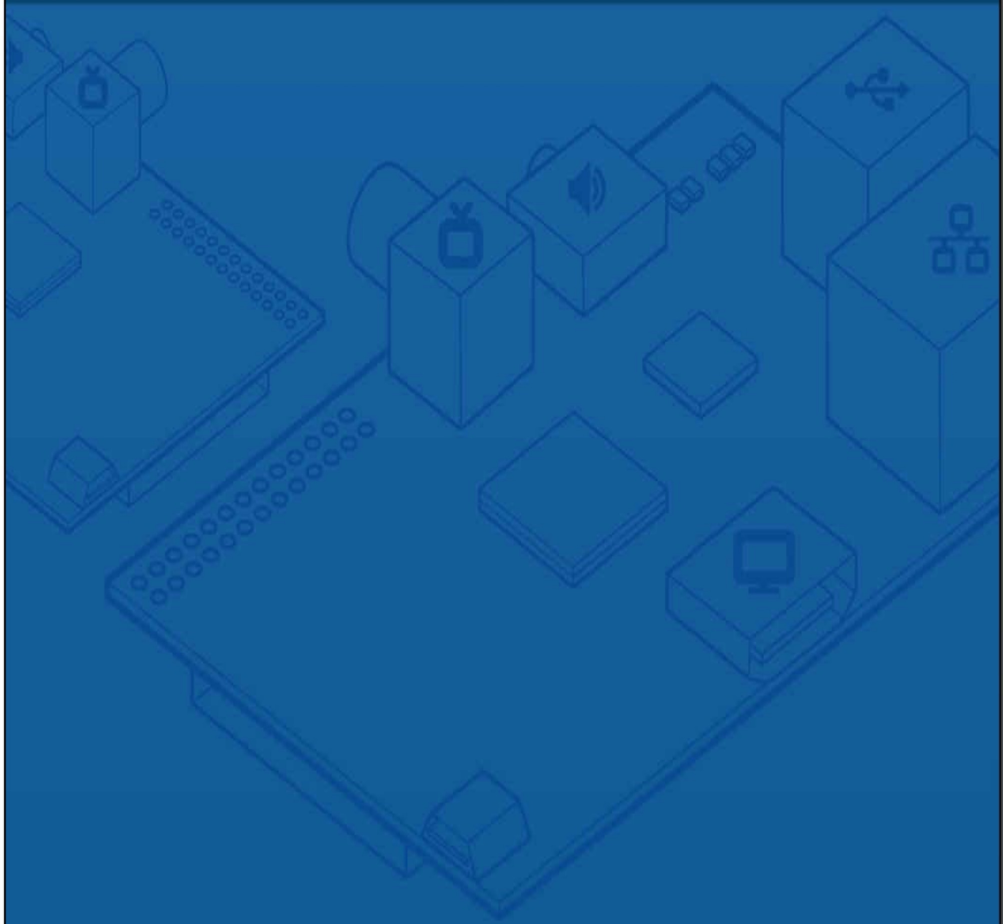


VILROS

Raspberry PI *User's Guide*

Raspberry PI User’s Guide

ISBN: 978-0-692-31936-9

Conventions Used in this Book

The following table describes the text conventions used in this book.

Convention	**Meaning**
Italic	Text that appears in italics refers to file names, variable and function names, or other code. Within the context of giving instruction, italic text should be typed exactly as shown.
Bold	Within the context of giving instruction, items in bold text are user interface elements, such as key strokes, menu items, or button labels.
Monospace font	A monospace font is used for shell commands and Python code that should be typed in on the keyboard.

Contents

1 – Getting Started

1.1 Introduction

The Raspberry Pi is a small computer, a *very* small computer.

It consists of mostly the same parts as a standard desktop computer or laptop. A central processing unit (CPU) acts as a brain, random access memory (RAM) and long-term storage devices are used to hold data, a video display shows you what is happening, and you interact with all of this using mice, keyboards, joysticks, and other universal serial bus (USB) devices. The Pi may be less powerful than your Windows PC or Macintosh, but it is still impressive that it fits all of this on a board only slightly larger than a credit card.

The original goal of the UK-based Raspberry Pi Foundation was to create a device that would address their perception of falling standards in the teaching of computer science. As computers have evolved, they have become more difficult to write software for at a "low-level" – with close interaction between hardware and software. And as they have become more integrated in our daily lives, the consequences of breaking your computer by experimenting have become more severe…and expensive.

So, unlike conventional systems, the Pi is a machine that is designed to be played with and used for experiments. Its diminutive form factor and relatively low cost mean that you can do what you want with it, and this attracts far more diverse groups of users than just students and teachers.

It is suitable for a wide range of applications, including:

Using the Pi for General Computing
You can install a variety of operating systems on the Pi, and many of these have full desktop environments. With support for external hardware devices, Internet connections, and downloading and installing software, you can use the Raspberry Pi as a regular computer. It can do just about everything your main system can do…only a little slower.

Playing Games
The range of games available on the Raspberry Pi's operating systems is extensive. But you can also install "emulators", which are pieces of software that allow programs from other machines to be run on the Pi. This gives you access to titles on classic machines such as the Atari 2600, Nintendo Entertainment System, Sega Genesis, Sharp X1, MSX, Panasonic 3DO, and many more. All Raspberry Pi models support HDMI video output and USB game controllers, making the Pi a popular choice for fans of retro computing and retro video games.

Playing Movie Files and Music on Your TV
Put the Pi in a case, connect it to a television using an HDMI cable, and install XBMC. Then you have a capable media center that can load movie files from the SD card, USB hard drives, or across your local network.

Providing Network Services
As a small, standalone device that connects to the Internet and local networks, the Raspberry Pi makes an excellent server. You can use it to serve files and webpages, answer domain name system (DNS) requests, share hardware devices (for example, printers) across a network, and almost anything else you might need a server for.

Setting up Development Environments
The simple nature of the Pi, and its support for many different programming languages, make it an ideal system to use when learning how to program. The Pi contains an ARM processor, and these are popularly used in cellphones, tablets, games consoles, and in the computer-controlled equipment used in manufacturing. The ability to program one is a skill that is in high demand.

Building Controller Boards and Interaction with Electronics Its small size, low cost, and ease of programming make the Raspberry Pi very useful in “embedded” applications. In these types of project, the Pi is used along with other electronics circuits to create everything from 3D printers, to home automation systems and robots.

1.2 Technical Specifications

The Model B+ is a revised version of the Model B that will eventually replace the earlier model. It has more general purpose input/output (GPIO) pins than earlier models, and twice as many USB sockets.

Model B+
CPU Broadcom microprocessor running at 700 MHz (ARMv6 architecture)
GPU Broadcom VideoCore IV, 250 MHz

OpenGL ES 2.0
MPEG-2 and VC-1, 1080p h.264/MPEG-4 AVC decoder

Memory 512 MB Video Output Video Output pole 3.5 mm jack, LCD panels through DSI. Audio Output 4-pole 3.5 mm jack, HDMI, I2S USB Ports 4 (2 dual sockets) Input/Output 17 x GPIO, UART, I2C, SPI Networking 10/100 Mb/s Ethernet, USB devices Storage microSD card slot Power 5V through micro-USB socket or GPIO header

The diagram below shows the available connectors on the Raspberry Pi Model B+.

Figure 1. The Raspberry Pi and its connectors
1 General purpose input/output (GPIO) header – these pins can be controlled from software.

2 Universal serial bus (USB) socket for attaching peripheral devices such as mice, keyboards, or memory sticks. This is a dual socket and two USB devices can be connected at the same time.

3 10/100 Mb/s Ethernet (RJ45) socket for connecting to a network router.

4 4-pole 3.5 mm output jack for audio and video.
5 Camera serial interface (CSI).
6 High-definition multimedia interface (HDMI) video output.
7 Power in via micro-USB socket.

8 Display serial interface (DSI) flexible flat cable connector for liquid crystal displays (LCDs).
9 microSD socket. The card socket is attached to the underside of the board.

1.3 Basic Setup

To start using your Pi, you need to connect it to: 1. Power

2. An SD card with an operating system or “bootable” program installed
Without these, the Pi will do nothing, not even output a video signal. For the initial configuration, you may also need to connect:

1. A display (using HDMI or composite video)
2. A USB keyboard
3. An Ethernet cable or USB Wi-Fi “dongle”

The display, and any USB devices you may connect, can be removed when they are not in use. For example, a Pi acting as a file/print server may only accept input from computers on the network and does not always need a keyboard or screen.

Fitting a Heat Sink
When the components in a computer system work hard, they generate heat. And above a certain level, this heat can reduce the lifespan of the components or even break them altogether. A *heat sink* is a carefully designed block of metal that takes the heat away from the electronic component and then passes it into the air surrounding the device.

There are three chips on a Raspberry Pi that can get very hot if the device is working hard: the central processing unit (1), the chip that controls the Ethernet and USB ports (2), and the power regulator (3).

To install a heat sink:
1. Unplug the Pi and leave it to cool before attempting to handle the device.
2. On the bottom of the heat sink, peel away the plastic backing that covers the adhesive.

3. Press the heat sink down firmly and directly onto the chip. Hold the pressure for a few seconds to allow the adhesive to work.

If you buy heat sinks for your Pi, only use the thermal adhesive that they arrive with; never use any other type of adhesive or sticky plastic to install a heat sink on a Raspberry Pi. The adhesive must be a special compound so that it effectively transfers heat from the chip on the Pi to the metal of the heat sink.

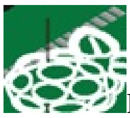


Figure 2. The three heat-out points on a Pi (left); and installing a heat sink on the CPU (right)

Connecting Power
The Raspberry Pi requires a power supply of 5 V that can provide at least 700 mA of current. Before connecting the Pi and turning it on, you should check the rating of the power supply carefully.

Many cellphone chargers will work, but some supply less current than the Pi needs. Using inadequate power supplies, or even powering the Pi from the USB port of another computer, is not recommended as the lack of current may make the Pi unstable. You should certainly avoid doing this if you need to connect any other devices to the Pi.

Power can be fed in to the Pi through the micro-USB socket or, if you have a suitable connector, through the general purpose input output pins. However, you should be aware that providing power through the GPIO header pins bypasses the on-board protection circuitry that is designed to prevent damage to the device. For this reason, it should only be attempted by people who are experienced in building electronic circuits.

It is worth noting that the Raspberry Pi does not have an on/off switch. Some operating systems can power down the device or put it into standby mode but when you want to turn it off, you will often have to remove the power supply or switch it off at the wall socket.

There are two holes on the printed circuit board (PCB) of the Model B+. These holes are labelled “RUN” and you can solder a switch across these two connections to create a reset button.

Connecting a Display

HDMI offers a high-quality video and audio signal, and is the preferred way of connecting all models of Raspberry Pi to a modern television. To connect a high-definition television:

• Plug one end of an HDMI cable into the Raspberry Pi’s HDMI socket, and the other end into an HDMI input on your TV.

If your display does not support HDMI, you can connect the composite video and audio outputs to the auxiliary A/V input of most televisions. These connections are colored yellow, red, and most televisions. These connections are colored yellow, red, and pole jack on one end, and three RCA plugs on the other (one yellow, one red, and one white). To use these cables:

1. Plug the 4-pole 3.5 mm jack plug into the 3.5 mm jack socket on the Raspberry Pi Model B+.
2. Connect the yellow composite video plug to the yellow video input socket on your TV.
3. Connect the red RCA plug to the red audio socket on your TV.
4. Connect the white RCA plug to the white audio socket on your TV.

If your TV does not have the yellow, red, and white auxiliary inputs, you can plug the composite outputs into a SCART adapter and connect that to your TV instead.

When using the composite video, you always have the option of connecting the video to a television and the audio output (the red and white connectors) to other devices, such as headphones, powered speakers, or amplifiers.

To use a computer monitor as the Pi’s display, you will need an HDMI to VGA, or HDMI to DVI adaptor. However, when connecting to a monitor using an HDMI to VGA/DVI adapter, you will initially be unable to connect speakers to the audio connector. When the HDMI cable is connected, the composite video and audio output ports are turned off. This can be changed once you have an operating system running on the Pi.

Although not covered in this book, all Raspberry Pi models support DSI for connecting LCD panels.
Figure 3. Connecting a display to the Model B+

Connecting USB Devices

With most operating systems that you run on the Pi, human interface devices (HIDs) that connect using USB (such as mice, keyboards, and game controllers), and storage devices (such as USB memory sticks and hard drives) will work without any problems or installation processes. You may need to install *drivers* – pieces of software written to pass messages between the operating system and the hardware device – to use more complicated peripherals like soundcards.

To connect a USB device, insert its USB connector into an available socket on the Raspberry Pi. You can usually do this safely whether or not the Pi is turned on and running.

The Raspberry Pi can only supply a limited amount of power to USB devices. It is recommended that you do not connect any devices that draw over 100 mA. To use more power-hungry devices, you can use a powered USB *hub* – a device that allows multiple USB peripherals to be connected to a single port, and that has its own power supply. This also allows you attach more devices to the Pi.

Figure 4. Connecting the Pi to a 4-port USB hub

Connecting to a Network

To make a wired network connection between the Pi and a network router:

1. Plug one end of a CAT5 or CAT6 Ethernet cable with RJ45 connectors into the Ethernet socket on the Raspberry Pi.

2. Plug the other end of the cable into your network router.
The actual network connection is made by the software or operating system running on the Pi.

In theory, Wi-Fi is a faster technology than the 10/100 Mb/s Ethernet circuit that is built-in to Model B+. In practice, however, this is not always the case. As the Pi cannot process data as quickly as a brand new desktop computer, the difference in speed between Ethernet and Wi-Fi is less noticeable and you should use whichever is most convenient for you.

Not all Wi-Fi and Ethernet adaptors are compatible with the Raspberry Pi. Before buying, check that the adaptor is known to be working by looking at the list at
http://elinux.org/RPi_USB_Wi-Fi_Adapters

1.4 SD Cards

In the next chapter, you will learn about Raspberry Pi operating systems (in particularly, Raspbian Linux) and how to install them onto an SD card. This will complete the set-up of the Pi.

Unlike conventional desktop computer systems, the Pi does not have a hard drive from which to load an operating system. Instead, it uses the card socket to load files from a memory card.

When the Pi is first turned on: 1. The main ARM processor and SDRAM (memory) are disabled.

2. The graphics processing unit (GPU) loads the “firststage bootloader” that is built-in to the Pi, and this contains all of the code necessary to work with memory cards.

3. The first-stage bootloader enables SDRAM, detects the presence of a memory card, and loads the “second-stage bootloader” from it.

4. The second-stage bootloader loads the operating system or bootable program from the card.

Inserting and Removing SD Cards
It is likely that you will need a memory card with at least 4 GB capacity if you are intending to run an operating system on the Pi. 4 GB SDHC microSD cards are widely available and generally very cost-effective.

To insert an SD card:
1. Ensure the Pi is unplugged.
2. Locate the SD card socket on the underside of the Pi’s board.
3. When looking from above, the SD card’s contacts should be facing up.
4. Gently, but firmly, push the SD card into the socket until it clicks into place.

Figure 5. Inserting SD cards into the Raspberry Pi

To remove the SD card:
• First push the microSD further into the device until it clicks. Then pull the card out.

When inserting and removing SD cards in other computers, it is not always necessary to turn off the device. However, as the memory card contains its operating system, the Pi may access it at any time. Removing the card while the Pi is accessing it can corrupt data and, in extreme cases, may stop the card working at all.

2 – Introducing Raspbian

2.1 Raspbian and Linux

An operating system (OS) is a unique type of application that you run on your computer. It is an environment in which many other applications can run at the same time, with a consistent user interface and sharing the same resources. Microsoft Windows and Apple's Mac OS X are probably the two most well-known operating systems, but there are others.

Linux is one of a small group of operating systems that are "free". It usually doesn't cost anything to use and people can modify the OS, repackage it with other software, distribute their version, and generally do what they want with it. Because of this, you can find Linux running on most types of computer – from large servers used by corporations such as Google, to small devices like the Raspberry Pi. A bundle of the Linux core with other applications (such as desktop environments, file managers, and web browsers) is called a "distribution" (or "distro").

Raspbian is a Linux distribution that is based on Debian, another popular version of Linux. It is designed for the Raspberry Pi and is the OS recommended by the Raspberry Pi Foundation. Although different Linux distributions can usually all run the same applications, this book is focused on using Raspbian and it may be helpful for you to run this OS while you are learning. Once you are familiar with Raspbian, you will find that you are able to use other varieties of Linux without much help.

2.2 Installation of Raspbian with NOOBS

New out of Box Software (NOOBS) is a tool that you can run on your Raspberry Pi and it will help you install an OS. It stays on the SD card, even after the OS installed, and can also be used to edit the Pi's main configuration file or replace the installed OS if problems occur. The SD card supplied with your Pi already contains NOOBS.

Installing NOOBS to an SD Card
There are two versions of NOOBS available: NOOBS and NOOBS Lite. NOOBS Lite is quicker to download, but does not come with the installation files for Raspbian. Instead, your Pi connects to the Internet to download the relevant files when you start the installation process. It does not support USB Wi-Fi adapters, and this means that you will need an Ethernet cable plugged in to your Raspberry Pi to use NOOBS Lite.

The full version of the latest version of NOOBS only comes with the files needed to install Raspbian. To install a different OS, you need an Internet connection.

The space that NOOBS occupies on the SD card is not available for use by the OS that you are installing. Even a 4 GB SD card will sometimes not be large enough to hold NOOBS and an OS, and still have adequate space for your files and programs.

If you cannot use NOOBS, you can install an OS "disk image" to your SD card. This is described in section 2.3 Installation of Raspbian using a Disk Image on page 20.

The process for installing NOOBS to an SD card is the same whether you are using the full NOOBS package, or NOOBS Lite. On Windows (8/7/Vista/XP):
1. Insert the SD card into a suitable card socket or USB card reader.
2. Press the **Windows logo key + R**. Type *explorer* and press **Enter**.

3. In the **Windows File Explorer** window, right-click the card device (usually labelled "SDHC" or "Removable Disk") and then click **Format**.

4. In the **File system** list, click **FAT32**.
5. In the **Allocation unit size** list, click **Default allocation size**.

6. Click **Start**.
7. Download one of the NOOBS packages from *http://www.raspberrypi.org/downloads/*

8. Find the *.zip* file you downloaded and right-click it. Point to **Open with** and then click **Windows Explorer**.

9. Press **Ctrl + A**, to select all of the files in the *.zip* archive.
10. Press **Ctrl + C**, to copy the selected files into memory.
11. In the **Windows File Explorer** window, right-click the card device.
12. Click **Paste** to copy the files to the SD card. 13. In the **Windows File Explorer** window, right-click the card device.

14. Click **Eject**.
On Mac OS X:

1. Insert the SD card into a suitable card socket or USB card reader.

2. On the dock, click **Finder**.

3. On the sidebar, click **Applications**.
4. Click **Utilities**, and then double-click **Disk Utility**.

5. In the left panel, click the SD card (usually labelled “NO NAME” if the card was not formatted with a volume name).

6. On the **Erase** tab, from the **Volume Format** list, click **MS-DOS File System**.
7. Click **Erase**.
8. Download one of the NOOBS packages from *http:/ /www.raspberrypi.org/downloads/*
9. In **Finder**, locate the .zip file you downloaded.

10. Double-click the NOOBS zip file and it will unzip to a new folder. For example, a folder named *NOOBS_lite_v1_3_9*.[1]

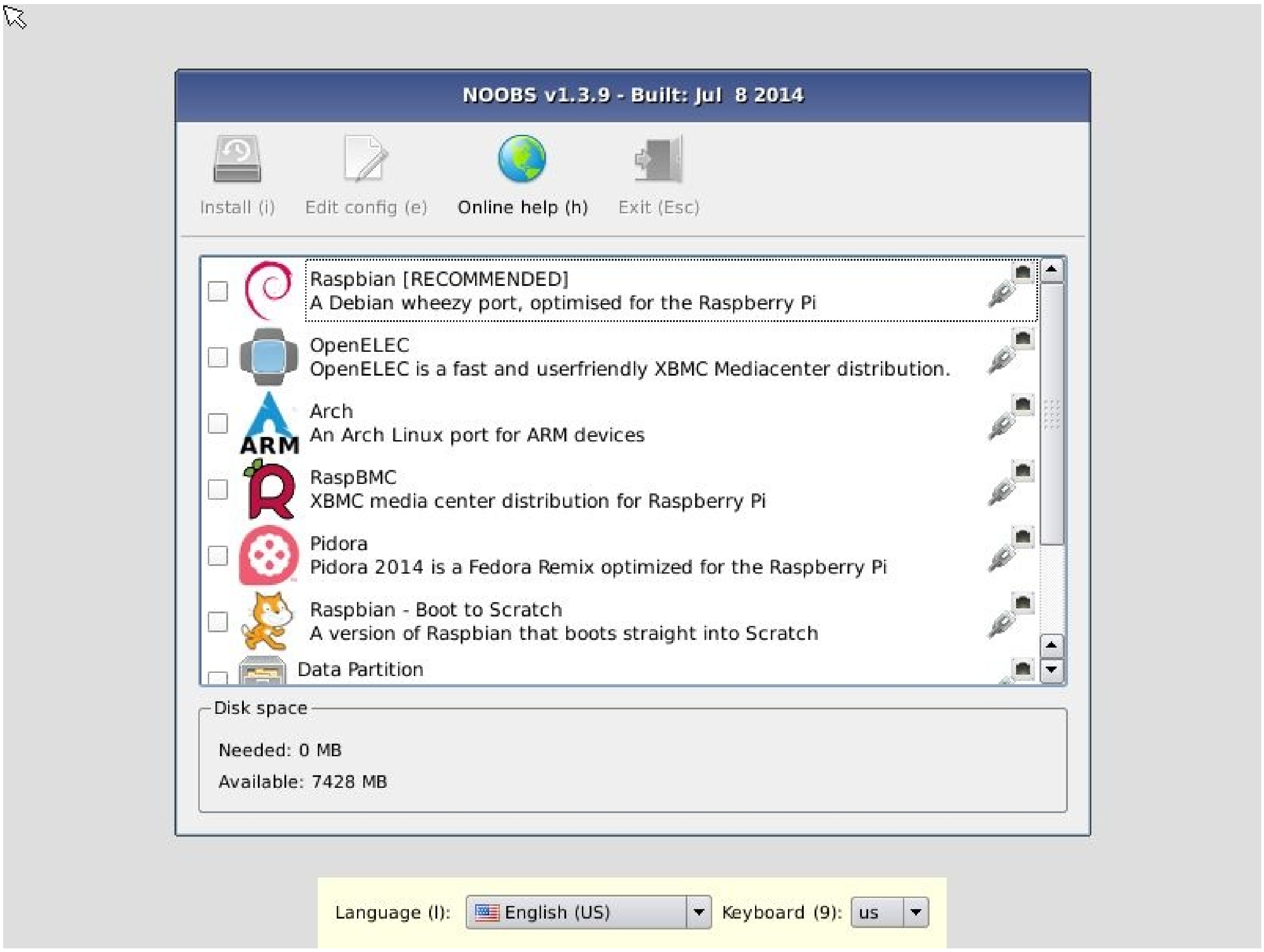
11. Double-click the folder to open it.
12. Press **Cmd + A** to select all of the files.
13. Press **Cmd + C** to copy the files into memory.

14. In the left panel, click the SD card (usually labelled “NO NAME” if the card was not formatted with a volume name).

15. Press **Cmd + V** to copy the files to the SD card. 16. In the left panel, click the **Eject** icon next to the name of the SD card.

Installing Raspbian
To install Raspbian, you will need to connect a keyboard, mouse, and display to your Pi.

First, ensure the Pi is completely off and unplugged, and then insert the SD card into the Pi’s card socket. Reconnect the power to your Pi. The Pi will start and will load the NOOBS tool. When everything is ready, you should see a window similar to Figure 1.


NOOBS v1.3.9 - Built: Jul 8 2014

Raspbian [RECOMMENDED]
A Debian wheezy port, optimised for the Raspberry Pi

OpenELEC
OpenELEC is a fast and userfriendly XBMC Mediacenter distribution.

Arch
An Arch Linux port for ARM devices

RaspBMC
XBMC media center distribution for Raspberry Pi

Pidora
Pidora 2014 is a Fedora Remix optimized for the Raspberry Pi

Raspbian - Boot to Scratch
A version of Raspbian that boots straight into Scratch

Data Partition

Disk space

Needed: 0 MB

Available: 7428 MB

Language (l): English (US)   Keyboard (9): us

Figure 1. NOOBS on a Raspberry Pi

The window shows a list of the operating systems that you can install.

1. If you have archiving software installed on your Mac, the NOOBS file might open with this instead..

To install Raspbian: 1. In the list, click the box next to **Raspbian**.

2. On the toolbar, click **Install**.

When the installation is complete, the Raspberry Pi restarts and loads the raspi-config tool. This helps you to change certain important settings. For more information, see section 2.4 Raspiconfig on page 22.

If you put the SD card back into a Windows PC then the card will appear to have shrunk in capacity. If you want to wipe the card, you can use the command-line tool *diskpart* to destroy the partitions and create a new "primary" partition using all of the available space.

2.3 Installation of Raspbian using a Disk Image

A *disk image* is a special type of file for making a copy of storage devices, such as floppy disks, CDs, hard drives, and memory cards. It not only contains all of the files from the device, it also stores all of the information necessary to recreate the same file structure and physical layout on another device.

Many operating systems that have already been made to work on the Raspberry Pi are available for download as disk images. To write the image to an SD card, you can use a Windows PC, Mac or Linux computer.[1]

Download the Raspbian disk image from *http://www.raspberrypi.org/downloads/* – it is a .zip archive that you need to unzip before it can be used.

To unzip the file on Windows (8/7/Vista/XP):

1. Find the *.zip* file you downloaded and right-click it. Point to **Open with** and then click **Windows Explorer**.

1. Only instructions for Windows and Mac OS X are covered in this guide.

2. Click the *.img* file, and then press **Ctrl + C**.
3. In the left panel, click **Desktop**.
4. Press **Ctrl + V** to copy the *.img* file to your Desktop.

To unzip the file on Mac OS X:

1. On the dock, click **Finder**.
2. Find the *.zip* file you downloaded.
3. Double-click the *.zip* file to unzip it to a new folder.

Users of Windows will need to download and install Win32DiskImager from *http://sourceforge.net/projects/ win32diskimager/* before continuing. Users of Mac OS X should install ApplePi-Baker from *http://www.tweaking4all.com/ hardware/raspberry-pi/macosx-apple-pi-baker/*

To write the Raspbian disk image to an SD card on Windows (8/ 7/Vista/XP):
1. Insert the SD card into a suitable card socket or USB card reader.

2. If you are running Windows XP: on the **Start** menu, under a group named **Image Writer**, click
Win32DiskImager.

3. If you are running a later version of Windows: on the **Start** menu, under a group named **Image Writer**, right-click **Win32DiskImager**. Then click **Run as administrator**. You may be asked to confirm this by a window titled “User Access Control”. Click **Yes**.

4. In Win32DiskImager, under **Image File**, click the folder icon. In the left-hand side of the Window titled “Select a disk image”, click **Desktop**, and then double-click the *.img* file that you unzipped.

5. Under **Device**, ensure that the drive letter shown matches the one that is displayed in Windows Explorer next to your SD card.

6. Click **Write**.
7. When the process is complete, click **Exit**.

8. Press the **Windows logo key + R**. Type *explorer* and press **Enter**.

9. In the Windows File Explorer window, right-click the card device (usually labelled "SDHC" or
"Removable Disk") and then click **Eject**.

On Mac OS X:
1. Insert the SD card into a suitable card socket or USB card reader.

2. On the dock, click **Finder**.
3. On the sidebar, click **Applications**.
4. In the right panel, double-click **ApplePi-Baker**.

5. If multiple SD cards are listed under **Pi-Crust: Possible SD-Cards**, click the one you want to write to.

6. Under **Pi-Ingredients: IMG Recipe**, click **IMG to SD-Card**.
7. Locate the *.img* file that you unzipped, then click **Open**.
8. Enter the administrator password for your Mac.

9. When the process is complete, click the **Close** button. Then in **Finder**, click the **Eject** button next to the SD card.

Ensure the Pi is completely off, and then insert the SD card into the Pi's memory card socket. Reconnect the power to your Pi.

2.4 Raspi-config

When Raspbian starts for the first time, it runs a tool called raspiconfig. You can use this to change several configuration options that affect how Raspbian operates. Raspi-config is a commandline application and is controlled using the keyboard:
• Use the **Up** and **Down Arrow** keys to highlight menu

items.
• Press **Enter** to activate the highlighted item.

• Press the **Tab** key to move the highlight down to the two options at the bottom of the window – **<Select>** and **<Finish>** – and then use the **Left** and **Right Arrow** keys to choose between those two options.

• To move back to the main menu, press the **Tab** key again.
When you have finished editing the settings, highlight the option **<Finish>** and press **Enter**.

When Raspbian's graphical desktop is running, you can access the raspi-config tool at any time by double-clicking LXTerminal on your desktop. Then enter the following command and press Enter: sudo raspi-config

Exploring the Settings in Raspi-config

The table below summarizes the options available from the main menu in raspi-config:

Menu Option Description
Expand Filesystem Make the full capacity of the SD card available to the operating system – if is not already available.
Change User Password
Change the password for the default user (*pi*).

Enable Boot to Desktop/Scratch Raspbian can automatically load different applications when it starts. This setting changes that.

Internationalisation Options
Configure language, keyboard and culture settings (such as date format and time zone).

Enable Camera If you have a Raspberry Pi Camera module, you need to enable it using this menu option.
Add to Rastrack

Rastrack (rastrack.co.uk) shows the location of Raspberry Pi devices around the world. Use this option to add your Pi to the map.

Overclock Overclocking your Pi makes it run faster than the

manufacturer designed it to. However, this can also cause the system to overheat or become less stable.

Advanced Options See below. About 'raspi-config' Display information about the raspi-config tool.

Selecting **Advanced Options** from the menu takes you to another menu:

Menu Option Description
Overscan The display in many older televisions is slightly

larger than the visible area. If your Pi's display runs off the edge of the screen, you can enable black borders (overscan) to ensure that you can see all of the picture.

Hostname Change the name given to the Pi when it is

connected to a local network. This can be useful when searching for the device, or when you have several Raspberry Pi devices.

Memory Split The Pi's memory is shared between the CPU and GPU. You can change how much memory is allocated to the GPU using this setting.
SSH Enable or disable SSH access. SPI When connecting certain SPI devices, it can be

useful to load the SPI kernel module when the Pi starts. Use this setting to change whether the module is loaded automatically.

Audio

Used to send the audio signal to the audio output connector, even when using an HDMI cable for video.

Update Connects to the Internet and updates the raspiconfig tool to the latest version.

For some users, the default settings are ok. However, there are three settings in particular that many users might want to change at this stage:

Changing the Default Password

When Raspbian is installed, it creates a new user called *pi*. The default password for this user is *raspberry*.

To change this:

1. Press the **Down Arrow** key to highlight **Change User Password**, and then press **Enter**.
2. Press **Enter** again to select **<OK>**.
3. Type a new password and press **Enter**.
4. To confirm the password, type it again and press **Enter**.
5. Press **Enter** to select **<OK>**.

If you connect your Pi to the Internet and are unsure whether your network router allows incoming connections then it is a good idea to change this password so that other people cannot login to your device.

Expanding the File System

If you installed Raspbian to the SD card using a disk image, it is possible that the OS cannot use the full capacity of your SD card.

To fix this:

1. Use the **Up** and **Down Arrow** keys to highlight **Expand Filesystem** and then press Enter.
2. When the process is complete, press **Enter** to select **<OK>**.
3. Press the **Tab** key and then the **Right Arrow** key to highlight **<Finish>**. Press **Enter**, and when you are prompted to reboot the Pi, select **<Yes>**.

Changing the Default Boot Sequence

The default setting is for Raspbian to finish loading and place you at the command line.

From the raspi-config tool, you can instruct Raspbian to load the graphical environment every time the Raspberry Pi starts up. To do this:

1. Use the **Up** and **Down Arrow** keys to highlight **Enable Boot to Desktop/Scratch**, and then press **Enter**.
2. Press the **Down Arrow** key to highlight **Desktop Log in as user ‘pi’ at the graphical desktop**, and then press **Enter**.

2.5 Raspbian's Desktop Environment

If you have not changed the setting in raspi-config, the Raspberry Pi will boot into Raspbian's command line. To start the desktop environment:

1. Type *pi* as the username, then press **Enter**.
2. Type your password, then press **Enter**.[1]
3. Type the following command and press **Enter**:

```
startx
```

After a short period of loading, you will see a screen similar to Figure 2.

1. If you have not changed the default password, it is raspberry.

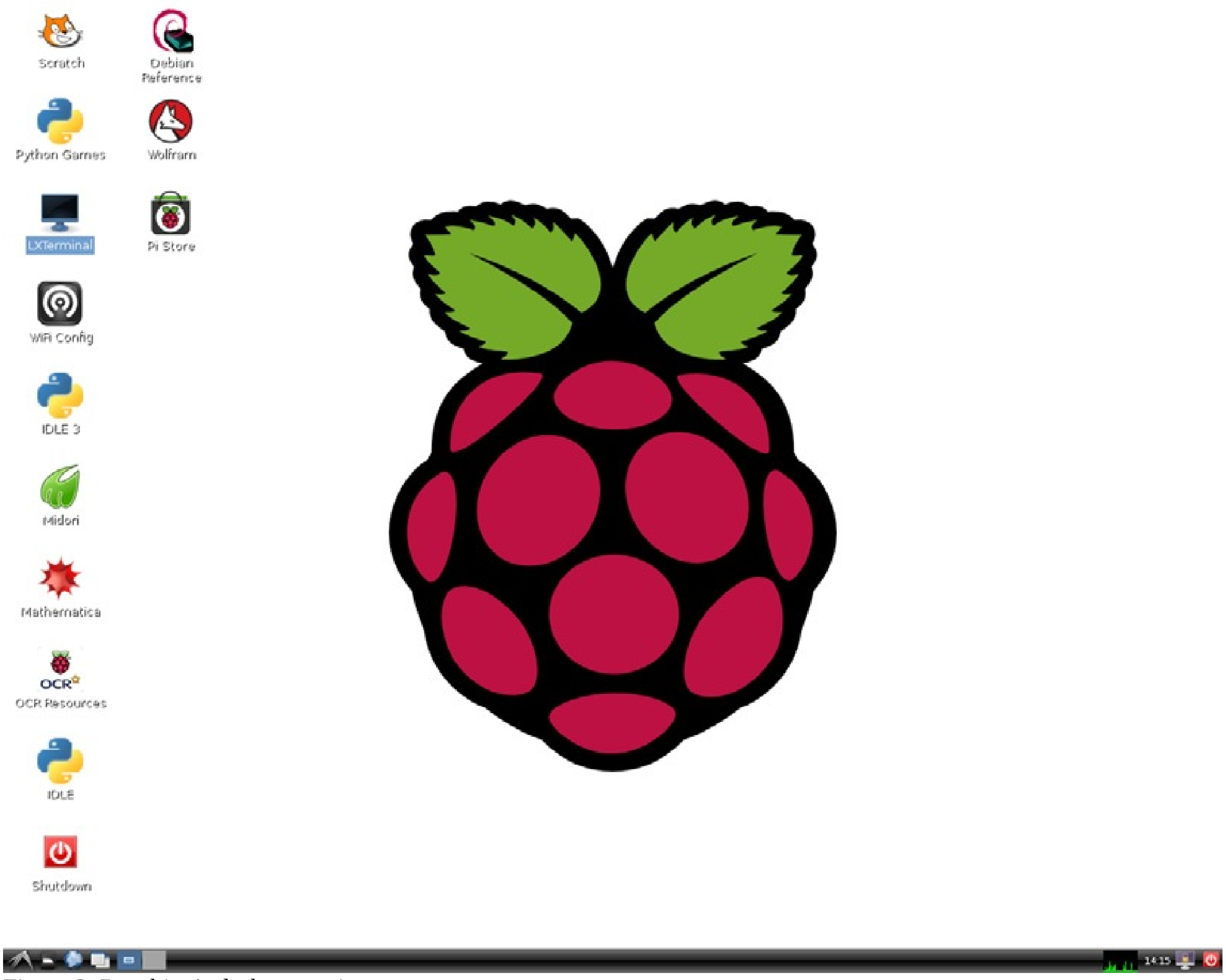


Figure 2. Raspbian's desktop environment

Raspbian's desktop is a customized version of LXDE, which stands for lightweight X11 desktop environment. It is similar to Microsoft Windows and many of the ways that you use it are the same.

The screen is divided into two distinct areas: the desktop, and the LXDE panel.

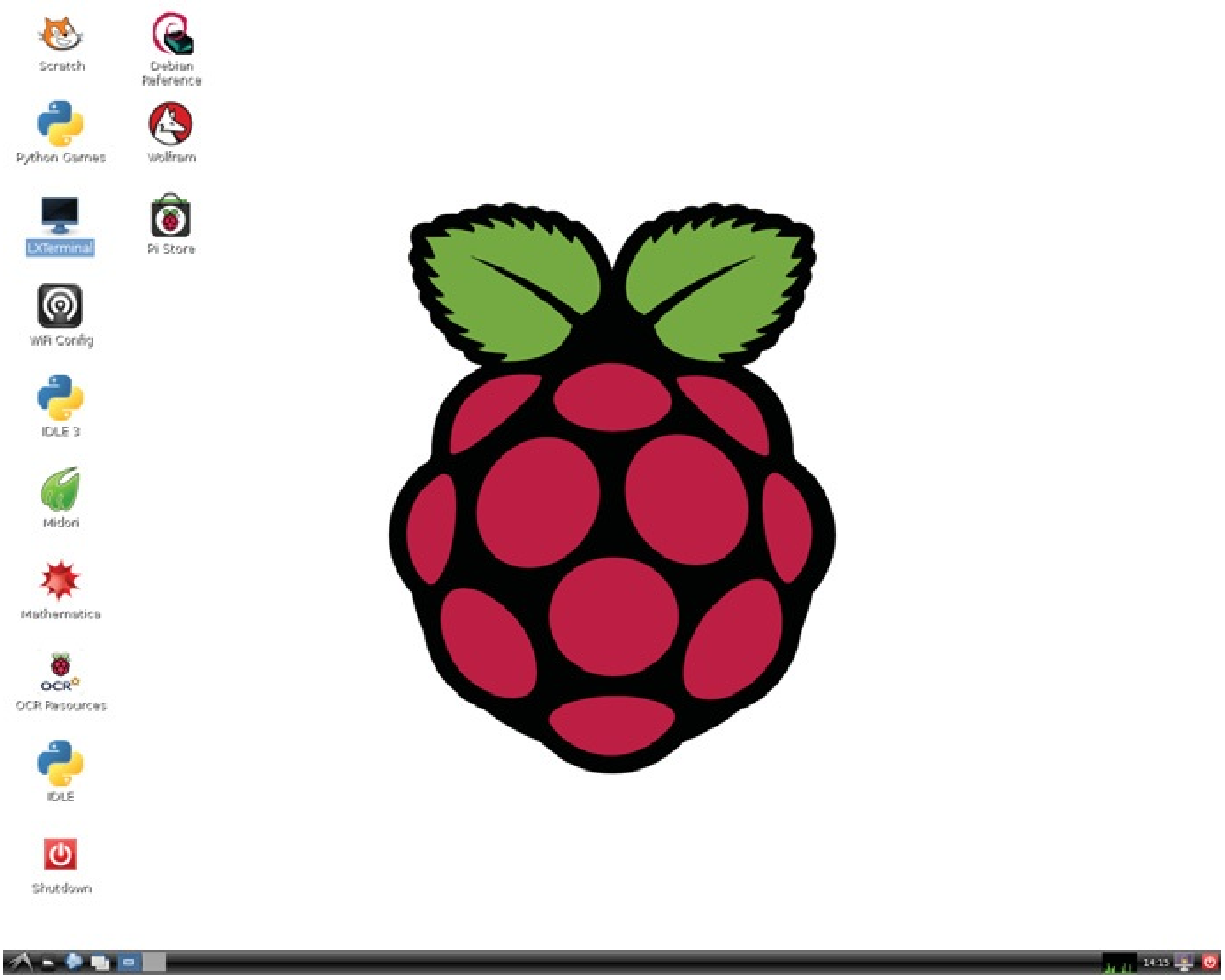


Figure 3. Elements of the Raspbian desktop environment 1 Desktop area. Here you can find files and links (shortcuts) to applications. Double-click the icons to open the file or application that it links to.

2 LXDE panel. Contains the main menu, shortcuts to commonly-used applications, the desktop pager, and the system tray. 3 Menu. The LXDE main menu provides access to many of the system settings, tools, and any applications that you have installed. 4 Desktop Pager. For more information, see Switching the Desktop on page 29.

5 Taskbar. Open applications are shown as an icon and description in this area. If you minimize an application window, you can click the icon in the LXDE panel to restore the window to full view.

6 System Tray. This is usually used by system processes and applications that run in the background, so that the user has some way of interacting with them. By default, the system tray contains the CPU monitor (which shows how busy is the CPU is), the current time, and the application launch bar (another place you can store icons to open commonly-used applications). You can usually double-click the icons in this area to open their window, or rightclick the icon to access the application's menu.

Opening and Closing Applications

On the desktop, and in the menu that you access from the LXDE panel, you can see that many applications are already installed on your system.

There are several ways of running applications in Raspbian, but the three most common are: clicking icons on the desktop, using the menu, and opening an application from the command line.

For example, to open the web browser Midori:
• On the desktop, double-click **Midori**.
• On the **LXDE panel**, click the **Menu** button. Point to **Internet**, and then click **Midori**.

• From a command-line terminal running in the desktop environment, type *midori* and then press **Enter**. If you run applications from the command line, you will return to the prompt when they are finished. With applications that open a window in the graphical environment, you can close them in three ways:

• Press **Alt + F4** on the keyboard.
• Click the **Close** button in the top-right of the window. It looks like an x.
• Click the icon in the top-left of the window, and then click **Close**.

Switching the Desktop
On most Linux desktop environments, you can have two (or more) *workspaces*. A workspace is collection of the windows and files that are currently open. Switching to another workspace hides all of the windows that are currently visible, and displays all of the windows from the other workspace. In Raspbian, workspaces are also called "desktops".

You can switch between these using the two desktop buttons that make up the Desktop Pager.
For example:
1. On the desktop, double-click **Midori**.
2. Drag the window to the top-right corner of the display.
3. On the desktop, double-click **LXTerminal**.

4. On the **LXDE panel**, point to the light gray square. After a few moments, the text "Desktop 2" will appear. Click the button.

5. On the desktop, double-click **Pi Store**.

6. Click the "Desktop 1" button (to the left of the "Desktop 2" button) to switch back to the first desktop.

7. Click the "Desktop 2" button to switch to the second desktop again.
To move a window to another desktop: click the icon in the topleft of a window, point to **Send to desktop**, and then click the name of the desktop.

To increase the number of desktops that you can use:

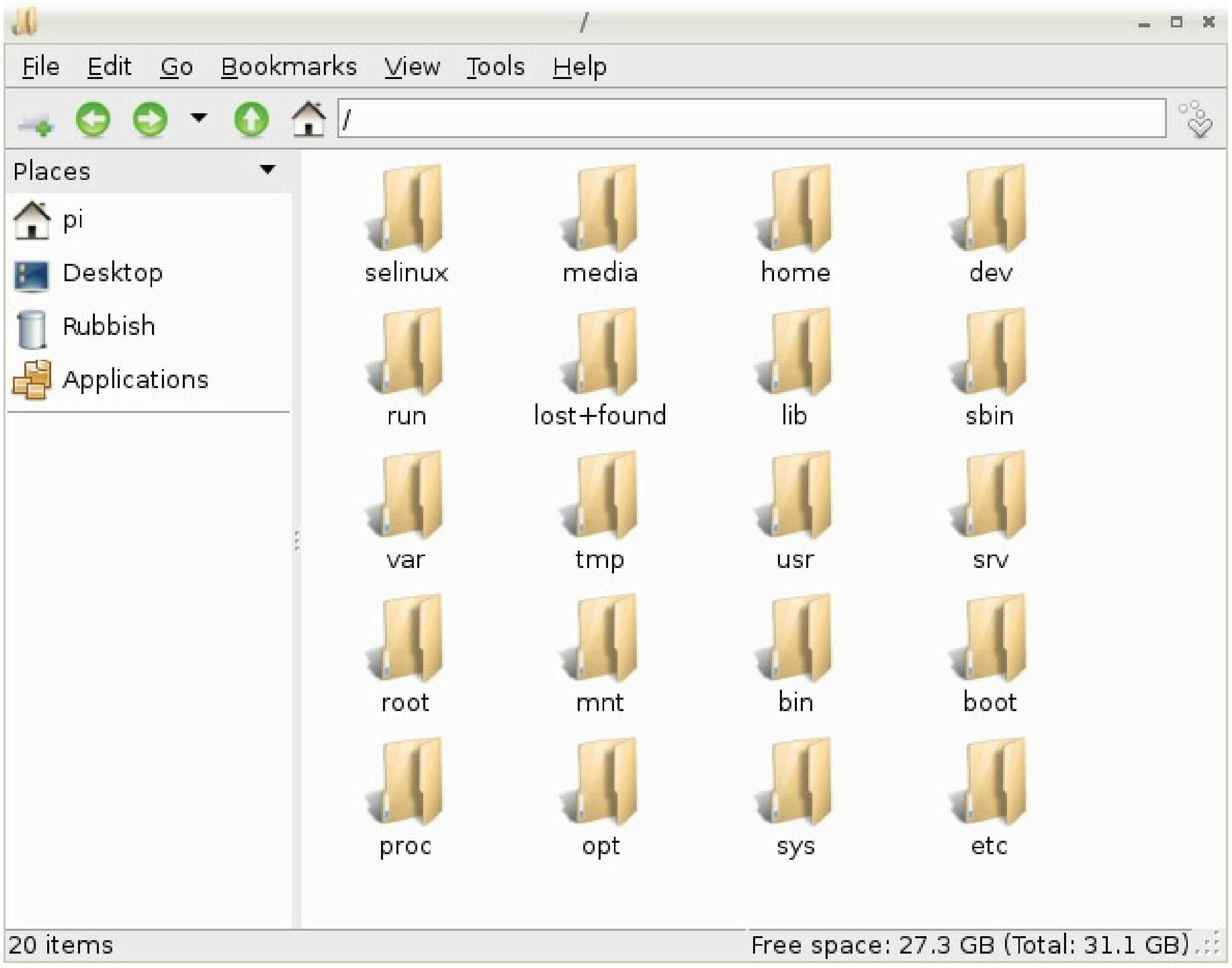
1. On the **LXDE panel**, right-click the **Desktop Pager**.
2. Click **"Desktop Pager" Settings**.

3. On the **Desktops** tab, type a number into the **Number of desktops** box, or use the up and down arrow buttons to increase and decrease the number of desktops.

4. Click **Close**.

Using the File Manager
You can use the File Manager to copy, rename, delete, and change the properties of files that are stored on the SD card or any USB storage devices that you have attached.

To open the file manager:
• On the **LXDE panel**, click the **File Manager** button – the second button from the left; or
• On the **LXDE panel**, click the **Menu** button, point to **Accessories**, and then click **File Manager**.



2
3
4 5
6

Figure 4. The File Manager

1 Application menu.
2 Toolbar – navigation buttons and address bar.
3Tab bar.
4 Left panel – for selecting places, devices, and file trees.
5 Right panel – for selecting files and folders.
6 Status bar.

Like a web browser, the File Manager window can open multiple folders at the same time by using tabs. To open a new tab:
• On the toolbar, click the first button from the left.

This will create a new tab in the same folder, and will show the tab bar. You can use the tab bar to switch between the tabs that are currently open.

If you have used the Windows Explorer in Microsoft Windows or the Finder in Mac OS X, you already know how to complete many of the most common tasks. However, the table below is a brief summary of how to use the Raspbian File Manager.

To Do This

Switch between the “Places” view and the full directory tree
In the left panel, click **Places** and then click **Directory Tree**.

Change the way icons files and folders are displayed in the right panel

On the application menu, click **View**, and then click on one of the four view options – **Icon View**, **Thumbnail View**, **Compact View** or **Detailed List View**.

Open a folder In the left panel, click the name of a folder. In the right panel, doubleclick folder's icon.
Open a folder using a file path
In the address bar, type the full file path and then press **Enter**.
To Do This

Open the current user's home directory
On the application menu, click **Go**, and then click **Home Folder**; or on the toolbar, click the **Home** button.

Open the desktop folder On the application menu, click **Go**, and then click **Desktop**.

Open the current folder in the command line
On the application menu, click **Tools**, and then click **Open Current Folder in Terminal**.

Select a file or folder In the right panel, click an icon. Select multiple files or folders In the right panel, hold **Ctrl** and click the icons of the files and folders that you want to select. Delete a file or folder Right-click an icon and then click **Delete**.
Rename a file or folder Right-click an icon and then click **Rename**.

View the properties of a file or folder
Right-click an icon and then click **Properties**.

Open a file in its default application Double-click a file's icon.

Move a file or folder to a new location
Right-click an icon and then click **Cut**. Browse to the folder that you want to move the item to, right-click in the right panel, and then click **Paste**.

Copy a file Right-click an icon and then click

Copy . Browse to the folder where you want a copy to be created, right-click in the right panel, and then click **Paste**.

Create a new folder

Right-click in the right panel, point to **Create New…**, and then click **Folder**.

If you are used to Microsoft Windows, the Linux file system might seem a little strange at first. It contains a large number of folders, and many of these have specific purposes. The key ones are:

Folder Description

home Contains a folder for each user account on the system. For example,
/home/pi is the home folder for the default user on Raspbian installations. This is where you should store all of your personal files.

etc Contains most of the configuration files for the operating system.

dev Linux creates file system entries in this folder for hardware devices that are attached to the system. Media devices (for example, DVD drives) are placed in */mnt* or */media* instead.

media When storage devices such as CDs, memory cards, and USB flash drives, are inserted, the system creates a "mount point" for them in this folder. Mount points provide access to the file system on the device.

root This is the home folder for the system administrator account.
tmp Temporary files used by the system. The contents of this folder are deleted when the Pi starts. boot Contains files needed to start Raspbian.

var Stores variable and temporary files that are created by the user or applications running as the user.

bin Contains applications that are used by all user accounts, the system, and the system administrator.

usr Contains applications and documentation for all user applications.
sbin Contains applications that are only available to the system and the system administrator.
Folder Description

lib Contains libraries – collections of code and information – that must be shared among all applications.

mnt Similar to */media*, this folder contains mount points for non-removable media.

In general, try to keep all of your data in the */home/pi* folder. When installing new software and hardware devices, they will use the special folders if they need to.

Accessing the Command Line
While most tasks can be completed using the desktop environment, there are still some things that you will have to do from the command line.

To access the command line from the graphical environment:
• On the desktop, double-click **LXTerminal**.
• On the **LXDE panel**, click the **Menu** button, point to **Accessories**, and then click **LXTerminal**.

In the command line, files and folders have file names and file paths. The file path includes the names of the folders and subfolders that the file is stored in. For example, a file *ocr_pi.png* in the *pi* user's home folder has the full file path: */home/pi/ ocr_pi.png*

File paths that begin with a forward slash start from the top-level folder. So the example above tells the system: start at the toplevel and move into the *home* folder, then move into the *pi* subfolder, and then find the file *ocr_pi.png*.

If you are already in the *home* folder, you can refer to the same file as: *pi/ocr_pi.png*

And if you are in a subfolder of */home/pi*, you can refer to the file using two dots that symbolize moving to the parent directory: *../ocr_pi.png*

To run commands: type the command using the keyboard and then press **Enter**. There are a large number of commands available (too many to list here), but these are some of the most common:

To Do This
List the contents of the current folder
Type *ls* and then press **Enter**.
List the contents of a specific folder Type *ls* followed by a space and then the name of the folder. Then press **Enter**.
Move into a subfolder Type *cd* followed by a space and then the name of the folder, and then press **Enter**.
Move to a parent folder Type *cd* .. and then press **Enter**. Create a subfolder Type *mkdir* followed by a space and then the name of the folder, and then press **Enter**.
Delete a folder Type *rmdir* followed by a space and then the name of the folder, and then press **Enter**.
Delete a file Type *rm* followed by a space and then the name of the file, and then press **Enter**.
Copy a file or folder Type *cp* followed by a space and

then the name (or full file path) of the file to be copied. Type another space and then the name (or full file path) to call the copy. Press **Enter**.

Rename or move a file or folder Type *mv* followed by a space and

then the name (or full file path) of the file to be copied. Type another space and then the name (or full file path) to call the copy. Press **Enter**.

Clear the command line window Type *clear* and then press **Enter**.

To close the command line, do one of the following:
• Press **Alt + F4** on the keyboard.
• Click the **Close** button in the top-right of the window.

• Click the icon in the top-left of the window, and then click **Close**.
• On the application menu, click **File** and then click **Quit**.
• Type the following command and then press **Enter**:

```
exit
```

In the *Accessories* group of the LXDE panel menu, there is another terminal. This one is called “Root Terminal”. When you open the command line using this application, you are automatically granted system administrator privileges.

If you only want to run a single command, you can do this from the Raspbian menu. On the LXDE panel, click the Menu button and then click Run. Type your command and then press Enter.

Understanding Linux Users and Superusers

If you have used more recent versions of Microsoft Windows then you may be used to running certain applications as an administrator. This concept is also in Linux.

Superusers (often called “root”) have full access to the system. Any applications you run as a superuser will also have full access to the system. To protect the system from accidental or malicious damage, you rarely login to Raspbian as a superuser.

Normal users have less access to the core files needed by the OS, and this means that any applications that they run also have less access to the system. In Raspbian, *pi* is the user that logs into the desktop environment, and it is a normal user.

When you do need to change part of the system, or accomplish a task that only superuser can do, you can:

• On the **LXDE panel**, click the **Menu** button, point to **Accessories**, and then click **Root Terminal**; or

• From the normal command line, type *sudo* followed by a space before the command that requires superuser access privileges.

2.6 Common Tasks

In this section, you can find instructions for some of most common tasks that people perform after installing Raspbian. These instructions continue into the next section, 2.7 Network Connections and Remote Access on page 40.

Restarting and Shutting Down the Raspberry Pi To restart the Pi:
1. On the **LXDE panel**, click the **Menu** button and then click **Run**.
2. Type the following command and then press **Enter**: `sudo shutdown –r now`
To shut down the Pi from the desktop:
1. On the desktop, double-click **Shutdown**. 2. Click **Yes**.
To shut down the Pi from the command line:
• Type the following command and then press **Enter**: `sudo shutdown –h now`

Configuring the TV Service
If you connect your Pi to a television using an HDMI cable, Raspbian automatically selects a screen size that it decides will be the best for your setup. However, this is not always the highest resolution.

The configuration setting that controls how Raspbian selects the screen size is in the file */boot/config.txt* and you can adjust this setting with a text editor:
1. On the **LXDE panel**, click the **Menu** button, point

to **Accessories**, and then click **Root Terminal**. 2. Type the following command and then press **Enter**:
```
tvservice –d edid
```
3. Type the following command and then press **Enter**:
```
edidparser edid
```

4. Find the video mode that you want to use, and make note of the mode number and whether it says “DMT mode” or “CEA mode”.

5. Type the following command and press **Enter**: `nano /boot/config.txt`

6. Press the **Down Arrow** key until you find the lines that begin with
#hdmi_group=
#hdmi_mode=

7. Remove the # from the start of both of those lines. 8. If the video mode is DMT, change the hdmi_group to 2:

```
hdmi_group=2
```
9. If the video mode is CEA, change the hdmi_group to 1:
```
hdmi_group=1
```
10. Type the mode number after *hdmi_mode=*, for example:
```
hdmi_mode=16
```
11. Press **Ctrl + X**.
12. Press **Y**, and then press **Enter**.
13. Restart the Raspberry Pi.

Installing a Graphical Package Manager

When installing new applications, there are a few steps that need to be taken:

1. Find out whether the application is compatible with

the OS.

2. Install any other pieces of software that this new application requires.
3. Download the application from the Internet. 4. Install the application to the correct folders.

In Linux, the *package manager* handles all of this for you. You can also use it to uninstall applications that you do not want anymore. Package managers connect to *repositories* – online libraries of software. And since Linux is open-source, most of the software in these repositories is also open-source (and free).

The package manager that is built-in to Raspbian is accessed from the command line. However, you can also install one that runs in the graphical desktop environment.

One such tool is Synaptic. To install it, use the command-line package manager:

1. On the **LXDE panel**, click the **Menu** button, point to **Accessories**, and then click **Root Terminal**.

2. Type the following command and then press **Enter**:
```
apt-get install synaptic
```

3. When the process is complete, type the following command and then press **Enter**:
```
exit
```

To run the Synaptic package manager:

• On the **LXDE panel**, click the **Menu** button, point to **Preferences**, and then click **Synaptic Package Manager**.

2.7 Network Connections and Remote Access

Setting up Wi-Fi
Before buying a USB Wi-Fi adapter for use with the Raspberry Pi, check that the adaptor is known to be working by looking at the list at *http://elinux.org/RPi_USB_Wi-Fi_Adapters*

To install a USB Wi-Fi adapter on your Pi and connect to your network:
1. Insert the USB device into an available USB socket on the Pi, or on a USB hub.
2. On the desktop, double-click **WiFi Config**.

3. In the **Adapter** list, click **wlan0**. If you have no adapters in the list, your USB device is not
compatible with the Raspberry Pi.

4. Click **Scan** to open the "Scan results" window.
5. Click **Scan**.
6. Double-click the name of your Wi-Fi network.

7. Check that the **Authentication** and **Encryption** settings match how your other computers connect to your router.

8. Type your network password into the **PSK** box. 9. Click **Add**, and then click **Close**.

Finding the IP Address
To connect to the Pi from another computer on your network, you will need to know its *IP address*. To find this:

1. On the desktop, double-click **LXTerminal**.
2. Type the following command and then press **Enter**: `ifconfig`

The address you need is called "inet addr" and consists of four numbers separated by dots. If you are connected to your network using an Ethernet cable, you can find the address in the section labelled "eth0". If you are connected to your network using a wireless adaptor, you can find the address in the section labelled "wlan0".

Using a Static IP Address
When the Pi restarts, or reconnects to the network, your network router will give the Pi an IP address. However, this can change every time.

You can configure the Pi so that it always uses the same IP address. You need a few pieces of information and you will need to edit a configuration file in a text editor.

To do this:

1. On the desktop, double-click **LXTerminal**.
2. Type the following command and then press **Enter**: `ifconfig`

3. If you are using an Ethernet cable, in the section labelled "eth0", make a note of the values "inet addr", "Bcast" and "Mask".

4. If you are using Wi-Fi, in the section labelled "wlan0", make a note of the values "inet addr", "Bcast" and "Mask".

5. Type the following command then press **Enter**: `netstat -nr`

6. Make a note of the values in “Gateway” and “Destination”. If you can see two entries in the table, ignore the values 0.0.0.0 and use the value from the other line.

7. Type the following command and then press **Enter**: `sudo nano /etc/network/interfaces`
8. If you are using Wi-Fi, skip to step 12.

9. If you are using an Ethernet cable: change the line that reads *iface eth0 inet dhcp* to
```
iface eth0 inet static
```

10. Add the following lines, starting on a new line after the word “static”. Replace the values in angled brackets with the values from the previous steps:

```
address <inet addr>
netmask <mask>
network <destination>
broadcast <bcast>
gateway <gateway>
```

11. Skip to step 14.
12. If you are using Wi-Fi: change the line that reads *iface wlan0 inet dhcp* to

```
iface wlan0 inet static
```
13. Add the following lines, starting on a new line after
the word “static”.

```
address <inet addr>
netmask <mask>
gateway <gateway>
```

14. Press **Ctrl + X**.
15. Press **Y**, and then press **Enter**.
16. Restart the Raspberry Pi.

Transferring Files to and from the Raspberry Pi
You can use secure copy (SCP) to transfer files from your computer to a Pi, or to copy files from the Pi to another computer. Raspbian already supports this, and so you only need to install a client on your main computer.

On Windows, WinSCP is very popular application for transferring files. You can download it for free at *http://winscp.net*

On Mac OS X, you can use an application called Cyberduck. This is a free download from *http://cyberduck.io/* or you can download it from the Mac App Store.

The details you need to make a connection to the Pi are largely the same regardless of which SCP client you use:

Server/Hostname: the IP address of your Pi
Username: *pi*
Password: this is *raspberry* if you have not yet changed it. Protocol: *SCP*

To download files from the Pi, drag the icons from the SCP client’s window to a Windows Explorer or Finder window. To upload files to the Pi, drag files from a Windows Explorer or Finder window and drop them onto the SCP client’s window.

Connecting to the Pi with SSH

Secure shell (SSH) is a way of talking to your Pi over an encrypted transmission. It is typically used when you want to access the Raspberry Pi's command line from another computer. As long as your Pi has a network connection, you can connect to it remotely and type commands.

On Windows, you will need an application that understands the SSH protocol. There are many of these available, but PuTTY is one of the more popular ones. Download it from *http:// www.chiark.greenend.org.uk/~sgtatham/putty/*

You do not need an additional SSH client to connect to the Pi on Mac OS X.
To connect to your Pi over SSH on Windows (8/7/Vista/XP): 1. Locate the file *putty.exe* that you downloaded, and then double-click it.
2. In the **Host Name (or IP address)** box, type the IP address of your Pi.

3. Click **Open**.
4. At the login prompt, type *pi*.

5. At the password prompt, type your password and press **Enter**. This is *raspberry* if you have not changed it.

To connect to your Pi over SSH on Mac OS X:

1. On the dock, click **Finder**.
2. On the sidebar, click **Applications**.
3. Click **Utilities**, and then double-click **Terminal**.

4. Type the following command, replacing <IP> with the IP address of your Raspberry Pi:

```
ssh pi@<IP>
```

5. Press **Enter**.
6. At the password prompt, type your password and press **Enter**. The default password is *raspberry* if you have not changed it.

Connecting to the Pi with Remote Desktop

Microsoft Windows comes with an application called "Remote Desktop". You can use this to connect to your Raspberry Pi and use the desktop environment from a Windows computer. Mac OS X users need to download "Microsoft Remote Desktop" from the Mac App Store.

You also need to install a small software application on the Pi before you can connect using Remote Desktop.
To install the software from the command line:
1. On the **LXDE panel**, click the **Menu** button, point to **Accessories**, and then click **Root Terminal**. 2. Type the following command and then press **Enter**: `apt-get install xrdp`
To start a Remote Desktop connection from Windows (8/7/Vista/ XP):
1. Press the **Windows logo key + R**. Then type *mstsc* and press **Enter**.
2. In the **Computer** box, type the IP address of your Raspberry Pi, and then press **Enter**.
3. In the "Login to xrdp" window, in the **username** box, type *pi*.
4. In the **password** box, type your password.[1]
5. Click **OK**.

Browsing the Web

If you are using the latest version of Raspbian, the web browser is called Epiphany and it is installed when you install the OS. In other versions, the web browser is Midori.

To open a web browser:

• On the **LXDE panel**, click the **Menu** button, point to **Internet**, and then click **Midori** or **Epiphany Web Browser**.

2.8 Other Operating Systems

Raspbian Linux is not the only OS that you can run on a Raspberry Pi. Many varieties of Linux have been repackaged so that you can use them, and there are several non-Linux operating systems that also work.

Arch Linux
Arch Linux is an extremely lightweight version of Linux that is ideal for servers or devices with unusual purposes – where many of the packages that are normally installed to create a usable desktop environment are not needed.

You can download a disk image of Arch Linux from *http://www.raspberrypi.org/downloads/*, or it can be installed using NOOBS.

RISC OS
RISC OS is available at *http://www.raspberrypi.org/downloads/* or you can install it using NOOBS. It has been in use since 1987, and was designed for Archimedes line of computers made by Acorn Computers Ltd. These machines used the first ARM processors.

1. If you have not changed the default password, it is raspberry.

RetroPie
RetroPie is a customized version of Raspbian that repackages the OS with a variety of emulators for classic computers and video games consoles. Find out more at *http://blog.petrockblock.com/retropie/*

Plan 9
Plan 9 is developed by Bell Labs, and has its origins in an OS that was released in the 1980s. It is very light on system resources, and quite different from the operating systems that people are used to today. You can download a disk image of Plan 9 from *http://plan9.bell-labs.com/wiki/plan9/download/*

AEROS
AROS is designed to be a successor to the operating system AmigaOS, which is itself based on the operating system used by the Commodore Amiga range of 16-bit computers. The Raspberry Pi version is called AEROS. The current version is only available to registered users, but older versions are available at *http://www.aeros-os.org/styled-11/index.html*

Android
Android is an operating system for tablets and smartphones, but it is not yet fully usable on the Raspberry Pi. The webpage for this project is *http://androidpi.wikia.com/wiki/Android_Pi_Wiki*

No Operating System
If you are an experienced software developer and you do not need to run any other applications, then you might be able to run your code without installing an operating system. This is called “bare-metal” programming and there are various tutorials and samples available on the web to help you.

3 – Building a Media Center with XBMC

3.1 XBMC

XBMC is a media player that is developed by the XBMC Foundation. As a multi-platform, highly-customizable, and opensource software package, it is extremely popular and is used on a wide variety of devices.

XBMC can play audio and video files in many formats, including: 3GP, AAC, APE, AVI, CDDA, FLV, MIDI, MKV, MP3, MP4, M4A, MPEG, OGG, WAV, WMA, WMV, and many more.[1] It can even play DVDs from ISO disc images.

Many people have a Raspberry Pi specifically for the purpose of running XBMC. The Pi's compact size, low cost, support for USB devices and networking, and built-in HDMI output make it an excellent choice for a media player that sits next to the TV and does not take up a lot of space.

The current version of XBMC for the Raspberry Pi is 13.2 “Gotham”. When the XBMC Foundation release version 14 of the software, they will rename XBMC to “Kodi”.

In this chapter, you can see how to install XBMC on your Pi. You will also learn the basics of how to work with the user interface, and how to add media files to the system.

To complete this tutorial, you need:
• A Raspberry Pi (any model) connected to your local network and with access to the Internet.
• A keyboard and mouse.
• A USB storage device, such as a memory stick or hard drive.

1. For the complete list of container formats and codecs that XBMC supports, see http://wiki.xbmc.org/index.php?title=Features_and_supported_formats

3.2 Installation

If you have a blank SD card that you will be using for your media center then you can install the operating system and XBMC together. RaspBMC and OpenELEC are two Linux distributions that already contain XBMC, and you can install these from NOOBS. For more information about installing operating systems using NOOBS, see section 2.2 Installation of Raspbian with NOOBS on page 16.

To start NOOBS when an existing OS is already on the SD card: turn off the Pi, and then hold the Shift key as you turn the Pi back on and until you see the NOOBS window.

NOOBS can either remove your existing OS, or add RaspBMC or OpenELEC as an additional option. To remove an OS, uncheck the box next to it before you click Install. If you leave your existing OS on the SD card, you will be asked to choose which one you want to load when the Raspberry Pi starts up.

If you cannot use NOOBS, you can also download disk images of the distributions from *http://www.raspberrypi.org/ downloads/* and then refer to section 2.3 Installation of Raspbian using a Disk Image on page 20 for more information about how to write the file to an SD card.

If Raspbian is already installed on your card and you want to add XBMC to it, installing XBMC usually only takes a few minutes. To begin, you need to add Michael Gorven's repository to the list of application sources in Linux:

1. On the **LXDE panel**, click the **Menu** button, point to **Accessories**, and then click **Root Terminal**.

2. Type the following command and then press **Enter**: `nano /etc/apt/sources.list.d/mene.list`
3. Type the following text as the first line in the file:

```
deb http://archive.mene.za.net/raspbian wheezy contrib
```
4. Press **Ctrl + O**, and then press **Enter**.
5. Press **Ctrl + X**.
6. Type the following command on one line, and then press **Enter**:
```
apt-key adv —keyserver keyserver.ubuntu.com
-recv-key 5243CDED
```
7. Type the following command and then press **Enter**: `apt-get update`
To install XBMC from the command line:
• Type the following command and then press **Enter**: `apt-get install xbmc`

XBMC does not work correctly if 64 MB or less memory is allocated to the GPU. As this is the default setting for Raspbian installations, you need to change it in raspi-config:

• In the **Root Terminal**, type the following command and then press **Enter**:

```
raspi-config
```
• or, on the **LXDE panel**, click the **Menu** button, and click **Run**. Type the following command and then press **Enter**:
```
sudo raspi-config
```

When the Raspberry Pi Software Configuration Tool appears:

1. Press the **Down Arrow** key repeatedly until **Advanced Options** is highlighted, and

then press **Enter**.

2. Press the **Down Arrow** key twice to select **A3 Memory Split**. Press **Enter**.
3. Change the value to *128* and then press **Enter**. 4. Press the **Right Arrow** key twice to select **<Finish>**. Then press **Enter**.
5. Press **Enter** to select **<Yes>** and reboot the Pi.

Running XBMC
XBMC is a standalone application and does not run the desktop environment. To run it, you must drop back to the command line: press **Ctrl + Alt + F1**. To return to the desktop environment, press **Alt + F7**.

To start XBMC:
• Type the following command and then press **Enter**: `xbmc-standalone`

However, starting XBMC this way reduces the amount of memory and system resources available to it. If you want to use your Raspberry Pi as a dedicated media player and automatically run XBMC when it starts up:

1. On a command line or **LXTerminal**, enter the following command and press **Enter**:

```
sudo nano /etc/default/xbmc
```

2. Change *ENABLED=0* to

```
ENABLED=1
```

3. Change *USER=xbmc* to

```
USER=pi
```

4. Press **Ctrl + X**, then press **Y**, and then press **Enter**. 5. Type the following command and then press **Enter**:

```
sudo raspi-config
```

6. Press the **Down Arrow** key twice to highlight **Enable boot to Desktop/Scratch** and then press **Enter**.
7. Highlight **Console Text console, requiring login (default)** and then press **Enter**.
8. Press the **Right Arrow** key twice to highlight **<Finish>**, and then press **Enter**.
9. Press **Enter** to select **<Yes>**.

3.3 First Steps

Once XBMC is finished loaded, you should see a screen similar to Figure 1 below.

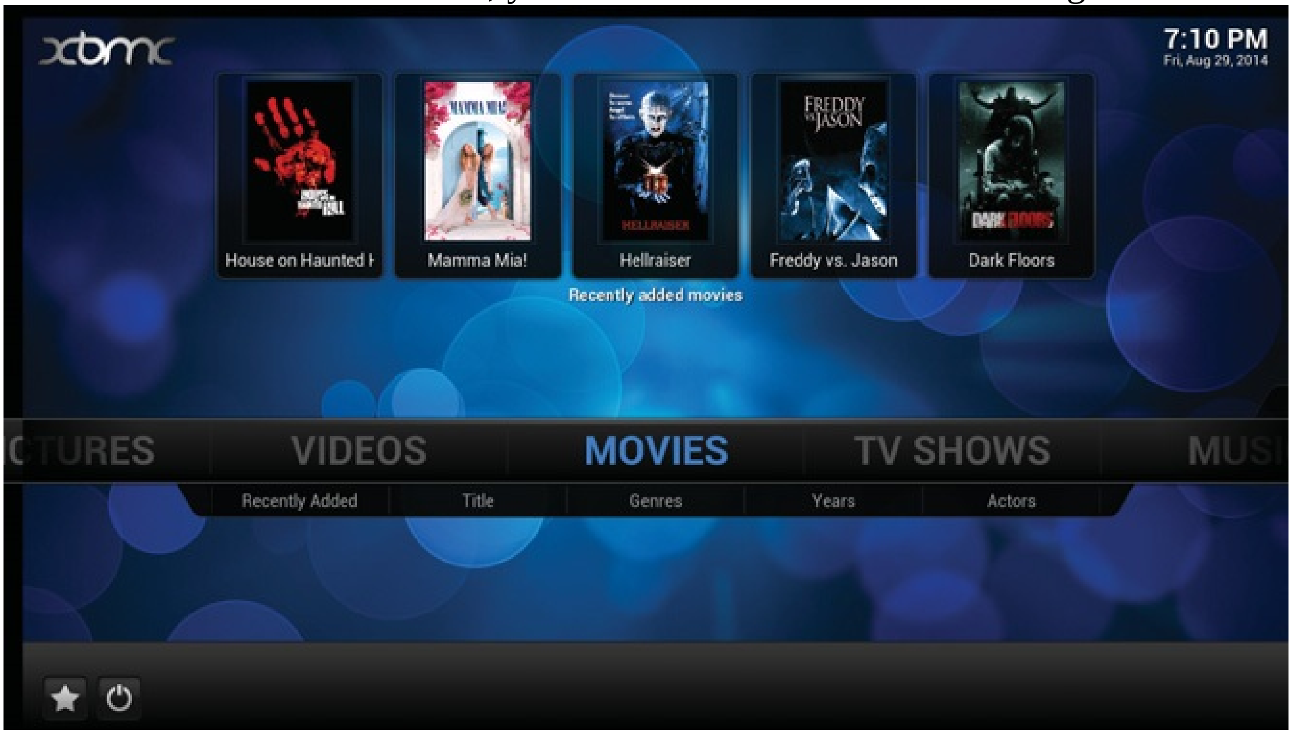


Figure 1. The XBMC home screen

1 Recent videos. When the **Movies** or **TV Shows** menu option is highlighted, XBMC will show the files that have most recently been added to your library. You can click one of these items to start playing.

2 Main menu.

3 Favorites. Click this button for quick access to items that you have placed in your favorites list. To add an item to your favorites, rightclick it to open its context menu.

4 Exit. To close XBMC and return to the Linux command line, click the **Exit** button, and then click **Exit**.

All of XBMC's features can be accessed using its mouse-driven menu system, you generally only need to use the keyboard for typing in configuration settings.

To control the home screen menu:
• Move the mouse pointer to the far left and far right sides of the menu to scroll it.
• Point to a menu item to bring up any options that are available in that category.

• Click a menu item (or option underneath) to open it. Most menu items open a different screen when you click them. To control these screens:

• Click with the left mouse button to activate menu items and buttons.

• Right-click an area of the screen that does not contain a menu item or button to move to the previous screen.

• Right-click a media file or folder to bring up a context menu which has additional options that affect the selected item.

• To open the sidebar, move the mouse pointer over to the tab on the left side of the screen. To close the sidebar, move the mouse pointer away to the right.

• Click the **Back** button to return to the previous screen.
• Click the **Home** button to return to the home screen.

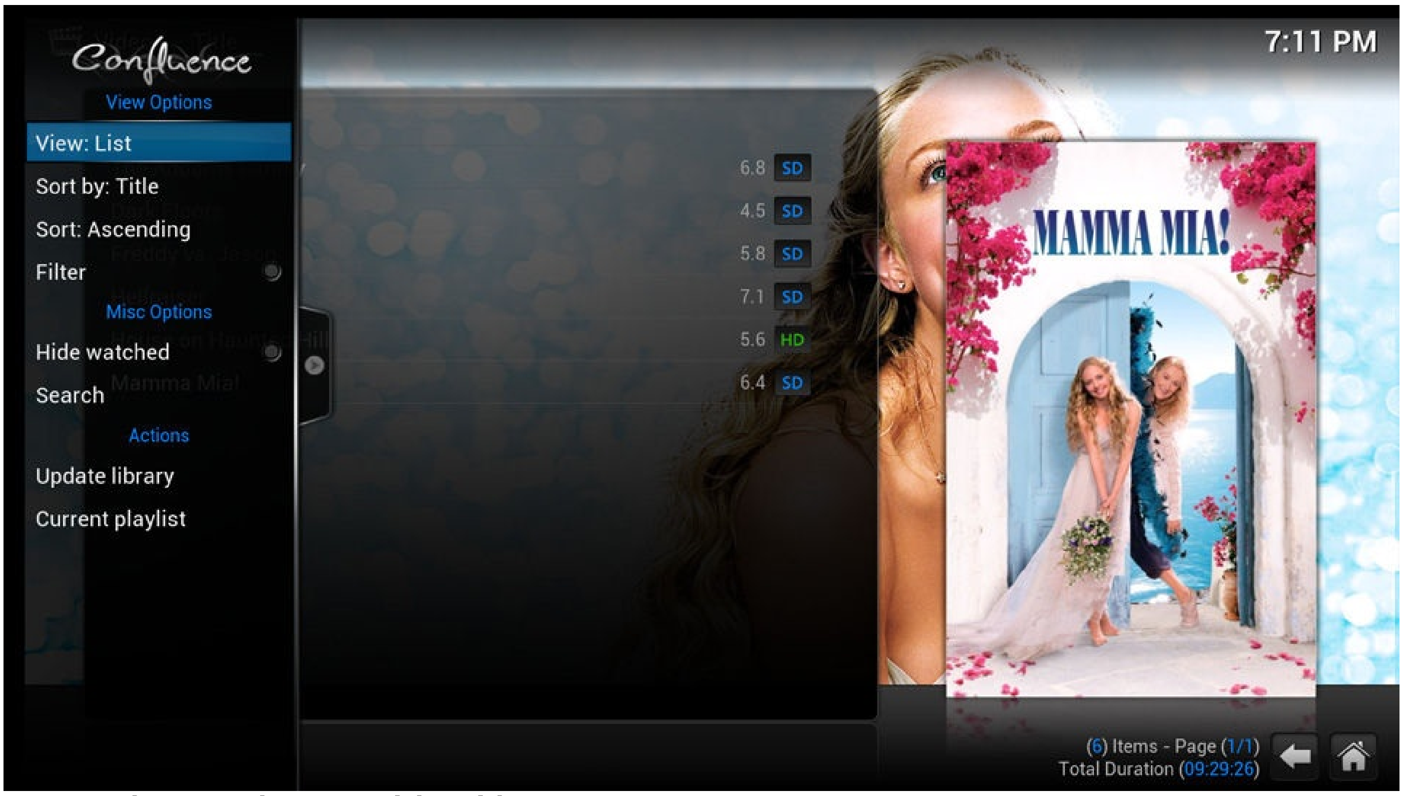


Figure 2. The Movies browser and the sidebar

XBMC divides media files into categories depending on whether they are a movie file, an episode of a TV show, a music file, a picture or photo, or a program. The different types of file are accessed using different menu items on the home screen menu.

Changing the XBMC Skin

The Raspberry Pi version of XBMC uses an interface theme (or “skin”) called “Confluence”. If you want to change this, you can download new skins directly from the XBMC system settings.

To download and enable a new skin:
1. On the XBMC **Home** screen, click **System**.

2. Click **Appearance**, and then click the first item in the right panel. This item has the word “Skin” on one side, and “Confluence” on the order.

3. Click **Get More…**
4. Point to each skin to load a preview and
description.
5. To download a skin, click it, and then click **Install**. 6. When asked if you want to switch to this skin, click **Yes**.
To change the skin back to Confluence:
1. On the XBMC **Home** screen, click **System**.
2. Click **Appearance**, and then click the first item in the right panel.
3. Click the Confluence skin.

The basic controls for using XBMC do not change with different skins. However, the layout, colors, and the way media files are displayed and previewed can change.

3.4 Media Files and the Library

Using XBMC, you can view files from USB devices (such as memory sticks and hard drives) or over your local network.

In addition to browsing for files each time you want to play something, you can add devices and network folders to your XBMC *library*. The library is a list of all of the files that you have available, and usually contains extra information such as preview images and descriptions.

Note that even if you add a file to your library, it is still stored on the original device and will not be available if that device is unplugged or turned off.

Playing Media Files from USB Devices
Keeping your media files on USB storage devices means that you are not taking up space on the SD card used to run XBMC. And you can remove USB devices at any time – making this a convenient way of transferring files from your PC to be played on your media box.

To play a file without adding it to your library:

1. Insert the USB device into a free USB slot on the Pi or USB hub. You can do this when the Pi is on and running XBMC.

2. On the XBMC **Home** screen, point to **Videos** and then click **Files**.
3. Click the USB device that contains the file you want to play.
4. Browse to the file and then click it.

Adding Files to your Library
Your XBMC library is made up of a list of *sources*, and information about the media files that you have on your system. A source is XBMC's term for a file path that it remembers between sessions, and that it associates with a particular type of content (for example, a movie). When XBMC updates your library, it opens each source folder and examines all of the compatible media files inside.

During this process, XBMC runs special programs called "scrapers". These add-ons find the information about the files in your library and for this to work, the file names of movies and TV shows that you want to add must follow a set pattern. If the scraper cannot identify the file from the file name, XBMC will not add the file to your library.

The instructions below are an example to get you started; for more detailed information, see *http://wiki.xbmc.org/ index.php?title=Naming_video_files*

If your USB device contains movie files and TV shows, create two subfolders on the device and use one for movies, and the other for the TV show episodes.

When naming movie files, use the pattern: *Title (Year).ext* For example: *House On Haunted Hill (1999).mp4* To add the movies folder on the USB device as a source, and load the files into your library:

1. Insert the USB device into a free USB slot on the Pi or USB hub. You can do this when the Pi is on and running XBMC.

2. On the XBMC **Home** screen, point to **Videos** and then click **Files**.

3. Click **Add Videos…** and then click **Browse**.
4. Click **Root filesystem**.

5. Click **media**, and then click the “name” of the USB device.

6. Click the movies folder, and then click **OK**.
7. Click **OK**.

8. From the **This directory contains** list, select **(Movies)**.

9. Under **Choose a Scraper**, click **The Movie Database** (or another scraper if you have already installed one).

10. Click **OK**.
11. If you are asked whether you want to refresh info for all items, click **Yes**.

If The Movie Database scraper cannot find your movie in its listings, XBMC will not add the file to your library. There are other scrapers available and you can install them in section 3.6 Add-ons on page 65.

To add TV shows to your library, it is advisable to create a directory structure inside the TV show folder on the USB device.

Create a folder for each TV series. For example, create a folder called *The Addams Family*. Then create subfolders inside that for each season. For example, *The Addams Family/Season 1* and *The Addams Family/Season 2*.

Place the video files for each season in the correct subfolder, and name each file with the pattern: *Anything Including Spaces_sXXeYY.ext*. Replace XX with the season number, and YY with the episode number. For example: *The Addams Family_s01e01.mp4*

To add the TV shows folder on the USB device as a source, and load the files into your library:

1. Insert the USB device into a free USB slot on the Pi or USB hub. You can do this when the Pi is on and running XBMC.

2. On the XBMC **Home** screen, point to **Videos** and then click **Files**.

3. Click **Add Videos…** and then click **Browse**.
4. Click **Root filesystem**.

5. Click **media**, and then click the “name” of the USB device.

6. Click the TV shows folder, and then click **OK**.
7. Click **OK**.

8. From the **This directory contains** list, select **(TV Shows)**.

9. Under **Choose a Scraper**, click **The TVDB** (or another scraper if you have already installed one).
10. Click **OK**.

11. If you are asked whether you want to refresh info for all items, click **Yes**.

If the scraper cannot find your file in its listings, XBMC will not add the file to your library. There are other scrapers available and you can install these using the Add-ons menu.

Playing and adding music files to your library works in the same way. On the **Home** screen, point to **Music** and then click **Files**.

Removing USB Devices

Make sure that you properly eject USB devices from the system. This process stops XBMC from reading or writing to the device while you unplug it, and ensures that all of the changes that need to be made to the file system are completed.

To safely remove a USB device:
1. On the XBMC **Home** screen, point to **Videos** and then click **Files**.
2. Right-click the device, and then click **Remove safely**.
3. Remove the USB device from the USB socket.

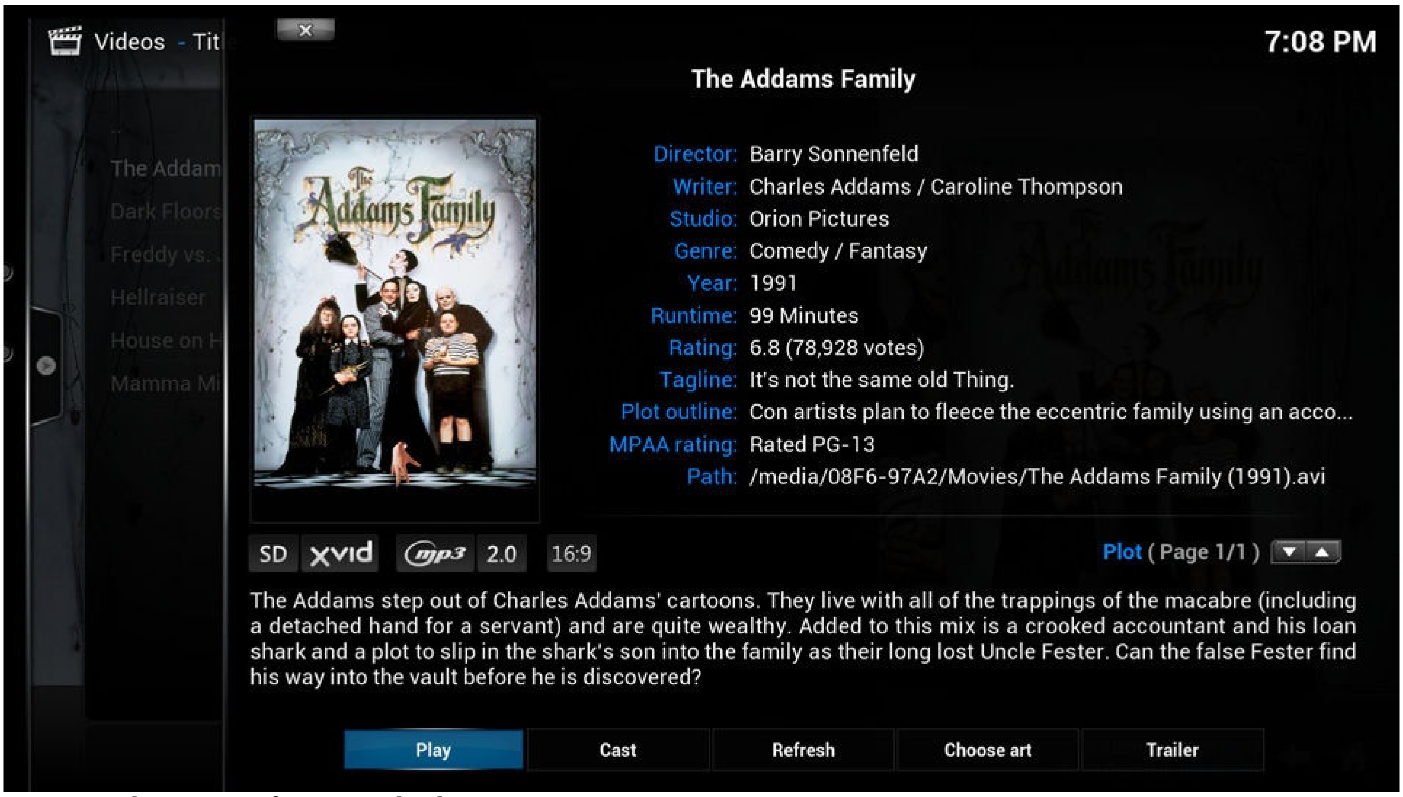
If the USB device contains the files for items in your library, those items will be unavailable until you plug the device back in to your Pi.

Playing Videos from Your Library

Once you have added some videos to your library, you can access them from the XBMC home screen. To play a movie, either:

• On the XBMC **Home** screen, point to **Movies**, and then click the preview of the file that you want to play; or
• On the XBMC **Home** screen, click **Movies** to load the movie browser.

If you use the movie browser, click the name of a movie to start playing, or right-click the movie and then click **Movie information** to bring up a window containing all the information that the scraper could find out about that file.



The Addams Family

Director:	Barry Sonnenfeld
Writer:	Charles Addams / Caroline Thompson
Studio:	Orion Pictures
Genre:	Comedy / Fantasy
Year:	1991
Runtime:	99 Minutes
Rating:	6.8 (78,928 votes)
Tagline:	It's not the same old Thing.
Plot outline:	Con artists plan to fleece the eccentric family using an acco...
MPAA rating:	Rated PG-13
Path:	/media/08F6-97A2/Movies/The Addams Family (1991).avi

The Addams step out of Charles Addams' cartoons. They live with all of the trappings of the macabre (including a detached hand for a servant) and are quite wealthy. Added to this mix is a crooked accountant and his loan shark and a plot to slip in the shark's son into the family as their long lost Uncle Fester. Can the false Fester find his way into the vault before he is discovered?

Figure 3. The movie information display

To play a TV show, either:

• On the XBMC **Home** screen, point to **TV Shows**, and then click the preview of the file that you want to play; or

• On the XBMC **Home** screen, click **TV Shows** to load the show browser.

If you use the TV show browser, click the name of a series to load the episodes or right-click it and then click **TV show information** to view the details of that series. Click an episode to start playing, or right-click the episode and then click **Episode information** to bring up a window containing all the information that the scraper could find out about that particular episode.

3.5 Network Devices and Other Computers

Connecting to SMB Shares
Server message block (SMB) is a networking protocol for sharing printers, files, and folders between devices on a local network. It is a well-established system, and support for it is built-in on most popular operating systems. Using SMB, XBMC can access folders on your Windows, Mac OS X, or Linux PCs as if they were local devices.

SMB shares must be added to XBMC as sources, and so it is usually useful to have your movies, TV shows, music, and pictures separated into different folders.

To create a new shared folder on Windows (8/7/Vista/XP): 1. Press the **Windows logo key + R**. Type *explorer* and then press **Enter**.

2. Browse to a location on your PC that either all users can access (such as an extra hard drive partition) or that belongs to the user that is currently logged in.

3. Right-click in the right panel, point to **New**, and then click **Folder**.
4. Type a name (for example, *Videos*) and then press **Enter**.
5. Right-click the folder, and then click **Properties**. 6. On the **Sharing** tab, under **Network File and Folder Sharing**, click **Share**.

7. In the **File Sharing** window, click **Share**.
8. Click **Close**.

9. In **Windows Explorer,** double-click the folder. Right-click in the right panel, point to **New** and then click **Folder**. Type a name (for example, *Movies*) and then press **Enter**.

10. Right-click in the right panel, point to **New** and then click **Folder**. Type a name (for example, *TV Shows*) and then press **Enter**.

11. Copy or move your movie files into the movies folder, and TV show episodes into the TV shows folder.

To create a new shared folder on Mac OS X (10.5 or later) using *Guest Sharing*:

1. On the dock, click **Finder**.
2. Browse to your home folder.

3. On the **File** menu, click **New Folder**. Type a name (for example, *Videos*) and press **Enter**.
4. Click this folder.
5. On the **File** menu, click **New Folder**. Type a name (for example, *Movies*) and press **Enter**.
6. On the **File** menu, click **New Folder**. Type a name (for example, *TV Shows*) and press **Enter**.

7. Copy or move your movie files into the movies folder, and TV show episodes into the TV shows folder.

8. On the **Apple** menu, click **System Preferences**.
9. Click **Sharing**.
10. Click the box next to **File Sharing**.

11. Below the **Shared Folders** list, click the **+** button.

12. Browse to your videos folder, double-click it, and then click **Add**.
13. Next to **Everyone**, click **Read Only**, and then click **Read & Write**.
14. Click **Options**.
15. Click the box next to **Share files and folders using SMB**.
16. Click **Done**.

Now that sharing is enabled and you have folders that you want XBMC to look at, add the SMB share as an XBMC video source:

1. On the XBMC **Home** screen, click **Videos**.
2. Click **Add Videos…** and then click **Browse**.
3. Click **Add network location…**[1]
4. In the **Protocol** list, click **Windows network (SMB)**.
5. Next to **Server**, click **Browse**.

6. Click the name of your workgroup, and then click the name of the Windows PC or Mac.

7. In the **Username** box, type your Windows
username (or the name of a user account that has access to the shared folder.) Do not enter a username if you are using Guest Sharing on Mac OS X 10.5 or later.

8. In the **Password** box, type the password for that user.
9. Click **OK**.
10. Click the new network location. This is *smb://*, followed by the name of your PC or Mac.

11. Browse to your movies folder and then click it.
12. Click **OK**.
13. Click **OK**.

14. From the **This directory contains** list, select **(Movies)**.

15. Under **Choose a Scraper**, click **The Movie Database** (or another scraper if you have already installed one).

16. Click **OK**.
17. If you are asked whether you want to refresh info for all items, click **Yes**.

To add the TV shows folder, repeat this process but select **(TV Shows)** from the **This directory contains** list and **The TVDB** scraper from the **Choose a Scraper** list.

1. The "Add network location…" option allows for more control over the username and password than the "Windows network (SMB)" option.

XBMC's scrapers will attempt to add the files from your sources to your library. This means that the files in your shared folder must follow the naming conventions described in Adding Files to your Library on page 54.

If the scraper cannot find information about the file, XBMC does not add the file to your library. However, you can still use the *Videos* file browser to access any files that were not added.

The most common cause of problems when creating SMB shares and adding them to

XBMC is user permissions. If you cannot connect to your share, or cannot see the files inside it, then check the user account permissions on your PC or Mac. If necessary, you can delete a source and start over.

To delete a source: 1. On the XBMC **Home** screen, point to **Videos**, and then click **Files**.
2. Right-click a source, and then click **Remove source**.

Connecting to UPnP Devices
Universal plug and play (UPnP) is a collection of networking protocols that allows devices on a local network to find each other, to find out what type of services and files other devices offer, and to share those files.

You can setup your PC or Mac so that XBMC on the Raspberry Pi can play the media files from it. On Mac OS X and Linux machines, you will need to install and configure a UPnP media server, such as MediaTomb (*http://www.mediatomb.cc*). Microsoft Windows has a built-in UPnP server, but this is usually disabled when the operating system is installed.

To enable the UPnP server on Windows (8/7/Vista):

1. Press the **Windows logo key + R**. Type *control.exe /name Microsoft.NetworkAndSharingCenter* and then press **Enter**.

2. On Windows 8/7: click **Change advanced sharing settings**. On Windows Vista: click the arrow to the right of **Network Discovery**.

3. Click the box next to **Turn on network discovery**, and then click **Save changes** (or **Apply** on Vista). To enable the UPnP server on Windows XP:
1. Press the **Windows logo key + R**. Type *appwiz.cpl* and then press **Enter**.

2. Click **Add/Remove Windows Components**.
3. Click **Networking Services**, then click **Details**.

4. If it is not selected, click the box next to **Internet Gateway Device Discovery and Control Client**. 5. If it is not selected, click the box next to **UPnP User Interface**.

6. Click **OK**.
7. Click **Next**.

8. Press the **Windows logo key + R**. Type
services.msc and then press **Enter**.
9. Double-click **SSDP Discovery Service**.
10. On the **General** tab, in the **Startup type** list, click **Automatic**.
11. Click **OK**.
12. Restart your PC.

UPnP devices and their folders should be added to XBMC as sources. Although files stored on these types of devices cannot be added to the XBMC library, they can still be accessed using the videos and music file browsers.

To add a video source:

1. On the XBMC **Home** screen, click **Videos**.
2. Click **Add Videos…** and then click **Browse**.

3. Click **UPnP Devices**, and then click the name of your device.

4. Browse into a folder, if you want to add it and its subfolders only.
5. Click **OK**.
6. Click **OK**.

To add a music source:

1. On the XBMC **Home** screen, click **Music**.
2. Click **Add Music…** and then click **Browse**.
3. Click **UPnP Devices**, and then click the name of your device.
4. Browse into a folder, if you want to add it and its subfolders only.
5. Click **OK**.
6. Click **OK**.

If your UPnP device appears in the UPnP Devices list but you cannot open it to browse its content, there are a few things to check:

• If you are running a firewall on a Windows PC with UPnP enabled, you may need to create an exception for the Windows Media Player Network Sharing Service or UPnP Framework.

• On Windows, place your video and music files in the *Videos* and *Music* folders of the user that is currently logged-in. Check that the Windows user *Everyone* is able to read from those folders.

• Check that your network router allows devices on your network to talk to each other.
• Update your operating system to the latest version.
• Update XBMC to the latest version.

3.6 Add-ons

Add-ons extend the functionality of XBMC in unique ways, and there are many add-ons available. These vary from making small changes to the interface, to letting you watch live TV and stream videos over the Internet.

To download and install an add-on: 1. On the XBMC **Home** screen, point to **System**, and then click **Settings**.
2. Click **Add-ons**.
3. Click **Get Add-ons**, and then click **XBMC.org Addons**.

4. Click the type of add-on that you want to install. For example, "Video Add-ons" add new ways of watching videos.

5. In the list of available add-ons, click one to bring up a description.
6. Click **Install** to download and install the add-on.

Installing Movie Information Add-ons
Movie information add-ons are a type of extension for finding information about the video files in your library. By default, the scraper that finds information about your movie files uses The Movie Database website. To install a different movie scraper, and configure a video source to use it:

1. On the XBMC **Home** screen, point to **System**, and then click **Settings**.
2. Click **Add-ons**.
3. Click **Get Add-ons**, and then click **XBMC.org Addons**.
4. Click **Movie information**, and then click **Universal Movie Scraper**.

5. Click **Install**.
6. On the XBMC **Home** screen, point to **Videos**, and then click **Files**.

7. Right-click your video source, and then click **Change content**.
8. Under **Choose a Scraper**, click **Universal Movie Scraper**.
9. Click **OK**.
10. If you are asked whether you want to refresh info for all items, click **Yes**.

Installing Program Add-ons
Program add-ons are powerful extensions that include programs for accessing the web and social media sites, clients for file sharing and file hosting services, backup utilities, and even torrent clients.

To open the list of program add-ons:
1. On the XBMC **Home** screen, point to **System**, and then click **Settings**.
2. Click **Add-ons**.
3. Click **Get Add-ons**, and then click **XBMC.org Addons**.
4. Click **Program Add-ons**.

3.7 Remote Controls

XBMC has a built-in web server that you can use to control it. This turns any device with a web browser into a remote control for XBMC. There are also several remote control apps available for smartphones and tablets. These apps usually talk to XBMC using the web server.

To enable the web server:

1. On the XBMC **Home** screen, click **System**.
2. Click **Services**.
3. Click **Webserver**.

4. Ensure **Allow control of XBMC over HTTP** is selected.
5. In the **Port** box, type a port number. This number must be over 1024.
6. In the **Username** box, type *xbmc*.
7. In the **Password** box, type a password that you want to use.
8. Press the **Back** or **Home** buttons to leave the system settings.

To connect to XBMC, you need to know its IP address on your local network. The IP address is shown in the **Summary** and **Network** tabs of the System info screen. To find this:

• On the XBMC **Home** screen, point to **System**, and then click **System info**.

Test the remote control feature by opening a web browser on a device on your local network. In the address bar, type *http://* followed by the IP address of the Raspberry Pi, then a colon, and then the port number that is set in the XBMC settings.

For example, to access the remote control web interface over port 8080 on a Pi with the IP address 192.168.0.9, type http://192.168.0.9:8080

Login with the username *xbmc*, and the password that you set in the XBMC settings.

If you cannot connect to XBMC, first check that the Raspberry Pi has an active network connection and then verify that the connection settings you are using match those that you entered in the XBMC settings. If you still cannot connect, try a different port number.

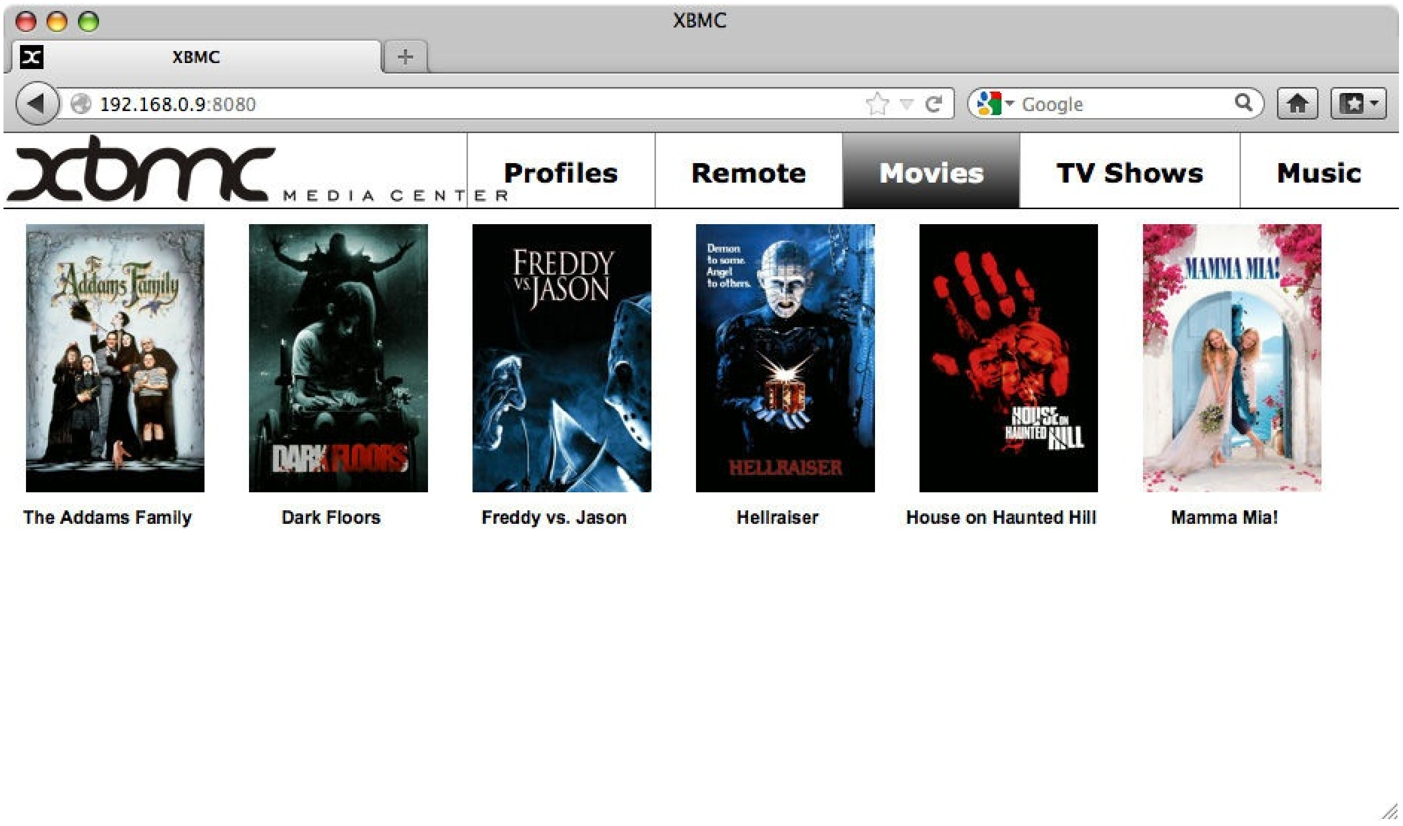


Figure 4. The XBMC web interface

To use a smartphone or tablet app, you will need the same details that you used to access the remote control with a web browser. Apps are available for iOS, Android, and Windows Phone. However, the Official XBMC Remote app is only available for iOS and Android devices.

At the time of writing, the current version of the Official XBMC Remote has a few problems. But developers update apps extremely frequently, and the issues may be fixed by the time you read this. If you cannot get the official app working, search your device's app store and you will find many others to try.

4 – Programming with Scratch

Scratch is a visual programming language that helps teach the concepts of programming to people who have never worked with traditional programming languages. It is aimed primarily at people who want to learn to build small games. However, the combination of its simple interface and rich feature set also make it useful for presentations, simulations, and prototypes of software that will be implemented in other languages.

The current version of Scratch for the Raspberry Pi is 1.4. Version 2.0 of Scratch is available on many platforms, however, it is not compatible with the Pi.

4.1 The User Interface

To start Scratch on Raspbian:
• On the desktop, double-click **Scratch**; or
• On the **LXDE panel**, click the **Menu** button, point to **Programming**, and then click **Scratch**.
The Scratch window is divided into several distinct areas:

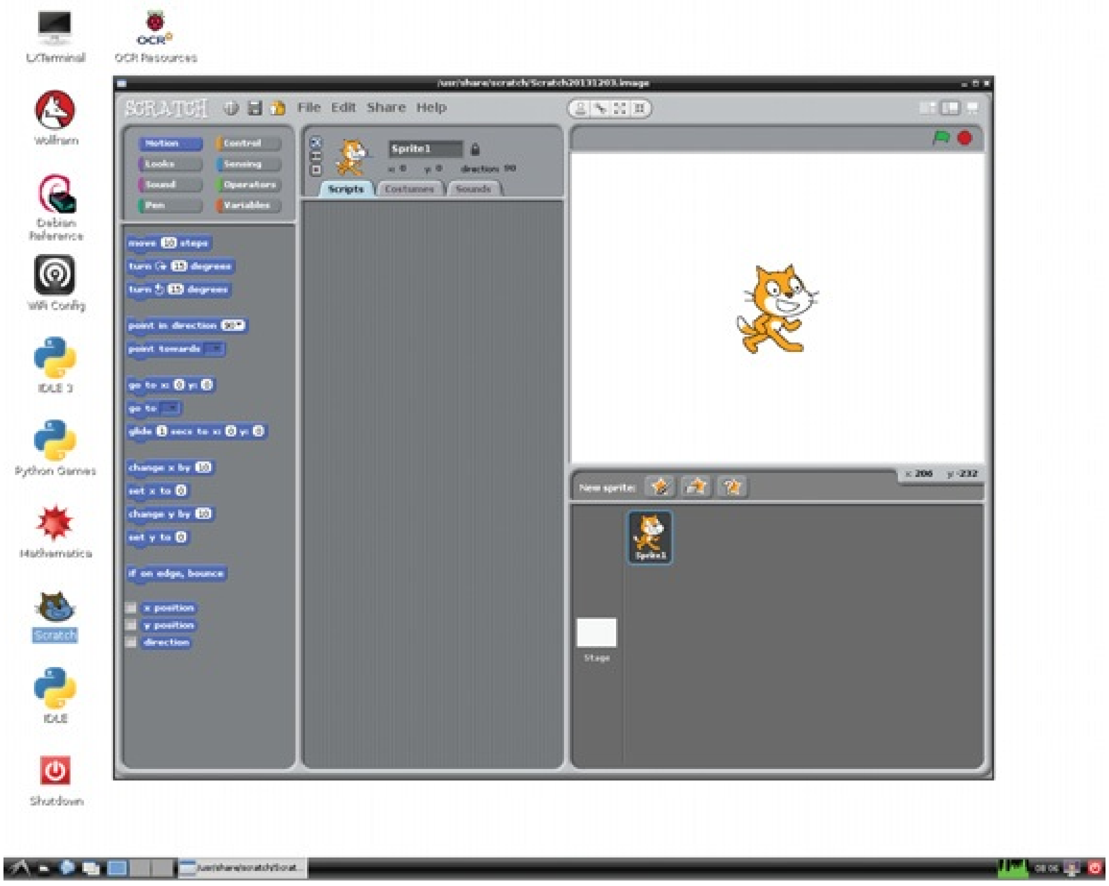


Figure 1. Scratch on Raspbian 1 Cursor Tools. These buttons are for duplicating, deleting, growing, and shrinking the sprites on the stage. Click the cursor tool, and then click a sprite to activate the function.

2 Menu.
3 Blocks Palette.

4 Scripts Area. This is where you combine blocks to control the sprite that is selected in the Sprite List.
5 Sprite List. All of the sprites in your project are shown here. Click a sprite to show the scripts that you have added for a particular sprite. 6 Stage. This is where your sprites move and interact with each other.

The key concept behind Scratch is that everything you need to move, animate, or control is a *sprite*. Each sprite is a collection of images and scripts, and is independent of all of the others. You combine the colored blocks from the Blocks Palette to set the behavior of a specific sprite.

Opening and Saving Projects

Scratch projects are saved in a special file format that uses the extension *.sb*. These files contain all of the graphics and sound files that you import into Scratch, and all of the scripts that you create.

To save a project to a new file:
1. On the **Menu** bar, click **File**, and then click **Save As…**
2. In the **New Filename** box, type a name for your project and then click **OK**.

If you have previously saved your project and you want to save it again:
• On the **Menu** bar, click the **Save this project** button (it looks like a floppy disk); or
• On the **Menu** bar, click **File**, and then click **Save**. To close the current project and open another: 1. On the **Menu** bar, click **File**, and then click **Open…** 2. Browse to the file that you want to open, click it, and then click **OK**.

Using the Stage

All of the sprites in your project appear on the *Stage*, and this is where all of your actions and scripts take place. A sprite cannot completely leave the Stage, however, you can hide sprites that are not needed at specific parts of your project.

In Scratch, the Stage is always 480 steps wide and 360 steps tall. It never changes size, it never moves, and it never resets. For example, once a sprite moves it remains at that position on the stage until another script moves it. The position of each sprite is described with a coordinate – two numbers, one that represents the horizontal position (x) and one that represents the vertical position (y). The top-left of the Stage is -240,180, the bottomright of the Stage is 240,-180, and this makes the center of the Stage 0,0.

In the Sprite List, there is an image for the Stage. If you click this, you can add blocks to the Stage itself. When the Stage is selected, the tabs at the top of the Scripts Area change.

You can change the background of the Stage from the Backgrounds tab. You can either use the *Paint Editor* to draw a new background (The Paint Editor is described in section 4.9 The Paint Editor on page 89), or import graphic files. Imported graphics should be 480 pixels wide and 360 pixels tall, or Scratch will position them in the center of the Stage surrounded by borders of the current background color.

You can have multiple backgrounds in a Scratch project, but only one can be visible at a time.
You cannot move backgrounds in Scratch, so if you want areas of the background to move then you have to create them as sprites.

Building and Running Scripts

This section describes how to work with blocks from the Blocks Palette to create *scripts* – sequences of blocks. Each sprite (and the Stage itself) can contain many scripts.

To attach a block to a sprite or the Stage itself, select it in the Sprite List and then drag blocks from the Blocks Palette to the Scripts Area. For example:

1. In the **Sprite List**, click **Stage**.
2. In the **Blocks Palette**, click **Control**.

3. Click the block **when <green flag> clicked** and, while holding the mouse button, drag the block over to the **Scripts Area**. Release the mouse button.

4. In the **Blocks Palette**, click **Looks**.

5. Drag the block **change color effect by 25** to the **Scripts Area** and underneath the other block. The white line indicates that this block will snap
together with the one above it. Release the mouse button.

6. In the **Scripts Area**, on the block **change color effect by 25**, click **color**, and then click **brightness**. 7. On the block **change brightness effect by 25** click the white box that

contains *25*. Change this to *-25*. 8. In the **Blocks Palette**, click **Control**.

9. Drag the block **wait 1 secs** to the **Scripts Area** and snap it to the bottom of the block **change**
brightness effect by -25.

10. On the block **wait 1 secs**, click the white box that contains *1*. Change this to *0.5*.
11. In the **Blocks Palette**, click **Looks**.

12. Drag the block **change color effect by 25** to the **Scripts Area** and snap it to the bottom of the wait block.

13. On the block **change color effect by 25**, click **color**, and then click **brightness**.
Figure 2. A short script

Above the Stage, there are two buttons: the green flag button starts all scripts in the project, and the red octagon stops all of the scripts.

The first block that is in the script, *when <green flag> clicked*, states that the script should run when the green flag button above the Stage is clicked. Click the green flag button now and you should see that the background briefly changes color.

To Do This
Run a script (for testing) In the **Scripts Area**, double-click any of the blocks in the script.

Insert a block above an existing block
In the **Blocks Palette**, drag a block over to the **Scripts Area** and over the top of an existing block. The white line indicates where the block will be inserted.

Detach a block (and any blocks that follow it) from a script
In the **Scripts Area**, click the block and drag it away from the block above it. All of the blocks below the selected block will move with it.

Remove a block, a series of
connected blocks, or a script from the project

In the **Scripts Area**, click the first block in the series that you want to remove and drag it over to the **Blocks Palette**.

To Do This

Duplicate a block, a series of connected blocks, or a script In the **Scripts Area**, right-click the first block in the series that you want to copy, click **duplicate**, and then click in the **Scripts Area** to position the new blocks.

Fitting the Blocks Together Scratch uses the color and shape of blocks to indicate how they can be connected to each other. Blocks with a curved top, such as the *when <green flag> clicked* block, are the start of a script and can only have blocks connected underneath them.

Figure 3. You can only snap in blocks below this one, not above.

A notch in the top of a block indicates that it can be attached to blocks above it. A tab on the bottom indicates that you can snap other blocks into position below it.

Figure 4. A block that allows other blocks to be connected on top, or underneath.

An eight-sided white box in the block means that you can type numbers in there, but also that you can add variables and blocks from the Operators section of the Blocks Palette.

To use an operator or variable block instead of a number:
• Drag an eight-sided block from the **Blocks Palette**, into the **Scripts Area**, and over the white box.

Some blocks have square boxes that you can type in. These accept numbers, strings (sequences of characters), variables, and eight-sided operator blocks.

Blocks like *if* in the Control section of the Blocks Palette have spaces for six-sided blocks. Six-sided blocks represent the values *Yes* or *No*, or *True* or *False*. These are described in more detail in section 4.4 Decisions on page 82.

4.2 Sprites

In Scratch, you will spend the majority of your time moving and changing the appearance of sprites in response to key presses.

Adding and Removing a Sprite

To add a sprite to the current project and make it appear on the Stage:

1. Above the **Sprite List**, next to **New Sprite**, click the **Choose new sprite from file** button. This is the second button in that area.

2. In the **New Sprite** window, browse to a sprite that you want to add, click it and then click **OK**.
3. On the **Stage**, click and drag the sprite to a new location.

If a sprite is the wrong size, click on the **Shrink sprite** or **Grow sprite** button in the Cursor Tools area and then click on the sprite:

To delete a sprite:
• In the **Sprite List**, right-click a sprite and then click **delete**.

Creating a New Sprite

There are two ways of creating new sprites in Scratch: you can use the built-in drawing tools to paint a new sprite, or you can import graphic files from other tools like Adobe Photoshop and KolourPaint.

In either case:
• Above the **Sprites List**, next to **New Sprite**, click the **Paint new sprite** button. This is the first button in that area.

If you want to import a graphic file, click **Import** in the Paint Editor. The editor is described in section 4.9 The Paint Editor on page 89.

Understanding Costumes

In most projects, and especially in games, you will need to change the appearance of your sprites as the project is running. These can be frames of an animation, as is the case when the sprite is "walking". Or these can be alternative versions of the sprite that use different colors or hold different weapons.

In Scratch, variations of a sprite are known as "costumes". If you create a new sprite, it will only have one costume. To add another costume to the selected sprite:

1. In the **Scripts Area**, click **Costumes**.
2. If you want to draw a new sprite using the built-in Paint Editor, click **Paint**.
3. If you want to import a graphic file to use as a costume, click **Import**.

To Do This

Set the current costume of a sprite from the Scripts Area (not from a script)

In the **Scripts Area**, click **Costumes**, and then click the image of the costume.

Rename a costume In the **Scripts Area**, click **Costumes**.

Click the textbox that contains the current name (for example, *costume1*) and then type a new name.

Duplicate a costume

In the **Scripts Area**, click **Costumes**. Next to the costume that you want to duplicate, click **Copy**.

To Do This
Delete a costume In the **Scripts Area**, click **Costumes**. Next to the costume that you want to remove, click **x**.
Edit a costume using the built-in Paint Editor

In the **Scripts Area**, click **Costumes**. Next to the costume that you want to change, click **Edit**.

Scripting Your Sprites Motion blocks set the position of sprites. On the Stage you can click and drag sprites where you want them, but to move them from scripts (for example, in response to the user pressing an arrow key) you need to use one of the motion blocks. These are:

Block Description
move ? steps Moves a sprite a number of steps in the direction it is currently pointing.

turn (clockwise) ? degrees
Rotates a sprite a number of degrees in a clockwise direction.

turn (counter
clockwise) ? degrees Rotates a sprite a number of degrees in an counter-clockwise direction.

point in direction ? Rotates or flips a sprite to face the specified direction.
point towards ? Rotates or flips a sprite to face the mouse pointer or another sprite.
go to x: ? y: ? Sets the position of a sprite. go to ? Moves a sprite to the location of the mouse pointer or another sprite.

glide ? secs to x: ? y: ?
Animates the movement of a sprite to the specified location.

change x by ? Moves a sprite horizontally by the number of steps specified. This number can be negative.
set x to ? Moves a sprite horizontally to the specified position.
change y by ? Moves a sprite vertically by the number of steps

specified. This number can be negative. set y to ? Moves a sprite vertically to the specified position.

if on edge, bounce Changes the direction of a sprite when it reaches the edge of the Stage.

Each sprite also has three variables that are changed by the motion blocks. These can be used in decision-making and arithmetic blocks.

Variable Description
x position The horizontal position of the sprite on the Stage. y position The vertical position of the sprite on the Stage.
direction The direction that the sprite is currently facing.

When sprites are added to the Stage, Scratch assumes that they face to the right. When you change the direction that a sprite faces, Scratch rotates the sprite. To tell Scratch that it should flip a sprite and not rotate it:

1. In the **Sprite List**, click a sprite.

2. At the top of the **Scripts Area**, next to the picture of the sprite, click the **only face left-right** button. This is the second button of the three that are arranged vertically.

The Looks blocks in the Blocks Palette change the appearance of sprites on the Stage. Some of these blocks change a sprite's costume, and some of them apply effects (such as fades).

switch to costume ? Sets the costume displayed for this sprite. next costume Changes to the next costume. If the current costume is the last one, the first costume will be used instead.
say ? for ? secs Displays a speech bubble next to the sprite. say ? Displays a speech bubble next to the sprite. This bubble remains until you use a **say** block with no text.
think ? for ? secs Displays a thought bubble next to the sprite. think ? Displays a thought bubble next to the sprite. This bubble remains until you use a **think** block with no text.

change ? effect by ? See below. set ? effect to ? See below. clear graphic effects See below. change size by ? Grows or shrinks a sprite by the amount specified. set size to ? % Grows or shrinks a sprite by setting its size. This is a percentage of the size of the graphic used to create the sprite, and it can be greater than 100%. show Shows the sprite on the Stage if it is currently hidden.
hide Hides a sprite if it is currently shown. go to front Moves a sprite in front of all others. go back ? layers Moves a sprite behind other sprites.

There are seven different effects that you can add to sprites:

Effect Description color Changes the color (hue) of a sprite. fisheye Distorts the sprite. **Effect Description**
whirl Distorts the sprite into a swirl around its center point.
pixelate Makes the sprite appear to be made up of larger pixels.
mosaic Makes the sprite appear to be made up smaller clones of itself.

brightness When used with positive numbers, this increases the brightness (how close the colors are to white). When used with negative numbers, this decreases the brightness (how close the colors are to black).

ghost Makes the sprite translucent (so that other sprites can be seen through it) or completely transparent (the sprite is invisible).

You control these effects using three blocks:
• To set an effect to a specific value, use the **set ? effect to ?** block.

• To increase or decrease the amount of an effect that is applied to a sprite, use the **change ? effect by ?** block.

• To remove all active effects from a sprite, use the **clear graphics effects** block.
All sprites have two variables related to their appearance:
Variable Description
costume # The number of the costume that is currently displayed for this sprite.
size The size of a sprite on the Stage. This is a percentage of the size of the graphic used to create the sprite, and it can be greater than 100%.
You can choose whether Scratch should display these variables on the Stage. To show the value of a variable:
• In the **Blocks Palette**, click the box next to the variable name.
Click the box again to remove the variable from the Stage.

4.3 Arithmetic and Variables

In the Operators section of Blocks Palette, there are eight blocks that you can use to perform basic math operations, such as adding two numbers together. These blocks are eight-sided, green blocks and they fit into any other block that accepts a number.

Block Description

? + ? Adds two numbers together. ? - ? Subtracts the second number from the first. ? * ? Multiplies two numbers together. ? / ? Divides the first number by the second.

pick random ? to ? Picks a random number between the first number and the second number.

? mod ? Performs modular arithmetic – the first number "wraps around" if it reaches the value of the second number.

round ? Rounds the specified number to the nearest whole number.
? of ? Select from a range of additional math functions.

You can use arithmetic operator blocks as inputs to other arithmetic blocks. For example, the expression *1 + ((14 / 3) x 3.14)* can be created with three blocks:

Figure 5. Nested operator blocks

Creating and Using Variables

A variable is a named area of the computer's memory in which you can store pieces of data.

To create a variable in Scratch:

1. In the **Blocks Palette**, click **Variables**.
2. Click **Make a Variable**.
3. Type a name for your variable.

4. If you want all sprites to be able to access the variable, click the circle next to **For all sprites**. 5. If you only want this sprite to be able to access the variable, click the circle next to **For this sprite only**. 6. Click **OK**.

When you create a variable, Scratch adds a block for it in the Variables section of the Blocks Palette. These orange variable blocks will fit into blocks that accept a number.

There are four blocks in the Variables section of the Blocks Palette for working with variables:

Block Description

set ? to ? Sets the value of a variable. change ? by ? Increments or decrements a variable. show variable Adds the specified variable to the Stage. hide variable Removes the selected variable from the Stage.

4.4 Decisions

There are six blocks that you use to compare numbers and make decisions. The result of these blocks is a “Boolean” value – either *true* or *false*..

Block	Description
? < ?	Is *true* if the first number is smaller than the second, or *false* if it is not.
? = ?	Is *true* if the two numbers are the same, or *false* if they are not.
? > ?	Is *true* if the first number is larger than the second, or *false* if it is not.
? and ?	Is *true* if the both of the attached blocks are *true*, or *false* if either (or both) of them is *false*.
? or ?	Is *true* if either of the two attached blocks is *true*. If neither are *true*, then it returns *false*.
not ?	Is *true* if the attached block is *false*, and *false* if the attached block is *true*.

Two blocks in the Control section of the Blocks Palette are used with the Boolean operator blocks to make decisions.

Block	Description
if ?	Runs the blocks inside it, if the attached Boolean operator is *true*.
if ? else	Runs the blocks in the top section if the attached Boolean operator is *true*, and the blocks in the bottom section if the attached operator is *false*.

You can snap any number of blocks into the middle of the *if* and *if…else* blocks. This includes adding other *if* blocks to create highly-complicated decision-making.

Controlling Your Scripts with a Keyboard and Mouse Users play Scratch games and interact with projects using the mouse and keyboard.

There are two blocks in the Controls section of the Blocks Palette that you can use to start a script when a key is pressed, or when the user clicks with the left mouse button. These blocks have curved tops and so you cannot place them in an existing script, they must always start a new script.

Block	Description
when ? key pressed	Runs the blocks snapped below when the user presses a key on the keyboard.
when sprite clicked	Runs the blocks snapped below when the user clicks on a sprite.

In games, the *when key pressed* block is used many times to move a sprite when the player presses one of the arrow keys.

Figure 6. Multiple scripts to control a sprite

All sprites can listen and react to the same key presses.

There is a special set of operators in the Sensing section of the Blocks Palette that you can use for making decisions based on keyboard or mouse input from the user. Unlike the control blocks, these blocks must be used as part of an *if* decision or with a Boolean operator block.

Block	Description
mouse down?	A Boolean value that is *true* if the user is currently holding the left mouse button down.
key ? is pressed?	A Boolean value that is *true* if the user is currently pressing the specified key on the keyboard.

You can use them in combination with the control blocks. For example, to create two different actions depending on whether the user clicks a sprite normally or holds the up arrow key when they click a sprite.

Detecting a Collision between Sprites

A *collision* is when two sprites overlap each other on the screen. The blocks that you need are in the Sensing section of the Blocks Palette, and there are two ways you can detect a collision in Scratch:

• Use **touching** to check if sprites are touching; or
• Use the **touching color** and **color touching color** blocks.

Checking if two colors touch is useful in games when you need to know if a specific part of a sprite is making contact with a specific part of another. But because Scratch is programmed to ignore transparent areas of your sprites, the *touching* block is all you need for many games.

The *touching* block does not detect collisions when sprites are hidden with the *hide* block. However, it does detect collisions when sprites are made invisible using the *ghost* effect.

4.5 Loops

Loops have space in the middle for you to snap in other blocks, and you can use them to pieces of a script repeatedly. There are three types of loop in Scratch: infinite loops, which run until the script is stopped; definite loops, which run a set number of times; and indefinite loops, which run repeatedly until a certain condition is met.

Block Description
wait ? secs This is a special kind of loop that repeatedly does nothing until the specified number of seconds has elapsed.
forever Repeatedly runs the blocks until the script is stopped.
repeat ? Repeatedly runs the blocks a set number times.
Block Description
forever if ? Checks the Boolean operation and if it is *true* then

Scratch runs the blocks that are in the middle of the loop. Then it checks the condition again. If the Boolean operation is *false*, the looping ends.

wait until ? This is a special loop that repeatedly does nothing until the Boolean operation is *true*.
repeat until ? Checks the Boolean operation and if it is *false*

then Scratch runs the blocks that are in the middle of the loop. Then the loop checks the condition again. If the Boolean operation is *true*, the looping ends and Scratch runs the block that is snapped onto the bottom of the **repeat until** block.

The *forever if* and *repeat until* blocks seem very similar. However, there are two important differences:

1. *Forever if* runs blocks while the Boolean operation is *true* – it ends when the operation is *false*. *Repeat until* runs blocks while the Boolean operation is *false* – it ends when the operation is *true*.

2. You cannot snap other blocks to the bottom of a *forever if* block.

4.6 Strings

A *string* is a sequence of characters, like a name, word, or sentence. Many of the blocks that you can use to work with strings are introduced earlier in this chapter:

Block Section Description
set ? to ? Variables Sets the value of the specified variable to a string. say ? for Looks Displays a speech bubble next to the sprite for the ? secs number of seconds specified.
say ? Looks Displays a speech bubble next to the sprite. **Block Section Description**

think ? Looks for ? secs
Displays a thought bubble next to the sprite for the number of seconds specified.

think ? Looks Displays a thought bubble next to the sprite.

ask ? and Sensing wait
Shows the specified string and a text input box to the user. When the user presses **Enter** or clicks the button, their answer is stored in the variable **answer**.

But in the Operators section of the Blocks Palette, there are three operator blocks that only work with strings:

Block Description
join ? ? Joins two strings together. letter ? of ? Extracts the character at the specified position of the string.
length of ? Calculates the number of characters in the string. Figure 7. A basic login prompt using strings

4.7 Messages

In Scratch, all scripts are associated with an individual sprite and run independently of any other scripts. *Messages* allow the different sprites on the Stage to communicate and synchronize with each other. These messages are never shown to the user. In the Controls section of the Blocks Palette, there are two blocks for sending messages, and one for receiving them:

Block	**Description**
broadcast ?	Sends the specified message to all sprites in the project.
broadcast ? and wait	Sends the specified message to all sprites in the project and then waits. When all of the sprites that listen for that message have finished their work, Scratch runs the blocks that are snapped onto the bottom of the **broadcast and wait** block.
when I receive ?	Starts a new script when a sprite receives the specified message.

To send a message from a sprite: 1. In the **Blocks Palette**, drag a **broadcast** block over to the **Scripts Area** and into position.

2. On the block, click the down arrow, and then either click the name of the message that you want to send, or click **new** to create a new message.

3. If you are creating a new message: in the **Message name** box, type a unique name for the message, and then click **OK**.

To receive a specific message when it occurs:
1. In the **Blocks Palette**, drag a **when I receive block** into free space in the **Scripts Area**.
2. Click the down arrow, and then click the name of the message.

3. Snap blocks onto the bottom of the **when I receive** block. When the message is received, Scratch runs these blocks.

4.8 Sound and Music

There are two ways to add sound effects and music to your project: import .*wav* and MP3 files into Scratch, or create music by playing notes.

Sound files are attached to individual sprites, and only the sprite that the sound is attached to can play it. To import a sound file:
• In the **Scripts Area**, click **Sounds**, and then click **Import**.
There are five blocks in the Sound section of the Blocks Palette for working with imported sound files:
Block Description
play sound ? Starts playing the specified sound. play sound ? until done
Plays the specified sound and until the end.
stop all sounds Stops all sounds that a sprite is playing. change volume by ? Increases or decreases how loud a sound plays. set volume to ? % Sets how loud the sound plays, as a percentage of how loud that sound is when imported.

Play sound and *play sound until done* appear to do the same thing, but there is an important difference between them. If you snap two *play sound* blocks together then Scratch plays both sounds at the same time. But with a *play sound until done* block, Scratch will play the sound file all the way to the end before running the blocks below.

4.9 The Paint Editor

Although it cannot compete with the features of dedicated software packages such as Adobe Photoshop, Scratch's Paint Editor has all of the tools that you need to draw sprites from within Scratch.

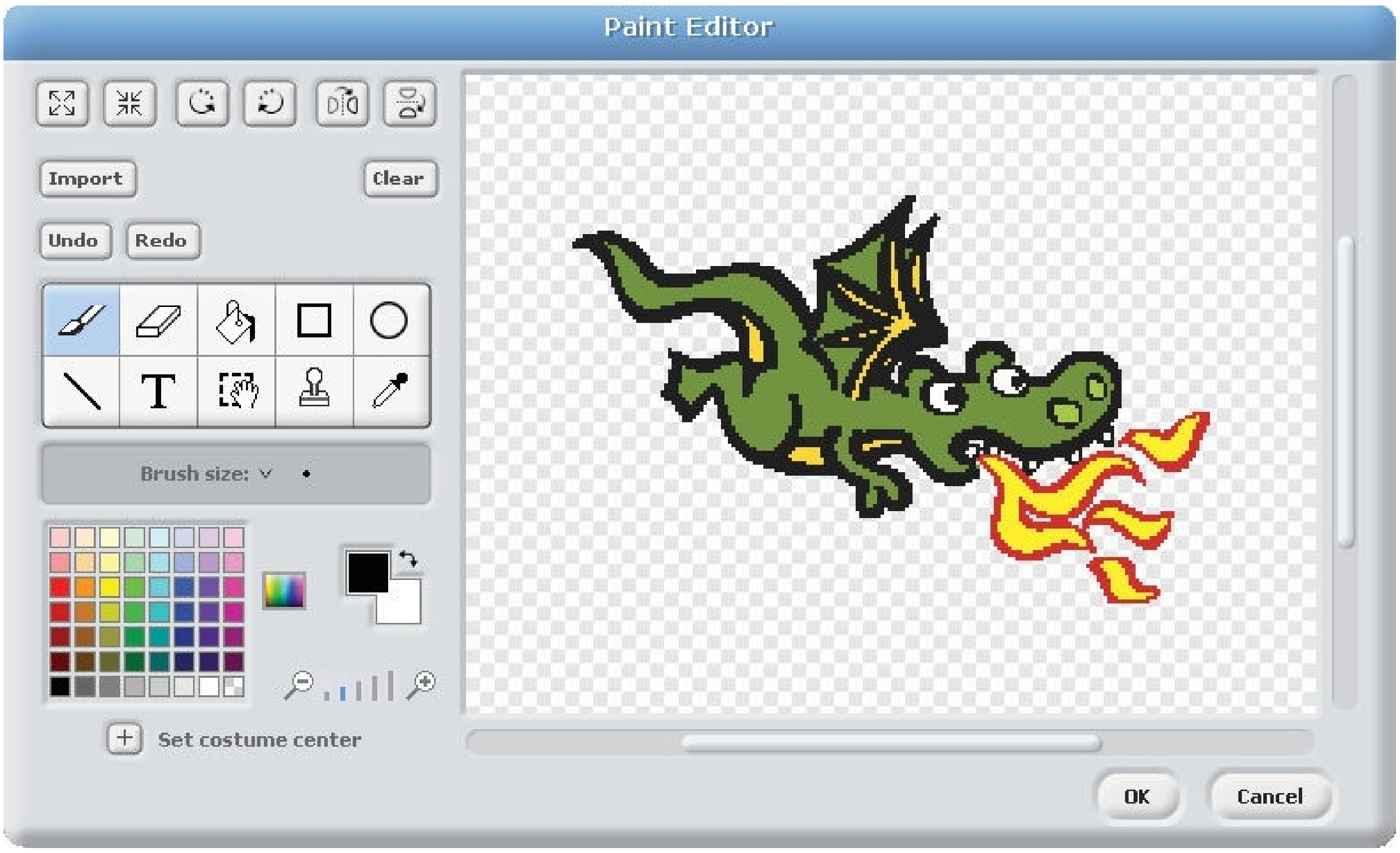


Figure 8. The Paint Editor
1 Tools.
2 Options. This area changes depending on which tool you have selected.
3 Color palettes. Use these options to change the color that you are drawing with.

4 Sets the center point of the picture. You can use this for controlling how sprites rotate and where on the picture the location variables of the sprite refer to.

5 Zoom. Click the - button to look at your picture from further away, or the + button to view your picture close up.
6 Drawing area.

The tools area contains tools for painting and drawing your sprites and backgrounds..

To Do This
Make the picture bigger In the **Tools** area, click the **Grow** button.
Make the picture smaller In the **Tools** area, click the **Shrink** button.

Rotate the picture counterclockwise
In the **Tools** area, click the **Rotate counter-clock-wise** button.

Rotate the picture clockwise
In the **Tools** area, click the **Rotate clock-wise** button.
To Do This

Flip the picture so that it faces in the opposite direction
In the **Tools** area, click the **Flip horizontally** button.

Flip the picture so that it faces the same direction, but is the other way up
In the **Tools** area, click the **Flip vertically** button.

Import a graphic file See below. Clear the drawing area In the **Tools** area, click **Clear**. Reverse the last change you made In the **Tools** area, click **Undo**. Remake the last change that you undid
In the **Tools** area, click **Redo**.
Close the editor without saving the changes you made to the picture
Click **Cancel**.

The ten larger buttons in the tools area are selected by clicking them, but only activate when you do something in the drawing area.

Tool Description
Paintbrush Draw freehand shapes in the drawing area. Eraser Click and hold the left mouse button in the drawing area to erase parts of the picture under the mouse pointer.
Fill Fills an area with a color or gradient. Rectangle Draws rectangles and squares when you click and drag the mouse pointer around in the drawing area.
Ellipse Draws ellipses and circles when you click and drag the mouse pointer around in the drawing area. Line Draws a straight line between two points. Text Type text onto the picture.
Tool Description

Selection Click and drag out a rectangle to select a part of the picture. You can then move, rotate or change this selection without affecting the rest.

Stamp Click and drag out a rectangle to select a part of the picture and copy it into memory. You can then paste it down onto the picture repeatedly. To stop stamping, select another tool.

Eyedropper Click anywhere in the drawing area. The eyedropper sets the current color in the color palette area to match whatever is under the mouse pointer.

Using Transparencies
The background of the drawing area is a grey and white checkerboard pattern. This indicates that the area is transparent. Sprites and backgrounds that are underneath the sprite on the Stage will show through these transparent areas.

When using the Paint Editor to draw or change sprites, anything in the drawing area that is not transparent is part of the sprite.

Importing Your Graphics
If you have a graphic file that you want to use as a sprite, costume, or background then you can import it into the Paint Editor.

To import a *.jpg* or *.png* file:
1. In the **Paint Editor**, click **Import**.
2. Browse to the file, click it, and then click **OK**.

If you are importing graphic files for use as sprites, you may need to remove the solid-color background in the Paint Editor. In the color palettes area, click the transparent color (the grey and white checkerboard pattern) and then use the *Fill* tool to clear the sprite’s background.

4.10 Automatic Startup of Scratch Projects

Scratch has a full-screen option that it calls “presentation mode”, where only the Stage and three buttons are shown on the screen. By editing a few configuration files in Raspbian, you can open a project in presentation mode when the Pi starts.

There are two steps you should take before you do this. First, save your project in a folder that does not contain spaces. For example, */home/pi/MyGame.sb*. Next, you will need to disable remote sensor connections for your project. Open the file in Scratch, and then:

1. In the **Blocks Palette**, click **Sensing**.

2. Right-click one of the two sensor value blocks at the bottom of the **Blocks Palette**, and then click **disable remote sensor connections**.

3. On the **Menu**, click **File** and then click **Save**. Now, use the raspi-config tool to change the boot order:

1. On the desktop, click **LXTerminal**.
2. Type the following command and then press **Enter**: `sudo raspi-config`

3. Use the **Down Arrow** key to highlight **Enable Boot to Desktop/Scratch**, and then press **Enter**.

4. Press the **Down Arrow** key to highlight **Scratch Start the Scratch programming environment upon boot**, and then press **Enter**.

5. Press the **Right Arrow** key twice to highlight **<Finish>**, and then press **Enter**.

6. Press **Enter**.
7. When Scratch appears, press **Ctrl + Alt + F1**.
8. Press **Ctrl + C**.
9. When prompted, press **Ctrl + C**.
10. Type the following command and then press **Enter**:

`sudo nano /etc/profile.d/boottoscratch.sh` 11. On the line that ends with *xinit dev/stdin*, change it to *scratch presentation* followed by the path to your
Scratch project. For example:
`scratch presentation /home/pi/MyGame.sb` 12. Press **Ctrl + O**.
13. Press **Enter**.
14. Press **Ctrl + X**.
15. Type the following command and then press **Enter**:
`sudo shutdown –r now`

4.11 Sharing Your Finished Projects

When you are finished with a Scratch project, you can share it with the world. However, the *.sb* files that Scratch uses can only be opened in the Scratch environment.

There are two other ways that you can distribute your projects:

Putting Your Project on the Scratch Website
The official Scratch website is at *http://scratch.mit.edu* and if you are a registered user then you can upload your Scratch projects. Visitors to the site can then play your games in any web browser that supports Adobe Flash.

Converting Your Scratch Project Files to Executables Several Scratch users ("Scratchers") have written software to convert *.sb* files into standalone executables. You can find an up to date list of all of these types of conversion tools on the Scratch wiki at *http://wiki.scratch.mit.edu/wiki/Porting_Scratch_Projects*

5 – Building an Arcade Game in Scratch

In this tutorial, you can see how to build a very basic version of the arcade game Double Dragon®[1]. Double Dragon® is a sidescrolling action game in which the player takes control of a character and must walk from the left-hand side of the level, to the right. In the way are numerous enemies that the player must defeat by punching and kicking. If you are not familiar with this game, it might be useful to watch a few videos of it on YouTube.

This project pushes Scratch on the Raspberry Pi to its limits. If Scratch struggles to run the game when you click the green flag, try Scratch in “presentation mode”.

Compared to many of the example games that you find on the Scratch website, the game you are building may seem complicated. It is intended to help you learn Scratch, having a working game at the end is a bonus.

5.1 The Title Screen

Even simple games usually start by showing some kind of title screen to the player. When the game ends, the title screen appears again so that the player can have another go.

To begin, start Scratch on Raspbian and delete the cat: 1. On the desktop, double-click **Scratch**.
2. In the **Sprite List**, right-click **Sprite1**, and then click **delete**.
The Stage's background color should match the background color of the logos and graphics that you use. To change it: 1. In the **Sprite List**, click **Stage**.
2. In the **Scripts Area**, click **Backgrounds**, and then click the **Edit** button next to **background1**. 1. Double Dragon is a registered trademark of Bally Gaming, Inc. 3. In the **Paint Editor**, in the **Color Palettes** area, click the black square in the grid of colors.
4. In the **Tools** area, click the **Fill** tool, and then click in the drawing area.
5. Click **OK**.
To create a line of text that tells the player to press the Space key:
1. In the **Sprite List**, click **Paint new sprite**.
2. In the **Tools** area, click the **Zoom out** button so that you can see the entire drawing area.

3. In the **Tools** area, click the **Text** tool, and then click the white square in the grid of colors in the **Color Palettes** area.

4. Type *Press Space*, and then click **OK**.
5. On the **Stage**, drag the text into position.

6. In the **Scripts Area**, click the box that contains **Sprite1** and change the name of the sprite to *PressSpace*.

You need a logo graphic for your title screen. Click **Paint a new sprite** and either draw a logo in the Paint Editor, or import one. Rename this sprite *TitleLogo* using the box in the Scripts Area.

To tell the two sprites on the title screen to fade in when the Scratch project begins, you might normally use the *when <green flag> clicked* block. However, the title screen needs to re-appear at the end of the game and the green flag is not clicked then.

To create a new start point for the title screen, you can use a *broadcast* block attached to the Stage:
1. In the **Sprite List**, click **Stage**.

2. In the **Blocks Palette**, click **Control**, and then drag a *when <green flag> clicked* block to the **Scripts Area**.

3. From the **Blocks Palette**, drag a **broadcast** block over to the **Scripts Area** and snap it into position underneath the **when <green flag> clicked** block.

4. On the **broadcast** block, click the **down arrow**, and then click **new…**
5. In the **Message name** box, type *StartGame* and then click **OK**.

You can use the *brightness* effect to create a simple fade-in animation. You can see this script in Figure 1, in the *when I receive StartGame* scripts.

The *when <green flag> clicked* block is currently the only place where the game is started. But later, other sprites can broadcast "StartGame" to reset the project.

To finish the title screen, you need to detect when the user presses the Space key and then broadcast a message to tell the first level to start. You could also make the "Press Space" sprite flash and play a sound effect when this happens.

To detect the key press:
1. In the **Sprite List**, click **PressSpace**.

2. In the **Blocks Palette**, click **Control**, and then drag a **wait until** block over to the **Scripts Area** and snap it into position below the **repeat** block.

3. In the **Blocks Palette**, click **Sensing**, and then drag a **key space pressed** block over to the **Scripts Area** and drop it onto the six-sided slot on the **wait until** block.

4. In the **Blocks Palette**, click **Control**, and then drag a **broadcast** block over to the **Scripts Area**. Snap it into position below the previous block.

5. On the **broadcast** block, click **StartGame** and then click **new…**

6. In the **Message name** box, type *Level1* and then click **OK**.
If you have a sound effect in *.wav* or *.mp3* format that you want to play when the user presses the Space key on the title screen:

1. In the **Sprite List**, click **PressSpace**.
2. Import the sound file.
3. In the **Scripts Area**, click **Scripts**.

4. In the **Blocks Palette**, click **Sound** and then drag a **play sound** block over to the **Scripts Area** and underneath the **wait until key space pressed** block.

5. Make sure the correct sound file is shown in the **play sound** block.
To hide the *TitleLogo* when *PressSpace* broadcasts "Level1". 1. In the **Sprite List**, click **TitleLogo**.

2. From the **Blocks Palette**, drag a **when I receive** block over to the **Scripts Area**. Click the **down arrow** and then click **Level1**.

3. In the **Blocks Palette**, click **Looks**, and then drag a **hide** block over to the **Scripts Area** and snap it into underneath the previous block.

Figure 1. Original artwork ©1987 Technos Japan Corp.

5.2 The Level Backgrounds

Scrolling is an effect where the background of a game moves in the opposite direction to the player. This makes it look like the player's character is moving when they are actually standing still.

However, Scratch cannot move the project background so your scrolling has to be created using sprites. But it has two further restrictions that make this complicated:

• Sprites cannot move completely off the Stage; and
• The maximum size of a sprite is 480 steps x 360 steps. You cannot use one sprite for the entire background.

Preparing the Background Sprites
You can find an image of the NES version of Double Dragon® on The Spriters Resource (*http://www.spriters-resource.com*). To use this background, you need to make it twice as large – 2030 pixels wide and 384 pixels tall. Then you will have to trim off 24 pixels from the top.

Scratch's Paint Editor is not ideal for working with large files, so you should use a different paint editor, such as Adobe Photoshop, Microsoft Paint, or KolourPaint.

If you are using Adobe Photoshop: when resizing the image, next to *Resample Image* click *Bicubic* and then click *Nearest Neighbor*.

Next, you need to divide the background into four images that measure 480px by 360px and one image that measures 110px by 360px. Save each piece to its own file in portable network graphics (PNG) format.

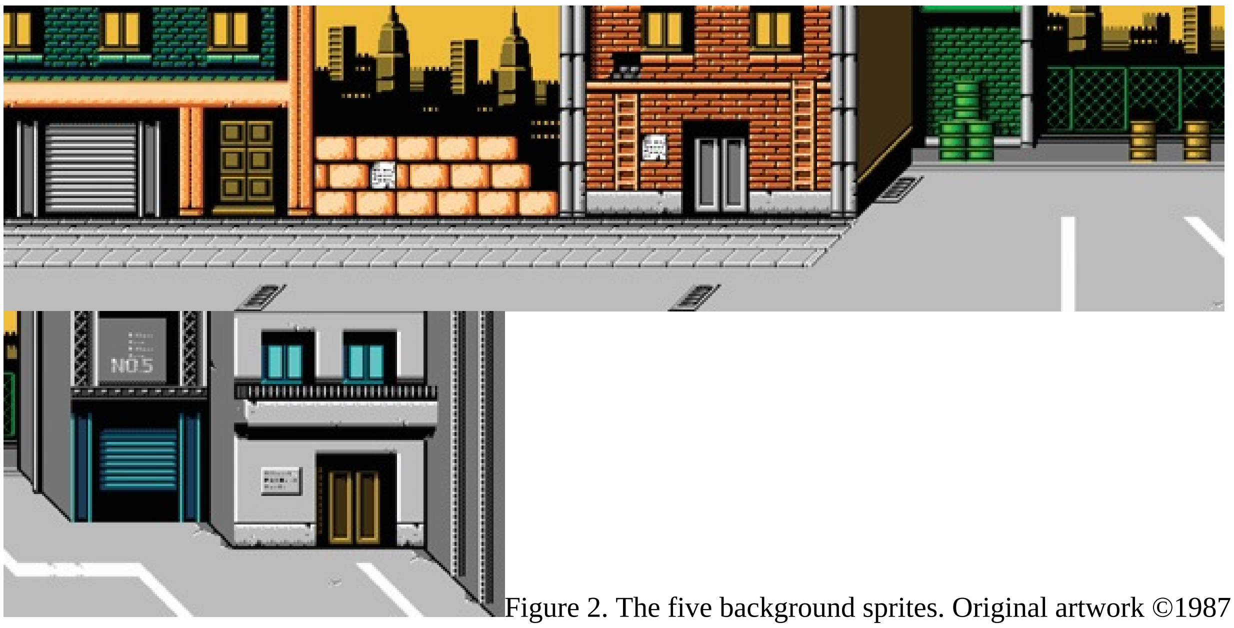


Figure 2. The five background sprites. Original artwork ©1987 Technos Japan Corp.

If you prefer to draw your own background then you can do this in Scratch's Paint Editor instead of importing your graphics in the next section. Try to make the overall shape and design similar to the image above, as this will help in later sections of the tutorial.

Importing the Background Graphics
To import each background sprite:

1. In the **Sprite List**, click **Paint new sprite**, and then click **Import**.
2. Browse to your background file, click it, and then click **OK**.
3. Click **OK**.

4. In the **Sprite List**, click your new sprite and then in the **Scripts Area**, rename the sprite. For example: *L1M1* for the first sprite, *L1M2* for the second, and so on.

Hiding the Background on the Title Screen
The furthest position that a sprite of this size can move to the left is -462. This means that a 480px wide background sprite cannot move far enough to the left to hide the last 18 pixels. The furthest position that a sprite of this size can be positioned to the right is 462. This means that the first 18 pixels of the image are always visible.

These two 18-pixel-wide areas at the edge of the Stage cause visible errors in the appearance of the scrolling and are hidden later.

When the game starts, the title screen broadcasts the message “StartGame”. At this point, the background sprites should not be visible on the screen. You need to add a script to each background sprite so that it hides when that message is received. Make sure that when the background sprites are made visible, that they are on the lowest layer. Do this by bringing each sprite to the front, and then sending it back six layers. To do all of this, add the following script to the *L1M1* sprite:

Now copy this script into the other four background sprites: 1. Right-click the **when I receive StartGame** block, and then click **duplicate**.
2. Move the mouse pointer to the **Sprite List**, and click **L1M2**.
Repeat this process for the remaining background sprites.
Scrolling the Background
The scrolling in this project works like this:
1. There is a variable called *scrollX* that controls the position of the background.
2. When *scrollX* is zero, all of the background sprites are in their default position.
3. Each background sprite calculates its position slightly differently, so that they appear one by one.

4. If *scrollX* is changed, the background sprites recalculate their position. For example, if *scrollX* is changed to 10, then all of the background sprites will move 10 pixels to the left.

5. Sprites that have an x position of greater than 462 will stay in the area on the right side of the Stage.

6. Sprites that have an x position of less than -462 will all be in the unusable area on the left-hand side of the Stage.

Add the variable *scrollX* to the project:
1. In the **Sprite List**, click **Stage**.
2. In the **Blocks Palette**, click **Variables**, and then click **Make a variable**.
3. In the **Variable name** box, type *scrollX* and then click **OK**.
4. In the **Blocks Palette**, click the box next to *scrollX*.

When the title screen broadcasts the message “Level1” to start the game, the first background sprite should make itself visible. Add the following script to the *L1M1* sprite:

Since the first background sprite starts in the center of the Stage, its x position is *0 - scrollX*. So if *scrollX* is 10, then the x position is -10. This is how increasing the value of *scrollX* causes the background sprites to move to the left.

Copy that script into *L1M2*. Since *L1M1* starts at coordinate 0,0 and is 480 steps wide, *L1M2* starts at 480,0. So you need to change the value in the *set x to* block to *480 - scrollX*.

Copy that script into *L1M3*. Since *L1M2* starts at 480,0 and is 480 steps wide, *L1M3* starts at 960,0. So change the *set x to* block to *960 - scrollX*.

Copy the script into the next background sprite, *L1M4*. This starts at 1440,0. So change the *set x to* block to *1440 - scrollX*.

The last background sprite is different. Because it is less wide than the others, its start point is 1735,0. And when this sprite is at coordinate 195,0 it is fully on-screen. As the last sprite in the chain, *L1M5* must stop any more scrolling from happening. To do this, create a variable called *CanScroll*:
1. In the **Blocks Palette**, click **Variables** and then click

Make a variable .
2. In the **Variable name** box, type *CanScroll*.
3. Click the box next to **For all sprites**.
4. Click **OK**.

5. In the **Blocks Palette**, click the box next to
CanScroll to remove it from the Stage.
To make *L1M5* scroll with the others and set *CanScroll* to zero, add this script to *L1M5*:
Before continuing, it is important that these two variables are set to default values when the game begins:

1. In the **Sprite List**, click **Stage**.
2. In the **Blocks Palette**, click **Variables**.

3. Drag a **set ? to 0** block over to the **Scripts Area** and position it above the **broadcast StartGame** block. 4. On the **set ? to 0** block, click the **down arrow** and then click **scrollX**.
5. Drag a **set ? to 0** block over to the **Scripts Area** and snap it below the previous block.
6. On the **set ? to 0** block, click the **down arrow** and then click **CanScroll**.
7. On the **set CanScroll to 0** block, click the white box and change the value to *1*.

Masking the Awkward Areas
To hide the two areas at the far left and far right of the Stage where the scrolling does not work properly, create a new sprite that adds 20 step (or pixel) thick black borders around the Stage. Name this sprite *Border*, and add the following script:

Adding the Background Music
Arcade games from the time period when Double Dragon® was released have background music that plays in a loop. You can use any *.wav* or *.mp3* file for this.

To import a file to use as background music in Scratch: 1. In the **Sprite List**, click **Stage**. 2. In the **Scripts Area**, click **Sounds**, and then click **Import**.

3. Browse to the file, click it, and then click **OK**. To play the sound when Level 1 starts:

1. In the **Scripts Area**, click **Scripts**.
2. In the **Blocks Palette**, click **Control**. Then drag a **when I receive** block over to the **Scripts Area**. 3. On the **when I receive** block, click the **down arrow** and then click **Level1**.
4. Drag a **forever** block over to the **Scripts Area** and attach it underneath the previous block.

5. In the **Blocks Palette**, click **Sound**. Then drag a **play sound until done** block over to the **Scripts Area** and snap it in the middle of the **forever** block.

6. On the **play sound until done** block, click the **down arrow** and then click the name of the background music.

5.3 The Player Sprite

For the main sprite, you need 10 costumes:

• Four “walking” sprites.
• One costume for when the character throws a punch.
• Two costumes for when the character kicks.
• One costume for when the character is hit.
• Two costumes for when the player is knocked over.

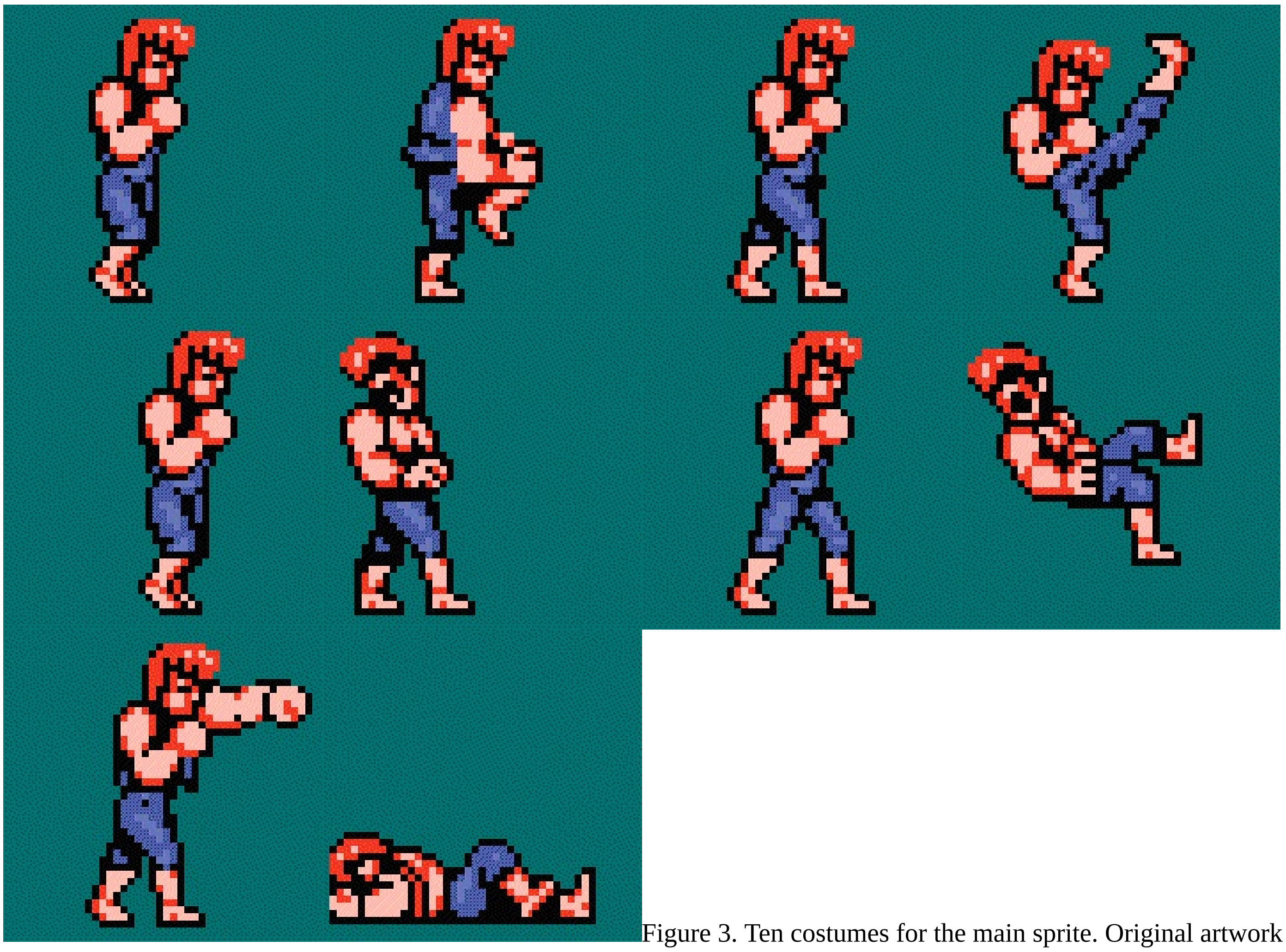


Figure 3. Ten costumes for the main sprite. Original artwork ©1989 Technos Japan Corp.

You can use Scratch’s Paint Editor to create your costumes. However, it is often easier to use an external editor. On The Spriters Resource, you can download images of the main sprite from the NES game. However, many of the enemies are not available. Instead, use the sprites from Double Dragon 2.

When preparing the costumes, keep them on a solid-color background and make each graphic file the same size. Save each costume to a new file in portable network graphic (PNG) format.

Create a new sprite in Scratch by importing the first costume. Name this sprite *Billy*, and then import the other costumes. This tutorial assumes that the costumes are named *Billy-1*, *Billy-2*, *Billy-3*, and so on. You may need to rename the first costume.

When you have imported all of the costumes, you can remove the solid-color background

using the *Fill* tool in the Paint Editor, without Scratch moving the sprites.
Finally, drag the *Billy* sprite into its starting position on the Stage. And then in the Scripts Area, click the *only face left-right* button.

Hiding the Sprite on the Title Screen
The main sprite should not be visible until the title screen broadcasts the message “Level1”. Add these scripts to *Billy*:

Controlling and Moving the Main Sprite
The player will control an invisible “ghost” sprite which tells the *Billy* sprite to move by broadcasting messages. Add these blocks:
The *if* block and costume changes cycle the sprite through the four frames of animation that show the character walking.

The ghost sprite broadcasts two other messages: *FakeWalk* and *MoveToGBilly*. These are used later, so add these two scripts to the *Billy* sprite:

The ghost sprite is not created until section 5.4 Collision Detection on page 109. If you want to test the movement with the keyboard, add scripts that *broadcast and wait* the move messages when the arrow keys are pressed.

When the player presses the Z key to punch or the X key to kick, the *Billy* sprite changes costume. It then waits, so the player has time to see the attack and enemies have a chance to be hit, before changing the costume again. Note that kicking has an extra costume (an extra frame of animation).

Add these scripts to *Billy*:

At this point, the player sprite can move across the Stage. But it can move anywhere on the screen (including up into the air and through walls) and the movement of the sprite does not cause the background to scroll. These limitations are addressed in section 5.4 Collision Detection on page 109.

Adding Variables for Health, Lives Remaining, and the Player’s Score
Since the *Billy* sprite represents the player in the game, now is a good time to create three variables that are needed for the game to work: *Health*, *Lives*, and *Score*.

Add these three variables and select the *For all sprites* option so that all sprites can see and modify the variables.

Health should never be displayed on the Stage. *Lives* and *Score* should be displayed on the Stage when the level starts, but not on the title screen. So:

1. In the **Sprite List**, click **Stage**.

2. In the **Blocks Palette**, click **Variables**, and then drag a **hide variable** block over to the **Scripts Area** and snap it into position underneath the **stop all sounds** block.[1]

3. On the **hide variable** block, click the **down arrow** and then click **Lives**.

4. From the **Blocks Palette**, drag a **hide variable** block over to the **Scripts Area** and snap it into position underneath the previous block.

5. On the **hide variable** block, click the **down arrow** and then click **Score**.

Then add three *set variable to* blocks below the *hide variable* blocks that you have created. Use the first to set *CanScroll* to *1*, the second to set *Health* to *5*, and the third to set *Lives* to *3*. Then

1. From the **Blocks Palette**, drag a **show variable** block over to the **Scripts Area** and snap it into position before the **forever** block in the **when I receive Level1** script.

1. If you do not have a “stop all sounds” block because you did not add background music, snap it to the bottom of the “broadcast StartGame” block.

2. On the **show variable** block, click the **down arrow** and then click **Score**.

3. From the **Blocks Palette**, drag a **show variable** block over to the **Scripts Area** and snap it into position underneath the previous block.

4. On the **show variable** block, click the **down arrow** and then click **Lives**.

5.4 Collision Detection

To prevent the player sprite from moving through walls and into areas that it is not supposed to move, you can use what Scratchers call a “wall-sensing sprite”.

This “ghost” sprite is invisible and moves ahead of the player sprite. When the player presses an arrow key on the keyboard, the ghost sprite moves. If it collides with a wall, or an area that the player sprite is not supposed to go, then it moves back to its previous position. If it does not collide with a wall, it tells the player sprite to move to the same location.

To make this work, create five new sprites (“masks”) – one for each of the five backgrounds. These new sprites are also invisible, but their costume shows the areas that the player sprite cannot move. The areas in which the player sprite can move are transparent.

You can create these masks in any image editor, but you can also do this using Scratch’s built-in tools.
To create the mask for the first background sprite, *L1M1*: 1. In the **Sprite List**, right-click **L1M1** and then click **duplicate**.
2. Click the new sprite and then rename it *ML1M1*.

3. In the **Scripts Area**, remove the **go to front** and **go back 6 layers** blocks from the **when I receive StartGame** script.

4. In the **Scripts Area**, click **Costumes**, and then click **Edit**.

5. Using a single, solid color (for example, bright green), draw shapes over the areas that the player sprite cannot walk.

6. Using the **Eraser** tool and **Selection** tool, delete the parts of the image where the player can walk. These areas must be transparent.

Repeat this process for the other background sprites – for the second mask, copy the second the background; for the third mask, copy the third background, and so on. Name the masks *ML1M2*, *ML1M3*, *ML1M4*, *ML1M5* respectively.

Make sure all of the areas in your mask are joined. Do not have two separate pieces of green – it will work, but causes the collision detection to run much too slowly for this project.

Because you copied most of the blocks from the background sprites, these masks will scroll with the background. It does not matter what layer they appear on as you will not see them. Scratch cannot detect collisions between hidden objects, and so use the *ghost* effect to make the masks invisible.

In the *when I receive Level1* scripts on each of the masks: 1. In the **Blocks Palette**, click **Looks**.

2. Drag a **set color effect to** block over to the **Scripts Area**, and snap it into position above the **show** block.

3. On the **set color effect to** block, click **color** and then click **ghost**.

4. Click the white box, and change the value to *100*. Then create the wall-sensing sprite:

1. Above the **Stage**, click the **green flag** to start the game.
2. Press the **Space** key at the title screen.
3. When the main sprite appears, stop all of the scripts.

4. In the **Sprite List**, click **Paint new sprite**.
5. Click **Import**.

6. Browse to the graphic file for the first costume of the main sprite, click it, and then click **OK**. 7. Draw a solid color rectangle (any color) around the feet of the main sprite.
8. Using the **Eraser** and **Selection** tools, delete the rest of the sprite.

9. Click **OK**.
10. In the **Sprite List**, click the new sprite.
11. In the **Scripts Area**, rename the sprite *GBilly*.
12. Add the following scripts to the **Scripts Area**:

Scratch will put the current sprite coordinates into the *go to x: y:* block. You do not need to change them. The rectangle should be positioned correctly over the feet of the *Billy* sprite. If it is not, you may have to move it in the Paint Editor.

Make a note of the difference between the y position of the ghost sprite and that of the *Billy* sprite. Then go back to the *Billy* sprite and change the number in the *when I receive MoveToGBilly* script from *34* to this value.

Moving the Ghost Sprite
To move the wall-sensing sprite, first:

1. In the **Sprite List**, click **GBilly**.
2. In the **Blocks Palette**, click **Control**.

3. Drag a **forever** block over to the **Scripts Area**, and snap it into position below the **show** block.
The player should not be able to control their movement when they are attacking, or when they have been hit.
1. Drag an **if** block over to the **Scripts Area**, and snap it into position inside the **forever** block.
2. In the **Blocks Palette**, click **Operators**.
3. Drag a **? < ?** block to the **Scripts Area** and snap it into the **if** block.
4. On the **? < ?** block, click the second white box and type the value *5*.
5. In the **Blocks Palette**, click **Sensing**.
6. Drag a **x position of ?** block to the **Scripts Area** and snap it into the first white box on the **? < 5** block.

7. Click **x position** and then click **costume #**.
8. Click the **down arrow** in the box, then click **Billy**.

9. Add the blocks on the following page into the middle of the **if costume # of Billy < 5** block:

Where it says “See Above” on the diagram, you need to create an operator block that reads: touching ML1M1 *or* touching ML1M2 *or* touching ML1M3 *or* touching ML1M4 *or* touching ML1M5.

When building scripts that are made of this many blocks, assemble it in small pieces in unused space in the Scripts Area. Then slowly combine the pieces.

If the right arrow key is pressed then the script checks that the *GBilly* sprite is in the far right of the Stage. If it is, and scrolling is enabled, then there is a block to increment *scrollX* (which in turn causes the background sprites to move to the left). If scrolling the background causes a wall to come in contact with the wallsensing sprite then one of the touching blocks will be *true*. In response, the script pushes the ghost sprite and the *Billy* sprite to the left.

If the player is not at the far right of the Stage, then the *change x by 10* block moves the ghost sprite to the right. It then checks whether it collides with a wall. If it does, it moves back to its previous position. If not, it tells the *Billy* sprite to walk to the right.

To move left or up, add the following blocks underneath the entire *if key right arrow pressed?* block. Duplicate the long operator block, and all of the *touching* blocks, and insert them where the diagram says "See Above".
When moving up, this script relies on the background masks to stop the player sprite. But the script to move down uses the *y* position to restrict the player's movement. Add these blocks underneath the *if key up arrow pressed?* block:

When you move the Billy sprite, your walking sprites may not line up with *GBilly*. To fix this, edit the costumes in the Paint Editor and "nudge" the costumes to the left or right.

5.5 Enemies

When the player reaches set points in the level, the scrolling stops and the game “spawns” enemies that the player must defeat. Only having two enemies on screen at once (like the NES game) is a sensible limitation for this project.

Prepare the graphics for an enemy sprite in a similar way as described in section 5.3 The Player Sprite on page 105. Enemies only need nine costumes, as shown in Figure 4 below.

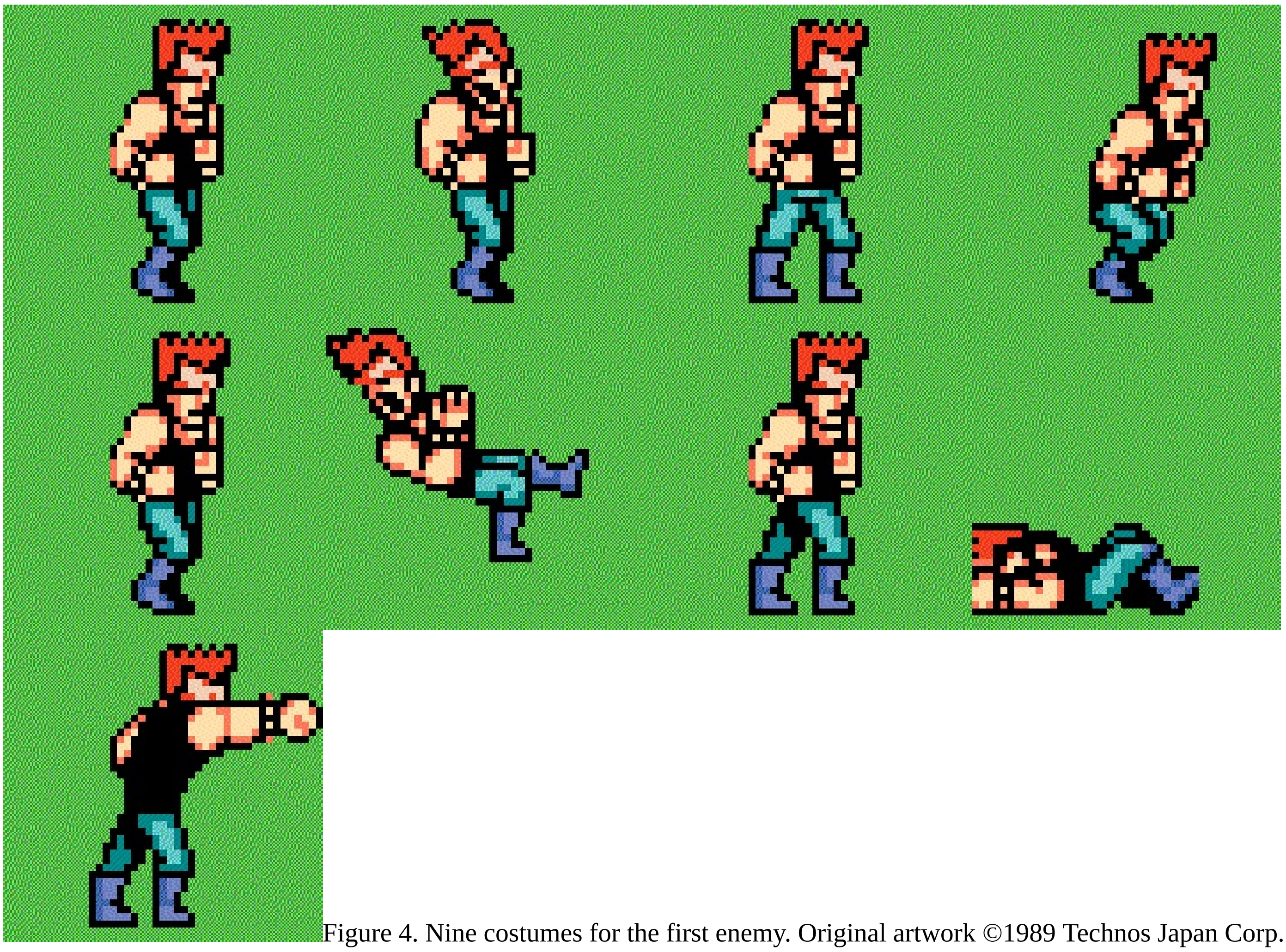


Figure 4. Nine costumes for the first enemy. Original artwork ©1989 Technos Japan Corp.

Name the sprite “Williams1” in Scratch and set it to only face left and right. Drag the sprite where you want it to appear on the Stage.

Add the following script to this sprite:

The enemy needs two variables: *Defeated* and *Hits*. *Defeated* is used to ensure that the sprite only moves and attacks until the player defeats it. *Hits* is the health of the enemy and is reduced when the player hits the sprite. Add these variables to the *Williams1* sprite only by selecting the *For this sprite only* option. You already have the variable *CanScroll* which is used to stop the player scrolling the background when enemies appear or when the end of the level is reached. But you need one that holds how many enemies must be defeated before scrolling is re-enabled. Call this *EnemiesRemaining*, and make it visible to all sprites.

To detect when the player reaches a point when enemies appear, create another invisible sprite. This is sometimes called a “spawn point”.

1. Start the game and then stop it on the first screen.
2. In the **Sprite List**, click **Paint new sprite**.
3. Draw a solid-color rectangle and then click **OK**.

4. This sprite must be tall enough to block the player’s path. Drag the sprite into position on the **Stage**, and edit the costume if it is not large enough.

5. In the **Scripts Area**, rename the sprite *Spawn1*.
6. Make a note of the **x position** of the sprite.
7. Add the following scripts to the spawn point:

The position of the spawn point is calculated using *scrollX*, in the same way as the background sprites and masks are positioned.

Assuming the x position of the spawn sprite is 88 and the first screen starts at coordinate 0,0 then the position of the spawn sprite can be calculated as *88 - scrollX*.

When the spawn point touches the *Billy* sprite, it: 1. Sets *CanScroll* to 0 to stop the background
scrolling.

2. Sets *EnemiesRemaining* to 1.
3. Broadcasts the message “Spawn1”.

4. Hides itself so that no further collisions between it and the *Billy* sprite are detected. Now you need to make *Williams1* respond to this message. Add these two scripts to it:

To create a new spawn point for the second screen, calculate the position of the new sprite with the second background sprite fully in view. The second background sprite starts at 480,0, so the position of the second spawn point is the x value (from the Scripts Area) plus 480 and then minus *scrollX*.

Making Williams Attack

To make the *Williams1* sprite attack the player, you need to add a loop that runs until the sprite is defeated. This loop:

1. Points the sprite in the direction of the *Billy* sprite.
2. Calculates if it is in striking range and if so, punches.
3. Walks a few steps if it is not in striking range.

Add the following blocks underneath the *go to x: y:* block:

Eventually, there will be two copies of this sprite on the screen at the same time. To avoid them moving identically, notice how the script picks its direction by looking at *Billy* and then changing the direction by a random number of degrees. And the sprite intentionally moves slower than the player’s character – otherwise it would be impossible to escape and the game would be very difficult.

Updating the Order of Layers

Because of the perspective used in the background graphics, when the sprites move up the screen they are actually moving into the distance. When the sprites move down the screen, they are coming closer. However, Scratch does not know that sprites with a greater y position should be behind other sprites.

The solution is to check the y position of sprites, and use the *go back ? layers* block to re-order the sprite layers. You also need to allow for sprites disappearing when they are defeated, and bear in mind that the names of enemy sprites may change.

One way to do this is to make a “list” of the sprites that are on the screen. Lists store strings and numbers, and give each item in the list a number. The first item is number 1. The second is number 2, and so on. Where this becomes useful, is that you can use an item in a list with many of the blocks in Scratch; in particular, the *? of ?* block in the Sensing section of the Blocks Palette.

First, make a list called *Mobs* (short for “mobiles”) on the Stage (so that it is accessible by all sprites).
To add *Billy* to the *Mobs* list:

1. In the **Sprite List**, click **Billy**.
2. In the **Blocks Palette**, click **Variables**.

3. Drag an **insert ? at ? of ?** block over to the **Scripts Area** and snap it into position between the **when I receive Level1** block and the **show** block.

4. On the **insert ? at ? of ?** block, click the first box and then type *Billy*.
5. Click the second box and then type *1*.
6. Click the third box and then click **Mobs**. Do the same for the *Williams1* sprite:

1. In the **Sprite List**, click **Williams1**.
2. In the **Blocks Palette**, click **Variables**.

3. Drag an **insert ? at ? of ?** block over to the **Scripts Area** and snap it into position between the **when I receive Spawn1** block and the **show** block.

4. On the **insert ? at ? of ?** block, click the first box and then type *Williams1*.
5. Click the second box and then click **last**.
6. Click the third box and then click **Mobs**.

The script that looks at the *Mobs* list and determines the order in which sprites are placed on layers will be on the Stage. For it to work, you need two other lists: *ZSprites* and *OZSprites*. Create these two lists in the same way that you created the *Mobs* list.

You also need to clear *Mobs* at the start of the game:

1. In the **Sprite List**, click **Stage**.
2. In the **Blocks Palette**, click **Control**.

3. Drag a **when I receive** block to the **Scripts Area**, and change it to be *when I receive StartGame*. 4. In the **Blocks Palette**, click **Variables**.

5. Drag a **delete ? of ?** block over to the **Scripts Area** and snap it into position underneath the previous block.

6. Click the **down arrow** on the first box and then click **all**.
7. Click the **down arrow** on the second box and then click **Mobs**.
Create the script to order the player sprite and active enemies: 1. In the **Blocks Palette**, click **Control**, and then drag a **when I receive** block to the **Scripts Area**.
2. Change the selection to **Level1**.

3. Drag a **forever** block over to the **Scripts Area** and snap it into position underneath the previous block. 4. Add the following blocks to the middle of the **forever** block:
ZSprites is used to store the sprite names in the order that they should be displayed on the Stage.

The *if* block checks to see whether the sprite that is second in the *Mobs* list has a y position that is less than the item in position one of the *ZSprites* list.[1] If it does, then the sprite is lower down on the Stage and should be displayed in the front, so the script inserts the second item from *Mobs* into the first position of *ZSprites*. This pushes *Billy* into second position in *ZSprites*. If it does not have a lower y position, the script puts the item in *ZSprites* after *Billy*.

The next *if* block checks whether the sprite that is third in the *Mobs* list has a y position that is less than the first item in *ZSprites*. If it does, the script puts this item first. If it does not, the script checks whether it has a lower y position than the second item – if it does, the script adds the item to *ZSprites* as the second item; otherwise the item is added to the end of the list. It does not matter if there is only two (or one) items in the list.

1. The item in position one is always the Billy sprite.

Add the next half of the script inside the *forever* block, but below the previous blocks. Where the diagram says “See Above”, create an operator block that reads:
not (item[1] of ZSprites = item[1] of OZSprites and item[2] of ZSprites = item[2] of OZSprites and item[3] of ZSprites = item[3] of OZSprites)

This reduces the number of times that Scratch re-orders the layers. The *OZSprites* list contains a copy of the *ZSprites* list from the last time the script was run. If *ZSprites* has changed, then two things happen: the script copies the items from *ZSprites* into *OZSprites*, and then broadcasts three messages to tell the sprites which layers to move to.

It builds a message by combining the name of the sprite with the strings “GF”, “GL2”, or “GL3”, depending on which layer the sprite should move to. You need to add three scripts to *Billy*: Copy these into *Williams1*, and then change the *when I receive* blocks to “GFWilliams1”, “GL2Williams1”, and “GL3Williams1”.

Layer re-ordering is not instantaneous as the Raspberry Pi is not powerful enough. The approach may work very quickly in other projects, but if it slows this game down too much then remove it.

Hitting Williams

Enemies can have any name, so it is easier for them to detect the collisions than to have the player sprite do it. They need to listen to the player’s attack keys, so add these blocks to *Williams1*:

There are also two changes that you need to make to the *when I receive Spawn1* script. This will reduce how frequently the enemy attacks:
1. Create an **if** block that reads *if costume # < 5*. Then

duplicate it.

2. Drag one of the **if** blocks directly underneath the **repeat until Defeated = 1** block, so that it is inside the loop and surrounds the other blocks in the loop.

3. Drag the other **if** block directly underneath the **repeat 4** block, so that it is inside the loop and surrounds all of the other blocks in the loop.

4. Add a *wait 0.1 secs* to the bottom – inside the **repeat until Defeated = 1** block but underneath the **if costume # < 5** block.

To detect when the enemy is defeated and reactivate scrolling, create these blocks underneath the *switch costume to Williams1* block in the *when Z key pressed* script.

The final block moves the sprite to a known location off the Stage – this is to allow the script that re-orders the sprite layers to remove "dead" sprites from the *Mobs* list. You need to add a few blocks for this to happen. Select the Stage in the Sprite List, and then add the following blocks as the first blocks inside the *forever* loop:

To finish the scripts for the *Williams1* sprite, click it on the Stage. Then, duplicate the whole script that begins with *when Z key pressed* and make the following changes:

1. Change *when Z key pressed* to *when X key pressed*. 2. Change the *switch to costume Williams-6* blocks to *switch to costume Williams-7*.
3. Change the *change by Hits by -1* blocks to *change by Hits by -2*.
4. Change the *if costume # = 6* block to *7*.
5. Add a *wait 0.1 secs* block to the top of the script.

Hitting the Player
When the *Billy* sprite is hit, the variable *Health* must be decremented. When it reaches zero, the sprite is knocked back 50 steps. However, this could cause the player to be knocked off the screen, and the ghost sprite would be in the wrong position. To fix this, the spawn point that triggers the enemies declares an *x* and y position on the Stage that is safe for the player. When they lose a life, a script tells the ghost sprite to move to this safe location. The ghost sprite then tells the *Billy* sprite to update its position.

Two new variables are needed for this: *RespawnX* and *RespawnY*. Create them on the Stage so that all sprites can see the values. The spawn points must set these variables:

1. In the **Sprite List**, click **Spawn1**.

2. Under the **Broadcast Spawn1** block, create a *set RespawnX to* block and enter the x position where the player sprite will reappear.

3. Create a *set RespawnY to* block and enter the y position where the player sprite will reappear. Now, add the following script to the *Billy* sprite: And this script to the *GBilly* sprite:

5.6 More Enemies

By duplicating the spawn point and *Williams1* sprites, you can fill up the rest of the level with enemies:
1. In **Spawn1**, change **set EnemiesRemaining to 1** to *set EnemiesRemaining to 2*.
2. In the **Sprite List**, right-click **Williams1** and then click **duplicate**.
3. Click the new sprite and rename it to *Williams2*.

4. In the **when I receive Spawn1** script, change **insert Williams1 at last of Mobs** to *insert Williams2 at last of Mobs*.

5. Change the block **when I receive GFWilliams1** to *GFWilliams2*.
6. Change the block **when I receive GL2Williams1** to *GL2Williams2*.
7. Change the block **when I receive GL3Williams1** to *GL3Williams2*.

8. If you want this sprite to spawn at a different location: change the values in the **go to x: y:** block under **when I receive Spawn1**.

9. If you want to give this sprite a slightly different random movement, change the values in **pick random -65 to 65**.

To complete this project, you need to add at least one more enemy – the end of level “boss”. The process is largely the same: 1. In the **Sprite List**, right-click **Spawn1** and then click **duplicate**.
2. Click the new sprite, and then rename it *SpawnE*.

3. Change the block **set x to 88 - scrollX** to *set x to 1700 - scrollX*. This moves the spawn point to the far right of the fourth background sprite.

4. Change the block **set EnemiesRemaining to 2** to *set EnemiesRemaining to 1*.
5. Change the **broadcast** block to send *SpawnE*.

Following the previous instructions, make another copy of *Williams1*. Name this sprite *Williams3* and change the *when I receive Spawn1* block to *when I receive SpawnE*.

In the *when Z key pressed* and *when X key pressed* blocks, replace the *set CanScroll to 1* blocks with *broadcast WinGame* blocks.

5.7 Game Over

The game broadcasts two messages that are not yet received by any sprites: “FailGame” is sent when the player loses all their lives, and “WinGame” is sent when the player defeats the enemy in the last part of the level. You can use these to return to the title screen.

If you are converting your Scratch project to an *.exe*, *.app*, or *.jar* file, then it will run a lot quicker and you may be able to add more into the game. There are a lot of areas in this project that you can extend. For example:

To Do This

Stop the enemies from walking through walls
Use a wall-sensing sprite for each enemy.

Add another level Change the final enemy so that it broadcasts “Level2”.
Add weapons Include an extra set of costumes for the player and enemy sprites that shows them holding a weapon.
Make the animation smoother

Use more costumes, reduce the number of steps in the movements, and the amount of seconds in *wait* blocks.

6 – Programming in Python

Python is a widely-used programming language that is available on most modern operating systems and computers. On the Pi, Python is a great way to write software and games, and to control the GPIO pins.

Unlike programming languages such as C++ or Objective-C, Python is an *interpreted* language. This means that special software reads your code files and runs them. You do not have to *compile* Python files to executables or .app files before you can use them.

When programmers write code, they usually do it in a piece of software called an integrated development environment (IDE). An IDE is a text editor that is packaged with other tools to make writing code a little easier. On Raspbian, there are two Python IDEs on the desktop – IDLE and IDLE3. This is because there are two versions of Python, and they are incompatible with each other. In this chapter, you will learn Python 3 using the IDLE3 IDE.

6.1 Your First Python Program

“Hello, World!” is a short program that people often write as their first program in a new environment or language. It is a test, and only displays the message “Hello, World!” to the user.

In Python, you can write this program with one line of code, and it is a good way of introducing you to the IDE and how to run Python programs.

To start: 1. On the Raspbian desktop, double-click **IDLE3**. 2. Type *print(“Hello, World!”)* and then press **Enter**.

The first thing to note is that the window opened by IDLE3 is titled “Python Shell”. The Python Shell works like the Linux terminal – it runs commands that you type here when you press the Enter key.

Python programs (or “scripts”) are text files that contain all of the commands that you want to run. You can use any text editor to create these files, but there is one built-in to IDLE3:

1. In the **Python Shell**, on the **Menu**, click **File** and then click **New Window**.
2. Type *print(“Hello, World”)*
3. On the **Menu**, click **File** and then click **Save As**. Save your file with the extension *.py*.

4. On the **Menu**, click **Run** and then click **Run Module**. The results of your program are shown in the Python Shell. To run your program from a Linux terminal:

1. On the Raspbian desktop, double-click
LXTerminal.
2. Type *python3* followed by a space, and then the full file name (including the file path) of your *.py* file.

If you want to run your program by clicking it on the desktop then you need to create a *desktop shortcut*. These are short text files that create additional icons on your desktop and describe what happens when the user clicks them.

To create a desktop shortcut for a Python script:
1. On the **LXDE panel**, click the **Menu** button, point to **Accessories**, and then click **Leafpad**.

2. Type the following text into the document. Replace *My Test Script* with what you want to call the icon, and the file name after *python3* with the location of your *.py* program.

```
[Desktop Entry]
Type=Application
Name=My Test Script
Exec=lxterminal —command “python3 /home/pi/ Desktop/test.py”
Terminal=False
Categories=None
```

3. On the Leafpad **Menu**, click **File**, and then click **Save As**.
4. Save the file to your desktop with the extension *.desktop*.

If you double-click the icon on the desktop, you should see a terminal window open and display the text “Hello, World!” However, the terminal window closes as soon as the program completes.

While you are developing your programs, it can be useful to keep the program running (and keep the terminal window open) until the user presses a key. In IDLE3, add the following command to your *.py* file. Put it on a new line underneath *print(“Hello, World!”)*.

```
input(“Press any key to close this window.”)
```

6.2 Python

In the following sections of this chapter, you will learn about the features of the Python 3 language.

Adding Comments to Python Programs
As your Python scripts become more complicated, you can put in explanations of what parts of your code do. This is to help you in case you need to a change a program that you wrote a long time ago.

To add a comment to a *.py* file:
• On a new line, type a # symbol and then any text.

Performing Basic Arithmetic
You have seen the function *print()* that displays information to the user. From a Python *.py* file, you need to use this to show the result of arithmetic calculations. But from the Python Shell, you can type the expression and the result is automatically shown.

For example:
• In the **Python Shell**, type the following expression and then press **Enter**:

```
5 + 5
```

To Do This
Add two numbers together Type one number followed by a + symbol and then another number.

Subtract the second number from the first
Type the first number followed by a
- symbol and then the second number.

Multiply two numbers together Type one number followed by a * symbol and then another number.

Divide the first number by the second
Type the first number followed by a / symbol and then the second number.

Divide the first number by the second, and return a whole number Type the first number followed by // and then the second number.

Using Variables In programming terms, a *variable* is an area of the computer's memory that you give a name to. You can store information in these variables and access it later.

To create a variable: specify a name, followed by an equals sign, and then the value it should contain. For example:

```
result = 5 + 5 print(result)
```

Variable names must start with a letter (a–z, or A–Z). The remaining characters in the name can be letters, numbers, or an underscore (_). You cannot use punctuation marks or other special characters in the names of variables, and there are several words (such as "print") that you cannot use as a variable name because they are already used for something else.

In Python, variables can hold any type of information. To extend the example above, you could assign a string (a sequence of characters) to the variable result, even when it currently contains a number.

```
result = "Hello"
```

You can use variables wherever you would otherwise type a value. For example, in math expressions such as *x + 5* and *x / y*.

Python is a “weakly typed” language. This means that you do not have to declare what type of data a variable holds. For example, if result contains “Hello” then result+5 is “Hello5”. However, 5+result causes an error when the program is run because Python tries to use result as a number.

There are many types of data in Python, but these are some of the most common:

Type	**Description**
Boolean	Can either be *True* or *False*.
Number	A number or fraction of any size. However, you can specify the type (and size) of number when you need to be specific.
String	A sequence of characters (letters). For example, “Hello, World!” is a string.
List	An ordered sequence of items. Each item can be any type of data (including other lists). See Using Lists and Dictionaries below.
ByteArray	A list of bytes (small, 8-bit numbers). This is mainly used for writing binary files (see section 6.8) and when working with the Pi’s GPIO pins.
Tuple	Like a list, except that the contents of tuples cannot be changed while the program is running.
Set	An unordered list where each item is unique.
Type	**Description**
Dictionary	A collection of key-value pairs. See Using Lists and Dictionaries below.

However, most things in Python are *objects*. You will learn more about these in section 6.6 Classes and Objects on page 141.

Using Lists and Dictionaries

Lists collect items together into a single variable. They are ordered – each slot in the list is given a number starting at zero.

To create a list, use square brackets. And then each item in the list is accessed using square brackets and the item number:

```
Beatles = [“John”, “Paul”, “Pete”, “George”]
print(Beatles[0])
```

The items in a list can be changed:

```
Beatles[2] = “Ringo”
```

Lists are useful when you do not want to give each item a separate variable name, or when the amount of data that you have changes when the program is running.

You can find the length of a list using the function *len()*. For example, *len(Beatles)* gives the result *4*.

To delete an item from a list, use *del()*. When an item is deleted, all of the items that follow it move up one position. So, in this example, deleting the first item makes “Paul” the new first item. “Ringo” would then be second item, and “George” third.

There are also several useful “methods” that you access using a dot after the variable name. For example:

```
Beatles.reverse()
```

Method	**Description**
append(obj)	Adds the specified item to the end of the list.
count(obj)	Counts how many times the specified item appears in the list.
index(obj)	Searches the list for the specified item and returns its position in the list.
insert(position, obj)	Adds an item to the list in the specified position. All list items that follow are moved down a position.
remove(obj)	Removes the specified item from the list. For example: Beatles.remove(“John”).
reverse()	Reverses the order of items in the list.

A dictionary is a special type of list. Instead of accessing the item by its position in the list, you use a name. You create a dictionary using curly braces. For example:

```
Beatles = {“Lead”:“John”,“Bass”:“Paul”,“Drums”:“Pete”,“Guitar:”,“George”}
```

Each item is now made up of a pair of values – a *key* (the name), and a *value*. To access the items in a dictionary, use the key inside square brackets. For example:

```
Beatles[“Drums”] = “Ringo”
```

The *len()* and *del()* functions work the same way for dictionaries as they do for lists. And there are also several useful “methods” that you access using a dot after the variable name.

Method Description
clear() Removes all items from the dictionary. copy() Returns a copy of the dictionary. has_key(key) Returns *True* if the specified key is in the dictionary, or *False* if it is not.
keys() Returns a list of the keys in the dictionary. update(dic) Adds the items from the specified dictionary to the current dictionary.
values() Returns a list of the values in the dictionary.

6.3 Decisions

In Python, making a decision involves calculating whether an expression is *True*, or whether it is *False*. Depending on the result of that calculation, you can run different blocks of code.

The basic decision-making command in Python is the *if* statement.

```
if variable == value: doThis()
andThis()
```

The == operator compares the two values and returns *True* if they are the same, or *False* if they are not. If the result is *True* then Python runs the commands below the *if* statement.

It is very important that you pay attention to the white space that you use around the commands under an *if* statement. Use the Tab key to indent lines that you want to run if the result of the comparison is *True*.

You can extend the *if* statement with another block of code that runs if the two values are not the same:

```
if variable == value: doThisIfTrue()
else:
doThisIfFalse()
```

There are several other operators that you can use in Python:

Operator	**Description**
!=	Returns *True* if two values are not equal, and *False* if they are.
>	Returns *True* if the first number is greater than the second, and *False* if it is not.
<	Returns *True* if the first number is less than the second, and *False* if it is not.
>=	Returns *True* if the first number is greater than or equal to the second, and *False* if it is not.
<=	Returns *True* if the first number is less than or equal to the second, and *False* if it is not.

You can combine two (or more) expressions by surrounding each with parenthesis and then using one of the logical operators below.

For example:

```
if (variable1 == value) or (variable2 == value): doThis()
```

Operator	**Description**
and	Returns *True* if both expressions are *True*, and *False* if either one is *False*.
or	Returns *True* if either of the expressions is *True*. Returns *False* if they are both *False*.

You can use *not()* to reverse the result of an expression, turning *True* to *False* and *False* to *True*. For example:

```
if not(variable == value): doThis()
```

By using correct spacing, you can also put *if* statements inside the code that runs if an earlier *if* statement is *True*. However, using *elif* is often a little easier to read.

The example below runs *doThis1()* if the value of variable is *1*. If it is not, it checks whether the value is *2* and runs *doThis2()*. If it is not *2* either, it checks whether the value is *3* and runs *doThis3()*. If the value is not *1*, *2*, or *3* then it runs *doThisInAllOtherCases()*.

```
if variable == 1: doThis1()

elif variable == 2: doThis2()
elif variable == 3: doThis3()
else:
```

doThisInAllOtherCases()

6.4 Loops

Loops are blocks of code that repeat a sequence of Python commands a set number times, or continually until a certain condition is met.

There are two types of loop in Python:

Loop	**Description**
while	Checks a condition in the same way as the *if* statement, and then runs the code underneath if the condition is *True*. Then it checks the condition again and if it is still *True*, it runs the code underneath again, and so on. The loop ends when the condition is *False*.
for	Runs the block of code for each item in a list or sequence. The loop ends when there are no more items.

There are two main ways to use a *while* loop. To run something a set number of times, use a variable to keep track of how many times the loop is run. For example:

```
count = 0
while (count < 5): print(“Hello”)
count = count + 1
```

This sends the message “Hello” to the terminal while the variable *count* contains a number that is less than 5. In effect, it prints “Hello” five times.

You can also use a *while* loop to run code until something happens. The example below continually prompts the user to type some text and echoes what they type back to them, until they press the Enter key without typing anything.

```
text = “?”
while not(text == ””):
text = input(“Type something (or nothing to quit). >”)
print(text)
```

The *for* loop is for “iterating” over a list or sequence. It runs as many times as there are items in the list, and assigns each one to a variable so that you can access it. For example, using the *Beatles* list that you saw earlier, you can print each item using a *for* loop.

```
Beatles = [“John”, “Paul”, “Ringo”, “George”] for beatle in Beatles:
print(beatle)
```

It is easy to write a loop that never ends. To stop a program that is running in the Python Shell or Linux terminal, press Ctrl + C.

There are two Python commands that you can use inside *while* and *for* loops, and that affect how a loop runs.

Command	**Description**
break	Immediately exits the loop.
continue	Restarts the loop, ignoring any other commands that are underneath the continue command.

6.5 Functions

Functions are one of the ways that you break a Python program into smaller chunks. This can help you manage projects, and it makes it easier for you to reuse parts of one project in another.

You have already seen a few examples of functions built-in to Python, but you can also create your own. When writing a function, you can choose whether it should accept arguments (values that are passed into the function), and whether it should return a value. To create a function in Python:

1. Type the keyword *def* followed by the function's name.
2. Type opening and closing parenthesis – ().

3. If you want to accept arguments: type them between the parentheses and separate each one with a comma.

4. End the line with a colon.
5. (Optional) Add a string on the next line that
describes the function.

6. Place any lines of code you want to include in the function on following lines, and use the Tab key to indent them.

7. If you want to return a value, use the command *return*.
For example:

```
def myFunction():
“A function that prints a message. This line is ignored.” print(“My first function”)
```

```
def myFunction2(val1, val2):
“A function that adds two numbers and returns the result.” return val1 + val2
```

Function names follow the same rules as variable names. They must start a letter (a–z, or A–Z), and the remaining characters in the name can be letters, numbers, or an underscore (_).

To call one of your functions from other parts of your code, use the function name and then parentheses. For example:

```
myFunction()
z = myFunction2(x, y)
```

To specify the arguments in a different order, you can use *keyword arguments*. As shown below, keyword arguments specify the parameter name before the argument.

```
myFunction2(val2=y, val1=x)
```

An argument is a variable or value that you pass into a function. In the examples above, *x* and *y* are arguments. Parameters are part of the function's definition and refer to the names given to the data when it is passed to the function. For example, *val1* and *val2* are parameters.

It is important to realize that when you change an argument inside a function, the value of it also changes outside of the function. This is because Python actually passes a reference to the argument, not a copy of it.

For example:

```
def Inc(val):
val = val + 1
```

```
x = 1
print(x) Inc(x)
print(x)
```

Making Parameters Optional

Sometimes it can be useful to call a function without including all of the arguments. You can do this by specifying the default value that a parameter should have if an argument is not passed into the function.

```
def PrintAndMultiply(val1, val2 = 1):
print(val1 * val2)
```

You can call this function with *PrintAndMultiply(5,2)* to see the value *10*, or just *PrintAndMultiply(5)*. If you do not pass the second argument, then Python assigns the value *1* to *val2* because that is the default specified in the function's definition.

6.6 Classes and Objects

Python is an object-oriented language. Object-oriented programming (OOP) is a way of writing code that treats data and concepts as standalone *objects*. These objects can contain multiple items of data (“fields”), and even functions that operate on that data (“methods”). As a simple example, consider an entry in your phone’s address book or contact list. You can think of each entry as an object, and inside each object there are numerous fields. “First name” is one field, “Last name” is another, and so on.

Introducing Classes

In Python, a *class* is like a blueprint. It describes the structure of objects and defines their methods. To use an object, you need to create an *instance* of the class that describes it.

You create your own classes in Python by using the *class* keyword. For example:

```
class AddressBookEntry:
firstname = ””
lastname = ””
telephone = ””
```

To create an instance of that class and get a usable object:

```
Alice = AddressBookEntry()
```

To access to the fields and methods that is inside an object, use the dot syntax.

```
Alice.firstname
Alice.lastname
```

You can add methods to the class by indenting them. To refer to fields inside the object from methods, use the keyword *self*.

```
class AddressBookEntry:
firstname = ””
lastname = ””
telephone = ””

def printName(self):
print(self.firstname + ” ” + self.lastname)
```

To call the *printName()* method from other parts of your code, specify the variable name, then a dot, and then the method name. For example:

```
Alice.printName()
```

__init__ is a special method that Python runs when an instance of the class is created. You can write your own and use this to ensure that the correct arguments are passed into the code when you try to create an instance of the class. For example:

```
class AddressBookEntry: firstname = ”” lastname = ””
telephone = ””

def __init__ (self, firstname, lastname): self.firstname = firstname
self.lastname = lastname

Mark = AddressBookEntry(firstname=“Mark”, lastname=“White”)
```

Inheriting Fields and Methods

Inheritance is where objects that are similar share parts of their structure. Using the address book example again, you could have a separate object for business contacts. Since this object contains the same basic information (name, phone number, and so on), it can inherit those fields from the standard entry object. But then you can add other fields that are specific to business contacts (such as company name and website).

```
class AddressBookEntry:
firstname = ””
lastname = ””
telephone = ””

def __init__ (self, firstname, lastname):
self.firstname = firstname
self.lastname = lastname

def printName(self):
print(self.firstname + ” ” + self.lastname)
class BusinessContactEntry(AddressBookEntry):
company = ””
```

Create an instance of *BusinessContactEntry* using the syntax:

```
Bob = BusinessContactEntry(firstname=“Bob”, lastname=“Smith”)
```

Even though the *BusinessContactEntry* class only defines the field *company*, the fields *firstname*, *lastname*, and *telephone* are also usable. It inherits these fields from *AddressBookEntry*.

```
Bob.firstname = “Bob”
Bob.company = “Bob’s Widgets, Inc.” Bob.printName()
```

This relationship does not work the other way. Since *AddressBookEntry* does not inherit from *BusinessContactEntry*, you cannot access the *company* field from an instance of *AddressBookEntry*.

Overriding Inherited Methods

In the example above, the *BusinessContactEntry* class inherits the *printName()* method from *AddressBookEntry*. However, if you include a new definition of __*init*__ and *printName()* in *BusinessContactEntry* then you can override the inherited versions.

```
class BusinessContactEntry(AddressBookEntry):
company = ””

def __init__(self, firstname, lastname, company):
self.firstname = firstname
self.lastname = lastname
self.company = company

def printName(self):
print(self.firstname+” “+self.lastname+” (” + self.company + “)”)
```

Where this gets interesting, is if you have a function like the example below:

```
def printEntry(entry): entry.printName()
```

This function accepts either type of object, and does not know the difference. It simply calls the method *printName()* and the two different types of object respond slightly differently.

Hiding Fields and Methods

In Python, you can stop other parts of your code accessing fields and methods in an object by defining them with a name that starts with two underscore characters.

6.7 Modules and Packages

A module is a collection of functions and classes that are related to a specific purpose. You can download modules that are written by other programmers and use them in your projects. Raspbian already has a large number of Python modules preinstalled. To see a list of all the modules that are installed on your Pi:

1. On the desktop, double-click **IDLE3**.
2. In the **Python Shell**, type the following command and then press **Enter**:

```
help(“modules”)
```

Installing a Module

You can find many additional Python modules that you can download and install with the Package Manager in Raspbian.

If you download a module without using the Package Manager, you will have to install it yourself. Modules usually come with an installation script (*setup.py*) that copies the module(s) to the correct location on your system. To run it:

1. On the desktop, double-click **LXTerminal**.
2. Move into the folder that contains the module that you want to install. Use the **cd** command.
3. Type the following command and then press **Enter**:

```
python3 setup.py install
```

Using a Module

Before you can use a class or function from a module, you need to *import* it into your project.

To import the entire module, put an import statement at the top of your *.py* script. For example, to import the *math* module use the statement:

```
import math
```

Then access items from a module using the dot syntax, as you do when working with objects. For example, to call the *floor()* function in the *math* module:

```
math.floor(10.3)
```

If you only want to import a specific item from a module, rather than everything that the module contains, you can use the *from…import* syntax. In the example below, only the *floor()* function is imported and so the call to the *ceil()* function generates an error.

```
from math import floor print(floor(10.3)) print(ceil(10.7))
```

Learning about Packages

Packages group modules together in *namespaces* – organized and hierarchical trees of modules. Working with namespaces is a little like working with the file system, except that you use dots instead of slashes.

You use package names when importing modules and reading their documentation. So you will become familiar with how this works without having to make a special effort.

6.8 File I/O

Working with files and folders from Python is important on Raspbian, because many of the system processes and devices are mapped into the file system.

Opening a File

To read from or write to a file in Python, you need to open it using the built-in function *open()*. In its basic form, *open()* accepts two arguments:

```
open(filename [, mode])
```

If the file is not in the same directory as your Python script, you need to specify the full path to the file.

Mode is optional and if you do not specify it then Python opens the file in read-only mode. This means that you cannot change the file, only read data from it. The most-common values that are used here are:

Mode Description

r Open the file for reading. rb Open the file for reading in binary format.

w Open the file for writing only. This overwrites the existing file or, if it does not exist, creates the file. wb Open the file for writing only, in binary format.

a Open a file for "appending". Anything you write is added to the end. This overwrites the existing file or, if it does not exist, creates the file.

ab Open a file for appending binary data.

Python makes a distinction between working with a text file and working with a binary file. The key difference between the two is that with text files Python deals with the text encoding and line break formats for you. In binary mode, you have access to each byte that makes up the file, and it is up to you to decide how to process them.

When reading and writing to a text file, you can only use *string* objects. When reading and writing to a binary file, use *bytes* and *bytearray* objects.

Reading from a Text File

To read from a file, open it using the mode "r", or any of the other modes that support reading. Then use the method *read()* to read a number of characters from the file.

You must remember to close files once you are finished working with them. Do this with the *close()* method. If you do not close a file then you might be unable to work with the file again, until you restart the Pi.

For example, to read the "message of the day" (MOTD) file:

```
motd = open("/etc/motd", "r")
message = motd.read()
print(message)
motd.close()
```

If you do not specify how many characters to read, *read()* will fetch the entire contents of the file. To read only ten characters:

```
motd.read(10)
```

If you specify more characters to read then the file has left, Python reads as many characters as it can. When Python reaches the end of the file, the string returned by a call to *read()* will have a length of zero.

Writing Text to a File

To create a new text file and write a test message to it, open the file using mode “w”. Then use the method *write()*. For example:

```
test = open(“/home/pi/Desktop/Test.txt”, “w”)
test.write(“Hello from Python!”)
test.close()
```

If you run this example twice, you will see that the text file on your desktop still only contains one sentence.

Appending Text to a File

To add content to the end of a file, use one of the “append” modes when opening the file. For example:

```
test = open(“/home/pi/Desktop/Test.txt”, “a”)
test.write(“Another hello from Python!”)
test.close()
```

If you open the file *Test.txt* now, you should see that the string has been added to the end of the file.

Renaming and Deleting Files

To rename a file, import *os* and then use the *rename()* function:

```
import os
os.rename(“/home/pi/Desktop/Test.txt”, “/home/pi/Desktop/Test2.txt”)
```

To delete a file, use the *remove()* function:

```
import os
os.remove(“/home/pi/Desktop/Test2.txt”)
```

Working with Folders

The *os* module also includes many functions for creating and working with folders. Here are some of the ones you will use often:

Function	**Description**
mkdir(path)	Creates a new folder at the specified path.
makedirs(path)	Creates a new folder at the specified path, and any folders above it that do not exist yet.
listdir(path)	Returns a “list” of the contents of the specified folder.
removedirs(path)	Removes (deletes) the specified folder and everything inside it.
rename(path, path)	Changes the name of a folder.

6.9 Graphical User Interfaces (GUIs)

Although all of the examples in this chapter have assumed that you are running Python programs from the Python Shell or the Linux terminal, it is possible to build full GUIs in Python. To do this, you need to use a GUI toolkit (or “framework”) that supports all of the different types of controls that you need – windows, textboxes, and buttons, for example.

To create a Python program with a GUI that also runs on other operating systems, you have to use a framework that is available on all of the machines on which you want to run your program. A full list of cross-platform GUI toolkits is available at *http://wiki.python.org/moin/GuiProgramming*. Of these, the *TkInter* framework is particularly popular and is already installed on Raspbian.

7 – Controlling Input and Output Pins

One of the differences between the Raspberry Pi and the desktop computers, smartphones, and tablets that you may have used, is that the Pi is designed with a 40-pin connector that you can control from your own software. This makes it similar to development boards like the Arduino, but the Pi has the power and speed of a regular computer.

The 40-pin header is made-up of many general purpose input and output (GPIO) pins. In this context, "general purpose" means that you can use them for whatever you like – for example: turning on light-emitting diodes (LEDs); working with sensors (devices that read things like temperature and light levels); moving servos and motors; and talking to electronic circuits that you build yourself.

In this chapter, you will learn some of the techniques and knowledge that you need to build basic electronic circuits, and then see how to control the GPIO pins from Raspbian.

Poorly designed or incorrectly assembled circuits can damage the Raspberry Pi, and may be hazardous to the person using or building them. Always seek advice from a qualified person if you are not sure about what you are doing.

7.1 Electronic Circuits, Voltage and Current

Electricity is a form of energy that flows in a circuit (a loop). It can be helpful to think of it starting from one connector on a battery or power supply, and then moving around the circuit until it reaches the other connector. Along the way, it provides power to the components that you place in the circuit. If the circuit is broken at any point, then electricity cannot flow and so nothing will receive power.

A *schematic* is a drawing that shows how different components can be connected together. For example:

Figure 1. Three types of circuit – closed (left), open (middle), and short (right).

1 Battery or power source. This supplies the circuit with electricity.
2 Light-emitting diode (LED). When power flows, it lights up.
3 Resistor. See below.

In a *closed* circuit, electricity can flow from the + terminal, to the
- terminal (*ground* in digital electronics). It passes through the LED, which causes it to glow. But in an *open* circuit, electricity cannot flow all the way around to ground so it does not move at all. In this example, the LED does not light. A switch works by opening and closing the circuit.

A *short* circuit can be dangerous. There is a direct connection between the + and - terminals on the battery. With no components in between, there is nothing to use or limit the amount of electricity that flows.

Voltage is the amount of force that pushes electricity around a circuit and it is measured in volts (V). For example, 5 V or 9 V. You will often see 3.3 V written as “3V3” in diagrams where a dot might be difficult to read.

Current is the amount of electrical charge in a circuit. It is current that makes things happen, like lighting an LED. It is measured in amps and this is shown with an A. For example, 1 A or 3 A. But electronic circuits usually use very small amounts of current, and so milliamps (mA, or thousandths of an amp) can be used instead.

The *resistance* of a circuit limits the amount of current that flows and protects components. It is measured in ohms, and written with the symbol Ω. *Resistors* have color-coded stripes that indicate the amount of resistance that they put in a circuit.

7.2 Solderless Breadboards

Breadboards (or “solderless breadboards”) are easy ways of connecting components temporarily.

The various rows of holes are joined together inside the board. On a standard breadboard, there are four long rows. The holes in each row are joined together inside the board, but none of these rows are joined to each other.

In the middle of the board, the holes are divided into columns. Column 15 is highlighted on the diagram below.

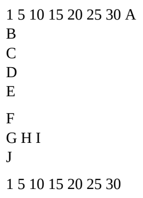


Figure 2. Layout of a typical breadboard

A15–E15 are joined, and F15–J15 are also joined. But the two groups are not connected together or to any other columns.

Connecting the Raspberry Pi to a Breadboard

In electronics, a male connector is a pin, plug, or wire that slots into a *female* connector. Female connectors are usually sockets or jacks. Unlike the designers of other development boards, the Raspberry Pi Foundation use *male* pins for the GPIO headers on the Pi. To connect the Pi to a solderless breadboard, you can either use:

- A 40-pin female-to-female ribbon cable; or
- Individual female-to-male jumper wires.

If you use a female-to-female ribbon cable, you need to use regular male-to-male jumper wires or a Raspberry Pi B+ GPIO breakout board to connect the socket on the cable to holes on the breadboard.

7.3 The GPIO Header

The 40-pin GPIO header is located in the top-left of the Pi. The pins are numbered from left to right, but odd-numbered pins are on the lower row and even-numbered pins are on the top-row.

Figure 3. Location of the GPIO header (left), and the power and ground connections (right)

You can configure most of the GPIO pins as inputs or outputs and control them. But some have additional uses:

Pin #	**Name**	**Description**
3	GPIO2	Input/output. Or I2C wire SDA (I2C1_SDA).
5	GPIO3	Input/output. Or I2C wire SCL (I2C1_SCL).
7	GPIO4	Input/output. Or provides a master clock output (GPCLK0) to external circuits.
8	GPIO14	Input/output. Or UART transmit pin (UART_TXD).
10	GPIO15	Input/output. Or UART receive pin (UART_RXD).
11	GPIO17	Input/output.
12	GPIO18	Input/output.
13	GPIO27	Input/output.
15	GPIO22	Input/output.
16	GPIO23	Input/output.
18	GPIO24	Input/output.
19	GPIO10	Input/output. Or SPI master-output wire (SPI_MOSI).
21	GPIO9	Input/output. Or SPI master-input wire (SPI_MISO).
22	GPIO25	Input/output.
23	GPIO11	Input/output. Or SPI clock wire (SPI_SCLK).
24	GPIO8	Input/output. Or SPI device select 0 (SPI_CE0).
26	GPIO7	Input/output. Or SPI device select 1 (SPI_CE1).
27	ID_SD	Reserved.
28	ID_SC	Reserved.
29	GPIO5	Input/output.
31	GPIO6	Input/output.
32	GPIO12	Input/output.
33	GPIO13	Input/output.
35	GPIO19	Input/output.
36	GPIO16	Input/output.
37	GPIO26	Input/output.
38	GPIO20	Input/output.
40	GPIO21	Input/output.

The GPIO pins are rated for 3.3 V. They are not 5 V tolerant and so applying more than 3.3 V to an input pin can break the Pi.

7.4 Basic Output

Blinking an LED is the electronics equivalent of the “Hello, World!” program. In this example, you will:

1. Make a circuit using an LED and 270 Ω resistor.
2. Learn how to set a GPIO pin to an output.

3. Turn the LED on and off from a terminal window, and from a Python script.

LEDs must be placed in a circuit in the correct direction as electricity only flows one way through them. One leg of an LED is longer than the other – this is called the “anode” and it is the one that receives power. The other leg is called the “cathode”, and this the output leg. In Figure 4 you can also see that on the anode side, the piece of metal inside the LED is smaller.

Figure 4. Light-emitting diodes (LEDs)

It does not matter in which direction you place the resistor.

Unplug the Pi, and then build the circuit in Figure 5. If you do not have a 270 Ω resistor, then you can use a resistor of a slightly greater value. Try to avoid using smaller value resistors – if there is not enough resistance in the circuit then the LED will draw too much current and break.

Each input/output pin on the GPIO header can only supply 16 mA of current, and the Pi can only supply 50 mA of current to all of the pins in total.

Figure 5. Lighting an LED

Check that:

1. The + leg of the LED is on the same row as the 3.3 V wire, and the - leg of the LED is not on the same row.

2. The top wire is connected to pin 1 (3.3 V).
3. The bottom wire is connected to pin 6 (ground).

Connect the Pi to its power source and the LED will turn on. If it does not, unplug the power immediately, and then check that:

1. You have placed the LED in the correct direction. The anode receives power, the cathode allows the power to flow out and into the next component.

2. You have placed the LED, resistors, and wires in the correct holes on the breadboard.
3. You have an appropriate value resistor. If the resistance is too great, you won’t be able to see that the LED is on.

So far, you are using the Pi as a battery. But now that you know your circuit works, you can place the LED under the Pi’s control:

1. Shutdown the Pi and unplug it from the power. 2. Disconnect the red 3.3 V wire from header pin 1, and then connect it to pin 3 (GPIO2).
3. Plug the power back into the Raspberry Pi, and allow Raspbian to load.

When the Pi boots, *GPIO2* is often brought high. A *high* signal is 3.3 V, a *low* signal is connected to ground (0 V).

Controlling GPIO2 from a Terminal Window
You can control the GPIO pins from a terminal window in Raspbian. But only the superuser can do this. For more information about users and superusers, see Understanding Linux Users and Superusers on page 36.

If you want to control the pins from the terminal, you first need to *export* the pin. This creates a special folder in the file system that you can use to change the pin from an input to an output, send the pin high, or send the pin low.

1. On the **LXDE panel**, click the **Menu** button, point to **Accessories**, and then click **Root Terminal**.

2. Type the following command and then press **Enter**: `echo 2 > /sys/class/gpio/export`
3. Type the following command and then press **Enter**: `ls /sys/class/gpio/gpio2`

The files that you see in the *gpio2* directory control the state of *GPIO2*. When exporting GPIO pins, the number you use is the GPIO number not the pin number.

To set *GPIO2* as an output and turn it on:
1. In the **Root Terminal**, type the following command and then press **Enter**:
`echo out > /sys/class/gpio/gpio2/direction`
2. Type the following command and then press **Enter**: `echo 1 > /sys/class/gpio/gpio2/value`
To turn *GPIO2* off:
• In the **Root Terminal**, type the following command and then press **Enter**:
`echo 0 > /sys/class/gpio/gpio2/value`
When you are done, it is a good practice to *unexport* the pin so that it is no longer under the control of the file system entries:
• In the **Root Terminal**, type the following command and then press **Enter**:
`echo 2 > /sys/class/gpio/unexport`

As a final note, outputting *0* actually connects the GPIO pin to ground inside the Pi. So anything connected to that pin is also brought to ground.

Controlling GPIO2 from Python 3
From Python, you can control the GPIO pins using the *RPi.GPIO* module. This example sets *GPIO2* as an output, and then blinks the LED by repeatedly turning it on and off again. To do this, you also need to import the *time* module, which contains a function for causing a delay.

1. On the desktop, double-click **IDLE3**.
2. In the **Python Shell**, on the **Menu**, click **File**, and then click **New Window**.
3. On the **Menu**, click **File**, and then click **Save As**. 4. Save the file to your desktop, with the extension *.py*.
5. At the top of the file, type the following statements: `import RPi.GPIO`
`import time`
6. On a new line, type the following statement: `RPi.GPIO.setmode(RPi.GPIO.BCM)`
setMode() is a function in the *RPi.GPIO* module that sets how you are going to refer to pins. There are two options for this:

• *BOARD* states that when you pass the value *2* into functions in the *RPi.GPIO* module, you are referring to pin 2.

• *BCM* states that when you pass the value *2* into functions in the *RPi.GPIO* module, you are referring to GPIO2.

Now you need to configure *GPIO2* as an output:
• On a new line, type the following statement:

```
RPi.GPIO.setup(2, RPi.GPIO.OUT)
```

The function *setup()* accepts two arguments: the first is the pin or GPIO number (depending on whether you use *BOARD* or *BCM* in the call to *setMode()*); and the second can be *RPi.GPIO.IN* or *RPi.GPIO.OUT*. To make a pin an output, use *RPi.GPIO.OUT*.

Now add the following code to the script and save the file:

```
while True:
RPi.GPIO.output(2, True)
time.sleep(1)
RPi.GPIO.output(2, False)
time.sleep(1)
```

The function *output()* sets the output pin high when the second argument is *True*. It sets the output pin low when the second argument is *False*. If you prefer, you can use *RPi.GPIO.HIGH* and *RPi.GPIO.LOW* instead of *True* and *False*.

The call to *sleep()* causes Python to wait one second before moving on to the next instruction. If you want to blink the LED slower, increase the value that you pass into *sleep()*. For example, *sleep(2)*. If you want to blink the LED faster, decrease the value that you pass into *sleep()*. For example, *sleep(0.5)*.

You cannot run this script from IDLE3, because controlling the GPIO pins requires superuser access. To run your script:

1. On the desktop, double-click **LXTerminal**.
2. Browse to your script. For example, if you saved the file as *test.py* on the desktop:

```
cd Desktop
```

3. Type the following command and then press **Enter**.
Change *test.py* to the name of your script.

```
sudo python3 test.py
```

The LED will now blink on and off. Because the *while* loop in this script never receives *False*, the script runs forever. To stop the script, press **Ctrl + C**.

7.5 Basic Input

With the GPIO pins, input involves finding out whether the voltage that is coming into a pin from your external circuit is *high*, or whether it is *low*. In this section, you will see how to detect when a switch is closed. Any switch will work for this, but two-terminal on/off switches and four-terminal tactile push buttons are the ones that people use most often.

With four-terminal push buttons, it can be difficult to see which way round in the circuit you should place them. The top-left and top-right terminals are always joined. The top-left and bottomleft terminals are only joined when the switch is pressed. So if your circuit always detects that the switch is pressed, rotate the button 90° in a clockwise direction.

Do not wire a switch on its own between the 3.3 V power-output pin and a GPIO input. When the switch is pressed, 3.3 V flows into the input pin and brings the input high. But there are two potential problems with this:

1. When the switch is not pressed the input pin can be *floating*. When a pin floats, it is neither high nor low, and can change between the two seemingly at random.

2. When the switch is pressed, there is very little resistance in the circuit and so this can damage the Pi. It is, effectively, a short circuit.

Introducing Pull-Up and Pull-Down Resistors
In the diagram below, when the switch is open, electricity flows through the resistor and into the GPIO pin. So when the switch is open, the GPIO pin is high. When the switch is closed, all of the electricity is drawn to ground, and this creates a low on the GPIO pin. So the pin is always in one or state or the other, it is never floating.

Pulling the line up in this way is called using a *pull-up resistor*.
Figure 6. A switch and a pull-up resistor

A *pull-down* resistor works in the opposite way: when the switch is open, the GPIO pin is tied to ground and is low. But when the switch is closed, electricity flows into the GPIO pin and it is high.

The Raspberry Pi has pull-up and pull-down resistors built-in to all of the GPIO pins. And you can choose which ones you want to use from your software programs.

Building the Circuit
On the Pi, the circuit above has a problem. If the GPIO pin (the middle line on the diagram) is accidently set to a high output (either by the user or by the Pi when it is starting-up) then pressing the switch causes a short circuit and can damage the Pi.

Build the circuit in Figure 7. It relies on the Pi's internal pull-up resistor but also adds a 330 Ω resistor to protect the Pi if the GPIO pin is changed to an output.
If the GPIO is a high output then the electricity flows through the 330 Ω resistor and this is enough to protect the Pi. The resistor must be a high-enough value to protect the Pi but low enough so that electricity prefers to flow to ground when you press the switch, rather than into the GPIO pin.

Figure 7. Basic input – schematic (left), and breadboard (right)

With the GPIO pin configured as an input and the Pi's internal pull-up resistor enabled, if the switch is open then the pull-up brings the GPIO pin high. If the switch is pressed, the connection to ground drains the electricity from the pin and this causes it to go low.

Reading a GPIO Pin from a Terminal Window

To read the state of *GPIO2* from the terminal or shell, you need to export the pin to the file system.

1. On the **LXDE panel**, click the **Menu** button, point to **Accessories**, and then click **Root Terminal**. 2. Type the following command and then press **Enter**:

```
echo 2 > /sys/class/gpio/export
```

To configure the pin as an input and enable the internal pull-up resistor:
1. In the **Root Terminal**, type the following command and then press **Enter**:

```
echo in > /sys/class/gpio/gpio2/direction
```

2. Type the following command and then press **Enter**:

```
echo 0 > /sys/class/gpio/gpio2/active_low
```

To read the pin, you can use the *cat* command, which is used to read files and display them in the terminal:
1. In the **Root Terminal**, type the following command and then press **Enter**:

```
cat /sys/class/gpio/gpio2/value
```

2. Type the following command but do NOT press the Enter key:

```
cat /sys/class/gpio/gpio2/value
```

3. Press the switch (hold it if you are using a push button), and then press **Enter**.

When you are done, unexport the pin so that it is no longer under the control of the file system entries:
• In the **Root Terminal**, type the following command and then press **Enter**:

```
echo 2 > /sys/class/gpio/unexport
```

Reading a GPIO Pin from Python 3

As for output pins, you can use the *RPi.GPIO* module to read from an input pin, and the *setMode()* method to state how you want to refer to pin numbers.

1. Create a new Python script and save it with the extension *.py*.
2. At the top of the script, type the following statements:

```
import RPi.GPIO
RPi.GPIO.setmode(RPi.GPIO.BCM)
```

Now you need to configure *GPIO2* as an input and enable the internal pull-up resistors:
• Add the following statement on one line:

```
RPi.GPIO.setup(2, RPi.GPIO.IN, pull_up_down=RPi.GPIO.PUD_UP)
```

The *pull_up_down* parameter can be one of three values:

Value	Description
RPi.GPIO.PUD_UP	Activates the internal pull-up resistor.
RPi.GPIO.PUD_DOWN	Activates the internal pull-down resistor.
RPi.GPIO.PUD_OFF	Disables the pull-up and pull-down resistors.

Add the remainder of the code to the script:

```
while True:
if RPi.GPIO.input(2) == RPi.GPIO.LOW: print("Switch pressed.") break
```

```
RPi.GPIO.cleanup()
```

This script loops until the switch is pressed and *GPIO2* goes low. At that point, it prints a message to the user, *breaks* from the *while* loop and then releases its control over the GPIO pin.

7.6 Communication between 3.3 V and 5 V Devices

Even though the Pi has a 5 V power supply, it is a 3.3 V device and 3.3 V is the maximum voltage that you can safely connect to its input pins. However, most other popular development boards, sensors, and components are 5 V devices.

To connect 5 V outputs to an input on the Pi, you will need to convert the voltage levels so that high signals are only 3.3 V.

The high signal from the Pi is usually enough to ensure that the 5 V device detects the line as high. This section is primarily concerned with protecting the Pi's input pins.

If you need to convert more than a two or three signals, then a bi-directional logic level converter chip is a useful purchase. But if you only need to convert one or two signals then there several methods you can use. Two are shown here.

1. Using a Voltage Divider

In the example below, the two resistors form a potential divider. This divides the voltage and allows enough of it to pass to ground so that the GPIO pin only receives around 3.3 V. The exact resistor values are not too important – but the first resistor must be slightly over half the value of the second resistor or the divider does not work correctly.

Figure 8. Using resistors for 5 V to 3.3 V interfacing

2. Using a Diode and Pull-Up Resistor

Diodes are passive components that only allow electricity to flow in one direction. On schematics, they resemble an arrow and this indicates the direction in which electricity can flow. On an actual diode, there is a thick line that corresponds to the vertical line on the schematic symbol.

In Figure 9, when the output on the 5 V device is low then it is actually connected to ground. This means that the electricity from the 3.3 V supply passes through the diode and into the 5 V device, pulling low the GPIO pin on the Pi. When the 5 V device outputs a high signal, the diode blocks the voltage. But because the diode is blocking, the 3.3 V cannot flow to the 5 V device and so it flows into the GPIO pin on the Pi – creating a high signal.

Figure 9. Using a diode for 5 V to 3.3 V interfacing

7.7 Serial Peripheral Interface (SPI)

Serial peripheral interface (SPI) is a serial data protocol that is used by microcontrollers and small electronic devices to exchange information. The term “serial” means that each bit of a binary number (for example, the number 255 is made up of eight bits in binary – 11111111) is sent one at a time, on the same wire.

SPI divides devices into two categories: *masters*, and *slaves*. The master is the device that starts the communication. The slave receives instructions from the master and does what it is told. A master can talk to many slave devices, but usually only one at a time.

To use SPI, you need four pins:

Pin Name Description
MOSI Master output, slave input. The master uses this line to send information to the slave. MISO Master input, slave output. The master uses this line to read information from the slave. SCK Serial clock. Each bit is read on the “edge” of the clock signal (see below).

SS Slave select. Use one of these for each slave device in the circuit. The master uses this wire to indicate which slave device should listen and respond to instructions.

When the master sends information to a slave, it usually: 1. Pulls the *SS* line low for the selected slave. 2. Brings the *MOSI* line high if the bit it is sending is *1*; or brings *MOSI* low if the bit is *0*.

3. *Pulses* the *SCK* line. For example, if the clock line is low when SPI is not being used, then pulsing the line involves briefly bringing the clock line high and then bringing it low again.
4. Repeats this process until all of the bits are sent.

5. Brings the *SS* line high.
When the master reads information from a slave, it usually:

1. Pulls the *SS* line low for the selected slave.
2. Pulses the *SCK* line.
3. Reads whether the *MISO* line is high, or low.

4. Repeats this process until it has all of the information that it is expecting.
5. Brings the *SS* line high to end communication with the slave.

With some devices, the master sends a command to the slave device to tell it to send data. In these cases, the master moves between the sending and reading phases without changing the *SS* line.

There is some variety in how different SPI devices expect these two processes to work. Some require that the clock line is high when it is not in-use, and others require it to be low. Some SPI slave devices expect their *SS* line to be high when the master wants to communicate with it, and others expect the *SS* line to be low.

Enabling SPI on the Raspberry Pi

You can use any of the GPIO input/output pins for SPI since it only involves bringing pins high and low, and reading input in the same way as described earlier in this chapter. This approach is often called “bit-banging”. However, five of the Raspberry Pi’s GPIO pins have alternative uses for communicating over SPI. By using these pins you can use prebuilt libraries and tools instead of writing as much code.

The SPI pins on the GPIO header are: MOSI – pin 19; MISO – pin 21; SCK – pin 23. The header has two slave select pins – pin 24 and pin 26.
By default, the SPI functions of these pins are disabled in Raspbian. To enable them:

1. On the Raspbian desktop, double-click **LXTerminal**.
2. Type the following command and then press **Enter**:
```
sudo raspi-config
```
3. Press the **Down Arrow** key seven times to select **Advanced Options**, and then press **Enter**. 4. Press the **Down Arrow** key four times to select **SPI**, and then press **Enter**.

5. Press **Enter**.
6. Press **Enter**.

7. Press the **Right Arrow** key twice to select **<Finish>**. Then press **Enter**.

8. Type the following command and then press **Enter**: `sudo nano /etc/modules`

9. On a new line at the end of the file, add the following text:

```
spi-dev
```
10. Press **Ctrl + O**, and then press **Enter**.
11. Press **Ctrl + X**.
12. Type the following command and then press **Enter**: `sudo shutdown –r now`

When the Pi restarts, the SPI modules load automatically.

Using SPI from Python 3
To use SPI from Python, you need to install the *python3-dev* libraries, and a module for Python that makes the SPI devices accessible.

1. On the Raspbian desktop, double-click
LXTerminal.
2. Type the following command and then press **Enter**:

```
sudo apt-get install python3-dev
```
3. When the installation is complete, type the
following command and then press **Enter**:
`git clone git://github.com/rpodgornypy-spidev` 4. Type the following command and then press **Enter**:
```
cd py-spidev
```
5. Type the following command and then press **Enter**:
```
sudo python3 setup.py install
```

When connecting the Pi to SPI devices, the GPIO pin *MOSI* (master output, slave input) connects to the *SI* (slave input) pin on the external device. Similarly, the GPIO pin *MISO* (master input, slave output) connects to the *SO* (slave output) pin.

To use SPI in Python:
1. On the desktop, double-click **IDLE3**.
2. In the **Python Shell**, on the **Menu**, click **File** and then click **New Window**.
3. Type the following at the start of the script:
```
import spidev
import time
```

Figure 10. Connecting a 23K640 SPI memory chip to the Pi

To create an instance of the *SpiDev* class and open a connection, add the statements:

```
spi = spidev.SpiDev() spi.open(0, 0)
```

The first parameter to *open()* is the device number. Only one SPI master device is available on the GPIO header and so this is always zero. The second parameter is the slave select pin that you want to use. *0* tells the module to use pin 24, and *1* tells it to use pin 26.

To write a byte to the SPI device, you can use the *xfer()* method. For example:

```
spi.xfer([2,0,0,8])
```

The *xfer()* method accepts one argument, and that is an array of bytes. When you call this method, the slave select pin is brought low and then it sends the values in the byte array to the SPI device. The clock signal is generated for you.

You also use the *xfer()* method to read from an SPI device. The slave device expects the master to generate the clock pulses that it needs to transmit data. So you may need to include extra bytes in the call to *xfer()*. For example, if an SPI device expects the master to send the sequence 3,0,0 before it sends back a byte then you would pass an extra 0 into *xfer()*:

```
spi.xfer([3,0,0,0])
```

You can use several properties of the *SpiDev* class to change how the SPI methods work. This is useful when you are using SPI devices that do not follow the usual process.

Property	**Description**
cshigh	When *True*, a high signal is used to tell the slave device that it should listen and respond. When *False*, a low signal is used. *False* is the default.
max_speed_hz	Not all SPI devices can run as fast as the Pi. Decrease the value of this property to slow the SPI transmissions down.
mode	Sets the clock polarity and phase. Can be 0–2.

7.8 I2C Communication

I2C (pronounced eye-too-see) is another serial communications protocol, but it uses fewer wires. The master device controls I2C communications, and you connect all slave devices to the same wires. Each slave device has a unique number (called an “address”) and it only responds to messages that it receives which specify this address. This means that you cannot use two slave devices that have the same address.

The two I2C lines are called *SDA* and *SCL*. You need to use a pull-up resistor on these lines. The value of the resistors is not very important, but around 10 KΩ is generally suitable.

Any GPIO pins can be used for I2C, if you want to write all of the code. But it is a more complicated protocol to implement from scratch. To use prebuilt I2C libraries and modules, the two I2C wires are pin 3 and pin 5 of the GPIO header.

Figure 11. I2C communication between the Pi and multiple devices

Enabling I2C

On Raspbian, I2C is disabled. To enable it, you must first remove I2C from the module “blacklist”. This is a file that stops Raspbian loading certain modules. To remove I2C from the list:

1. On the desktop, double-click **LXTerminal**.
2. Type the following command and then press **Enter**:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

3. If you see lines in this file that read *blacklist spi bcm2708* or *blacklist i2c-bm2708* then add a # symbol at the start of the lines. This makes the line into a comment so that Raspbian ignores it.
4. Press **Ctrl + O**, and then press **Enter**.
5. Press **Ctrl + X**.

To add the *i2c-dev* module to the list of modules that Raspbian loads when the Pi is started:

1. In **LXTerminal**, type the following command and then press **Enter**:

```
sudo nano /etc/modules
```

2. On a new line, add:

```
i2c-dev
```

3. Press **Ctrl + O**, and then press **Enter**.
4. Press **Ctrl + X**.

Now you need to install the *i2c-tools* package and build the *pysmbus* module:

1. In **LXTerminal**, run the following commands one by one:

```
sudo apt-get install i2c-tools
sudo shutdown -r now
```

2. In **LXTerminal**, type the following command and then press **Enter**:

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

3. Run the following commands:

```
sudo apt-get install libi2c-dev
cd /home/pi/Desktop
```

4. Type the following command and then press **Enter**:

```
wget http://ftp.de.debian.org/debian/pool/main/i/i2c-tools/i2c-tools_3.1.0.orig.tar.bz2
```

5. Run the following commands:

```
tar xf i2c-tools_3.1.0.orig.tar.bz2
cd i2c-tools-3.1.0/py-smbus
```

6. Type the following command and then press **Enter**:

```
wget https://www.dropbox.com/s/
am9wb0g3wpixq3k/smbusmodule.c?dl=1 –O smbusmodule.c
```

7. Run the following commands:

```
python3 setup.py build
sudo python3 setup.py install
```

Using I2C from Python 3

At the top of your Python script, you need to import the module *smbus* and create an instance of the *SMBus* class.

In the example below, the code sends the value 88 to the I2C device with the address 23:

```
import smbus
bus = smbus.SMBus(0)
bus.write_byte(23, 88)
```

To read from an I2C device, use the *read_byte()* method and pass the device's address as an argument. For example: `v = bus.read_byte(23)`

7.9 Serial UARTs

A universal asynchronous receiver/transmitter (UART) is a chip that translates the parallel data used by a microprocessor or microcontroller to serial data for use with communications ports.

Serial ports are *asynchronous*, which means that you can send data at the same time as receiving it, and both devices can initiate connections and send data when they want to.

On the Pi, the GPIO header has two UART pins for making connections to other devices using a traditional serial port. These are pin 8 (UART_TXD) and pin 10 (UART_RXD). The TXD pin of the Pi is connected to the RXD pin of the other device, and the RXD pin of the Pi is connected to the TXD pin of the other device. And to make electricity flow, you also need to connect a wire between the ground of each device.

The Pi's serial port uses 3.3 V logic levels. You need logic level converters to connect the Pi to the 5 V ports used by most USB serial port adapters.

In Raspbian, the Pi's serial port is */dev/ttyAMA0*. This device is usually configured for console input and output, and so if you want to control the port from your own programs then you need to change two configuration files:

1. On the Raspbian desktop, double-click
LXTerminal.

2. Type the following command and then press **Enter**: `sudo nano /etc/inittab`

3. On the line that reads *TO: 23: respawn: /sbin/getty*
–L ttyAMA0 115200 vt100, add a # symbol at the start of the line. This makes the line into a comment so that Raspbian ignores it.

4. Press **Ctrl + O**, and then press **Enter**.
5. Press **Ctrl + X**.
6. Type the following command and then press **Enter**: `sudo nano /boot/cmdline.txt`

7. Remove all references to */dev/ttyAMA0* so that the line reads:

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/ mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

8. Press **Ctrl + O**, and then press **Enter**.
9. Press **Ctrl + X**.
10. Type the following command and then press **Enter**: `sudo shutdown –r now`

Making Serial Communications from Python 3
To send and receive data from the serial port in Python, you can use the *pySerial* module:

1. At the top of your script, include the following line: `import serial`
2. Create an instance of the *Serial* class, using the line:

```
ser = serial.Serial("/dev/ttyAMA0", 19200, timeout=0)
```

3. Use the *write()* method to send data. For example: `ser.write("Hello")`

The first parameter of the call to *Serial()* is the name of the serial port device. */dev/ttyAMA0* is the serial port that is available on the Pi's GPIO headers. The second parameter is the connection speed (also known as "baud rate"). The other device needs to open its serial port at the same speed.

The remaining settings that you need to use on the other device are: parity: none; data bits: 8; stop bits: 1; handshaking (hardware flow control): off.

To read from the serial port, you can either call the *read()* method with no arguments to read a single byte, or pass the number of bytes that you want to read into the method. The *timeout* parameter that you specified earlier defines how long the Pi will wait to receive data. If *timeout* is zero then *read()* will wait until it receives all of the bytes that you request.

To wait until the other machine sends a specific number of bytes before you attempt to read from the port, you can use the method *inWaiting()*. This returns the number of bytes that your script has not yet read.

When you are finished with the serial connection, you should close the serial port by calling the *close()* method of the *Serial* class.

8 – Building an IP Camera

IP cameras are a type of camera that you do not need to connect to a computer in order to view the videos and images that they record. Instead, they connect to your local network router and make the output available over Internet protocols (IP). This allows the videos and images to be viewed across the Internet.

Because the Raspberry Pi is small, and has good Internet connectivity, it is well-suited for building this type of camera. In this chapter, you will see how to connect a Pi camera module to the Model B+ and configure Raspbian.

You need:

1. A Raspberry Pi (any model) running Raspbian.
2. An Ethernet or Wi-Fi connection setup on the Pi.
3. A Raspberry Pi Camera Module.

4. Administrator access to your network router (to make the video accessible over the Internet).

8.1 The Pi Camera Module

The Raspberry Pi camera module is a small, high-definition video camera that connects to the Raspberry Pi using the camera serial interface (CSI) connector.

It has a fixed-focus, 5-megapixel sensor, capable of taking still images in resolutions up to 2592 pixels wide and 1944 pixels tall, and video in resolutions up to high-definition 1080p at 30 frames per second.

Connecting the Camera
The camera module uses a long, flexible cable, and this is useful when you need to fit the camera and the Pi into a case. However, this type of cable can be damaged easily. Do not bend or twist the cable excessively, and do not use it as a weight-bearing support.
To connect to the camera module to the Pi:

1. Ensure the Pi is unplugged.
2. Open the CSI connector on the Pi by grasping the top of the CSI connector and pulling it upwards.

3. Take the flexible cable and gently push it down into the CSI connector. It will only push in by a few millimeters. The blue stripe on the Pi end of the cable should face the Ethernet port.

4. Hold the flexible cable in position and then press down on the top of the CSI connector to close it. 5. Reconnect the Pi to its power supply.
Figure 1. Connecting the camera module to the Model B+
Enabling the Camera
By default, the camera is disabled in Raspbian. To enable it: 1. On the Raspbian desktop, double-click
LXTerminal.

2. Type the following command and then press **Enter**: `sudo raspi-config`

3. Press the **Down Arrow** key four times to highlight **Enable Camera**, and then press **Enter**.
4. Press the **Right Arrow** key to highlight **<Enable>**, and then press **Enter**.

5. Press the **Right Arrow** key twice to highlight **<Finish>**, and then press **Enter**.
6. Press **Enter**.

Disabling the LED
The Pi's camera module has a red light-emitting diode (LED) on the board. To disable it:

1. On the Raspbian desktop, double-click
LXTerminal.

2. Type the following command and then press **Enter**: `sudo nano /boot/config.txt`

3. Use the **Down Arrow** key to move to the end of the file, and then add the following text on a new line:

```
disable_camera_led=1
```
4. Press **Ctrl + O** and then press **Enter**.

5. Press **Ctrl + X**.
6. To restart the Pi, type the following command and then press **Enter**:

```
sudo shutdown –r now
```

8.2 Motion

Motion is a command-line tool that is designed for working with cameras from Linux. It takes still images, records videos, detects movement, and streams live video feeds across the Internet.

Motion-mmal is a version of Motion that is designed to work with the Pi camera module. To install Motion-mmal, you must first install several files and pieces of software that it works with.

1. On the Raspbian desktop, double-click **LXTerminal**.
2. Type the following command on one line, and then press **Enter**:

```
sudo apt-get install –y libjpeg62 libjpeg62-dev libavformat53 libavformat-dev libavcodec53 libavcodec-dev libavutil51 libavutil-dev libc6-dev zlib1g-dev libmysqlclient-dev libpq5 libpq-dev
```

3. Change to your desktop folder. For example: `cd /home/pi/Desktop`
4. Type the following command and then press **Enter**:

```
wget https://www.dropbox.com/s/
xdfcxm5hu71s97d/motion-mmal.tar.gz
```

5. Type the following command and then press **Enter**: `tar zxvf motion-mmal.tar.gz`
6. Type the following command and then press **Enter**: `sudo mv motion /usr/bin/motion`
7. Type the following command and then press **Enter**: `sudo mv motion-mmalcam.conf /etc/motion.conf`
8. Type the following command and then press **Enter**: `sudo chown root:root /usr/bin/motion`

Steps 6–8 above move Motion into a system folder and change its owner.

Configuring the Software
Before starting Motion, you should check its configuration settings. When you start Motion with no command-line arguments, it reads its configuration settings from the file */etc/ motion.conf*. To open this file:

1. On the Raspbian desktop, double-click **LXTerminal**.
2. Type the following command and then press **Enter**: `nano /etc/motion.conf`

There are many settings that you can change, but the key ones are:

Setting Description width The width of still images and video frames in pixels.
Setting Description
height The height of still images and video frames in pixels.
framerate The maximum number of frames to capture per second.
rotate You can use this setting to rotate the image. Acceptable values are 0, 90, 180, and 270. target_dir The folder where Motion saves images and videos.
stream_port The port number you want to use when accessing the live stream from other devices.
stream_maxrate The maximum number of frames to send per second to devices viewing the live stream. output_pictures Set this to *off* if you do not want to save pictures when Motion detects movement.

ffmpeg_output_movi es
Set this to *off* if you do not want to record video files when Motion detects movement.

ffmpeg_video_codec If you have difficulty viewing the video files that Motion records, change this to use a different video codec. For example, *msmpeg4*.
max_mpeg_time The maximum length of video that Motion saves to each file before starting a new one.

webcontrol_port The port number for accessing the control panel from a web browser.
webcontrol_localhost When this is *on*, only web browsers running on the Pi can access the control panel.

When taking images or recording video, the larger the image size (*width* and *height*) and the number of images taken (*framerate*) greatly affect how much space the files will use on the SD card. If you are streaming the video feed to other devices, the *width*, *height*, and *stream_maxrate* settings affect how fast each image or frame is sent to connected devices. The default image size of 1024 x 576 is a little large for the Pi to stream to connected web browsers. The size suggested in *motion.conf* of 352 x 288 works much better and at higher frame rates.

If you only want to access the live video feed, it is recommended that you set *ffmpeg_output_movies* and *output_pictures* to *off*. Otherwise you may run out of space on the SD card very quickly.

To change the configuration of Motion: 1. On the Raspbian desktop, double-click **LXTerminal**.

2. Type the following command and then press **Enter**: `nano /etc/motion.conf`
3. Change any settings.
4. Press **Ctrl + O**, and then press **Enter**.
5. Press **Ctrl + X**.

Password-Protecting your Camera Stream
If you want to stop others from accessing your video stream, you can configure Motion to require a username and password.

To do this:
1. On the Raspbian desktop, double-click
LXTerminal.

2. Type the following command and then press **Enter**: `nano /etc/motion.conf`
3. Change the setting *stream_auth_method* to *1*.

4. On the line that starts with *;stream_authentication*, remove the semi-colon and change *username* and *password*.

5. Press **Ctrl + O**, and then press **Enter**.
6. Press **Ctrl + X**.

Starting Motion

With the default settings, you start Motion from the command line. Once the software is configured how you want it, and you have tested it, you can set Motion to start automatically when the Pi starts and loads Raspbian.

To start Motion manually:
1. On the Raspbian desktop, double-click
LXTerminal.
2. Type the following command and then press **Enter**: `motion`
When you want to stop Motion, press **Ctrl + C**.
To configure Motion to start automatically:
1. On the Raspbian desktop, double-click
LXTerminal.

2. Type the following command and then press **Enter**: `nano /etc/motion.conf`
3. Change the setting *daemon* to *on*.
4. Press **Ctrl + O**, and then press **Enter**.
5. Press **Ctrl + X**.
6. Type the following command and then press **Enter**: `sudo nano /etc/rc.local`

7. Before the line that reads *exit 0*, add the following text on its own line:

```
motion
```

8. Press **Ctrl + O**, and then press **Enter**.
9. Press **Ctrl + X**.
10. To restart the Pi, type the following command and then press **Enter**:

```
sudo shutdown –r now
```

Viewing the Live Stream from Devices on your Network To view the video from a device on your local network, ensure Motion is running and then open a web browser on the device.

You need to know the IP address of the Raspberry Pi. For more information, see Finding the IP Address on page 40.

Type *http://* followed by the IP address of your Pi, then a colon and the port number (*stream_port* in *motion.conf*) into the address bar of the web browser. Then press the Enter key.

Internet Explorer on Windows is not able to open the stream this way, but other browsers can. You can also use VLC
(*http://www.videolan.org*) to open the stream using its address.

Viewing the Live Stream over the Internet

To access the live stream from devices outside your local network, configure your router to allow incoming requests on the port*stream_port* (in *motion.conf*). To do this, you will use the Pi's IP address and so it is helpful if the Pi is using a static IP, see Using a Static IP Address on page 41.

You need to create a rule that allows incoming traffic on a TCP port of your choosing (for example, 8081), and states that this traffic should be forwarded to the Pi on port 8081 (or the value of *stream_port* in *motion.conf*).

To access the stream over the Internet, you need to know the external IP address of your router. Open a web browser on a device connected to your network and visit *http:// www.whatsmyip.org*. Then you can open the live stream using an address that begins *http://*, followed by the external IP address of your router, a colon, and then the incoming port number that you set in your router's port forwarding.

When your router restarts or reconnects to the Internet, your IP address may change. Using a dynamic domain name system (DDNS) service provider, you can map your IP address to an Internet domain name and use this domain name to access your live stream.

9 – Building a Smarter Doorbell

A doorbell is a push-button switch that closes a small electrical circuit when a visitor presses the button. With the circuit closed, electricity flows into components that make the chime or buzzing sound.

With a Raspberry Pi, you can make a much smarter doorbell – one which is capable of doing a lot more than playing a sound. Anything that the Pi can do, you can do in response to someone ringing the doorbell.

In this chapter, you will see how to make a prototype Raspberry Pi-powered doorbell.
You need: 1. A Raspberry Pi running Raspbian.
2. A set of speakers. For more information about suitable speakers, see section 9.2.

3. A door chime sound effect as an MP3.
4. A doorbell ("bell push") or momentary push switch.
5. Some wire.
6. One 330 Ω resistor.

7. A solderless breadboard, perfboard, or stripboard for attaching wires and components.

The doorbell on the front of your house is usually wired into the "mains" electrical supply. Do not connect the Pi doorbell to the mains supply, and consult a qualified electrician before attempting to remove your current doorbell.

9.1 The Button Circuit

The push-button circuit used in this project is the same as you can see described in section 7.5 Basic Input on page 160. You can use a push button for this project, or a standard doorbell switch. Doorbells usually have two screw terminals inside, which make electrical connections by tightly holding wires onto metal contacts. To connect a wire to a terminal:

1. Loosen the screw by turning it counterclockwise.
2. Strip 25 mm–35 mm of insulation from the wire.
3. Wrap the exposed part around the screw.

4. Tighten the screw by turning it in a clockwise direction.
Repeat the process for the other terminal.
The direction that you connect the wires to your circuit does not matter.
1. Connect one wire from the bell to *GPIO2* (pin 3). 2. Connect the other wire from the doorbell to one leg of the 330 Ω resistor.
3. Connect the other leg of the resistor to a *ground* on the GPIO header.

9.2 Speakers

There are two types of speaker that work well for this type of Raspberry Pi project: stereo, desktop PC speakers; and speakers that have an HDMI input and built-in amplifier.

Speakers that have an HDMI input extract the audio information from the HDMI signal. If you are using the HDMI output then you do not have to change your configuration to use these speakers.

Stereo PC speakers come in two varieties: passive, and active. Active speakers have their own power supply and can play Active speakers have their own power supply and can play pole 3.5 mm stereo plug that you can connect into the Pi's 3.5 mm output jack.
By default, if you connect an HDMI cable to the Pi then any audio signals will also use that. You can force the audio out through the 3.5 mm output jack with the *amixer* command:

1. On the Raspbian desktop, double-click
LXTerminal.
2. Type the following command and then press **Enter**: sudo amixer cset numid=3 1

9.3 A Basic Doorbell

To make a doorbell, you can use a Python script to detect when the button is pressed and then play a sound.
To begin: 1. On the desktop, double-click **LXTerminal**.
2. To make a new folder for your doorbell project, type the following command and then press **Enter**:

```
mkdir Doorbell
```

3. On the desktop, double-click **IDLE3**.
4. In the **Python Shell**, on the **Menu**, click **File** and then click **New Window**.
5. On the **Menu**, click **File** and then click **Save As**.
6. Save the file to your folder, as *doorbell.py*.
7. Add the Python code below, and then save your script again:

```
import RPi.GPIO
RPi.GPIO.setmode(RPi.GPIO.BCM)
RPi.GPIO.setup(2, RPi.GPIO.IN, pull_up_down=RPi.GPIO.PUD_UP)

def Ring():
pass
while True:
if RPi.GPIO.input(2) == RPi.GPIO.LOW:
Ring()
```

This code detects a button press in the same way as described in Chapter 7 – Controlling Input and Output Pins on page 150. It then calls the function *Ring()*, which does nothing yet. To play a sound file from Python, you can use *mpg321*. This is a command line tool for playing MP3 files on Raspbian, but you can start it from a Python script. To install mpg321:

• In **LXTerminal**, type the following command and then press **Enter**:

```
sudo apt-get install mpg321
```

Copy the sound effect *.mp3* into your *Doorbell* folder. Add the following import directive to the top of the script:

```
import os
```

Then in the *Ring()* function, remove the *pass* statement and add the following code; replace the file name of the MP3 file if necessary.

```
os.system(“mpg321 ring.mp3”)
```

The *system()* function in the *os* module runs command line tools from Python scripts. To run the script:
• In **LXTerminal**, type the following command and then press **Enter**:

```
sudo python3 Doorbell/doorbell.py
```

Press the switch to hear the audio file play. If you hold the button down, the script repeatedly plays the sound.

To stop the script from looping the sound, you can make sure that the button is released before the script checks for another press:

• Add the following statements to your script file, in the *Ring()* function, after the call to *os.system()*:

```
while RPi.GPIO.input(2) == RPi.GPIO.LOW:
pass
```

The *while* loop continually runs the *pass* instruction (which does nothing) until the button is released and *GPIO2* goes high.

Starting the Doorbell Script Automatically

If you want your doorbell script to run when the Raspberry Pi starts up, you can use a feature built-in to Linux called *cron*. Cron is a way of starting programs in the background.

First, create a shell script that starts your Python program: 1. In **LXTerminal**, type the following command and then press **Enter**:

```
nano Doorbell/doorbell.sh
```

2. Add the following text to the file:

```
#!/bin/sh
cd /home/pi/Doorbell
sudo amixer cset numid=3 1
sudo python3 doorbell.py
```

3. Press **Ctrl + O**, and then press **Enter**.
4. Press **Ctrl + X**.
5. Type the following command and then press **Enter**: `sudo crontab -e`

6. At the end of the file, add the following text on a new line:

```
@reboot sh /home/pi/Doorbell/doorbell.sh
```

7. Press **Ctrl + O**, and then press **Enter**.
8. Press **Ctrl + X**.
9. Type the following command and then press **Enter**: `sudo shutdown –r now`

Step 6 tells cron to run the command *“sh /home/pi/Doorbell/ doorbell.sh”* when Raspbian starts. To stop the script from starting when the Pi starts up, repeat step 5 and remove the line.

9.4 An Enhanced Doorbell

In this section, you will see a short example of how you can extend the Pi doorbell script so that Python takes a picture from the Pi camera module when a visitor presses the doorbell. The Python *picamera* module contains the functions for accessing the camera. *Time* contains functions for working with the date and time. Add the following import statements to the top of your *doorbell.py* script:

```
import picamera import time
```

Then create an instance of the *PiCamera* class:

• Underneath the two *RPi.GPIO* functions, add the following statement:

```
camera = picamera.PiCamera()
```

To take a picture from the camera module, use the method *capture()*. This method accepts one argument – the file name where the image is to be saved. In the example below, the current date and time is combined with the string “Ring .jpg” to create a unique file name each time the doorbell is pressed.

Add a call to *capture()* above the *os.system()* call. For example:

```
camera.capture(“Ring %s.jpg” % time.strftime(“%Y%m%d-%H%M%S”))
```

The string passed into *strftime()* determines the format of the date and time string. For more information about the *time* module, see *http://docs.python.org/3/library/time.html*

The full Python script should look like this:

```
import RPi.GPIO
import os
import picamera
import time
RPi.GPIO.setmode(RPi.GPIO.BCM)
RPi.GPIO.setup(2, RPi.GPIO.IN, pull_up_down=RPi.GPIO.PUD_UP) camera = picamera.PiCamera()

def Ring():
camera.capture(“Ring %s.jpg” % time.strftime(“%Y%m%d-%H%M%S”)) os.system(“mpg321 ring.mp3”)
while RPi.GPIO.input(2) == RPi.GPIO.LOW:

pass
while True:
if RPi.GPIO.input(2) == RPi.GPIO.LOW: Ring()
```

10 – Making Free Phone Calls with Google Voice

Google Voice is a form of voice over Internet protocols (VOIP) that is now available in Google Hangouts. You can use it to make free phone calls to numbers in the US and Canada[1], and it gives you a landline number that others can use to call you – either from their Google account, or from a regular phone.

You can use Google Voice from the Google Mail website, or the Google Hangouts apps for smartphones. However, if you set up your Raspberry Pi as a private branch exchange (PBX) then you can allow any SIP-compatible phone or softphone to make and receive calls using your Google Voice account.

In this chapter, you will learn how to install the RasPBX distribution and configure it to let SIP phones use your Google Voice account.

You need: 1. A Raspberry Pi, and either an Ethernet connection or a USB Wi-Fi dongle.
2. A spare microSD card.
3. A SIP phone, SIP app for your smartphone, or a SIP softphone for your desktop PC.

10.1 SIP and Softphones

Session initiation protocol (SIP) is a VOIP standard that defines how phones and softphones (software that runs on your computer and pretends to be a phone) communicate with the servers that route phone calls over the Internet. Unlike regular phones, SIP phones connect to a local area network using Ethernet instead of connecting to a phone line. A PBX server often connects to a phone line and to computer networks, and it routes landline calls to the SIP devices.

1. You can make international calls, but these are not free and require you to add credit to your account.

If you do not have a SIP phone, there are many softphones available for Windows, Linux, and Mac OS X desktop PCs, and there are also SIP apps that run on smartphones. It is beyond the scope of this guide to review all of the softphones that are available. However, as a starting point, you may want to download Linphone (*http://www.linphone.org*) or Zoiper (*http:// www.zoiper.com*).

There are also hosted SIP service providers that give you a landline number which is associated with their SIP servers and PBXs. You can connect your SIP phone or softphone to these servers and receive free incoming calls. However, outgoing calls usually incur charges. Using Google Voice means that you do not have to pay for outgoing calls to numbers in the US or Canada.

The registration process for Google Voice requires that you give them a phone number where you can be reached. If you do not have one then you can get a free number and SIP service that you can use for the registration.

To do this:

1. Open a web browser and visit *http://sip.pregi.net*
2. Click **Create Account**.

3. Complete the registration form (you do not need to enter a value for the phone number field), and then click **Register**.

4. Check your email and confirm the registration.
5. In your web browser, visit *http://www.ipkall.com*
6. Click ***Sign-Up***.

7. In the **SIP URI username** box, enter your sip.pregi.net username.
8. In the **SIP URI @ hostname** box, enter *sip.pregi.net*. 9. Complete the rest of the form and then click **Submit**.

When you register with IPKall, they give you a Washington state phone number and associate this number with your SIP account at sip.pregi.net. To test the number:

1. Install a SIP softphone on your desktop PC.

2. In the softphone, create an account using the following details:
Domain: sip.pregi.net
Username: <your sip.pregi.net username>
Password: <your sip.pregi.net password>
Transport: UDP

Port: 5060

3. From a landline or cellphone (if you are using a SIP app, use a different phone for this step), call the number that IPKall gives you.

If your SIP phone does not ring, check all of your settings and try again.

10.2 Google Voice

To register for Google Voice, you must: be located in the US or Canada[1]; have an existing phone number (only for the registration process); and have a Google account.

During the process, you create a new Google Voice landline number. This can be in any area code (you are not restricted to the area code that you live in) where they have numbers available.

If you do not have a Google account, create one at *http:// accounts.google.com/SignUp* and then create a Google Voice account at *http://www.google.com/voice*

When you have finished the Google Voice registration process and have your new phone number:

1. Once you register and obtain your Google Voice number, you can use the service from international locations to call numbers in the US or Canada.

1. On the Google Voice webpage, click the **Settings** button.
2. On the **Phones** tab, under **Forward calls to**, deselect the checkbox next to your phone number. 3. On the **Calls** tab, next to **Call Screening**, click **Off**.

10.3 RasPBX

RasPBX is a Linux distribution for the Raspberry Pi that packages together all of the telephony, server, and database software that most users need. Some of these tools are quite complicated to install, so using a prebuilt RasPBX disk image saves you a lot of time and effort.

FreePBX is one of the key pieces of software that this distribution contains. It is the server software that you use to setup extensions (equivalent to user accounts), passwords, and configure access to your Google Voice account.

To install RasPBX: 1. Download the latest RasPBX disk image from *http://www.raspberry-asterisk.org/downloads/*
2. Extract the *.img* file from the zip archive.

3. Write the disk image to a new microSD card. For more information, see section 2.3 Installation of Raspbian using a Disk Image on page 20.

4. Safely eject the microSD from your PC.
5. Unplug the Pi, and insert the microSD card.
6. Reconnect the Pi.

If you are using an Ethernet cable to connect the Pi to your network, and you do not have a display, make an SSH connection. For more information, see Connecting to the Pi with SSH on page 43.

Login in to the Pi with the username *root*, and the password *raspberry*.

If you are using an Ethernet cable, configure the Pi with a static IP address. For more information, see Using a Static IP Address on page 41.

To use a USB Wi-Fi adapter, you must first install the *wpasupplicant* tool and then configure it. To do this: 1. In the terminal, type the following command and then press **Enter**:

```
apt-get install wireless-tools wpasupplicant
```

2. Type the following command and then press **Enter**:

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

3. Add the following text to the end of the file[1]:

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
wpa-ssid <YOUR NETWORK SSID>
wpa-psk "<YOUR NETWORK PASSWORD>"
address <AN IP ADDRESS TO USE>
netmask <YOUR NETWORK NETMASK>
gateway <YOUR NETWORK GATEWAY>
```

4. Press **Ctrl + O**, and then press **Enter**.
5. Press **Ctrl + X.**
6. Type the following command and then press **Enter**:

```
shutdown –r now
```

RasPBX is primarily a server distribution. However, you can still access the graphical desktop environment using the *startx* command. If you want to install a softphone (such as *sflphone*) on the same Pi that runs RasPBX then you can also do that.

Accessing the Admin Panel
FreePBX includes a web-based administration panel that you can use to complete this tutorial. To access it:
1. On a device connected to your local network, open

1. If you are not sure about the values to enter into this configuration, look at the network connection properties of other devices on your network.

a web browser.
2. Visit *http://raspbx* or *http://* followed by the static IP address of your Pi.

3. Click **FreePBX Administration**.
4. In the first box, type *admin*.
5. In the second box, type *admin*.
6. Click **Continue**.

Creating the SIP Extensions
Before you start routing the calls, you need to create extensions (the PBX equivalent of a user account) for each SIP device that you want to use.

To create a SIP extension:

1. In the **FreePBX Administration** panel, on the **Menu**, point to **Applications** and then click
Extensions.

2. In the **Device** list, click **Generic SIP Device**.
3. Click **Submit**.

4. In the **User Extension** box, type a number. For example, *2001*.
5. In the **Display Name** box, type the name of the person who uses this extension.
6. In the **secret** box, type a password for this
extension.
7. Click **Submit**.
8. At the top of the page, click **Apply Config**.
To register a SIP device on your local network to this extension: 1. In the settings of your SIP phone or softphone, create a new account.

2. In the username box, enter the extension number. For example, *2001*.
3. In the password box, enter the secret that you set in the FreePBX configuration.

4. In the domain box, enter the IP address of your Pi. 5. Check that the phone connects to the server using *UDP* port *5060*.

It can be useful to create a second extension, for example *2002*, so that you can test the server. If the SIP device on extension 2001 dials *2002* then the second device should ring. If it does not, check that both SIP devices are showing “registered”. This indicates that they are communicating with the Raspberry Pi ok. To see what is happening on the Pi:

1. Make an SSH connection or login to the terminal on the Pi.
2. Type the following command and then press **Enter**: `asterisk -rvvvv`

When the Asterisk CLI tool is running, you see all of the actions that occur – including when SIP devices connect to the Pi, and when they disconnect. This can be helpful in debugging. Some SIP softphones drop off the network without warning, so if you cannot dial a device try using different softphone software.

When you are finished with the CLI tool:
• Type the following command and then press **Enter**: `exit`

Setting up Google Voice and Outgoing Calls
RasPBX contains a FreePBX plugin for communicating with Google Voice. This plugin is called “Motif” and it works over extensible message and presence protocol (XMPP).

To add a Google Voice account to the system:

1. In the **FreePBX Administration** panel, on the **Menu**, point to **Connectivity** and then click **Google Voice (Motif)**.

2. In the **Google Voice Username** box, type your Google Mail email address.
3. In the **Google Voice Password** box, type your Google Mail password.

4. In the **Google Voice Phone Number** box, type the phone number that you obtained from Google Voice.

5. Click the box next to **Add Trunk**.
6. Click the box next to **Add Outbound Routes**.

7. Click the box next to **Send Unanswered to Google Voicemail**.

8. Click **Submit**.
9. At the top of the page, click **Apply Config**.

10. On the right-hand side of the screen, under **Add Google Voice Account**, click your Google Voice account.

11. Check that the **Status** says *Connected*. If it does not, check your Google Voice account details.

Motif creates the outbound route for you, and all SIP extensions can now use this outbound route to make calls. The default can now use this outbound route to make calls. The default digit phone number or a *1* followed by the 10-digit number.

Routing the Incoming Calls
To receive the incoming calls from your Google Voice number, you need to direct them somewhere. FreePBX has a lot of features that work with incoming calls, but this guide only covers directing calls to a specific extension. To route calls to an extension:

1. In the **FreePBX Administration** panel, on the **Menu**, point to **Connectivity** and then click
Inbound Routes.

2. In the **Description** box, type a name for this rule. For example, *Google Voice*.
3. If you want to create a rule that only applies to incoming calls from one specific Google Voice account: in the **DID Number** box, type the phone number that Google Voice gave you.

4. In the **== choose one ==** list, click **Extensions** and then click the extension that you want to connect the incoming calls to.

5. Click **Submit**.
6. At the top of the page, click **Apply Config**.

If you add multiple Google Voice accounts to your system and want to direct them to different extensions, create different rules for each Google Voice account and specify the phone number in the *DID Number* box for each rule.

10.4 SIP Phones across the Internet

If you want SIP devices or softphones to be able to access your PBX from outside your local network, you need to configure your network router to allow incoming traffic on the following ports and route them through to the same port numbers on the Pi:

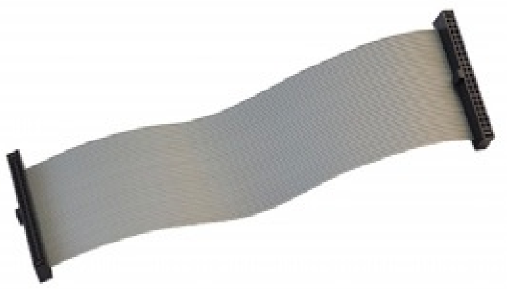
Start Port	End Port	Type	Description
5060	5061	UDP	The two ports used for SIP signaling and control.
10001	20000	UDP	Ports for media (voice and video) transmission.

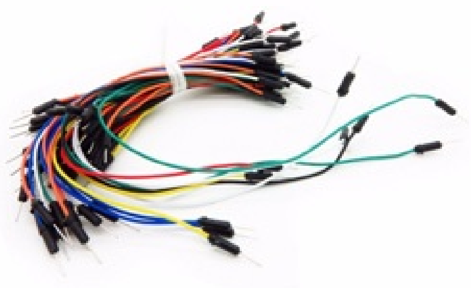
Your SIP devices need to connect to the external IP address of your router, not to the internal IP address of the Pi as you have used so far.

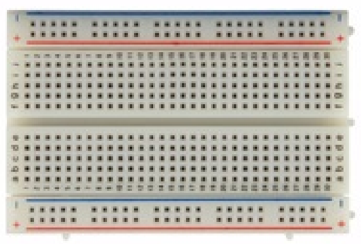
When your router restarts or reconnects to the Internet, your IP address may change. Using a dynamic domain name system (DDNS) service provider, you can map your IP address to an Internet domain name and use this domain name to access your PBX.

11 – Accessories

One of the great things about the Raspberry Pi is the number of peripherals, expansion boards, connectors, and components that you can use with it. Here is a selection of a useful accessories that you can use with the projects and information in this guide.







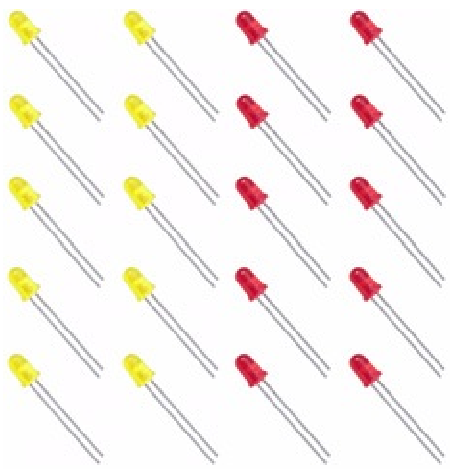
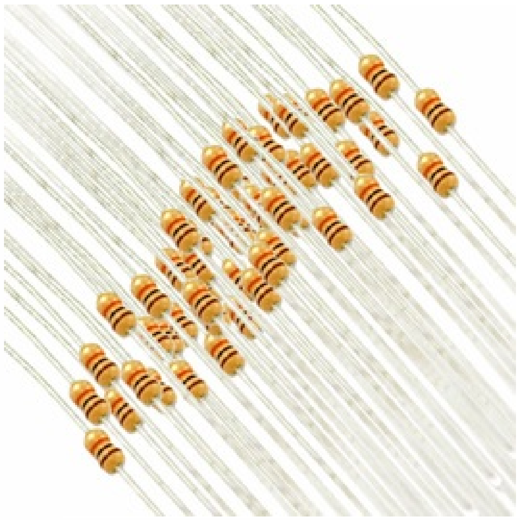
USB to TTL Cable
This handy cable connects to a PC using the USB connector, and to the Pi using individual female sockets that fit directly onto the serial port pins of the Pi's GPIO header. The built-in circuitry handles voltage level conversions.
http://www.vilros.com/usb-to-ttl-cable.html

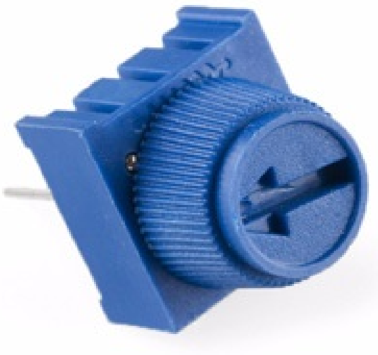
40-pin GPIO Ribbon Cable
A female to female cable for connecting external circuits to the Raspberry Pi. It can also be used to connect male to male jumper cables to the Pi when prototyping your circuits on a breadboard. *http://www.vilros.com/40-pin-gpio-ribboncable.html*

Jumper Wires
Jumper wires (also called "jump wires") have reinforced ends that make it easy to connect components on a breadboard, or make connections with devices that have female sockets.
http://www.vilros.com/jumper-wires.html

Breadboard
A standard, 400-hole breadboard is invaluable for prototyping and testing your circuits.

Their use is explained in Chapter 7 – Controlling Input and Output Pins on page 150. *http://www.vilros.com/breadboard.html*



Light-Emitting Diodes (LEDs)
LEDs are components that you will use lots of. In addition to making them flash or using them as status indicators, they can be very helpful for checking which parts of a circuit are working. *http://www.vilros.com/red-led.html*

Resistors
Almost all electronic circuits contain at least one resistor. Unlike other components that you may buy specifically for the project that you are working on, you will use so many resistors that it is worth keeping a stock of different values.
10 KΩ: *http://www.vilros.com/10k-resistors.html* 330 Ω: *http://www.vilros.com/330-resistors.html*

Buttons
These high-quality, push-button switches are ideal for making input circuits using the Pi's GPIO header. When the conveniently-large button is pressed and held, electricity flows between two of the switch's terminals. Exactly the type used in section 7.5 Basic Input on page 160.
http://www.vilros.com/big-12mm-buttons.html

10K Trimpot
Also known as "potentiometer" or "variable resistor", these trimpots are used when you need to change the amount of resistance in a circuit without disassembling it. For example,

to reduce the brightness of an LED. These ”pots” have resistance that changes as you turn the knob, from 0 Ω through 10 KΩ, and slot neatly into place on a breadboard.
http://www.vilros.com/10k-trimpot.html