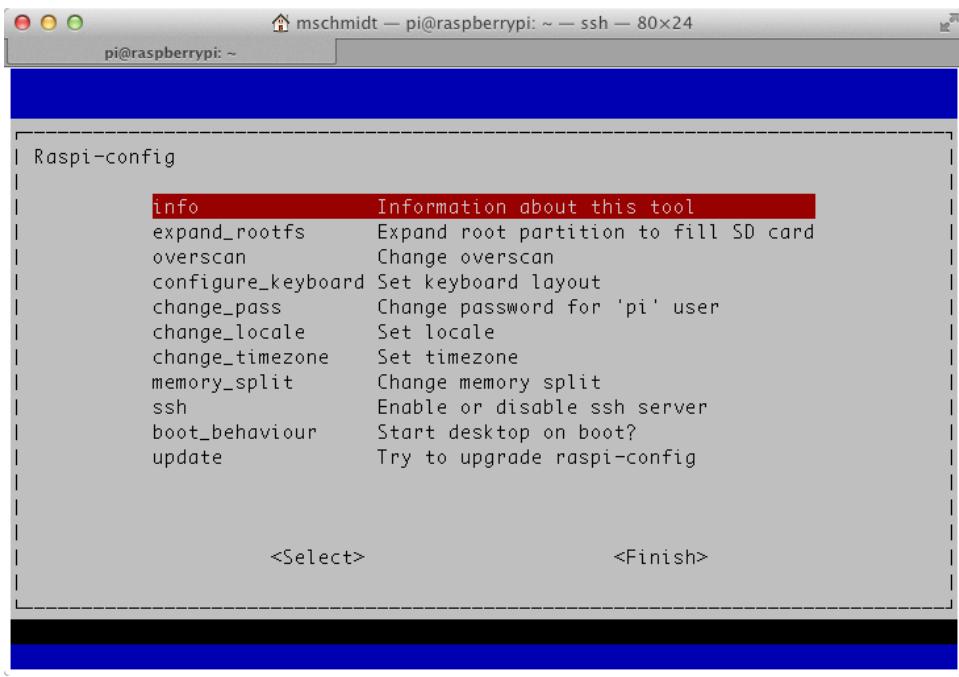
Figure 4—Raspi-config makes most configuration tasks a breeze.

To get familiar with Raspi-config, select the Info menu item first. This will open a new window briefly explaining what Raspi-config is for. Click the Ok button and press the spacebar to return to the main menu.

Most menus in Raspi-config also have a Cancel button. To cancel the current operation, press the Tab key until the Cancel button is highlighted, and then press the spacebar or the Return key.

The main menu has a Finish button that exits Raspi-config. Most changes you can perform with Raspi-config require a reboot of the Pi. So, when you press the Finish button in Raspi-config, it asks you whether you'd like to reboot.

Raspi-config will not start automatically the next time you boot the Pi. Don't worry. You can always invoke it in a terminal like this:

```
pi@raspberry:~$ sudo raspi-config
```

In the next section, you'll learn what most of the Raspi-config options are for.

3.2   Customize Your Installation with Raspi-config

Before you do anything else with the Pi, you should adjust the most important aspects of your Raspbian installation with Raspi-config. For example, you should increase the space that's available on your SD card, and you should set the right locale.

In this section, you'll get to know the most important menu items in Raspi-config. Of course, you'll learn about the rest later in the book also.

Use All the Space on Your SD Card

The Raspbian image limits your root file system to 2GB. In other words, no matter what the real capacity of your SD card is, you'll be limited to 2GB. You could copy the image to a 16GB SD card, for example, but you still can use only 2GB.

With the expand_rootfs menu in Raspi-config, you can easily change this situation. Select the menu item, and after the next reboot, the Pi will grab all the space it can get on your SD card. Depending on your SD card's capacity and speed, this will take a while.

Keep in mind that Raspi-config will not start automatically again. You have to log in with the username pi and the password raspberry. To start Raspi-config again, run the following command:

```
pi@raspberry:~$ sudo raspi-config
```

Configure the Pi's Overscan Mode

The Raspberry team wanted the Pi to work with as many displays as possible, so they had to take overscan and underscan into account. In the case of underscan, the video output does not use the whole display size, so you can see a black frame around the actual video output. In the case of overscan, the opposite happens, so in some cases you cannot see the whole output because it gets clipped at the display's borders. With the overscan menu in Raspi-config, you can enable or disable the overscan mode completely. In Section 4.3, *Configure the Video Output,* on page 38, you'll learn how to control video output in a more fine-grained manner.

Remap Your Keyboard and Change Your Locale

By default Debian assumes you're using an English keyboard layout, which might lead to some confusion if you're not. You can change the keyboard layout by choosing the configure_keyboard menu item in Raspi-config. This

will spawn a configuration program that first asks for your type of keyboard (see Figure 5, *Choose your keyboard type*, on page 24).
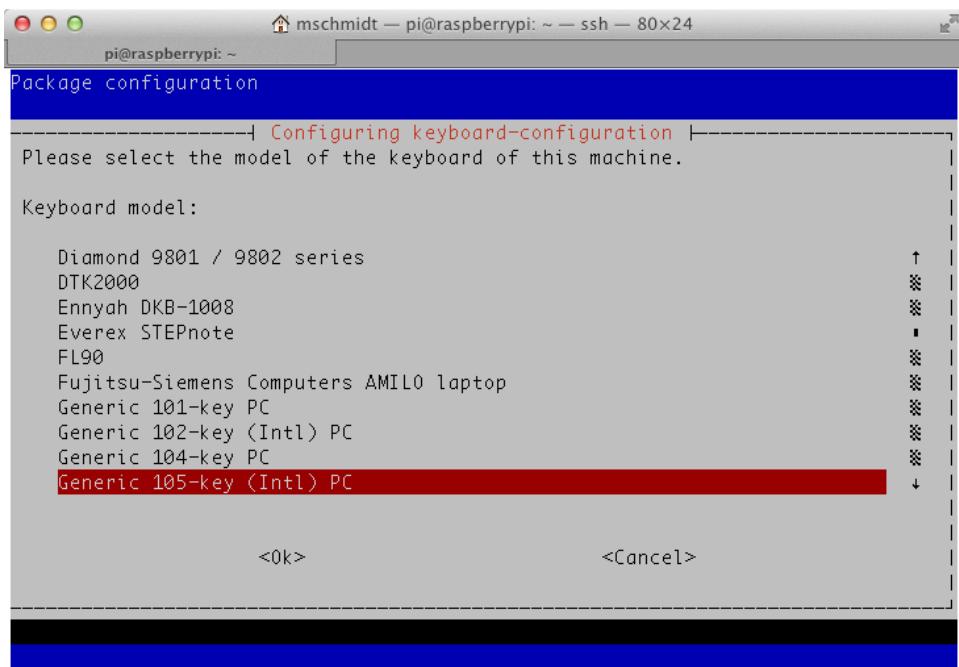
Figure 5—Choose your keyboard type.

Next you'll have to specify the language you're using, and after that, you'll have to configure the behavior of a few special keys.

To enable the new keyboard layout, you have to exit Raspi-config using the Finish button and reboot the Pi, but before that, you should consider changing the locale with the change_locale menu, too. A locale consists of more than a mere keyboard layout. It determines how data such as text or dates get sorted and formatted, for example. Also, it affects the language the system uses to display information such as menu texts in applications. In Figure 6, *A German version of LXDE*, on page 25, you can see a German version of the LXDE desktop, for example. You can configure your locale using Raspi-config's change_locale menu.

This starts a configuration program that greets you with the menu in Figure 7, *Generate your locale*, on page 25.

Here you can select which locales Raspbian should generate. You can select several and switch between them if necessary. Use the cursor keys to move
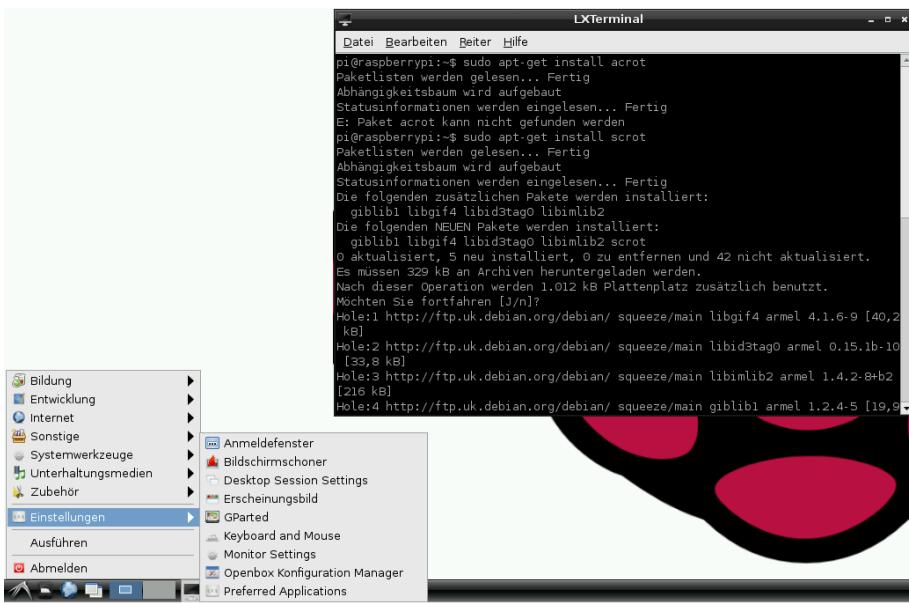
**Figure 6—A German version of LXDE**
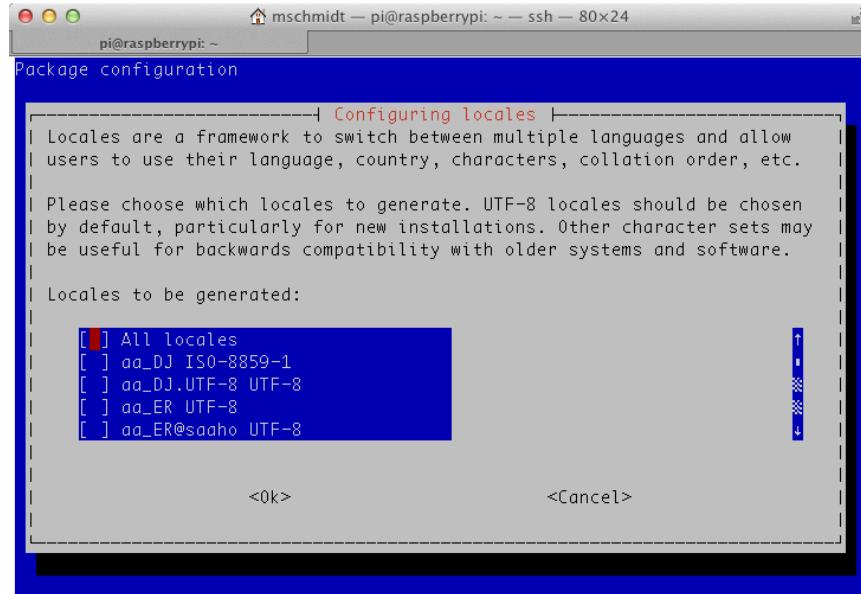
Figure 7—Generate your locale.

through the list, and use the spacebar to select or deselect a locale. With the Tab key, you can move the focus between the list of locales and the Ok and Cancel buttons. Hit the Return key to select a button.

After you select a list of locales and press the Ok button, you can choose your default locale. Hit the Ok button again, and you're done.

Set Your Time Zone, Time, and Date

To reduce costs, the Pi does not have a real-time clock, so it does not store the current date and time internally. Setting the correct date and time is not only a nice feature but is critical for cryptographic operations such as validating certificates. So, you need correct time information for many purposes. Raspbian contacts a time server on the Internet when it boots and sets the current time and date automatically.

So, internally the Pi knows the exact date and time in the UTC time zone, but it doesn't know your time zone. That's what the change_timezone menu item is for in Raspi-config. Select it, and it will ask a few questions to determine where you live exactly. Then it will store the time zone information in your profile, so the next time you boot your Pi, it will know what time zone you live in.

If you haven't connected your Pi to the Internet, you can manually set the date and time like this:

```
pi@raspberry:~$ sudo date --set="2012-07-31 13:24:42"
```

This solution has a few disadvantages. It's not as accurate as possible, and you have to repeat it whenever the Pi boots, so you can forget it easily.

Change Your Password

At the time of this writing, you have to enter the username pi and the password raspberry to log into the Pi. If you've been one of the lucky few who got one of the first boards, you also got a flyer with wrong credentials. In previous releases, the password was suse, so to be completely sure, check the credentials on the download page.[1]

Select the change_pass menu item in Raspi-config to change the password. Raspi-config asks you for a new password, and it also asks you to confirm the password. Note that for security reasons you cannot choose trivial passwords such as 123 or aaaa. If you want to learn more about users and passwords, take a look at Section A1.4, *Manage Users*, on page 106.

----

1. http://www.raspberrypi.org/downloads

By the way, raspberry is a really bad password, not only because you can guess it easily but mainly because it contains the character *y*. For all the people who do not have an English or American keyboard layout, this will certainly lead to some frustrating login sessions, because by default Debian uses a QWERTY keyboard layout. In Germany, people usually use a QWERTZ layout, for example, so if you're absolutely sure you've typed the password correctly for the tenth time, try raspberrz.

3.3 Start the Desktop

In contrast to other operating systems, a desktop environment is optional on Linux. So, it's not uncommon that you have to start it manually. Alternatively, you can start a desktop environment automatically whenever the Pi boots. Choose Raspi-config's boot_behaviour menu item to enable this behavior. If you rarely use the command line, this is a convenient option. Otherwise, the Pi will greet you with the login prompt:

After you've successfully logged in, you'll still see not much more than a boring shell prompt. Start the desktop to see some more colors using the

following command. (It reminds you of the good ol' MS-DOS times when you had to run win to start the real action, right?)

```
pi@raspberry:~$ startx
```

After a few seconds, the Raspberry Pi presents a nice desktop with a colorful raspberry in the background:

The desktop environment you've just started is named LXDE,[2] and although it does not use many resources, it still comes with some nice features. For example, it has virtual screens you can manage with the buttons in the toolbar at the bottom.

Starting applications is similar to starting them on Windows systems prior to Windows 8. Click the small LXDE logo on the left of the toolbar at the bottom of the screen to see which applications are available. Move the mouse to navigate through the pop-up menu, and start an application by clicking its name. In Figure 6, *A German version of LXDE,* on page 25, you can see the pop-up menu in action.

Also, you can configure a lot, such as the look and feel of all UI elements, the desktop resolution, and so on. You can change most of the settings using the system preferences menus; for example, in Figure 8, *You can change many preferences in LXDE,* on page 29, you can see some of them.

To leave LXDE, use the small power switch icon at the bottom right of the screen. If you've configured Raspi-config to always start the desktop, the Pi will shut down completely, when you log out from LXDE. Otherwise, it will return to the Pi's boot terminal. To shut down the Pi from there, run the following:

```
pi@raspberry:~$ sudo shutdown -h now
```

----

2.   http://lxde.org/
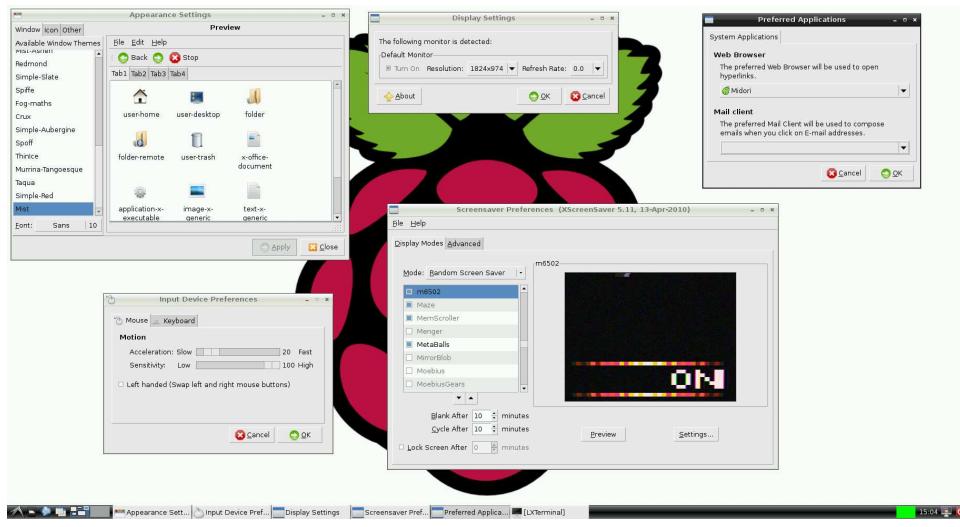
Figure 8—You can change many preferences in LXDE.

3.4 Manage Your Software with apt-get

Now that you have Debian set up, you probably want to add more software. Back in the old days, it was a pain in the neck to install new software on Linux systems. Usually you had to download a program's source code and compile and install it yourself. If the program depended on other projects or libraries, you learned about it when the compiler or the linker spat out some nasty error messages, and then you had to resolve the dependencies yourself —you had to download, compile, and install even more programs, and so on.

Fortunately, those days are long gone; all modern Linux distributions come with a package manager now that automates the whole process of downloading and installing new software. Not only do package managers resolve all dependencies automatically, they also save a lot of time by downloading binary packages instead of compiling them locally. Oh, and they also help you get rid of stuff you no longer need.

Debian comes with a package manager, too, and its name is *apt-get* (apt stands for Advanced Packaging Tool). In this section, you'll learn how to perform more operations such as adding, updating, and removing software.

Installing New Software

The Pi's Debian distribution comes with a minimalistic set of applications. This makes sense, because the Pi does not have a hard drive, but to get the

most out of it, you'll probably have to install a few programs. The good news is that installing software on the Pi does not differ from installing software on a regular PC running Debian. You'll get your software mostly from the same sources, and there are a lot of applications to choose from. Unfortunately, not all packages are available for the Pi's ARM architecture, and some applications simply do not run, because they need more resources than the Pi has to offer. Still, you can find plenty of useful programs.

In this section, you're going to install a PDF reader on your Pi. If you've worked exclusively with Microsoft Windows or Apple's Mac OS X before, you probably did not worry a lot about PDF viewers, but on some Linux systems, and especially on the little Pi, you cannot take a good PDF viewer for granted.

Interestingly, you can choose from a whole variety of different tools, and there's even a website dedicated to free PDF readers.[3] Two of these readers look especially interesting: xpdf[4] and evince.[5] You're going to install them both, try them, and uninstall the one you don't like.

You can install new packages with the install command. To install xpdf and evince, run the following command (just make sure you're connected to the Internet):

```
pi@raspberry:~$ sudo apt-get install xpdf
pi@raspberry:~$ sudo apt-get install evince
```

Alternatively, you can install more than one package at once like this:

```
pi@raspberry:~$ sudo apt-get install xpdf evince
```

Now you have both PDF readers installed as independent packages, and you can start and test them to see which one suits your needs better. You can find shortcuts for both applications in the Graphics section of the LXDE desktop's start menu. Also, you can start them from a terminal by running either the following (only if you've started the desktop environment before):

```
pi@raspberry:~$ evince
```

or the following:

```
pi@raspberry:~$ xpdf
```

In , you can see both programs in action rendering the same PDF document.

---

3. http://pdfreaders.org/
4. http://www.foolabs.com/xpdf/
5. http://projects.gnome.org/evince/
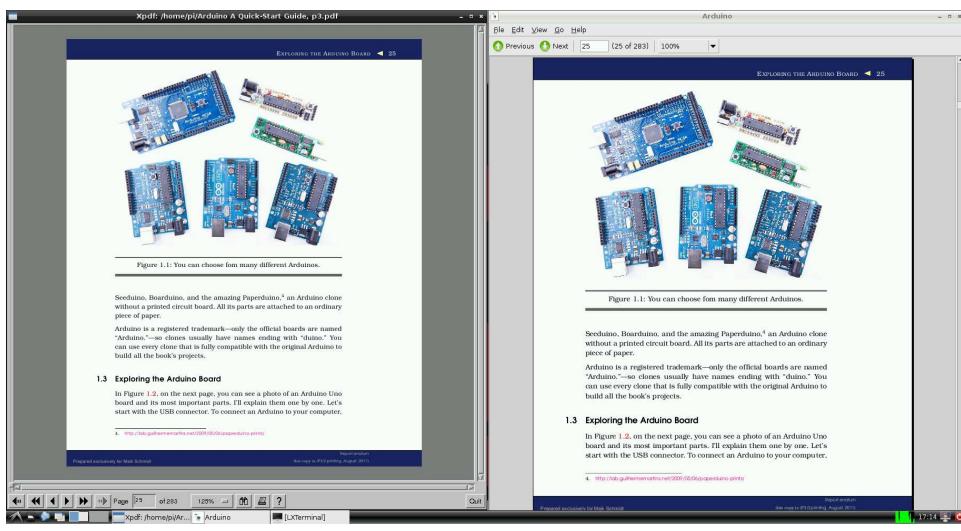
Figure 9—Two PDF readers showing the same document

Remove Software

Play around with both applications for a while and see which one you like best. Let's assume that you prefer evince; then you can uninstall xpdf with the following command:

```
pi@raspberry:~$ sudo apt-get remove xpdf
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
lesstif2 poppler-data ghostscript poppler-utils libxp6
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
xpdf
0 upgraded, 0 newly installed, 1 to remove and 42 not upgraded.
After this operation, 487 kB disk space will be freed.
Do you want to continue [Y/n]?
(Reading database ... 49208 files and directories currently installed.)
Removing xpdf ...
Processing triggers for menu ...
Processing triggers for man-db ...
```

The xpdf application is gone now without any traces.

Keeping Your Software Up-to-Date

To make the software installation process as easy and as unobtrusive as possible, apt-get comes with a small database containing a list of all available

packages and their dependencies. This database consists of only a few files. Usually you'll never work with it directly, but you should run the following command occasionally to update it:

```
pi@raspberry:~$ sudo apt-get update
```

This downloads the latest package lists from a central server and updates apt-get's local database. So, if you run this command before you install a new package, you can be sure that you will get the latest version available. Note that in some cases you have to run this command twice. apt-get is nice enough to tell you when this is the case.

If you already have installed software on your Pi using apt-get, you probably want to update it from time to time. The following command upgrades all the software that is currently installed on your Pi:

```
pi@raspberry:~$ sudo apt-get upgrade
```

Running this command will take a while, but when it's done, you have the latest version of every single application and library on your Pi. For this, apt-get has to download a lot of files that you no longer need after apt-get has installed the applications. You can delete these obsolete files easily.

```
pi@raspberry:~$ sudo apt-get autoclean
Reading package lists... Done
Building dependency tree...
Reading state information... Done
```

Sometimes dependencies between packages and package versions change, so it might happen that you no longer need some of the installed packages. You can remove them with the following command:

```
pi@raspberry:~$ sudo apt-get autoremove
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
ghostscript lesstif2 libxp6 poppler-data poppler-utils
0 upgraded, 0 newly installed, 5 to remove and 0 not upgraded.
After this operation, 15.3 MB disk space will be freed.
Do you want to continue [Y/n]?
(Reading database ... 49185 files and directories currently installed.)
Removing ghostscript ...
Removing lesstif2 ...
Removing libxp6 ...
Removing poppler-data ...
Removing poppler-utils ...
Processing triggers for man-db ...
```

In principle, this is all you need to know about managing software on a Debian system. There is one more helpful tool that you should know about, and you'll learn about it in the next section.

Find Packages with apt-file

If you know the exact name of a package you'd like to install, apt-get is all you need, but in some cases, you might not know the name. For example, you still have to install a lot of software from source and compile it yourself. If this software depends on a certain library that you do not have installed, the compiler or the linker will stop with an error message. Usually, the error message contains the name of the missing file, so it'd be great to have a tool that searches for all packages that contain this file. apt-file is such a tool, and you can install it as follows:

```
pi@raspberry:~$ sudo apt-get install apt-file
```

Like apt-get, the apt-file command depends on a local database containing a list of all packages and their dependencies. To update this database, you should run the following command:

```
pi@raspberry:~$ sudo apt-file update
```

Now you can use apt-file to search for a package containing a certain file. Let's assume you've heard about this cool PDF reader for the Pi named evince, and you do not know which package you have to install to use it. The command shown here is all you need:

```
pi@raspberry:~$ apt-file -l search evince
evince
evince-common
evince-dbg
evince-gtk
gir1.0-evince-2.30
libevince-dev
libevince2
python-evince
```

This outputs a list of all packages referring to evince so you can decide which one you'd like to install.

You can also use apt-file to list the content of a package, even if you have not installed the package.

```
pi@raspberry:~$ apt-file list evince
```

Package managers are really helpful, and every modern Linux distribution has one. On Debian it's apt-get, on Fedora it's yum, and on Arch Linux it's

pacman. Although they slightly differ in their syntax, they all offer the same operations and behavior.

3.5 Next Steps

In this chapter, you booted the Pi for the first time, and you configured many aspects to suit your personal preferences. Also, you know how to manage software on the Pi—how to install, update, and remove it.

Installing and configuring the Pi's operating system are important steps, but in contrast to regular PCs, the Pi needs some more configuration. In the next chapter, you'll learn about the Pi's firmware and how to adjust it to your needs.

Configure the Firmware

The Pi not only needs an operating system but also needs firmware that controls its hardware on a low level. For example, the firmware controls and configures the GPU, the card reader, and in some regards even the CPU. It's a vital piece of the Raspberry Pi, and you can solve many issues, such as problems with the video output, by setting the right parameters. In this chapter, you'll learn how to configure and update the Pi's firmware.

Also, you have to update the Linux kernel itself from time to time. The kernel is the heart of a Linux system, because it manages all processes and the hardware. All applications depend on the kernel, and in this chapter, you learn how to keep it up-to-date.

4.1 Update the Firmware/Kernel

The Debian image already comes with a firmware for the Pi, but the developers of the Linux kernel and the Pi's firmware release updates frequently. New releases usually contain bug fixes and improvements, so it's beneficial to update both the kernel and the firmware from time to time. To check which versions of the kernel and the firmware are installed on your Pi, run the following commands:

```
pi@raspberrypi ~ $ uname -a
Linux raspberrypi 3.1.9+ #168 PREEMPT Sat Jul 14 18:56:31 BST 2012
armv6l GNU/Linux
pi@raspberrypi ~ $ /opt/vc/bin/vcgencmd version
Jul 14 2012 13:14:40
Copyright (c) 2012 Broadcom
version 325444 (release)
```

You can always find the latest version of all files on GitHub,[1] and you only have to download them to the SD card. To install a new kernel and a new firmware, you have to replace a few files in the Pi's /boot directory. The /boot directory belongs to the SD card's boot partition, which is formatted with the FAT file system. So, you can read and write it not only with the Pi but with nearly every computer in the world. In the following screen capture, you can see its content:

The file start.elf contains the firmware, and the kernel is in kernel.img. All files beginning with *arm* also contain firmware images, but with different memory configurations (see Section 4.2, *Adjust the Memory Layout to Your Needs*, on page 37 to learn more about it).

So, you could download the new kernel and firmware files using your regular PC and copy them to the SD card using a card reader. This would still be tedious and error-prone. Fortunately, rpi-update[2] automates the whole process. After you've installed it on the Pi, it checks whether a new firmware version is available and downloads it if necessary. But before you can install rpi-update on the Pi, you have to install some packages that it needs.

```
pi@raspberry:~$ sudo apt-get install ca-certificates git-core
```

Now you can download rpi-update and make it executable.

```
pi@raspberry:~$ sudo wget http://goo.gl/1BOfJ -O /usr/bin/rpi-update
pi@raspberry:~$ sudo chmod +x /usr/bin/rpi-update
```

After that, run rpi-update.

---

1.   https://github.com/raspberrypi/firmware
2.   https://github.com/Hexxeh/rpi-update

```
pi@raspberry:~$ sudo rpi-update
aspberry Pi firmware updater by Hexxeh, enhanced by AndrewS
Performing self-update
Autodetecting memory split
Using ARM/GPU memory split of 192MB/64MB
We're running for the first time
Setting up firmware (this will take a few minutes)
Using HardFP libraries
If no errors appeared, your firmware was successfully setup
A reboot is needed to activate the new firmware
```

As you can see, the Pi has a new firmware, and you have to reboot it to activate it. Note that rpi-update even tries to determine and keep the current memory split. In the next section, you'll learn what a memory split is and how to configure it.

4.2   Adjust the Memory Layout to Your Needs

As you've learned in *What's on the Pi,* on page 1, the Pi has 256MB of RAM. Let's check whether this is true.

```
pi@raspberry:~$ free -m
             total       used       free     shared    buffers     cached
Mem:           186         37        149          0          5         19
-/+ buffers/cache:         12        174
Swap:          127          0        127
```

Oops! Apparently the Pi has much less than 256MB of RAM. How can that be? Don't worry: everything is OK with your hardware, and your Pi has 256MB of RAM. It just splits it between the CPU and the GPU (the device responsible for processing graphics). By default the CPU gets 192MB of RAM, while the GPU gets 64MB. For most use cases, this is reasonable, but for some cases, a different setup might make more sense. If you use the Pi as a server, for example, you won't need much graphics power but more RAM for the CPU.

You can change the memory configuration by using a different GPU firmware. Raspbian comes with a few preconfigured firmware files, and you can find them in the /boot directory.

```
pi@raspberry:~$ ls -l /boot/*.elf
-rwxr-xr-x 1 root root 2053456 Jul 19 06:47 /boot/arm128_start.elf
-rwxr-xr-x 1 root root 2053456 Jul 19 06:47 /boot/arm192_start.elf
-rwxr-xr-x 1 root root 2053456 Jul 19 06:47 /boot/arm224_start.elf
-rwxr-xr-x 1 root root 2053456 Jul 19 06:47 /boot/start.elf
```

The files starting with *arm* contain firmware images. The number after *arm* is the amount of main memory allocated for the CPU.

For example, arm128_start.elf allocates 128MB for the CPU and 128MB for the GPU. The file start.elf contains the current firmware. To activate a new one, you have to copy it to start.elf and reboot the Pi (alternatively, you can use the memory_split menu in Raspi-config to change the memory layout).

```
pi@raspberry:~$ sudo cp /boot/arm224_start.elf /boot/start.elf
pi@raspberry:~$ sudo reboot
```

Now check whether the Pi uses the new memory layout.

```
pi@raspberry:~$ free -m
             total       used       free     shared    buffers     cached
Mem:           218         36        182          0          4         19
-/+ buffers/cache:          12        205
Swap:          127          0        127
```

Excellent! The CPU now has plenty of RAM, so the current setup is perfect to run the Pi as a server.

4.3 Configure the Video Output

You can configure the firmware's behavior in many ways using the /boot/config.txt file. It contains all configuration parameters for the Pi's firmware; it's a good idea to bookmark their descriptions[3] in your web browser, because sooner or later you'll probably want to tweak a few things. Using the configuration file, you can adjust video and audio output, and you can even change the CPU's clock rate.

Most of the defaults work quite well on most systems, but the video output does not always work properly. The main problems are overscan and underscan, especially when using composite video output. In case of underscan, the video output does not use the whole display size, so you can see a black frame around the actual video output. In case of overscan, the opposite happens, so you cannot see the whole output because it gets clipped at the display's borders. In Figure 10, *Overscan problems can lead to a clipped output, on page 39*, the last line is not fully visible, for example. You can solve both problems by setting a few configuration options.

As you learned in Section 4.1, *Update the Firmware/Kernel, on page 35*, you can access all files in the /boot directory directly from your PC. If you prefer to edit /boot/config.txt on the Pi, open it with the nano text editor or a text editor of your choice.

```
pi@raspberry:~$ sudo nano /boot/config.txt
```

----

3.   http://elinux.org/RPi_config.txt
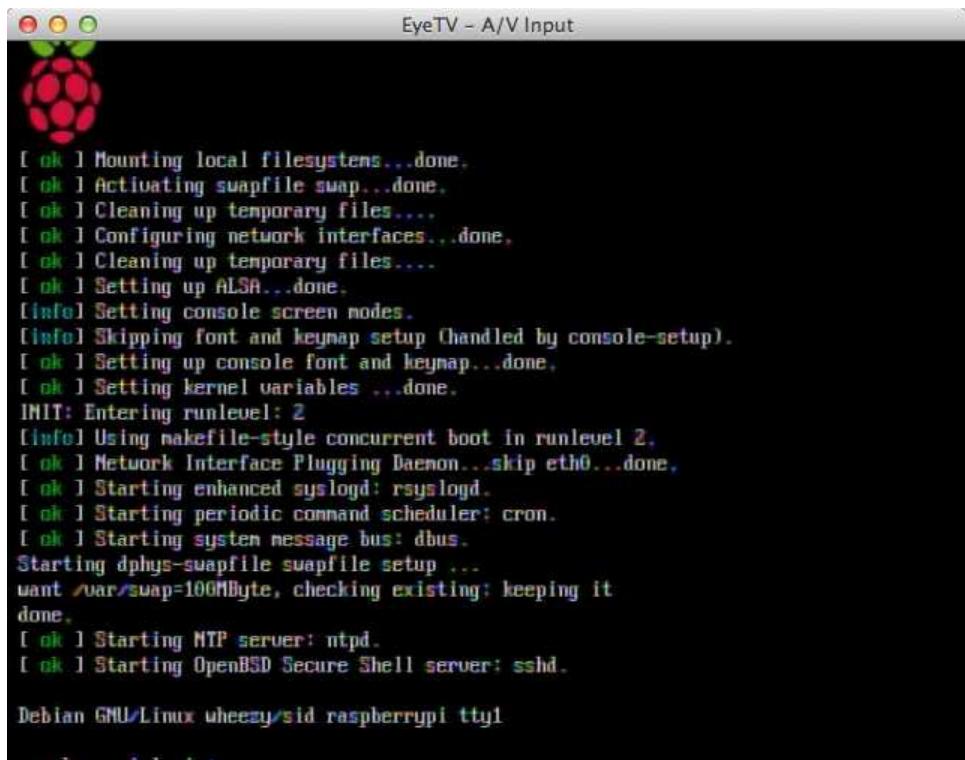
Figure 10—Overscan problems can lead to a clipped output.

To adjust the display's overscan, add these lines to the configuration file:

```
# Adjust overscan.
overscan_left=10
overscan_right=20
overscan_top=0
overscan_bottom=10
```

Configuration parameters have a simple format: they start with a name followed by an equal sign and a value. You can also add comments to the configuration file—they start with a # character. The previous example sets the overscan_bottom option to 10 pixels so that after a reboot, the Pi will skip 10 pixels on the bottom of the display. In Figure 11, *Overscan problems can be solved easily,* on page 40, you can see the effect.

You can use the same set of options to solve underscan problems, that is, to make the display area larger and remove the black frame. To achieve this, you have to set the option values to negative numbers.

Figure 11—Overscan problems can be solved easily.

```
overscan_left=-20
overscan_right=-10
```

You have to reboot the Pi every time you change /boot/config.txt, so getting the display settings right may take a while.

It's a good idea to have at least a quick look at the list of all configuration parameters so you remember them in case of any problems. You can play around with most options, but beware: setting some options, for example overclocking options, will void your Pi's warranty! If your Pi does not start any longer or if the display is no longer readable, edit /boot/config.txt on a separate PC and revert your last changes. If you're totally lost, delete /boot/config.txt, and the Pi will start with its defaults. Of course, you can also copy a fresh Raspbian image to the SD card.

4.4 Test and Configure the Audio System

Audio output is still a bit problematic on Linux systems, but Raspbian enables audio by default. For a first sound test, run the following commands:

```
pi@raspberry:~$ cd /opt/vc/src/hello_pi/libs/ilclient
pi@raspberry:~$ make
pi@raspberry:~$ cd ../../hello_audio
pi@raspberry:~$ make
pi@raspberry:~$ ./hello_audio.bin
```

This will compile a small test program that plays a siren sound. It outputs the sound via the analog audio jack, so plug in some headphones or speakers to hear it. Alternatively, you can play the sound via HDMI.

```
pi@raspberry:~$ ./hello_audio.bin 1
```

The Pi determines the best way for outputting audio automatically. It uses HDMI when available and analog output otherwise. You can change this behavior using amixer. This is a small tool that allows you to configure the sound hardware. Run it as follows to see which options you can change:

```
pi@raspberry:~$ amixer controls
numid=3,iface=MIXER,name='PCM Playback Route'
numid=2,iface=MIXER,name='PCM Playback Switch'
numid=1,iface=MIXER,name='PCM Playback Volume'
```

You can change only three options, and you must reference them by their *numid*. It'd be much nicer if the options had a real name, but the developers of amixer decided to use numerical IDs instead. The option for setting the playback route has a numid value of 3. Set it as follows:

```
pi@raspberry:~$ sudo amixer cset numid=3 1
```

This sets the playback route to 1 (analog output). You can also set it to 0 (automatic) or 2 (HDMI). When everything works as expected, you can add the amixer command to the /etc/rc.local file, so the Pi runs it automatically at start-up. Open a text editor such as nano, and add the following line to /etc/rc.local:

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

Add it to the end of the file, but do not make it the last line. Put it in front of the exit 0 statement.

By the way, some displays do not detect an HDMI cable correctly, so you might have video output via HDMI while audio output does not work. You can change this by setting the firmware configuration parameter hdmi_drive to 2 in /boot/config.txt. See Section 4.3, *Configure the Video Output,* on page 38 to learn how to do this.

4.5 Next Steps

In this chapter, you learned how to configure the Pi's firmware. You are able to solve problems with the display, and you know how to adjust system parameters such as the memory layout to your needs. In the next chapter, you'll take a short break and turn the Pi into a kiosk system.

Intermezzo: Build a Kiosk with the Pi

If you've been in a waiting room lately, chances are good that you've seen a kiosk system.[1] Usually, it consists of an old TV set with a DVD player and sometimes even a VCR still. In a doctor's waiting room, you'll see and hear a lot about new and expensive treatments, while repair shops often bombard you with lots of advertisement for useless parts. Often the video signal flickers a lot, the whole show contains many typos, and overall you feel that you should leave and look for someone more professional.

On the other side, you can find really good kiosk systems, for example while waiting for the metro. Here you can see the latest news, weather forecasts, and cartoons on huge screens. The biggest difference between the good and the bad kiosks is that the good ones usually do not repeat the same stuff for all eternity. They often update their content via the Internet.

The Raspberry Pi is a perfect platform for building a cheap but powerful kiosk system. In this chapter, you'll learn how to turn the Pi into a kiosk that displays Twitter live search information.

5.1 Display Twitter Live Search Information
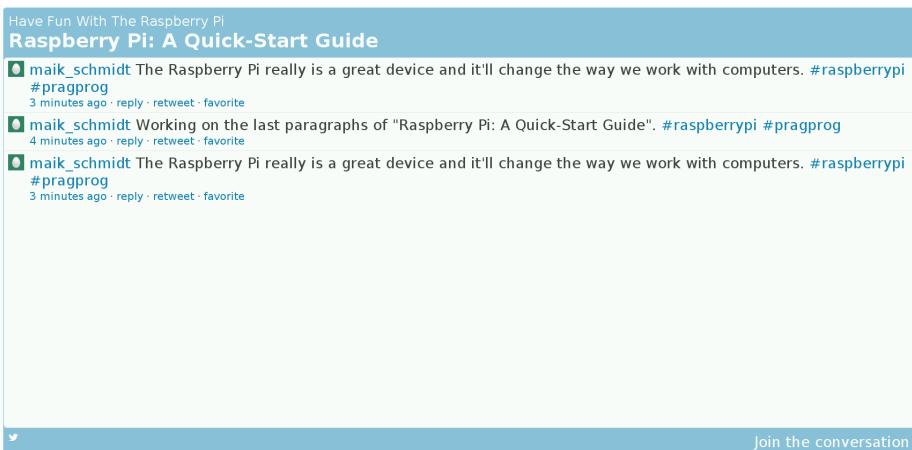
The only piece of software you need to build most kiosk applications is a web browser. Web browsers are very good at displaying multimedia content, so you have to make only the information you'd like to present available as an HTML page.

Also, you have to disable all the browser's menu bars and make sure the browser refreshes the content automatically at a certain interval. Most modern browsers already have a kiosk mode that does all this automatically.

---

1. http://en.wikipedia.org/wiki/Kiosk_software

In Chapter 6, *Networking with the Pi*, on page 49, you'll learn a bit more about Midori, the web browser that ships with Debian. All in all, the browser is very useful, and you can use it to build an impressive kiosk system.

The system you're going to build will display a live search of a list of terms on Twitter. For this book, for example, it's natural to search for the terms *maik_schmidt*, *pragprog*, and *raspberrypi*. In the following screen capture, you can see the final result.

It automatically updates its content every ten to thirty seconds, smoothly scrolling the last message out of the window while it scrolls in the next at the top. On a 46-inch screen, this system looks really impressive, and it would be a nice addition to your company's foyer.

Perhaps the screenshot reminds you of a bloated Twitter widget. This is no coincidence, because it actually is one. Usually, you'd embed such a Twitter search widget into your websites, but they are also the right technology for a kiosk system. The trick is to use the widget not as a widget but as your whole kiosk application. Therefore, you only have to make it really big and increase the size of your web browser's font.

One of the greatest things about widgets is that you usually do not have to create them yourself. For example, you can customize and download a Twitter search widget for free on Twitter's website.[2] You can choose the widget's title, width, height, some colors, and a search term. Then you can generate the JavaScript you need to embed the widget into your own site. The final result looks like this:

---

2.   https://twitter.com/about/resources/widgets/widget_search

kiosk/widget.html

```
Line 1  <html>
   -      <head>
   -        <style>
   -          .twtr-widget {
   5            text-align: center;
   -          }
   -          .twtr-doc {
   -            text-align: left;
   -            margin: auto;
   10           }
   -        </style>
   -
   -        <script charset="utf-8" src="http://widgets.twimg.com/j/2/widget.js">
   -        </script>
   15
   -        <script>
   -          new TWTR.Widget({
   -            version: 2,
   -            type: 'search',
   20           search: 'maik_schmidt pragprog raspberrypi',
   -            interval: 10000,
   -            title: 'Have Fun With The Raspberry Pi',
   -            subject: 'Raspberry Pi: A Quick-Start Guide',
   -            width: 1700,
   25           height: 800,
   -            theme: {
   -              shell: {
   -                background: '#8ec1da',
   -                color: '#ffffff'
   30             },
   -              tweets: {
   -                background: '#ffffff',
   -                color: '#444444',
   -                links: '#1985b5'
   35             }
   -            },
   -            features: {
   -              scrollbar: false,
   -              loop: true,
   40             live: true,
   -              behavior: 'default'
   -            }
   -          }).render().start();
   -        </script>
   45     </head>
   -
   -      <body>
   -      </body>
   -  </html>
```

The Twitter website generates the whole JavaScript code for you. You only have to add some boilerplate HTML. Also, lines 4 to 10 contain some style sheets to center the widget. If you need to reconfigure the widget later, you do not have to generate it again. The most important configuration parameters are in lines 20 to 25. Here you can alter the widget's title, its search terms, its dimensions, and its update interval. Note that you have to specify the update interval in milliseconds. Also note that Twitter does not guarantee the interval, so expect your widget to update less often than you like.

To get the widget code onto your Pi, you can open a text editor such as nano and type it in. A better approach is to download the ZIP archive containing the book's source code from the book's website[3] or click the file name above the preceding code example. By default Midori stores all downloads in the /tmp directory, so after Midori has downloaded the file, open a terminal and run the following commands:

```
pi@raspberrypi ~ $ cd /tmp
pi@raspberrypi ~ $ unzip msraspi-code.zip
```

Now you can find the widget code in /tmp/code/kiosk/widget.html. Before you run it in Midori, open it with a text editor such as nano, and adjust the widget's search term, its width, and its height to your needs.

```
pi@raspberrypi ~ $ nano /tmp/code/kiosk/widget.html
```

When you're done with your changes, press Ctrl+X in nano, confirm that you'd like to save your changes, and then confirm the filename.

To run the kiosk widget, start the LXDE desktop on your Pi, and then start the Midori browser. After that, choose the Open menu in Midori, and select the file containing the widget code. The widget will start right away, but to see the full effect, press F11 to enable Midori's full-screen mode.

The widget now covers the whole screen and updates automatically. Press Ctrl+plus a few times to increase the font size. Without writing a single line of code, you've turned the Pi into a kiosk system.

5.2   Refresh Websites Automatically

The Twitter widget you used in the preceding section has a nice feature: it updates its content automatically using JavaScript. Unfortunately, not all web pages are so nice, so you have to update some of them yourself. One solution is to add a <meta> tag to the head section of the HTML page.

---

3.   http://media.pragprog.com/titles/msraspi/code/msraspi-code.zip

```
<meta http-equiv="refresh" content="120"></meta>
```

The previous element reloads the page every 120 seconds. It's an easy solution, but it works only when you are allowed to change the page and have access to the server hosting it. So, the best solution would be if you could tell your browser to reload the current page at a certain interval. Most browsers have such a function, and usually it's called *kiosk mode*. Midori has a kiosk mode, too. To use it, you have to start Midori from the command line. With the -a option, you specify the address Midori should display. To set the update interval, use the -i option, and to turn on full-screen mode, use -e. The final command looks like this:

```
pi@raspberrypi ~ $ midori -i 30 -e Fullscreen \
  -a "https://mobile.twitter.com/search?q=pragprog"
```

This shows the results of a mobile Twitter search for the term *pragprog* and refreshes the page every thirty seconds. Increase the font size by hitting Ctrl+plus a few times, and you're done. Note that Midori supports many more options for controlling it from the outside. You can list them with the following command:

```
pi@raspberrypi ~ $ midori --help-execute
```

5.3   Next Steps

In this chapter, you learned how to turn the Pi into a kiosk system in only a few steps. Look around in your company for possible applications. For example, you can permanently display the status of your most critical systems on a big screen. Also, you could display the current number of customers/orders somewhere. Of course, you can be conservative and simply display a set of slides explaining how wonderful and great your company is.

Networking with the Pi

Like every computer, the Pi gets even more exciting the moment you connect it to a network. Suddenly you can use the Pi for everyday tasks such as surfing the Web or tweeting messages. You also can make the Pi accessible via Secure Shell, so you can securely and conveniently work on it from another computer. It's even possible to share the Pi's graphical desktop with your PC over a network, and vice versa.

On top of all that, you can use the Pi as a cheap but powerful web server that not only serves static content but is also able to run web applications written in languages like PHP, for example.

6.1 Perform Everyday Tasks on the Web

You're probably used to performing a lot of tasks using only your web browser: checking email, reading RSS feeds, watching videos, sending tweets, and so on. This is all possible because most browsers today support HTML5, JavaScript, Flash, and Java. Without these technologies, the Web would still look very much like 1995.

All these nice things work only with modern browsers such as Google Chrome or Mozilla Firefox. Although you can already install Chromium on Raspbian,[1] it will take a while until it will run sufficiently fast on the Pi. Also, it does not support all features like video at the moment. The biggest limitation is because of the Pi's small memory, because modern browsers use a lot of it. The browser that ships with the Debian distribution for the Pi is Midori.[2] It is a pretty good web browser that does not use a lot of memory. Unfortunately, it's still in its infancy and has some limitations. For one thing, technologies

---

1.   http://hexxeh.net/?p=328117859
2.   http://twotoasts.de/index.php/midori/

such as Flash and Java are not available on the Pi currently. So, if you come across a website that uses Flash or a Java applet, it will not work in any browser running on the Pi.

> ### But I Want Chrome
>
> At the time of this writing, the Chromium port is new, and it's far from being stable. Also, it's rather slow, but if you still want to install Chromium on the Pi, run the following command:
>
> ```
> pi@raspberrypi ~ $ sudo bash <(curl -sL http://goo.gl/5vuJI)
> ```
>
> Then you can start Chromium from a terminal with the following command:
>
> ```
> pi@raspberrypi ~ $ chrome --disable-ipv6
> ```
>
> To make it run as fast as possible, use the 224MB memory split (see Section 4.2, *Adjust the Memory Layout to Your Needs*, on page 37). If you're brave enough, you can also overclock your Pi.[a]
>
> _____
>
> a. http://www.rpiforum.net/forum/tutorials/article/8-how-to-overclock-your-raspberry-pi/

Although Midori understands HTML5, CSS3, and JavaScript, it's not capable of interpreting all modern websites properly. For example, Google Mail does not run out of the box, because Midori does not understand some of the site's JavaScript. Also, Midori needs some time to render the default view of Google Mail. You can improve this situation by disabling JavaScript using Midori's Preferences > Behaviour menu and by choosing Google Mail's basic HTML view. Even then, you still cannot work properly with all input fields, such as the fields to enter an email's recipients. It's best to select addresses only from the address book or to copy them from another text field and then paste them into the To and CC fields.

Disabling JavaScript can improve the usability of other websites, too. When you disable JavaScript in your browser, many websites will return a plain HTML version of their service. This HTML version usually does not have all the bells and whistles of the original site, but at least you can use it.

For Twitter, it's similar. Midori can render it, but it's rather slow. A good solution is to use Twitter's mobile website.[3] It will not have all the features of the original site, but it still provides a very good Twitter experience, and it works on Midori.

_____

3. http://mobile.twitter.com

Another trick that sometimes helps is to change Midori's user agent. All web browsers send a unique identifier with every request that tells the web server exactly what kind of browser has sent the request. This identifier is named *user agent*, and some websites change their response depending on its value. For example, some websites generate an error message if they do not know the user agent. In Midori, you can change the user agent in the Preferences > Network menu. Here you can tell Midori it should pretend to be a Mozilla Firefox browser or Safari or an iPhone.

Some sites will not work at the moment no matter what you try. YouTube,[4] for example, depends on video support in the browser, be it HTML5 video or Flash. Midori does not support either of them at the moment, so you cannot render YouTube properly. You're out of luck, too, if a website depends on some JavaScript code that does not work on Midori or if it needs a Java applet. Fortunately, Java applets are not very popular today, but some online banking sites still use them.

In addition to all this, Midori sometimes uses a lot of resources, especially the CPU. It happens often that Midori needs nearly the whole CPU to render a website—sometimes you have to wait for a few minutes for just one page to render.

If you keep in mind that you can disable JavaScript and that most popular websites have a mobile version, Midori is sufficient for most situations. Also, the Pi's popularity might result in major improvements in the near future.

6.2   Use Secure Shell with the Pi

It's very likely that you will connect your Pi to a network so you can access the Pi from other computers, and vice versa. One of the best ways to communicate securely between two computers is *Secure Shell* (SSH), a network protocol for secure data communication. Debian comes with everything you need to use SSH; you have to configure only a few things.

Accessing the Pi Using a Password

If you want to access only other computers from the Pi, you don't have to configure anything. For example, you can connect as the admin user to the host maik-schmidt.de by starting SSH on the Pi with the other computer's name and password.

---

4.   http://youtube.com

```
pi@raspberrypi ~ $ ssh admin@maik-schmidt.de
admin@maik-schmidt.de's password:
Last login: Wed May  2 09:41:34 2012 from 94.221.82.250
admin@maik-schmidt.de:~$ exit
logout
Connection to maik-schmidt.de closed.
```

However, if you want to access the Pi using SSH, you first have to enable the SSH server on the Pi using Raspi-config.

```
pi@raspberrypi ~ $ sudo raspi-config
```

Choose the ssh menu item, and enable the SSH server. Then click the Finish button to leave Raspi-config. After you reboot, the Pi's boot log contains a new message.

```
Starting OpenBSD Secure Shell server: sshd
My IP address is 192.168.2.109
```

This means you can access the Pi via SSH now and that the Pi's IP address is 192.168.2.109. In your case, the IP address will very likely be a different one. If you need to determine your Pi's IP address later, you can run this:

```
pi@raspberrypi ~ $ ip addr | grep 'inet .* eth0'
inet 192.168.2.109/24 brd 192.168.2.255 scope global eth0
```

The first IP address that appears in the output is your Pi's address. Using this address, you can access the Pi from all the other computers on your network.

From a Mac or a Linux system, you can start SSH from the command line, passing it the IP address and password of the pi user.

```
maik> ssh pi@192.168.2.109
pi@192.168.2.109's password:
Linux raspberrypi 3.1.9+ #171 PREEMPT Tue Jul 17 01:08:22 BST 2012 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Type 'startx' to launch a graphical session

Last login: Fri Jul 20 08:06:17 2012
pi@raspberrypi ~ $ exit
logout
Connection to 192.168.2.109 closed.
```
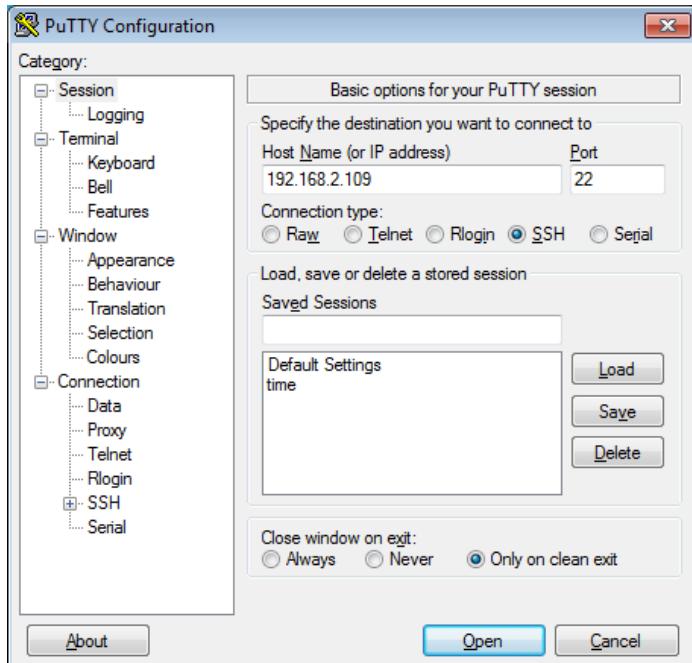
To access the Pi from a Windows box, you need an SSH client, and one of the best is PuTTY.[5] It's a very small program that you do not even have to install. Download the executable, start it, and you'll see the configuration screen shown here:

PuTTY allows you to configure a lot of things, and it also lets you store your configuration for each type of session you need. To log into the Pi, you only have to enter its IP address and click the Open button. Then you'll see the Pi's regular login prompt, as in .

Accessing the Pi with a Public-Private Key Pair

If you have to access your Pi via SSH often, it might get tedious to always enter the password when you log in. A more convenient way is to use the public-private key mechanism. To do that, you have to generate a key on your PC. The key has two parts, one private and one public, and you copy the public part to the Pi. The next time you log in to the Pi from your PC, SSH is able to verify your identity by checking whether the public and private parts of the key match. If you want to access the Pi from several computers, you have to run the following steps for each of them.

---

5.   http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
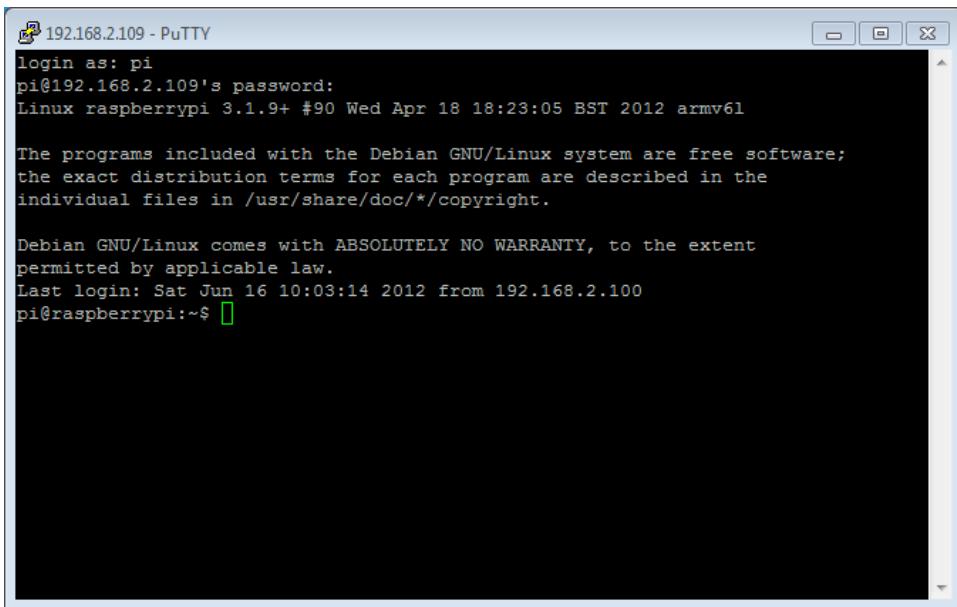
Figure 12—Accessing the Pi from Windows is easy.

Before you generate a new key pair, you should check to see whether you have one already. On Linux or Mac OS X, open a terminal and run the following command:

```
maik> ls ~/.ssh/id_rsa.pub
/Users/maik/.ssh/id_rsa.pub
```

The file id_rsa.pub contains the public key, and the command shown earlier tries to list it. If the output looks like the previous output, you already have a key, and you can skip the key generation and copy the public key to the Pi, as described next. If you get a "No such file or directory" message instead, you have to generate a key, as in the following:

```
maik> ssh-keygen -t rsa -C "your_email@youremail.com"
Generating public-private rsa key pair.
Enter file in which to save the key (/Users/maik/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
/Users/mschmidt/.ssh/id_rsa.
Your public key has been saved in
/Users/mschmidt/.ssh/id_rsa.pub.
The key fingerprint is:
f0:09:09:49:42:46:42:6f:42:3b:42:44:42:09:6a:e8
your_email@youremail.com
```

```
The key's randomart image is:
+--[ RSA 2048]----+
| . .o..          |
|+  ..o + .       |
|o.o + B o        |
|.+ o o B .       |
|  E   = S .      |
| .   . o         |
|       .         |
|                 |
|                 |
+-----------------+
```

This generated a key pair in your home directory. You can find the public key in a file named id_rsa.pub. You now have to transfer this file to the Pi where SSH keeps a list of all authorized keys in a file named .ssh/authorized_keys in the pi user's home directory. The following commands append the id_rsa.pub file to the Pi's list of authorized keys:

```
maik> scp ~/.ssh/id_rsa.pub pi@192.168.2.109:/tmp
maik> ssh pi@192.168.2.109 "cat /tmp/id_rsa.pub >> ~/.ssh/authorized_keys"
```

The first command copies id_rsa.pub to the Pi's /tmp directory, and the second command appends the file's content to the ~/.ssh/authorized_keys file. If you do not plan to keep several keys in the authorized_keys file, you can copy id_rsa.pub directly, of course.

```
maik> scp ~/.ssh/id_rsa.pub pi@192.168.2.109:/home/pi/.ssh/authorized_keys
```

On a Windows box, you can use some additional tools from the PuTTY download page to generate the keys and to copy them to the Pi. In , you can see the PuTTYgen application that generates keys.

To copy the generated public key file, use PSCP. It works exactly like scp, so from a DOS prompt, run the following command:

```
C:\> pscp id_rsa.pub pi@192.168.2.109:/home/pi/.ssh/authorized_keys
```

Your Pi is a full-fledged member of your network now.

6.3   Share Desktops with the Pi

Logging into the Pi using SSH is convenient and opens a new world of possibilities. For example, you can access the Pi's file system, start and stop processes, and monitor what's happening on the Pi at the moment. The biggest disadvantage of the SSH solution so far is that it works only in a text terminal.
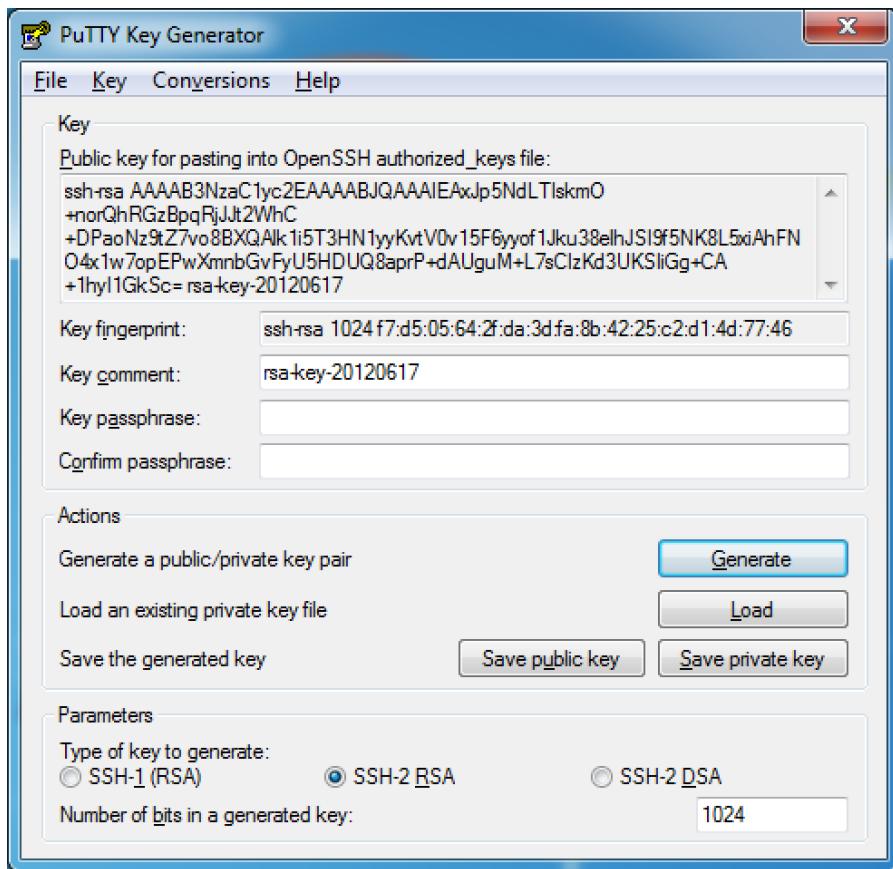
Figure 13—PuTTYgen generates keys on Windows.

You can easily overcome this limitation and control the Pi's desktop, keyboard, and mouse using another computer. The solution is Virtual Network Computing (VNC),[6] a technology that transmits the whole screen and all mouse and keyboard events from one computer to another.

To enable VNC, you need a VNC client and server. The server runs on the machine you'd like to control, and the client runs on the controlling machine. So, if you want to control the Pi using your PC, you have to install a VNC server on your Pi. You can choose from several, but one of the best is TightVNC.[7] It is available for free for all major platforms, and you can install it via apt-get.

---

6. http://en.wikipedia.org/wiki/Vnc
7. http://www.tightvnc.com/

```
pi@raspberrypi ~ $ sudo apt-get install tightvncserver
pi@raspberrypi ~ $ tightvncserver

You will require a password to access your desktops.

Password:
Verify:
Would you like to enter a view-only password (y/n)? n

New 'X' desktop is raspberrypi:1

Creating default startup script /home/pi/.vnc/xstartup
Starting applications specified in /home/pi/.vnc/xstartup
Log file is /home/pi/.vnc/raspberrypi:1.log
```

When you run tightvncserver for the first time, it asks you to set a password. You have to enter this password in the VNC client later to prevent unauthorized people from accessing your Pi. In addition, TightVNC allows you to optionally define a view-only password. This password gives clients read-only access so they can see the screen, but they cannot control the keyboard and mouse. This is useful for presentations, for example.

After you've defined the passwords, TightVNC creates a new virtual screen that you can access from your PC or from your Mac. The great thing about VNC is that it allows you to create as many virtual screens as you need. These screens do not necessarily have to correspond to physical screens. They are purely virtual, so many users can access your Pi, for example, and they all get their independent desktop environment.

To address a virtual screen, you need two things: the IP address of the Pi and the port address of the screen. VNC's base port is 5900, so to access screen number 1, you have to use port 5901. To access the screen you've created with the previous commands, you have to use the network address 192.168.2.109:5901. Keep in mind that your IP address will probably be different.

Now that you know the address of the Pi's VNC server, you can access it from a PC or a Mac with a VNC client. On a Mac it's very easy, because it comes with a VNC client already. You can actually connect to a VNC server using the Safari web browser. Simply enter the web address vnc://192.168.2.109:5901, and Safari will spawn the Screen Sharing application. Enter the password you defined earlier, and you're done. See the result in .

Figure 14—Controlling the Pi from a Mac

On Windows and Linux the procedure is very similar, but you have to install a VNC client first, which is easy, because TightVNC runs on Windows and Linux, and it contains a client, too.

Controlling the desktop of your PC or Mac from the Pi is easy, too. First you have to install a VNC server on your PC, and again TightVNC is a great choice for Windows and Linux. On the Mac it's even easier, because Mac OS X has an integrated VNC server that you only have to enable. In the System Preferences, select Sharing and then enable Screen Sharing. Click the Computer Settings button to set a password (you can see the preferences panel in).

Now you need a VNC client on the Pi, and xtightvncviewer is a good one. Install it using apt-get.

```
pi@raspberrypi ~ $ sudo apt-get install xtightvncviewer
```

Then open a terminal on the Pi's desktop, and start the client, passing it your PC's IP address and VNC port.

```
pi@raspberrypi ~ $ xtightvncviewer 192.168.2.100:5900
```
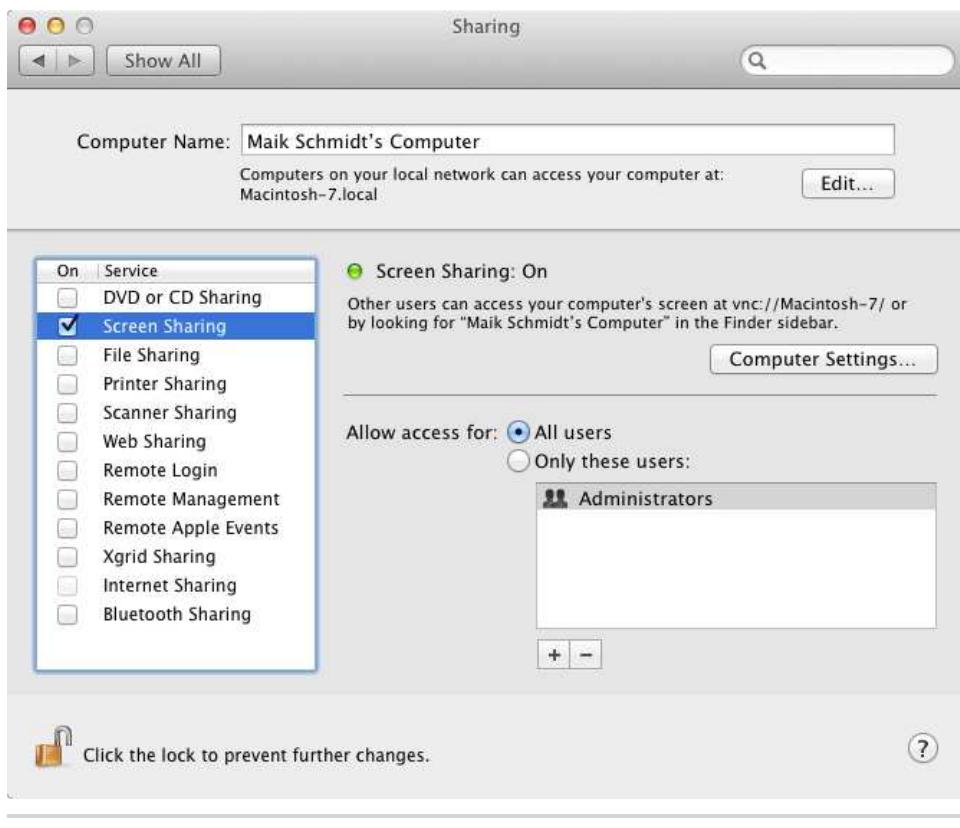
Figure 15—Sharing the Mac's screen is easy.

In Figure 16, *Controlling a Mac from the Pi,* on page 60, you can see a Mac's desktop inside a window on the Pi's desktop. If it doesn't work, make sure you use the right IP address and the right port number. Usually, it's 5900, but it might vary in different VNC servers.

6.4 Turn the Pi into a Web Server

Although the Pi looks like a toy compared to modern web server hardware, it's still powerful enough to serve interesting information in your local network. Not only can it serve static websites, but it can also generate dynamic content using databases and web applications. In addition, it can even provide access to its GPIO ports via web technologies.

The first thing you need to turn the Pi into a web server is an HTTP server, a network service that understands the Hypertext Transfer Protocol (HTTP).

Figure 16—Controlling a Mac from the Pi

You can choose from several great products such as the Apache HTTP server[8] or Nginx,[9] but Lighttpd[10] is a good choice for the Pi because of its very low memory footprint.

Installing/running Lighttpd is a piece of cake.

```
pi@raspberrypi ~ $ sudo apt-get install lighttpd
```

After the installation has completed, Lighttpd is up and running, and you can point your PC's web browser to your Pi's IP address. For example, in Figure 17, *Lighttpd's welcome page, on page 61*, you can see the server's welcome page.

To create your own web pages, you have to add them to Lighttpd's document root, a directory containing all files it should serve for a website. Lighttpd's document root by default is the /var/www directory. You should make sure that only members of the operating system group www-data have permission to write to it. The following commands add the pi user to the www-data group and set the permission flags of the /var/www directory accordingly:

---

8. http://httpd.apache.org/
9. http://nginx.org/
10. http://www.lighttpd.net/
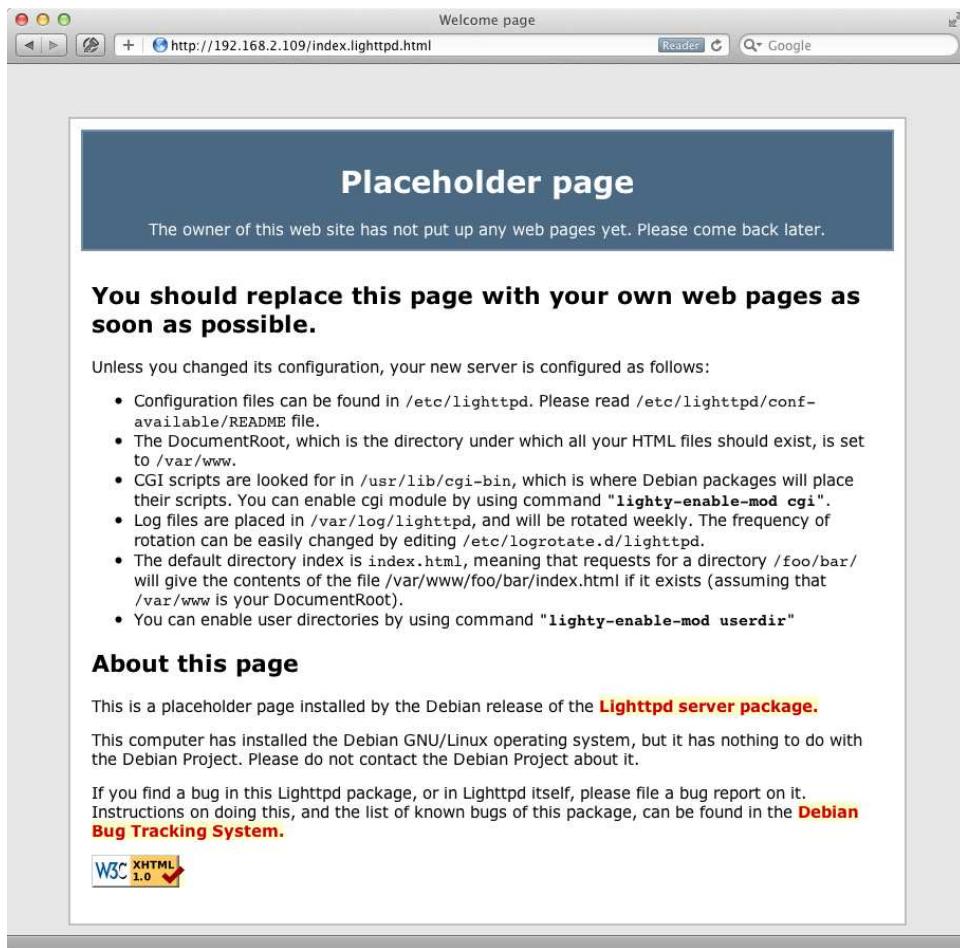
Figure 17—Lighttpd's welcome page

```
pi@raspberrypi ~ $ sudo adduser pi www-data
pi@raspberrypi ~ $ sudo chown -R www-data:www-data /var/www
pi@raspberrypi ~ $ sudo chmod -R 775 /var/www
```

After the next login, the pi user is allowed to create new web pages. You can do it with any text editor, such as nano. The following command creates a new file named index.html that will be the start page of your first website:
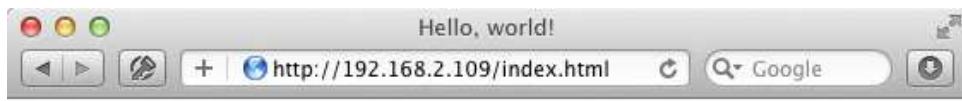
```
pi@raspberrypi ~ $ nano /var/www/index.html
```

Enter the following text:

Networking/index.html

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

When you're done, press Ctrl+X to leave nano. Press Y to confirm that you want to save the file, and then press Enter to confirm the filename. After that, point your browser to the new web page and see the result likke this:

By the way, if you prefer, you can edit the index.html file on your PC and copy it to the Pi afterward (remember to replace the IP address with your Pi's IP address).

```
maik> scp index.html pi@192.168.2.109:/var/www
```

With only a few commands, you've turned the Pi into a full-blown web server that is able to serve static content such as HTML pages. This is nice and useful, but sometimes you need more dynamic content. For example, you might want to embed data from a database into your pages, or you might even embed environmental data you collect with sensors attached to the Pi.

To create dynamic content, you need a programming language. You can choose from a lot of alternatives. For the Pi, PHP[11] is a good choice, because it does not use a lot of resources, and you can install it easily.

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get install php5-cgi
pi@raspberrypi ~ $ sudo lighty-enable-mod fastcgi
pi@raspberrypi ~ $ sudo /etc/init.d/lighttpd force-reload
```

---

11. http://www.php.net/

These commands install a PHP interpreter and enable the FastCGI module in the Lighttpd server. FastCGI[12] speeds up dynamic websites tremendously, so it's a good idea to enable it. To finalize the installation, you have to edit Lighttpd's configuration file.

```
pi@raspberrypi ~ $ sudo nano /etc/lighttpd/lighttpd.conf
```

Add the following lines at the end of the file to enable PHP and FastCGI:

```
fastcgi.server = (".php" => ((
  "bin-path" => "/usr/bin/php-cgi",
  "socket"   => "/tmp/php.socket"
)))
```

When you have saved the changes to the configuration file, restart the web server.

```
pi@raspberrypi ~ $ sudo service lighttpd restart
```

To test whether everything works as expected, create a file named /var/www/index.php with the following content:

Networking/index.php
```
<?php
  phpinfo();
?>
```

Point your web browser to the newly created file, and you should see something similar to Figure 18, *Serving dynamic web pages from the Pi,* on page 64.

This is PHP's info page that contains a lot of information about the system PHP is currently running on. PHP generates it dynamically. You can see that everything works fine, and now you can start to build your own web applications on the Pi. In Chapter 9, *Tinker with the GPIO Pins,* on page 87, you'll build a web application that controls some external hardware attached to the Pi.

6.5   Add WiFi to the Pi

Wireless networks are everywhere. Coffee shops, airports, and hotels all offer free WiFi to their customers these days. You probably run a wireless network at home, too, so you can conveniently access your most important devices from your smartphone while you're having a barbeque with the family in the garden. On Windows or Mac OS X you usually do not have to think much about joining wireless networks because the process is nearly seamless.

––––––––––––––

12. http://www.fastcgi.com/

Figure 18—Serving dynamic web pages from the Pi

On the Pi it's different. In this section you'll learn how to connect the Pi to a wireless network.

Joining wired networks via Ethernet is usually a piece of cake: plug in the Ethernet cable and you're done. To access a wireless network you need to do a bit more. First, you need a WiFi stick for the USB port, because the Pi's hardware does not support WiFi by default. Plug your WiFi stick into one of the Pi's USB ports and run the lsusb command to see if the Pi recognizes it properly:

```
pi@raspberrypi ~ $ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 050d:0237 Belkin Components F5U237 USB 2.0 7-Port Hub
Bus 001 Device 005: ID 04e8:2018 Samsung Electronics Co., Ltd WIS09ABGN
                                  LinkStick Wireless LAN Adapter
Bus 001 Device 006: ID 046d:c312 Logitech, Inc. DeLuxe 250 Keyboard
Bus 001 Device 007: ID 046d:c05a Logitech, Inc. Optical Mouse M90
```

In this case, device 005 is a WiFi stick manufactured by Samsung. You can take a closer look at the Pi's boot message with the dmesg command and see whether the WLAN stick has been initialized properly:

```
pi@raspberrypi ~ $ dmesg | less
...
usb 1-1.3.6: new high speed USB device number 5 using dwc_otg
usb 1-1.3.6: New USB device found, idVendor=04e8, idProduct=2018
usb 1-1.3.6: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-1.3.6: Product: 802.11 n WLAN
usb 1-1.3.6: Manufacturer: Ralink
usb 1-1.3.6: SerialNumber: 1.0
...
```

Press the space bar to go down one page and press 'b' to go up one page. Press 'q' to go back to the shell prompt. As you can see in the current case the Samsung stick uses the WiFi chipset from a company (manufacturer) named Ralink. This chipset is pretty popular, so Debian recognized it out of the box. If the output of dmesg contains any errors right after the initialization of your WiFi stick, check the Pi's Wiki.[13] Often you have to download the firmware for your WiFi stick manually and reconfigure the Linux kernel.

If no errors occured, Debian Linux has recognized your WiFi stick successfully. You can use the following command to get the current status of your Pi's wireless network interfaces:

```
pi@raspberrypi ~ $ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11abgn  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

---

13. http://elinux.org/RPi_VerifiedPeripherals#USB_WiFi_Adapters

At the moment the Pi is not connected to a wireless network, but the wlan0 interface is up and running. The following command searches for a wireless network:

```
pi@raspberrypi ~ $ sudo iwlist scan | grep ESSID
        ESSID:"darknet"
        ESSID:"valhalla"
```

In this case, two wireless networks named darknet and valhalla are within reach. To connect to one of them you have to edit the configuration file /etc/network/interfaces using a text editor, such as the nano text editor, for example. To connect to darknet add the following lines to the file:

```
auto wlan0
iface wlan0 inet dhcp
wpa-ssid darknet
wpa-psk t0p$ecret
```

These lines will activate the wlan0 interface automatically the next time you boot the Pi. Also they'll make the Pi obtain an IP address using DHCP. The Pi will try to join the network named darknet and it'll use the password 't0p$ecret;. Of course, you have to adjust the network name and the password accordingly.

If you're impatient, you do not have to reboot the Pi. Run the following command to make the Pi join your wireless network:

```
pi@raspberrypi ~ $ sudo ifup wlan0
Internet Systems Consortium DHCP Client 4.2.2
Copyright 2004-2011 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/wlan0/00:12:fb:28:a9:51
Sending on   LPF/wlan0/00:12:fb:28:a9:51
Sending on   Socket/fallback
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 8
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 14
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 14
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPOFFER from 192.168.1.1
DHCPACK from 192.168.1.1
bound to 192.168.1.101 -- renewal in 2983 seconds.
```

The Pi has the IP address 192.168.1.101 now and is connected to your network wirelessly (your IP address probably will differ). Use the ping command to check whether you can access a web site like Google, for example:

```
pi@raspberrypi ~ $ ping -c 3 google.com
PING google.com (173.194.69.100) 56(84) bytes of data.
64 bytes from google.com (173.194.69.100): icmp_req=1 ttl=45 time=26.7 ms
64 bytes from google.com (173.194.69.100): icmp_req=2 ttl=45 time=32.3 ms
64 bytes from google.com (173.194.69.100): icmp_req=3 ttl=45 time=34.8 ms

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 26.752/31.338/34.863/3.395 ms
```

As you can see, the Pi is connected to the Internet via WiFi now. Run the following command to get some statistics about the signal strength etc.:

```
pi@raspberrypi ~ $ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11abgn  ESSID:"darknet"
          Mode:Managed  Frequency:2.442 GHz  Access Point: 54:E6:FC:CF:77:8A
          Bit Rate=135 Mb/s   Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off    Fragment thr:off
          Power Management:on
          Link Quality=40/70  Signal level=-70 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:1  Invalid misc:6   Missed beacon:0
```

Keep in mind that a computer like the Pi can have more than one IP address. If you connect it via Ethernet and WiFi, for example, your boot message will display something like the following:

```
My IP address is 192.168.2.109 192.168.1.101
```

That means that you've connected your Pi using two network interfaces and for each interface it has a different IP address.

6.6   Next Steps

In this chapter, you learned how to integrate the Pi into your network. You can now conveniently access the Pi via SSH, and you can even use it as a web server. In the next chapter, you'll do something completely different: you'll turn the Pi into a multimedia center.

Turn the Pi into a Multimedia Center

The Pi's small size, its low power consumption, and its graphics capabilities make it a perfect candidate for a fully integrated multimedia center just like a PlayStation or an Apple TV. To turn the Pi into such a multimedia center, you need a special piece of software named XBMC.[1]

XBMC is a media player on steroids that can turn nearly every PC into an entertainment hub for digital media. The Pi is no exception. In this chapter, you'll learn how to run XBMC on the Pi.
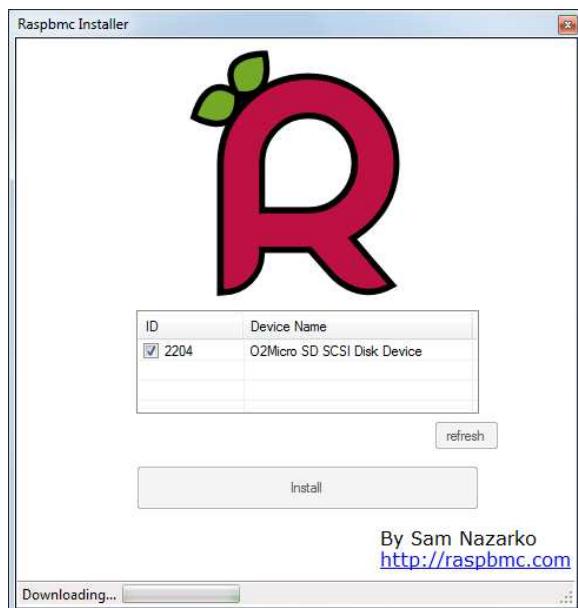
7.1 Install Raspbmc

XBMC is a really big software project, and its installation and configuration can be tricky. Fortunately, you do not have to do it yourself for the Pi, because you can benefit from the glorious efforts of the Raspbmc[2] team. Raspbmc is a Linux distribution for the Pi that does nothing but run XBMC. You copy an image of this distribution to an SD card as usual, and then you use this SD card to boot the Pi. Instead of starting a terminal or a desktop environment, the Raspbmc distribution starts XBMC automatically.

In contrast to other Linux distributions for the Pi, the Raspbmc team does not offer a complete image file for an SD card for download. They decided to create an installer for all major platforms instead. This installer downloads the latest version of Raspbmc from the Web and copies it to your SD card automatically.

---

1. http://xbmc.org/
2. http://www.raspbmc.com/

If you're using a Windows PC to prepare a Raspbmc card, download the installer,[3] extract it to your hard drive, and start the program named installer.exe. You'll see a window similar to this:

Insert an SD card, select your SD card reader, and click the Install button. The installer then downloads the latest version of Raspbmc and copies it to the SD card. Note that the installer will delete all the data on the SD card!

The Raspbmc installer for Linux and Mac OS X does not have a fancy UI, but it's easy to use, too. It's a Python program, and after you've downloaded it,[4] you can run it from a terminal as in the following example:

```
maik> sudo python install.py
```

```
Raspbmc installer for Linux and Mac OS X
http://raspbmc.com
----------------------------------------
Please ensure you've inserted your SD card, and press Enter to continue.

Enter the 'IDENTIFIER' of the device you would like imaged, from the following list:
#:              TYPE NAME          SIZE        IDENTIFIER
1:              EFI                209.7 MB    disk0s1
2:        Apple_HFS Macintosh HD   499.2 GB    disk0s2
3:       Apple_Boot Recovery HD    650.0 MB    disk0s3
#:              TYPE NAME          SIZE        IDENTIFIER
```

---

3. http://www.raspbmc.com/wiki/user/windows-installation/

4. http://svn.stmlabs.com/svn/raspbmc/testing/installers/python/install.py

```
1:      Windows_FAT_32              67.1 MB    disk1s1
2:          Linux_Swap            129.0 MB    disk1s2
3:              Linux              16.0 GB    disk1s3
```

⇒ **Enter your choice here (e.g. 'disk1s1'): disk1s1**
❮ It is your own responsibility to ensure there is no data loss!
  Please backup your system before imaging
⇒ **Are you sure you want to install Raspbmc to '/dev/disk1s1'? [y/N] y**
❮ Downloading, please be patient...
  Downloaded 49.52 of 49.52 MiB (100.00%)

  Unmounting the drive in preparation for writing...
  Unmount of all volumes on disk1 was successful
  Please wait while Raspbmc is installed to your SD card...
  (This may take some time and no progress will be reported until it has finished.)
  0+3022 records in
  0+3022 records out
  198000640 bytes transferred in 27.034211 secs (7324077 bytes/sec)
  Installation complete.
  Finalising SD card, please wait...
  Disk /dev/rdisk1 ejected

  Raspbmc is now ready to finish setup on your Pi, please insert the
  SD card with an active internet connection
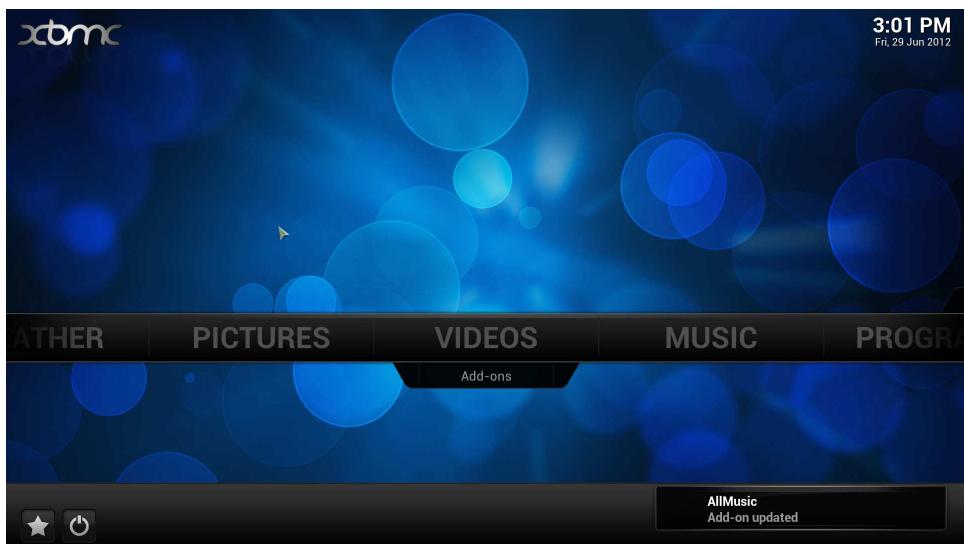
The installation program shows a list of all the drives connected to your PC including the SD card reader. On your PC the output will be different, but in the previous example, the SD card consists of the three partitions named disk1s1, disk1s2, and disk1s3. Enter the name of the first partition—disk1s1 in this case—and then confirm that you'd like to install Raspbmc. Make sure you chose the right drive, because the installer deletes all the data on the device you've selected!

After you've created a bootable SD card, insert it into the Pi and turn it on. Surprisingly, the Pi will not start XBMC right away but boots a minimalistic Linux system and starts the actual installation of Raspbmc. First it repartitions the SD card and formats the newly created partitions. Then it downloads and installs the root file system, the kernel, some kernel modules, and a few libraries. After a reboot, it eventually downloads and installs the latest version of XBMC.

All these steps require no user interaction. Depending on the speed of your SD card and your Internet connection, they take about twenty minutes. So, you can safely go for a walk or have a cup of your favorite hot beverage.

7.2 Start Raspbmc for the First Time

After the installation process has finished, it starts Raspbmc automatically. When Raspbmc starts for the first time, it will probably open a few dialog boxes informing you about a couple of broken add-ons. You can safely disable them by clicking Yes. After that, XBMC is ready, and its main menu looks like the following:
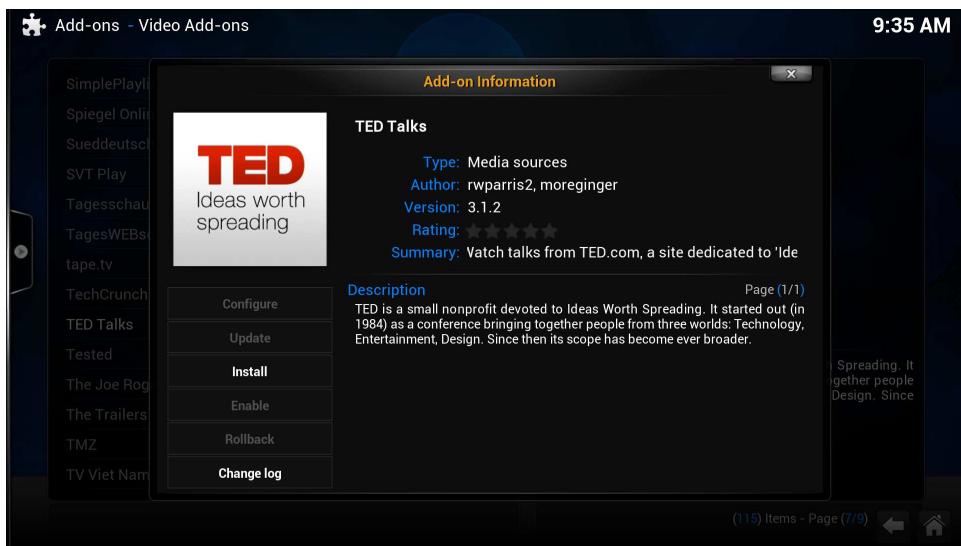
At first sight, XBMC looks like many other media players. It has menu items for viewing photos, watching videos, playing music, and configuring some system preferences. These functions are mainly self-explanatory—for playing or viewing any kind of content, you can simply select media files from the SD card or a USB device, and XBMC will output them.

To attach a USB device such as a hard drive or a USB stick to the Pi, you have to use a USB hub, or you have to temporarily disconnect your mouse and control XBMC using your keyboard. Instead of choosing a menu item by clicking it with the mouse, you can move the focus with the cursor keys and press Return to select an item. With the Escape key, you go back one step in the menu hierarchy.

But XBMC is more than a simple media player; you can improve and enhance it using many add-ons that are available for free on the Web. Simply put, add-ons give you access to media on the Web. For example, you can find add-ons that aggregate the content of certain TV stations or add-ons that give you access to the music of the greatest video games. XBMC even provides a very

convenient way for managing add-ons. In the following image, you can see the TED add-on, which lists the latest and greatest TED conference videos.

Take a few minutes, and browse the list of add-ons to see whether there's something interesting for you. When in doubt, install it and take a look. It's easy to remove an add-on if you do not like it. Note that you need enough bandwidth for most add-ons, because they stream a lot of data.

Depending on the speed of your SD card and your Internet connection, you'll experience a noticeable lag when choosing menu items in XBMC. This might get better in future releases, but for the moment you have to live with it and be patient when navigating through XBMC's menus. Playing content works fine, without any lags or staggering.

Finally, you should take a look at the Systems > Settings menu and see whether all settings match your local setup. If you're using composite video, for example, you have to choose analog as your audio output device in Systems > Settings > Audio.

7.3 Add Files to XBMC

In XBMC you can easily add new movies, TV shows, or music using the Add Files menu in the Videos or Music menu. Before you start to add your media files, you should know how XBMC works internally. XBMC is more than a simple media player. It is a full-blown media library that tries to get as much information about your media files automatically as possible. For example, XBMC reads additional information about your favorite TV shows from web

databases and adds them to your library. This can be anything ranging from an episode's original air date to a short summary. To do this, XBMC depends on a certain filenaming scheme—you can read all about it on the project's wiki.[5] Note that choosing the right filenames and directory structures even affects XBMC's main menu. If you add a directory for TV shows, for example, XBMC adds a TV shows menu item to its main menu. So, to get the most out of XBMC, you should rename your media files accordingly before you import them.

Video and Music Formats

XBMC supports nearly all container formats and codecs on the market. You'll rarely find a multimedia file that you cannot play with the Pi. There's one problem, though: the Raspberry foundation licensed hardware acceleration only for the H.264 video codec. Fortunately, this is one of the most popular codecs available, but if you have video files using a different codec, you might experience some performance problems.

The easiest way to add files to XBMC is to attach a USB device containing your media files directly to the Pi. This solution works fine, but it also has some disadvantages. The USB device often is bigger than the Pi itself, and it probably consumes more power, too. In addition, it needs at least one of your valuable USB ports. A better solution is to store media on the SD card or to stream media from your local network. Because of XBMC's great network integration, you can implement both solutions easily.

XBMC supports FTP, SSH, NFS, and Samba out of the box, and you do not have to configure much to get them all up and running. XBMC enables SSH by default, so to copy data to the SD card, you can use scp, for example, as you did in Section 6.2, *Use Secure Shell with the Pi*, on page 51. Run the following command from your PC's terminal to create a folder named Movies in XBMC's home directory:

```
maik> ssh pi@192.168.2.109 "mkdir /home/pi/Movies"
```

Replace the IP address with the address of your Pi. Note that Raspbmc also comes with a user named pi who also has the password raspberry at the moment. Now you can copy media files using scp and add them to the XBMC library afterward.

```
maik> scp Pulp\ Fiction\ (1994).avi pi@192.168.2.109:Movies
```

---

5. http://wiki.xbmc.org/index.php?title=Adding_videos_to_the_library/Naming_files

Even if you use an SD card with plenty of space, it will probably not be enough to store your whole media library. Also, it does not make sense to always copy files before you can watch a film or listen to some music. That's what network file systems such as NFS and Samba were built for, and XBMC supports them all.

Using NFS or Samba, you can host all your media files on your regular PC and stream them to the Pi when you want to use them. Configuring NFS[6] or Samba[7] is beyond the scope of this book, but the XBMC wiki has excellent documentation for all major platforms.

As soon as your media files are available in your home network via NFS or Samba, you can easily access them using XBMC. For NFS you usually do not have to do anything, and you can configure your Samba settings in the System > Service > SMB client menu.

7.4 Control XBMC Remotely

If you want to use the Pi as a multimedia center in your living room, you'll sooner or later feel the need for a remote control. You can use some special hardware with an infrared dongle,[8] but the easiest way is just to use your smartphone.

Not only does XBMC have add-ons for managing your multimedia files, it also allows you to install nice web interfaces for controlling XBMC remotely. Go to the System > Settings > Services > Webserver menu and enable the "Allow control of XBMC via HTTP" option. Then click the "Web interface" button and select Get More…. In Figure 19, *XBMC comes with several web interfaces*, on page 76, you can see the web interfaces that are currently available. Simply enable all of them so you can try them all and choose your favorite.

After you've enabled the web interface, you can use it with every browser that has access to your network. The interface listens on port 8080, so in your browser, you have to enter a URL like http://192.168.2.109:8080 to open XBMC's web interface (remember to replace the IP address with your Pi's IP address).

In Figure 20, *The AWXi web interface in action*, on page 76, you can see the AWXi web interface in action, for example. It has all the usual buttons such as play, pause, and stop, and it allows you to search your whole media library.

---

6. http://wiki.xbmc.org/index.php?title=NFS
7. http://wiki.xbmc.org/index.php?title=Samba
8. http://www.raspbmc.com/wiki/user/configuring-remotes/

report erratum • discuss
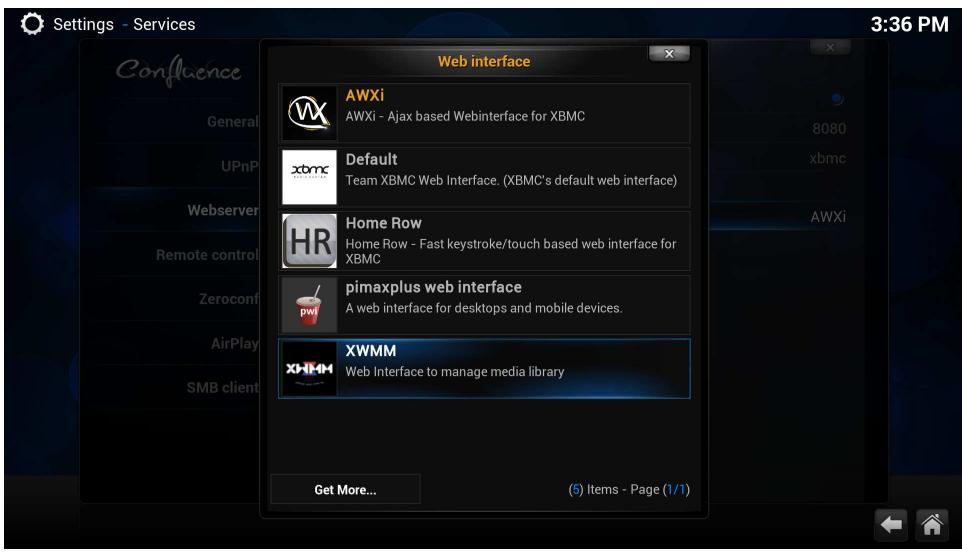
Figure 19—XBMC comes with several web interfaces.
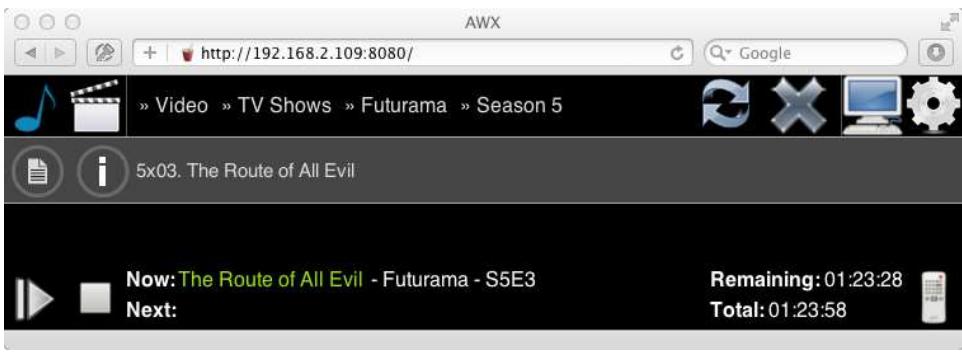
Figure 20—The AWXi web interface in action

If you have an iPad/iPhone[9] or an Android phone,[10] you can even install a native remote control application for XBMC. In Figure 21, *Control XBMC on an Android device*, on page 77 and Figure 22, *It looks and works like a regular remote*, on page 77, you can see the Android version. It not only looks beautiful but also provides convenient access to all XBMC functions. In many regards, it's way better than regular remote controls for TV sets.

---

9. http://itunes.apple.com/gb/app/official-xbmc-remote/id520480364
10. https://play.google.com/store/apps/details?id=org.xbmc.android.remote
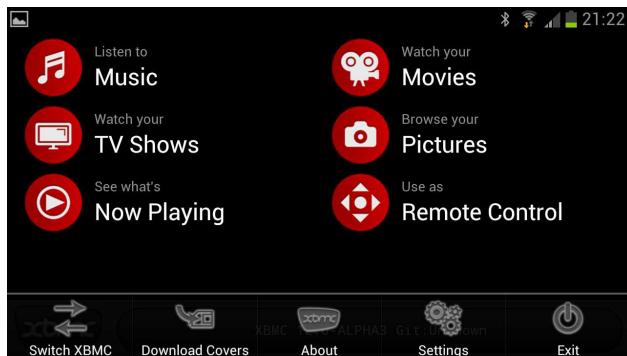
Figure 21—Control XBMC on an Android device.
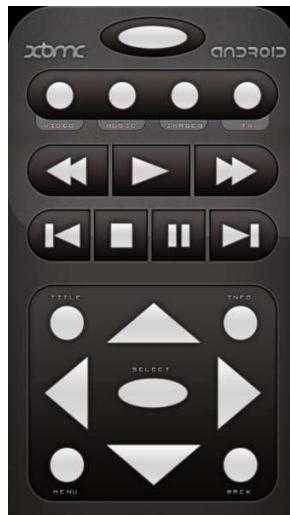
Figure 22—It looks and works like a regular remote.

7.5 Next Steps

In this chapter, you learned to do something completely different with the Pi. For the first time you did not use it as a regular PC but turned it into a special-purpose device, a multimedia center. In the next chapter, you'll learn how to run even more multimedia applications on your Pi, and you'll play some entertaining games.

Play Games on Your Pi

Linux has never been a popular gaming platform. Even though the situation has improved over the years, it will probably take some time until the first blockbuster titles become available. Still, you can already play some entertaining and sometimes even addictive games on the Raspberry Pi.

For example, you can enjoy thousands of text adventure games on the Pi. Although commercial publishers abandoned this genre long ago, it still has an active and enthusiastic fan base that releases new games frequently. And if you haven't played classic games such as *Zork*, you should give them a try.

Another classic genre is point-and-click adventure, including games such as *The Secret of Monkey Island* and *Day of the Tentacle*. Thanks to the efforts of the open source community, you can play most titles on your Pi.

Even if the Pi is not powerful enough to run modern games, it still has enough power to run some native Linux games such as *Quake III*. It even has enough power to emulate some home computers and game consoles from the past. For example, you can play all games made for Atari's VCS 2600 on the Pi.

8.1 Play Interactive Fiction

In the era of the first home computers, text adventure games were very popular. In contrast to modern games with spectacular 3D graphics and surround sound, text adventures look very spartan. They output only text, and you control the game by typing commands on your keyboard. Here you can see the opening of *Zork*, one of the most famous text adventures.

```
ZORK I: The Great Underground Empire
Copyright (c) 1981, 1982, 1983 Infocom, Inc. All rights reserved.
ZORK is a registered trademark of Infocom, Inc.
Revision 88 / Serial number 840726
```

```
West of House
You are standing in an open field west of a white house, with a
boarded front door.
There is a small mailbox here.
```

⇒ **>open mailbox**
❮ Opening the small mailbox reveals a leaflet.

⇒ **>take leaflet**
❮ Taken.

⇒ **>read leaflet**
❮ "WELCOME TO ZORK!

```
ZORK is a game of adventure, danger, and low cunning. In it you will
explore some of the most amazing territory
ever seen by mortals. No computer should be without one!"
```

Don't be misled by the presentation of the games. Many of them tell great stories and will entertain you for many hours.

Even though no commercial text adventures have been released for decades, the genre still has an active community that produces exciting games. Most of these games tell long and elaborate stories, so their authors prefer to call their creations *interactive fiction*.

One of the first companies to produce interactive fiction was Infocom. Over several years it has created some of the greatest text adventure games ever. The Infocom developers realized early that they could reduce their efforts by creating a domain-specific language for describing interactive fiction. They called this language *Z-language*, and authors of interactive fiction still use it today to create games.

To run programs written in Z-language, you need an implementation of a virtual machine named Z-machine,[1] and one of the best is Frotz.[2] You can install it as follows:

```
pi@raspberrypi ~ $ sudo apt-get install frotz
```

To play a text adventure using Frotz, you only need the game's Z-language file. A great place to start your search for interactive fiction is the Interactive Fiction Archive,[3] because it hosts thousands of games.

---

1. http://en.wikipedia.org/wiki/Z-machine
2. http://frotz.sourceforge.net/
3. http://www.ifarchive.org/

If you're new to interactive fiction, you should start your journey with the *Zork* trilogy. This series of games made Infocom famous, and although they are a couple of decades old, they are still as fresh as on the first day. Meanwhile, they are available for free,[4] so download *Zork I*,[5] and start it as follows:

```
pi@raspberrypi ~ $ unzip zork1.zip
pi@raspberrypi ~ $ frotz zork1/DATA/ZORK1.DAT
```

This invokes the Z-machine interpreter and runs the game stored in ZORK1.DAT. You might need a few moments to get used to this kind of game,[6] but it's certainly well worth it.

If you enjoy playing interactive fiction, you might also enjoy creating it with today's tools. It's really easy,[7] at least on a technical level. You still have to come up with a nice and original story.

8.2   Play Point-and-Click Adventures

Another genre that has always been popular is point-and-click adventures. In these games, you control the main character using the mouse. You can make the character move to a certain place on the screen by clicking the place, and you can perform actions by clicking the action's name on the screen. Popular point-and-click adventures are *The Secret of Monkey Island*, *Day of the Tentacle*, and *Maniac Mansion*.

There's always been a demand for new point-and-click adventures, but over the last years very few have been released. The big publishers did not believe in them and preferred to release countless first-person shooters such as *Call of Duty* or *Battlefield* instead.

Tim Schafer, one of the creators of *The Secret of Monkey Island*, got frustrated by this situation and tried to raise some money on Kickstarter.com to fund a new point-and-click adventure. He raised more than $3.3 million and proved that people are still very interested in this genre.

By the time of this writing, Tim's new game has not been yet released, but fortunately you can play many classic games. Similar to text adventures, most point-and-click adventures run on a virtual machine. The most popular one is SCUMM, which stands for Script Creation Utility for Maniac Mansion.

---

4.   http://www.infocom-if.org/downloads/downloads.html

5.   http://www.infocom-if.org/downloads/zork1.zip

6.   At http://pr-if.org/doc/play-if-card/play-if-card.html you can find a nice help sheet.

7.   http://inform7.com/

Originally it was created by the developers of LucasArts to implement the game *Maniac Mansion*, and they have used it to implement many other games since.

The ScummVM[8] project implements a virtual machine that interprets SCUMM games, and it is available for free. You can install it as follows:

```
pi@raspberrypi ~ $ sudo apt-get install scummvm
pi@raspberrypi ~ $ sudo apt-get install beneath-a-steel-sky
pi@raspberrypi ~ $ sudo apt-get install flight-of-the-amazon-queen
```

This will not only install ScummVM but also two great games you can play with it. These games will appear in the Games section of your desktop's Start menu. In Figure 23, *Beneath a Steel Sky is still a great game*, on page 82, you can see *Beneath a Steel Sky*.
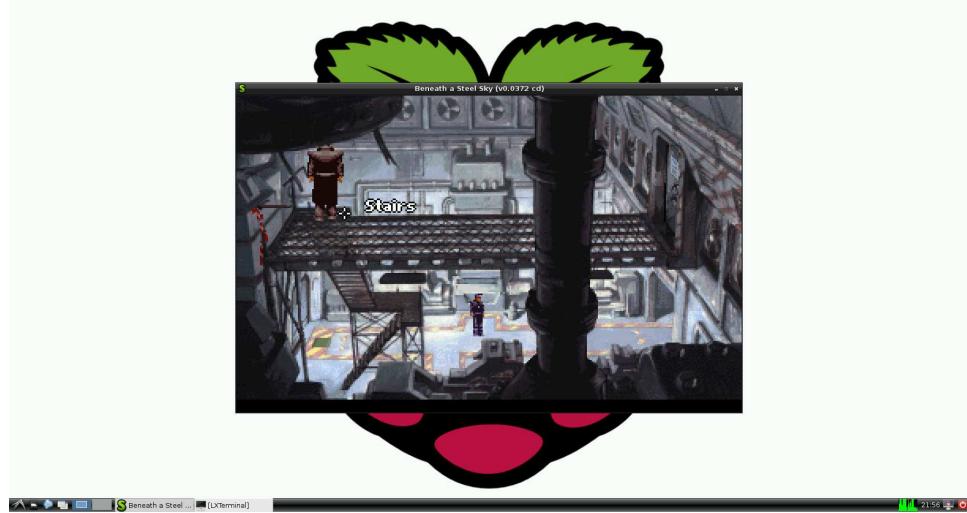
Figure 23—*Beneath a Steel Sky* is still a great game.

Beneath a Steel Sky and *Flight of the Amazon Queen* are freeware, so you can safely install them. Most other games are not available for free, so you are allowed to install them only if you own the original. If you own other games that are compatible with ScummVM, you can start ScummVM directly and add them.

---

8. http://www.scummvm.org/

8.3 Emulate Other Platforms

Another way to play some fine games on the Pi is to emulate other platforms. Many emulators are available for Linux, and they reanimate classic computers and game consoles such as the Commodore 64, Sega's Mega-Drive, the Nintendo Entertainment System (NES), and many more. Probably there's at least one emulator for every single system from the past.

An emulator rebuilds the hardware of a certain computer or game console in software. So, you run the emulator on the Pi, and then you can work with the emulated system as if you're using the original hardware. Most importantly, you can run all the old software and games that were available for the old system.

Emulating a complete computer is extremely difficult even for simple systems, and most emulators suffer from two major problems. The first one is accuracy; often an emulator is not able to emulate the original system at 100 percent. The second problem is performance, because emulating even very old and slow hardware needs a tremendous amount of resources. For example, the Commodore 64 ran at a clock speed of only 1MHz, but you need a lot of computing power to emulate it. Even the Pi's hardware is not powerful enough to emulate the C64 at a reasonable frame rate at the moment, although it has a multiple of the C64's resources. This might change with better graphics drivers for the Pi.

Still, the Pi is powerful enough to emulate some cool game consoles, and one of them is the Atari VCS 2600.[9] This device was very popular from 1977 until the early 90s, and you could play great games like Pac-Man, Centipede, or Pitfall on it. The console was so popular that several emulators exist for it, and one of the best is Stella.[10] You can install and start it as follows:

```
pi@raspberrypi ~ $ sudo apt-get install stella
pi@raspberrypi ~ $ stella
```

First Stella asks you where it should look for game ROMs. The games for the VCS 2600 shipped on cartridges containing a few kilobytes of read-only memory (ROM). To play a game using Stella, you need a copy of its cartridge's ROM. You can copy the content of a cartridge to your PC using a special device. Fortunately, you can find ROM files for all games on the Web,[11] but there's one big problem: although most games for the VCS 2600 are very old,

---

9. http://en.wikipedia.org/wiki/Atari_2600
10. http://stella.sourceforge.net/
11. http://atariage.com/

they are still copyrighted. So, in most countries, it's illegal to download and use ROM files of games that you do not actually own!

You can buy used cartridges on the Web for very little money, and some publishers still sell Atari games collections that ship on CDs. These collections do nothing but run an emulator and play the original ROM files.

The size of a ROM file is usually between 4KB and 8KB, and the filenames end with the extension .bin. So, Pac-Man's ROM file is named pacman.bin, for example. If you have copied a ROM file to the Pi, you can select it in Stella's main menu, and it will start immediately. By default, you can use the cursor keys for movements and the spacebar for actions. Stella allows you to remap all keys, and it also has support for joysticks. On top of that, you can change countless video and audio options, but note that the Pi will not emulate the VCS 2600 properly in the most demanding video modes.

Playing some classic games might bring back warm and fuzzy childhood memories, but the VCS 2600 has an incredibly active user group that still creates games.[12] Many of these homebrew titles even look and sound better than most of the original games, and they are usually available for free. In Figure 24, *People still create great games for the VCS 2600,* on page 85, you can see *A-VCS-tec Challenge*, for example.[13] Some of these home-brew games are available on cartridges even today.

By the way, developing games or demos for the VCS 2600 is very difficult, but you can learn a lot from it, and it can be fun! Most people cannot imagine how limited the hardware was. It ran at a clock speed of 1.19MHz, it had only 128 bytes of RAM, and it had no frame buffer for the video display. Developing software for this machine was a really painful act back in the old days, but today's tools and documentation make it much easier. For example, Stella comes with a debugger that allows you to see and change the state of a game while it is running. To enable the debugger, start Stella like this:

```
pi@raspberrypi ~ $ stella -debug
```

To invoke the debugger, press the backquote key ('), and remember that you can freely remap Stella's actions to other keys if you cannot find it on your keyboard. In Figure 25, *Stella comes with a powerful debugger,* on page 85, you can see the debugger in action.

---

12. http://en.wikipedia.org/wiki/Atari_2600_homebrew
13. http://www.quernhorst.de/atari/ac.html
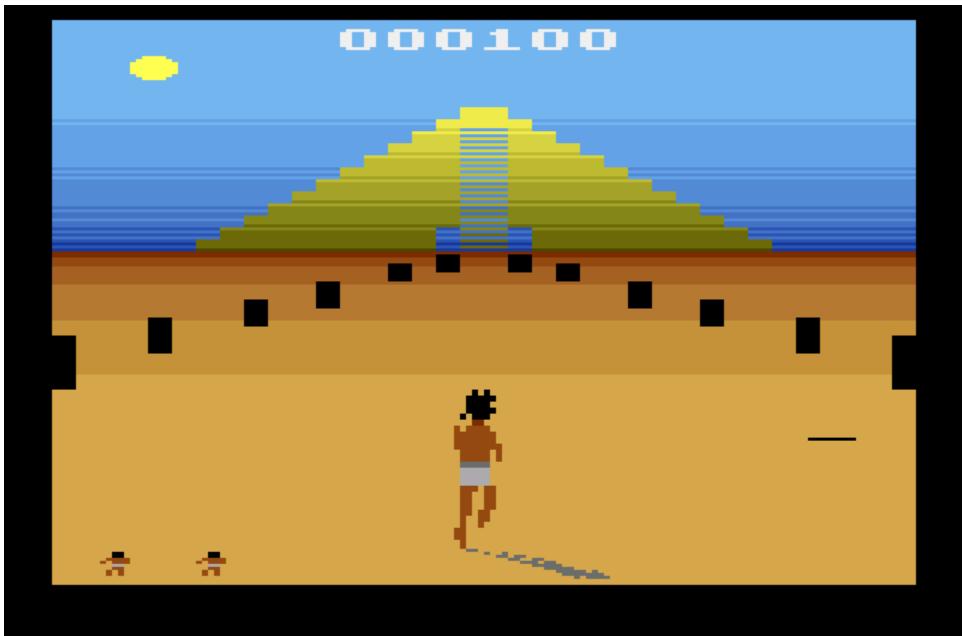
Figure 24—People still create great games for the VCS 2600.
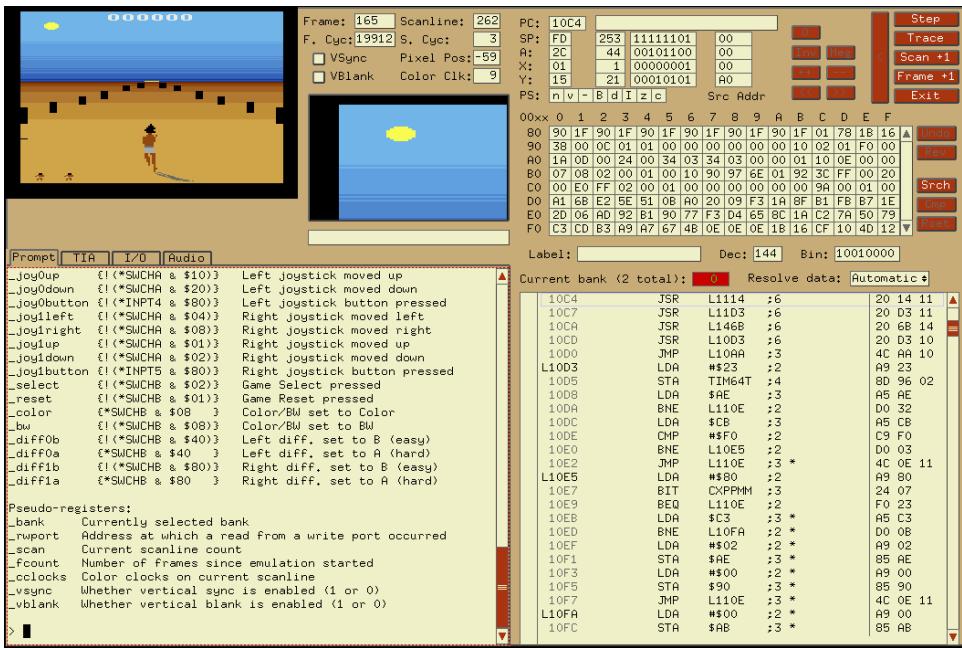
Figure 25—Stella comes with a powerful debugger.

All in all, Stella works pretty good on the Pi, because the VCS 2600 is not a very strong machine. Emulators for other systems do not work so well at the moment. The C64 emulator Vice,[14] for example, works in principle on the Pi, but its frame rate is too low for playing most games. The same is true for arcade machine emulators such as MAME,[15] but the situation might change soon. Until then, you could work on your own VCS 2600 game.

8.4   Play Native Games

In the preceding sections, you've learned about technologies that help you run games using virtual machines and emulators, but there are also native games for Linux. For example, on the LXDE desktop, you can find a compilation of some classic games written in Python: *Four in a Row*, *Snake*, and so on.

Because the Pi is a regular Linux system, you can run every game compatible with your current distribution as long as the Pi's resources are sufficient. If you search the Web for classic games like Tetris or Pac-Man, you'll quickly find really good clones.

```
pi@raspberrypi ~ $ sudo apt-get install ltris pacman
```

It's worth trying to install all the games you find, but many are too demanding for the Pi. For example, the pinball and extremetuxracer packages will not work on the Pi.

Surprisingly, the Pi is capable of running *Quake II* and *Quake III* (two famous first-person shooters) at a decent frame rate. At the moment of this writing, they have some problems with sound output, but they're still playable. So, keep looking for new Pi games.

8.5   Next Steps

In this chapter, you learned how to kill some time playing classic games on the Pi. The Pi might not be an Xbox or a PlayStation, but it runs some entertaining games that you won't find on most modern video game consoles. The next chapter deals with a completely different topic. You'll learn how to build and attach electronics projects to the Pi's GPIO ports.

---

14. http://vice-emu.sourceforge.net/
15. http://mamedev.org/

Tinker with the GPIO Pins

The Raspberry Foundation built the Pi not only for teaching kids how to program but also to teach them how to tinker with electronics. That's why the Pi has an expansion header that makes it easy to connect it to your own electronics projects.

In this chapter, you'll learn how to build your own small electronics devices and control them with the Pi. You'll start slowly and build a very basic circuit that makes a light-emitting diode (LED) shine. After that, you'll control the LED using the Pi's expansion header, and you'll turn the LED on and off by issuing commands from the Pi.

Then you'll build a small memory alarm that is a device showing how much memory is left on your Pi. It will work like a traffic light where a red light means that the amount of remaining memory is critically low. In addition, you'll make the results of the memory alarm available in your web browser.

9.1 What You Need

To build all projects in this chapter, you need only a few cheap parts (you can see them all in Figure 26, *The parts you need*, on page 88).

- A half-size breadboard
- Three 5mm LEDs (red, yellow, and green)
- Three resistors in the range of 220Ω to 1kΩ
- Four female/male jumper wires
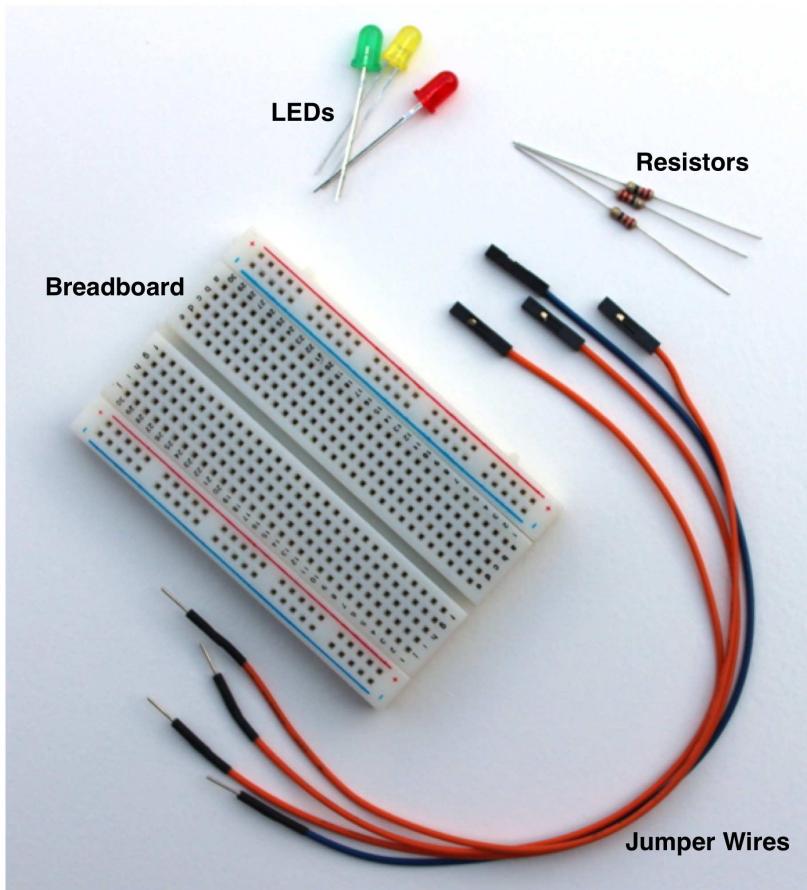
Figure 26—The parts you need

You can get these parts at any shop selling electronic parts, for example RadioShack,[1] sparkfun,[2] Mouser,[3] Digi-Key,[4] and Adafruit.[5] Note that buying single LEDs, resistors, or jumper wires does not make much sense. You get these parts much cheaper when you buy them as a pack. For example, RadioShack sells packs of LEDs (catalog number 276-1622) and resistors (catalog number 271-308). Adafruit has a nice pack of jumper wires (product ID 826), and it also sells breadboards (product ID 64).

---

1. http://radioshack.com
2. http://www.sparkfun.com/
3. http://mouser.com
4. http://digikey.com
5. http://adafruit.com

9.2 Meet the Pi's GPIO Pins

To connect your own electronics projects to the Pi, you can use the expansion header in the top-left corner of the Pi (see Figure 1, *The front side of a Model B*, on page 2). It consists of 26 pins arranged in two rows containing 13 pins each. The top row contains the even-numbered pins, and the other row contains the odd-numbered pins. That is, the first pin in the lower row is pin 1, and you can find the label "P1" on the Pi below the pin.

In Figure 27, *The Pi's GPIO pins*, on page 89, you can see the meaning and the numbering of the pins. With pin 6, the Pi can share a common ground with your electronics projects. Using pins 1 and 2, you can power external devices connected to the Pi with 3.3 volts or 5 volts. The Pi limits the output of pin 1 to 50mA, while pin 2 allows for a current draw that depends on the USB input current. If you power the Pi with a 1A power supply, for example, you can draw up to 300mA from pin 2, because the Pi Model B needs 700mA for itself.

	5V	-	Ground	GPIO14	GPIO15	GPIO18	-	GPIO23	GPIO24	-	GPIO25	GPIO8	GPIO7
	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
Pin	2	4	6	8	10	12	14	16	18	20	22	24	26
Pin	1	3	5	7	9	11	13	15	17	19	21	23	25
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	3v3	GPIO0	GPIO1	GPIO4	-	GPIO17	GPIO21	GPIO22	-	GPIO10	GPIO9	GPIO11	-

Figure 27—The Pi's GPIO pins

Pins 4, 9, 14, 17, 20, and 25 are reserved for future enhancements, so you cannot use them in your own projects. The remaining pins are general-purpose input/output (GPIO) pins that you can use as digital input or output pins. Note that the GPIO pin names do not correspond to the pin numbers of the expansion header.

You can use the GPIO pins, for example, to read the state of a push button or to turn an LED on and off. For this chapter's examples, you can assume that all GPIO pins work the same, but you should know that some of the Pi's pins are special. Pin 12, for example, supports Pulse Width Modulation (PWM),[6] which can be handy for controlling motors. If you're going to build more complex projects, you should take a look at a more detailed description of the Pi's pins.[7]

---

6. http://en.wikipedia.org/wiki/Pulse_width_modulation
7. http://elinux.org/RPi_Low-level_peripherals

9.3 Build a Basic Circuit

To warm up, you'll build one of the most basic circuits possible. You'll connect an LED to the Pi and make it shine as long as the Pi is running. For this you need an LED, a resistor, a breadboard, and two female/male jumper wires. Using these parts, you'll build the circuit in Figure 28, *A basic circuit*, on page 90.
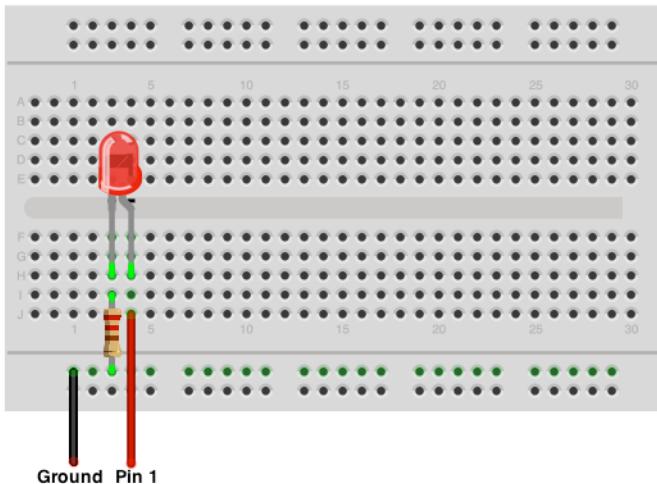
Figure 28—A basic circuit

Before you actually build the circuit, you should know what all the parts are for and how they work. Breadboards are useful tools for prototyping circuits. You can simply plug in parts like LEDs and resistors, so you do not have to solder them. Breadboards come in various sizes, but they all look very similar. On all of them, you can find many sockets arranged in columns. Most breadboards also have two rows of sockets at the top and at the bottom.

The main trick of a breadboard is that it automatically connects the sockets belonging to a certain column and to a certain row. In the basic circuit in Figure 28, *A basic circuit*, on page 90, you connect the Pi's Ground pin to the second-to-last row of the breadboard. This automatically connects all sockets in this row to the Pi's ground (that's why all the sockets in this row have a light green color). The same happens in the two columns connected to the LED. The resistor indirectly connects the Pi's Ground pin to one of the LED's connectors. In addition, you connect the Pi's pin 1 directly to the other connector of the LED by plugging it into a socket in the same column.

By the way, LED stands for light-emitting diode, so an LED is basically a diode. Diodes are very useful, because they let electricity pass in only one direction. That's true for LEDs, too, and in addition, LEDs emit light as a side effect.

Working with LEDs isn't very difficult, but you have to take care of a few things. First, you have to connect them the right way. LEDs have two wires, and one of them is a bit shorter than the other. The shorter wire is named *cathode* (negative), and you have to connect it to the Pi's ground pin. The longer wire is named *anode* (positive), and you have to connect it to one of the Pi's power supply or GPIO pins. You can also identify the anode and cathode by taking a close look at the LED's case. The flat side belongs to the cathode and the round side to the anode. In Figure 28, *A basic circuit,* on page 90, the anode is slightly bent.

Also, you always have to put a resistor in front of an LED. If you don't, the LED consumes too much power and will be destroyed. Simply put, a resistor limits the amount of current that flows through an electric connection and protects the LED. Calculating the resistor value for a certain type of LED is not difficult, but it's beyond the scope of this book. Simply keep in mind that the lower the resistor value, the brighter the light will shine. When in doubt, use a 330Ω or 470Ω resistor.

Now it's time to actually build the circuit. First, connect the LED to the breadboard. Make sure that the direction of the LED is right, and plug it in. You have to press firmly but not too hard—otherwise, you'll bend the connectors, and they won't fit. It's usually easier to plug parts in after you've shortened the connectors. When cutting the connectors, wear safety glasses to protect your eyes!

The resistor is next, and this time the direction doesn't matter. Before plugging the resistor in, you have to bend its connectors. Also, it usually helps to shorten them.

Finally, connect the two jumper wires to the Pi and to the breadboard. Connect the female side to the Pi and the male side to the breadboard. Make sure you're using the right pins on the Pi, and then turn on the Pi. If you've connected everything correctly, the LED will turn on, too. Otherwise, take a look at Section 9.7, *What If It Doesn't Work?,* on page 98.

9.4 Control an LED Using the GPIO Pins

Making an LED shine is a nice exercise, but it gets boring pretty quickly. In this section, you'll learn how to control an LED using software—you'll turn it on and off by issuing commands on the Pi.

Programming hardware directly is usually a difficult task. Using the WiringPi project,[8] it becomes a piece of cake. WiringPi is an open source project that hides the ugly low-level functions behind a nice and clean interface. If you have worked with the popular Arduino project[9] before, WiringPi will look very familiar because it tries to bring most of the Arduino goodies to the Pi. WiringPi not only makes programming the Pi's hardware easier, but it also comes with a small command-line utility named gpio that allows you to control the hardware without writing code.

You can install WiringPi on the Pi as follows:

```
pi@raspberry:~$ cd /tmp
pi@raspberry:~$ wget http://project-downloads.drogon.net/files/wiringPi.tgz
pi@raspberry:~$ tar xfz wiringPi.tgz
pi@raspberry:~$ cd wiringPi/wiringPi
pi@raspberry:~$ make
pi@raspberry:~$ sudo make install
pi@raspberry:~$ cd ../gpio
pi@raspberry:~$ make
pi@raspberry:~$ sudo make install
```

These commands install the WiringPi libraries and the gpio command. You can use WiringPi from many programming languages such as C, C++, Python, Ruby, and so on. In this chapter, you'll use it exclusively from the command line and in a short but effective shell script.

All you need for your first interactive electronics experiments is the gpio command. It supports only a few options and three commands: mode, read, and write. For example, the following command sets the GPIO18 pin into output mode:

```
pi@raspberry:~$ gpio -g mode 18 out
```

All GPIO pins can be in one of two modes: in and out. To read digital signals from a GPIO pin, set its mode to in. Set it to out if you'd like to emit digital signals. You have to set a pin's mode only once, and it will remember its mode until you set it to a different one.

---

8. https://projects.drogon.net/raspberry-pi/wiringpi/
9. http://arduino.cc

After you've set GPIO18's mode to out, you can turn it on like this:

```
pi@raspberry:~$ gpio -g write 18 1
```

Turning it off works similarly:

```
pi@raspberry:~$ gpio -g write 18 0
```

Finally, you can read GPIO18's current state using the read command:

```
pi@raspberry:~$ gpio -g read 18
0
```

This command returns 0 if there's currently no signal and 1 otherwise.

With gpio, you can control the LED on your breadboard easily. You have to connect it to only one of the Pi's GPIO pins instead of connecting it directly to a power supply pin. For example, you can use GPIO18, which is pin 12 in Figure 27, *The Pi's GPIO pins*, on page 89. Choosing and addressing the right pins can be a bit confusing, because of the different naming and numbering schemes. By default WiringPi uses its own numbering scheme,[10] but for your first experiments you should use the "official" GPIO pin names. Fortunately, gpio accepts them when you pass it the -g option.

So, in your circuit disconnect the jumper wire from pin 1 and connect it to pin 12 instead. Then run the following commands:

```
pi@raspberry:~$ gpio -g mode 18 out
pi@raspberry:~$ gpio -g write 18 1
```

These commands will turn on the LED, and the following command turns it off:

```
pi@raspberry:~$ gpio -g write 18 0
```

With only a few parts and a single command-line tool, you've created your first circuit. Even better, you control it with the Raspberry Pi! In the next section, you'll create a more complex project using the techniques you've learned so far.
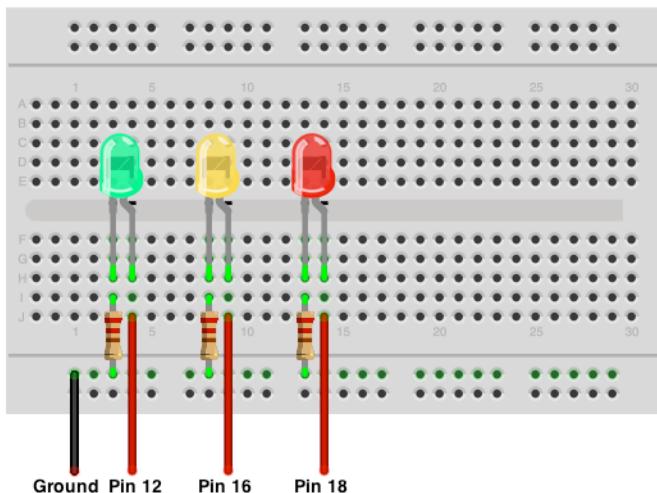
9.5 Build an "Out of Memory" Alarm

Turning an LED on and off using software is an important exercise, but it's certainly not a very useful project. Usually, you won't control LEDs manually but use them as status indicators. For example, many USB devices use LEDs to show whether they are reading or writing data at the moment.

––––––––––––––––

10. https://projects.drogon.net/raspberry-pi/wiringpi/pins/
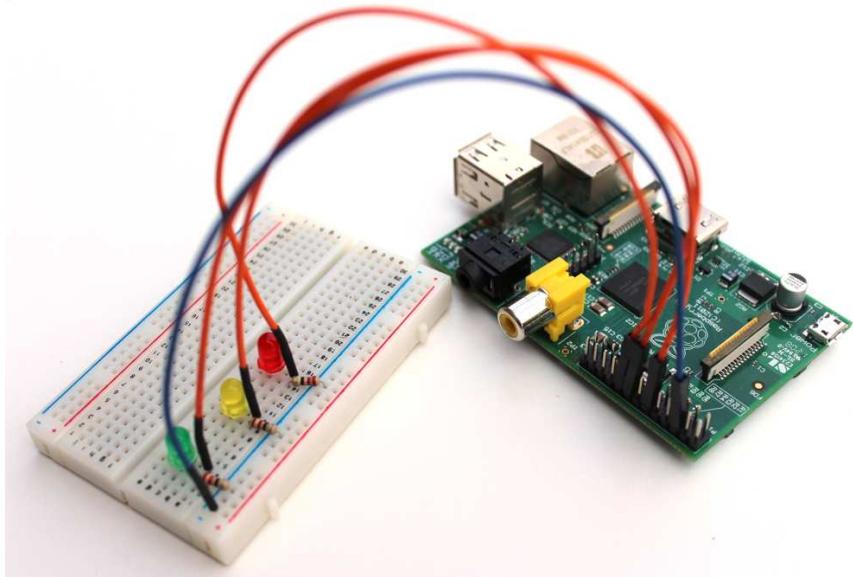
In this section, you'll use three LEDs as status indicators for the Pi's current memory usage. The LEDs will have the same colors as a traffic light. If the Pi's memory is critically low, you'll turn on the red light. If a lot of memory is available, you'll turn on the green light. Otherwise, it will be yellow.

In this image, you see the circuit of the memory alarm.

Ground Pin 12 Pin 16 Pin 18

In the following image, you can see it in real life. It is very similar to the last circuit you built. You have to copy the original LED circuit only two times to control three LEDs.

To make the circuit do something useful, you have to write some software. You could choose from a broad range of programming languages, but thanks to the gpio command, you can implement the project's software with a simple shell script. The following statements define some constants and functions. Also, they perform some initializations.

gpio/memwatch.sh
```
#!/bin/bash
green=18
yellow=23
red=24

init_leds()
{
  for i in $green $yellow $red
  do
    gpio -g mode $i out
    gpio -g write $i 0
  done
}

set_led()
{
  led_status=`gpio -g read $1`
  if [ "$led_status" -ne 1 ]
  then
    init_leds
    gpio -g write $1 1
  fi
}

cleanup()
{
  init_leds
  exit 0
}

init_leds
trap cleanup INT TERM EXIT
```

The first three lines define constants for the GPIO pins you've connected the LEDs to. Note that the numbers in the constants refer to the pin names GPIO18, GPIO23, and GPIO24. They do not refer to the expansion header's pin numbers 12, 16, and 18.

The init_led() function in line 6 sets the mode of all three LEDs to output and turns them off. The set_led() function turns on a certain LED and turns off the other ones. Before turning on the LED, it checks whether the LED is already

on. This makes sure the LED does not flicker in case the current status hasn't changed. Finally, the cleanup() function turns off all LEDs and exits the program.

To initialize the program, you have to call init_leds(). To make sure that the program cleans up before it stops, you can use the trap command in line 32. This command binds the cleanup() method to the most common exit signals, so if you terminate the script, it will call cleanup() before it shuts down.

Now that you have initialized everything properly, you can write the alarm's business logic. Therefore, you need to find a way to determine the Pi's current memory status, and the free command is the perfect solution to this problem.

```
pi@raspberry:~$ free
             total       used       free     shared    buffers     cached
Mem:        190836      40232     150604          0       6252      21068
-/+ buffers/cache:      12912     177924
Swap:            0          0          0
```

To calculate the percentage of memory available, you have to cut out the amount of total memory and the amount of free memory from the free command's output. After that, you have to turn on the right LED depending on the percentage value. Here's how to do this:

gpio/memwatch.sh
```
Line 1  while :
     -  do
     -     total=`free | grep Mem | tr -s ' ' | cut -d ' ' -f 2`
     -     free=`free | grep Mem | tr -s ' ' | cut -d ' ' -f 4`
     5     available=$(( free * 100 / total ))
     -     echo -n "$available% of memory available -> "
     -
     -     if [ "$available" -le 10 ]
     -     then
     10       echo "Critical"
     -         set_led $red
     -     elif [ "$available" -le 30 ]
     -     then
     -         echo "Low"
     15        set_led $yellow
     -     else
     -         echo "OK"
     -         set_led $green
     -     fi
     20    sleep 10
     -  done
```

The memory monitor should permanently check the current memory status so the whole logic runs in an endless loop. Lines 3 to 5 calculate the percentage of memory available by calling the free command two times and cutting

out the relevant information. Note that this solution has a potential flaw. If the current memory usage changes significantly between the two calls to free, you'll get bad results. This is not very likely, but it might happen, so for a production system, you'd have to find a more sophisticated way of determining the current memory usage. For a prototype, it is sufficient.

The following lines compare the memory available to a few threshold values. If the available memory is less than or equal to 10 percent of the total memory, the program turns on the red LED. If it's between 10 and 30 percent, it turns on the yellow LED. Otherwise, the green LED will shine. Finally, the program goes to sleep for ten seconds, so it does not waste a lot of CPU time.

Now type in the shell script or download it by clicking the file name above the code, and copy it to the Pi afterward. The following statement will make the script executable:

```
pi@raspberry:~$ chmod a+x memwatch.sh
```

Then you can start it as follows:

```
pi@raspberry:~$ ./memwatch.sh
78% of memory available -> OK
```

To test the "out of memory" alarm, start the LXDE desktop environment and run the script in a terminal. Then open a few applications and see how the amount of available memory shrinks.

9.6 Display the GPIO Status in a Browser

In Chapter 6, *Networking with the Pi,* on page 49, you've learned how to set up a web server and PHP5 on the Pi. Now you can use this web server to display the current memory usage of your Pi in a web browser. Copy the following PHP program to your Pi's /var/www directory:

```
gpio/memwatch.php
<?php
    function led_is_on($number) {
        $status = trim(@shell_exec("/usr/local/bin/gpio -g read " . $number));
        if ($status == "0") {
            return False;
        } else {
            return True;
        }
    }

    $green = 18;
    $yellow = 23;
```

```php
    $red = 24;

    echo "<h1>Memory Usage is ";
    if (led_is_on($green)) {
        echo "OK";
    }
    elseif (led_is_on($yellow)) {
        echo "Low";
    }
    elseif (led_is_on($red)) {
        echo "Critical";
    }
    else {
        echo "Unknown";
    }
    echo ".</h1>";
?>
```

Then point your browser to the memwatch.php file. If your Pi's IP address is 192.168.2.109, for example, you have to use the URL *http://192.168.2.109/memwatch.php.* Your browser will display a short message explaining the current memory situation on the Pi.

Even if you've never used PHP before, you should be able to understand what the program does. In the led_is_on() function, it calls the gpio command and reads the current status of a GPIO pin. If the pin is currently on, the function returns True; otherwise, it returns False. After that, the program checks which of the LEDs is on and emits a corresponding message.

As you've seen, it's easy to make the status of an electronics device available in your network. Of course, you can make this page more colorful and turn it into a real traffic light display, but this is beyond the scope of this book.

9.7   What If It Doesn't Work?

Building your own electronics devices is not rocket science, but it's not trivial either. If you've never worked with a breadboard, LEDs, and resistors before, a lot of things can and will go wrong. Even if you have a lot of experience, you'll still make mistakes.

If something does not work as expected, don't panic! Usually, the cause of the problem is very simple. The first thing you should check is whether you've connected all parts to the right pins. Then you should check the direction of the LEDs. Also make sure that all parts fit correctly into place. Plugging parts into breadboards can be tricky, especially when the breadboard is new.

Don't forget to plug in the power supply, and do not proceed with your project until you have the simplest version working.

9.8 Next Steps

In this chapter, you learned how to build your own electronics projects and control them with the Raspberry Pi. Even though you used only a few cheap parts, you could actually build something fun and useful.

If you'd like to build more ambitious projects, you should consider buying an extension board for the Pi. The Gertboard[11] and the Adafruit Prototyping Pi Plate,[12] for example, come with some nice features and make prototyping much easier.

---

11. http://www.raspberrypi.org/archives/411
12. http://adafruit.com/products/801

report erratum • discuss

A Linux Primer

The most popular operating system for the Pi is Linux, especially the Debian Linux distribution (Raspbian). If you've exclusively worked with operating systems such as Windows or Mac OS X until now, Linux might produce a little culture shock for you, mainly because on Linux systems the graphical user interface (GUI) is optional. That means you can run a Linux system without using a mouse and without double-clicking colorful icons.

Still, you need a way to interact with the system, and on Linux you have to use a shell for this purpose. A *shell* is a program that awaits your commands via a keyboard and passes them to the operating system (Linux). After Linux has run the command, the shell passes the results back to you.

The shell itself runs in a terminal that goes back to the very beginnings of computing. In these times, you had to connect to the "real" computers using a more or less dumb terminal that basically forwarded your inputs and displayed the computer's outputs. Today you no longer have to use explicit terminal devices, but the metaphor still lives on, and Linux depends on it.

So, whenever you log into a Linux computer, it usually starts a shell in a terminal session. In the shell, you can invoke commands that actually run on the Linux computer.

In modern times, Linux comes with graphical desktop systems that are very similar to Windows and Mac OS X. Still, these desktop systems ship with a terminal emulator you can use to invoke commands directly. The LXDE desktop, for example, comes with a desktop emulator named LXTerminal. You can find a shortcut on the LXDE desktop, so double-click it, and the Pi will start a new terminal session.

A1.1 A First Encounter

After you've logged into the Pi or after you've started a new terminal from the desktop, you'll see the following prompt awaiting your commands:

```
pi@raspberrypi ~ $
```

It does not look like much, but it already gives you a lot of information. For example, the first part (pi@raspberrypi) tells you that the host name of your computer is raspberrypi. It also tells you that your username is pi. This is an important piece of information, because Linux is a multiuser operating system. This means that multiple people can work on the same computer simultaneously (over a network, for example). Also, you can switch to another user account whenever you need to, so it's good to know who you are at the moment.

The next part of the prompt contains the file system path you're currently in. Here it consists only of the tilde character (~), which is an abbreviation for the user's home directory. Every Linux user has a home directory for storing personal data and configuration files. It's similar to the My Documents folder on Windows or the Documents folder on Mac OS X. The dollar character at the end marks the end of the prompt.

To see the content of the current directory, type *ls* and press the Enter key to run the ls command.

```
pi@raspberrypi ~ $ ls
Desktop   python_games
```

The current directory contains two items named Desktop and python_games. From only their names, you cannot tell whether they are regular files or directories. Fortunately, you can control the behavior of most Linux commands with command options. Usually, these options consist of only a single letter preceded by a dash. The ls command supports many options, and when you pass it the -l (short for "long") option, it displays more information about the files in the current directory.

```
pi@raspberrypi ~ $ ls -l
total 8
drwxr-xr-x 2 pi pi 4096 Jul 15 19:36 Desktop
drwxr-xr-x 2 pi pi 4096 Jul 15 19:36 python_games
```

At first it looks a bit scary, but it's really easy to understand. For every item in the directory, ls displays the information shown in Figure 29, *The different components of ls output,* on page 103.
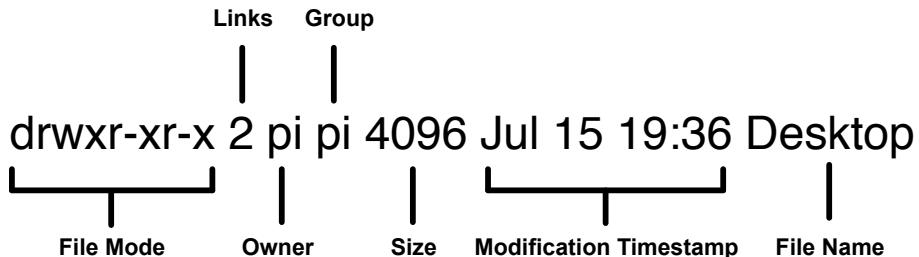
Figure 29—The different components of ls output

The file mode contains the file type and its permissions. If the first character is a dash, the file is a regular file. If it contains a *d*, it is a directory. So, both Desktop and python_games are directories.

The following nine characters encode the file permissions of three groups of people: the owner, the group, and others. The first three characters are rwx, and they mean that the owner of the file is allowed to read (r), write (w), and execute (x) it. In case of a directory, execute means "enter the directory."

Every file on a Linux system belongs to a user and to a group. Groups help to build teams who work together on the same resources. So, for every file, Linux stores permissions for the group, too. In the current case, they are r-x, which means that group members can read and execute the file but are not allowed to change it.

Finally, Linux stores permissions for other users who are not a file's owner and do not belong to the file's group. Again, r-x means that other users might read and execute the file but are not allowed to change it.

The next information ls outputs is the number of links to a file. For your first tour through Linux, you can safely ignore it.

Then you can find the name of the file's owner and of its group. In this case, it's both pi; that is, on the current Linux system, there's both a user named pi and a group named pi.

Next you can find the file size. On Linux, directories are files too, and they simply contain the names of the files stored in the directory. By default Linux allocates some space for this list of files up front, and in case of Debian on the Raspberry Pi, it's 4,096 bytes.

To the right of the file size, you can see the date the file has been modified for the last time. And, finally, ls outputs the file's name.

A1.2 Navigate Through the File System

The pwd (print working directory) command outputs the directory you're currently in.

```
pi@raspberrypi ~ $ pwd
/home/pi
```

As you can see, your home directory (~) expands to the absolute path /home/pi. Linux distinguishes between absolute and relative paths. Absolute paths always begin with a forward slash (/) and reference the same file no matter where you are in the file system. Relative paths on the contrary are relative to your current position in the file system. The following example will clarify the difference between absolute and relative paths.

As you saw in the preceding section, the pi user's home directory contains two directories named Desktop and python_games.

```
pi@raspberrypi ~ $ ls
Desktop   python_games
```

With the cd (change directory) command, you can change the current directory to another one.

```
pi@raspberrypi ~ $ cd Desktop/
pi@raspberrypi ~/Desktop $
```

Now your current working directory has changed. You can see that your prompt has changed, and you can also check it with the pwd command.

```
pi@raspberrypi ~/Desktop $ pwd
/home/pi/Desktop
```

To go back to the pi user's home directory, you have several options. First, you can invoke cd with the absolute path to the home directory.

```
pi@raspberrypi ~/Desktop $ cd /home/pi
pi@raspberrypi ~ $
```

Alternatively, you can use a relative path like this:

```
pi@raspberrypi ~/Desktop $ cd ..
pi@raspberrypi ~ $ pwd
/home/pi
```
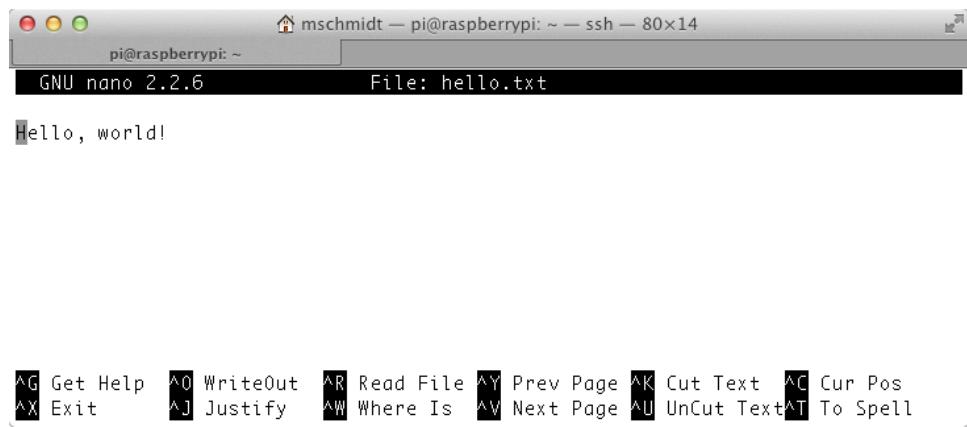
The abbreviation .. stands for the parent directory of the current directory. In the previous command, your current working directory is /home/pi/Desktop. When you run cd .., you change the working directory to the parent directory of Desktop, which is /home/pi.

A1.3 Edit Text Files

Many Linux tools depend on configuration files. Most of these files are regular text files, and you have to edit them from time to time. On Linux, you'll find many powerful text editors for the terminal. If you're used to graphical text editors, most Linux text editors look a bit awkward at first. One of the easiest and most intuitive editors is nano. It permanently displays shortcuts to its most important commands, so you do not have to remember them. The following command starts nano and creates an empty text file named hello.txt:

```
pi@raspberrypi ~ $ nano hello.txt
```

In the following screen capture, you can see how nano looks in your terminal.

You can use most of the screen for editing the text, so type in a few words and move the cursor around with the cursor keys. At the bottom of the screen, you see the most important nano commands. To invoke them, you have to press the Ctrl key and the letter belonging to the command. (The ^ character is an abbreviation for the Ctrl key.) For example, you can exit nano by pressing Ctrl+X.

When you do this, nano does not simply discard your changes and exits. It asks you if you'd like to save your changes (see Figure 30, *Saving a file with the nano text editor,* on page 107). Enter y if you'd like to save your changes and n otherwise. If you've pressed y, you aren't done yet, because nano asks

you to confirm the filename (see Figure 31, *nano always asks you to confirm the filename*, on page 107).

Usually, you'll press Enter only to eventually save the file. At the bottom of the screen, you can see some useful options allowing you to store the file in different formats, for example.

If you're going to work with Linux more often, you should get familiar with one of its text editors. For beginners, nano is an excellent choice, so play around with it for at least a few minutes.

A1.4 Manage Users

Linux is a multiuser operating system—you can work with several different users on the same computer at the same time. In this book, you'll mainly use the user pi, because it comes with the Raspbian image automatically. This is convenient, but sometimes it's handy to create different users for different tasks. Also, the user pi is a very powerful user who has full administrative rights and can change nearly every aspect of the system. You do not want to grant all privileges to all users. It's always a good idea to work with only the administrative rights you need to get the job done. That way, you cannot harm the system by accident.

Adding a new user to Linux is easy using the adduser command.

```
pi@raspberrypi ~ $ sudo adduser maik
Adding user `maik' ...
Adding new group `maik' (1002) ...
Adding new user `maik' (1002) with group `maik' ...
Creating home directory `/home/maik' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for maik
Enter the new value, or press ENTER for the default
	Full Name []: Maik Schmidt
	Room Number []:
	Work Phone []:
	Home Phone []:
	Other []:
Is the information correct? [Y/n] Y
```

You have to provide only a username (by convention it should contain only lowercase letters), a password, and a few optional attributes such as your full name. After you've confirmed that all information is correct, Linux creates
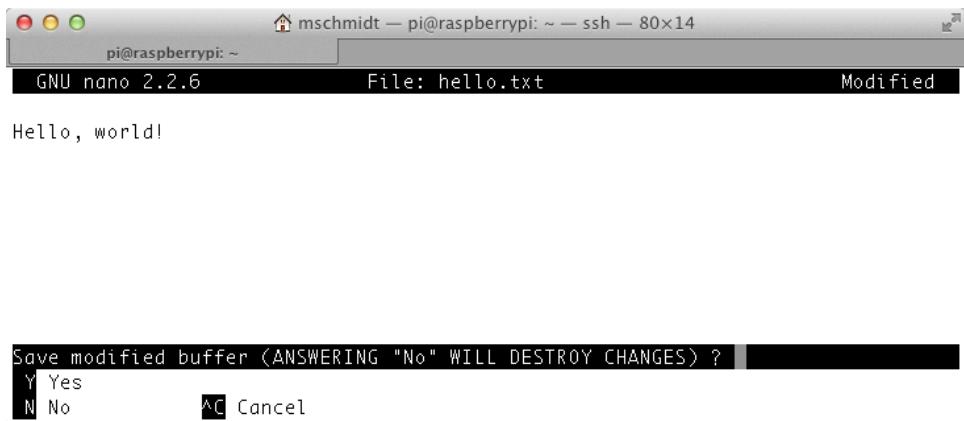
Figure 30—Saving a file with the nano text editor
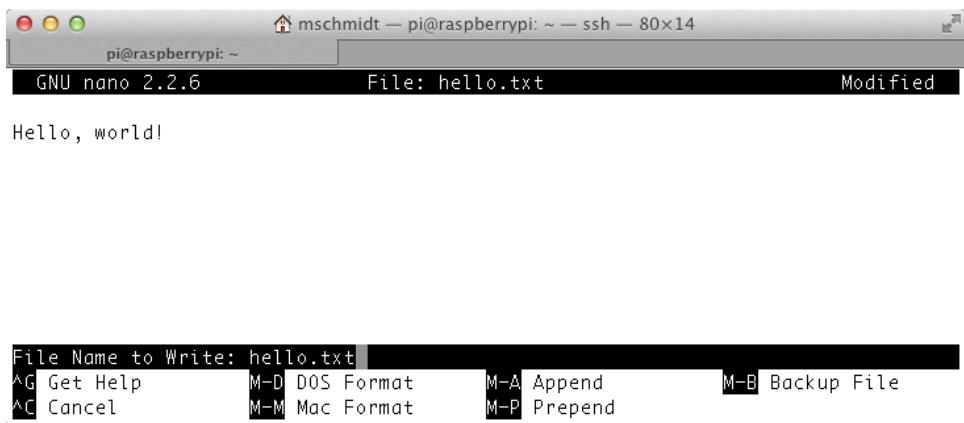
Figure 31—nano always asks you to confirm the filename.

a new user with its own home directory. The next time you boot the Pi, you can use it to log into the system. If you're impatient, you can use the su (substitute user identity) command to switch to the new user.

```
pi@raspberrypi ~ $ su - maik
Password:
maik@raspberrypi ~ $ pwd
/home/maik
maik@raspberrypi ~ $ startx
```

su asks for the user's password, and if it's correct, it switches to the new user. The pwd command prints the current working directory; in this case, it's the home directory of the newly created user. If you start the LXDE desktop with

the startx command, it greets you with the standard LXDE background image
(see Figure 32, *The default look of LXDE,* on page 109), because in contrast to
the pi user, the new user starts with the desktop's defaults.

When working with the pi user, you've often used sudo to run commands with
administrative privileges. See what happens if you try to delete a file that you
do not own with the rm (remove file) command.

```
maik@raspberrypi ~ $ sudo rm /boot/config.txt

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for maik:
maik is not in the sudoers file.  This incident will be reported.
```

The command prints a warning and then asks for your password. Obviously,
the new user is not allowed to delete files in the /boot directory, so Linux
refuses to invoke the rm command.

While it's a good default behavior to deny new users access to dangerous
operations, sometimes users need more privileges. If you want to give your
new users the same rights as the pi user, you have to add the user to the
sudoers file. This file contains a list of all users who are allowed to run the sudo
command, and it also specifies which operations the users are allowed to
perform. You cannot edit the sudoers file directly; you have to use the visudo
command, which invokes the text editor vi by default. If you want to edit the
file using a different text editor such as nano, you have to specify it on the
command line (make sure you're using the pi user again).

```
pi@raspberrypi ~ $ sudo EDITOR=nano visudo
```

This opens the /etc/sudoers file using the nano text editor. In the file, you'll find
a section that looks like this:

```
# User privilege specification
root  ALL=(ALL) ALL
suse  ALL=(ALL) ALL
pi    ALL=(ALL) ALL
```

Add a new line that looks exactly like one of the previous three lines, but
replace the username with the name of your new user. If you're using nano to
edit the file, press Ctrl+X and confirm that you would like to save the changes.

Figure 32—The default look of LXDE

Then confirm the filename, and you're done—your new user now has the same rights as the original pi user.

In case you no longer need a certain user, it's reasonable to delete it.

```
pi@raspberrypi ~ $ sudo userdel maik
```

The previous command will delete only the user's account but not the user's files. The user can no longer log into the system, but all the files he or she has created in the home directory are still available. If you want to delete the files as well, run the following:

```
pi@raspberrypi ~ $ sudo userdel -r maik
```

If you ever need to change a user's attributes, such as their home directory, you can use the usermod command. You can use it to lock or unlock accounts, for example.

```
pi@raspberrypi ~ $ sudo usermod -L maik
```

This will lock the account of the user named maik. The user can no longer log into the system. To unlock the account, run the following command:

```
pi@raspberrypi ~ $ sudo usermod -U maik
```

You can read usermod's manual page (and the manual page of every other Linux command) using the man command.

```
pi@raspberrypi ~ $ man usermod
```

This displays the command's manual pages. You can stop the man command by pressing Q.

One important action is changing a user's password. For this, you can use the passwd command.

```
pi@raspberrypi ~ $ passwd
Changing password for maik.
Old Password:
New Password:
Retype New Password:
```

passwd asks for your current password and then for your new password. If everything is OK, it prints no message, and your user has a new password.

A1.5 Manage Processes

Whenever you run a command or an application on a Linux system, the operating system's kernel spawns a new process. You can list your current processes with the ps command.

```
pi@raspberrypi ~ $ ps
 PID TTY          TIME CMD
1880 pts/2    00:00:00 bash
1892 pts/2    00:00:00 ps
```

At the moment, you own only two processes. The first has the process ID (PID) 1880, and it belongs to a command named bash (the process IDs on your system will vary). This is the process that belongs to the shell you're currently working in. The process with the PID 1892 belongs to the ps command you have used to list your current processes. At the moment, you see the output of the ps command; process 1892 will be gone already. To see the effect, run ps again.

```
pi@raspberrypi ~ $ ps
 PID TTY          TIME CMD
1880 pts/2    00:00:00 bash
1894 pts/2    00:00:00 ps
```

As you can see, the shell still has the PID 1880, but your latest call to ps was handled by a new process with PID 1894.

You can get more information about your processes with the -f option.

```
pi@raspberrypi ~ $ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
pi        1880  1879  0 12:51 pts/2    00:00:00 -bash
pi        1895  1880  0 12:58 pts/2    00:00:00 ps -f
```

Now you can see the user ID (UID) of the user who has spawned a certain process. Unsurprisingly, the UID is pi for all your processes. In addition to

the PID, you can see the parent process ID (PPID). This is the ID of the process that has created another process. For example, the ps -f command you've run before has the PPID 1880. This is the PID of the shell you're using. So, the shell is the parent of the process created by the ps -f command.

To see all information about all processes currently running on your Pi, run the following command:

```
pi@raspberrypi ~ $ ps -ef
```

This will output a fairly long list of processes that contains every single Linux service your Pi has started.

Getting a list of all active processes is useful, but most often you'll be looking for a certain process for a reason. Perhaps the process uses too many resources or takes too long, and you'd like to terminate it. But how can you terminate a process?

Terminating a long-running process is easy when you start it directly from the shell. To demonstrate, the following command searches for all text files on your SD card, so it will take a long time until it's finished:

```
pi@raspberrypi ~ $ find / -name '*.txt'
```

While the process is running, you can terminate it by pressing Ctrl+C on your keyboard. When you press Ctrl+C, the shell recognizes your keypress and sends a signal to the process that is currently running. Signals are small messages that all processes listen for in the background. Pressing Ctrl+C generates a signal named SIGINT that tells a process it got interrupted. Whenever a process receives a SIGINT signal, it usually cleans up and terminates.

For processes that are still running in your terminal, pressing Ctrl+C is a good option, but what if you need to terminate a process that is running in the background? For example, most Linux services run in the background by default, and you do not start them yourself. In this case, you have to find out the PID of the process and pass it to the kill command.

```
pi@raspberrypi ~ $ kill 4711
```

The previous command sends the SIGTERM command to the process with the ID 4711. You can send other signals with the kill command, too. For example, the following command will terminate the process with the PID 4711 in any case:

```
pi@raspberrypi ~ $ kill -KILL 4711
```

Of course, you need to have the permission to terminate a process. Usually you are allowed only to kill your processes.

A1.6 Shut Down and Reboot the Pi

When you're done with your work, you should not simply switch off the Pi. It might result in the loss of data. Always shut it down with the following command:

```
pi@raspberrypi ~ $ sudo shutdown -h now
```

If you need to reboot the Pi, use the following command:

```
pi@raspberrypi ~ $ sudo reboot
```

A1.7 Getting Help

Since its beginnings, the Unix/Linux operating systems came with a great manual system named *man pages*. Whenever you need to look up the options of a certain command, you can display its manual with the man command. To look up all options of the ls command, for example, run the following command:

```
pi@raspberrypi ~ $ man ls
```

To scroll down a line, press the cursor-down key. Press cursor-up to scroll up a line. To scroll down a page, press the spacebar. Press Ctrl+B to scroll up a page. To leave the program, press Q.

The man command itself has many more options. To learn more about it, run the following:

```
pi@raspberrypi ~ $ man man
```

The Pragmatic Bookshelf

The Pragmatic Bookshelf features books written by developers for developers. The titles continue the well-known Pragmatic Programmer style and continue to garner awards and rave reviews. As development gets more and more difficult, the Pragmatic Programmers will be there with more titles and products to help you stay on top of your game.

Visit Us Online

This Book's Home Page
http://pragprog.com/titles/msraspi
Source code from this book, errata, and other resources. Come give us feedback, too!

Register for Updates
http://pragprog.com/updates
Be notified when updates and new books become available.

Join the Community
http://pragprog.com/community
Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

New and Noteworthy
http://pragprog.com/news
Check out the latest pragmatic developments, new titles and other offerings.

Buy the Book

If you liked this eBook, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: *http://pragprog.com/titles/msraspi*

Contact Us

Online Orders:	*http://pragprog.com/catalog*
Customer Service:	*support@pragprog.com*
International Rights:	*translations@pragprog.com*
Academic Use:	*academic@pragprog.com*
Write for Us:	*http://pragprog.com/write-for-us*
Or Call:	+1 800-699-7764