NEW

Raspberry Pi for Beginners

All you need to know to get started with your Raspberry Pi

For all RasPi models

- 100% independent
- Create fun Pi projects
- Learn Pi programming
- Set up your Raspberry Pi

Welcome to
Raspberry Pi for Beginners

The Raspberry Pi has taken the world by storm. The credit-card size computer has sold over a million units and has captured the imagination of kids and adults all over the globe. For the former, it has proved a great way to get into the world of programming and for the latter, it has allowed their creativity to soar. With a few peripherals and basic programming knowledge, your Raspberry Pi can do things you never imagined. Raspberry Pi for Beginners teaches you just that. We begin with the peripherals you'll need, how to set it up and install distros familiar to you. Raspbian tutorials then introduce you to the creative potential of the device. A weather station, music streamer and Twitter-powered lamp are a few of the many project tutorials we've shown you how to create and there's also a section on how to get started with Scratch and Python programming. Every new venture comes with its own set of challenges, so we've concluded with a section on Troubleshooting and FAQs.

Raspberry Pi for Beginners

Imagine Publishing Ltd
Richmond House
33 Richmond Hill
Bournemouth
Dorset BH2 6EZ
☎ +44 (0) 1202 586200
Website: www.imagine-publishing.co.uk
Twitter: @Books_Imagine
Facebook: www.facebook.com/ImagineBookazines

Publishing Director
Aaron Asadi

Head of Design
Ross Andrews

Production Editor
Hannah Kelly

Senior Art Editor
Greg Whitaker

Designer
Abbi Denney

Photographer
James Sheppard

Printed by
William Gibbons, 26 Planetary Road, Willenhall, West Midlands, WV13 3XT

Distributed in the UK, Eire & the Rest of the World by
Marketforce, Blue Fin Building, 110 Southwark Street, London, SE1 0SU
Tel 0203 148 3300 www.marketforce.co.uk

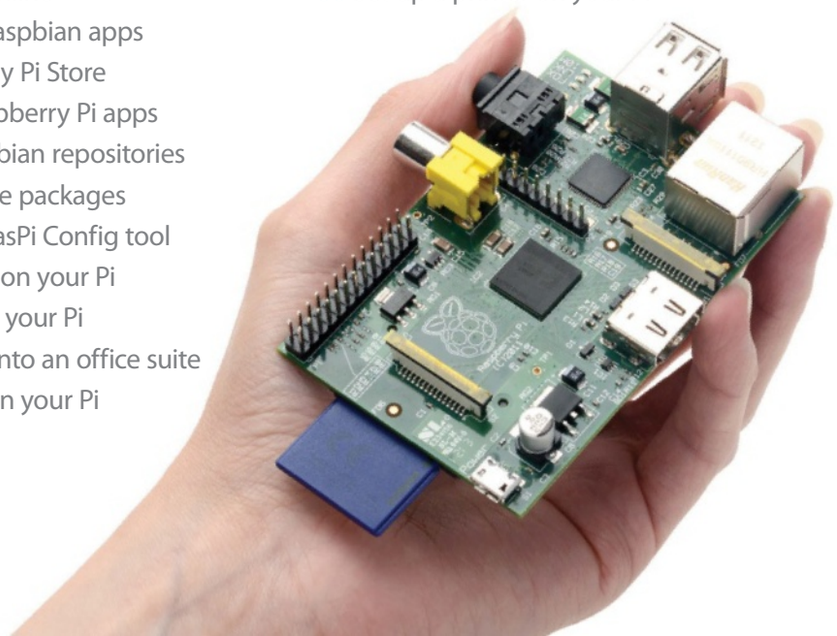
Distributed in Australia by
Network Services (a division of Bauer Media Group), Level 21 Civic Tower, 66-68 Goulburn Street, Sydney, New South Wales 2000, Australia Tel +61 2 8667 5288

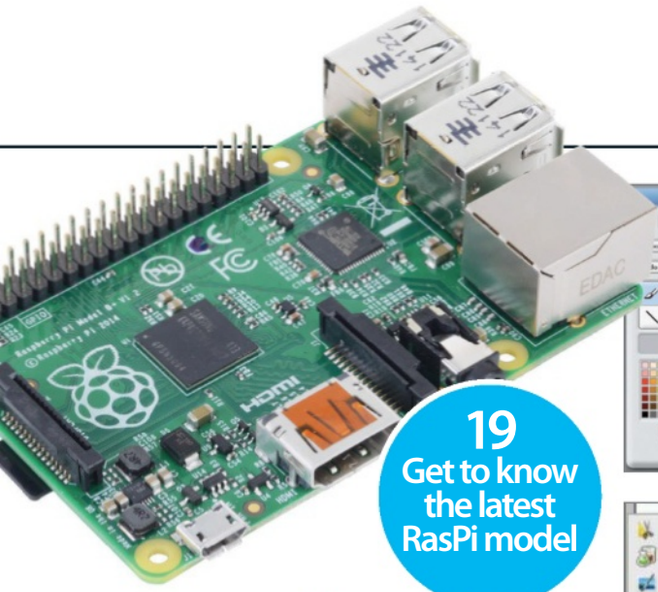
Part of the LinuxUser & Developer bookazine series

IMAGINE PUBLISHING

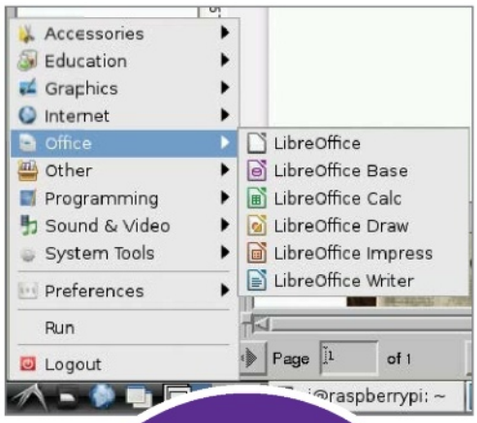
Contents

The basics





19
Get to know the latest RasPi model



Projects

172
Glossary
Decoding the Pi jargon

Programming

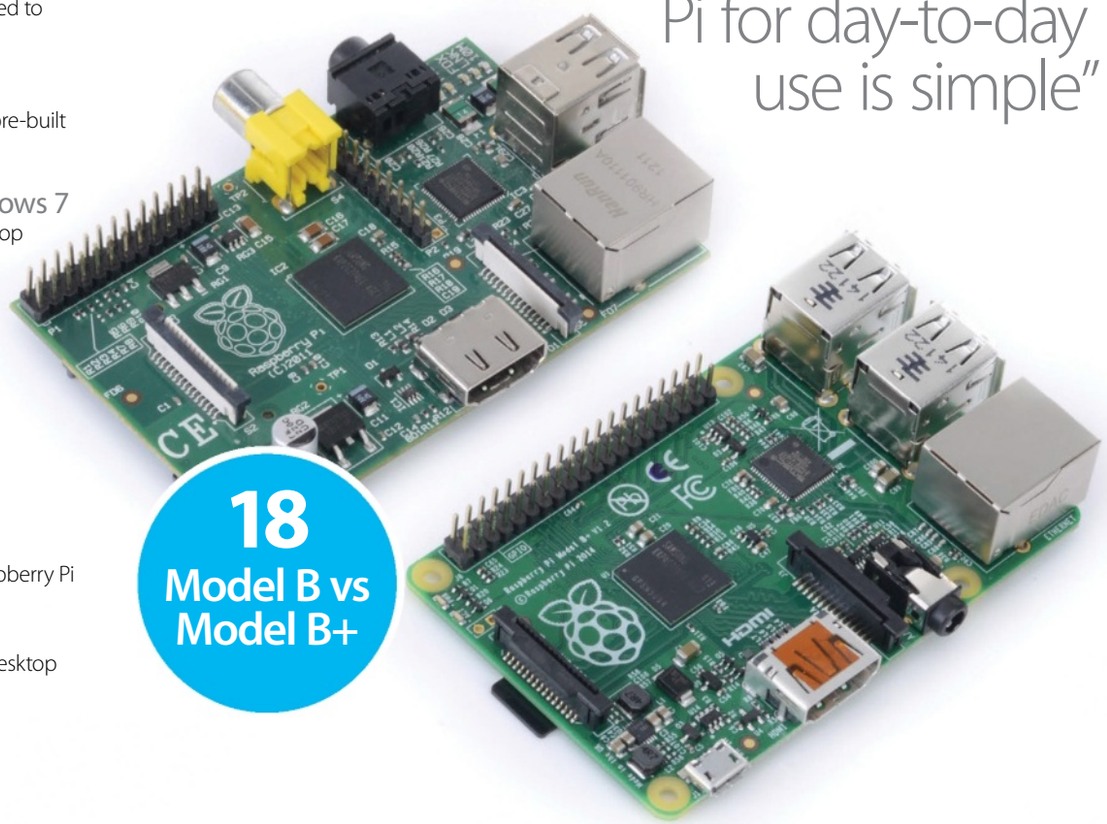
Troubleshooting

"You're more likely to be overwhelmed by choice than stuck for options"

The basics

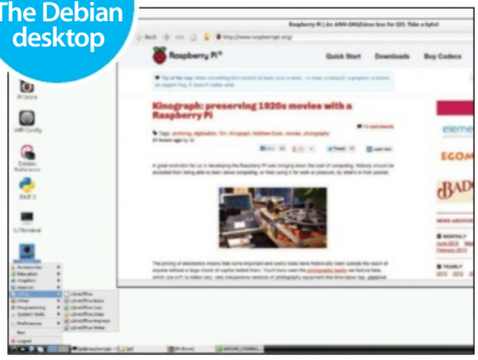
Set up your Raspberry Pi, install the latest distros and get started with Raspbian

"Setting up your Pi for day-to-day use is simple"



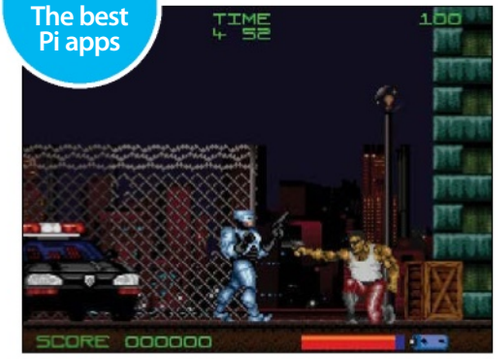
18
Model B vs Model B+

32 The Debian desktop

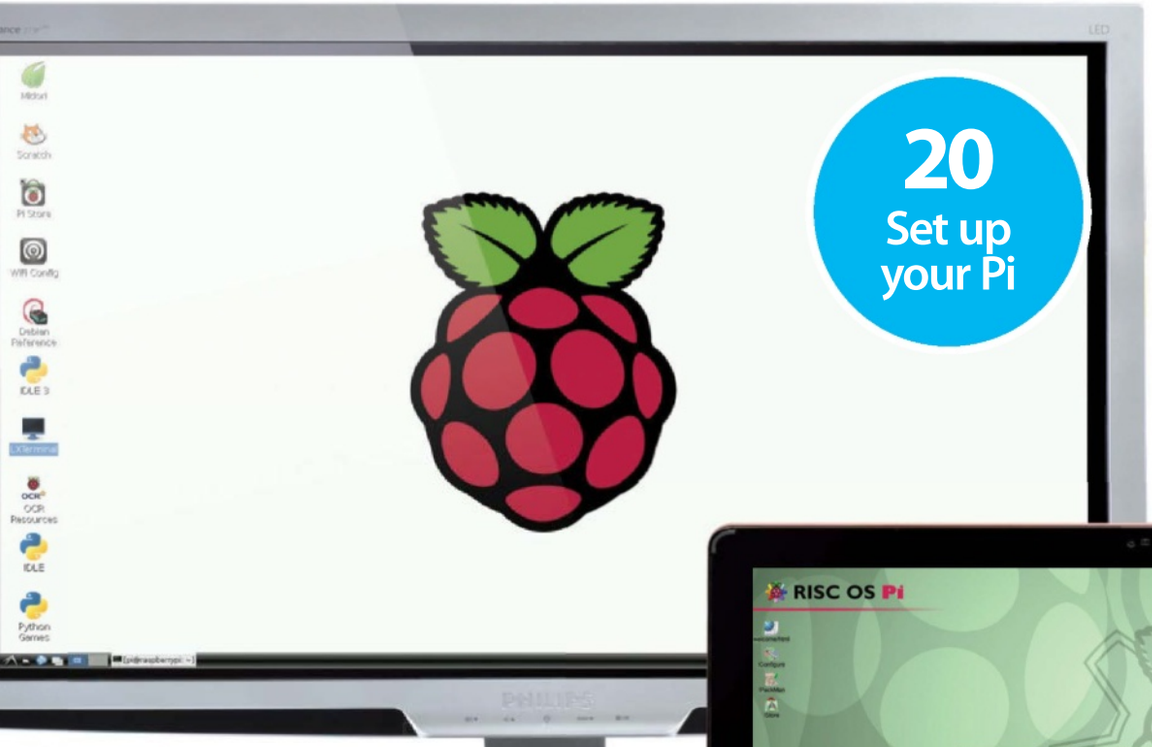


38 The Pi Store

40 The best Pi apps



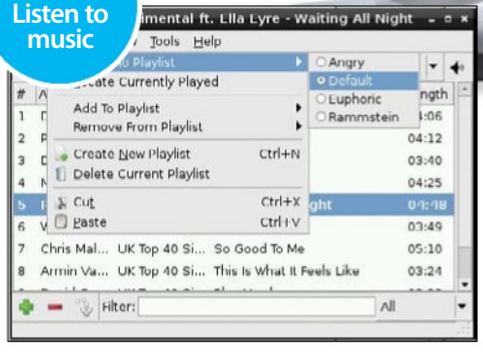
20 Set up your Pi



"The potential for your Raspberry Pi's capability is limited only by your imagination"

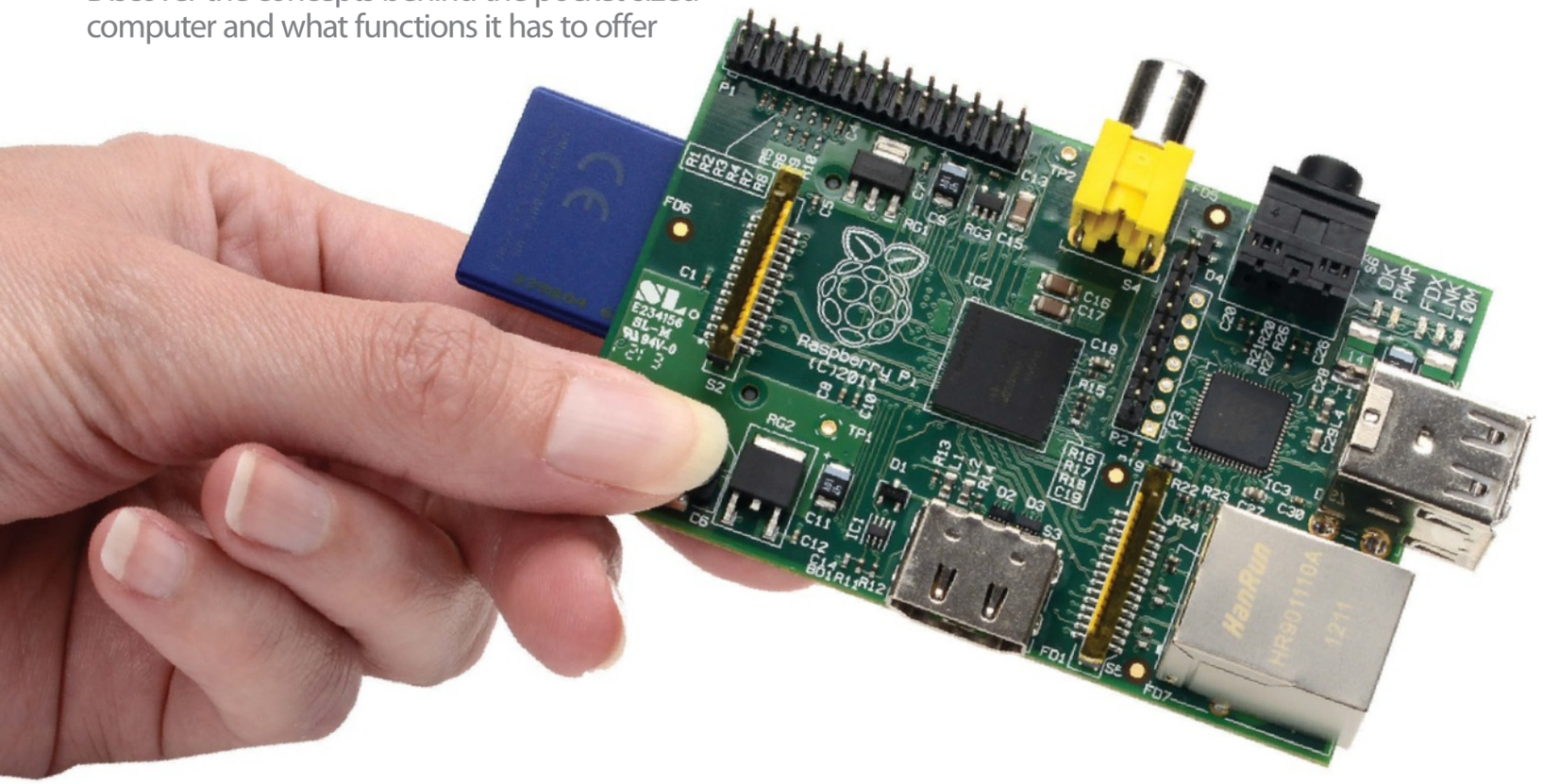
31 Install RISC OS

50 Listen to music



What is Raspberry Pi?

Discover the concepts behind the pocket-sized computer and what functions it has to offer



The Raspberry Pi sprang out of a desire by colleagues at the University of Cambridge's Computer Laboratory to see a return to the days of kids programming inexpensive computers. The rise of expensive PCs and games consoles put paid to the BBC B, Spectrum and C64 generation of home programmers, leading to applicants for computer studies courses lacking the necessary skills. After spending a few years designing prototypes, Eben Upton, formerly of the University but now working as a chip designer for Broadcom, joined forces with his old university colleagues, Pete Lomas of hardware design company Norcott Technologies and David Braben, co-author of classic BBC Micro game *Elite*, to form the Raspberry Foundation. Three years later, the Raspberry Pi went into mass production with the Model B (main image), and then later, lower-capacity RAM version, but cheaper, Model A.

The basic concepts behind the Raspberry Pi were to make it as affordable as possible, supply only the basics and provide it with a programming environment and hardware connections for electronics projects. The Pi runs a modified version of Linux called Raspian with Wheezy Raspian being the preferred option for newcomers to the device. Raspian runs directly on an SD card and provides a command-line interface for using the operating system. However, as this was likely to be a little overwhelming for novices, there's a more friendly face to the Pi, and that appears when you type "StartX" to launch the desktop (Fig 4).

Connecting it up

As mentioned, the Pi comes as a bare bones circuit board but you can buy all the extras needed to make it fully functional. The first thing that is needed is an SD card to act as storage for the operating system and any software you want to install. Although it will work on a 2GB card, it is recommended you use a 4GB card as a minimum. This should be Class 4 speed and it is better to have a branded card so that it is more reliable. You'll need to format the card, download Raspian from the Raspberry Pi website, and install it, using either a Windows (easier) or Mac (not as easy) computer. With the operating system installed on the card, all that remains is the various plugs and connections. You'll need a USB keyboard and mouse, either a HDMI or analogue video cable and a micro USB power supply that provides 700mA at 5v. The HDMI connection also includes audio so if you are using it with a TV or a monitor with speakers then sound will come from those. If not, there's a separate 3.5mm audio jack for output. The final option is that on the Model B there is an Ethernet connection for wiring into your internet modem. This is a faster and more hassle-free way of getting the Pi online. The final element is to get a fancy looking case to put the Pi in (Fig 1).

Programming the Pi

There are two programming languages supplied by default with the Pi: Scratch v1.4 and Python. They both have shortcut icons on the graphical desktop. Scratch was designed by Lifelong Kindergarten Group at the MIT Media Lab as a first-step in programming. It uses tile-based language commands that can be strung together without having to worry about syntax. Commands are split up into easy-to-understand groups and dragged onto the scripting area. Sounds, graphics and animation can be added. Projects can be saved or uploaded to the Scratch website and shared for remixing by other users.

■ Fig 1: The Pi is ready to use with connections to peripherals, a monitor and the OS installed on an SD card

■ Fig 2: There are two programming languages supplied with the Pi. Scratch is for beginners to get started, while Python is a little more advanced

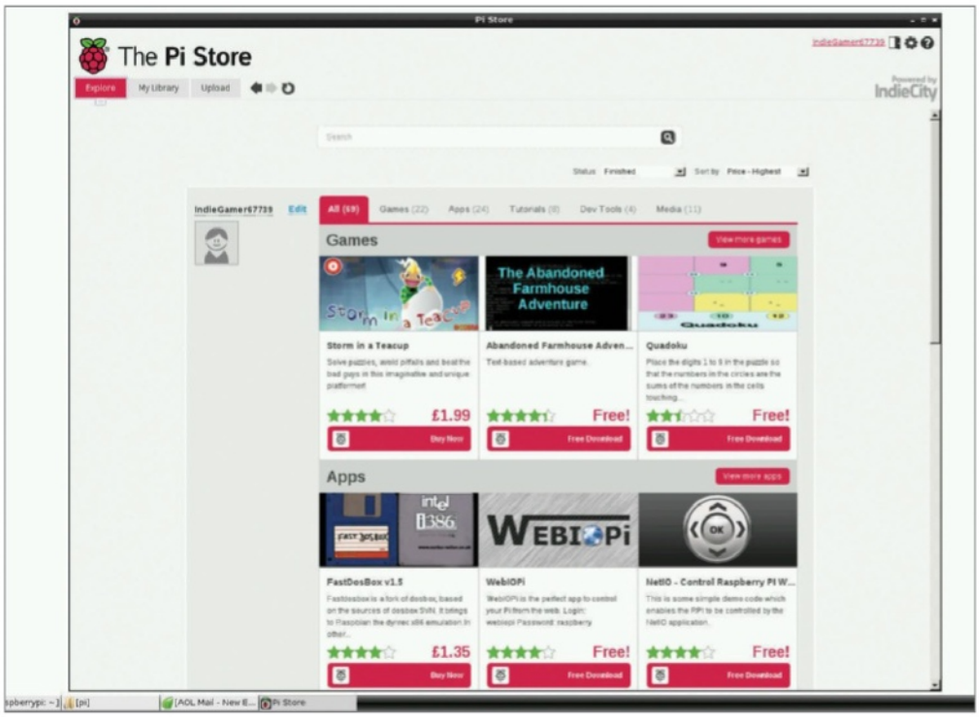
The other language is Python v3.2.3 (Fig 2) which starts with a Python Shell. Python is an interpreted language, where commands are read and implemented one line at a time. The high level commands and data structures make it ideal for scripting. The previous version of Python is still quite popular and has more development aids, so is also supplied. The icons for these are IDLE 3 and IDLE, which stands for Integrated Development Language for Python.

Using the Pi Store

Of course, using your Pi would just be a labour of love if there weren't other people already out creating things you can run on it. To get more out of your Pi and see what other people are up to, there's the Pi Store (Fig 3). This houses a collection of games, apps, tutorials, development tools and media, such as the latest issues of The MagPi – a community-led magazine dedicated to the Pi of course.

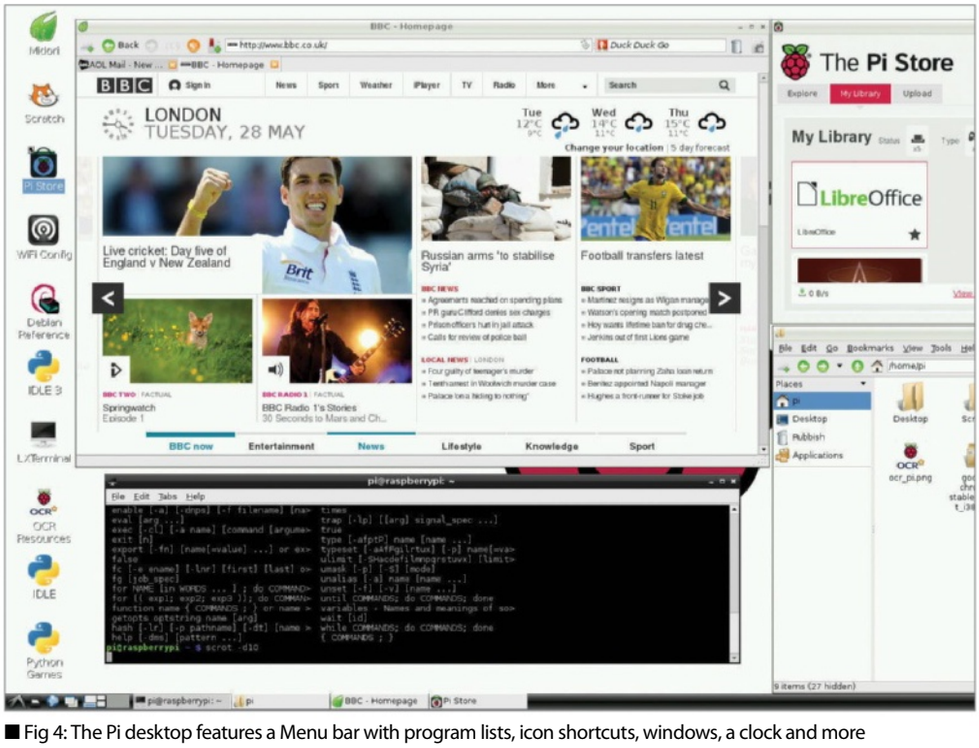
You'll first need to create an account to log in. The content of the Pi Store can then be downloaded and installed for use. Some create shortcut icons, others use the Menu, but many can only be run directly from the Store interface itself. This has a My Library tab where downloaded content can be managed as well as run.

Most of the content on the Pi Store is free, though a few things carry a charge. You can also upload content you have created to the store, though you need to register as a developer to do this.



■ Fig 3: There's a wealth of apps, development aids and media available on the Pi Store

"A more friendly interface to the Pi appears when you type 'StartX' to launch the familiar-looking desktop"



■ Fig 4: The Pi desktop features a Menu bar with program lists, icon shortcuts, windows, a clock and more

What is Linux?

The Linux OS is ubiquitous. It's on your computer, laptop, smartphone and now, your Pi. But what is it?

Linux is the operating system (OS) used for your Raspberry Pi. Its role is exactly the same as Windows, Mac OS X, Android (in fact, Android is based on a Linux kernel), iOS or any other OS you care to mention. That role is to provide a platform for everything else to run on. It talks to the hardware and it talks to you, the user.

But what makes Linux different to any other OS out there? Well, for a start it's free (more about that later), immensely powerful, highly customisable and the best bit is it's been created for users by users.

However to call Linux 'an operating system' is a bit of an understatement. It's not 'one operating system' in the same way that Windows 8 or Mac OS X is. No, it's many operating systems… hundreds even! As we'll talk about in the next section, Linux consists of different components, each of which has many different variants. These have all been wrapped into easy-to-install distributions to meet different needs. Want a simple desktop replacement? There's a Linux distribution for that. Want a home media server? There's a distribution for that too. If you can think of it, someone in the Linux community is probably already developing for it.

How it works

One of the great things about Linux (apart from it being free) is just how customisable it is. Let's take a look at the main components (Fig 1) that make up a Linux install (there are more, but these are the ones you'll meet most often along your way):

The kernel: The brains of the operation. It talks to the hardware and can be compiled to run on different CPUs, such as the ARM one in the Pi. Any application you run that needs access to hardware – such as keyboard input, monitor output or access to the hard drive – will have to go through the kernel.

The shell: A good old-fashioned command-line interface (Fig 2). There's nothing you can't do here, from installing software to viewing system resources and scripting common tasks. It can look daunting at first, but you'll soon realise it's not so scary.

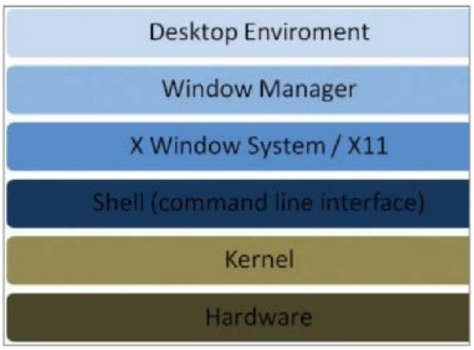
Desktop environment: Of course it's no fun just looking at text all day. This is where the desktop environment (Fig 3) comes in and what makes it look like an operation system you're probably more used to. However, unlike most OSs, you're not limited to one desktop: you can mix and match and get exactly the look and feel you want.

Applications: Not part of the OS as such, but a key part of any Linux installation. Whether it's an office suite or a media player, you'll find it for Linux.

Set yourself free

If there's one reason to use Linux, it's cost; there is none! Welcome to the world of freeware and open source software. For most users, that means you just download the OS or your chosen application, install it and use it. No money changes hands and no laws have been broken. You see, Linux and thousands of applications written for it are created for the love of creating software and sharing it with the world. It's people just like you who have an interest in computing and a problem they want to solve.

But there's more to open source than just free software. It's open software: you can look inside



■ Fig 1: An overview of the Linux architecture hierarchy. The kernel sits right above the physical hardware

■ Fig 2: Also known as the 'terminal', here you can see it being used to keep your system up to date

the source code, play with it, change it and make a difference to the whole community. Fed up waiting for a new feature in your favourite application? With a bit of knowledge you can add it yourself.

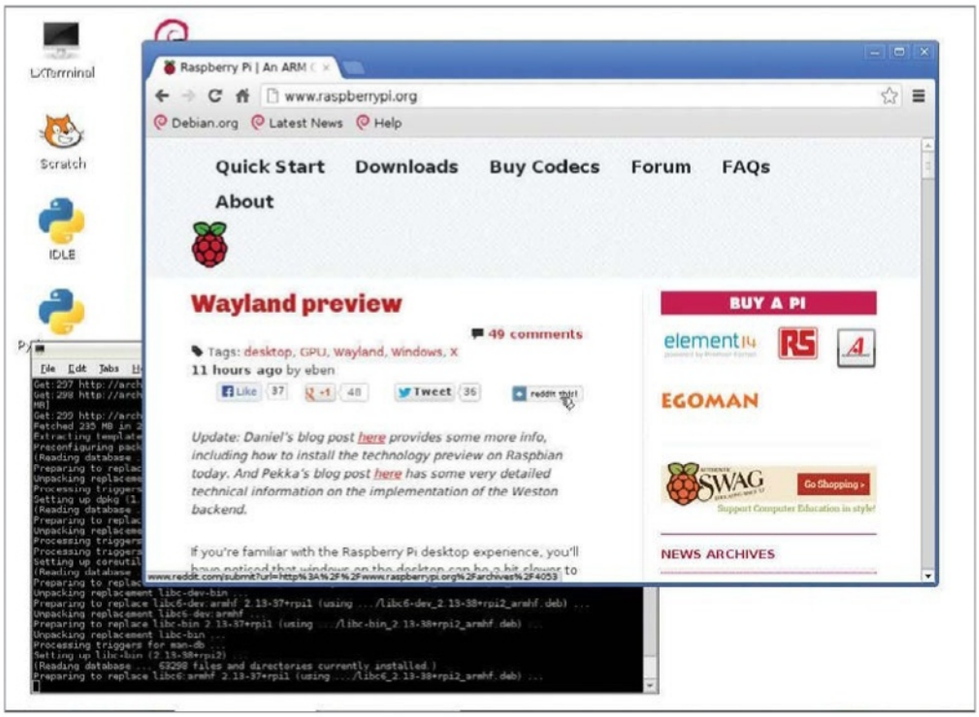
Many of the people starting out with the Raspberry Pi today will be the Linux programmers of tomorrow.

Choosing a distribution

With so much choice in Linux, where do you start? Luckily for us, that's where the distribution (or 'distro') comes in (Fig 4). These are a combination of kernel, desktop environment and applications chosen for a specific purpose. From general desktop use, use on older hardware, web servers, media servers to development environments, there's always a choice of distros to meet your needs.

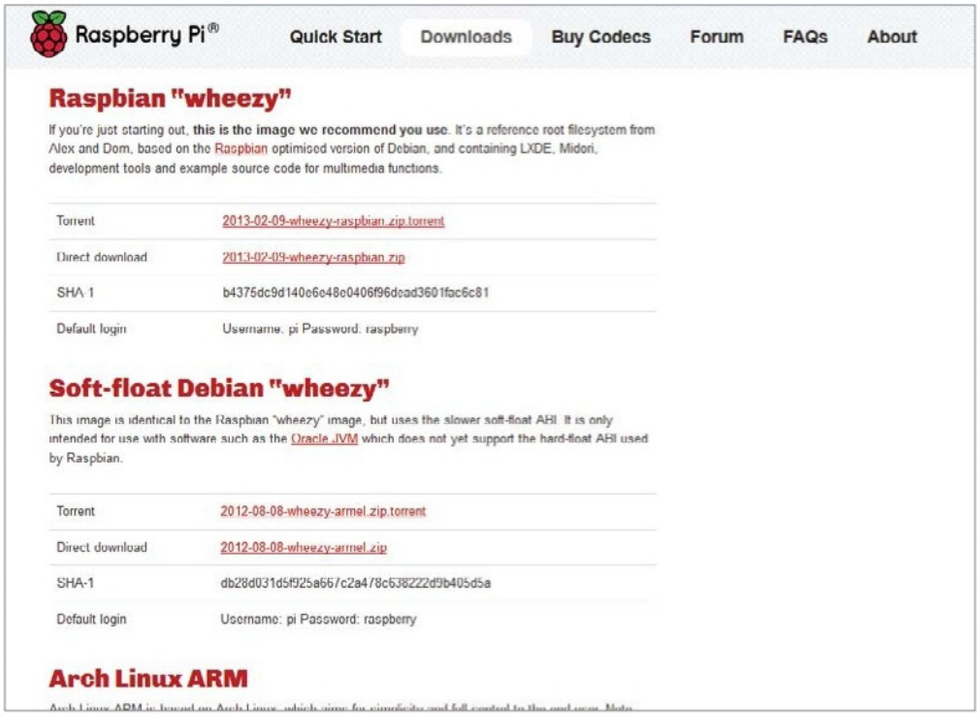
As a Raspberry Pi user, you'll have a more limited choice of distribution due to the ARM processor used, but the number is growing all the time. Most users will begin with Raspbian, a modified version of Debian that comes bundled with a wealth of applications and development tools. The choice of lightweight LXDE as a desktop environment ensures a smooth user experience.

If you're feeling brave, check out Arch – it's fast, easy on resources and the perfect partner if you plan to turn your Pi into an embedded masterpiece. But be warned: you'll be greeted with a flashing cursor and nothing more.

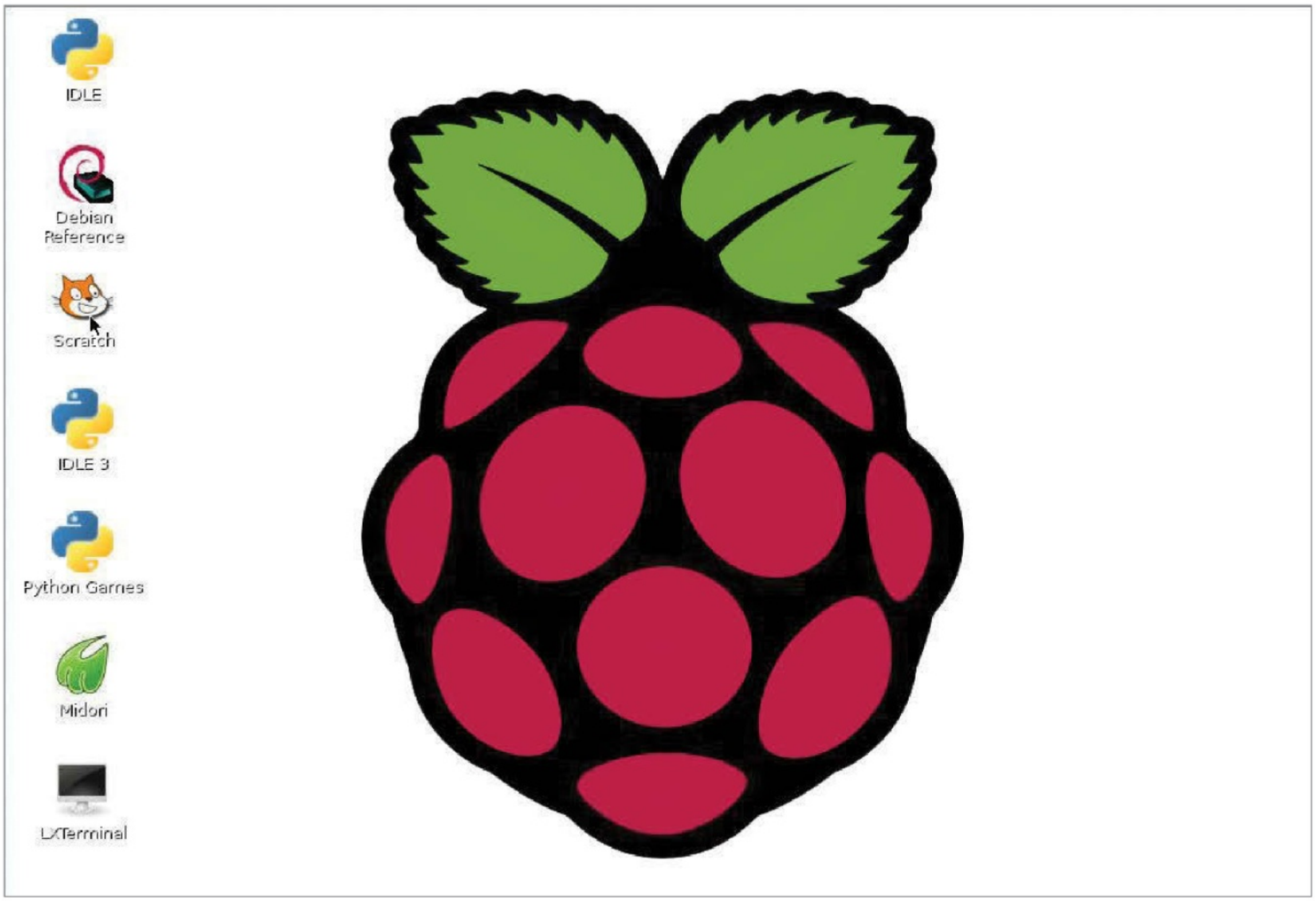


■ Fig 3: Linux has many desktop environments. Anyone who has used Windows or Mac OS X will feel right at home

"To call Linux 'an operating system' is a bit of an understatement. It's many operating systems... hundreds even!"



■ Fig 4: Go to **www.raspberrypi.org** and download a distro. Raspbian 'Wheezy' is recommended for novices



Choose the operating system

The Raspberry Pi has a number of Linux distros to choose from. Here are three of the best…

The Raspberry Pi has captured the hearts and imaginations of the public since its launch a little over a year ago. There's rarely a day gone by where we haven't heard of some wonderful, extreme or incredible hardware project from the ever-growing community of Raspberry Pi enthusiasts. Just looking through the pages of this book will reveal some examples of what can be done. The potential for the Raspberry Pi's achievements are limited only by the users' imagination.

However, it's not just the hardware that's worthy of consideration. The Raspberry Pi, being an ARM CPU-based 'computer', is capable of running a variety of different operating systems. These – known as distributions, or distro for short – are as varied as the many Raspberry Pi projects, and each can offer the user a different look and feel with their interaction with this fantastic little piece of technology.

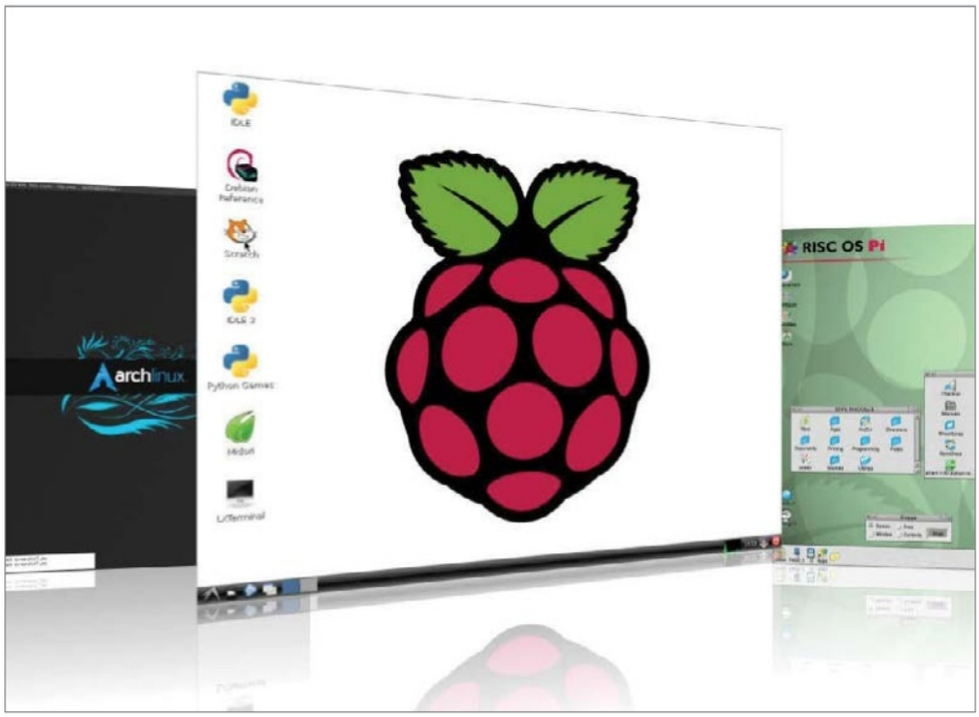
In this instance, we'll look at a selection of three of the most popular distros available for the Raspberry Pi: Raspbian, Arch Linux and RISC OS. Each of these distros is optimised especially for the hardware in the Raspberry Pi, utilising armhf, which is a set of instructions that support the Raspberry Pi hardware floating-point arithmetic architecture. What this essentially means is that the operations contained within the operating system itself will be executed many times faster and more efficiently than an operating system without the armhf instructions.

Raspbian

Raspbian is the first distro every new Raspberry Pi owner should ideally use. It's based on Debian Linux and, as we mentioned, is fully optimised for the Raspberry Pi's hardware. It's also an excellent starting point as it contains a plethora of pre-installed programs that will help get you up and running, and into the wonderful world of the Raspberry Pi.

Raspbian's approach to the Raspberry Pi is one of ease of use. It's a fully documented operating system with a very vibrant community and is continually improving with each successive release. The installation and initial setup of Raspbian has also been designed to be as intuitive as possible.

The Raspbian operating system itself is very well constructed and developed. Running LXDE (Lightweight X11 Desktop Environment) as the desktop environment, and Openbox as the window manager, makes this a considerably streamlined operating system ideally suited to the limited system resources of the Raspberry Pi. What's more, you won't require a PhD in computer science



■ Fig 1: Out of these three great distros for the Raspberry Pi, Raspbian is the best distro for beginners

in order to use Raspbian, and with a little work it can be made into a fully functional desktop operating system (Fig 2).

Raspbian leans heavily toward the educational aspects of the Raspberry Pi, as per the ethos of the Raspberry Pi Foundation. Packaged within you'll find such programs as Scratch – the child-friendly graphical beginner programming language whereby games and live story books can be created. Python is also included, a programming language that's ideal for beginners and resembles BASIC from years ago.

Arch Linux

Arch is a Linux distro that has been active for roughly 11 years. It's a fantastic operating system and prides itself on its minimalism, code correctness and elegance. This particular port for the Raspberry Pi – Arch Linux ARM – aims for simplicity and offers full control of the operating system to the user. It's fast (booting to a command prompt in less than ten seconds) and is incredibly light on the Raspberry Pi's system resources.

However, while it's a more user-control-orientated operating system, its very design means that it's not as friendly to the beginner as the aforementioned Raspbian. Whereas Raspbian will hold your hand, to some degree, in the setting up of the operating system and finally present you with a GUI (graphical user interface) with which to work from, Arch will boot to the command prompt and expect you to install everything else from then on (Fig 2).

Although there is no desktop to start with, you'll find that everything is in place for you to start building the ultimate personalised operating system with Arch.

■ Fig 2: Arch Linux for the Pi takes some setting up, but once that's done it's a wonderful system

> "The potential for the Raspberry Pi's achievements are limited only by the users' imagination"

RISC OS

Designed in Cambridge and first released in 1987, RISC OS is an operating system that can trace its roots back to the very team that developed the ARM processor – in fact it's a direct descendant of the OS used on BBC Micros. This version, designed for the Raspberry Pi, is RISC OS Open and is, in itself, an extremely charming system to use.

Part of that charm lies in the way RISC OS for the Raspberry Pi looks. If you can imagine the look and feel of a desktop from the early nineties, then you won't be too far away from this version of RISC OS. The artwork and scheme for the graphical user interface is very simplistic, but at the same time offers the user a retro and whimsical aspect.

Admittedly, the first use of RISC OS requires some patience since it isn't Linux, nor is it UNIX. Therefore things are conducted in a slightly different manner.

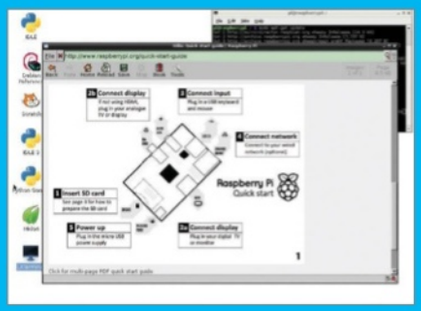
Which distro is best?

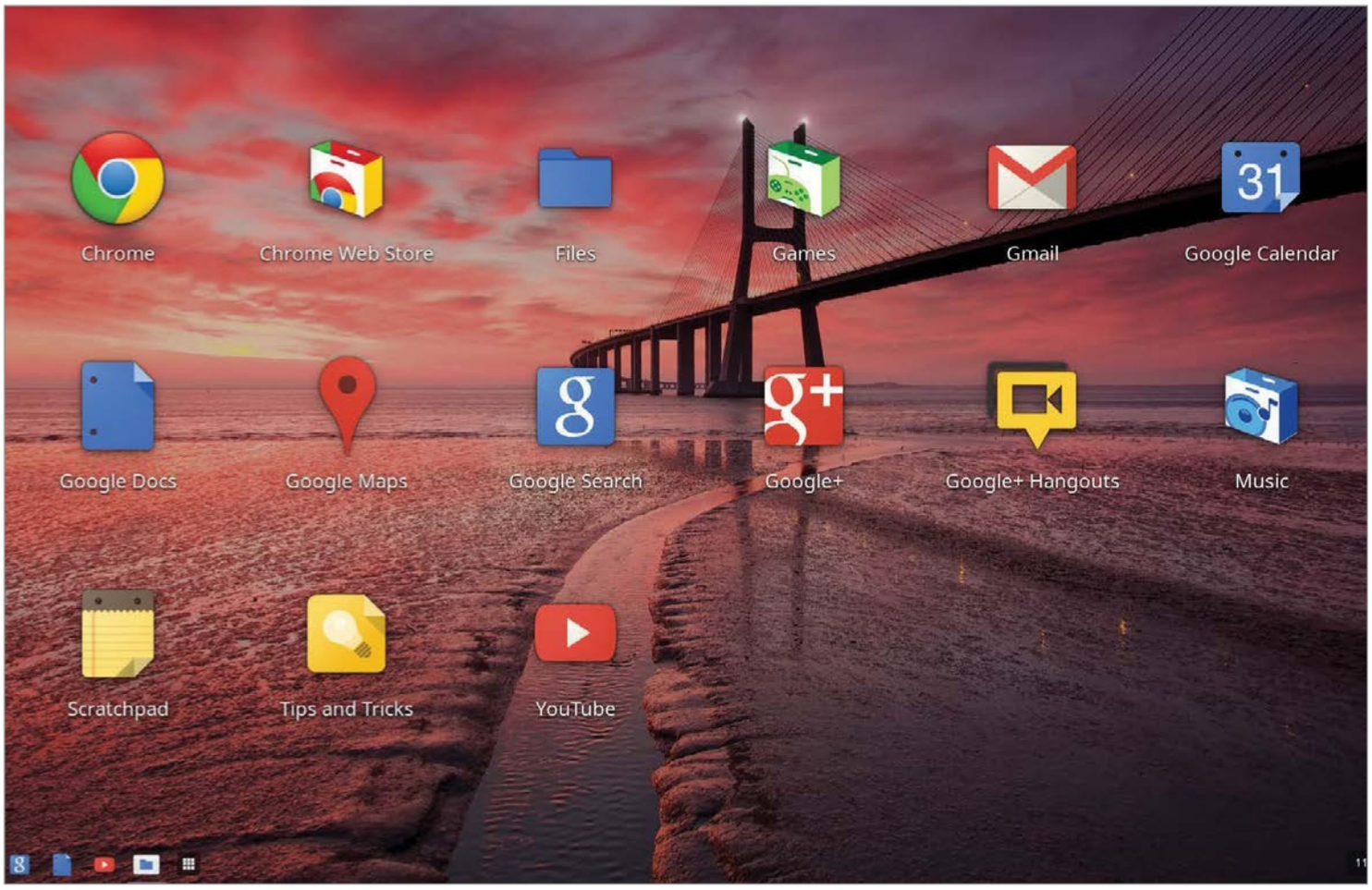
So which distro should you start with?

Clearly the ripest fruit to begin with is Raspbian, as it provides an ideal environment to learn and experiment in. But don't discount Arch or RISC OS: they're best appreciated when you can claim a little more knowledge of the system, Linux and how everything works.

Raspbian is a super OS. It's friendly, easy to use and particularly easy to customise and tweak, once you start to get to grips with the way Linux works. It is widely accepted as the best distro in which to start your Raspberry Pi adventures, is excellent as a teaching base and can offer the most advanced interaction with the Raspberry Pi as further technologies are developed.

However, many users who started with Raspbian have gone onto using Arch after spending time learning how Linux is developed, and as a result now rank Arch higher than Raspbian, due to the complete control it offers the user.





Other operating systems

Learn about the efforts to bring Android and Chromium to the Raspberry Pi

Although the Raspberry Pi has many operating systems that are compatible and configured for use with its hardware, there are some which the users of the Raspberry Pi have clamoured for since its arrival. Two of these in particular are Android – the operating system used on smartphones and tablets; and Chromium OS – the development operating system based on that used in Google Chromebooks.

At this time, there are examples of each available to download and test on the Raspberry Pi. However, they are extremely problematic and are considered as being experimental by the developers and the community as a whole. But that doesn't mean they are impossible to install or experiment with.

The main issue with these other distros is purely one of experience. By this we mean that when embarking on testing these distros you, as a novice user, will most likely be stretching your working knowledge of the Raspberry Pi to its very limits. While this is a perfectly enjoyable experience for some, for others it can be quite stressful, especially when you've spent some time preparing the work only to have it fail without reason. Therefore our advice would be to take it easy, keep the experimentation of the other operating systems a fun project and accept that in the world of computing, things do go inexplicably wrong from time to time. Still, don't be afraid to go for it!

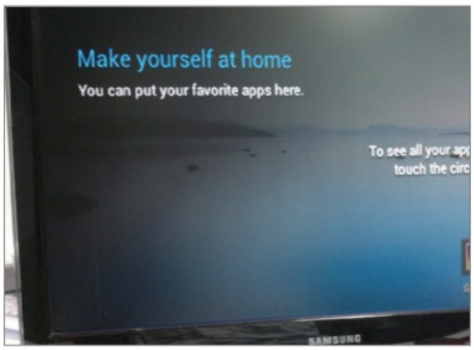
Android on Raspberry Pi

Android is a Google-funded and developed Linux-based operating system designed for mobile devices such as smartphones and tablets.

Android comes in many versions, the latest being version 4.4 (codenamed KitKat), and is a very versatile and modern operating system. There are literally thousands of apps available for it, and it is continually improving and implementing new technologies with every new release. However, herein lies the problem: The modern smartphone is a deceptively powerful device, several times more powerful than the Raspberry Pi, with specifically designed hardware inside. So porting such an advanced operating system to a lesser specified hardware base comes with many challenges.

In the summer of 2012, the Raspberry Pi Foundation announced that a port of Android version 4.0 (Ice Cream Sandwich) is in the works and features hardware accelerated video and graphics, but lacks audio. Unfortunately though, it was pretty unusable by anyone's standards when installed on the Raspberry Pi (Fig 1).

The Android on Pi project has since stalled, as Liz Upton, who handles the PR work for the Raspberry Pi Foundation, stated on the Pi forum: "We've got a finite amount of engineering resources, and we're putting it into things that are important for the educational purpose of the Pi, and making the wider user experience better… Android



■ Fig 1: Sadly, the Android on Raspberry Pi project has stalled, but it's possible it could return in the future

development is dormant for now; we have things that will benefit the *whole* community we're working on."

Chromium OS

Chromium OS is another Linux-based operating system. Designed by Google, it is the open source development version of the Google Chrome OS used on the firm's Chromebooks.

Liam McLoughlin, aka Hexxeh, has been working tirelessly over the last year or so to try to bring the development Chromium OS to the Raspberry Pi. Unfortunately though, he no longer has the time to dedicate himself to the project. To quote his blog entry: "I'd still like to see Chrome on the Raspberry Pi, but I don't have time for it at present, and I'm kind of at a dead end in terms of where to go with it. It's not dead, so I might have something to share with you in the future for this one."

It is possible to install Chromium OS onto the Raspberry Pi (Fig 2). However, the bad news is that it's terribly slow, to the point of being unusable, and you'll need to have some advanced knowledge of how to compile and build images and packages, so it's not exactly beginner-friendly by any stretch of the imagination.

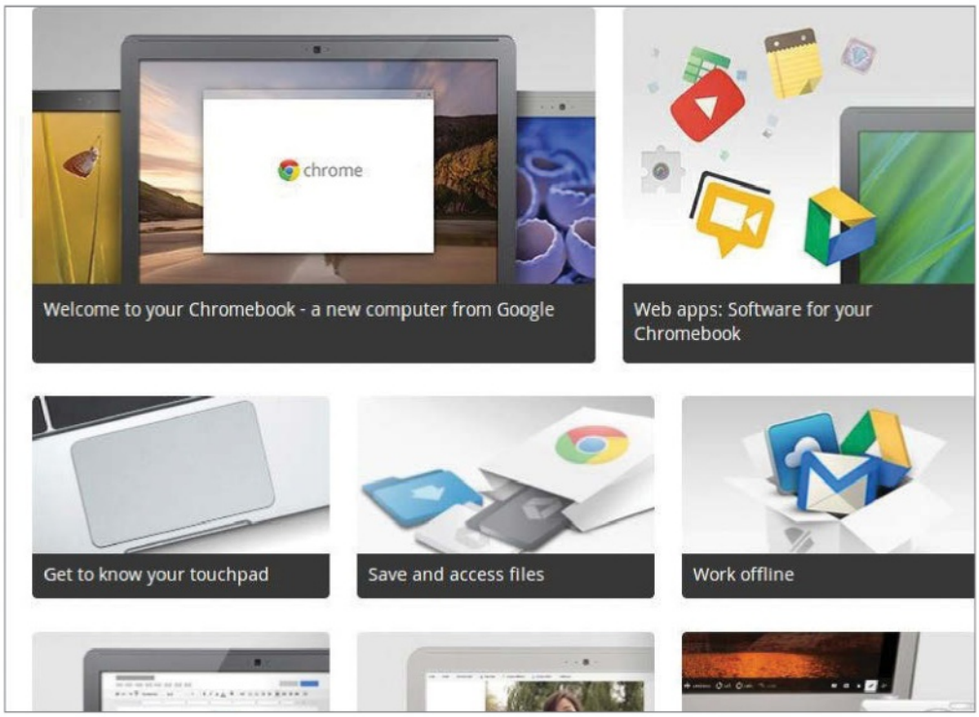
Try them out

Should you wish to try the Android port on the Raspberry Pi, take a moment to point your browser to **goo.gl/gT5De**. This is the official wiki page for the project and towards the bottom of the page you'll find the links to the relevant images.

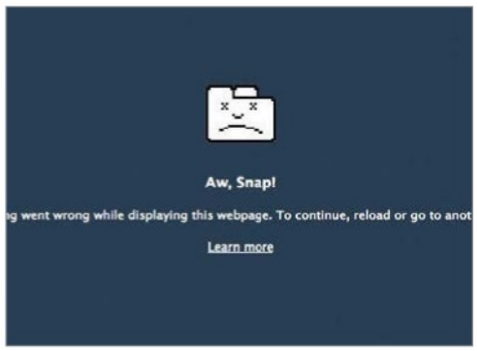
Simply download the image and transfer it to an SD card in the same way you transferred your primary Raspberry Pi operating system. Then, insert the SD card into the Raspberry Pi and apply power.

In time you should be at the Android start screen, but be warned, it's painfully slow, and using a mouse is nowhere near as intuitive in Android as using touch-screen technology.

Getting Chromium OS to run on the Raspberry Pi is a little more complex than that of the Android build. For starters, there are a few prerequisites necessary; however, a blog post located at **goo.gl/bEQdk** should get you on track and ready to test Chromium OS on the Raspberry Pi.



■ Fig 2: Chromium OS can be installed on the Raspberry Pi, but it's terribly unstable and impossibly slow to use



■ If you do decide to test Chromium on the Raspberry Pi, then get used to screens like this

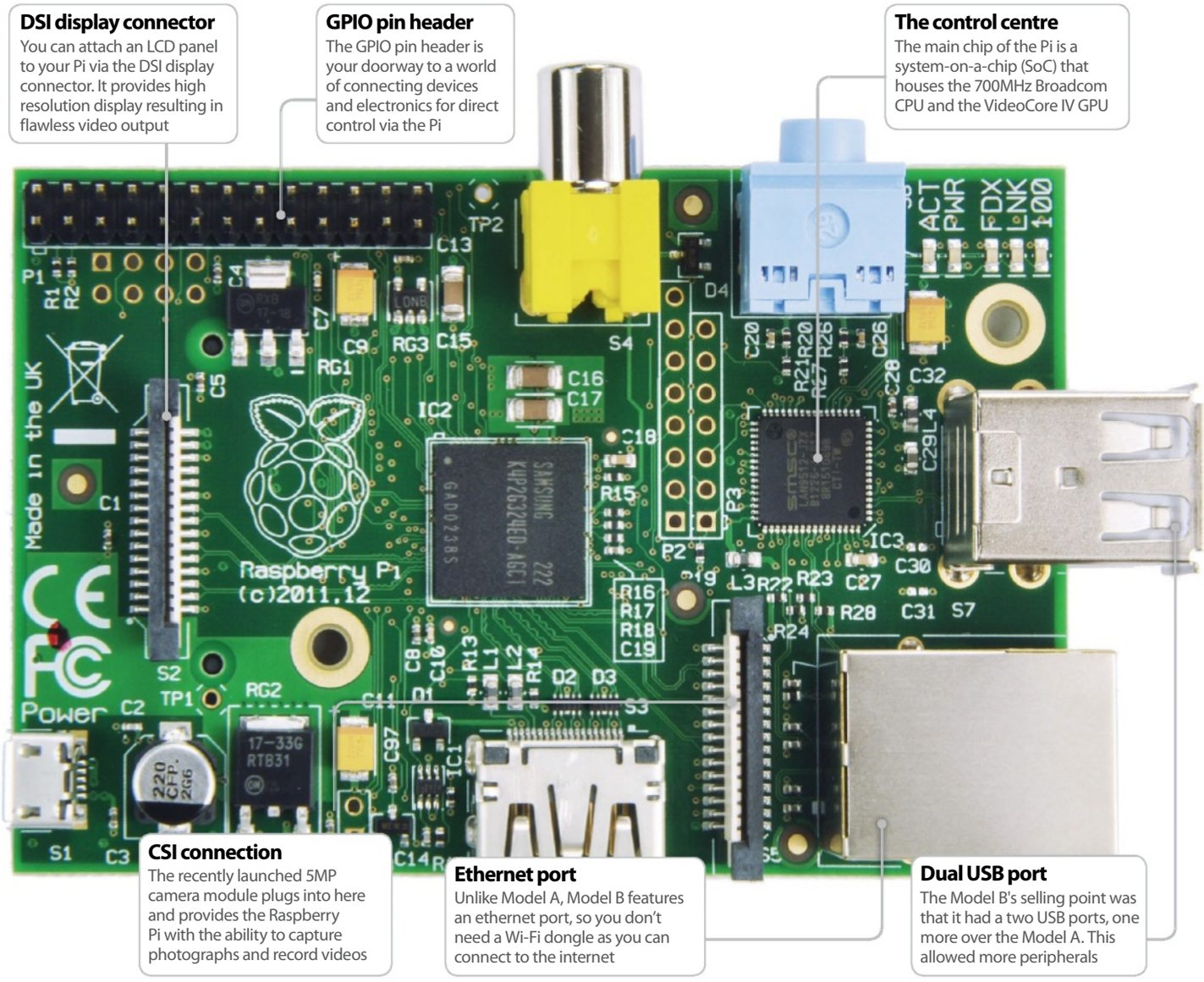
■ Don't be put off by the experimental nature of Android or Chromium for Pi. They just need more work

Even more distros

Take your Pi even further with a different distro

The Raspberry Pi has the potential to play host to some great distributions. We have only scratched the surface here, and there are more being tested, packaged and created almost by the day. There is very little doubt that you will come into contact with some, if not all, of the distros that are mentioned throughout the pages of this book. And we are fairly sure that you will enjoy the time you have with each. One thing to remember when trying out a new distro, though, is that it's a personal thing. What works fantastically with one may not work well with another.

If you are interested in looking into even more distros, then take a moment to browse on over to this Raspberry Pi page: **goo.gl/PPc8z**, and have a look at the lengthy selection of available distros.



Raspberry Pi (Model B)

For most users Model B was the version of choice, and it still is compared to the Model A

The decision to name the Pi versions Model A and Model B comes from the BBC Micro of the early Eighties. The BBC Model A had fewer interfaces and RAM and so was cheaper; the Model B eventually completely replaced it.

With the Pi, the situation was the same and for most people, the Model B was the obvious choice. It had more RAM and, instead of one USB port connected directly to the SoC, it had a hub with an Ethernet (10/100Mb) RJ45 port and two USB ports. This means that a keyboard and mouse can be connected via USB and internet access is available with a direct Ethernet connection to your modem/router. It meant the Model B was easier and quicker to set up and required fewer extra hardware purchases to make operational.

Aside from the RAM and USB hub, the hardware was the same, so both models also featured the 8-pin General Purpose Input/Output (GPIO) header that can be used for interfacing with electronics projects. There's also a CSI input connector for adding the newly launched 5MP camera module.

There's no hard drive in the Pi; it uses solid-state storage courtesy of an SD card that plugs in. The operating system and card must be set up on a Windows or Mac computer, before booting the Pi.

The 26 pins of the GPIO offer a number of different technologies for interfacing with the outside world. Pins 3 and 5 are an Inter-Integrated Circuit (I2C), which is a serial bus interface for connection to simple devices using just two wires. It's limited to fairly slow speeds of up to 100Kbps. On pins 8 and 10 is a Universal Asynchronous Receiver/Transmitter (UART) for accessing the serial console. There's a clock signal pulse on pin 7 and for more sophisticated sensors and simple displays, there's the Serial Peripheral Interface Bus on pins 19, 21, 23, 24 and 26. This is a two-way serial connection with synchronised timing on pin 23, the SCLK. Finally, over on pin 12 is a Pulse Width Modulation (PWM) power supply for lighting up LEDs and controlling simple motors.

Raspberry Pi (Model B+)

The latest version of the Raspberry Pi improves on the Model B in many ways, notably with more USB ports

The Model B is far and away the most popular version of the Pi, with a lot more functionality and a tiny bit more power for only marginally more money. The Model B already received a stealthy update shortly after it was released by giving it some more RAM but the Raspberry Pi Foundation has seen it fit to release a third iteration of the Model B called the B+.

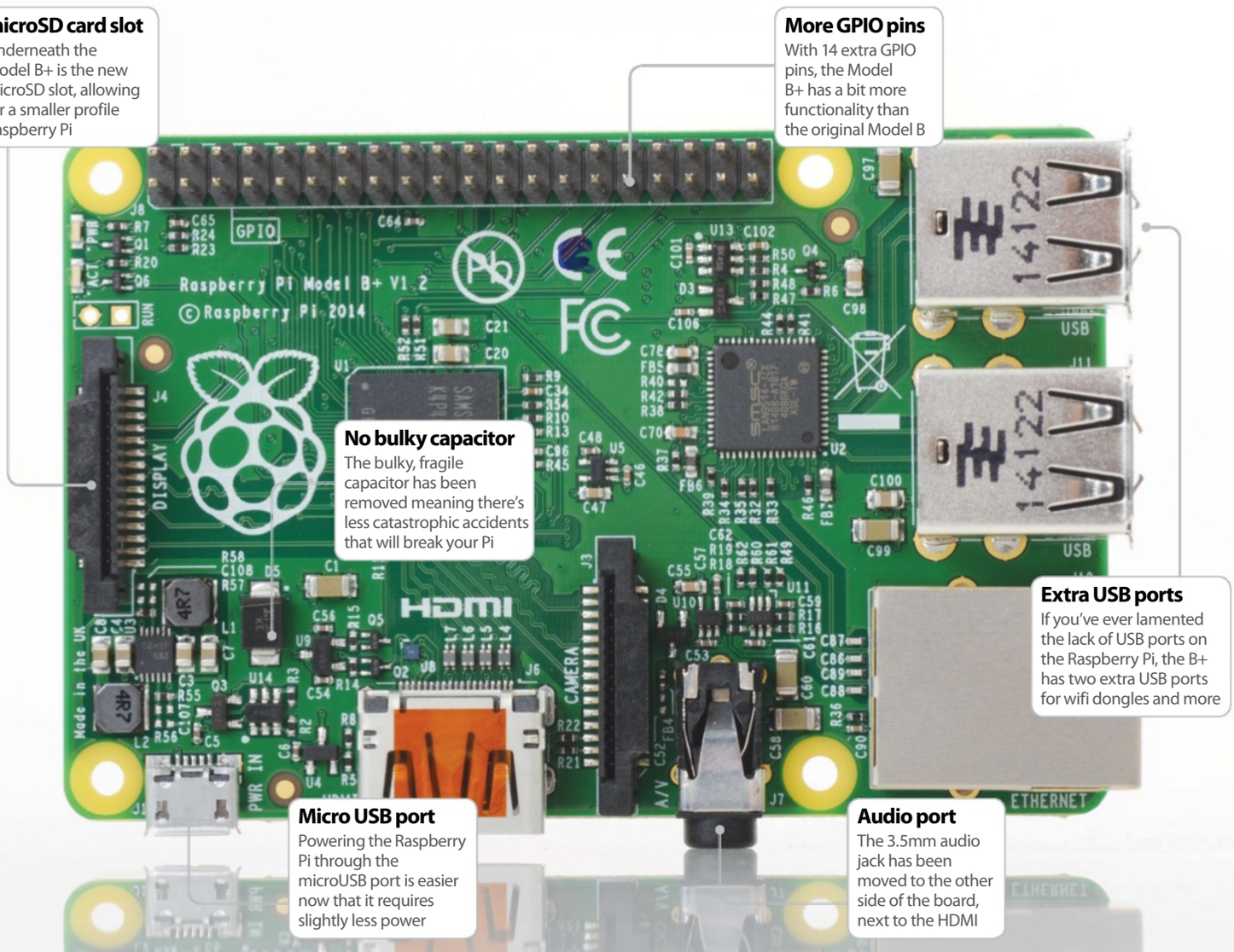
Technically, all the core chips and RAM are the same in the B+ meaning it's no faster or more capable than the Model B; in fact benchmarking shows it's exactly the same. The main difference is the rearranged components on the board itself along with the removal of the Video out port, changing the SD card port to microSD and adding on two more USB ports.

The new arrangement also has all the inputs and outputs along two sides of the Raspberry Pi, allowing for a more tidy workspace or project. There are also 14 extra GPIO pins to bring the count up to 40, with the first 26 having the exact same functions as the original layout. This means your old projects will still work on the new Pi model with the same pin configuration.

The are two more clock signal pulses on pins 29 and 31, along with three extra Serial Peripheral Interface Bus pins on 36, 38 and 40. There's some Ground pins and a few hybrids on the rest of the pins but it's basically more of the same so that you can have a few more options.

One of the important changes not well advertised is that the Model B+ also draws a lot less power than the Model B – this efficiency has a number of benefits such as being powered easier from a computer's USB port to just saving a little bit of money on the electric bill.

If you have the original Model B it's definitely not an essential upgrade but it is a very logical one – it's also supposed to be the last version of the Model B.



microSD card slot
Underneath the Model B+ is the new microSD slot, allowing for a smaller profile Raspberry Pi

More GPIO pins
With 14 extra GPIO pins, the Model B+ has a bit more functionality than the original Model B

No bulky capacitor
The bulky, fragile capacitor has been removed meaning there's less catastrophic accidents that will break your Pi

Extra USB ports
If you've ever lamented the lack of USB ports on the Raspberry Pi, the B+ has two extra USB ports for wifi dongles and more

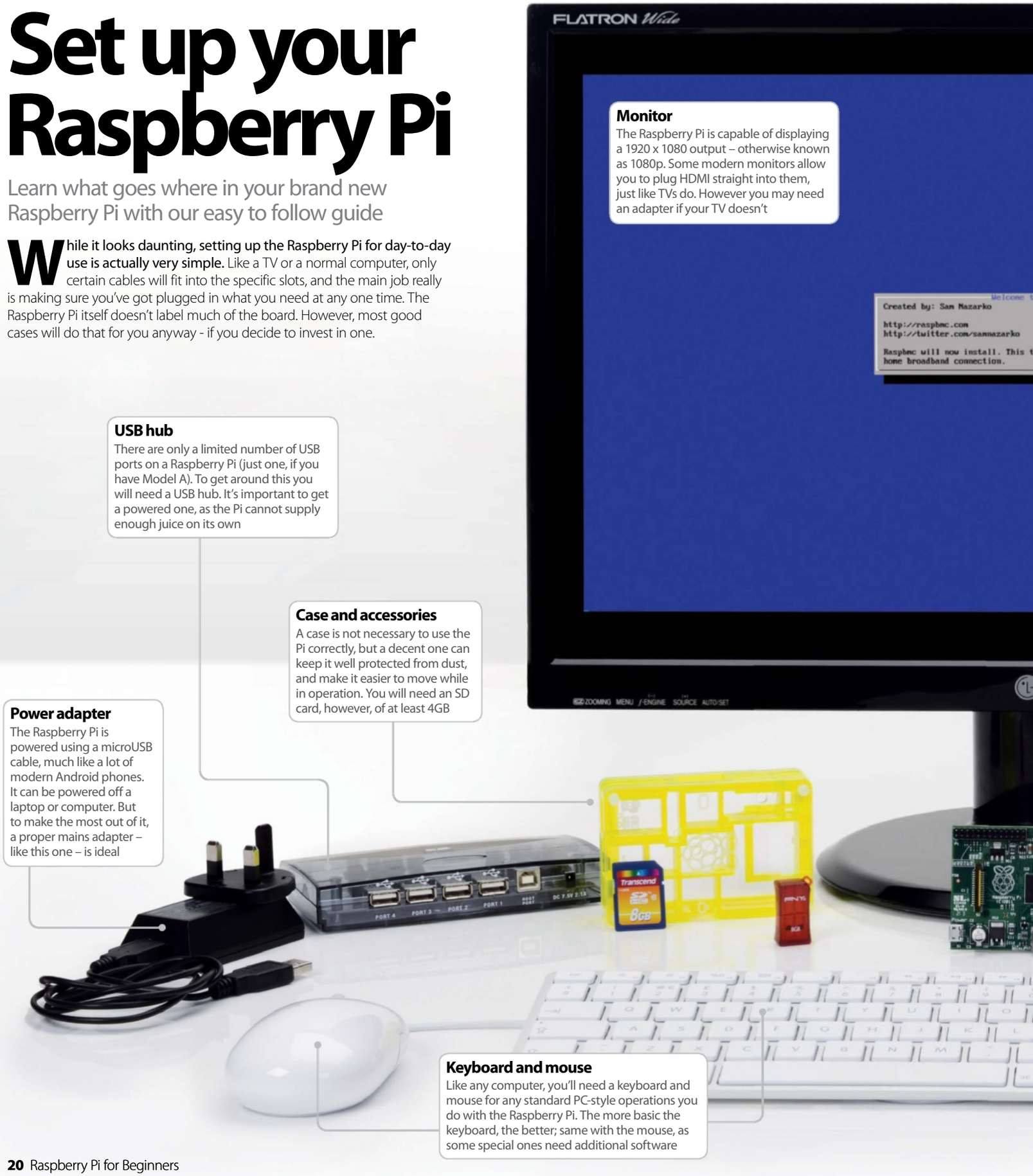
Micro USB port
Powering the Raspberry Pi through the microUSB port is easier now that it requires slightly less power

Audio port
The 3.5mm audio jack has been moved to the other side of the board, next to the HDMI

Set up your Raspberry Pi

Learn what goes where in your brand new Raspberry Pi with our easy to follow guide

While it looks daunting, setting up the Raspberry Pi for day-to-day use is actually very simple. Like a TV or a normal computer, only certain cables will fit into the specific slots, and the main job really is making sure you've got plugged in what you need at any one time. The Raspberry Pi itself doesn't label much of the board. However, most good cases will do that for you anyway - if you decide to invest in one.



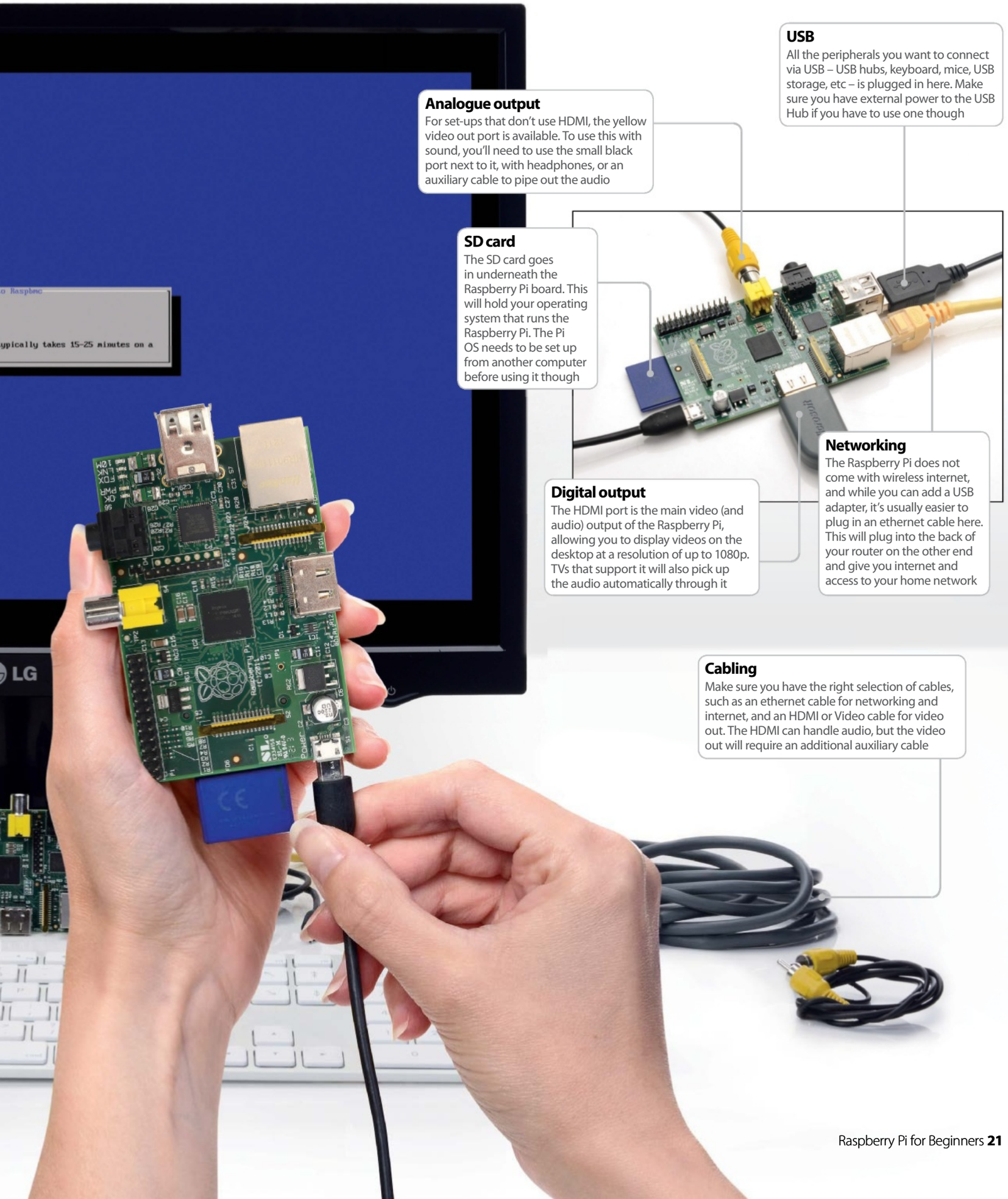
Monitor
The Raspberry Pi is capable of displaying a 1920 x 1080 output – otherwise known as 1080p. Some modern monitors allow you to plug HDMI straight into them, just like TVs do. However you may need an adapter if your TV doesn't

USB hub
There are only a limited number of USB ports on a Raspberry Pi (just one, if you have Model A). To get around this you will need a USB hub. It's important to get a powered one, as the Pi cannot supply enough juice on its own

Case and accessories
A case is not necessary to use the Pi correctly, but a decent one can keep it well protected from dust, and make it easier to move while in operation. You will need an SD card, however, of at least 4GB

Power adapter
The Raspberry Pi is powered using a microUSB cable, much like a lot of modern Android phones. It can be powered off a laptop or computer. But to make the most out of it, a proper mains adapter – like this one – is ideal

Keyboard and mouse
Like any computer, you'll need a keyboard and mouse for any standard PC-style operations you do with the Raspberry Pi. The more basic the keyboard, the better; same with the mouse, as some special ones need additional software



USB
All the peripherals you want to connect via USB – USB hubs, keyboard, mice, USB storage, etc – is plugged in here. Make sure you have external power to the USB Hub if you have to use one though

Analogue output
For set-ups that don't use HDMI, the yellow video out port is available. To use this with sound, you'll need to use the small black port next to it, with headphones, or an auxiliary cable to pipe out the audio

SD card
The SD card goes in underneath the Raspberry Pi board. This will hold your operating system that runs the Raspberry Pi. The Pi OS needs to be set up from another computer before using it though

Networking
The Raspberry Pi does not come with wireless internet, and while you can add a USB adapter, it's usually easier to plug in an ethernet cable here. This will plug into the back of your router on the other end and give you internet and access to your home network

Digital output
The HDMI port is the main video (and audio) output of the Raspberry Pi, allowing you to display videos on the desktop at a resolution of up to 1080p. TVs that support it will also pick up the audio automatically through it

Cabling
Make sure you have the right selection of cables, such as an ethernet cable for networking and internet, and an HDMI or Video cable for video out. The HDMI can handle audio, but the video out will require an additional auxiliary cable

The Raspberry Pi starter kit

There's more to your Pi than first meets the eye. Here are some vital peripherals to get you started

In order to get the very best experience from your Raspberry Pi, you're going to have to get hold of a few extras on top of the actual Raspberry Pi itself. For example, you're going to need a keyboard and mouse with which to enter commands and navigate with. Also, an SD card on which to store the operating system. There's also power in some form or another, maybe even a USB hub so you can attach more USB devices.

Perhaps you'll need a Wi-Fi adapter of some description, or maybe just a length of network cable. Then there's the basic electronics side of the Raspberry Pi, what would you need to start some of the beginner electronics and control experiments? Clearly, there's more to the Raspberry Pi than you have first thought.

When we say peripherals, we of course mean other hardware that can be attached and utilised by the Raspberry Pi. They could be something as simple as a decent HDMI cable with which to hook up the RPi to your TV with, or they could be the newest RPi bespoke gadget that enhances the project capabilities of the Raspberry Pi.

There is an entire world of possibilities available for the Raspberry Pi; from robot arms to remote-controlled helicopters. The only limit being the hardware available.

Keyboard and mouse

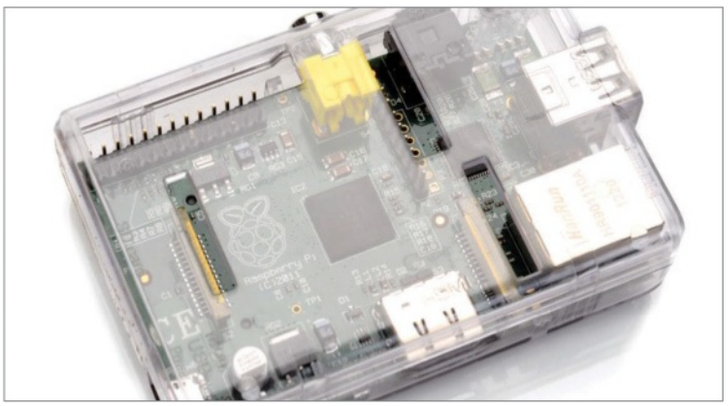
Let's start with the most basic of components, the keyboard and mouse. Generally speaking, virtually any USB keyboard and three-button scroll mouse will work with the Raspberry Pi, and although for some projects you won't even need a keyboard and mouse, it's a good idea to start off with them. For a full list of confirmed working USB keyboards and mice, check out **goo.gl/YjXNG**, and **goo.gl/2cbhW**.

SD card

The SD card is the storage device that contains the Raspberry Pi operating system, whatever you decide that to be. These can be bought pre-installed with the OS, or blank from a number of sources. SD cards come in a variety of sizes and speeds, but for the basics a card of 2GB or over is acceptable as the base model. A minimum of 4GB is recommended for additional programs. For more info on SD cards, have a look online.

Power cable

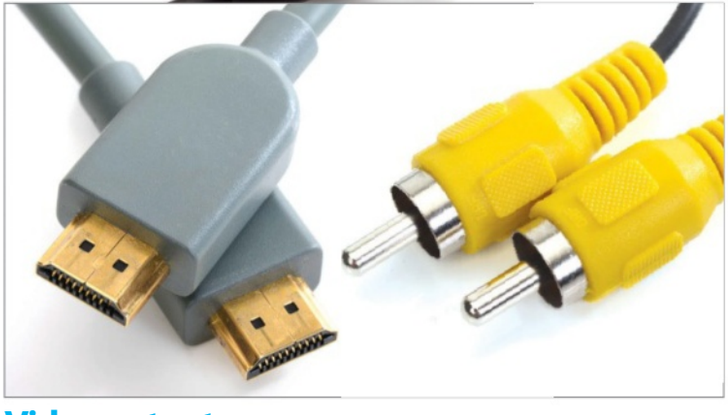
The Raspberry Pi uses a standard micro USB connector for its power input, which should run at 5V. In most cases a micro USB to USB cable will suffice, of which one end can be plugged into a laptop or PC USB port, as will a HTC phone charger (although these can be too much for the Pi depending on the charger). Generally speaking, a 5.25V 1500mA, will be perfect. Again, for more info have a look at this Raspberry Pi link: **goo.gl/2rmFS**.



Case

Although not totally essential, placing your Raspberry Pi in a case will obviously protect it and prevent it from ever being thrown away by accident, or damaged by some other catastrophe. Aside from accidental damage, the case can make your Raspberry Pi a more attractive or striking unit, for use as a living room media centre for example. There are many cases available online, in a range of styles and colours.

"With the right hardware, you can use your Raspberry Pi to create some wonderful projects"



Video output

The Raspberry Pi comes with two video output ports, a HDMI port and an RCA Socket. Most of us are familiar with the HDMI port, which is more than likely to be the primary video-output connector for most users. However, if you don't have HDMI on your TV or monitor, then the RCA video-out port can be used to connect TV's, monitor's or to a SCART cable if necessary. For a more detailed explanation, follow this Raspberry Pi link: **goo.gl/sJNCg**.

Powered USB hub

Having extra USB ports handy is worth considering when you begin to expand your Raspberry Pi. The Raspberry Pi only comes with two USB ports, which can be taken with a keyboard and mouse, so the more you have, the more you can attach. Using a powered USB hub will stop any power being drained from the Pi, and allow you to attach the likes of an external hard drive, for example.

Raspberry Pi camera board

This is a custom designed add-on that attaches to one of the Raspberry Pi's on-board sockets via a flexible cable bus. It's extremely small, but remarkably powerful, having a native resolution of five megapixels and supporting 1080p video. It's currently for sale from a few locations, chiefly Element 14: **goo.gl/SXCWo**. In addition, the Raspberry Pi Foundation have a great tutorial on how to activate it in Raspbian, found here: **goo.gl/qBwHK**.

USB Wi-Fi adaptor

Using a USB Wi-Fi adaptor will free up the location of the Raspberry Pi, to a degree. You'll no longer have to run an Ethernet cable from your router, and it could be used for more advanced projects where running a wired internet connection isn't a valid option. Whatever the reason, there are plenty of USB Wi-Fi adaptors available that are compatible with the Raspberry Pi; check out this list for an idea on what to look for: **goo.gl/ZoPuz**.

Connect your Pi to a network

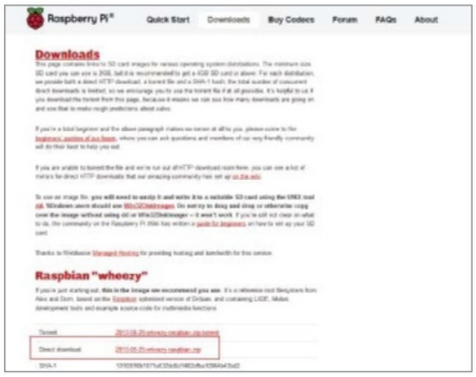
Getting the Raspberry Pi up, running and connected to your network has never been easier

Before we begin anything, let's make sure we've ticked off everything on our checklist. You'll need: a Raspberry Pi (Model B), compatible SD card, micro USB cable/charger, network cable, HDMI cable, compatible TV, USB keyboard and mouse. You can also use the analogue TV out, but HDMI is highly recommended.

We'll be focusing on the most common and useful configuration for new users; a combination of the Raspbian Linux distribution (also known as 'wheezy') with an HDMI output from an existing Windows PC. Starting with the OS download we'll walk through creating the image on your SD card, connecting the peripherals and the installation process. After that we'll walk through the Raspberry Pi Config Tool options. We'll expand your SD card partition to fill the full card allowing you more space for applications etc. We'll change your password from the unsecure default, and we'll set up all your localisation settings; keyboard layout, locale and timezone. If the command prompt isn't your thing we'll set the Pi to boot directly into the desktop on start up. We'll even enable SSH (secure shell) access so you can open a terminal window across your network. Finally we'll find out your existing network settings and configure your Pi accordingly.

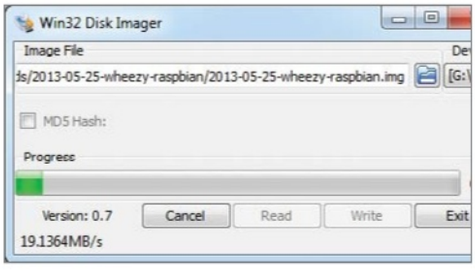
On completing these steps you'll have a fully functional Raspberry Pi as a operation desktop machine. Use as you would any other PC for day-to-day internet, email and word processing or break it out and let it live up to its full potential with some of the great projects in these pages.

> "We'll be focusing on the most common and useful configuration for new users"



01 Download the software

Head to **http://www.raspberrypi.org/downloads** and download the latest Raspbian image, if you download the compressed zip, unzip it to a location of your choosing. Head to **http://bit.ly/VOUamj** and download Win32 Disk Imager. Unzip and run Win32DiskImager.exe.

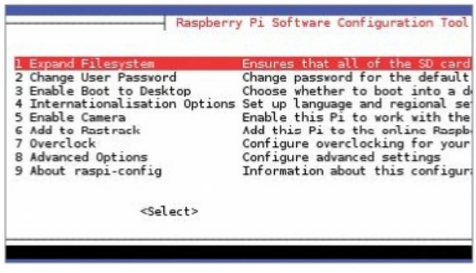


02 Creating your image

Insert the SD card into an SD card reader. A good tip here for reusing and resetting the partitions on an SD card is to use a camera to format it. Select the drive of the SD card under Device and point to the unzipped image file and hit Write.

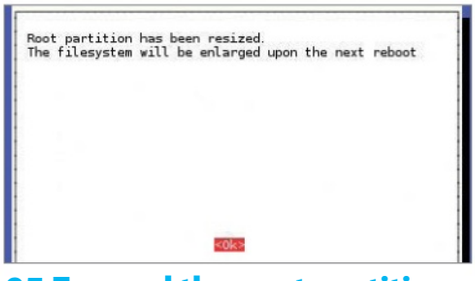
03 Connecting your Pi

Connect the keyboard and mouse to the USB ports. Connect the network cable from an existing network connection such as your router into the network port and connect the HDMI cable from a TV. Insert the SD card into the Pi. Insert the micro USB cable from a power source.



04 The Config tool

On boot up, after the initial scrolling text you'll see the Config tool. Here you can configure many of the Raspberry Pi features quickly and easily. You can also run this tool at a later date with the command sudo raspi-config. Esc, Tab and Space can be used to navigate the tool.

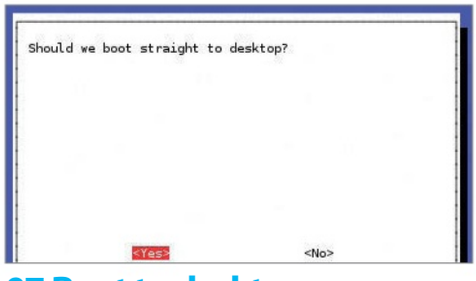


05 Expand the root partition

By default not all the storage on the SD card is used. By selecting the Expand Filesystem option, you can expand that partition to take up the full capacity of the SD card. This is highly recommended as it will give you far more storage capabilities.

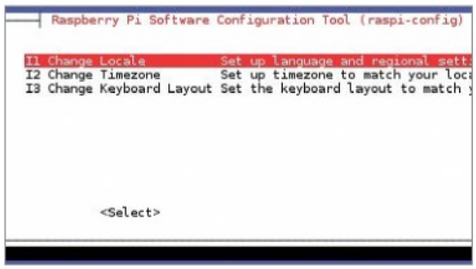
06 Changing the password

The default username and password for the Raspberry Pi is "pi" and "raspberry" respectively. It's highly that you change the password to something more secure. Change User Password will guide you through that process. Remember this, if you lose it you will not be able to recover it.



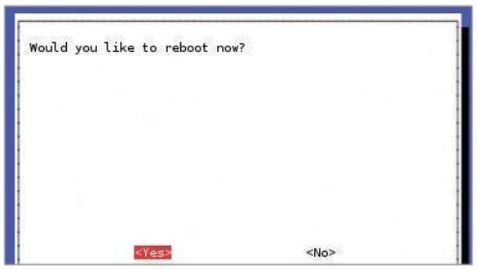
07 Boot to desktop

The Config tool is very useful, but you don't want to see it on every boot. While you can use the startx command to boot to the desktop it's far more useful to Enable Boot to Desktop. The Pi will then boot directly into your desktop environment on every boot.



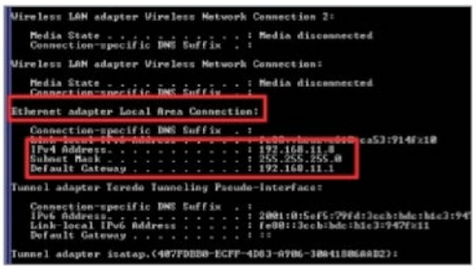
08 Localisation options

Use Internationalisation Options to change the default locale, timezone and keyboard layout. The setup will guide you through the various options with a recommended default. This is important as it's used by applications being installed to import language settings, customisations and more.



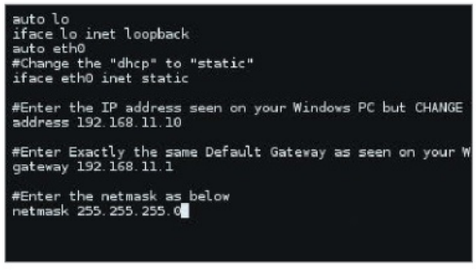
09 Finalising the changes

After making your required changes select Finish. You'll be asked if you would like to reboot, select Yes. This initial reboot will take longer than normal as all the changes, such as resizing the partition, are made. When done you'll be taken to your shiny new Raspberry Pi desktop.



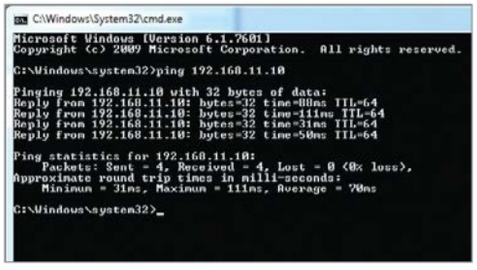
10 Finding network details

To find out your current network details head back over to your Windows PC. Use Start>Run> "cmd" to bring up a command prompt. Enter the command ipconfig. Look for your IP address and Default Gateway and note them down, you need these for the Pi.



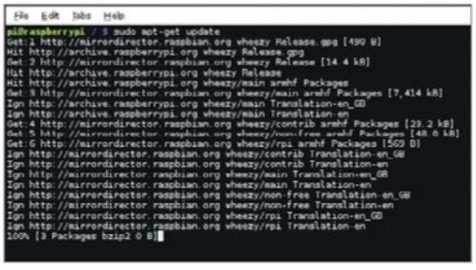
11 Network settings

Open a Terminal window on the Pi. Enter the command sudo nano /etc/network/interfaces. Change the line starting iface eth0 to read "static" instead of "dhcp". Enter the address details as shown in the image. Only the last digits of the IP address should differ from your Windows PC.



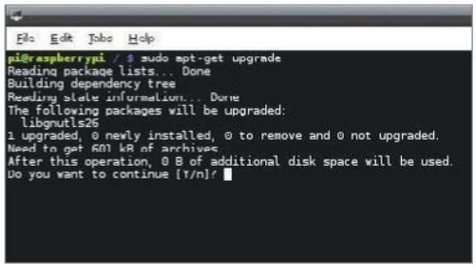
12 Test your connection

Back on your Windows PC open the command prompt again (Start>Run>"cmd"). This time run the command ping [enter IP Address of Pi]. So if you just gave the Pi an IP Address of 192.168.11.10 you would use ping 192.168.11.10. You should see a reply and no timeouts.



13 Up-to-date repositories

Linux uses what we call repositories. These are centralised locations of the latest and greatest software packaged and that can be downloaded and installed. These need to be kept up-to-date. Open the Terminal and type sudo apt-get update. This will get the latest locations for your software.

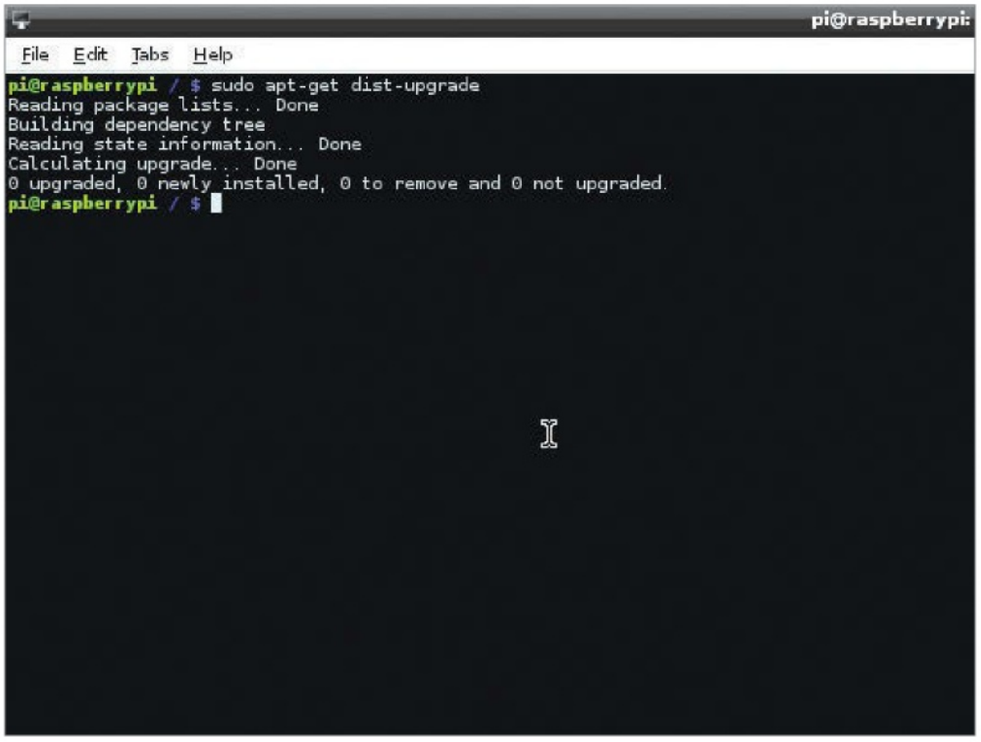


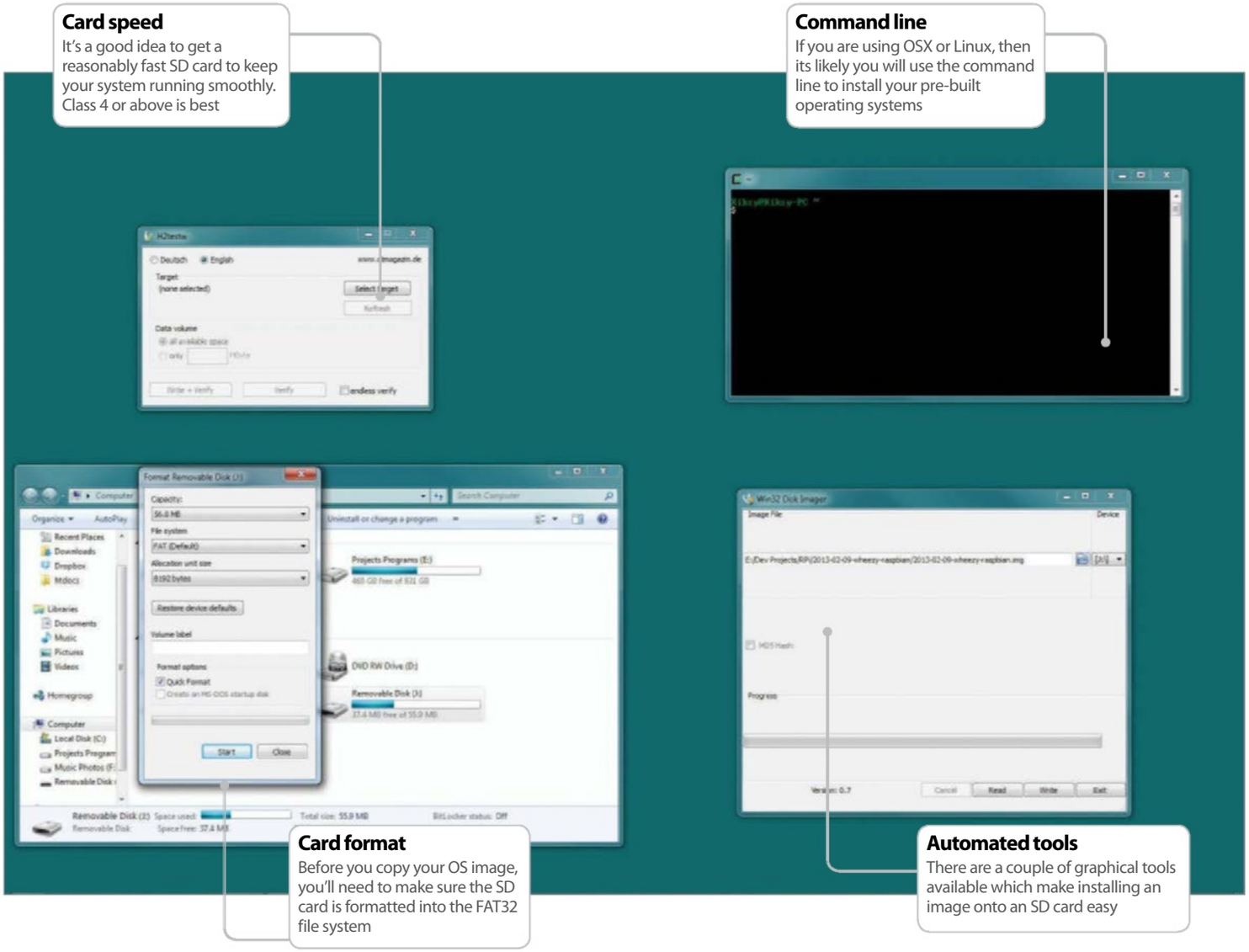
14 Update your software

Now we have the latest repository information we can make sure all the installed software is up-to-date as well. Again from the Terminal prompt enter the command sudo apt-get upgrade. If prompted answer "y" to any questions, this will upgrade any out of date application software.

15 Updating the distribution

Finally we need to make sure that our distribution is up-to-date. This will include the base operating system updates such as kernel improvements, driver updates or even some distribution specific application. This time, you will need to use the command sudo apt-get dist-upgrade. Again answer "y" to any installation questions and then you're done.





Install an operating system

Taking a look at some of the key aspects involved in installing a pre-built OS

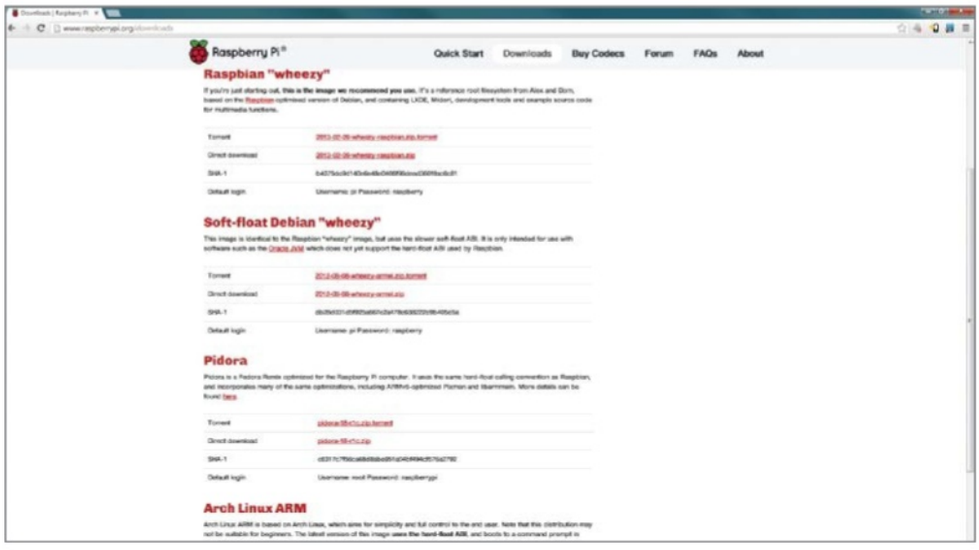
With its small size and cheap price, many people might be fooled into thinking that the Raspberry Pi is only usable for basic tasks, and learning to program on. While one of the primary goals of the Pi was to increase computer literacy at a lower level rather than just learning how to create Excel spreadsheets, the Pi has many other great uses.

As the Raspberry Pi is essentially a mini PC, with an HDMI and analog TV output, rather than a traditional monitor connection, it can perform many common tasks that a laptop or desktop is often used for. While it doesn't really have the processing power or RAM to run the latest version of Windows, there are other options.

There are a wealth of fully fledged operating systems, many forked from their desktop big brothers that have been optimised specifically for the Pi. One of the most popular of these is Raspbian, which is a port of Debian. Debian is a key part of the Linux ecosystem, and many other popular open source distributions are forked from the Debian source code. The original Debian was released in 1993, and its come a long way since. Raspbian needed work to get performance levels up to standard, as the Pi uses the older ARMv6 architecture. Its now a great everyday desktop. Another popular Linux distribution is Pidora, which is a Fedora Linux remix, again specifically tailored for the Raspberry Pi.

Resources

Raspberry Pi downloads
http://www.raspberrypi.org/downloads

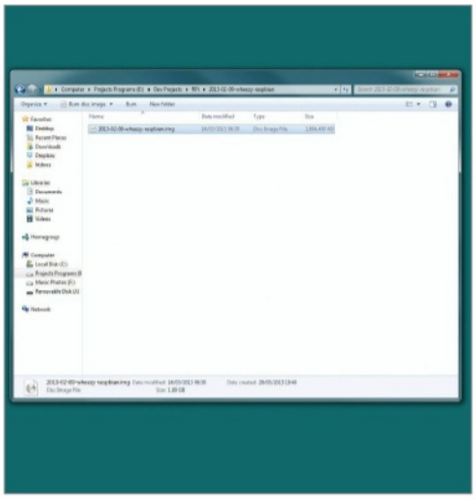


01 Obtaining OSs

One of your first questions may be: "Where can I find some operating systems to download?" Most of the common images can be found on the main Raspberry Pi site: **http://www.raspberrypi.org/downloads**. These are stable and well-tested systems, and the best place to start.

02 Using NOOBS

On the OS download page, you'll also notice NOOBS. It's an easy-to-use program to install many OSs to your Raspberry Pi. All you need to do is unzip the files to a freshly formatted SD card, and follow the instructions on your Raspberry Pi.

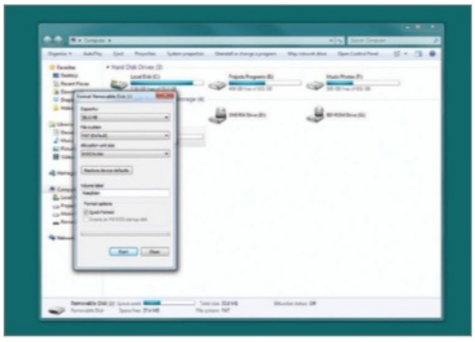


03 OS Format

Within the zip, most likely there will be a file with a .img or .iso extension. These are the equivalent of a 'snapshot' of an installation CD or DVD. Simply copying the file to the Sdcard won't do anything, you'll need to use a program to extract it.

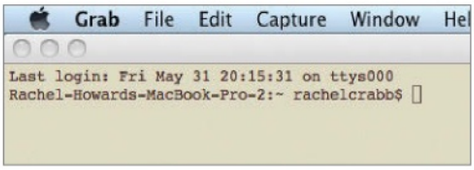
04 SD card format

The SD card that you'll boot from needs to be blank, so make sure there is nothing important on it first. You'll also need to format it to use the FAT32 file system. This is a common system, used by most USB sticks and cameras.



05 Formatting the card

In Windows, to format the card simply insert and wait for it to mount. Then click on 'My Computer' and then right click on the cards icon. After that choose format and then 'FAT32' from the dropdown menu.



06 Using the terminal

If you are using OS X or Linux, then you'll have to use the terminal to copy the image. In OS X, the Terminal app comes installed by default, and most Linux versions come with one in some form or other. It may be referred to as the 'console' or 'command line'.

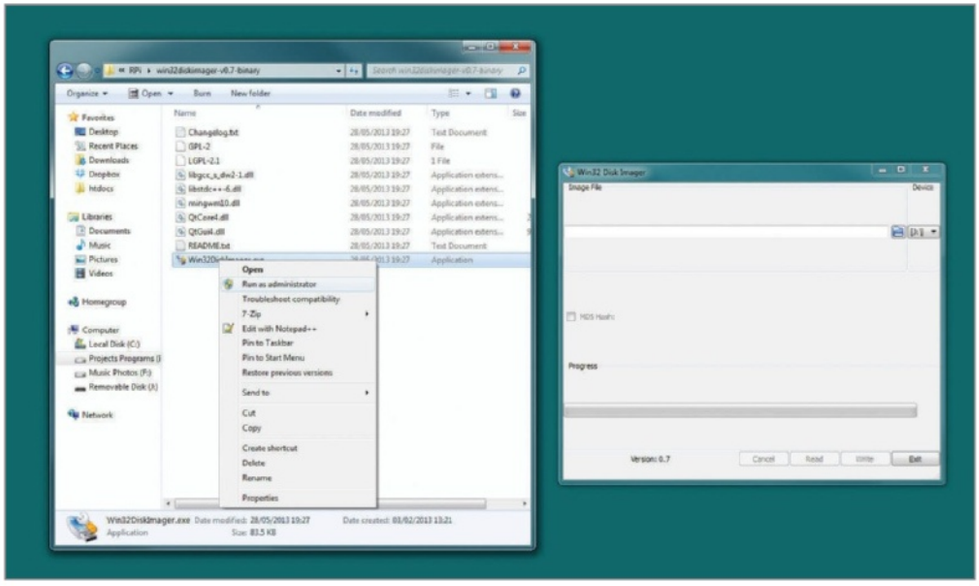
07 DD command

The command you need to use is called 'dd'. This is entered in the format of 'sudo dd bs=1m if=[img] of=/dev/[sdcard]'. An example of this can be seen below:

```
sudo dd bs=32m if=/Users/rachelcrabb/Desktop/ArchLinux/archlinux-hf-2013-02-11.img of=/dev/disk1
```

08 Win 32 Disk Imager

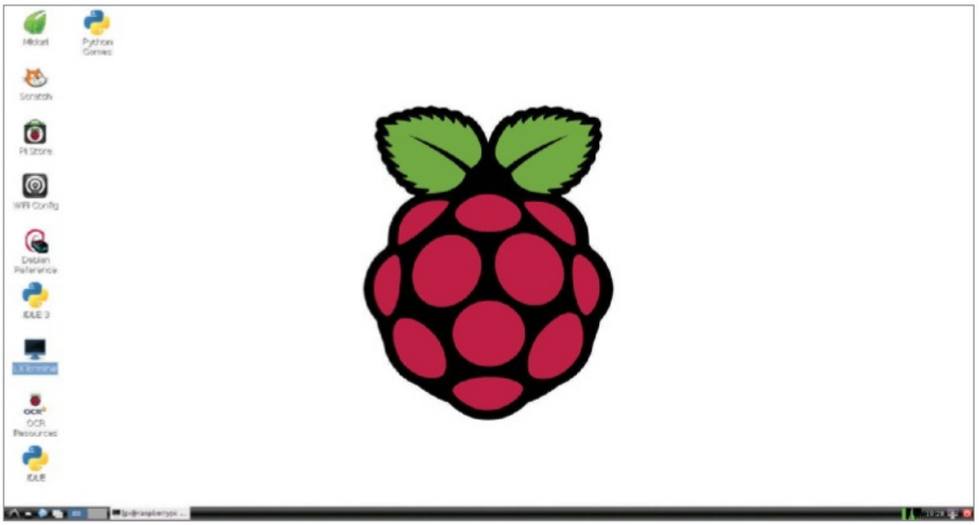
If you are using Windows, there is a handy tool called Win32 Disk Imager, which means you don't have to worry about entering any terminal commands. Once you've downloaded the tool, simply right click on the .exe, and choose 'run as administrator'.



Install Raspbian from Windows 7

Learn how to install one of the most popular desktop solutions for your Pi

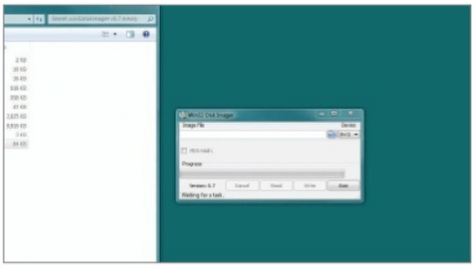
As soon as the Raspberry Pi was announced, there was great excitement among the developer community as to what could be possible with the inexpensive computer. There have been lightweight versions of popular Linux distributions before, and it wasn't long until early builds of Pi-compatible Debian were released to the public. One of the most popular and well supported is Raspbian. One hurdle that was to be overcome revolved around the processor support. The mainline Debian builds only supported ARMv7, whereas the Pi uses the slightly older ARMv6 architecture. Raspbian supports many of the most popular software packages and is a great all-in-one desktop solution for your Pi. Regularly updated and maintained, it offers decent performance while still being fully featured. This guide will take you through the steps of installing Raspbian on your Pi from a Windows 7-based machine, but it's possible on any operating system.



■ Raspbian's LXDE desktop environment will feel familiar to most computer users

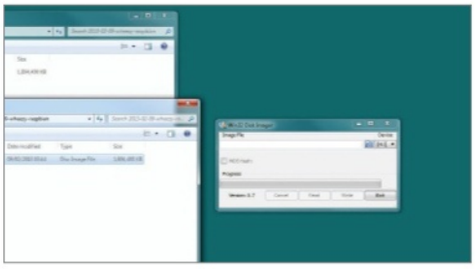
01 Download Raspbian

To get things started, on your desktop or laptop go to **www.raspberrypi.org/downloads** and download the latest 'Wheezy' release of Raspbian. The download is around 500MB – much bigger than some of the other distros – which reflects its nature as a fully featured GUI-based distro.



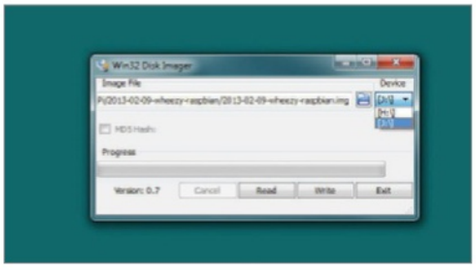
02 RPI installer

Now head to **http://sourceforge.net/projects/win32diskimager/** and download the Win32diskimager software. This is what we'll use to copy our Raspbian image to the SD card. Unzip the **wheezy-raspian.zip** file when it has finished downloading to your desktop.



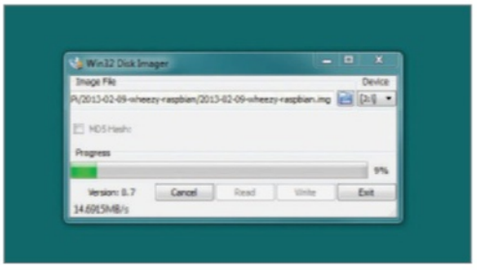
03 Select image file

Select the image file in Win32 Disk Imager by clicking on the small folder icon just under the Image File title. This will bring up a file choice window. Select the Raspbian image that we just extracted from the zip file and then click on OK to select it for copying.



04 Choose target device

Now, under Device, choose your SD card. Make sure that it is formatted as FAT32 (the file system required by the Pi) first. Also make sure that you choose the correct drive here, as all data on the target device will be lost. Once that's done, click Write to start the process.



05 Writing image

After confirming that you really want to erase everything on the card, the image will be written to the SD card. Depending on the speed of the card, this may take a few minutes. If you have an issue writing the file, try running the Win32 Disk Imager tool as administrator.

06 Boot Raspbian

Once the image has finished being written, eject the SD card from the computer and insert it into the Raspberry Pi. Power up and you'll first see a setup menu. Press Enter as the defaults are fine for now. At the command prompt, type 'startx' and press Enter to load the GUI.

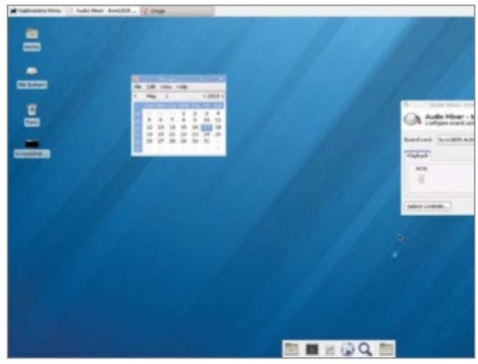
Install Pidora

Learn how to install a Fedora distro optimised specifically for the Raspberry Pi

While many Linux distributions are based on Debian, there are plenty of others with different goals and aims. One great alternative is Fedora, which has fast release cycles with relatively short support terms, so new features are added regularly. There is a Raspberry Pi-specific remix available, known simply as Pidora. Pidora incorporates some of the hardware acceleration features that Raspbian has, so performance is good while still being a complete desktop solution.

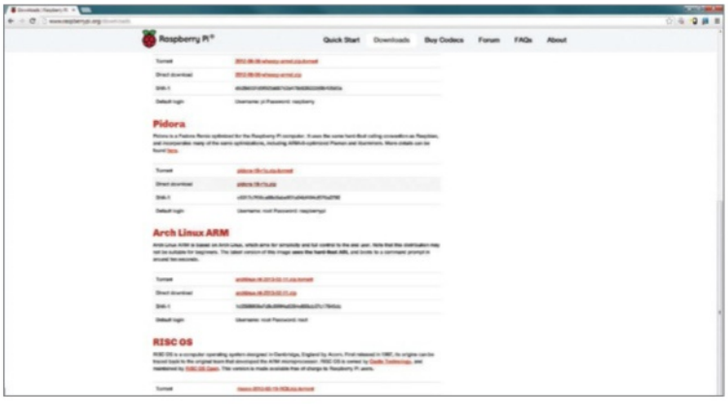
So why would you use Pidora over Raspbian? Firstly, Pidora offers a different desktop environment, the lightweight Xfce, whereas Raspbian uses LXDE. Both have their merits but depending on your requirements, one may suit over the other. The other difference is the packages and package manager used to install software: Raspbian uses Apt, while Pidora uses YUM.

In this short guide, we'll show you how to install and set up Pidora on your Raspberry Pi.



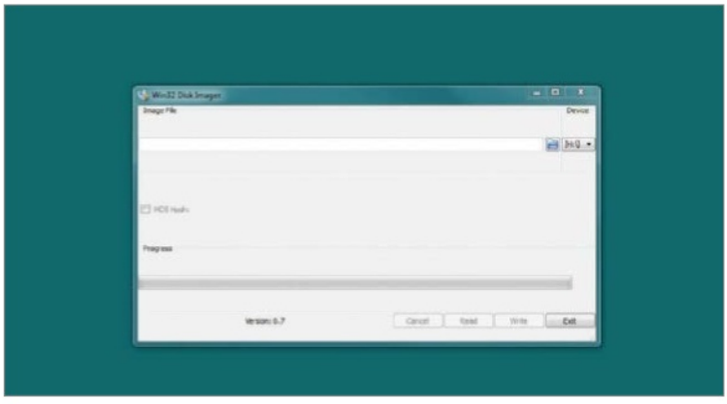
■ Pidora's Xfce desktop environment uses few system resources while being easy to use

"Fedora has fast release cycles, so new features are added regularly"



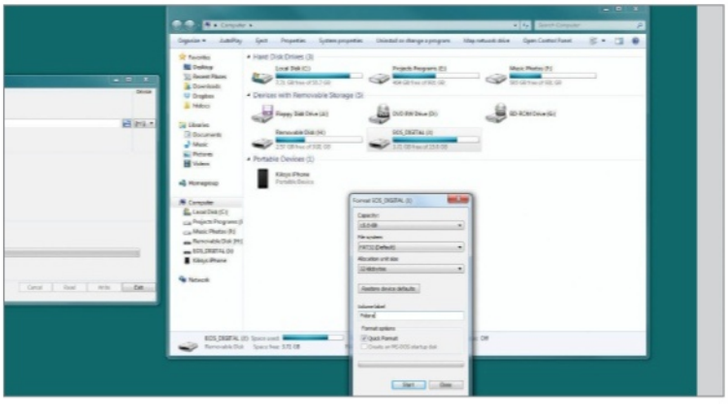
01 Download Pidora

First, you need to get a copy of the latest Pidora image. This can be found on the main Raspberry Pi download page: **www.raspberrypi.org/downloads**. The image is around 500MB in size. Once it has finished downloading, unzip it and you'll see the pidora.img file.



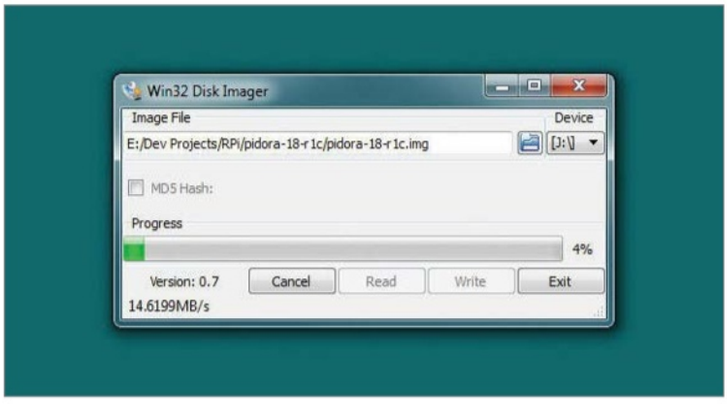
02 Win32 Disk Imager

If you haven't already, download a copy of Win32 Disk Imager from **http://sourceforge.net/projects/win32diskimager/**. Once it has finished downloading, install it on your Windows PC and open it up by right-clicking and selecting 'Run as administrator'.



03 Format card

Insert your SD card into your PC and make sure it's formatted to FAT32. This is done by clicking on My Computer and then right-clicking on the SD card icon, choosing Format and then FAT32 as the file system. Give the card a label and click Start.



04 Copy image

Once that's complete, open up Win32 Disk Imager and, from the Device menu, choose your SD card volume. Then, in the Image File section, choose your Pidora.img you extracted earlier. Click Write to start the process, then insert the card into the Pi once it's completed.

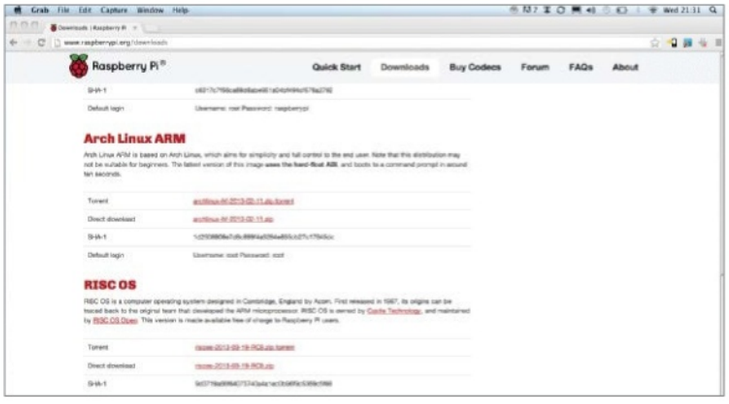
Install ArchLinux

Learn how to install the powerful but lightweight Linux distro ArchLinux on your Raspberry Pi using OSX

One of the must impressive aspects of the Raspberry Pi is that it's incredibly easy to turn you small, inexpensive Pi into a fully fledged desktop computer very easily. There are plenty of different ways in which you can achieve this, but one popular route is to install ArchLinux. ArchLinux is a complete but lightweight distribution that includes only the parts you really need. This helps keep it running smooth on the Pi's relatively modest hardware. Any extra components you then need can be added manually at a later date. Launched in 2002 the primary goal of the Arch development team was to focus on simplicity, in contrast to say the Ubuntu team who focus highly on usability and being fully featured from the initial installation. Its easy to install Arch from any operating system, and there are plenty of software tools to help you out in the process, but this guide will take you through the process of installing Arch using the command line in OSX.

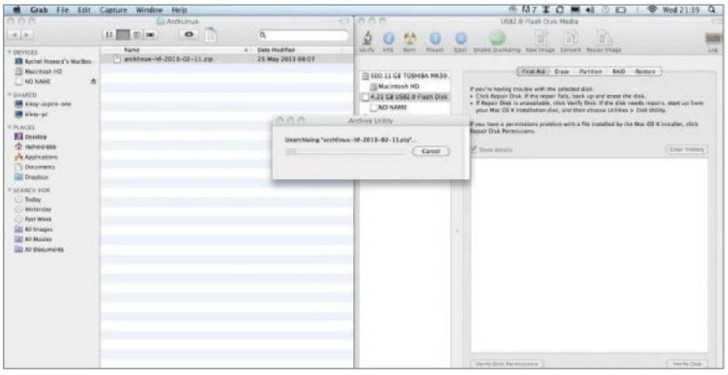
"ArchLinux is complete but lightweight, only including parts you really need"

■ The stripped back gui of ArchLinux allows for improved performance



01 Download the image

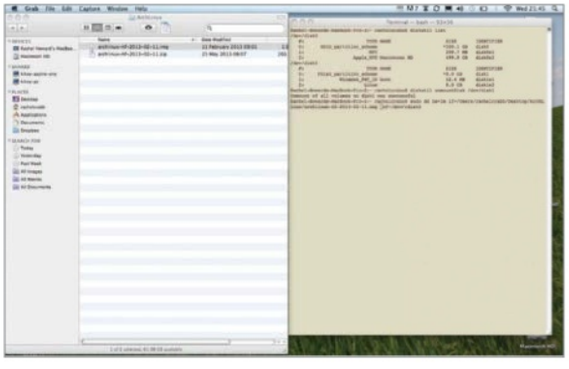
The first step is to download the latest ArchLinux operating system image from **http://www.raspberrypi.org/downloads**. The file should be around 200MB in size, so its advisable to have at least a 1GB Sdcard to boot from, but bigger is always better.



02 Unzip and format card

Next you need to format your Sdcard. It needs to be in the FAT32 format. This can be done using the Disk Utility application. After that, extract your ArchLinux from its zip archive by double clicking on it or entering unzip ~/Downloads/archlinux-hf-2013-02-11.zip from a new terminal window, making sure the date matches your download.

03 Copy image



Now in the terminal enter: diskutil list and make a note of the disk. This will be something like 'Disk1'. Then unmount the disk using:

```
001diskutil unmountDisk /dev/disk1
```

finally:

```
001sudo dd bs=1m if=archlinux-hf-2013-02-11.img of=/dev/disk1
```

04 Boot up Arch

At the command prompt, login with the username 'root' and the password 'root'. To get the Xorg graphical interface just type:

```
001'pacman -Syu'
002to update the package manager and then
003'pacman -S xorg-server xorg-xinit xorg-utils xorg-server-utils xorg-twm xorg-xclock xterm'
004 and then 'startx'
```

Install RISC OS from Linux

Get the classic Acorn OS on your Raspberry Pi for free

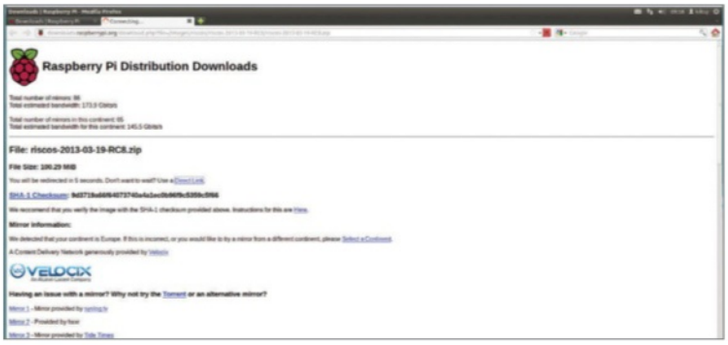
For those people who went to school during the early Nineties, you may have remembered using the Archimedes computers. Archimedes were produced by Acorn in the UK, who also produced the very popular BBC Micro in the Eighties. For the Archimedes, Acorn developed a brand new graphical user interface they called RISC OS. RISC OS was quite advanced for its time, sharing many elements with Apples OS and was arguably ahead of Microsoft's Windows effects.

The Archimedes used an ARM based CPU, which happens to be the same architecture as many modern smart phones and the Raspberry Pi. This makes the porting of RISC OS an obvious next step. RISC OS is interesting as unlike many other popular distros, its not Linux based, and is its own standalone entity.

This guide will run through the process of installing RISC OS onto your Pi from Linux but the process is very similar from all operating systems.

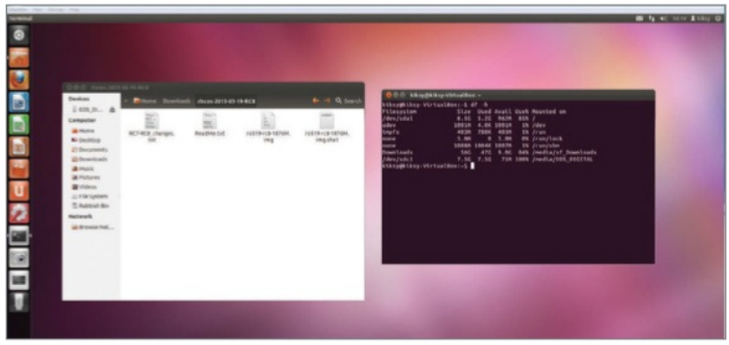
"RISC OS was quite advanced for its time, sharing many elements with Apple's OS"

■ While not at the cutting edge visually, RISC OS is snappy and has lots of great apps



01 Download RISC OS

Firstly you need to grab the latest RISC OS image from **http://downloads.raspberrypi.org/download.php?file=/images/riscos/riscos-2013-03-19-RC8/riscos-2013-03-19-RC8.zip** and then unzip it. This should give you a couple of readme txt files, and the main OS image file with an .img extension. Once you have this, you will now need to open up new terminal window.

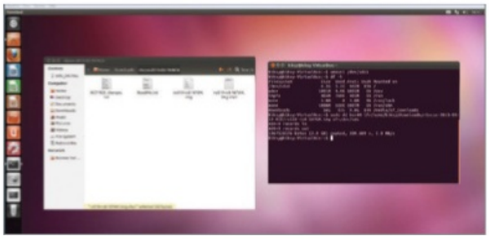


02 Get card info

Insert your SD card and type 'df -h ' into the terminal. This will list the current devices, which name, eg /dev/sdc1. Then unmount the card using:

```
zumount /dev/sdc1
, replacing 'sdc1' with which ever your device is called. Note that the number represents the partition.
```

03 Copy image



Once you've done this, the next step is to copy the image to the card. This is done using the command structure as below:

```
dd bs=4M if=[your RISOS img] of=/dev/[your device, excluding partition]
```

for instance:

```
dd bs=4M if=/home/kiksy/Downloads/riscos-2013-03-19-RC8/ro519-rc8-1876M.img of=/dev/sdc
if you get 'permission denied' , you'll need to add sudo to the start of the command eg.
Sudo dd bs=4M if=/home/kiksy
```

04 Boot into RISC OS

Once the image has finished copying, all you need to do now is insert the SD card into your Pi, plug it in making sure that you have a keyboard and mouse attached and wait for RISC OS to boot up. You'll notice the boot time is much quicker than many other distros.

The Debian desktop

Although the Raspberry Pi's operating system is closer to the Mac than Windows, it's the latter that the desktop most closely resembles

Icons

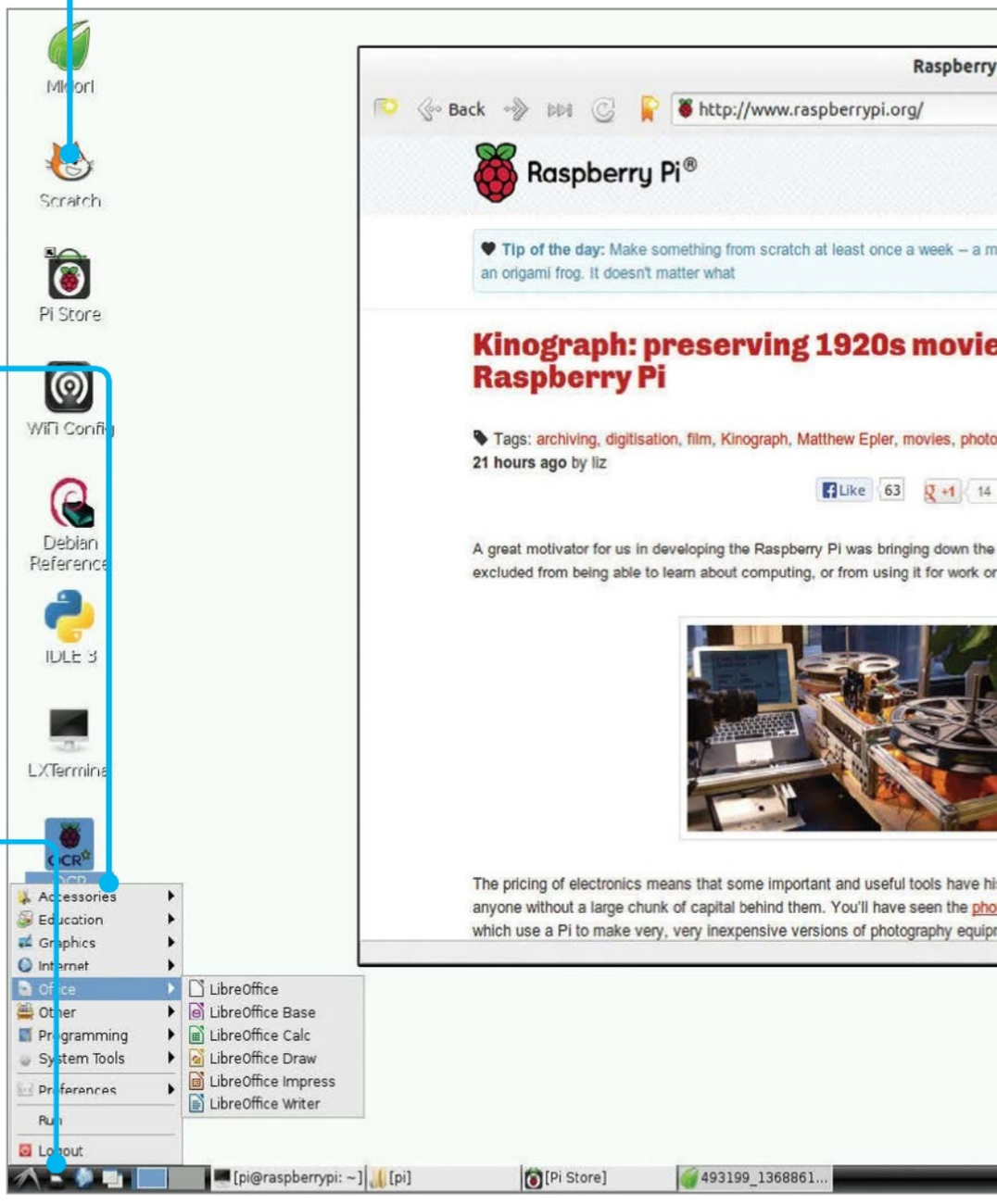
A number of the pre-installed apps have icons on the desktop. To run the associated app, simply double-click on the icon. The icons can be moved and dropped into other locations. Click, hold and drag to move them. There are icons for a variety of tasks – from apps such as Python and Scratch for programming, to Midori for web browsing, the Pi Store, LXTerminal for issuing commands to the system directly, to reference materials. Icons can be renamed or even deleted if you don't want them on the desktop any more. Right-click over any icon to see the options available for it.

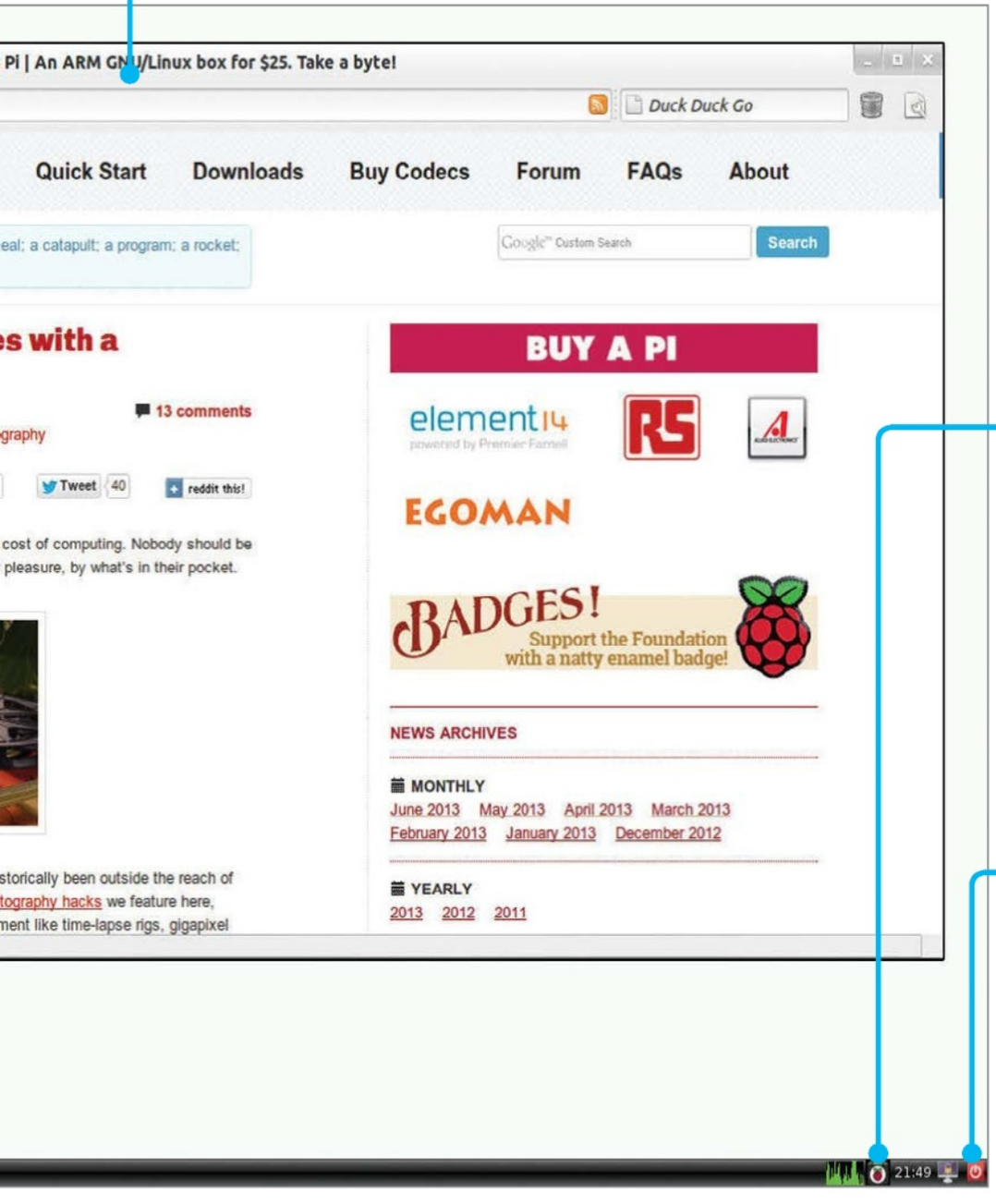
Menu bar

Just like Windows' Start menu, click in the corner to bring up a menu of programs and options. The main categories include accessories, apps for educational use, graphics, internet (including some alternative browsers), programming, a general assortment of apps and utilities, and the system tools. You will also find that some programs you download from the Pi Store (like LibreOffice) will appear here. There are also a set of preference options for configuring how certain elements of the system work, such as keyboard and mouse. The Run option is just like the one in Windows – it opens a command-line interpreter to run commands.

File manager

No computer would be complete without a file manager. Click here to bring it up. Files can be copied, renamed and deleted by dragging to the dustbin in the window. You can also create tabbed windows in the file browser or open further ones. Folder locations can be bookmarked for easy access and files themselves can be viewed as icons or in a detailed list. If you need to do any command-line work, the current graphical folder can also be opened in the terminal as the current folder there. Next to the file manager is a shortcut to Midori, an option to minimise all windows, and a dual desktop feature.





Web browser

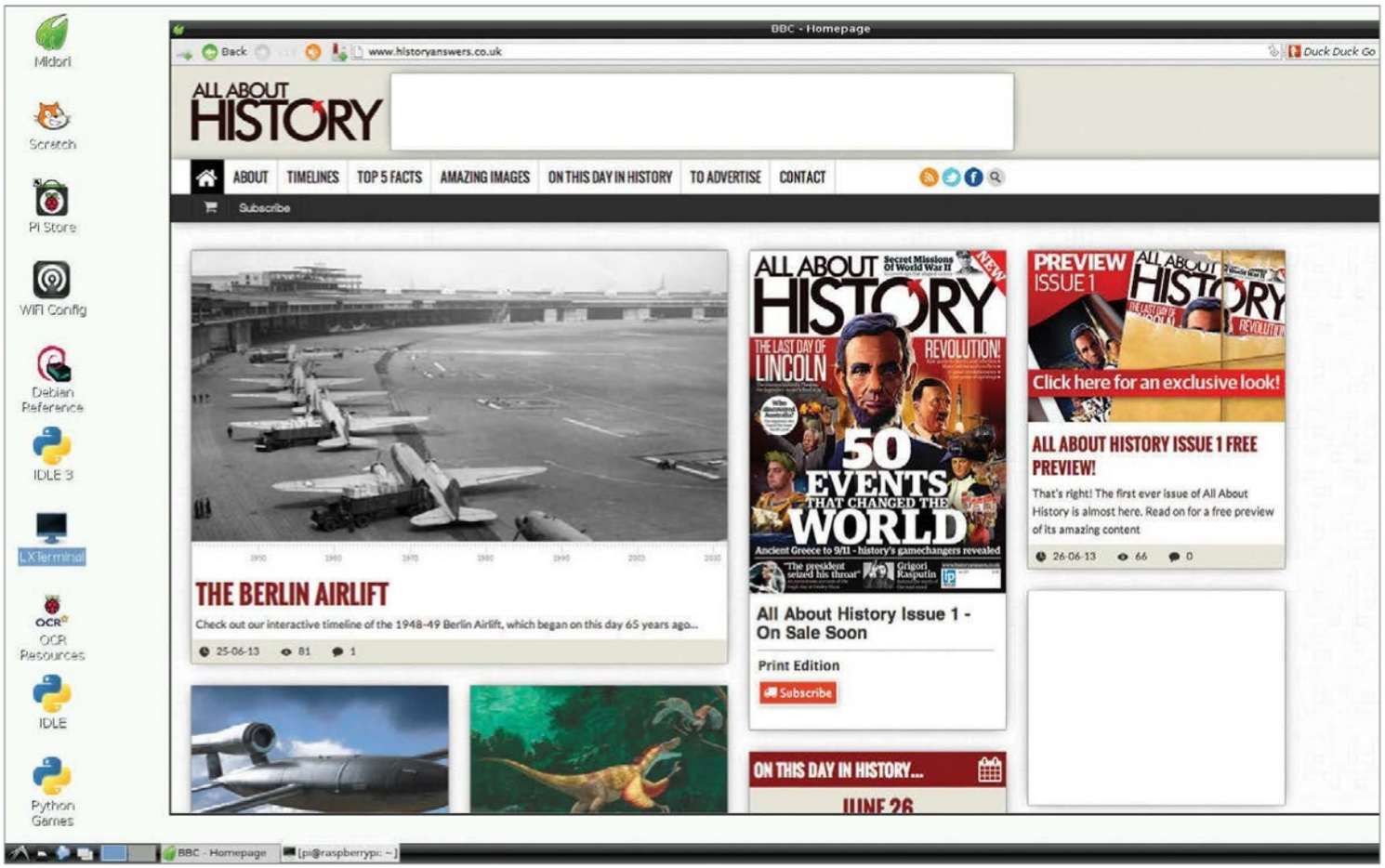
The Pi features a number of browsers – the default one being Midori. There's a desktop icon to launch it. The window can be resized by hovering the cursor over the sides or corners. Web addresses are typed into the long address bar whereas search queries are typed into the shorter bar with the words Duck Duck Go. There are standard navigation buttons for going back and forth through webpages. Midori supports multiple tabbed windows and features private browsing and the ability to clear browsing data. Click on the Options icon at the end of the tool bar to access these functions.

Pi Store

There's a desktop shortcut to the Pi Store and, once running, an icon appears here as well as a minimised window on the taskbar. You can access the Store again by clicking on it and also, messages from the Store will appear here. It works much like Steam, Google Play or the App Store. You need to sign up for an account with IndieCity, which can be done directly on the Store interface. For future visits you will need to log in. Applications downloaded and installed from the Store then appear under a tab called My Library. Some can only be run from here.

Power and time

Rather than just pulling the plug on the Pi when it is running the desktop, it is better to click here and select 'Logout of the desktop'. This closes any temporary files and leaves the desktop, dropping you back to the command line. You can then unplug the Pi. Next to the power symbol is a clock; click on this to reveal a calendar, highlighting the current date. If you right-click on the time, you get the option to go to the Digital Clock customisable settings or to remove it from the panel altogether.



Get your Pi online

To access a world of utilities, apps and resources you need to get online. This is how to do it

The easiest way to get online is to buy a Raspberry Pi Model B or Model B+, as both come with an RJ45 Ethernet socket. The Model A not only lacks the Ethernet port, but is handicapped by only having one USB port. That means you will have to buy a power USB hub in order to get online. Back to the Model B/B+ though and to get online, simply plug an Ethernet cable into the socket on the Pi and connect it to a similar port on the back of your internet modem/router. Turn your Pi on and launch the desktop, then double-click on Midori and you should see the internet appear (main image). To check that it's working, look at the lights on the Pi itself. The red power light should on. Above this is the green light that flickers when accessing the SD card. Below the power light are the three Ethernet-related lights. Note that the Model A does not have these LEDs because it doesn't have the Ethernet socket. The middle light is green and comes on when it detects a Full Duplex LAN connection. This means it is able to send and receive data to the internet. The next light is green and flashes when actually accessing the internet by sending or receiving data. The last light is yellow and will come on and stay on when a 100Mb LAN connection has been detected.

The Wi-Fi option

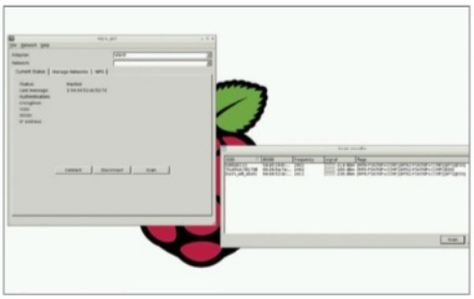
If you aren't close enough to the modem/router to be able to plug in the Ethernet connection, or you simply have a Model A, then a powered USB hub is required. This plugs into a USB port on the Pi. You can then plug a Wi-Fi dongle into this. Boot up the Pi and launch the desktop. Then double-click on the Wi-Fi Config icon. You should see a name for the dongle in the Adapter section. Click on Scan to look for networks and a list of those found should appear (Fig 1). Double-click on the one you want to connect to and the details for it will be listed. Almost all home networks use a network key, which is usually written on the modem itself. Click on PSK, which stands for pre-shared key, and type it in (Fig 2). Then click on Add. It will process this, then associate the connection and then finally, a new IP address for the Wi-Fi connection will appear. If you click on the Manage Networks tab, the network will now be listed and have an Enabled radio button active. To get on the internet, simply launch Midori and you'll be connected. The Wi-Fi utility will remain running on the bottom right of the panel. If you right-click on the Wi-Fi icon you will see options to Disconnect or Reconnect, event history and the results of the most recent scan. Click on Status to see how it's performing.

Checking the connection

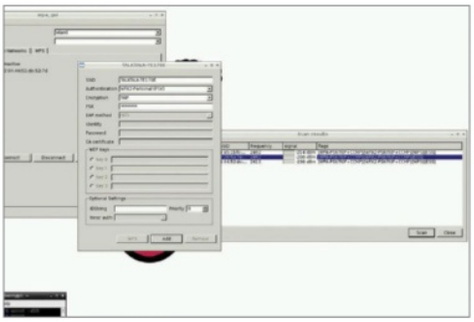
To check that the Pi has a valid internet connection, double-click on LXTerminal. Enter this command:

```
ip addr
```

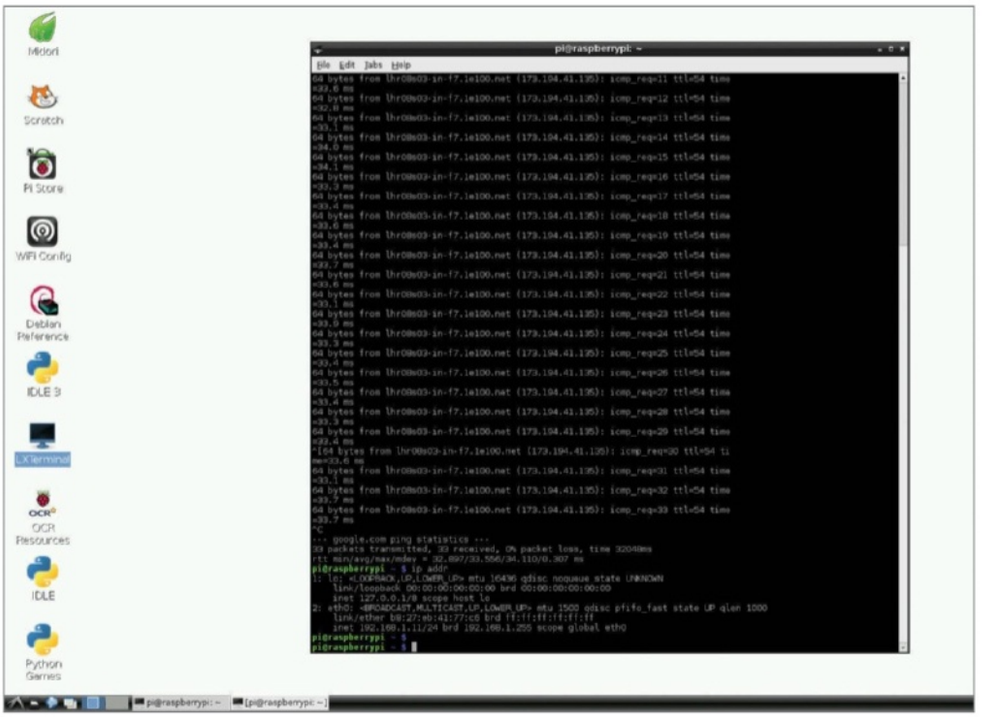
You should see a list of numbers, with the bottom line starting 'inet' and then the IP address of the Pi connection (Fig 3). Typically this is something like 192.168.1.11 and this shows that the connection is working because the Pi has been assigned an IP address based on the one used by your internet modem/router. If this doesn't come up then there



■ Fig 1: With a Wi-Fi dongle attached, run the utility and scan for available networks



■ Fig 2: Enter the pre-shared key in order to connect to your home router



■ Fig 3: If it doesn't look like the connection is working, there are some easy ways of checking

may be a problem at the router end. The modem/router should be running a DHCP server and when the Pi connects to it, it will be given the IP address. If it isn't running then nothing else connected to it will be able to access the internet either. Use the web interface with another device to log onto 192.168.1.0 or whatever is your modem's actual IP address in order to check that the DHCP server service is turned on. Finally, in the terminal, type:

```
ping google.com
```

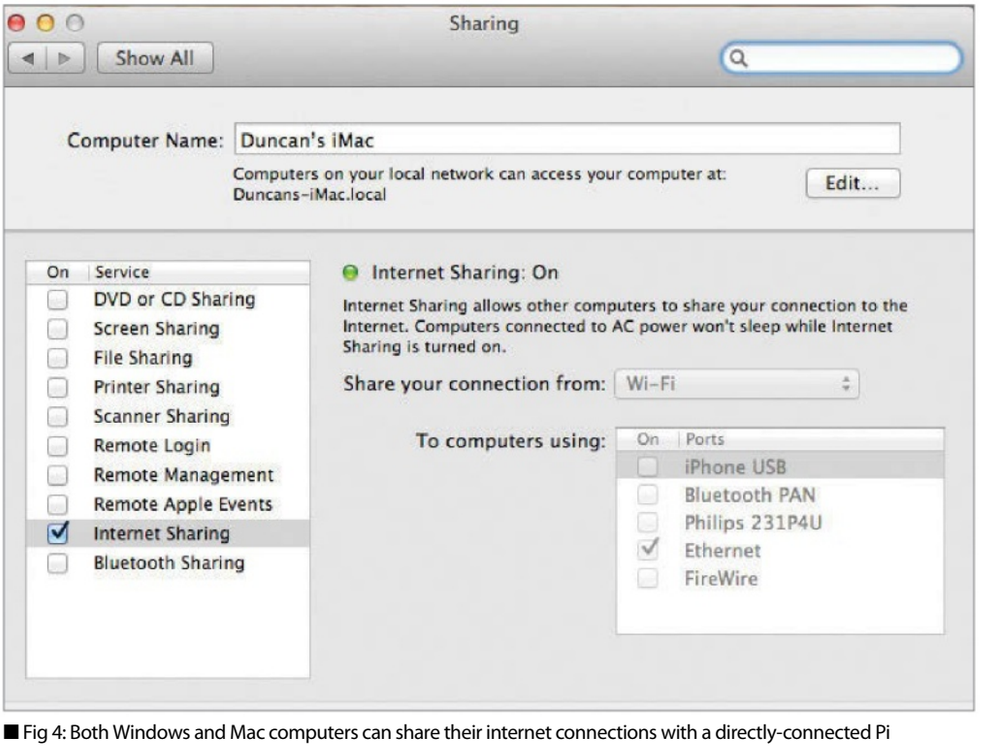
Give it a few seconds and press Ctrl-C to stop. You'll see that it has successfully sent and received a number of packets.

Sharing a connection

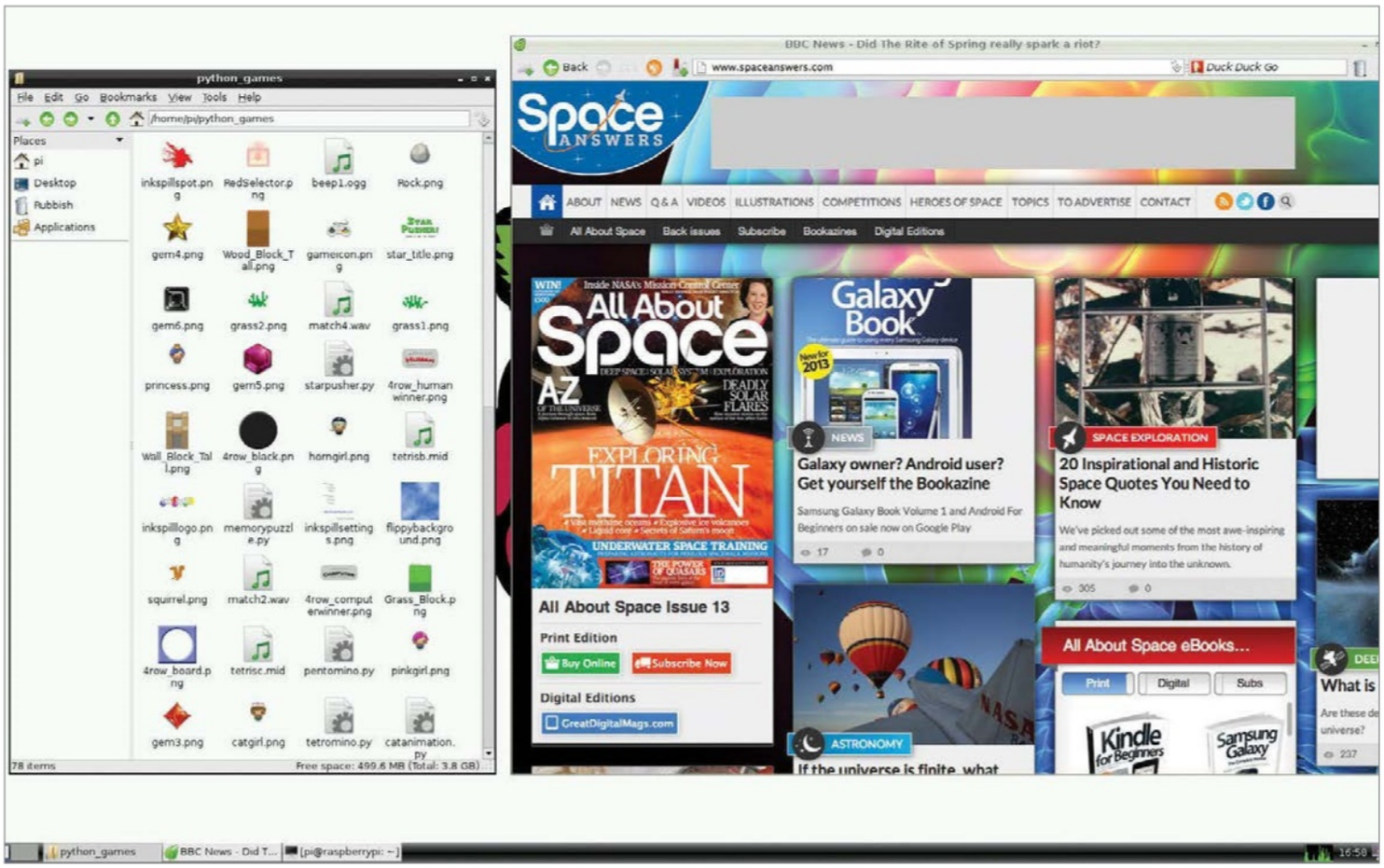
If you don't have a Wi-Fi dongle, a powered USB hub or a long enough Ethernet cable, but do have another computer connected to the internet, there's another way of getting access. On a Mac, connect it to the Pi via a USB or Ethernet cable. Launch System Preferences; under Internet & Wireless, click on Sharing. Click on Internet Sharing, then select Wi-Fi (or AirPort) as the connection type to share, and select how the Pi is connected to your Mac (Fig 4). Close it and turn on your Pi. It'll now connect to the internet via the interface with the Mac.

On a PC, go to Windows Explorer>Networking>Networking and Sharing Center>Change Adapter Settings. Select the Wireless hotspot and the Ethernet connections by holding down Ctrl and clicking on each in turn. Right-click on these and select Bridge Connections. Plug in your Ethernet cable from the PC to the Pi, then turn the Pi on and it will use the Wi-Fi via the Ethernet connection.

"The Raspberry Pi Model B and B+ come with a RJ45 Ethernet socket"



■ Fig 4: Both Windows and Mac computers can share their internet connections with a directly-connected Pi



Start using Raspbian apps

It takes software to make a system usable. Here's your guide to the apps in the Pi's Raspbian distro

Getting online is very handy for downloading new software, tutorials and resources, but you're also going to want to browse the internet at some point as well. The standard browser for this is Midori and there's an icon on the desktop to enable you to run it – just double-click it. You can also run Midori by going to the menu under Internet, where there's an option to run Midori in Private Browsing mode. There are also the NetSurf and Dillo browsers as alternatives. The latter is very quick because it doesn't load images by default, but then it doesn't render pages properly either. One of the best options for web browsing isn't actually installed at the time of writing, but it is available. Open the LXTerminal, or at the command line before starting the desktop and type:

```
sudo apt-get install iceweasel
```

Iceweasel is the Debian version of Firefox. Neither Midori or Iceweasel play Flash video by default, though you can install it. Open a terminal and type:

```
sudo apt-get install gnash
sudo apt-get install browser-plugin-gnash
```

Close the terminal and restart the browser and then see how you get on. There are other ways of watching YouTube and Flash videos, but they are a little more technical. One thing you will notice is that rendering webpages is fairly slow and videos crawl because the CPU does all the rendering for the desktop. There are moves to have the GPU do this, but programming isn't expected to be completed until later this year.

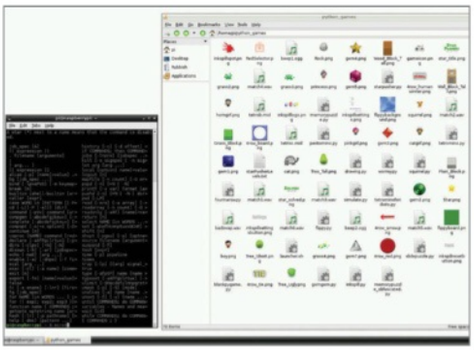
Programming the Pi

There are two main programming platforms supplied with the Pi and they are Scratch and Python. Both have shortcuts on the desktop. There's also a link to Squeak (a version of Smalltalk) on the menu system, but this doesn't get any further than looking for a Squeak image because half the files necessary for it to work are missing. You can download them, though.

Scratch is aimed at getting children interested in programming. It uses a tile-based system of commands that can be strung together without having to worry about program syntax, and BBC Micro BASIC arrived recently as well. Python is the main language, though, and is a high-level interpreted language where each line of code is read and implemented by the Python host application itself. Due to its nature, Python is often used as a scripting language in 3D applications. There are two versions supported: the latest v3.2 and an older v2.7 because there are more resources for that and the community around it is still active. Launching either version of Python from the menu or shortcuts activates its integrated development environment (IDE), otherwise listed as IDLE, or IDLE 3 in the case of the latest version. Programs can then be created or loaded and run, examined, altered or debugged. There's a selection of Python games installed and the code for these can be loaded to see how they work, to amend or improve them yourself.

The accessory programs

There are a number of accessory apps that come



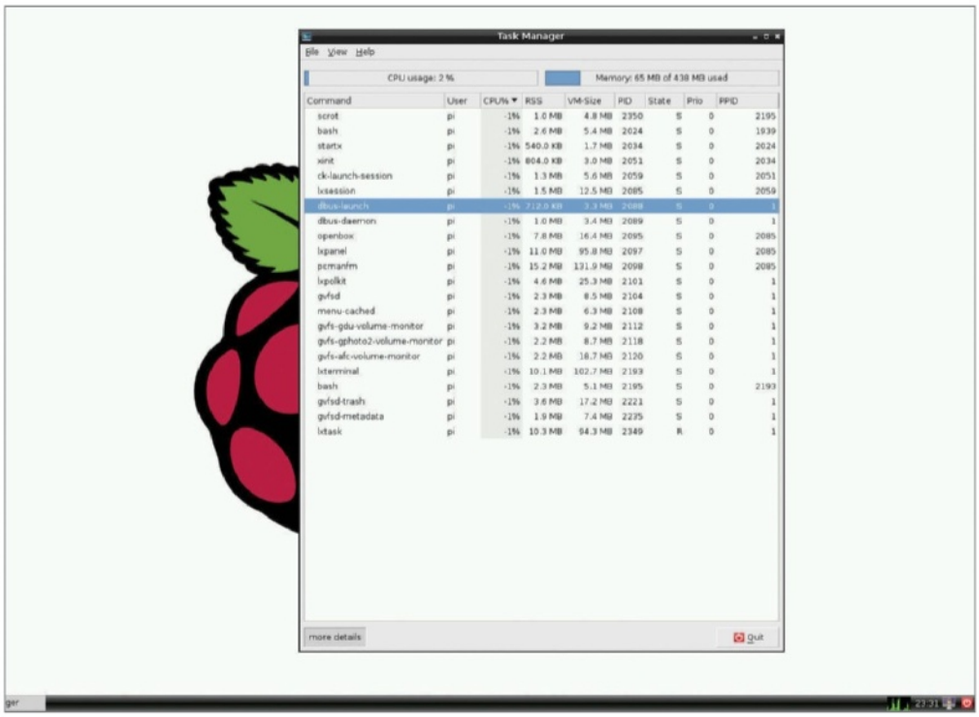
■ Fig 1: The file manager is the essential method of navigating your way around the filing system while inside the desktop environment

pre-installed, including an image viewer, calculator, notepad, archiver, file manager and a couple of versions of the terminal. Look to the main menu under Accessories to see them listed. There's also a reference manual for Debian there. The Image Viewer is a basic viewing program that automatically runs when you double-click on a graphic file. Images can be rotated and saved. The two most useful apps are the file manager (Fig 1) and the LXTerminal. The former gives a familiar windows file-access structure with shortcuts and icons. The view can also be changed to thumbnail, compact or detailed list views. Favourite locations can be saved as bookmarks for quick access later. Interestingly, because you need to use command-line instructions a lot more than when using the Windows or Mac OS X operating systems, you can open a current location with the terminal window, making it easier to find precise locations. The LXTerminal and Root Terminal allow commands to be sent directly to the system from within the desktop environment.

Useful software

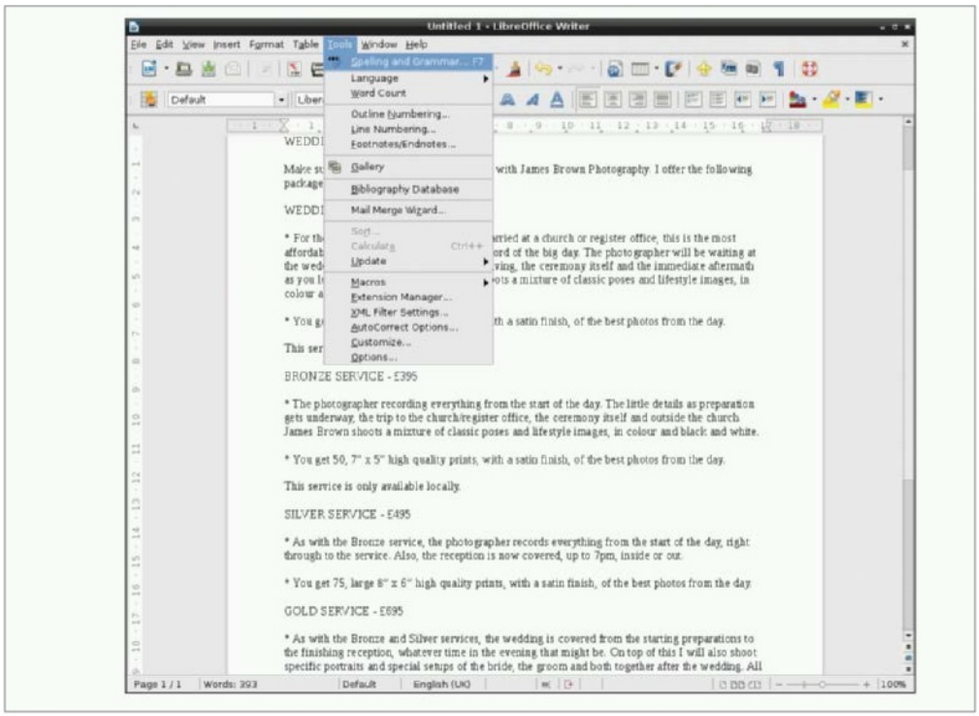
There are a brace of other handy accessories, listed under Other on the menu. They are a PDF viewer and the Task Manager (Fig 2). This displays commands and CPU percentages being used, as well as memory overheads. The view can be trimmed down by showing user, root or other tasks. To stop a task or command from running, left-click to select it, then right-click for the menu. From here it can be stopped, removed or given a high or lower priority.

For yet more useful software you need to go to the Pi Store. Here you can find resources for programming and also the free LibreOffice package. This includes a word processor, presentation software, spreadsheet, flowchart and database. It's the LibreOffice Writer (Fig 3) app that will probably be most useful as it offers Microsoft Word file compatibility as well as a host of familiar features like styles, formatting, image wrapping, bullet points, spell-checking and a word count. There's more available on the Pi Store, though, so it's worth having a look to see what you can find.

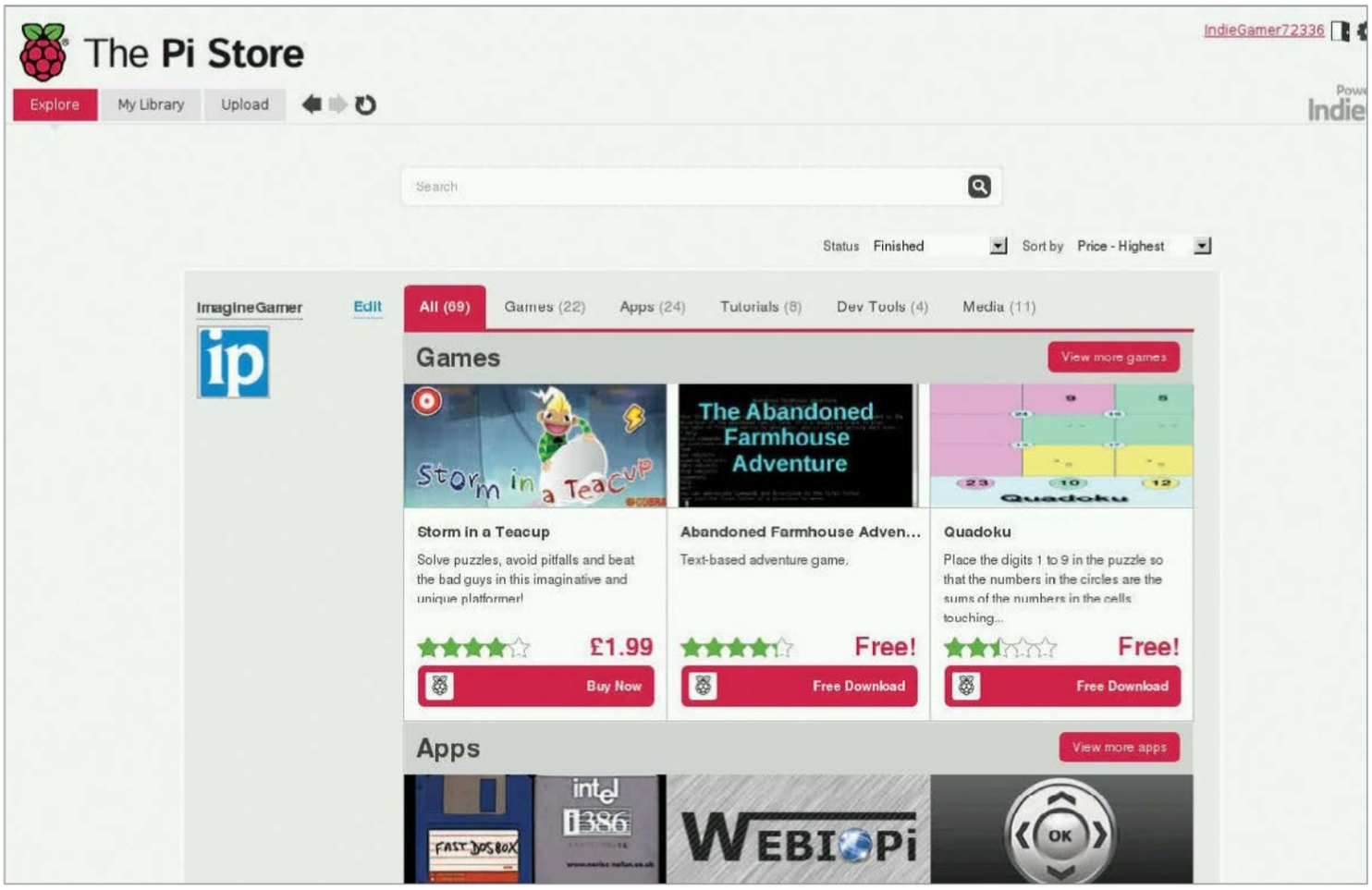


■ Fig 2: The Task Manager is like Windows. It shows you how much CPU load and memory a process is using

"Launching either version of Python from the menu or shortcuts activates its integrated development environment"



■ Fig 3: Make LibreOffice your first download from the Pi Store. It is loaded with features and MS Office compatibility



The Raspberry Pi Store

There's a mixture of free and paid-for games and applications in the Pi Store

Launched in 2012, the Pi Store aims to provide a hub for both professional and amateur developers who want to make their software available for the Pi. The service was created by the Raspberry Pi Foundation in co-operation with companies IndieCity and Velocix.

It's possible to browse the Store and log in as a customer using a web browser or the dedicated Pi application. So far, most of the items on the Pi Store are free or low priced, and it's clear that the developers of the site hope to build a thriving community. In particular, they have emphasised that they would like to see developers of all age groups. This approach ties in with the whole strategy of using the Pi to encourage interest in software development. In addition, The Pi Store is possibly the easiest way for beginners to expand their Pi.

As well as being a platform for browsing, installing and buying applications, the Pi Store has integrated features for developers to create a profile and upload projects. Customers can also create a profile and participate in the social side of things.

Browsing the Store

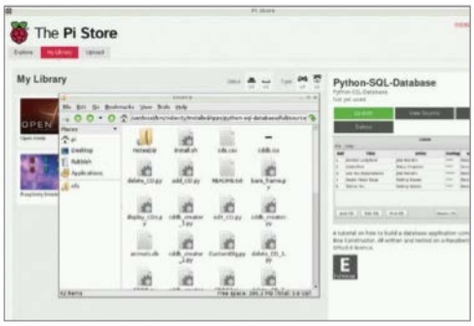
The store can be browsed via the web (**http://store.raspberrypi.com**) or by using the dedicated Pi app. To install the app, type `sudo apt-get install pistore`. The Store should then be visible as a clickable icon on the backdrop.

Upon launching, you are presented with a two-pane window with tabs along the top and a main area below (main image). The first tab is Explore, and this is where you go to find new applications.

The main window itself makes use of more tabs for the content categories. The Apps and Games categories contain the meat of what people expect from a device-specific app store. Each of these items, like all store items, features customer feedback in the form of a rating out of five and a comments section. Many of the products are versions of existing programs that have been repackaged specifically for the Pi. The Dev Tools are resources and tools to help developers, such as ready-made sound and graphics packs.

The Tutorials are resource packs designed to educate you on various aspects of the Pi. The power of this feature is that these packages aren't limited to flat text files and may contain other resources. For example, one tutorial demonstrates how to build a database program using the Python programming

> "The Pi Store is possibly the easiest way for beginners to expand their Pi"



■ Fig 1: The tutorial is a working piece of software to learn from along with documentation

language along with SQL, a language for accessing databases. The tutorial package includes the finished program along with the source code and documentation explaining how it works (Fig 1).

My Library

You can see what apps you have purchased and installed by clicking on the My Library tab. From here you can launch apps that you have installed, delete them from your system or reinstall them. This last feature is handy because it means that you can wipe and reinstall your Raspberry Pi without permanently losing the content you have obtained from the Store. This section also includes a running count, in icon form, of the applications that you currently have installed (Fig 2).

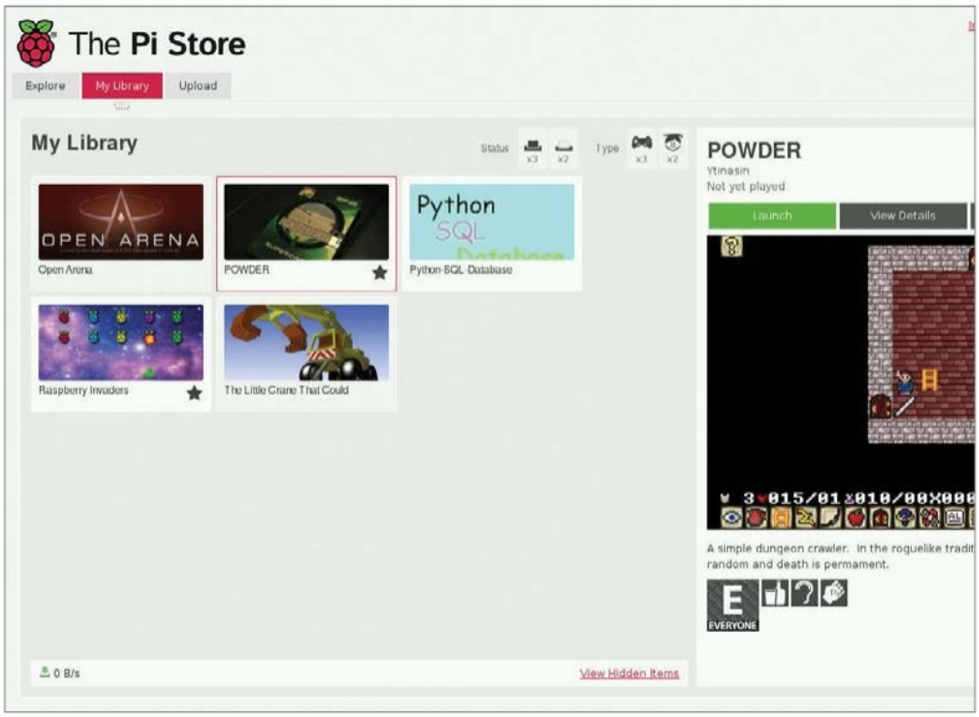
If you click on your username, you can edit your profile (Fig 3). You can change your username from the one that the store assigns you when you first sign up and add a link to your website and a photo. Doing this helps you to establish a presence within the community when you're giving feedback.

Note that when the Store app is running, it creates a status icon on the control panel at the bottom of the screen. This is particularly useful as some of the apps take a long time to install.

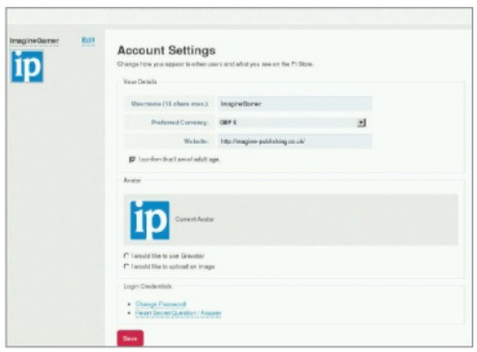
Uploading content

Clicking on the Upload tab allows you to create a developer profile (Fig 4). This gives you a comprehensive homepage on the site that you can customise. You can choose the colour scheme and banner image, and you can connect it to your Facebook and Twitter streams. Note that once you have created a developer page and saved it, you cannot change the name in the future.

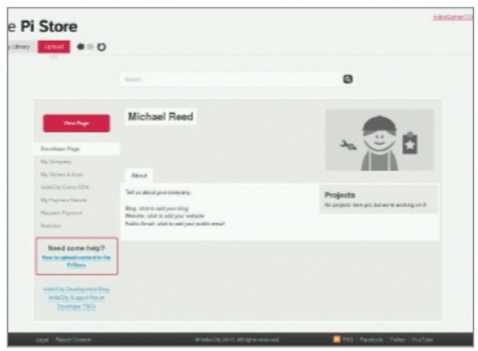
From here, you can also add a product. When you add a product, you can specify that it will be publicly viewable or hidden and that it is finished or a work-in-progress. You can also specify the category of your project (game, application, tutorial, dev tools or media), add screenshots, box art, age rating, tags, description and prices. Once the project has gone public, you can view statistics relating to number of page views and downloads. You can also specify your preferred licence for the product. For example, you may opt to allow other people permission to remix what you have created.



■ Fig 2: This section involves a list of all the applications that you have installed. From here you can also launch installed apps or choose to delete them



■ Fig 3: Customise your profile with a recognisable username and a picture



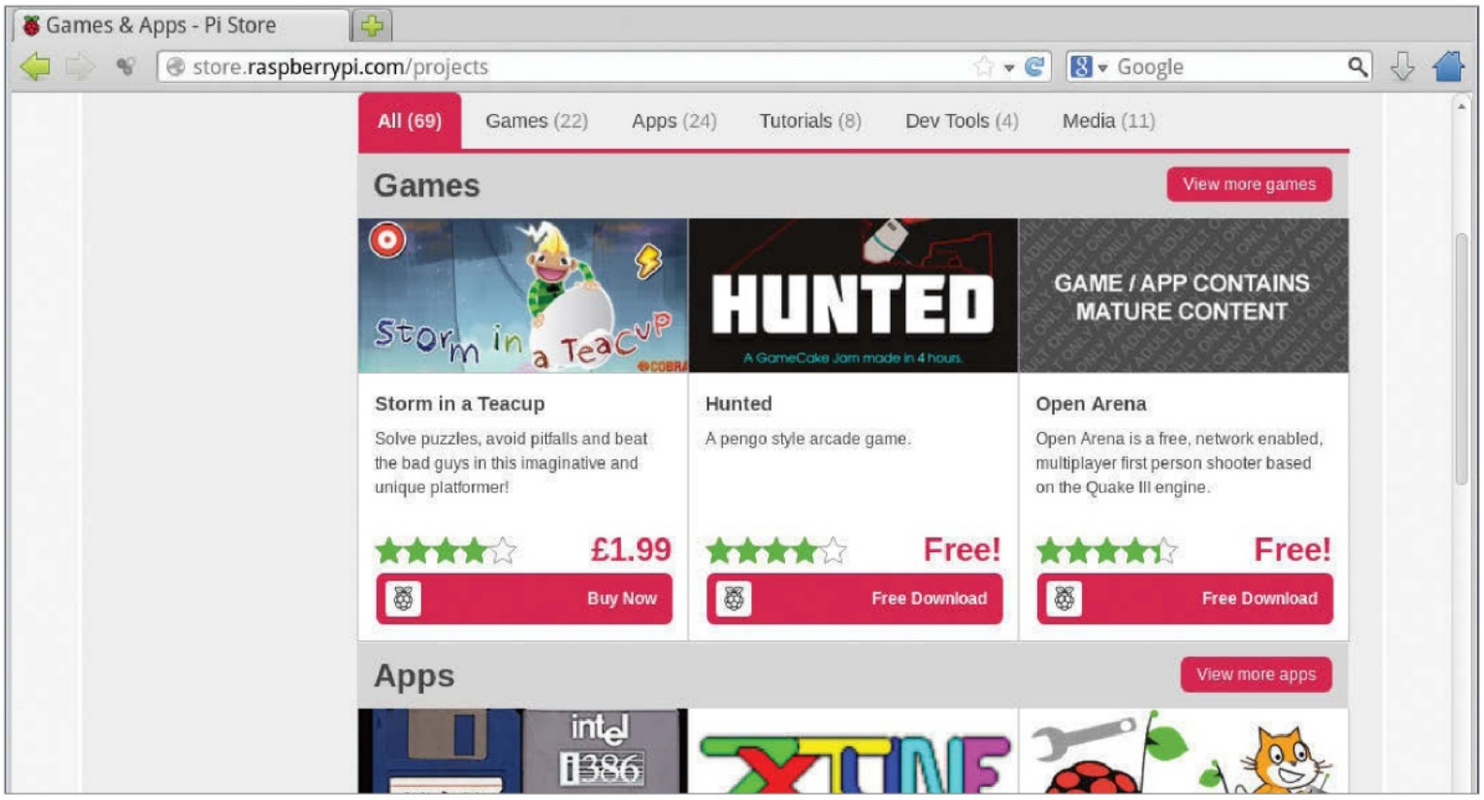
■ Fig 4: Click on the Upload tab to begin editing your developer profile

The files themselves

Time to get our hands dirty

Things run smoothest if you don't directly manipulate the files themselves, but it's handy knowing where they are if something goes wrong. Files relating to your profile and configuration are not designed to be edited by hand, but they can be found in [your home directory]/indiecity. A useful file for troubleshooting is [your home directory]/indiecity/pistore/Scripts.log. This file logs all store activity and can be an invaluable source of information when troubleshooting. Right-click on this file in order to open it in Leafpad. The files that make up the applications

themselves can be stored in various locations on your SD card, but many of the biggest game and application-related files end up in /usr/local/bin/indiecity/.



The best Raspberry Pi apps

We take a look at some apps that exemplify the best of what the system has to offer

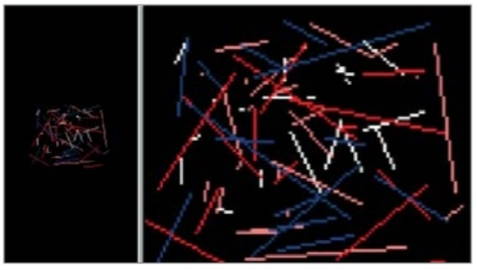
The Pi Store has been around for a while now, and it's still one of your main sources for getting applications on the Raspberry Pi. Most of it has been ported over from Linux workstations, and this has given the Pi a handy lead for a new platform. It's an encouraging sign that the developer community is already becoming established enough to support the Pi in this way.

The other, equally important, half of the equation – the user community – is similarly becoming established around the Store. By now, nearly all of the products have had some user feedback in the form of comments and star ratings. In fact, if you're not able to create your own software to place on the Pi Store, trying the software out and giving feedback may be one of the best ways of contributing to the software ecosystem. As one of the developers put it in the initial announcement for the launch of the Store: "If you rate and review constructively, it means the really great content that gets submitted will percolate up to the top, where everyone can see it."

Here we present a cross-section of the best of what the Pi Store has to offer.

LibreOffice

The Raspberry Pi can be a desktop computer replacement if you use it correctly, and LibreOffice can help the Pi fulfil this task. With full compatibility with Microsoft Office, it's the best suite of Office applications that isn't made by the Windows folks.

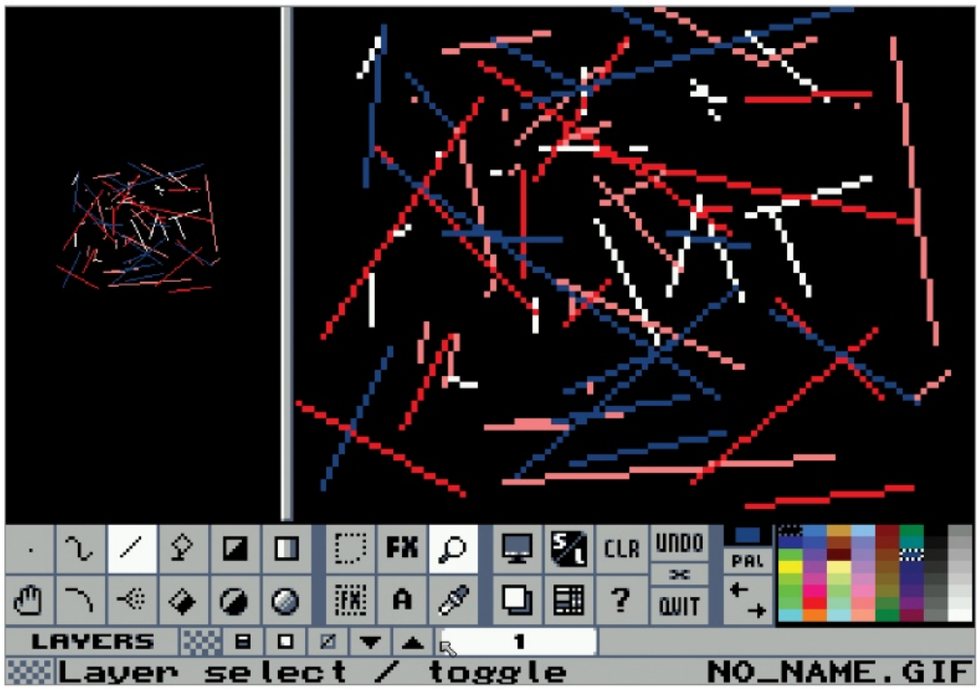


GrafX2

GrafX2 takes its inspiration from an earlier generation of 2D pixel art packages that ran on computers such as the Commodore Amiga. This means it has everything you need to get started within an intuitive layout, and it aims to become a favourite among indie game developers.

VirtualHere

VirtualHere allows you to share USB devices across a network. It takes the form of server which runs on the Pi and a client device driver that runs on a Microsoft Windows PC. It's free when used to share a single device, but the version that can share many devices costs money.



■ GrafX2 is an art package that harks back to an earlier era, combining immediacy with some powerful features

Pi gaming

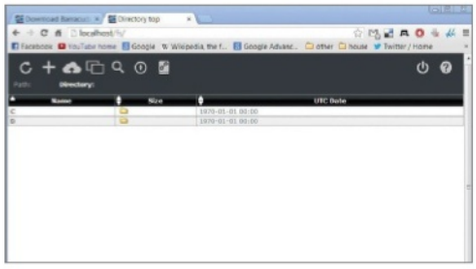
Discover a world of indie Pi games only on the Pi Store

The Pi Store isn't just home to apps and tools to help you use your Raspberry Pi. There's also a selection of home-grown video games to get your teeth into, that are developed by the Raspberry Pi community.

The Games tab is where you'll find all these titles, including a ported version of FPS classic Quake 3 Arena that works at full power on the Raspberry Pi.

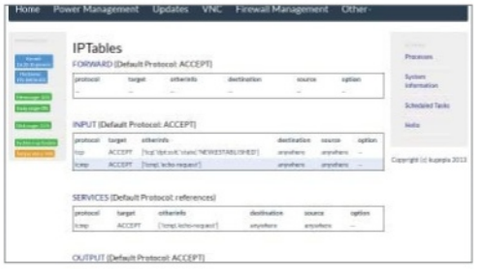
A lot of the other titles are made by children and teenagers that use the Raspberry Pi to learn to code, from dungeon crawlers to text adventures and even complicated RPGs.

You will also find some emulators on here, although our retro-gaming tutorial might be a little bit better for playing old games.



BarracudaDrive

This software allows you to turn your Pi into a cloud-based storage server, so you can access your files and documents from wherever you are. It's also highly standards compliant, and this means that you should be able to access files from most platforms - including mobile devices.



Pi-Web-Agent

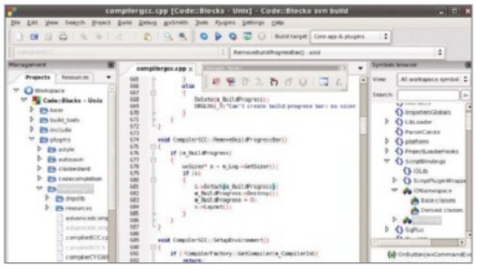
Controlling your Raspberry Pi from a web browser sounds like the stuff of science fiction, however it has been made real by Pi-Web-Agent. It allows you to control some of the functions of the Raspberry Pi that would otherwise need the command line, and helps monitor what it's doing.

Asterisk

Love them or hate them, voicemail systems are an essential function for some businesses. This repackaged version of Asterisk provides voicemail, VoIP (voice over Internet Protocol) and a whole lot more. It could add up to an interesting project when coupled with a Raspberry Pi.

NetPy

NetPy allows you to create websites by using drag and drop – and it outputs HTML, the basic language of simple websites. The bonus is that this means that although it is easy to use, by looking at its output you can learn HTML – a very handy skill to have.

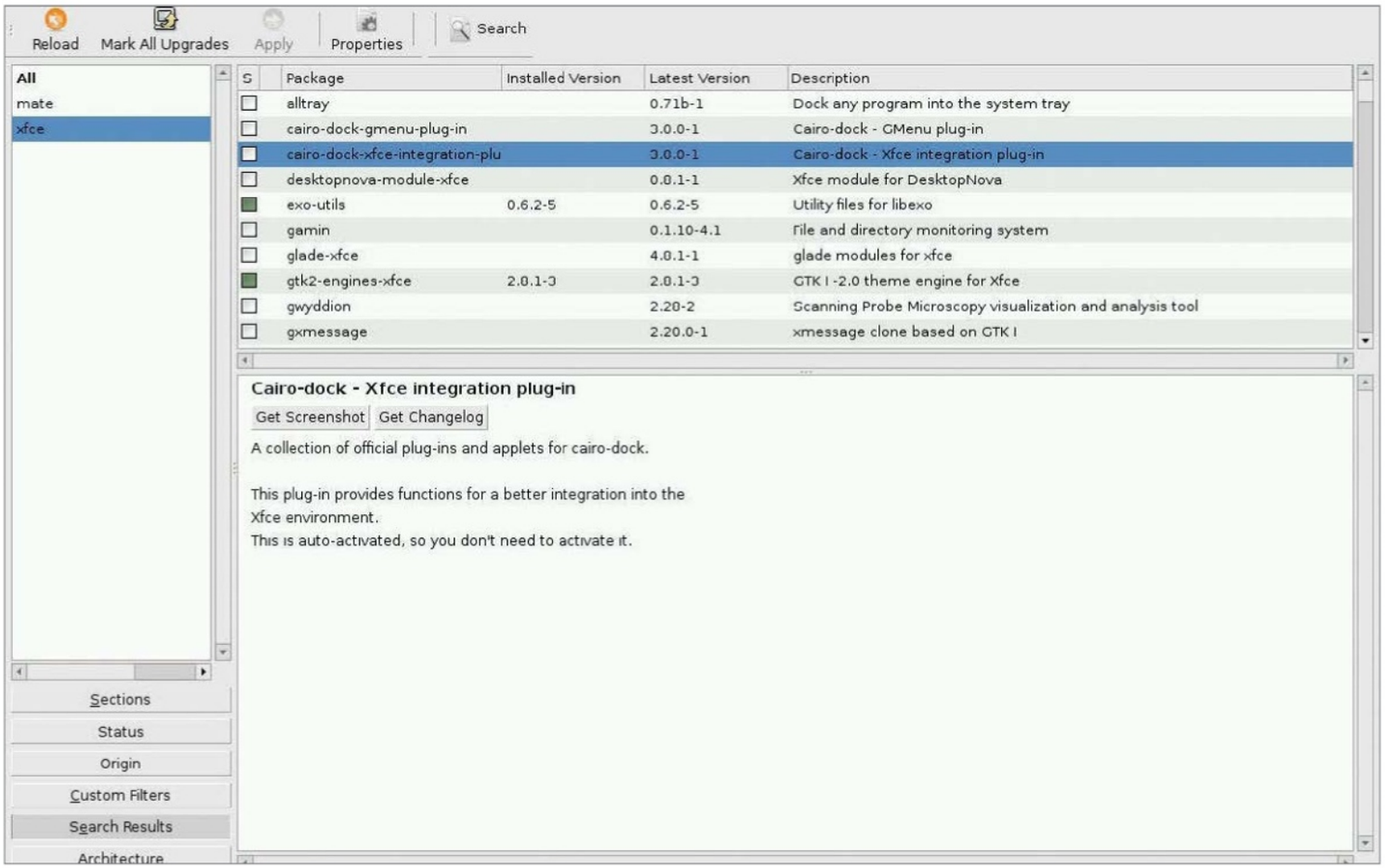


Code::Blocks IDE

If you plan to start doing some very serious programming or developing, then Code::Blocks is one of the best tools to help you along the way. It makes code easier to read, gives suggestions on what you can do, and you can modify it with special extensions.

Schism Tracker

This music sequencer package is inspired by 'tracker'-style programs that began to appear in the mid-1980s. It's an ideal system for composing music on a Qwerty keyboard and there are thousands of MOD format tunes that can be freely downloaded to learn from.



Use the Raspbian repositories

The Raspbian repository contains over 35,000 software packages that you can use to extend your Pi

The Raspbian repository contains over 35,000 freely available software packages. These range from small software components to full applications. In fact, the entire Raspbian operating system itself is made out of these packages.

To make this clearer, let's try an example. You've probably tried Midori, the open source web browser that comes with Raspbian. Although we may think of Midori as a single application, it is actually made up of a collection of smaller components. If one package requires other packages to work, the latter are called 'dependencies'.

You can list the dependencies of Midori by typing `apt-cache depends midori` into the terminal (Accessories>LXTerminal from the program launcher). As you can see from the output (Fig 1), Midori depends on nearly 20 other packages to work. However, don't worry about this as the Raspbian operating system takes care of downloading and updating dependencies for you. For example, if you didn't have Midori installed on your Pi, typing `sudo apt-get install midori` would download and install the latest version of Midori and all of its dependencies. Libxml2, a library of facilities for working with XML files, is one of the packages that Midori depends upon. Like all packages, it's actually a DEB file stored on the Raspberry repository.

Back to the roots

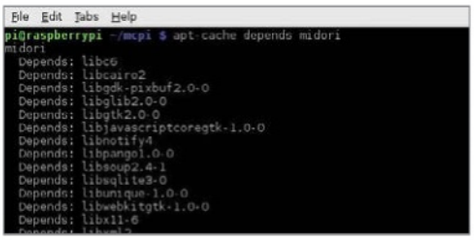
Linux is an operating system kernel – the 'hub' of the system – that was created in 1991 by Linus Torvalds, a Finnish computer science student. Linux powers servers that run a large portion of the internet as well as desktop computers and laptops, and it is also at the heart of Android-powered smartphones, in addition to the Raspberry Pi. A collection of packages in combination with the Linux kernel is called a distribution, and Raspbian is a Linux distribution derived from another distribution called Debian. Due to the open source nature of Debian, every time a software expert makes an improvement to Debian, that improvement is ported over to Raspbian.

The Raspbian repository consists of a set of DEB files that are stored on a server. You can browse the actual files that make up the repository by pointing a browser at **http://archive.raspbian.org/raspbian/**. The pool/main/m/midori subdirectory (Fig 2) contains a number of support files including the DEB package itself and a DSC file which contains a description of the package.

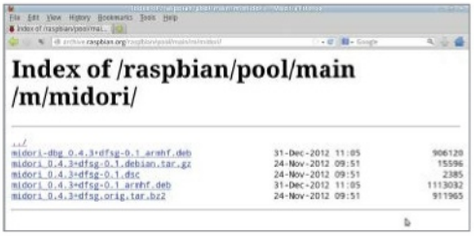
If you were to type `sudo apt-get install libxml2`, the Advanced Packaging Tool (Apt) connects to the repository, downloads the corresponding DEB file, unpacks and installs the files and updates the package database on your Pi. This means that when a new version of Libxml comes out, typing `sudo apt-get upgrade` can update it automatically.

Finding the packages

Installing packages from the command line is an efficient way of working, but using a front-end with a graphical environment is the easiest way to explore the Raspbian repository. There are a few choices, but we're going to work with a package



■ Fig 1: These are the dependencies of the package that provides the Midori web browser



■ Fig 2: Browsing the Raspbian repository

called Synaptic. Type `sudo apt-get install synaptic` to install it. Once the process is complete, launch Synaptic by typing `sudo synaptic`.

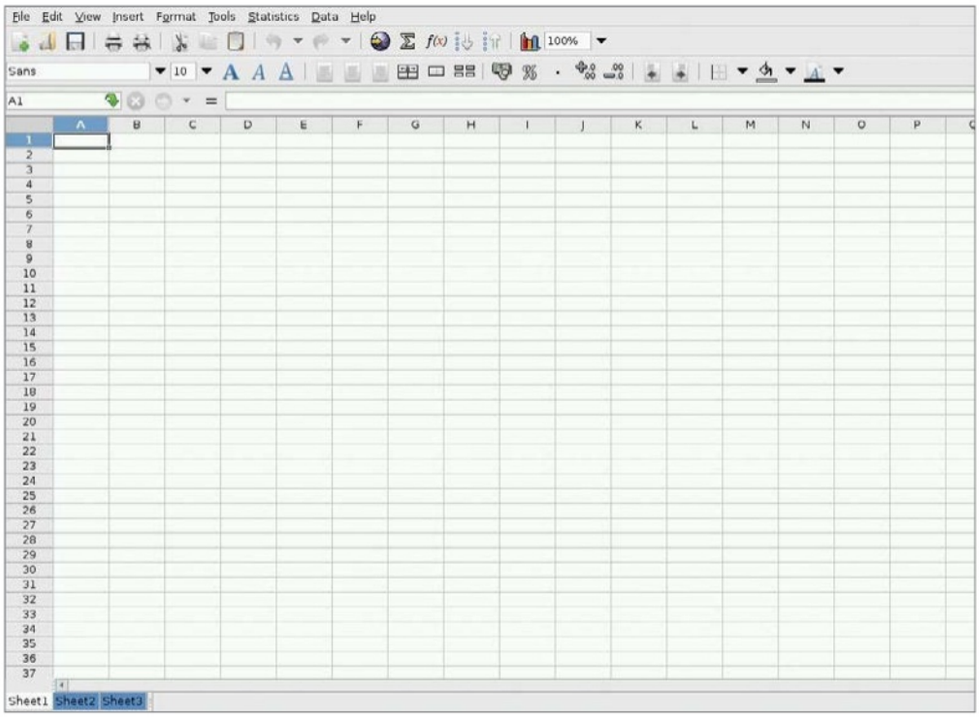
From the main image you can see that Synaptic employs a three-pane layout. The top section shows a list of packages, the sidebar has a list of categories and the main window shows a description of the currently selected package. Click on the Search icon at the top of the window and type 'word processor' into the search box. The first result of the search is AbiWord. Click on the title 'abiword' rather than the checkbox next to it and you will be given a description of the package. Note that if you do prefer the command line, you can always use `apt-cache search [search term]` to search through package names and descriptions.

Choosing packages

The Raspberry Pi is relatively powerful for such a small computer, but it doesn't have the memory, storage or processor power of a desktop PC. For this reason, it's often a good idea to choose lightweight applications wherever possible. For example, when it comes to office applications, although LibreOffice is the most fully featured Linux office suite, consider the aforementioned AbiWord as a word processor along with Gnumeric (Fig 3) as a very powerful spreadsheet application.

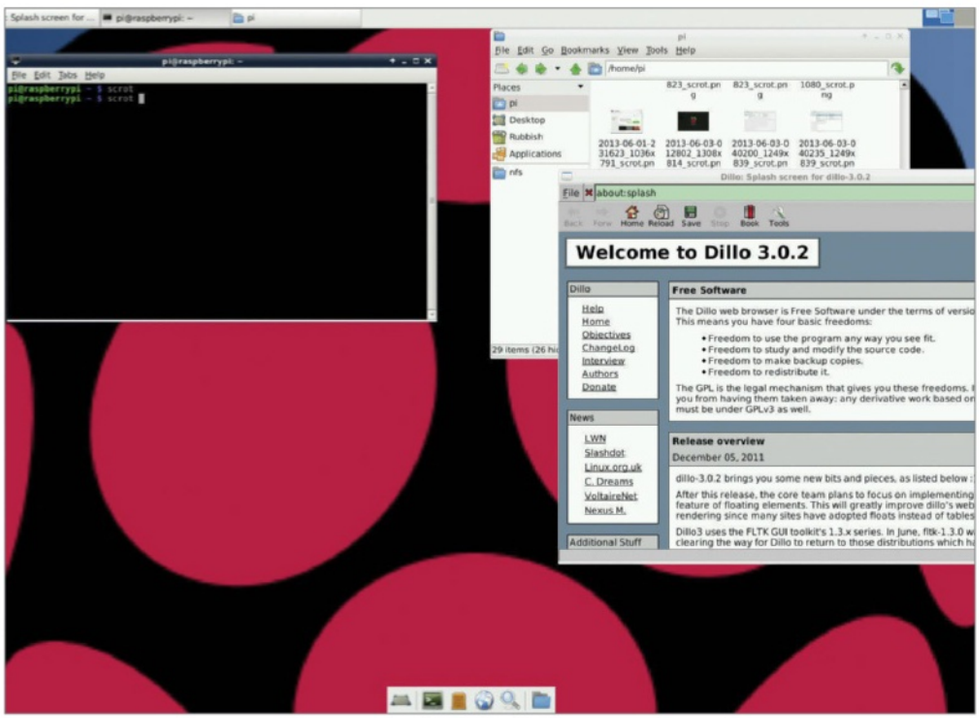
Note that the repository doesn't just contain applications and utilities. The package Xfce4 is an alternative window manager (front-end) which is efficient and fast enough for the Pi and yet more configurable than the default choice of LXDE (Fig 4). At the same time, install SLiM so that you can choose between front-ends at startup.

Debian-derived distributions like Raspbian are phenomenally flexible. Whether you need a web server like Apache, which runs over 60 per cent of all the websites, or a word processor such as AbiWord, you're more likely to be overwhelmed by choice than stuck for something to install.

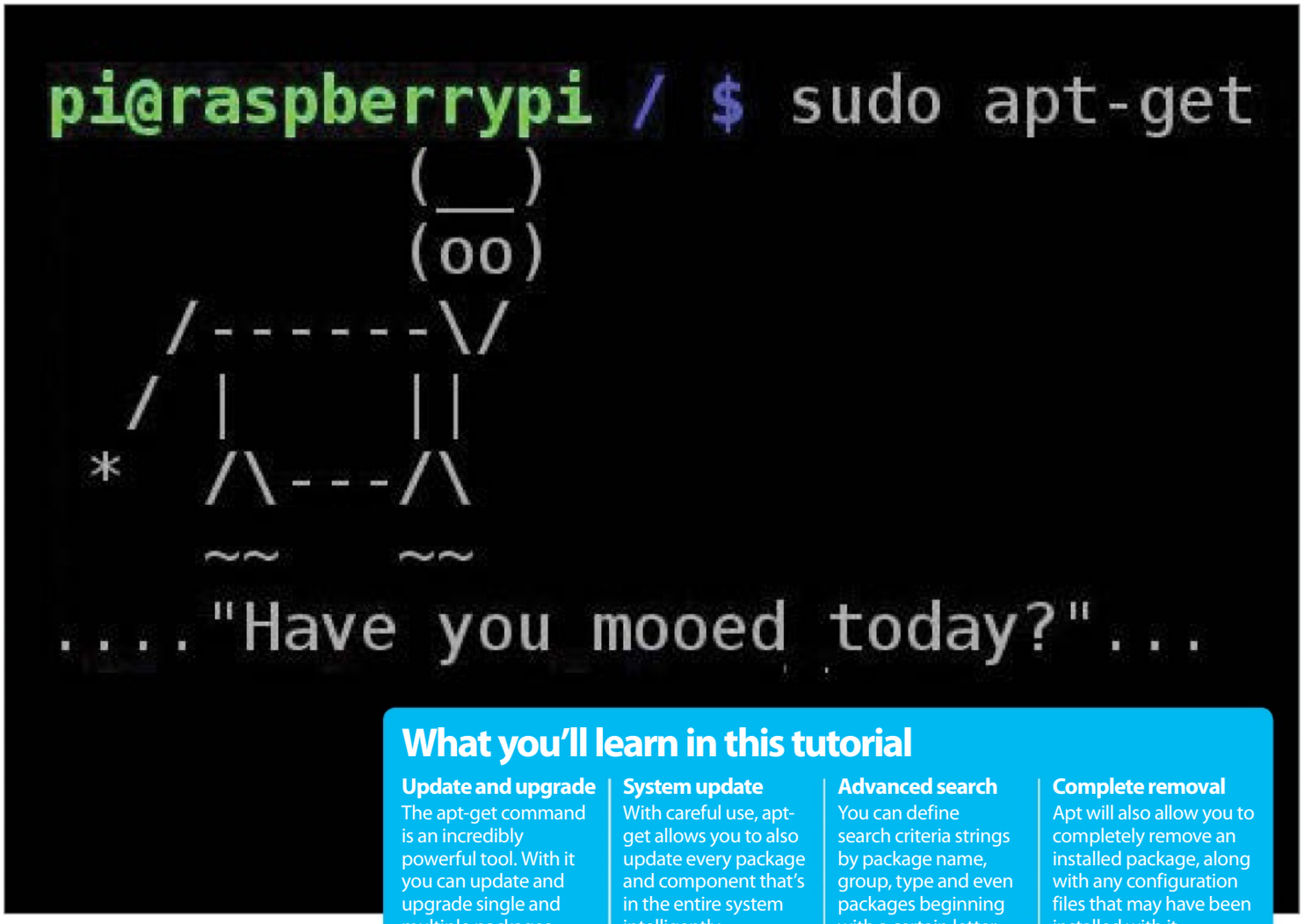


■ Fig 3: Gnumeric is an excellent spreadsheet application that is both comprehensive and light on resource usage

"You're more likely to be overwhelmed by choice than stuck for something to install"



■ Fig 4: The Xfce desktop is a good alternative to the default LXDE



What you'll learn in this tutorial

Update and upgrade
The apt-get command is an incredibly powerful tool. With it you can update and upgrade single and multiple packages.

System update
With careful use, apt-get allows you to also update every package and component that's in the entire system intelligently.

Advanced search
You can define search criteria strings by package name, group, type and even packages beginning with a certain letter .

Complete removal
Apt will also allow you to completely remove an installed package, along with any configuration files that may have been installed with it.

Install and use packages

The Raspberry Pi is great, but it's made better with the software you install onto it

Resources

Apt command help page:
http://linux.die.net/man/8/apt

Apt-get help page:
http://manpages.ubuntu.com/manpages/lucid/man8/apt-get.8.html

On its own, the Raspberry Pi is a near-perfect mini computer. It already contains a wealth of educational software, a few games, some programming utilities and a number of system tools. But, as with most computers, this is only the tip of the proverbial iceberg. By installing more programs, you can turn your Pi into a fully functioning desktop computer, a networked connected server, a games server, or a retro games machine, or even a state-of-the-art media centre.

These programs, known as packages, are as wide and as varied as the developers who originally designed them. In Linux, if there's a need for a particular program, then someone develops one. They then put it out to the world and make the source code freely available, hence 'open source'. Once the program has been tested, it will eventually make its way onto one of the many remote servers for that particular Linux distro.

These remote servers, called repositories, or repos, contain all the elements of the package in order for it to be downloaded and installed onto your system.

In our case, the Raspberry Pi repos are filled with every conceivable item of software you can imagine. But rather than list them all, we think it would be best to demonstrate how to actually get these packages installed onto your Raspberry Pi and how to use them, or execute them, when they are on there.

01 Update and upgrade

Getting hold of a package on the Raspberry Pi involves dropping into the command-line terminal, via the LXTerminal icon on the desktop, and entering a few commands. But before we do that, we need to make sure the system is up to date. Enter the following into the terminal:

```
001 sudo apt-get update
002 sudo apt-get upgrade
```

Or…

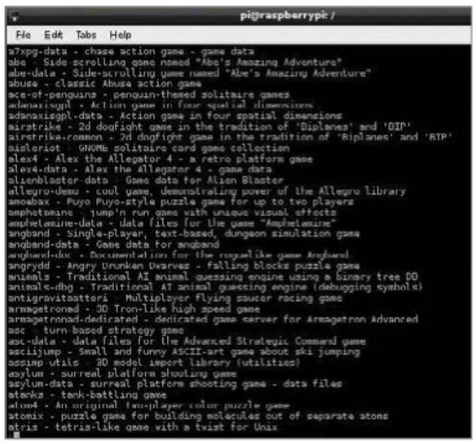
```
001 sudo apt-get update && sudo apt-get upgrade
```

02 Search for a package

The apt-get command (Advanced Package Tool) is the key to downloading and installing packages on the Raspberry Pi. In the previous instance, we updated the existing packages and system, upgraded any that needed it, and updated the current package list.

Now, let's search the list of server packages for available games.

```
001 apt-cache search game | less
```



03 Apt searching

The current list you find yourself in is the name of all the packages labelled as 'games' from the available server. In the list, the part before the hyphen tells you the name of the package, which is what you will need to know to be able to install it.

Use the arrow keys up/down to navigate; press 'Q' to exit.

04 Installing a package

Using the up and down arrow keys, navigate the list. If you find something you like the look of, say *Angry Drunken Dwarves*, remember the name of the package, in this case 'angrydd', and press 'Q' to exit the list.

To install the package, enter the following in the terminal:

```
001 sudo apt-get install angrydd
```

05 Executing the package

The result of the previous command should be the successful download and installation of the game, *Angry Drunken Dwarves*. To execute the newly installed package, you can either run it from the LXDE Menu under Games>Angry Drunken Dwarves, or by typing in the following into the terminal:

```
001 angrydd
```

06 Remove a package

This installing of packages is perfectly fine, and you can see just how powerful a command Apt really is. But, what if you want to remove a package?

Using the Apt command again, let's say we want to completely remove all trace of *Angry Drunken Dwarves* from the Raspberry Pi.

```
001 sudo apt-get --purge remove angrydd
```

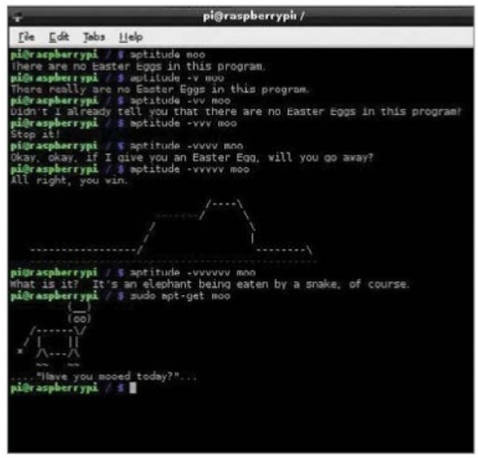
Enter 'Y' to accept the removal.

07 Apt Easter eggs

The Apt command is a shorter, non-menu-driven variant of the Aptitude command. This command has a long history in Linux, and as a result has some rather special 'features', also known as Easter eggs.

Purely for a bit of fun, type in the following commands and see the results:

```
001 aptitude moo
002 aptitude -v moo
003 aptitude -vv moo
004 aptitude -vvv moo
005 aptitude -vvvv moo
006 aptitude -vvvvv moo
007 aptitude -vvvvvv moo
008 sudo apt-get moo
```

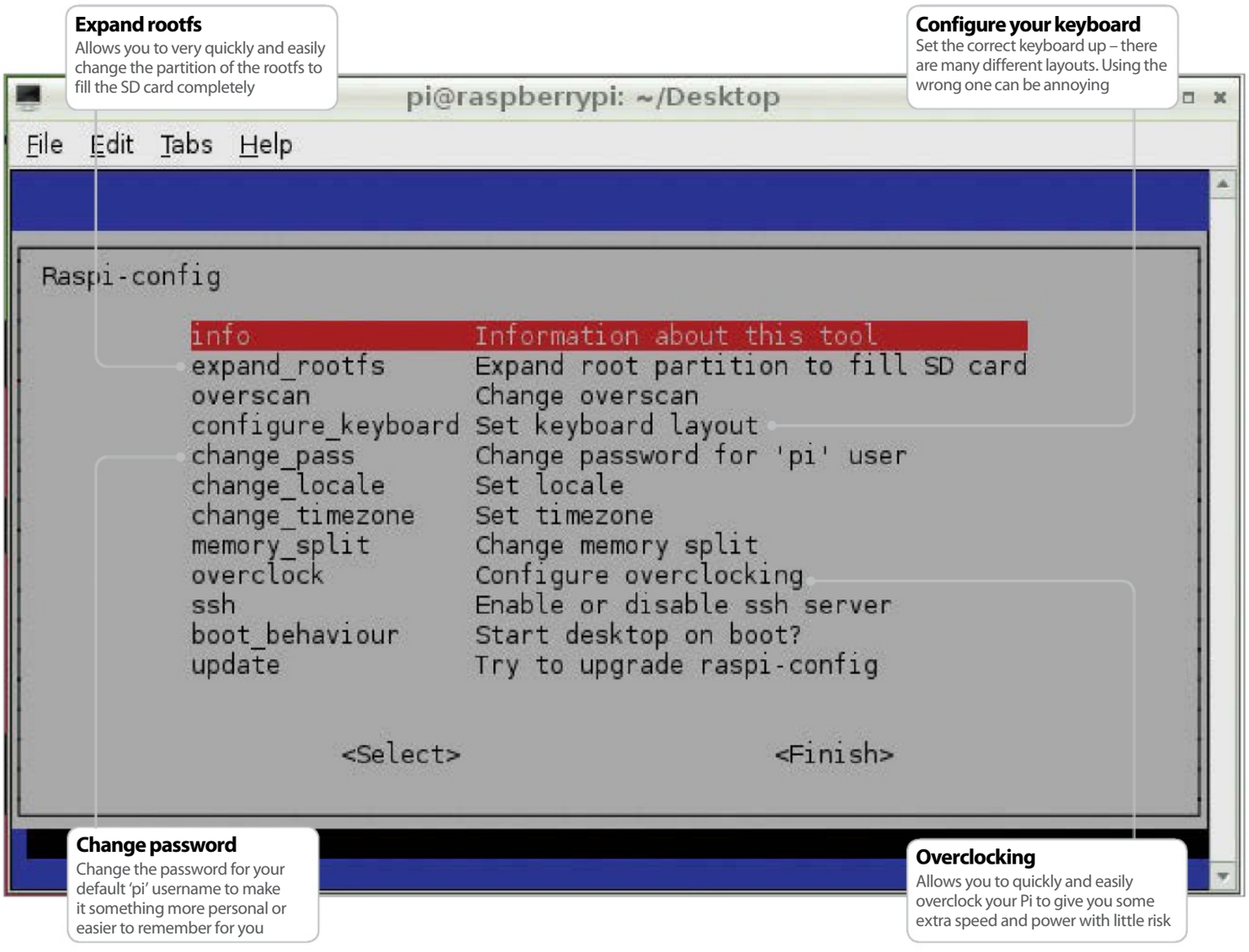


08 Man the Apt command

As you can see, there is more to the simple Apt command than what first meets the eye. There are many different sub-commands that you can run, and many different variations in which to run them.

If you want to see what else the Apt command can do, enter the following:

```
001 man apt
```



Master the RasPi Config tool

Easy setup for your Raspberry Pi using the built-in config tool

There is a configuration tool that comes with the Raspbian OS. Anyone who installed Raspbian themselves will have seen this once before – the 'RasPi Config' tool allows configuration of your system that would otherwise be trickier in the Linux environment. Tasks such as setting the date and time or regional settings for your keyboard are often done in a command-line interface with no dialogs, no additional help – for a new user, this is a nightmare.

There are some further specifics for the Pi and Raspbian itself, such as being able to easily enable overscan for your TV; change the split of memory to the computer/graphics card to steer performance in one direction or even overclock your system to make it a little faster; enable remote SSH access to the system; or stop the system booting into the desktop environment.

Another powerful option is being able to expand the root partition to the full size of your SD card – useful if you want to store actual data on your system.

While this tool is run from the command line, fortunately it gives the user common configuration options to make the experience more enjoyable.

Use of this tool is not a one-off either; you can get back to it at any point. The aim of this tutorial is to show you how to do that, and show practical use of some of the settings available to you.

Resources

Rasbian:
www.raspbian.org

01 Open the raspi-config tool

Start by double-clicking the LXTerminal icon on your desktop. This will start the command prompt, where you'll be able to run the config tool. To do this you'll need to run a command.

```
001 sudo raspi-config
```

When asked for your password, you won't actually see it being typed.

When you've typed the password and pressed Enter to submit it, the config screen will be shown to you. There are a few settings of particular interest that we'll cover in this section, although they all have their uses in the running of your Pi.

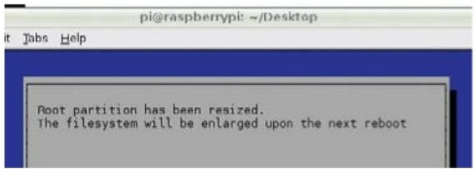
Some of the settings in this menu are important and some are irreversible, so use them with caution!

02 Expand the root file system

By default, the Raspbian root file system will be 2GB – this is done so that the image provided for it can fit on as many different SD cards as possible, by requiring the smallest possible footprint.

However, if, say you have anything bigger than a 2GB memory card, it's annoying as you can't use all of the space on your card – which means you can't store that much stuff on there, or get that much use out of it.

The 'expand_rootfs' option is what you need. Upon using this option, the command will be executed immediately. Proceed with caution when using this, although it should not have a negative consequence as long as no other partition exists. Reboot your system to see the changes.



03 Configure your locale, time zone and keyboard

Locale is the language and regional settings that your Pi is using – while this generally has little impact on what you'll see, it is also responsible for any default currency settings etc – so could prove to be an irritant at a later time if wrong.

Upon selecting the option, you'll be taken through a wizard. Use the arrow keys to check the built locale before building more (it takes a while).

Timezone will take you to a tzdata screen where you can use the arrow keys to select the correct time zones. Enter applies it immediately.

Finally, keyboard settings take you through a wizard to identify the kind of keyboard you're using so that characters you type match those on your keyboard. A must for secure passwords!

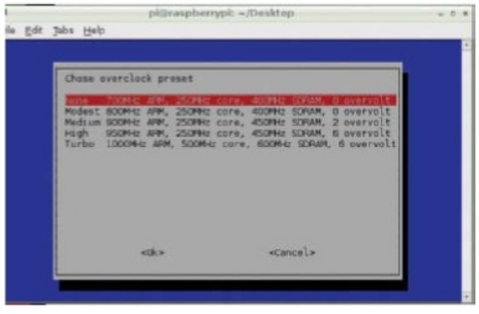
04 Overclock your Pi

In raspi-config there is the ability to set the clock speed and voltage of your Pi to several different presets. Setting the clock speed and voltage at higher rates than the specification may cause instability or a shortened lifetime for the system.

If you see any noticeable instability, it's recommended that you run this wizard again and set the clock speed back down to something lower – which should make things more stable again.

For the scope of this tutorial, a Modest/Medium overclock is recommended – it seems to give a little extra performance with no noticeable side effects. It is also recommended to reboot your system after making this change.

If you get any major problems, hold down the Shift key when rebooting to temporarily disable overclocking.



05 Change the memory split

Changing the memory split of the Pi allows you to give either the system or the GPU a larger amount of memory.

Type the amount of memory you wish the GPU to have. The value you give to it must be either 16/32/64/128/256.

Largely, this depends on what you want and what you are using your Raspberry Pi for.

The recommendations for different use cases are as follows:

32MB GPU memory for a Linux desktop distro or heavy non-GUI applications that do NOT need to play video or render 3D.

64MB GPU memory for desktop distros that want to play video or have 3D effects.

128MB GPU memory for applications and games that do extensive multimedia or play 3D rendered games.

For most people, 64MB of memory will suffice. But play with this to find what works best for you.

06 Change the boot behaviour

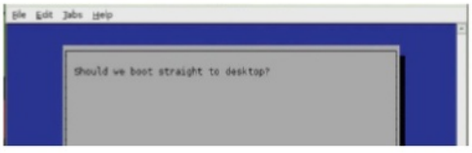
By default, the Raspbian distro will boot into a command-line interface, whereby you have to first log in as 'pi'.

If you then want to run a window manager (in this case, it's called 'X'), you have to give the system a command to let it know that's what you want to do.

For a lot of people this isn't really ideal since command lines scare them. Because of this, there's an option to start X automatically, on boot. Set this option to 'Yes' to enable this behaviour by default.

You can obviously revert this at any time to not boot – whereby you'll be presented with a text-based login where you have to start manually:

```
001 startx
```



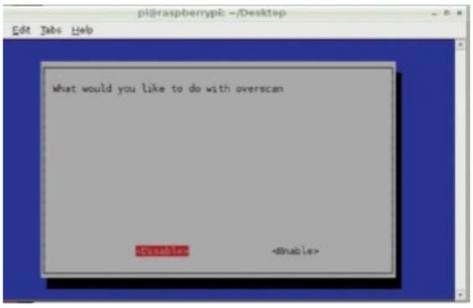
07 Turn overscan on and off

You may have noticed one of two behaviours if you're using your Pi with a modern HDTV.

- There is a black border the whole way around the image output by the Pi – it just doesn't fit correctly. This is caused by underscan.
- You can't see the edges of your screen to get to them. This is caused by overscan.

If you have the former issue, you may need to either turn on overscan, or enable a 'zoom' mode or similar on your TV.

If you have the latter issue, you need to turn overscan off so that you can see the edges. You may then also have to enable your TV to 'zoom' to fit the image to the screen.

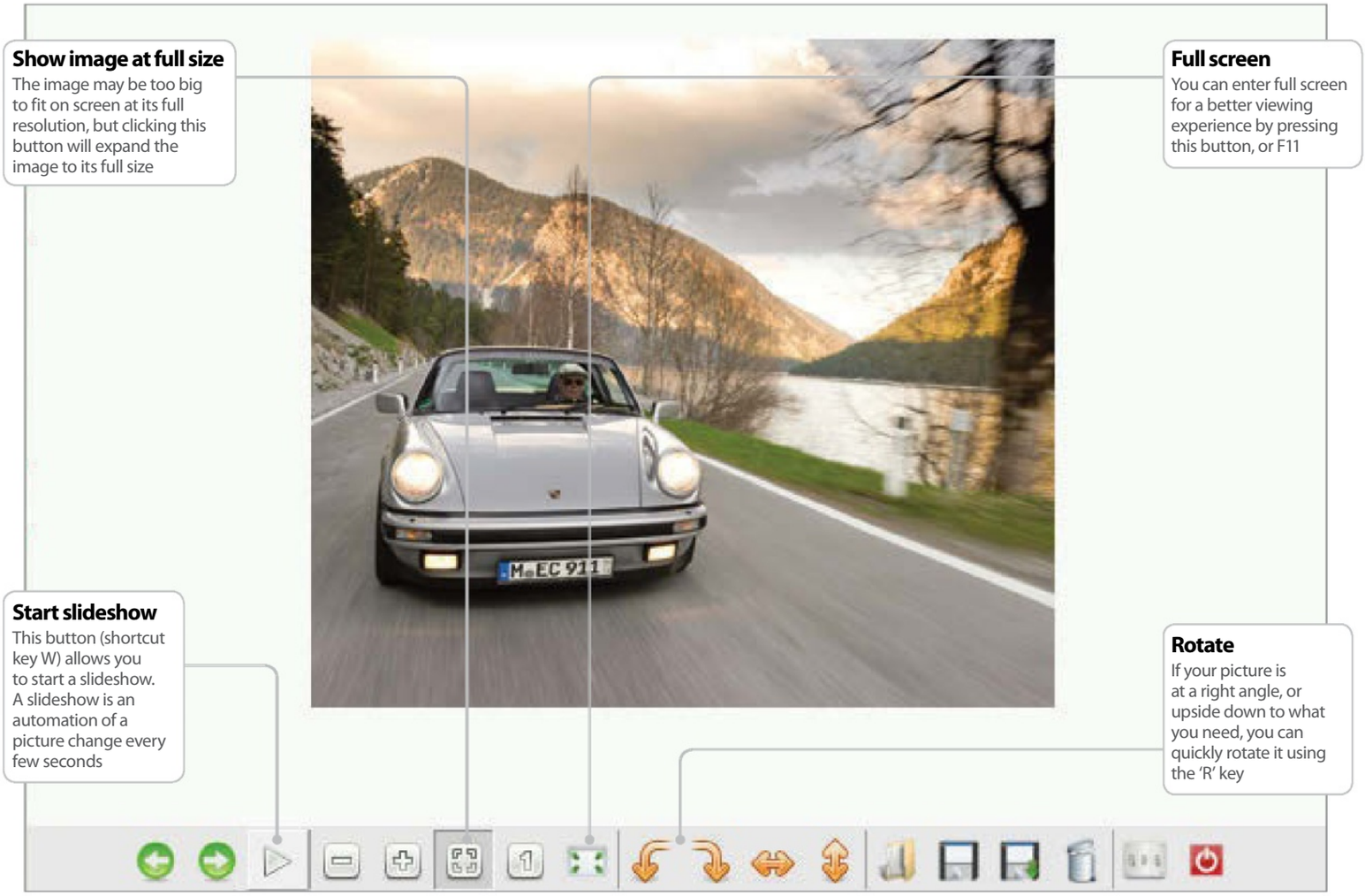


08 Update raspi-config

The raspi-config tool receives updates from time to time. This is generally to either add more features or tidy up the user interface, or both!

It's not a bad idea to run the updater when you use the tool – before you start changing any system settings. While it's (much) more likely that it'll be updated to look better or do more things, it's not impossible there could be miscellaneous bug fixes hidden within that would otherwise cause you some grief.

Remember, though, when you're trying to update your copy of the raspi-config tool, you'll need an active internet connection, either through an LAN cable or wireless dongle. Without them, it's never going to get any newer. Always try and make sure you're on the latest version.



View images on your Pi

Use your Pi to store an image gallery of your best collections

One of the things you might fancy using your Raspberry Pi for is some sort of portable projector. We don't mean a projector in the sense of strapping a bulb and a bunch of fancy electronics onto it (although trust us – it's been done!).

We're thinking more along the lines of a portable computer with a memory card big enough for a couple of hundred pictures – maybe your all-time favourites, maybe some of the last family holiday or the new dog; whatever you want to show.

The Pi being as small as it is, clearly this is a pretty handy way to carry something around with you that's got a little more scope than passing the family iPad around. We're talking about being able to put your pictures up on your friend's 50" plasma TV – using his or her surround-sound system for background music (though you'll need a music player application).

In more corporate environments, Pis are being used as a low-cost alternative to expensive presentation systems in welcome areas and receptions. This is down to the general ease of configuration and high flexibility involved when dealing with something so small.

There are a couple of obvious options when it comes to viewing pictures and slideshows on your Pi. Some require no additional work other than getting your files to them; others require rather more fiddling. We're going to take a look at a couple of these right now, and you can decide for yourself which you want to play around with. You'll also see the fastest ways to use them with keyboard shortcuts.

Resources

RaspBMC:
www.raspbmc.com

gThumb:
www.gthumb.com

01 Prepare yourself, and Pi

The first thing that you want to be absolutely sure of is that you have some content to display, so the very first step is to get some of your favourite images to your Pi.

We suggest downloading images straight from your camera or alternatively copying them from a USB drive, because these should be the easiest methods. While it is possible to copy files from a network source, you can't use them in situ – which, other than transferring, actually makes it a hindrance more than a help. For the purpose of this tutorial, create a Pictures folder on your desktop and copy the pictures on your drive or from your camera to here.

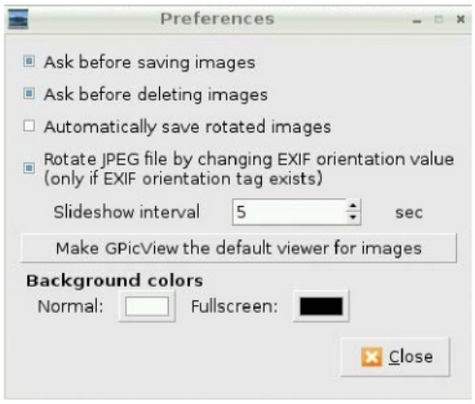
02 Let's have a look

So now you've got all of your pictures in a central place, let's go and have a look at them. You placed them in a 'Pictures' folder in your desktop. Open that and you'll see all of your files. Double-click one and by default it'll open up in a picture viewer. Along the bottom you'll see a bunch of different buttons – from left to right we have:
Previous Image / Next Image / Play Slideshow ||
Zoom Out / Zoom In / Fit Image to Window /
Show Full Size Image / Full Screen ||
Rotate Left / Rotate Right / Flip Horizontal /
Flip Vertical ||
Open / Save / Save Copy / Delete ||
Settings / Close Viewer

03 Show me your slides

Those of you who read carefully in the last step will have noticed that one of the buttons along the bottom toolbar is labelled 'Slideshow'. To start a simple slideshow, simply press this button. The screen will maximise to full and a slideshow will play, changing the picture every five seconds. To end it at any time, press Esc.

As mentioned, the default time for each slide is five seconds – which (usually) probably isn't enough. However, this is easily solved. Do so by hitting the 'Settings' button and either typing or using the arrow buttons to select your desired frame time in seconds on the 'Slideshow Interval' time. This setting is maintained when you reopen the picture viewer.



04 It has its limits

What you have above is basically the extent of the built-in picture viewer's slideshow abilities. This is kind of to be expected, given the relative simplicity of the pre-packaged applications. They're made to work, not to do anything particularly fancy or out of the ordinary. If you're looking for something more media-focused, you could check out other projects such as RaspBMC (**www.raspbmc.com**). If you want a more better photo management option, get gThumb. This can be used inside Raspbian and can be obtained using the Package Manager with:

```
001 sudo apt-get install gthumb
```

Once installed, open from the system menu. The next step will tell you a little more.

05 But wait, there are others!

gThumb is a more fully-featured picture manager. It has a tree-based view for easily navigating your file system to sets of different photos. In your Pictures directory, you could have lots of other directories that you could use as albums, or you could merge multiple albums into a single Catalogue. You'd be able to switch quickly between them. It also allows bookmarks (to get back to special albums quickly) and viewing of the camera's EXIF metadata. Another awesome feature of gThumb is that it has sharing options built into the interface – and provides a simple point-and-click interface to upload to Flickr, Photobucket and Facebook etc. You can even export to a web album to upload to your own website.

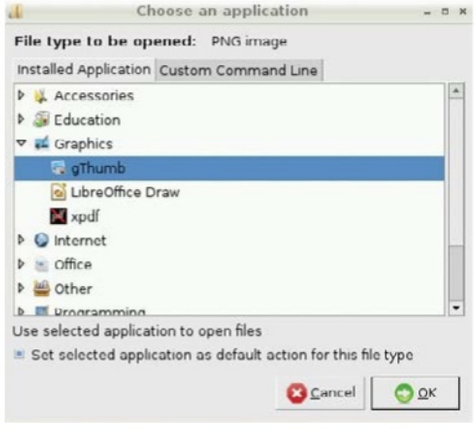
06 Play music

Using either the built-in picture viewer or gThumb does not really give you a lot of scope for adding any effects or transitions to your pictures. To make your slideshow better, you could perhaps construct a playlist or play a song. You could use an MP3 player application such as LXMusic to do this. Simply start the music playing in LXMusic, minimise or close it, switch back to your picture viewer or gThumb application and click the slideshow button. Now your holiday snaps will be accompanied by your choice of music. Sadly, when using these applications there's no way to start it in sync, but having audio play at the same time makes a pretty big difference to the overall ambience.

07 Switching your viewer

If you've installed gThumb, as mentioned in this article, it will automatically take over as your default image viewer. This has its own advantages and disadvantages. The advantages? It's a prettier interface, it has more options and it shows you more information about the image, and shows you thumbnails of others in the directory. The disadvantages? It's slower and it's a more heavy, bloaty program.

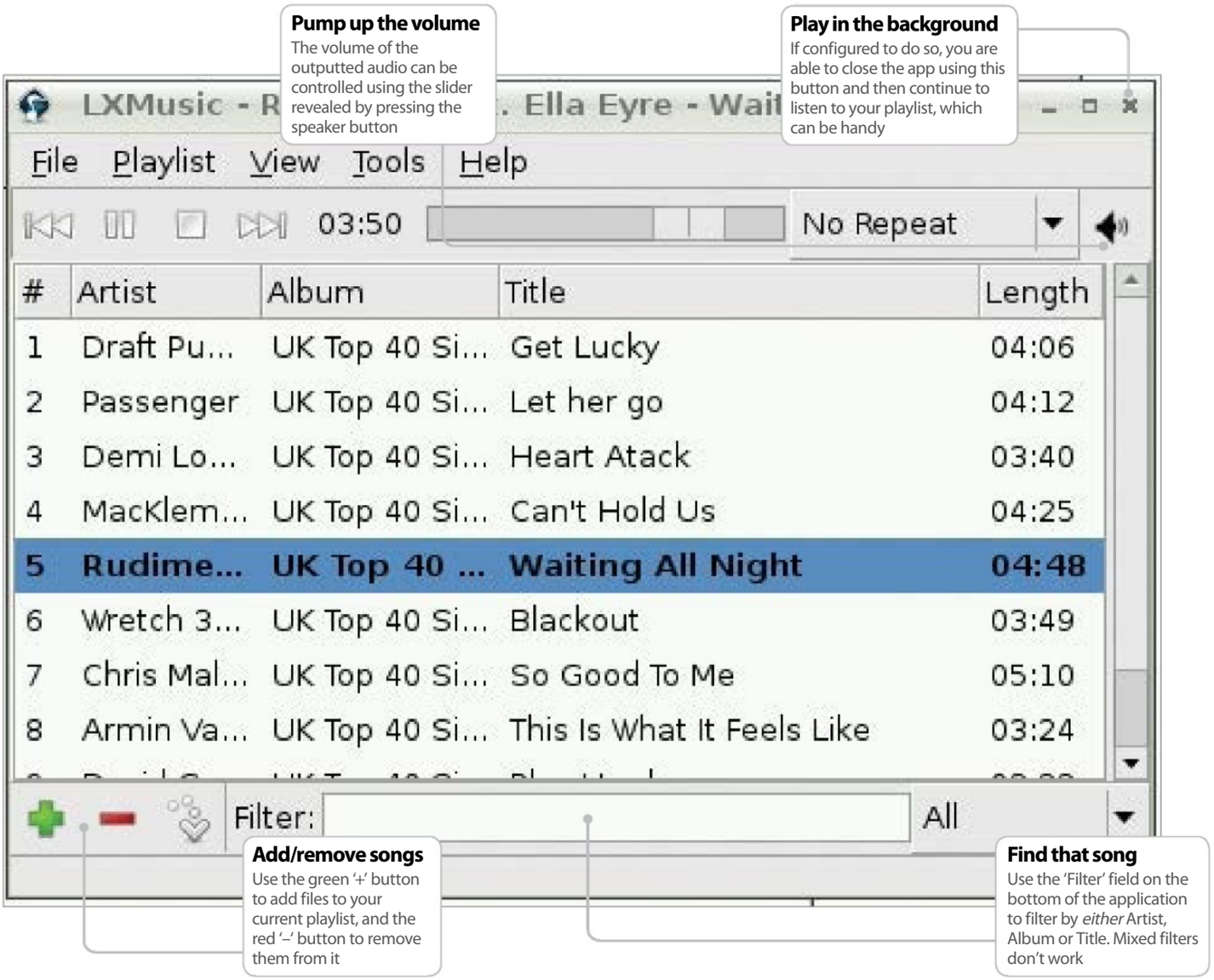
To change your default program, right-click an image and select 'Open With'. Choose either Image Viewer or gThumb, select the 'Set selected application as default' checkbox, and click OK. From now on, the application will load as default. You can experiment with whichever you prefer.



08 Some shortcuts

The picture viewer has a few keyboard shortcuts that can be used to reduce the need for using the mouse. This can be particularly useful if you don't have a good surface to use your mouse on – or if you like to be quick. Using these commands may seem rather difficult to get to grips with at first, but once you get used to them it'll frustrate you greatly any time they're not there.

The **left** and **right arrow** keys work for previous and next pictures.
F11 puts picture viewer into full screen.
G shows the picture full size.
Q quits.
W starts a slideshow.
R rotates.
S saves.
D deletes.
V flips vertical.
H flips horizontal.
Ctrl and +/- to zoom in/out.



Play MP3s on your Pi

Discover how to play MP3s on your Pi with a GUI music player

Resources

LXPlayer:
Linked in article – obtained through apt-get

So finally you've got your Pi all set up and have a desktop showing on your TV. You've got lots of different icons, but not many of them really seem to relate to things you'd frequently do – such as play music. While this isn't the general 'aim' of the Pi, using it as an MP3 player is well within its scope – but like many things, it requires a little more setup out of the box.

This tutorial will step through getting an easy-to-use, GUI (graphical user interface) MP3 player installed onto your system and to set it up in a way that you might be familiar with, either with Mac OS X or Windows systems – including being able to play music with it minimised and creating playlists.

We'll be using the Raspbian package manager Apt to install the LXMusic player – which involves use of the command line. It's easier than it sounds, so don't panic! Following this, we will walk you through the steps of creating a desktop shortcut for it the manual way and then move onto general use of your newly installed MP3 player.

One final note before we begin: LXMusic doesn't willingly play nice with any kind of source on a Windows 'SMB' network share. MP3s should be kept either locally (on your SD card) if you have the space, or on a USB drive with a FAT32 partition for smooth running. Directory structure does not matter.

01 Open terminal and install

On your desktop, double-click the 'LXTerminal' icon to open a command prompt that we'll now use to get down to business and install our MP3 player. We'll be doing this through use of the Raspbian package manager, Apt. When the command line is open, you'll see a black screen that you'll want to type into.

```
001 sudo apt-get install lxmusic
```

When asked for your password, beware that you won't see any characters that you type into it. You've just got to have faith that your fingers work correctly.

When you've entered the command to install LXMusic, lots of text will flash by you until you see 'Setting up lxmusic …' When this happens, we're good to go!

02 Create a desktop shortcut

Now we've got the player installed, it would probably be a good idea to have an easy way to open it. This is really simple. You can do it by opening the system menu (the 'Start'-like menu at the bottom left-hand side of the screen). Hover over Sound & Video until the menu expands. In here , you'll likely see a lonely item for 'Music Player' – right-click it and select 'Add to Desktop'. You will immediately see an icon added to the next available space of your desktop.

You can now access the Music Player with ease. Proceed by double-clicking the icon and actually opening the application – now we can listen to some music!

03 Know your player

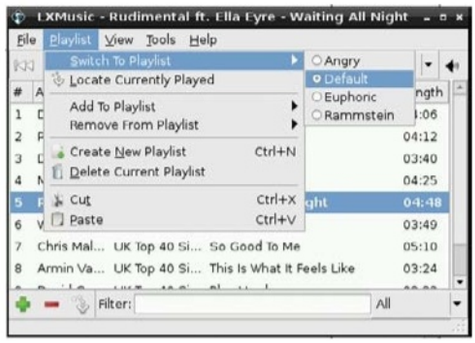
The interface of LXMusic is pretty similar to a lot of other MP3 players about. It helps to know where things are, however, so it's a good idea to know what you're dealing with. Along the very top you've got your menu bar. File, Playlist and View are the most useful menus here. Next you have the actual player controls: back, play/pause, stop and forward. Following this is the repeat preference and volume controls. Underneath, you have your playlist (which can be turned off totally in the View menu). This shows all currently queued songs in your playlist. Finally, underneath all this are the playlist add and remove buttons, locate current track (switches straight to the track in the playlist) and the filter box for searching your playlist.

04 Manage your playlist

As you'll probably know from almost any other operating system/MP3 player combination in the world, you can browse to the location of your files using the file manager application. That's great for stuff you've downloaded, but you can also create playlists so you can leave things playing without having to go back to it every few minutes. You'll notice the two brightest buttons on the application are a green '+' and a red '–'. These are your interface for adding and removing music. To add, click the green '+', upon which you'll be prompted to either add files (and given a traditional 'open' dialog) or add a URL to stream. To remove, simply highlight the song(s) (hold Ctrl and click to select multiple tracks) you don't want, and click the '–' icon.

05 Manage multiple playlists

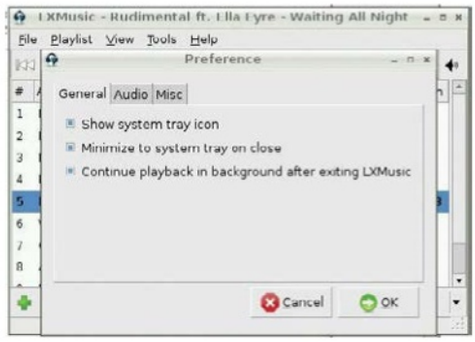
You're probably used to having more than one playlist too, aren't you? Some for different artists? Maybe a mix? Different moods? Well, we're able to manage multiple playlists in LXMusic – although you may not be used to the way that they're presented in this application. To start, create a new playlist by using either Ctrl+N or going to Playlist>Create New Playlist. You'll be asked for a name – type one that's fitting. Proceed to add items to your playlist as normal. These items will be automatically saved into your playlist – when you want another, just repeat the action and create a new one with a different name. To open a previous playlist, simply click the Playlist menu, then 'Switch to Playlist' – then to whichever of your new playlists you want.



06 Set your preferences

There's a limited amount of configuration you can do with LXMusic, – it *is* a simple Music Player so there's nothing particularly fancy about it. However, there are some options that aren't switched on by default that you'd probably expect. To get to the preferences panel, click File followed by Preferences on the menu bar. The General tab has just three options:

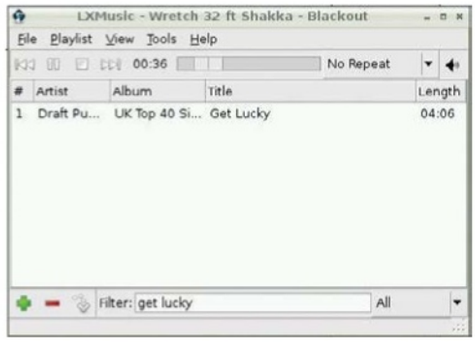
- (OPTIONAL) Show system tray icon – Show an icon in the system tray to control and get back to the music player
- (OPTIONAL) Minimise to the system tray on close – instead of exiting the application, just minimise (allows playing of music without the window)
- (OPTIONAL) Continue playback in background after exiting LXMusic – playing the current playlist even if the application exits.



07 Search for music

If you end up with a big playlist, there will likely come a time when you don't want to listen to things in order, or you'll want to listen to a song that you can't see. At the bottom of the LXMusic interface, you'll see a text box with 'Filter:' to the left of it, and a drop-down with 'All' to the right.

This is a simple filter box – it'll allow you to filter your current playlist by the terms you specify in this box. It's not smart enough to do matching in multiple columns simultaneously (so you can't type, say, 'Rammstein Mutter' to search for the artist Rammstein and album Mutter). You can, however, stop search results coming back from multiple columns by changing the 'All' filter type to something specific.

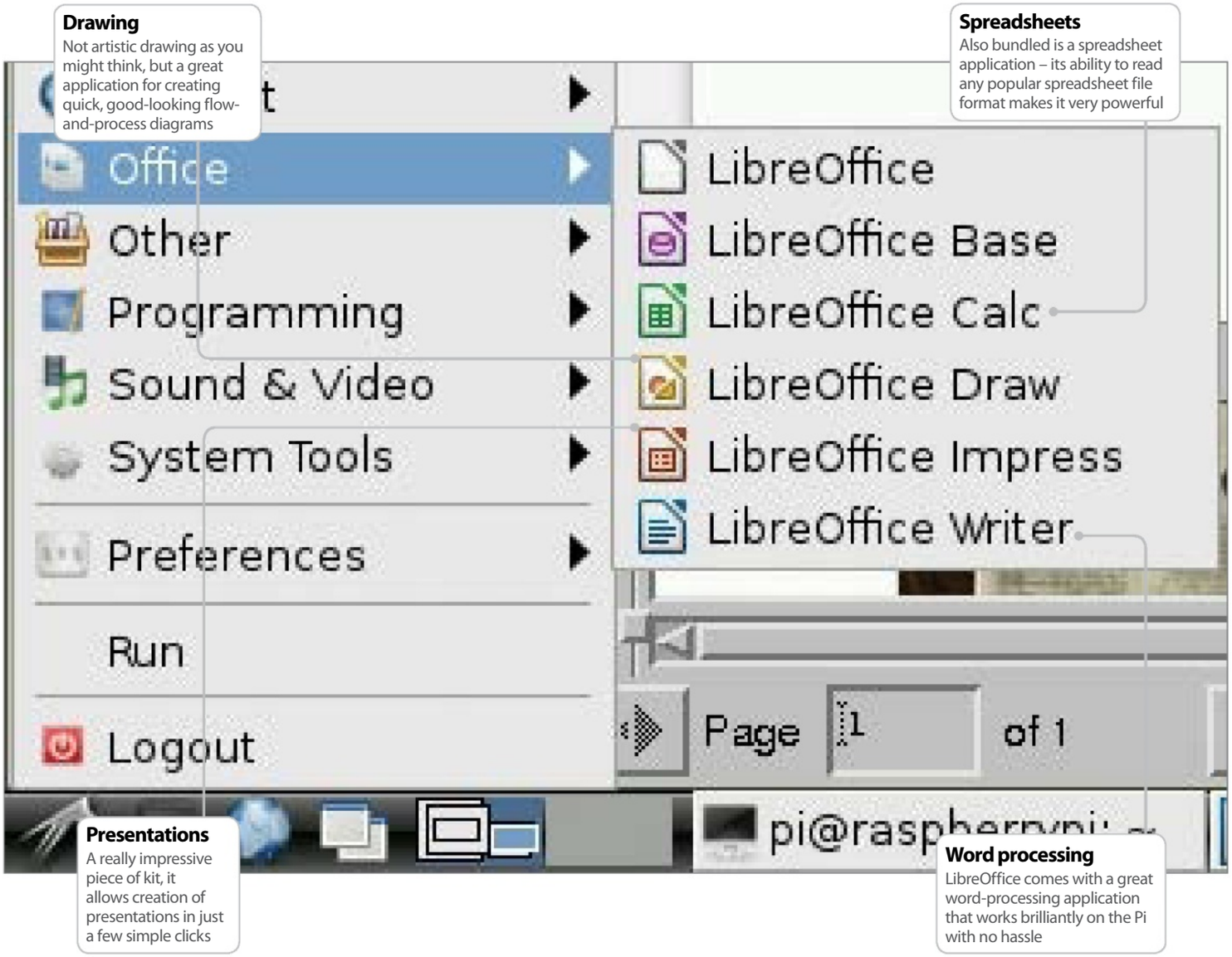


08 Odds and ends

As we've covered all of the main features of LXMusic in this article, we'll now cover a couple of random bits.

If you've got the option to have a system tray icon switched on and then exit to the desktop from it, you can continue to do other things and listen to your music. At some point you'll want to get back to it – do this by single-clicking the musical note icon in your system tray.

Another useful feature is the ability to view information about your songs. This dialog can be viewed through File>File Properties. From here you'll be able to see any ID3 information about the selected file, as well as its location, quality and file type.



Turn your Pi into an office suite

How to add a full, free office suite to your Raspberry Pi

Common tasks when using any computer are word processing, spreadsheets and presentations. This is well within the scope of the Raspberry Pi. Raspbian comes bundled with a couple of text editors which are fine for making simple notes. However, as soon as you come to creating actual documents and want any kind of formatting or pictures inside them, these editors really aren't up to the task and so you'll want to move to something new.

The best way to get round this is to install another application – or, as is the case here, a full office suite! This is really easy on the Pi and the best option available is open source and completely free: LibreOffice.

Installing it couldn't be much easier – and it gives a lot of scope to what you're able to use your Pi for. It's fully featured and so long as you're not creating huge documents with hundreds of images in, will be more than adequate for most of your needs.

You'll need a working internet connection for this tutorial, so make sure your LAN cable or wireless dongle is plugged in and functional, because we'll be using the built-in package manager.

Installation time can vary a little, but aim to set around 30 minutes aside and you won't be far wrong. The screen shouldn't stay still for too long, so you'll know it's doing something!

Resources

LibreOffice:
Installed through tutorial using apt-get

01 Install LibreOffice

The best thing about open source software is the fact that you can install it any time (so long as you have an internet connection or installation medium anyway) at your total convenience – without having to worry about handing over payment details. So, let's get right on with doing exactly that. Open your terminal – you can do this by double-clicking LXTerminal from your desktop. Install the LibreOffice office suit by using the package manager:

```
sudo apt-get install libreoffice
```

Now type in your password. Press 'Y' and Enter when prompted to install other dependency packages.

Sit back, relax and enjoy. Note that when asked to type your password for sudo, you won't actually see it being typed on screen.

02 Explore the suite

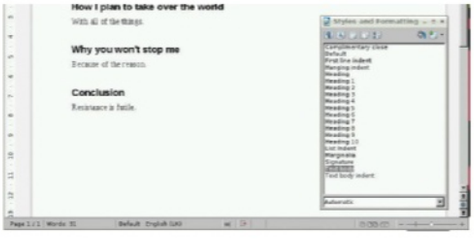
When you've installed LibreOffice, you can admire your new applications by hitting the system menu. You'll see a new 'Office' category that's shown up, underneath which you'll see the following applications have been installed. That's right, it's not just for word processing!

LibreOffice Base – A database management application.
LibreOffice Calc – For looking after spreadsheets.
LibreOffice Draw – To make flowchart/visualisations.
LibreOffice Impress – For presentations.
LibreOffice Writer – The word processor.

These applications are perfectly able to open a majority of documents created in their expensive Microsoft Office counterparts and are almost as feature rich. But we'll now explore a few of the Writer features in more detail.

03 Add some style

We're focusing on some of the word-processing features on the Pi specifically here, so go on ahead and open up LibreOffice Writer. When you're writing documents, it helps to put some formatting in. It helps with reading the document, and it looks prettier too. When you start writing, to the left of the font drop-down menu there's another drop-down that currently reads 'Default'. Select some text, then use this drop-down to change the current applied style. If you don't like the defaults, select 'More' from the drop-down, or press F11. This will allow you to customise the styles that exist already, switch an existing style to properties of the selected text, or add new ones.



04 Add an index

One of the great things about using formatting is that it allows you to create a contents page for your work really easily. So long as you use consistent headings for the different levels of your document, it's a quick and easy way to give a guide to what's in your document. Another advantage is that if afterwards you export the file to PDF, you'll end up with each heading in the contents hotlinking directly to the correct section within your document.

To create your index, create some space at the top of your document, and then insert your index by using the menus to go to 'Insert Indexes and Tables>Indexes and Tables'. Review the selected options and click OK when you're happy. It'll insert to the section of the document that your cursor is on.

05 What about images?

There's often a requirement to put images in a document, and naturally this is something we can do relatively easily.

There are a couple of ways of doing this – the easiest one by far is to drag an image in from the file manager directly into the document. Open the folder in file manager. Say it's on your desktop, simply drag the file into LibreOffice Writer and you will see the icon change; upon letting go, your image will appear where you drop it.

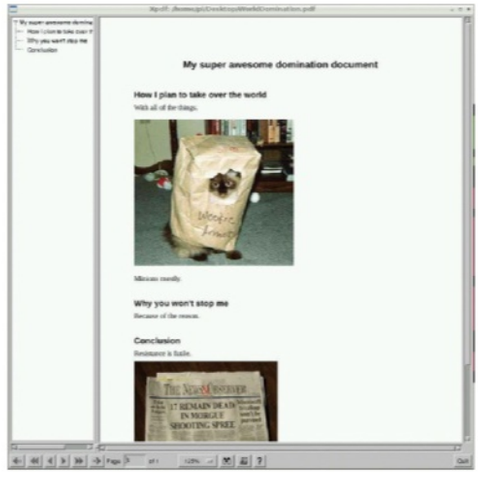
Alternatively, you can use the Insert>Picture>From File option – that gives you an open dialog to do the same thing.

06 Export as a PDF

The PDF format is a very common way of exchanging documents – mainly because in its simplest form, it's an easy way of stopping accidental changes/modifications to a document. It's also easy to set passwords on PDFs and stop purposeful modifications to them.

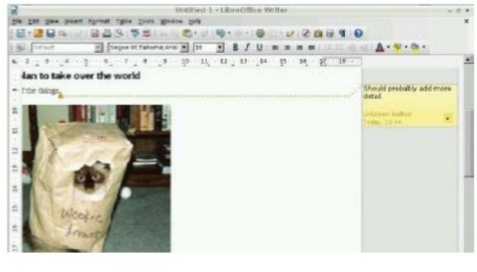
The main advantage of PDFs, however, is that you don't really need to worry about the recipient having a specific application to look at them in – as most modern operating systems will have some sort of PDF viewer built in.

With LibreOffice, this is easy. You can save your document as a PDF using the File>Export to PDF menu option. Simply choose your desired options, click Export and save it in the appropriate place.



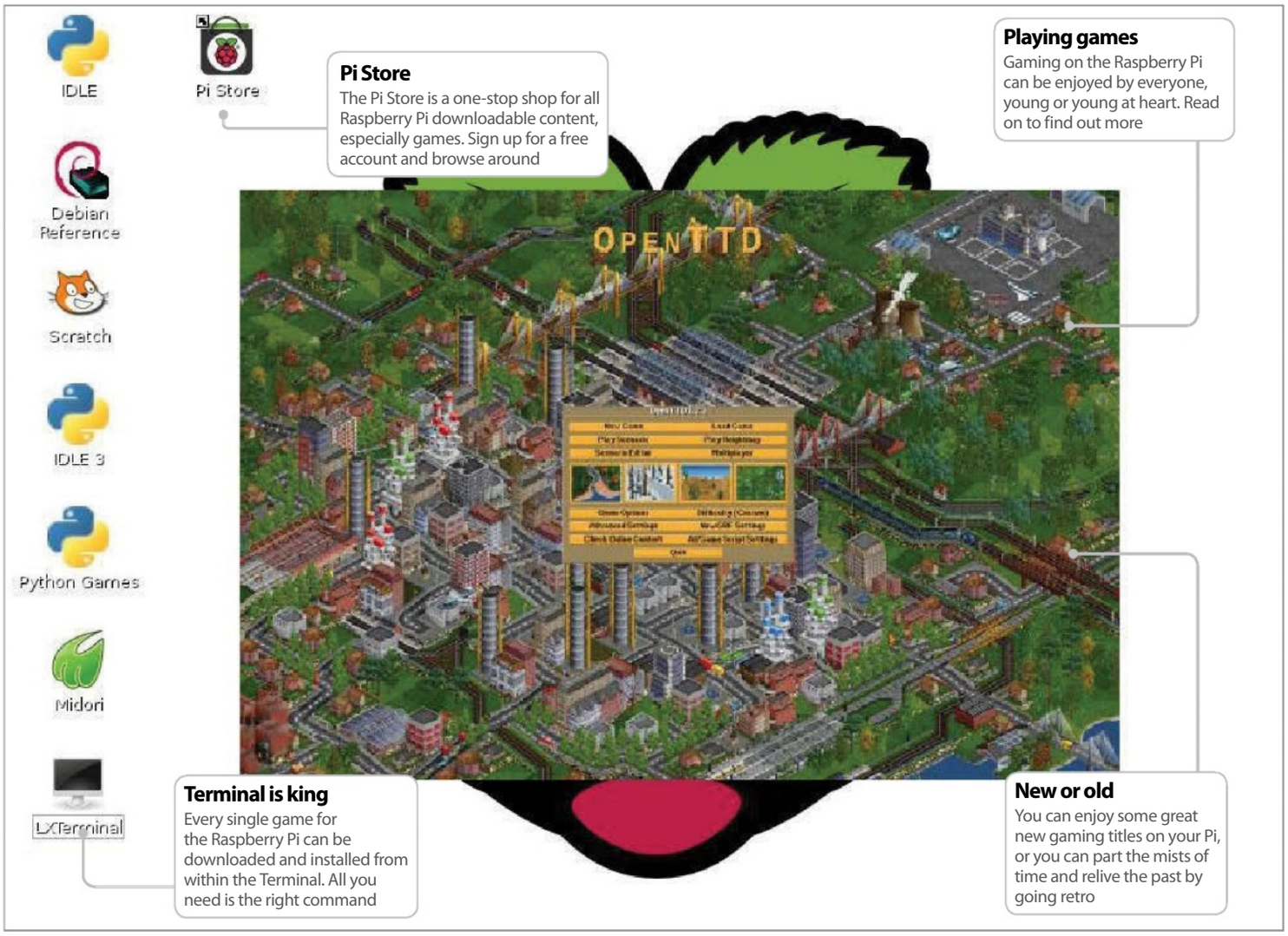
07 Help yourself by using comments

When writing a long document you'll often end up thinking of things that you should really check up on or add to the document later. When you think of these things, you can make notes about them – but physical notes aren't always that ideal. There's a little-known function in most word processors now for 'Comments' – even more useful when there are multiple people moving through the document. Comments can be added at the current point in the document with Ctrl+Alt+C or by using the menu option Insert>Comment. If you want to get rid of the extra 'Comments' pane, you can use the View menu to toggle them: View>Comments.



08 Linking out to the world

Say you've used outside resources in a document that you want to reference, or maybe you don't directly need to include the information though it's useful – it might be handy to have a way of getting back to that document later on. You could have an Appendix of information at the end of your document that links to these other resources. You can hyperlink to these easily using the Hyperlink toolbar button or the same from the Insert>Hyperlink menu. Select your text, then click it and you'll be presented with a dialog to link to one of the appropriate places. It's important to note that direct document links use full paths. If it's being sent to someone else, avoid using these as they'll usually break. Ideally use it only a personal reference guide.



Play games on your Pi

There's plenty of fun to be had with a Raspberry Pi, but even more when you start gaming on one

Although the Raspberry Pi may be small on resources, it has just enough under the hood to enable us to enjoy some great gaming.

Naturally it won't be able to play the latest triple A-rated titles; the likes of *Assassin's Creed 4* are well and truly beyond its capabilities. However, that doesn't mean it can't play some endearing games from a wealth of genres. Indeed, the Linux community of developers and clever coders have spent years giving us access to many a free game which is just as addictive as the latest and greatest on offer.

The trick of course is to know how to get those games onto the Raspberry Pi, and where to look for them in the first place. Thankfully, that's the easy part. Finding and installing a game is a simple enough affair. Getting constantly beaten by your 12-year-old offspring is a tad more difficult to comprehend.

So, if you have stretched your Pi with projects involving electronics, motors, or sending to the very edge of space, why not relax a little, unwind, and enjoy some free time with a game or two? Perhaps even a spot of retro gaming? At least then, we 40-something-year-old mums and dads may have a chance of beating those youngsters when it comes to *Manic Miner* on a ZX Spectrum emulator! Although we can't guarantee that.

Resources

The Pi Store:
www.store.raspberrypi.com/projects

01 The Pi Store Games

The first port of call to look for some fantastic Raspberry Pi games is via the Pi Store. Searching it is simple enough, but first you have to launch it. To open up the Pi Store, either double-click the icon on the desktop, or type the following into the Terminal:

```
pistore, and press Enter
```

02 Getting hold of a game

There's something like 36 games available via the Pi Store at present, with more being added all of the time. Click on the 'Games' tab, along the top of the main Pi Store window, and have a browse through the selection. When you find the one you want, simply click on the 'Free Download' or 'Buy Now' buttons.

You must sign in with an IndieCity account to download games from The Pi Store. Once the game is downloaded and installed, you can play it via the Pi Store by clicking on the 'Launch' button.

03 Using Synaptic

Synaptic is a package manager for Debian-based operating system, such as Raspbian. With it you can search every available program based on its category, in this case: games.

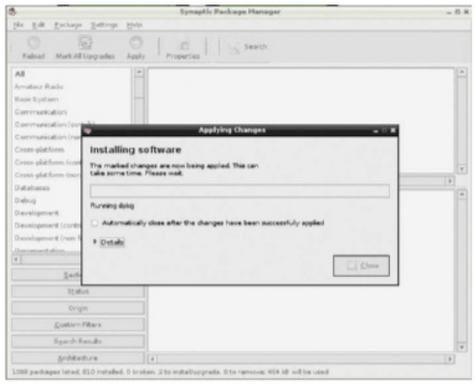
To install Synaptic, simply drop into a Terminal and enter the following command:

```
sudo apt-get install synaptic
Answer 'Y' to any prompts
```

04 Installing a game via Synaptic

Once installed, find it in Menu>Preferences, click the Synaptic icon, and scroll down the list in the left-hand pane to 'Games and Amusement'.

Next, scroll down the list in the right-hand pane, and find a game you like the look of, then click the box to 'Mark for Installation'.



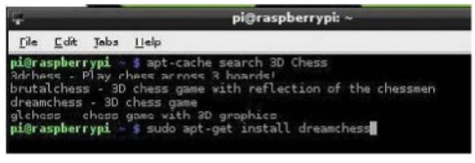

```
Click to 'Mark for Installation'
In the pop-up window, click the 'Mark' button
Click 'Apply' located in the top menu to install
Click the 'Apply button in the pop-up window
Click 'Close' when the operation is complete
```

05 Using apt-cache to search

Synaptic, although visual, is not great and it is very slow on the RPi. Instead, drop into a Terminal and use 'apt-cache search' in order to search for a game. Apt-cache search interrogates the Raspbian repositories for the string you enter in the command.

For example, to search for a 3D Chess game, do the following:

```
apt-cache search 3D Chess, and press Enter
```



06 Installing a game from the Repositories

Once you have found a game that you like the look of, using the 3D Chess example from above, all you need to do is enter its title into an 'apt-get' command.

For example, installing 3D Chess from the repositories can be accomplished by entering the following command:sudo apt-get install 3dchess, and press Enter.

07 Going retro

You can't beat a bit of retro gaming. In this example, we'll install a ZX Spectrum emulator. First browse to The World of Spectrum site, and download a TAP format game, such as Manic Miner (goo.gl/U4AvSM). Unzip it, then install a Spectrum emulator and run the game by entering the following commands:

```
sudo apt-get install fuse-emulator-common, and
press Enter
Press 'Y' to confirm the install
Sudo apt-get install spectrum-roms fuse-emulator-
utils
Fuse-gtk MMiner.tap (the location of the unzipped
TAP file)
```

08 Useful Links

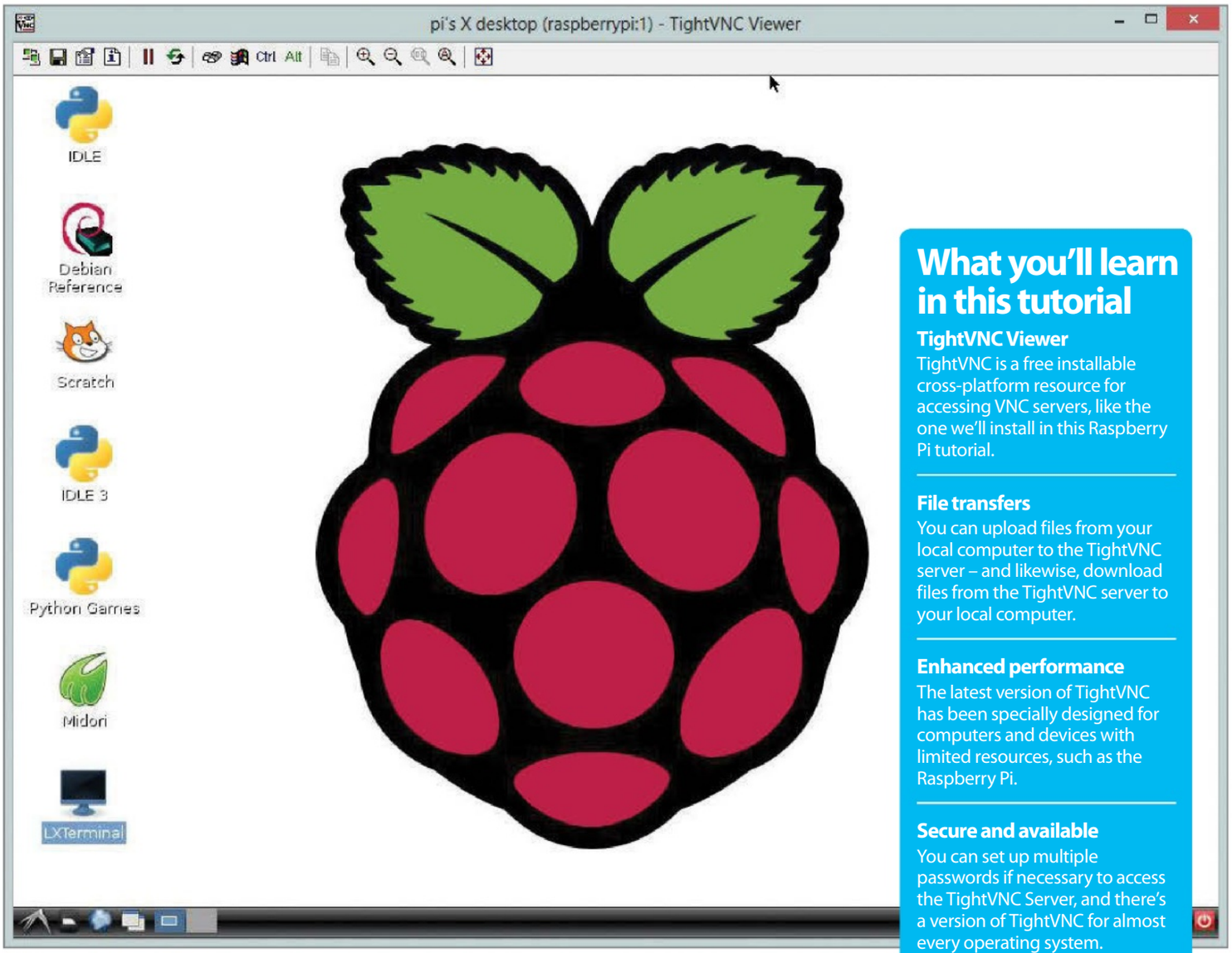
While it would be great to list every game available for the Raspberry Pi, unfortunately we can't do this.

To that end, here are some rather useful links to games for the RPi, and how you can get them. Beyond that, all you need to do is enjoy them – great!

```
Raspberry Pi Games List - An impressive colour-
coded list at goo.gl/1rtspS
Raspberry Connect - Another handy list of games
at goo.gl/Czjj90
Raspberry Pi site Games Tag - Liz Upton
shows off Pi gaming at goo.gl/8auImd
RetroArch Tutorial - The Raspberry Pi
forums offer some help goo.gl/IZtez9
```

This is just the tip of the iceberg; there are literally hundreds of games that are capable of being played on the Raspberry Pi, not to mention the back catalogue of retro titles that stretch out over 40 years.

Although it may not look it, the Raspberry Pi is a nice little games machine, and to impress that point even more, check out Games Tag from the Raspberry site, where Liz and Eben Upton regularly update the gaming scene, including what is new.



What you'll learn in this tutorial

TightVNC Viewer
TightVNC is a free installable cross-platform resource for accessing VNC servers, like the one we'll install in this Raspberry Pi tutorial.

File transfers
You can upload files from your local computer to the TightVNC server – and likewise, download files from the TightVNC server to your local computer.

Enhanced performance
The latest version of TightVNC has been specially designed for computers and devices with limited resources, such as the Raspberry Pi.

Secure and available
You can set up multiple passwords if necessary to access the TightVNC Server, and there's a version of TightVNC for almost every operating system.

Access your Pi desktop remotely

Learn how to get access to your Raspberry Pi without it being in front of you

VNC (Virtual Network Computing) is a graphical desktop access and sharing system that allows a user to remotely control and use the desktop of another computer from their own system.

It's handy in many ways: to give you control over a remote system; to help out another user elsewhere; to allow a computer to remain powered on, but without the need for a keyboard, mouse or even a monitor.

In our case, we aim to allow access to the Raspberry Pi desktop without it being hooked up to the aforementioned peripherals. This way, you can do everything you would normally do on the Raspberry Pi but with just the power supply and access to your home network, thereby freeing up space and the need for extra expense.

By the end of this tutorial, you should have your Raspberry Pi ready to accept incoming VNC requests from within your home network, with it sitting somewhere in your home, discreetly out of the way; next to your router, perhaps. You'll be able to access the desktop of your Pi as if it were in front of you, and you'll be able to do so from a variety of systems. As long as the system and program uses VNC, then you'll be able to connect to the Raspberry Pi. In real-world terms, this means you could potentially access the Raspberry Pi desktop via an Android tablet, or even your phone!

Resources

TightVNC:
www.tightvnc.com/release-2.7.php

01 Static IP address

The first step for us is to make sure the Raspberry Pi has a static IP address. Basically, an IP address is a group of numbers that your network assigns to devices in order to tell them apart. To set up a static IP address, double-click LXTerminal and type the following and then press Enter:

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

02 Static IP part 2

This file controls the IP addressing for the Pi. You need to scroll down to the 'iface eth0' line and remove DHCP and replace it with static. Now, on the line directly below, enter the IP address that you want force your Pi to have, along with the subnet mask and the gateway:

```
address 192.168.1.93
netmask 255.255.255.0
gateway 192.168.1.254
```

03 Static IP part 3

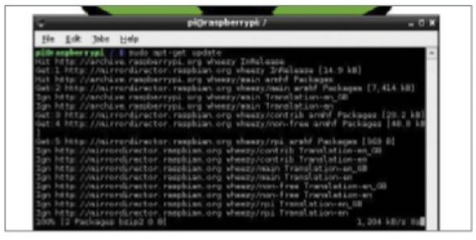
After you've entered those details, exit nano by pressing Ctrl+X, followed by Y to accept the changes, and then press Enter a couple of times to get back to the command prompt in the terminal. You can now either reboot your Pi, or type the following into the terminal:

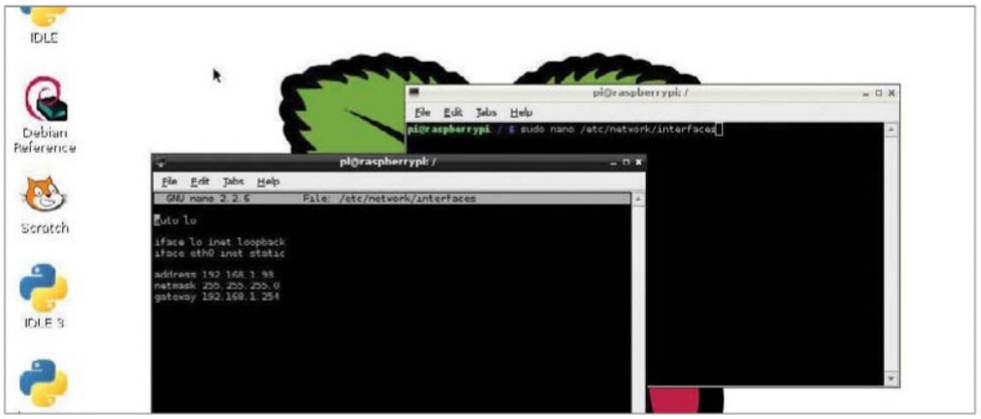
```
sudo /etc/init.d/networking stop
sudo /etc/init.d/networking start
```

04 Installing VNC part 1

Now we'll install VNC and ensure that it starts automatically whenever the Pi is booted up. This used to be a bit annoying under older Raspbian versions, as configuring services often had a nasty habit of breaking the system. But no longer. Enter the following commands, pressing Enter after each one:

```
sudo apt-get update
sudo apt-get install tightvncserver
tightvncserver
```





05 Installing VNC part 2

When the packages have downloaded and installed, follow the instructions on screen (see below) to set up a password and confirm it, but answer 'n' to the view only option. This really is just a security feature and since you are only accessing the Pi at home, it's not absolutely necessary – but, it's still good practice.

```
You will require a password to access your
desktops
Password:
Verify:
Would you like to enter a view-only
password (y/n)? n
New 'X' desktop is raspberrypi:1
```

06 Configuring VNC server

That's the VNC server installed, up and running. Now we need to make sure it loads up as a service every time the Raspberry Pi reboots, so you can access it even if the Pi undergoes a power cycle. To configure the Pi to do this, type the following in the terminal and press Enter.

```
sudo nano /etc/init.d/tightvncserver
```

07 Configuring the boot service

As you can see, we're back in the nano text editor again, and we'll need to enter some lines of commands in order to allow the Raspberry Pi to activate the VNC server every time it boots, or whenever the service is restarted. In the editor, type the following:

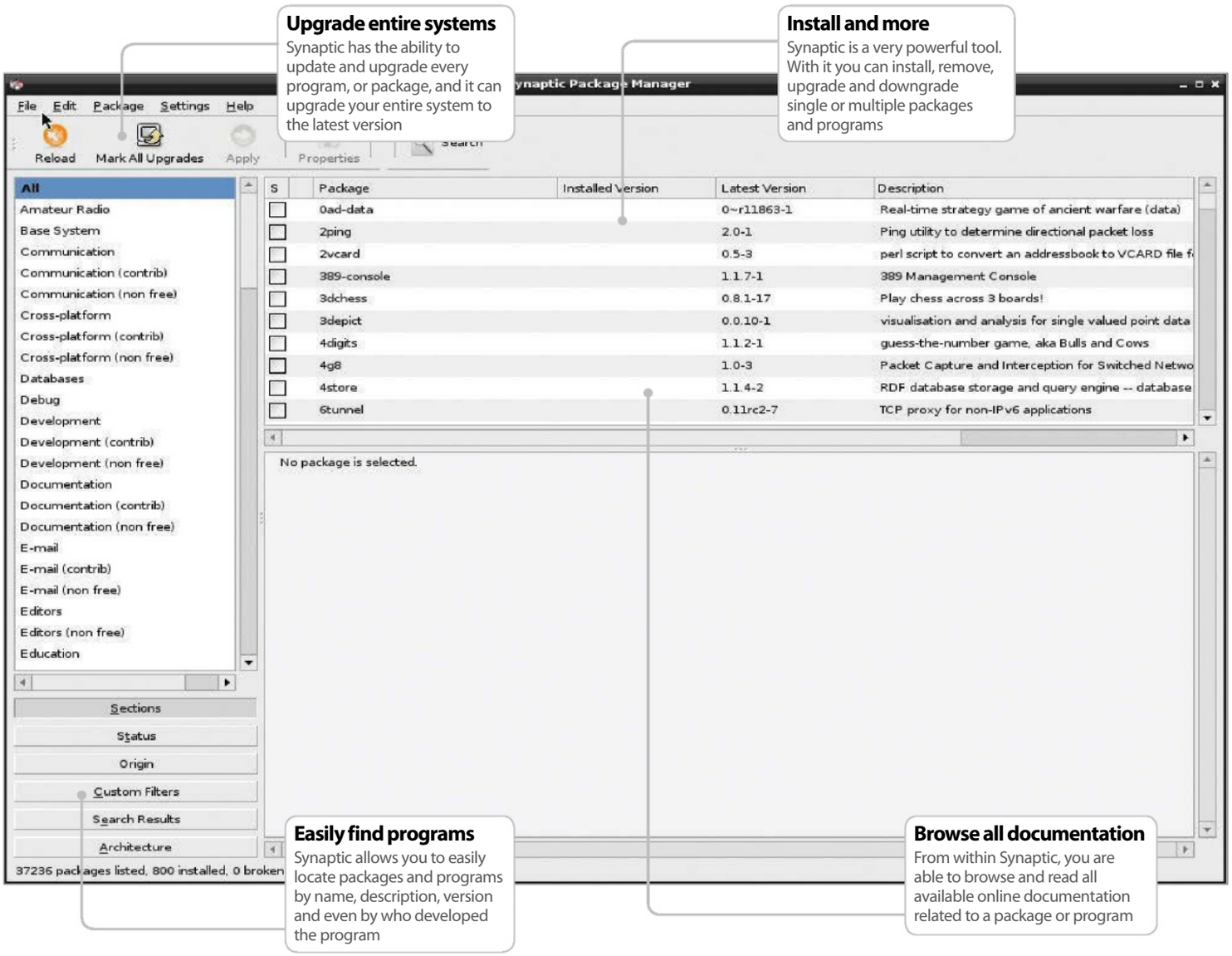
```
    #!/bin/sh
    # /etc/init.d/tightvncserver
   # Set the VNCUSER variable to the name of
the user to start
    tightvncserver under
    VNCUSER='pi'
    case "$1" in
    start)
    su $VNCUSER -c '/usr/bin/tightvncserver :1'
    echo "Starting TightVNC server for $VNCUSER"
    ;;
    stop)
    pkill Xtightvnc
    echo "Tightvncserver stopped"
    ;;
    *)
    echo "Usage: /etc/init.d/tightvncserver
{start|stop}"
    exit 1
    ;;
    esac
    exit 0
```

08 Reboot and ready to go

Now press Ctrl+X, then Y to save, followed by Enter a couple of times to get you back into the Terminal. What we need to do now is edit the permissions of the script we've just created so that it's executable and active. Do this by typing the following commands into the terminal, ensuring that you press Enter after each one otherwise it will not be registered:

```
sudo chmod 755 /etc/init.d/tightvncserver
update-rc.d tightvncserver defaults
sudo reboot
```

Once you have completed these steps, the final thing that you will need to do is to unplug the Raspberry Pi and locate it somewhere that has easy access to a network cable. If you install the likes of TightVNC Viewer, or any other remote access software (as long as it uses the Tight protocol) then you should be able to point the client to the IP address 192.168.1.93:1 (or whatever the static IP address is of the network that you wish to connect the device to) and have full access to the Raspberry Pi.



Manage application installations

Do you need a graphical interface to install new programs? If so, then read on

If you're new to Linux, you may find using its built-in Apt package management tool a bit intimidating and confusing. The apt-get command is used for installing applications through the internet, connecting to the remote servers – called repositories – which house the programs as packages. But it is used through the terminal command prompt, which can be daunting, so we need an alternative: a desktop environment interface method of getting hold of packages.

This is where Synaptic comes in. Synaptic is a friendly-looking graphical interface to the apt-get terminal command which allows you to manage your application installations, and removals, through the already familiar desktop environment. Think of it as a kind of online shop where you can pick and choose the programs you want and have them downloaded and installed onto your Raspberry Pi without you having to drop into the terminal.

It's remarkably easy to install and set up, and within a few minutes you should be completely at home with its many intricacies. However, the terminal command line is the more powerful tool, so if you can, it's worth making the effort to try to learn how to use it.

In the meantime, though, settle down to some GUI-based program management. There's always time enough to dip your toes into the terminal at a later date.

Resources

Synaptic:
www.nongnyu.org/synaptic/

01 Update the system

Unfortunately, if you have an aversion to dropping into the command-line terminal, then you're going to be stuck at the first step. Before we install anything, we need to make sure that the Raspberry Pi is fully updated and any existing packages are upgraded. Simply enter the following into the LXTerminal:

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

02 Installing Synaptic

To install Synaptic, you'll first need to enter the LXTerminal and run the following command – don't forget to type Y to any prompts asking you to accept the installation:

```
sudo apt-get install synaptic
```

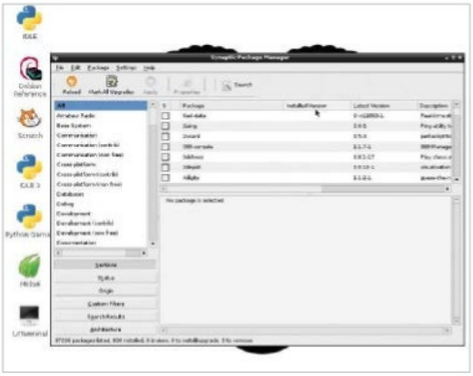
03 Running Synaptic – Part 1

In essence, that's all you need to do. Synaptic is now installed and ready to use. However, due to its complexity, there may be some bugs that need ironing out first, so it's best to follow these steps. To test if Synaptic is working okay, first enter the following command into the terminal:

```
gksudo synaptic
```

04 Running Synaptic – Part 2

You should be now looking at the Synaptic program window, where you can scroll through the list of available programs and click on



each to download and install. Now we need to test whether it will run from the LXDE menu. Click on the icon in the bottom left, then go to Preferences>Synaptic Package Manager in the menu.

05 Running Synaptic – Part 3

Running Synaptic from the LXDE menu results in an error. Don't panic, however: all it's doing is asking for a password. This is due to permissions, as the menu command to run Synaptic is expecting the same command as you entered with gksudo. Enter the following password into the box:

```
raspberry
```

06 Fixing Synaptic – Option 1

This will temporarily fix the issue, but to permanently resolve it, do one of the following. First, right-click the Synaptic icon in the menu and left-click Properties. In the Command text box, change the text to add the gksudo command. So instead of 'synaptic-pkexec', it will read:

```
gksudo synaptic-pkexec
```

07 Fixing Synaptic – Option 2

The second, and best, option is to drop back into the terminal and alter the way in which the program is executed from the menu. All that's needed is to change one line to another, so that the gksudo command is again used instead of the plain synaptic-pkexec. From the terminal, type:

```
sudo nano /usr/share/applications/
synaptic.desktop
```

Change 'Exec=synaptic-pkexec' to…

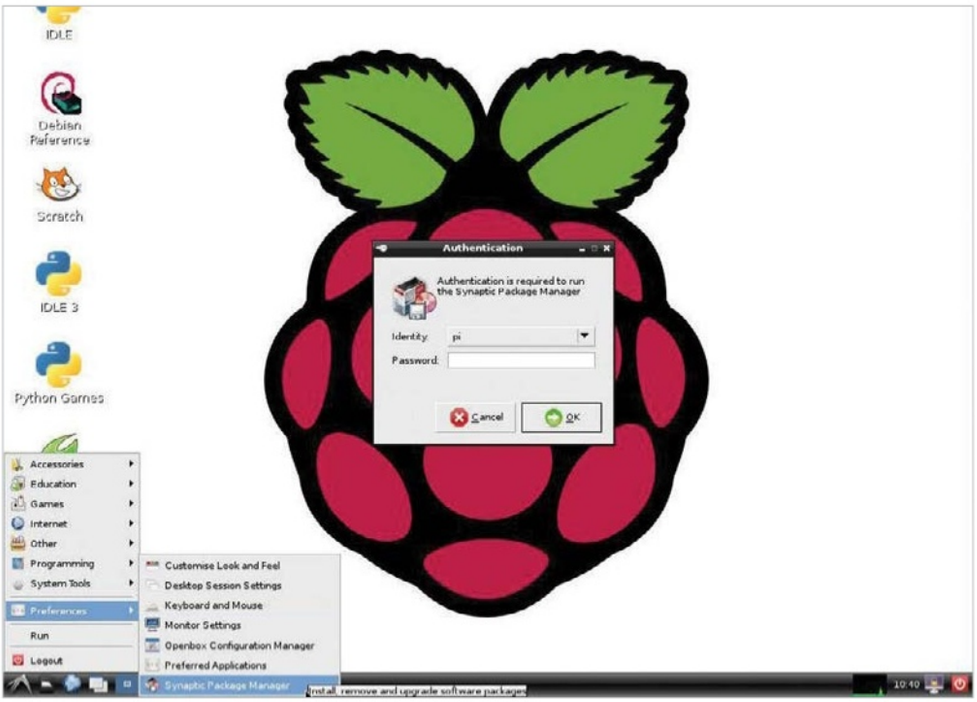
```
Exec=gksudo synaptic-pkexec
```

08 Synaptic fully working

After you've entered those changes, exit nano via Ctrl+X, followed by Y to accept the changes, and then press Enter a couple of times to get back to the command prompt. You can now launch Synaptic from the menu, or by entering the following command when you're in the terminal:

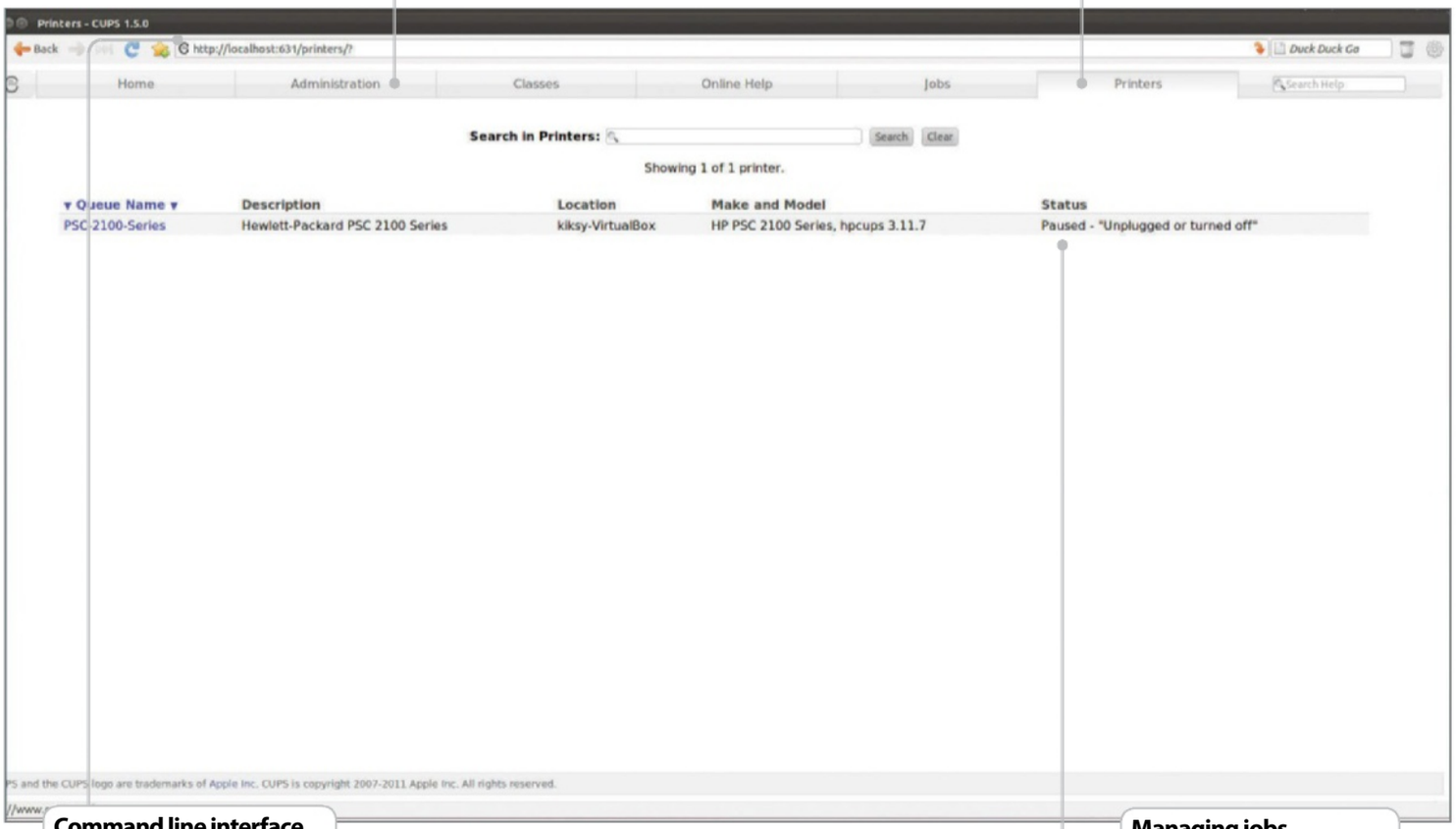
```
gksudo synaptic
```

"A friendly-looking graphical interface to apt-get"



Security
CUPS has a basic authentication process for users, meaning that you have to log in before gaining any access to the main administration panel

Using network printers
CUPS allows you to manage all of the printers on your network in one single location, which makes things straightforward and means you can keep better track of them



Command line interface
You can also use CUPS using just the command line, which is great if you want to script up a cron job to automatically print out reports at a certain time

Managing jobs
You can use your Pi as a handy print server and monitor, and pause and resume jobs from anywhere else on your network using the browser interface

Set up a printer on your Pi

Learn how to print out documents and photos using your Raspberry Pi running Raspbian OS

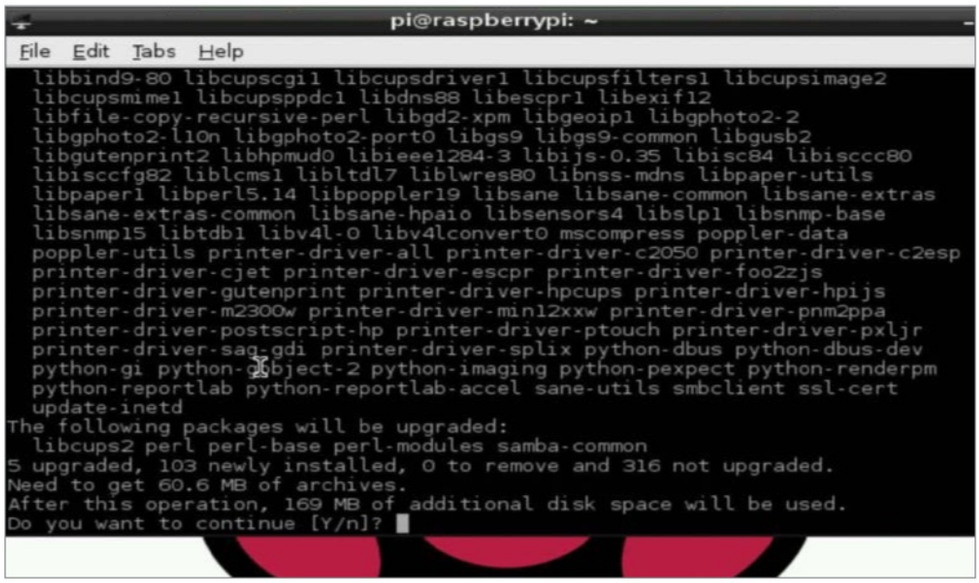
With its diminutive size, the Pi makes a great portable computer; just pop it in a bag and plug it in to any TV or projector with an HDMI port. One reason people carry personal computers around with them is for basic word-processing. Of course, the Pi with Raspbian shines at this, as it has access to hundreds of software packages via APT (Advance Packaging Tool).

Once you have finished working on your latest novel, shopping list, or presentation, there's a good chance you'll want to print it. Thankfully setting up printing in Raspbian is straightforward, even if it's not quite plug 'n' play like it can be within Windows or OSX. This guide will take you through all of the steps needed to install a printer using CUPS (Common Unix Printing System).

CUPS was first released in 1999, and in 2002 was incorporated into OSX. Later in 2007 Apple purchased the code. More information can be found at **www.cups.org** if you're interested. The process will involve a little bit of command-line work to begin with, but then CUPS offers a nice web-based interface with which to manage your printer and jobs. As a next step on from this tutorial, you could set up your Pi to allow remote access from your network or even the internet. From there you could then control print jobs from anywhere in the world.

Resources

Raspbian:
www.raspbian.org



■ Step 2: Install CUPS in order to get the printing process underway

01 Get Pi ready

Firstly, make sure your Pi Raspbian installation is up and running and that you don't have any background programs running. Also make sure the Pi has internet access. Then open up a new terminal window, which is done by clicking LXTerminal. To update the APT manager run:

```
sudo apt-get update
```

02 Install CUPS

CUPS is the main package that we need to install in order to get printing up and running. Installation is a simple case of listing the packages we need using APT.

```
sudo apt-get install cups
```

03 Install Nano

We need to edit some text files, and if your Raspbian installation doesn't have it already, it's a good idea to grab Nano. Nano is an easy-to-use text editor that runs from within a terminal window. Installation is again done using APT.

```
sudo apt-get install nano
```

04 Edit config

Now to edit the CUPS config files. Under: # Default authentication type, when authentication is required, add:

```
# Show shared printers on the local network.
Browsing On
BrowseOrder allow,deny
BrowseAllow @LOCAL
```

To make the printer accessible over the network:

```
sudo nano /etc/cups/cups.d.conf
sudo usermod -a -G lpadmin pi
```

05 Make accessible

Continuing on the config file, change the section under # Restrict access to the server to the following code:

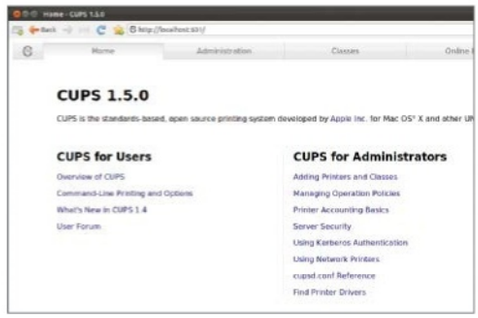
```
<Location/>
Order allow,deny
Allow localhost
Allow 172.20.22.*
</Location>
```

Once that's done you can restart the server with: #/etc/init.d/cupsys restart

06 Access locally

Now on the Pi open up a browser and go to the address below. From there you can then choose 'add printer'. When asked, add your username and password. You should then see any network printers or attached printers shown on the list.

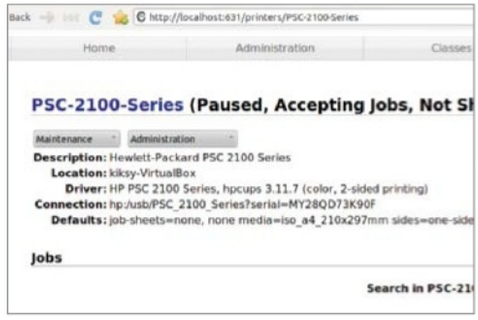
```
http://localhost:631/
```



■ Step 6: Follow some simple steps and you will see any network printers or attached printers on the list

07 Test the printer

To print a test page, click on 'administration' and then 'manage printers'. You should see yours listed. Click on the name of the printer to see more information about it. From the maintenance drop-down menu you should then be able to choose 'print test page'.

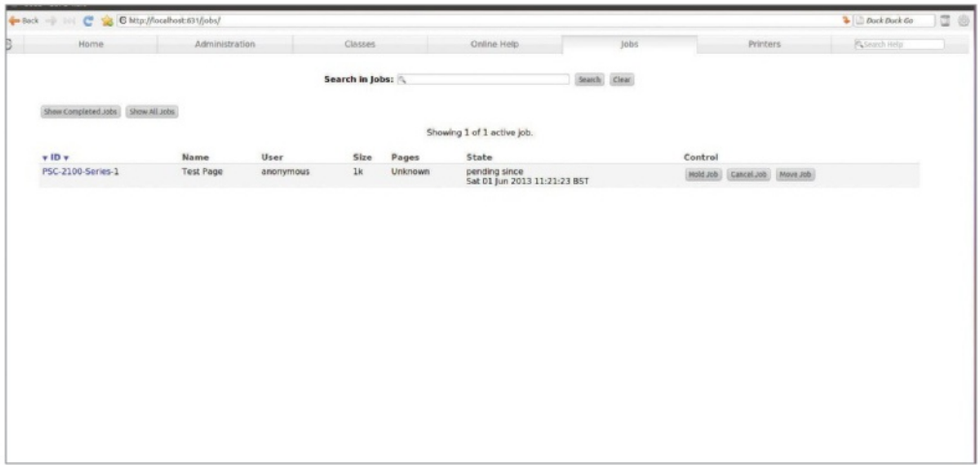


■ Step 7: It's useful to print a test page in order to see if your selected printer is working correctly

08 View jobs

After printing your test page, click on the 'jobs' tab, and you should see the print test listed in the queue. As the Pi doesn't have huge CPU power or RAM, printing can sometimes take a few minutes, so give it time.

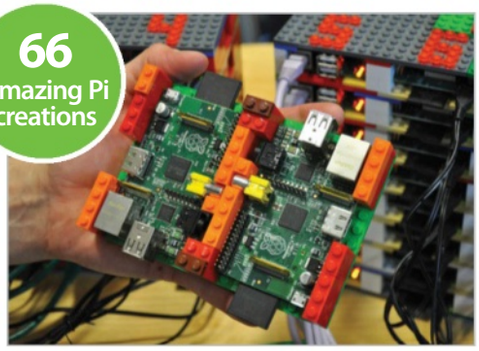
```
sudo apt-get update
```



■ Step 8: You should see the print test listed in the queue under the 'jobs' tab. Be patient when printing

Projects

Get inspired by others' projects and learn to create your own

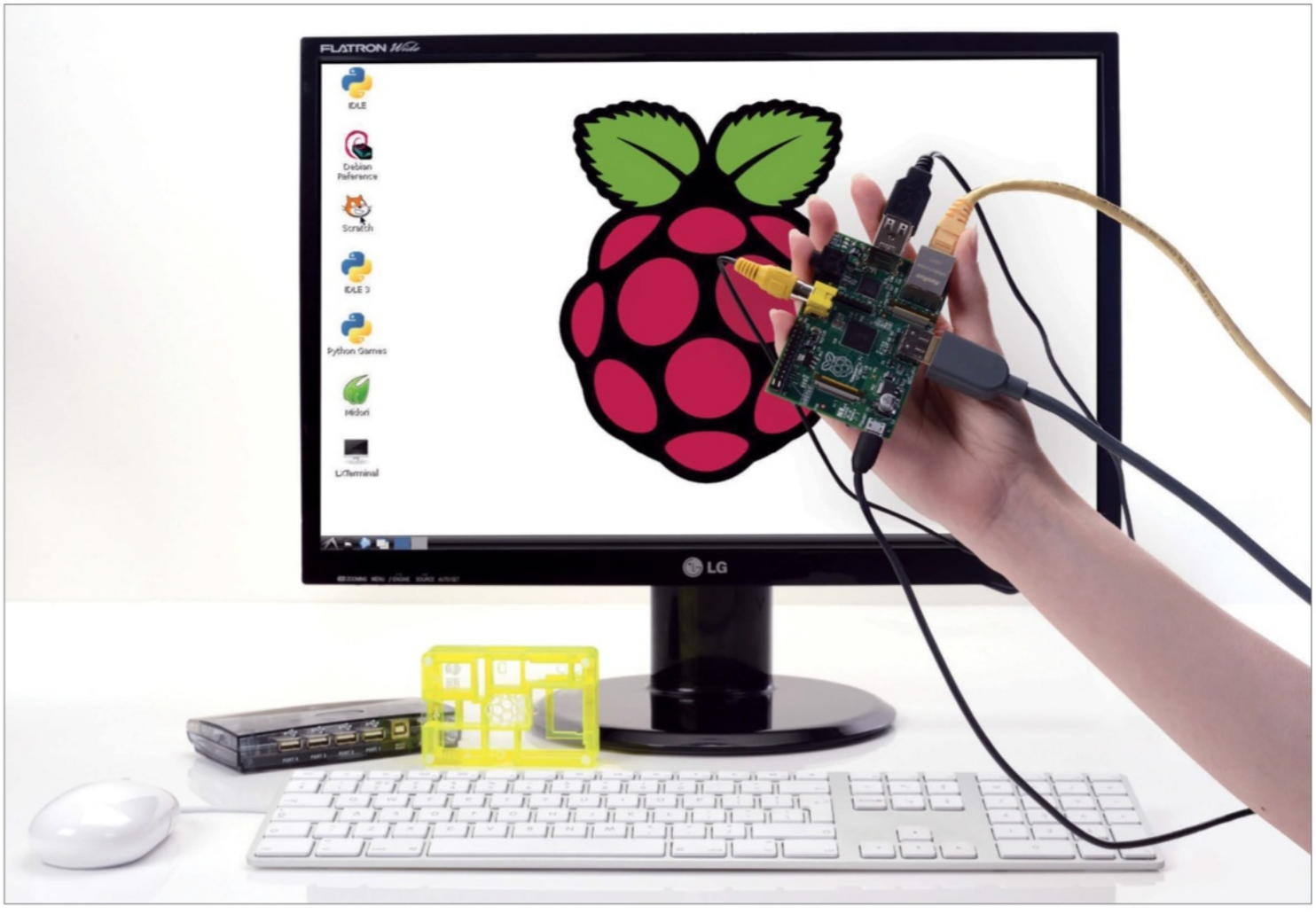


"Turn your Raspberry Pi into an array of useful tools and fun projects with ease"

"Discover how you can let the Pi be the hub of your digital life"

A guide to Pi projects

This is where things start to get really interesting. Learn how you can build some truly amazing machines



The Raspberry Pi is an ideal foundation for your building your own projects. It's cheap, powerful, has an open environment aimed at development and is very portable. But perhaps the most exciting aspect of the Pi is the General Purpose Input/Output (GPIO) header. These pins are there for the sole purpose of allowing you to interact with the environment around you. When used with Python, the GPIO instantly opens the Pi up to the physical world around it.

Before you get started on a project, we would recommend a little shopping. These items can all be found very cheaply (the most expensive is £10) from popular electronics sites and online auctions.

Number one on your list is a 'breadboard', aka a prototype board. These are simple solderless boards that allow you to create your own electronic circuits. Number two is male-to-female and male-to-male breadboard wire, a must for wiring up your contraptions. Number three is a beginner's set of electronic components. These will have all the common resistors and LEDs you'll need before you even know you need them. Last on the list is a small low-power LCD – we like the HD44780.

Building external projects

Broadly speaking, your Raspberry Pi project will fall into one or more of three categories: external, software or embedded projects.

External projects are when we build our own electronic circuits, connect them to the Pi via the GPIO and write software (most likely in Python) to communicate with them. For us, these are the most exciting projects you can build; they bring together Linux, programming and electronic knowledge in such a way that just wasn't possible for most home users before the Pi.

If you're new to any of these fields then don't be put off. With a little help from a great community and some fantastic tutorials, you'll soon be building some basic circuits and controlling them with software correctly. Then your projects will get bigger and you'll be starting your own from scratch. Truly, the only limitation is your imagination.

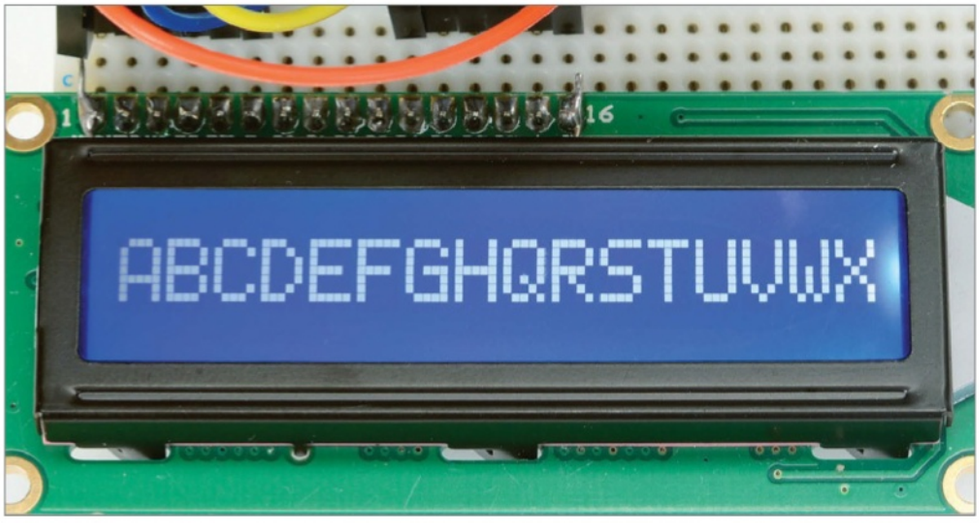
To whet your appetite you'll find some great examples in these pages. The Twitter-powered lamp tutorial is a lot simpler than it sounds and is a fantastic project to start with – it is a great stepping

With a few components and some basic bit of programming you can convert your Raspberry Pi into a secure file server (p. 80)

Ever missed the nostalgia of retro gaming? Use your Pi and an XBox controller to revisit classic games that you played as a child (p. 98)

Detect motion and take a photo of the cause for home security or time-lapse photography purposes (p. 94)

"Projects such as these are the perfect way to integrate the modern computing revolution into your home"



With a few different components, your Pi will accurately display weather information (p. 102)

stone to bigger and better things. Or why not combine your Pi with our nation's favourite topic of conversation and build a weather station? Look no further, these tutorials are sure to bring the inventor out in all of us.

Building software projects

Many of the projects we want (or need) to develop won't require the full GPIO capabilities of the Pi and will just use it purely as a computer to write and run our own applications. These are software projects. The Pi is more suited to this task than most desktops or laptops as it has been specifically designed as a development and learning platform. From the child-friendly Scratch programming environment to the powerful and diverse Python language, there's a starting point for everyone here regardless of previous experience or ability.

Whether it's recreating classic games or creating your own, writing a script to make system administration easier or designing an application to fill a hole in the market; whatever software project you have in mind, you'll find the tutorials here a great place to start. To get you started why not integrate your Pi into your network and create a Tweet Robot to automatically tweet on your behalf?

Building embedded projects

Think of embedded projects as a hybrid project where the Pi is used to bridge the gap between two devices or add new functionality to a device.

The perfect example of this is the Media Centre project. HDTVs are yesterday's news – everyone wants a smart TV now. No problem: the Broadcom chip on the Pi can easily play high-definition video and output this via the HDMI port. Combine that with the fact the Pi can even be powered from the USB port of many TVs and suddenly you have a cheap and powerful upgrade your existing HDTV. This project brings out the Pi's full potential.

Want to expand your network and create a centralised storage space for the whole family? Then check out the File Server project to see just how easy the Raspberry Pi makes this.

Embedded projects such as these are the perfect way to integrate the modern computing revolution into your home.

We've got these projects and a whole lot more, including a list of inspirational ones to start you off.

8 INSPIRATIONAL PI PROJECTS

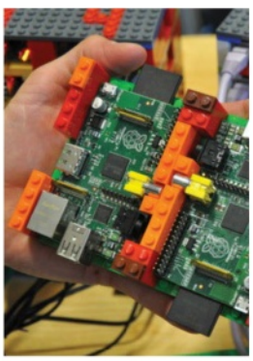
The £25 marvel has changed the way people approach computing, and here we're going to look at some of the best projects you can complete yourself

As portable computing goes, the Raspberry Pi could hardly be bettered. Small enough to slip inside a pocket, it can go anywhere and everywhere with you. Yet to use the Pi as a standard Linux machine kind of misses the point – or, at the very least, the opportunities afforded by this small-form-factor, high-spec wonder.

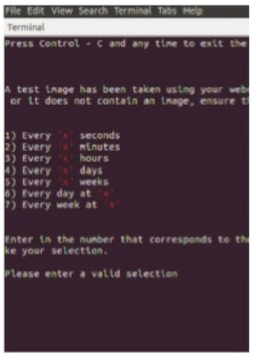
Since its initial release, this inexpensive machine – produced as an educational plaything – has taken centre stage for a whole host of projects. Some have satisfied a craving for fun; others for exploration and indulgence. More still have satisfied the current trend for performing real-world tasks at the lowest possible price. The Pi is the recession buster with the potential to empower a future generation (as one schoolboy emphatically showed).

Here we present a selection of those projects. Each of them, we believe, encapsulates all that is good about the Pi, making great use of two things: the machine itself, with its tiny credit card size; but above all, the bold imagination of the creators. And the drive and determination to keep going to realise those dreams, of course, but that would be three things.

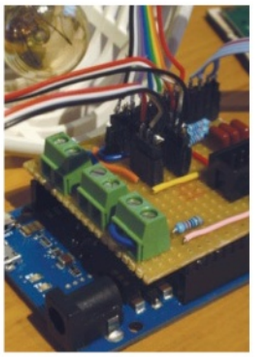
Over the next few pages, the creators of these projects talk us through their designs. We've packed in some incredible images of the projects, and we'll also tell you where you can find tutorial instructions, where available, to allow you to replicate them yourself. But more than all of this, we hope it provides you with inspiration for your own projects. You never know – you may see your creation here in our next edition!





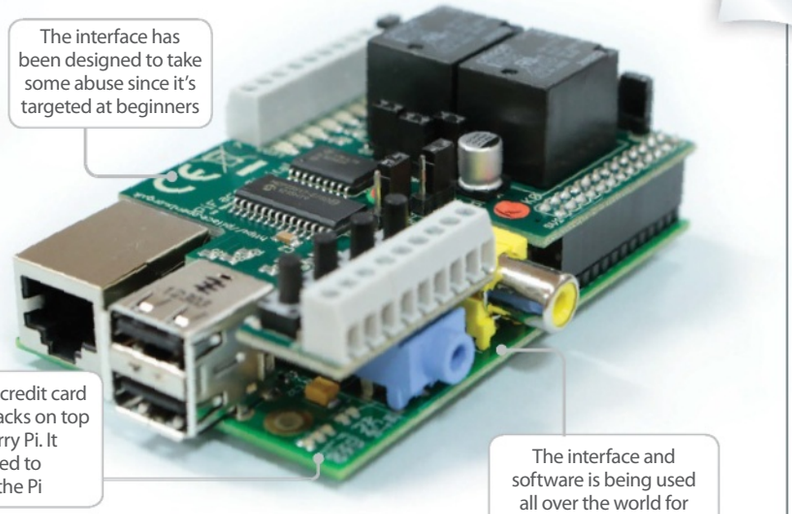








Pi-Face – the digital interface

Discover how you can let the Raspberry Pi be the hub to your digital life



Being able to control the real world using a computer is the sort of thing that we wanted back in the mid-Eighties when watching *Back To The Future*. There have been lots of solutions for this issue, but none has been as cost effective and wide open as the Pi-Face Digital Interface created by scientists at Manchester University.

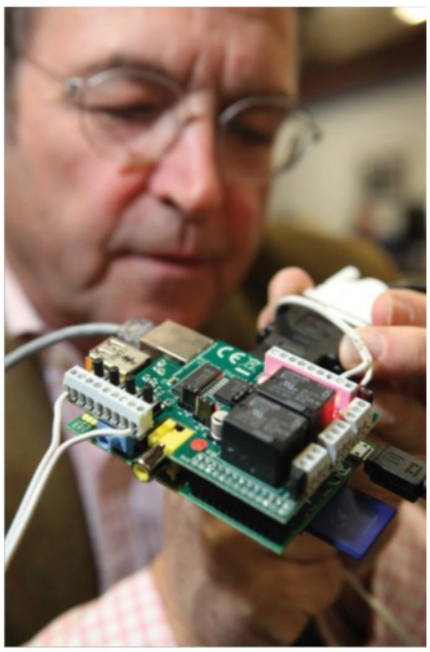
"I'd been working on getting more people interested in computing for a few years," explains Dr Andrew Robinson, the brains behind the device. "I was really excited to hear Raspberry Pi was coming along, but worried that people wouldn't know what to do with it. I saw the interface as a missing link, to allow people to put the Raspberry Pi where they wouldn't put a PC and connect it up to the world. So I designed Pi-Face to be ready for when the Raspberry Pi came out, and planned some fun activities – like the twittering chicken."

The first design was published on 15 December 2011 and the interface board is the same size as the Raspberry Pi. It slots onto the GPIO pins and has screw terminals so it's very easy to connect up switches to sense things and lights/motors for the Raspberry Pi to control. It's very easy to program in Scratch or Python too. It's not just about the interface, either. The project is also about creating the fun activities and tutorials to support using the Raspberry Pi.

"It took four prototypes to get right," says Dr Robinson. "The first one has a yellow wire tacked on because a connection got missed. It was a bit like the early Raspberry Pis that also had bits of wire tacked on by hand."

Apart form that and any other minor mishaps, it has all gone rather smoothly, however. "The interface has been designed to take some abuse since it's targeted at beginners," he adds. "We have connected things up wrongly a few times, but so far not seen any smoke!"

"On the workshop side of things, the children have really and truly loved it. We have a harder time when we run the workshops for teachers! Still we've learned from it."



The hardware architecture was altered during the project so that it would work with mobiles (with Android and USB OTG) and desktop PCs too. It works with Raspberry Pi through SPI, but also through USB.

Pi-Face integrates with existing tools already prevalent in the classroom, such as Scratch and Python

TRY IT YOURSELF

The Kit: Pi-Face is available complete from Farnell for £20, which is cheaper than buying the parts separately – piregistration.element14.com/signup.html

The Knowhow:
pi.cs.man.ac.uk/interface.htm

MAKER PROFILE: DR ANDREW ROBINSON

Andrew Robinson can trace his first interest in electronics back to making a model lighthouse aged five. At the University of Manchester, Andrew did a PhD in low power processor architectures (which included looking at ways to make ARM processors more efficient). While working in the group Andrew became interested in 'public engagement', to get more people interested in Computer Science.

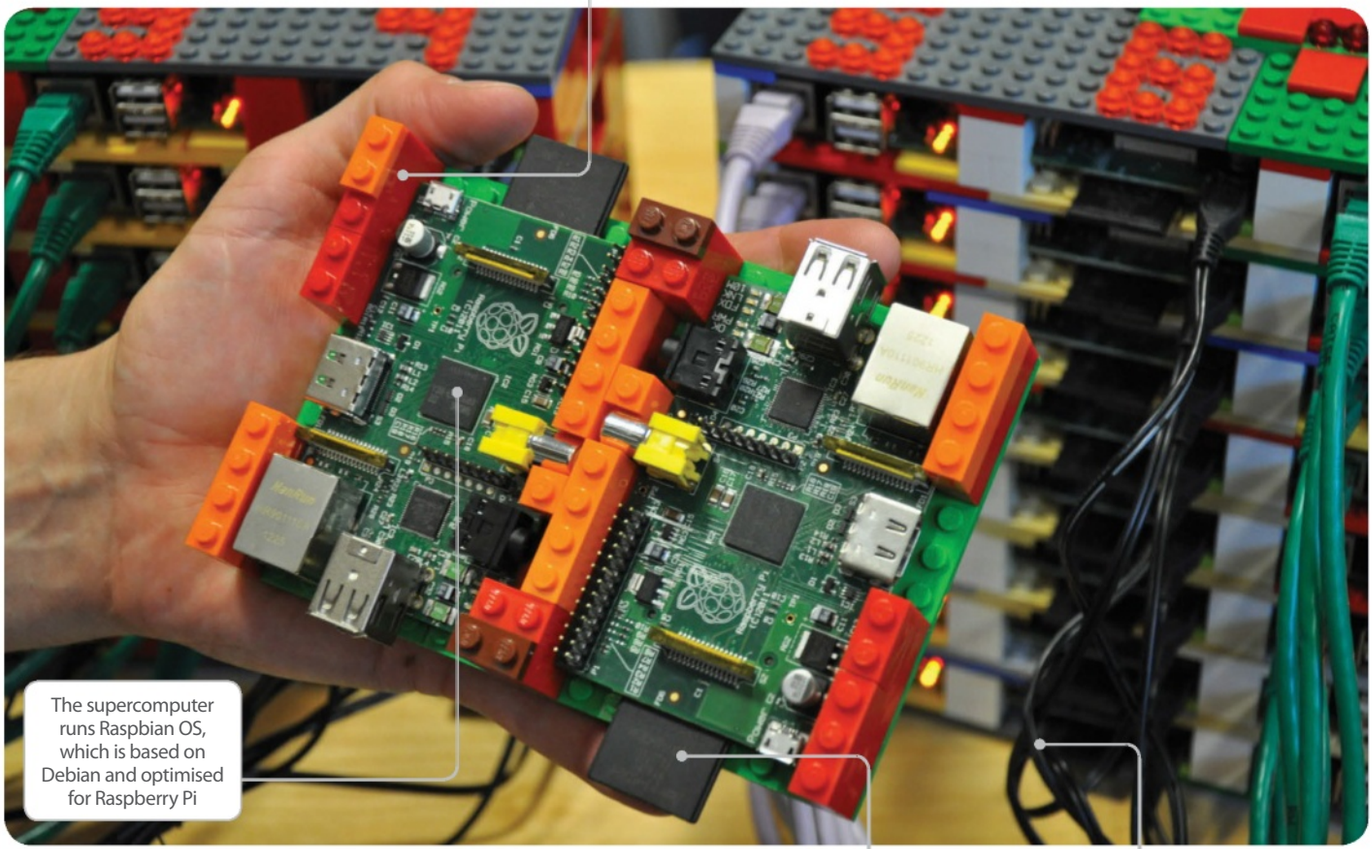
The Raspberry Pi Supercomputer

Take some Lego and 64 Pis for a delicious slice of processing power

Since the ultimate aim of the Raspberry Pi is to encourage children to experiment with computers and understand their inner joy, the supercomputer project built by computational engineers at the University of Southampton could not be a better example of the magical things being done with this miniature marvel.

The creation by Professor Simon Cox and his team cost less than £2,500 to build, excluding the switches, but it also had a special ingredient: Prof Cox's six-year-old son, James.

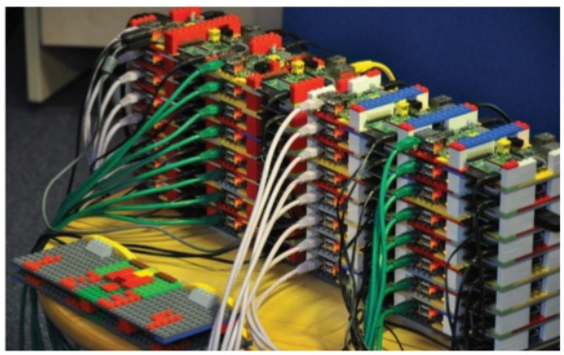
It was while playing around with a Pi with his son that the supercomputer expert decided it would be an interesting experiment to buy 64 of the machines and produce something rather spectacular. And it was the young boy who provided specialist support on Lego and system testing, providing an eye-catching aspect to the entire project. Running off a single 13 amp mains socket and using MPI to communicate



MAKER PROFILE: **PROF SIMON COX**

Simon Cox is Professor of Computational Methods at the University of Southampton. His research is about applying and developing high-performance computing and big data to tackle problems in science and engineering. The university of has a powerful 12,000-core Intel-based supercomputer which cost millions of pounds.

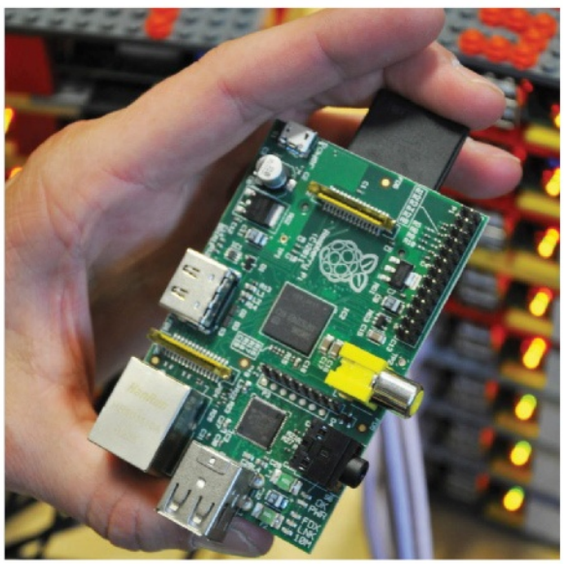
"James used his design skills to build the racking out of little plastic bricks"



Lego features highly in the project, but so does research. Although the supercomputer uses an ordinary downloadable system image from the Raspberry Pi website, the performance and various other parts of the firmware and drivers have been improved by a factor of 50 or more.

between nodes using Ethernet, it offers a staggering amount of power when you consider the low cost.

Professor Cox had been hugely impressed straight away by how great the Raspberry Pi was for playing around with electronics using the GPIO connectors. "The fact you could turn LEDs on and off and link a little Python computer program to a bar graph LED caught the imagination of myself and my son who built a Lego case for it," he tells us. So he waited for a large supply of Pis to became available. "We then had to decide on the network switches. My background researching supercomputers and IT meant I knew you could spend a lot of money on switches. We begged, stole and borrowed some old switches that were being decommissioned from our computing service, which meant we could link them together. I got some power over Ethernet switches from another project I was working on."



Since the drivers for the Pi's video card are open source, the potential to use that video chip for processing opens up 24 gigaflops of general-purpose computer performance.

The project started when Prof Simon Cox and his son, James, began to play with a Pi

James used his design skills to build the racking out of little plastic bricks before testing it using both Python and Scratch. But one of the most time-consuming processes was getting all of the images set up. Professor Cox found it was one thing to download one image on to one SD card, but quite another doing 64 of them.

"The Pi, for the first time, has meant you can assemble a supercomputer for a couple of thousand of pounds. And if you take just four of the units, suddenly you are at £100 and that means that large-scale supercomputing, or the principle of it, can be seen in schools. That's very special."

TRY IT YOURSELF

The Kit:

- 64 Raspberry Pis Model B/256MB = £1,475.20
- 64 Kingston Ultimate X 16GB Class 10 (SD10G2/16GB) = £622.72
- 64 metres of CAT 5E (misc colours) = £90.88
- 64 micro-USB power supply adaptors, UK, 1.2A = £244.48

The Knowhow:

www.soton.ac.uk/~sjc/raspberrypi

Plus items that were in the lab:

- 3 Netgear ProSafe 24-port smart switches with PoE / 192W total 4 SFP (GS724TP) = £269.99 each
- DrayTek Vigor 2820N router
- Keyboard
- Monitor
- Mouse
- HDMI-to-monitor cable
- Three CAT 5E cables to connect the switches together

Picade arcade cabinet

No need to stand around in arcades – bring the action to your home

Games have often been said to be a key driver of technology and so we were extremely excited to see the Picade project when it emerged on Kickstarter. Created by Jonathan Williamson and Paul Beech, it looks like a mini, retro-style cabinet and comes in kit form. All people need to get up and running with this project is a screwdriver, a pair of pliers and an hour of time to put it together.

"The idea had been in our heads for a while," says Jonathan. "Paul and I have been through a number of startups and we had frequently seen a JAMMA cabinet in the office. It seemed the thing to have. We thought that only people with too much money or time on their hands were getting enjoyment out of them, so we decided it would be good to build one ourselves."

The pair have been interested in technology for a long time and the Picade has been an accumulation of knowledge. From reading about joysticks to monitors, all of those ideas have come together. "We've also benefited from Kickstarter and globalisation," says Paul. "Having the Pi was amazing."

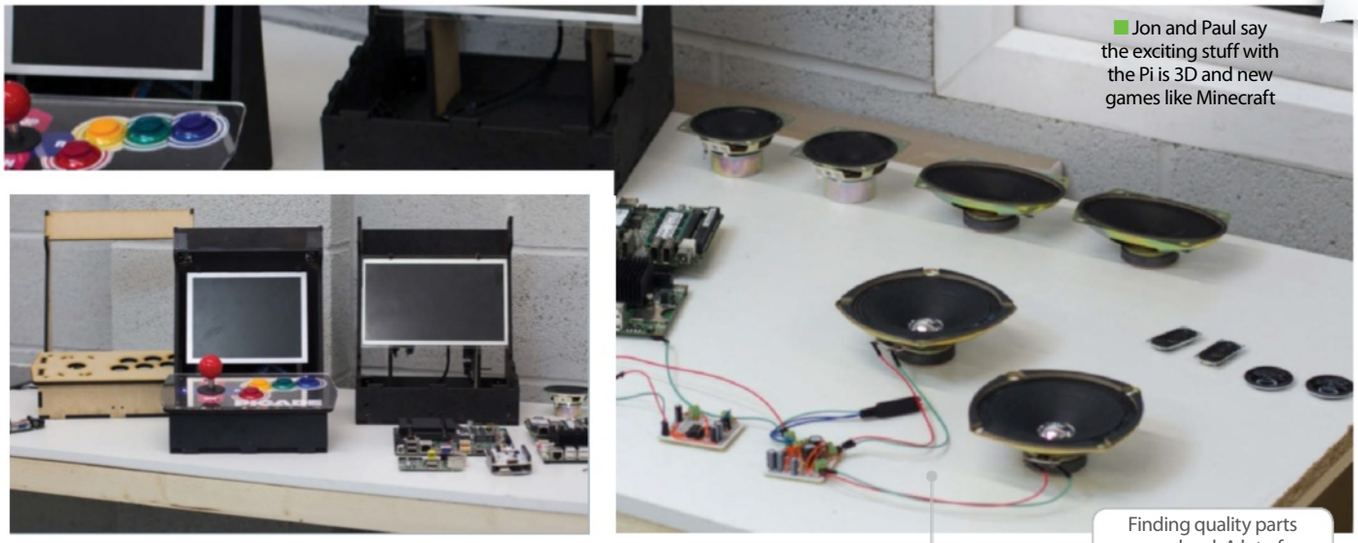
As if to underline just how the project has turned heads, Ian Stewart, the founder of Sheffield's Gremlin Graphics – a games developer big in the

Ian Stewart from Gremlin Graphics stepped forward and allowed his old games to be available on the Picade

"The Picade suddenly becomes a great way to play legal games"

MAKER PROFILE: **JONATHAN WILLIAMSON AND PAUL BEECH**

Paul is a designer, hacker and maker. He designed the Raspberry Pi logo and is the brand and design lead for the Foundation. Jon is a software guy and electronics hobbyist. He is the co-founder and technical lead for Netcopy (a digital archiving solution).



Jon and Paul say the exciting stuff with the Pi is 3D and new games like Minecraft

There is no profit margin on this project. It is something the pair just wanted to build

Finding quality parts was hard. A lot of components had to come from the Far East

Eighties and early Nineties – has allowed the use of the firm's old-school games. "This means we can provide a games machine that can be used straight away," says Paul. "And when you look at the Pi Store, the Picade suddenly becomes a great way to play legal games."

Jonathan is now starting to get very excited about the future. "It's the 3D games which I am loving," he enthuses. "It will show off the Pi as a great gaming machine." However, there have been some problems encountered by the pair.

"Screens," says Paul. "There are tons of models out there and some of these are 3mm in width. You can't believe how thin they are. We had to look at how easy they are to mount, and separating wheat from the chaff. There is always a delay in stock too and we have to get on with other stuff while waiting for things to appear. Trying to get quality parts on time is difficult. A lot of components have to come from the East, such as arcade joysticks. But it's been brilliant to work on."

The creators noticed lots of startups had JAMMA cabinets in the break room and associated them with people with a lot of money. Picade opens up cabinets to the masses

Getting a quality screen was important for Jon and Paul. Delays in stock caused problems

Both Paul and Jon have accumulated knowledge of items from joysticks to monitors following years of reading and research

TRY IT YOURSELF

The Kit:

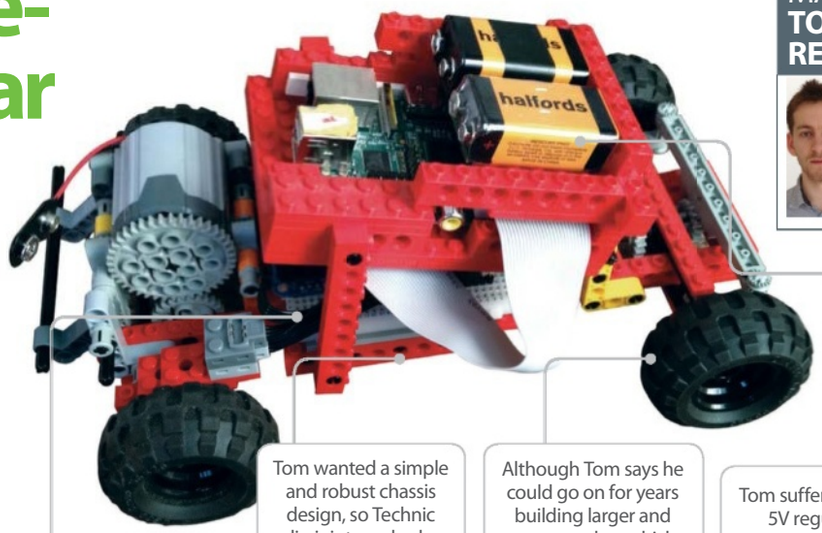
- Cabinet panels and fasteners
- LCD panel mount with protective overlay
- LCD panel and driver board with inputs (at least 8" for the Picade Mini and 12" for the Picade)
- Amplifier and speakers
- 3.5mm stereo panel-mounted headphone socket
- Panel-mounted video input socket (allows you to use your Picade as a second display for your computer or laptop)
- A proper arcade stick
- Illuminated microswitch arcade buttons (at least four on the Picade Mini and six on the Picade)
- All other required components and cables

The Knowhow: www.kickstarter.com/projects/pimoroni/picade-the-arcade-cabinet-kit-for-your-raspberry-p

Lego remote-controlled car

A Pi-powered self-contained remote-controlled car guided by an Xbox controller

MAKER: **TOM REES**



The steering is controlled via a high-torque 180-degree TowerPro servo and a Lego gear is attached with superglue

Tom wanted a simple and robust chassis design, so Technic clip joints and axles are used through everything. Servos and motors need to be held together tightly and two motors are required

Although Tom says he could go on for years building larger and more complex vehicles with increased power, four-wheel drive, weaponry, cameras, caterpillar treads, he is keen to invent something new

Tom suffered dodgy 5V regulators and overloaded components. He also found that the cheap micro-USB cable going into the Pi could barely carry half an amp and had to be replaced

Very often what you didn't have in your childhood inspires you in adulthood. That was the case with the Lego remote-controlled car created by Tom Rees using the Raspberry Pi. Having always wanted a remote-controlled car when he was young, the desire never left him.

And thanks to the power of the Pi, he was also able to go one better. Rather than build something controlled on a long tether to a computer or via Wi-Fi from a laptop, Tom could integrate the computer into the car itself. This means it became a 'real' remote-controlled car comprising just the vehicle and the controller.

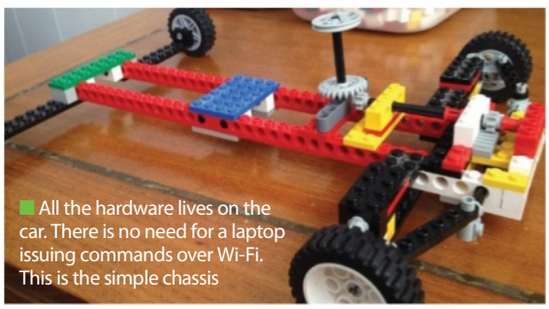
The original plan was to build an RC Lego DeLorean, but all he could produce was an ugly white box. He then came up with different prototypes, most of which looked nothing like a car and were essentially motors, servos and LEDs attached to Technic towers. "I remember the very first thing I built was a pair of LEDs which were turned on and off by the Xbox controller," he says. "It's important to begin with the simplest possible proof-of-concept to show all parts of the system working."

To even begin to prototype, jumper wires, breadboards, a multimeter and lots of batteries were needed, plus a selection of resistors, capacitors, LEDs and a soldering iron. Once Tom had all the necessary supplies, he just decided to dive in. He learned as he went along.

"I had plenty of disasters," he admits. "The cheap inductors I used in the 5V power regulator kept catching fire because the Pi consumes so much current. Smoke pouring out of the Lego Technic holes is pretty terrifying." Trickiest of all was powering the electronics aboard the car. "The Raspberry Pi has strict tolerances on its 5V input, and it will consume well over 1 amp when the Microsoft USB peripheral is attached," he says. But he is more than pleased with the result.

"Most stunning has been the reaction to this project. I knew I wanted to do something original with my Pi, to push the envelope very slightly, and be a little more imaginative than another 'internet-controlled LEDs' project. It's seen a great reaction and I'm very pleased to see some projects inspired by this one."

"The original plan was to build an RC Lego DeLorean, but all he could produce was an ugly white box"



All the hardware lives on the car. There is no need for a laptop issuing commands over Wi-Fi. This is the simple chassis

The car is controlled via a wireless Xbox 360 controller

Extra electronics are needed on top of the Lego and the Raspberry Pi. The Cobbler breakout kit from Adafruit broke out the power, GPIO, I2C and SPI pins from the 26-pin header onto a solderless breadboard

Light painting with the Pi

Artistic images that simply astound are possible with the power of Pi

There is seldom a more stunning effect than light painting. It uses long exposure times with lights in motion to produce amazing images. When Phil Burgess wanted to produce a machine that could create effective artistic images, he had a hunch that the Raspberry Pi would be able to streamline the process, not least because of the memory it could provide for the project.

The concept had been stewing for months, but no existing device made for a really satisfactory solution, Phil says, and by that he meant something that could be shared and easily built by others. The open source electronic prototyping platform Arduino was considered and Phil says it could handle the core task but it requires a ton of preprocessing and staging of the image data in flash storage. The Raspberry Pi could handle this – and because it is so inexpensive, it got over the worry of actually getting it damaged.

An Arduino could handle the core task, but required a ton of preprocessing and staging of the image data in flash storage. Another thought by Phil was to use his laptop and FTDI chip, but that machine is rather precious to him and he was not eager to strap it on a bike. The Pi was a middle ground – the horsepower and programming ease of the laptop, but the throwaway inexpensive nature of it. As you can see, the results are astounding

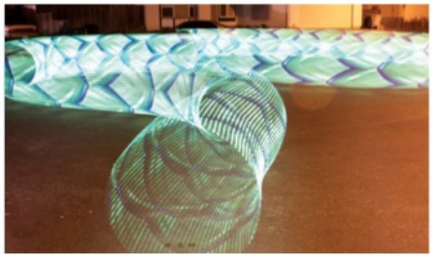
"So far we have created the prototype," Phil tells us. "The next version will be smaller and lighter with a proper battery pack. At the time I was just scrambling among parts in the garage and relied on an enormous camping inverter to quickly solve the problem at hand." Things went well too. "All the hardware and software was amazingly co-operative and everything fell into place," Phil adds.

The trickiest part of the project was the documentation. "Explaining things as concisely as possible, cleaning up the code to be presentable, providing clear diagrams and such is always hard," comments Phil. "Good documentation can elevate even a modest project into something desirable, but no amount of 'coolness' in a project can cover for a lousy explanation."

And there were other considerations. A 5V DC power supply was needed because the LED strip draws so much more power than the micro-USB connector can provide. A 26-pin IDC cable was also sacrificed to create a purpose-built cable between the Raspberry Pi GPIO header, LED strip and power supply in order to make the whole project a lot more robust, something that was considered a necessity.

Phil also says Python warrants a special mention: "My first attempts at communicating with the LED strip were in C, but if this project was to be shared, C with makefiles and extra libraries and such would add complexity. In 48 hours, starting with *zero* Python exposure, it was possible to get this not just working, but actually pretty well optimised as I got a better handle on the language. And it's all in one source file, easily shared."

The bike. Other than having the basic concept already, there was no planning: Phil just went straight into implementation, which took about two days



Phil just scrambled among parts in the garage and relied on an enormous camping inverter to get these effects

TRY IT YOURSELF

The Kit:

- Raspberry Pi = £25
- Adafruit Pi Cobbler breakout kit = $7.95
- Digital RGB LED weatherproof strip 32 LED = $29.95
- 4-pin JST SM receptacle cable = $1.50
- 4-pin JST SM plug cable = $1.50
- 5V 10A switching power supply = $25
- 5V 2A (2000mA) switching power supply = $9.95
- Female DC power adaptor: 2.1mm jack to screw terminal block = $2
- Plus a bike and mounting hardware (PVC pipe, hula hoop, zip ties)

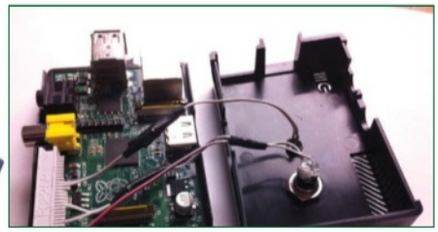
The Knowhow: learn.adafruit.com/light-painting-with-raspberry-pi

MAKER PROFILE: PHIL BURGESS

A Californian living in the San Francisco Bay area, Phil Burgess's roots in art and technology go back many decades. The light painting demo is the latest in a progression of LED projects inspired by artist/engineer Bill Bell in the 1980s. Today Phil develops kits and tutorials for Adafruit Industries, a firm at the centre of the open source and 'maker' movements.

The one-button audio player

Music to a grandma's ears… it's the easy-to-use audio player



The difficult part came in the automatic mounting of the USB flash drive

Michael Clemens's idea for his Raspberry Pi project came less out of hobbyism and more out of a solution to a very personal problem. His wife's grandmother is visually impaired and the couple find it difficult to keep her entertained. Listening to television is not ideal and she struggles to operate her CD player.

With her ninetieth birthday approaching, Michael sought a solution: to build an audiobook player that was inexpensive and easy to operate, with as few buttons as possible. It also needed to be open source. And after some careful consideration, it was obvious Raspberry Pi was the answer.

"The idea came quick, but planning it through and building it was kind of fragmented because my work includes travelling," he says. "When I knew what I wanted, I soldered the button and LED to a board that could be connected via the GPIO pins and then I used a little Python coding and configuration of standard Linux software for the rest. Not being an expert in electronics, this was the simplest way to achieve my goals."

The sound quality can crack when you play or resume an MP3. "But," Michael says, "that is a known problem in the Raspberry Pi community. The quality of the device itself is actually quite solid, but I'm thinking about building a new, large enclosure out of wood for the player together with the speakers."

As for how well the present was received, Michael laughs. "It was Christmas and she was obviously confused about all the people around her. On another day, we showed her how to use it and she was very happy about it."

TRY IT YOURSELF

The Kit:

- Raspberry Pi = €40
- ModMyPi enclosure = €12
- 1 button = €2
- 2 resistors (330 ohm, 10 kilo-ohm) = few cents
- 1 blue LED = few cents
- 1 (slow) 8GB SD card = €8
- Some wire = few cents
- A pair of speakers – he already had them, rough cost about €30

The Knowhow: blogs.fsfe.org/clemens/2012/10/30/the-one-button-audiobook-player

MAKER PROFILE: MICHAEL CLEMENS

Fascinated by electronics and mechanics from childhood, Michael began work as an IT apprentice and worked seven years as a UNIX and database administrator before becoming an IT security consultant.

Time-lapse photography with Pysnap

The competition-winning CLI tool created by a talented schoolboy

It was a contest for young programmers, but few would have expected to see quality that was produced by 12-year-old Aaron Hill, who ran away with first prize in the under-13 category. His success? Amazing software that turns a Raspberry Pi computer into a time-lapse camera. By connecting a USB camera to a Pi and running the software, it is possible to fine-tune the interval at which it takes pictures.

"My aunt has a motion-sensing camera set up in the woods near their hunting stand. That triggered the idea," Aaron tells us. "The first thing that I did was to find out how I'd get the program to take the pictures… I discovered a command-line program called Streamer, which was in the Debian and Ubuntu repos. I installed it and, after playing around with it, decided that I would use it for my project. Next, I started writing the code. I started off with just adding the option to choose different time intervals: seconds, minutes and hours. Later I got the idea to allow taking of pictures on a specific day, and a specific time."

One feature he initially liked but ended up discarding was having the program print out text one character at a time, instead of all at once. The effect wasn't very noticeable with a fast speed, and irritating to the user if it was slower, he says. However, there was enough to prove that Aaron was very much on to a good thing.

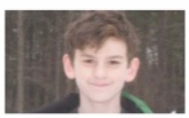
TRY IT YOURSELF

The Kit:

- PySnap software
- USB camera

The Knowhow: www.raspberrypi.org/archives/2544

MAKER PROFILE: AARON HILL



Aaron Hill is a 12-year-old schoolboy who loves to program and is very knowledgeable about computers.

Pi in the sky – 40km high photography

A record-breaking balloon mission records stunning near-space photos

It's hard to say what is more amazing – the photographs taken high up in space or Dave Akerman's Raspberry Pi/webcam/GPS/hydrogen balloon combination which enabled him to capture such magnificent shots.

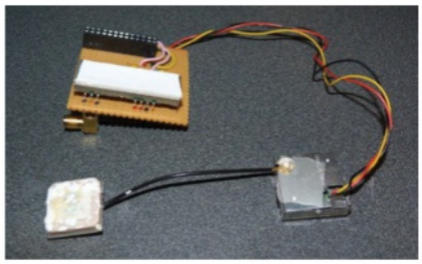
Not only are they believed to be the highest ever photographs transmitted live from an amateur device, the ability to thrust a Raspberry Pi into space is an amazing feat. And it barely took Dave any time at all to figure out, making it all the more extraordinary.

"The idea to combine the Pi and a weather balloon came after I took delivery of the Pi," he tells us. "I ordered it just because I was curious, and didn't really have any use planned for it. By then I'd been flying weather balloons with Arduino trackers for about a year, and thought that it would be fun to fly the Pi instead. The Arduino is ideal as a basic tracker, but the Pi does make it much easier to add things like live picture downloads."

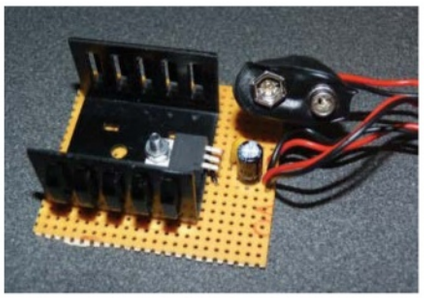
It took about an hour to wire up a radio transmitter and GPS, and another hour to port Dave's Arduino code over to the Pi. At that point he had a working Pi tracker and he was keen to fly it just to be the first to do so. But the weather remained poor and he couldn't launch for two weeks. That gave him time to add some more capability.

"Adding a live image download was obvious, since the Pi can interface very easily to a USB webcam," Dave says. "It was a couple of days' work to get that working. After that I had to make a regulator so the tracker could run straight from batteries."

He kept quiet about his project because he wanted to do it before anyone else. On top of that, he didn't say when the launch was planned either. "The launch itself was fairly stressful – as they usually are – but particularly this time because it was a new tracker and the first time I'd transmitted live images," he admits. "I had two trackers to get ready, one as a backup, and a friend added his payload with a GoPro camera. So it was quite a complex launch and it made an impressive sight as it went up."



Above is the radio transmitter for sending data back, while below it is the GPS receiver



This is the power regulator to reduce the battery voltage to 5V for the Pi

> "Not only are they believed to be the highest photos transmitted live from an amateur device, the ability to thrust a Pi into space is an amazing feat"

In the future, Dave wants to use the forthcoming Pi camera: "That will mean I can record high-definition images and video"

MAKER PROFILE: DAVE AKERMAN

David Akerman is a high-altitude ballooning enthusiast who hit the headlines when he tethered a Raspberry Pi, complete with a webcam to photograph its amazing progress, to a helium balloon.

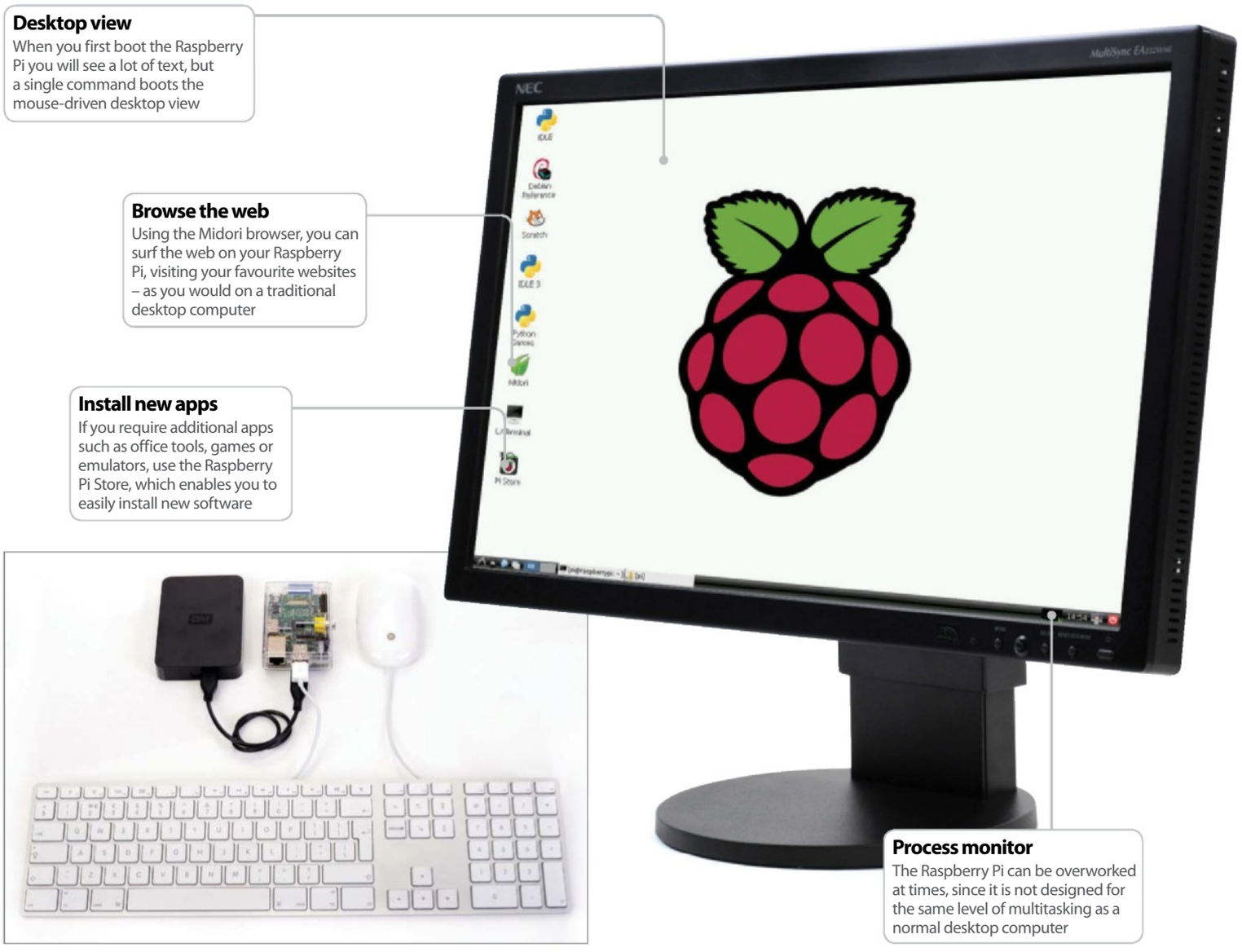
TRY IT YOURSELF

The Kit:

- Raspberry Pi = £25
- NTX2 radio transmitter = £25
- Webcam = £13
- Lassen IQ GPS receiver and antenna = £30
- 1200g Hwoyee balloon = £70
- Gas cylinder (Helium = £80)
- Parachute = £30
- 6x Energizer AA Lithium = £8
- Yupiteru radio scanner = £80
- Radio aerial = £15
- Plus polystyrene foam, duct tape, nylon cord

The Knowhow:

www.daveakerman.com/?p=592
www.ukhas.org.uk



Use your Pi as a desktop PC

You can do so much with the Raspberry Pi – but start with the basics!

Resources

Raspbian Image:
www.raspberrypi.org/downloads

W32 Disk Imager:
sourceforge.net/projects/win32diskimager

Cross-Platform Raspbian installation:
elinux.org/RPi_Easy_SD_Card_Setup

LED status:
www.elinux.org/R-Pi_Troubleshooting

A brand new Raspberry Pi can be used for so many different tasks and is perhaps the most flexible personal computer you will ever own.

But before you start launching it into space, controlling robots and learning how to code, it's worth spending some time with the Pi to find out just how useful it can be with everyday computing tasks such as checking email, word processing and browsing the web.

As well as being the go-to piece of kit for anyone wanting to build a compact media centre or home NAS box, the Raspberry Pi can be used as a desktop computer. You probably won't be able to perform any advanced tasks such as HD video editing or Bitcoin mining, but there remains a wide number of uses for the Raspberry Pi as a desktop computer.

To use the Raspberry Pi in this way you will need to have the usual power supply, SD card and network connection available. You should also have to hand a USB keyboard and mouse – and if you want to connect the Pi to other USB devices, a powered USB hub.

Ideal for compact offices (the Raspberry Pi can be hidden under a desk, in a drawer or even within one of the table legs), this palm-sized digital wonder can be set up to perform all of the typical desktop tasks you've come to expect from the term 'computer' in a matter of minutes.

01 Build your Pi PC

To use your Raspberry Pi as a desktop computer, you should have your basic setup of a power supply, a formatted SD card, HDMI cable and a wired network connection.

On the matter of the SD card, a good tip is to have a dedicated card for each distro or purpose for your Raspberry Pi. This way you can easily change modes by shutting down the Pi and swapping cards.

In addition, you will need a USB keyboard and mouse, and perhaps a powered USB hub for additional storage options. If you're using wireless networking, you'll need to connect your wireless dongle to the computer and either the keyboard or mouse to the USB hub.

With everything ready and connected (except the PSU), just download the Raspbian image and appropriate installer tool.

02 Installing the OS

The Raspberry Pi requires another desktop (or laptop) computer to install the operating system for it onto the formatted SD card. You will of course also need a memory card reader. Which desktop computer platform you choose (Windows, Mac OS X or Linux) will determine which method of installation you use, and how long it takes.

There are several Raspberry Pi-compatible operating systems available, ranging from dedicated media-centre distros to versions of Android, with ARM-specific releases of Debian and Arch Linux in between. For this tutorial we're installing Raspbian, a special version of Debian, but the installation routine is the same for all Raspberry Pi distros.

Windows users will require the Win 32 Disk Imager software to copy Raspbian to the SD card, while Mac OS X and Linux users can install using native tools. Use the resources listed on the previous page to install using the appropriate guide for your platform.

03 Booting your Pi

With the required cables connected, and the Raspbian SD card inserted, switch on your Raspberry Pi. Note the LED grid next to the USB ports – this can tell you if the device is booting correctly. Any errors booting are indicated here and can be referred to online. If installation has completed correctly, you should see the Raspbian configuration screen.

Navigate the configuration screen using the arrow keys, and begin by selecting the 'change_pass' option. This will allow you to change the default Raspbian password to secure your Raspberry Pi. (Note that the default username is 'pi', while the password is 'raspbian')

Meanwhile if you are using a high-capacity SD card, take advantage of this space by expanding the partition using the 'expand_rootfs' option on the configuration screen.

04 Further configuration of Raspbian

The configuration screen provides several options to tailor Raspbian to your purposes. Before proceeding you should check for updates by selecting the 'update' option. This will download the latest version of Raspbian and make the necessary changes.

Toggling 'ssh' in the configuraiton menu allows remote control of your Raspberry Pi via the command line.

You can toggle whether the Pi boots into the configuration screen or the GUI with the 'boot_behaviour' option.

When you're done, use the keyboard to select Finish. This will take you to the terminal. Now enter the command below which will open the X graphical user interface:

```
startx
```

05 Booting into Raspbian

Soon the OS will launch, ready to be controlled by mouse and keyboard. From here you can launch applications, and use the Raspberry Pi like any other desktop computer.

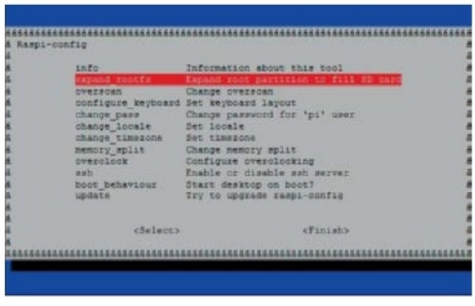
Should you decide that you need to change an aspect of the configuration, open LXTerminal and enter the following command:

```
raspi-config
```

This will launch the configuration screen again, this time in a window and enable you to make any changes that you feel are needed.

Using LXTerminal is essentially the same as booting up the Raspberry Pi and using the command-line full-screen, enabling you to use all of the same text-based commands and instructions to configure Raspbian.

LXTerminal can also be used to install software on your Raspberry Pi.



06 Installing new software

After booting, the first thing you should do is add software via the Pi Store, which can be installed via this terminal instruction:

```
sudo apt-get update && sudo apt-get install
pistore
```

Open the Pi Store using the new desktop shortcut and use the search tool and category tabs to find apps such as *Minecraft* or a DOS emulator.

The Pi Store is purely for apps dedicated to the Raspberry Pi. For a wider selection you will need to use the apt-get command in LXTerminal.

There are many applications that you can install via this method, available via the Raspbian repository. You will need to know the name of the software – for instance, this instruction will install the Scrot screen-capture tool:

```
sudo apt-get install scrot
```

07 Start using your tiny new desktop computer!

Raspbian is the best Raspberry Pi operating system for performing standard desktop tasks.

Along with the pre-installed programming utility Scratch, the Midori web browser enables you to visit your favourite websites, although note that Gmail can be slow so choose the HTML option at the login screen. Social networking sites like Facebook might be slow too, but you can overcome this with a different browser. Using the apt-get command, you might install the Chromium browser:

```
sudo apt-get install chromium-browser
```

Open Office is also available:

```
sudo apt-get install openoffice.org
```

The AbiWord word-processing app and Gnumeric spreadsheet are also available:

```
sudo apt-get install abiword-common
sudo apt-get install gnumeric
```

08 Safely shut down the Pi

Shutting your Raspberry Pi down safely will prevent a lot of problems in the future. While there is no standard 'shutdown' option from within X, you will be able to switch the computer off using LXTerminal.

One of the most common ways of breaking your Raspberry Pi's OS is to switch off the device at the mains while the Raspbian is still running. Doing this will corrupt the operating system, necessitating reinstallation.

Correctly shut down with:

```
sudo shutdown -h now
```

You can now unplug the computer. If you want to restart your Raspberry Pi, use:

```
sudo shutdown -r now
```

To schedule the shutdown five minutes later, switch 'now' to '+5':

```
sudo shutdown -r +5
```

Build a Raspberry Pi media centre

With an XBMC-based distro and HDMI cable you can watch TV with your Raspberry Pi

Resources

Raspbmc XBMC distro for Raspberry Pi:
http://download.raspbmc.com/downloads/bin/installers/raspbmc-win32.zip

OS X/Linux SD card installation tips:
www.raspbmc.com/wiki/user/os-x-linux-installation/

Raspbmc wiki:
www.raspbmc.com/wiki/

MPEG-2 Licence:
www.raspberrypi.com/mpeg-2-license-key

While flipping through the pages you've probably noticed just how versatile the Pi is. Many of these projects use the device on a limited basis, however. To really test the computer and bring it into the heart of your home, why not set it up as a fully featured XBMC-powered media hub, capable of serving entertainment to your TV and home network?

XBMC is perhaps the most popular open-source media centre solution and Raspberry Pi owners have their own dedicated distro, Raspbmc. Installation is fast, configuration simple and you'll soon find yourself enjoying videos, music and photos stored on your SD card, an external hard disk drive, elsewhere on your home network and from the internet. You might even investigate ways to secrete your Pi to avoid a tangle of wires hanging below your HDTV – these range from attaching it to the back of your TV with an adhesive hook-and-loop fastener to mounting your Pi inside your external HDD case.

A wide selection of plug-ins are available for XBMC, adding YouTube video channels, movie streaming services, some cable TV stations and even sports channels. These can be added to your Raspbmc installation, which essentially means that your low-cost small computer can act as a stand-in for expensive media centres costing hundreds of pounds!

01 Preparation

Building a media centre has never been this easy. All you need to get started is a Raspberry Pi (preferably in some sort of case), a blank formatted SD card, an HDMI cable, your device's power supply, a keyboard and/or mouse, and a network connection (Ethernet – don't use wireless until the device setup is complete).

You should also have an external hard disk drive ready, complete with music, videos and photos to enjoy on your new media centre.

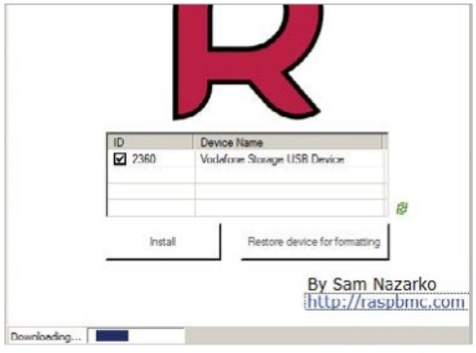
Note that if you have too many USB devices for the two USB ports on the Raspberry Pi, you should employ a powered USB hub. Standard hubs that draw their power from the host device are unsuitable as they will deny much-needed energy to the Raspberry Pi.

02 Using the Raspbmc installer

Different methods are used for installation depending upon your operating system. Windows users should download the Raspbmc installer (see Resource list) and extract the zipped data into a folder. With the SD card inserted into your card reader, browse to setup.exe, right-click and select 'Run as administrator'. Your SD card should be listed in the installer tool, so select this and click Install, waiting for confirmation before removing the card.

Mac OS X and Linux users will need to download Raspbmc via the command line and install using superuser permissions. Full steps and troubleshooting suggestions on completing this can be found in the Resource list.

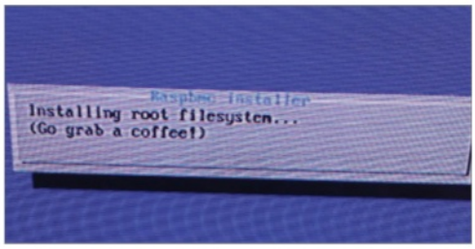
The end result is the same across all platforms, however – an SD card with Raspbmc installed, ready for you to insert into your Raspberry Pi.



03 Raspbmc first boot

With all of your cables connected and the SD card inserted, make sure that the Ethernet cable is connected and your router switched on. Next, plug your Raspberry Pi power supply into the device to power up and begin the boot process.

On the first run, the full installation of the Raspbmc platform is installed – what you installed to the SD card was just the installer framework. The online installation enables you to be sure that the latest version is downloaded to your Raspberry Pi and installed to the SD card. The message 'Installing root filesystem…' will be displayed, along with the suggestion to make some coffee. As long as your internet connection remains stable, installation will eventually complete, so be patient and have that hot drink!



04 Configuring audio

When Raspbmc finally boots, the slick user interface will load up. This can be controlled using a keyboard, mouse or any number of mobile apps on Android, iPhone and Windows Phone.

The first thing you should do is configure Raspbmc to your setup, via the System>Settings menu. For instance, in the Audio submenu you can choose between Analog audio and HDMI. Meanwhile you can use a USB TV decoder to send live TV from your aerial to your Raspbmc device, but make sure you're using the MythTV add-on (see below for more on add-ons). There are many configuration options, so if you're stuck visit the Raspbmc wiki.

Updates for Raspbmc are issued regularly and downloaded automatically. Your device will check for new versions when it starts up.

05 Get familiar with Raspbmc

The user interface for Raspbmc should be extremely easy to use but, as with any new setup, you might find yourself getting a little lost.

You'll mostly use the Pictures, Videos and Music screens. These are essentially libraries for your media, the easy-to-open repositories where you will find your favourite music, photos and movies. Each of these menus has a choice between Files and Add-ons, with the latter opening up to display the list of available media. Raspbmc will display any related meta information too, providing additional details about the media files you're browsing.

Also listed is the Programs menu, listing TV shows recorded using a PVR solution such as MythTV and video files on your hard disk drive that have been indexed as a program rather than a movie.

06 Add network drives

While you might download media to your SD card or connect a hard disk drive or other USB storage device to your Raspberry Pi for viewing in Raspbmc, you can save time by connecting to a network resource that is sharing movies, music and photos.

This device might be a second computer on your network or a NAS drive (it might even be a Raspberry Pi-powered NAS!) and you will need to establish a connection from Raspbmc to your network resource.

To do this, open System>Settings>Network and click Add. You will then need to select the Protocol from the drop-down menu, input or browse for the Server Name, select the name of the shared folder and add security credentials. Seconds later, you'll be streaming media across your network!

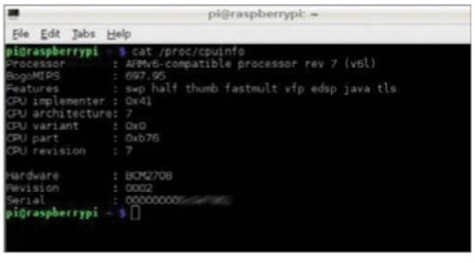
07 Install add-ons and codecs

The basic Raspbmc setup can be enhanced beyond viewing the contents of your network drive to stream media online via a wide selection of add-ons. The majority of these can be installed in the **System>Add-ons** screen using the Get Add-ons option.

You will be unable to view MPEG videos until you unlock the MPEG-2 decoder. To add this, find the motherboard serial number by quitting Raspbmc and entering:

```
cat /proc/cpuinfo
```

Note the number, then visit the licence key site (listed in Resources on the previous page) to purchase. Once received by email, the licence must be entered manually as a new line in /boot/config.txt, found by opening the SD card on your PC. Save and close when done, then reboot Raspbmc to enjoy MPEG-2 videos.



08 Sit back and relax!

So you've got your hard disk drive and network storage to call on for locally streamed movies and music and you've found plug-ins for YouTube, Vimeo and some of your favourite websites. The MPEG-2 codec licence has been bought (just £2.40) and installed and you've configured the system to play video and music with the right audio settings for your sound system setup.

All that is left is to conceal the device and the cables, perhaps swap the Ethernet connection with a wireless network dongle and find a decent remote control app.

Let's just assess things: you've built a media centre from a Raspberry Pi, streaming media from the web and now you're controlling it remotely with a smartphone app.

Life is pretty good with the Raspberry Pi!

What you'll need for this project

USB hub
You can either just plug a hard drive or USB stick directly into one of the Pi's USB ports, or use a powered USB hub for more options.

Stream video
The Pi makes a great central store for all your videos and can stream to another PC, a Pi running XBMC, or an Xbox or PS3.

Silent runnings
As the Pi has no fans, it is completely silent, meaning it can be left in a bedroom or cupboard without you being worried about it disturbing anyone.

Low power
The Pi uses a lot less power than a normal desktop PC, since it features a similar processor to modern smartphones.

Set up a file server

Learn how to use your Pi as a basic file server on your network or over the internet

Resources

External hard drive:
Any external hard drive or USB stick

Powered USB hub:
If you need extra USB ports

Samba:
www.samba.org

Nowadays many of us have large music, video and photo collections, possibly all stored on a few external hard drives. If you want to dig out some old family films or listen to some music, you have to traipse upstairs, turn on the PC, plug in the hard drive and copy the media to a USB stick. Then go downstairs again and plug the stick into your Xbox, PS3 or hi-fi to access the media. Wouldn't it be much better to just be able to access all your media on your network from one central location? NAS, or 'network attached storage', solutions have been available for a while, but can be quite expensive. It's cheaper and much more fun to build your own, and the Pi is perfect for this!

As the Raspberry Pi only has two USB slots, and four on the Model B+, if you have a large number of external drives and USB sticks, you may well want to purchase a small, powered USB hub to plug all your hard drives into. Since the Raspberry Pi has such low power usage, and is silent, it's perfectly acceptable to leave it running for long periods of time without worrying about racking up a large electricity bill.

This guide will take you through the process of setting up a simple NAS which will act as a server using the popular Samba system. Samba is supported in Windows, Mac OS X and is easily configured in your favourite Linux distro.

01 Setting up

To start with we need to get everything set up on our Pi. You'll need to make sure that you have a suitable operating system running – Raspbian is perfect, but you can use others too. You'll also need a keyboard and mouse plugged into the Pi, and internet access.

02 Starting a terminal

Now open up a new terminal window. This is done by clicking on the icon in the bottom left, then choosing Accessories, then LXTerminal. If you are using a different operating system to Raspbian, the location and program name may be slightly different.

03 Updating Apt

We need to make sure our package manager is up to date, so to do this we have to use the 'update' command. It's a good idea to do this before you attempt to install any new software onto the Pi from the terminal.

```
sudo apt-get update
```

04 Adding NTFS support

If you primarily use Windows, it's likely that most of your drives are set to use the NTFS file system. As Linux doesn't support this by default, we need to add some drivers to our Pi. The NTFS-3G package has been around for a long time and is also a popular way to get NTFS support on OS X.

```
sudo apt-get install ntfs-3g
```

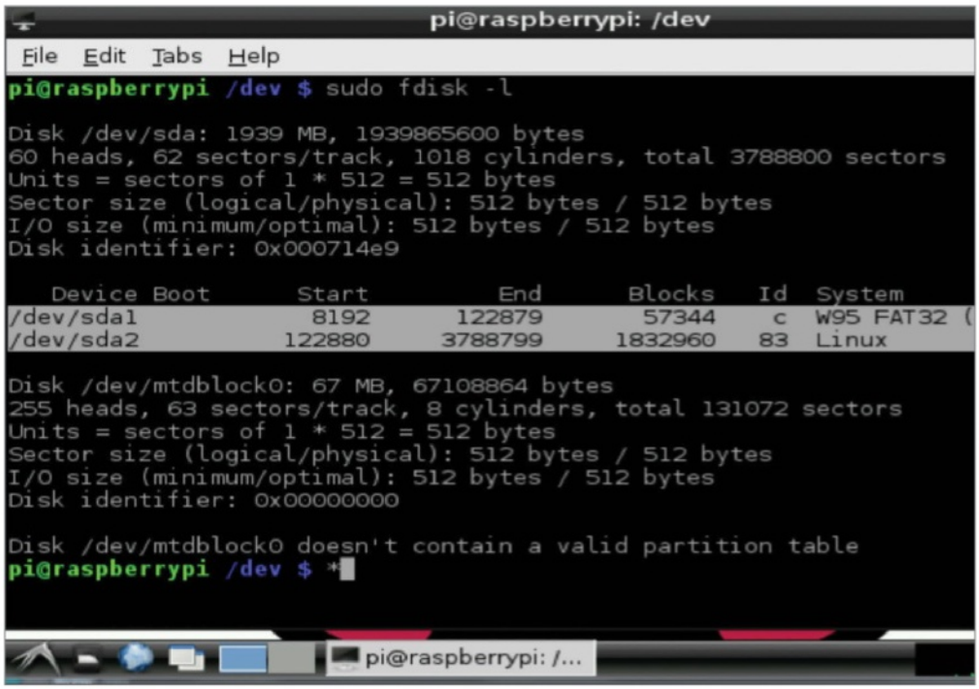
05 Detecting mounted devices

It's necessary to set up all the mount points for each of the hard drives we have attached to the Pi, and that requires a little more work than in Windows or OS X. Firstly we have to list all the attached devices, using fdisk.

```
sudo fdisk -l
```

06 Understanding the list

The list on your screen may be a little confusing if you're new to Linux. The first disk, in this case **/dev/sda**, is the SD card that we have booted our Pi from. We don't need to worry about this one, as we won't be adding that to our NAS.



07 Listed external drive

The next devices listed are the ones we are interested in. Depending on the number of drives you have attached and how they are partitioned, you may see slightly different results. Here – see picture above – we have one drive (sda) with two partitions (sda1 and sda2).

08 Set mount points

In Linux we have to define mount points a little differently to Windows. We need to make a couple of directories where we will access the stored files from. The standard place to put these is in the **/media** directory, although theoretically they can go anywhere.

```
sudo mkdir /media/Music
sudo mkdir /media/Videos
```

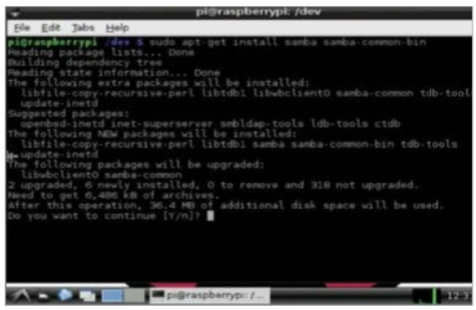
09 Mounting drives

To actually mount the hard drives, we have to use the 'mount' command. Then you need to enter the actual device name and partition, and then the location of the mount point that we just set up. The code below gives you an example, but you'll need to adjust the relevant parts accordingly for your setup.

```
sudo mount -t auto /dev/sda1 /media/Music
sudo mount -t auto /dev/sda2 /media/Videos
```

10 Adding shared directory (optional)

It's possible that you don't want to share everything that's contained within your external drive – maybe it has a pile of old work documents which you don't want cluttering up the network.



Simply add a separate **/shared** directory to each mount point.

```
sudo mkdir /media/Videos/shared
sudo mkdir /media/Music/shared
```

11 Installing Samba

The next step is to install our sharing protocol, which is Samba. This allows for easy setup on Windows machines, as well as XBMC, OS X or Linux. Again, this is done from the command line using the Apt tool. Simply choose Y when asked at the prompt.

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

12 Backup config

We need to make some changes to the main Samba config file, and it's always a good idea to make a backup before doing anything which may break your setup like this. We just need to copy it using the 'cp' command.

```
sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.backup
```

Fig 1

Fig 2

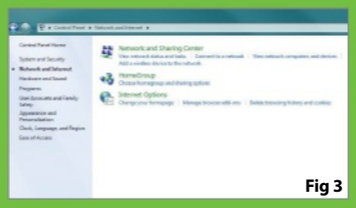


Fig 3

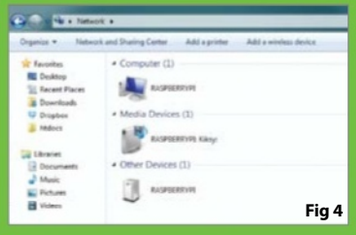


Fig 4


Fig 5

■ Fig 1: Samba is a great way to share files between Linux and Windows systems

■ Fig 2: To adjust the settings from the Windows machine, go to **Start>Control Panel**

■ Fig 3: Click 'Network and Internet'. Ensure that the Pi and PC are on the same workgroup

■ Fig 4: Click 'View network computers and devices'. You should see the Pi listed

■ Fig 5: To change workgroup, open System settings in Control Panel, then 'Change settings'. Click 'Computer Name/Domain Changes' and edit the 'Member of' setting

13 Edit config

To edit the config file we use a simple text editing tool called 'nano'. Nano comes pre-installed with Raspbian, but you could also use a graphical editor if you wanted, or install an alternative such as 'vi'. If you really don't like using text-based editors, then you could use AbiWord, which comes with Raspbian.

```
sudo nano /etc/samba/smb.conf
```

14 Setting workgroup

If you have a Windows network, the default workgroup is called 'WORKGROUP'. Many people change this to something else, however – if this is the case, scroll down using the cursor keys and enter your actual workgroup network name. Note that you can't use the mouse to navigate from within nano.

```
workgroup = WORKGROUP
```

15 Setting security options

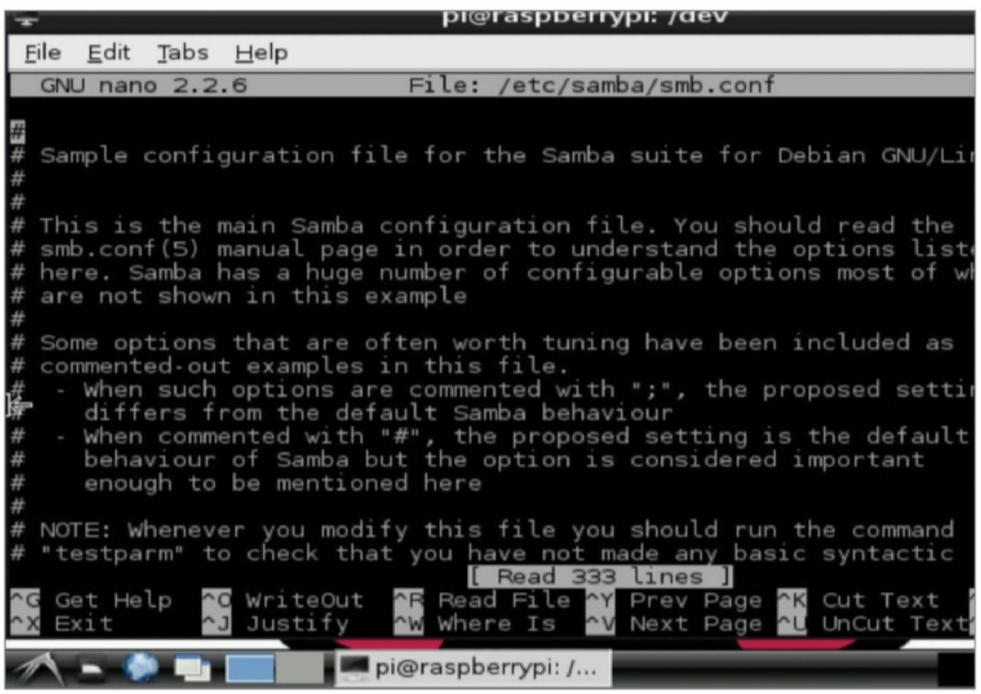
By default there is no security on your NAS shares. If you're okay with this, then you can skip this step. If not, then you need to scroll down and uncomment the 'security = user' line by removing the # symbol. This is found in the #Authentication# section.

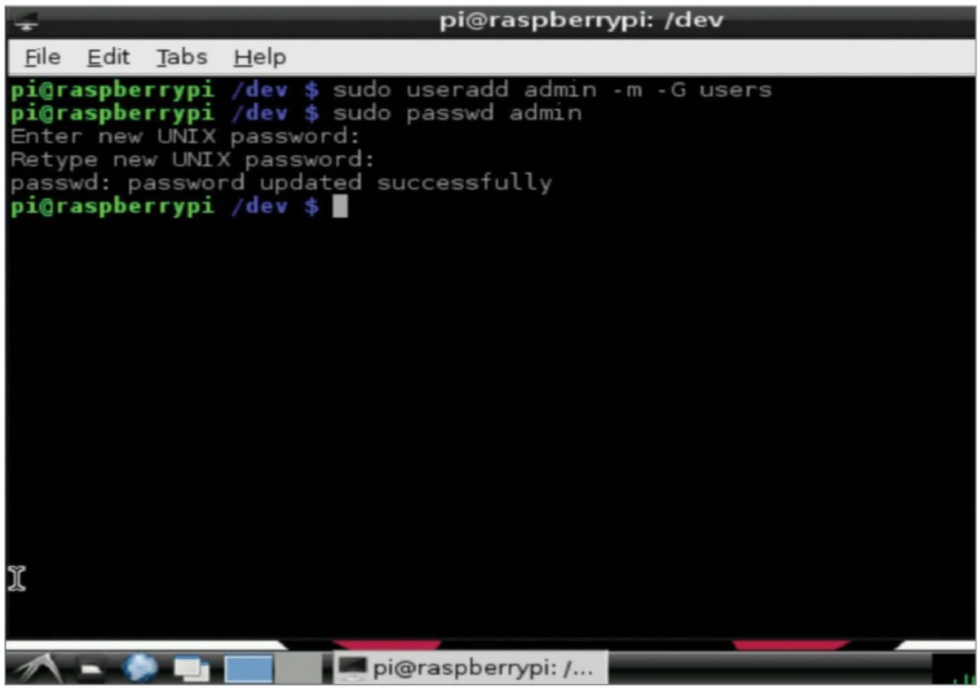
```
security = user
```

16 Adding a share

To add a share, we need to add a new block at the end of the config file. This defines our new share location and who can access it. Edit the path variable to be the one you set up earlier in the tutorial.

```
[Shares]
comment = Shares Folder
path = /media/Music/shared
valid users = @users
force group = users
create mask = 0660
directory mask = 0771
read only = no
```





"We can go to another machine on our network and make sure we can access the shares on our Pi"

17 Adding a user

Back in the terminal, we'll add a user with access to the shares – obviously, change the password from 'admin' to something much more secure.

```
sudo useradd admin -m -G users
sudo passwd admin
```

18 Add user to Samba

Now we need to add that user to the Samba users. Enter the command and you'll be asked to enter a password, then repeat it to make sure its correct. Make sure to change the 'admin' part to whichever user you wish to add.

```
sudo smbpasswd -a admin
```

19 Restart Samba

Next, we restart our Samba service to activate all the changes. This should only take a few seconds, once the 'starting Samba daeomons: nmbd smbd' message has appeared.

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

20 Check the shares

Once the server is running, we can go to another machine on our network and make sure we can access the shares on our Pi. On Windows, ensure you're connected to the correct workgroup, go to **Start>Computer** and click the network icon.

21 Make mounts permanent

We want our drives to automatically mount as soon as we turn the Pi on. To do this we have to edit the file system table from within nano.

```
sudo nano /etc/fstab
```

22 Edit fstab

We just need to add a couple of lines to the bottom of the file. As you can see already, the root drive and partition are already set to mount automatically, so we do the same, just using the 'auto' option to choose the file system automatically for us.

```
/dev/sda1 /media/Music auto noatime 0 0
/dev/sda2 /media/Videos auto noatime 0 0
```

Setting up automatic backups

Set up your Raspberry Pi to run automatic backups of your media

If you have lots and lots of media, photos and other files, you might want to make sure that you don't lose any by running regular backups. Doing this manually can be a real pain, so you might be interested in knowing that the Pi can act as a backup device as well as a NAS. To perform the backups we need two different things. Firstly we need a little Linux program called rsync. To install, simply run:

```
sudo apt-get install rsync
```

The second thing we need to do is edit our cron table. The cron table is a small file that is used to set up programs or scripts to run at predefined intervals automatically. Crontab comes ready installed with Raspbian, so to edit we just need to enter:

```
crontab -e
```

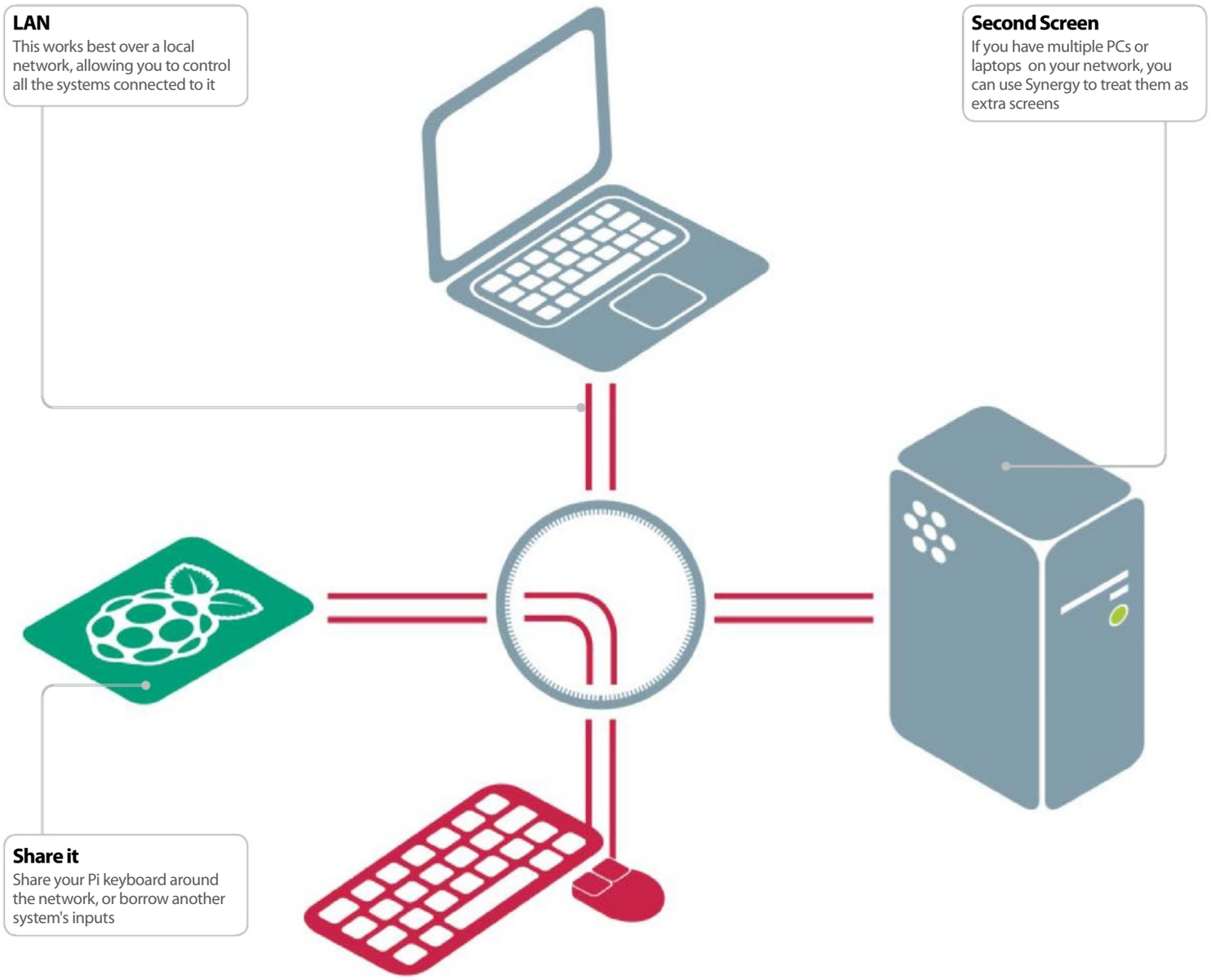
Then we need to add this line into the table. Obviously you'll need to edit the actual location of the directory or device you wish to back up, as well as the target drive.

```
0 5 * * * rsync -av --delete /media/
Videos/shared /media/Video/shared/backups
```

Now to run the backup to test it out, all you need to do is enter:

```
rsync -av --delete /media/Videos/
shared /media/Videos/shared/
```

The first time it runs, it may take a while. However, the next time it will only recopy new files, or ones which have been modified or deleted.



Network and share your keyboard and mouse using Synergy

Borrow a mouse and keyboard from another PC, using only your Raspberry Pi and Synergy

Resources

Latest Raspbian Image
raspberrypi.org/downloads

Synergy

Host computer

One issue we sometimes find with the Raspberry Pi is the lack of USB ports if you are using Model A or Model B. We don't always have the luxury of using a powered USB hub, and it can become a bit of a hassle to juggle a mouse and keyboard with other devices. The Model B+ has four USB ports which does help, it can sometimes still not be enough. Instead of using a hardware solution for this, you can always try a software solution – one that opens up the uses of the Raspberry Pi as well.

The Synergy program lets you share the mouse and keyboard of one system with other systems on the same network, acting as a virtual KVM.

In this tutorial we'll learn how to use your main computer's input devices on your Raspberry Pi, as well as how you can look into making it a server.

01 Install Synergy

Synergy is available from the Raspbian repositories. We can install it by using the following code:

```
001  $ sudo apt-get update
002  $ sudo apt-get install synergy
```

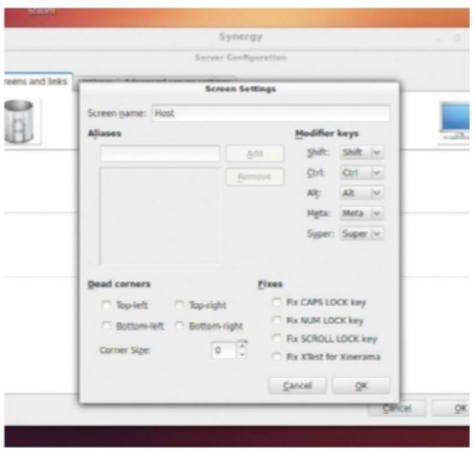
This will go through the basic installation process as normal and Synergy will be put in the Accessories folder.

02 Start up Synergy

Start up Synergy on the host computer and choose the 'Server' option for the moment. We'll cover how to use it as a client later, and how to use the Raspberry Pi as a server for the mouse and keyboard.

03 Encryption and passwords

Here, you can set up whether or not the connection is encrypted. This is useful for stopping key loggers from being able to snoop on your information, or random clients from connecting and hijacking your mouse. Provide a password and then click Finish.



04 Server naming

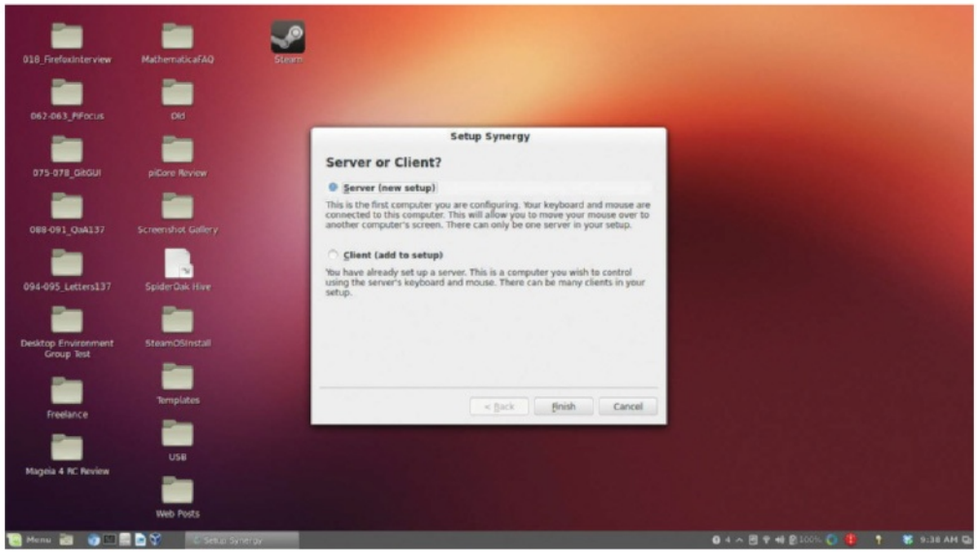
Once the process has finished, go to Configure Server. Your host computer will be put virtually in the centre of your array of displays and you can drag and drop it around, along with any connected screens. Double-click on the server to change its name, making it easier to remember and find. Add a new screen and call it pi.

05 Starting and connecting

Once you're happy with the setup, click Start to be able to accept client connections. To connect from a Raspberry Pi, enter the following:

```
001  $ synergyc --name pi [IP Address of host]
```

It will be recognised as 'pi' on the host system.



■ Above – Select the Server option to use this machine's mouse and keyboard for the Pi

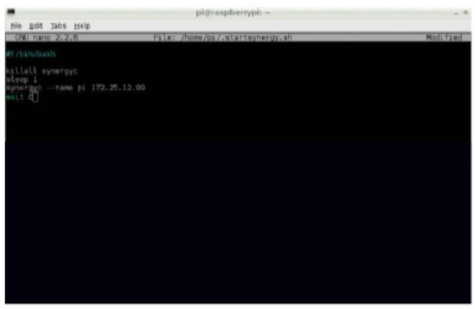
06 Autostart Synergy

To make sure it starts every time you turn on the Pi, we need to create an LXDE autostart file by using the following:

```
001  $ sudo mkdir -p ~/.config/lxsession/LXDE
002  $ sudo touch ~/.config/lxsession/LXDE/
       autostart
003  $ sudo nano ~/.config/lxsession/LXDE/
       autostart
```

And then add the following to autostart:

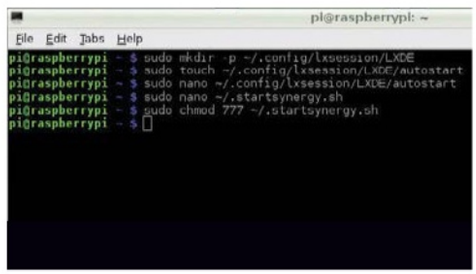
```
001  ~/.startsynergy.sh
```



07 Start file

Open and populate the startsynergy file with:

```
001  $ sudo nano ~/.startsynergy.sh

002  #!/bin/bash

003  killall synergyc
004  sleep 1
005  synergyc --name pi [IP address of host]
006  exit 0
```



08 Permissions

Finally, to finish it off you'll need to run:

```
001  sudo chmod 777 ~/.startsynergy.sh
```

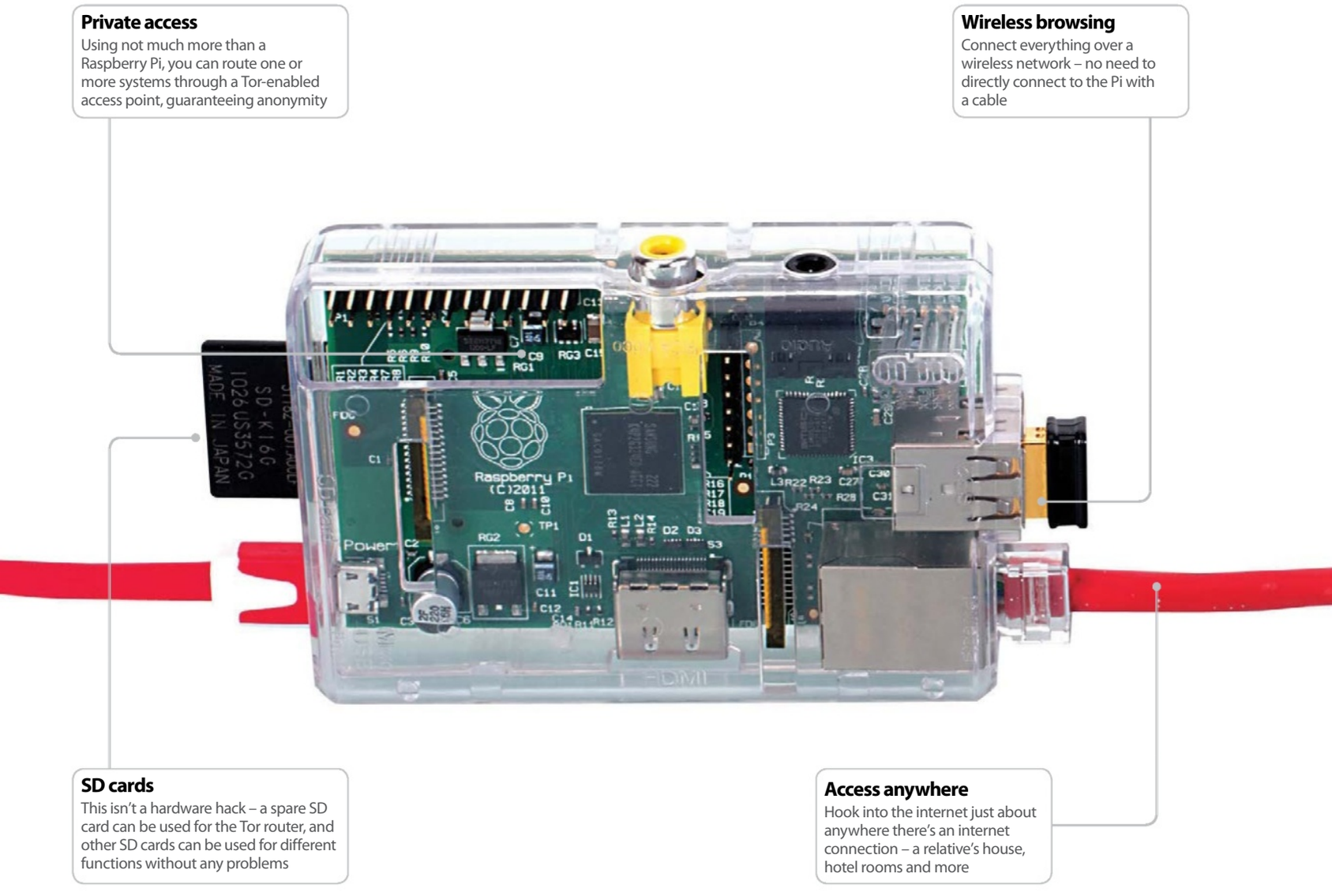
This will autostart, and hopefully autoconnect, Synergy whenever you turn it on.

The Raspbian client is a little old, so if you get a problem you may need to compile the latest version from source.

09 Pi server

Setting up the Raspberry Pi as a server is a little more involved and uses the synergys command. It allows you to listen for clients on specific addresses. You then need to create a separate configuration file to arrange the displays – however, you can load one from a different computer and edit it.

"Synergy lets you share a mouse and keyboard… acting as a virtual KVM"



Browse privately with Onion Pi

Turn your Raspberry Pi into a highly secure and very portable router to keep your system safe and your browsing anonymous, wherever you are

Resources

A Raspberry Pi

Raspbian
www.raspberrypi.org/downloads

Compatible Wi-Fi adaptor
www.adafruit.com/products/814

You can easily turn your Raspberry Pi into the ultimate portable wireless router, requiring very little power and giving you a wireless network wherever there's the most basic of internet connections. What if it's not enough to know you can search the web, though? What if you want to be wholly secure as you do it? Then it's time to upgrade the router with Tor to protect your privacy on the internet.

This 'Onion Pi', as dubbed by Adafruit, combines Raspbian and Tor to create and secure a wireless access point using just a Raspberry Pi. This project is fairly straightforward: after setting up the wireless access point, we install Tor and do some basic setup tasks so that it routes traffic properly, and securely. This will keep you anonymous online – a handy feature in a time of privacy concerns all around the web.

When the Pi is not connected to the internet, it should still function as a wireless router, allowing at the very least a wireless LAN in your location.

So, here's how to set up online anonymity.

01 Install Raspbian

Raspbian is the Raspberry Pi distro we'll be using for the Onion Pi. Download the zip file, extract the image and then apply it to an SD card using:

```
001 $ dd bs=4M if=[version number]-wheezy-
      raspbian
002 img of=/dev/[SD card location]
```

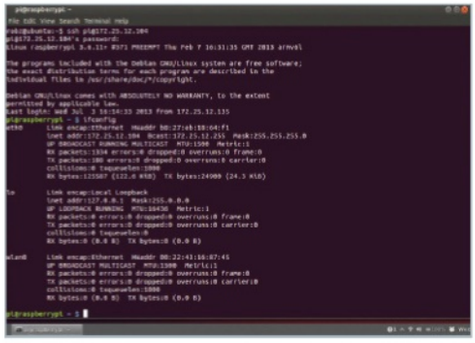
You can also use NOOBS to install Raspbian if you wish.

02 Set up Raspbian

Go through the initial Raspbian setup and make sure to turn on the SSH server, and to disable autoboot to desktop – this is unnecessary and will only use extra power. You can also tell it to fill up the rest of the card if there's room for it.

03 Pi IP

We'll be accessing your Raspberry Pi via SSH to set it up. To do this we need to know its IP address – you can find it by typing `ifconfig` into the command line. Make a note of it and turn off your Pi.



04 SSH connection

Plug your USB wireless adapter into the Pi and turn it back on. On another computer connected to the same network, open a terminal or type into the command line:

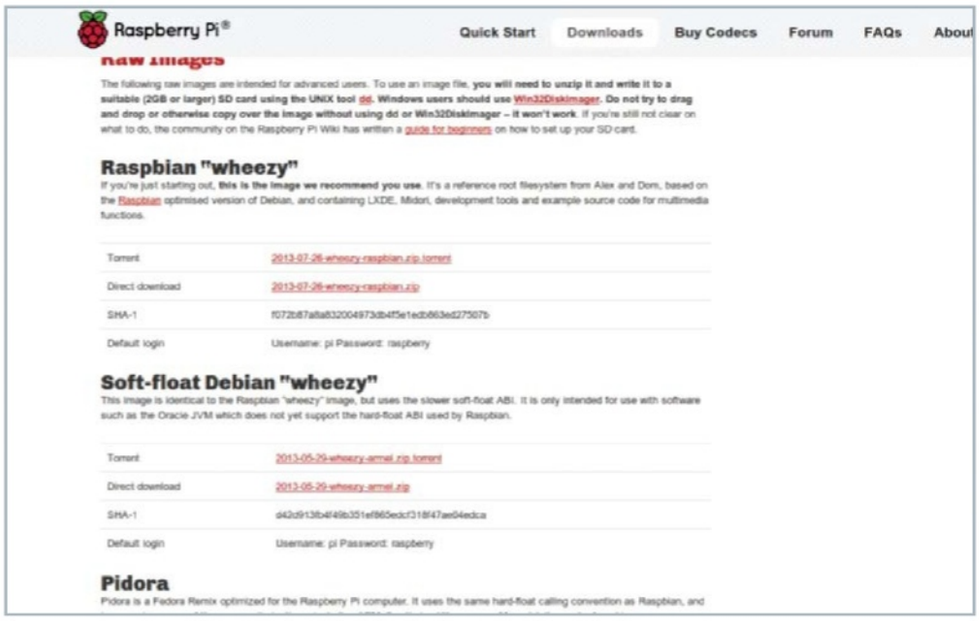
```
001 $ ssh [user]@[IP address]
```

Then enter the password for your Raspbian if it asks for it.

05 Install DHCP

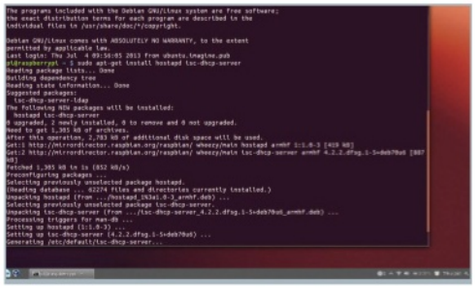
To make life easier for any system connecting to the Pi access point, we need to install a DHCP server to it. We do this with:

```
001 $ sudo apt-get install hostapd isc-dhcp-
      server
```



■ Above – Download and install the Raspbian "wheezy" zip file

DHCP will automatically assign IP addresses to network-attached devices, meaning you won't need static IPs.



06 Set up DHCP

Now we need to configure the DHCP server. Edit the configuration file with:

```
001 $ sudo nano /etc/dhcp/dhcpd.conf
```

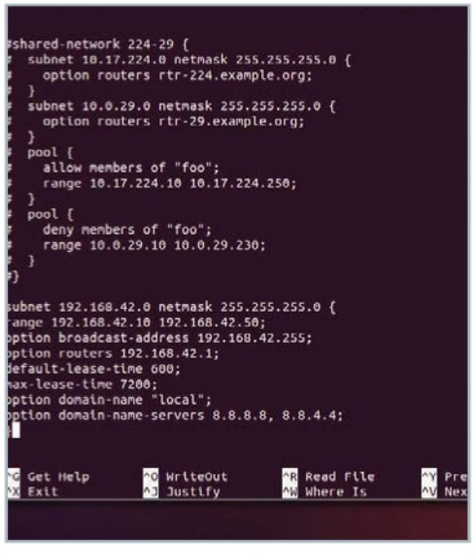
And start by putting a # in front of the two option domain-name entries, then remove the # in front of authoritative, seven lines down.

07 Server address

At the end of the configuration file, add the following:

```
001 subnet 192.168.42.0 netmask 255.255.255.0 {
002 range 192.168.42.10 192.168.42.50;
003 option broadcast-address 192.168.42.255;
004 option routers 192.168.42.1;
005 default-lease-time 600;
006 max-lease-time 7200;
007 option domain-name "local";
008 option domain-name-servers 8.8.8.8,
    8.8.4.4;
009 }
```

Save and exit.



08 DHCP server

Edit the server configuration files so that it's set to work in conjunction with the wireless adaptor:

```
001 $ sudo nano /etc/default/isc-dhcp-server
```

Scroll to INTERFACES and change it to:

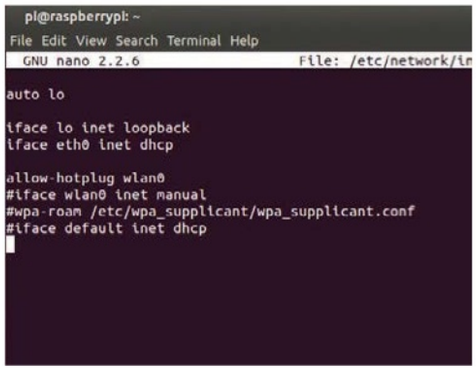
```
001 INTERFACES="wlan0"
```

09 Incoming Wi-Fi

We need to set up the Wi-Fi adaptor to be both static and accept incoming signals. First:

```
001 $ sudo nano /etc/network/interfaces
```

Put a # in front of iface wlan0 and following lines with wpa roam, iface default and any other affecting wlan0.



10 Static IP

Now give the wireless interface a static IP – after the line allow-hotplug wlan0, enter the following:

```
001  iface wlan0 inet static
```

```
001  address 192.168.42.1
002 netmask 255.255.255.0
```

Save and exit, and then set wlan0's address with:

```
001  $ sudo ifconfig wlan0 192.168.42.1
```

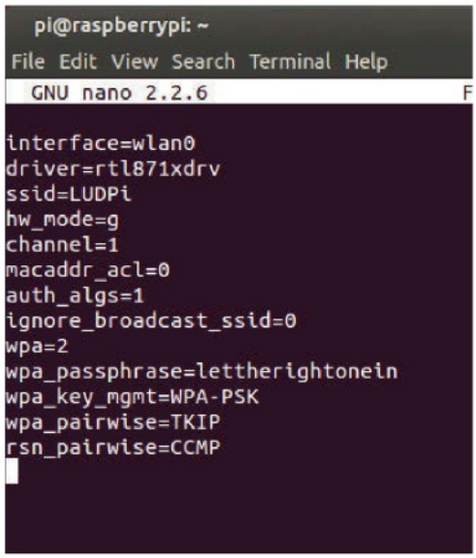
```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0

iface wlan0 inet static
  address 192.168.42.1
  netmask 255.255.255.0

#iface wlan0 inet manual
#wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
#iface default inet dhcp

up iptables-restore < /etc/iptables.ipv4.nat
```

11 WLAN creation

We need to create a new file that holds all the information for our wireless network. We are going to make it password protected so that only the people we want to can access it. To create the file, start with:

```
001 $ sudo nano /etc/hostapd/hostapd.conf
```

And then enter the text from the next step.



12 WLAN configuration

```
001 interface=wlan0
002 driver=rtl871xdrv
003 ssid=[access point name]
004 hw_mode=g
005 channel=1
006 macaddr_acl=0
007 auth_algs=1
008 ignore_broadcast_ssid=0
009 wpa=2
010 wpa_passphrase=[password]
011 wpa_key_mgmt=WPA-PSK
012 wpa_pairwise=TKIP
013 rsn_pairwise=CCMP
```

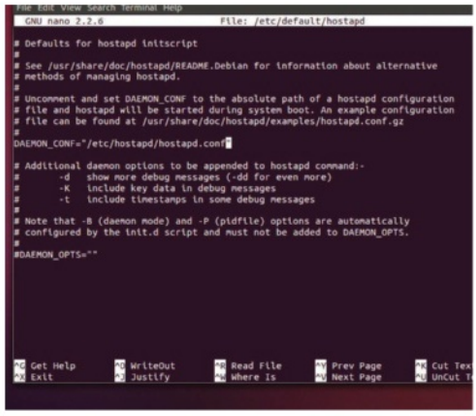
13 Hostapd

After saving and exiting, we need to edit hostapd to point it to this new file. Open it with:

```
001 $ sudo nano /etc/default/hostapd
```

And then find the line #DAEMON_CONF="". Remove the #, and change it to:

```
001 DAEMON_CONF="/etc/hostapd/hostapd.conf"
```



14 Network addressing

Setting up a NAT will allow multiple clients to connect. To do this, run:

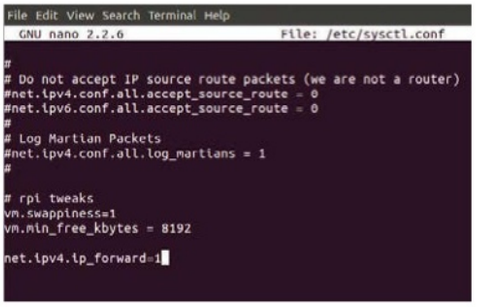
```
001 $ sudo nano /etc/sysctl.conf
```

And add to the bottom of the file:

```
001 net.ipv4.ip_forward=1
```

Save this, and then finish by running:

```
001 $ sudo sh -c "echo 1 > /proc/sys/net/ipv4/
    ip_forward"
```



15 IP tables

Run the following three commands to make sure the internet connection is forwarded correctly:

```
001 sudo iptables -t nat -A POSTROUTING -o eth0
    -j MASQUERADE
002 sudo iptables -A FORWARD -i eth0 -o wlan0
-m state --state RELATED,ESTABLISHED -j ACCEPT
003 sudo iptables -A FORWARD -i wlan0 -o eth0
-j ACCEPT
```

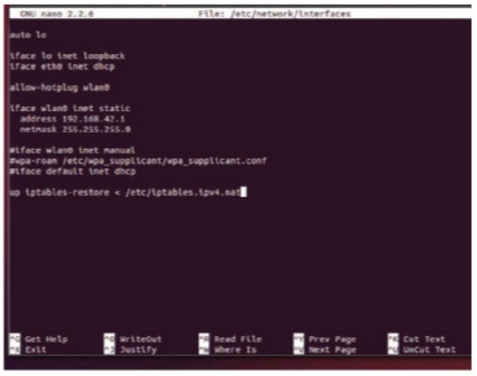
16 Apply configuration

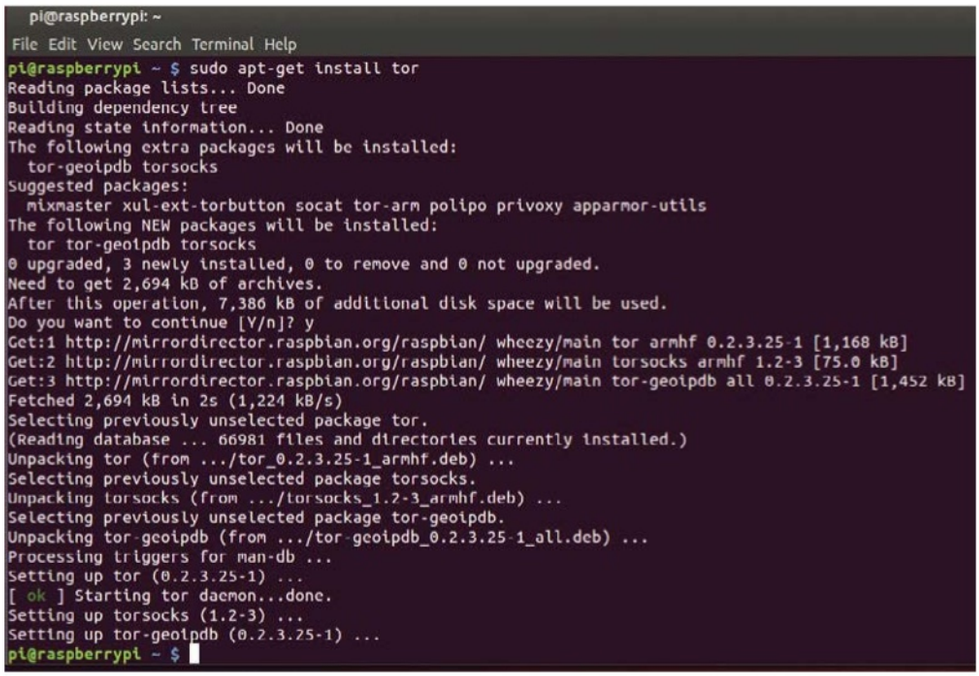
So that this still works after a reboot, type:

```
001  $ sudo sh -c "iptables-save > /etc/
    iptables.ipv4.nat"
```

Then add to the end of /etc/network/interfaces:

```
001 up iptables-restore < /etc/iptables.ipv4.
    nat
```





17 Wi-Fi final

Finally, set it up as a daemon so it runs at boot with the following commands:

```
001 sudo service hostapd start
002 sudo service isc-dhcp-server start
003 sudo update-rc.d hostapd enable
004 sudo update-rc.d isc-dhcp-server enable
```

And the wireless access point part will be finished.

18 Install Tor

After a reboot, we now need to install Tor. Do this simply with:

```
001 $ sudo apt-get install tor
```

Once it's installed, you'll need to edit the Tor config file with:

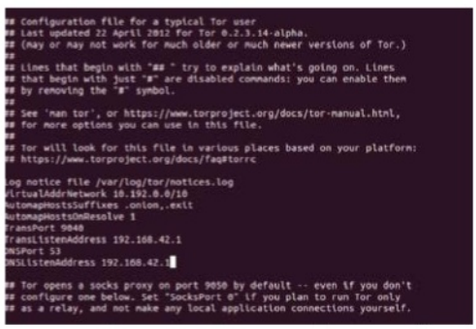
```
001 $ sudo nano /etc/tor/torrc
```

Follow the next step to add all the necessary information to it.

19 Tor configure

Put this below the FAQ comment:

```
001 Log notice file /var/log/tor/notices.log
002 VirtualAddrNetwork 10.192.0.0/10
003 AutomapHostsSuffixes .onion,.exit
004 AutomapHostsOnResolve 1
005 TransPort 9040
006 TransListenAddress 192.168.42.1
007 DNSPort 53
008 DNSListenAddress 192.168.42.1
```



20 Table flush

We now need to flush the current IP tables so that we can get the routing to go through Tor. First of all, do:

```
001 $ sudo iptables -F
002 $ sudo iptables -t nat -F
```

If you want to keep SSH open to connect remotely, you'll need to make an exception for that with:

```
001 $ sudo iptables -t nat -A PREROUTING -i
    wlan0
 -p tcp --dport 22 -j REDIRECT --to-ports 22
```

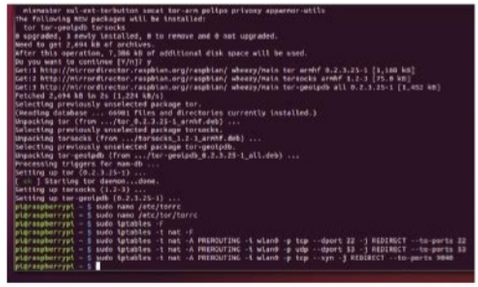
21 Reroute

Route all DNS traffic first, using:

```
001 $ sudo iptables -t nat -A PREROUTING -i
   wlan0-p udp --dport 53 -j REDIRECT --to-ports
    53
```

And then route any TCP traffic with:

```
001 $ sudo iptables -t nat -A PREROUTING -i
  wlan0 -p tcp --syn -j REDIRECT --to-ports 9040
```



22 Check and save

You can check the table setup with:

```
001 $ sudo iptables -t nat -L
```

If you're happy with it, save it to the NAT file like before with:

```
001 $ sudo sh -c "iptables-save > /etc/
                              iptables.ipv4.nat"
```

23 Logging

We should create a log file in case you need to debug later. To do this, use these three commands:

```
001 $ sudo touch /var/log/tor/notices.log
002 $ sudo chown debian-tor /var/log/tor/
    notices.log
003 $ sudo chmod 644 /var/log/tor/notices.log
```

You can also check it with:

```
001 $ ls -l /var/log/tor
```

24 Secure the router

Finally, we can activate the Tor service so that we can start using the access point securely with:

```
001 $ sudo service tor start
```

You can check this if you wish with:

```
001 $ sudo service tor status
```

To make it turn on at boot, you simple add it to rc.d with:

```
001 $ sudo update-rc.d tor enable
```

"Create a log file in case you need to debug later"

Automate tasks
Create automated video- and photo-related tasks easily

USB free
The Pi Camera doesn't take up any USB slots, instead plugging directly into the Raspberry Pi, which also powers it

Multifunctional
Use the camera for time-lapse photography, a robot's eyes, or a webcam in any number of suitable projects

Take pictures and record videos

Follow our tutorial on how to get your new Pi Camera set up on your Raspberry Pi, and how to use it

Resources

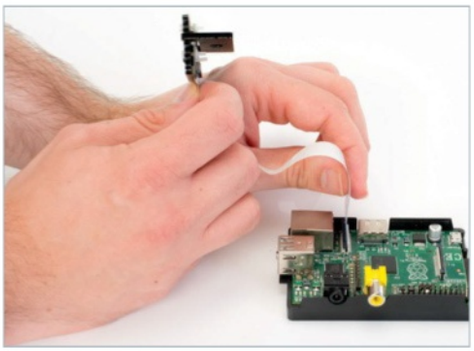
Raspberry Pi Raspbian:
www.raspberrypi.org/downloads

Pi Camera Ashton's picam module:
https://github.com/ashtons/picam

One of the most recent Raspberry Pi accessories is the tiny Pi Camera board – a small PCB with a camera sensor mounted to it that connects via a ribbon to the Raspberry Pi. Because of this, it is not exactly plug-and-play, so you will need to do some extra setup on your Raspberry Pi in order to get it to work properly.

The Pi Camera has multiple functions, such as for time-lapse photography, using as a webcam, or even using as an optical sensor for a Pi-powered robot. Because it does not take up any USB slots, and draws very low power, it is possible for it to be a lot more versatile than a standard webcam is.

The Pi Camera itself is not a low-quality piece of kit either – with a 5MP sensor, it is also able to create up to 1080p quality video footage, which is in fact the same as the Raspberry Pi's HDMI output. So, grab your Raspbian SD card and get started by making the absolute most out of your Pi Camera.



01 Attach Camera

To attach the Camera to the Raspberry Pi, locate the slot between the Ethernet and HDMI port and gently lift up the fastener. Insert the ribbon of the Camera board, making sure to align the ribbon's connectors with those on the Pi.

02 Pi preparation

Before we try to enable the Raspberry Pi Camera, we need to make sure our firmware and software are all up to date with a quick software upgrade. In Raspbian, we do this by opening the terminal and using:

```
$ sudo apt-get update
```

…followed by:

```
$ sudo apt-get upgrade
```

03 Pi config

Once that has finished, run in the terminal or command line:

```
$ sudo raspi-config
```

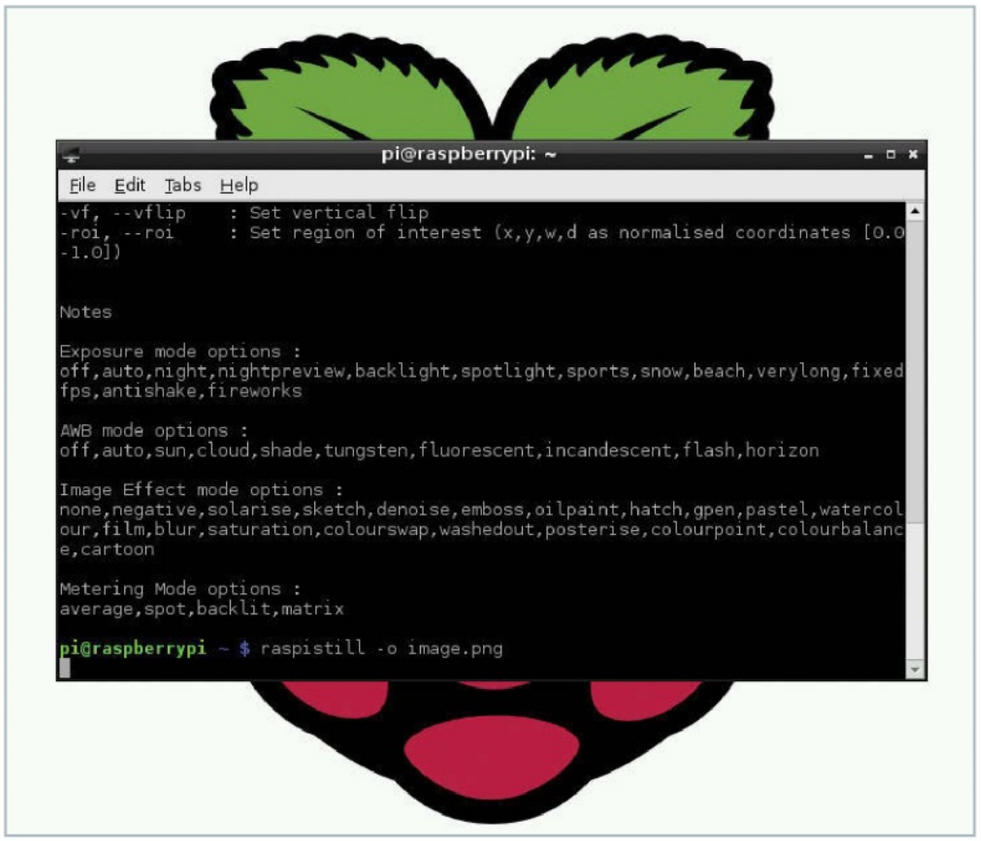
…to start the standard configuration screen. Navigate down to Enable Camera, press Enter and then simply key over to enable and confirm with another press of Enter. Select Finish and then reboot.

04 Take pictures

To take pictures with the Raspberry Pi Camera, you'll simply need to enter:

```
$ raspistill -o image.png
```

This will show a five-second preview of the input of the camera and then capture the last frame of the video.



05 Record video

To record a video, we use a similar command, raspivid, like so:

```
$ raspivid -o video.h264
```

It will also take five seconds of video by default.

06 Picam

If you want to do a little more with the Pi Camera, there's a simple Python wrapper currently available called picam. You'll need to install it first, though, and we'll use pip for that. Install pip with:

```
$ sudo apt-get install python-pip
```

…and then enter:

```
pip install https://github.com/ashtons/picam
/zipball/master#egg=picam
```

07 Picam photos

We can now use Python to construct a script to take photos with the picam module. Very simply, all you need to do is enter:

```
import picam
i = picam.takePhoto()
i.save('/home/pi/test.jpg')
```

And running it will take a photo called test.jpg.

08 Advanced photos

You can have it take photos of specific size and quality with a time-based name by editing the code to look like this:

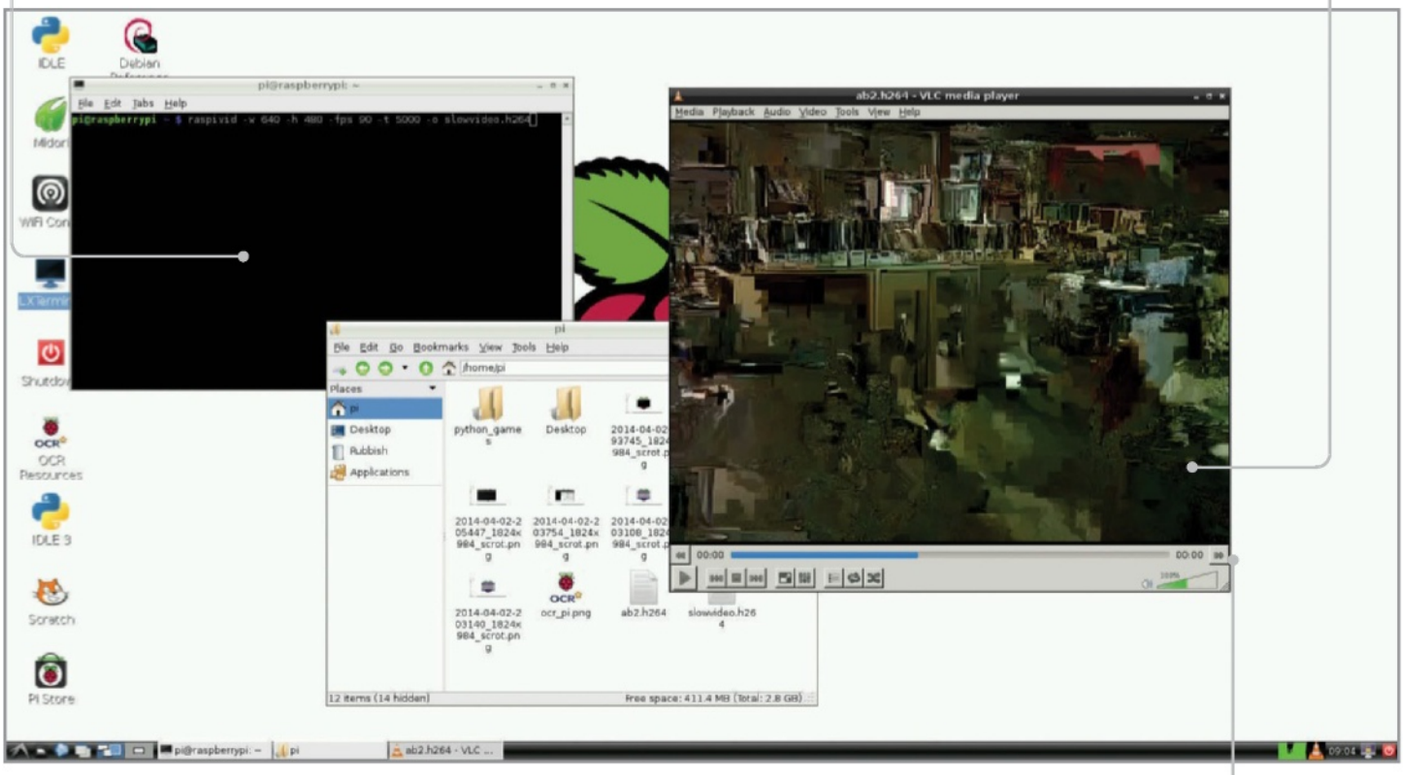
```
import picam
import time
ii = picam.takePhotoWithDetails(640,480, 85)
filename = "/tmp/picam-%s.jpg" % time.
strftime("%Y%m%d-%H%M%S")
ii.save(filename)
```

09 Picam video and more

Picam also allows you to take video in a similar way to the above, with the main difference being that you'll use the recordVideo command. You can use the code to take photos or video at regular intervals for time-lapse, or have it trigger during a specified event.

Slow-mo
Learn how to record multiple types of slow-motion videos for yourself

Video player
Allow your Raspberry Pi to play back the video recording to get the full effect



High frame rate
Play it at normal speed for high frame rate video for specific video projects

Record slow-motion videos with the Raspberry Pi

Use the updated Raspberry Pi firmware to record slow-motion videos with the camera module

Resources

Latest Raspbian Image
raspberrypi.org/downloads

Raspberry Pi camera board

Internet connection

The Raspberry Pi camera has been an interesting little add-on for the Raspberry Pi, allowing people to have slightly more control when it comes to projects that require pictures and video.

The functionality of the camera is always being slightly enhanced and now a new function has made its way to the camera: slow-motion video.

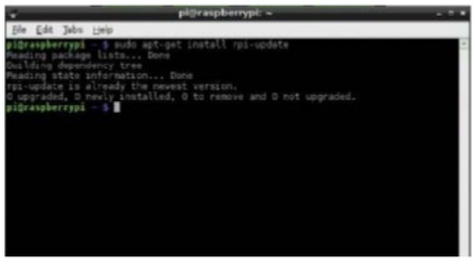
With this you can create short clips as slow as 33 per cent normal speed, with all the same control as you have with the normal camera functions. Unfortunately, at the time of writing, our favourite PiCamera Python module doesn't include the ability to access the slow-motion functions.

"You can also use the slow-motion function to create high frame rate videos"

01 The firmware updater

The first thing you need to do is make sure you update the firmware on your Raspberry Pi. Boot into Raspbian and open the LXTerminal. To install the firmware updater, simply type in the following command. For newer versions of Raspbian, it will likely be already installed.

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

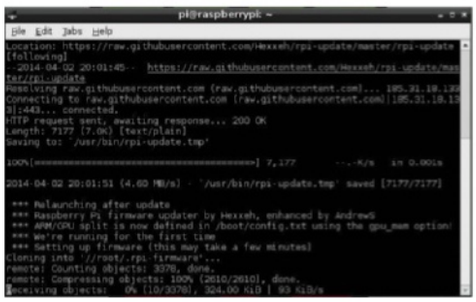


02 Update the firmware

Updating the firmware is very straightforward; back in the terminal just type in the following:

```
001  $ sudo rpi-update
```

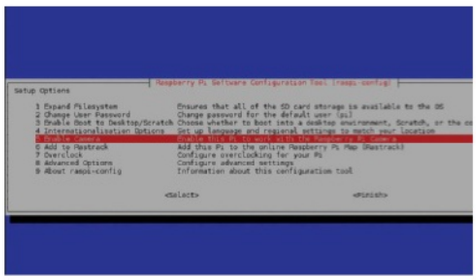
And it should automatically download and install all the necessary files. Once this is complete, reboot your Raspberry Pi to activate the new firmware

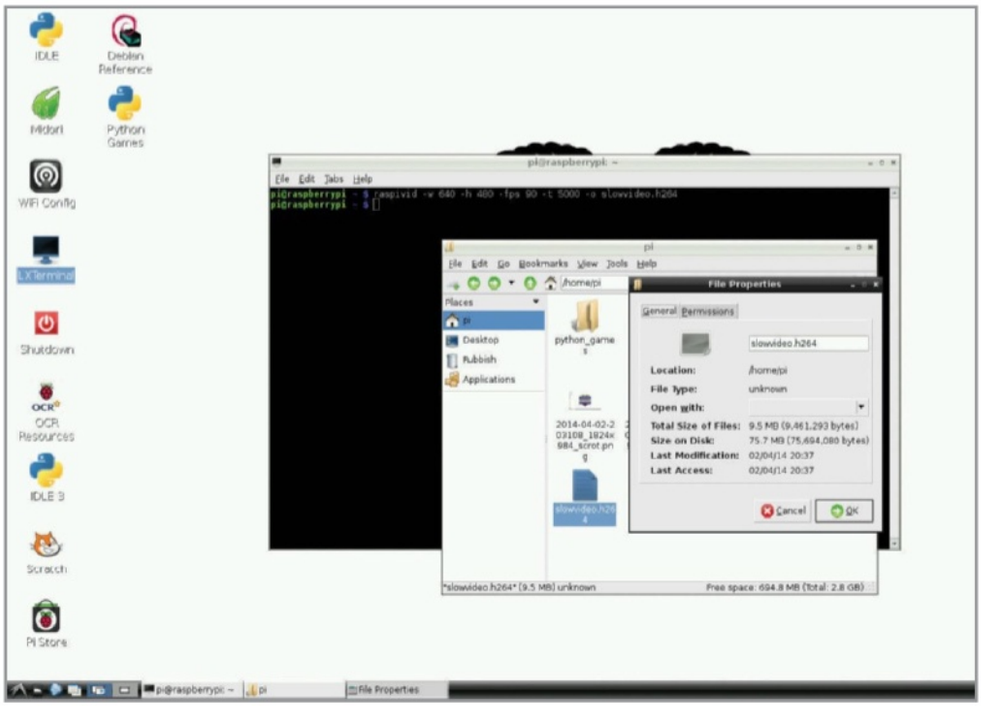


03 Alternate updating

Problems have been known to occur with the firmware update and there are two main ways you can try and fix them. First of all, try updating Raspbian with apt-get update followed by apt-get upgrade. If that method doesn't work, try using the firmware updater like so:

```
001  $ sudo UPDATE_SELF=0 rpi-update
```





■ Above – The maximum resolution is 640 x 480 for this speed of video

04 Camera setup

Just in case you're using a new install of Raspbian, you'll need to make sure that the camera module is enabled.

Make sure that you've plugged in the camera while it's off, and then in the LXTerminal use:

```
001  $ sudo raspi-config
```

Go to the Enable Camera option and enable it. You may need to restart in order for it to actually take effect.

05 First tests

We can now do a quick test of the slow-motion capabilities by entering the following:

```
001  $ raspivid -w 640 -h 480 -fps 90 -t 5000
     -o slowvideo.h264
```

We've told it to make a video at 640 x 480 resolution, to film it at 90 frames per second and to do so for five seconds.

This is the maximum resolution for this speed of video. (See the image above.)

06 Other recording modes

As you may already be aware, as well as being able to record video at 90 frames per second, you can also go down to 60.

You cannot increase the resolution though, so it's stuck to 640 x 480 for both 90 frames per second and 60 frames per second video for the time being.

07 Playing it back

Although your Pi has the means to record the video, it can't play it back. You can either grab the video from the SD card, or install VLC using the following command, which should have no trouble viewing the video file.

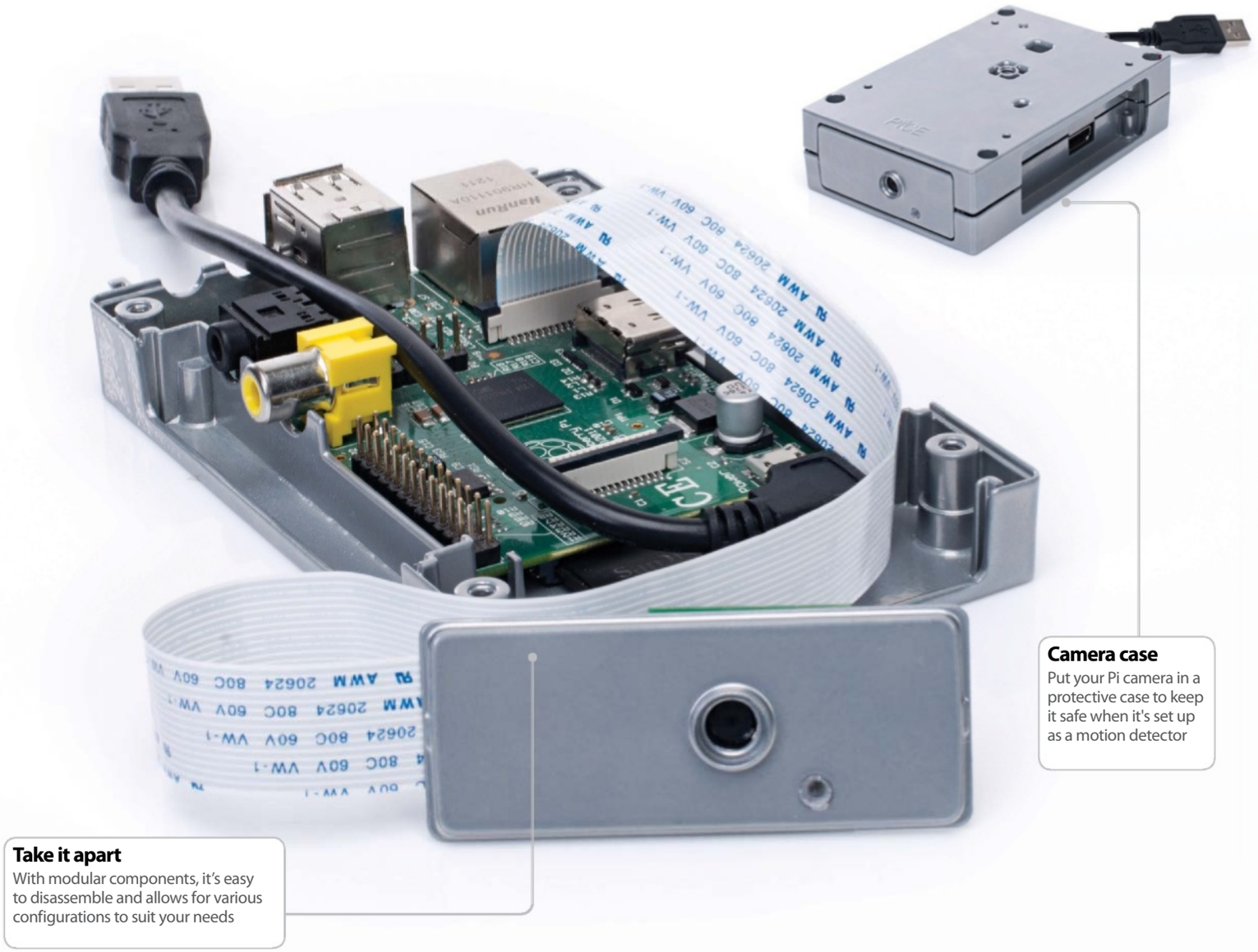
```
001  $ sudo apt-get install vlc
```

08 Other uses

If you do plan to use the slow-motion function, obviously you can use it as intended. However, you can also use it to create high frame rate videos. Video editors such as Kdenlive will let you increase the playback speed, and you can also view them in VLC at normal speed.

09 Final warnings

Doing these slow-motion videos – even at the low resolution we're using – is taxing on the processor. Ideally, try to only capture short clips while using it and definitely not a lot of them in rapid succession.



Camera case
Put your Pi camera in a protective case to keep it safe when it's set up as a motion detector

Take it apart
With modular components, it's easy to disassemble and allows for various configurations to suit your needs

Set up a Pi motion detector

Use the Raspberry Pi camera to detect motion and take a photo of the cause for home security or time-lapse photography

Resources

SD card with up-to-date version of Raspbian

picamera Python module

Raspberry Pi camera module

The Raspberry Pi camera is an excellent piece of kit for many projects. With a decent sensor, video capabilities and great Python support from picamera, there's plenty you can do with it. You can easily do surveillance and time-lapse photography with the camera – but what if you could combine the two? Having the camera take photos when motion occurs reduces the strain on the Pi for surveillance and can create a better time-lapse for more human situations.

The code is fairly simple as well, plus you'll be able to learn some special methods of storing photos.

"Combine time-lapse photography and surveillance"

01 Enable the camera module

Before we start, make sure to update your Raspberry Pi and its firmware with apt-get update and apt-get upgrade followed by rpi-update. Once that's done, enter the raspi-config by typing exactly that in the terminal and choosing 'Enable' in the Enable Camera option. Hit Finish to reboot.

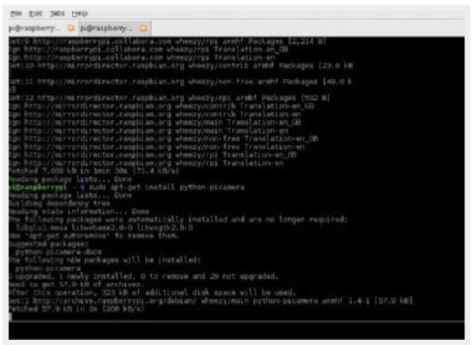
02 Attach the camera

Turn the Raspberry Pi off either during the reboot process or afterwards. Disconnect the power and find the slim port between the ethernet port and the HDMI port. Lift up the fastener and slot in the camera module's ribbon, with the silver side facing towards the HDMI port. (See Fig 1.)

03 Install the Python modules

The code requires a couple of extra Python modules to work. First, the all-important picamera so we can better control the camera, along with the Python Image Library so we can inspect the images. Install them both with:

```
001  $ sudo apt-get install python-picamera
     python-imaging-tk
```



04 Position your camera

Where are you placing your camera? Does the Raspberry Pi have adequate access to power? Are you controlling it with SSH or will it be in range of a display and keyboard/mouse? Set up your Raspberry Pi accordingly and take a sample picture to make sure it's in the right location with:

```
001  $ raspistill -o test.jpg
```

05 Tweak the sensitivity

In our code, difference and pixels are used to determine when a picture should be taken. The difference variable is the amount a pixel needs to change in colour to count as a change, while pixels is the number of these changed pixels that will be used to determine whether or not enough motion has been created to take a picture.

06 Cross the stream

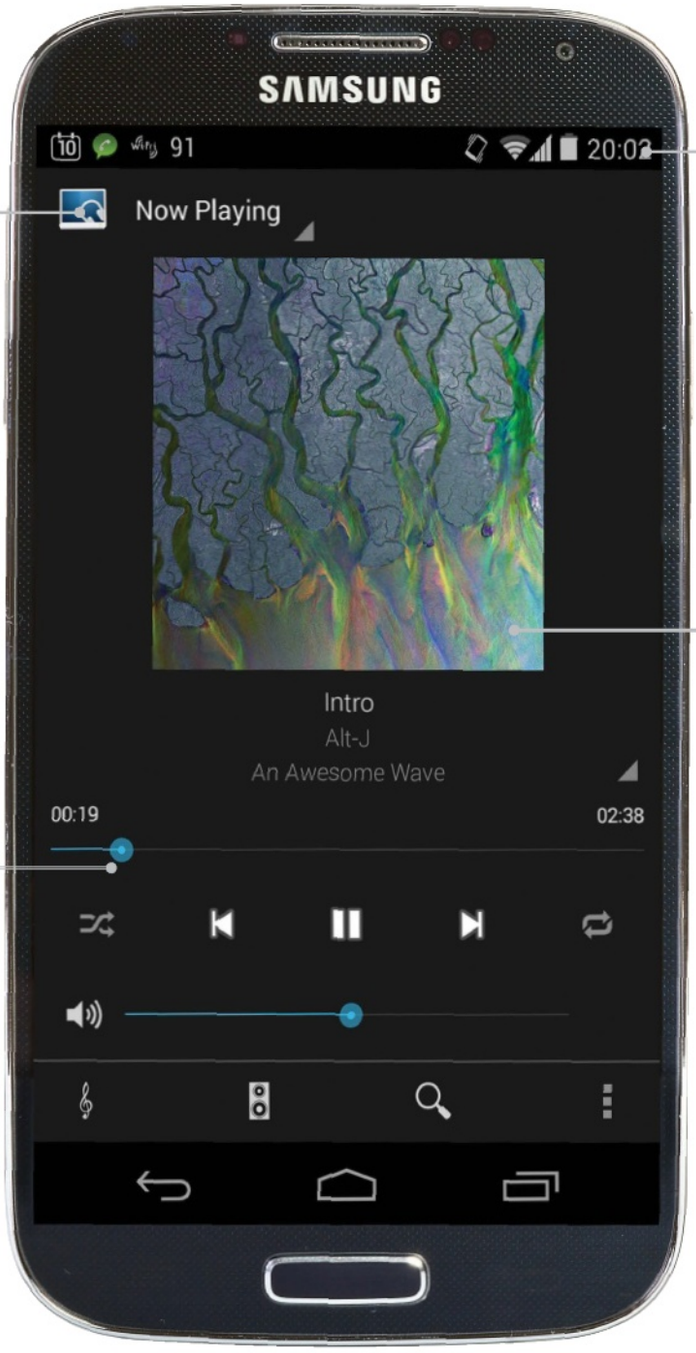
To make the comparison, we're taking a sample image and inputting it into a stream we've created using io.BytesIO. It's stored in memory and then compared with the previous image that has been saved later on in the code – this determines whether or not a new photo should be taken.

Fig 1. – Position the camera module's ribbon so that the silver side is facing the HDMI port

Full Code Listing:

```
import io
import os
import picamera
import time
from datetime import datetime
from PIL import Image

camera = picamera.PiCamera()

difference = 20
pixels = 100

width = 1280
height = 960

def compare():
    camera.resolution = (100, 75)
    stream = io.BytesIO()
    camera.capture(stream, format = 'bmp')
    stream.seek(0)
    im = Image.open(stream)
    buffer = im.load()
    stream.close()
    return im, buffer

def newimage(width, height):
    time = datetime.now()
    filename = "motion-%04d%02d%02d-
%02d%02d%02d.jpg" % (time. ↵
year, time.month, time.day, time.hour,
time.minute, time.second)
    camera.resolution = (width, height)
    camera.capture(filename)
    print "Captured %s" % filename

image1, buffer1 = compare()

timestamp = time.time()

while (True):

    image2, buffer2 = compare()

    changedpixels = 0
    for x in xrange(0, 100):
        for y in xrange(0, 75):
            pixdiff = abs(buffer1[x,y][1]
- buffer2[x,y][1])
            if pixdiff > difference:
                changedpixels += 1

    if changedpixels > pixels:
        timestamp = time.time()
        newimage(width, height)

    image1 = image2
    buffer1 = buffer2
```

"Having the camera take photos when motion occurs reduces the strain on the Raspberry Pi"



Music collection
Your music collection must be stored in the directory /var/lib/mpd/music on your Raspberry Pi. Once the daemon is set up, you can access it from any mobile device or computer

Android & iOS
While there are lots of applications suitable for this project, we recommend MPDroid for Android and MPoD for Apple's iOS devices

Server connection
Once your Android or iOS app is set up, we can connect to our music server using the Raspberry Pi's IP address

Streaming daemon
We can configure the Raspberry Pi Music Player Daemon to listen on all interfaces so we can access the music from all kinds of devices

Raspberry Pi music streamer

Remotely control a Raspberry Pi that plays your music collection and stream your music to your phone

Resources

Raspbian:
www.raspberrypi.org

A web-connected device:
Tablet or smartphone

Music Player Daemon (MPD) is a piece of software that acts as both a music player and a music server, which is handy. It is capable of outputting audio to any sound card that is connected to the system, and be controlled by an MPD client.

Clients are available for almost any platform, including iOS and Android. MPD is also able to output audio to a stream, which can then be used by most clients. This is really great for people who have rather large music libraries and cannot fit them all on their device.

01 Install the required software

Log into the Raspbian system with the username pi and the password raspberry. First, find the IP address of the Pi using ip addr | grep inet and note it down for use later. Get the latest package lists using the command sudo apt-get update. Then install MPD using sudo apt-get install mpd. There may be some errors, but you should be able to ignore those.

02 Add music

The default music directory for mpd is /var/lib/mpd/music. We will first make this folder world-readable, writable and executable so that the Pi user can write to it. Do this with sudo chmod 777 /var/lib/mpd/music. Then find some music you'd like to copy on your Linux computer and use scp to copy it. For example: scp -r Alt-J/ pi@192.168.157.28:/var/lib/mpd/music/

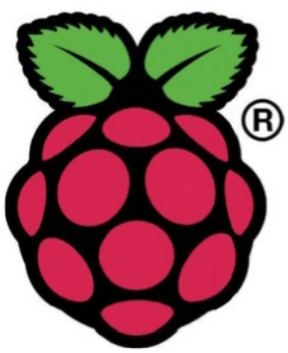
03 Fix permissions

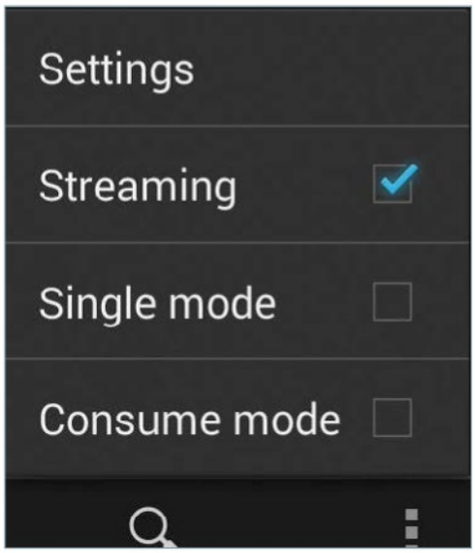
The files that we just copied will be owned by the Pi user, which isn't what we want. We're going to change the ownership of the music directory, and all subfiles/subdirectories, to the mpd user and the audio group: s**udo chown -R mpd:audo /var/lib/mpd/music**

04 Configure the daemon

We want to edit /etc/mpd.conf (using sudo). The first change is to make the daemon listen on all interfaces, so we can use MPD clients from other devices. Do this by changing the line:

```
 bind_to_address      "localhost"
to…
 bind_to_address      "any"
```





■ Stream your music library to a mobile device using a compatible client

05 Configure a stream

At the moment, the only audio output is the 3.5mm one on the Pi. To set up a stream, scroll down the config file until you find the httpd stream output that is commented out. Uncomment the entry, and change the format line to produce stereo output instead of mono. Our entry was as follows:

```
audio_output {
    type        "httpd"
    name        "My HTTP Stream"
    encoder     "vorbis"
    port        "8000"
    quality     "5.0"
#bitrate        "128"
    format      "44100:16:2"
}
```

Save the changes and restart the daemon with **sudo /etc/init.d/mpd restart**

06 Set up a client

It's difficult to walk through setting up a client on each different platform, but the steps translate fairly easily. For Linux, we suggest Sonata, for Android we suggest MPDroid, and for iOS we suggest MPoD. We're going to set up MPDroid on Android, so go ahead and download that from the Play Store.

"Music Player Daemon can output audio to any sound card connected to the system, and be controlled by an MPD client – on almost any platform"

■ This is the Sonata client for Linux

07 Connect to the server

Once in the MPDroid app, select WLAN-based connection and choose your access point. Then fill in the Host field with the IP address of your Pi and fill in the 'Streaming host' field with the same details. Everything else should be the default. Once you've done this, go to the Now Playing screen. We need to update the music library, as it has never been scanned before. To do this, press the Menu button,and go to Settings. Then select the Update option, with the caption 'Refresh MPD's Database'.

08 Playing music

Press the 'treble clef' button in the bottom-left corner in order to go to the music library. This will take you to the Artists section of the library. To play music from an artist, long-press on the artist and select 'Add, replace and play'. If you have speakers or headphones connected to the Pi, you should hear music coming out of them. Use the volume slider on the Now Playing screen to adjust the volume.

To enable the stream, press the Menu button and tick the Stream option. After about ten seconds of buffering, the sound will be coming out of your Android device.

Although this might sound like a rather long time to buffer, once you have a playlist, the device will play it absolutely seamlessly. You may be able to reduce this buffer time by looking at the improvements section…

09 Further improvements

This article has illustrated a very simple MPD setup. Further possible improvements include:

■ Putting music on an external hard drive so that you have more storage space;

■ Tweaking the streaming settings to tax the Pi's CPU less (look at the 'encoder plugins' section of the user manual at **www.musicpd.org/doc/user);**

■ Setting up a Samba share, to give access to the music files over the network.

Play retro games on your Pi

Install RetroPie and enjoy the ultimate classic-gaming experience on your Raspberry Pi

Resources

RetroPie:
http://blog.petrockblock.com/download/retropie-project-image/

Download the W32 Disk Imager:
http://sourceforge.net/projects/win32diskimager/

Installation Tips for Mac and Linux:
http://elinux.org/RPi_Easy_SD_Card_Setup

Abandonia:
www.abandonia.com/en/game/all/Abandonware/

Fans of retro gaming have been flocking to the Pi to take advantage of its comparative power, flexibility and its modest dimensions in order to create the ultimate retro gaming device.

Small enough to sit inside any device, from an old-school console to a full-sized MAME cabinet, the Raspberry Pi can be easily converted to run games from a wide selection of classic systems.

In order to play the games, of course, you must first own them. ROMs can be made from your own copies of these old games, or you can find abandonware online. Once obtained, you'll need to copy them to your SD card running the RetroPie system.

Abandonware games are those that have had their copyright run out, or the rights have lapsed and no one specifically owns them. Some companies, such as iD software, have made the original DOOM open source and public domain.

To turn your Raspberry Pi into a retro-gaming rig, you will need a minimum 4GB SD card, USB keyboard and mouse (for setup), USB gaming controller(s), network connection, suitable display and appropriate cables, a desktop computer to install RetroPie onto the SD card, and FTP software.

From beginning to end, this tutorial should take no more than 30 minutes to complete.

01 Install RetroPie

From the site listed in Resources on the previous page, download the RetroPie image and save it to your computer. You should also download and unzip the Win32 Installer tool for SD cards.

Insert your SD card into your card reader, browse to the unzipped Win 32 Installer folder, right-click Win32DiskImager.exe and select 'Run as administrator'. Check that the correct disk is selected under Device, and browse for the extracted Image File. When you're ready, click Write and wait for notice that the image has been burnt to the SD card.

If you're using Mac OS X or Linux, check the appropriate resource for details on how to install images to SD cards.

02 Boot RetroPie

Safely remove your SD card, insert it into your Raspberry Pi and switch on. The RetroPie uses EmulationStation to manage the various emulators that are installed, enabling you to easily launch game ROMs.

You should see the controller setup screen, from where you can configure your USB controller to work with the EmulationStation software. Follow the instructions to configure a controller or your keyboard, keeping a note of the controls for future reference.

This will enable you to navigate around the RetroPie setup. To calibrate for gaming, select Menu and Exit EmulationStation.

03 Calibrate the controller

We use the RetroArch configuration script in the Linux command line to configure game controllers. First, connect your USB gaming controller. Next, enter the following commands:

```
cd RetroPie/emulators/RetroArch/tools
./retroarch-joyconfig >> ~/RetroPie/configs/all/retroarch.cfg
```

Another controller calibration tool will be displayed, so follow the prompts. If you find that the tool is asking you to configure buttons you don't have on your device, use the nearest comparative option. If you plan to use an Xbox 360 or similar controller, skip to step 7.

04 Configuring retro-style controllers

To configure your controller you will need to edit the retroarch.cfg file. The quickest way to do this is to connect to the Pi via FTP. You might use FileZilla or Cyberduck – the FTP software must support SFTP (SSH File Transfer Protocol).

Find your Raspberry Pi's IP address:

```
ifconfig
```

You need to make a note of the 'inet addr', which will be in the form 192.168.x.x.

Next, run your FTP software, create a new connection and select the SFTP option; add the IP address and input the username and password as 'pi' and 'raspberry'. Browse to RetroPie>Configs> all and download and open retroarch.cfg.

05 Editing retroarch.cfg

Viewing the file in a text editor, you can make various changes that allow you to use your game controller of choice. For instance, to configure the popular USB version of the classic Super Nintendo controller, find and delete the following:

```
input_player1_l2_btn = "4"
input_player1_r2_btn = "4"
input_player1_l3_btn = "4"
input_player1_r3_btn = "4"
```

Save the config file to use the controller.

If your chosen controller doesn't have an analog stick, you should also delete:

```
input_player1_l_x_plus_btn = "x"
```

06 Alternative retro controller configuration

Rather than messing about with FTP, you can use the X graphic user interface in Raspbian to browse to the directory to make the changes to retroarch.cfg, as outlined above.

Begin by exiting EmulationStation. At the command prompt, enter: `startx`

Next, open the file manager, which you will find on a toolbar in the lower-left corner of the display. Browse to RetroPie>Configs> all, open retroarch.cfg and make your controller configuration changes.

When you're done, save retroarch.cfg and log out to return to the command line.

07 Using an Xbox controller

You can also use your Xbox 360 controller. Begin by installing the driver from the initial setup script by finding the "Install drivers for Xbox 360 Wired Controllers".

Edit **/etc/rc.local**, adding:

```
xboxdrv --trigger-as-button --wid 0 --led 2
--deadzone 4000 --silent &
sleep 1
```

The driver will launch as the computer boots. Switch '--wid' to '--id' for wired controllers. Next, open:

```
cd ~/RetroPie/emulators/RetroArch/tools
```

Here, add:

```
./retroarch-joyconfig -o p1.cfg -p 1 -j 0
```

Increase digits by one for each extra controller.

Choosing a game controller

Choose the best controller for your retro gaming

Setting up your Raspberry Pi as a retro gaming centre is a great idea. However, despite the immense flexibility of the system (EmulationStation can be manually configured to bring additional emulators into the mix, allowing you to play games from other platforms), you may find that you run into problems with choosing the right controller solution.

The options are considerable, based purely on the number of USB controllers that are currently available on the market. For instance, you might prefer to keep the tone strictly retro and rely only on arcade machine-style joysticks or NES controllers (which can be bought very cheaply online with USB connectors). On the other hand, your retro-gaming dream might be to sit playing Civilization, in which case you will of course require a mouse rather than a controller.

At the other end of the scale, however, are the modern, multi-button, twin-joystick analogue controllers that can be used with Xbox 360 and PlayStation 3 consoles. You might even fancy setting up a pair of Nintendo Wii Remotes to connect to your Raspberry Pi with a Bluetooth dongle (although this can leave you spending more time setting things up than actually playing games).

Of course, your choice depends on the platform of retro gaming that you favour.

Although joysticks and gamepads/controllers are supported, there is a lot to be said for old-fashioned keyboard control, and it is perfectly simple to set up a pair of USB keyboards for two-player action on your RetroPie.

The resulting files that are created should be added to retroarch.cfg:

```
sudo cat p*.cfg >> ~/RetroPie/configs/all/retroarch.cfg
```

Save and reboot to finish.

08 Setting up dual controllers

For the best results in two-player gaming, you should use a pair of identical controllers.

In **retroarch.cfg**, find this line:

```
input_player1_joypad_index = "0"
```

Select this and the lines that follow and copy them, leaving a blank line, then pasting the selection.

After duplicating the details of the first controller profile, edit the new block of code so that every instance of 'player1' now reads 'player2'.

For instance, the first two lines should read:

```
input_player2_joypad_index="0"
input_player2_a_btn = "1"
```

Save and close once all changes have been made.

09 Exit games without restarting

Exiting a game can be risky by default as it involves removing the Raspberry Pi power cable, which can corrupt the SD card. You will need to add some code to the retroarch.cfg file to exit:

```
input_enable_hotkey_btn = "X"
input_exit_emulator_btn = "Y"
```

Replace X and Y with the button numbers that correspond to the buttons on your controller that you want to use to exit a game and return to EmulationStation.

10 Navigating EmulationStation

With everything set up, you will be able to relaunch EmulationStation from the command line using the command:

```
emulationstation
```

Using the key mappings configured earlier, navigate around EmulationStation using the left and right controls, where you will find emulators such as an MS-DOS emulator called PC (x86). All emulators that are listed are only those that have game ROMs added to them already. To display all emulators within EmulationStation, you will need to find ROMs for them and save them to your RetroPie – this 'activates' the corresponding emulators.

11 Getting ROMs

With a lot of cartridge-based games, you are able to get readers that allow you to read the ROM – the data on the solid state memory. From there, you can get software to rip the ROM from the cartridge, which is legal for backup purposes as long as you're not breaking any DRM. A lot of these older systems work on RetroPIE, with a full list on the website.

Otherwise, a lot of software can be found as Abandonware on the internet. Here, Google is your friend, and while there's no one site that is best for getting this software, you may find a variety in your search. We've also included a link to Abandonia in the resources. In the intro we mentioned DOOM – that's already installed on RetroPie, so there's no need to hunt it down.

12 Copying ROMs to RetroPie

After obtaining your ROMs, you will need to copy them to your RetroPie. If you already have them saved on the SD card somewhere, meanwhile, they may need moving to the correct folder.

Using your FTP software, connect to the Raspberry Pi with the IP address, username and password, as explained in step 4.

Browse to the RetroPie>roms folder, where you will see a collection of subdirectories, each labelled according to the emulator that uses them. All you need to do now is to copy the ROM files from your computer to the corresponding directory on your RetroPie gaming station.

When done, you'll be ready to play!

13 Launching a game

Starting a game on your RetroPie means cycling through the list of emulators left and right, then using the up/down control to find and select the game you want to launch.

Each emulator has its own 'screen' of this very large, media centre-style menu system, and within each menu you will find the available game ROMs that you can load up and play.

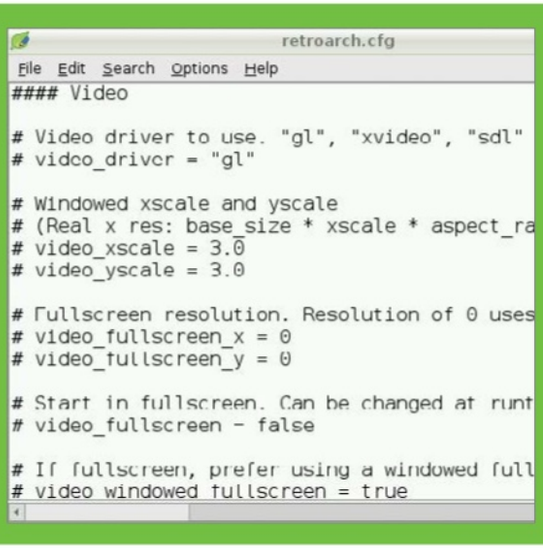
Using the controller as configured when you first booted up your RetroPie, you can then navigate around the emulators to find the game you want, scrolling up and down through menus and making your selection.

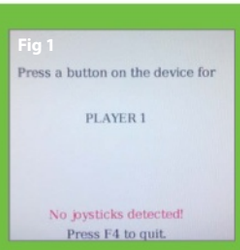
As you add more and more ROMs, so the menus will increase in size, so take advantage of the 'Jump to letter' option that you set on your controller for fast navigation.

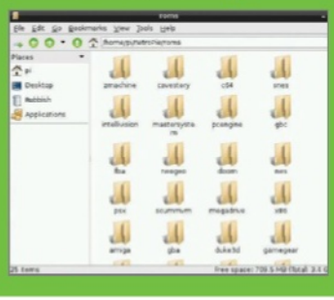
14 Configuring EmulationStation with retroarch.cfg

If you're used to Windows or Mac OS X or even some of the more aesthetically pleasing versions of Linux, you'll probably be happy enough using checkboxes and forms for setting up drivers, adjusting audio and video settings and generally customising things to your liking when using applications.

Not so on the Raspberry Pi. When using the EmulationStation for your RetroPie gaming centre







■ Fig 1: Working through the joystick/ controller configuration screen will enable you to navigate around the EmulationStation platform. However, separate controller calibration will be required before you can play games

■ Fig 2: When you boot your RetroPie you will see this splash screen, a 16-bit themed graphic that hints at the retro gaming fun that is set to follow…

project, tweaks and adjustments you make must be entered and applied directly into the **retroarch.cfg** file.

At first glance, this may not seem an ideal way of making the changes you need to run your emulated games successfully, but in actual fact, editing the config file can prove extremely useful in testing and assessing the changes you make to things like audio and video settings.

15 Managing audio on RetroPie

There is a chance that the emulator or ROM that you choose to use fails to run the audio correctly. If this happens to you, be aware that the problem is not with the Raspberry Pi itself, but with the audio driver (although you should spend time checking your audio cabling to and from the mini-computer).

In the event that this turns out to be a problem for you, connecting via FTP or booting your RetroPie into Raspbian and editing the retroarch.cfg file will help you to troubleshoot and resolve the problem.

Find the section labelled '## Audio' and alter the settings as appropriate.

For instance:

```
audio_latency = 64
```

…will adjust the audio latency in milliseconds – which should prove very useful if sound is out of sync with the video.

16 Adjust video settings

Similarly, you might find that you have problems when you are viewing games that are presented through the EmulationStation emulators due to display issues.

Once again you should connect to your RetroPie box via FTP to open and edit **retroarch.cfg** (or browse to the folder using Raspbian – see above for details on how to achieve this), this time looking for the section headed '## Video' (without quotes).

Although you might like to change video drivers, the most common adjustments are to the resolution of the display. Old systems use a much lower resolution so appear even blockier when presented on modern displays.

Find and adjust the following instructions to change the resolution (0 is default):

```
# video_fullscreen_x = 0
# video_fullscreen_y = 0
```

17 More video display tweaks

Many more video display tweaks can be used to fix problems.

For instance, you might prefer the system to boot into a non-full-screen display:

```
# video_fullscreen = false
```

> "You could even go one step further and build a full-sized cabinet"

Toggling the vsync setting can also prove useful:

```
# video_vsync = true
```

Meanwhile, reduce latency in the video settings with:

```
#  video_hard_sync = false
```

Other audio/video latency problems can be addressed by finding and adjusting the following:

```
# video_refresh_rate = 59.95
```

Note that there are several video settings in the retroarch.cfg file, and changes that you make should be carefully noted and subsequently tested. Some may cause stability problems, so take care when making adjustments.

18 Buy a case for your Pi

Although the Raspberry Pi ships as a small computer board with the minimum two USB ports, HDMI out, Ethernet and the SD card slot, it often isn't the first thing on the new Pi owner's mind to get a case for the device.

A vast supply of cases are already available, however, ranging from downloadable card constructs, builds made from popular plastic bricks and hobby boxes to those designed perfectly to accommodate every connector and GPIO slot.

As a retro gaming unit, your Raspberry Pi is at strong risk of being knocked around. Its default case-free position is often being suspended by the assortment of thick cables that are attached to it – but given the activity that gaming can inspire in people, a case is a very good idea.

19 The standard Pi case

We've established that you need a case for your RetroPie setup – but where do you start?

If budget is an issue, you might want to consider some of the suggestions above. The design for the cardboard case can be downloaded from the official Raspberry Pi website, while the dedicated cases can be purchased for a few pounds.

The problem with a small case, however, is similar to not having a case at all. You will probably experience difficulty in keeping the Raspberry Pi positioned exactly where you want it. In this situation your best solution is to attempt to fix it in place. This might be done with flat-head screws (some cases have slots to accommodate this) or using adhesive hook-and-loop fastener strips to 'lock' the computer to your desk or table.

20 Customise a console

Another good solution is to take an existing retro games console, remove the internals and fit the Raspberry Pi inside. There are various websites that explain how to do this, but the general principles are to fix the Pi in place securely without damaging the board, and to use extender cables to connect to the existing ports.

For instance, with a Nintendo NES case, you'd connect to short USB extension cables from the Raspberry Pi to where the original controller ports were fixed, gluing the USB cables in place. Alternatively you might mount a bare USB hub.

When using a USB hub with the Raspberry Pi, make sure you use a powered device as the Pi doesn't provide enough electricity for an unpowered hub.

21 Dedicated gaming case

Finally, you might prefer a dedicated solution to your RetroPie storage requirements. After all, the Raspberry Pi is an affordable computer so leaving it purely as a retro game station isn't an unrealistic prospect.

If this is a scenario you're considering, you should be looking at building a permanent, dedicated retro gaming case, taking advantage of existing cases, hobby boxes or even developing a custom design.

Your RetroPie box might, for instance, have a couple of built-in retro joystick controllers. You could even go one step further and build a full-sized cabinet, big enough to mount a monitor in, or even purchase an old arcade machine cabinet (standing or horizontal) and use this as your RetroPie-powered retro game station.

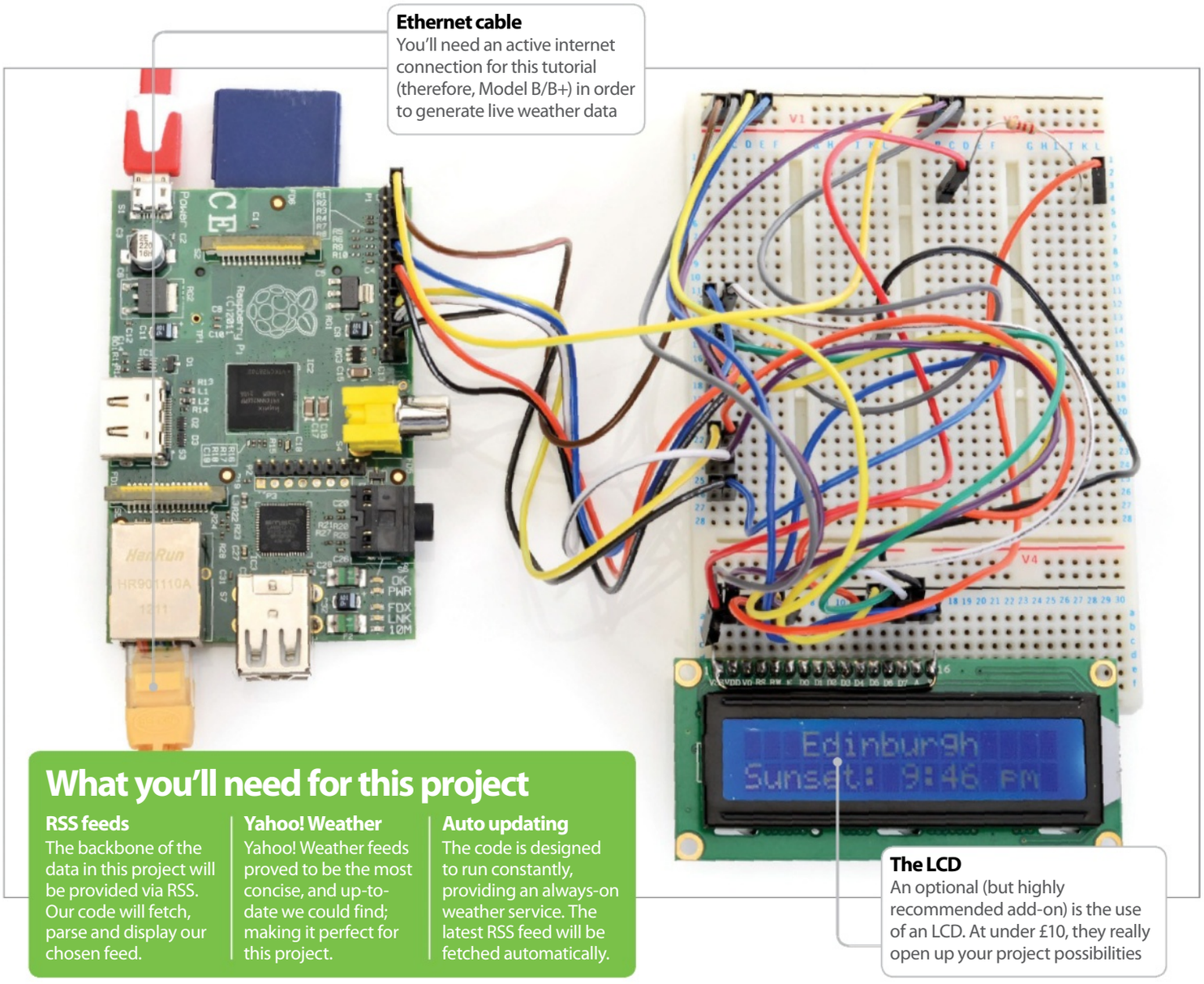
As with everything Raspberry Pi-related, the possibilities are endless.

22 Your retro gaming hub

With your favourite retro games downloaded copied to your Raspberry Pi 'RetroPie' and your USB game controllers connected to your mini-computer, you'll be just about ready to start using the device as your all-in-one retro gaming hub.

Browsing through the emulators is fast, and loading up the games is usually completed in a fraction of the time of their original release – particularly for those that were first available on cassette!

The power demands of the device are low, but you will perhaps find that there are various audio and video settings that you might need to consider for optimum enjoyment, particularly of the older titles. Perhaps best of all is the wonderful feeling of liberation that you get from playing a PlayStation game on the modest Raspberry Pi hardware!



Ethernet cable
You'll need an active internet connection for this tutorial (therefore, Model B/B+) in order to generate live weather data

The LCD
An optional (but highly recommended add-on) is the use of an LCD. At under £10, they really open up your project possibilities

What you'll need for this project

RSS feeds
The backbone of the data in this project will be provided via RSS. Our code will fetch, parse and display our chosen feed.

Yahoo! Weather
Yahoo! Weather feeds proved to be the most concise, and up-to-date we could find; making it perfect for this project.

Auto updating
The code is designed to run constantly, providing an always-on weather service. The latest RSS feed will be fetched automatically.

Construct a digital weather station

Learn to produce a localised weather feed with an LCD output for a low power weather station

Resources

Yahoo! Weather Feed
http://developer.yahoo.com/weather/

HD44780 LCD controller
http://en.wikipedia.org/wiki/Hitachi_HD44780_LCD_controller

Full Source Code
http://neil-black.co.uk/raspberry-pi-magazine-articles-code

When making our weather station we're going to split the process into two parts. Part one will be building the software and displaying the weather information on screen for those of us who do not yet have the LCD display. Part two will be connecting up the LCD and editing our code to display the information there. For the software side, our information will be coming in the form of an RSS weather feed from Yahoo!. This proved to be an excellent source of weather information, very concise and updated many times during the day. Head over to **http://weather.yahoo.com/** find your local area; it covers a massive selection of areas for all over the world.

Next we'll be looking at the LCD. After a lot of research we opted for the HD44780 LCD, it's small and low powered and has existing support from the Raspberry Pi community. You'll find you won't have to worry about breaking the bank during the build, and it should prove to be rather painless.

The two parts will integrate nicely with the end result being your local weather information (minimum temperature, maximum temperature, humidity, sunrise and sunset) being displayed constantly on the LCD and updated at regular intervals. But it doesn't stop there, as there's plenty of scope for your own hacks in both the software and hardware on this project.

01 Setting up the environment

Before we begin writing any code we need to install the FeedParser package for Python. We're going to do this the easy way and use PIP, a tool for installing and managing Python packages. But first we need to install the Python Setup Tool. Once installed this will give us access to the Easy Install command which can then be used to install PIP. Lastly we use PIP to install FeedParser. If, during this entire process, you are prompted along the way, enter "y".

```
sudo apt-get install python-setuptools
sudo easy_install pip
sudo pip install feedparser
```

02 Creating a new project

Now we can give our project a place to live. Create a directory in your home user account (default is /home/pi). For example /home/pi/WeatherStation. Remember where this is, we'll need it later on. We're going to be using the command line editor Nano to write our code, so create a new file with the .py extension; sudo nano /homr/pi/WeatherStation/WeatherStation.py. Use Ctrl+X to exit and "y" to confirm file name and save.

```
sudo mkdir /home/pi/WeatherStation
    sudo nano /home/pi/TwitterBot/
WeatherStation.py
```

03 Importing modules

Now we can begin to start writing the code. Start with the shebang, the #! location of Python, followed by importing the modules time, feedparser and os. We'll be using time to tell our code when to update the weather information, feedparser to fetch and parse our information and os will be used to clear the prompt for users with no LCD.

```
#!/usr/bin/env python
#import
import time
import feedparser
import os
```

04 Setting our feed

We're going wrap our main body of code in an infinite while loop and we'll decide later how often the loop iterates. But before we do anything else we need to get the correct location for our weather feed. Head over to **http://weather.yahoo.com/** and enter your city. When you land on the weather page for your chosen location take a look at the url, you'll see at the end a set of between four and seven digits; this is your location code. Now we just need to set up the RSS feed url in the following format http://weather.yahooapis.com/forecastrss?w=[lcoation code]&u=[unit]. Enter your location code and a value of c or f (Celsius or Fahrenheit) for u. Don't include the square brackets.

"We're going to be using the command line editor 'Nano' to write our code"

```
while True:
rss_link = 'http://weather.yahooapis.com/
forecastrss?w=19344&u=c'
```

05 Fetching the data

So we know where our RSS feed lives now, we just need to pull that information into the code. Luckily feedparser makes this whole process very simple. We use the method feedparser.parse passing in the variable rss_link where our custom weather feed URL is being stored. We've store this in an object called d, all the information we need is in here. That's it! It really is that simple to pull in the remote data, not only that but feedparser has already parsed it ready for our use.

```
d = feedparser.parse(rss_link)
```

06 Extracting the data

As the data has already been parsed we can pull most of what we need right away. In the object d we have the feed details in a dictionary (more about that in the next step) as defined in the feed. The ones we're interested in are yweather_location (contains the locations dictionary), yweather_atmosphere (contains the atmospheric conditions dictionary) and yweather_astronomy (contains the astronomy dictionary). Te acquire these we will use feedparser as feedparser allows us to take a simple structured approach to accessing this data.

```
location  = d.feed.yweather_location
    atmosphere = d.feed.yweather_atmosphere
    astronomy = d.feed.yweather_astronomy
```

07 Accessing the data dictionaries

In the last step we pulled out the dictionaries we want to access for our weather information. Dictionaries are very useful data structures in Python. Think of them as lists (much like an array) but each item in the list is actually a pair; the data and the name of that data. So city = location['city'] will pull the value for the data named "city" in the dictionary we stored in location. We do exactly the same for humidity, sunrise and sunset in their associated dictionaries.

```
city = location['city']
humidity = atmosphere['humidity']
sunrise = astronomy['sunrise']
sunset = astronomy['sunset']
```

08 What shall we do about temperature?

Yep, we've missed probably the most important data that people want in a weather station, the temperature. These aren't in the feed details but in the feed entries which again can be easily accessed with feedparser. Let's try printing the high temperature with d.entries[0].yweather_forcast['high']. Again we traverse the data and pull the element needed for the dictionary. But wait! That's not today's high temperature, it's tomorrow's. If you look at the feed you'll see two entries for weather_forecast, today's and tomorrow's. We should be able pull out the first

in an array, but fortunately feed parser falls over here. But that's okay, we can go the long way. Let's start but pulling in the summary.

```
summary = d.entries[0].summary
```

09 Splitting data with Python

Summary has all the data we need in the following format: ... Fri - Sunny. High: 28 Low: 14
 Sat. So what we want to do is pull the two values from the string. Python includes a very handy way of helping us do this with a method called splitting. For example, take the string "one, two, three" and pretend it's a variable called "test". Using test.split(",") will split the string test into different sections every time it sees the substring ",". We would be left with an array called "test" that we can use to pull each element. For example, print test[0] would print "one", print test[1] would print "two" and so on.
So we can start to do the same with our string called summary.

```
temp = summary.split("High: ")
```

10 Getting the high temperature

We've split our summary into sections on the string "High :" and stored that in an array called temp. The spilt will remove the string we matched on. The first piece of data we want is the high temperature for the day, that's going to be in the second chunk of data that we split for the original source, so we can access that with temp[1]. If you print that data you'll see it has the value we need at the beginning, but a whole load after it. So let's split again on the substring right after the value we need; " Low: ". Grabbing the first element from this will give just the data we need.

```
temp = temp[1].split(" Low: ")
high = temp[0]
```

11 Getting the low temperature

Finally we can get the last piece of data we need, the low temperature. After our last split we got the high value in temp[0]. The data we want now is in the same array, but in the next element. Try printing temp[1] and again you'll see the data we need at the start followed be more text we're not interested in. As last time, we just need to do one final split on the text right after the data we want, in this case the HTML line break "
". This will return the low temperature value in the first element of the new array.

```
low = temp[1].split("<br />")[0]
```

12 The OS library

So we have all the data we need all ready to be displayed. At this point we're just going to show that information in the terminal window, we'll get into the LCD later. The problem with this approach is the terminal window will soon become very cluttered and hard to read, ideally we need to clear that window at the beginning of ever iteration of our while loop. That's where the library os comes in, it effetely allows access operation system dependant functionality from within your Python code. This is incredibly useful and you'll probably find yourself using this time and time again. In this case we're just going to pass it the parameter "clear". This does exactly the same as it would if entered directly at the command prompt; it clears all the left over junk from the screen.

```
os.system('clear')
```

13 Printing the data

Now we just have to print each one of our variables on screen. As you can see we've hardcoded some static text and concatenated the variables using +. Note that the terminal output is mono spaced, meaning you can easily align your variable output in the code with spaces and know it will line up when the script is run.

```
print ("Weather for "+city)
print ("Low Temp:  "+low)
print ("High Temp: "+high)
print ("Humidity:  "+humidity)
print ("Sunrise:   "+sunrise)
print ("Sunset:    "+sunset)
```

14 Refreshing the data

The final part to our code for this version of the weather station is the sleep function, ie how long to wait until we go through the whole while loop again, refresh the data from the source and output the results on screen. For the non-LCD version we've chosen five minutes (300 seconds). This should be more than adequate for most people's needs and also means we're not hitting the free Yahoo! feed too often. Finally, to run the script simply point Python to the script you created, for example sudo python /home/pi/WeatherStation/WeatherStation.py.

```
time.sleep(5)
```

15 Connecting the LCD

Hopefully you've managed to source one of the excellent HD44780s around eBay or Amazon. We went for the 16 character by two row model at just under £10, quite a bargain. If you fancy some more room to move around in, go for the larger four row by 20 character models, although expect to pay at least twice as much. Follow the wire mapping in the image provided and you shouldn't have any problems, however there are a few caveats to note. LCD Pin 15 can connect to the +5V on the Pi (Pin 2). If you wish to use the backlight, we would recommend the use of a resistor to control the brightness and prevent it from burning out. Similarly when connecting the

Inside of the board

Header	Pin	Pin	Header
+3.3 V	1	2	+5V
I²C SDA	3	4	Do Not Connect
I²C SCL	5	6	Ground
General Purpose Clock	7	8	UART Transmit (TXD)
Do Not Connect	9	10	UART Receive (RXD)
GPIO Pin 17	11	12	GPIO Pin 18
GPIO Pin 21	13	14	Do Not Connect
GPIO Pin 22	15	16	GPIO Pin 23
Do Not Connect	17	18	GPIO Pin 24
SPI MOSI	19	20	Do Not Connect
SPI MISO	21	22	GPIO Pin 25
SPI SCLK	23	24	SPI Chip Select 0
Do Not Connect	25	26	SPI Chip Select 1

Our code uses the pin numbering under the heading "Pin" in the green column

■ Above: A sample of the Yahoo! RSS feed. It's this feed that will provide all your local weather data for this project

■ Left: Mapping for the Raspberry Pi's GPIO functions to the pin numbering used in the tutorial. It's the numbering in green we use

■ Top right: Mapping from the LCD functions and pin numbering to that of the Raspberry Pi. Use the middle columns in green to see what pins map

LCD Function	LCD Pin	Pi Pin	Pi Function
Ground	01	06	Ground
+5V	02	02	+5V
Contrast (via resistor or potentiometer)	03	06	Ground
RS	04	26	SPI Chip Select 1
RW	05	06	Ground
E	06	24	SPI Chip Select 0
Data 0	07		
Data 1	08		
Data 2	09		
Data 3	10		
Data 4	11	22	GPIO Pin 25
Data 5	12	18	GPIO Pin 24

contrast, if you can't find a potentiometer you can try different rated resistors to get the required effect.

16 Adding the LCD code

You can download our code for this section if you want to see the full LCD driver code in action with the rest of the code. Alternatively head over to Raspberry Pi Spy to get an idea how this code should look and what it will consist of. The code should be inserted right at the top of the code, right after your import section. Note that we no longer need to import os so that line can be removed. Also the LCD driver code imports time, we only need to do this once, so one of them can be removed.

17 Cleaning up the output

We're going to be making heavy use of the GPIO here and of course the LCD itself. This can make for some errors and warning when trying to connected to a GPIO Pin connection that's not been closed correctly or some left over output on the LCD. So right at the top, after the import section run the GPIO modules built-in cleanup function, this will ensure everything is reset and we'll not get any errors. Now head back down, after the LCD code and before our weather code. Call lcd_init() to clear all the output on the LCD.

```
GPIO.cleanup()
...
lcd_init()
```

18 Exiting gracefully

We're going to wrap our main while loop in a try statement and enter a condition of KeyboardInterrupt. This will continue to run the code indefinitely unless we see some keyboard input (Ctrl+C) to kill the code. At that point we'll run the clean up commands before exiting.

```
try:
    while True:
    ...
except KeyboardInterrupt:
    lcd_init()
    GPIO.cleanup()
```

19 Controlling the feed

We're going to want the LCD to be updated every few seconds, but we want to be kind to feed source and not hit it too hard (it's good etiquette). So we'll add a simple counter here that says to only pull the feed on every 300 iterations of the loop (around every 20 minutes).

```
i = 0
try:
    while True:
    rss_link = 'http://weather.yahooapis.com/
forecastrss?w=19344&u=c'
    if i == 0 or i == 300:
        d = feedparser.parse(rss_link)
        i = 1
    i += 1
```

20 Removing the old output

The old code we created to clear the terminal window and print all the weather data there is now redundant. We're going to replace that with the LCD output. So remove or comment out the os.system command along with all the print statements. We can also remove the sleep timer.

```
    #os.system('clear')
    #print ("Weather for "+city)
    #print ("Low Temp:  "+low)
    #print ("High Temp: "+high)
    #print ("Humidity:  "+humidity)
    #print ("Sunrise:   "+sunrise)
    #print ("Sunset:    "+sunset)
    #time.sleep(300)
```

21 Displaying data

lcd_byte sets up the line we're going to display on, one is the top and two is the bottom. lcd_string allows us to send the text we want to display. We decided we would like to display the city on line one and the different data on line two; note how we add the data we pulled from the feed before. lcd_string also accepts parameters for justifying the text. One for left, two for centre and three for right. Finally we sleep for four seconds before printing the next piece of data to the LCD.

```
        lcd_byte(LCD_LINE_1, LCD_CMD)
        lcd_string(city, 2)
        lcd_byte(LCD_LINE_2, LCD_CMD)
        lcd_string("Low Temp: "+low, 2)
        time.sleep(4)
```

22 Display more data

With our code now working we're going to set the script to run at start up. This will allow you to create a standalone device that just need to be plugged in and no user interaction is required to start displaying the data. To do this, edit rc.local; this file contains commands that will be run at boot up. On the last line before exit 0 add the python command to run the script, remember to include sudo.

```
    sudo nano /etc/rc.local
    ...
    sudo python /home/pi/WeatherStation/
    WeatherStation.py
```

"We're going to set the script to run at start up"

The LCD driver software

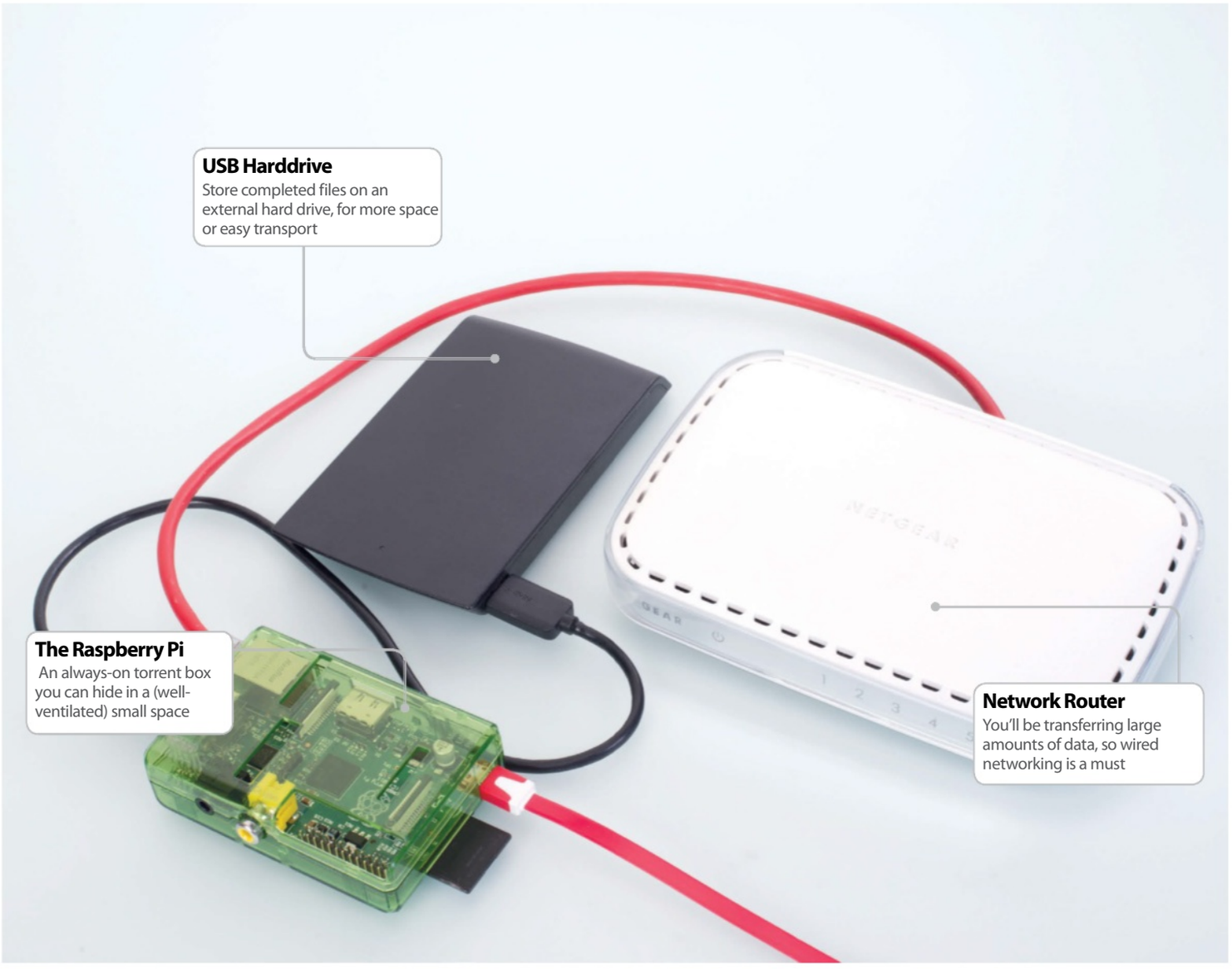
Getting your LCD to work in Python is easy

Getting the HD44780 to work with the Raspberry Pi is now thankfully very easy. We did a lot of research around this and there are various modules available for Python that are designed for this very purpose (the most obvious one being "hd44780"). However, the most complete and robust solution we found was over at Raspberry Pi Spy, not so much as an installable package as some very concise and well written code you can place any of your Python scripts using the LCD. The LCD actually supports 8-bit and 4-bit modes, but the author has chosen 4-bit mode in order to save a GPIO pin, but he's done this without losing any of the functionality. You'll find full and complete instructions in the blog for both wiring the LCD and using the code. Once you have the code in place it's incredibly easy to use. To write on line one simply use lcd_byte(LCD_LINE_1, LCD_CMD) followed by lcd_string("This is line 1",1). For line two simply use lcd_byte(LCD_LINE_2, LCD_CMD). lcd_byte takes three option's parameter values of one, two and three. These left, centre and right justify the text respectfully. It really is that easy to get data displayed on the LCD. And to clear data? It's as easy as lcd_init(). Finally you'll also find optional wiring for features such as turning the back light on and off and controlling the contrast. The backlight wiring can even be controlled in the software.


Python

Here is the updated code :

```
#!/usr/bin/python
#
# HD44780 LCD Test Script for
# Raspberry Pi
#
# Author : Matt Hawkins
# Site   : http://www.raspberrypi-spy.co.uk
#
# Date   : 03/08/2012
#

# The wiring for the LCD is as follows:
# 1 : GND
# 2 : 5V
# 3 : Contrast (0-5V)*
# 4 : RS (Register Select)
# 5 : R/W (Read Write)       - GROUND THIS PIN
# 6 : Enable or Strobe
# 7 : Data Bit 0             - NOT USED
# 8 : Data Bit 1             - NOT USED
# 9 : Data Bit 2             - NOT USED
# 10: Data Bit 3             - NOT USED
# 11: Data Bit 4
# 12: Data Bit 5
# 13: Data Bit 6
# 14: Data Bit 7
# 15: LCD Backlight +5V**
# 16: LCD Backlight GND

#import
import RPi.GPIO as GPIO
import time

# Define GPIO to LCD mapping
```



Build an always-on torrent box

Get the latest distros, packages and test builds faster with a low-power, mini torrent box

Resources

Raspbian Image:
http://www.raspberrypi.org/downloads

W32 Disk Imager:
sourceforge.net/projects/win32diskimager

Cross-Platform Raspbian installation:
elinux.org/RPi_Easy_SD_Card_Setup

Deluge:
deluge-torrent.org/

Torrenting has got a bad reputation, however it's a fantastic tool for downloading and sharing legitimate files and other content. Obtaining open source software this way has a number of advantages – it can be faster, alleviates bandwidth and allows you to share back with the community. Distros, packages and more are available via torrents, and the Raspberry Pi makes for a tiny, low-wattage, always-on torrent box to better manage your files.

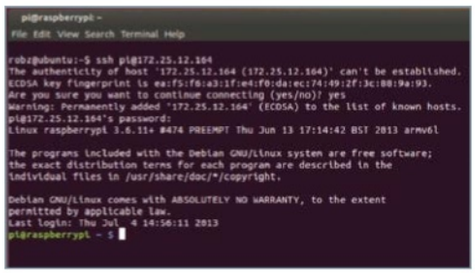
It can't do it on its own though – afterall, the average Raspberry Pi SD card will only be between 2GB and 8GB in size. You'll quickly run out of space, which is where an external hard drive comes in.

Portable, USB hard drives are the perfect companion to your Raspberry Pi's storage needs. Not only do they not require an extra power supply, they don't need much more than what the Raspberry Pi can already provide via its USB ports. This way, the entire set-up can run off one plug socket, and draw very little electricity in the process.

Some websites provide feeds for torrents as well, so with the right set-up, you can always have the most up-to-date package or ISO as soon as they're available. This also allows you to share the content for everyone else to use, ensuring a speedy and efficient download process for all.

01 Install Raspbian

Raspbian works just fine for our torrent box. Install the image on an SD card and go through the basic setup process, making sure that you enable SSH in the advanced options and to disable the desktop.



02 Remote access

Type ifconfig into your Pi's command line to find the IP address. At this point you can unplug the monitor and set it up remotely, but either way you can now access the Pi by typing:

```
$ ssh [user]@[IP address]
```

…and entering your password to log in.

03 Mount hard drive

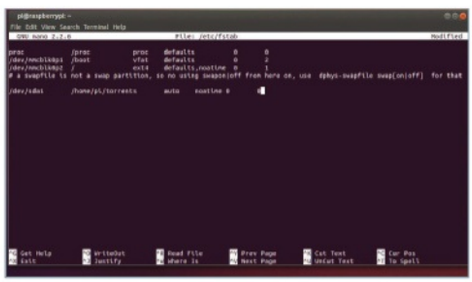
Unless you plan to reformat your portable drive, you'll need to install NTFS support onto your Pi. Type:

```
$ sudo apt-get install ntfs-3g
```

Add the hard drive to **/etc/fstab** (open it with sudo nano /etc/fstab) by adding the line:

```
/dev/[hard drive address] [mount point] auto noatime 0 0
```

Use fdisk in order to find the name of the storage, and then create a mount point such as /home/pi/torrents with mkdir. Reboot for it to mount.



04 Install Deluge

We'll use Deluge for our torrents. Install it with:

```
$ sudo apt-get install deluged deluge-console
```

Now start and then stop Deluge so it creates a config file we can edit with:

```
$ deluged
$ sudo pkill deluged
```

And finally, run the following to copy the config file in case we mess up:

```
$ cp ~/.config/deluge/auth ~/.config/deluge/auth.old
```

05 Basic configuration

Edit the file with:

```
$ nano ~/.config/deluge/auth
```

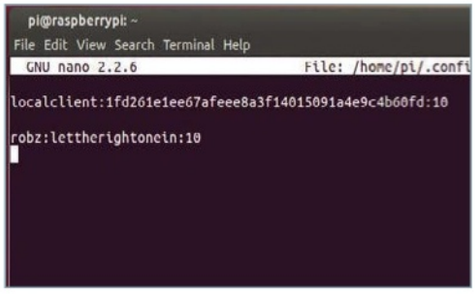
And add to the bottom:

```
[user]:[password]:10
```

…to restrict access.

Now start it up with:

```
$ deluged
$ deluge-console
```



06 Remote connection

Now you're in the client, type the following three commands:

```
config -s allow_remote True
config allow_remote
exit
```

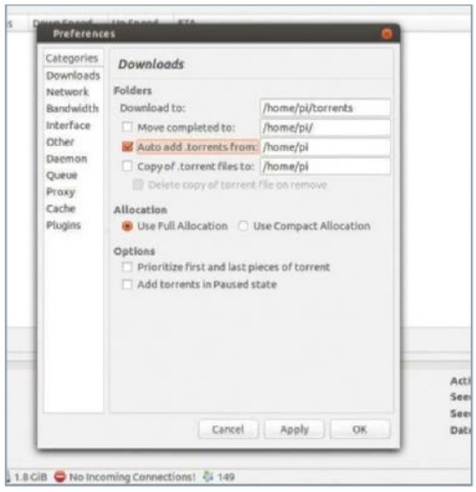
Restart the Deluge daemon with:

```
$ sudo pkill deluged && deluged
```

Now open the graphical client on your Linux PC.

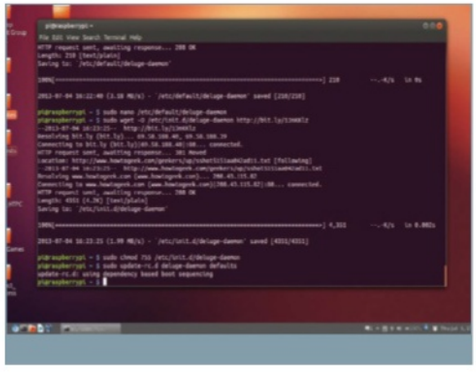
07 Remote interface

Go to Edit>Preferences>Interface, then disable Classic Mode and restart Deluge. Click Add on the Connection Manager, and enter the IP in Hostname and the user we set up earlier. Click Connect to see any torrents you have downloading or uploading.



08 Download location

Go again to Edit then Preferences, and change to the Downloads tab if it's not on there already. Set the download location to the directory we mounted the hard drive to, and enable 'Auto add .torrents', setting it to any destination if you plan to dump torrent files to the Pi.



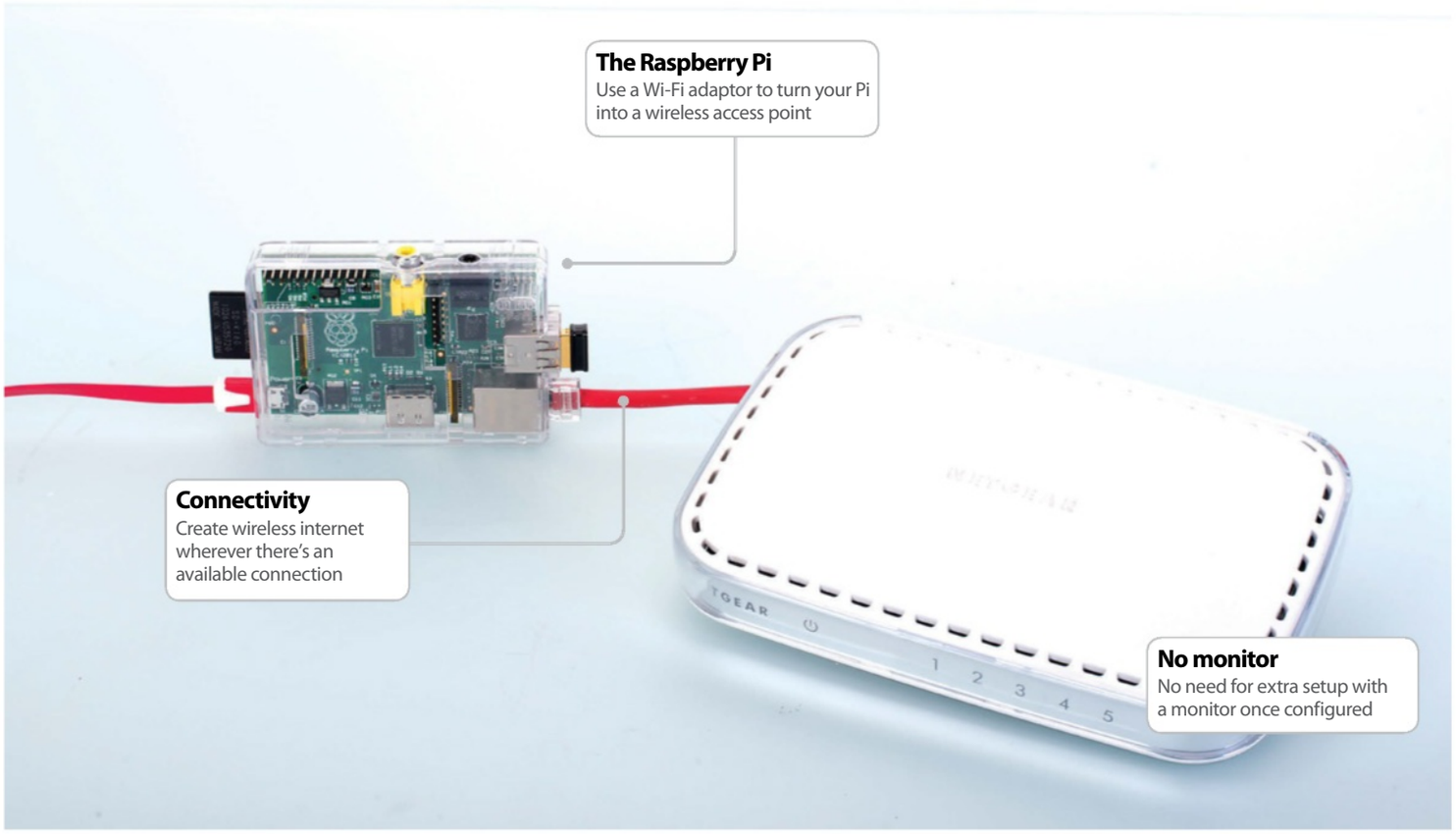
09 Start on boot

An init script from Ubuntu can be used to have Deluge start on boot. Download it with:

```
$ sudo wget -O /etc/default/deluge-daemon http://bit.ly/13nKOSj
```

Open **/etc/default/deluge-daemon** with nano and change the username to the one we set up earlier. Save it, then download the full init script and update with:

```
$ sudo wget -O /etc/init.d/deluge-daemon http://bit.ly/13nKKlz
$ sudo chmod 755 /etc/init.d/deluge-daemon
$ sudo update-rc.d deluge-daemon defaults
```



Create a portable wireless access point

With the help of a Wi-Fi adaptor, turn your Raspberry Pi into a wireless access point for other devices

Resources

Raspbian Image:
http://www.raspberrypi.org/downloads

W32 Disk Imager:
sourceforge.net/projects/win32diskimager

Cross-Platform Raspbian installation:
elinux.org/RPi_Easy_SD_Card_Setup

Adafruit Access Point:
learn.adafruit.com/

The Raspberry Pi's portability is an incredible asset to the tiny SoC, allowing people to put it anywhere around the house, or indeed in any project that needs some computing power.

What about using it outside the home though? Well, you could put it in a variety of outdoor or office projects, but the board itself has a variety of uses on its own. With a bit of prep, it could be a portable PC, however an overlooked and very useful method of utilising the Pi is by turning it into a portable WiFi hotspot.

While travelling, you might not always have available wireless internet. Whether you are visiting less technically-adept family members, or at a hotel for a business trip, the odds of getting a decent wireless connection can vary wildly. There may not even be a wireless access point.

Internet in these situations may be limited to an ethernet connection, limiting your mobility and keeping you stuck in one spot. With the Raspberry Pi and a wireless dongle, you can create a secure, wireless router that distributes internet to all your laptops, phones, tablets and other mobile devices. The best part is, it takes up a tiny amount of luggage space, and doesn't require any extra configuration after the initial setup.

The Raspberry Pi is robust enough to handle multiple devices connecting at once, so give it a try.

01 Install Raspbian

For this project, we can use Raspbian to power our access point. Install the image on an SD card and go through the basic setup process, making sure to enable SSH. You can also turn off the desktop during setup as well if you don't plan to use it.

02 Connect through SSH

Find the IP address of your Raspberry Pi by typing ifconfig into the command line, and make a note of it. Turn off the Pi, plug in your wireless adaptor, and turn it back on. In a networked computer's terminal, type:

```
$ ssh [user]@[IP address]
```

03 Install DHCP

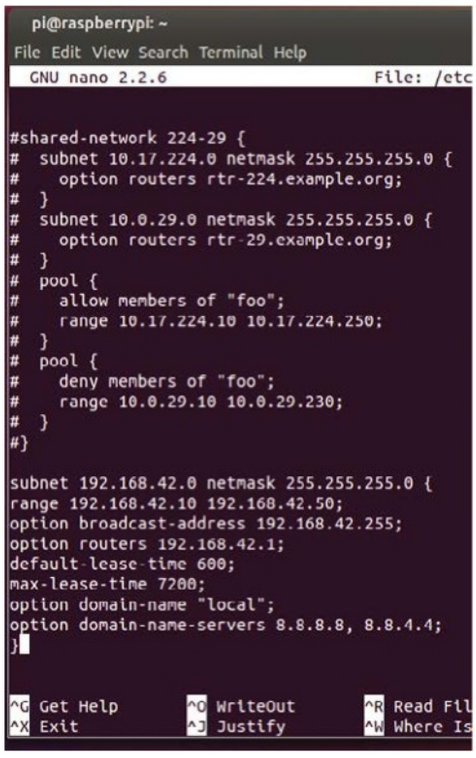
Install a DHCP server to your Pi with:

```
$ sudo apt-get install hostapd isc-dhcp-server
```

Now we need to set it up. Edit the configuration file with:

```
$ sudo nano /etc/dhcp/dhcpd.conf
```

And start by putting a # in front of the two option domain-name entries, then remove the # in front of 'authoritative;', seven lines down



04 Server address

At the end of the configuration file, add the following lines:

```
subnet 192.168.42.0 netmask 255.255.255.0 {
range 192.168.42.10 192.168.42.50;
option broadcast-address 192.168.42.255;
option routers 192.168.42.1;
default-lease-time 600;
max-lease-time 7200;
option domain-name "local";
option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

Save and exit.

05 Disable Wi-Fi

Edit the server more with:

```
$ sudo nano /etc/default/isc-dhcp-server
```

Set INTERFACES to 'wlan0' and save. Now open:

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

Put a # in front of 'iface wlan0' and the following lines with 'wpa roam', 'iface default' and any others affecting wlan0.

06 Enable access

After the line 'allow-hotplug wlan0', you need to enter the following:

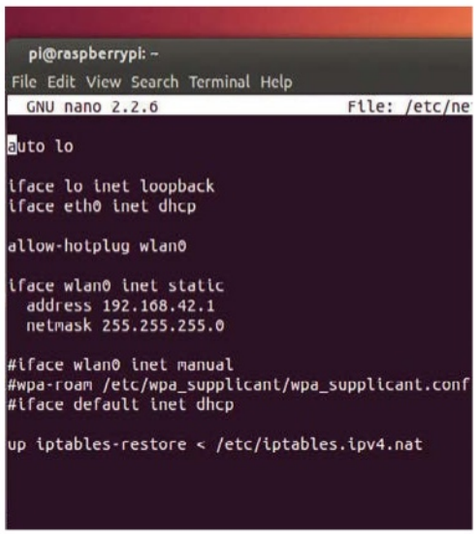
```
iface wlan0 inet static
        address 192.168.42.1
        netmask 255.255.255.0
```

Save and exit, then set wlan0's address with:

```
$ sudo ifconfig wlan0 192.168.42.1
```

Now create a new file to use to start creating the wireless network:

```
$ sudo nano /etc/hostapd/hostapd.conf
```



07 Wireless networking

You create your wireless network with the following code:

```
interface=wlan0
driver=rtl871xdrv
ssid=[access point name]
hw_mode=g
channel=1
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=[password]
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Save and exit. Now edit **hostapd** to point it to this new file with:

```
$ sudo nano /etc/default/hostapd
```

And then add:

```
/etc/hostapd/hostapd.conf to DAEMON_CONF=""
```

08 Network addressing

Run:

```
$ sudo nano /etc/sysctl.conf
```

And add net.ipv4.ip_forward=1 to the bottom of the file.

Save this, and then finish by running:

```
$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

Run the following three commands to make sure the internet is forwarded correctly:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

09 Finish up

ASo that this works after a reboot, type:

```
$ sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Then add up iptables-restore < /etc/iptables.ipv4.nat to the end of the **/etc/network/interfaces** file.

Finally, set it up as a daemon with:

```
sudo service hostapd start
sudo service isc-dhcp-server start
sudo update-rc.d hostapd enable
sudo update-rc.d isc-dhcp-server enable
```

"The Raspberry Pi's portability makes it ideal for carrying around as an emergency wireless router"

Build a Twitter-powered lamp with Python

Using the GPIO port, let's see if we can set up an LED light to flash in response to the Twitter API…

Resources

tweepy
Internet connection
Breadboard
Male-to-female cables
300ohm resistor
LED

Over the last eight years Twitter has become one of the most prominent social media networks in the world. Twitter is built on excellent open source technology and code, meaning it's very easy to work with alongside your Raspberry Pi. With all the focus on the tiny 140-character limit, you'd probably be surprised to hear that behind each tweet is over 3,000 characters of raw data!

While we're not going to harness the full power of Twitter's API in this tutorial, we are going to do enough to allow us to make a Twitter-powered lamp – a light that flashes whenever our chosen phrase or word is mentioned on Twitter. You'll need to set up a Twitter account and to install tweepy.

This project is actually a lot easier than you might think, so let's get started…

01 Set up an app

Before you can do anything else, you need to make sure that you've got a Twitter account and that you're signed in. In order to be able to use the API, you need to create a Twitter app, so head over to **https://apps.twitter.com** and click 'Create New App'. Fill out the form, but don't worry about the URL sections – just put any website for now, since we won't be using this functionality.

02 Configure the app

Once the app has been created you can alter the access to Twitter by modifying app permissions. While we don't need the ability to 'write' to Twitter for this project, we will do in the next one, so click 'Modify App Permissions' and change it to 'Read and write'. Once this changed, you can click the 'Create my access token' button at the bottom of the page. When this has refreshed, you'll have access to the info you need in the 'API Keys' tab. Copy the API key along with its secret and the access token and its secret into a new document called 'twitter_lamp.py' in the format shown in our code listing on the right.

03 Install tweepy

The next step in creating our Twitter-powered lamp is installing and setting up tweepy. In the terminal, type:

```
001  git clone https://tweepy/tweepy.git
```

Once it's downloaded, move into the directory (cd tweepy) and install it like so:

```
001  sudo python setup.py install
```

Once tweepy is installed we don't need the contents of the folder anymore. Go back to your home folder (cd ~) and delete it with:

```
001  rm -rf tweepy
```

You can test that the library is installed correctly by typing python in the terminal to open the Python Interpreter. Now type import tweepy – if you don't get an error when you hit Enter, you're all set!

04 Connect the LED

We're going to use exactly the same simple breadboard circuit used in our candlelight project at bit.ly/1mCxQqO – we're even using the same GPIO pins. With the circuit connected, we can turn our attention to the code listing on the right. The GPIO aspect of the code is very similar to the candlelight project, other than the fact we're using the GPIO.HIGH and GPIO.LOW commands to flash the light, as opposed to PWM. After our required imports at the top of the code listing, we've laid out the api_key and access_token phrases that point to our Twitter app attached to our Twitter account. Obviously we've masked over ours, since they're meant to be a secret. Don't share yours (even on GitHub)!

05 Using tweepy

After setting the keys and secrets as variables, we need to get tweepy to use them to authorise us. Create the auth variable and use the OAuthHandler to call them and then set the access token details as shown. We then use auth to allow us access to the Twitter API on our account and test the connection by printing our Twitter handle using the api.me() method. We want to use the real-time Twitter stream for our project, though, so we need to override the StreamListener class in tweepy to make our lamp work. The on_status method in that class is where the magic happens; whenever there's a status update that's relevant to us, it pings on_status and we trigger our GPIO pins to make the light flash on.

06 Test and tweak

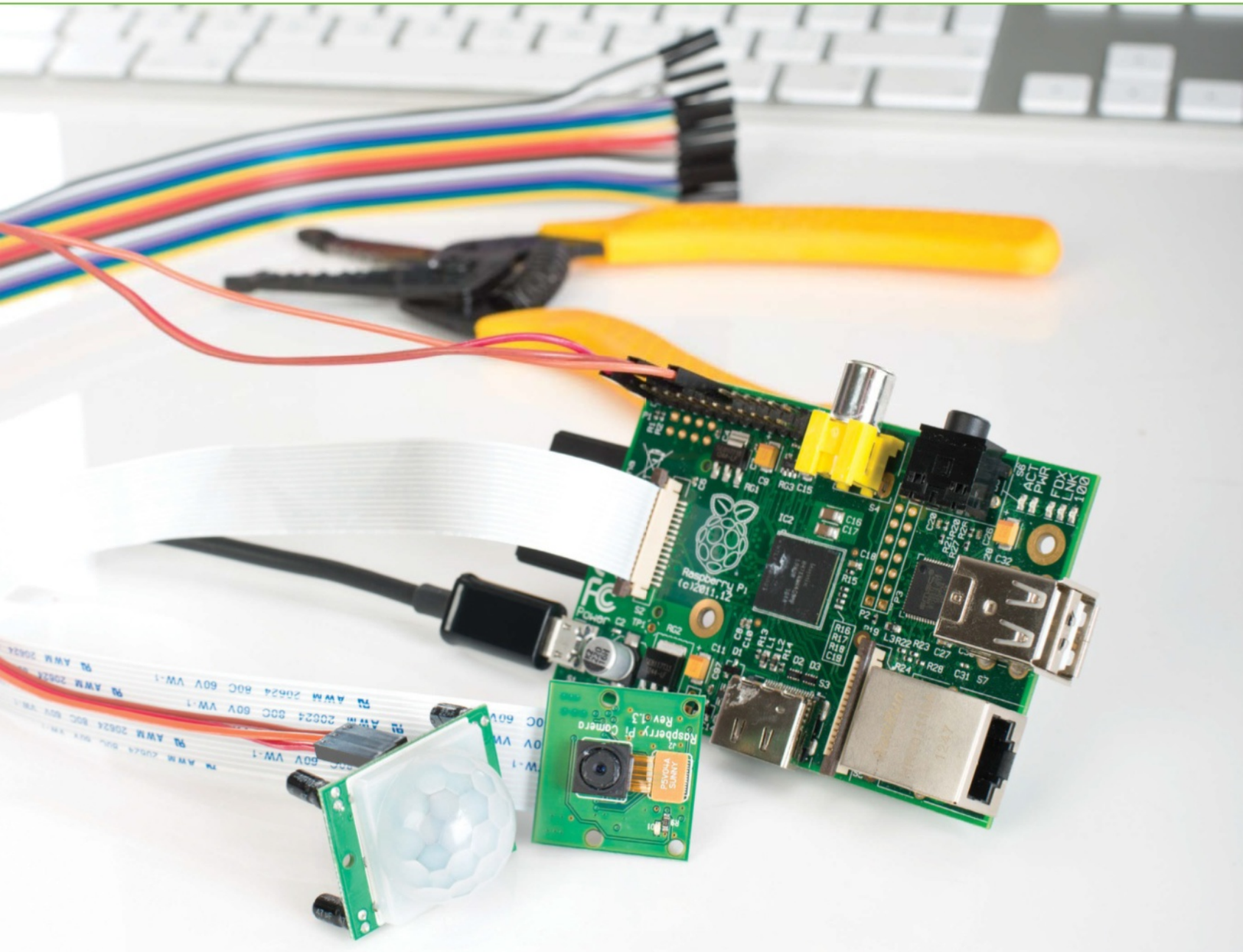
Next we put our search terms into a list called terms. As you can see, we've elected for our lamp to flash whenever one of the Raspberry Pi terms we've set is mentioned. You don't need to include '@' or '#' tags – the API takes care of that for us. Now all that's left to do is create an instance of the StreamListener() class, start the Twitter live stream and then use the filter method to set it to only track the search terms we're interested in. It's amazing what you can achieve in just 50 lines of Python! To test your new Twitter-powered lamp prototype, at the terminal type:

```
001  sudo python twitter_lamp.py
```

Full Code Listing:

```
#!/usr/bin/env python

#import the required libraries
import tweepy
import time
import RPi.GPIO as GPIO

# Set your access keys as configured
# at https://apps.twitter.com
api_key = 'your_api_key'
api_secret = 'your_api_secret'
access_token = 'your_access_token'
token_secret = 'your_token_secret'

# Initiate the OAuth process
auth = tweepy.OAuthHandler(api_key, api_
secret)
auth.set_access_token(access_token,
token_secret)

# Assuming the keys are good, you'll be
# able to test your Twitter app
api = tweepy.API(auth)
my = api.me()

print my.name, "is connected! Press CTRL
+ C to quit."

# Configure the GPIO port as we did in
the last project
led = 18

GPIO.setmode(GPIO.BCM)
GPIO.setup(led, GPIO.OUT)

# Set the led light to be 'on'
GPIO.output(led, GPIO.HIGH)

# We've tweaked the class that monitors
Twitter's stream
class StreamListener(tweepy.
StreamListener):

# Whenever a status occurs that we're
interested
# in we flash the LED
  def on_status(self, data):
    print "Flash the light"
    # We're using a simple for loop that
turns the light
    # off and on three times using an
interval of a
    # quarter of a second
    for i in xrange(3):
      GPIO.output(led, GPIO.LOW)
      time.sleep(0.25)
      GPIO.output(led, GPIO.HIGH)
      time.sleep(0.25)

  def on_error(self, error_code):
    print "Error:", error_code
    return False

# These are the words we're looking out
for on Twitter
terms = ['raspberry pi', 'raspberrypi',
'raspi']

# Configure the stream and filter only
our chosen terms
try:
  listener = StreamListener()
  stream = tweepy.Stream(auth, listener)
  stream.filter(track = terms)

except KeyboardInterrupt:
  print "\nQuitting"

# Don't forget to clean up after so we
# can use the GPIO next time
finally:
   GPIO.cleanup()
```

"It's amazing what you can achieve in just 50 lines of Python"



Create a tweeting bird watcher

Create an Internet of Things device able to takes pictures of wildlife before tweeting them to the web

Resources

tweepy

Internet connection

HC-SR501 PIR Infrared sensor

Camera board & picamera library

This is an exciting project that uses all of the skills and technology we've covered so far and raises the bar to include things like automated Twitter updates and event detection.

We're going to create our own little Internet of Things device that incorporates a simple PIR sensor (just £2.99/$5 from **www.modmypi.com**), the Raspberry Pi Camera board and the power of the internet to let us automatically capture images of birds (and other wildlife), as if by magic, before tweeting them to a Twitter account of our choice whenever activity occurs.

Let's get started with this fun project…

"Automatically capture images of birds"

01 Configure the infrared sensor

Since this an automated device, we need a way to trigger the camera when movement is sensed in our camera's target area. One affordable and easy-to-configure solution is the HC-SR501 Infrared Motion Sensor. The device itself has three pins – VCC (5V), Ground and a signal pin that sits in the middle. We've configured our script for the pir pin to trigger GPIO pin 17 (it sits opposite the PWM pin). The VCC pin is connected directly to the 5V power pin and the Ground to the same Ground pin we've used throughout the tutorials.

02 Test the PIR

We don't want to send millions of accidental tweets to Twitter, so we should do some pretty extensive testing with the PIR first, using simple print statements to show when motion has been detected. Visit github.com/russb78/tweety-pi and explore the project to find the pir_testing.py script. Copy it onto your Pi and run it with your PIR connected. You'll probably find that it's over-sensitive for your needs by default. You'll find two tiny adjustable screws on the PIR. Gently adjust them to the left to lower the sensitivity and test thoroughly until you get the desired result.

03 Set up the project

With the camera connected as per our previous camera project earlier in the week, we need to tie the camera, motion sensor and Twitter code together to make a tangible project that can be left to do its job. We'll walk through the script, but it's worth looking around our project by cloning it from GitHub. In the terminal type:

```
001  git clone https://github.com/russb78/
     tweety-pi.git
```

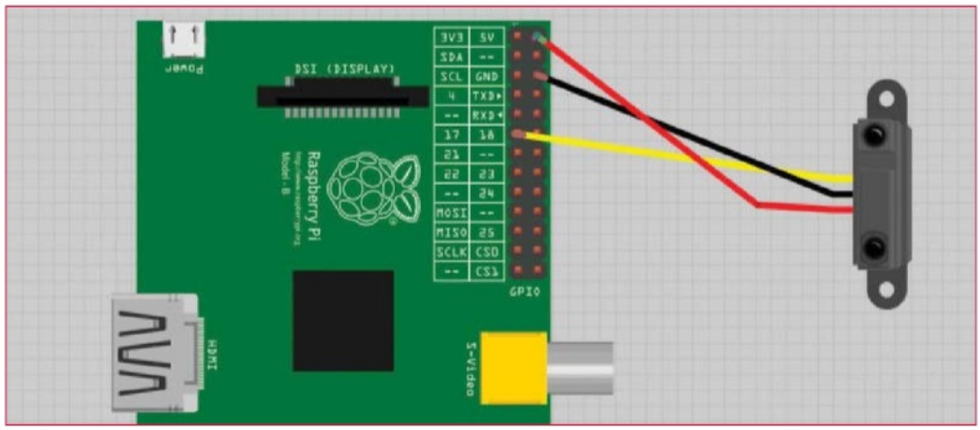
Enter the project with cd tweety-pi to take a look around. It's been set up as a full (if a little basic) project with a readme, licence and even a folder for our pictures to sit in.

04 The callback function

One of the big changes in this project compared to our previous ones is that we're using the GPIO as an input, instead of an output. Since we want the PIR sensor to alert us to movement, we really want the PIR to interrupt our script to let us know – that's where our callback function motion_sense() comes in. Looking further down the script to the main program loop you'll see a GPIO.add_event_detect. Whenever the assigned GPIO pin gets pinged, the script will stop what it is doing and jump to the named callback function (in this case motion_sense). This simple function then calls the take_picture function below it.

05 Simple chain

The entire chain of main functions that make up the meat of the project are laid out in trigger order and all initiated from that initial callback function. Once motion is detected the take_picture function is called. As soon as the image has been saved to the /pics folder we call the update_twitter function. Here, we're loading our previously saved image and using the Twitter API's update_with_media method to allow us to tweet our picture to the outside world. We can set our status from within this line, but instead of repeating the same phrase we use the random module's choice method to pick from a list of three we'd assigned to the variable tweet_text earlier in the script.

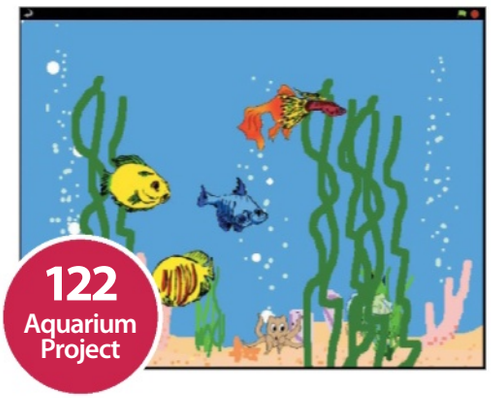


Full Code Listing:

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import random, time, os
import tweepy
import picamera

pir = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(pir, GPIO.IN)

### TWITTER SETTINGS ###
# Set your access keys via https://apps.
twitter.com
api_key = 'your_api_key_number'
api_secret = 'your_api_secret_number'
access_token = 'your_access_token_number'
token_secret = 'your_token_secret_number'
auth = tweepy.OAuthHandler(api_key, api_
secret)
auth.set_access_token(access_token,
token_secret)
api = tweepy.API(auth)
my_twitter = api.me()
print my_twitter.name, "is connected!
Press CTRL + C to quit."
# Three  statuses. We'll pick one at
random to go with our pic
tweet_text = ['Another shot taken with
tweety-pi!',
              'Just spotted with my
Raspberry Pi',
              'Snapped automagically with
my Raspberry Pi camera!']

### CAMERA SETTINGS ###
camera = PiCamera()
cam_res = (1024, 768)
camera.led = False # Turn off LED so we
don't scare the birds!
pics_taken = 0
time.sleep(1)

### MAIN FUNCTIONS ###
def motion_sense(pir):
  print "Motion detected... Taking
picture!"
  take_picture(cam_res)

def take_picture(resolution):
  global pics_taken
    camera.resolution = resolution
    # Capture a sequence of frames
    camera.capture(os.path.join(
      'pics', 'image_' + str(pics_taken)
+ '.jpg'))
    pics_taken += 1
    print "Picture taken! Tweeting
it..."""
    update_twitter()

def update_twitter():
  api.update_with_media(os.path.join(
    'pics', 'image_' + str(pics_taken -1)
+ '.jpg'),
    status = random.choice(tweet_text))
  print "Status updated!"
  #We don't want to tweet more than once
per minute!
  time.sleep(60)

### MAIN PROGRAM LOOP ###
try:
  GPIO.add_event_detect(pir, GPIO.RISING,
callback=motion_sense)
  while True:
    time.sleep(60)
except KeyboardInterrupt:
  print "\nQuitting"
finally:
  camera.close()
  GPIO.cleanup()
```

Programming

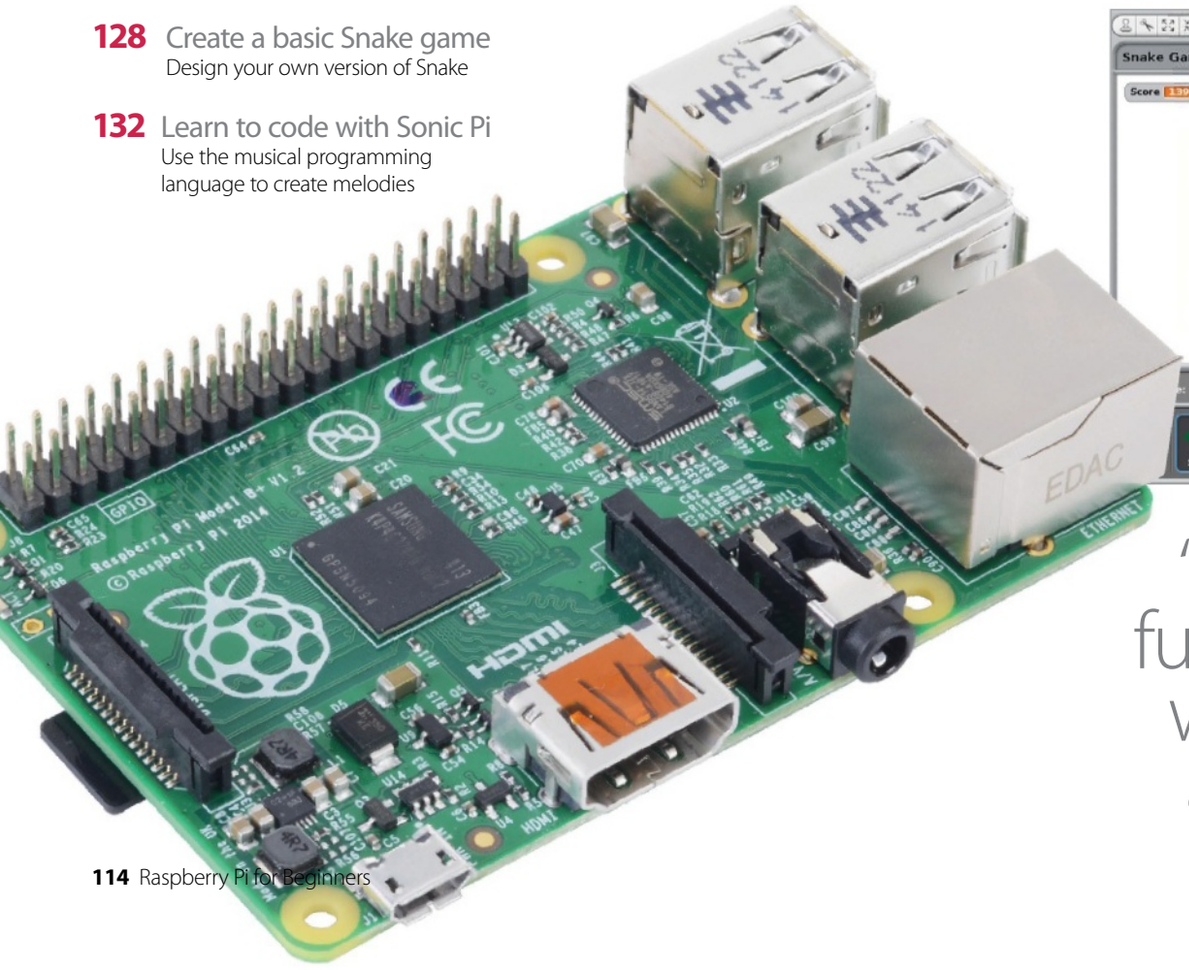
Learn the basics of Pi programming using Scratch and Python



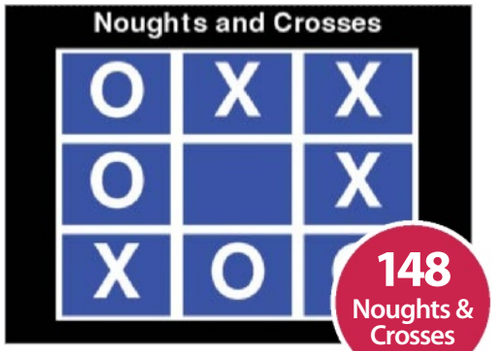
122 Aquarium Project

124 Drawing application

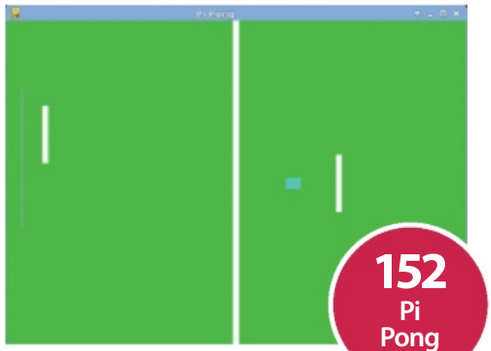
128 Snake Game



"Explore the fun of coding with Python and Scratch"



148 Noughts & Crosses



152 Pi Pong

116 Scratch & Python

142 Rock, Paper, Scissors

"Bring your creative ideas to life using simple lines of code"

134 Python Masterclass

Use Python and Scratch

Explore the fun of coding with Python and Scratch

Python and Scratch are pre-installed on the Raspbian distribution for the Raspberry Pi. Both languages are well suited to the novice coder. However, each takes quite a different approach to code construction.

If you have never done any programming before, the easiest of the two to get to grips with is Scratch. While primarily aimed at school children, it enables anyone to create fun interactive stories, games and animations using its simple toolset.

Python is a little harder to learn, but is a powerful object-oriented programming language that was designed to be highly extensible.

Read on for more information on Scratch and Python, along with some other Pi languages.

Python

Why Python?

Python has a clear and easy-to-grasp syntax. It's a very concise language. Useful tasks are executed in just a few statements, and complete apps can be created with relatively few lines of code. With Python's interactive mode, you enter statements and see the results immediately. It's a great way to experiment and gain confidence with the language. IDLE, a simple Python editor, is already included. Other Python-friendly editors, such as Geany, are easy to install.

Large module collection

Python's built-in functionality is supplemented by hundreds of specialised modules. Developers use these modules to create tools, games, websites, smartphone apps, hardware controllers and much more.

Consequently, Python enjoys huge community support. Google developers, astronomers, robotics engineers, space scientists, nuclear physicists and bioinformatics researchers all use Python.

Valuable skills

As the language is free to use and distribute, Python is popular with software companies, research laboratories and academic institutions across the world.

And the programming skills you'll acquire can be applied to other languages, such as PHP, Java and C.

Python links

Python home: **python.org**
Python docs: **python.org/doc/**
Python wiki: **wiki.python.org**

Scratch

What is Scratch?

Scratch, from MIT's innovative Media Lab, has a visual interface and is aimed at anyone old enough to use a keyboard and mouse. In fact, you hardly need to use the keyboard at all.

Nevertheless, it's possible to create intricate and sophisticated programs, including colourful animations and entertaining games. Do have a play around with the Scratch pre-installed on Raspbian – you'll have fun and be surprised at some of the results you can quickly achieve.

Block scripting

Scratch does away with the traditional editor and symbolic language approach. In its place is a collection of graphical code blocks.

These blocks have different shapes and snap together rather like a code jigsaw puzzle. Scratch coding is fast and fun.

Scratch Studio

Scratch has its own integrated development tool, the Scratch Studio. It includes everything you'll need to create complete applications.

There's an extensive collection of example projects, images and sounds to help get started. And the Scratch apps you create can be shared online or run on a Windows or Linux PC, or a Mac.

Scratch links

Scratch home: **scratch.mit.edu**
Scratch help: **scratch.mit.ed/help**
Scratch projects: **scratch.mit.edu/explore**

Other Pi programming languages

A quick guide to Raspberry Pi development languages

When coding on the Raspberry Pi, you're not limited to Python or Scratch. With a full Linux platform at your disposal, a wide range of other languages are supported

Shell scripts

A shell script can call any combination of Linux commands. So the potential is enormous. Get started with simple 'Bash' shell utilities to automate repetitive operations. Then try creating more advanced code to manage system resources and process data. Useful for writing shell scripts, the 'nano' and 'vi' text editors are included in most Linux distributions.

C

The C language offers the ultimate in portability and execution speed. Compilers exist for virtually every chipset and operating system. With a Linux-based Raspberry Pi, the C language is always available, since it's used to build downloaded source file packages. Although C typically takes a little while to learn, the compiled apps will be fast in operation and small in size. Consequently, C is ideal for fast-action games or DIY hardware projects.

Java

Java is a popular choice of software organisations. Its simplified C-style syntax is easy to read and write, plus it runs on the vast majority of platforms. Java developers can create almost any kind of app or tool. It's used for Android apps, desktop office and development tools, web servers plus various aviation and space mission systems. Java needs plenty of free memory, so runs best on the 512MB version of the Raspberry Pi Model B boards.

PHP

PHP is a scripting language that's easy to learn. Developers typically use PHP to create rich, data-driven websites, such as personal blogs, online image libraries, wiki pages and complete eCommerce sites. Code can be added to existing webpages, embedded in the HTML, or located in separate '.php' files stored on the web server. PHP is often combined with the Apache web server and MySQL database – which are also completely free to download and use.

BASIC

As the name suggests, BASIC is aimed at novice programmers. Its straightforward English-like words are easy to understand and remember. Tiny BASIC is one popular Pi-friendly option. And the RISC OS distribution can emulate the BBC Micro Model B computer, complete with the infamous BBC BASIC language. Unfortunately, BASIC doesn't benefit from the extensive range of community-supported modules and libraries enjoyed by Python, PHP, Java and C.

Links

Check out the following links that you might find rather useful for further information:

C:
cprogramming.com

Java:
java.net

PHP:
php.net

Apache:
apache.org

MySQL:
mysql.com

RISC OS:
riscosopen.org

Tiny BASIC:
raspberrypi.org/archives/tag/tinybasic

Code with Scratch studio

An interactive guide to coding with Scratch studio

Would you like to delve into the world of animation and game creation? Do you want to bring your creative ideas to life without learning a software development language? With Scratch you can do all this, and much more.

Scratch 1.4 is installed on the official 'wheezy' Raspbian operating system image. If your Raspberry Pi doesn't already have Scratch installed don't worry, just hop on over to the official MIT Scratch website (**http://scratch.mit.edu/scratch_1.4**) to find the download and install instructions.

To begin all we need to do is open the Scratch Studio. Click on Scratch's cat-icon on the desktop, or find the Scratch in the LXDE desktop menu.

The Scratch Studio is a complete development environment. It's divided up into a number of separate panels. Each panel has a specific role in the app construction process and its own specific set of features and tools.

In this guide we'll take a tour through each of these Studio panels. Our tour will be based around one of the example Scratch projects. By understanding how an example project is constructed we'll acquire the knowledge we need to create our own Scratch projects.

First we'll see this example app in action, then uncover the sprites and other elements that make up this app and finally we'll discover how Scratch's block-based scripting underlies all the activity.

Scratch studio projects

Located at the top of the Studio are three quick-access icons and the main menu (see Fig 1).

The first globe-style icon sets the language for the Studio. The other two buttons provide rapid access to the project save and share features.

Under the 'File' menu there's a typical set of file management features to open, save and import Scratch projects. There's also a 'Project Notes' option where we can enter feature descriptions and comments.

The 'Edit' menu contains a mixed bag of animation, image and audio editing tools. While the 'Help' section provides access to the browser-hosted help pages.

Interestingly the 'Share' menu allows us to share our projects with the world. Any Scratch project can be posted onto the Scratch community website via the "Share This Project Online…' option and the 'Upload To Scratch Server' form (see Fig 2).

Let's load the 'Aquarium' example project. Select the 'Open…' option in the 'File' menu to display the Open Project dialog. From the list of large buttons on the left, click on the one called 'Examples'. Next, on the right, select the 'Animation' folder with a double click. Then select the '6 Aquarium' item. The open dialog window contents should look like the one in Fig 3.

With the Aquarium project loaded our Scratch Studio should look similar to the main image here. We'll begin our Studio tour with the staging area.

Scratch studio stage

The stage is where all the action takes place and is located at the upper right of the Scratch Studio.

The stage is constructed from graphical elements called sprites. Here we have plants, bubbles, fish and other creatures.

At the top of the 'Staging Area' there's a green flag and a red circle. Click on the green flag to bring the aquarium to life. Now spend a little

■ Fig 1: Scratch Studio Menu – Scratch Studio's main menu and shortcut icons

■ Fig 2: Project Share Dialog – We can share our projects with the world using the Studio's Share menu

time studying the Aquarium animation. Note the creature's movements and rising bubbles. The red circle icon stops the action.

We can set the view mode with the three buttons located just above the green flag.

The two left hand buttons increase or decrease the size of the Staging Area panel. A smaller Staging Area means the central area of the Studio increases in relative size.

The right hand button is the Presentation Mode which displays the stage in full screen mode (see Fig 4). Exit presentation mode with the curly arrow button at the top left, or press the 'esc' key.

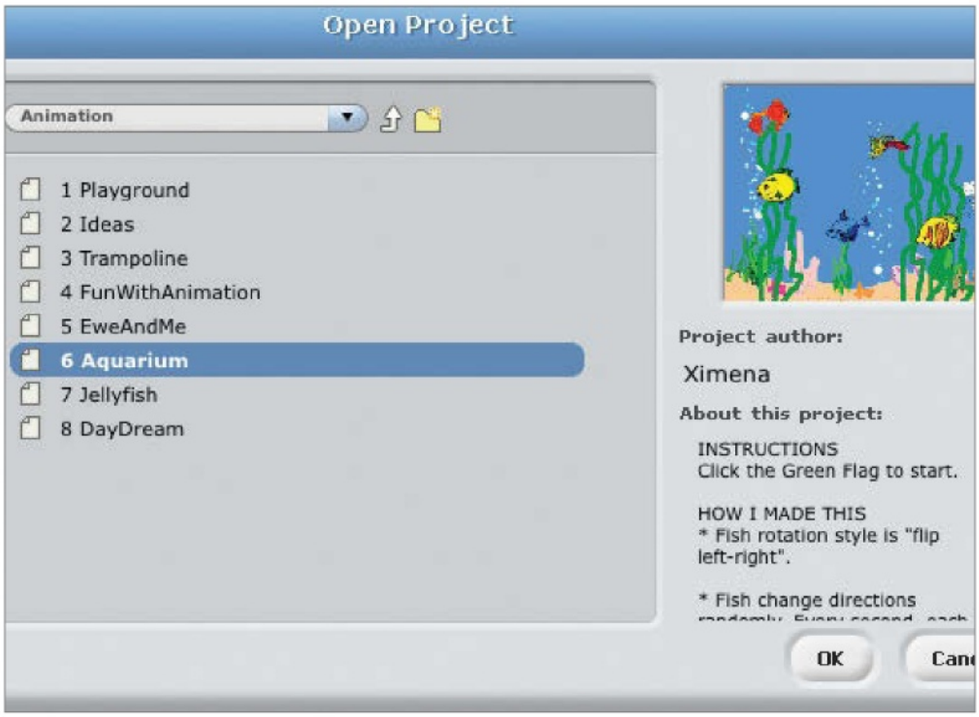
Scratch studio sprites

Beneath the Staging Area is the collection of sprites for this project.

The 'Stage' sprite is separated from the rest. It's a little different to the others and acts as the background image for the stage.

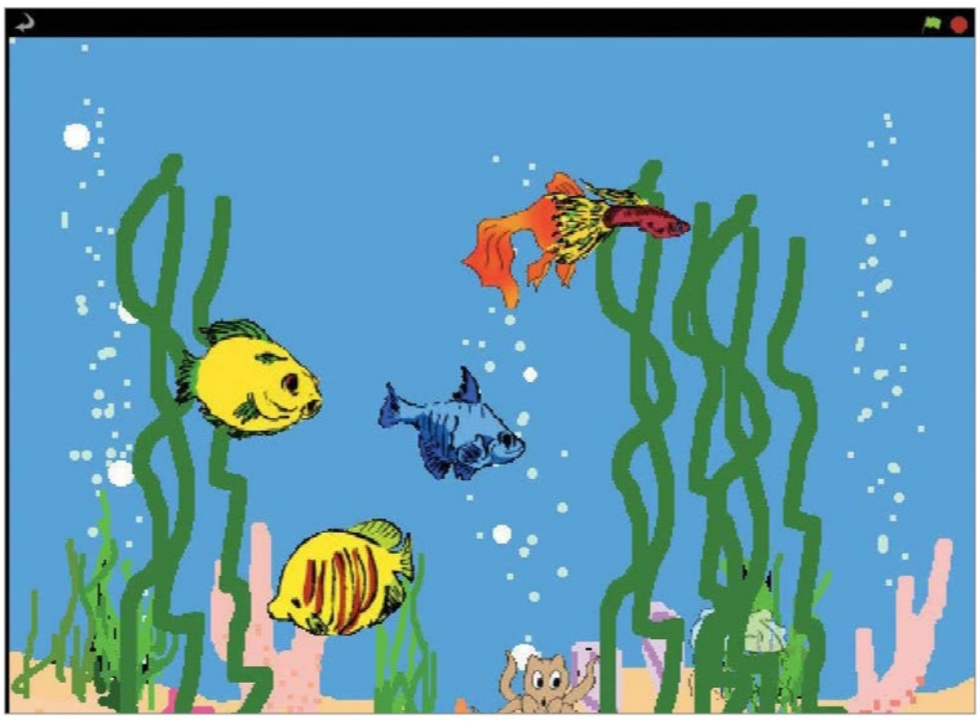
The three buttons across the top of this area offer various ways to create a new sprite. The first button opens up a blank canvas in the Paint Editor. The second button creates a new sprite based in an image file, as selected by the popup file section dialog window. The third will select a random image from the pre-installed image collection.

We can manage sprites directly from the stage using the four buttons to the right of the main menu. Here we click on a particular button and then a sprite on the stage.

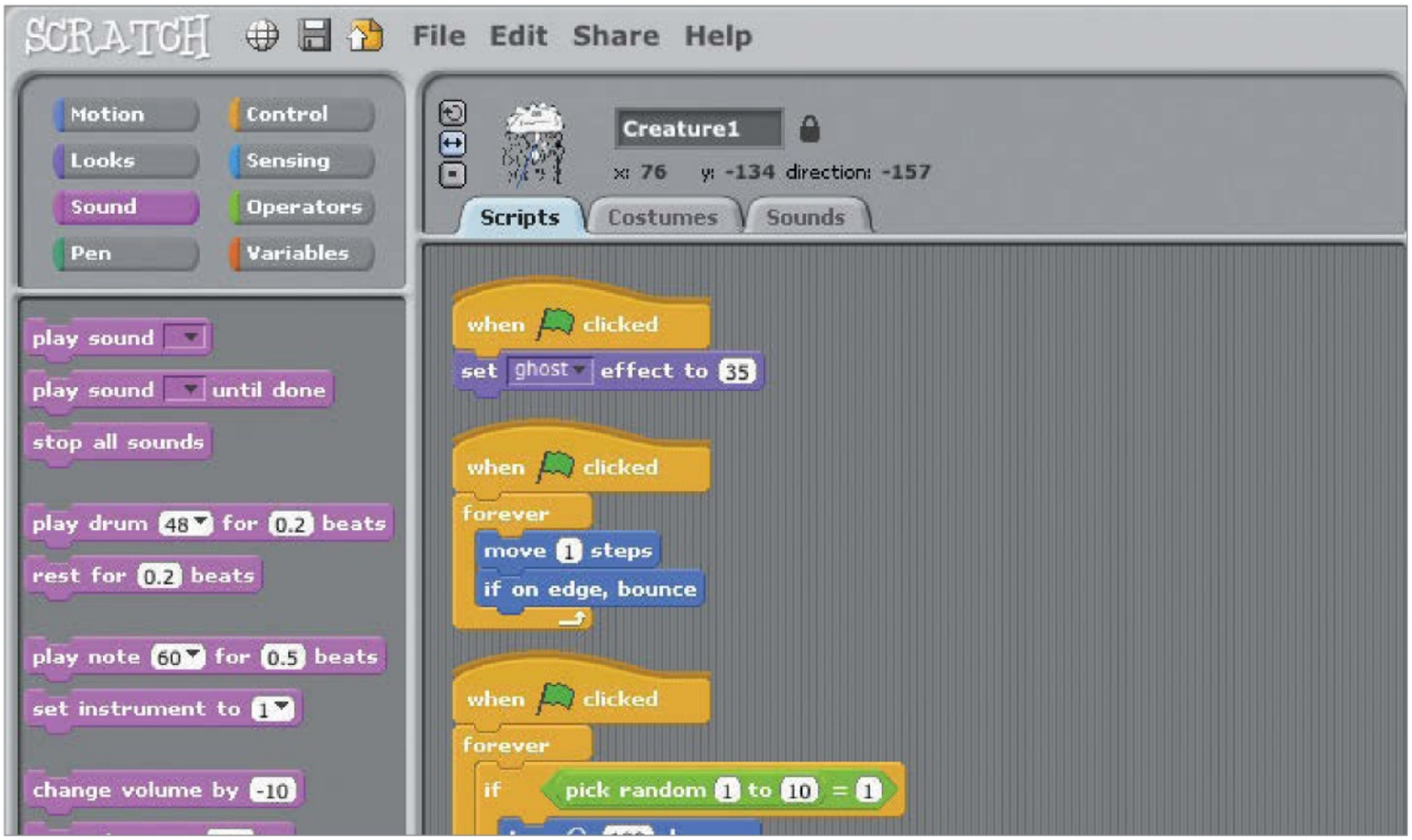


■ Fig 3: File Open Dialog – Use the Studio's File menu and 'Open…' option to load the Aquarium project

"Do you want to bring your creative ideas to life without learning a software development language?"



■ Fig 4: Stage Presentation Mode – The Studio's stage presentation mode is the best way to see the project in action



Use Scratch blocks and tools

Getting to grips with the Scratch Studio toolbox

Situated in the centre of the Studio is the Edit Panel. The panel contents relate to the currently selected sprite.

Let's start by selecting the jelly fish sprite, called Creature1, from the Sprite Collection area.

At the top we have the sprite's image and name, plus an indication of its current stage coordinates and direction. On the left are three animation control buttons. The top button will rotate the sprite, the second switches between left and right facing states, and the third turns animation off.

Below are three Edit Panel tabs. The script tab is where block scripts are created. Here's where we'll drag and drop our blocks, snapping them together in various combinations.

To change a sprite's visual appearance we'll use the 'Costumes' tab. Each sprite can have one or more costumes. For example, the jellyfish has two costumes (see Fig 1). Each costume has buttons to edit, copy and delete. New costumes can be painted, imported or captured using the three 'New Costumes' buttons.

The sound tab allows us to add audio to our project. A sprite can be associated with one or more sounds. We can have some fun recording sounds with the 'Record' button (see Fig 2) or simply load an existing sound file using the 'Import' button.

Scratch Block Styles

The 'Blocks Palette' contains the complete collection of scripting blocks. Blocks come in three basic styles, namely hats, stacks and reporters (see Fig 3).

A hat-style block will start block script execution based on a specific event. The classic hat block is the 'green flag' click event. However, there are numerous others, including hat blocks that start script execution after a specific key press, a mouse click and even following sensor event from the some GPIO connected hardware.

Reporter blocks allow us to specify textual, numeric and boolean values. They fit into specific shaped 'holes' in other blocks. A rectangular reporter will contain a text string. While the rounded end reporters are associated with numeric values, angle-ended reporters contain boolean true and false values.

Stack blocks are the core script building elements. They interconnect with other blocks via their top-edge notches and bottom-edge bumps. Many stack blocks contain 'holes' for reporter style blocks, which will modify their operation depending on the specified reporter block values.

The Scratch block collection is divided into groups. We select a block group using the eight buttons located at the top of the Block Palette panel, namely 'Motion', 'Control', 'Looks' and so on. These groups are colour coded. Apart from aiding block selection this colour coding provides a visual clue to a block's type when reading a block script in the Edit Panel.

Scratch Block Help

As we've seen, there are many blocks, each with their own specific functionality and capabilities.

In one way this is great news. A large block collection means Scratch can be used in a vast range of software projects, such as games, animation, music, graphics, math, science, robotics, electronics and much more.

However, the wide selection of blocks can be quite a challenge for the novice Scratch coder. To help with this problem the Scratch Studio designers

■ Fig 1: Jellyfish Sprite Costumes – The jellyfish sprite has two different costumes

■ Fig 2: Sound Recorder Tool – Scratch Studio includes a tool to record our own sounds

have included an informative set of block-centric help pages.

A simple right click on any block will display a pop-up help page option. The help page contains context-specific descriptions, graphical images and, where appropriate, a script example of how to use this particular block (see Fig 4).

It's a terrific feature which greatly simplifies the process of deciding which blocks to use. More importantly, studying these help pages is a highly effective way to enhance our scripting skills and discover the potential contained within Scratch's feature-rich block collection.

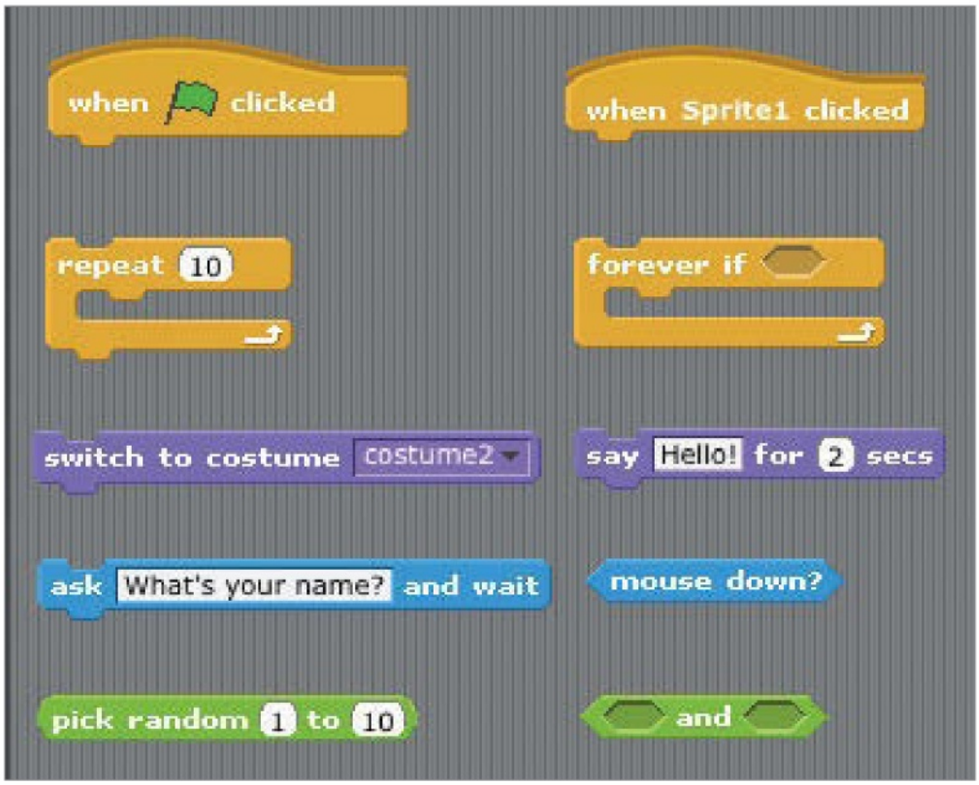
Block Script Walkthrough

Let's dig deeper into how a block script works in practice. For this, we will use a simple Aquarium project block script. From the 'Sprite Collection' panel select the Stage sprite. Then go back to the central 'Edit Panel' and select the 'Scripts' tab.

There's just a single block script. Starting at the top there's a 'green flag' hat-style block to kick off the activity. Next there's a 'forever loop'. The blocks inside this loop are actioned until the stop button is pressed. This forever loop block contains two other blocks.

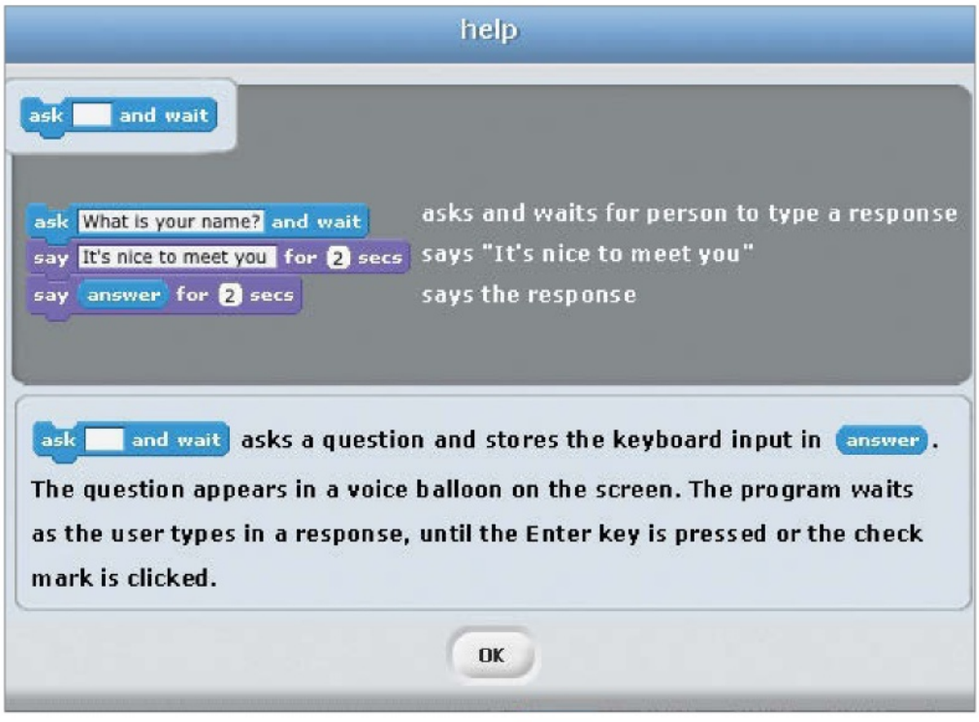
The first inner script block selects the next background image. Click on the 'Backgrounds' tab to view all the stage images. The second inner block simply pauses execution for a number of seconds. Setting the value to '1' means that this script will pause for a second before then performing the action specified by the next block.

When following this tutorial, it is important that you remember that these two blocks are enclosed in the forever loop block. So, the stage background images will be displayed in sequence for one second each.



■ Fig 3: Block Style Examples – Scratch blocks come in a number of different styles

"We can have some fun recording sounds with the 'record' button"



■ Fig 4: Block Help Window – Example of the help window associated with an 'ask and wait' block

Customise the Aquarium project

How to make changes to the Aquarium project

We've seen the Aquarium project in action and looked at some of the sprites, scripts and costumes. Now we're ready to make some changes. Let's start by creating a new Stage sprite background image. In the 'Edit Panel' select the 'Backgrounds' tab. Now click on the 'Copy' button on the third image. A fourth image will appear in the list. Click the associated 'Edit' button to open the Paint Editor.

There are many useful features in the Paint Editor's toolbox (see main image). In addition to freehand drawing and painting with the multi-sized brush, we can create predefined shapes, block fill areas with colour and add text. Selected regions can be shrunk, enlarged, rotated or flipped horizontally and vertically. Alternatively, we could start with an imported image file and then apply customisations.

Using this toolbox we can add or remove bubbles, move a starfish, or anything else we fancy. The zoom feature, located next to the colour palette, helps with fine details. Click the 'OK' button to save the changes.

Remember, the stage script block script will loop through every Stage sprite background. So, we can just click the green flag and watch what happens. To make further changes simply repeat the editing process.

Any sprite costume can be edited using this process. And, of course, we can use the same Paint Editor to design brand new sprites.

Aquarium Bubbles Audio

While the animation looks good it's very quiet. However, a sprite can be associated with one or more sounds.

Adding a sound effect is a two step process. The first step is to assign one or more sounds to a particular sprite. Step two involves the construction of a block script to play the sound.

We're sticking with the Stage sprite, so make sure it is selected in the Sprite Collection panel. Now, in the central Edit Panel click on the 'Sounds' tab. Here we have two buttons, namely 'Record' and 'Import'.

Click the 'Import' to display the 'Import Sound' dialog. We can select any suitable MP3, WAV, AIF or AU sound file on the Raspberry Pi. However, the Scratch installation includes a range of useful sounds in the 'Sound' folder.

With this 'Sound' folder selected we can see a list of sub-folders, including one called 'Effects' (see fig 1). Double click on 'effects' to see a list of sound files. Click on any sound file to hear the audio clip.

Notice there's a sound file called 'Bubbles'. This seems quite appropriate so select it and click the 'OK' button. Now the Sound tab will contain the 'Bubbles' sound (see Fig 2).

Any number of sounds can be added to a sprite. Each is triggered by a block script event, such as a mouse click, key press or as a sprite moves to certain position on the stage.

Aquarium Bubbles Script

Now for step two, building the block script.

Let's take a moment to consider what this script should do. We want our chosen sound to start on a green flag click event, and continue until the red stop button is clicked.

In the Block Palette panel select the 'Control' group. Drag and drop the hat-style 'when clicked' block with the green flag onto the 'Scripts' tab in the central 'Edit Area' panel. Place this block a little below the existing block script.

■ Fig 1: Adding sounds to the project is simple with the 'Import Sound' dialog

■ Fig 2: The Stage sprite's Sound tab after adding the 'Bubbles' sound effect

Next we need a loop. Drag the 'forever' loop block from 'Control' group and snap it onto the 'when clicked' block.

Now select the 'Sound' group. To play the whole sound sample we'll need the 'play sound until done' block. Drag it over to the 'Scripts' tab and drop it into the slot that we've made inside the 'forever' loop block.

Ensure the sound field value is set to 'Bubbles' and we're done (see Fig 3). Click on the green flag and listen to those bubbles.

Aquarium Project Experiments

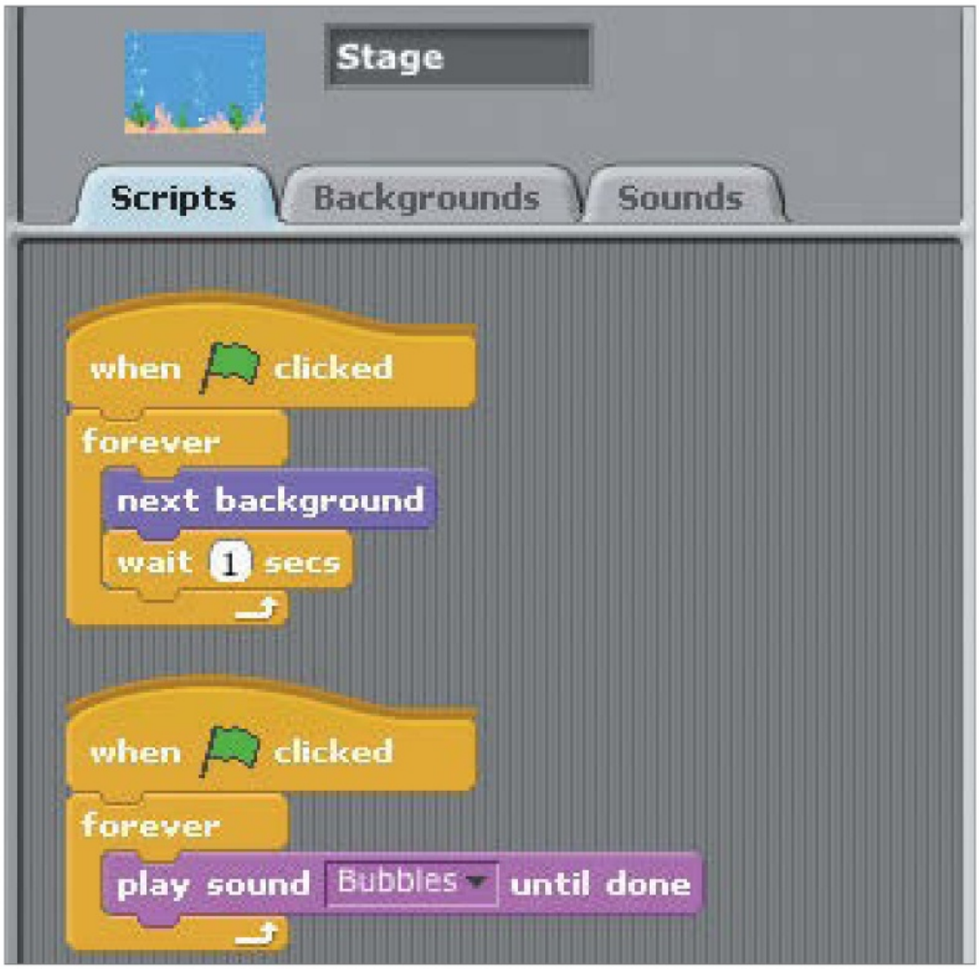
Before we start to experiment let's save our changes. Click on the File menu and select the 'Save As…' to display the 'Save Project' dialog (see Fig 4).

Choose a suitable name for the project, fill in the author field and add some comments. By default, projects are saved in a folder called 'My Projects'. However, we could save it in any folder.

Time to have some fun. Here's just a few ideas. Try adding more sounds in the 'Sounds' tab, then change the value in the 'play sound until block'. Alternatively have a go at recording your own sounds using the 'Record' button and the Sound Recorder tool.

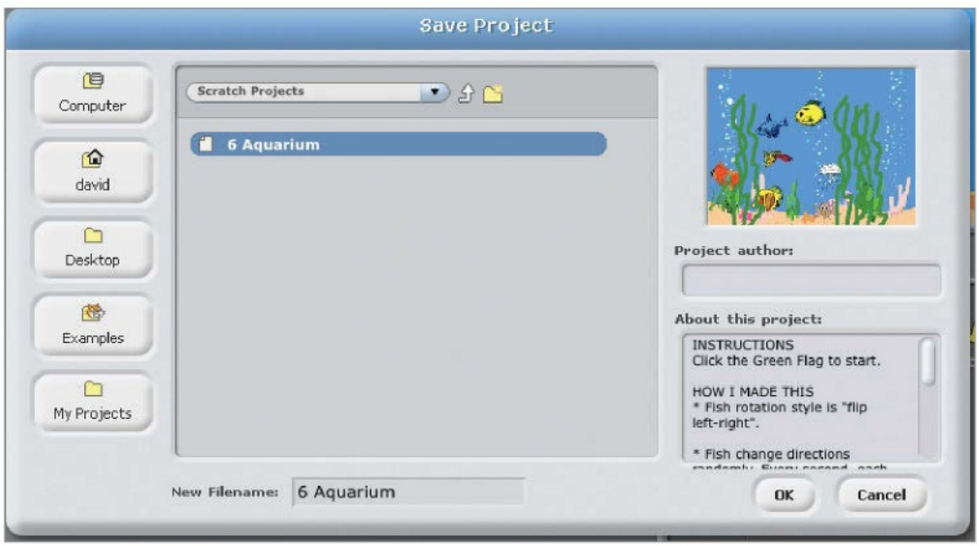
Next try to trigger sounds from different events, for example when a key is pressed. This will require a different hat-style block.

Comments can be a useful reminder of the changes we've made. In the 'Script' tab right click on the background and select 'add comment' from the popup menu. Then type in your comment.

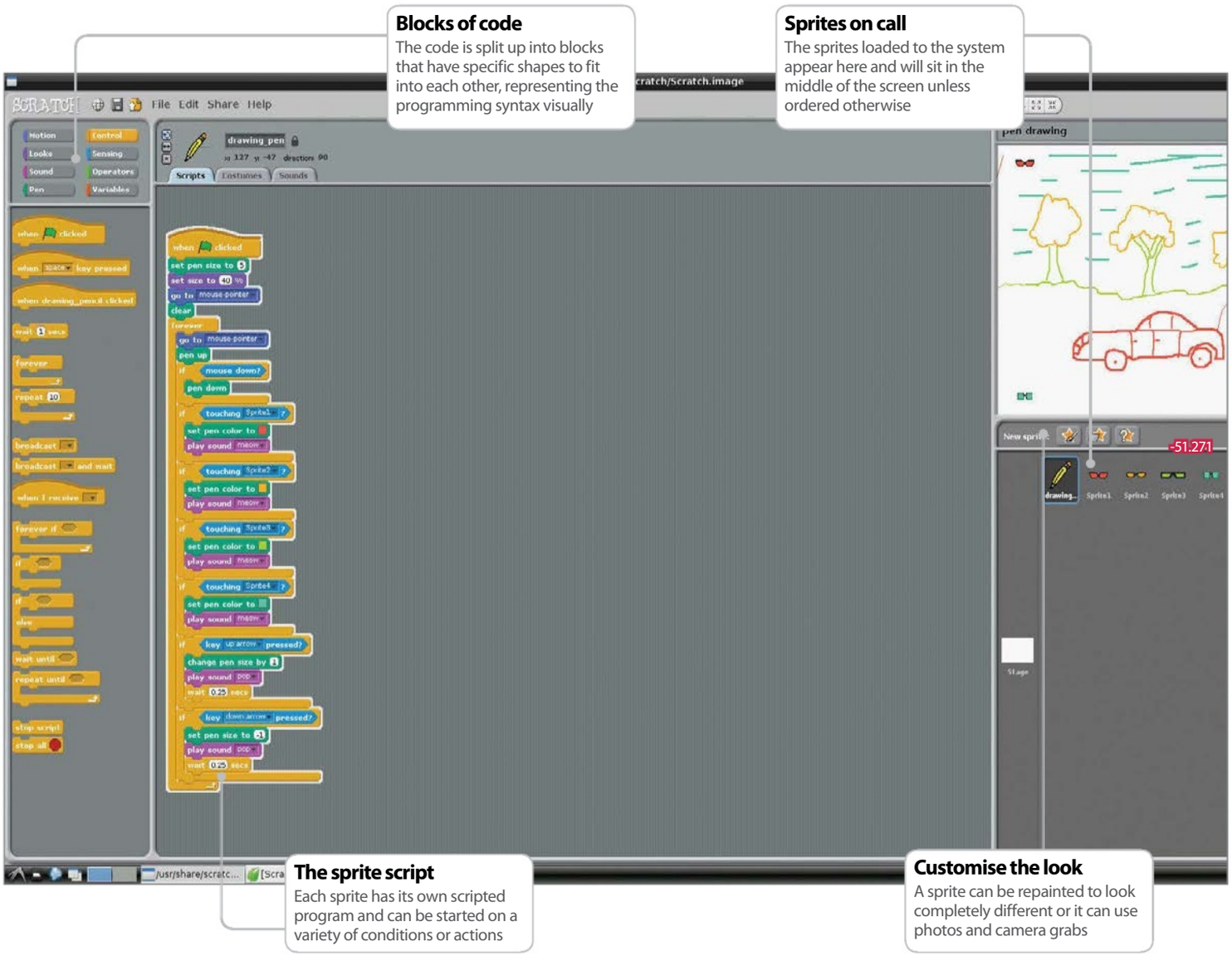


■ Fig 3: The Stage sprite's Script tab after adding the 'Bubbles' sound play block script

"When you're done, click on the green flag and listen to those bubbles"



■ Fig 4: Save all your project changes using the 'Save Project' dialog



Create a simple drawing application

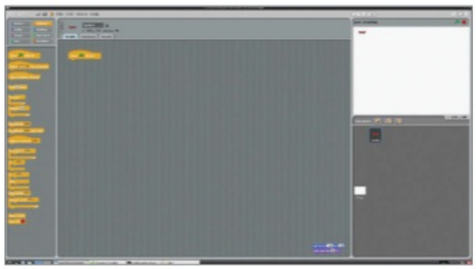
Here's how to add colour and pencil sizes using the built-in pen tool

Resources

No resources required to complete this guide

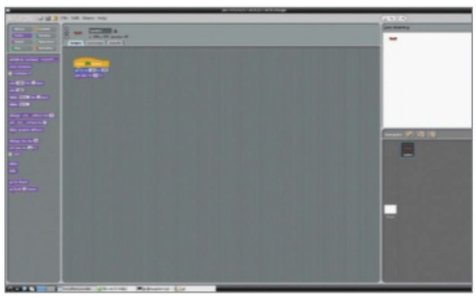
The aim of this tutorial is to use the Pen tool in Scratch to develop a more useful drawing application that you can then expand on yourself. It uses only the sprites that are bundled with the installation on the Pi but you could easily make some custom ones to generate the colour changes. In fact, one possibility is to have a colour palette and use this to select colours from. At the moment, this uses four sprites to make the colour selection from. All you have to do is move the pencil cursor over the colour sprite for it to start using that colour. Then simply click and draw. Any time the colour is changed a sound is played to let you know. Pressing up or down on the keyboard arrow keys increases and decreases the size of the pencil nib, not the sprite for it. A sound plays for this as well.

If you've never tried Scratch before there are a couple of things to look out for. The first is that it's fairly slow because of the graphical interface so you have to be fairly precise and persistent when moving blocks of code around. Scratch is object based so that all objects, or sprites, have their own scripts. Unlike other languages, there is no main code. The scripts for each sprite activate when you tell them to. For this reason, it's better to base most of the script around what the user will be doing with a sprite or the mouse cursor. Click on the green flag to run the scripts, the red circle to stop.



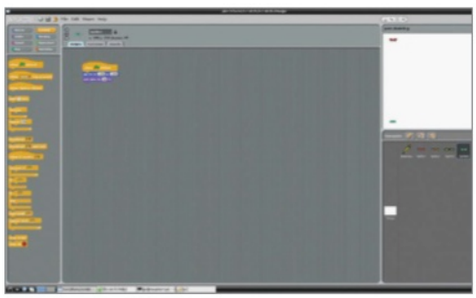
01 Load up sprites

Click in the top-right to use the Full Stage option. Then, in the sprite window, click on Load Sprite. Select the red sunglasses and load them. They will appear in the sprite window. Click on Control in the commands window. Drag the 'when clicked' header into the Script window.



02 Position in corner

Click on Motion and drag 'go to x:' into the Scripts window. Lock it to the bottom of 'when clicked'. Enter values of x=-200 and y=155. Click on Looks and drag a 'set size to' block and lock that on. Set the value to 40% by clicking and typing it in.

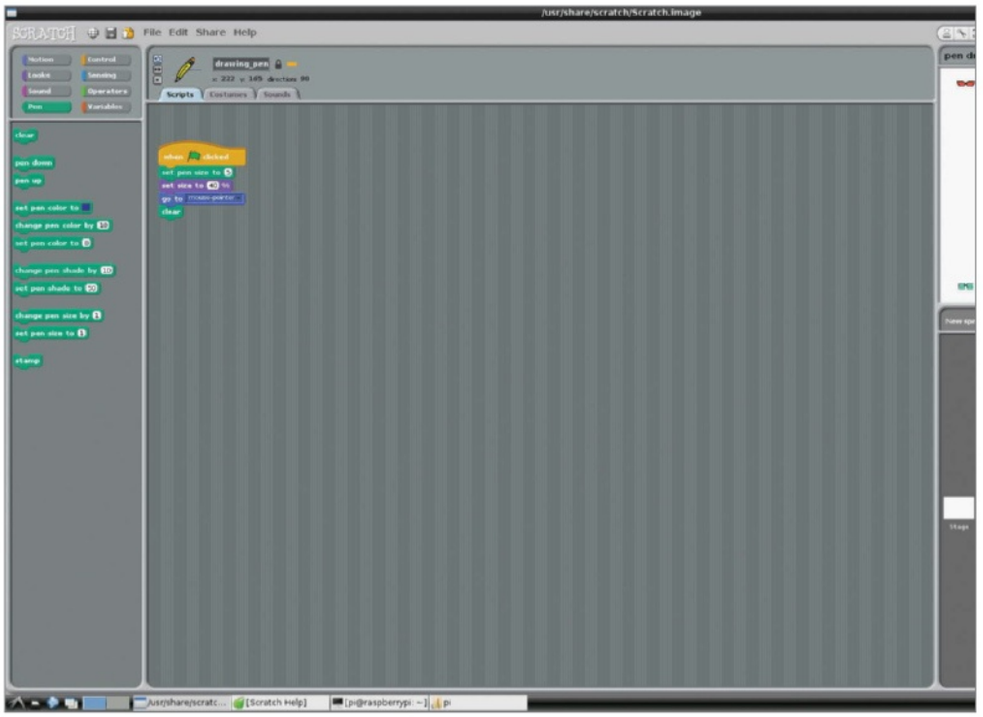


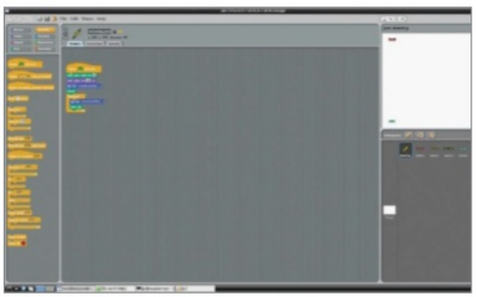
03 More sprites for colour

Repeat this process for the orange, green and blue sunglasses sprites. Create the code blocks for each, positioning them at x=200, y=155 as we did with the red. Go back to Load Sprites and load the Drawing pen. Delete the script that comes with this by right-clicking over each element and selecting Delete.

04 Set up the drawing

Under Control, drag a 'when clicked' header in. Get a 'set pen size' from under Pen. Click on the variable and enter 5. Go to Motion and get the 'go to' block. Click on the drop down arrow and select 'mouse-pointer'. Go to the Pen group and drag out 'clear'.





05 Controlling the pen draw

From the Control group drag out a 'forever' loop which the rest of the code goes inside. Go to Motion and drag out a 'go to'. Click on the down arrow and select 'mouse-pointer'. Go to Pen and grab a 'pen up'. Tuck this under the 'go to'.

06 Drawing and colour control tools

Go to Control and grab an 'if'. Grab a 'mouse down?' from the Sensing group and place it into the 'if' operator. Get a 'pen down' from the Pen group. Put this under and inside the 'if'. The next step is repeated four times, once for each colour and sprite.

07 How to control the colour changes

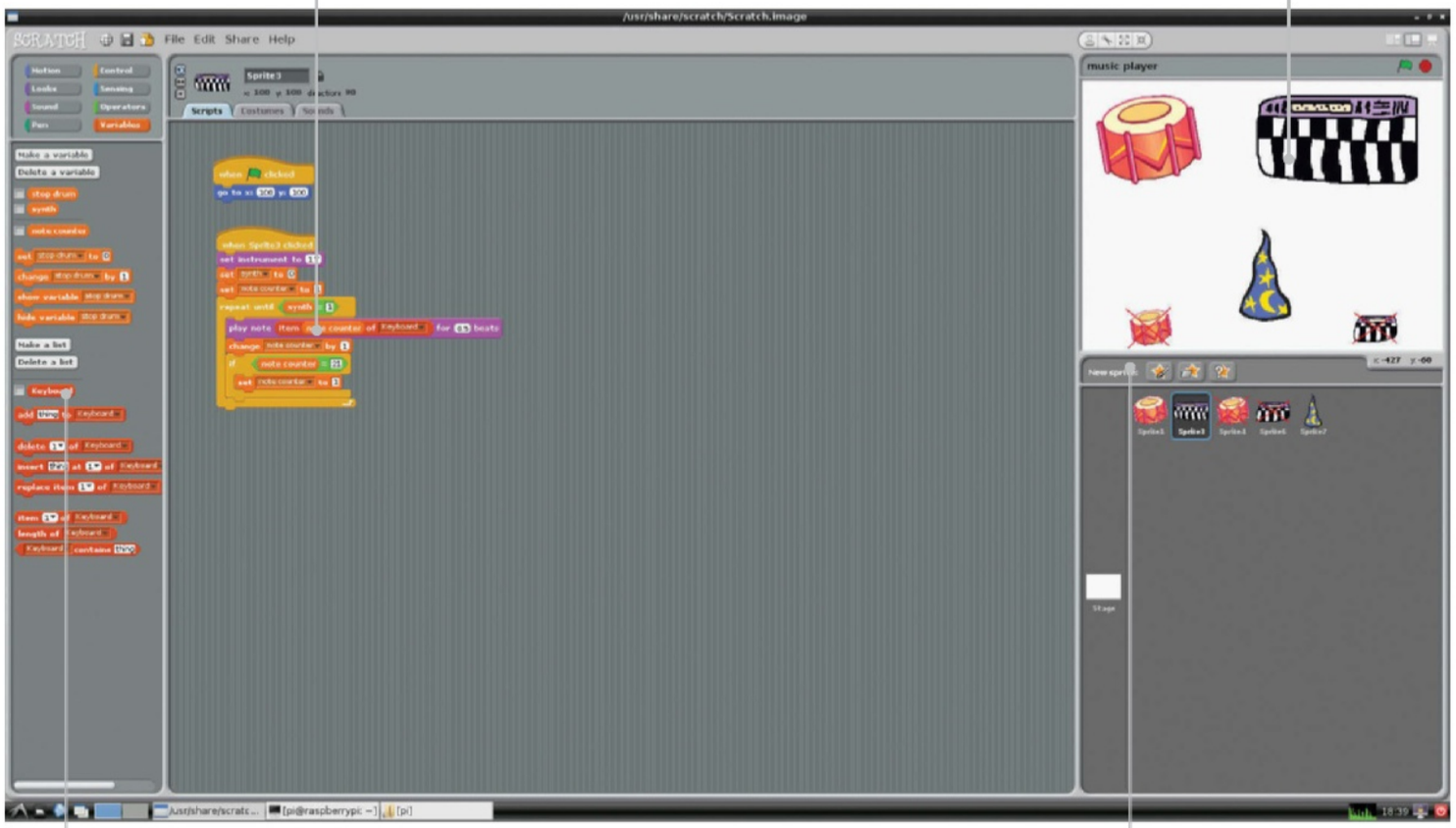
From Control grab an 'if' and from Sensing put a 'touching' condition inside. Set trigger to Sprite1. From Pen get 'set pen color to', click the colour square and use the dropper to sample the sprite. From Sound get 'play sound' and select 'meow'.

08 Setting the pen size

Add another 'if' and put 'key pressed' as the subject. Select 'up arrow' as the trigger. Get 'change pen size by' and enter 1. Use a 'play sound' with 'pop' then add a 'wait' for 0.25secs. Repeat this group but with the 'down arrow' and pen size of -1 instead.

Variables and lists
'Keyboard' is the list, while 'item' is the specific element that is being referred to by the increasing 'note counter' variable

The music instruments
Once running, click on the drum to start a drum loop and the synth to play a loop of keyboard notes



Populate the list
The list has to be visible to enter values into it directly. It can then be toggled back off to remove from the screen

New sprite please
Click on the Paint sprite icon to create an entirely new sprite. When finished they can be exported back to the library

Make music and play sounds

Discover how to use the musical instruments and sound effects built into Scratch

Resources

No resources required to complete this guide

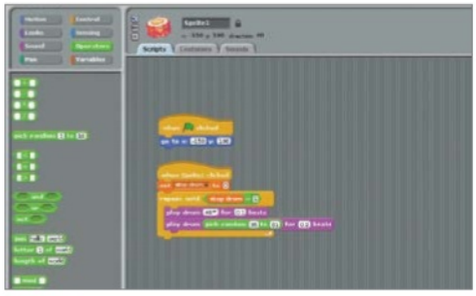
Getting your Pi to play music is relatively easy, if we're talking about MP3 files. Getting it to coherently play musical notes though, is a little trickier because the timing loops aren't precise when running Scratch. However, what this tutorial aims to do is use the drum and instrument commands, combined with a note sequence and some free hand effects to give you a basic music player. The project uses a 20-value array, or list as they are called in Scratch, to hold the notes for the instrument, while the drum sounds are played in a loop. The instruments can be turned on and off. To add to the cacophony, there's an option for playing a randomly selected sound over the top. The drum and synth sprites have smaller versions with a cross through them. This sets a toggle to one that the main drum and synth sprites use to detect when to stop playing. The synth is the one that uses an array. The array has 20 elements, each with a note value in.

This shows that the basic control commands in Scratch are very flexible. Where it looks like there is only room for one modifier for a command, you can insert variables and arrays as well to make it more complex. It's worth annotating your script to make it more readable.

Further things to remix with this project would be to add volume control, record some voice samples and have triggers for them as well.

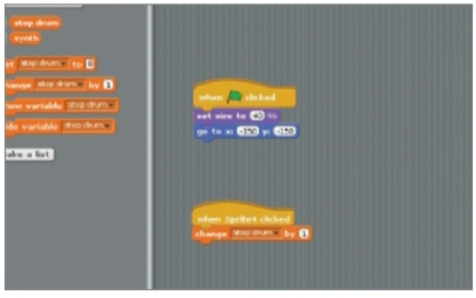
01 Position and set up

Load the drum sprite then drag out to 'When clicked' and 'Go to x:-150, y:100' to position in the top left. Drag out a 'When Sprite1 clicked'. Click on Variables and create 'stop drum'. Drag out 'Set () to ()' and enter 'stop drum' as the variable and a value of 0.



02 Repeat the drum pattern

Drag out a 'Repeat until ()' Control and then go to Operators. Drop in '()=()'. In the first half of the operator, drop in the variable 'stop drum' and set to =1. Then put in two 'Play drum' commands from Sounds. Set one of the drums to be selected at random.

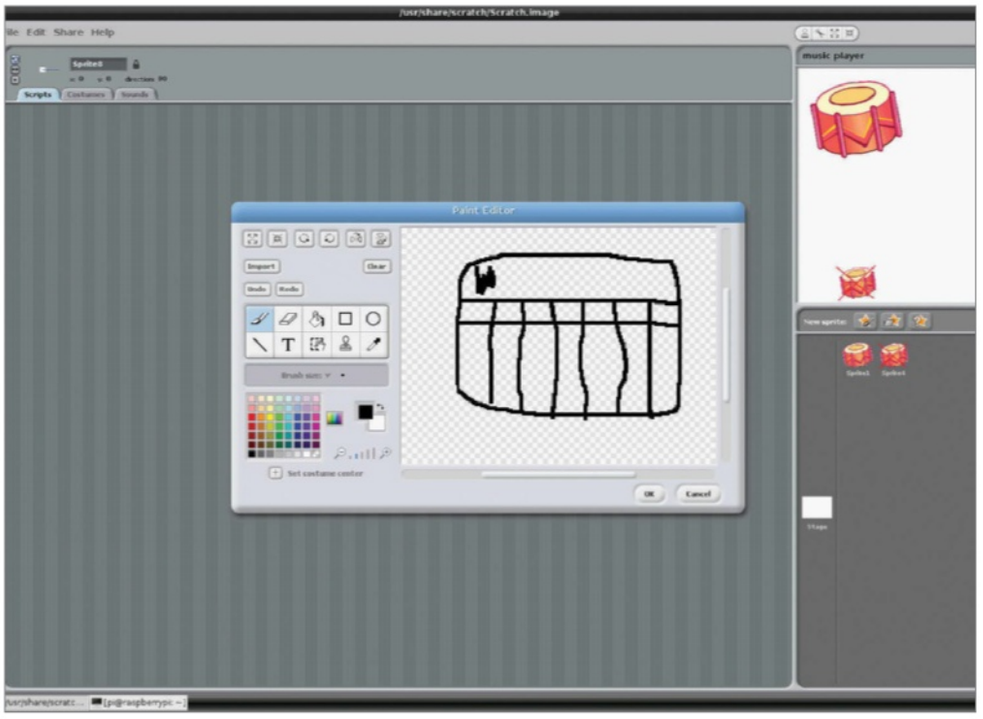


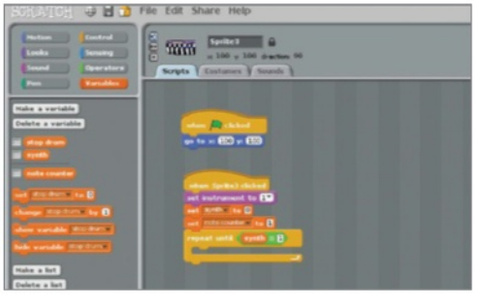
03 Stop the drum sound

To create, click on Paint new sprite. Then click on import and select the same drum. Draw a cross and close. Drag in the 'When clicked' header to size and position. Then drag out a 'When SpriteX clicked' and 'Change () by ()'. Enter the 'stop drum' variable and 1.

04 Create a synth sprite

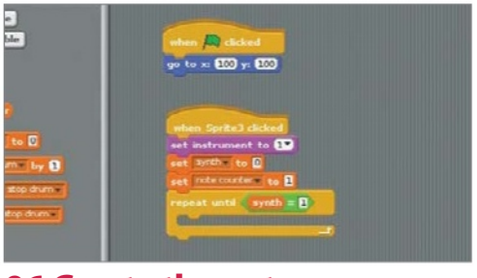
For this one, the sprite is entirely new. Click on Paint new sprite and draw a synth. Click on OK when done. Grab a 'When clicked' header and set the position underneath it to 'go to x:100, y:100' to place in the top left corner. Drag out a 'When SpriteX clicked' header.





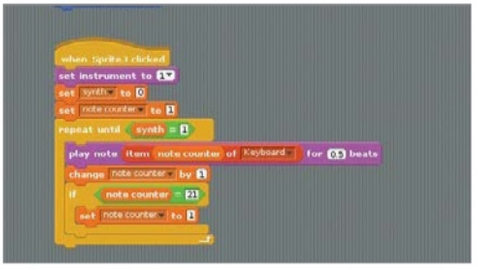
05 Define instruments and variables

Add a 'Set instrument to ()' and pick one you like. Go to variables and create ones called 'synth' and 'note counter'. From Variables, use 'Set synth to 0' and 'Set note counter to 1'. Drag out a 'Repeat until ()'. From Operators add '()=()' and enter 'synth' and 1.



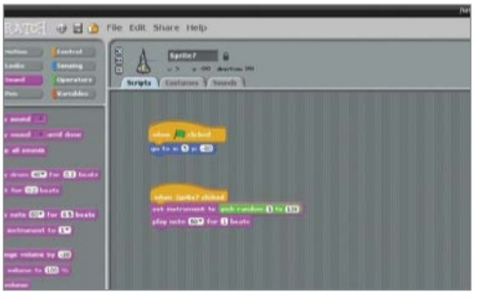
06 Create the note array

Go to Variables and create a list called 'keyboard'. In the Stage window the list appears. Click on the Plus to add a field and enter the values. Repeat so that there are 20 entries. Untick the Keyboard list in Variables to remove from the screen.



07 Play notes from array

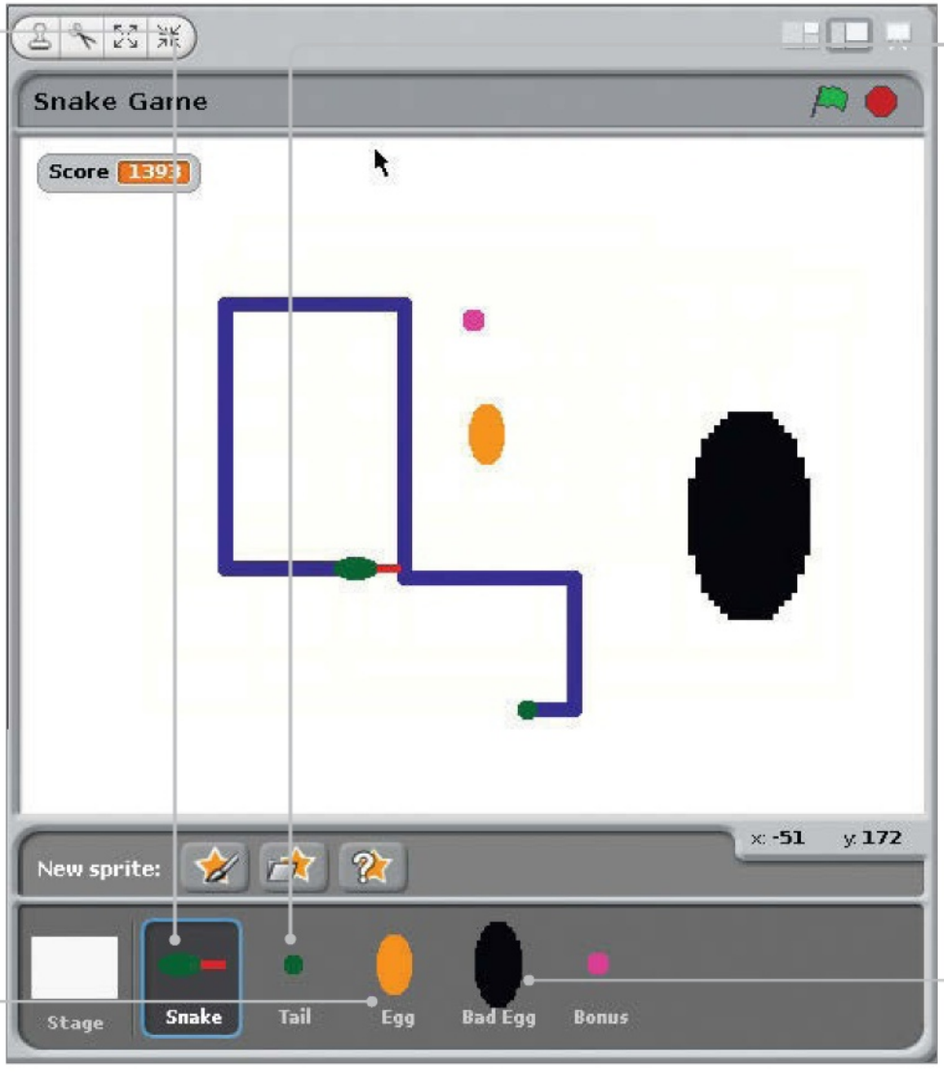
Add 'Play note () for 0.5 beats'. In the variable drop 'Item () of Keyboard'. Drag in the 'Note counter' variable. Add 'Change note counter by 1'. Drag in an 'If ()' condition and drop the '()=()' operator inside. Make this 'note counter = 21'. Add 'Set note counter to 1'.



08 Play some random sounds

Create another sprite for turning the keyboard off. Position bottom right and use the 'synth' variable as the toggle, like the drum off sprite. Create a new sprite and position. Add a 'When SpritX clicked' and then 'Set instrument to ()'. Drag in a 'Pick random' operator. Add a 'Play note'.

Snake sprite
The Snake sprite moves the head around the stage and draws the body behind it. It also detects collisions with the body or edge



Tail sprite
The Tail sprite follows the head, erasing the end of the body so the snake moves and pausing when it needs to grow

Egg sprite
The Egg sprite appears randomly on the screen and lets other sprites know when it has been eaten (touched by the snake tongue)

Bad Egg sprite
The Bad Egg sprite also appears randomly but decreases the score when eaten. It also grows in size, getting harder to avoid

Create a basic Snake game

Design a snake game where you must avoid hitting the snake body or the edge

Here, we will create a version of the classic Snake game where you move the snake around the Scratch stage using the arrow keys. You control the head of the snake and must avoid a collision with either the body of the snake or the edge of the stage.

The snake body grows longer each time you eat an egg. You get points added to your score for eating good yellow eggs and lose points for eating bad black eggs. There are also bonus sprites to eat for extra points.

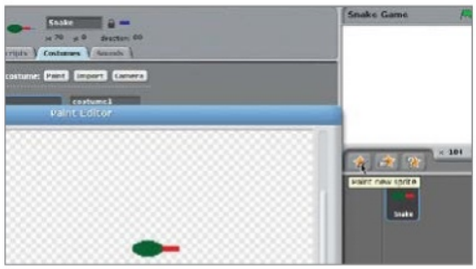
By following this tutorial you will learn to create your own simple sprite graphics, send and receive broadcast events, use a list variable to store data, play sound effects, generate random numbers and use sensing commands to detect when a sprite is touching something.

You will draw a graphic for each sprite using the built-in Paint Editor. Each sprite will have one or more scripts that run when you click the green flag above the stage and additional scripts that respond to key presses or events. For some sprites you will import a sound from the Scratch library via the Sound tab.

The game uses five sprites: Snake, Tail, Egg, Bad Egg and Bonus – you can delete Scratchy the cat. We'll build up the game one sprite at a time and you can test your project as you go along by clicking the green flag, then using the arrow keys to control your snake.

Resources

Scratch project archives
http://http://scratch.mit.edu/explore/?date=ever

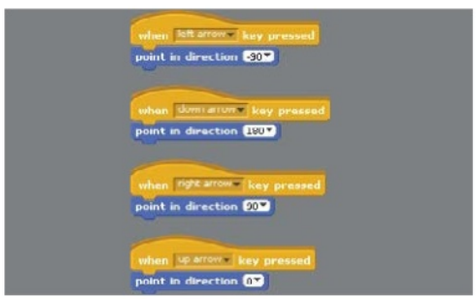


01 Paint the Snake sprite

Click the New Sprite: Paintbrush icon to paint the Snake sprite. In the Paint Editor, draw a small green ellipse for the snake head and add a red rectangle for the tongue. It's important that the tongue is a different colour to the head. Name your sprite Snake.

02 Add a Snake sound

When the Snake tongue touches the snake body or the edge of the stage, we are going to play a Game Over sound. We need to add this sound to the Snake sprite. With the Snake sprite selected, click the Sound tab and choose Import. Select the Electronic>Screech sound.

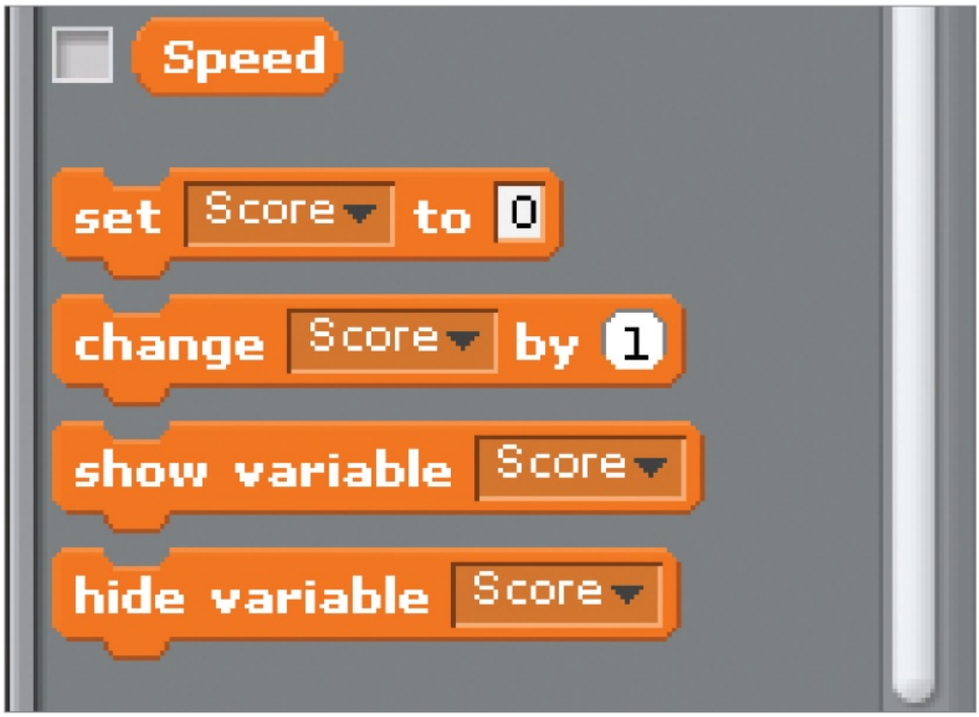


03 Respond to arrow keys

Drag four **when key pressed** commands from the Control palette, and four **point in direction** commands from the Motion palette. Configure them as shown so that the up arrow changes the direction to 0 degrees (up) and so on. Click the green flag above the stage to test this.

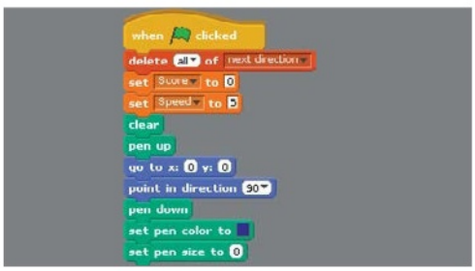
04 Make Snake variables

Click on the Variables palette. Make two variables, Score and Speed, that are visible to all sprites. Make a list called Next Direction which is visible to all sprites; it will store the sequence of directions that the head takes. Only have the Score variable checked so it appears on the stage.



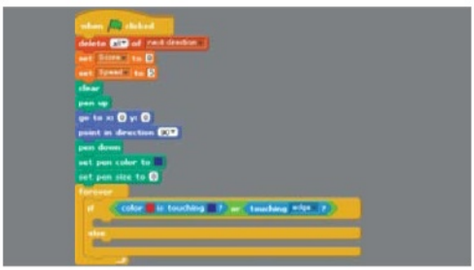
05 Initialise Snake variables

Use a 'when green flag clicked' Control command and initialise the Snake variables as shown, using commands from the Variables palette. We want to start each game with an empty Next Direction list, so delete all of its entries. The Score must start at zero. The Speed sets the game difficulty.



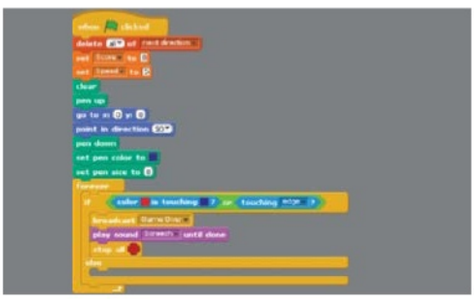
06 Draw Snake body

Use commands from the Pen palette to control the drawing of the snake body. Use commands from the Motion palette to move the snake to the centre of the stage and point left at the beginning of each game. The pen is up until the snake is in the starting position.



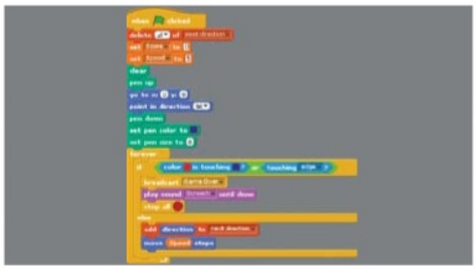
07 Add main action loop

Use a 'forever' command with an 'if-else' command nested inside. We have a collision if the red tongue is touching the blue body (the head is always touching the body) or the Snake sprite is touching the edge. Use the Eyedropper tool to select colours within Scratch.



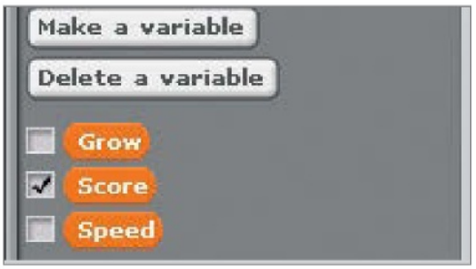
08 Handle Game Over

When a collision has been detected, broadcast a Game Over event (you'll need to create a new event) to the other sprites so they can also react. Also, play the Screech sound effect and stop all scripts so that the snake freezes in its current position at the end of the game.



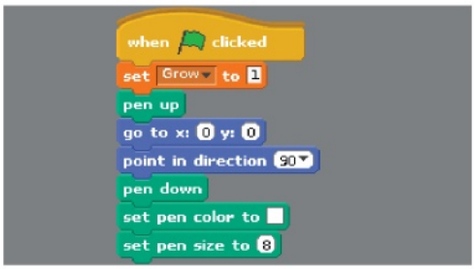
09 Handle movement

Now handle the typical case where there is no collision and the snake must move in its current direction. The pen is down so it will draw the body. The Speed variable determines how many steps to move. Add the current direction to the Next Direction list for the tail to read.



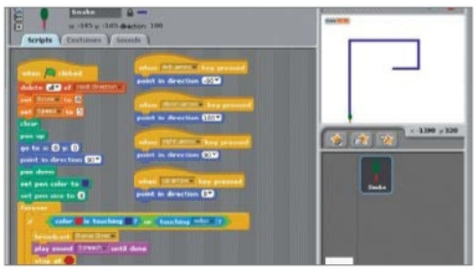
12 Make a Grow variable

Make a Grow variable which is for this sprite only – no other sprites need access to it. The Grow variable is used to determine when the snake body needs to grow and the tail therefore needs to pause before following to allow the head to get further ahead.



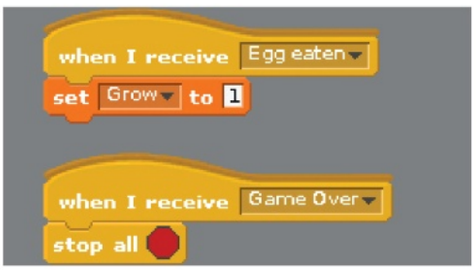
14 Initialise the tail

When the green flag is clicked to start the game, Grow is set to 1 so the snake gets a short body. Move to the centre of the stage with the pen up and configure the pen to draw a trail the same colour and size as the stage background so that it erases the body.



10 Try out the snake

You can now try out your Snake sprite. It will move around the screen in response to pressing the arrow keys. It will draw its body, which will just get longer and longer because we need the tail to erase it. And it will screech and end the game on detecting a collision.



13 Handle events

The Tail needs to listen for two new events which you create as you need them. When it receives an Egg Eaten event from one of the Egg sprites, it sets Grow=1 so that the tail can pause. And when it receives a Game Over event from the Snake, it must freeze.

15 Grow and move

Use a 'forever' command to keep the tail moving. If Grow is 1 it should pause and reset Grow to 0 – this makes the body grow longer. Use the first value from Next Direction to set the direction and remove it so you get a new value next time. Move Speed steps.

11 Paint the tail

Click the New Sprite: Paintbrush icon to paint the Tail sprite. Draw a small green circle to represent the end of the tail. Name this sprite Tail. The Tail sprite will follow the Snake and erase the end of its tail so that the snake body doesn't grow indefinitely.

"You can now try out your Snake sprite. It will move around the screen in response to pressing the arrow keys"

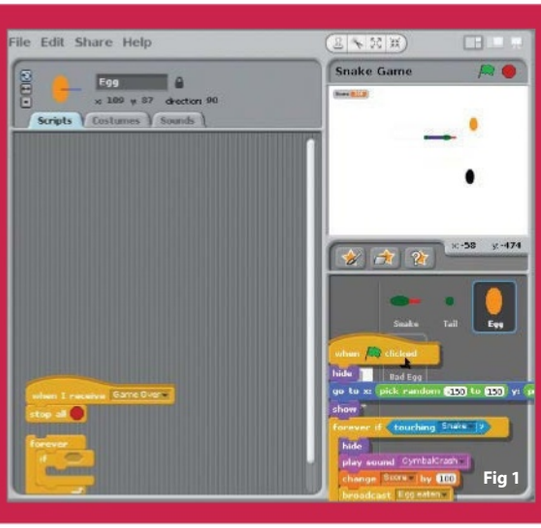


Fig 1

■ Fig 1: To copy a script to another sprite, drag the script over the sprite until a grey box appears around the target sprite and release

■ Fig 2: To make a circle in the Paint Editor tool, hold down the Shift key while you draw with the ellipse tool

■ Fig 3: When you have completed creating the game, you can switch to presentation mode to play it full-screen

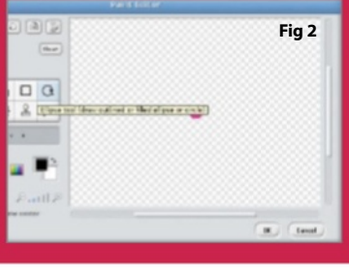


Fig 2

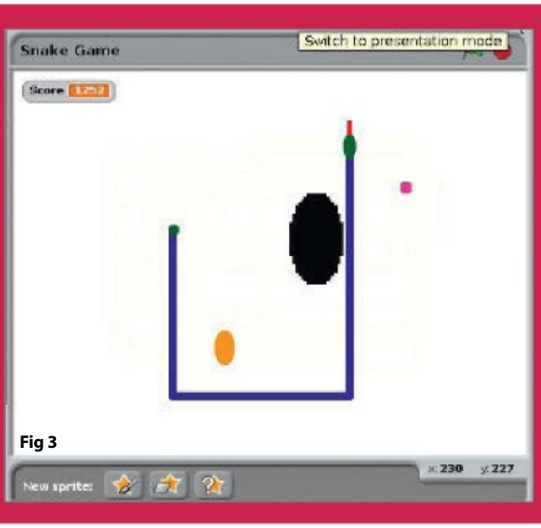
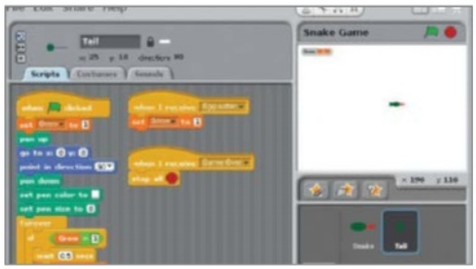


Fig 3



16 Try out the tail

Now you can try out the Tail sprite. The snake won't keep growing yet because it won't receive any Egg Eaten events. But the tail will follow the snake head around the stage, erasing the snake body as it goes by drawing over it with a white pen (the same colour as the background).

17 Paint the Egg sprite

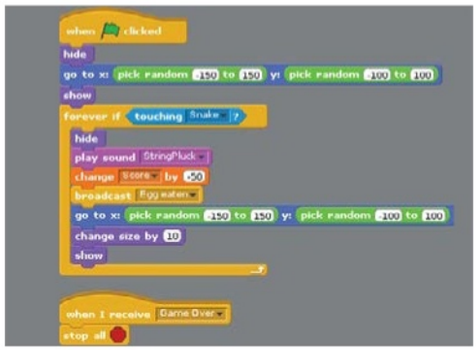
Click the New Sprite: Paintbrush icon to paint a new sprite. In the Paint Editor, draw a small yellow ellipse. Name the sprite Egg. The Egg will appear randomly on the stage and cause the snake to grow and increase its score.

18 Add Egg sound

Go the the Sounds tab for the Egg sprite and import the Percussion>Cymbal Crash sound. Or you can choose a different sound if you like. This sound will play when the snake eats an egg.

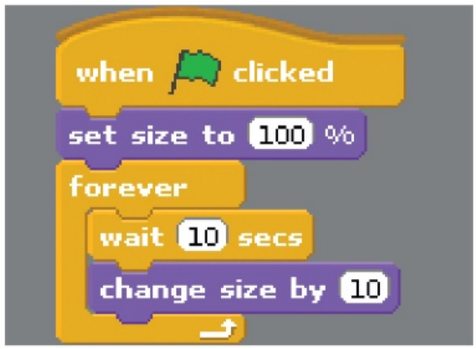
19 Add Egg scripts

Copy the Egg script so that the Egg appears randomly at the start of the game. When the Egg senses that it has been eaten, it must hide, play the Cymbal sound (or whatever you chose in step 18), update the score, broadcast the Egg Eaten event and then randomly appear again. When the Game Over event is received, it must stop.



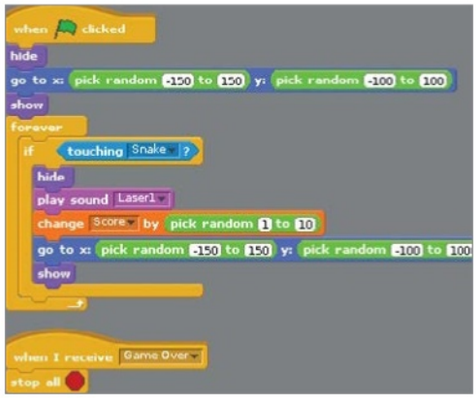
20 Make Bad Egg

Create the black Bad Egg in the same way as the Egg but using a different graphic and the Instruments>StringPluck sound. You can drag the Egg's green flag scripts onto the Bad Egg to copy them – just change the sound that's played and reduce the score instead of increasing it.



21 Grow Bad Egg

Add another 'when green flag clicked' script to the Bad Egg so it sets its size to the default 100% when a new game is started and then increases its size by 10 every 10 seconds. The Bad Egg will get bigger and bigger and harder to avoid.



22 Create Bonus sprite

Create the Bonus sprite in a similar way. You can choose the shape and sound for the Bonus. Its scripts are similar to the Egg ones so you could drag one of those to the Bonus sprite and work from that. Make sure you change the sound and increase the score by a random bonus.

Using multiple sprites in Scratch

Each sprite operates independently and they can communicate using events

The characters or objects in a Scratch game are called sprites. Scratch comes with a library of sprites. You can draw your own, as we did in the Snake game, using the Scratch Paint Editor or you can create them in an external drawing tool like Inkscape and import them.

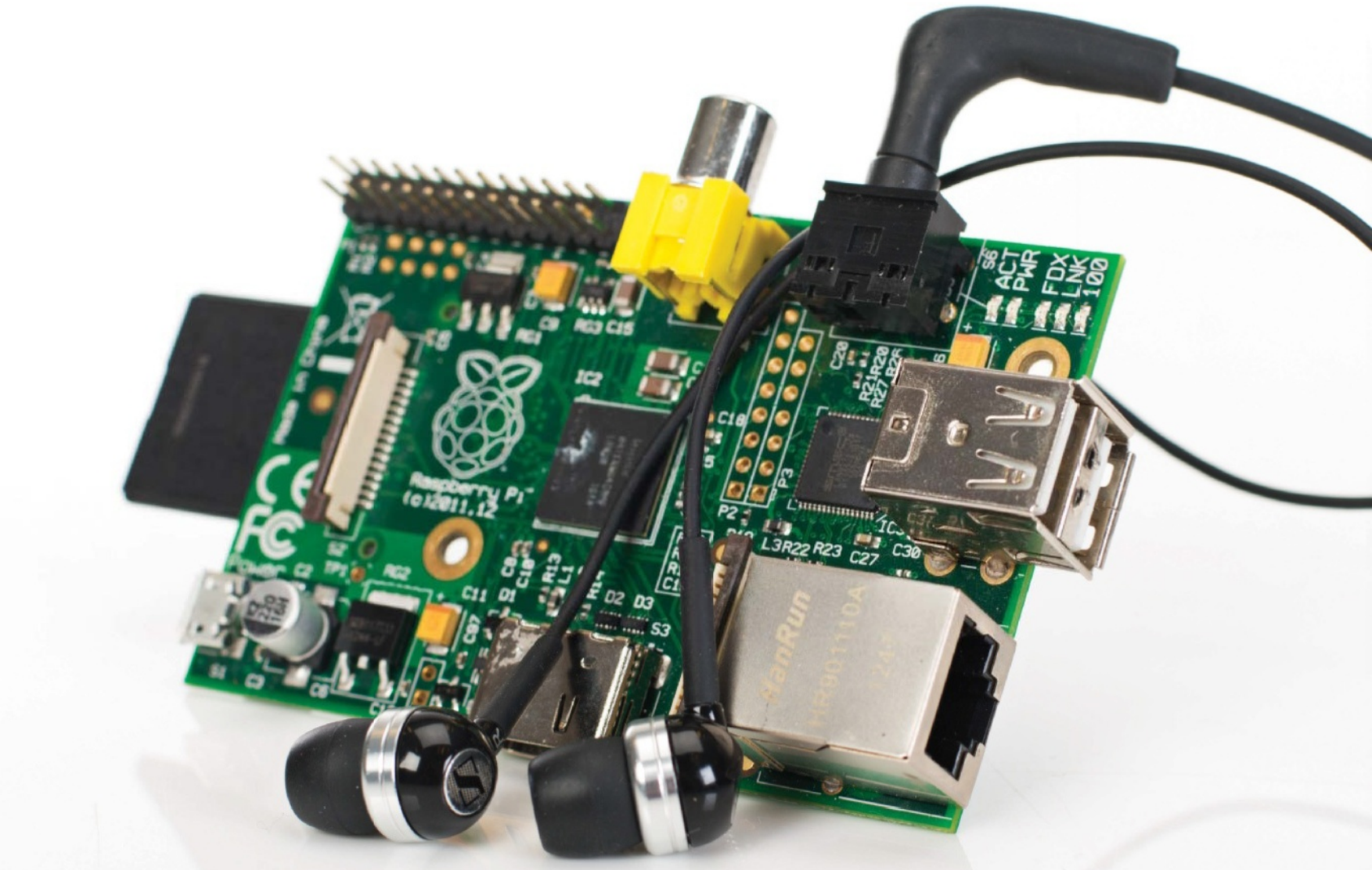
Sprites can have multiple costumes to change their appearance. We only used one costume per sprite, but you could extend the project so that the snake head changes appearance briefly when it eats a Bad Egg.

Complex Scratch projects often use multiple sprites to manage complexity. In the Snake game, each sprite has its own responsibilities.

The sprites communicate with each other using global variables that are visible to all sprites, and events that can be broadcast and received by sprites. We used Egg Eaten and Game Over events to communicate between sprites. And the Speed, Score and Next Direction variables were all used by more than one sprite.

If you have sprites that are similar to each other, you can right-click on a sprite and choose Duplicate to create another sprite the same as the first. You can then modify your first sprite. We could have used this approach to create the Bad Egg and Bonus sprites from the Egg sprite. You could use this technique to add a new kind of good or bad sprite to the game.

When you design your own Scratch projects, think about how you can logically divide the behaviour between multiple sprites to keep things simple.



Learn to code with Sonic Pi

Take the next step in programming and create your own melodies with Sonic Pi, the musical programming language

Resources

Speakers or headphones

Sonic Pi

With the Scratch tutorial on pages 126-127, we've learned how to operate under the logic of programming. The next step is to then use that within a programming language – the problem is that many of the available languages can look a little intimidating. This is where Sonic Pi comes in, offering a very simple language style that can ease you in to the basics of working with code. It's quite straightforward to use as well – Sonic Pi allows you to choose from a small selection of instruments and select a tone to play with it. These can be turned into complex melodies using loops and threads and even some form of user input.

Here's how to get started…

"Choose from a small selection of instruments"

01 Install Raspbian

If you've installed the latest version of Raspbian, Sonic Pi will be included by default. If you're still using a slightly older version, then you'll need to install it via the repos. Do this with:

```
$ sudo apt-get install sonic-pi
```

02 Get started with Sonic Pi

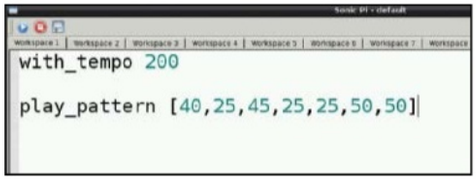
Sonic Pi is located in the Education category in the menus. Open it up and you'll be presented with something that looks like an IDE. The pane on the left allows you to enter code, and then you can save and preview it as well. Any errors are displayed separately from the output.

03 Your first note

Our first thing to try out with Sonic Pi is simply being able to play a note. Sonic Pi has a few default notes already pre-set, so we can get started with:

```
play 50
```

Press run and the output window should show you exactly what is happening.

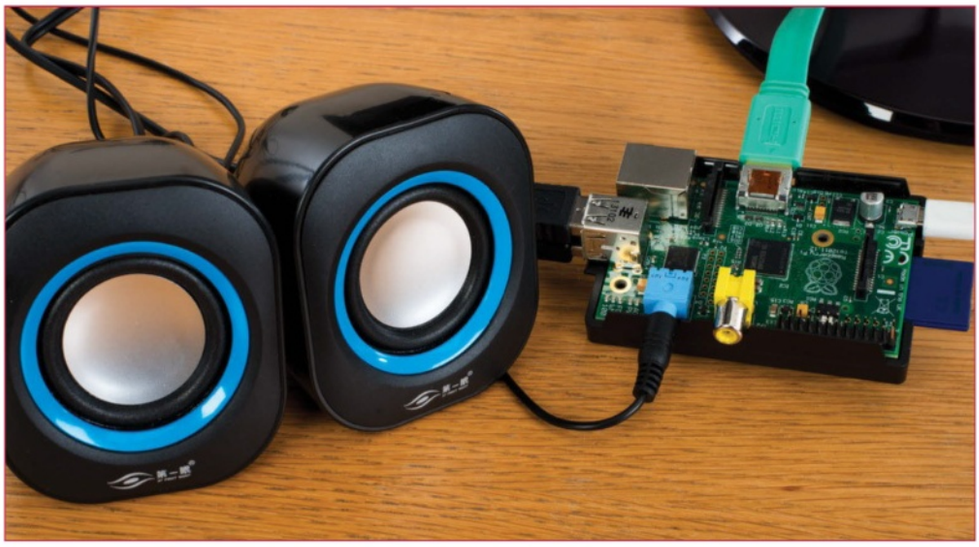


04 Set the beat

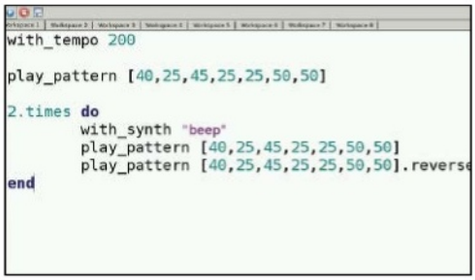
For any piece of music, you'll probably want to set the beat. We can start by putting:

```
with_tempo 200
```

At the start of our code. We can then test this out by creating a string of midi notes using play_pattern.



"Sonic Pi offers you a very simple language style that can ease you into the basics of working with code"



05 Advance your melody

We can start making more complex melodies by using more of Sonic Pi's functions. You can change the note type by using with_synth, reverse a pattern, and even create a finite loop with the x.times function. 'Do' and 'end' signify the start and end of the loop.

06 Play a concert

Using the in_thread function, we can create another thread for the Sonic Pi instance and have several lines of musical code play at once instead of in sequence. Here we've made it create a series of notes in a random sequence.

Full Code Listing:

```
with_tempo 200

play_pattern [40,25,45,25,25,50,50]

2.times do
   with_synth "beep"
   play_pattern [40,25,45,25,25,50,50]
   play_pattern [40,25,45,25,25,50,50].
reverse
end

play_pad "saws", 3

in_thread do
   with_synth "fm"
   6.times do
         if rand < 0.5
               play 30
         else
               play 50
         end
         sleep 2
   end
end

2.times do
   play_synth "pretty_bell"
   play_pattern [40,25,45,25,25,50,50]
   play_pattern [40,25,45,25,25,50,50].
reverse
end
```

Python masterclass

Always wanted to have a go at programming? No more excuses, because Python is the perfect way to get started!

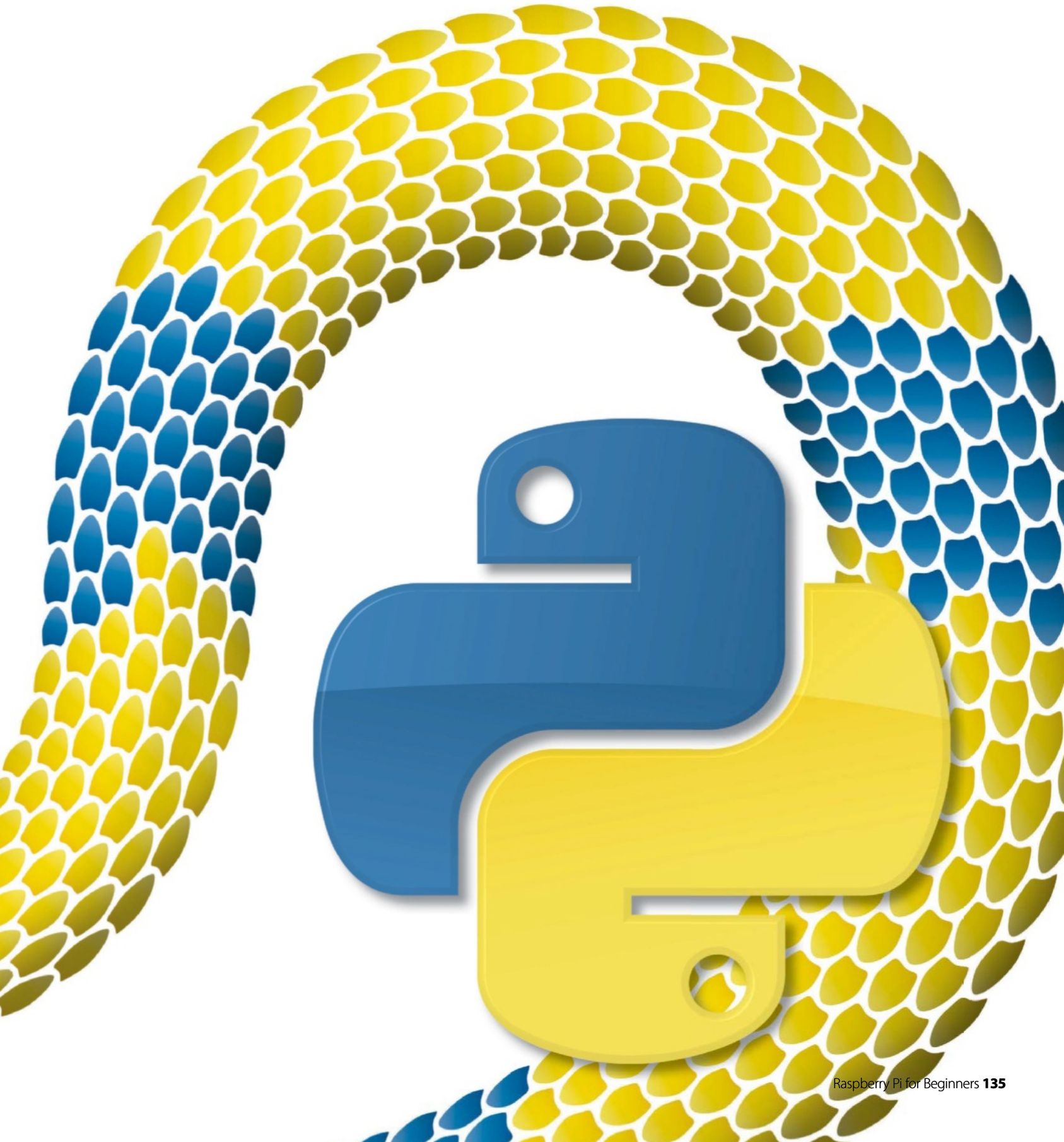
Python is a great programming language for both beginners and experts. It is designed with code readability in mind, making it an excellent choice for beginners who are still getting used to various programming concepts.

The language is popular and has plenty of libraries available, allowing programmers to get a lot done with relatively little code.

You can make all kinds of applications in Python: you could use the Pygame framework to write simple 2D games, you could use the GTK libraries to create a windowed application, or you could try something a little more ambitious like an app such as creating one using Python's Bluetooth and Input libraries to capture the input from a USB keyboard and relay the input events to an Android phone.

For this tutorial we're going to be using Python 2.x since that is the version that is most likely to be installed on your Linux distribution.

In the following tutorials, you'll learn how to create popular games using Python programming. We'll also show you how to add sound and AI to these games.



TIP

If you were using a graphical editor such as gedit, then you would only have to do the last step of making the file executable. You should only have to mark the file as executable once. You can freely edit the file once it is executable.

Hello World

Let's get stuck in, and what better way than with the programmer's best friend, the 'Hello World' application! Start by opening a terminal. Its current working directory will be your home directory. It's probably a good idea to make a directory for the files we'll be creating in this tutorial, rather than having them loose in your home directory. You can create a directory called Python using the command mkdir Python. You'll then want to change into that directory using the command cd Python.

The next step is to create an empty file using the command 'touch' followed by the filename. Our expert used the command touch hello_world.py. The final and most important part of setting up the file is making it executable. This allows us to run code inside the hello_world.py file. We do this with the command chmod +x hello_world.py. Now that we have our file set up, we can go ahead and open it up in nano, or any text editor of your choice. Gedit is a great editor with syntax highlighting support that should be available on any distribution. You'll be able to install it using your package manager if you don't have it already.

```
[liam@liam-laptop ~]$ mkdir Python
[liam@liam-laptop ~]$ cd Python/
[liam@liam-laptop Python]$ touch hello_world.py
[liam@liam-laptop Python]$ chmod +x hello_world.py
[liam@liam-laptop Python]$ nano hello_world.py
```

Our Hello World program is very simple, it only needs two lines. The first line begins with a 'shebang' (the symbol #! – also known as a hashbang) followed by the path to the Python interpreter. The program loader uses this line to work out what the rest of the lines need to be interpreted with. If you're running this in an IDE like IDLE, you don't necessarily need to do this.

The code that is actually read by the Python interpreter is only a single line. We're passing the value Hello World to the print function by placing it in brackets immediately after we've called the print function. Hello World is enclosed in quotation marks to indicate that it is a literal value and should not be interpreted as source code. As expected, the print function in Python prints any value that gets passed to it from the console.

You can save the changes you've made to the file in nano using the key combination Ctrl+O, followed by Enter. Use Ctrl+X to exit nano.

TIP

Python has plenty of great online documentation. Usually the best way to find things is to simply Google them and the first result will be the official Python documentation. For example, there is a very detailed page on Python's built-in types here: **docs.python.org/2/library/stdtypes.html**

```
#!/usr/bin/env python2
print("Hello World")
```

You can run the Hello World program by prefixing its filename with ./ – in this case you'd type: ./hello_world.py.

```
[liam@liam-laptop Python]$ ./hello_world.py
Hello World
```

Variables and data types

A variable is a name in source code that is associated with an area in memory that you can use to store data, which is then called upon throughout the code. The data can be one of many types, including:

Integer	Stores whole numbers
Float	Stores decimal numbers
Boolean	Can have a value of True or False
String	Stores a collection of characters. "Hello World" is a string

As well as these main data types, there are sequence types (technically, a string is a sequence type but is so commonly used we've classed it as a main data type):

List	Contains a collection of data in a specific order
Tuple	Contains a collection immutable data in a specific order

A tuple would be used for something like a co-ordinate, containing an x and y value stored as a single variable, whereas a list is typically used to store larger collections. The data stored in a tuple is immutable because you can't change values of individual elements in a tuple. However, you can do so in a list.

It will also be useful to know about Python's dictionary type. A dictionary is a mapped data type. It stores data in key-value pairs. This means that you access values stored in the dictionary using that value's corresponding key, which is different to how you would do it with a list. In a list, you would access an element of the list using that element's index (a number representing the element's position in the list).

Let's work on a program we can use to demonstrate how to use variables and different data types. It's worth noting at this point that you don't always have to specify data types in Python. Feel free to create this file in any editor you like. Everything will work just fine as long as you remember to make the file executable. We're going to call ours variables.py.

> "A variable is a name in source code that is associated with an area in memory that you can use to store data"

Interpreted vs compiled languages

An interpreted language such as Python is one where the source code is converted to machine code and then executed each time the program runs. This is different from a compiled language such as C, where the source code is only converted to machine code once – the resulting machine code is then executed each time the program runs.

```
#!/usr/bin/env python2

# We create a variable by writing the name of the variable we want followed
# by an equals sign, which is followed by the value we want to store in the
# variable. For example, the following line creates a variable called
# hello_str, containing the string Hello World.
hello_str = "Hello World"

hello_int = 21

hello_bool = True

hello_tuple = (21, 32)

hello_list = ["Hello,", "this", "is", "a", "list"]

# This list now contains 5 strings. Notice that there are no spaces
# between these strings so if you were to join them up so make a sentence
# you'd have to add a space between each element.

hello_list = list()
hello_list.append("Hello,")
hello_list.append("this")
hello_list.append("is")
hello_list.append("a")
hello_list.append("list")

# The first line creates an empty list and the following lines use the append
# function of the list type to add elements to the list. This way of using a
# list isn't really very useful when working with strings you know of in
# advance, but it can be useful when working with dynamic data such as user
# input. This list will overwrite the first list without any warning as we
# are using the same variable name as the previous list.

hello_dict = { "first_name" : "Liam",
               "last_name"  : "Fraser",
               "eye_colour" : "Blue" }

# Let's access some elements inside our collections
# We'll start by changing the value of the last string in our hello_list and
# add an exclamation mark to the end. The "list" string is the 5th element
# in the list. However, indexes in Python are zero-based, which means the
# first element has an index of 0.

print(hello_list[4])
hello_list[4] += "!"
# The above line is the same as
hello_list[4] = hello_list[4] + "!"
print(hello_list[4])
```

The following line creates an integer variable called hello_int with the # value of 21. Notice how it doesn't need to go in quotation marks

The same principal is true of Boolean values

We create a tuple in the following way

And a list in this way

You could also create the same list in the following way

We might as well create a dictionary while we're at it. Notice how we've aligned the colons below to make the code tidy

Notice that there will now be two exclamation marks when we print the element

TIP

At this point, it's worth explaining that any text in a Python file that follows a # character will be ignored by the interpreter. This is so you can write comments in your code.

"Any text in a Python file that follows a # character will be ignored"

Remember that tuples are immutable, although we can access the elements of them like so

Let's create a sentence using the data in our hello_dict

A tidier way of doing this would be to use Python's string formatter

```
print(str(hello_tuple[0]))
# We can't change the value of those elements like we just did with the list
# Notice the use of the str function above to explicitly convert the integer
# value inside the tuple to a string before printing it.

print(hello_dict["first_name"] + " " + hello_dict["last_name"] + " has " +
    hello_dict["eye_colour"] + " eyes.")

print("{0} {1} has {2} eyes.".format(hello_dict["first_name"],
                                     hello_dict["last_name"],
                                     hello_dict["eye_colour"]))
```

Control structures

In programming, a control structure is any kind of statement that can change the path that the code execution takes. For example, a control structure that decided to end the program if a number was less than 5 would look something like this:

```
#!/usr/bin/env python2

import sys # Used for the sys.exit function

int_condition = 5

if int_condition < 6:
    sys.exit("int_condition must be >= 6")
else:
    print("int_condition was >= 6 - continuing")
```

The path that the code takes will depend on the value of the integer int_condition. The code in the 'if' block will only be executed if the condition is true. The import statement is used to load the Python system library; the latter provides the exit function, allowing you to exit the program, printing an error message. Notice that indentation (in this case four spaces per indent) is used to indicate which statement a block of code belongs to.

'If' statements are probably the most commonly used control structures. Other control structures include:

- For statements, which allow you to iterate over items in collections, or to repeat a piece of code a certain number of times;
- While statements, a loop that continues while the condition is true.

We're going to write a program that accepts user input from the user to demonstrate how control structures work. We're calling it **construct.py**.

The 'for' loop is using a local copy of the current value, which means any changes inside the loop won't make any changes affecting the list. On the other hand however, the 'while' loop is directly accessing elements in the list, so you could change the list there should you want to do so. We will talk about variable scope in some more detail later on. The output from the above program is as follows:

More about a Python list

A Python list is similar to an array in other languages. A list (or tuple) in Python can contain data of multiple types, which is not usually the case with arrays in other languages. For this reason, we recommend that you only store data of the same type in a list. This should almost always be the case anyway due to the nature of the way data in a list would be processed.

Indentation in detail

As previously mentioned, the level of indentation dictates which statement a block of code belongs to. Indentation is mandatory in Python, whereas in other languages, sets of braces are used to organise code blocks. For this reason, it is essential that you use a consistent indentation style. Four spaces are typically used to represent a single level of indentation in Python. You can use tabs, but tabs are not well defined, especially if you happen to open a file in more than one editor.

```
[liam@liam-laptop Python]$ ./construct.py
How many integers? acd
You must enter an integer

[liam@liam-laptop Python]$ ./construct.py
How many integers? 3
Please enter integer 1: t
You must enter an integer
Please enter integer 1: 5
Please enter integer 2: 2
Please enter integer 3: 6
Using a for loop
5
2
6
Using a while loop
5
2
6
```

"The 'for' loop is using a local copy of the current value, which means any changes inside the loop won't affect the list"

```
#!/usr/bin/env python2

# We're going to write a program that will ask the user to input an arbitrary
# number of integers, store them in a collection, and then demonstrate how the
# collection would be used with various control structures.

import sys # Used for the sys.exit function

target_int = raw_input("How many integers? ")

# By now, the variable target_int contains a string representation of
# whatever the user typed. We need to try and convert that to an integer but
# be ready to # deal with the error if it's not. Otherwise the program will
# crash.
try:
   target_int = int(target_int)
except ValueError:
   sys.exit("You must enter an integer")


ints = list()

count = 0

# Keep asking for an integer until we have the required number
while count < target_int:
   new_int = raw_input("Please enter integer {0}: ".format(count + 1))
   isint = False
   try:
      new_int = int(new_int)

   except:
      print("You must enter an integer")

   # Only carry on if we have an integer. If not, we'll loop again
   # Notice below I use ==, which is different from =. The single equals is an
   # assignment operator whereas the double equals is a comparison operator.

   if isint == True:
      # Add the integer to the collection
      ints.append(new_int)
      # Increment the count by 1
      count += 1


print("Using a for loop")
for value in ints:
   print(str(value))
```

The number of integers we want in the list

A list to store the integers

These are used to keep track of how many integers we currently have

If the above succeeds then isint will be set to true: isint =True

By now, the user has given up or we have a list filled with integers. We can loop through these in a couple of ways. The first is with a for loop

```
# Or with a while loop:
print("Using a while loop")
# We already have the total above, but knowing the len function is very
# useful.
total = len(ints)
count = 0
while count < total:
    print(str(ints[count]))
    count += 1
```

Functions and variable scope

Functions are used in programming to break processes down into smaller chunks. This often makes code much easier to read. Functions can also be reusable if designed in a certain way. Functions can have variables passed to them. Variables in Python are always passed by value, which means that a copy of the variable is passed to the function that is only valid in the scope of the function. Any changes made to the original variable inside the function will be discarded. However, functions can also return values, so this isn't an issue. Functions are defined with the keyword def, followed by the name of the function. Any variables that can be passed through are put in brackets following the function's name. Multiple variables are separated by commas. The names given to the variables in these brackets are the ones that they will have in the scope of the function, regardless of what the variable that's passed to the function is called. Let's see this in action.

The output from the program opposite is as follows:

> "Functions are used in programming to break processes down in"

TIP

You can define defaults for variables if you want to be able to call the function without passing any variables through at all. You do this by putting an equals sign after the variable name. For example, you can do: `def modify_string (original=" Default String")`

```
#!/usr/bin/env python2

# Below is a function called modify_string, which accepts a variable
# that will be called original in the scope of the function. Anything
# indented with 4 spaces under the function definition is in the
# scope.
def modify_string(original):
    original += " that has been modified."
    # At the moment, only the local copy of this string has been modified

def modify_string_return(original):
    original += " that has been modified."
    # However, we can return our local copy to the caller. The function
    # ends as soon as the return statement is used, regardless of where it
    # is in the function.
    return original


test_string = "This is a test string"

modify_string(test_string)
print(test_string)

test_string = modify_string_return(test_string)
print(test_string)

# The function's return value is stored in the variable test string,
# overwriting the original and therefore changing the value that is
# printed.
```

We are now outside of the scope of the modify_ string function, as we have reduced the level of indentation

The test string won't be changed in this code

However, we can call the function like this

```
[liam@liam-laptop Python]$ ./functions_and_scope.py
This is a test string
This is a test string that has been modified.
```

Scope is an important thing to get the hang of, otherwise it can get you into some bad habits. Let's write a quick program to demonstrate this. It's going to have a Boolean variable called cont, which will decide if a number will be assigned to a variable in an if statement. However, the variable hasn't been defined anywhere apart from in the scope of the if statement. We'll finish off by trying to print the variable.

```
#!/usr/bin/env python2
cont = False
if cont:
    var = 1234
print(var)
```

In the code above, Python will convert the integer to a string before printing it. However, it's always a good idea to explicitly convert things to strings – especially when it comes to concatenating strings together. If you try to use the + operator on a string and an integer, there will be an error because it's not explicitly clear what needs to happen. The + operator would usually add two integers together. Having said that, Python's string formatter that we demonstrated earlier is a cleaner way of doing that. Can you see the problem? Var has only been defined in the scope of the if statement. This means that we get a very nasty error when we try to access var.

```
[liam@liam-laptop Python]$ ./scope.py
Traceback (most recent call last):
  File "./scope.py", line 8, in <module>
    print var
NameError: name 'var' is not defined
```

If cont is set to True, then the variable will be created and we can access it just fine. However, this is a bad way to do things. The correct way is to initialise the variable outside of the scope of the if statement.

```
#!/usr/bin/env python2

cont = False

var = 0
if cont:
    var = 1234

if var != 0:
    print(var)
```

The variable var is defined in a wider scope than the if statement, and can still be accessed by the if statement. Any changes made to var inside the if statement are changing the variable defined in the larger scope. This example doesn't really do anything useful apart from illustrate the potential problem, but the worst-case scenario has gone from the program crashing to printing a zero. Even that doesn't happen because we've added an extra construct to test the value of var before printing it.

Coding style

It's worth taking a little time to talk about coding style. It's simple to write tidy code. The key is consistency. For example, you should always name your variables in the same manner. It doesn't matter if you want to use camelCase or use underscores as we have.

Comparison operators

The common comparison operators available in Python include:

<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal
!=	not equal

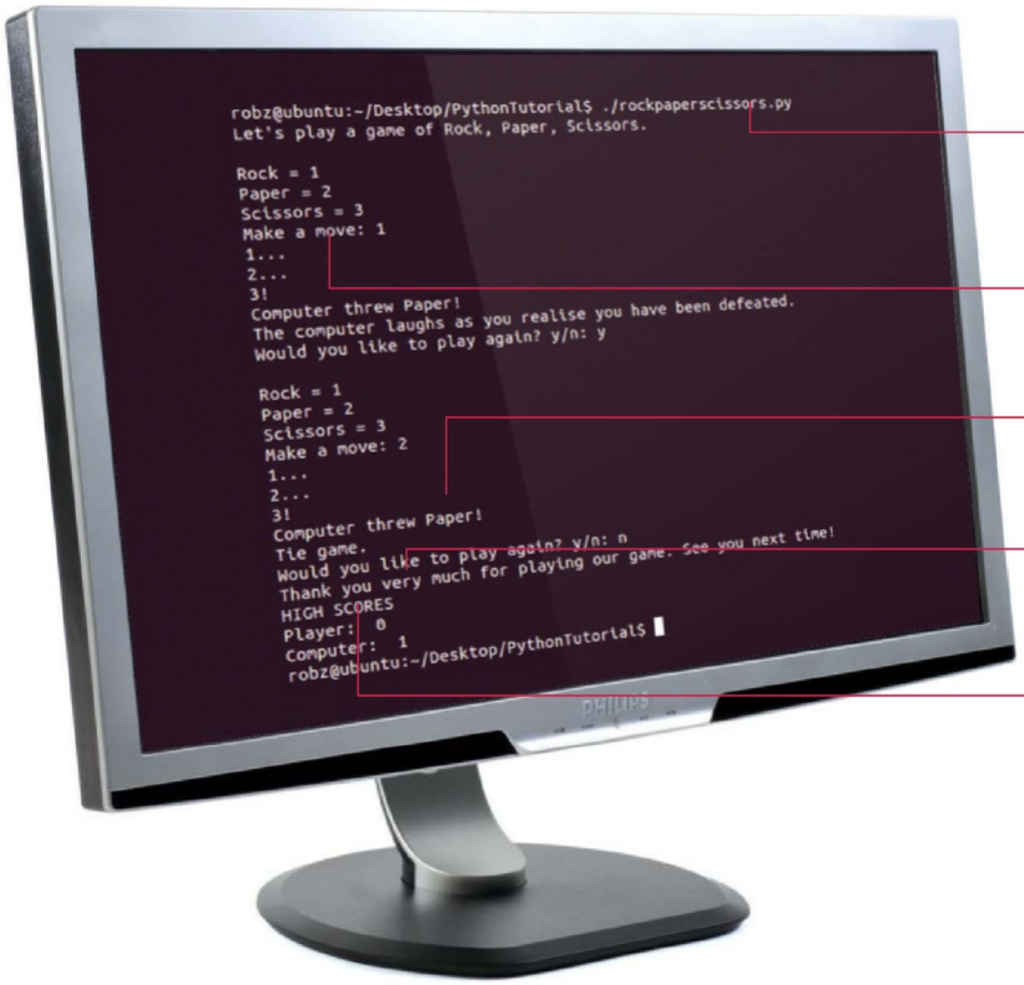
One crucial thing is to use self-documenting identifiers for variables. You shouldn't have to guess what a variable does. The other thing that goes with this is to always comment your code. This will help anyone else who reads your code, and yourself in the future. It's also useful to put a brief summary at the top of a code file describing what the application does, or a part of the application if it's made up of multiple files.

Summary

This article should have introduced you to the basics of programming in Python. Hopefully you are getting used to the syntax, indentation and general look and feel of a Python program. The next step is to learn how to come up with a problem that you want to solve, and break it down into small enough steps that you can implement in a programming language.

Google, or any other search engine, is very helpful. If you are stuck with anything, or have an error message you can't work out how to fix, stick it into Google and you should be a lot closer to solving your problem. For example, if we Google 'play mp3 file with python', the first link takes us to a Stack Overflow thread with a bunch of useful replies. Don't be afraid to get stuck in – the real fun of programming is solving problems one manageable chunk at a time.

Happy Programming!



Allow the Python script to run in a terminal, and outside the IDE

Human input in the form of integers is used for comparing moves and, ultimately, playing the game

Use deduction to determine one of three outcomes

Loop the code over again and start from the beginning

Append to integer variables to keep track of scores and more

Learn basic coding by building a simple game

Perform some basic Python coding by following our breakdown of a Rock, Paper, Scissors game

Resources

Python 2:
www.python.org/download

IDLE:
www.python.org/idle

To complement our main Python feature, we've put together a tutorial to guide you through making a Rock, Paper, Scissors game in Python. The code applies the lessons from the feature and doesn't require any extra Python modules to run.

"Rock, Paper, Scissors is the perfect game to show off a little more about Python. It's also easy to adapt and expand"

Rock, Paper, Scissors is the perfect game to show off a little more about Python. Human input, comparisons, random selections and a whole host of loops are used in making a working version of the game. It's also easy enough to adapt and expand as you see fit, adding rules and results, and even making a rudimentary AI if you wish.

For this tutorial, we also recommend using IDLE. IDLE is a great Python IDE that is easily obtainable in most Linux distributions and is available by default on Raspbian for Raspberry Pi. It highlights any problems that there may be with your code and allows you to easily run it to make sure that everything is working properly.

The code dump

01 This section imports the extra Python functions we'll need for the code – they're still parts of the standard Python libraries, just not part of the default environment.

02 The initial rules of the game are created here. The three variables that we are using and their relationship is defined. We also provide a variable so we can keep score of the games.

03 We begin the game code by defining the start of each round. The end of each play session comes back through here, whether we want to play again or not.

04 The game is actually contained all in here, asking for the player input, getting the computer input and passing these on to get the results. At the end of that, it then asks if you'd like to play again.

05 Player input is done here. We give the player information on how to play this particular version of the game and then allow their choice to be used in the next step. We also have something in place in case they enter an invalid option.

06 There are a few things going on when we show the results. First, we're putting in a delay to add some tension, appending a variable to some printed text, and then comparing what the player and computer did. Through an if statement, we choose what outcome to print, and how to update the scores.

07 We now ask for text input on whether or not someone wants to play again. Depending on their response, we go back to the start, or end the game and display the results.

```
#!/usr/bin/env python2

# Linux User & Developer presents: Rock, Paper, Scissors: The Video Game

import random
import time

rock = 1
paper = 2
scissors = 3

names = { rock: "Rock", paper: "Paper", scissors: "Scissors" }
rules = { rock: scissors, paper: rock, scissors: paper }


player_score = 0
computer_score = 0

def start():
    print "Let's play a game of Rock, Paper, Scissors."
    while game():
        pass
    scores()

def game():
    player = move()
    computer = random.randint(1, 3)
    result(player, computer)
    return play_again()

def move():
    while True:
        print
        player = raw_input("Rock = 1\nPaper = 2\nScissors = 3\nMake a move: ")
        try:
            player = int(player)
            if player in (1,2,3):
                return player
        except ValueError:
            pass
        print "Oops! I didn't understand that. Please enter 1, 2 or 3."

def result(player, computer):
    print "1..."
    time.sleep(1)
    print "2..."
    time.sleep(1)
    print "3!"
    time.sleep(0.5)
    print "Computer threw {0}!".format(names[computer])
    global player_score, computer_score
    if player == computer:
        print "Tie game."
    else:
        if rules[player] == computer:
            print "Your victory has been assured."
            player_score += 1
        else:
            print "The computer laughs as you realise you have been defeated."
            computer_score += 1

def play_again():
    answer = raw_input("Would you like to play again? y/n: ")
    if answer in ("y", "Y", "yes", "Yes", "Of course!"):
        return answer
    else:
        print "Thank you very much for playing our game. See you next time!"


def scores():
    global player_score, computer_score
    print "HIGH SCORES"
    print "Player: ", player_score
    print "Computer: ", computer_score

if __name__ == '__main__':
    start()
```

The breakdown

01 As we explained earlier, we need to start with the path to the Python interpreter. This allows us to run the program inside a terminal or otherwise outside of a Python-specific IDE like IDLE. Note that we're also using Python 2 for this particular script, which we need to specify in the code to make sure it calls upon the correct version from the system.

02 We're importing two extra modules on top of the standard Python code so we can use some extra functions throughout the code. We'll use the random module to determine what move the computer will throw, and the time module to pause the running of the code at key points. The time module can also be used to utilise dates and times, either to display them or otherwise.

03 We will be setting each move to a specific number so that once a selection is made by the player during the game, it will be equated to that specific variable. This makes the code slightly easier later on, as we won't need to parse any text for this particular function. If you wish to do so, you can add additional moves, and this will start here.

```
01 #!/usr/bin/env python2

   # Linux User & Developer presents: Rock, Paper, Scissors: The Video Game
02 import random
   import time
03 rock = 1
   paper = 2
   scissors = 3
04 names = { rock: "Rock", paper: "Paper", scissors: "Scissors" }
   rules = { rock: scissors, paper: rock, scissors: paper }
05
06 player_score = 0
   computer_score = 0
```

04 It is in this section here we need to specify the rules for the game, and the text representations of each move for the rest of the code. Once we have done this, when it is called upon, our script will print the names of any of the three available moves, this is mainly to tell the player how the computer has moved. These names are only equated to these variables when they are needed – this is because this way, the number assigned to each of them is maintained while it's needed.

05 Similar to the way the text names of the variables are defined and used only when needed, the rules are done in such a way that when comparing the results, our variables are momentarily modified. Further down in the code we'll explain properly what's happening, but basically after determining whether or not there's a tie, we'll see if the computer's move would have lost to the player move. If the computer move equals the losing throw to the player's move, you win.

06 To put it very simply, this creates a variable that can be used throughout the code to keep track of each player's scores. We need to start it at zero now so that it exists, otherwise if we defined it in a function, it would only exist inside that function and not elsewhere. The code adds a point to the computer or player depending on the outcome of the round and so their score will increase if they have been successful. We have no scoring system for tied games in this particular version so no points are awarded.

Python modules

There are other modules you can import with basic Python. Some of the major ones are shown to the right. There are also many more that are included as standard with Python.

string	Perform common string operations
datetime and calendar	Other modules related to time
math	Advanced mathematical functions
json	JSON encoder and decoder
pydoc	Documentation generator and online help system

07 Here we define the actual beginning of the code, with the function we've called 'start'. It's quite simple, printing our greeting to the player and then starting a while loop that will allow us to keep playing the game as many times as we wish. The pass statement allows the while loop to stop once we've finished, and could be used to perform a number of other tasks if so wished. If we do stop playing the game, the score function is then called upon – we'll go over what that does when we get to it.

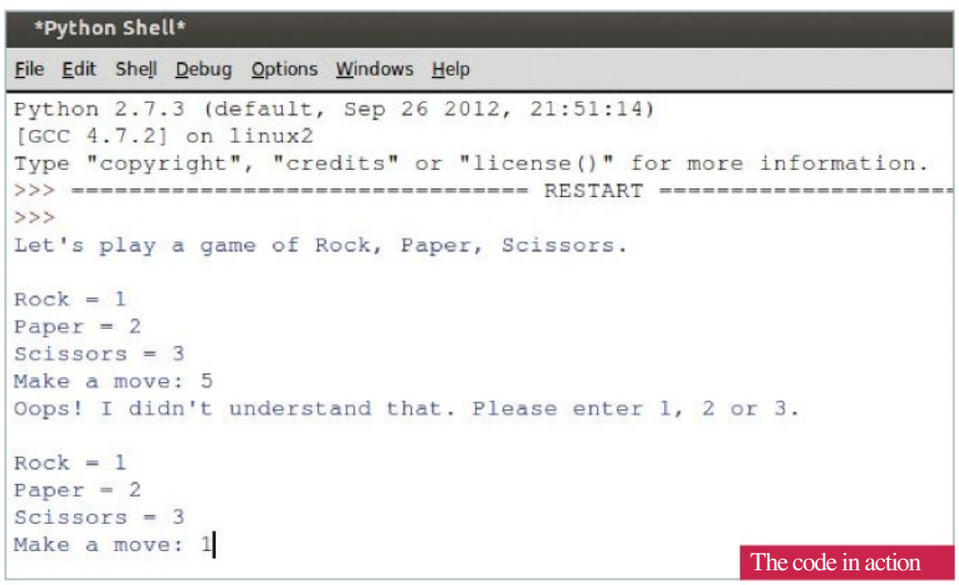
08 We've kept the game function fairly simple so we can break down each step a bit more easily in the code. This is called upon from the start function, and first of all determines the player move by calling upon the move function below. Once that's sorted, it sets the computer move. It uses the random module's randint function to get an integer between one and three (1, 3). It then passes the player and computer move, stored as integers, onto the result function which we use to find the outcome.

07
```
def start():
    print "Let's play a game of Rock, Paper, Scissors."
    while game():
        pass
    scores()
```

08
```
def game():
    player = move()
    computer = random.randint(1, 3)
    result(player, computer)
    return play_again()
```

09
```
def move():
    while True:
        print
        player = raw_input("Rock = 1\nPaper = 2\nScissors = 3\nMake a move: ")
```

10
```
        try:
            player = int(player)
            if player in (1,2,3):
                return player
        except ValueError:
            pass
        print "Oops! I didn't understand that. Please enter 1, 2 or 3."
```



The code in action

09 We start the move function off by putting it into a while loop. The whole point of move is to obtain an integer between one and three from the player, so the while loop allows us to account for the player making an unsupported entry. Next, we are setting the player variable to be created from the player's input with raw_input. We've also printed instruction text to go along with it. The '\n' we've used in the text adds a line break; this way, the instructions appear as a list.

10 The try statement is used to clean up code and handle errors or other exceptions. We parse what the player entered by turning it into an integer using int(). We use the if statement to check if it is either 1, 2, or 3 – if it is, move returns this value back up to the game function. If it throws up a ValueError, we use except to do nothing. It prints an error message and the while loop starts again. This will happen until an acceptable move is made.

11 The result function only takes the variables player and computer for this task, which is why we set that in result(player, computer). We're starting off by having a countdown to the result. The printed numbers are self-explanatory, but we've also thrown in sleep from the time module we imported. Sleep pauses the execution of the code by the number of seconds in the brackets. In this example, we've put a one-second pause between counts, then half a second after that to show the results.

12 To print out what the computer threw, we're using string.format(). The {0} in the printed text is where we're inserting the move, which we have previously defined as numbers. Using names[computer], we're telling the code to look up what the text version of the move is called from the names we set earlier on, and then to insert that where {0} is.

13 Here we're simply calling the scores that we set earlier. Using the global function allows for the variable to be changed and used outside of the variable, especially after we've appended a number to one of the scores of either the player or computer.

```
11 def result(player, computer):
       print "1..."
       time.sleep(1)
       print "2..."
       time.sleep(1)
       print "3!"
       time.sleep(0.5)
12     print "Computer threw {0}!".format(names[computer])
13     global player_score, computer_score
14     if player == computer:
           print "Tie game."
15     else:
           if rules[player] == computer:
               print "Your victory has been assured."
               player_score += 1
16         else:
               print "The computer laughs as you realise you have been defeated."
               computer_score += 1
```

14 The way we're checking the result is basically through a process of elimination. Our first check is to see if the move the player and computer used were the same, which is the simplest part. We put it in an if statement so that if it's true, this particular section of the code ends here. It then prints our tie message and goes back to the game function for the next step.

15 If it's not a tie, we need to keep checking, as it could still be a win or a loss. Within the else, we start another if statement. Here, we use the rules list from earlier to see if the losing move to the player's move is the same as the computer's. If that is the case, we will print the message saying so, and add one to the player_score variable from before.

16 If we get to this point, the player has lost. We print the losing message, give the computer a point and it immediately ends the result function, returning to the game function.

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.3 (default, Sep 26 2012, 21:51:14)
[GCC 4.7.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Let's play a game of Rock, Paper, Scissors.

Rock = 1
Paper = 2
Scissors = 3
Make a move: 5
Oops! I didn't understand that. Please enter 1, 2 or 3.

Rock = 1
Paper = 2
Scissors = 3
Make a move: 1
1...
2...
3!
Computer threw Rock!
Tie game.
```

The code in action

17 The next section of game calls upon a play_again function. Like the move function, this requires human input, asking the player if they would like to play again via a text message with raw_input, with the simple 'y/n' response.

18 Giving users an option of y/n like we have should expect a response in kind. The if statement checks to see if any of our defined positive responses have been entered. As Python doesn't differentiate between upper or lower case, we've made sure that it accepts both y and Y. If this is the case, it returns a positive response to game, which will start it again. Entering n or N will return a negative response to the game and so it will not restart.

19 If we don't get an expected response – that is either a y or n – we will assume that the player does not want to play again. If this is the case, we will print a goodbye message, and that will end this function. This will also cause the game function to move onto the next section and not restart the game again. This ensures that the game does not run without an end point.

```
17 def play_again():
       answer = raw_input("Would you like to play again? y/n: ")
18     if answer in ("y", "Y", "yes", "Yes", "Of course!"):
           return answer
19     else:
           print "Thank you very much for playing our game. See you next time!"

20 def scores():
       global player_score, computer_score
       print "HIGH SCORES"
       print "Player: ", player_score
       print "Computer: ", computer_score

21 if __name__ == '__main__':
       start()
```

```
Python Shell
File  Edit  Shell  Debug  Options  Windows  Help
Python 2.7.3 (default, Sep 26 2012, 21:51:14)
[GCC 4.7.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Let's play a game of Rock, Paper, Scissors.

Rock = 1
Paper = 2
Scissors = 3
Make a move: 5
Oops! I didn't understand that. Please enter 1, 2 or 3.

Rock = 1
Paper = 2
Scissors = 3
Make a move: 1
1...
2...
3!
Computer threw Rock!
Tie game.
Would you like to play again? y/n: n
Thank you very much for playing our game. See you next time!
HIGH SCORES
Player:  0
Computer:  0
>>>
```

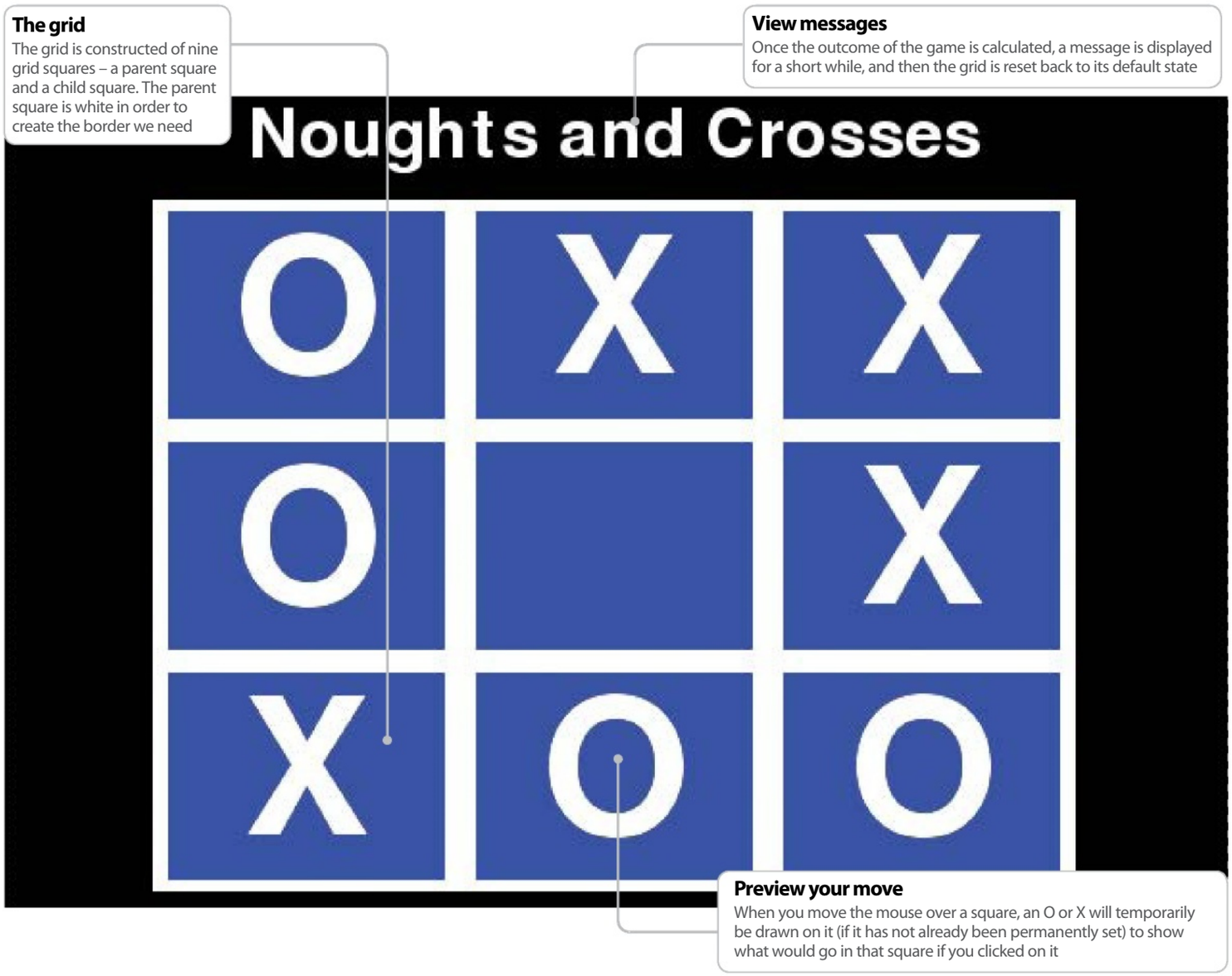
The code in action

ELIF

IF also has the ELIF (else if) operator, which can be used in place of the second IF statement we employed. It's usually used to keep code clean, but performs the same function.

20 Going back to the start function, after game finishes we move onto the results. This section calls the scores, which are integers, and then prints them individually after the names of the players. This is the end of the script, as far as the player is concerned. Currently, the code won't permanently save the scores, but you can have Python write it to a file to keep if you wish.

21 The final part allows for the script to be used in two ways. Firstly, we can execute it in the command line and it will work fine. Secondly, we can import this into another Python script, perhaps if you wanted to add it as a game to a collection. This way, it won't execute the code when being imported.



Build a game of noughts and crosses for the Raspberry Pi

We demonstrate how to make your own version of this classic game using the Pygame framework

Resources

Raspberry Pi with the necessary peripherals

Raspbian distro:
www.raspbian.org

Before we begin, download the latest Raspbian image from **www.raspberrypi.org/downloads**. Flash the image to your SD card as you usually would. Instructions can be found at **http://bit.ly/QDkoS5** if needed. You'll only need to go up to the step where you write the image to the SD card. You'll have to adapt the instructions slightly for using the newer Raspbian image rather than the Debian one.

Noughts and crosses is a very simple strategy game played across a 3x3 grid. It is often played by young children due to its simple nature, which makes it well suited to the Raspberry Pi. Since we have already covered how to create artificial intelligence players in a previous tutorial, this game will be played by two human players, with the focus instead being on introducing new things such as an algorithm to work out which player has three in a row. We'll also be working out the sizes of everything in proportion to the size of the display, rather than explicitly specifying their positions in pixels. Without further ado, let's get started.

01 Creating our project

Double-click on the IDLE (not IDLE 3) icon on the Raspberry Pi's desktop to open up our Python development environment. IDLE starts as a Python shell by default, so go to the File menu and select New Window to open an Editor Window. Once the Editor is open, select the File menu and click Save. We don't actually need to make a project folder for this project as the game will be done entirely in code and won't require any external resources such as image files. We called our file O&X.py. Click Save once done.

02 Starting with the basics

Begin with the usual line of '#!/usr/bin/python', which tells the shell to run the code in the file using Python. After that, write a summary of what your program will do. We'll also import the Pygame and system modules and import everything thing from the Pygame library, as well as a bunch of constants used by Pygame to make our lives easier. Keep saving by hitting Ctrl+S.

```
    #!/usr/bin/python
    # A simple, Noughts and Crosses game for
two human players implemented using
    # the PyGame framework. Created by Liam
Fraser for a Linux User and Developer
    # tutorial.
    import pygame # Provides the PyGame
framework
    import sys # Provides the sys.exit function
we use to exit our game loop
    from pygame import *
    from pygame.locals import * # Import
constants used by PyGame
```

03 Starting the Game class

The init function of the Game class is going to look the same as it usually does and is explained thoroughly by the comments you'll see in the following code.

We also keep track of who the current player is, so that we can change it. Notice that the reset function doesn't reset this value; this means that whoever goes first in the next game will be different from whoever went last in the previous one.

```
    # Our game class
    class OAndX:
        def __init__(self):
            # Initialize PyGame
            pygame.init()
            # Create a clock to manage the game
loop
            self.clock = time.Clock()
            # Set the window title
            display.set_caption("Noughts and
Crosses")
            # Create the window with a
resolution of 640 x 480
            self.displaySize = (640, 480)
            self.screen = display.set_
mode(self.displaySize)
            # Will either be O or X
            self.player = "O"
```

04 The Background class

Our Background class is very simple. We're going to create the surface and not fill it so that the background is black. After that, we write the text 'Noughts and Crosses' at the top of the window, horizontally centred. Below, we pick the font size based on the width of the display. The displaySize tuple contains two values: the width and the height, which are accessed with an index of 0 or 1 respectively. Notice the use of the Pygame Color function, which takes a string which is the name of a colour and returns the RGB value for that colour. The True value that's passed to the font renderer makes the text look smoother by enabling anti-aliasing.

```
 # The background class
    class Background:
        def __init__(self, displaySize):
            self.image = Surface(displaySize)
            # Draw a title
            # Create the font we'll use
            self.font = font.Font(None,
(displaySize[0] / 12))
            # Create the text surface
            self.text = self.font.
render("Noughts and Crosses", True,

(Color("white")))
            # Work out where to place the text
            self.textRect = self.text.get_
rect()
            self.textRect.centerx =
displaySize[0] / 2
            # Add a little margin
           self.textRect.top = displaySize[1] *
0.02
            # Blit the text to the background
image
            self.image.blit(self.text, self.
textRect)
        def draw(self, display):
            display.blit(self.image, (0, 0))
```

05 Grid squares

We're going to have a Grid class which makes up the playing area of the game. However, before we do that, we're going to make a class for each individual square in the grid. This is so we can know things such as the state of the square (either X or O) and if that state is permanent (or just while the mouse is hovering over the square). The GridSquare class inherits from sprite.Sprite, so that the squares can be added to a sprite group, which can be updated in one go.

The position is a tuple in the form of (column, row). The grid size is a tuple in the form (width, height).

```
    # A class for an individual grid square
    class GridSquare(sprite.Sprite):
        def __init__(self, position, gridSize):
            # Initialise the sprite base class
            super(GridSquare, self).__init__()
            # We want to know which row and
column we are in
            self.position = position

            # State can be "X", "O" or ""
            self.state = ""
            self.permanentState = False
            self.newState = ""
            # Work out the position and size of
the square
            width = gridSize[0] / 3
            height = gridSize[1] / 3
            # Get the x and y co ordinate of
the top left corner
            x = (position[0] * width) - width
            y = (position[1] * height) - height
```

06 Grid square surfaces

Each grid square is actually made up of two rectangles. The parent rectangle is white, while the child rectangle is blue and 90% of the size of the parent rectangle. This effectively gives us a blue rectangle with a white border. The new thing here is that the child rectangle image is actually drawn to the parent rectangle surface, which means that all positioning is only relative to the parent rectangle. As far as the child rectangle is concerned, the co-ordinate (0, 0) is the top-left corner of the parent rectangle. We finish off the grid square's initialiser by creating the font we'll use to display the O and X letters.

```
            # Create the image, the rect and
then position the rect
            self.image = Surface((width,
height))
            self.image.fill(Color("white"))
            self.rect = self.image.get_rect()
            self.rect.topleft = (x, y)
            # The rect we have is white, which
is the parent rect
            # We will draw another rect in the
middle so that we have
             # a white border but a blue
center
            self.childImage = Surface(((self.
rect.w * 0.9), (self.rect.h * 0.9)))
            self.childImage.fill(Color("blue"))
            self.childRect = self.childImage.
get_rect()
            self.childRect.center = ((width /
2), (height / 2))
            self.image.blit(self.childImage,
self.childRect)
            # Create the font we'll use to
display O and X
            self.font = font.Font(None, (self.
childRect.w))
```

07 Constructing a grid

Now we have the basic structure of our grid squares, we'll create a Grid class to hold them in. The grid is going to be 75% of the size of the screen, and centred. Once all of the positioning code is out of the way, we have a nested loop which makes three rows, each containing three grid squares. Each grid square instance is stored in the self.squares list. We finish the grid initialiser off by putting the sprites into a sprite group, so that they can be updated and drawn as a group.

```
    # A class for the 3x3 grid
    class Grid:
        def __init__(self, displaySize):
            self.image =
Surface(displaySize)
            # Make a number of grid squares
            gridSize = (displaySize[0] * 0.75,
                        displaySize[1] * 0.75)
            # Work out the co-ordinate of the
top left corner of the grid so that it can be
centered on the screen
            self.position = ((displaySize[0] /
2) - (gridSize[0] / 2),
    (displaySize[1] / 2) - (gridSize[1] / 2))
            # An empty array to hold our grid
squares in
            self.squares = []
            for row in range(1,4):
                # Loop to make 3 rows
                for column in range(1,4):
                    # Loop to make 3 columns
                    squarePosition = (column,
row)
                    self.squares.append(GridSqu
are(squarePosition, gridSize))
            # Get the squares in a sprite group
            self.sprites = sprite.Group()
            for square in self.squares:
                self.sprites.add(square)
```

08 The grid draw function

The draw function of the Grid class updates each grid square sprite, draws them to the grid's surface (self.image), then draws to the display at the position calculated in the initialiser function.

```
        def draw(self, display):
            self.sprites.update()
            self.sprites.draw(self.image)
            display.blit(self.image, self.
position)
```

09 Finish the GridSquare

Now that we have the main parts of our Grid class in place, we can carry on with the GridSquare class. It needs a couple of functions before it is complete. One is the update function, which is mandatory for any class that inherits the Pygame Sprite class, and the other is a simple function that sets the state of the grid square. The setState function sets the newState variable, which in turn means that the grid square will update with the correct state (X, O or blank) on the next clock tick. setState can also make the state permanent in the case that the user has clicked on the square.

```
        def update(self):
            # Need to update if we need to set
a new state
            if (self.state != self.newState):
                # Refill the childImage blue
                self.childImage.
fill(Color("blue"))
                text = self.font.render(self.
newState, True, (Color("white")))
                textRect = text.get_rect()
                textRect.center = ((self.
childRect.w / 2),
                                   (self.
childRect.h / 2))
                # We need to blit twice because
the new child image needs to be blitted to the
parent image
                self.childImage.blit(text,
textRect)
                self.image.blit(self.
childImage, self.childRect)

                # Reset the newState variable
                self.state = self.newState
                self.newState = ""
        def setState(self, newState,
permanent=False):
            if not self.permanentState:
                self.newState = newState
                if permanent:
                    self.permanentState = True
```

10 Extending the game class

Now we have our Background and Grid related classes, we should add them into the initialiser of the main OAndX class. However, as we want to be able to reset the board, we'll put them in a reset function, which recreates the Background and the Grid and can be called many times. Once we have this function, we simply need to call self.reset() from the initialiser of the OAndX class.

```
    def reset(self):
            # Create an instance of our
 background and grid class
            self.background =
Background(self.displaySize)
            self.grid = Grid(self.
displaySize)
```

11 Finishing off the grid class

We need to add a couple of helpful functions to the Grid class that will come in very useful when working out who has won the game, or if it's a draw (if the grid is full).

```
    def getSquareState(self, column, row):
            # Get the square with the requested
position
            for square in self.squares:
                if square.position == (column,
row):
                    return square.state
        def full(self):
            # Finds out if the grid is full
            count = 0
            for square in self.squares:
                if square.permanentState ==
True:
                    count += 1
            if count == 9:
                return True
            else:
                return False
```

12 Working out the winner

This is the part that gets a little tricky. We're going to add a function to the OAndX class called getWinner, which will either return nothing, the winning player (O or X), or 'draw'. We start by defining players: a list of things we need to try to find a row of. If we don't find a winner by the time we've checked every possibility, we check if the grid is full, in which case we return 'draw'.

For each player, we go through up to four separate sets of loops (one for each possible direction). The possible ways of achieving three in a row are horizontally, vertically or diagonally, in both forward and reverse directions.

For each direction, a pair of loops check through each grid square and get the next two grid squares away from the current grid square in the direction that we are checking for. If a square doesn't exist, then nothing is returned by the getSquareState function. If all three squares have the same state, then the winning player is returned, which stops the function.

```
        def getWinner(self):
            players = ["X", "O"]

            for player in players:
                # check horizontal spaces
                for column in range (1, 4):
                    for row in range (1, 4):
                        square1 = self.grid.
getSquareState(column, row)
                        square2 = self.grid.
getSquareState((column + 1), row)
                        square3 = self.grid.
getSquareState((column + 2), row)
                        # Get the player of the
square (either O or X)
                        if (square1 == player)
and (square2 == player) and (square3 == player):
                            return player
                # check vertical spaces
                for column in range (1, 4):
                    for row in range (1, 4):
                        square1 = self.grid.
getSquareState(column, row)
```

```
                        square2 = self.grid.
getSquareState(column, (row + 1))
                        square3 = self.grid.
getSquareState(column, (row + 2))
                        # Get the player of the
square (either O or X)
                        if (square1 == player)
and (square2 == player) and (square3 == player):
                                return player
                # check forwards diagonal spaces
                 for column in range (1, 4):
                     for row in range (1, 4):
                        square1 = self.grid.
getSquareState(column, row)
                        square2 = self.grid.
getSquareState((column + 1), (row - 1))
                        square3 = self.grid.
getSquareState((column + 2), (row - 2))
                        # Get the player of the
square (either O or X)
                        if (square1 == player)
and (square2 == player) and (square3 == player):
                                return player
                # check backwards diagonal spaces
                  for column in range (1, 4):
                      for row in range (1, 4):
                     square1 = self.grid.
getSquareState(column, row)
                        square2 = self.grid.
getSquareState((column + 1), (row + 1))
                        square3 = self.grid.
getSquareState((column + 2), (row + 2))
                        # Get the player of the
square (either O or X)
                        if (square1 == player)
and (square2 == player) and (square3 == player):
                                return player
            # Check if grid is full if someone
hasn't won already
            if self.grid.full():
                return "draw"
```

13 Displaying a message when a winner is found

We want to display a message when a winner is found, and make it separate from the grid. The best way is to completely black out the screen and then write to the blank screen. The text is rendered straight to the screen. Notice how the display.update() function needs to be called for anything to go on the screen, as the time.wait function will stop the execution of everything, including the game loop we're going to write in a moment. The winner message ends by resetting the game to its initial state with a fresh grid.

```
        def winMessage(self, winner):
            # Display message then reset the
game to its initial state
            # Blank out the screen
            self.screen.fill(Color("Black"))
            # Create the font we'll use
            textFont = font.Font(None, (self.
displaySize[0] / 6))
            textString = ""
            if winner == "draw":
                textString = "It was a draw!"
            else:
                textString = winner + " Wins!"
            # Create the text surface
            text = textFont.render(textString,
True,
    (Color("white")))
            textRect = text.get_rect()
            textRect.centerx = self.
displaySize[0] / 2
            textRect.centery = self.
displaySize[1] / 2
            # Blit changes and update the
display before we sleep
            self.screen.blit(text, textRect)
            display.update()
            # time.wait comes from pygame libs
            time.wait(2000)
            # Set game to its initial  state
            self.reset()
```

14 The game loop

The game loop for this game is very easy, as all of the code is triggered by something else, namely the handleEvents function that we can begin to write, because we'll have all the necessary pieces in place once we finish the game loop. The game can quite happily run at 10fps because there is hardly any animation, simply text being drawn on-screen as the mouse moves between squares.

```
    def run(self):
       while True:
           # Our Game loop
           # Handle events
           self.handleEvents()
           # Draw our background and grid
           self.background.draw(self. screen)
           self.grid.draw(self.screen)
           # Update our display
           display.update()
           # Limit the game to 10 fps
           self.clock.tick(10)
```

15 Handling events

Now we tie everything together. The only two events important to us are the click event and mouse button up event, so that we know when someone has clicked on a square, indicating they want to permanently set the state of that square.

We need to get the co-ordinates of the mouse pointer, which are relative to the top left of the screen, but we can make them relative to the top left of the grid by subtracting the co-ordinate of the grid's top-left corner. Once we have the co-ordinate, we can loop through each square and work out which square the mouse is over using the rect.collidepoint function, which returns True if the co-ordinates that have been passed to it as a parameter are located in the rectangle.

If the mouse is clicked then we want to set the state of the square, passing through the current player and True, indicating the change should be permanent. We then go to the next player and call self.getWinner. If the getWinner function returns anything that isn't null (ie O, X, or draw), the value is passed to winMessage which displays a message with whoever won and resets the grid.

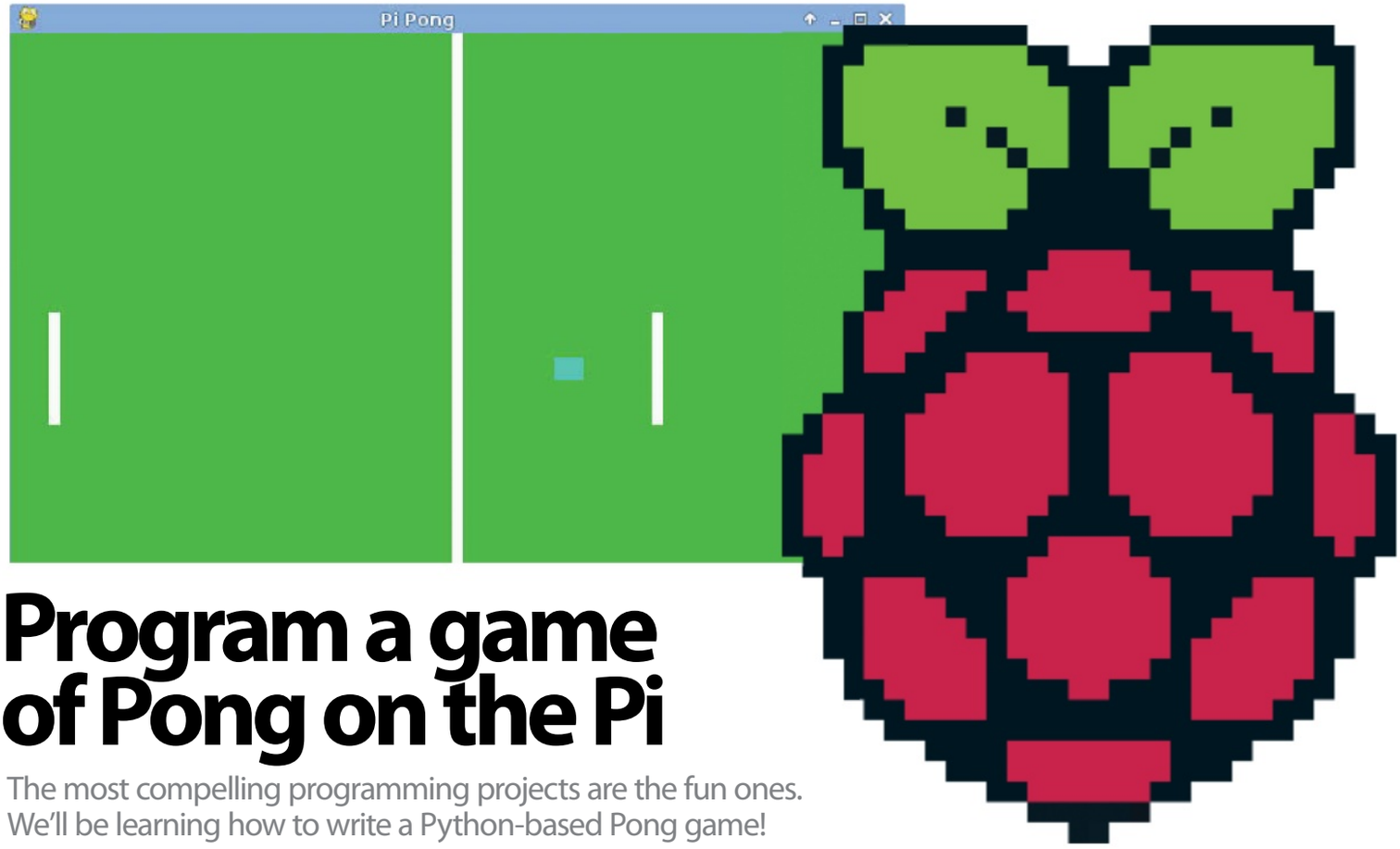
If the mouse isn't clicked, then the grid will change state to the current player, but will go blank again once the mouse moves away.

```
        def handleEvents(self):
            # We need to know if the mouse has
been clicked later on
            mouseClicked = False
            # Handle events, starting with quit
            for event in pygame.event.get():
                if event.type == QUIT:
                    pygame.quit()
                    sys.exit()
                if event.type == MOUSEBUTTONUP:
                    mouseClicked = True
            # Get the co ordinate of the mouse
            mousex, mousey = mouse.get_pos()
            # These are relative to the top
left of the screen,and we need to make them
relative to  the top left of the grid
            mousex -= self.grid.position[0]
            mousey -= self.grid.position[1]
            # Find which rect the mouse is in
            for square in self.grid.squares:
                if square.rect.
collidepoint((mousex, mousey)):
                    if mouseClicked:
                        square.setState(self.
player, True)
                        # Change to next player
                        if self.player == "O":
                            self.player = "X"
                        else:
                            self.player = "O"
                        # Check for a winner
                        winner = self.
getWinner()
                        if winner:
                        self.winMessage(winner)
                    else:
                   square.setState(self.player)
                else:
                    # Set it to blank, only if
permanentState == False
                    square.setState("")
```

16 The final piece

The final thing to do before our project will run is to add a couple of lines right at the end of the file that will create an instance of the Game class and then call the run function to start the game loop.

```
if __name__ == '__main__':
    game = OAndX()
    game.run()
```



Program a game of Pong on the Pi

The most compelling programming projects are the fun ones. We'll be learning how to write a Python-based Pong game!

We're going to be remaking the classic game Pong. To do this, we'll be using a Python module called Pygame, which is great, as it allows the programmer to create 2D games without having to worry about things such as rendering the graphics in too much detail. The main portion of the code will be that which makes up the game's structure and logic.

Pong is an ideal game through which to introduce the principles behind game development, as it is fairly simple, which makes it much easier to understand what's going on. This also means that it requires less code, making it ideal for those just starting out in programming.

Getting started

01 Create a file

The first step, of course, is to start up our development environment, which is the Geany editor, and create a new Python file using the 'with template' option. We only need the first line and the bottom couple of lines – delete the others so you only have the lines shown in Fig 1.

Fig 1

```
1  #!/usr/bin/env python
2
3  if __name__ == '__main__':
4      main()
```

02 Add comments

As usual, we're going to add some comments to the start of our program to help out anyone (including ourselves) who might end up using it. See Fig 2 for what we wrote.

03 Save the file

You will want to save your file inside a new folder for the project, which we are rather cleverly naming PiPong. You should save your file with the name 'PiPong.py'.

The standard stuff

Pretty much every game in Pygame starts off in a similar way: by importing the required modules and then having a game class with an initialisation method that is called when an instance of the class is created. This initialisation method is where we set everything up. Don't worry if this doesn't seem clear at the moment – you will see how it works in a moment.

01 Import modules

We need to start off by importing some modules after we have written our initial comments. It's good style to have each import on a separate line, because you can follow the import with a comment explaining what the library is used for. Import the modules as shown in Fig 3.

02 Import constants

After the imports, leave a blank line and then write the line 'from Pygame.locals import *'. This essentially means that everything is imported from the Pygame locals module.

This enables us to use constants (variables that can't be changed) from the Pygame module. An example of a constant would be the KEYDOWN

Fig 2

```
3  # PiPong - A remake of the classic Pong game using PyGame. Written by
4  # Liam Fraser - 28/07/2012 for the Linux User & Developer magazine.
```

Fig 3

```
6  import pygame # Provides what we need to make a game
7  import sys # Gives us the sys.exit function to close our program
8  import random # Can generate random positions for the pong ball
```

Member variables

This tutorial will be much easier once you understand member variables, so let's take this opportunity to learn about them.

A variable called 'self' is usually passed to every function in a class. The self variable is a reference to a particular instance of the class and allows the programmer to access member variables and methods from that particular instance. This is passed to all instance methods within the class. For example, we make the display size a member variable so that it can be accessed from every function in the class.

```
# Our main game class
class PiPong:

    def __init__(self):

        # Make the display size a member of the class
        self.displaySize = (640, 400)
```

■ Try and get to grips with member variables in order to make the process easier

event, which is triggered whenever a key is pressed. By importing the constants, we can use them in code as just that, rather than having to refer to them as Pygame.locals.KEYDOWN. This makes our code tidier, and easier to read.

We're going import the whole Pygame module too, so do that using the code 'from Pygame import *'. This just saves us having to write 'Pygame.' in front of a whole bunch of things.

03 Create game class

It's now time to create our game class. A class is defined very simply using the syntax 'class PiPong:'. Now that we have a class, it's time to move onto the initialiser function.

Before we begin

There has been a lot of activity in the Raspberry Pi community recently. As a result of this, there is now a new operating system for the Raspberry Pi named Raspbian. As you might have guessed, this is just a Raspberry Pi-specific version of Debian, which has been used in previous tutorials.

The advantage to Raspbian is that it really makes the most of all of the hardware available on the Raspberry Pi, offering a significant performance increase over the previous Debian images. The disadvantage of the Raspbian distribution is that the current image (2012-07-15-wheezy-raspbian.zip) does not have all of the packages that we require for this tutorial by default. A code editor such as Geany is missing (although there are command-line alternatives included).

It does not actually matter if you are still using an old Debian image or not though, as the programs that we are using will work in exactly the same way. Not to worry if you are using the new Raspbian image though, because all you need to do is simply enter the command 'sudo apt-get update' followed by 'sudo apt-get install geany python-Pygame'.

If there are any problems with this, then we recommend that you go back to the last Debian 6 image, which you can find by going to **http://downloads.raspberrypi.org/images/debian/6/debian6-19-04-2012**.

The game class initialiser

In the initialisation function of the game class, we will do a number of things, such as initialise the Pygame module, create a clock to manage time and speed in the game, create a window and many more things.

The background class

The background class is pretty simple: all it has to do is create a surface, fill it with a green colour and then draw a white line down the middle in proportion to the display size. This means that the code itself is also pretty simple, so we're only going to explain the less obvious parts.

Display sizes
displaySize[0] is the width component of the display and displaySize[1] is the height component of the display

Rectangle
Here, we are creating a rectangle with x and y co-ordinates (0, 0), which is the top-left corner. We are also passing the width and height following this

Colour
Here we are saying, draw the rectangle to the background surface, in the colour white (255,255,255 in RGB colour code)

Render
Blit stands for block image transfer. This function will allow us to render the background surface to the display which is passed in during the game loop, which we'll get onto in a moment

```
# The class for the background
class Background:

    def __init__(self, displaySize):

        # Set our image to a new surface, the size of the screen rectangle
        self.image = Surface(displaySize)

        # Fill the image with a green colour (specified as R,G,B)
        self.image.fill((27, 210, 57))

        # Get width proportionate to display size
        lineWidth = displaySize[0] / 80

        # Create a rectangle to make the white line
        lineRect = Rect(0, 0, lineWidth, displaySize[1])
        lineRect.centerx  = displaySize[0] / 2
        draw.rect(self.image, (255, 255, 255), lineRect)

    def draw(self, display):

        # Draw the background to the display that has been passed in
        display.blit(self.image, (0,0))
```

01 Coding the class initialiser

The first thing you need to do is copy the code from the 'Member variables' boxout (page 157), because that's the start of our initialiser class. The class initialiser function is always defined with the syntax 'def __init__():' where any parameters to be passed are put inside the brackets. Note that the lines before and after are made up of two underscores. The next thing to do is initialise Pygame, by calling Pygame.init(). We could just call init() because we have imported the Pygame module, but it looks a bit odd and it's clearer to be explicit in what we are calling things.

02 Create a clock

We now need to create a clock to manage the time in the game. The clock allows us to specify the number of frames per second the game should run at, among other useful things. A clock is formed by creating a new instance of the Pygame.time.Clock class in the following way: self.clock = Pygame.time.Clock(). Another bit of housekeeping we need to do is to set the window title using the syntax: display.set_caption("Pi Pong").

03 Create a window

For now, the final part of the class initialiser is creating the window, which we do using the syntax: self.display = display.set_mode(self.displaySize). The display size, which we've set to 640x480, was set earlier on in the initialisation function. We've chosen 640x480 as the resolution for the game as that should be pretty much suitable for everyone. We'll have to come back to the class initialiser later on, to do things such as creating the background, the bats, ball and so on.

The game loop

Almost every game works using a loop; a continuous piece of code that repeats itself. A typical game loop may take the following structure: handle any input events update any objects accordingly, draw the objects to the display surface, update the display, wait until it is time for the next frame. Interestingly, if the game doesn't wait a certain amount of time before looping again, it will hog the CPU by trying to use it all of the time because the loop will start again as soon as it is finished. This is where the clock object is useful.

Before we start the game loop, it makes sense to go back to the initialiser function of the PiPong class so we can create a background. After the line where the display is created, you need to create an instance of the background class, passing through the display size, by typing 'self.background = Background(self.displaySize)'. Now we can move on to the game loop, because we'll actually have something to render on the display.

Understanding co-ordinates

Understanding co-ordinates and axes is crucial to game development – so let's take a little recap.

The co-ordinate system in 2D game development is usually always the same. The x axis goes from left to right, with the far left value being 0. The y axis goes from bottom to top. However, the highest point at the top has the value 0, which is different to if you were drawing a graph. Co-ordinates are written in the form (x, y). The top-left corner of the screen has the value (0, 0).

After the line where you have created the background, we need to make a new function called run, which is where the code for the game loop will go. Leave a blank line and be sure to move the indentation back a level. Define the function like you would any other, passing self through as usual, with the syntax 'def run(self):'. We then need to start the while loop with 'while True:'. Using the constant True means that this is an infinite loop and will run until the program exits. The while loop code is self-explanatory, so it should be absolutely fine for you to copy it in.

The first run

We are now at a stage where we can finally get some of our first Pygame code running.

Now all we have to do to get this to run is to change the main code from the very start, which is how the program starts. The main code needs to look like this:

```
if __name__ == '__main__':
game = PiPong()
game.run(
```

This code will call the init function of the PiPong class when the new instance is created, and then call the run function to start the game loop. You can press the button with the three cogs on it to run the game. Note that we haven't handled the quit event yet, so you'll have to use the red stop button to stop the game.

"To get this to run... change the main code from the very start"

Creating the bats

The bat class we're going to make will inherit Pygame's Sprite class. Doing this allows us to put them into a sprite (a graphical object) group and makes them easier to manage in the game loop. To inherit the Pygame Sprite class, we must define our bat class in the following way: class Bat(sprite.Sprite):. We prefix the Sprite class with 'sprite.' because the Sprite class is part of Pygame's sprite module.

The initialisation function for the bat class takes two extra parameters: the display size, and the player (either player 1 or player 2). The display size is passed so that the bat can work out its width and height in proportion to the display size. The most important thing to understand about the initialisation function is the super command, which allows us to call methods and access variables on the class we have inherited. To initialise the instance of sprite.Sprite, we use: super(Bat, self).__init__(), where Bat is the class inheriting the base class. As usual, most of the initialisation function is simple, as it is just setting off starting values.

```
def startMove(self, direction):

    # Set the moving flag to true
    self.direction = direction
    self.moving = True

def update(self):

    if self.moving:
        # Move the bat up or down if moving
        if self.direction == "up":
            self.rect.centery -= self.speed
        elif self.direction == "down":
            self.rect.centery += self.speed

def stopMove(self):

    self.moving = False
```

■ Fig 4: Add this code to the bat class to make the bat move when a key is pressed

The sprite update function

Now it's time to learn about how basic animation works.

The update function for every sprite instance, is run on each frame, which in the case of this game is 30 frames a second. This means that we render 30 new images for each second that the game runs. As you can see in the bat movement code, if we are moving up, we subtract the speed value from the y value of the bat. If we are moving down, we add the speed value to the y value of the bat. The speed value is set to 13 in the class initialisation function, so the bat moves 13 pixels per frame. This gives us a fairly smooth animation.

> "The initialisation function for the bat class takes two extra parameters: the display size, and the player (either player 1 or player 2)"

Now that we've had a recap about co-ordinates, there shouldn't be anything difficult to understand in the bat class as long as you realise how the location is set depending on the player. Co-ordinates on the left side start at zero, so the display width divided by 20 can be the x co-ordinate of the left side. However, for the right side, we need to mirror the same effect, so we effectively subtract the x co-ordinate of the left side from the entire display width. Also note that the speed value of 13 is just something that we worked out using trial and error. Here is the code for the init function of the bat class:

```
[
def __init__(self, displaySize, player):

# Initialize the sprite base class
super(Bat, self).__init__()
# Make player a member variable
self.player = playe

# Get a width and height values
proportionate
to the display size
width = displaySize[0] / 80
height = displaySize[1] / 6

# Create an image for the sprite using the
width and height
# we just worked out
self.image = Surface((width, height))

# Fill the image white
self.image.fill((255, 255, 255))

# Create the sprites rectangle from the    image
self.rect = self.image.get_rect()

# Set the rectangle's location depending on  the
player if player == "player1":
# Left side
self.rect.centerx = displaySize[0] / 20
elif player == "player2":
# Right side
self.rect.centerx = displaySize[0] -
displaySize[0] / 20
# Center the rectangle vertically
self.rect.centery = displaySize[1] / 2
# Set a bunch of direction and moving  variables

self.moving = False
self.direction = "none"
self.speed = 13
]
```

Now, we need to make a function that will start the bat moving. Without this function, you'd have to repeatedly tap the arrow keys to keep the bat moving. This is because when a key is pressed down, the key press event is only called once. So instead of moving while a key is held down, we start moving when the key is pressed and then stop moving when the key is raised again. We'll get onto this shortly, but for now you need to add the code from Fig 4 into your bat class.

Adding the bats to the game

Now that we have a bat class, we need to go back up to the PiPong initialiser and create two bats. We also want to create a sprite group to put them in, so that we can call the update function on every sprite in the group easily. After the line where we create the background, you need the code from Fig 5 below.

We pass "player1" and "player2" through to the respective bats so they know which side to position themselves on. Now all that we have to do to be able to see our bats is add a couple of lines to the game loop, but before we do that we might as well write the event handling code so that we can move the bats with the keyboard.

Event handling

You need to add a function called handleEvents to the PiPong class. In this code, we loop through every event and then check if it's a certain type. If it's a QUIT event then we quit Pygame and exit the program using sys.exit(). If it's a key down event then we find out which key it is and start the bat moving in the appropriate direction. Notice that

■ Fig 5

```
# Create two bats and add them to a sprite group
self.player1Bat = Bat(self.displaySize, "player1")
self.player2Bat = Bat(self.displaySize, "player2")
self.sprites = sprite.Group(self.player1Bat, self.player2Bat)
```

"The most complicated part of this game is the ball class, because it involves collisions and then bouncing off things"

we have two if/else if statements – one for each bat. If it's a key up event then we will need to find out exactly which bat's keys were raised and subsequently stop that bat from moving. The code for this is here:

```
[
def handleEvents(self):

# Handle events, starting with the quit    event
for event in Pygame.event.get():
if event.type == QUIT:
Pygame.quit()
sys.exit()

if event.type == KEYDOWN:
# Find which key was pressed and start    moving
appropriate bat
if event.key == K_s:
# Start moving bat
self.player1Bat.startMove("down")
elif event.key == K_w:
self.player1Bat.startMove("up")

if event.key == K_DOWN:
self.player2Bat.startMove("down")
elif event.key == K_UP:
self.player2Bat.startMove("up")
if event.type == KEYUP:

if event.key == K_s or event.key == K_w:
self.player1Bat.stopMove()
elif event.key == K_DOWN or event.key
== K_UP:
self.player2Bat.stopMove()
]
```

Back to the game loop

We now need to go back to the game loop. The first thing we need to do as part of our game loop is handle any events. We do this by calling the self. handleEvents() function.

We then need to add some code to update and draw the sprite group that our two bats are in. This should ideally go after the code to draw the background. Once you have done this, the start of your main loop should look exactly like it does in Fig 6 – alongside.

Now that we've done that, you'll be able to run your game and move the bats around with the W and S keys for player1, and the UP and DOWN arrows for player2. You'll also be able to close the game with the X button in the top corner because we handle the quit event in the handleEvents function we wrote previously.

Fig 6

```
# Handle Events
self.handleEvents()

# Draw the background
self.background.draw(self.display)

# Update and draw the sprites
self.sprites.update()
self.sprites.draw(self.display)
```

The ball class

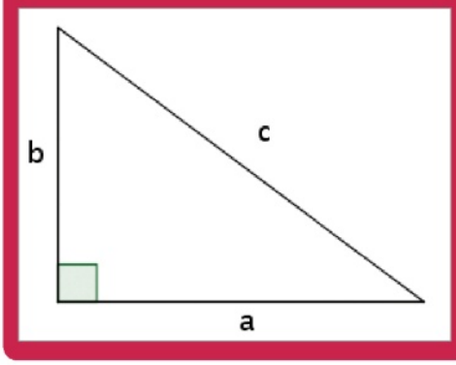
The most complicated part of this game is the ball class, because it involves collisions and then bouncing off things. Don't worry, though; the physics of the ball isn't 100 per cent accurate, so it i s much easier to understand. This is just supposed to be a very basic implementation; not a polished product.

The initialisation function of the ball follows the same structure as the one from the bat class. It starts by calling the super function to initialise the base sprite class. The display size passed in during initialisation is stored as a member variable, as it will be needed to check collisions with the sides later on. The ball class starts off like this:

```
[
# The class for the ball
class Ball(sprite.Sprite):

def __init__(self, displaySize):

# Initialize the sprite base class
super(Ball, self).__init__()

# Get the display size for working out
collisions later self.displaySize =
displaySize
# Get a width and height values
proportionate to the display size
width = displaySize[0] / 30
height = displaySize[1] / 30

# Create an image for the sprite
self.image = Surface((width, height))

# Fill the image blue
```

Vectors

We represent the speed and direction of the ball with a variable called vector, which is essentially made up of an x and y value. The size of these x and y values decide the angle and speed of the ball. It helps to think of the direction as the hypotenuse (c) of a triangle. To move in the direction of c, we have to move a (x) and b (y) by a certain number at the same time.




```
self.image.fill((27, 224, 198))

# Create the sprites rectangle from the    image

self.rect = self.image.get_rect()

# Work out a speed
self.speed = displaySize[0] / 110
# Reset the ball
self.reset()
]
```

Take a look at the last line of code. We're going to make a reset function for the ball that places it in the centre of the screen and then starts it moving either left or right, deciding the direction with a random number generator. This is because we'll need to reset the ball every time that it hits a wall behind the bat.

Now that we have learnt a little about vectors, let's take a look at the code for the ball's reset function – Fig 7.

The random.randrange function using the range 1 to 3 will either give the number 1 or 2. If the number is 1 then we'll set the x vector to -1, which will start moving the ball to the left. Otherwise, we'll set it to +1 which will start moving it to the right.

The ball's update function

As part of the ball's update function, we are going to check if the ball has hit any of the walls. Remember that the edge of the left side has the x co-ordinate zero, and the edge of the top side has the y co-ordinate zero. Also remember that the x

```
def reset(self):

    # Start the ball directly in the centre of the screen
    self.rect.centerx = self.displaySize[0] / 2
    self.rect.centery = self.displaySize[1] / 2

    # Start the ball moving to the left or right (pick randomly)
    # Vector(x, y)
    if random.randrange(1, 3) == 1:
        # move to left
        self.vector = (-1, 0)
    else:
        # move to right
        self.vector = (1, 0)
```

■ Fig 7: When the ball is reset, this code puts it in the middle of the screen and moves it randomly left or right

co-ordinate is the first element and accessed with a [0], and the y co-ordinate is the second element and accessed with a [1]. If we hit the left or right side, then we reset the ball to the centre. Otherwise, we call the reflectVector function to make it bounce.

The other part of the update function makes the ball move in the direction of the vector by simply adding the vector, multiplied by the ball speed, to the current x and y values. Notice that if the vector is a negative then the value will be subtracted,

```
def update(self):

    # Check if the ball has hit a wall
    if self.rect.midtop[1] <= 0:
        # Hit top side
        self.reflectVector()
    elif self.rect.midleft[0] <= 0:
        # Hit left side
        self.reset()
    elif self.rect.midright[0] >= self.displaySize[0]:
        self.reset()
    elif self.rect.midbottom[1] >= self.displaySize[1]:
        # Hit bottom side
        self.reflectVector()

    # Move in the direction of the vector
    self.rect.centerx += (self.vector[0] * self.speed)
    self.rect.centery += (self.vector[1] * self.speed)
```

■ Fig 8: The ball's update function

because adding a negative becomes subtraction. This isn't perfect, because the ball sometimes moves slower if the vector is small. However, this is just a basic implementation to explain the concepts. The ball's update function is shown in Fig 8.

The final steps

01 Create a ball

We now have to create a ball in the PiPong initializer, by creating an instance of the ball class, and passing through the display size. We also need to add it to the sprite group. Once you have done this, the end of your init function should function.

```
# Check for bat collisions
self.ball.batCollisionTest(self.player1Bat)
self.ball.batCollisionTest(self.player2Bat)
```

■ Fig 9

02 And finally

We need to go into the game loop and call the bat collision test function for each bat. Add this code (Fig 9) after the code to update and draw the sprites, and before the code to update the display.

Congratulations, you have just finished your first game on the Raspberry Pi. However, it can still be improved, so why not take a look at our frequently asked questions (right) before moving on to a different article?

Expanding the basics

Question:
How can I replace a colour of a surface that we have with an image?

Answer:
This is very simple. Instead of making self.image a surface, we use the following code for it: self.image = Pygame.image.load(background_file_name).convert().

Q How can I change the colour of the surfaces we have?

A We used **www.colorpicker.com** to get the RGB values for the colour we wanted. Click on a colour and replace the numbers in the self.image.fill code, which looks like this: self.image.fill((27, 210, 57)).

Q How can I add a score counter to the game?

A You would have to have a score class that kept count of the score, had a function to update the score, a function to reset the score, and a draw function. The draw function would use Pygame's font module.
The documentation for this, along with many other things, can be found over at: **www.Pygame.org/docs/ref/**.

Collisions and bouncing off surfaces

We are going to add in a very basic function called reflectVector, which will invert the y component of vector, by putting a negative sign in front of it. Think of it as having a vertical mirror where the ball bounces so the angle is reflected in the mirror. The reflectVector function looks like the screen to the right. All it is doing is getting the current x value, getting the current y value and making it negative, and then setting the same x and new inverted y value to become the new vector of the ball.

We are also going to make a function called batCollisionTest, which will be called twice from the game loop, passing a different bat in each time. This uses the Rect.colliderect function of Pygame, which takes two rectangles as parameters and returns True if they collide. If they do collide, then we need to work out the difference between the point at the centre of the bat and the centre of the ball, and then set the ball's new direction. The code for batCollisionTest will look like this:

```
def batCollisionTest(self, bat):

    # Check if the ball has had a collision with the bat
    if Rect.colliderect(bat.rect, self.rect):

        # Work out the difference between the start and end points
        deltaX = self.rect.centerx - bat.rect.centerx
        deltaY = self.rect.centery - bat.rect.centery

        # Make the values smaller so it's not too fast
        deltaX = deltaX / 12
        deltaY = deltaY / 12

        # Set the balls new direction
        self.vector = (deltaX, deltaY)
```

Add sound and AI to Pi Pong

Now we are going to build upon the Pong game that we just created in the earlier tutorial. We'll polish it off by adding a menu, sound effects and an artificial intelligence player

The skills that you will learn in this tutorial follow on from the previous one on programming a game on the Raspberry Pi, where we showed you how you can write a remake of the classic Pong game. Now that you are probably a little bit more comfortable with the way that game programming works, and Python in general, we are able to extend our original Pong remake by adding a menu, artificial intelligence and finally some interesting sound effects! So let's get started and have a go.

Before we begin

The Raspbian operating system for the Raspberry Pi has now become the recommended distribution to use. It comes with Pygame installed by default, reliable sound output and a Python development environment called IDLE.

As mentioned elsewhere in this book, the other enormous benefit to Raspbian is that it takes advantage of all of the hardware that is available on the Raspberry Pi, offering a significant performance increase over the previous Debian images.

Unfortunately, Geany is missing from the image, but we will walk you through how to use IDLE as it is just as good.

You are able to download the latest Raspbian image from **www.raspberrypi.org/downloads**. From here, flash the image to your SD card as you usually would. Instructions can be found at **www.linuxuser.co.uk/tutorials/how-to-set-up-raspberry-pi** if you are unsure on this. You'll only need to go up to the step where you write the image to the SD card.

01 Getting the original Pong remake code and sound effects

Double-click on the LXTerminal icon to open up a terminal. Use the command 'mkdir PiPong' to create a directory for the project. Then type 'cd PiPong' to change into the PiPong directory. Download the PiPong.zip file using the command 'wget liamfraser.co.uk/lud/PiPong.zip'. Unzip the zip file using the command 'unzip PiPong.zip'.

You can now verify that the files have been extracted using the ls command. Finally, remove the PiPong.zip file using the command 'rm PiPong.zip'. You can now close the terminal.

02 Introducing IDLE

IDLE stands for Integrated Development Environment and has actually been in existence since 1998. It is for Python only and is therefore more specific and capable than Geany once you get used to using the interface. You can start it up by double-clicking on the IDLE icon on the desktop (note that we are using IDLE 2, not IDLE 3). You will be greeted with a Python shell once IDLE has started (see Fig 1).

03 Opening the PiPong.py file

Go to the File menu and then select the 'Open Module' option. Navigate into the PiPong folder by double-clicking on it and then double-click on the PiPong.py file. The code will load up in an editor window with syntax highlighting very similar to Geany's. You can run the code using F5 should you want to and can save as normal using Ctrl+S. When you run the code, the Python shell will move to the front and print a line that looks like this:

========== RESTART ==========

You may also notice that once you close your program, it prints a red traceback message containing the last called functions. As long as the last called function was sys.exit(), your program exited cleanly.

```
# The class for our main menu
class Menu():
```

04 Adding a Menu class

Our menu's main purpose is to allow the user to pick if they would like to play against an artificial intelligence second player, or with two human players. It also means that the game doesn't start straight away. We're going to start our menu off by creating a Menu class at the bottom of the existing code, just above the following line: 'if __name__ == '__main__':'.

> "We are able to extend our original Pong remake by adding a menu, artificial intelligence and some sound effects"

05 Build a menu

In Pygame, a menu has to be drawn in the game loop, just like everything else. The game loop will check for a variable in the Menu class to decide if it should be drawing the menu or not, and then do the appropriate action. We'll create a class initializer, which creates everything needed. The draw function later on simply draws what we create in here to the screen. Passing 'None' to the font constructor just means use the default Pygame font. When creating the text surface, the True value means use font smoothing; the final parameter is the colour to render the font specified as (Red, Green, Blue).

```
def __init__(self, displaySize):
# Should we be drawing a menu?
self.active = True
# Create the font we'll use
self.font = font.Font(None, 30)
# Create the text surface
self.text = self.font.render(
"Press 1 for a CPU Opponent or 2 for "
"two Human players", True, (255,0,0))
# Get the text rectangle and center it
self.textRect = self.text.get_rect()
self.textRect.centerx = displaySize[0] / 2
self.textRect.centery = displaySize[1] / 2
```

06 Draw the menu and handling events

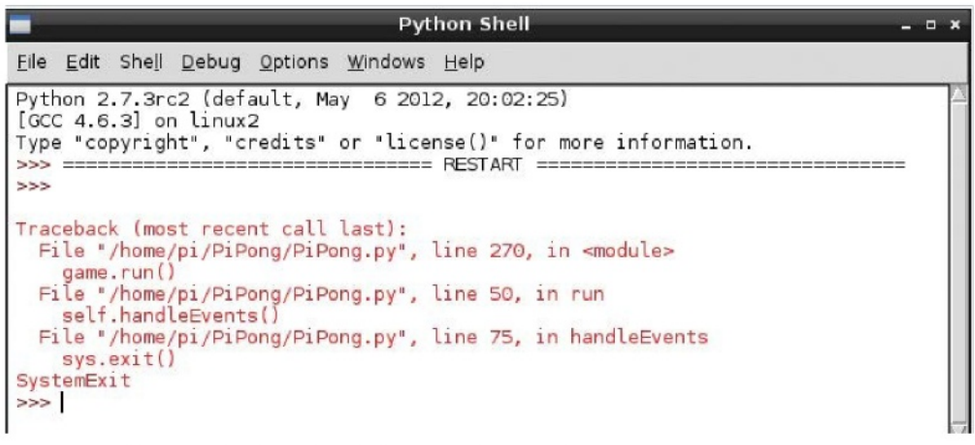
Drawing the menu is easy. We already have it, we just need to draw it to the screen using the blit function (blit stands for block image transfer) at the location of the rectangle that we created before. We'll also need to handle the 1 and 2 key events that will eventually set which type of game to play. For now, the 1 and 2 keys will just start the same type of game.

```
def draw(self, display):
# Draws our menu to the screen
display.blit(self.text, self.
textRect)
def handleEvents(self):
# Handle events, starting with
the quit event for event in
pygame.event.get():
if event.type == QUIT:
pygame.quit()
sys.exit()
if event.type == KEYDOWN:
if event.key == K_1:
# Disable the menu
self.active = False
elif event.key == K_2:
self.active = False
```

07 Add the menu to the existing game

We need to head up to the top of the code now and then add the following line to the bottom of the PiPong initializer function: 'self.menu = Menu(self.displaySize)'.

Scroll down a little bit to the very beginning of the run function. This is where we need to add an if statement and then either run a normal game or display a menu. We have moved the code in order to draw the background outside of



■ Fig 1: You will be greeted with a Python shell once IDLE has started

"We're going to make a Sound class that will make things nice and easy for us to play sound from anywhere in the game"

the if statement so that there is a background in both cases.

```
def run(self):
# Runs the game loop
while True:
# The code here runs when every frame is
drawn
# Draw the background
self.background.draw(self.display)
if self.menu.active:
# Draw our menu and handle keys 1 + 2
self.menu.draw(self.display)
self.menu.handleEvents()
else:
# Handle Events
self.handleEvents()
# Update and draw the sprites
self.sprites.update()
self.sprites.draw(self.display)
# Check for bat collisions
self.ball.batCollisionTest(self.player1Bat)
self.ball.batCollisionTest(self.player2Bat)
# Update the full display surface to the
screen
display.update()
# Limit the game to 30 frames per second
self.clock.tick(30)
```

08 The artificial intelligence bat

The AI bat is going to be a separate class from the player-controlled bat. However, we'll inherit the Bat class as the base class of our AI bat, which means we don't have to do much work to change what it does. We'll start the class just below the end of the Bat class with the line 'class AIBat(Bat): '. We then need to create the initializer method, which takes the exact same parameters as in the Bat class. We then use the super command to call the initializer method of the Bat base class and pass on the parameters that the AIBat Class received. We also need to set a speed for the bat to override the default speed of the normal Bat class. Having a slower AI bat is the easiest way of making it possible to defeat it. This also means that you can make the game easier or harder by changing the speed of the AI bat.

```
# The AIBat
class AIBat(Bat):
def __init__(self, displaySize, player):
# Initialize the Bat base class
super(AIBat, self).__init__(displaySize,
player)
# Make our bat slower so we can't always win
self.speed = 9
```

09 The AI movement code

The AI movement code is located in the update method of the class and is actually easier than you may think. All that it has to do is check where the ball is and then move either up or down to try to hit it back in time. The bat will basically follow the ball wherever it goes. Once we have set the direction of the bat, we call the update method of the base class, which actually moves the bat a certain distance according to the speed that is set. Using this method means that the AI bat can't intelligently hit the ball to try to defeat the human player, but it keeps the code simple and at least the ball is returned to the human player.

```
def update(self):
# Move ourselves so we are on line with the
ball
if game.ball.rect.centery < self.rect.
centery:
self.startMove("up")
elif game.ball.rect.centery > self.rect.
centery:
self.startMove("down")
else:
self.stopMove()
# Call the base class update method
super(AIBat, self).update()
```

10 Selecting a bat

We need to add a function to the main PiPong class that changes the bat. We've created ours just after the handleEvents function at the bottom of the class. It's a bit more complicated than assigning the new type of bat to where the old bat used to be, as the sprite group tries to use a bat that no longer exist, which doesn't end well. To combat this problem, we start by removing the current bat from the sprite group, then replace the old bat with the new type of bat, and finally add the new bat back into the sprite group.

```
def changeBat(self, batType):
# Change the player2 bat type
self.sprites.remove(self.player2Bat)
if batType == "cpu":
self.player2Bat = AIBat(self.displaySize,
"player2")
elif batType == "human":
self.player2Bat = Bat(self.displaySize,
"player2")
self.sprites.add(self.player2Bat)
```

11 Choosing the AI bat

We'll need to go back and change the handleEvents function of the Menu class before it's possible to select which bat you would like. This is easy though. All you need to do is call the function game.changeBat and pass through the bat type for both keys 1 and 2. 'Game' refers to the instance of the PiPong class that is running. You may have noticed that we haven't bothered changing the game event handling to ignore any keys for the player 2 bat if it's in AI mode. This is because the update event runs so often that it's almost impossible to make it move with the keys – and even if you did, it would correct itself immediately anyway.

```
if event.type == KEYDOWN:
if event.key == K_1:
# Disable the menu
game.changeBat("cpu")
self.active = False
elif event.key == K_2:
game.changeBat("human")
self.active = False
```

12 Wired for sound

Pygame comes with a mixer module that allows us to easily play sound. The Raspbian distribution comes with everything set up by default – so you should just be able to plug your headphones or speakers straight into the Pi.

13 The Sound class

We're going to make a Sound class that will make things nice and easy for us to play sound from anywhere in the game, as well as provide a layer of stability if the Pygame mixer was not able to initialise correctly. The most likely cause of this problem is that you are running an older distribution with a disabled or unreliable sound module. We placed our Sound class just after the Menu class. The initialisation method of the class begins by checking if mixer.get_init() returns None. If this is the case, then the mixer did not initialise correctly and enabled gets set to False. From this point on, the class does nothing useful if enabled is false.

```
# A class to handle playing sound effects
class Sound():
def __init__(self):
# Check if the mixer has loaded
if mixer.get_init() == None:
# If not there is a problem with sound
self.enabled = False
else:
self.enabled = True
```

14 A dictionary of samples

A dictionary holds a collection of key:value pairs. This is similar to an array, where the elements are accessed by an index, but instead they can be accessed by any key you like. In this case, we will

be using the name of a sound effect as the key, and a Pygame Sound object as the value. This is really useful because you can call a method on an object once, such as the play function of a Pygame Sound object, but change the object (and therefore the sound that plays) by changing the key that is passed to the dictionary. You will see how this works in the play function of our Sound class.

The start of the image is the remainder of the initializer function of the Sound class. The curly brackets are Python's syntax for constructing a dictionary. The dictionary in this case is called 'effects' and is a member variable of the class, so is prefixed with 'self'. Each sound effect is then added to the dictionary with a key, such as 'drop', followed by a colon to separate it from the value. The value is Sound object, which is returned from calling the mixer.Sound function, and passing through the path to a sound file.

The play function takes a parameter of what effect to play, which is expected to be a string that is one of the keys in our sound dictionary or just "hit", which will play one of the three hit samples. The play function only tries to do anything if the mixer has been loaded correctly. It's much tidier to check that here than having to repeat code and check it every time you want to play sound in the game.

The next part of the code checks whether the string "hit" has been passed through as the effect and then constructs a string "hit" followed by a number from 1 to 3, making sure that the number is converted to a string. The second value in the random.randrange function is not inclusive, which is why we've put 4 instead of 3. This gives us the key of a random hit sound to play.

Finally, we get to see how handy the dictionary is in helping us to have concise code because we only need to have one line that plays sound. The code self.effect[key] returns effectively puts the Sound object for that key in the place of self.effect[key] and then the play() function can just be called as if we were referring to a normal object.

Doing this without some kind of collection would probably involve having a bunch of if statements which call play on a different sound depending on the effect that has been passed through – this is much neater.

```
if self.enabled:
# Load up our sound effects into a
dictionary
self.effects = { "drop":mixer.Sound("audio/
drop.wav"),
"hit1":mixer.Sound("audio/hit1.wav"),
"hit2":mixer.Sound("audio/hit2.wav"),
"hit3":mixer.Sound("audio/hit3.wav") }
def play(self, effect):
# Only play if sound is enabled
if self.enabled:
# If effect is hit - randomly pick one of
the three
if effect == "hit":
effect = "hit" + str(random.randrange(1,
4))
# Play the sound effect with the key
specified
self.effects[effect].play()
```

15 Add the Sound class

As you may have guessed, it's now time to go back up to the initializer of the main PiPong class and create an instance of the Sound class. We've put ours just after the code that creates an instance of the Menu class.

```
# Create an instance of the sound class
self.sound = Sound()
```

16 Playing sound effects

We've placed code in three different places to play a sound effect:

1. Just before the main loop starts – to play an effect of a ball dropping. 2. When the ball collides with a bat – to play one of the hit sounds. 3. When the ball bounces off a wall – to play one of the hit sounds.

You can place yours wherever you like. The beginning of our run function from the PiPong class now looks like the screenshot above.

```
def run(self):
# Runs the game loop
# Just before we start the game play
the ball drop
self.sound.play("drop")
while True:

# The code here runs when every
frame is drawn
```

17 Playing sound effects (continued)

Playing sound from outside of the main loop is a little bit different because, rather than using self.sound, you will be using game.sound. The code for playing a sound when the ball hits the bat is part of the batCollisionTest function from the Bat class – see screenshot above.

```
def batCollisionTest(self, bat):
# Check if the ball has had a collision
with the bat
if Rect.colliderect(bat.rect, self.rect):
# Play a sound effect
game.sound.play("hit")
```

18 Giving it a go

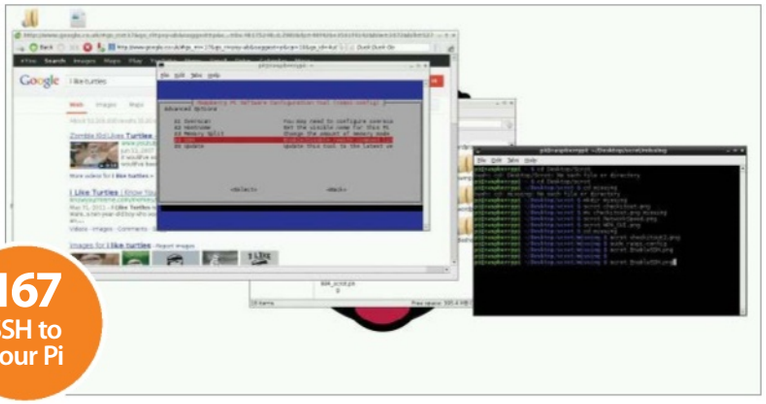
Press F5 in order to run your game and then select 1 or 2 from the menu in order to decide which type of player you would like to play against. You should find that if you select 1, the CPU-controlled bat will move up and down the screen to be in line with the ball. You should also hear sounds. Now everything is programmed and as it should be, you can sit back and enjoy your game!

■ Decide which type of player you'd like to play against, then enjoy the game

Troubleshooting

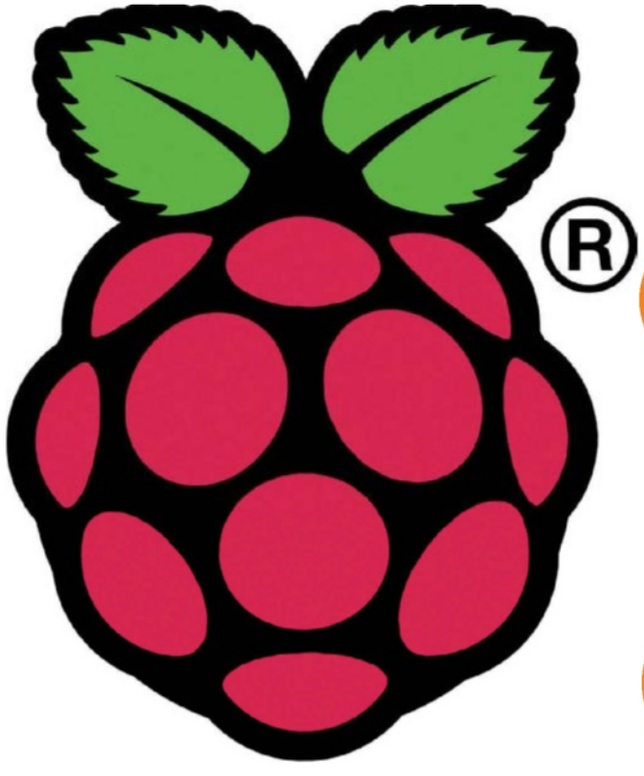
Overcome common Raspberry Pi problems

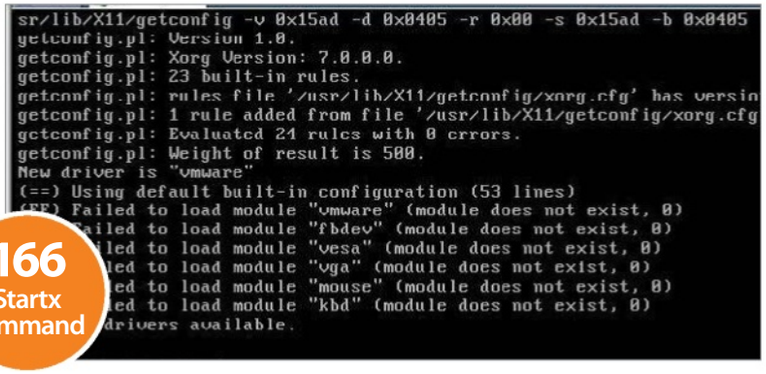
"No matter how simple a device, problems will occur"



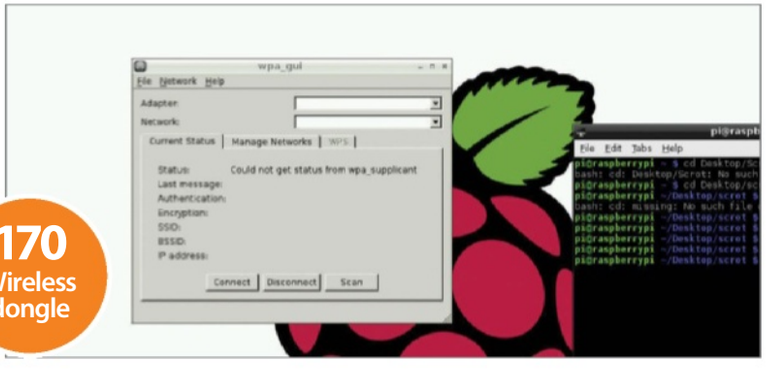
167 SSH to your Pi

164 Ensure no noise



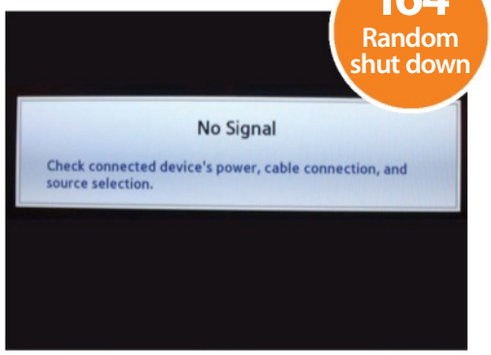


166 Startx command



170 Wireless dongle

168 ModMyPi case



164 Random shut down

169 Storage devices

165 Ethernet stops

"Discover just why your Pi works in quite the way it does"

171 Operating systems

Troubleshooting – Hardware

A list of common Raspberry Pi hardware problems and what to do when they occur

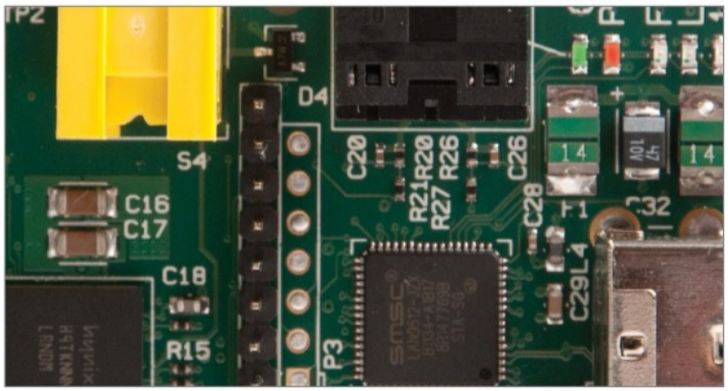
We've all been there. We've come back to something, and for whatever reason, the thing we're trying to do just won't work! We've plugged everything back in where we thought it came from, but there's nothing. So, what do we do when it comes to this? Usually there are a few different solutions to any one problem – but sometimes there's only one way, and finding that one key to victory might not be all that easy.

Maybe you've come back to your Pi after several months of it being sat in a drawer. You plug it in and boot it up and… X, Y or Z has happened!

What can we try when it just seems that everything has completely fallen over and nothing will work?

Well, hopefully your problem will be listed as one here. These are common problems for any Raspberry Pi user to come across at one point or another while using their device. Have a look through and if you're experiencing one of these issues, we've got a solution on hand for you.

Some of these things you'll want to slap yourself in the face for not thinking of – but that's all part of the joy of computers, right? The simplest of things can put you to the test for the longest of times, just ready for that eureka moment when it all finally comes together.



Problem 1: The green LED flashes and nothing is on screen

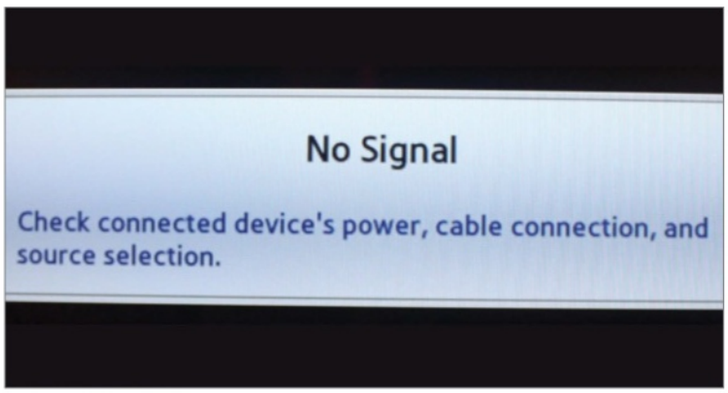
Solution: This is more common than you might think. It's likely down to either a blank SD card, or a lack of correct data on the SD card causing a boot failure. First, check your SD card is fully inserted. With the latest Pi firmware, if you have:

3 flashes – start.elf is missing
4 flashes – start.elf not launched
7 flashes – kernel.img not found

Problem 2: I can't see my USB hard drive

Solution: This is also another really common one and is generally caused by using a non-powered portable USB hard drive in a non-powered USB hub. The easiest way to fix this is to power one side or the other. If using a portable, USB-powered hard drive, you should really use a powered USB hub.

If you definitely have power, the hard drive could be in an unreadable format. Try using FAT32.



Problem 3: Pi shuts down or restarts randomly

Solution: If your Pi is particularly unstable, the first thing to check is the power supply you're using. First, swap out the cable. If you're still having problems, swap out the power supply. If your Pi is overclocked, it's also worth using the **raspi-config** tool to set it back to defaults. You may just be overheating it!

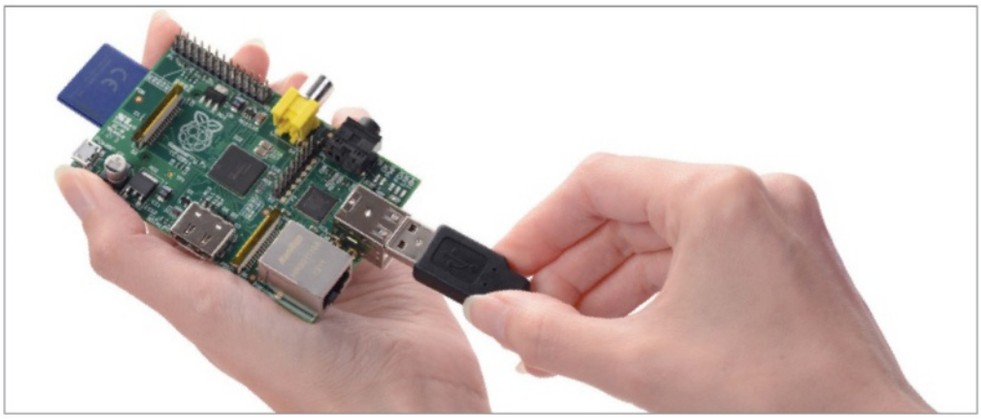
If setting it to default clock speeds helps, increase again gradually until you find something stable.

Problem 4: The HDMI image from the Pi has noise

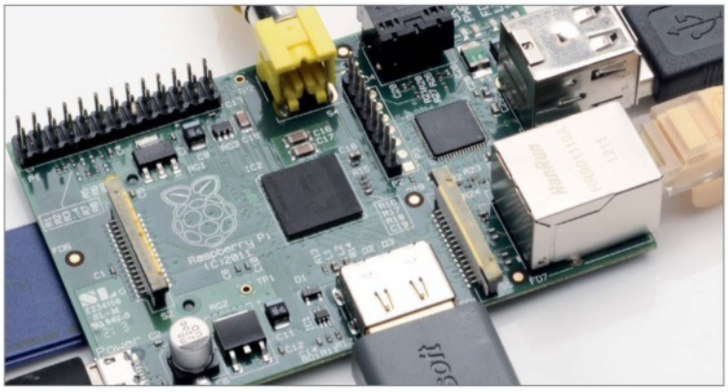
Solution: If the picture from your Pi is noisy (screen is shaking, dots or green lines, etc, moving around) then it's probably a dirty connection. Make sure the HDMI or composite cable is connected properly and free from dust and other contaminants. If you still have a problem, try another HDMI cable.

You can also try adding the following to your SD card's **config.txt**:

```
config_hdmi_boost=4
```

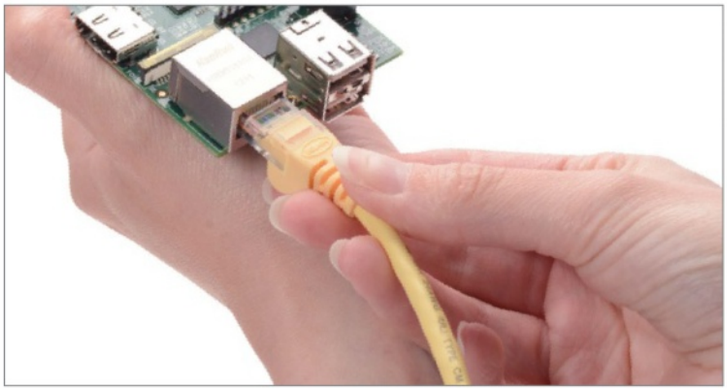


"Try the failing device in a powered USB hub – rather than directly into the Raspberry Pi"



Problem 5: I broke a part off!

Solution: If the part that broke off happens to be a silver cylindrical piece near the Pi's power input, this is easy enough to fix – this has been broken off by many other people due to its large surface/small mount area. This piece of the Pi reduces noise and stops power spikes. You can either reattach it (depending on your soldering skills), but you can use the Pi just fine without it – with most power supplies.



Problem 6: Ethernet stops working with some USB devices

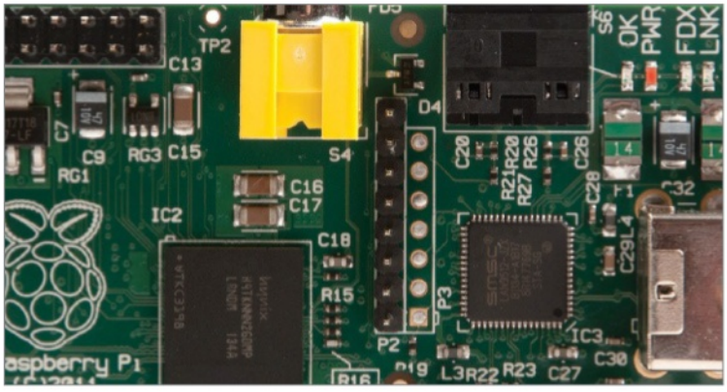
Solution: The first thing that you should do in this situation is check your power supply is working! It has been known for some power supplies to provide an inadequate source of power. First, try changing out your cable (it may just be a poor-quality cable).

Also, try the failing device in a powered USB hub – rather than plugging it directly into the Raspberry Pi.

Problem 7: Pi doesn't (always) boot

Solution: This is likely to either be a bad mount of an SD card or a power supply issue. Check your SD card: it may well be that it's not seated correctly. Take it out and put it back in straight – and make sure it goes in all the way it possibly can.

If it was definitely in correctly, try cleaning the contacts and restarting to see if it works. If this fails, try another power supply to see if it works there.



Problem 8: Red power LED is on, nothing on display

Solution: First, check the power cable is properly seated.

Check that your SD card has a valid image on it. Can you still read the SD card in the PC that the image was created on?

If you can, try booting your Raspberry Pi with just the power supply connected. Watch whether the 'OK' light flashes. If it does, add the cables back in one by one.

Troubleshooting – Software

A list of common Raspberry Pi software problems and what to do when they occur

Software can be just as much of a pain as hardware. The problem with software, however, is that different configuration options can lead to a really bad, frustrating time. Hardware we have attached can affect the configuration of our system and lead to further issues – sometimes the only way to go is from the ground up. You have got to think logically about these things.

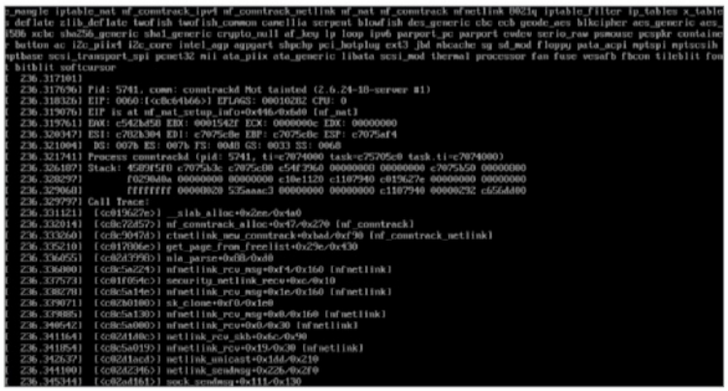
Some problems will seem to have little to no correlation with their answers, but those scratch-your-head moments soon go when you realise that actually the thing is working and doing exactly what you wanted it to again. There's a lot you can do from the command line to help with any of this stuff, but things aren't always as clear as they seem.

Thankfully there are a lot of helpful tools to start making a dent in the most common problems.

In this section we feature a list of the most common software problems with their solutions.

If the worst comes to the worst then you can always rebuild it. Due to the Pi's nature, doing this is a simple reflash of a single image, rather than painstaking hours of driver and software reinstalls like some other operating systems out there.

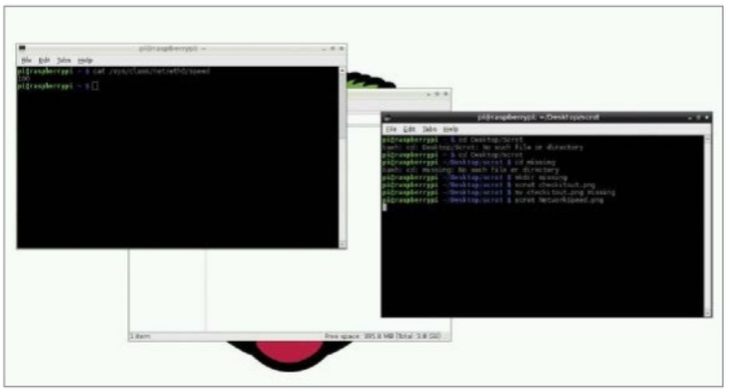
As always, be sure to keep frequent backups of your most precious data, though.



Problem 1: I'm getting a kernel panic on every boot

Solution: If you're seeing kernel panic messages each time you start your Raspberry Pi, the first thing to try is booting it without any USB devices in – and adding them one by one afterwards.

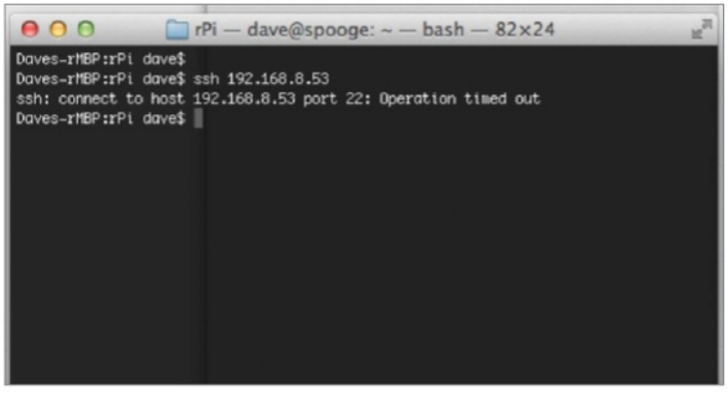
If you're still getting the messages, it's likely that your flash of the SD card was unsuccessful. Reflash the card and try again. Make sure you use admin privileges to do it!



Problem 2: My Pi's Ethernet connection is working at 10Mbit, not 100Mbit

Solution: This may or may not be inaccurate. On the earlier Pi revision boards, the 100Mbit LED was mislabelled – it was actually a 10Mbit socket. On newer revision boards, however, it was correctly labelled on the newer boards that actually have a 100Mbit socket.

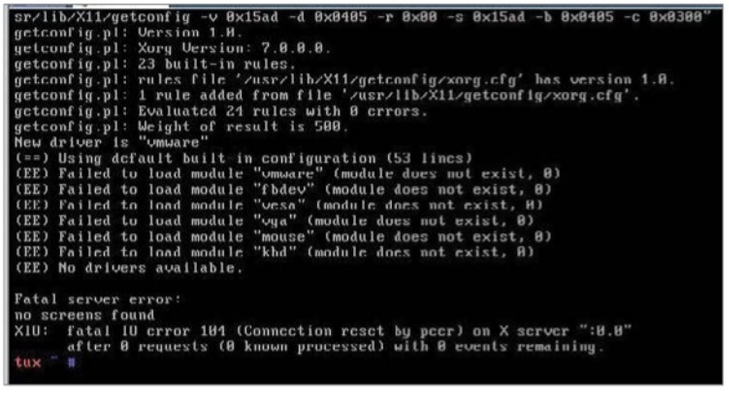
You can check your link speed with `cat /sys/class/net/eth0/speed`



Problem 3: I cannot SSH to my Raspberry Pi

Solution: If you're receiving a connection time-out error when you try to SSH to your Pi from another machine, it probably means that SSH access to it is disabled.

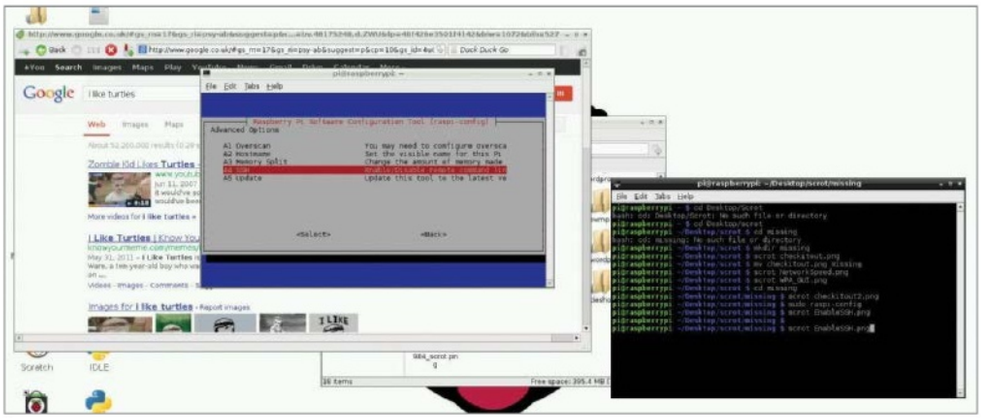
You need to open up the **raspi-config** tool, then use the option 'ssh' to enable SSH. You should now be able to connect to your Pi. The default user/password combination is pi/raspberry.



Problem 4: Startx fails to start window manager

Solution: If you just get errors instead of a working desktop when trying to use the **startx** command, you may be out of disk space. This possibly either means you need to expand the rootfs using the **raspi-config** tool if you have an SD card bigger than 2GB. If not, you probably need to get a bigger SD card.

If you know there's space, try renaming the **.Xauthority** file under your home directory, temporarily.



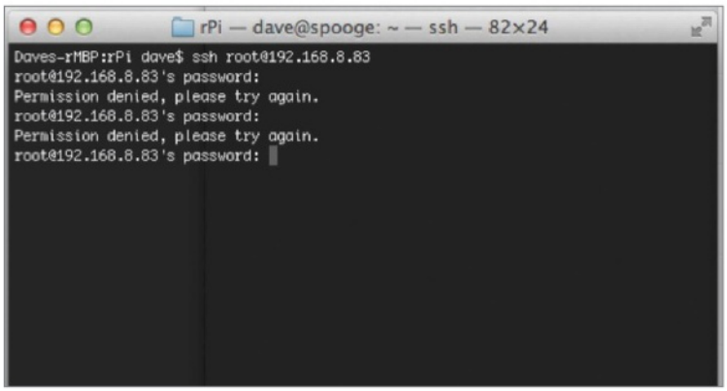
"Open up the raspi-config tool, then use the option 'ssh' to enable SSH. You should now be able to connect to your Pi"

Problem 5: The visible desktop goes off the screen

Solution: You probably want to turn overscan off. Use the **raspi-config** tool to disable overscan.

If you now see a big black border round the edges of your screen, you can usually use your TV's settings to zoom in.

If not, try setting the overscan manually in the **config.txt** using the properties overscan_left, overscan_right, overscan_top and overscan_bottom.

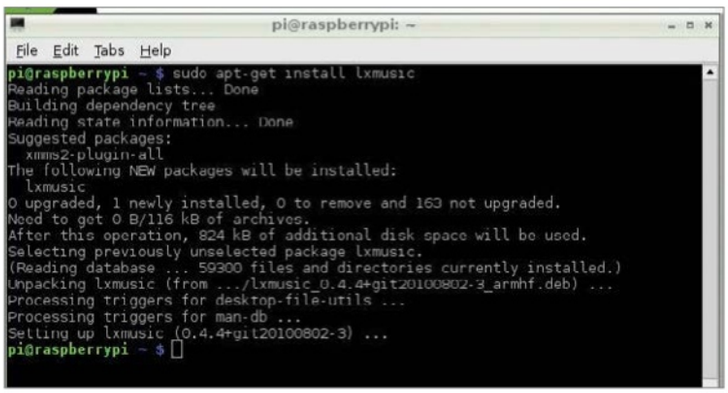


Problem 6: I don't know the root password

Solution: This is probably because the root account in many newer Linux distributions is disabled by default – and as such doesn't have a password. You can enable a password by entering the following command in the terminal:

```
sudo passwd root
```

Exercise caution when doing this, however, since meddling with root accounts can be dangerous.

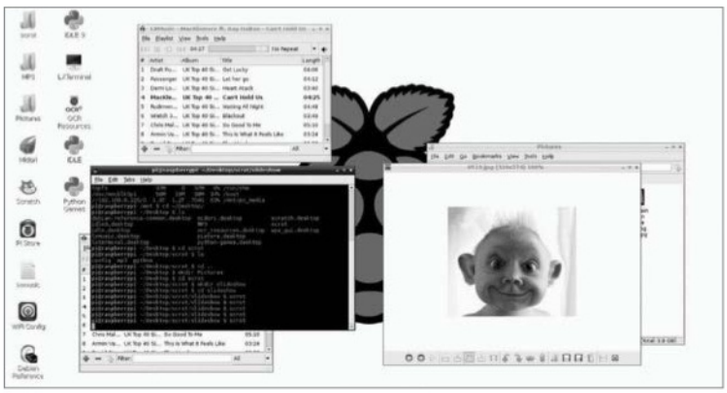


Problem 7: I can't install new software with apt-get

Solution: If you're seeing the error 'Package xx is unavailable' you probably need to update the Apt tool first. Seeing this error means that your software list is out of date.

Start by running the command `sudo apt-get update`, let it update its list of packages (you'll need a network connection) – then `sudo apt-get upgrade`.

When this completes, try installing your package again.



Problem 8: Composite output is only black and white

Solution: The Pi's composite output defaults to NTSC (American). Some PAL (European) TVs may either now show and image, or display it in black and white. To solve this, change the **config.txt** on the SD card to add/modify:

```
sdtv_mode =x
```

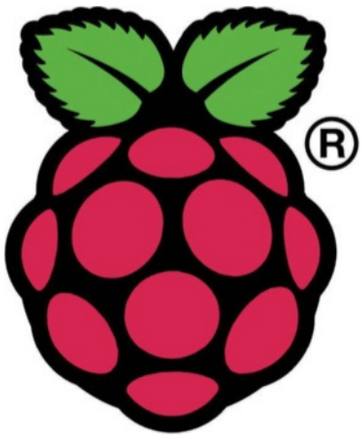
Where x is: 0 – NTSC; 1 – Japanese NTSC
2 – PAL; 3 – Brazilian PAL

Your questions answered

The most frequently asked questions about the Raspberry Pi answered for you

There are many questions thrown around about the Raspberry Pi, due to its wide range of both users and possible uses. Its ease of accessibility and low cost make the Pi appealing to many different kinds of people, from complete computing novices to more advanced users who want to use it for out-of-the-ordinary projects because of the Pi's low-risk nature. This leads to a lot of different questions being asked about the Raspberry Pi: Why is it so cheap? What operating systems can I use on it? What SD cards can I use with it?

Well, we have put together a list of a few of the most commonly asked questions regarding the Raspberry Pi here. With limited space, there's no way we could possibly cover everything you'd ever need to know, but hopefully it'll go some way to explaining queries you have.



■ This is the Raspberry Pi logo – you'll be familiar with this from the Raspberry Pi board, website, or Raspbian desktop

Why is the Raspberry Pi so cheap?

■ **The Raspberry Pi Foundation is a charitable organisation registered with the Charity Commission for England and Wales in May 2009.** Its aim is to promote the study of computer science and related topics, especially at school level – and to put the fun back into learning computing.

The purpose of the Pi was to create a small computer that was accessible to every kind of end user – from the average home user that wants something to play around with or come up with a new and inventive use, to the education sector whereby currently, technology in many parts of this country (and across the world) are inaccessible in schools due to cost – leading to little to no understanding of computer science and creating a big gap in the skills that children in schools are learning.

The price you pay reflects the cost of manufacturing and shipping costs for the assembled alone – nothing more, nothing less.

There are two models of the Raspberry Pi – the Model A and Model B boards. The main distinguishing factor between the two is that the Model B has 512MB RAM and an Ethernet port, whereas the Model A has 256MB RAM, no Ethernet port and just a single USB port.

A lot of effort and research went into the Raspberry Pi, keeping costs down to a minimum by removing certain components that weren't critical – such as an on-board real-time clock which required a battery and drove up prices – and made the physical requirement for space much larger. Lots of non-critical components can be added to the Pi should you need them.

Where can I get a case for my Raspberry Pi?

■ **It's not ideal using the Raspberry Pi on its own, out of a case.** It lends itself well to be quite vulnerable and easily breakable if it's not in some way enclosed in a case.

There are a lot of people now selling cases for your new Raspberry Pi. The first commonly available case was the Raspberry Pi case from ModMyPi (**www.modmypi.com**) – it features a two-part moulded plastic case that can be ordered in two different colours.

The other common Raspberry Pi case is the 'BerryBlack' case – available from **http://shop.pimoroni.com**. This again features a two-part plastic case, but requires Model B users to knock out some plastic inserts to be usable.

Other than these, a quick Google or eBay search will reveal lots of different cases available.

It's worth noting that given the relative size and lack of complexity of the Raspberry Pi board, there's a vast community of people that create their own cases – in anything from Lego boxes to cigarette rolling tins. So long as you can get the required ports in the required places, you can use whatever you like.

In the Raspberry Pi site archives, there's also the option to print off a template for a DIY case – made entirely from cardboard. Not exactly sturdy, but a good way to make your Pi a little more noticeable. See **www.raspberrypi.org/archives/1310**

■ A Raspberry Pi sporting a ModMyPi case. You don't have to have one made of plastic, though – why not try cardboard?

■ Windows cannot be used as an operating system on your Raspberry Pi, although there are ways to mimic its use

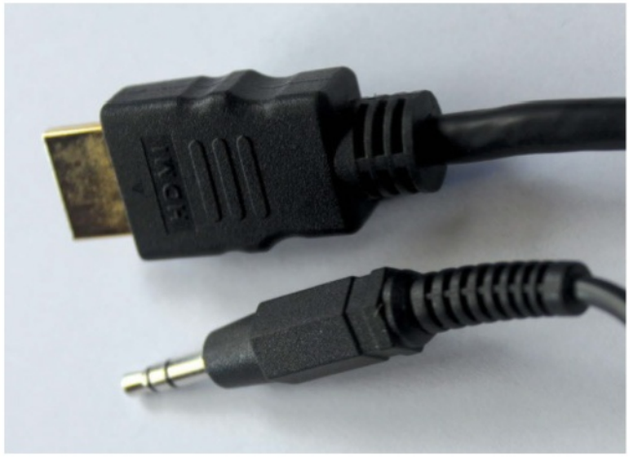
Can I use Microsoft Windows on my Raspberry Pi?

■ **Due to the Raspberry Pi's ARM architecture, unfortunately it is not possible to run the Windows operating system on a Raspberry Pi.** This is because the ARM processor does not understand the same instructions as a processor with an x86 or x64 architecture.

However, there are other ways to give the impression of running Windows on a Pi. These are mostly through remote desktop-like solutions such as VNC or Citrix.

'Remote desktop' means that you have a physical Windows machine or server, sat on a normal desktop for example – then you use a piece of software that will run on your Raspberry Pi to connect to it over a network. When connected, you're able to use the other computer as though you were sitting at it – though the operating system itself is not actually running from the Pi.

While not that useful on its own, for people with a Citrix server looking to deploy thin clients out to their business, this is a very cheap way of deploying lots of clients at low cost.



■ The cable at the bottom is a standard 3.5mm audio jack; above it is an HDMI cable. You can use either to output Pi audio

What are my options for audio on the Raspberry Pi?

■ **The Pi is able to play high-quality audio.** It has two options: a 3.5mm audio jack, and audio out over HDMI.

These open up options of getting audio out through almost anything. A set of PC speakers will plug into the Pi's 3.5mm audio jack, but this is also what you'd use to plug in your headphones.

At a slightly higher level, rather than using the analogue audio jack you could use the HDMI port – which means that the audio goes out digitally. As well as giving a higher quality of audio, it also means that you can run this into your TV or other HDMI advice, then out through your TVs optical connection which may go to your surround sound system or a whole bunch of other stuff.

If you want to specifically switch between the two audio outputs, you can do so at the ALSA mixer level, by using the following command in your terminal:

```
amixer cset numid=3 <n>
```

Where n is:
0 – auto
1 – audio jack
2 – HDMI

What's the biggest SD card the Raspberry Pi supports?

■ This is an SD card – it's your Raspberry Pi's boot and storage device, though you can add more with a USB drive

■ **Officially, the biggest SD card size that's been tested by the Raspberry Pi team is a 32GB card.**

As far as choosing an SD card goes, the best way to select is just to not go for the absolute cheapest one you can find.

Go with something of a recognised brand. While class isn't critical, it's worth noting that the Pi will be considerably slower with a card of a lower class.

There is a list of officially tested cards on the eLinux site – using this you'll be able to look for cards of a specific size and/or class that meet your requirements: **http://elinux.org/RPi_SD_cards**

On here there's only one tested 32GB card, but it's confirmed to work without issue. The minimum sized card you can use is 2GB but it is recommended to use at least 4GB to run additional programs on it.

If you simply require more storage space for your Raspberry Pi, one of the easiest and cheapest ways to achieve this is to have a smaller SD card just to run your operating system, a USB hub and a powered USB drive to store all your data. Mechanical hard drives are still much cheaper to buy than flash storage, but the drive would be bigger than the Pi – and as the Pi has limited power available, it requires powering separately.

How can I add internet to my Raspberry Pi?

01 By ethernet cable

Generally, the least configuration-intensive way is to use a network cable. It guarantees compatibility because it'll work out of the box with any of the available Pi distros. Simply plug both ends of the cable in and let the Pi do the rest; it should go and get an IP address.

02 By wireless dongle

Plug your Wi-Fi dongle in and if using Raspbian, use the 'WiFi config' tool that lives on the desktop. Select your adaptor and network, then type in your network key. As long as your dongle is supported by the Pi, this should be all you need to get going.

03 Check it out!

When you've done either of the steps above, start your favourite browser and do the age-old 'am I connected?' test by trying to search for a (non-indecent) term. If you yield any results, you can be sure that you're online. If you get any errors, things might not have gone so well.

Can I overclock my Raspberry Pi for more speed?

■ **Yes you can, but if you start seeing problems it's best to switch back to your original settings – or keep dropping them until you maintain stability.**

You can set your overclock settings with the **raspi-config** tool, or by modifying **/boot/config.txt** on your SD card.

There are several different presets for overclocking in the raspi-config tool. Generally speaking, the 'Modest' overclock gives a little performance increase all round, and is the safest and most stable for the hardware.

If you'd like to be a bit more adventurous and play around with the config.txt for overclocking, you should just be able to modify the values:

```
arm_freq=xxx
sdram_freq=xxx
```

Beware that this is clearly more dangerous since you're inputting raw values to be read upon startup of the system.

It's also worth noting that if you manually modify the config.txt and set the `force_turbo=1`, your warranty will be null and void.

It's also possible to check the current CPU clock speed using the terminal and a simple command:

```
cat /proc/cpuinfo
```

You'll see a few different lines of information being thrown out, but the one you're most interested in is the second which reads 'BogoMIPS' – and by default should be somewhere around 700.

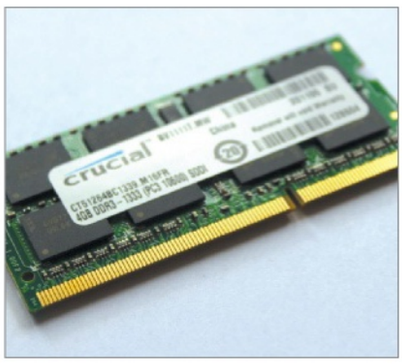
Each time you make changes to your Raspberry Pi's overclocking settings and have rebooted the system, run this again and you should see that this number has changed in accordance with what you've set as your desired clock speed.

"The 'Modest' overclock gives a little performance increase all round"

Can I add more memory to my Raspberry Pi?

■ **Unfortunately not.** The Raspberry Pi is a single PCB. The memory is not removable – nor is it swappable. If you buy a 256MB Model A, you're stuck with a 256MB Model A. You are unable to use any means to upgrade your Raspberry Pi, be it memory, CPU or graphics.

You are able to overclock your device, though (see above). This will potentially allow all components to run marginally faster, but you cannot add more of them.



■ This is a piece of normal, DDR3 laptop memory. It's not possible to add more memory to the Pi

■ This is a powered USB hub. It is known as 'powered' because it has a power supply to provide power to connected devices

How can I connect more devices to my Raspberry Pi?

■ **The easiest way to be able to connect more devices to your Raspberry Pi is by using a USB 2.0 hub.** There are two kinds of USB hubs – powered and non-powered.

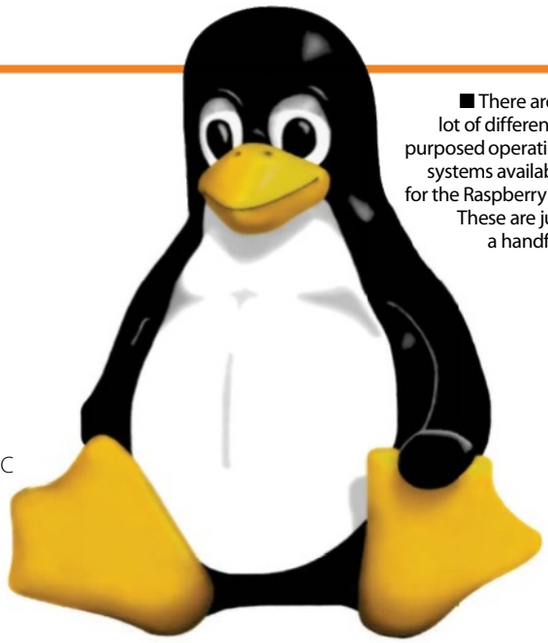
If you just want to connect very low-powered devices such as a USB flash drive, keyboard and mouse, you should be okay with a non-powered hub.

However, if you want to add other devices – such as a USB-powered hard disk – then you'll require a powered hub (one that comes with its own power supply.)

Failure to use a hub of this kind will often lead to devices not being recognised by the Raspberry Pi and being inaccessible – or, at the worst case, leading to system instability.

If you're making a decision about a hub and don't know what to get, always go for one that's powered.

What operating systems can I run on my Pi?



■ There are a lot of differently purposed operating systems available for the Raspberry Pi. These are just a handful!

■ **Already, there are a lot of different operating systems available for the Raspberry Pi.** Anything that runs on an ARM system should be okay.

However, there are a few distributions that are favoured in the community.

Debian (Squeeze) was the default distro used on the early versions of the Raspberry Pi board. It was the first fully functional OS with a desktop, browser and set of development tools.

Raspbian is the most common all-rounder and is used as a generic, all-purpose operating system. It's based on the Debian distro.

Arch Linux ARM is the ARM variant of Arch Linux. It's another general-purpose operating system, specifically lightweight (small) and simple, but not so easy to use for novices.

RISC OS is another desktop-driven OS, originally used on Acorn computers. Stable releases are not that common, with the last being December 2009, though the last unstable release was February 2012.

OpenELEC and Raspbmc are both standalone operating systems that run XBMC – a common media-centre application for playing music/video and looking at pictures. If you run either OpenELEC or Raspbmc, it will boot straight into XBMC – there is no other user interface. They are single-purpose distros.

Is it possible to boot from a USB device?

■ It's not possible to boot directly from USB, although it can be used after boot for more storage

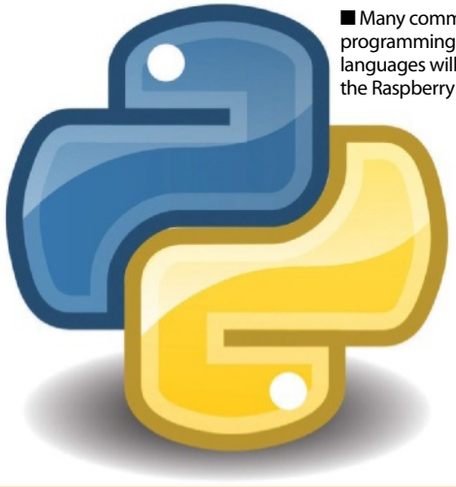
■ **No – the config.txt is the first thing that's read upon starting the Pi.** Before the CPU's initialised, the GPU checks this file on the SD card, at which point there's no USB support.

For this reason, it's not possible to start a Raspberry Pi without an SD card being present with some sort of data on it.

It is, however, possible to have a USB boot device take over duties as the main storage device once the system has booted.

"It's not possible to start a Raspberry Pi without an SD card being present"

What programming languages can I use on the Pi?



■ Many common programming languages will work on the Raspberry Pi

■ **The 'official' programming language that's being supported as the educational learning language on the Raspberry Pi is Python.**

However, any programming language that will compile for ARMv6 is usable as a means for making applications for the Pi.

Two languages have been included by default in Raspbian. These are:

Scratch – This is an entry-level language that doesn't require syntactically correct code, but helps you understand mathematical and computational concepts.

Python – As it's the 'official' language of the Pi, it'd be silly not to include it, right? It's powerful and not too difficult to learn.

Other languages that'll work on the Pi already include several other widely known, lower-level ones: HTML5, Java, JavaScript, C, C++ and Perl.

To use and compile any of these directly on the Pi, you'll need to install your own compilers – though these can be obtained using the Aptitude package manager without too much hassle.

Your Raspberry Pi glossary

There will be a lot of new terms to learn while using your Raspberry Pi. Here are some of the most common ones…

Apt-get
To install or update software manually, you may sometimes find yourself using the apt-get command. It's part of aptitude, the package manager for Raspbian and some other Raspberry Pi operating systems.

Arch Linux
Arch Linux is a Linux distribution that is light on system resources and encourages development from its involved community. However, unlike other distros, its focus is simplicity from a developer point of view, not from the end-user. While highly customisable, it can be tricky for novices.

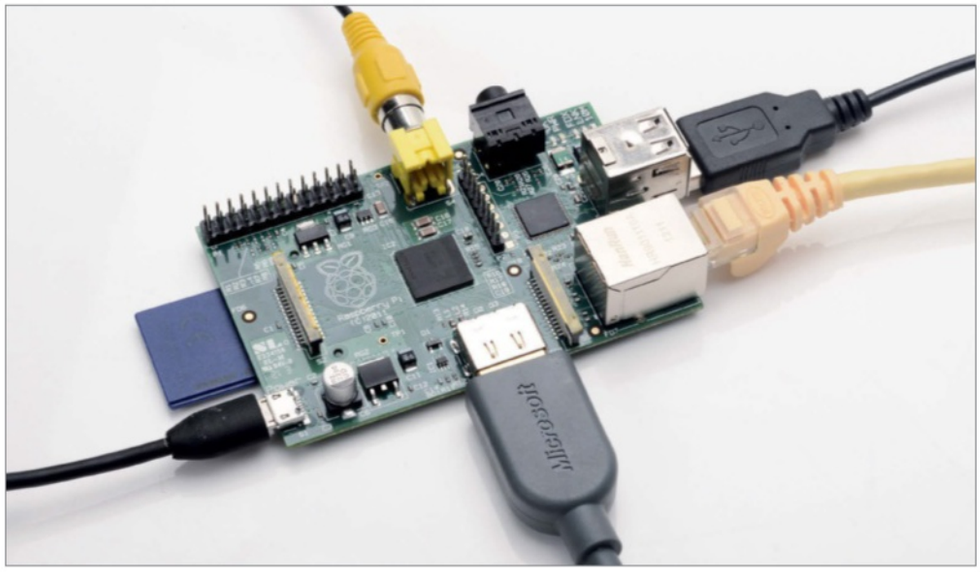
Arduino
A lot of people like to use their Raspberry Pi in conjunction with an Arduino. It's a board that allows you to easily power motors and lights and other physical objects that can make use of computers

ARM
The Raspberry Pi uses an ARM chip. ARM is one of the types of processor or CPU that powers modern computers. ARM type chips themselves are used in mobile devices, such as phones and tablets

Audio jack
An audio jack is either an input or output socket for sound. The standard 3.5mm audio output on the side of a Raspberry Pi is a way of getting audio out of the system – generally if HDMI is not available. Such jacks are commonly found on MP3 players, phones, TVs and so on.

Camera Module
While you are able to use a USB camera on the Raspberry Pi, there is also a specific camera that is specially made for the Pi. It can be connected to one of the special ports on top of the Pi, and controlled with commands.

CLI
Short for command line interface, Arch Linux uses one of these instead of a graphical desktop. Instead of using a mouse to click icons, all functions are done with written commands. You should rarely come across it.



■ An operational Raspberry Pi - with USB, ethernet, HDMI, micro-USB and video out ports in use. One or more of these will be connected, depending on what you're using your Pi for. The Model B+ does not have a video out port

Cold boot
A cold boot is when you start a computer from a completely 'off' state. It allows all hardware to completely reset, and can often solve anomalies – particularly with a GPU or CPU.

Command prompt
A command prompt is a text interface to a computer system. It allows a user to type in commands rather than use a graphical interface. It's often used as a way of recovery or when a normal desktop environment isn't required (such as on a file server).

Composite Video
As well as having high-definition video output, the Raspberry Pi also has a traditional yellow video port, used for SCART-style displays. Not all TVs use an HDMI port, so it's a good alternative to have in case.

Config tool
The raspi-config tool is a convenient way of accessing many of the operating system's general settings very quickly. It allows you to set parameters for overclocking, overscan, memory splitting, language and more.

CPU
The central processing unit, all commands and fed through here one way or another. It's the brain of the Raspberry Pi, and one of the most core components that makes it work.

Cron job
A 'cron job' is the Linux equivalent of a Scheduled Task. It allows you to run a specified task at one or more specified times or events – such as when booting a machine – or every day.

Desktop environment
Raspbian's graphical desktop is called LXDE, which is one of many Linux desktop environments. LXDE is

■ A USB hub is one the first peripherals you should invest in when you buy your Raspberry Pi

used in particular as it is simple, and doesn't need a very powerful computer to run it.

Distro

Distro is short for 'distribution'. There are many different distros of Linux – each contributed to by different groups of people, and generally aimed at performing specifically different tasks. Some are targeted more at user desktops, some at enterprise servers.

Ethernet

A wired internet cable, and what you'll be using if you don't have access to a USB wireless adapter. There are a few varieties of this kind of cable, so always check to make sure the speed is the same as your routers.

External hard drive

SD cards don't have a lot of space, so the cheapest and best way to add more storage to your Raspberry Pi is to have a USB-powered hard drive. This is great if you plan to keep a lot of files on the Pi

File system

A file system is your computer's filing cabinet. It arranges things on your hard disk into a way that can be read by the software on it – and ultimately, the end-user.

Firmware

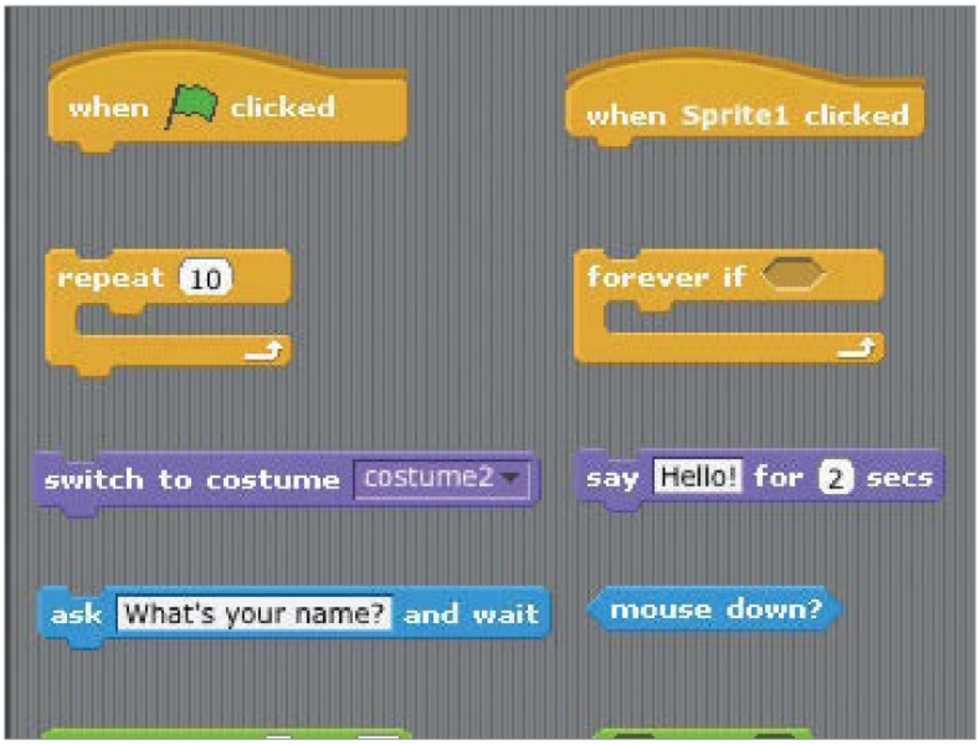
This is the core code that runs the Raspberry Pi. Sometimes updates are required to get new software or hardware to work. The config tool helps you update the firmware, so keep an eye on the Raspberry Pi site for these.

Free software

The saying goes "free as in speech, not as in beer." What it means is that free software isn't free to buy; it's free for anyone to use as they wish, such as modifying it, using parts of it for your own project and so on. It's a software philosophy.

GertBoard

Created by one of the original Raspberry Pi



■ Scratch is one of the easiest programming languages for the Raspberry Pi

developers, a GertBoard allows you to extend the functionality of the Pi to be able to interact with physical objects. This is used as the base of the GertDuino.

GertDuino

A GertBoard/Arduino hybrid, the GertDuino allows for easier integration of an Arduino project into the Raspberry Pi, at a more affordable price. It contains the same parts as an Arduino Uno, and works with all accessories for the Uno.

GPIO

The line of pins along one side of the Raspberry Pi. This is used in some more advanced projects to better control other circuitry or a breadboard. For most applications, you won't need to use them

GPU

GPU is short for 'graphics processing unit' – this is the chip that processes the information required to display graphics and then outputs that information to something that's human-readable.

HDMI

HDMI is short for High-Definition Multimedia Interface. It's a compact interface for audio/video, used for sending uncompressed digital data from one device to another.

IDLE

The integrated development environment (IDE) for the Python programming language. It lets you easily write Python code and test it, giving reasons for why it may have failed. You can also test out commands without writing code.

Image

In this case, not a picture, but a file that contains the operating system to run a Raspberry Pi. The images you download from the Raspberry Pi website are then used to create the same operating system on your SD card.

Linux

Linux is a variant of the UNIX operating system. It is totally open source. Open source means that anybody can take the source code and make their own changes. This is particularly common when security is paramount.

Maker

A maker is a person that makes things. This may sound simple, but usually it applies to people that create ingenious homemade devices. They tend to use the Raspberry Pi and Arduino in projects.

Micro-USB

A micro-USB cable has a full size USB plug at one end – and a micro-USB plug at the other – this is the name of the cable that gives your Pi power.

Model A

One of the types of Raspberry Pi you can buy. This version only has one USB port, less RAM and no wired internet connection. However, it is cheaper, and still holds the same processing power, so some people still have use for it.

Glossary

Model B and Model B+
The Model B version of the Raspberry Pi is the main one you'll see. It comes fully powered, and with more ports than the Model A. You can basically use it as a real computer, as well as for hobby projects. The new Model B+ comes with a microSD slot, extra GPIO pins and two extra USB ports.

MP3
An MP3 is an 'MPEG Layer 3' file. It's a very common file format for audio – it's generally quite heavily compressed to give a small file size to be easily downloadable from the internet.

NOOBS
A newer way to install an operating system to the Raspberry Pi, NOOBS is a selection of files you put on the SD card that allow you to download and install all the major distributions to the card. It's a lot easier, and faster, than writing an image to the card.

OpenELEC
One of the two most popular media centre distros for the Raspberry Pi, OpenELEC actually encompasses a range of PCs and devices. It uses XBMC media centre with some of their own modifications, and has a better support team.

Open source
Open source is a term used for computer software where the source code is freely available. This allows communities of people (or individuals) to make changes to the code, or to freely distribute the compiled contents of the computer package. It's all about the spirit of collaboration by programmers to solve problems and be creative.

OS
OS is short for 'operating system' – it is a software application that runs upon the startup of a system. It's basically a way of getting a computer to do useful things – it is able to interpret commands in a way a user can understand.

Overclock
Overclocking in its very basic form is the process of running a computer component at a higher clock speed than it was manufactured for (hence over-clocking). There are other ways to overclock, but this is the simplest.

Overscan
An area of extra image around the outside of a video picture that might be missed by the viewer. Turn overscan off if you can't see all of your screen.

Packages
Packages are bundles of software used in the Linux world. There are package managers to simplify installation of them and their dependencies. Dependencies are where one package requires one or more others to run. This is common due to the open source nature of Linux.

Pidora
A specific Raspberry Pi distro based on Fedora, a popular distribution in the Linux world with lots of up-to-date and cutting-edge software. Early versions of the build were supposed to be the official Pi operating system, however it was a little bit buggy.

Python
Python is a general-purpose and high-level programming language. It's often used as a scripting language, but can also be used to create standalone executable applications.

RAM
The computer's memory. Memory is different from hard-drive space; it's where a computer stores the current tasks it's working on, and the software that's running. The Raspberry Pi is no different, and requires RAM.

Raspberry Jam
An event for intrepid Raspberry Pi users to show off or learn new skills, while talking with people on how to advance computer science in schools. They pop up around the country, and can be found on the Raspberry Pi site.

Raspberry Pi
The Raspberry Pi is a low-power, low-cost, tiny computer – generally used for small projects, education and development purposes. It's a perfect complete system to learn about computers, with little impact if it all goes wrong, since the hardware is so inexpensive.

Raspberry Pi Foundation
The organisation behind the Raspberry Pi lives in Cambridge, and makes sure that Pi-related news and products get out to the people who care. Their website is the main hub for Raspberry Pi information and support.

Raspbian
Raspbian is a Linux distro. Based on Debian 'Wheezy', it is specifically aimed at the Raspberry Pi as a desktop operating system, complete with a graphical user interface (GUI) and several built-in applications. It is designed to be compact a nd lightweight to run well on the Pi's rather limited resources.

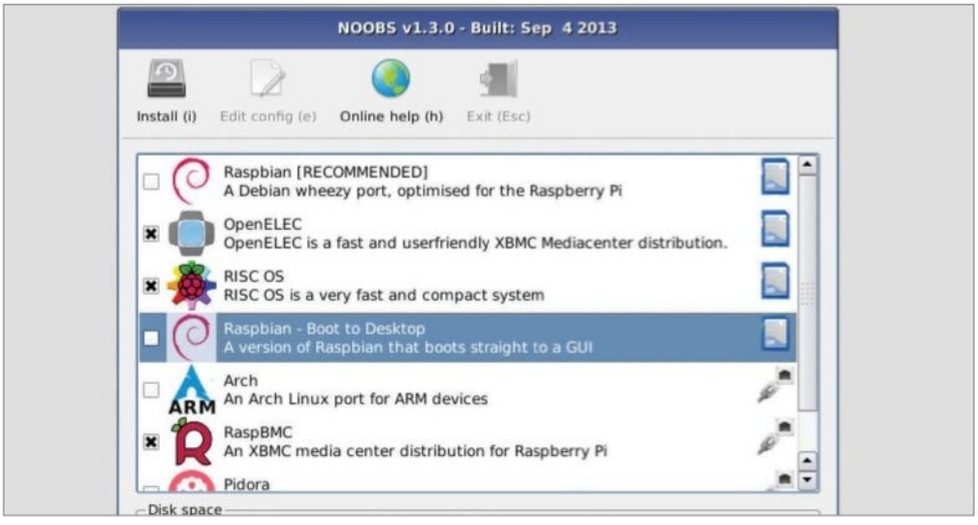
RaspBMC
This is a Raspberry Pi specific media centre operating system, turning your Raspberry Pi into an all-in-one entertainment centre that can be hidden away behind your TV, while making watching or streaming files easy.

Reboot
A 'reboot' is a software shutdown and restart of a computer. It's the easiest way of clearing everything from memory and starting the system afresh with nothing running.

RISC OS
RISC OS was a computer operating system released by Acorn Computers in 1987. It's still in development, and today it is used for computers running an ARM processor. A version of RISC is available for the Pi.

Scratch
Scratch is a simple programming interface. It is designed to be an introductory step to learning to program computer applications using a low-level language without having to learn syntactically correct coding.

Script
A script is a sequential list of actions that are to be performed specifically by a computer, generally used in some way to automate a certain function – for instance, it could be to update all of your packages regularly.



■ NOOBS is one of the best, and easiest ways, to install a distro to your Raspberry Pi

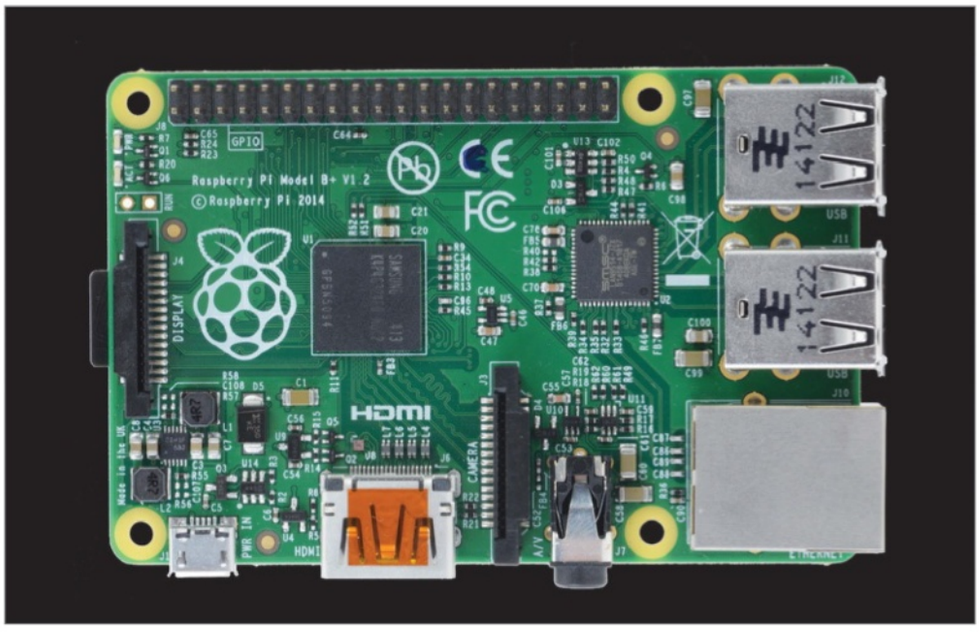
SD card
SD is short for 'Secure Digital' – it's a card used for storing data on. Your Raspberry Pi uses one as its boot device – effectively as a hard drive. You've likely seen one before in your camera.

Server
A centralised computer that serves files and/or internet to normal, desktop computers. A lot of people like to use their Raspberry Pis as simple servers for holding their files, as it draws very little electricity in the process.

Shutdown
When you've finished working on your Raspberry Pi, or any other computer, it's a good idea to shut down – this is the process of clearing down everything running to a state whereby you can safely power off the system. This is the ideal way to then restart from a 'cold boot'.

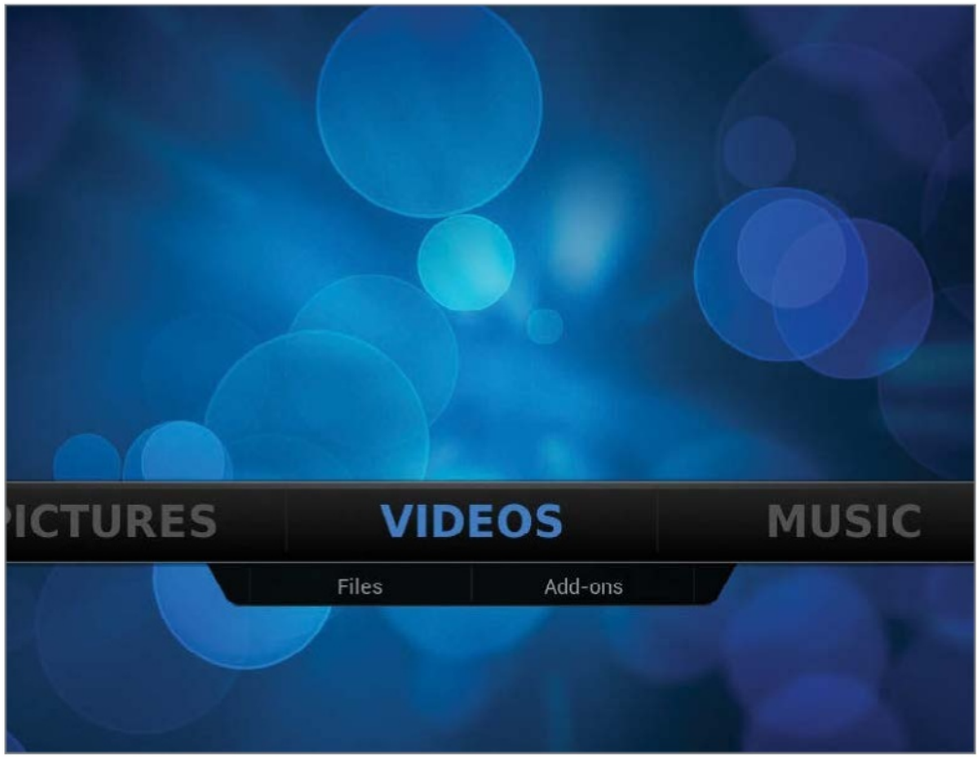
Slideshow
A slideshow is the idea of taking lots of different pictures and displaying them one after the other in an automated way. This means that rather than having to sit by your computer clicking through photos, it'll do it for you.

SSH
SSH is short for 'Secure Shell' – it's a way of giving a command-line interface to a system and is generally used for remote access. It's possible to operate a system completely remotely in this way.



■ The components on the board of the new Model B+ have been rearranged to make a tidier workspace or project

sudo
The term 'sudo' is a Linux command that is used in order to temporarily elevate the current user's privileges – therefore it allows the user to perform actions that would normally only be available to the administrator user, which can be very useful. It requires re-authentication of the current user.



■ Using your Raspberry Pi as a home theatre PC is a cheap and efficient solution

Terminal
A terminal emulator is one of the tools used by Linux operating systems to write command prompts. It uses the same syntax and commands as a normal command-line interface, and allows you to perform these tasks while using a desktop environment.

Torrent
A legal way of downloading and sharing files, such as a Linux distribution, where 4MB blocks are sent at a time. The system allows you to connect to several people at a time, meaning there's less strain on any one server.

USB
USB is short for 'Universal Serial Bus' – well known because it's an industry-standard connector. It doesn't matter what's on the other end, so long as the computer knows what to do with it.

USB Hub
A way of adding more USB ports to a computer, the Raspberry Pi can also make use of them. However, if you plan to use one on the Pi, you will need to get a powered hub.

XBMC
The software used on OpenELEC and RaspBMC, it's a media centre that allows you to play just about any kind of video file. With years of development, it has every convenience for playing.

X
'X' is the short name for the X Window Manager. It's an open source product that presents a graphical user interface (GUI) to a user. It's also known as 'X Windows' or 'X11'.

About the mag

The magazine for the GNU generation

Written for you

Linux User is the only magazine dedicated to advanced users, developers & IT professionals

In-depth guides & features

Written by grass-roots developers & industry experts

4.5GB DVD every issue

Four of the hottest distros feature every month - insert the disc, reboot your PC & test them all!

subscribers to...

LinuxUser & Developer™

Try three issues for £5 in the UK*
or just $7.85 per issue in the USA**
(saving 54% off the newsstand price)

For amazing offers please visit
www.imaginesubs.co.uk/lud
Quote code ZGGZIN

Or telephone UK 0844 249 0282 overseas +44 (0) 1795 418 661

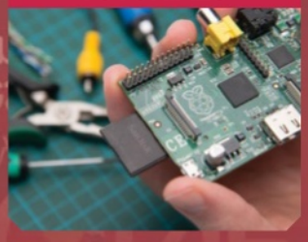
RasPi

Digital magazine for Raspberry Pi makers

Free sample issue


PROJECTS


INTERVIEWS


TUTORIALS


EXPERT ADVICE

Download the **free** app on iTunes for iPhone and iPad now

www.linuxuser.co.uk/RasPiCode

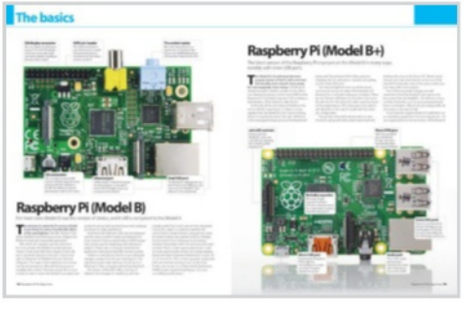
facebook.com/LinuxUserUK

#raspimag

Raspberry Pi for Beginners

All you need to know to get started with Raspberry Pi

Learn Pi basics, how to program, and create simple projects



Basics

Connect your Pi and learn to use the best open-source software for it

Projects

Use your Pi to take videos, detect motion, play games and much more

Programming

Get to grips with Scratch, Sonic Pi and Python programming

ip IMAGINE PUBLISHING
DigitalEdition
GreatDigitalMags.com
VOLUME 1 SECOND REVISED