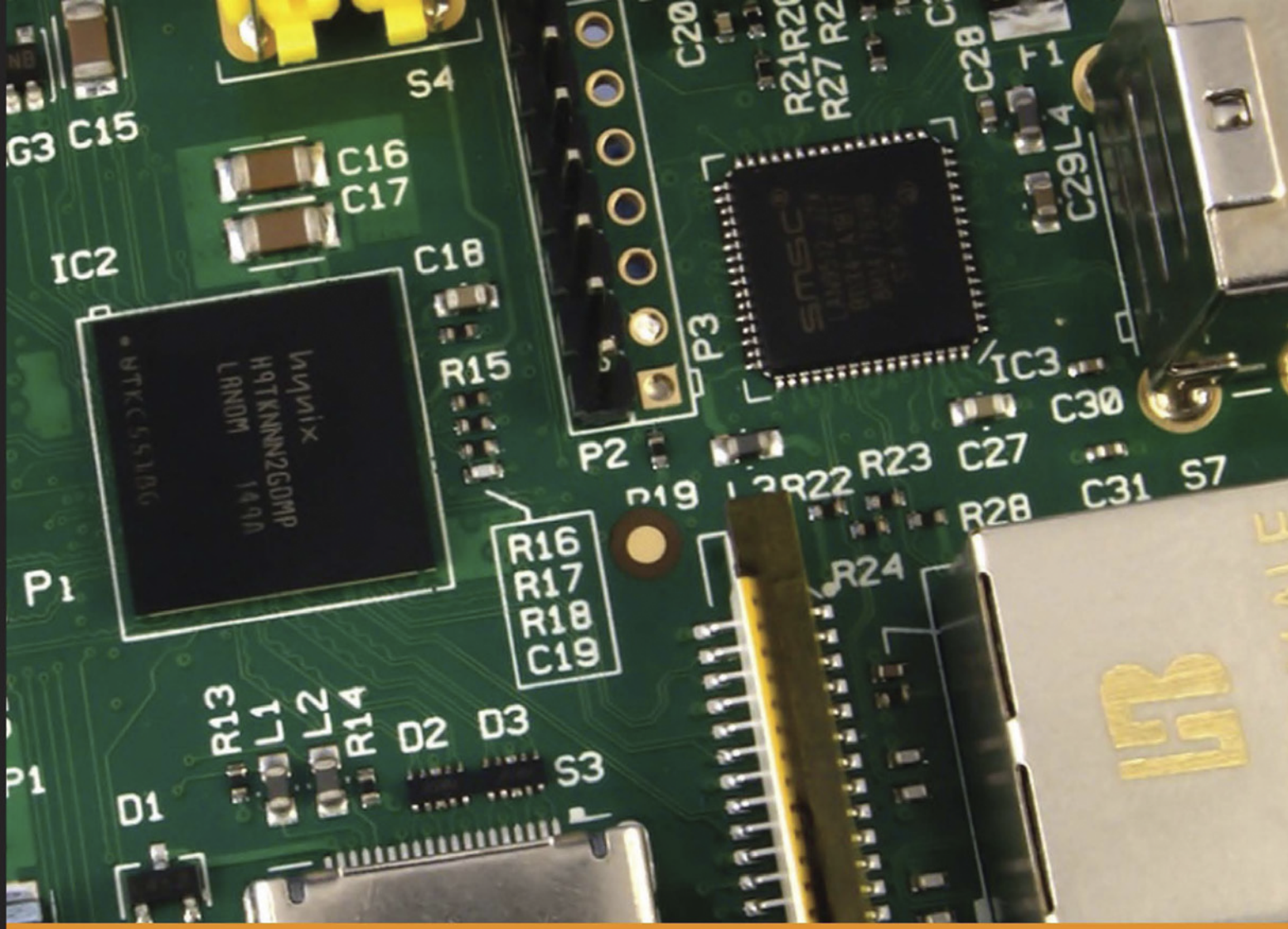


Community Experience Distilled

Raspberry Pi Server Essentials

Transform your Raspberry Pi into a server for hosting websites, games, or even your Bitcoin network

Piotr J. Kula

[PACKT] open source* PUBLISHING
community experience distilled

Raspberry Pi Server Essentials

Transform your Raspberry Pi into a server for hosting websites, games, or even your Bitcoin network

Piotr J. Kula

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

Raspberry Pi Server Essentials

First published: February 2014

Production Reference: 1030214

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78328-469-6

www.packtpub.com

Cover Image by Piotr J. Kula (info@piotrkula.com)

Credits

Author
Piotr J. Kula

Reviewers
Teemu Lätti
Warren Myers

Acquisition Editor
Kunal Parikh

Commissioning Editor
Manasi Pandire

Technical Editors
Shubhangi Dhamgaye
Nachiket Vartak

Copy Editors
Alisha Aranha
Roshni Banerjee
Gladson Monteiro
Adithi Shetty

Project Coordinator
Sageer Parkar

Proofreaders
Maria Gould
Paul Hindle

Indexers
Monica Ajmera Mehta
Tejal Soni

Graphics
Ronak Dhruv

Production Coordinator
Shantanu Zagade

Cover Work
Shantanu Zagade

About the Author

Piotr J. Kula developed an interest in computers when he was six years old. He was introduced to the world of technology by his father who came from an electronics engineering background. Piotr has lived, studied, and gained experience in three countries. Today, he is a Microsoft Certified Professional and works with reputed companies offering complex software solutions. In his spare time, he enjoys working on electrical engineering projects and also enjoys doing some home improvement projects with his wife.

I want to thank my wife Katarzyna Kula for always supporting me during my projects.

About the Reviewers

Teemu Lätti works as a software specialist for Elektrobit (`http://elektrobit.com`) in Kajaani, Finland. He has over 15 years of experience as a professional Java and C++ developer. He is specialized in embedded software on different platforms, for example, Raspberry Pi, Android, and Windows Phone. He has a wide experience in various software, from device drivers to user interfaces and web development. He hosts a private web page (`http://cupla.net`) and builds home automation experiments with Raspberry Pi and Arduino.

Warren Myers is a Data Center Automation and Management Engineer with seven years of experience with the HP automation stack. He has an extensive background and interest in technical arenas. He started programming when he was 10, and has always strived to learn new things on a regular basis. He currently works for Avnet Services as a Solutions Architect in the Cloud & Automation Practice. He has also written *Debugging and Supporting Software Systems* (`http://cnx.org/content/col11350`), a freely available e-book.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface

The purpose of this book is to get you started with the Raspberry Pi. We will try and cover many different topics to demonstrate the flexibility of the Raspberry Pi. The main goal of the book is to get you started on this project that you were just not so sure about.

What this book covers

Chapter 1, *Getting Started with Raspberry Pi*, will check the basic equipment that we need to use with this book. We will have to look into the other peripherals that we buy and see how the Raspberry works. We then will be seeing how to flash the newest Raspbian image to our SD card.

Chapter 2, *Preparing the Network*, will illustrate how to set up LAN and a wireless connection to our network and connect to the Internet. We will set some network benchmarks and understand some limitations. We will also look into Dynamic DNS hosting.

Chapter 3, *Configuring Extra Features*, will illustrate how to update the software and the firmware of the Raspberry Pi. We will learn about the watchdog and how to buy extra decoder licenses.

Chapter 4, *Using a Fast PHP Web Server and Database*, will illustrate how to set up a fast web server using nginx with PHP, and decide if we want to use MySQL or SQLite.

Chapter 5, *Setting Up a File Server*, will illustrate how to attach a USB storage medium and how to format it. We will not only look into various ways of sending data to the Raspberry Pi, but also how to share media on the network. As an extra task, we will look into creating the hardware RAID!

Chapter 6, *Setting Up the Game Servers*, will explore open source game engines that are available on the repository. We will also have a sneak peek into the Jessie repository that is in beta testing.

Chapter 7, *Bitcoins – Pools and Mining*, will illustrate the concept of cryptographic currencies, and we will try and understand what Bitcoin is. We will set up a wallet and learn how to send and receive coins. We will also venture into mining equipment and learn about mining pools.

Chapter 8, *Streaming Live HD Video*, will explore the camera module and decide if we should use FFmpeg or grstreamer. We will compile a custom version of Nginx and start streaming videos to the World Wide Web!

Chapter 9, *Setting Up a Media Center*, will illustrate how to finally attach an HD monitor and play some videos that we recorded or downloaded earlier. We will also look into RaspBMC and the benefits of running it as a dedicated media player.

What you need for this book

You need at least a Raspberry Pi model B, recommended 8 GB (4 GB minimum) SD card, 1 Amp micro USB power supply, and a network cable.

Who this book is for

This book is mainly targeted at novices with a keen sense of hacking and learning things. Almost the entire book is explained using the power of command lines over SSH.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "The `kernel.img` connects the application to the hardware."

A block of code is set as follows:

```
#! /bin/sh
# /etc/init.d/btsync
#

# Carry out specific functions when asked to by the system
case "$1" in
start)
    /home/root/.btsync/btsync
    ;;
stop)
    killall btsync
    ;;
*)
    echo "Usage: /etc/init.d/btsync {start|stop}"
    exit 1
    ;;
esac

exit 0
```

Any command-line input or output is written as follows:

```
mkdir ~/.btsync && cd ~/.btsync

wget http://btsync.s3-website-us-east-1.amazonaws.com/btsync_arm.tar.gz

tar -zxvf btsync_arm.tar.gz
./btsync
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "**Password**: The password of that user."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Getting Started with Raspberry Pi

Connecting to a network should be as easy as plugging in a cable. The question is, what can we do on the Raspberry Pi after we are connected to the Internet or local network? This is why it is essential to learn about the hardware prerequisites and capabilities of the Raspberry Pi, so that your idea is theoretically possible to accomplish. Also, knowing your hardware will make troubleshooting problems much easier later in the book.

The most common problems are related to power. These problems can cause the Raspberry Pi to restart, or may show up as a rainbow screen during the boot process (if you have an external monitor connected).

This chapter is all about identifying your Raspberry Pi and the peripherals that you are using or may want to use with it. There are three main pieces of information you should know about your Raspberry Pi: Model, PCB Revision, and Revision.

From here on, I will refer to the Raspberry Pi as "Pi" for simplicity.

Hardware requirements

This book will assume that you are using Model B with at least PCB Version 1 and Revision 3. Model B has 512 megabytes of RAM and a built-in LAN port. You also need an SD card of at least 4 GB, but 8 GB is recommended.

The Pi, at the most basic level, needs only a power supply and an SD card to run; but to configure, it is recommended to have an HDMI cable, a compatible screen or television, and a USB keyboard. Even though you might have bought a Pi as a prepackaged kit, it does not always mean that the supplier has chosen the correct peripherals to go with it. These peripherals are important to achieve optimal performance and maximize its lifetime.

The most common power supply is a 1 AMP power supply, which is commonly supplied with smart phones. These chargers are made from good quality components and can easily handle the stress of additional power or power spikes. You should also pay attention to the USB cable that you are using. Some cables are cheaply produced and the copper wire inside them is very thin, and so they struggle to deliver a full 1 AMP current when needed. The original cable provided with the 1 AMP power supply works best. So, try to avoid buying cheap cables.

When you purchase a powered USB hub, they are usually supplied with a 2 AMP power supply. This is enough to power USB devices such as a Wi-Fi adapter, a USB hard drive, a few other peripherals, and even the Pi itself. For basic usage, a computer's USB port will suffice, but you may experience problems. So, it is recommended to avoid using these USBs as a power source.

SD cards may all look alike, but the actual controllers and memory chips vary in speed. The most trusted memory cards are genuine memory cards. But be careful, as the market is flooded with fake brands, and they usually use the slowest and cheapest components. Some even have the incorrect size. There are various speed classes, where Class 4 is the slowest and should be used minimally with the Pi class, and Class 10 is the fastest. The speed of the SD card should not be treated as the main performance gauge. Instead, we should use external storage devices such as hard drives, USB memory sticks, or **Network Attached Storage** (**NAS**) to do intensive storage operations. This book assumes that you are using at least an 8 GB, Class 4, genuine SD card. Tweaking performance requires a lot of time and is not covered in this book. The following figure shows an original HTC charger and cable, together with an original Kingston 8 GB Class4 SD card that was used during the writing of this book:

Extra peripherals

The Pi is branded as a computer, and it is expected that we can connect various different devices to it. Raspbian is based on Debian. A majority of drivers are available within Raspbian and are getting larger with each new update. You might have some old USB peripherals lying around, for example, a joystick. If you can find a driver for any other Linux platform, it should be possible to make it work with Raspbian. Plug it in, use the `lsusb` command line, and check to see if it has been detected. If you manage to get it working with your knowledge, you should share your knowledge on a forum for other users; but, the process is not covered in this book.

Essential peripherals

You should consider buying these peripherals and dedicating them to your Pi. They will really make it easier to set everything up and are even used for long term.

Wireless USB network adapters

As of the time of writing this book, the current Raspbian image supports a variety of wireless adapters—without the need to install any extra drivers. Many of the mini, nano, or micro versions run directly off the Pi's USB ports and do not require a power USB hub.

USB hubs

Because the Pi is limited to two USB ports, it might be wise to have a compatible, powered USB hub. Powered being the key word here, as this will allow you to plug in any USB device or several devices at the same time without affecting the Pi's power stability. As of this writing, Raspbian is not fully compatible with USB 3.0 hubs yet.

Keyboards and mice

Most wired keyboards and mice will run directly off the Pi (since Revision 3). Many Bluetooth keyboards and mice also work directly off the Pi's USB ports, but require initial setup using a wired keyboard. Some wireless keyboards, such as the Microsoft 3000 series, do not need any configuration as they emulate a wired keyboard and can be used during boot time.

Useful peripherals

As you grow more familiar with your Pi, you will think up much bigger ideas. With such ideas, you might need a few more useful devices to help you out.

Internet 3G dongles

You can connect to the worldwide network by using a 3G dongle. These require a lot of power, and they need to run from a powered hub to operate at full speed. But, they are a really easy way to connect your Pi to the Internet, even in the most remote places of your country. As long as you have the basic voice signal, you should always be able to use GPRS (single channel 57.5 KBps or dual channel 115 KBps). This can be enough to send plenty of logging data as text. Some countries offer free text messages, and these can also be used to send and receive a minimum amount of data. If you plan to run a server, it would be recommended to use LAN or Wi-Fi connected to an ADSL/DSL connection instead.

Sound cards

The Pi has its own sound hardware, which is really good at giving you high definition sound over HDMI or analog sound via its RCA connection (you cannot use both). You might find yourself in a situation where you would like to record audio from a line input or a microphone; you could then use any USB 1.1 or USB 2.0 device.

IR receivers

Infrared (IR) receivers are a great way to control your Pi using conventional remote controls. The FLIRC USB IR remote dongle is a great way for you to start doing this.

TV and radio receivers

This is the ultimate way to turn your Pi into a full DVR system. You can record, playback, or pause live TV from HD satellite or Digital TV. You can also listen to your favorite radio channels.

Webcams

The Pi has a port for its own dedicated HD camera module. Owning one of these cameras is a real treat, but they do not have proper V4L (Video for Linux) support like most USB cameras. This means that doing some easy tasks can be much more complicated. There is a chapter dedicated to this later in the book.

Multicard readers

These come in handy if you work with various types of cards. There is limited support on the generic types; but, the USB 3.0 USRobotics all-in-one card works really well, and you can mount all six cards at the same time.

Alamode

WyoLum is a startup business that creates useful add-ons for various applications. Specifically, the AlaMode is an Arduino-compatible board with a real-time clock and microSD slot that sits on top of the Pi. You can communicate with Arduino using the Pi's dedicated Universal Asynchronous Receiver/Transmitter (UART), and it can run on the Pi's power source. If you like electronic projects and are already familiar with Arduino, this is worth looking at. You can even use it to flash other Arduino compatible chips or upload firmware to run it on its own!

HDMI to VGA

Though you may not like to downgrade from HDMI to VGA, you can purchase inline HDMI to VGA converters from your favorite online auction shops or electronic stores. You must make sure that you buy an ACTIVE converter, which is slightly more expensive than the passive converter. Active means that it contains a microcontroller that uses power from the HDMI port to convert the digital signal into the VGA standard. The Raspberry Pi is capable of powering these types of devices.

Remember that this device contributes to the overall power consumption of the Pi. It would be recommended to use a 1 AMP power supply for such accessories.

Fun peripherals

You might have some of these lying around in your gadgets box. Hopefully, reading about some of these less-used devices might spark some creative ideas.

Joysticks

Microsoft's Xbox 360 controller works as a mouse in X using the Arch Linux distribution. Other joysticks might need a ported driver that can be found on Internet forums.

USB to SATA

You can purchase simple USB to SATA controllers that allow you to attach SATA hard drives by using the dedicated power supplies. The real fun begins when you use hardware RAID-based USB to SATA controllers that can be chained in various configurations. This can give you massive storage, high redundancy, and maximum performance. Be careful though, as the maximum throughput speed you can achieve is governed by the bandwidth of USB 2.0. In theory, this is a maximum speed of 60 MBps; but it is shared by all the devices on the controller and not per port. There is more information on this later in the book.

CAN bus

The CAN bus is the standard used in all modern motor vehicles. It is a standard port that gives mandatory data that can be interpreted by anybody. For example, throttle value, misfiring of cylinders, or air to fuel ratio. PEAK-System has a variety of peripherals and software that are compatible with the Pi. If you have access to the manufacturer-specific codes, you can even adjust engine mappings with these tools.

Adjusting some values may damage your ECU and will void any warranties.

Home automation

A compatible device called TellStick runs well as a third-party home automation device for the Pi. But as an advanced Linux user, you should strive to make your own applications; the best add-on for this is the AlaMode from WyoLum.

USB missile launcher

Available from a variety of websites and stores, this is the perfect gadget if you need to shoot plushy missiles at unidentified objects! This is purely an entertainment peripheral, but you could use it for DIY projects.

Fingerprint scanners

Futronic's fingerprint scanners work well with Raspbian, and there are many examples that can be found online. They are standalone programmable devices that communicate with the Pi using simple messages over USB-UART, and have extensive documentation available with the device.

Installing Raspbian on the Raspberry Pi

There are many distributions that can run on the Pi. Some are specific real-time operating systems like RISC OS, or mainstream operating systems such as Google's Android. Also, the Raspberry foundation has an image called New Out Of Box Software (NOOBS) that contains four different distributions: Raspbian, PiDora, and two flavors of XBMC.

In this book, we will be using Raspbian. It is supported by the foundation, has the best compatibility, and is easy to use. Raspbian is based on Debian, and it is similar to many other Linux operating systems.

The following steps will show you how to install Raspbian on the Raspberry Pi:

1. For Windows and Macintosh users, it is recommended by the Raspberry Pi foundation to use the SD Formatter from `http://www.sdcard.com`.

   For Windows, perform the following steps:

   1. Install the SD card formatting tool.
   2. In the **Option** menu, set the **Format size adjustment** option to **ON**.
   3. Make sure that you select the correct SD card.
   4. Click on the **Format** button.

   For Macintosh, perform the following steps:

   1. Install the SD card formatting tool.
   2. Select **Overwrite Format**.
   3. Make sure that you select the correct SD card.
   4. Click on the **Format** button.

   For Linux, perform the following steps:

   1. It is recommended to use the GParted or Parted tool in Linux.
   2. Format the entire disk as FAT.

2. You should download the latest NOOBS archive from `http://www.raspbian.org/`.
3. Unzip the archive.
4. Copy the extracted files on to the formatted SD card.
5. Insert the SD card into the Pi, plug in your HDMI or other video cable with a compatible keyboard, and power it up.

6. The Pi will boot up and present a list of operating systems. Select **Raspbian**.
7. If your display is blank, try to press the numeric keys given in the following list when the Pi is booted up:
    - [1] - HDMI Mode
    - [2] - HDMI Safe mode
    - [3] - Composite PAL
    - [4] - Composite NTSC

Understanding the design of the Raspberry Pi

The Pi has two identifiable microchips on the PCB, which are described as follows:

- In the center, the one clearly marked as BCM2835 is the main processor
- Near the USB port, there is a smaller chip that is either a USB hub or USB/LAN chip, depending on the model

The BCM2835 is actually a high performance OpenGL ES GPU (VideoCore IV) with a built-in 700 MHz ARM6 processor by its side. It is a System on Chip (SoC), which means that there is a small amount of space for code that executes when it gets turned on. This is known as Stage 1 in the boot process.

Boot process

Some network actions need to be done during the boot process, and it is good to understand the various stages in case you need to troubleshoot something. The boot process is as follows:

1. Stage 1 begins on the GPU and executes the code SoC firmware, which starts to load the Stage 2 code to the L2 cache.
2. Stage 2 reads `bootcode.bin` from the SD card. It initializes Synchronous Dynamic Random Access Memory (SDRAM) and loads Stage 3.
3. Stage 3 is the `loader.bin` file. This loads `start.elf`, which starts the GPU.
4. During `start.elf`, it prepares to load `kernel.img`.

5. Then, the kernel image reads `config.txt`, `cmdline.txt`, and `bcm2835.dtb`.
6. If the `.dtb` file exists, it is loaded at 0 × 100 and the kernel is loaded at 0 × 8000 in memory.
7. The kernel image is the first binary that runs on the ARM CPU, and it can be compiled with the custom support for a specific hardware.
8. The operating system starts to load.

All the source code in Stages 1 to 3 are closed source and protected by Broadcom. These closed source files are compiled and released by Broadcom only. You can update them on your SD card by running a firmware upgrade in Raspbian, which is covered later.

The `kernel.img` file connects the application to the hardware. Any computer with an operating system has a kernel of some sort. It is possible to compile your own kernel in Linux, and it might be the first file that you might want to amend yourself. This allows you to change the boot screen, load custom drivers, or perform other tasks that you might need. This is an advanced task and is not covered in this book.

Other capabilities

The BCM2835 processor also has dedicated audio hardware together with audio and video encoding/decoding. This allows the Pi to playback HD (MPEG-4) content such as videos or games. You can buy additional encoder/decoder licenses for extra functionality like MPEG-2 used in DVD video encoding and VC-1 that is used by Microsoft's WMV formats. The SD card is also directly interfaced by the Broadcom chip using the dedicated hardware inputs/outputs and interrupts.

All that dedicated hardware means that while some sections of the chip are fully utilized, the ARM CPU will be idle or hardly used. This will allow you to compute other transactions synchronously, and this is what makes the Raspberry Pi truly a unique, single board, credit-card-sized computer!

Hardware limitations

All this hardware crammed into one tiny space has its drawbacks. Some are deliberate and others are not. You should consider that these are theoretical calculations and that the real-world performance may vary. Usually, it is slower than the theoretical estimation.

Network speeds

It may be disappointing that the Raspberry Pi foundation decided to use a 100 MBps LAN chip instead of a gigabit one. But, we need to crunch some numbers to justify this decision. Let's convert megabits to more familiar megabytes. To get to "megabytes per second" from "megabits per second", we divide 100 MBps by 8 (there are 8 bits in a byte). This equates to 12.5 megabytes per second at 100 percent LAN capacity. For a single user, this is only roughly 20 percent of what the USB hub can handle. That means that, by design, this is an unchangeable bandwidth limitation for networking.

If you plan to share files with several users at the same time, each new user will bump down the other user's bandwidth to accommodate their own. As a work-around, you could add a gigabit USB LAN peripheral to increase the bandwidth. But, due to the speed constraints of the USB hub, you will only use approximately 48 percent of the gigabit LAN. To make matters worse, the hard drives running on the USB port will start to fight for bandwidth. The USB Controller has to share 480 MBps across all the four ports! One port is used by the 100 MBit network card, and the other connects the hub to the GPU. For one user, this means a maximum bandwidth of 240 MBps. Why 240 MBps? This is because 240 MBps goes to LAN and 240 MBps goes to the hard drive; and theoretically, there is no USB bandwidth left for anything else.

This can be a problem for a multiuser environment, but for home use, you would not run into any major problems as the bandwidth can accommodate HD video streams while serving other clients. This is why the cheaper 100 MBit version was used.

USB bottlenecks

As it was made clear by the bottlenecks found with LAN, the worst thing about USB bottlenecks is that there is no way to work around this problem! This is because the USB Controller connects to the Broadcom chip and the LAN chip on the PCB, without any possibility of expanding or replacing this chip.

Time

The Pi does not come with a real-time clock, so timekeeping is left to Internet-based time servers. For most people, this might not cause a problem. But if you want to create a remote, disconnected device that depends on events that are recorded at various times of the day, you might be left a little bewildered.

One easy and reliable way to do this is to connect a USB or I2C RTC that runs on a small battery. There is an easier and free option though, but it is not as accurate. You may want to install the `fake-hwclock` package. All you need to do is set the time once and the software will keep track of time by using a file. If you have a power outage, the software will read the file and set the time back to the last known time. The drawback is that you lose that time as there is no way to determine how long the outage lasted.

Summary

One of the most popular questions found on the Internet is how to increase the Pi's performance! This is because the Pi has been labeled as a small computer which directly associates with modern day desktop computers. This makes a lot of people think that the Pi will perform as efficiently as a full, multi-core computer.

The purpose of understanding the architecture is vital to a successful, long term project. The Pi works "like" any other computer, but it was designed purely for experimental and learning purposes. It should not be used in production environments, but it is an extremely attractive alternative.

It is an excellent platform for sharing media between friends at school. It is fantastic for streaming HD media on your TV, and it is robust enough for some standalone applications.

2
Preparing the Network

It is important to learn how your network works, especially if you plan to connect your **Raspberry Pi** to the Internet. A home user like yourself will typically use an Internet package designed for browsing websites and reading e-mails. Business packages on the other hand allow you to do a lot more than you might think. These two different ISP packages usually carry important technical differences that decide how your network can be reached.

In this chapter, we will learn how to connect to the Internet and look at how to solve some common problems for home users. We will also learn how to benchmark your network and try to isolate any network-related issues.

Local Area Network (LAN)

Using the standard Raspbian package, the essential drivers are included; most of the other drivers are also preloaded.

We will start by plugging in a network cable between the Pi and the router provided by your ISP. By default, this router has a DHCP server that automatically assigns an IP address to your Pi.

You may also use network switches to make a more complex network, but because the Pi has a 100 MB network port, it may downgrade the entire network to 100 MBs. Some switches can negotiate separate connections for lower-speed interfaces without downgrading the entire network, but you need to refer to the specifications of the device.

To check whether your LAN network is up and running, just type `ifconfig`, and you will get text containing your current settings:

```
eth0      Link encap:Ethernet  HWaddr b8:27:eb:45:bb:fa
          inet addr:192.168.1.135  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::ba27:ebff:fe45:bbfa/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:89 errors:0 dropped:0 overruns:0 frame:0
          TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7943 (7.7 KiB)  TX bytes:14104 (13.7 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- `HWaddr`: This is your Pi's MAC address that identifies the vendor and should provide a globally unique address.
- `inet addr`: This is your current LAN IP address that belongs to a private range of `10.x.y.z` (Class A), `172.16.y.x` (Class B), or `192.168.x.x` (Class C). They are known as private addresses because they do not exist on the Internet.
- `Bcast`: This is a reserved address calculated by your network mask and transmits global messages within your private network.
- `Mask`: This is used in advanced networks to divide them into subnetworks, and it depends on the class your network belongs to.
- `inet6 addr`: This is your IPv6 address. It will show only if your router supports IPv6, but it is more difficult to remember.
- `RX`: This represents how many packets have been received
- `TX`: This represents how many packets you have sent.

The eth0 port

Pi will typically use `eth0` as its onboard LAN interface. If you add USB LAN devices, they will have an incremented number after the *eth*; this will increment the interface number.

The wlan0 interface

When we add a wireless adapter, `ifconfig` will represent the network interfaces using `wlan` instead of the `eth` prefix. All the details are the same, but it helps to visually identify wired networks from wireless ones.

The lo interface

`lo` is the loopback interface. It is known as the home address because it refers to the local device and has a reserved range starting from `127.0.0.1` and ending with `127.255.255.254`. This is a virtual interface that bypasses local hardware interfaces and rules. It is commonly used for security reasons and during software testing. For example, you may only allow root access of MySQL to localhost (this will include `127.0.0.1` for IPv4 or `::1` for IPv6). This means that only the user or service on the computer will be granted access to this resource. It is reserved and neither the localhost nor `127.x.y.z` can be assigned to users on the Internet.

Wireless configuration – Wi-Fi

Wi-Fi is a very convenient way to allow your Pi to operate in a remote place within the wireless range. There are many types of wireless adapters available, and not all drivers are included with Raspbian. You might have to install a specific driver, but that is usually a simple process. However, when talking about wireless, you have to target the chip that is used on the adapter and not the end vendor who is selling it.

Recommended wireless adapters

There is a comprehensive list of compatible adapters available at `http://elinux.org/RPi_USB_Wi-Fi_Adapters`.

One of the most stable and affordable chipsets is the *Realtek RT8191*, which works with the **802.11n** standard. It is also compatible with the older **802.11b/g** specifications, just in case your router does not support the newer 802.11n specification. It does not support 5 GHz frequencies or 40 MHz dual band though. Shop around to find the best price and don't be fooled by overpriced adapters that claim to be the only and the best one for your Pi.

You should be aware that by default, wireless and wired adapters do not work in combined mode. After configuring a wireless adapter, you will need to remove your wired connection and reboot the Pi. This will configure the wireless interface correctly. Connecting the wired connection usually drops the wireless connection after boot.

Setting up from the desktop

Using the desktop is the most convenient way to connect to your wireless within minutes. Raspbian releases after October 2012 include a configuration utility that can be found on the desktop. Your SSID, which is the readable name of your wireless access point, has to be broadcasted by your router for this to work. If you have it hidden, you should use the console method instead:

1. Double-click on **Wifi Config**
2. Click on **scan**
3. Find your SSID and double-click on it.
4. Enter your password and click on **Add**
5. You should now see the Pi connect.
6. Take note of the IP address that the router has assigned to the Pi

Setting up from the console

If you are running your Pi in headless mode (where you do not have a monitor or keyboard connected), you cannot start the desktop and use the **Wifi Config** utility. In the latest distributions of Raspbian, the SSH service is enabled by default; you can SSH into your Pi, log in, and enter the following command:

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

This will open the included file editor called nano and present you with your current interface configuration. We are interested in the last lines. If you do not see some of the lines you may type them in manually.

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
auto wlan0

iface wlan0 inet dhcp
        wpa-ssid "ssid"
        wpa-psk "yourpassword"
```

To save the file, press *Ctrl* and then press *Y* followed by *Enter*.

The `allow-hotplug` command will do as the name suggests; it allows you to plug in and out wireless adapters assigned to the `wlan0` interface. Auto `wlan0` tells Raspbian to configure the interface automatically based on the settings you provide.

We need to replace the SSID and password with the details of your router, keeping the text enclosed with quotation marks. This is the most basic configuration that can be used.

There is one problem with this configuration though; if the wireless network is disconnected, the interface will not be brought back automatically. There are many scripts that try to solve this problem. The configuration requires that you know extra details about the wireless configuration. The next section offers a fully automatic way of doing this though. We will use the wpa supplicant that is now installed by default with wheezy. In `/etc/network/interfaces`, we change the last lines to the following:

```
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

We need to go to the `/etc/wpa_supplicant` directory, create the file, and edit `wpa_supplicant.conf`.

```
sudo touch wpa_supplication.conf
nano wpa_supplication.conf
```

In nano, the simple text editor we need to type in the following:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
  ssid="ssid"
  proto=WPA RSN
  key_mgmt=WPA-PSK
  pairwise=CCMP TKIP
  group=CCMP TKIP
  psk="password"
}
```

The configuration file contains extra details about the connection. The way supplicant works is that it will try to connect using the defined parameters for `proto`, `pairwise`, and `group`. You might have to adjust `key_mgnt`, but `WPA-PSK` is the most common configuration for modern wireless routers.

Using this method will also allow you to connect to your router if it does not broadcast its SSID.

It is a common misconception that disabling SSID broadcast protects your network. There are tools that can still find SSID names using normal wireless cards without effort, even though you have turned this off. The best form of security is to use the latest encryption protocol and a complex password.

Using wicd-curses

Using the `wicd-curses` package is a much simpler way of setting up wireless and even wired interfaces in the console. This package allows you to edit advanced options like static IP, DNS, and hostnames. A daemon also runs in the background that automatically reconnects the interface if the wireless signal drops out, and this is a more reliable way to maintain a wireless network in poor signal areas. To use this package, you should comment out any settings in the `wpa_supplicant` interfaces file.

Installing this package will install many other dependency packages and can take up in excess of 8 MB of extra space. Some sources also claim that running this daemon uses more overhead, but this seems like a fair, small price to pay for ease of use and reliability.

```
sudo apt-get install wicd-curses
sudo wicd-curses
```

You will get a console screen with a list of available wireless access points. Use the cursor keys to select your access point and press the right key to edit its properties.

If you see the message stating that no wireless networks are detected, try to open the preferences window by pressing *P* and type `"wlan0"` in the wireless interface field.

Once you are in the configuring preferences screen for wireless networks, you only need to enter your password in the key field. Save the settings by pressing *F10*.

If you set up a wireless connection using a wired connection, you would need to press *C* to connect to the access point. Keep an eye on the LED. You should see it start to flicker, and after a few seconds, it should become stable, which means a connection has been established. At this point, it is highly likely that the SSH session will be dropped because the wired connection gets disconnected. You should unplug the wired connection and reconnect to SSH using the wireless IP or hostname.

You can find the wireless IP by logging in to your router via the web interface and looking at the DHCP list.

Static network address

Some DHCP servers on routers tend to change your private address every now and again. Setting a static private address is a quick way to prevent this from happening, and it is easier to remember what the IP address is.

However, many newer routers have the ability to assign a preferred IP address in the DHCP settings or automatically assign a long term IP to the device based on its MAC address. Long term usually means until the router is reset to factory defaults or if it runs out of IP addresses and replaces the oldest entry in the DHCP list.

The downfall with using static addresses with most home routers is that the router will not know about this device. The reason is because it would have to scan the entire network endlessly, taking up valuable resources. If the DNS service does not advertise the IP address of the Pi, you will not be able to use its host name (by default, raspberry), and instead you have to type in the full IP address. Furthermore, if you want to use your Pi as a media center, it will take much longer or even not show up in the network list. Worst of all, some routers will not allow you to forward ports to your Pi if it is not in the DHCP client list.

It is good practice to always have a DHCP server control assigned IP addresses on the network so that the DNS server running beside it can work properly. Assigning a static IP address is easy with `wicd-curses`, but you could disable DHCP and DNS on your router and use your Pi as an advanced firewall, DHCP, and DNS server.

This is not covered in this book as it can introduce many other problems that may be specific to vendors and only a minority of people.

Testing and benchmarking your network

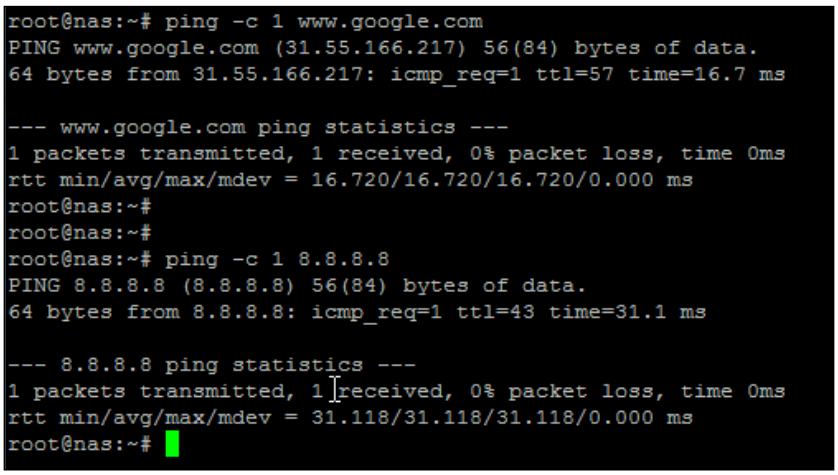
These are essential tests that can be carried out to troubleshoot network problems, but there are also some advanced techniques to benchmark your network.

Basic tests

The simplest way to check whether you are connected to the Internet is to ping a remote address.

We can ping `http://google.com`, but we can also use the shorthand method and a quicker address, `8.8.8.8`, which is Google's public DNS server. This IP address will resolve to the nearest Google DNS server in your area, and even if it goes down, there are many backup servers, making this a reliable test.

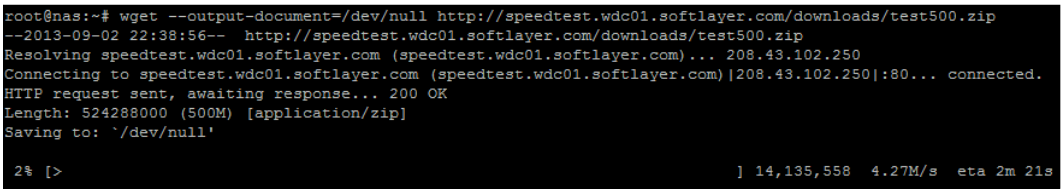
```
ping -c 1 www.google.com
ping -c 1 8.8.8.8
```



A ping can help you determine whether you have access to the public network. You may want to see how fast you can download files to the Pi. We use the popular `http://speedtest.com` server to help us with this directly in the command line.

```
wget --output-document=/dev/null
  http://speedtest.wdc01.softlayer.com/downloads/test500.zip
```

My Pi is connected over a fast wireless network connection, which is connected to a 75 Mbps downstream bandwidth. In the following screenshot, you can see that I achieve about 4.27 Mbps. This varies greatly from site to site. The server used here is in America.



Advanced benchmarking tools

You may be a bit more serious about the performance of your network. Here are a few advanced ways to push your network to the maximum.

Speedtest application

There is a speedtest application available on **github**. It is more advanced than the command line technique, it automatically picks the nearest server and starts to download a large file from there. The benefit of using the closest server is that it will better demonstrate the maximum capacity of your wireless network or your ISP downstream using a wired connection. Furthermore, the application benchmarks your upstream bandwidth. This may be important to you if you were thinking of hosting applications for publically using the Internet.

We will need to install git using aptitude's package manager. We do this by typing in `apt-get` into the command line.

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

Follow the on-screen instructions to install the git package. Using the `/tmp` directory is ideal for short-term applications. This directory in Raspbian is mounted as a RAM drive, and any contents stored there will be lost after a reboot or power failure. If you wish to keep the speedtest application for future use, create a new directory in your home directory and update the path as appropriate.

```
cd /tmp
git clone https://github.com/sivel/speedtest-cli.git
cd speedtest-cli
./speedtest.py
```

Iperf

Iperf is an application that is included with Raspbian. It is a common tool for system administrators for testing a network by creating TCP and UDP streams. Iperf has a client and server functionality, so it requires another computer known as the server. This can be another Pi or another computer that has Iperf installed on it.

This application will push the boundaries of your network interface and architecture. One of the more advanced features of Iperf is to detect packet loss along a complex network, like the Internet. If you have a virtual machine on the Internet, try installing iperf on it and compare the results of your local network against the Internet. Let's install Iperf on Pi and the remote computer.

```
sudo apt-get install iperf
```

The server which will be listening for connections type, you will type:

```
iperf -s
```

The client will show you all the statistics relating to the tests that are carried out.

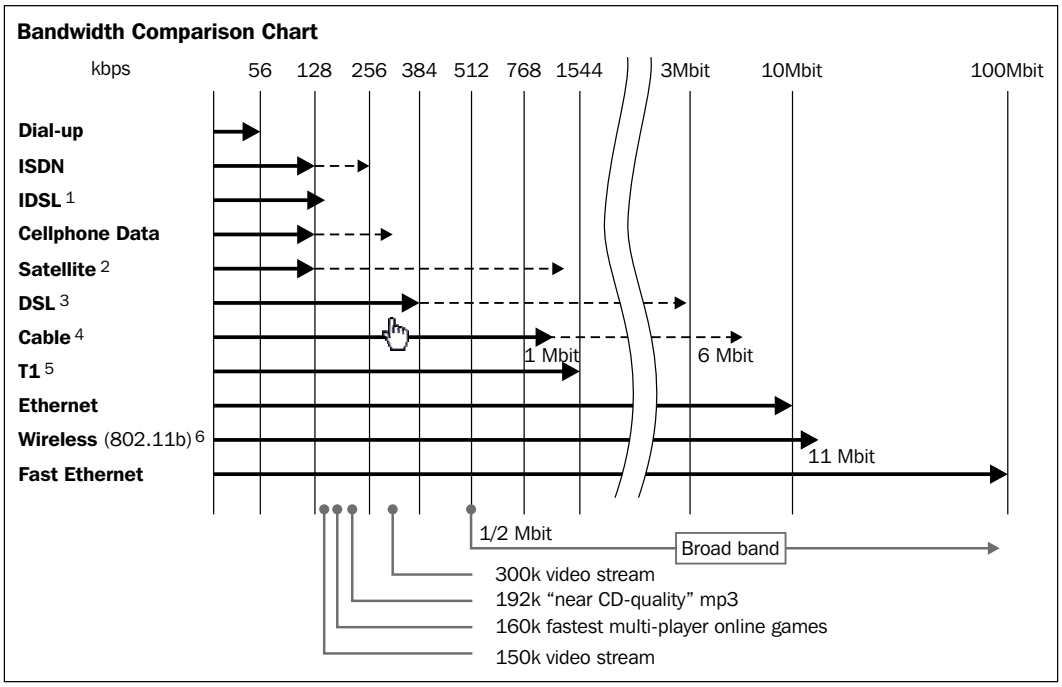
```
iperf -c <ip address of server or domain name of public server>
```

Recommended bandwidth

A basic bandwidth of 256 KBps of both upstream and downstream is recommended for low use hosting of any kind. You can get away with 64 KBps for personal use, say, using a basic website or transferring text data like JSON.

It is standard practice for home ISP providers to supply you with a much larger downstream bandwidth than an upstream one. This is one of the main differences between home and business packages. It is the upstream bandwidth that matters when trying to serve content to users on the Internet.

If you plan on using VoIP with Asterisk, a small upstream might cause terrible delays and jitter. But Asterisk comes licensed with GSM codecs that should work with upstream bandwidth of 64 KBps for a single call. You may purchase closed source codecs that work on much slower connections at the cost of voice quality.



Internet configuration

There are some obstacles to resolve before you can successfully host any kind of Internet application. The solutions are influenced by the package that your ISP provides to you, but almost every problem can be worked around.

ISP packages

Let's have a quick discussion on the two most popular packages supplied by ISP's.

Home packages

The most fundamental part of home packages is that you almost always have a dynamic IP address. Some ISPs will allow you to upgrade to a static IP, but there would still be some limitations. The problem with having a dynamic IP is that it may change without notice. So, if you try to access your network from the Internet using an IP, one day it will certainly stop working after a while. This lease time varies from ISP to ISP, some are as short as a few hours and others can last for years. If you don't know what your current Internet IP address is while you are in a remote place, finding out what it is can be quite inconvenient.

One of the best known solutions is to use a dynamic DNS service. Many Internet routers will have some kind of dynamic DNS setting available, even the one provided by your ISP. The dynamic DNS service is a program that automatically updates DNS records of a public domain name. There are a handful of free dynamic DNS services.

Business packages

Business packages will almost always include a static IP address or an option to purchase one. This really removes a lot of work for workarounds that need to be applied with home packages. You can use the IP address provided to you, but even on the rarest occasion, it may be changed to another IP; however, you will be warned about this in advance.

Other benefits are that ISP's will allow you to change the reverse DNS lookup of your IP, a service that is unavailable with home packages. This service is very important if you plan on running a small mail server. This is one of the prerequisites that need to be configured so that other mail domains know who you are. The only way to solve this problem at home is to use a paid for mail exchange for sending and receiving e-mails using custom domain names.

Many business packages offer unlimited and unshaped traffic. ISP's might deliberately throttle certain ports on home broadband to make them slower and less reliable. Some ISP's go to extreme measures and block these ports on home packages, for example, ports 25, 80, 110, and 443 that are used for mail and web servers.

If you plan on using the Pi for some home experiments, private use, or educational purposes, all you need is to spend a little extra time configuring everything correctly.

Be aware though that hosting production services on home broadband may be against the terms and conditions of your contract. If you anticipate high usage, it is a much better option to purchase a business package.

Dynamic DNS

Dynamic DNS is a way of updating name records for the public DNS. Paid solutions usually offer real-time updates, and free services have a minimum time before the change is published.

I decided to search for dynamic DNS providers online, and I picked a random result, `http://dnsdynamic.org`. You may use any service you like as the protocol is the same for everybody.

Installing the client

I created an account and registered a domain called `pi.craftx.biz`. It automatically detected my public IP and set the **A** record of the domain accordingly.

Use `http://mxtoolbox.com` to verify changes to domain name records. It also has several other useful tools.

a:pi.craftx.biz

Type	Domain Name	IP Address	TTL
A	pi.craftx.biz	31.52.239.30	1 min

Once you have created a dynamic DNS account, you can install the client on the Pi.

```
sudo apt-get install ddclient
```

Follow the onscreen instructions. If you are unsure of any question, you may press enter to omit it, as we will manually edit the configuration file later for fine-tuning. Go to `/etc` and edit the file: `ddclient.conf`.

```
#
# Configuration file for ddclient
#
# /etc/ddclient.conf
daemon=60                          # check IP every 60 seconds
syslog=yes                         # log update msgs to syslog
mail=root                          # mail all msgs to root
mail-failure=root                  # mail failed update msgs to root
pid=/var/run/ddclient.pid          # record PID in file.
ssl=yes                            # use ssl-support.
use=web, web=myip.dnsdynamic.com # get ip from server.
server=www.dnsdynamic.org          # default server
login=***                          # your login
password=***                       # your password
server=www.dnsdynamic.org,
protocol=dyndns2
pi.craftx.bix                      # your domain
```

After saving these changes, you must restart the `ddclient` service by typing in `sudo service ddclient restart`.

You now have a domain name that points to your home IP address. As we progress, you might want to test various applications, and for that, you will need to open ports on your router. Unfortunately, configuring your router is not part of this book, as each vendor handles this differently. Refer to the manual that came with your router to learn how to open and forward ports.

Dynamic DNS domain workaround

You might not like the domain names provided by Dynamic DNS providers, and instead you would like to use a domain you already own. I own the domain `piotrkula.com`, and I would like to use `pi.piotrkula.com` instead of the strange `pi.craftx.bix` domain name.

Log in to your domain name panel and add a new subdomain. Edit the DNS records and make sure that it has absolutely no `A` records associated with it. You will add a new CNAME record; the value in my case is `pi.craftx.bix`. Enter your dynamic DNS domain there.

Now the domain `pi.piotrkula.com` tells the visitor to look at the `pi.craftx.bix` DNS record instead. The visitor will find an `A` record there and that will send them to the correct IP address, which is your Pi. This is not a redirect, and the top domain name will not change. Visitors will not even know you are using a dynamic DNS unless they inspect your private domain record and track down that the CNAME record belongs to a dynamic DNS company. But as long as the domain works, there is no reason to inspect it.

The only drawback to using this workaround is speed. Whenever a visitor looks up the DNS records for the first time, it might take several seconds before a connection is established. After the first request, things go smoothly as the client usually caches the IP for the duration of the session, regardless of what TTL is set on your `A` record. Every new session might experience this delay as the TTL forces the client to read the DNS again. Some dynamic DNS providers might even deliberately slow down DNS lookups and require you to upgrade to premium DNS servers for a premium price.

Summary

It is important to understand how your networks are configured. This includes both the Internet side and your local network. This in-depth understanding will surely help you troubleshoot particular problems on your own.

We now have our Pi connected to our private network and we have configured a domain name that can be accessed by anybody on the Internet. In the next chapters, we will learn more about the various services that can be made available to friends over the Internet.

3
Configuring Extra Features

There are some extra features on the Broadcom chip that can be used out of the box or activated using extra licenses that can be purchased.

Many of these features are undocumented and are found by developers or hobbyists working on various projects for the Pi.

In this chapter, we will learn how to keep the Pi up to date and how to use the extra features of the GPU.

Updating the Raspberry Pi

The Pi essentially has three software layers: the closed source GPU boot process; the boot loader, also known as the firmware; and the operating system. As of writing this book, we cannot update the GPU code. But maybe one day, Broadcom or hardware hackers will tell us how to do this.

This leaves us with the firmware and operating system packages. Broadcom releases regular updates for the firmware as precompiled binaries to the Raspberry Pi foundation, who then release it to the public. The foundation and other community members work on Raspbian and release updates via the aptitude repository; this is where we get all our wonderful applications from.

In this early stage of development, it is essential to keep both the firmware and packages up to date so that you can benefit from bug fixes and new or improved functionality from the Broadcom chip.

Updating firmware

Updating the firmware used to be quite an involved process, but thanks to a user on GitHub who goes by the alias "Hexxeh," we now have some code that automatically does this for us.

The update tool has been included in the official repository, and you can simply install it using `apt-get` and then execute it as shown in the following command lines:

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

After the process is complete, you will need to restart the Pi in order to load the new firmware.

You may find that running the firmware update process does not always solve a particular problem. Sometimes, it may be easier to download the latest Raspbian package and burn a completely new image to the SD card.

Updating packages

Keeping Raspbian packages up to date is also very important as many changes might work together with fixes published in the firmware. First, we update the source list that downloads a list of packages and their versions to the aptitude cache. Then, as shown in the following command lines, we run the `update` command that will compare the packages that are already installed and compare their dependencies, then download and update accordingly:

```
sudo apt-get update
sudo apt-get upgrade
```

Updating some packages like PHP might break the existing code or applications.

Outcomes

If you have a long-term or production project that runs independently, it is NOT a good idea to log in from time to time to update the packages. Windows users are conditioned to update frequently, and it is very rare that something will go wrong. Although on Linux running updates will update your software and operating system components, it could cause incompatibilities.

Some Linux distributions such as CentOS only release yearly updates to keep production environments as stable as possible. With Raspbian, it might be better to download and reload the latest image instead of upgrading from the operating system. This ensures that the libraries are tested and compatible with the release.

Hardware watchdog

A hardware *watchdog* is a digital clock that needs to be regularly restarted before it reaches a certain time.

In the TV series Lost, there is a dead man's switch hidden on the island that needs to be pressed at regular intervals; otherwise, an unprecedented event will begin. In terms of the Broadcom GPU, if the switch is not pressed, it means that the system has stopped responding; the reaction event is to restart the Raspberry and reload the operating system.

Raspbian has a kernel module included that is disabled by default and deals with the watchdog hardware. A configurable daemon runs on the software layer that sends regular events (like pressing a button), referred to as a heartbeat to the watchdog, via the kernel module.

Enabling the watchdog and daemon

To get everything up and running, we need to do a few things in the console as follows:

```
sudo modprobe bcm2708_wdog
sudo vi /etc/modules
```

Add the line of text `bcm2708_wdog` to the file, then save and exit.

Next, we need to install the daemon that will send the heartbeat signals every 10 seconds. We use `chkconfig` and add it to the startup process and then enable it as follows:

```
sudo apt-get install watchdog chkconfig
sudo chkconfig --add watchdog
chkconfig watchdog on
```

We can now configure the daemon to do simple checks. Edit the following file:

```
sudo vi /etc/watchdog.conf
```

Uncomment the lines `max-load-1 = 24` and `watchdog-device` by removing the hash (#) character. The max load means that it will take 24 Raspberry Pis to complete the task in one minute. In normal usage, this should never happen and would only really occur when the Pi has hung.

You can now start the watchdog with that configuration. Each time you change something, you need to restart the watchdog using the following command:

```
sudo /etc/init.d/watchdog start
```

There are some other examples in the configuration file that you may find of interest.

Testing the watchdog

In Linux, you can easily place a function into a separate thread that runs in a new process by using the `&` symbol. By exploiting this feature together with some anonymous functions, we can issue a very crude but effective system halt. This is a quick way to test if the watchdog daemon is working correctly, and it should not be used to halt the Pi. It is known as a fork bomb, and many operating systems are susceptible to this.

The random-looking series of characters are actually anonymous functions that create new anonymous functions, which are placed in its own thread and process. This is an endless and uncontrollable loop. It most likely adopted the name of a bomb because once it starts, it cannot be stopped. Even if you try to kill the original thread, it creates several new threads that need to be killed. It is just impossible to stop, and eventually "bombs" the system into a critical state. Type these characters into the command line and press *Enter*:

```
: (){ :|:& };:
```

After you press *Enter*, the Pi should restart after about 30 seconds, but it might take up to a minute.

Enabling extra decoders

The Broadcom chip actually has extra hardware for encoding and decoding a few other well-known formats. The Raspberry Pi foundation did not include these licenses because they wanted to keep the costs down to a minimum, but they have included the H.264 license. This allows you to watch HD media on your TV or use the webcam module.

They provide a way for users to buy separate licenses if you would like to use these extra encoders/decoders.

At the time of writing, the only project to use these hardware codecs was the OMX project maintained by XBMC. We will compile GStreamer-OMX later in the book, which can be used to stream a video from the Pi to another computer or watch HD video from the console.

Buying licenses

You can go to `http://www.raspberrypi.com/license-keys/` to buy licenses that can be used once per device. Follow the instructions on the website to get your license key.

MPEG-2

Known as H.222/H.262, this is the standard of video and audio encoding widely used by digital television, cable, and satellite TV. It is also the format used to store video and audio data on DVDs.

This means that watching DVDs from a USB DVD-ROM drive should be possible without any CPU overhead whatsoever. This is only possible with using OMX-player or XBMC Media centre, discussed later in the book.

VC-1

VC-1 is formally known as SMPTE 421M and was developed by Microsoft. Today, it is the official video format used on the Xbox and Silverlight framework. The format is supported by HD DVD and Blu-ray players.

The only use for this codec would be to watch Silverlight packaged media, but as you might already know, it is not a widely used format.

Hardware monitoring

The Raspberry foundation provides a tool called `vcgencmd` that gives you detailed data about various hardware used in the Pi. This tool is updated from time to time and can be used to log the temperature of the GPU, voltage levels, processor frequencies, and so on.

To see a list of supported commands, type the following command in the console:

```
vcgencmd commands
```

As newer versions are released, there will be more commands available in here. To check the current GPU temperature, use the following command:

```
vcgencmd measure_temp
```

We can use the following command to check how RAM is split for the CPU and GPU:

```
vcgencmd get_mem arm/gpu
```

To check the firmware version, use the following command:

```
vcgencmd version
```

The output of all these commands is simple text that can be parsed and displayed on a website or stored in a database.

Summary

The Raspberry Pi has very capable GPU and ARM processors with features that might not have been disclosed or discovered yet. Open development is slowed down by the fact that Broadcom is keeping everything closed source, and even if somebody gets a peak at some source code, they enforce strict non-disclosure terms and conditions. It is still in the early stages of development, but the hype created around the Raspberry Pi has brought you, me, and thousands of people to this hack space.

This chapter's intention was to teach you about how hardware relies on good software, but most importantly, to it also shows you how to leverage hardware using ready-made software packages.

In the next chapter, we will look at how to configure a web server and a database.

4
Using a Fast PHP Web Server and Database

Working with nginx

Nginx, pronounced as engine x, is a lightweight HTTP server. It also has modules that allow it to work as a reverse proxy and mail server without installing any other dependencies. You can find out more information by visiting `http://wiki.nginx.org/`.

The Raspberry Pi does not have too much RAM to spare, and nginx is specially designed to use the least amount of resources but deliver content at higher speeds. You can tweak Apache, but it is still very bulky; this will not be discussed in this book because even the best optimized Apache server is not as good as the normal nginx installation.

Nginx streamlines input and output processes using the shortest route possible, where Apache relies on other modules to do the work, even if it is to serve a simple file. These modules in Apache take up unnecessary RAM and CPU power.

According to Netcraft, a company that provides web server market share analysis, nginx served roughly 15 percent of traffic for the top busiest sites during October 2013, and this number is growing yearly.

Installing nginx

The easiest way to install nginx is to use apt-get.

```
sudo apt-get install nginx
```

By default, nginx serves pages from `/usr/share/nginx/www`, but most Linux admins tend to be more familiar with the location `/var/www` that Apache uses by default. It does not matter where you keep the files, as long as you keep it well structured. We will use `/var/www` in this chapter.

```
sudo mkdir /var/wwww
```

Inside this directory, we will create more directories that will hold various virtual hosts. Virtual hosts are domain names that tell the web server about the configuration for that requested domain. We will create a holding directory named `local` and a directory named `web` that will hold the web files.

```
mkdir /var/www/local/web
```

Configuring virtual hosts

A virtual host is a specific website to be served off a given server. You may have many virtual hosts to serve different domains (or subdomains) on one system.

Nginx uses the directory `/etc/nginx/sites-available/` to store configuration files for virtual hosts. At startup, it will check another directory, `/etc/nginx/sites-enabled/`, and usually, these files are symbolic links to the configuration files in `sites-available`.

There should be a file called `default`, which is the default site that nginx serves. We will remove the symbolic link of the default site, create a new configuration file and symbolic link, and finally restart nginx. We will use the name `nginpi`, but feel free to use any name you like for the virtual host.

```
sudo service nginx stop
sudo unlink /etc/nginx/sites-enabled/default
sudo touch /etc/nginx/sites-available/nginpi
```

Edit the new nginx configuration file found at `/etc/nginx/nginx.conf` and add the following to it:

```
server {
    listen 80;
    root /var/www/local/web;
    index index.html index.html;
}
```

This is the most basic configuration to serve up some standard HTML markup to any browser. This can be enough for some people as you can include jQuery into these files and send AJAX requests to other sites or network locations to get various data without PHP running on the Pi. Let's enable the site:

```
cd /etc/nginx/sites-enabled
sudo ln -s ../sites-available/nginpi
sudo service nginx start
```

1. Create a file named `index.html` in the `/var/www` directory.
2. Place some basic HTML markup into it.
3. Use a browser on another computer on your network that can access the Pi on port `80` and browse to it.
4. You should see your basic web page now.

Restart nginx after each configuration change by typing in the command line `service nginx reload`.

Installing PHP

As mentioned before, nginx simply handles web requests as quickly as possible. To get PHP to work with nginx, we need to install an interface called **FastCGI**. This will allow nginx to send requests to PHP and receive and display the response. To help improve PHP performance, we will also install a PHP extension called **Alternative PHP Cache** (**APC**). It is much faster and uses fewer resources than the standard PHP caching.

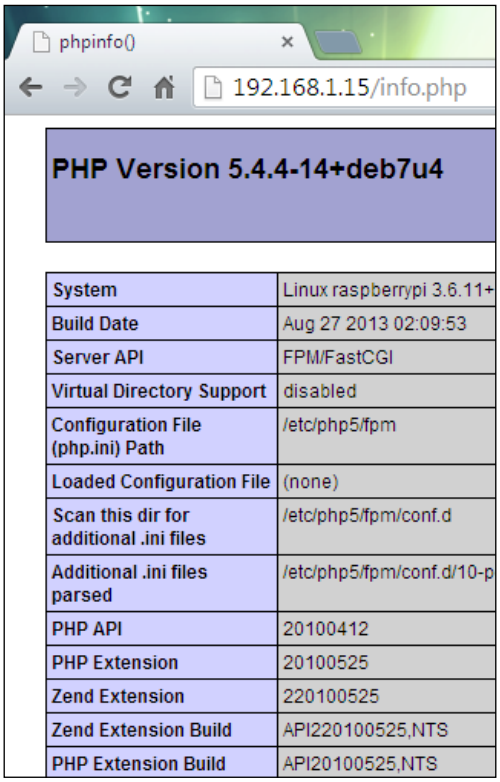
After installing, PHP should be running, but we will restart it to make sure all the extensions were loaded correctly.

```
apt-get install php5-fpm php-apc
service php5-frmp restart
```

We need to update the nginx virtual host's configuration file now to tell nginx how to handle PHP requests. This time, you need to restart nginx after you save the file.

```
server {
    listen 80;
    root /var/www/local/web;
    index index.php index.html index.htm;

    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php5-fpm.sock;
```

```
            fastcgi_index index.php;
            include fastcgi_params;
        }
    }
```



Create a new file named `info.php` in `/var/www /local/web/` and add these simple lines to it:

```
<?php
   phpinfo();
?>
```

Save the file and navigate to it in your browser.

You should see something similar to the preceding screenshot. This means that PHP is configured and you can use any PHP web application or write your own PHP scripts now.

Installing a database

When we think of a database on a Linux machine, the first thing that often comes to our mind is MySQL. It is the first choice because it is free, open source, and reliable, and comes with a rich tool set. However, you have to remember that the Pi is not a server, and you might just want to use SQLite instead.

According to Wikipedia, as of July 2013, MySQL is the world's second most widely used open-source relational database management system.

Installing MySQL

The most popular tool to manage MySQL is **phpMyAdmin**. We will install these two and then test the installation. This can take up to 15 minutes.

```
sudo apt-get update
sudo apt-get install mysql-server
```

You will be presented with blue-background screens asking you for the root user password. You should write these down or store them in an application like **KeePass**.

The MySQL root user must *never* be used by web applications, especially production ones available on the Internet. The phpMyAdmin application needs to run with root settings so that it can fully manage MySQL. The best practice for phpMyAdmin is to only allow local network connections. This is because it will be used to create or delete databases, users, and passwords and apply a permission access to those databases.

```
sudo apt-get install phpMyAdmin
```

PhpMyAdmin is a complex web application that uses many files and PHP libraries to allow you to manage MySQL and the databases found within.

During installation, you get another blue screen asking for the web server that should be automatically configured. Regrettably, nginx is *not* on the list, so click on **OK**. After a short while, a final screen will ask about `dbconfig-commin`; click on **NO**.

We need to manually configure phpMyAdmin just like any other website for nginx. PhpMyAdmin places its files in the shared location `/usr/share/phpmyadmin`, and anyone with access to the system and the username and password can use this shared site to access only their database.

In `sites-available`, create a new file called `phpmyadmin` and use the following configuration. Don't forget to create a symbolic link in `sites-enabled` and reload nginx.

```
server {
  location /phpmyadmin {
    root /usr/share/;
    index index.php index.html index.htm;
    location ~ ^/phpmyadmin/(.+\.php)$ {
      try_files $uri =404;
      root /usr/share/;
      fastcgi_pass 127.0.0.1:9000;
      fastcgi_index index.php;
      fastcgi_param SCRIPT_FILENAME
  $document_root$fastcgi_script_name;
        include /etc/nginx/fastcgi_params;
    }
  location ~* ^/phpmyadmin/(.+\.
  (jpg|jpeg|gif|css|png|js|ico|html|xml|txt))$ {
    root /usr/share/;
    }
  }
  location /phpMyAdmin {
    rewrite ^/* /phpmyadmin last;
  }

}
```

You can now browse to `http://pi/phpmyadmin/setup` and follow the onscreen instructions to get the most up-to-date and best practice information to set up everything correctly and securely.

Installing SQLite3

SQLite3 is a self-contained, easy-to-install, lightweight transactional database. The website boasts that it is the most deployed database in the world. As we know from Wikipedia, it is mostly deployed on embedded devices, but the most used database is still MySQL. None of this really matters to us because both these databases are very good at what they do, except that SQLite is a better choice if you are looking for performance on the Pi.

```
sudo apt-get install sqlite3 php5-sqlite
```

We will be using `phpliteadmin`, a single-file administration utility made specifically for SQLite Versions 2 and 3. Also, create a new subdirectory for `phpliteadmin`. You should visit the website to get the latest version and download that file instead.

```
mkdir /var/www/phpliteadmin/web
cd /var/www/phpliteadmin/web
wget https://phpliteadmin.googlecode.com/files/phpliteAdmin_v1-9-4-1.zip
unzip phpliteAdmin_v1-9-4-1.zip
```

The easiest way to get `phpliteadmin` working locally is by running it on its own dedicated port. Create a new virtual host file with the following contents:

```
server {
  listen 81;
  root /var/www/phpliteadmin/web/;
  index phpliteadmin.php;

  location ~ \.php$ {
      fastcgi_pass unix:/var/run/php5-fpm.sock;
      fastcgi_index phpliteadmin.php;
      include fastcgi_params;
  }
}
```

Reload nginx and navigate to your Pi site, `http://pi:81`. The default password is `admin`. At this point, you will get an error message saying there are no databases and the directory is not writeable. SQLite creates flat files that contain all the data, and you can manage where you would like to keep these files. There is no server managing these files and any SQLite-compatible client can connect to these files.

You can create a test database using the SQLite3 command line. In the web directory of `phpliteadmin`, type in the following:

```
sqlite3 test.db

BEGIN;
CREATE TABLE temps (theDate DATE, name TEXT);
COMMIT;

.exit
```

Refresh the web page where you navigated to `phpliteadmin`. You will see the newly created `test.db` file and the basic `temps` table displayed.

Nginx with custom modules

To configure custom modules, recompile nginx with specific flags or copy new files into specific locations. You can run multiple instances of nginx side by side, as long as the virtual hosts do not overlap.

Running two or more nginx side by side can be beneficial as it reduces overhead and has a high level of isolation. The standard nginx can serve up your PHP application, while another version of nginx will be streaming a live video and a third could be handing `Node.JS`. You could run them all in one instance of nginx. But if you need to monitor performance, then separating them like this gives you greater control while still being more efficient and faster than Apache.

The drawback is that you have configuration files for each instance of nginx, but that seems like a small price to pay for such flexibility.

You can find a list of common modules at `http://wiki.nginx.org/Modules`.

Summary

In this chapter, we covered the following topics:

- Installing the lightweight web server called nginx
- Installing and configuring PHP
- Deciding which database to install and configure

In the next chapter, we will set up the Pi to store and share files.

5
Setting Up a File Server

Windows, Linux, and Mac OS X all use different filesystems, and they have different ways of sharing files across a network. On top of all this, you also get network-specific protocols, for example, the ones that are used over the Internet. If you are wondering which is the best one, you will never find the answer; instead, you should use the technologies that you find most suitable for your application and with which you are most familiar.

Preparing the storage medium

At this stage, it is a very good idea to use an externally-powered USB hub. USB-attached hard drives will need the extra power to operate properly, and if you are using a wireless adapter and USB flash drive, you should move them over to the USB hub.

Please make sure that you are not using a drive with important files. This chapter will show you how to format drives and use other filesystems. All the data on your drives will be lost.

For simplicity, in this chapter we will be using an 8 GB USB flash drive; but the concept is the same for USB hard drives.

Listing the available drives

In the console, you can use the `fdisk` command to get a list of drives and partitions as shown in the following command line:

```
sudo fdisk -l
```

The `fdisk` command will show you the names of the disks used in Linux, their full sizes, and partitions. The drive under `/dev/mmcblk` is the internal SD card, and you do not want to do anything with that device. Instead, you should always look for drives marked as `/dev/sdxn`, where `x` is usually an alphabetical letter representing a physical drive and `n` is a number representing the partitions of the device in ascending order.

Unfortunately, the device names **sda** are not assigned in a particular order, and there is no guarantee that the same name will be used for the same drives. This can become a problem when you start to use two or more hard drives. The partition numbers never change though, and it represent the exact order in which the partitions where created.

The following screenshot shows that the USB flash drive has come up as a `/dev/sda` device, and it has 7803 MB with a FAT32 filesystem:

```
root@raspberrypi:~# fdisk -l

Disk /dev/mmcblk0: 3904 MB, 3904897024 bytes
4 heads, 16 sectors/track, 119168 cylinders, total 7626752 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000c7b31

        Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1            8192      122879       57344    c  W95 FAT32 (LBA)
/dev/mmcblk0p2          122880     7626751     3751936   83  Linux

Disk /dev/sda: 7803 MB, 7803174912 bytes
122 heads, 58 sectors/track, 2153 cylinders, total 15240576 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xc3072e18

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        8064    15240575     7616256    b  W95 FAT32
```

Formatting a drive

The Pi is capable of reading and writing to NTFS, which Windows uses. It can also read/write HFS+, which is used by Macintosh. Both these methods are fine for temporary attachment of removable media if you need to copy something quickly. The NTFS and HSF+ filesystems are not native to Linux, and they take a lot of overhead to convert data between what Linux understands and what the other filesystems understand. Some unexpected errors might occur and cause loss of data, which nobody wants!

Ext4 is the preferred storage filesystem in Linux. Media mounted using this filesystem in Linux is really fast and reliable. There are ways to use it on Mac OS X and Windows, but that is not the goal of this chapter. You should commit to using the media as a long term storage that will stay connected to the Pi as long as you would like it to work, hidden behind the coach or in the wardrobe—out of sight.

A word of warning: this step will destroy all data and create an Ext4 partition.

This command will quickly remove everything from your hard drive, including partitions and boot sectors. Be sure to use the correct device name that you saw in `fdisk`. In this case, it is `sda`, but you need to double check this on your Pi as it may be different. This can be done using the following command:

```
dd if=/dev/zero of=/dev/sda bs=512 count=1
```

Now, we will create a new partition using `fdisk`. I want to use the entire drive and use defaults that `fdisk` provides. You can use the `m` command in `fdisk` at any time for more help as shown in the following command line snippet:

```
fdisk /dev/sda
n "new partition"
<enter> "Uses p as the default"
<enter> "Uses 1 as the default"
<enter> "Uses default first sector"
<enter> "Uses last sector available on drive"
w "write partition data and exit"
```

The last line is used to write the Ext4 filesystem on the newly created partition found at `sda1`. The `-L` flag in the following command line provides the name of the partition. You can use any short name that you like.

```
sudo mkfs.ext4 /dev/sda1 -L nas001
```

Mounting the drives

Linux has a directory called `/mnt` where you create mount points for various hard drives or other network connections. There is also a directory called `/media` where you can mount drives. This is a convention used throughout Linux so that people have an idea of how to organize all these files. These are just normal directories; and if you wish, you can create your own directory called `/nas`, and mount all your drives associated with Network Attached Storage (NAS) there.

I will create a directory called `/nas` and mount my USB drive there. Then, I will create a subdirectory called `USB001`. We will also create directories that we will use for sharing as shown in the following command line snippet:

```
mkdir /nas
cd /nas
mkdir USB001
sudo mount -t ext4 -v /dev/sda1 /nas/USB001/
cd /USB001
mkdir public
mkdir work
```

Remounting a disk after reboot

The quickest way to get your drives remounted after a reboot is to add a few lines to the `fstab` file, which controls the filesystem configuration and can be found at `/etc/fstab`.

Simply add the following command line. The first part is the partition and the latter part is the mount point. You need to adjust these to your hardware. The final options are for auto mounting. Do not change these options.

```
/dev/sda1 /media/HDD1 auto noatime 0 0
  /dev/sda2 /media/HDD2 auto noatime 0 0
```

Accessing files

We will go over several ways of allowing access to files on the Pi on the network and the Internet. You should choose the method that suits you best, as enabling more than one way makes it easier to compromise your system or network.

FTP service

The File Transfer Protocol specification was originally published in 1971, but we currently use a specification from 1985 that everybody should really start moving away from. FTP uses port 21.

A much newer specification known as Secure FTP (SFTP) supports the IPv6 and Secure Socket Layer (SSL) encryption. Installing FTP will just be a waste of time as SSH comes with built-in support and is enabled by default to use SFTP. SFTP generally uses port 22, which is the same port as SSH.

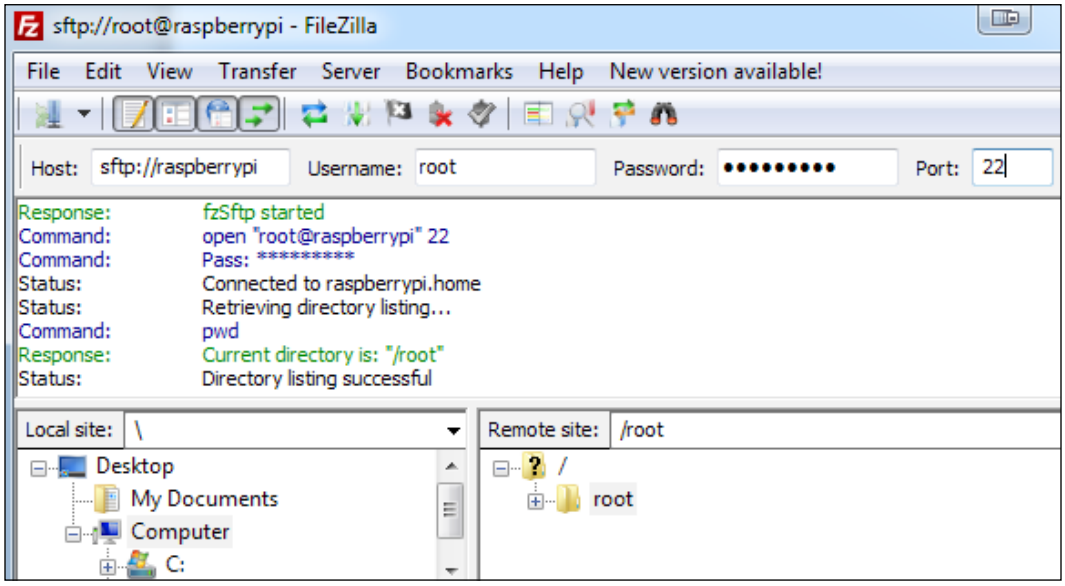
You should create and use a separate user for SFTP access. I will demonstrate how to connect to your Pi with two popular clients, using the root account for simplicity.

Connecting with FileZilla

FileZilla is open source and can be run on Windows, Mac OS X, or Linux. Download and install it on your computer. We connect to the Pi using the following steps:

1. In the **Host** field, enter the IP address or DNS name of your Pi.
2. In the **Username** field, enter the name of the user on the Pi.
   For example, `root`.
3. In the **Password** field, enter the password of that user.
4. In the **Port** field, enter the name of the port as `22`. This tells FileZilla that you want to connect using SFTP.

The following screenshot will show how FileZilla is connected to our Pi:



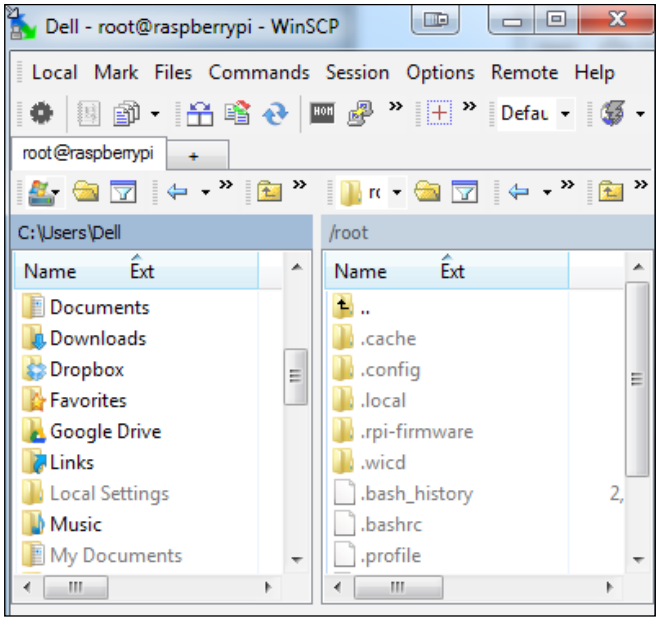
Connecting with WinSCP

WinSCP is the client preferred by Windows users who need to connect to any Linux box running the SSH or SFTP server.

WinSCP also offers Secure Copy Protocol (SCP), which is not specified by any standard. It uses Secure Shell for data transfers and the same authentication method.

Go to `http://winscp.net/` and find the **[Go to Download page]** link from where you will get the installer or portable binary.

Just like with FileZilla, you will enter the hostname, username, and password. You can use SFTP or SCP as shown in the following screenshot:



Samba service

Samba is an implementation of the SMB/CIFS networking protocol which is mostly used on computers running Microsoft Windows. It is basically a way of allowing a Windows computer to connect to Linux systems and access their shared files or printers. It is not supported by Windows, and was originally developed by Andrew Tridgell.

If you want to share media from Linux with other Windows computers on your network, Samba is the best software option.

Installing and configuring Samba

You need to install Samba and a common library used by Samba using the following command line:

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

You can then navigate to `/etc/samba` and edit the `smb.conf` file. On a private network for personal use, you may skip any steps that require the use of passwords to mount network shares. In an exposed or production environment, always use authentication.

If you personally do not like to put security on internal network shares that is perfectly safe, since these files cannot be accessed from the Internet, you may just want to browse your network freely and copy or move files around without the extra headache of entering passwords.

By default, Samba is set for anonymous (unauthenticated) access. To change this, scroll down and uncomment the line `security = user` by removing the preceding `#`. This will tell Samba to authenticate against the users that you have created.

For demonstration purposes, create a system user to access Samba. We will quickly create a user for the purpose of demonstration, as there is an extra step to add the user to the Samba authentication list. The following command line snippet is an example of creating a system user to access Samba:

```
useradd bond007 -m -G users
passwd bond007
smbpasswd -a bond007
```

Network shares

At the bottom of the `smb.conf` file, we will add two shares as shown in the following command line snippet. One account is for guests without authentication, while the other account will only be accessible by our new user, James Bond.

```
[public] comment = Media share
path = /nas/USB001/public
force user = "root"
read only = no
guest ok = yes
public = yes

[work]
comment = Work share
path = /nas/USB001/work
valid users = "bond007" #or @users to allow the group users access
force group = users
read only = no
writeable = yes
```

Samba can be quite difficult to understand in terms of permissions. Let's start off with the public share. You will notice that we put a line in `smb.conf` that says `force user = "root"`. We do this because we created the public directory as root user, so the permissions applied to the directory and all the files to be created later can only be for the user root! So, we tell Samba to imitate guests, as root, so that they can read/write as the root user.

It is bad practice to force the root user to use the Pi account that comes with Raspbian instead.

The same will apply for the user James Bond and his directory. If you created the work directory as the root user or Pi, even James Bond will not be allowed to write new files. To overcome this problem, you can simply change the owner of the folder by typing the following command line:

```
chown bond007:users /mnt/USB001/work
```

You can also assign a directory to an entire group by replacing `bond007` with users. The command would look like `chown users:users ....` This means that anybody in the group can write and delete files on that folder now.

The configuration file has many advanced features and I recommend reading the online manual to learn more about security, sharing printers, and even how to get Samba to act like a Windows domain server!

You can now use any Windows computer to browse the network and access the newly created shares.

AFP for Macintosh

Samba also works with Macintosh, but Apple has its own networking protocol called Apple Filling Protocol (AFP). There is a software that you can install on the Pi so that your Macintosh computers can detect the Raspberry Pi. Place it in the shared section of the sidebar.

Installing and configuring

Installing Netatalk has become really easy as most of the configuration is set up by default. At the time of writing this book, the version used is 2.3.3.

```
sudo apt-get install netatalk
```

Within 30 to 60 seconds, you should not see your Raspberry pop up in the shared section on your Macintosh. The default directory that is shared is the home directory of the user that you log in as, for example, user Pi.

Shares and Time Machine

We will create a few entries to use your external drive for the shares that you need, and this version of Netatalk also supports Time Machine.

You will need to edit this file to customize some of the settings as follows:

```
/etc/netatalk/AppleVolumes.default
#Ammend default to look like this
:DEFAULT: options:upriv,usedots,rw,tm
#Add this to end of the file
/nas/USB001/public "Public"
/nas/USB001/work "James Bond"
```

Read the configuration file for more options and adjusting security. After saving, you must restart Netatalk using the following command:

```
service netatalk restart
```

BitTorrent Sync

All the bad things that you might have heard about torrents were primarily caused by private companies who were deliberately placing infected files to try and track users sharing copyrighted material, or malicious users who were after your personal details.

Sync is not a public network; it only uses the revolutionary peer-to-peer technology from conventional torrents, which allows you to share an unlimited amount of data with an unlimited number of computers. Files are not stored on the cloud. They are stored on all the computers that have access to the private files, and you can control who has access and where these files will be stored. Sync includes basic file versioning, sharing with mobile phones, LAN synchronizing, and delta updates. It also allows massive file transfers of up to several gigabytes without any problem, as long as your filesystem can support such large files.

Installing Sync

At the time of writing this book, there were a few repositories that kept this package with the startup scripts. A word of warning: do not to trust sources other than the ones that are enabled by default. People can easily alter code and place their own version of the binary with some nasty backdoors. Go to the official website, or use the links provided by the official website.

We will create a new directory, download the ARM version of Sync, and extract it, as shown in the following command line snippet:

```
mkdir ~/.btsync && cd ~/.btsync

wget http://btsync.s3-website-us-east-1.amazonaws.com/btsync_arm.tar.gz

tar -zxvf btsync_arm.tar.gz
./btsync
```

You should see a message saying that Sync forked to the background. This means that everything is running well. You can now navigate a browser on the same network to your Pi's address: `http://raspberrypi:8888/gui`.

Add the recently created public directory, and copy the generated secret. Now, you should go and install the application on your Mac OS X, Windows, or other Linux machine. Use the secret that was generated on the Pi on other machines. It can detect machines on the same network, which increases the speed of synchronization.

Sync tries its best to configure ports and use protocols to communicate over the Internet. In theory, you can now call your friend on the phone, ask them to install Sync, and e-mail them the secret. They will now start seeing the same files as you.

BitTorrent Sync uses external servers for tracking purposes, but no files are stored on them.

BitTorrent Sync claims to use SRP for mutual authentication and Perfect Forward Secrecy, in which all data is then encrypted with AES-128 Counter Mode and unique session keys. It is a very complex way to ensure nobody can guess your key, even BitTorrent. It is all dynamically generated on the fly, and then the clients negotiate more keys during transfers. This sounds really good, but you would have to take them on their word, as the project is closed source.

Autostart

We need to create a startup script so that `btsync` will always run in the background. Create a new file in `/etc/init.d/btsync`, and type in the following code. You need to check the start directory and make sure that it's owned by the users that you used while you were logged in.

```
#! /bin/sh
# /etc/init.d/btsync
#

# Carry out specific functions when asked to by the system
case "$1" in
start)
    /home/root/.btsync/btsync
    ;;
stop)
    killall btsync
    ;;
*)
    echo "Usage: /etc/init.d/btsync {start|stop}"
    exit 1
    ;;
esac

exit 0
```

Then, change the permission of the file and register it to run at boot.

```
sudo chmod 755 /etc/init.d/btsync
sudo /etc/init.d/btsync start       # test that the script starts
sudo /etc/init.d/btsync stop        # test that the script stops
sudo update-rc.d btsync defaults
```

Hardware RAID

The Pi does not have a SATA controller on board, and there is no way to attach extra hard drives except via the USB ports. The cool thing is that you can get a USB RAID controller. This makes file storage on the Pi a very attractive option. Technically, the peripheral needed is called a USB SATA multiplier.

Configuration

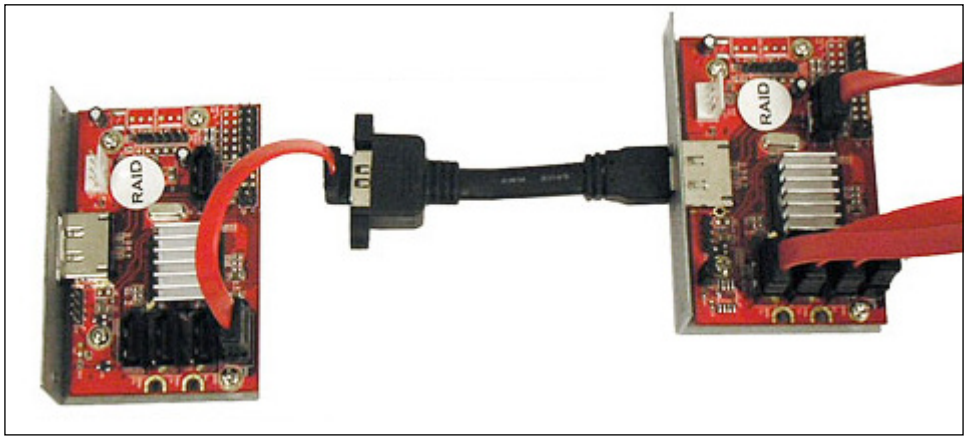
There is no one-step guide to configure these multipliers. Some need to be configured using software in Windows or Linux, while others may have DIP switches that configure the multiplier to a specific configuration.

Addonics is a well-known and fairly easy-to-source multiplier. You will need to search around on the Internet, online auctions, and shops for these. The prices are around the same as the Pi; but if you are looking for redundant storage, then these are the cheapest options you will find.

Massive storage

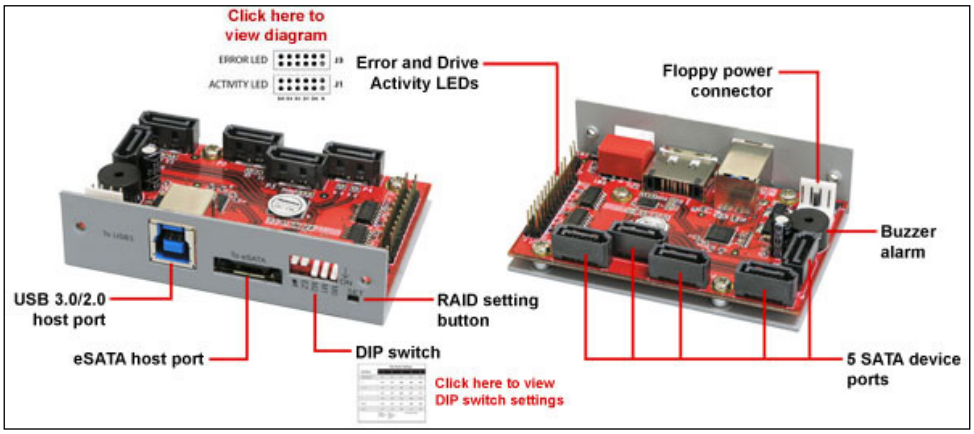
If you are looking to create a really massive storage, you can easily daisy chain a multiplier with another, sometimes up to three or four times. Recently, Addonics developed another device that can be daisy chained infinitely!

Let's assume that we can daisy chain up three levels and we use 2 TB hard drives. We configure all the multipliers to span data across all drives with no redundancy; that is, 5 times 5, times 5. We end up with 125 usable SATA ports and a total of 250 terabytes of storage. This figure is highly impractical for home users, because all those hard drives consume a lot of power. There are people who have hard drive rigs like these at home, so I would think using six multipliers to achieve 25 SATA ports would be a completely viable and cheap option for some readers! The following figure is a simple example of how to replicate 1 SATA port into 5 more Ports using SATA port multipliers:



Redundant storage

We all have lost important data way too many times in our lives, and so you should consider redundancy over massive storage. We can use a set of four 1 TB hard drives that are configured in RAID 5 + S. This gives us 3 TB of usable space. Also, if any one hard drive crashes, you can just replace it quickly and all the data that we resynchronize will be stored without any loss. The following is a figure explaining each feature of a SATA Port replicator:



Summary

This chapter demonstrates that the Pi is capable of doing anything that a normal computer or server would do. It is capable of sharing files with various platforms. You can connect to the Pi using the latest secure File Transport Protocols. It is capable of storing a great amount of data using some clever hardware add-ons.

In the next chapter, we will set up various gaming servers.

6
Setting Up the Game Servers

As great as it would be to run a Counter Strike server on Pi, it is just not possible due to the high requirements of running such game servers. Mostly, this is due to the shortfall of RAM, but some servers also require powerful processors to deliver low latency performance for all players, which could be up to 24 players per game.

In this chapter, you will be introduced to open source games that have been developed by people with a passion for a particular genre of games. These games were reverse engineered, built from scratch, or just became popular due to their simplicity.

Updating to Jessie

At the time of writing this book, the main distribution was Debian Wheezy; but Debian Jessie was added around June 2013 for testing. A lot of game server packages were updated only in Jessie because of newer dependencies.

You can upgrade the entire Pi to Jessie, but you are advised not to do this until the official image is released. Instead, we will install selected, specific packages from Jessie, while keeping the entire distribution stable in Wheezy. These steps can be skipped if you are already using Jessie.

Selective settings

We should add the following two lines into `/etc/apt/sources.list`:

```
deb http://mirrordirector.raspbian.org/raspbian/ jessie main contrib
  non-free rpi
deb http://archive.raspbian.org/raspbian jessie main contrib non-free
  rpi
```

The next step is to tell aptitude to use the wheezy repository for normal updates and that you would like to use Jessie from time to time. These settings should be typed into the file found at `/etc/apt/preferences`. Create this file if it does not exist.

```
Package: *
Pin: release n=wheezy
Pin-Priority: 900

Package: *
Pin: release n=jessie
Pin-Priority: 300

Package: *
Pin: release o=Raspbian
Pin-Priority: -10
```

After adding these settings and saving the files, you should run `apt-get update`.

We will be using `apt-get` with an extra switch to use the new packages. You should only use this switch when it is advised. Upgrading core packages could result in unstable behavior. The command we will be using is as follows:

```
apt-get <-t jessie> <install> <packages>
```

Games servers

We will only be focusing on running the server parts of games that do not require desktop interaction. Instead, you run the client on another computer and connect to Pi. Some clients can run on Pi's X desktop, but they might suffer from low frame rates.

OpenTTD

OpenTTD is a reimplementation of the original **Transport Tycoon** made by Micropose. The game is dangerously addictive and adding Internet play into the mix gives you an extra competitive edge.

You have to have a passion for logistics, strategy, and real-time simulation to enjoy this game. The goal is just something to do with controlling trains and trucks as efficiently as possible and make profit while transporting cargo.

Installing OpenTTD

The latest package is kept only on Jessie.

```
Sudo apt-get -t jessie install openttd
```

Several libraries will be installed during this time. You might be presented with a blue screen, which asks you if you want to restart some services in order to complete the package installation. It is safe to agree to this. This can take several minutes to complete.

Configuring OpenTTD

The configuration file `openttd.cfg` can be found at `~/.open`. It contains a few hundred settings that you can tweak according to your preferences; but some of the important ones are as follows:

- `lan_internet`: This sets the condition for the LAN access. For `lan0` allow Internet and for `lan1` use LAN only.
- `server_name`: This is a unique name to identify the server on the Internet.
- `server_advertise`: We set this to `true` so that it will be visible on the server list.
- `server_bind_ip`: We set this to `0.0.0.0`; this allows the address to bind to any network.
- **server_port:** For this, we keep the default value; this is how other people from the Internet will connect over to the network.

To allow people to connect from the Internet, you should forward the ports `3979` and `3978` to the Pi in your router.

Start the server with the following options:

```
./usr/games/openttd -D
```

Playing OpenTTD

You can now install and run OpenTTD on other computers and invite some friends to join you.

Click on **Multiplayer**, and select **Internet** if you have forwarded the ports for the game. Click on find, and a list of servers will be displayed.

Freeciv

> *This is a free and open source empire-building strategy game inspired by the history of human civilization.*
>
> *–Freeciv*

The game reminds me of *Civilisation* by *Sid Meier*. Its user interface is very similar, but once you start playing the game, you will notice a lot more features and possibilities that are built into the game.

Installing Freeciv

We can install the server from the Raspberry repository:

```
sudo apt-get -t jessie install freeciv-server
```

Several libraries will be installed during this time. You might be presented with a blue screen, which asks you if you want to restart some service in order to complete the package installation. It is safe to agree with this. This can take several minutes to complete.

Configuring Freeciv

The default settings are good enough for you to play straight away. You can start the server by typing the following in the console. The server should not be run with the `sudo` command or as the `root` user, but as a normal unprivileged user.

```
freeciv-server
```

This will actually present you with another console provided by the `freeciv` server. You can adjust settings of the server by typing in commands. You can try to type in `help` and press *Enter*.

If you would like to publish your server to the online community, you need to forward port `5555` to Pi through your router, and then start the server with the extra command line `--meta`.

Playing Freeciv

You can now install and run `freeciv` on your computer. If you are running the same versions of the server and client, you will see it in the LAN.

OpenArena

OpenArena is a free, first person shooter based on the Quake 3 engine. It aims to be a clone of Quake 3, but replaces propriety content with brand new features.

Installing OpenArena

The server installation can take up to 500 MB. Make sure that you have enough free space, and then we can install the game using the Jessie repository.

```
sudo apt-get -t jessie install openarena-server
```

Several libraries will be installed during this time. You might be presented with a blue screen, which asks you if you want to restart some services in order to complete the package installation. It is safe to agree with this. This can take several minutes to complete.

Configuring OpenArena

If you would like to share your server on the Internet, you should open and forward UDP ports `27960` and `27950` on your router. You should go through the settings found at `/etc/open-arena/server.cfg`.

- `set dedicated 2`: We use 2 for the Internet while 1 (default) is LAN
- `v_hostname "Raspberry Pi"`: This is used to set the name of your server
- `sv_master1 "dpmaster.deathmask.net"`: This sets the Internet server that keeps lists
- `sv_maxclients 16`: This is the maximum amount or number of clients that are allowed to connect
- `capturelimit 8`: This is used to set the capture limit in CTF
- `timelimit 15`: This is used to set the time limit
- `fraglimit 35`: This is used to set the frag limit
- `g_motd "This Pie is delicious!"`: This is used to set the message of the day
- `g_gametype 0`: This is used to set the gametype to *Free for All* also known as *Deathmatch*

You can find hundreds of extra settings on the OpenArena Wiki. Remember that the server must *not* be run as `root`.

The OpenArena server runs as a service. We should stop the service and run it on console to make sure everything runs fine. Once you are happy that everything is configured properly, you can start the service again.

```
Sudo service openarena-server stop
/usr/games/openarena-server +exec /etc/openarena-server/server.cfg
```

Playing OpenArena

The game can be played on Windows, Mac OS X, or Linux. You should go to the website for platform-specific clients.

Minecraft

Minecraft is a sandbox indie game where you are in control of a person that can build anything you imagine. There are complex blocks that can be used or programmed to do various server-side calculations. If you would like to see the full potential of Minecraft, you should visit **BitVegas**. This will be more difficult than the other servers as we need to install and configure a lot of dependencies.

Installing Java Hard-Float

We will need to go to `https://jdk8.java.net/download.html`, accept the terms, and copy the link to the Linux ARMv7 HardFP ABI. Now, we need to download the file, ignoring any certificate problems.

```
sudo wget --no-check-certificate
  http://www.java.net/download/jdk8/archive/b111/binaries/jdk-8-ea-
  b111-linux-arm-vfp-hflt-09_oct_2013.tar.gz
```

Then, enter these lines into the console and replace `<filename>` with the latest file you have downloaded.

```
mkdir -p /opt
sudo tar zxvf <filename> -C /opt
rm <filename>
sudo /opt/jdk1.8.0/bin/java –version
```

You can start typing the file name and then press *Tab*. This will autocomplete the filename or display a list of similar names for your convenience.

If you get a response from the version command line that means you now have Java Hard-Float installed on your Raspberry Pi.

Installing the Minecraft server

Minecraft servers are actively developed by various crowds, and there are many configurations to choose from. **Spigot** is a high performance Minecraft server implementation, and we can install it with the following command:

```
cd /home/pi
sudo wget http://ci.md-5.net/job/Spigot/lastStableBuild /artifact/Spigot-Server/target/spigot.jar


#This is the command line use on Model B with 512 ram
sudo /opt/jdk1.8.0/bin/java -Xms256M -Xmx496M -jar
  /home/pi/spigot.jar nogui
```

It takes about 15 minutes to start up for the first time. Once it is ready, you can log on locally and explore your new world. You should exit shortly and configure the server for performance.

Configuring Minecraft

To allow players from the Internet to connect, you will need to open and forward port `25565` to the Pi from your router.

These are the settings that seem to be best for running on the Pi. You can adjust them as you go along.

```
server-name=Raspberry Picraft!
motd=Pie is Delicious
allow-flight=false
spawn-monsters=true
generate-structures=true
enable-query=false
enable-rcon=false
level-name=world
spawn-protection=16
online-mode=true
difficulty=1
gamemode=0
spawn-animals=true
```

```
view-distance=4
level-seed=
hardcore=false
snooper-enabled=false
level-type=DEFAULT
pvp=true
texture-pack=
max-players=20
server-ip=
max-build-height=240
spawn-npcs=true
server-port=25565
white-list=false
generator-settings=
allow-nether=false
```

Playing Minecraft

To connect to your server, you will need to put in your Raspberry IP address in the network settings. If you would like other people to join from the Internet, you should read *Chapter 2, Preparing the Network*, to learn how to set up dynamic DNS.

Summary

In this chapter, we have learned how to use another non-default repository and how to set up various gaming servers on the Pi.

We have also learned how to use the Jessie repository for some packages; how to install OpenTTD, freeciv, and OpenArena from aptitude; and how to set up a high performance Minecraft server using Java Hard-Float. In the next chapter, we will be setting up and possibly even mining for some Crypto Coins.

7
Bitcoins – Pools and Mining

Bitcoin is the first fully implemented cryptocurrency. A cryptocurrency relies on cryptography to generate and validate transactions that become the currency itself. This data is decentralized, meaning it is stored on computers all over the world and it is shared using peer-to-peer technology. Digital counterfeiting is avoided by using a proof-of-work scheme; a concept that works well but cannot be easily explained.

Bitcoin uses SHA-256 as its proof-of-work scheme, but Litecoin, an alternative to Bitcoin, uses scrypt, a password-based key derivation function. Bitcoin and Litecoin are the two most popular cryptocurrencies. They are not compatible with each other, and require different hardware or configurations to work or mine with them. As of writing this chapter, there are more than fifteen currencies and hundreds of other implementations trying to get a place in the spotlight.

In this chapter, we will learn more about Bitcoins, the mining of Bitcoins, and managing your wallet. Using other cryptocurrencies is very similar to Bitcoin because they all use modified code, based on the Bitcoin client.

Installing Bitcoind

Bitcoind is the software that connects you to other nodes in the network. There is no single point of failure, and the more nodes there are the faster and more secure the network becomes. Peers are rewarded with a transaction fee for the first validation of a transaction and are assigned randomly.

After installing Bitcoind by using the following command, we have to synchronize it with the network, which means downloading millions of transactions exceeding a gigabyte in size to your Pi.

```
sudo apt-get install bitcoind
```

Before running the background daemon, you need to consider changing the data directory as the database files can take up many gigabytes of space. Mount a new storage place and always start Bitcoind with the following line:

```
bitcoind -datadir /mnt/HDD/bitcoin -daemon
```

After a few minutes, you can type in the following basic commands. You will need to wait until the entire block chain data is downloaded before you can do anything useful. During this time, the Pi might become less responsive, use a lot of bandwidth, and create a large amount of data in the data directory. This can take up to 12 hours.

```
bitcoind getinfo
bitcoind getblockcount
```

If you already have most of the block chain data downloaded on another computer, you can just copy all the data to the Pi. You can even have the same wallet and addresses in both locations. It is not recommended though, as that means it's twice as easy for hackers to get your wallet. Always encrypt your wallet with a very strong password and keep the password and wallet backup in a safe, offline place—like CDs or USB drives. If you lose the password, you lose access to the wallet, forever.

Bitcoin wallet

You need to generate addresses where other people can send you funds. Theoretically, you can generate an unlimited number of addresses, but practically managing all those addresses would become difficult. Bitcoin, by default, allows you to have a hundred addresses. You always get a default wallet address, which counts as the first address, and this address will receive network fees if you process any transactions for the first time.

Cryptocurrency transactions cannot be reversed. Sending coins to the incorrect address means your coins will be lost forever. On the other hand, it also protects you if you are receiving coins from somebody else, as they cannot reverse the transaction.

These addresses are globally unique, and can never be generated by anybody else in the universe or beyond it. It can get stolen by hackers or viruses, though. Imagine losing your wallet in real life. Unless a kind person returns it to you, the contents of that wallet will be lost forever.

You can back up your wallet file, but you should store it on any two USB flash drives or CDs, and store it in a safety deposit box at home or in the bank. You should never be tempted to copy your wallet to any kind of online storage or backup services, especially if it's unencrypted.

Creating a Bitcoin address

These examples will work with the command line, as you can re-use them to display data on your web server. All online wallets use the Bitcoind API to generate addresses and manage your wallet. As a word of warning, you should never use online wallet services as almost all of them have been hacked, resulting in massive numbers of coins getting stolen.

You can download the Bitcoin-QT client and run it in X if you prefer an easy way to manage your wallet.

The following command lines will create a new receiving address and then list all your addresses:

```
bitcoind getnewaddress
bitcoind listaccounts
```

Receiving Bitcoins

As long as you know your address, people can send you Bitcoins. You do not need to have Bitcoind running all the time as the transactions are processes and are stored in the network. It is just very important to keep a backup of your `wallet.dat` file if you want to send the Bitcoins somewhere else, like an online exchange.

Sending Bitcoins

As long as you have coins in your wallet, you can easily send coins to another address by using the `sendtoaddress` command followed by the address and the amount. Here is an example sending 0.01 Bitcoins to the author's tip jar:

```
bitcoind sendtoaddress 126HA8L1dP7fr7ECu2oEMtz6JpDRnzqp9N 0.01
```

You will get a response, which is a transaction ID. If you do not, the transaction has failed. You need to have at least 0.0001 Bitcoins reserved for the transaction fees. If you are sending large amounts of coins, this transaction fee will also become more expensive.

The value of Bitcoins

In July 2010, the value of Bitcoin was below 1 USD. Investing in Bitcoins was very controversial and a huge risk, since it was difficult to predict if the currency would be accepted or rejected by the community. As of writing this book, the Bitcoin value has exceeded 1100 USD.

You can search the Internet and find interesting articles on early adopters mining Bitcoins, or buying a few hundred dollars and leaving their wallets lying around. Just like one person in the UK who threw away his hard drive with 7500 BTC stored on it. After going public, thousands of people flocked to the public dump ground to search for the hard drive. Other stories include a student who invested a few hundred dollars in 2010, and sold them for hundreds of thousands of dollars. With such large amounts, it is important to consult your local TAX authority or council.

The trend seems to be that the more people find out about Bitcoins and the more media publicity it gets, the higher the value of the currency rises. This also looks like it applies to alternative currencies such as Litecoin.

Mining for Bitcoins

Without getting too deep into the specifics, mining is all about verifying transactions and solving blocks with mathematical computations. Each time a block is solved, the miner gets a reward in Bitcoins. It used to be 50 BTC, but is currently 25 BTC.

In the early days of mining, it was enough for a popular dual-core processor to generate enough hashes to keep the network going. As more and more people mined, the difficulty to accomplish these blocks arose. The next stage was to use the GPU found on powerful graphics cards, and that proved to be profitable for some miners. As of writing this chapter, it is no longer viable to mine with the CPU or GPU, and instead, you should invest in specialist devices, such as the ASIC Block Erupter.

Mining for Bitcoins in solo mode can take an extremely long time and there is no way to predict when you might solve a block. Instead, people join mining pools and combine their computational power in one place. Mining in pools can find up to 5 blocks in a day, but on bad days it can take up to a week with the same hash rate. Once a block is found, the reward is split up proportionally by the number of miners and their hash rate. These coins go to your Bitcoin address, and sometimes there is a small fee that goes towards running the pool.

You also get pay-share pools where you get paid periodically based on the amount of work you have done, whether a block has been found or not. This is preferred by many miners as they can see what is in their account and withdraw at any time.

Mining with ASICMiner

Block Erupters (Sapphire) run at a constant 330 Mh (Mega Hash) and can only calculate SHA. They draw 2.5 watts, and when technically running off a normal USB port on desktop computers or your Pi, it is recommended to run a dedicated powered USB hub. You can find these on Amazon, but they can be highly overpriced. You should join a well-known Bitcoin forum, and follow recommendations on where to get the best deals and best hardware at the time.

Installing CGMiner

CGMiner is one of the first mining software that was used to mine Bitcoins. It is actively updated, and more support for new devices and other cyrptocurrencies is added frequently.

It is best to compile CGMiner with settings to enable the Icarus protocol used for ASICMiner and disable the GPU mining. At the time of writing this, there is no known way to use the Raspberry Pi GPU to aid in computational power.

Compiling does not take long and ensures that CGMiner is stable. At the time of writing, the latest version was 3.6.4, and it is recommended to always use the most recent version for the best stability. The following command-line snippet will help you to install CGMiner:

```
sudo apt-get install git autoconf libtool libcurl4-openssl-dev
libncurses5-dev pkg-config yasm make libusb-1.0-0-dev libusb-1.0-0
libudev-dev


cd ~
sudo git clone git://github.com/ckolivas/cgminer.git cgminer


cd cgminer
sudo ./autogen.sh
export LIBCURL_CFLAGS='-I/usr/include/curl'
export LIBCURL_LIBS='-L/usr/lib -lcurl'


sudo ./configure --enable-bflsc --enable-icarus --disable-opencl


sudo make
```

You will now have a compiled and executable binary called `cgminer`, which you can run from within this directory, which should be `/home/pi/cgminer` or `/home/root/cgminer`, if you logged in as `root`.Connecting to a pool.

There are many pools available with different settings. The most important thing to look for in a pool is security. You do not want your coins to get stolen because a site is not secure. Even the biggest and the best are under constant attack and the slightest vulnerability will be used against them. 50 BTC is a very well kept site, but there was a situation in October 2013 where they were hacked into and all funds were stolen and never reimbursed.

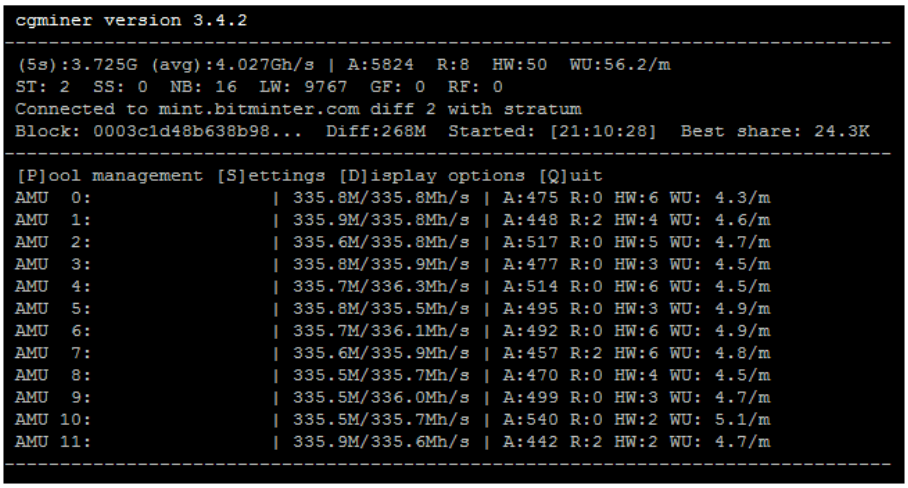
We will use a German run site called BitMinter. When mining on BitMinter, you also collect NameCoins at the same time. It has very low fees. You can even set it to zero percent if you don't want to insure yourself against orphaned blocks. It uses `OpenID` for authentication, and you should consider using a two-step authentication process to protect your account.

Create a file called `miner.sh` in your home directory, run `chmod +x miner.sh`, and add the following command lines to the script:

```
#!/binbash
cd cgminer
./cgminer -o http://mint.bitminter.com:3333 -u <username>_default -p
password
```

You may leave the password empty or use any short text, for example, password, as the servers rely on your username. The username will be the one that you selected during registration. You can now run your script by typing in `./miner.sh` in the command line.

CGMiner will start up and connect to the pool. You can now insert your ASICMiner and it will autodetect the USB device and configure it correctly as shown in the following screenshot:



Summary

In this chapter, we learned about Bitcoins and other cryptocurrencies. We also learned the following points:

- Installing a Bitcoin daemon
- Receiving and sending Bitcoins
- Mining for Bitcoins

In the next chapter, we will try streaming live HD videos.

8
Streaming Live HD Video

In this chapter, we are going to use the official HD camera module designed by the **Raspberry Pi Foundation**. Unfortunately, at the time of writing, there was no Video4Linux driver for this camera device and alternatives had to be used.

We will also be learning how to compile a custom version of the nginx-rtmp-module. It is an add-on created by *Roman Arutyunyan*, and it is capable of retransmitting RTMP video. It can also be used like a media center to list stored video files, play them on demand, or capture live streams from other sites or cameras and record or rebroadcast them.

Before starting with this chapter, you should have the camera module installed and enabled as described in the instruction that was included with the module.

Streaming with GStreamer

At the time of writing, GStreamer from the OMX project is the only encoder that supports hardware encoding and decoding on the Pi. It supports the extra licensed encoders, and you can activate them by reading *Chapter 3*, *Configuring Extra Features*. GStreamer works really well for transmitting streams of video between Linux and Mac OS X. This does not work on Windows as GStreamer is not supported on Windows. Streaming live video to the **VLC** player or **HLS** format is not supported properly using GStreamer.

We can add a repository and install the latest version of GStreamer from there. We need to add this line into `/etc/apt/sources.list`:

```
deb http://vontaene.de/raspbian-updates/ . main
```

Then, we update the aptitude and install the packages we need:

```
# apt-get update
# apt-get install gstreamer1.0
```

Streaming with FFmpeg

Arch Linux uses the latest branch of FFmpeg, but on Raspbian, we have to compile FFmpeg manually. This can take up to 6 hours, but it has the best compatibility for streaming directly from the camera module.

When you type in the last line, it will configure, compile, and install FFmpeg in one go. Let it run and do not close the terminal window; however, just come back from time to time to make sure there were no errors. If something goes wrong, you can rerun the command, and it will go much faster as it tries to pick up from where it stopped.

```
# cd /usr/src
# git clone git://source.ffmpeg.org/ffmpeg.git
# cd ffmpeg
# ./configure && make && make install
```

Raspivid

It is best to use the latest version of Raspivid. Let's download and compile the latest version that is supported by the **userland** repository. This normally takes about 20 minutes to compile and install everything we need.

```
# sudo apt-get install git-core gcc build-essential cmake

# cd /tmp
# sudo mkdir code
# cd code
# git clone git://github.com/raspberrypi/userland.git

# cd userland
# sudo sed -i 's/if (DEFINED CMAKE_TOOLCHAIN_FILE)/if (NOT DEFINED #
  CMAKE_TOOLCHAIN_FILE)/g' makefiles/cmake/arm-linux.cmake

# sudo mkdir build
# cd build
# sudo cmake -DCMAKE_BUILD_TYPE=Release ..
# sudo make

# sudo make install
```

After you install all the newest userland binaries, including `raspivid` and `raspistill`, on the Pi, Raspivid should now support the usage of `-t 0` and possibly some other features.

Compiling nginx-rtmp

Now, we need to compile nginx-rtmp. The following process will install nginx into the default locations. If you already have nginx configured, you will need to change some of the configuration settings or skip the installation process completely. We will install nginx from the repository just to make sure all the dependencies are installed. Afterwards, we will only remove nginx.

```
# sudo apt-get -y install nginx
# sudo apt-get -y remove nginx
# sudo apt-get clean
```

Then, we can download the latest version of the nginx-rtmp module and Version 1.4.1 of nginx. Later versions should be fine, but this version is verified to compile on the Pi and is stable with this module.

```
# cd /usr/scr
## clone the latest version of rtmp module for nginx
# sudo git clone git://github.com/arut/nginx-rtmp-module.git


##download nginx source tested with- 1.4.1
# sudo wget http://nginx.org/download/nginx-1.4.1.tar.gz
# sudo tar -xzvf nginx-1.4.1.tar.gz
```

We need to install some building dependencies. Some of these might be already installed; if they are, `apt-get` will skip over them.

```
# sudo apt-get update
# sudo apt-get install curl build-essential libpcre3-dev libpcre++-dev
zlib1g-dev libcurl4-openssl-dev libssl-dev
```

Now go into the nginx directory and configure it to use the `rtmp-module` plugin source before creating it. (Slashes are meant to be kept on the same line.)

```
# cd nginx-1.4.1
# sudo ./configure --prefix=/var/www \
    --sbin-path=/usr/sbin/nginx \
    --conf-path=/etc/nginx/nginx.conf \
```

```
    --pid-path=/var/run/nginx.pid \
    --error-log-path=/var/log/nginx/error.log \
    --http-log-path=/var/log/nginx/access.log \
    --with-http_ssl_module \
    --without-http_proxy_module \
    --add-module=/usr/src/nginx-rtmp-module
```

Then, create the directory that will hold your HTML files, and we can `make` and then `install`. This takes about 15 minutes.

```
# sudo mkdir -p /var/www
# sudo make
# sudo make install
# nginx –v
```

The last line should display the version number we just compiled and that would mean that our custom nginx is now installed. You can start nginx using `service nginx start` and browse to your Pi to see the default webpage. If it's all ok, then we can stop it using `service nginx stop`.

Configuring nginx

This file configures `nginx-rtmp` to accept FLV-wrapped video in h264 format and then allows clients to stream it on a web page using an open source flash stream player. You should edit the file located at `/etc/nginx/nginx.conf`.

```
http {
  server {
    listen 80;
  }
}
rtmp {
  server {
    listen 1935;
    #publish_time_fix off; #gstreamer only
    application src{
          live on;
    }
  }
}
```

Streaming video using the RTMP module

You can type in this line using the specific pipe for GStreamer. This does not use any hardware and does not always work with RTMP.

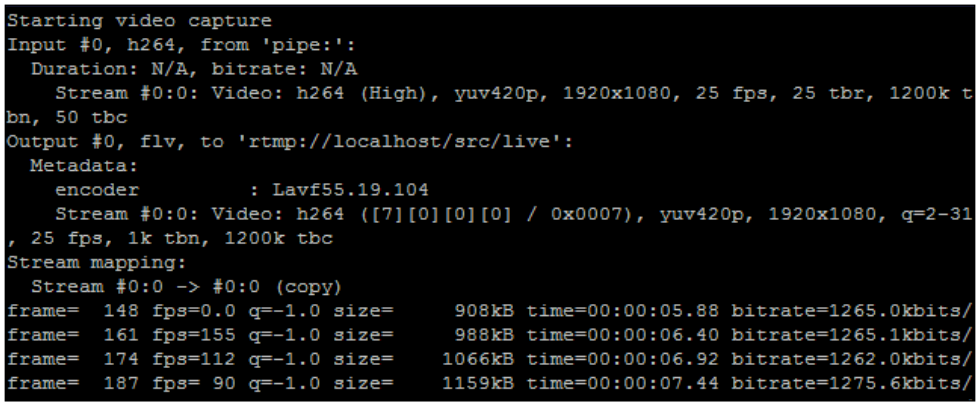
```
# raspivid -t 9999999 -fps 25 -b 1500000 -o - |
  gst-launch-1.0 -v fdsrc
  ! h264parse config-interval=1 ! flvmux
  ! rtmpsink location=\"rtmp://localhost/src/live live=1\"
```

The preferred method is using FFmpeg. It consumes about the same amount of resources as GStreamer, but it is much more stable with the RTMP module.

```
# raspivid -t 999999 -fps 25 -b 1500000 -o - |
  ffmpeg -i - -vcodec copy -an -f flv rtmp://localhost/src/live
```

Both these methods will stream full HD frames at a managed bitrate. If you pipe (| is a system reserved character, used to *pipe* data from one software to another using a memory stream) data from Raspivid to FFmpeg without any bitrate settings, there is a considerable loss of frames and the CPU consumption goes near 100 percent. It is rarely worth the little bit of extra detail, but you can tune that parameter to your liking.

This screenshot shows FFmpeg streaming video to `nginx-rtmp`.



Watching a video

There are many different ways to transport the video, but there is one problem that many people have, which is the SPS/PPS information that only gets sent at the beginning of the stream. There is a solution that works with FFmpeg that is hosted at `https://github.com/AndyA/psips`. Follow the instructions on the site to compile the filter.

RTMP streams

We will use an open source media framework to stream the flash content directly in a browser that supports flash plugins. Download the framework and extract it in the HTML directory.

```
# cd /var/www/html
# Wget http://sourceforge.net/projects/smp.adobe/files/latest/
  download?source=files
# unzip download?source=files
# sudo mv for\ Flash\ Player\ 10.1 osmf
```

In the HTML directory, create an HTML file called `flash.html` and copy this basic markup into it:

```
<!DOCTYPE html>
<html>
  <head>
    <script src=osmf/lib/swfobject.js></script>
    <script src=osmf/lib/ParsedQueryString.js></script>
  </head>
  <body>
    <script>
      var parameters = {
        src: "rtmp://raspberrypi/src/live", autoPlay: true,
          optimizeBuffering : false, optimizeInitialIndex: false,
          liveBufferTime: 0.1,
            liveDynamicStreamingBufferTime: 0.1,
          initialBufferTime : 0.1, expandedBufferTime : 0.1,
          minContinuousPlayback : 0.1, streamType: "live"
      };
      swfobject.embedSWF("osmf/StrobeMediaPlayback.swf",
        "StrobeMediaPlayback",
      1280, 720, "10.1.0", "expressInstall.swf", parameters,
```

```
            {allowFullScreen: "true", wmode: "opaque"},
            {name: "StrobeMediaPlayback"});
        </script>

        <h1>RTMP</h1>
        <div id="StrobeMediaPlayback"></div>
      </body>
    </html>
```

You should now browse with a browser that has flash installed, and you should see a live stream. (Technically, it is delayed by about 0.5 seconds, but this is as fast as it gets for now.)

MPEG streams

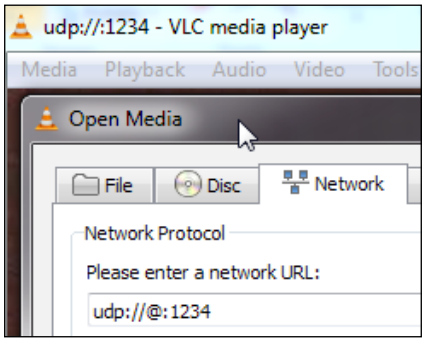
You can use FFmpeg to send MPEG frames, and we will configure FFmpeg to transmit UDP packets to a remote computer, that is, running the VLC player and listening for these packets. UDP is a popular way to send live video and audio because the protocol just sends data and does not care if it got there or not. This saves on bandwidth but receivers can sometimes miss a frame every now and again, resulting in freezing frames or garbled image or audio. It gets corrected as soon as the next set of correct UDP packets is received.

We need to use a PSIPS filter with this method and also wrap everything in a script file. Create a file called `udp.sh` in `/home/pi` and copy the following code into it. Replace the IP with the machine that will run VLC to receive the UDP frames.

```
#!/bin/bash

fifo="live.fifo.h264"

rm -f "$fifo"
mkfifo "$fifo"

raspivid  -w 1280 -h 720 -fps 25 -g 100 -t 99999 -b 1000000 -o -
 | /home/pi/psips > "$fifo" &

ffmpeg -y -f h264 -i "$fifo" -c:v copy -map 0:0 -f mpegts  "$url"
```

On the command line, perform a `chmod +x udp.sh` command and run the file by typing `./udp.sh`.

Start VLC on your other computer and under **Media**, click on **Open Network Stream...** and type in `udp://@1234`; then, click on **Play**. You need to make sure that the Pi is streaming when you click on **Play**. There is a lag while using this method as FFmpeg buffers into VLC. It decides on its own how much to buffer, and the delay could be from 1.5 seconds for a low bitrate up to 10 seconds for high bitrates.



Other streams

Using the examples provided, you can experiment with the parameters and use different protocols to achieve the effect that you are after.

You can experiment with `nginx-rtmp` and `HLS` for native iPhone support. The quality is really good, but it has a greater lag as the protocol saves chunks of files to the disk before it can stream them.

Summary

At the time of writing, it still seems very difficult to stream video. In this chapter, we learned how to stream video in various formats to various platforms, installing and using GStreamer, compiling and using FFmpeg, compiling a custom version of `nginx-rtmp`, and streaming using RTMP FLV and MPEGTS UDP.

9
Setting Up a Media Center

The Pi has an **HDMI** output that is capable of streaming High Definition Audio Video and also supports **CEC** for sharing remote control functions. In this chapter, we will look at how to use the Pi as a media server directly from the command line. We will look into displaying images and playing audio from the command line. We will briefly look at some other solutions that people have come up with and finally install **RaspBMC** (XBMC dedicated Pi image) that uses hardware decoding and CEC out of the box.

If you are going to use Raspbian for these examples, it would be advisable to give the GPU more RAM. A recommended value will be 256 MB or more. Most of these examples will benefit from more GPU RAM than CPU and will improve loading time and stability. You can change core system settings by typing `raspi-config` in the command line.

Slideshows

Linux users are familiar with the command-line program called **fbi** (**frame buffer imageviewer**). We will connect a widescreen monitor or HD TV using an HDMI cable.

You should be aware that using square aspect computer monitors smaller than 19 inches or non-HD TVs are not supported with HDMI output.

There is a project called **HDMIPi** that offers a 9-inch 1280 x 800 LCD screen, which plugs into the HDMI port without any extra parts. There are other smaller LCD screens available, but they can be quite expensive.

Using fbi

Find photos you like and resize them from 1920 x 1080 pixels to JPG using about 80 percent image quality for optimal size and quality. The reason you should resize is so that the Pi can load the data much quicker into the buffer. Using screen sizes from 19 inches to 32 inches will be pretty crisp, but if you are going to use a screen larger than 32 inches, you should resize them to 2560 x 1600. These are the recommended HD resolution guidelines for 16:9 aspect ratio screens.

Use the following command lines over SSH. This will install fbi and start a simple slideshow over SSH, but the output will be the attached monitor on HDMI.

```
# sudo apt-get install fbi
# cd /home/pi/photos
# sudo fbi -T 1 -a -noverbose -t 5 *.*
```

Fbi will now loop through all the images in the directory changing them every 5 seconds until you stop it.

The `T -1` command line tells fbi to use the terminal 1 output (in SSH only), which will be the HDMI port. We tell it to autoresize the images to fit the screen, while `-a`. `-t 5` changes the image every 5 seconds and `-noverbose` disables any debugging information.

You will notice that fbi will not block the console and runs in the background. It is possible to run fbi again and it will not complain about another instance. However, this may cause **Out of memory** errors and unwanted behavior.

To stop fbi, we use the `killall` command.

```
# sudo killall fbi
```

Watching movies

We can watch movies using **OMXPlayer**, which was specifically designed by the XBMC project to use hardware decoding on the Pi. Since OMXPlayer uses the hardware decoders, it is possible for you to use any additionally purchased video licenses.

OMXPlayer requires at least 256 MB GPU RAM to run.

Using OMXPlayer

Download an MP4 file to the Pi (for example, we can use the site **hd-trailers** to find some links). Use OMXPlayer to stream the video and sound to the HDMI port. You can also use RTMP streams directly instead of video files. Replace the URL in `wget` to an existing file on the Internet.

```
# sudo apt-get install omxplayer
# cd /tmp
# sudo wget http://videosite.com/hd.mp4
# omxplayer -o hdmi hd.mp4
```

Playing audio

The latest Raspbian image comes with all the sound drivers and utilities installed. The packages that are used belong to **ALSA**. The Pi has no way to record audio, as it has no microphone jack and the GPIO pins are all digital. To record audio using GPIO, we need to use an A/D (Analogue-to-Digital) device or a USB sound device that has a microphone input.

Using aplay

The following is a pre-installed package that plays WAV files:

```
# cd /tmp
# wget http://goo.gl/Ps3paV
# mv Ps3paV siren.wav
# aplay siren.wav
```

Using OMXPlayer

We can play back MP3 files using OMXPlayer.

```
# omxplayer audio-test.mp3
```

Using AirPlayer

There is a project called `shairport` which works really well on the Pi. It does not support videos or photos, but streaming music is very stable. We will need to get the project and compile it. This will only take a few minutes.

```
# cd /tmp
# git clone -b 1.0-dev git://github.com/abrasive/shairport.git
# cd shairport
# sudo ./configure && sudo make && sudo make install
```

The files on `tmp` will be deleted during the next boot, which is fine because the binaries are now installed on the Pi. Run the server with the following command and the AirPlay icon will show up on your Apple devices that play music:

```
# shairport -a 'ShairPortPi'
```

This will block the console and you will see some messages pop up from time to time about packets, but that is normal.

Using alsamixer

You can log in to another SSH console and control the volume by using alsamixer. The Pi only has one output by default, and pressing the up or down keys will make it louder or quieter.

```
# alsamixer
```

Installing RaspBMC

In this section, we will burn a new image on the SD card. You can either use another SD card or wipe the one you were using until now.

We can use a UI installer made for Windows, Mac OS X, and Linux. Open your browser and navigate to `http://www.raspbmc.com/download`. Select the UI download link and run it on your computer. This burns the network image to SD. What that means is on the first boot, it will download the latest files, so you need to have a network connected with Internet access.

The UI installer also gives us an option to install RaspBMC on a USB drive instead of an SD card. If you select this option, you will need a USB storage device connected during the first boot. You will also need a keyboard.

The first boot will take about 20 minutes to completely install RaspBMC. It will reboot and the installation will be complete.

Enabling other codecs

RaspBMC uses OMXPlayer, which is maintained by XBMC. As we know, OMXPlayer uses the Pi's hardware to decode video.

Navigate to **Programs | Raspbmc Settings | System Configuration | Advanced System Settings**. Enter your keys and reboot the system.

If you have MPEG-2 or VC1 encoded files, you can simply purchase the licenses and enter the codes using the XBMC interface. The H264 codec is already licensed in the price of the Pi.

Configuring RaspBMC

RaspBMC can be mostly configured from the user interface, but you can still log in to SSH to install your own services or tweak some advanced settings. As of writing this book, Frodo was the stable version, but any newer versions should be quite similar.

Wireless

You can use RaspBMC to configure your wireless networks using an add-on.

Simply navigate to **System | Add-ons | Get Add-ons | XBMC.org Add-ons | Program Add-ons | Network-Manager | Install**.

Plug in your USB wireless adapter and then go to **Programs | Network-Manager**. You should always plug in a USB wireless adapter before opening that menu.

Media sources

You can start to add the locations of the files that are stored on the USB drive. To do so, navigate to the following options:

Video | Files | Add Videos

Music | Files and Pictures | Files

To add network shares, you can click on **Add Network Location**. To access Samba or NFS, you need to type in a specific prefix to specify this protocol.

```
smb://192.168.0.1/movies
nfs://192.168.0.2/music
```

Using Add-ons

Before Version 10 of XBMC, there was no single repository for add-ons. Thankfully there is one now, and you can find some great plugins there.

Navigate to **System | Add-ons | Get Add-ons**. The following are some popular add-ons that we should give a try:

- **Last.fm** (navigate to **Add-ons | Album Information**): This is an online database for music files. It will search your music files and download extra information about the albums and songs.
- **YouTube** (navigate to **Add-ons | Video Add-ons**): This is used to share user-created video content, and it works really great on RaspBMC. It also allows you to pair with your smartphone to be used as a remote control.
- **XBMCFlicks** (navigate to **Add-ons | Video Add-ons**): This plugin enables you to watch media from Netflix.
- **MCERemote** (navigate to **Add-ons | Video Add-ons**): If you own a Microsoft Media Center remote, this plugin makes using the remote extremely easy.
- **Grooveshark** (navigate to **Add-ons | Music Add-ons**): This is a great plugin to stream music without any hassle.

But wait, there's more! We can now install a third-party repository that has even more plugins. Download the following ZIP file to a USB drive or a network shared drive from the following link:

```
http://code.google.com/p/bluecop-xbmc-repo/downloads/
detail?name=repository.bluecop.xbmc-plugins.zip&can=2&q=
```

Go to **System | Add-ons | Install From ZIP**. Now go back to the **Add-ons** menu and you will see a new repository by the name of **bluecop**. This repository is most popular for Hulu and Amazon plugins.

AirPlay

As an Apple user, you will appreciate that XBMC can be used as a target to stream your media to.

Navigate to **System | Settings | Network | Allow AirPlay**. You will now find the AirPlay icon on your Apple devices.

Enabling CEC

CEC (**Consumer Electronics Control**) is a standardized protocol used in new televisions, media players, and sound equipment. You need to have everything connected using HDMI cables, but because it's not a general standard, not all remote controllers or systems allow these functions.

If you have a smart TV, you should be able to at least control the volume on the Pi and navigate XBMC using the arrow buttons. More support is added every day, but sometimes there are hardware limitations that prevent certain buttons or functions from working.

The Raspberry Pi GPU supports all the CEC functions, so if something is not working, it is most likely your master device. Some home entertainment equipment can be upgraded with newer firmware that might resolve some of these issues.

To do so, navigate to **System | Settings | System | Input Devices | Peripherals | CEC adapter**.

Performance optimization

You might find that performance gets a bit sluggish at times, and you can tweak XBMC at the cost of visual effects to improve the speed.

Change the skin

You can download a skin called Quartz Skin from `https://github.com/pecinko/quartz/archive/master.zip` onto a USB drive.

Now, go to **Add-ons | Settings | Add from ZIP**. This prompts you to change the skin by navigating to **Settings | Appearance** | **Settings | Quartz Skin**. This is a very lightweight skin to improve interface performance, but it still looks good.

Overclocking

It is not a good idea to overclock. If you are running RaspBMC off an SD card and using WiFi, you might experience unstable behavior or corruption.

Programs | Raspbmc Settings | System Configuration

Change the **System Performance Profile** to **fast** and reboot the system.

NFS versus Samba

Using Samba shares on the Pi causes a higher CPU overhead. If your storage server can support NFS shares, it is worth the time to enable or configure NFS instead of Samba.

Summary

In this chapter, we learned how to play audio, make slideshows, and watch videos from the console. We also explored RaspBMC, a very mature media center with a lot of hardware support. We covered the following topics in this chapter:

- Using fbi to make a slideshow
- Playing movies using hardware decoding with OMXPlayer
- Creating an AirPlayer server and playing WAV and MP3 files from the console
- Installing RaspBMC and configuring add-ons and also some performance tips

Index

Symbols

A

B

G

H

I

J

K

L

S

T

U

V

W

X

Y

Thank you for buying

Raspberry Pi Server Essentials

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: `www.packtpub.com`.

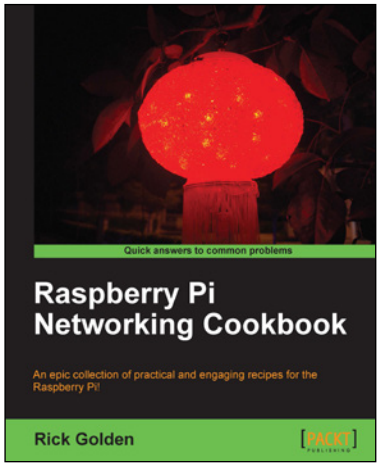
About Packt Open Source

In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licenses, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to `author@packtpub.com`. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

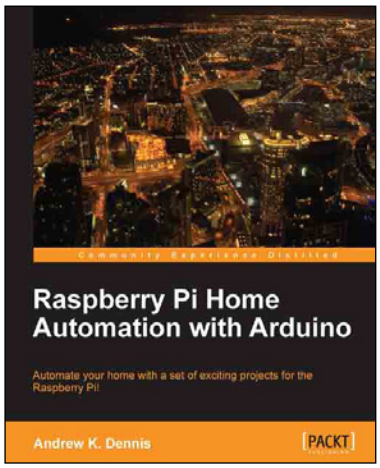


Raspberry Pi Networking Cookbook

ISBN: 978-1-84969-460-5 Paperback: 204 pages

An epic collection of practical and engaging recipes for the Raspberry Pi!

1. Learn how to install, administer, and maintain your Raspberry Pi
2. Create a network fileserver for sharing documents, music, and videos
3. Host a web portal, collaboration wiki, or even your own wireless access point
4. Connect to your desktop remotely, with minimum hassle



Raspberry Pi Home Automation with Arduino

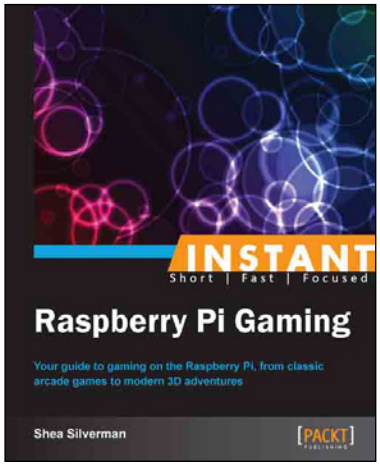
ISBN: 978-1-84969-586-2 Paperback: 176 pages

Automate your home with a set of exciting projects for the Raspberry Pi!

1. Learn how to dynamically adjust your living environment with detailed step-by-step examples
2. Discover how you can utilize the combined power of the Raspberry Pi and Arduino for your own projects
3. Revolutionize the way you interact with your home on a daily basis

.

[PACKT] open source
community experience distilled
PUBLISHING



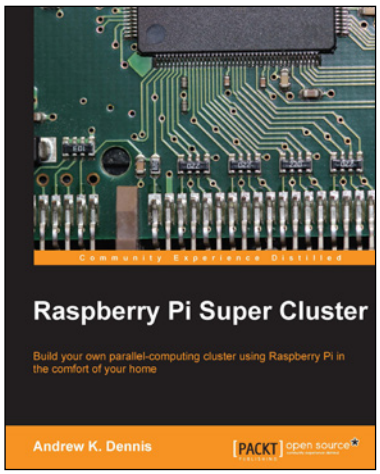
Instant Raspberry Pi Gaming

ISBN: 978-1-78328-323-1 Paperback: 60 pages

Your guide to gaming on the Raspberry Pi, from classic arcade games to modern 3D adventures

1. Learn something new in an Instant! A short, fast, focused guide delivering immediate results
2. Play classic and modern video games on your new Raspberry Pi computer
3. Learn how to use the Raspberry Pi app store
4. Written in an easy-to-follow, step-by-step manner that will have you gaming in no time

---



Raspberry Pi Super Cluster

ISBN: 978-1-78328-619-5 Paperback: 126 pages

Build your own parallel-computing cluster using Raspberry Pi in the comfort of your home

1. Learn about parallel computing by building your own system using Raspberry Pi
2. Build a two-node parallel computing cluster
3. Integrate Raspberry Pi with Hadoop to build your own super cluster