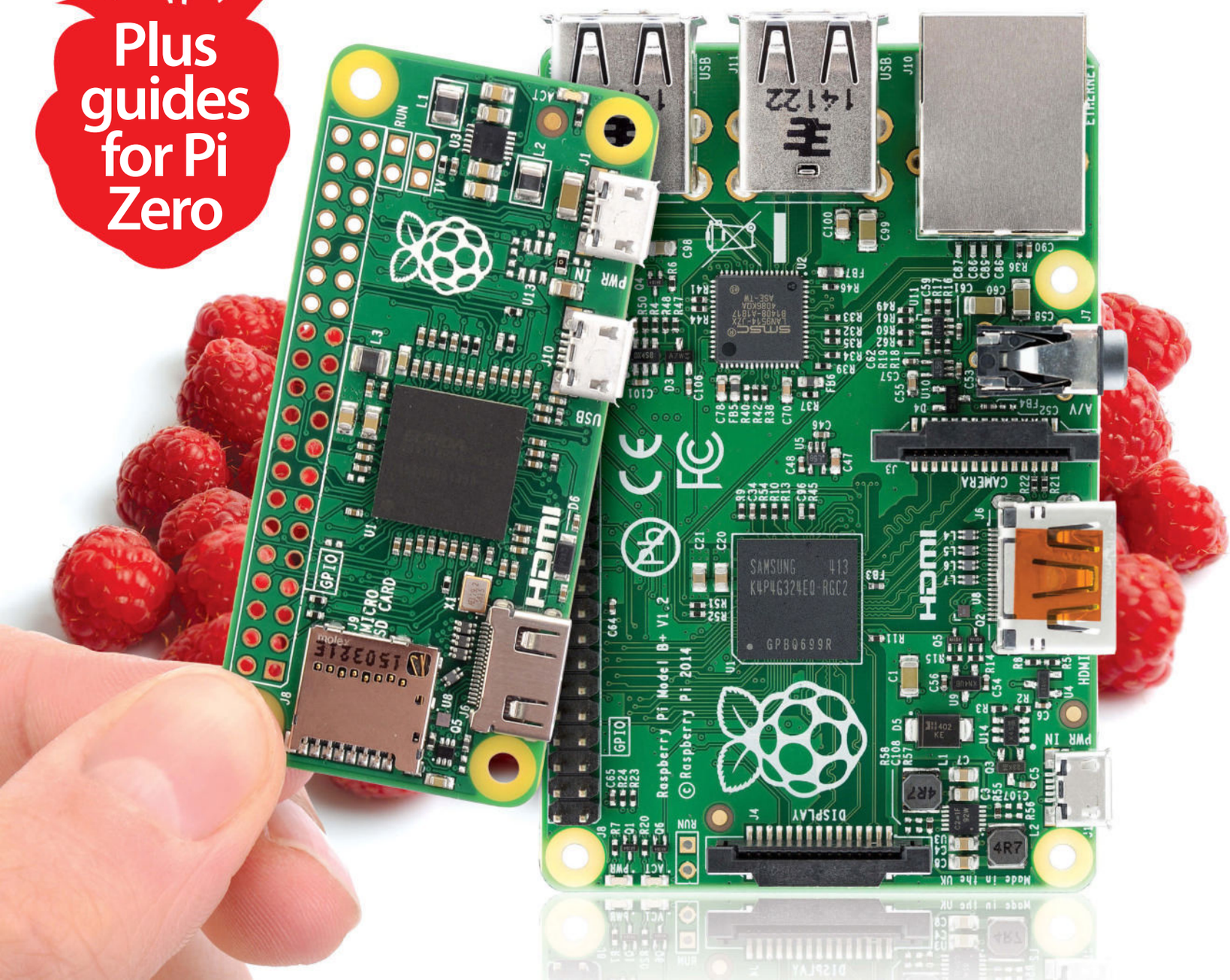
NEW

Raspberry Pi for Beginners

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

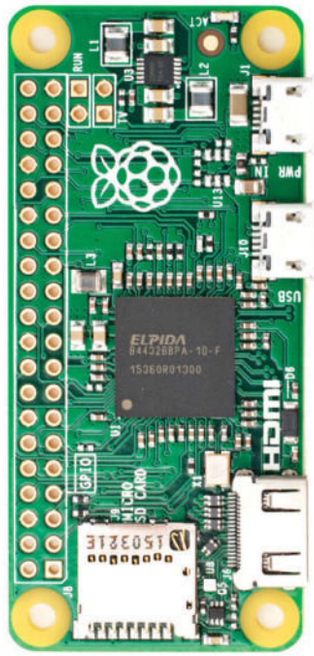
100% independent
Follow project tutorials
Understand Pi essentials

Plus guides for Pi Zero



Welcome to Raspberry Pi for Beginners

During 2015, the latest – and smallest – Raspberry Pi device was released, the Pi Zero, costing under a fiver, and smaller than your bankcard. It was truly one of the most innovative products released, and has begun to change the front of Pi programming. Within the first 24 hours of its release, the Pi Zero had 'sold out' with over 20,000 units shifted in that small time. Such is the strength of feeling towards this impressive - and low-cost - educational tool. For those stepping into the market for the first time, Raspberry Pi for Beginners will help you select the model for you, set up your new device and take your first steps with your new gadget. Once you have mastered the basics, we even provide 40 pages' worth of fun practical projects to sink your teeth into, as well as offering guidance on Python and Scratch programming. Perfect for adults and children alike, join the Raspberry Pi community and get excited about the power of the Pi!



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
Jen Neal

Senior Art Editor
Greg Whitaker

Designer
Perry Wardell-Wicks

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, 5 Churchill Place, Canary Wharf, London, E14 5HU
Tel 0203 148 3300 www.marketforce.co.uk

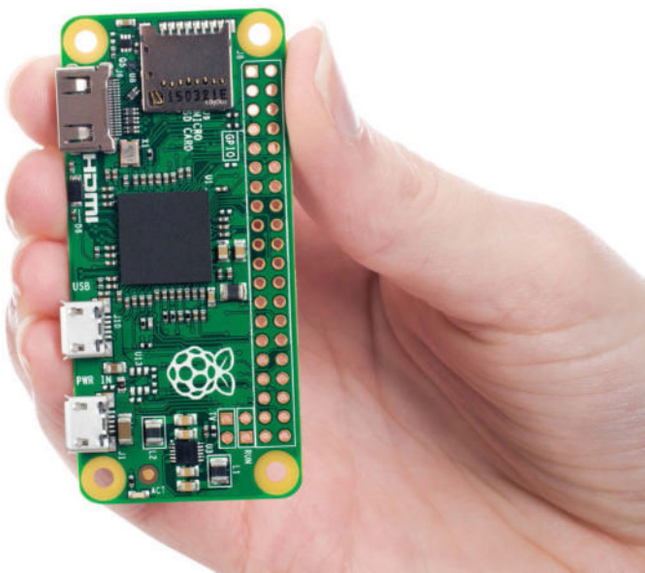
Distributed in Australia by
Network Services (a division of Bauer Media Group), Level 21 Civic Tower, 66-68 Goulburn Street, Sydney, New South Wales 2000, Australia Tel +61 2 8667 5288

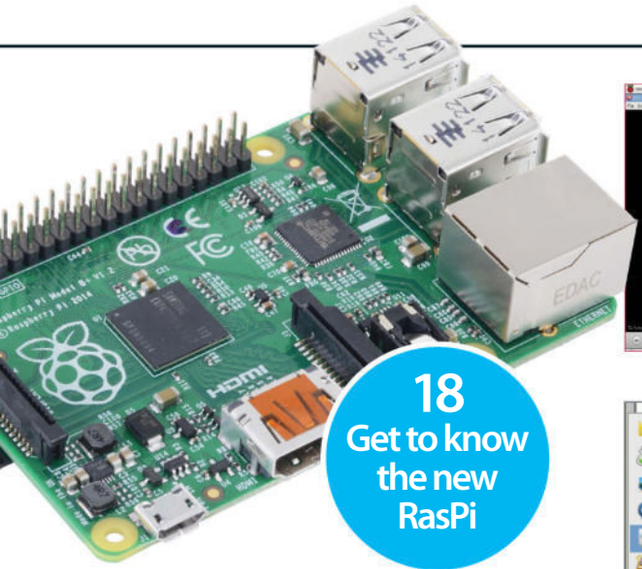
Part of the
LinuxUser & Developer
bookazine series

IMAGINE PUBLISHING

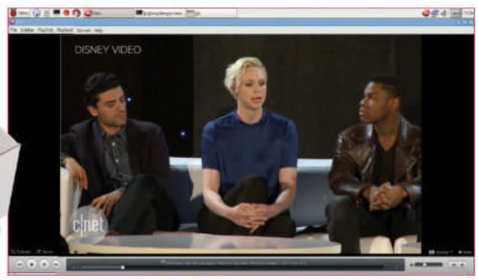
Contents

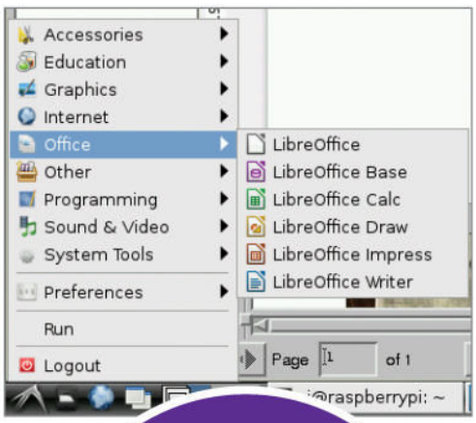
The basics





18 Get to know the new RasPi





156 Glossary Decode the Pi jargon

Programming

Troubleshooting

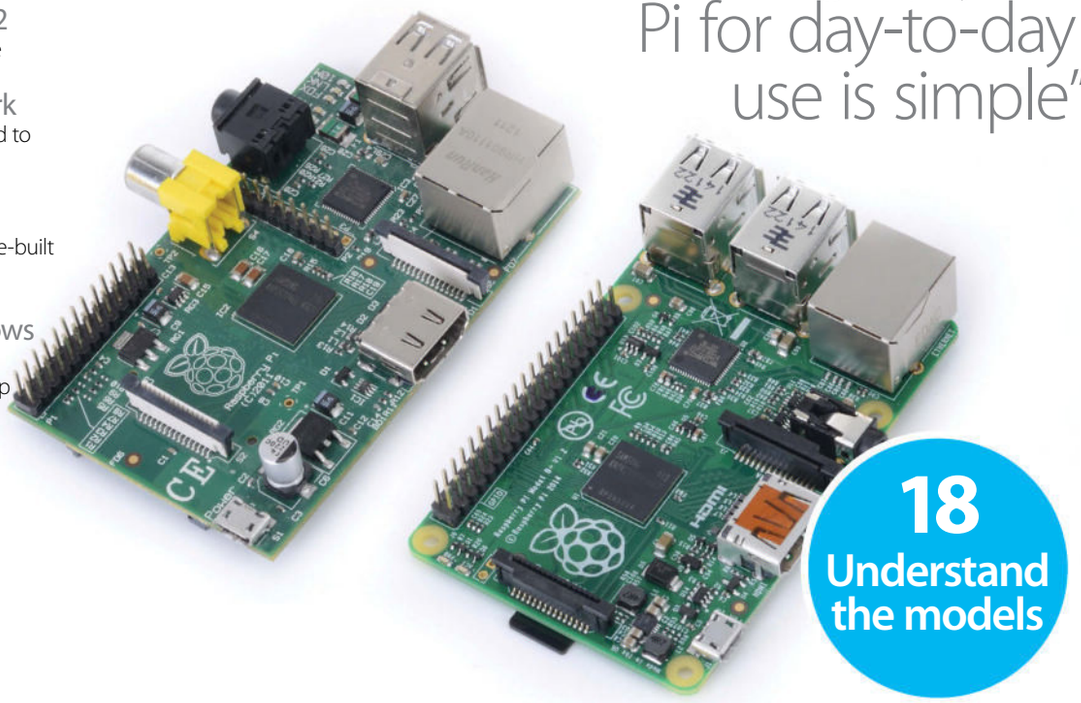
Projects

"You're far more likely to be overwhelmed by the choice that the Raspberry Pi's capabilities afford to you than stuck for interesting things to do"

The basics

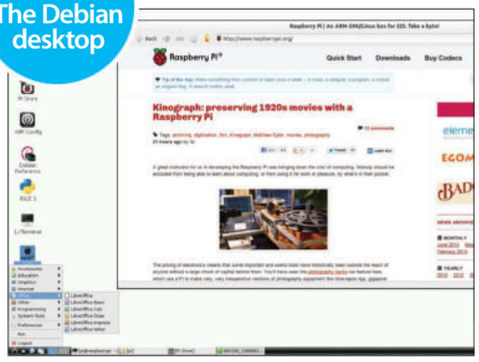
Set up your Raspberry Pi, install the latest distros and get started with Raspbian

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

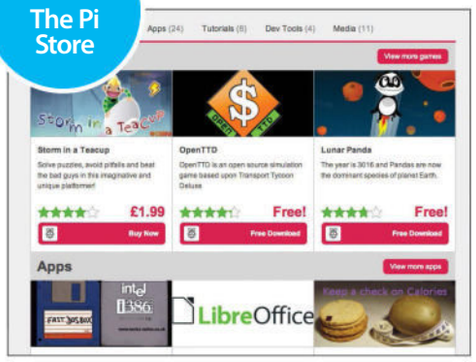


18 Understand the models

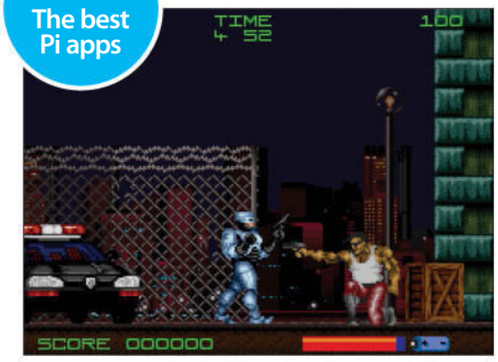
36
The Debian desktop

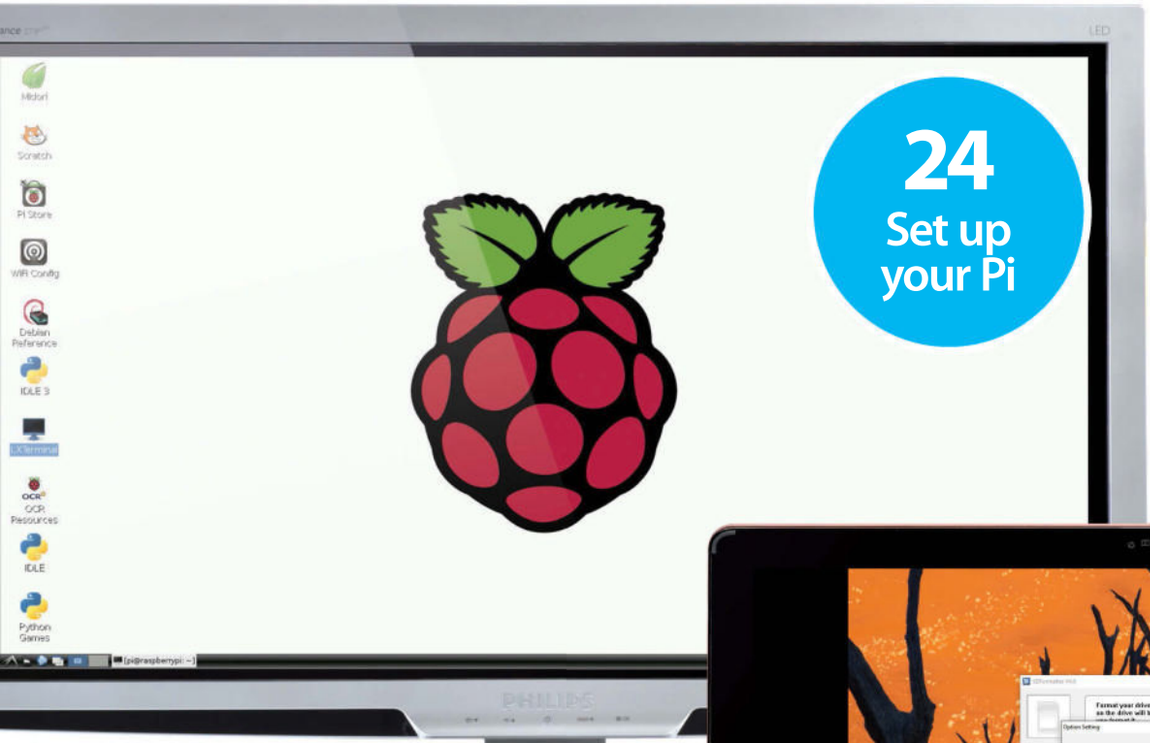


42
The Pi Store



44
The best Pi apps

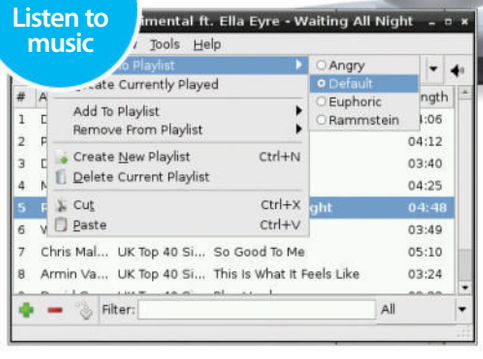




24
Set up your Pi

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

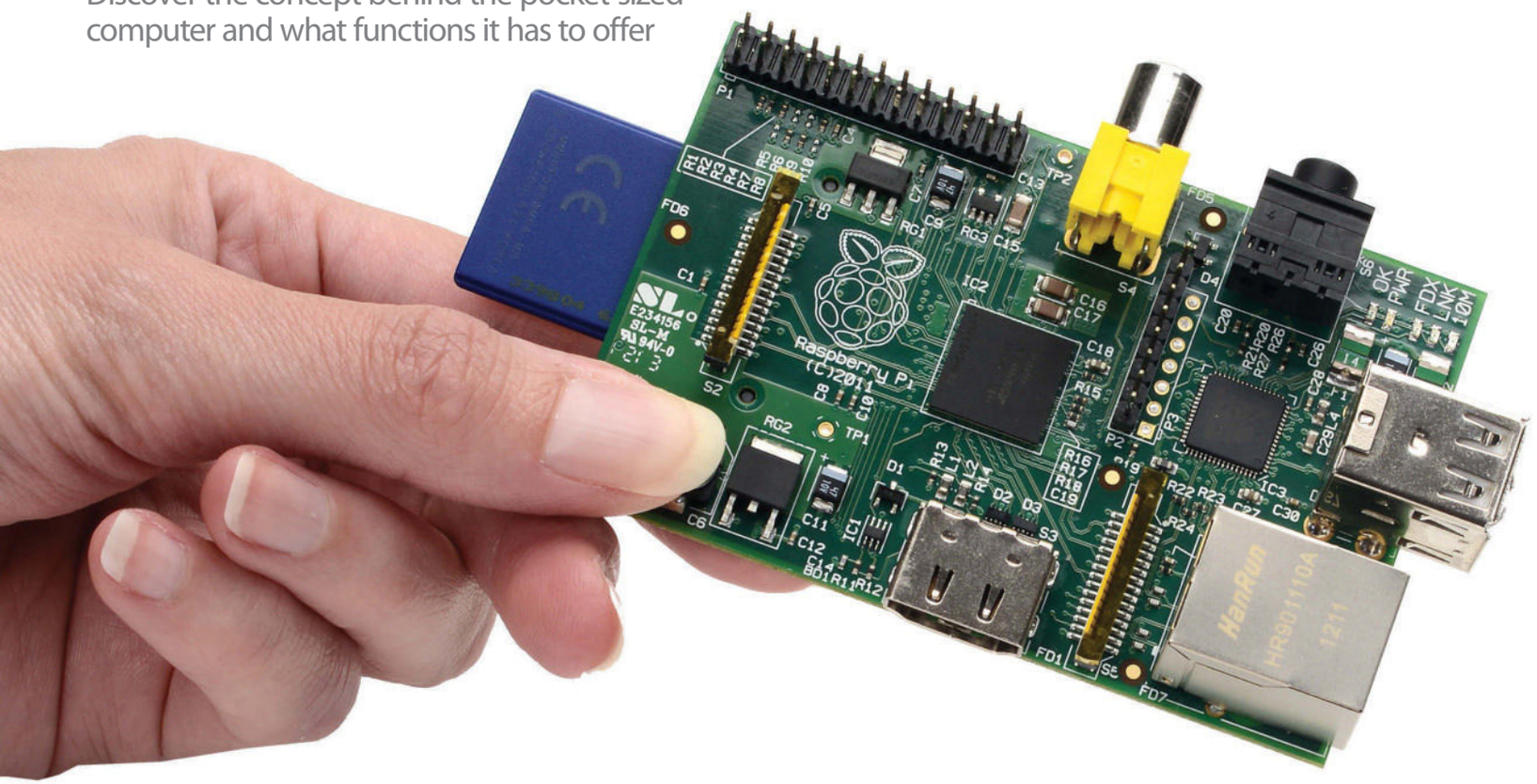
54
Listen to music



35
Install Ubuntu

What is Raspberry Pi?

Discover the concept 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 Micro, Spectrum and Commodore 64 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 on with the, lower-capacity RAM version, but cheaper, Model A.

Fast-forward to 2016 and the expanded Raspberry Pi family now includes the improved A+ and B+ models, the powerful Compute Module, the Raspberry Pi 2 model B, and the tiny Pi Zero.

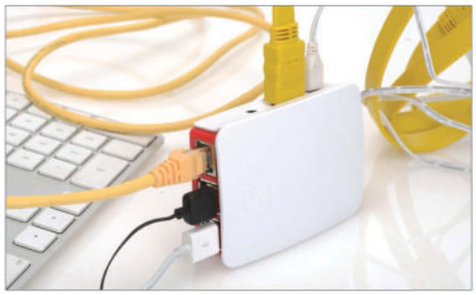
The basic concepts behind the Raspberry Pi were to make it as affordable as possible, supply only the basics and provide it with a programming environment and hardware connections for electronics projects. The Pi runs a modified version of Linux called Raspbian, based on the Linux distribution called Debian. Raspbian runs directly on an SD card and provides a graphical interface for using the operating system, as well as a terminal application for accessing the command line.

Connecting it up

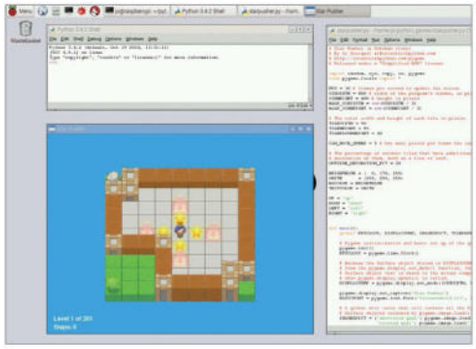
The Pi comes in both starter kits and as a bare-bones circuit board, for which you can buy all the extras needed to make it fully functional. The first thing that you need is an SD card to act as storage for the operating system and any software you want to install. Although it will work on a 2GB or 4GB card, we recommend that you use an 8GB card as a minimum. It is always best to use the official Raspberry Pi SD so that it is more reliable. Unless you bought one with NOOBS (New Out Of Box Software) on it from the Pi Swag Store, you'll need to download NOOBS from the Raspberry Pi website, format your SD card and move NOOBS onto it using another computer, and then install an operating system using NOOBS. Before you boot the Pi to be guided through NOOBS, though, you'll need to wire it up. You'll need a USB keyboard and mouse, either a HDMI or an analogue video/audio cable, and a micro USB power supply – the B+ and Pi 2 require 1.8A at 5V. To get your Pi online so that you can download updates and software, you'll need either a Wi-Fi dongle or an Ethernet cable – the dongle will need setting up, so if this is your first boot then go with an Ethernet cable. The final element is to get a fancy looking case to put the Pi in (Fig 1).

Programming the Pi

There are two main programming languages supplied by default with the Pi: Scratch and Python. They both have shortcuts in the start menu at the top-left of the screen. Scratch was designed by Lifelong Kindergarten Group at the MIT Media Lab as a first-step in programming. It uses tile-based plain-language commands that can be strung together without you having to worry about syntax. Commands are split up into easy-to-understand groups and simply 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 main programming languages supplied with the Pi. Scratch is for beginners to get started, while Python is a little more advanced

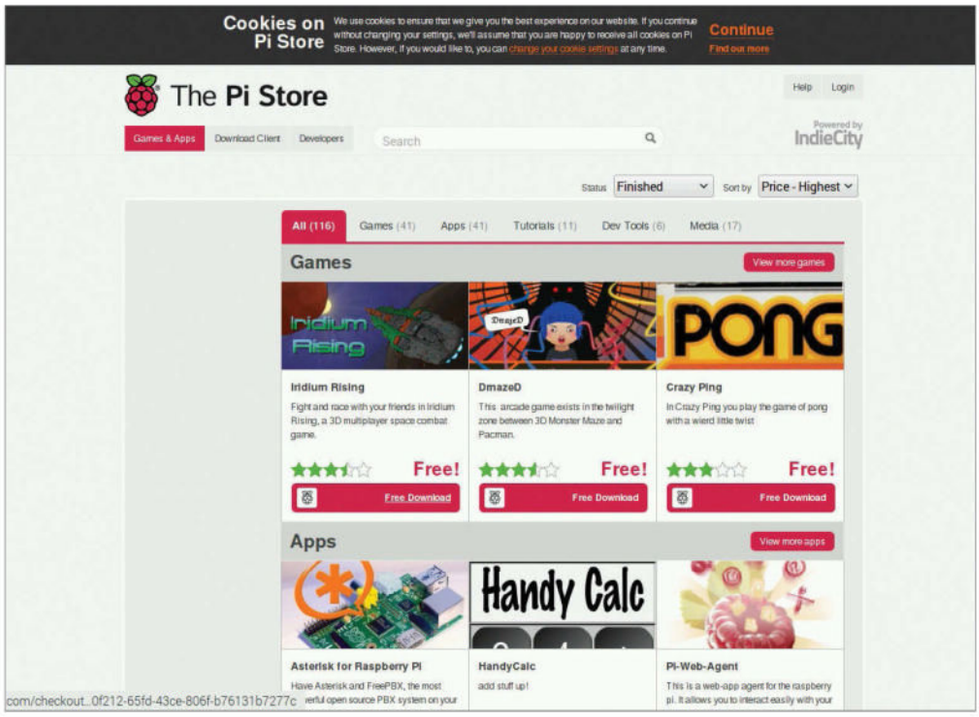
The other language is Python v3.2.3 (Fig 2) which starts with a Python Shell. Python is an interpreted language, where commands are read and implemented one line at a time. The high level commands and data structures make it ideal for scripting. The previous version of Python is still quite popular and has more development aids, so is also supplied. The shortcuts 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 website (Fig 3). This houses a collection of games, apps, tutorials, development tools and media, and is well worth browsing through for inspiration when you begin to tackle your very first programming projects.

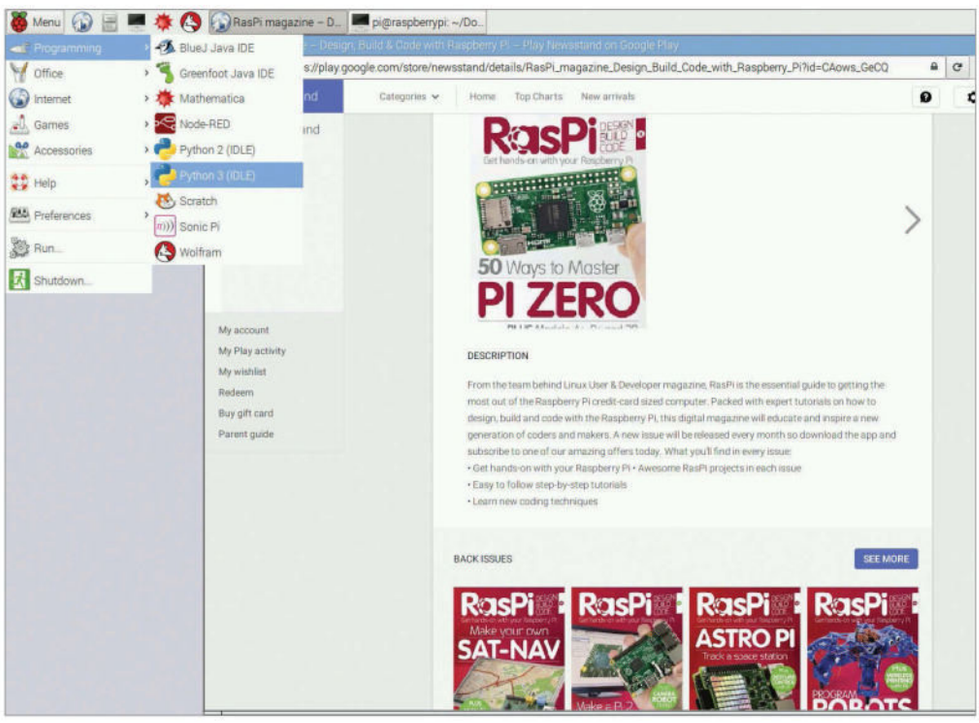
You'll first need to create an account to log in. The content of the Pi Store can then be downloaded and installed for use. Some create shortcut icons, others use the Menu, but many can only be run directly from the Store interface itself. This has a My Library tab where downloaded content can be managed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

"Raspbian runs directly on an SD card and provides a graphical interface for using the operating system, as well as a terminal"



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

What is Linux?

Linux operating systems are everywhere – computers, smartphones, your Pi. But what is Linux?

Linux is the operating system (OS) used for your Raspberry Pi. Its role is exactly the same as Windows, OS X on the Mac, Android (in fact, Android is based on the 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 a misnomer. It's not 'one operating system' in the same way that Windows 10 or OS X Yosemite 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 – this is pure Linux. 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 your keyboard, monitor or Wi-Fi dongle – will have to go through the kernel.

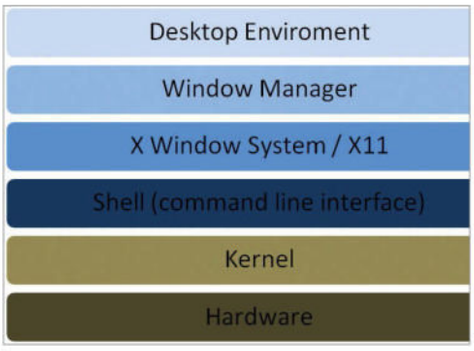
The shell: A good old-fashioned command-line interface (Fig 2). There's nothing you can't do here, from installing software to viewing system resources and scripting common tasks. It can look daunting at first, but you'll soon realise it's not so scary.

Desktop environment: Of course it's no fun just looking at text all day. This is where the desktop environment (Fig 3) comes in; the graphical interface that 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 to get exactly the look that you want.

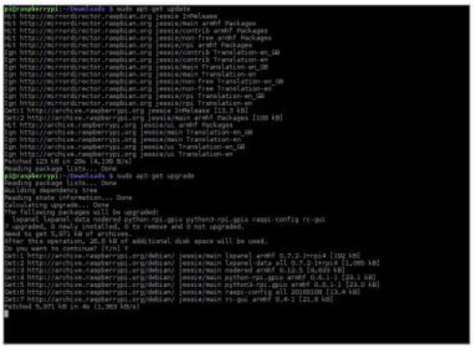
Applications: Not part of the OS as such, but a key part of any Linux installation. Whether it's an office suite or a media player, you'll find it for Linux.

Set yourself free

If there's one reason to use Linux, it's cost – there is none! Welcome to the world of free and open source software (FOSS). For most users, it 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 under a special 'copyleft' license called the GPL, which stipulates that anyone releasing FOSS must make the original source code available to the user. This model has encouraged development and sharing in a way that proprietary software cannot.



■ 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

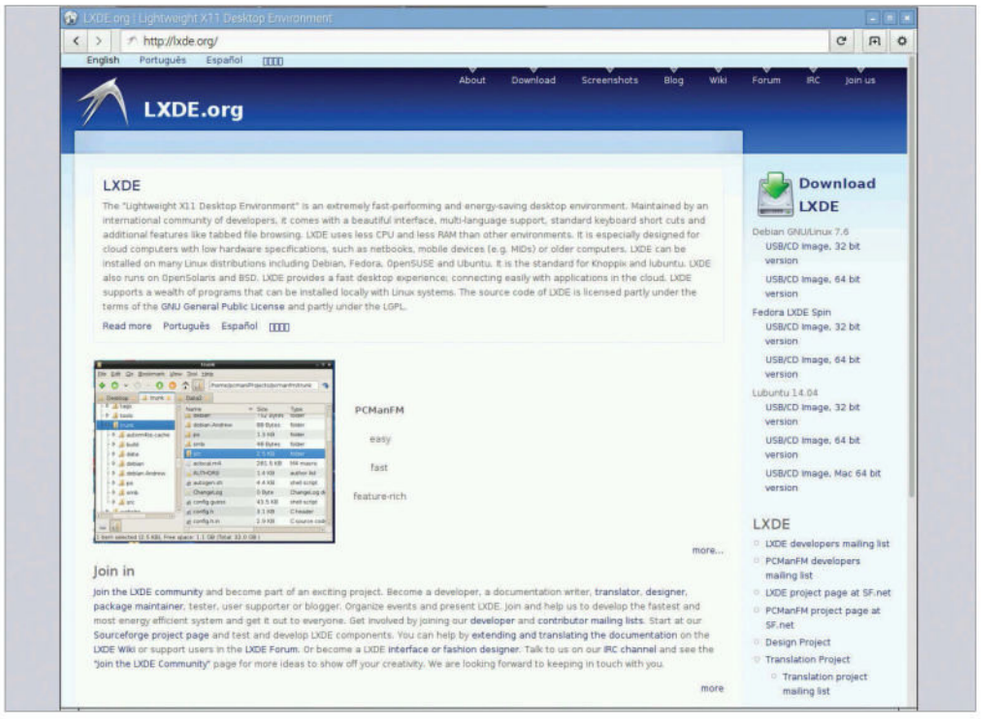
With open software, you can look inside the source code, play with it, change it and make a difference to the whole community. Fed up waiting for a new feature in your favourite application? With a bit of knowledge, you can add it yourself. Many of the people starting out with the Raspberry Pi today 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, OSs for older hardware, web servers, media servers, to development environments, there's always a choice of distros to meet your needs.

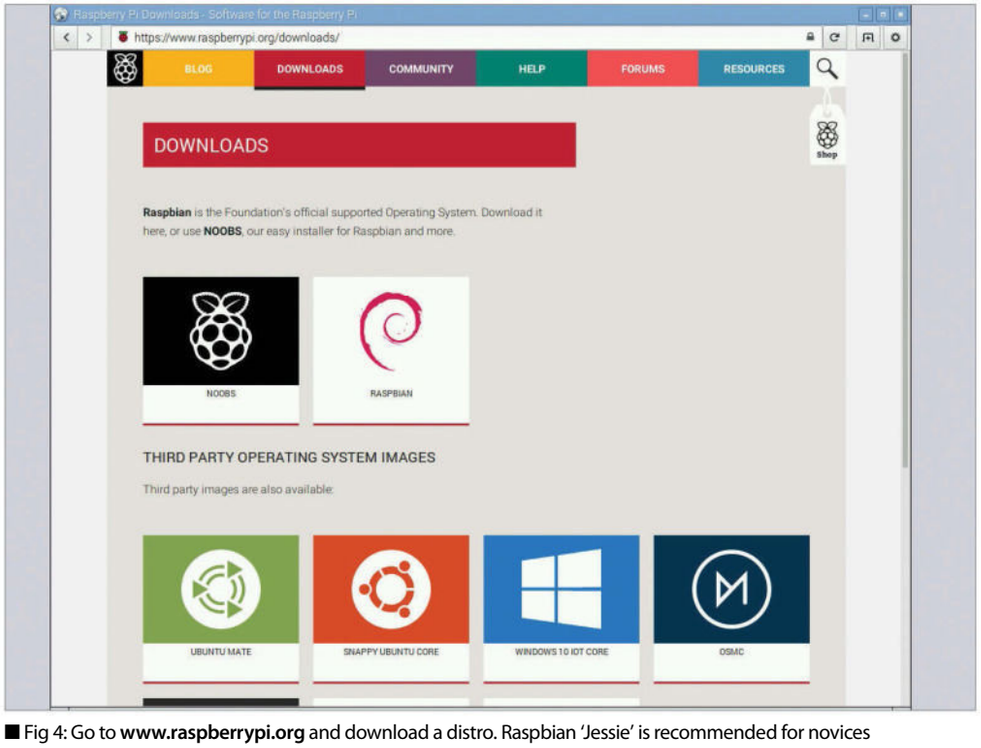
As a Raspberry Pi user, you'll have a more limited choice of distribution due to the ARM processor used, but the number is growing all the time. Most users will begin with Raspbian, a modified version of Debian that comes bundled with a wealth of applications and development tools. The choice of lightweight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

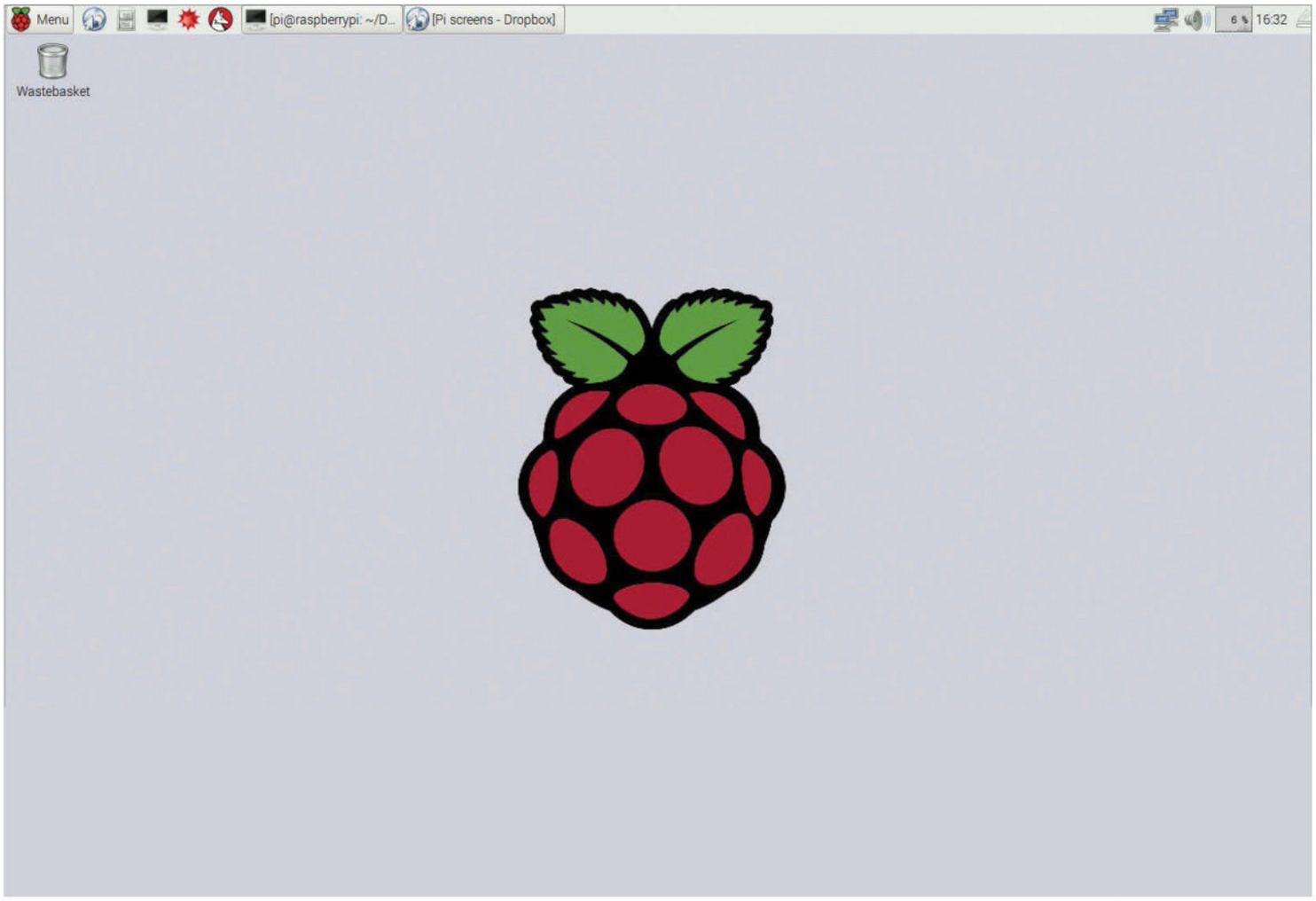


■ Fig 3: Linux has many desktop environments. Anyone who has used Windows or OS X will feel right at home

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



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



Choose an operating system

The Raspberry Pi has a number of operating systems to choose from. Here are some of the best…

The Raspberry Pi has captured the hearts and imaginations of people since its launch in 2012. There's rarely a day gone by where we haven't heard of some wonderful hardware or software project from the ever-growing community of Raspberry Pi enthusiasts. Just looking through the pages of this book will show you 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 is capable of running a variety of different Linux operating systems. These – known as distributions, or distro for short – are as varied as the many Raspberry Pi projects, and each offers a different look and feel, as well as different functionality. Let's take a look at some of the best distros available for your Pi, all available via **www.raspberrypi.org/downloads**.

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.

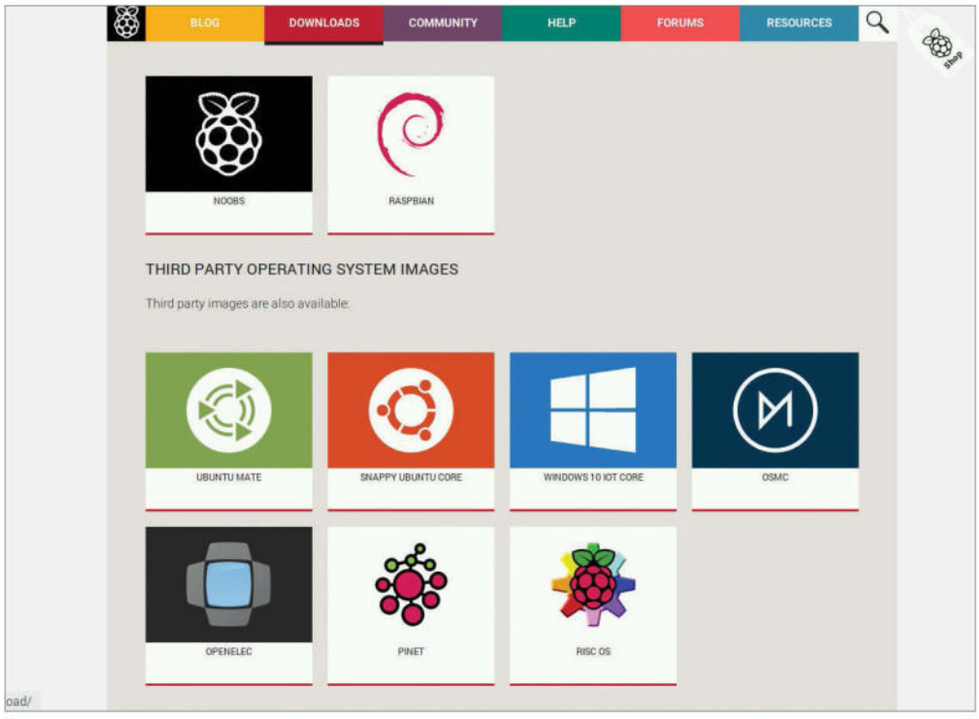
The Raspbian operating system itself is very well constructed and developed. Running LXDE (Lightweight X11 Desktop Environment) as the desktop environment makes this a considerably streamlined operating system, ideally suited to the limited system resources of the Raspberry Pi.

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.

Arch Linux

Arch is a Linux distro that has been active for roughly 14 years. It's a fantastic operating system and prides itself on its minimalism, code correctness and elegance. The 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 resource usage.

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,



■ Fig 1: The officially recommended distros can be downloaded from the official Raspberry Pi website

to some degree, in the setting up of the operating system, Arch will boot to the command prompt and expect you to install everything manually (Fig 2).

Ubuntu MATE and Snappy Core

Ubuntu, based on Debian (just like Raspbian), is the most popular Linux distribution for regular desktop and laptop computers. It is just as simple to use as Raspbian and is incredibly well-supported by its parent company Canonical.

For the Raspberry Pi, it comes in two flavours: Ubuntu MATE and Snappy Ubuntu Core. Ubuntu MATE is basically regular Ubuntu but with a different desktop environment – it uses the popular MATE desktop rather than Canonical's own Unity desktop. Snappy Ubuntu Core is designed to run on Internet of Things (IoT) devices, so it's a stripped-down version of the OS that is based on transactional updates. If you are developing a product that is powered by the Raspberry Pi, it's well worth a look.

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

Which distro is best?

With so many options on offer, choosing can be hard

Clearly the ripest fruit to begin with is Raspbian, as it provides an ideal environment to learn and experiment in.

Raspbian is 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.

A great next step is often Ubuntu MATE, since Ubuntu is the most popular Linux distribution used on full-size desktop and laptop computers. It's also easy to use.

People with specific goals in mind, such as running a Kodi-based media server, will want to check out OSMC and OpenELEC.

Many users will later progress to Arch after spending time learning how Linux is developed and how it works under the hood. It requires more advanced understanding, but is the most customisable platform on offer.

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

OSMC and OpenELEC

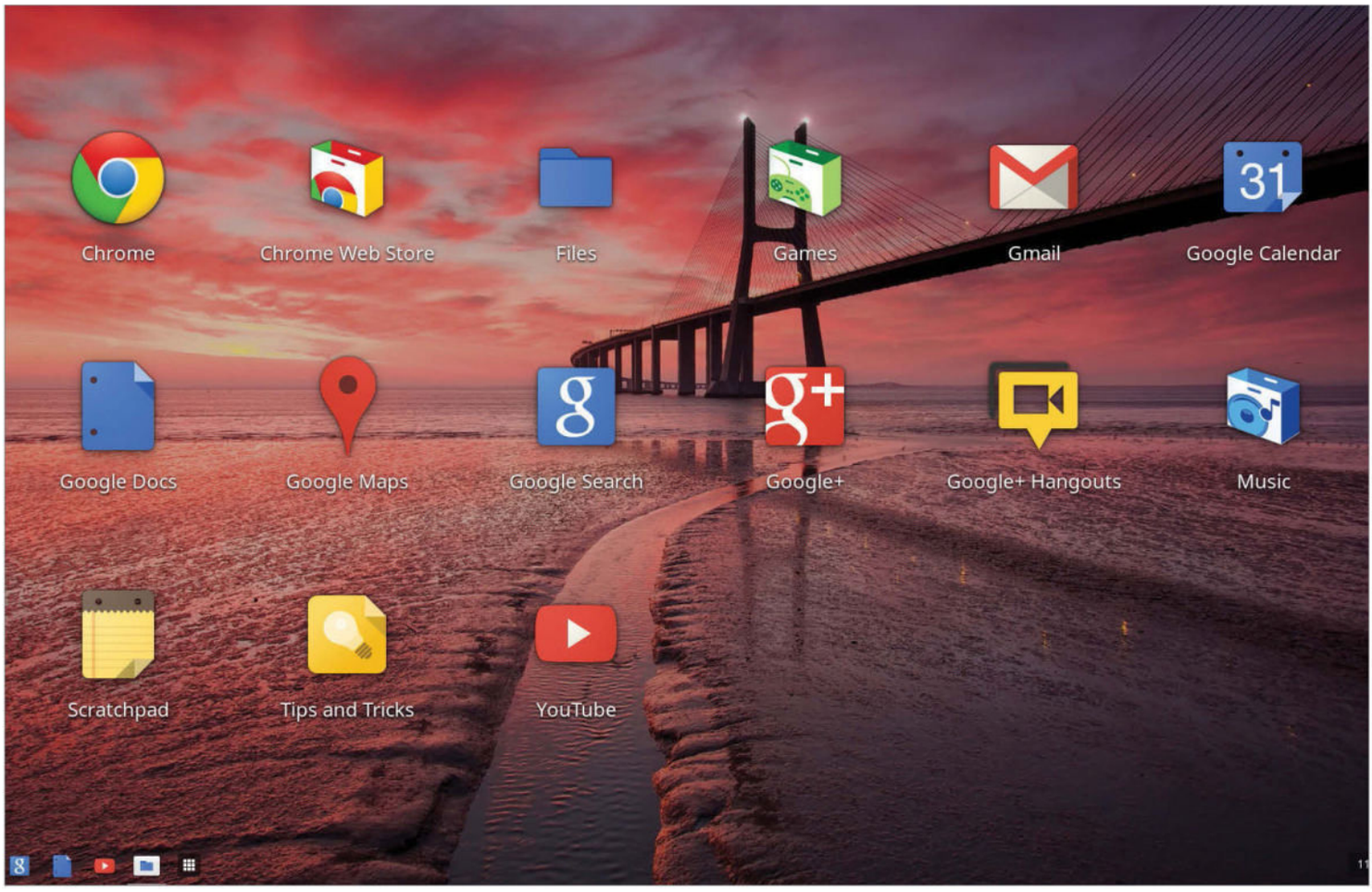
Many people use their Raspberry Pis as media servers – basically, a device to connect to your home network that can relay audio and video files around the house. You could use it to automatically download internet TV and podcasts, to stream your movie collection onto the big screen in your house, and to handle music playback over your sound system. OSMC is a great distro that is specifically optimised for this task. Another one worth investigating is OpenELEC, which is an older project that has an incredible amount of support in the community. Both are based on the Kodi media player, formerly known as XBMC.

Windows 10 IOT Core

Similar to Snappy Ubuntu Core, Windows 10 IOT Core is Microsoft's version of a stripped-back operating system designed to be used in embedded devices and IoT tech. It is, of course, not a Linux distro – it's a kind of Windows – but with Microsoft embracing open source technologies and practises in recent times, it's not to be ignored.

RISC OS Open

Designed in Cambridge and first released in 1987, RISC OS can trace its roots back to the team that developed the ARM processor – in fact, it's a direct descendant of the OS used on BBC Micros. RISC OS Open is a port designed for the Raspberry Pi, and is quite a charming, retro offering. If you can imagine the look and feel of a desktop from the early Nineties, you won't be far away from this version of RISC OS. Admittedly, it requires some patience since it isn't Linux, so things are done slightly differently.



Use other operating systems

Discover the myriad projects that are extending the Pi's OS choice

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. Many of these have since been absorbed into NOOBS or linked directly from the official Raspberry Pi website – things like Ubuntu MATE, Fedora, media centre distros like OSMC. There are far more operating systems for your Pi than just these, though. There is a well-maintained list of them over on the eLinux wiki page (**elinux.org**).

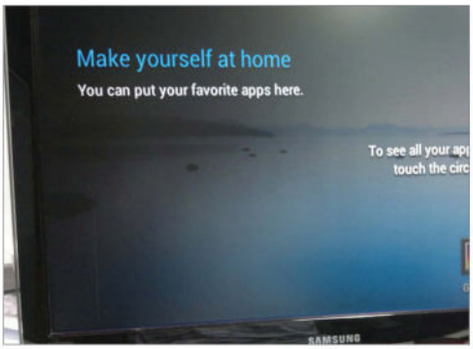
Two which have proven particularly problematic are Android, which is the operating system used on smartphones and tablets, and Chromium OS, the development operating system forming the base of Chrome OS, which powers Google Chromebooks. There are examples of each available to download and test on the Raspberry Pi. However, they are extremely buggy and are considered to be experimental by the developers and the community as a whole. That doesn't mean they are impossible to install or experiment with, but as a novice user, you will most likely be stretching your working knowledge of the Raspberry Pi to its limits! Still, don't be afraid to go for it and learn from the experience – that's how everyone starts!

Android on Raspberry Pi

Android is a Google-funded and well developed Linux-based operating system designed for mobile devices such as smartphones and tablets. Android comes in many versions, 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 quite unusable by anyone's standards when installed on the Raspberry Pi (Fig 1), and the Android on Pi project has since stalled.

However, 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. Be warned: it's painfully slow, and using a mouse is nowhere near as intuitive in Android as using touch-screen technology. It can, however, be a highly educational experience to try debugging yourself.



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

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.

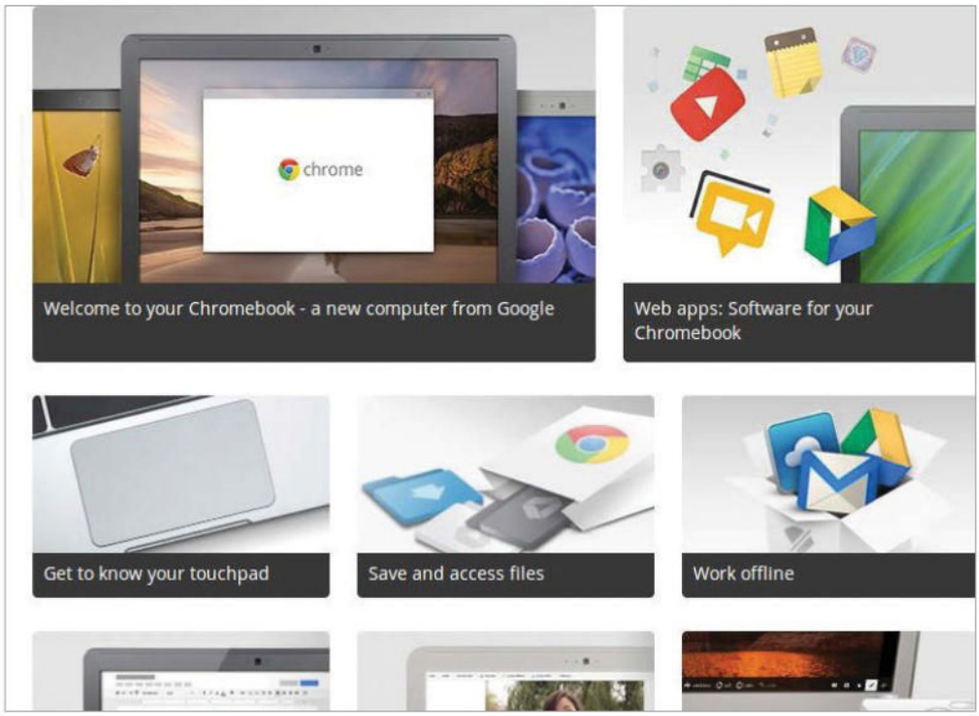
Liam McLoughlin, aka Hexxeh, initiated the project to try to bring the development Chromium OS to the Raspberry Pi. Unfortunately, though, he no longer has the time to dedicate himself to the project. To quote his blog entry: "I'd still like to see Chrome on the Raspberry Pi, but I don't have time for 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. But if you're interested, why not contribute to the project yourself?

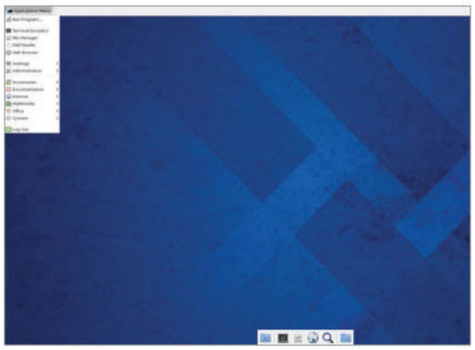
Getting Chromium OS to run on the Raspberry Pi is a little more complex than that of the Android build. For starters, there are a few prerequisites necessary; however, Hexeh's post located at **blog.hexxeh.net/?p=328117867** should get you on track.

More to explore

Beyond these two experimental, difficult, yet still desirable operating systems, there is a whole world of distros out there for you to explore. The most comprehensive list of available distros for the Raspberry Pi is maintained at elinux.org/Rpi_Distributions. Take a browse through the page and you'll find a host of media centre distros beyond OSMC, such as Xbian and RasPlex; builds for popular Linux distros such as Bodhi, Slackware and Gentoo; images that re-create classic operating systems such as that used by the Commodore 64; task-specific distros such as IPFire, the open source firewall distribution; and even news on upcoming projects such as openSUSE, Firefox OS and Puppy Linux for the Pi. More are being added all the time, as the clever folk programming on the Pi work hard to adapt their favourite tools for the tiny computer.



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



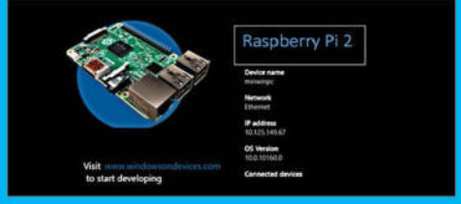
■ Pidora – a remix of the Fedora distribution – is a very popular and very useable alternative OS for your Pi

■ Kano OS was developed specifically for an educational kit that teaches children how to assemble and use the Pi

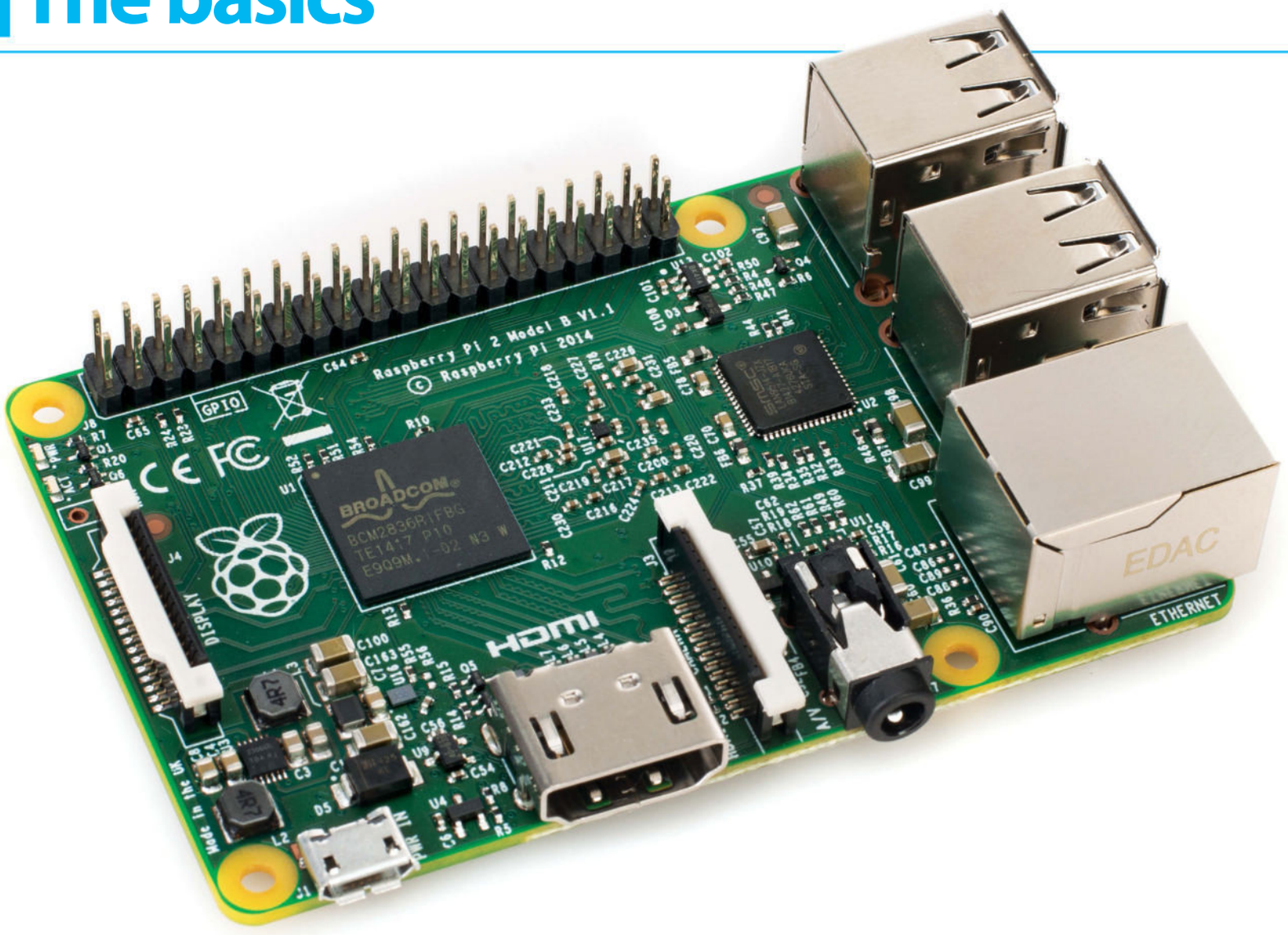
Raspberry Pi 2

The newer CPU opens up even more possibilities

The most important difference between the Raspberry Pi 2 Model B and the preceding models, in terms of which operating systems you can run on it, is the fact that the Pi 2 contains an ARMv7 core. All of the earlier versions used an ARMv6 core, which limited their potential to a certain extent. With ARMv7, it is possible to run more modern operating systems such as Ubuntu and Windows – this is why the Ubuntu MATE and Windows 10 IOT Core distributions only surfaced following the release of the Raspberry Pi 2, and this is also why they won't work on your old Model A or B.



Going forward, there is lots of potential to port other operating systems across to the Pi 2, so if you are keen to stay on the cutting edge then it is worth investing in the newer model. Plus, it makes a great change from Raspbian!



Raspberry Pi 2 Model B

The biggest surprise of 2015, this is the true second iteration of the greatest education and hobby tool since Arduino

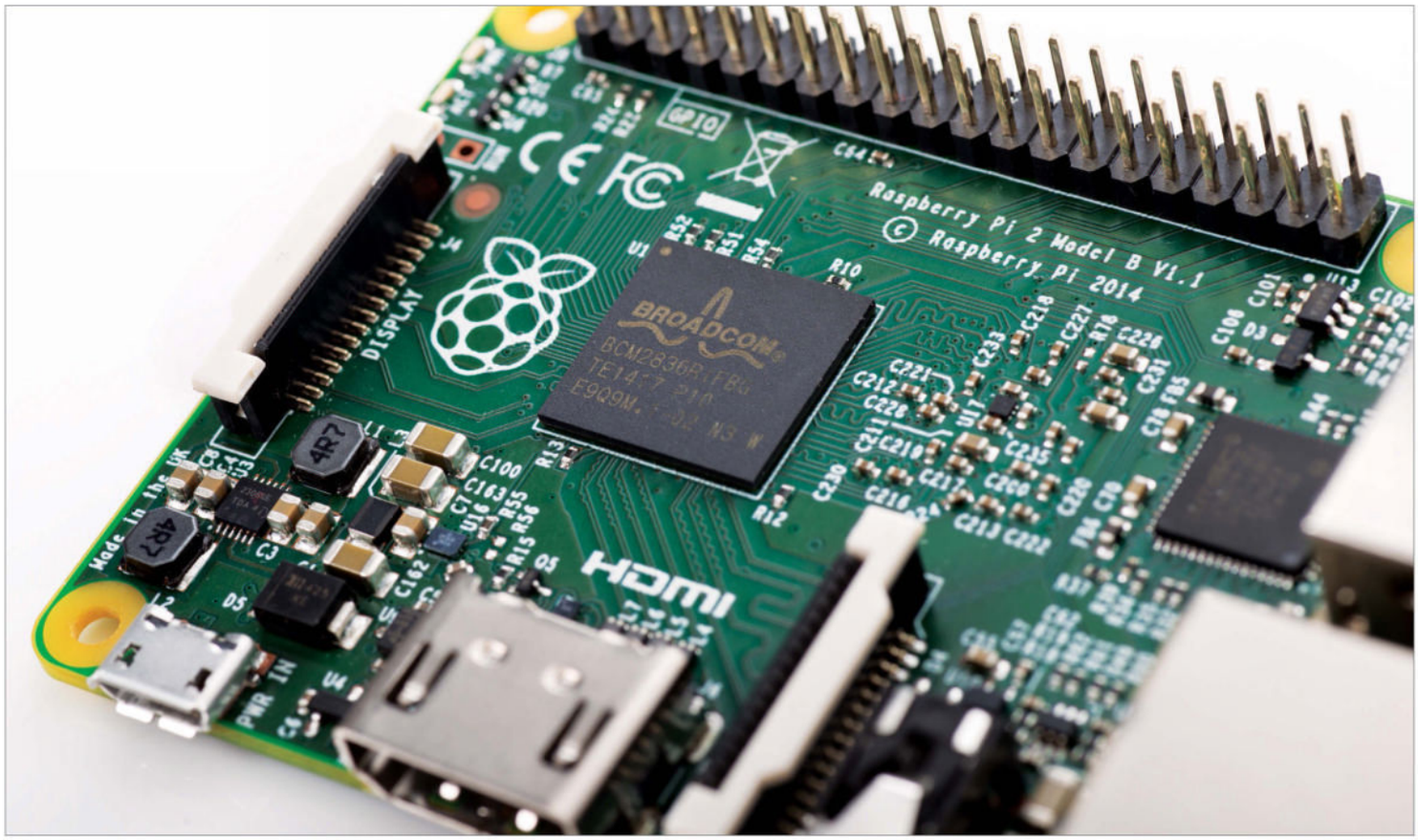
The Raspberry Pi family has certainly grown over the last few years and currently there are five devices to pay attention to: the Raspberry Pi 2 Model B, the Raspberry Pi Models A+ and B+, the Raspberry Pi Compute Module and the Raspberry Pi Zero. Anything older than that – i.e. the original Models A and B – are now ancient history. Most of you will want to use the Pi 2, which is the direct successor to the Model B+ (covered overleaf). As for the rest? In a nutshell: the Compute Module is for experienced engineers; the A+ (overleaf) is for anyone wanting to build small, embedded projects; the B+ is still a valid board, but if you are looking to buy then go straight for the Pi 2; the Zero is a super-small, bare-bones board that's great for wearable and other embedded projects – things like cosplay accessories or NES controllers that have been hacked to fit a Zero, which can run RetroPie.

We'll be honest, when the Raspberry Pi 2 hit our desk we were very excited to crack it open and try it out. From what we had been told this was basically the Raspberry Pi everyone had ever wanted, at least in terms of power. It was a bit of a have-your-cake-and-eat-it moment though, as we hooked up the board that was essentially a Model B+ and began using a very familiar Raspbian layout.

It worked fantastically well; while performing normal computer actions there was none of the classic slowdown the Raspberry Pi used to get. We could upgrade the system and comfortably do other tasks such as web browsing or even word processing. It's simple yet appreciated, and on the surface that's really about it for the Raspberry Pi 2. There is no real killer feature of the updated Raspbian that we can point to that illustrates the new Pi's full abilities.

However, this is frankly a great thing. The Raspberry Pi Foundation's mission has always been education, and with the number of excellent tutorials and software that already exist for the Raspberry Pi, making the Pi 2 very different would make the last three years of work obsolete.

Before we talk about that though, let's bring some context to exactly what the changes are to the Raspberry Pi 2. As we've said, and as is evident from the photos, it looks exactly the same as the Raspberry Pi Model B+. In our interview with Eben Upton, co-creator of the Raspberry Pi and co-founder of the Pi Foundation, it was mentioned that the redesign of the B+ was done to facilitate the changes planned for the Raspberry Pi 2. Specifically in this case, more room for the larger BCM2836 chip that powers the Pi 2. This is a modified version of the BCM2835 that was on the



■ Above: Aside from that powerful new BCM2836, it's basically a B+

original Pi, but it now has a much beefier, quad-core, ARMv7 processor with 512Kb of cache. At the time of writing, the Pi Foundation reckons it performs six times better than the original and they have been testing out some overclocking as well.

There's also more RAM, 1GB located on the rear of the board rather than PoP with the BCM chip. This has resulted in better heat dissipation, and overall the extra resources result in a far better Raspberry Pi as we've already described.

That's about it though for additions – there's no Wi-Fi as many people may have hoped for, but the Foundation has developed its own, very cheap dongle that it claims outperforms most current dongles recommended for the Pi.

Although you're getting basically the same functionality, the greatly improved power and experience makes the Raspberry Pi 2 a perfect successor and replacement for the original Raspberry Pi. While it's going to take a few months for some of the distros and packages to properly catch up with the new Pi, all the Python project code and add-on boards will work fine. All of the existing setup tutorials and project tutorials will work just as they did before, allowing people to replace their Pi in order to get a better experience without sacrificing the potential at all.

It's also just nicer to use, and this is what a lot of hobbyists have been wanting for a long time. The extra power makes it completely viable for a lot more day-to-day tasks and projects as well, opening up the Pi to many more possibilities – all for the same extremely low price.

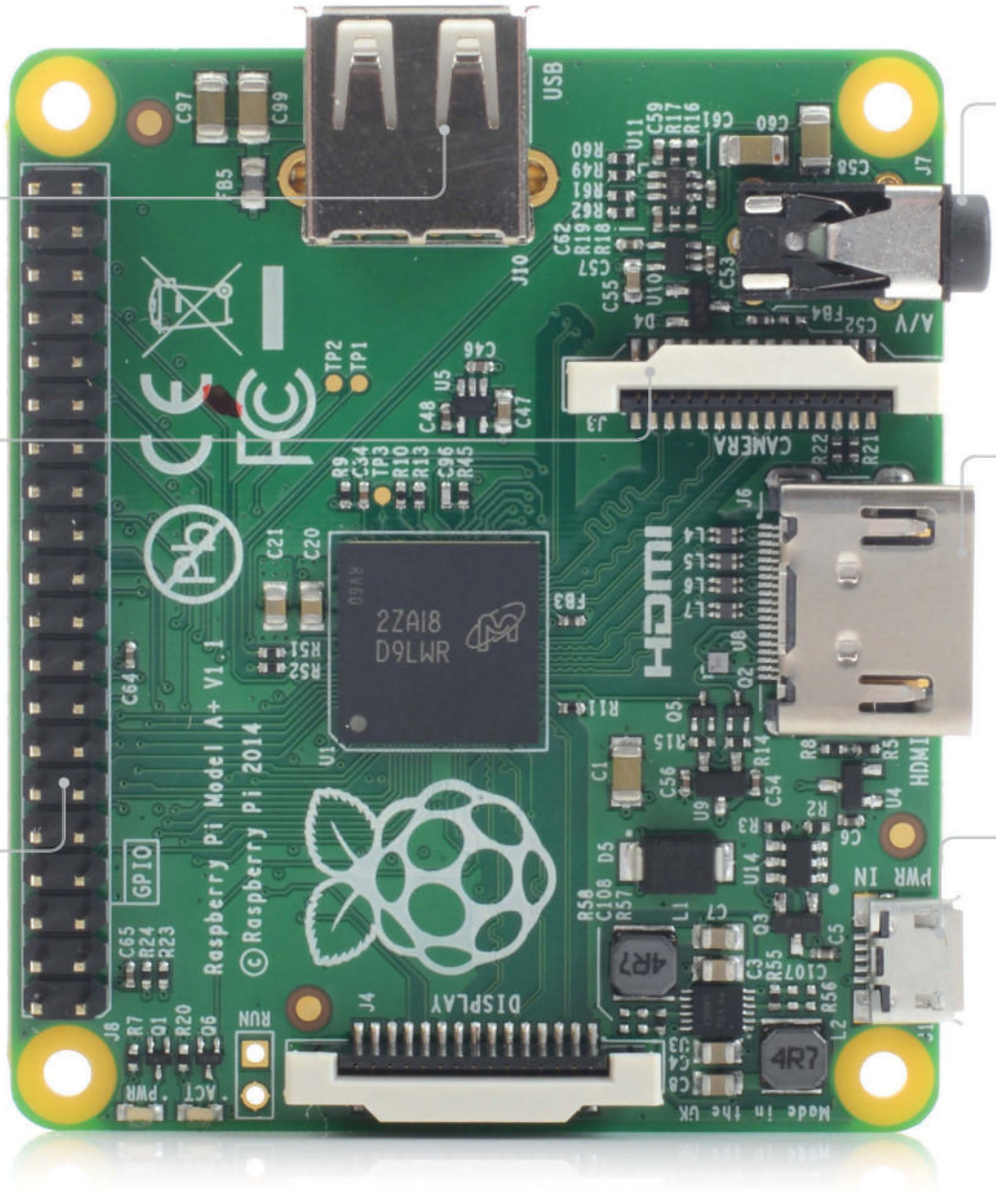
This is the Raspberry Pi everyone has always wanted. It successfully brings some new potential without alienating the user, and we can safely say that it completely delivers.

Technical specs

Processor	Broadcom BCM2836 custom SoC, quad-core ARMv7 800 MHz
Graphics	Broadcom VideoCore IV dual-core GPU included on BCM2836
Memory	1GB SDRAM
Dimensions	86 x 56 x 20mm
Weight	45g
I/O	4x USB 2.0, Ethernet 10/100 MB, HDMI, 3.5mm audio jack, camera module port, display module port, 40-pin GPIO, microSD card slot
Power	Recommended 2A microUSB charger
Price	£23 ($35)

Optimising the BCM2836 processor

According to Eben, there's plenty of unexplored potential in terms of the BCM2836's quad-core parallel processing. Currently, even single-threaded applications can post around 30-35 per cent utilisation of the four cores, as the application sets off things to render and the X server then renders them. Promising work is being to parallelise the rendering operations across the four cores, which could mean reaching 50-60 per cent utilisation when scrolling through and rendering web pages in the very near future. Further optimisation over the coming months and years will reap even greater rewards.



Raspberry Pi (Model A+)

The next version of the Raspberry Pi Model A is noticeably smaller than the original – what's different, what's new and what's it all about anyway?

The Raspberry Pi Model A is the budget version of the Raspberry Pi – the fabled $25 board that was part of the original promise. This version, the Model A+, is the successor to the original Model A and also slightly cheaper at $20 in the US and about £13 in the UK.

The Model A has fewer ports to connect devices to. Just a quick look and comparison to the Model B+ will show that it has fewer USB ports and no Ethernet port. It's also now much smaller than the B, giving you a benefit over the B aside from price.

While the Model A+ can do everything the Raspberry Pi can, albeit with the help of a few USB hubs, it's aimed a little more at the 'makers'. These are people who like to hack together devices and machines but need a small and low-power computer to get everything to work. With some creative use of the GPIO ports and the lone USB port, you can get full access to the CPU in the A+, which is the same one as in the B+.

What does the Model A+ offer, then? Well as previously mentioned, it has a single USB 2.0 port which can be expanded upon with a powered USB hub; the Pi doesn't actually give the port enough power to support the hub on its own. There's a HDMI port that allows for digital audio and video, a 3.5mm headphone jack that is higher quality than the previous board, a CSI camera port for the Raspberry Pi camera, a DSI port for displays and a 40-pin GPIO port. The system on a chip is a Broadcom BCM2835 with an ARM v6 700 Mhz processor, 256 MB of RAM and a VideoCoreIV graphics processor.

The 40-pin GPIO contains the same exact pin layout as the B+, with a mixture of power lines, grounds, communication pins, clock pins and others that can also be used for standard measurement across the pins. This makes it pretty much perfect for projects needing a small computer.

Raspberry Pi Zero

The latest version of the Raspberry Pi creates the opportunity for more of us to own a computer and start coding

Since its release in 2013 the Raspberry Pi has become the UK's bestselling computer. It was originally created with the intention of promoting the teaching of computer science, facilitated through a device that was affordable, accessible and fun to use. It was incredibly successful and adopted by many businesses, educational institutions, hackers and tinkerers alike.

Raspberry Pi like to push the boundaries and at the back end of November 2015 they launched their newest model, the Pi Zero. This ground breaking model is a stripped down version smaller in size than the A+ model. Yet it is three times faster than the original 2013 model. You can still run the same Pi images and software such as Sonic Pi, Minecraft, Open Office. At the heart is the on-board ARM Single-core CPU providing up to 1Ghz of processing and 512MB RAM. The board components have been scaled down, making use of a Mini-HDMI, USB On-The-Go ports and Micro USB (The change in USB and HDMI port size means that you have to purchase convertors for a few pounds) These modifications facilitate the reduction of the physical size of the Pi, to and impressive 6.5x3cm. Despite its small size it still boasts the same HAT-compatible 40-pin GPIO header as the previous models.

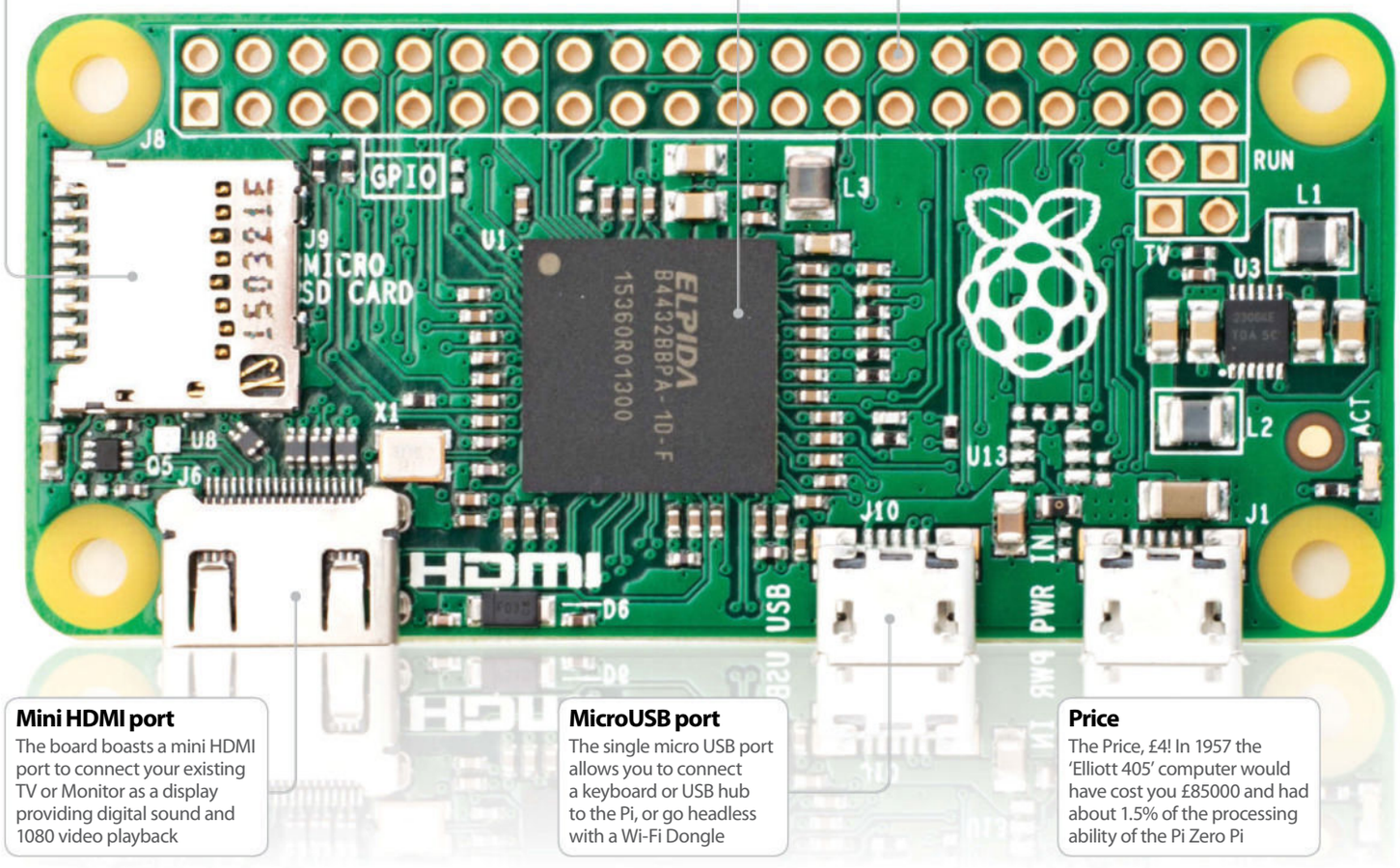
If this wasn't ground breaking enough Raspberry Pi gave away 10'000 Pi Zeros free with their magazine. Within days the Pi Zero sold out across the UK and you will probably still have to join a waiting list if you want one!

In conclusion, the Pi Zero is a computer with a sound specification. It is the first computer ever to be given away free on the front of a magazine. It is also small, frequently compared to the size of a stick of chewing gum, making it suitable for embedding into projects. However, the real outstanding factor is its price. Amazingly, it costs only £4. The Pi Zero is the most affordable computer.

microSD card
The Pi Zero has the same microSD card as the other Raspberry Pi. This means you can use your existing cards and software

Processor and RAM
Single Core Processor running at 700MHz (1GHz overclocked) and 512MB RAM, fast enough to run a range of open source software Open Office, Minecraft, Python, Sonic Pi, a web browser, email client

GPIO pins
Despite the Pi Zero's size there is still space for 40 General Purpose Input/Output, GPIO pins. Providing inputs outputs and 5v or 3.3v of power



Mini HDMI port
The board boasts a mini HDMI port to connect your existing TV or Monitor as a display providing digital sound and 1080 video playback

MicroUSB port
The single micro USB port allows you to connect a keyboard or USB hub to the Pi, or go headless with a Wi-Fi Dongle

Price
The Price, £4! In 1957 the 'Elliott 405' computer would have cost you £85000 and had about 1.5% of the processing ability of the Pi Zero Pi

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

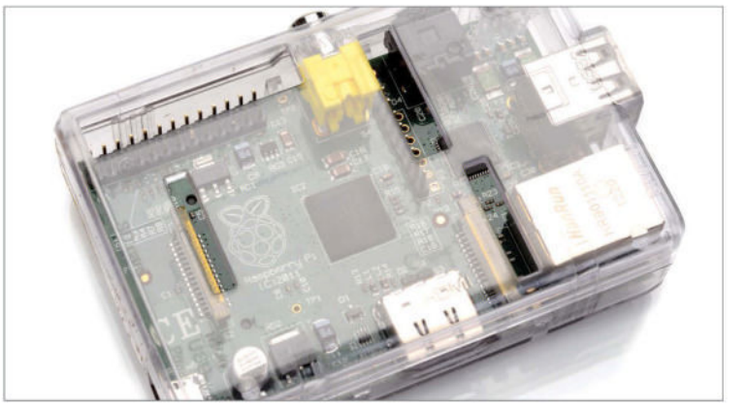
Let's start with the most basic of components, the keyboard and mouse. Generally speaking, virtually any USB keyboard and three-button scroll mouse will work with the Raspberry Pi, and although for some projects you won't even need a keyboard and mouse, it's a good idea to start off with them. For a 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.

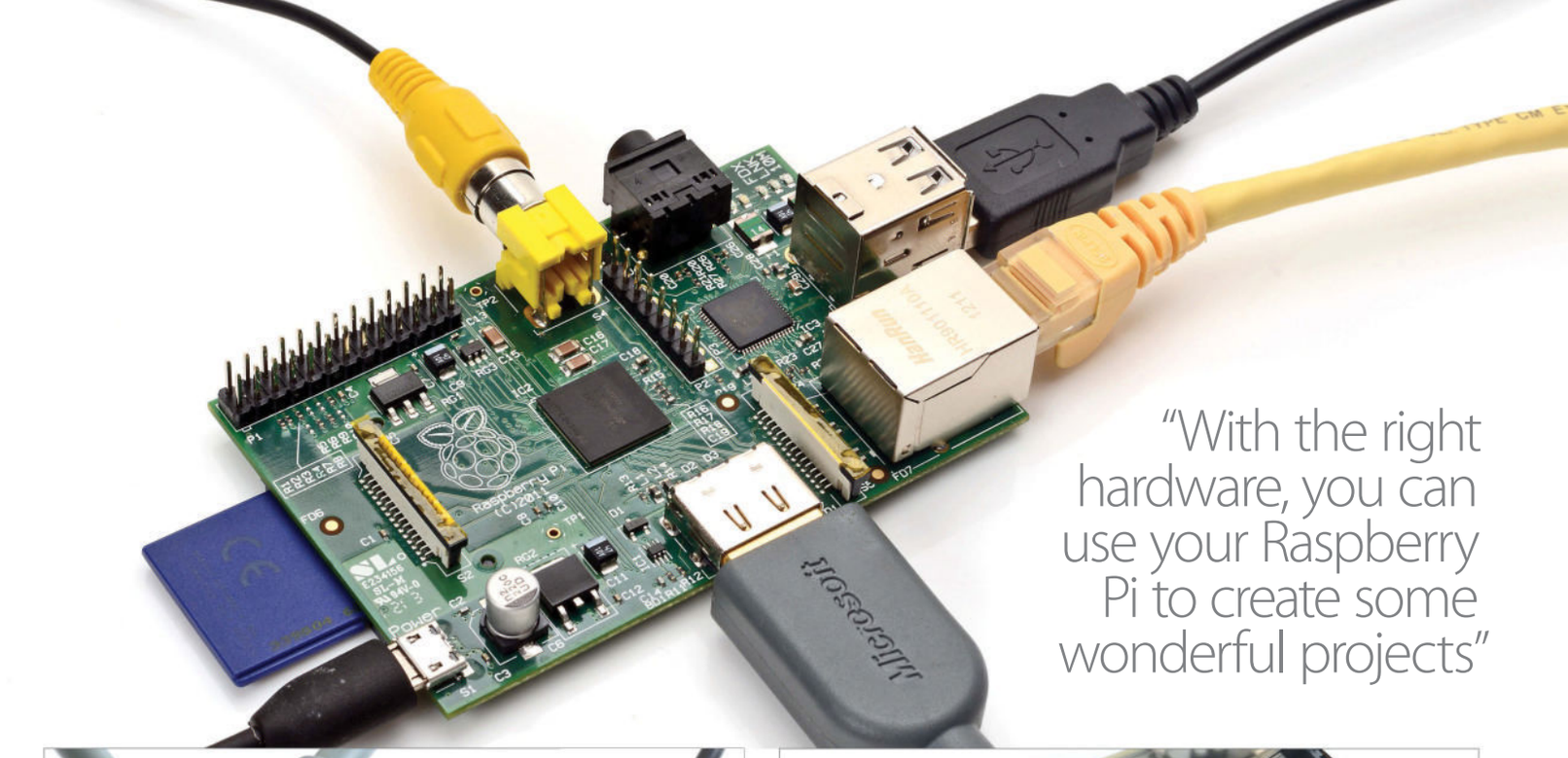
Power cable

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**.

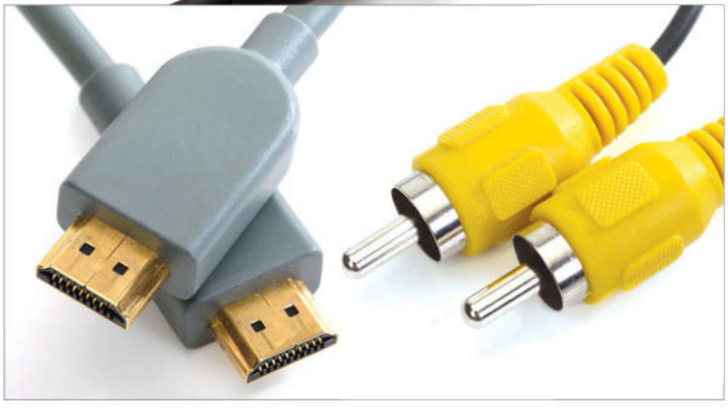


Case

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**.

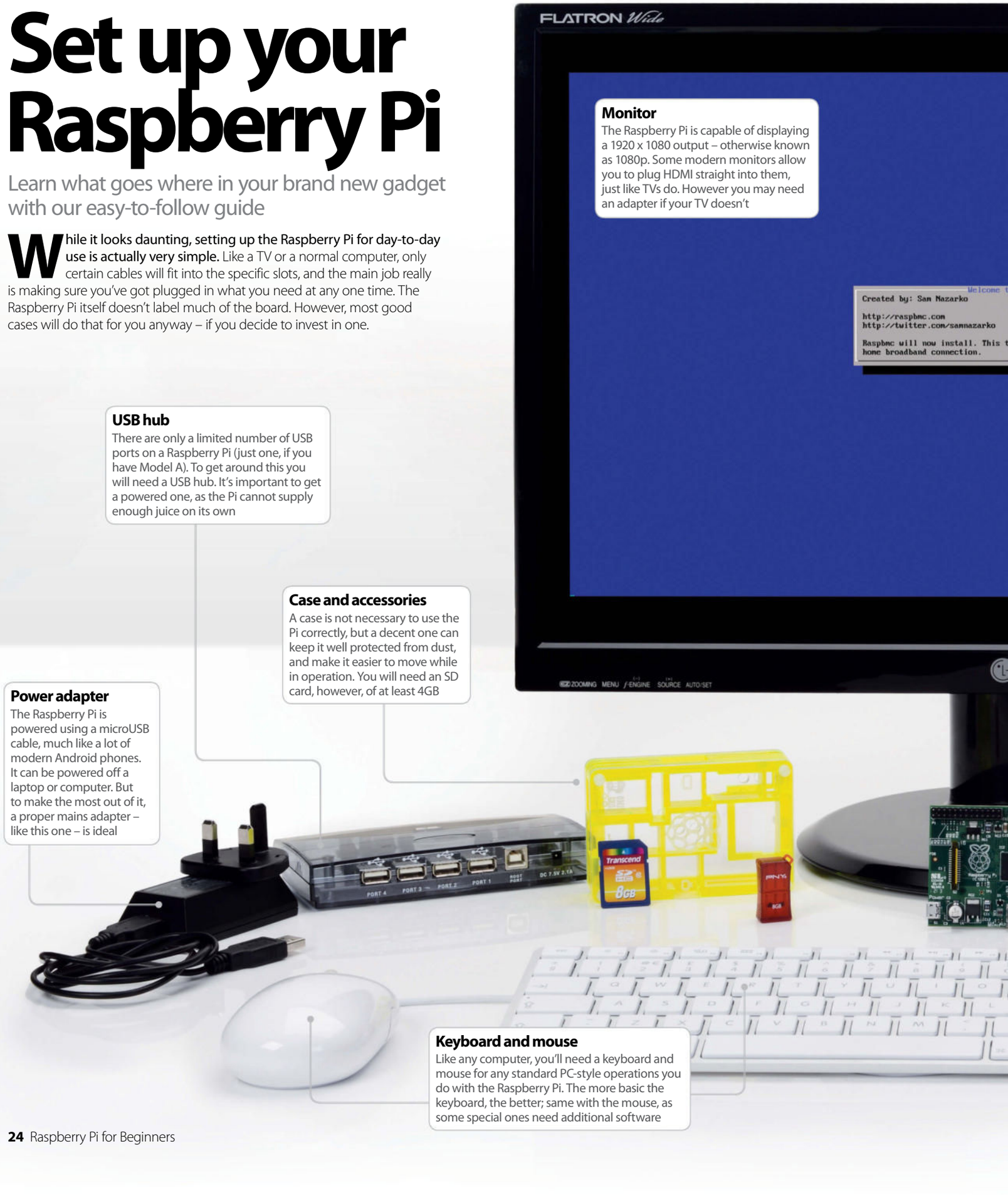
USB Wi-Fi adaptor

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**.

Set up your Raspberry Pi

Learn what goes where in your brand new gadget 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.



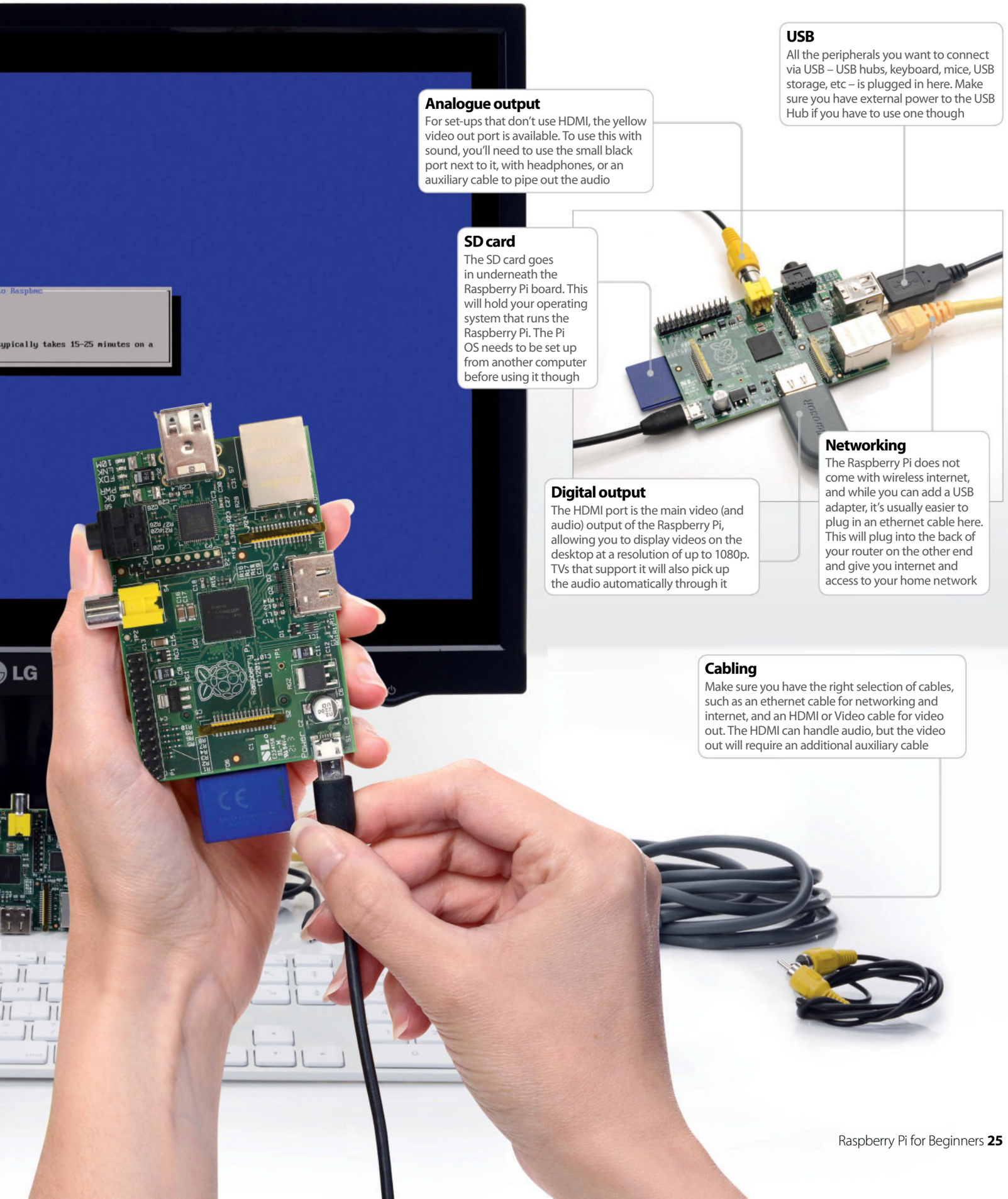
Monitor
The Raspberry Pi is capable of displaying a 1920 x 1080 output – otherwise known as 1080p. Some modern monitors allow you to plug HDMI straight into them, just like TVs do. However you may need an adapter if your TV doesn't

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

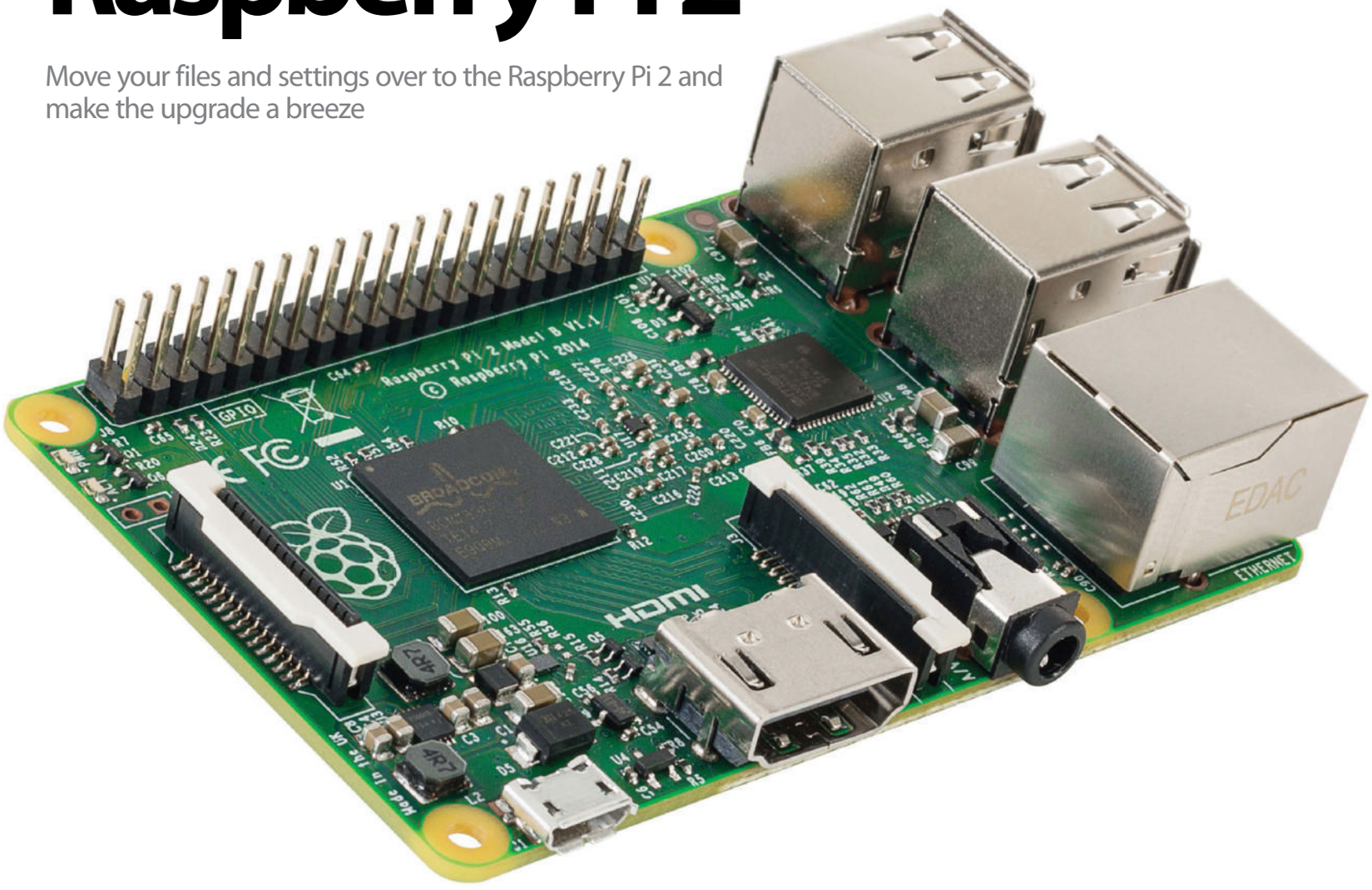
Networking
The Raspberry Pi does not come with wireless internet, and while you can add a USB adapter, it's usually easier to plug in an ethernet cable here. This will plug into the back of your router on the other end and give you internet and access to your home network

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

Migrate to the Raspberry Pi 2

Move your files and settings over to the Raspberry Pi 2 and make the upgrade a breeze



Resources

SD card reader
linux.die.net/man/8/apt

MicroSD to SD card adapter

Raspbian 2 image
raspberrypi.org/downloads

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 full 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 just develops it. They then put it out to the world and make the source code freely available, hence the term 'open source'. Once the program has been tested, it makes its way onto one of the remote servers for that particular Linux distro.

These remote servers, called repositories, or repos, contain all 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 item of software you can imagine. But rather than list them all, it's best to demonstrate how to install these onto your Raspberry Pi and how to use them, or execute them, when they are on there.

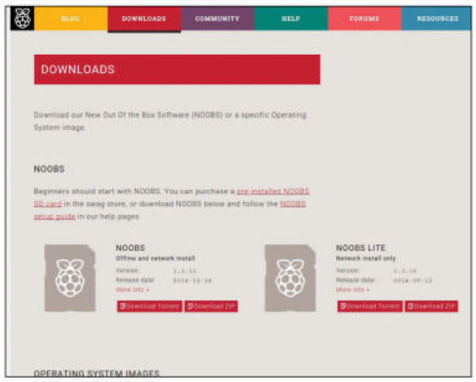
01: Clean up Raspbian

Now is the perfect time to do some probably overdue spring cleaning. Delete any unneeded files and pictures, perform a sudo apt-get dist-upgrade to make sure all the packages are the same as the new Raspbian and uninstall whatever software you don't need.

02: Copy package list

You can make a list of the packages you have installed and then use this list to install the same packages onto your new Pi. Do this by going into the terminal and typing:

```
001 dpkg --get-selections > installed-software.log
```



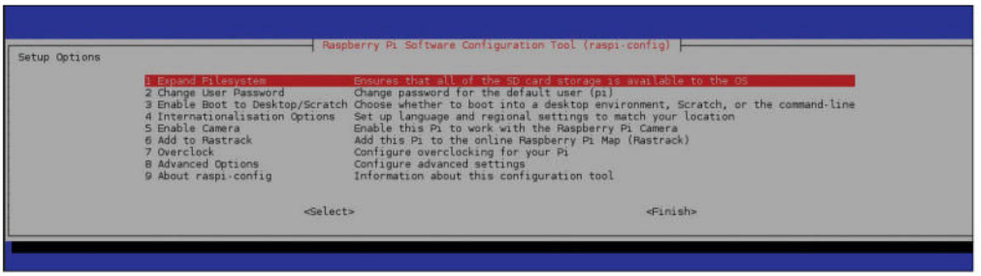
03: Prepare the new Raspbian

On a second, spare microSD card, download the image for the recently updated version of Raspbian from the Raspberry Pi website (**raspberrypi.org/downloads/**) and write it to the SD card using dd:

```
$ dd bs=1M if=raspbian2.img of=/dev/[SD card location]
```

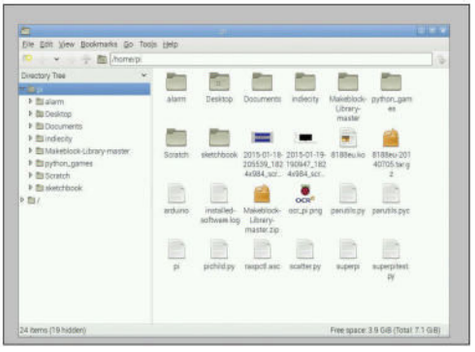
04: Transfer the contents of your home folder

Grab the contents of the home folder from your old Raspbian and either store them on your PC or put them straight onto your new Raspbian SD card if you can have two cards in one PC. Create a backup of this existing card and then re-use it however you want.



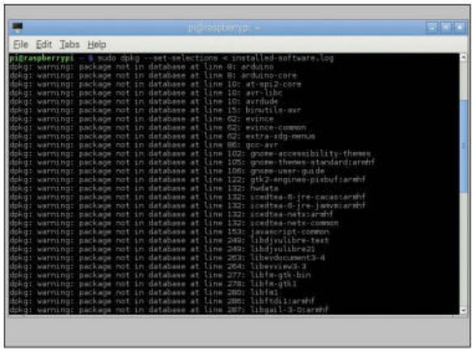
05: Set up the new Raspbian

You'll get to raspi-config when you start up the new Raspbian. Perform the usual updates that you'd want, such as turning the desktop on at boot, enabling the camera and expanding the filesystem. Finish and then reboot the Pi.



06: Check the files

Make sure all the files have properly transferred over, including the package list. If it was just the home folder files, then it should be easy to double-check them. Move anything you've put in there to the folders that they now need to go in.



07: Restore the files

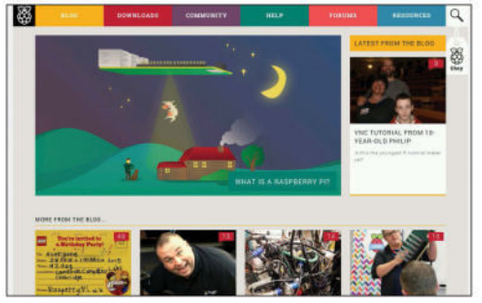
Open the terminal to restore the packages. Some may not work, so you might have to start manually editing the list of everything to install to remove them – most packages will already be installed as well. Restore with:

```
$ sudo dpkg --set-selections < installed-software.  log && sudo apt-get dselect-upgrade
```

"Some files may require an upgrade to a newer version"

08: Upgrade files

Some files may need an upgrade to a new version after you perform the transfer. To do this, do the normal sudo apt-get update followed by a sudo apt-get upgrade to make sure files are up to date.



09: Enjoy your new Raspberry Pi!

Now you should have successfully upgraded and you can get back to doing existing projects or starting new ones. Look out for the next issue of Linux User & Developer when we start doing new project tutorials for the Raspberry Pi 2.

New Raspbian, older Pi

You'll only see one Raspbian image on the Foundation website – there aren't separate versions for both the Raspberry Pi 2 and the older models. Don't worry, though. The way the updated Raspbian works, it now has both ARMv6 and ARMv7 kernels inside it, and it detects which kind of Pi you are using on boot and then it will load the appropriate kernel. It's all automatic, so there's no need to manually switch.

Connect your Pi to a network

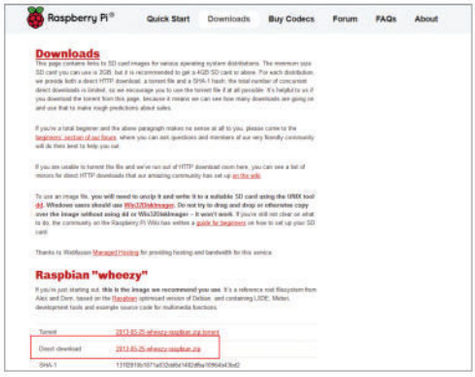
Getting the Raspberry Pi up, running and connected to your network has never been easier

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.

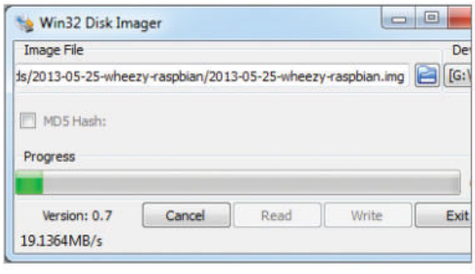
On completing these steps you'll have a fully functional Raspberry Pi as a operation desktop machine. Use as you would any other PC for day-to-day internet, email and word processing or break it out and let it live up to its full potential with some of the great projects in these pages.

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



01: Download the software

Head to **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 **bit.ly/VOUamj** and download Win32 Disk Imager. Unzip and run Win32DiskImager.exe.

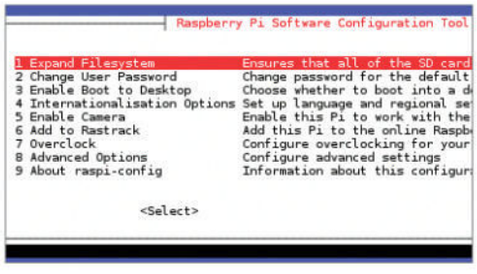


02: Creating your image

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

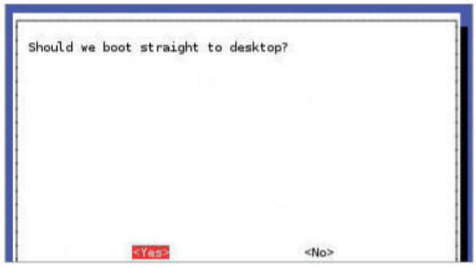
On boot up, after the initial scrolling text you'll see the Config tool. Here you can configure many of the Raspberry Pi features quickly and easily. You can also run this tool at a later date with the command sudo raspi-config. Esc, Tab and Space can be used to navigate the tool.

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.

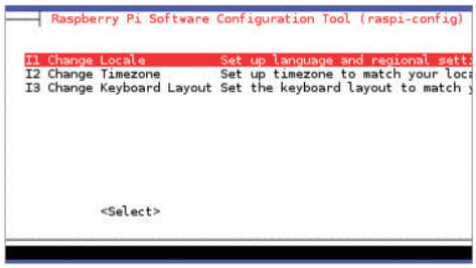
06: Changing the password

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.



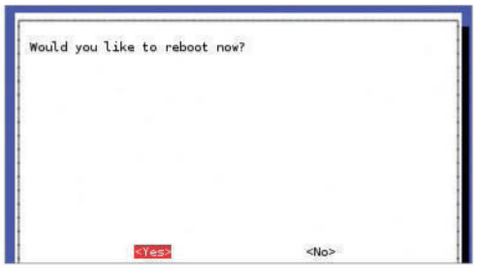
07: Boot to desktop

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.



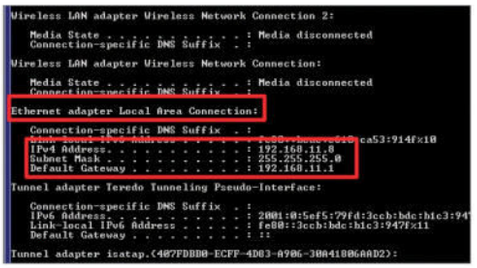
08: Localisation options

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.



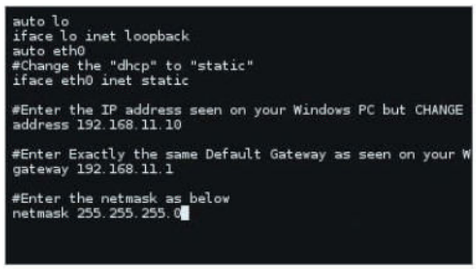
09: Finalising the changes

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.



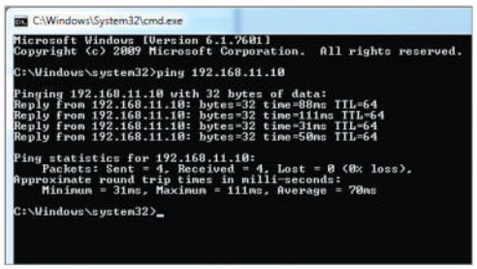
10: Finding network details

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.



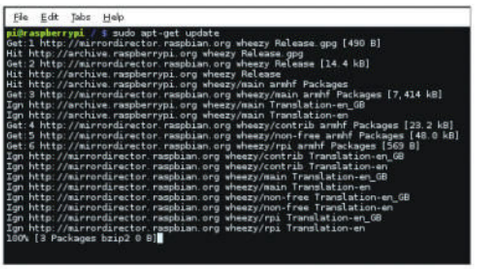
11: Network settings

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.



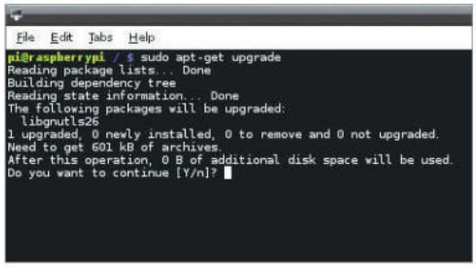
12: Test your connection

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.

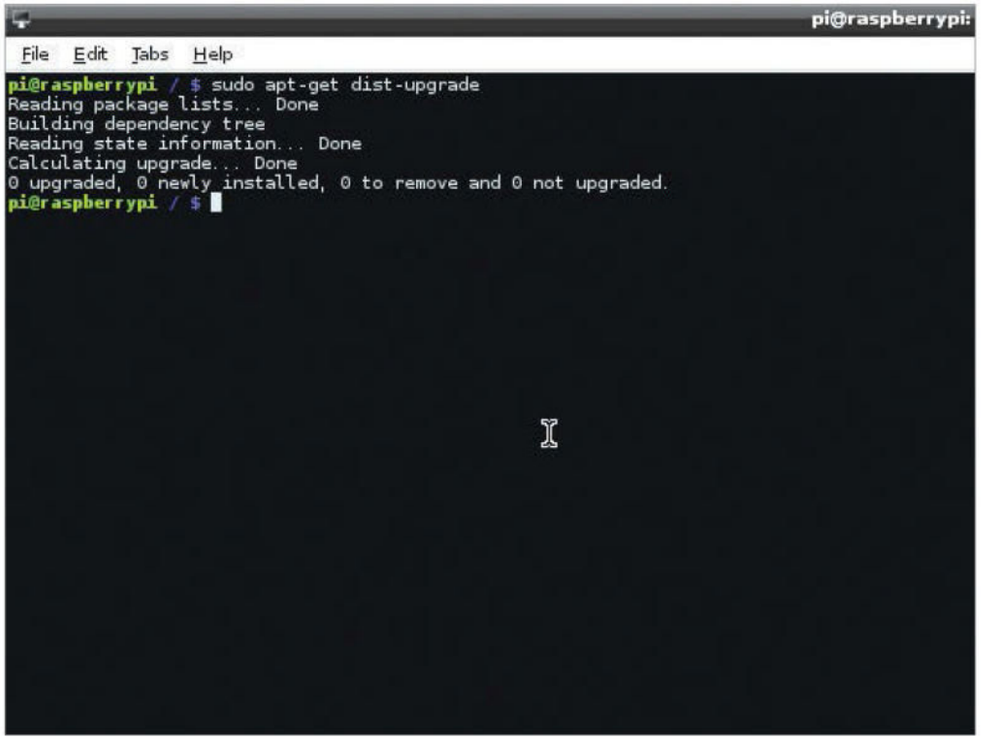


14: Update your software

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.

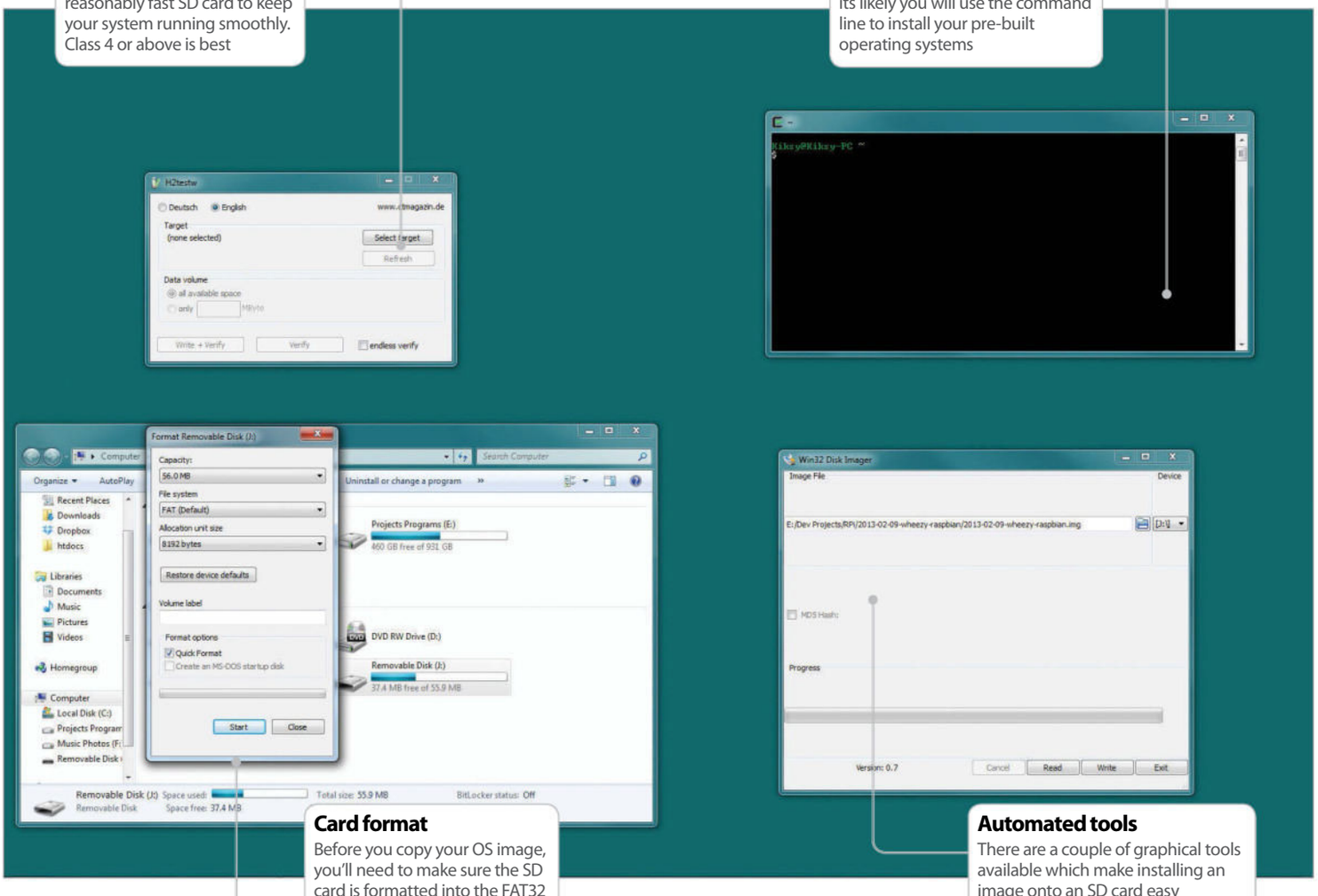
15: Updating the distribution

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.



Card speed
It's a good idea to get a reasonably fast SD card to keep your system running smoothly. Class 4 or above is best

Command line
If you are using OSX or Linux, then its likely you will use the command line to install your pre-built operating systems



Card format
Before you copy your OS image, you'll need to make sure the SD card is formatted into the FAT32 file system

Automated tools
There are a couple of graphical tools available which make installing an image onto an SD card easy

Install an operating system

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

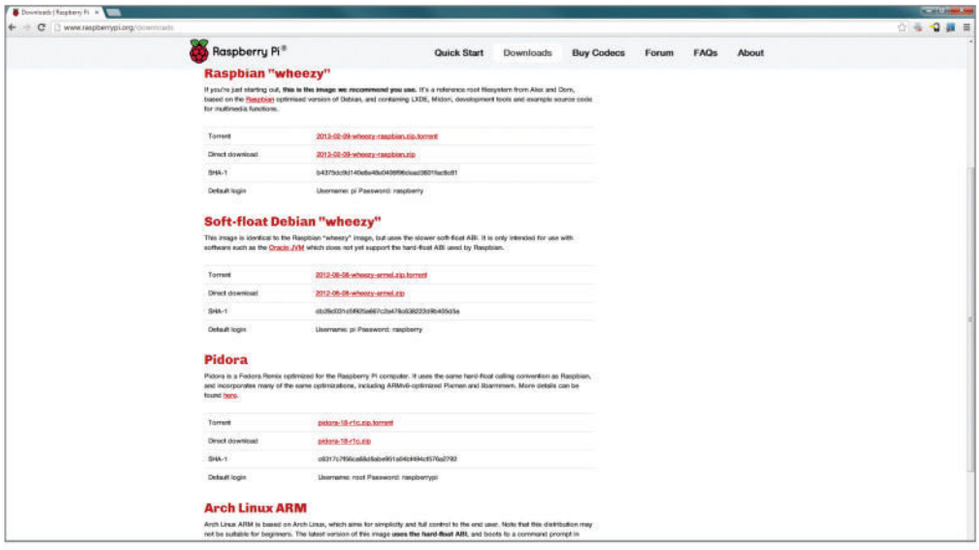
With its small size and cheap price, many people might be fooled into thinking that the Raspberry Pi is only usable for basic tasks, and learning to program on. While one of the primary goals of the Pi was to increase computer literacy at a lower level rather than just learning how to create Excel spreadsheets,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

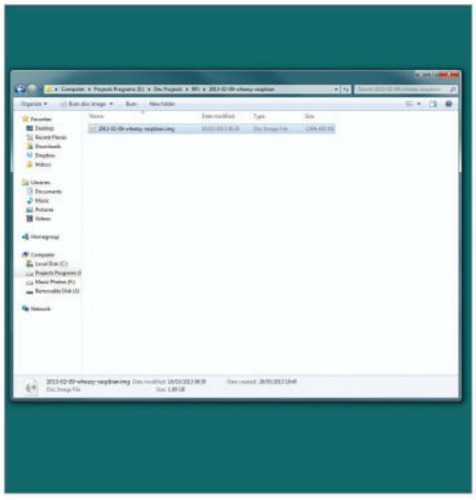


01: Obtaining OSs

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.

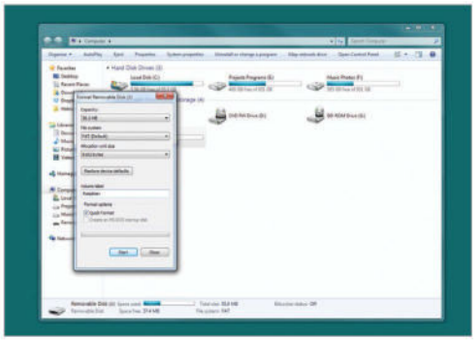


03: OS Format

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.

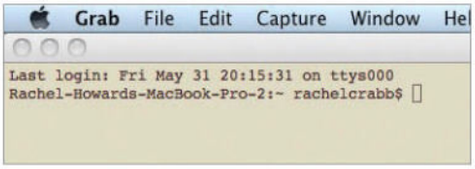
04: SD card format

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'.

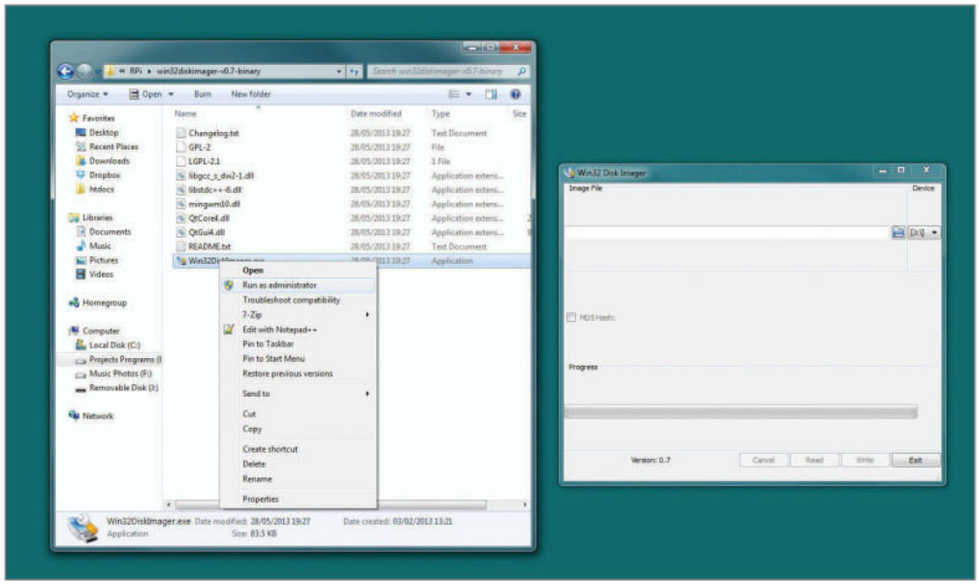
07: DD command

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

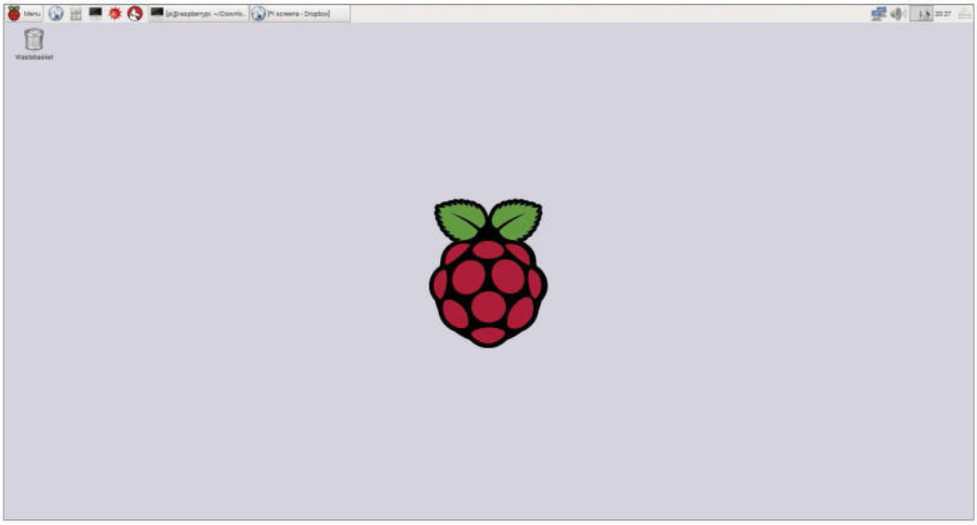
If you are using Windows, there is a handy tool called Win32 Disk Imager, which means you don't have to worry about entering any terminal commands. Once you've downloaded the tool, simply right click on the .exe, and choose 'run as administrator'.



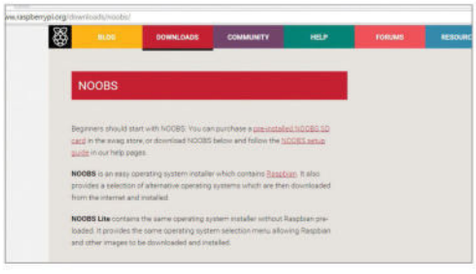
Install Raspbian from Windows 10

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. The most popular and well supported is Raspbian – it has long been the official Raspberry Pi distro. 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 by using a Windows 10 machine to prepare the SD card, but it's possible on any operating system. On OS X, for example, you would simply use the Disk Utility application to write Raspbian to your SD card, and in Linux you can use GNOME Disks.



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

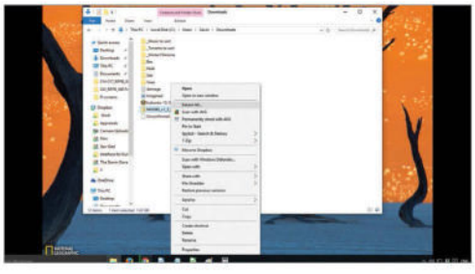


01: Download NOOBS

On your desktop or laptop, go to **www.raspberrypi.org/downloads**. Click the NOOBS thumbnail and then, on the next page, look for the red buttons beneath NOOBS on the left (not NOOBS LITE on the right). Click 'Download ZIP', or 'Download Torrent' if you use BitTorrent or similar.

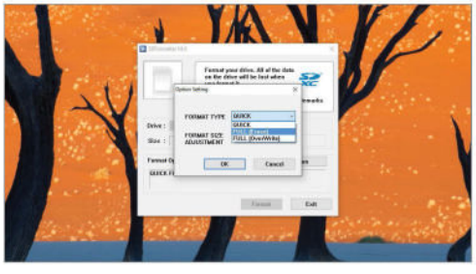
02: Download SD Formatter

Now head to **www.sdcard.org**, click the Downloads link at the top of the page and then choose 'SD Formatter for Windows Download'. Accept the terms and conditions to download the program, which we'll use to copy our Raspbian image onto the SD card.



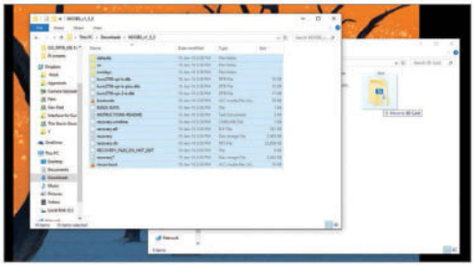
03: Prepare the files

Once NOOBS and SD Formatter have finished downloading, find them inside your Downloads folder. First, unzip the NOOBS file. Next, unzip SD Formatter and run the setup file within. Follow the simple instructions to install the software to your computer.



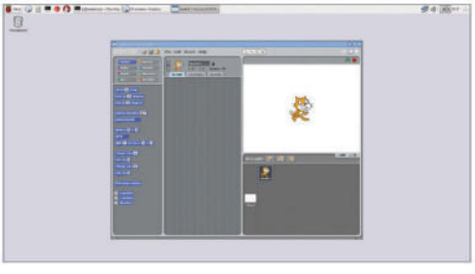
04: Format your SD card

Run the SD Formatter program once it has finished installing. Select the SD card that you've plugged into your SD card reader using the Drive drop-down. In the Format Options, select FULL (Erase). Click Format when you're ready to wipe your SD card clean.



05: Copy NOOBS across

Now, navigate to your SD card using your File Explorer and open it up – it should now be blank, ready for some fresh files to be written. Select all of the files inside the unzipped NOOBS folder, then drag them across to the SD card to begin copying them.



06: Install Raspbian

Now your have a fresh NOOBS SD card, simply eject it from your Windows 10 machine and then insert it into your Raspberry Pi's SD card slot. Power up your Pi and you'll be greeted with an easy-to-follow installer – select Raspbian from the menu to start the installation.

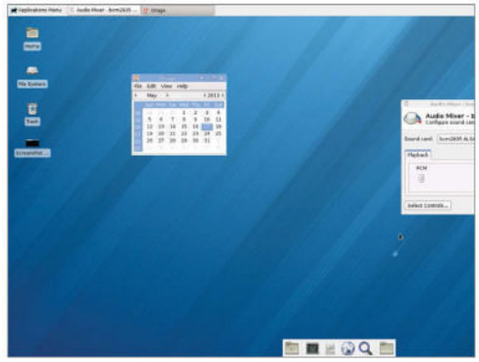
Install Pidora

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.

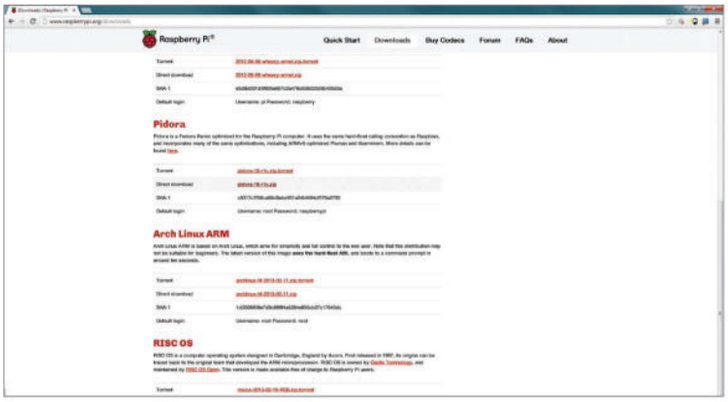
So why would you use Pidora over Raspbian? Firstly, Pidora offers a different desktop environment, the lightweight Xfce, whereas Raspbian uses LXDE. Both have their merits but depending on your requirements, one may suit over the other. The other difference is the packages and package manager used to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



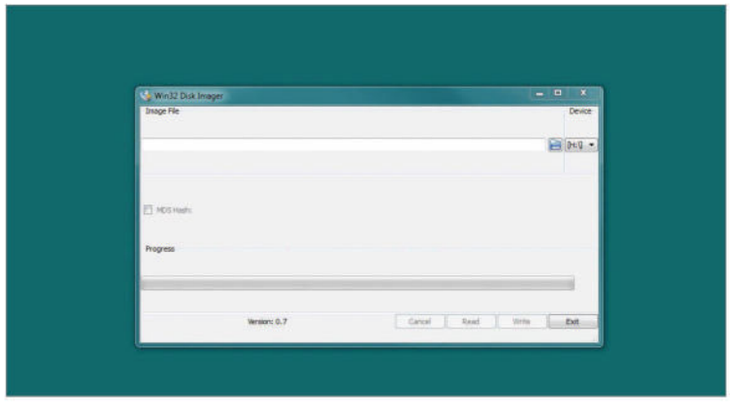
■ Pidora's Xfce desktop environment uses few system resources while being easy to use

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



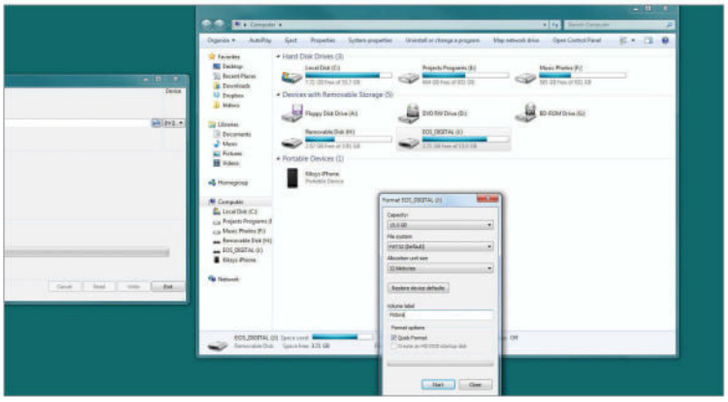
01: Download Pidora

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.



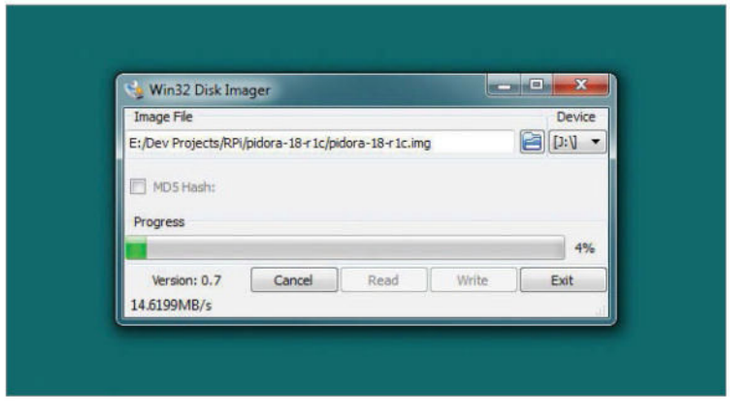
02: Win32 Disk Imager

If you haven't already, download a copy of Win32 Disk Imager from **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.

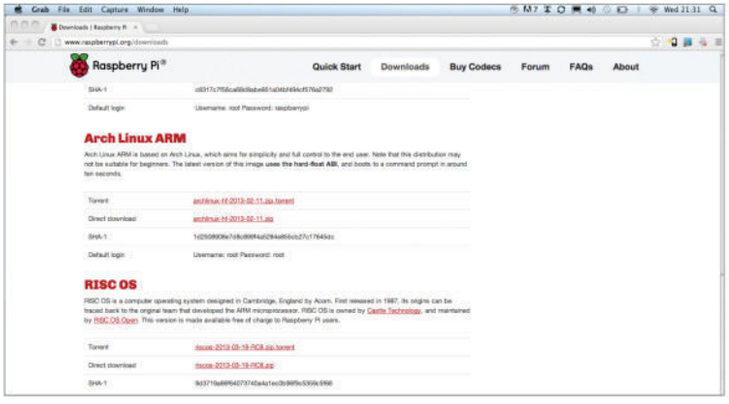
Install ArchLinux

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 your 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.

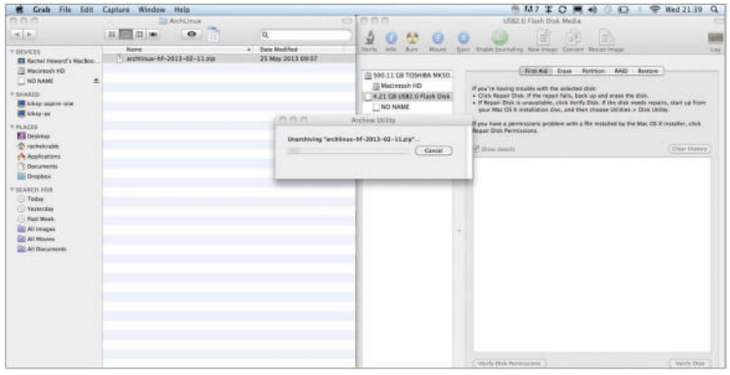
"ArchLinux is complete but lightweight, only including parts you really need"

■ The stripped back gui of ArchLinux allows for improved performance



01: Download the image

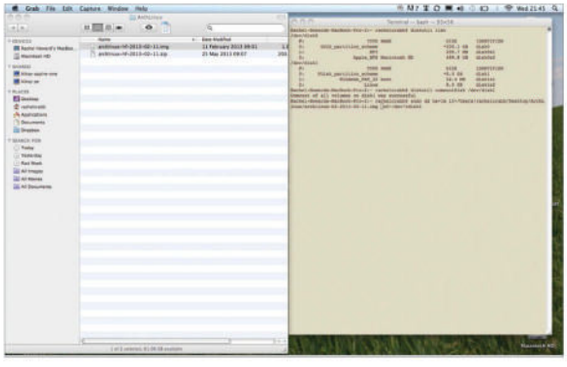
The first step is to download the latest ArchLinux operating system image from **www.raspberrypi.org/downloads**. The file should be around 200MB in size, so its advisable to have at least a 1GB Sdcard to boot from, but bigger is always better.



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:

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

finally:

```
sudo 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:

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

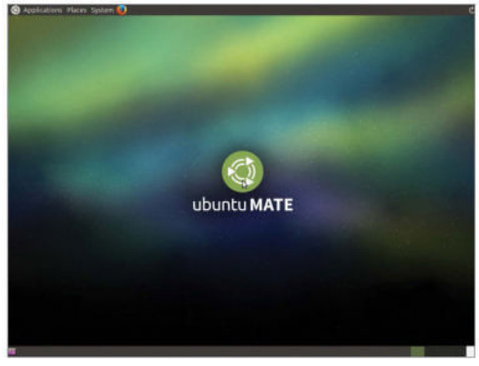
Install Ubuntu MATE

Raspberry Pi 2 owners can run the world's most famous distro

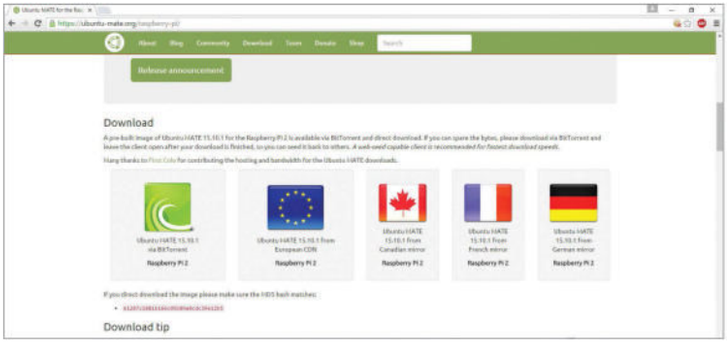
With the Raspberry Pi 2 came the ARMv7 core – replacing the ARMv6 core used in the preceding models – and due to this the most popular Linux distribution of them all finally came within reach of the humble Pi: Ubuntu. One of the more popular spins of Ubuntu is Ubuntu MATE – basically, Ubuntu with the MATE desktop environment installed, rather than the default Unity desktop. MATE is far more familiar to those who have come from Windows and OS X than Unity, and is of course an excellent desktop in its own right.

The Ubuntu MATE developers worked hard to port their Ubuntu spin to the Raspberry Pi 2, and now you can enjoy a desktop-class operating system on your single-board computer. Programs like LibreOffice are blazing fast in this distro, and it is a pleasure to use. If you're looking to explore life outside of Raspbian and learn more about 'mainline' Linux, there is no better place to start than here.

"MATE is far more familiar to those who have come from Windows and OS X"

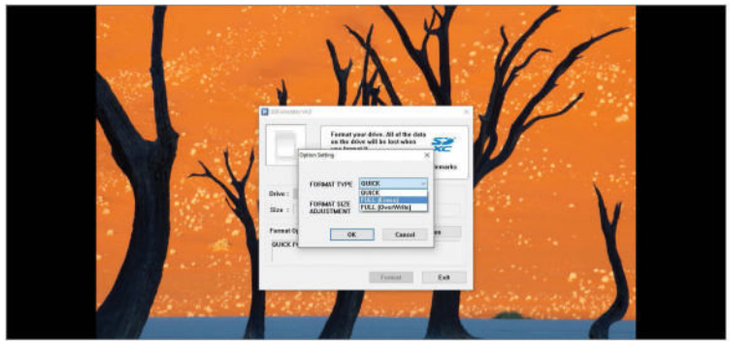


■ GrafX2 is an art package that harks back to an earlier era, combining immediacy with some powerful features



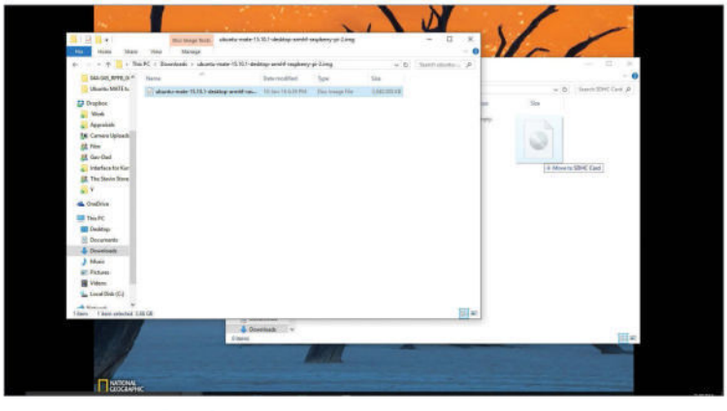
01: Download Ubuntu MATE

First, you need to go the official project website and then download the Ubuntu MATE image for the Raspberry Pi 2 – navigate to **https://ubuntu-mate.org/raspberry-pi**. Scroll down to find the download mirrors; simply choose the one closest to you (for the fastest download speed) and click to start downloading.



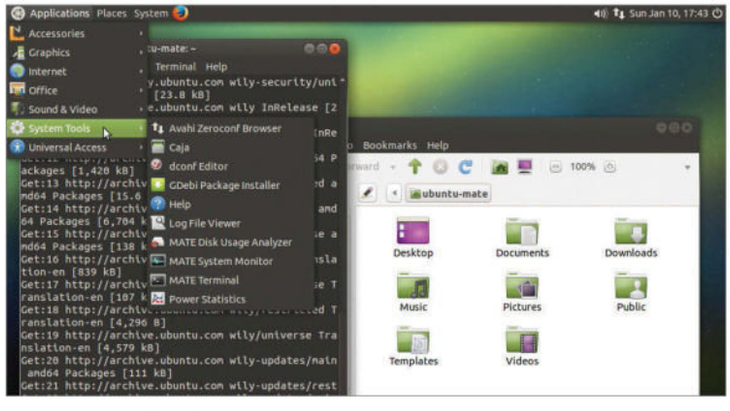
02: Prepare your SD card

You will need to use either a Class 6 or a Class 10 microSDHC card for Ubuntu MATE, with at least 4GB of free space for the OS image. First, format the card with your tool of choice – we recommend using the following: SD Formatter for Windows, Disk Utility for OS X, or GNOME Disks for Linux.



03: Copy the image

Next, you need to unzip the downloaded Ubuntu MATE file. Then, copy its contents across to your SDHC card – in Windows and Mac, you can drag and drop these across to write the files. In Linux, we recommend using GNOME Disks again, or GParted, if you're not familiar with the dd command.



04: Install Ubuntu MATE

With the Ubuntu MATE image copied across to your SDHC card, simply eject it from your main computer, pop it back into the Pi and then power it up. The installation process will automatically begin, and you'll be guided through a graphical interface. Upon completion, you'll be taken to your new desktop.

Navigate the Raspbian GUI

Raspbian's graphical user interface can be a much simpler way of interacting with your Raspberry Pi than the command line interface – here's a quick tour

Icons

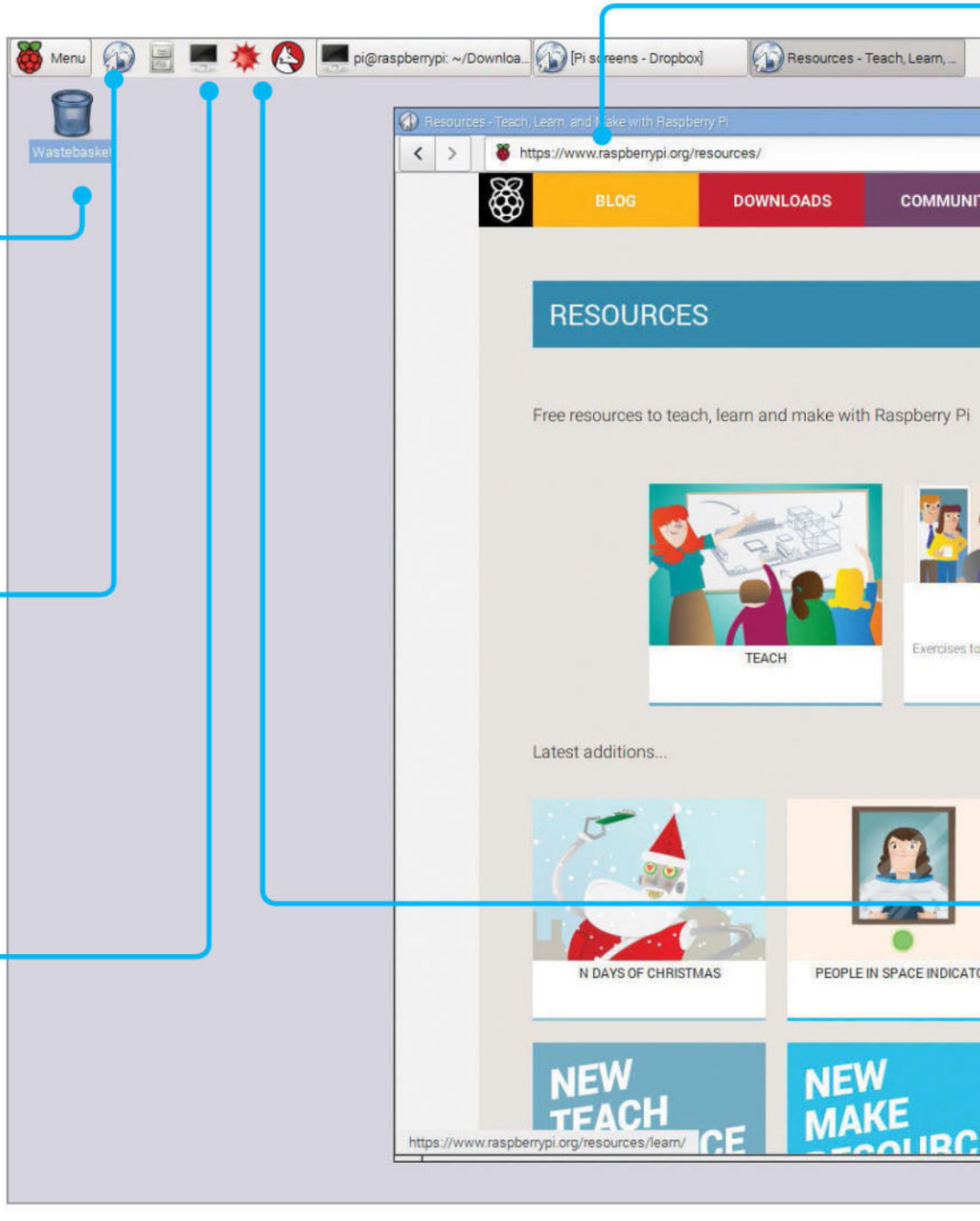
You can add apps to the desktop for easier access. To run the associated app, simply double-click on the icon. The icons can be moved and dropped into other locations. Click, hold and drag to move them. There are icons for a variety of tasks – from apps such as Python and Scratch for programming, to Empathy for web browsing, LXTerminal for issuing commands to the system directly, educational software like Wolfram Alpha and the Mathematica, to reference materials. Icons can be renamed or even deleted if you don't want them on the desktop any more.

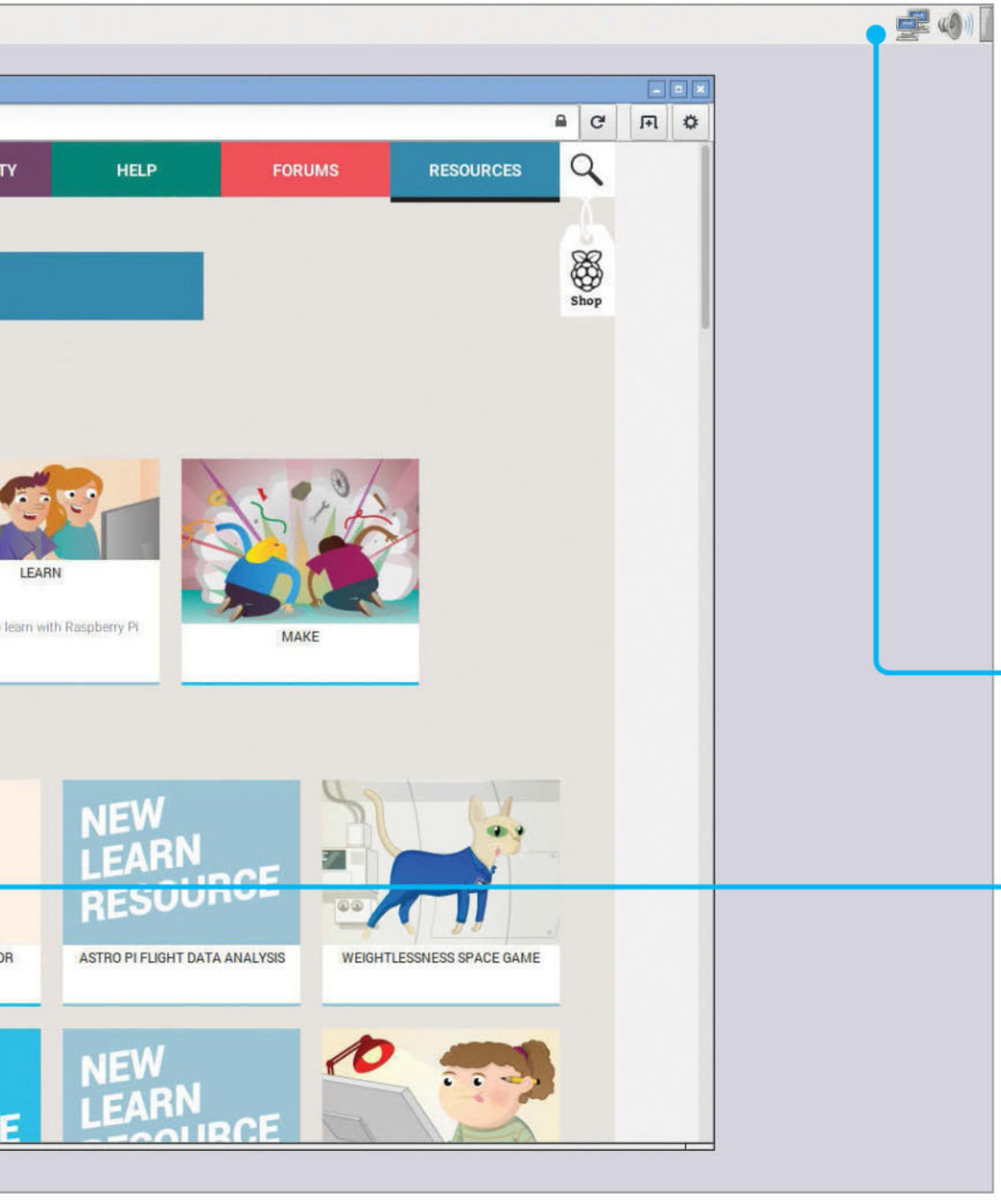
Menu bar

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, 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 VLC for playing videos) 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. All newly installed software will turn up inside your main start menu.

File manager

No computer would be complete without a file manager. Click here to bring it up. Files can be copied, renamed and deleted as in other operating systems. (Deleted files are moved to the Wastebin – you'll need to open this and re-delete them to properly remove them.) 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.





Web browser

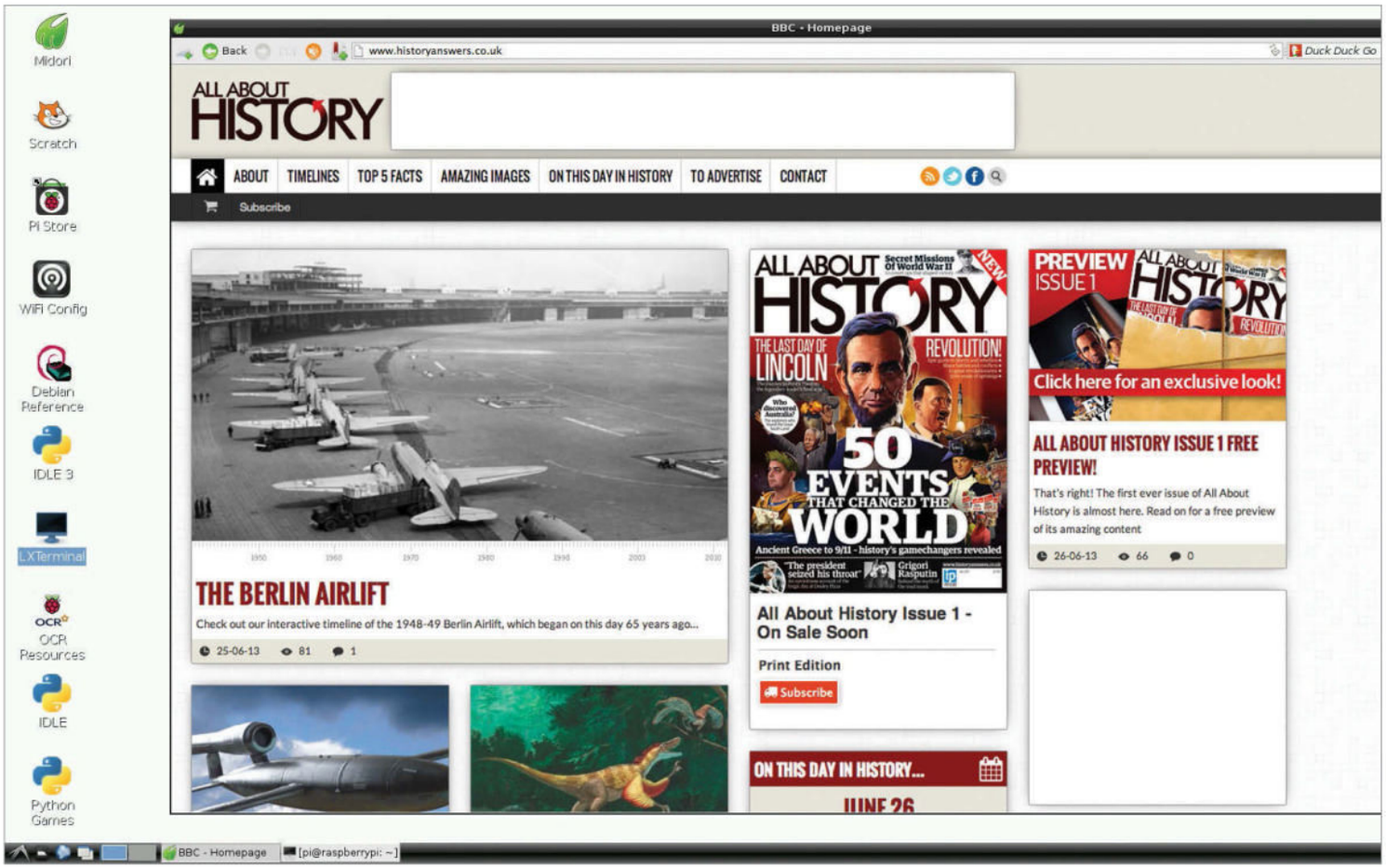
The default web browser on the Raspberry Pi is called Empathy. There's a quick launch icon to open it. The window can be resized by hovering the cursor over the sides or corners. Web addresses are typed into the long address bar whereas search queries are typed into the shorter bar with the words Duck Duck Go. There are standard navigation buttons for going back and forth through webpages. Empathy 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.

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.

Terminal

The terminal is used to bring up the Raspberry Pi's command line interface. Before Raspbian was made to boot automatically into the graphical environment that you see here, the Raspberry Pi would by default boot straight into the command line, and to get to the graphical interface you would need to run the startx command. That's not to say that this is now defunct, however – far from it. If you really want to master the Pi, learn to work with the terminal and you'll soon be able to do absolutely anything with just a few keyboard strokes.



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 that comes 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

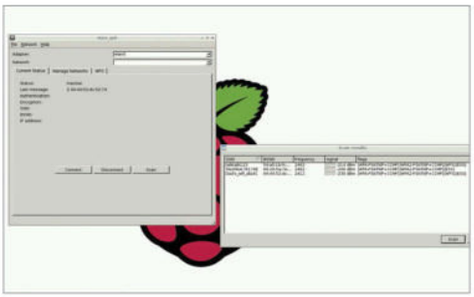
If you aren't close enough to the modem/router to be able to plug in the Ethernet connection, or you simply have a Model A, then a powered USB hub is required. This plugs into a USB port on the Pi. You can then plug a Wi-Fi dongle into this. Boot up the Pi and launch the desktop. Then double-click on 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

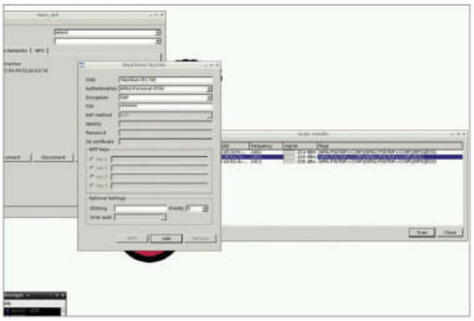
To check that the Pi has a valid internet connection, double-click on LXTerminal. Enter this command:

```
ip addr
```

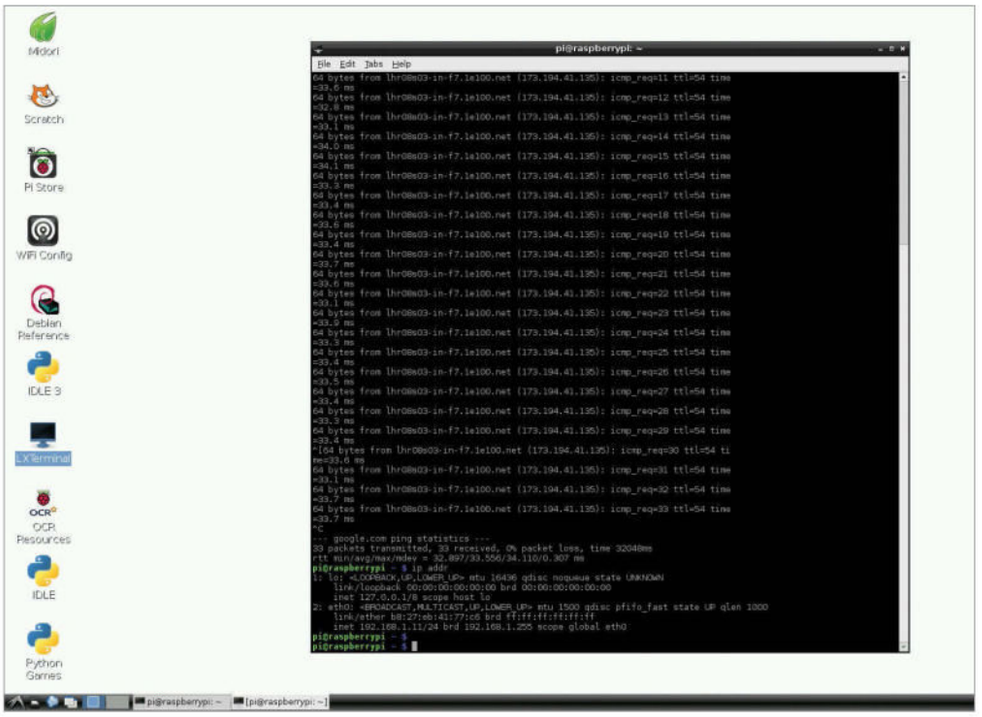
You should see a list of numbers, with the bottom line starting 'inet' and then the IP address of the Pi connection (Fig 3). Typically this is something like 192.168.1.11 and this shows that the connection is working because the Pi has been assigned an IP address based on the one used by your internet 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
```

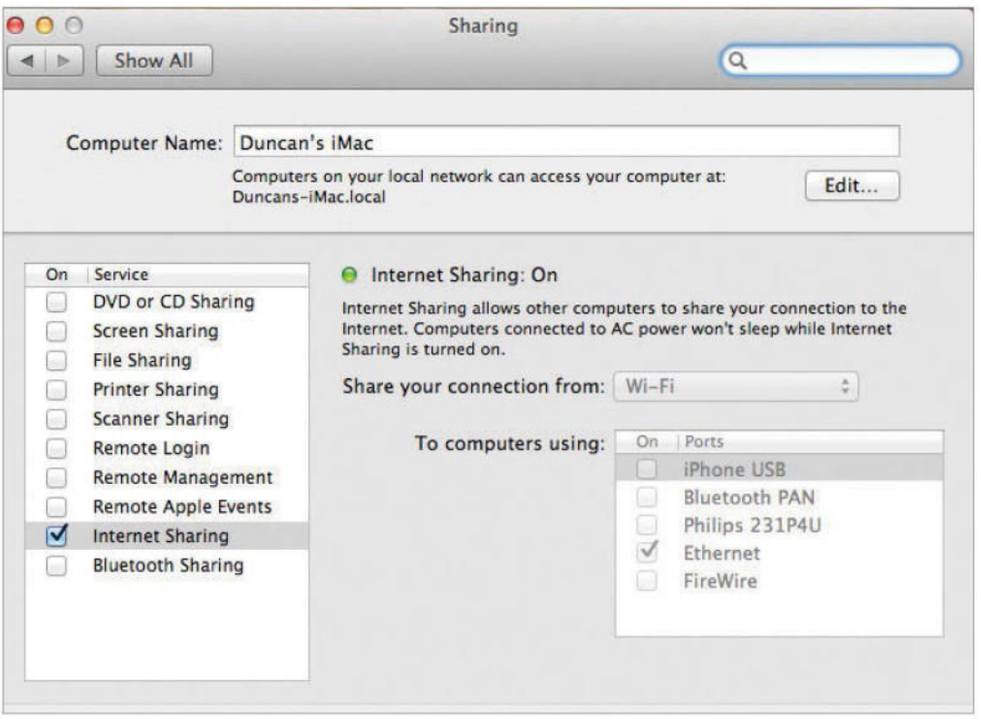
Give it a few seconds and press Ctrl-C to stop. You'll see that it has successfully sent and received a number of packets.

Sharing a connection

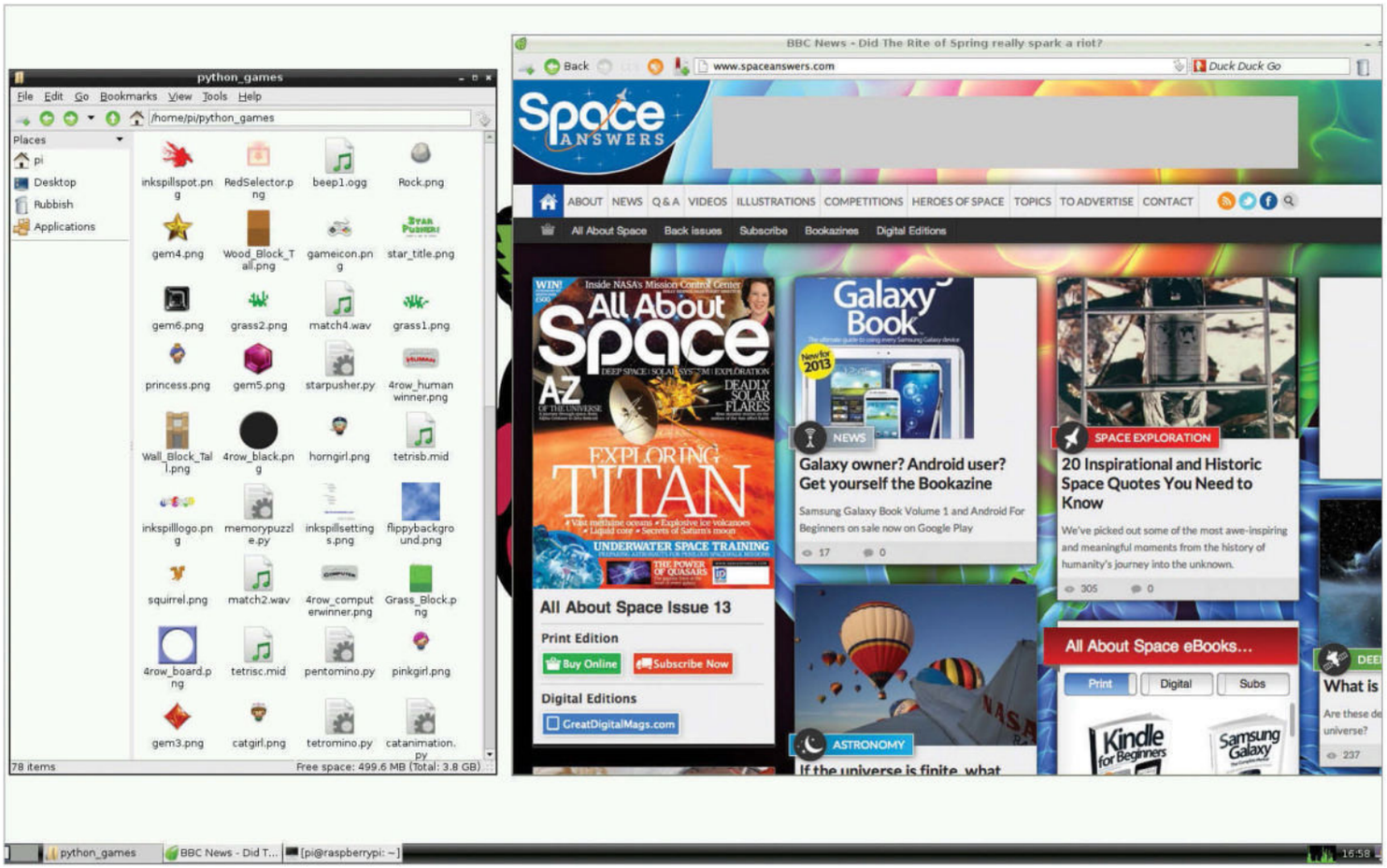
If you don't have a Wi-Fi dongle, a powered USB hub or a long enough Ethernet cable, but do have another computer connected to the internet, there's another way of getting access. On a Mac, connect it to the Pi via a USB or Ethernet cable. Launch System Preferences; under Internet & Wireless, click on 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 as well. The standard browser for this is Epiphany and there's an icon on the quick launch bar to enable you to run it with ease – just double-click on it to get started. You can also run Epiphany by going to the menu under Internet, where there's an option to run Epiphany in Private Browsing mode. On top of this, you also have the NetSurf and Dillo browsers as alternative choices. 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 alternative options for web browsing isn't actually installed, but it is available. Open the LXTerminal, or at the command line before starting the desktop and type:

```
sudo apt-get install midori
```

Midori is the original Raspbian browser, only replaced within the last year by Epiphany. It now works a lot better thanks to the Raspberry Pi 2 hardware though, but it doesn't have Flash by default. You can install it with:

```
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.

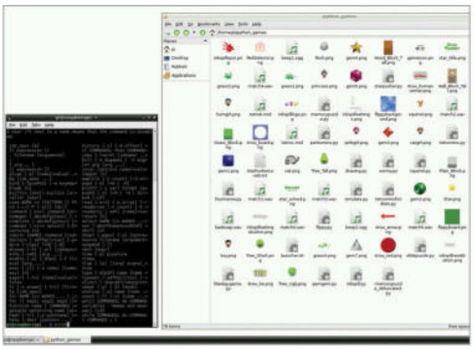
Programming the Pi

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 – still widely used as there are more resources available for it 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 are a selection of Python games installed and the code for these can be loaded up to see exactly how they work, with the scope to amend or improve them yourself.

The accessory programs

There are a number of accessory apps that come pre-installed, including an image viewer, calculator, notepad, archiver, file manager and a couple of



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

versions of the terminal. Look to the main menu under Accessories to see them listed. There's also a reference manual for Debian there.

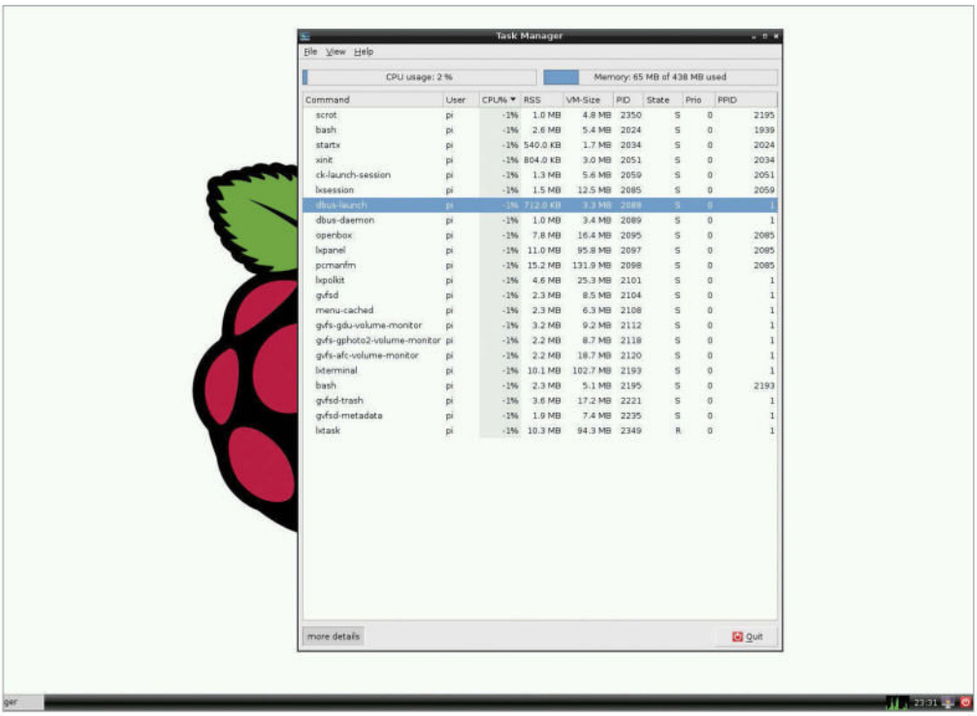
The Image Viewer is a basic viewing program that automatically runs when you double-click on a graphic file. Images can be rotated and saved. The two most useful apps are the file manager (Fig 1) and the LXTerminal. The former gives a very familiar-looking windowed file-access structure with shortcuts 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

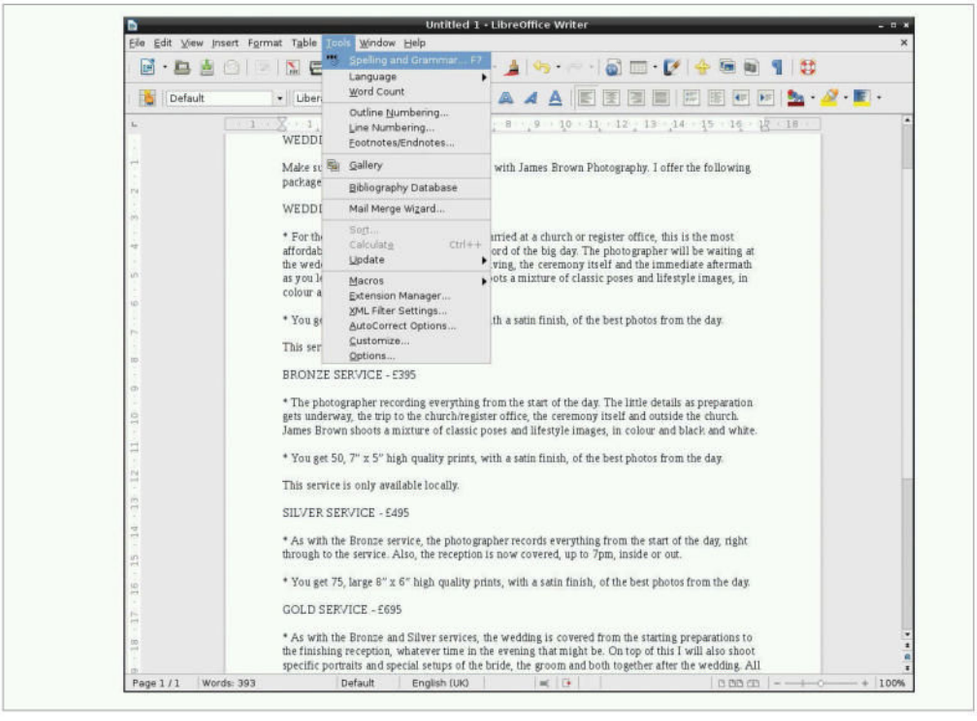
There are a brace of other handy accessories, listed under Other on the menu. They are a PDF viewer and the Task Manager (Fig 2). This displays commands and CPU percentages being used, as well as memory overheads. The view can be trimmed down by showing user, root or other tasks. To stop a task or command 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, a great place to go is the Pi Store. Here you can find averitable cornucopia of 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 take some time to have a look around and see what you can find.

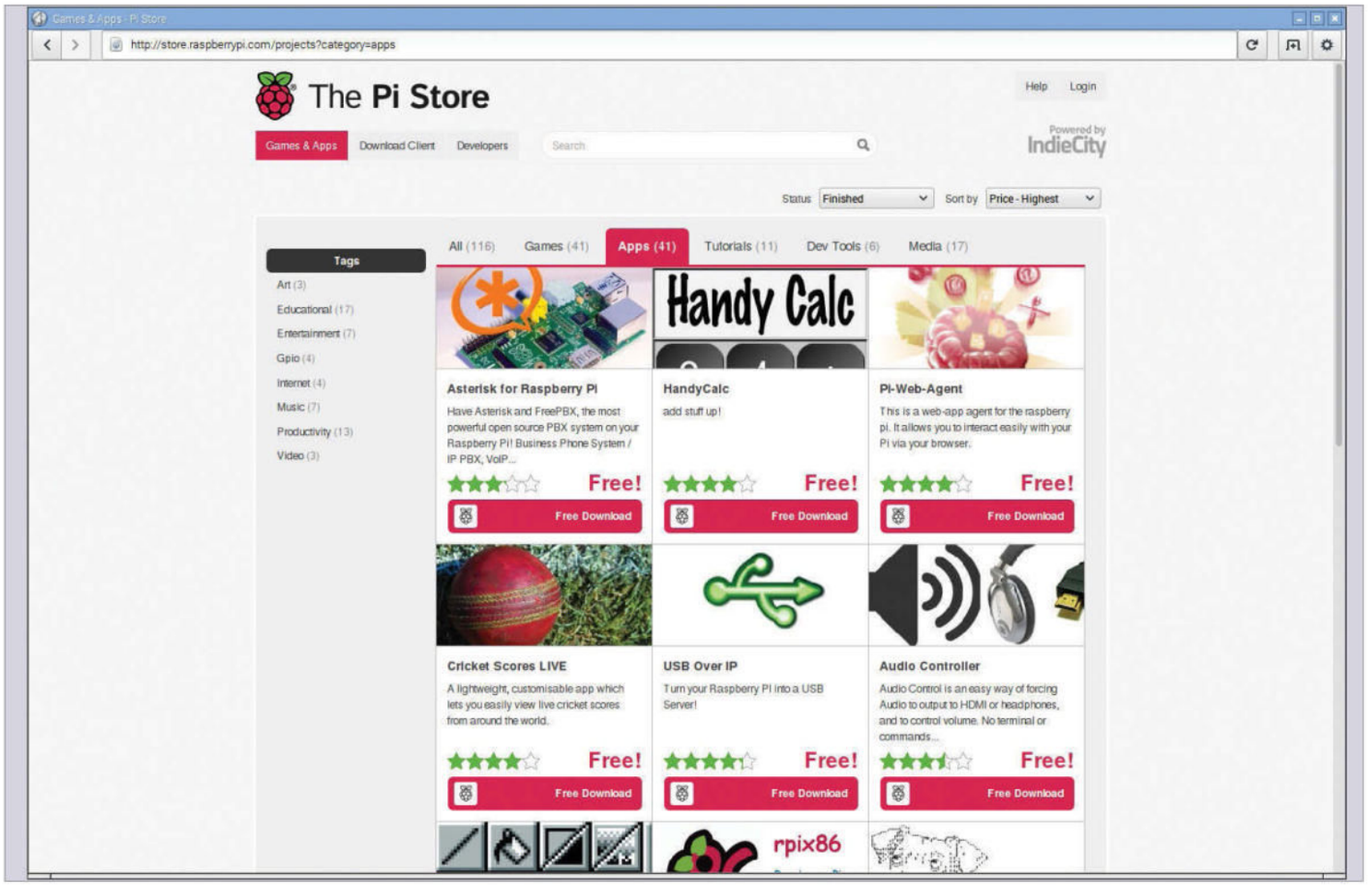


■ Fig 2: The Task Manager is like Windows. It shows you how much CPU load and memory a process is using

"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



Explore the 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 either a customer or a developer using your web browser. While there used to be a client program available to download straight onto your Pi, this has sadly been removed as the popularity of the Pi Store has dwindled. Most of the items on the Pi Store are free or low priced, and it's clear that the developers of the site originally hoped to build a thriving community. In particular, they 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. But despite the project's gradual disappearance into history, there's still a decent amount of utility to be found in the existing content that's been uploaded by users.

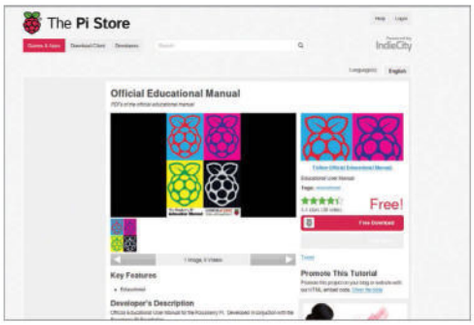
As well as being a platform for browsing, installing and buying applications, the Pi Store has integrated features for developers to create a profile and upload projects. Customers can also create a profile and participate in the social side of things.

Browsing the Store

To get to the store, head over to **store.raspberrypi.com/projects**. 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

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



■ Fig 1: Documents such as this make the Pi Store worth a visit, despite the platform's increasing obscurity

database program using the Python programming language along with SQL, a language for accessing databases. Another provides an reference manual for the Pi that's accessible offline (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).

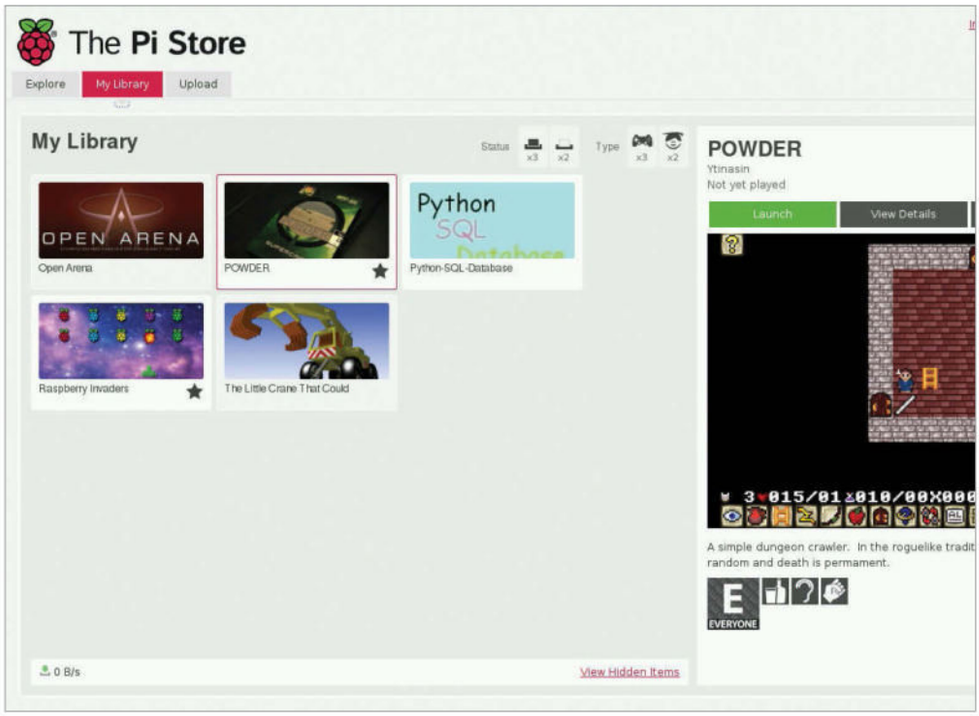
If you click on your username, you can edit your profile (Fig 3). You can change your username from the one that the store assigns you when you first sign up and add a link to your website and a photo. Doing this helps you to establish a presence within the community when you're giving feedback.

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.

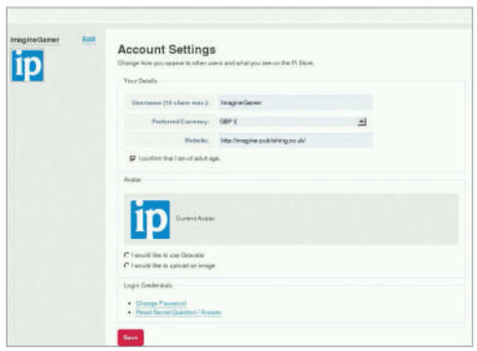
Uploading content

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.

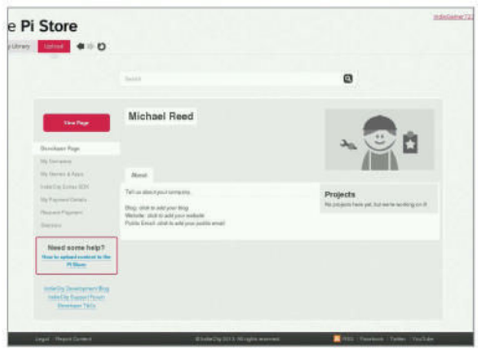
From here, you can also add a product. When you add a product, you can specify that it will be publicly viewable or hidden and that it is finished or a work-in-progress. You can also specify the category of your project (game, application, tutorial, dev tools or media), add screenshots, box art, age 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



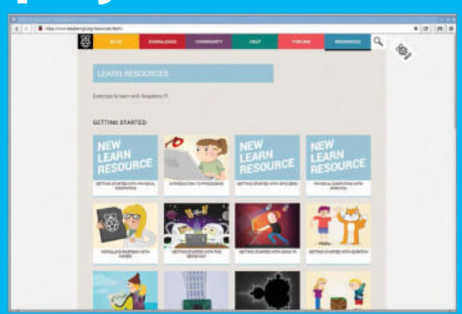
■ Fig 3: Customise your profile with a recognisable username and a picture



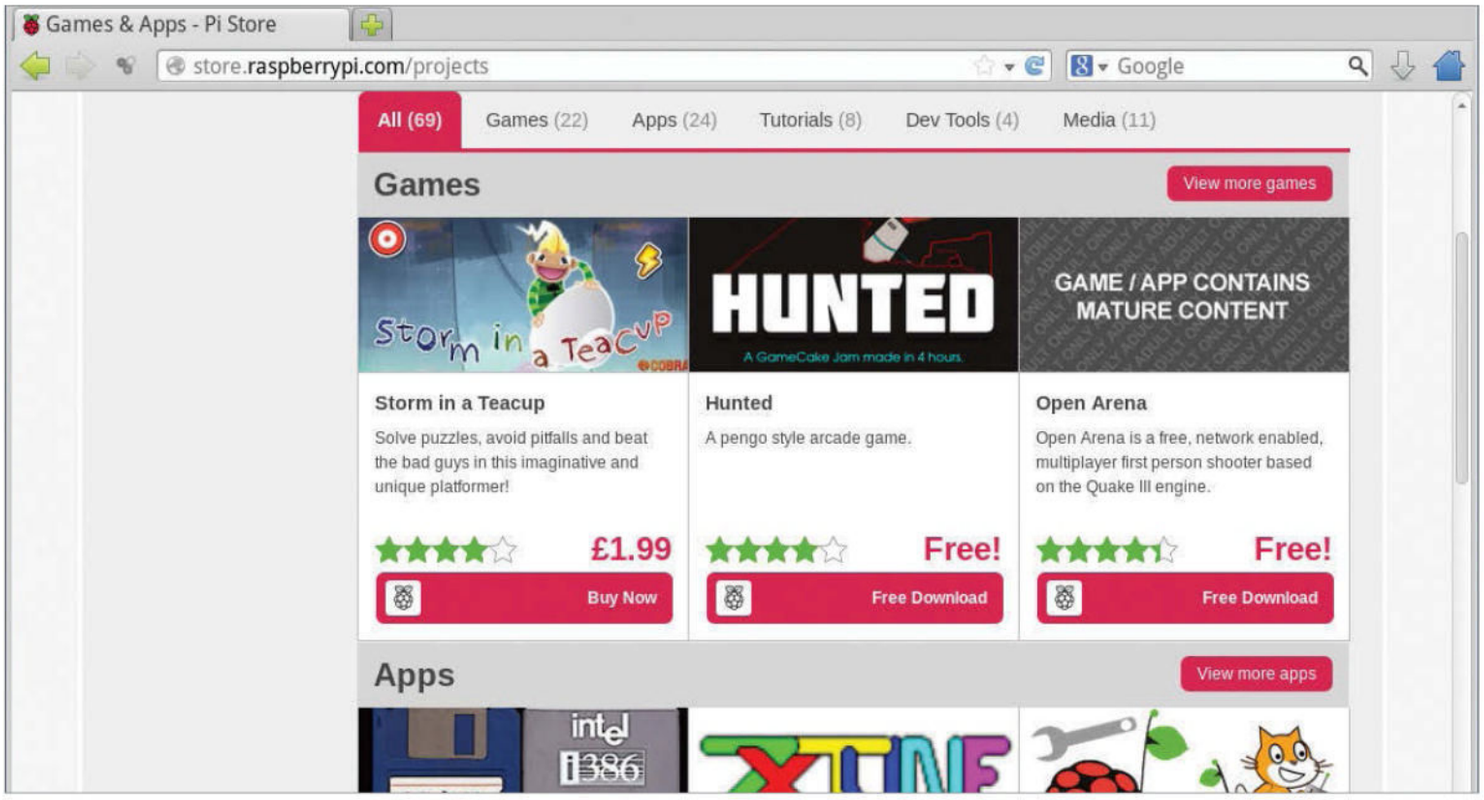
■ Fig 4: Click on the Upload tab to begin editing your developer profile

Where are the new projects?

Go beyond the Pi Store



With the Pi Store essentially being left to languish and few people contributing, where are the new resources being hosted? Easy – the official Raspberry Pi website! The Raspberry Pi Foundation is now actively creating and curating a collection of high-quality resources specifically designed to get you off the ground when you're first setting up, and then to inspire and educate you as you go forward in your journey to master the Pi and the underlying Linux technology. Go to **raspberrypi.org/resources** and then choose Learn or Make to find some excellent resources. IT teachers will want to check out the Teach page to find lesson plans and more.



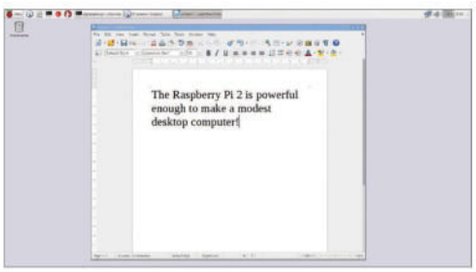
Discover the best apps

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

As we've said a few times, the Raspberry Pi is a Linux-based computer. From a software point of view, this means that you can run Linux programs on your Raspberry Pi – you don't need special Pi-only versions; all you need to do is check the software requirements and make sure that you meet the minimum recommended specifications. For most Linux programs, this shouldn't be too much of a problem because, unlike Windows and Mac software, they tend to be much lighter on resources. With the Raspberry Pi 2, you certainly shouldn't have anything to worry about.

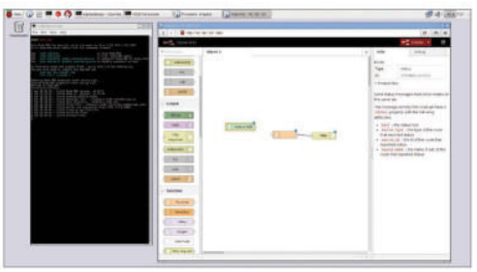
Open source software is fantastic, and for every proprietary program (such as Microsoft Word or Google Chrome) there is a free/libre equivalent that is often much less resource-intensive (such as LibreOffice Writer or Midori). Much of this software is hosted inside the Raspbian software repositories, meaning that you can download it with your standard **apt-get install** command. Places like **sourceforge.net** are also excellent sources.

Here we've rounded up a selection of some of the most useful software that you can install onto your Raspberry Pi. This is only a tiny, tiny taste, mind!



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.



Node-RED

Created by the clever people over at IBM, Node-RED is an application that enables you to program and connect together Internet of Things devices by using a graphical interface to drag, drop and link different elements, much like you would with chunks of code in Scratch.

RetroPie

Many people like to use their Raspberry Pi as an arcade emulator, enabling them to play classic games spanning everything from the Amiga to the Game Boy platforms. RetroPie is the best project for this sort of thing – just make sure you own the original games before you emulate!

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

Find packages

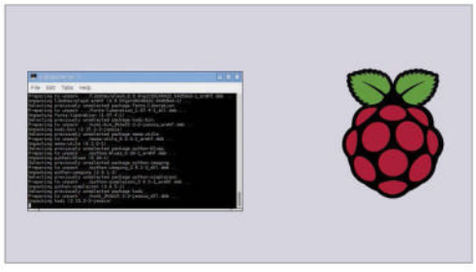
Find out what's inside the official Raspbian repositories

It's all well and good knowing that there are thousands of software packages available to download from the Raspbian repositories, but how do you know what to download in the first place? Well, often you'll hear about a piece of software and it's then a simple case of searching for it online – almost every Linux software will have a home page that details how to install it, including the package name that you'll need for your apt-get install.

Another option, though, is to hit up **www.raspberryconnect.com/raspbian-packages-list**. Here you'll find the packages in the repos broken down into categories and then alphabetised. If you have an idea of the kind of software you're looking for but don't know any names to search for, this is a great option.

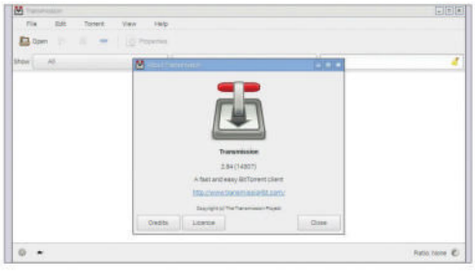
Kodi

While there are distros like OSMC and Xbian for turning your Raspberry Pi into a media server, you may just want to do that inside Raspbian. For this, go straight to the source and download Kodi, the program that the previously mentioned media server distros are all based on.



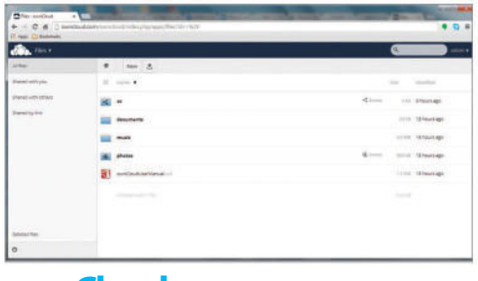
TightVNC

If you want to access your Pi remotely, you have two options. One is to use the ssh command on another computer to access the Pi on the command line. The other is to install TightVNC server on the Pi and then access the graphical interface on a PC with TightVNC client installed.



Transmission

Some people like to set up their Raspberry Pi as a torrent box, set to download their favourite podcasts or Linux distros automatically and then make them accessible on the home network. Transmission is a great torrent client for this sort of thing, and is also highly customisable.

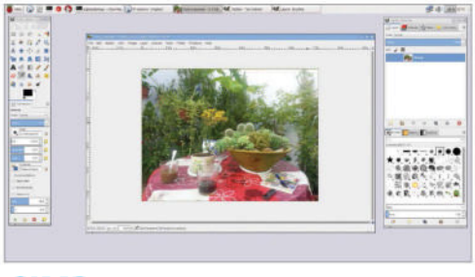


ownCloud

Don't trust services such as Dropbox with your data? Give ownCloud a shot. Once you set it up, your Pi can monitor a particular folder and, when it sees any updates to the files within, it relays those changes around to all the connected devices, just like Dropbox, iCloud or OneDrive.

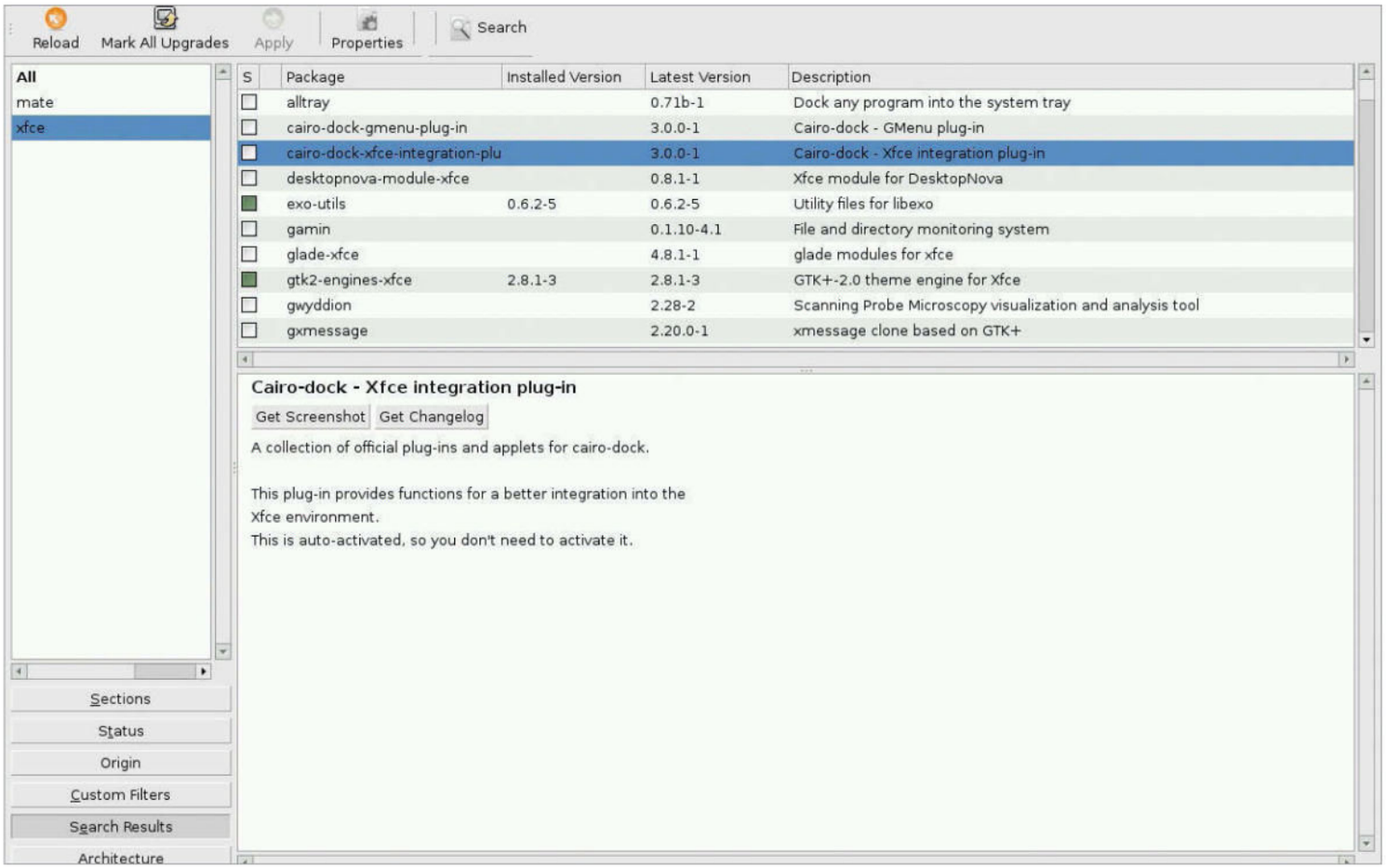
NagiosPi

The Raspberry Pi can also be used to monitor your home or office network for any untoward web traffic, boosting your security. NagiosPi is an excellent software for this, which runs on your Pi and is easily accessible and controllable through a web interface that you can go to in a browser.



GIMP

The Raspberry Pi can also be used to monitor your home or office network for any untoward web traffic, boosting your security. NagiosPi is an excellent software for this, which runs on your Pi and is easily accessible and controllable through a web interface that you can go to in a browser.



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

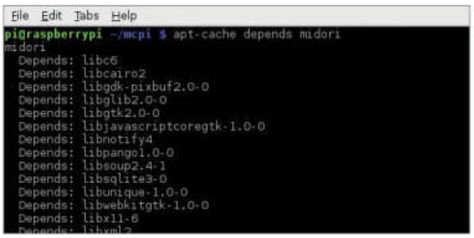
Linux is an operating system kernel – the 'hub' of the system – that was created in 1991 by Linus Torvalds, a Finnish computer science student. Linux powers servers that run a large portion of the internet as well as desktop computers and laptops, and it is also at the heart of Android-powered smartphones, 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 **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.

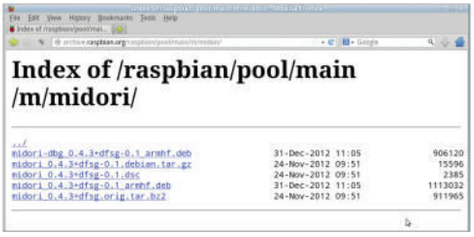
If you were to type `sudo apt-get install libxml2`, the Advanced Packaging Tool (Apt) connects to the repository, downloads the corresponding DEB file, unpacks and installs the files and updates the package database on your Pi. This means that when a new version of Libxml comes out, typing `sudo apt-get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

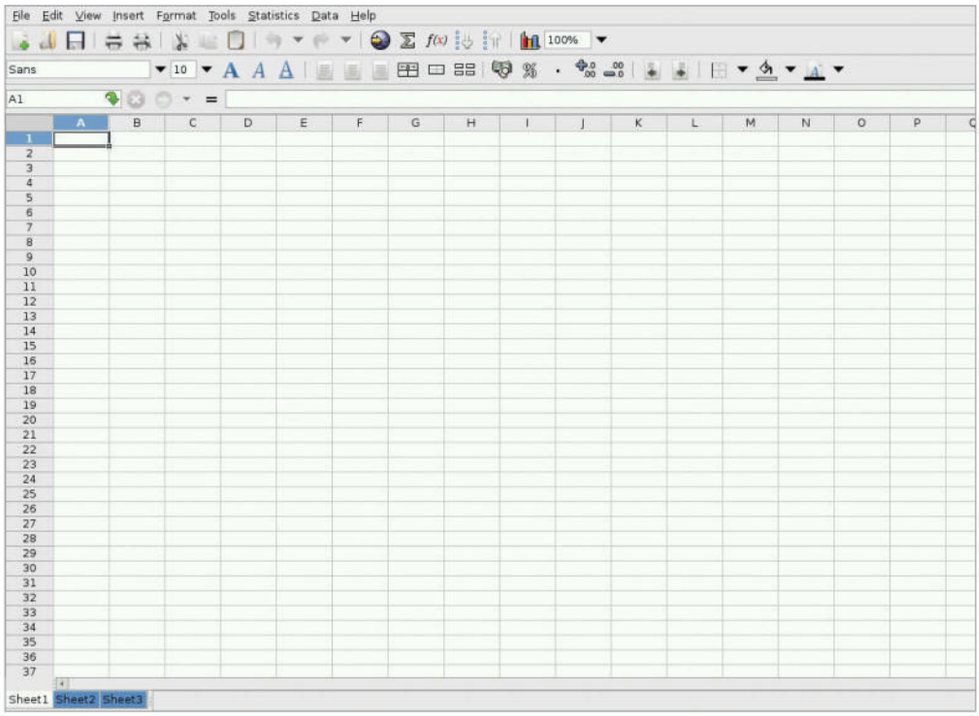
From the main image you can see that Synaptic employs a three-pane layout. The top section shows a list of packages, the sidebar has a list of categories and the main window shows a description of the currently selected package. Click on the Search icon at the top of the window and type 'word processor' 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.

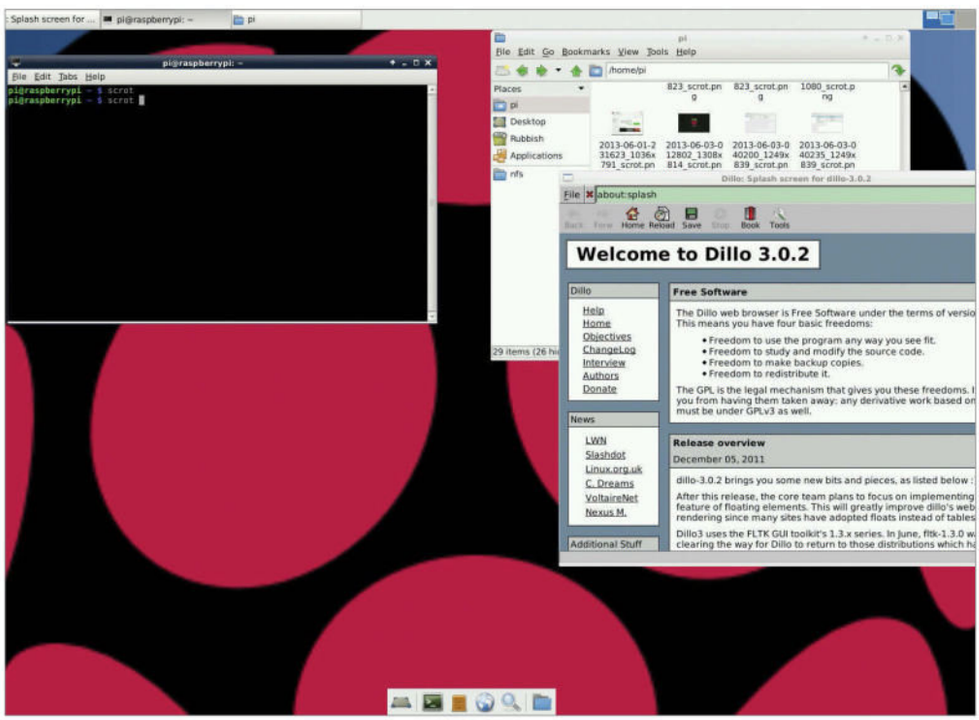
Note that the repository doesn't just contain applications and utilities. The package Xfce4 is an alternative window manager (front-end) which is efficient and fast enough for the Pi and yet more configurable than the default choice of LXDE (Fig 4). At the same time, install SLiM so that you can choose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

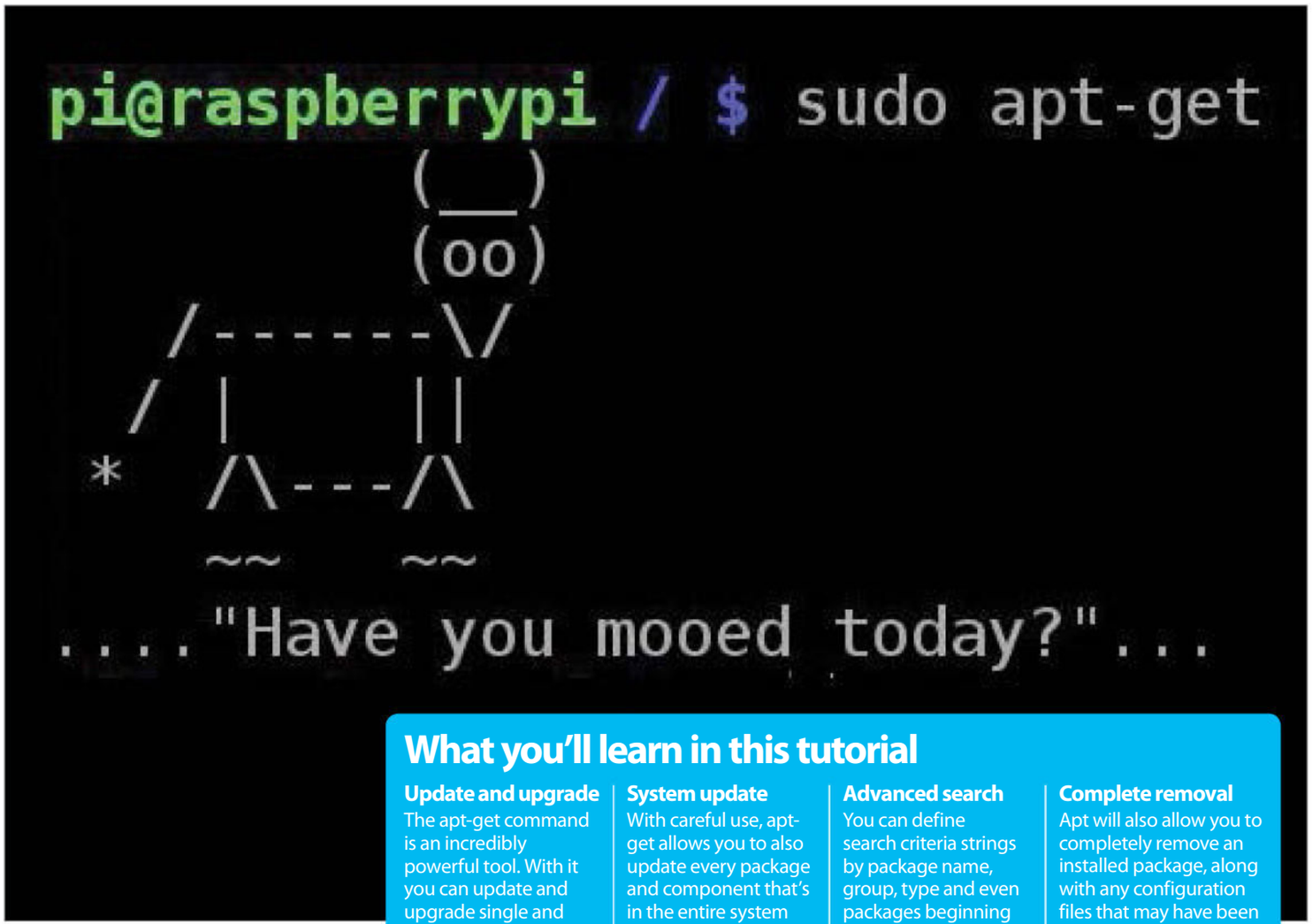


■ Fig 3: Gnumeric is an excellent spreadsheet application that is both comprehensive and light on resource usage

"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:
linux.die.net/man/8/apt

Apt-get help page:
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, 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:

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

Or…

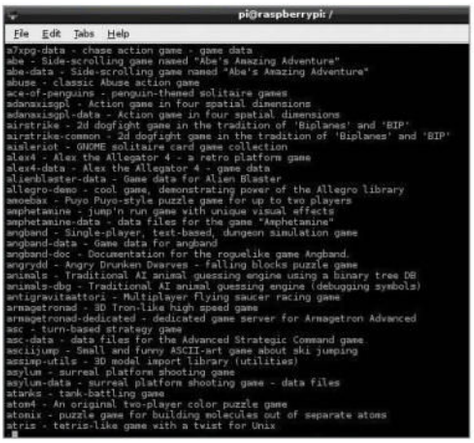
```
sudo apt-get update && sudo apt-get upgrade
```

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.

```
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:

```
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:

```
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.

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

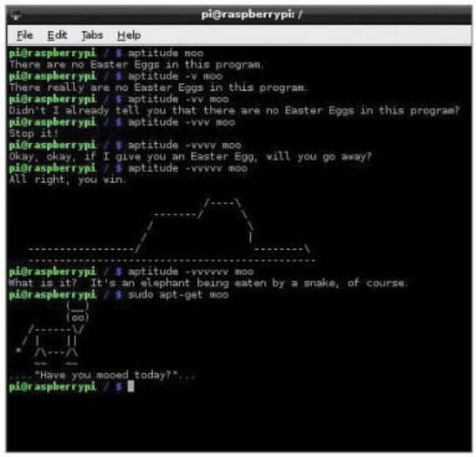
Enter 'Y' to accept the removal.

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:

```
aptitude moo
aptitude -v moo
aptitude -vv moo
aptitude -vvv moo
aptitude -vvvv moo
aptitude -vvvvv moo
aptitude -vvvvvv moo
sudo apt-get moo
```

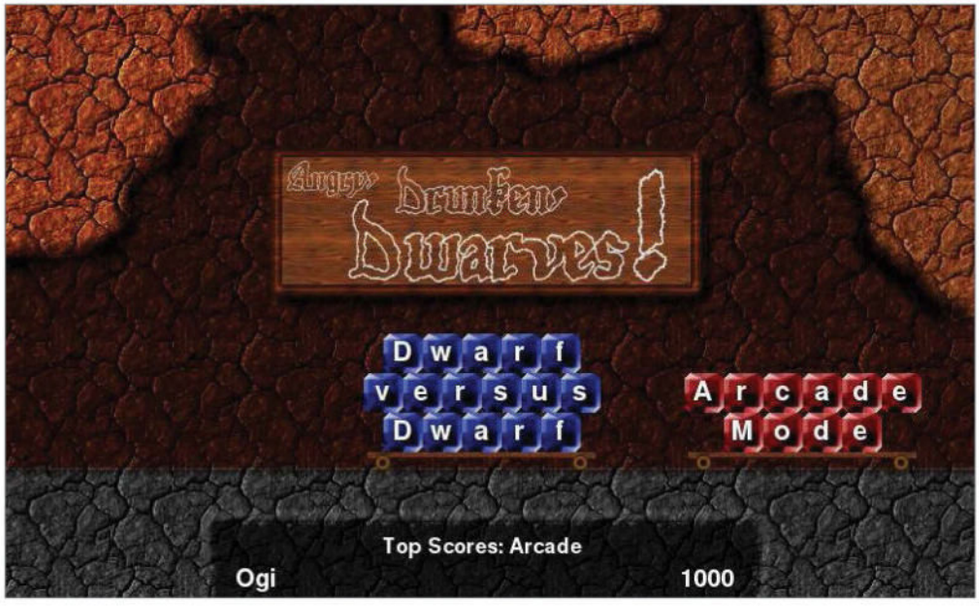


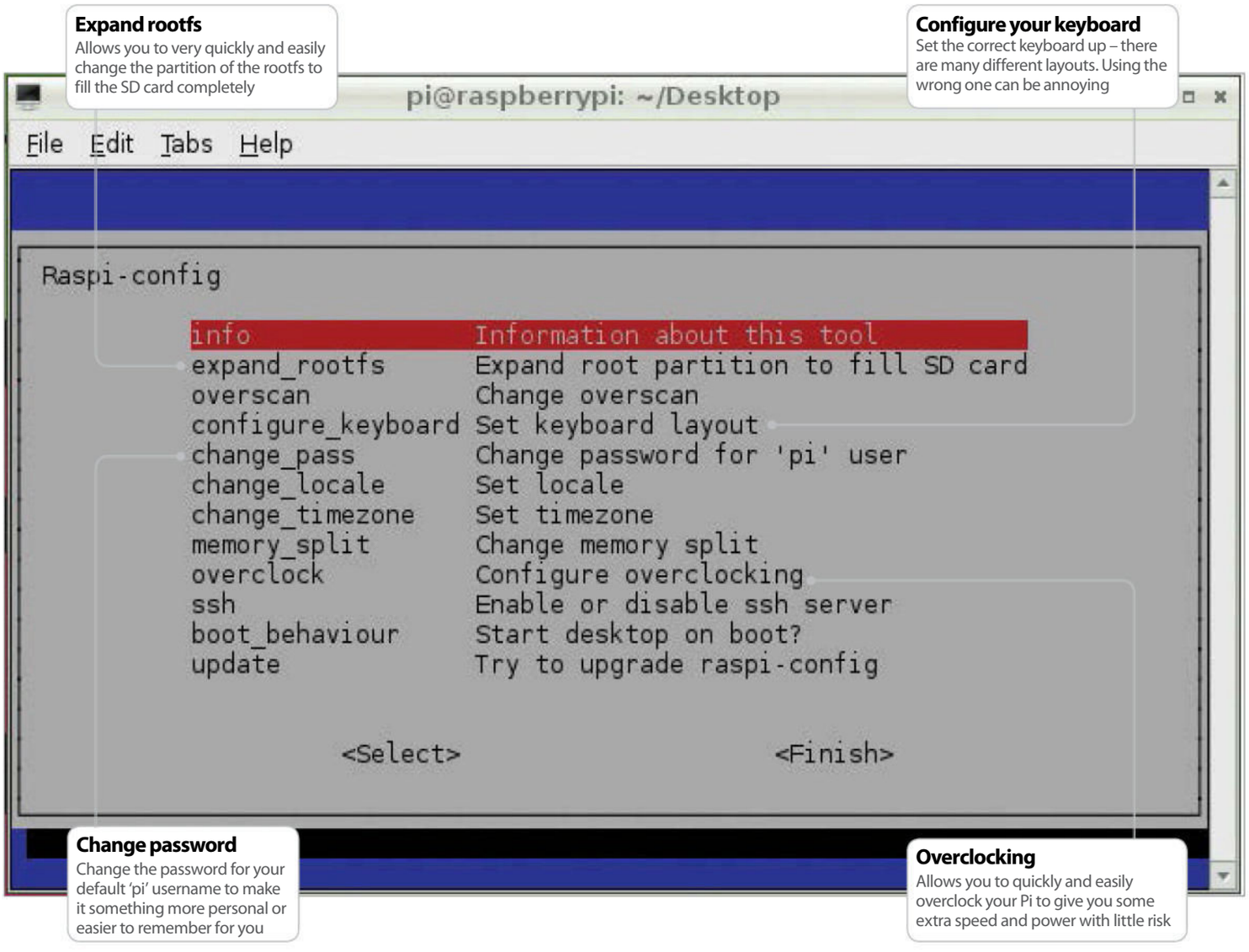
08: Man the Apt command

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:

```
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

Raspbian:
www.raspbian.org

01: Open 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.

```
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.

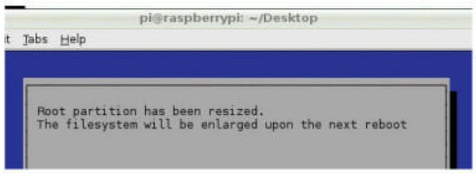
Some of the settings in this menu are important and some are irreversible, so use them with caution!

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!

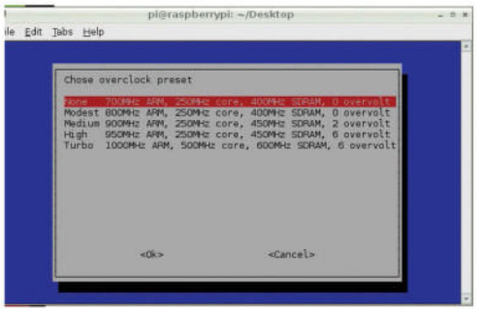
04: Overclock your Pi

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

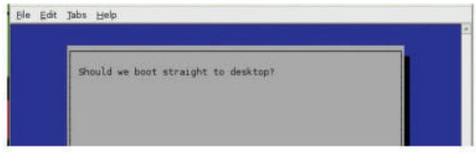
By default, the Raspbian distro will boot into a command-line interface, whereby you have to first log in as 'pi'.

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:

```
startx
```



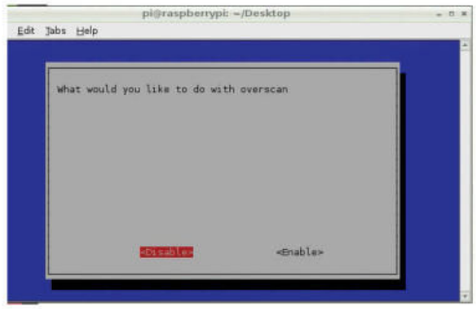
07: Turn overscan on and off

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.

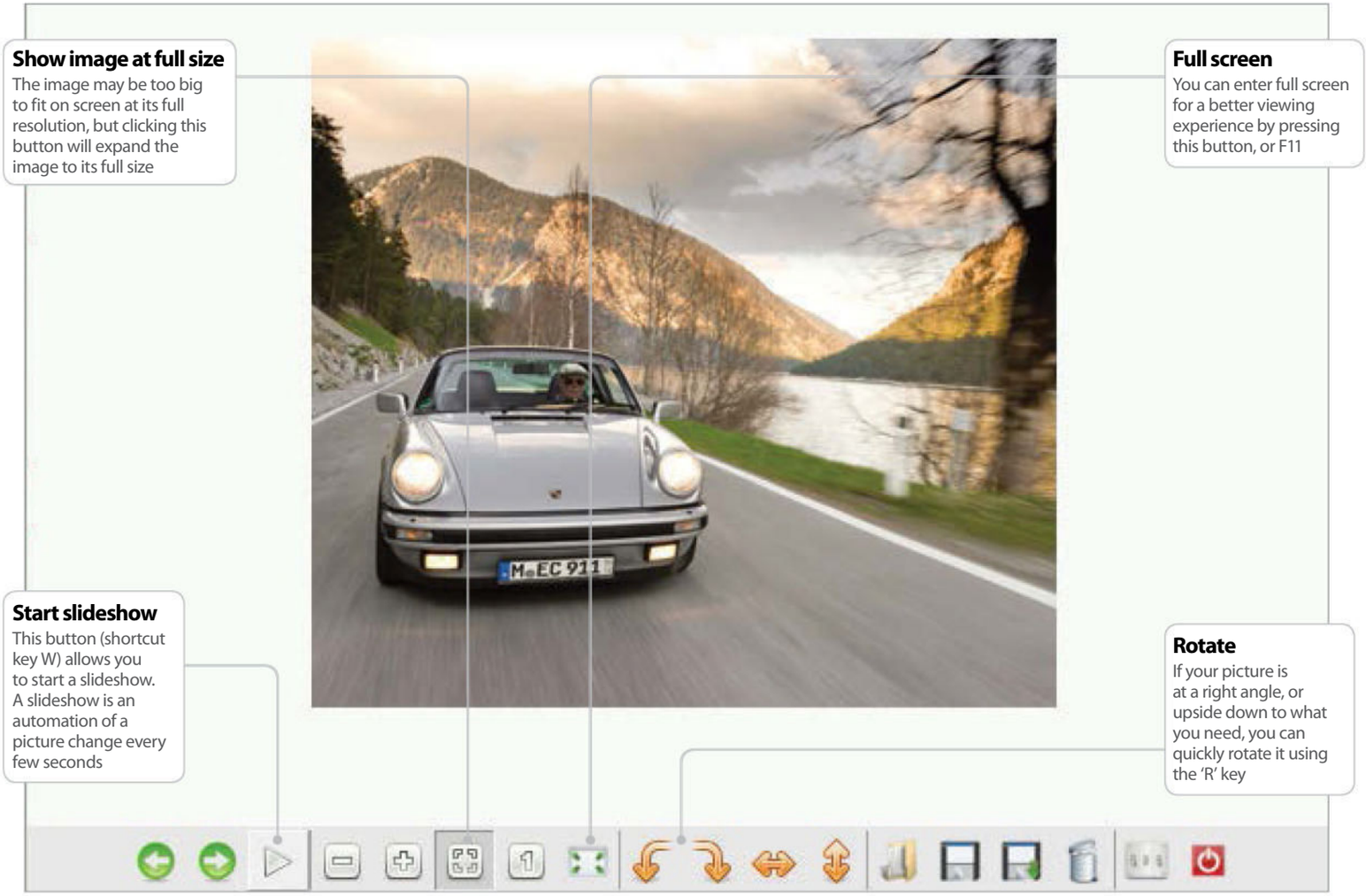


08: Update Raspi-config

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.

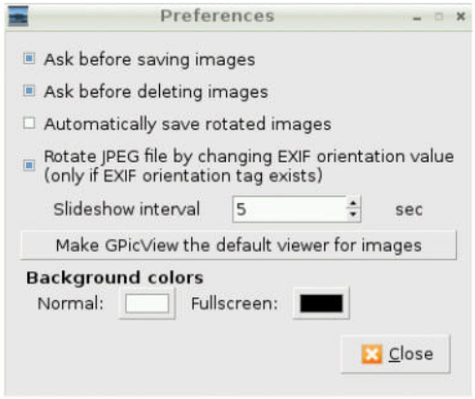
02: Let's have a look

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:

```
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.

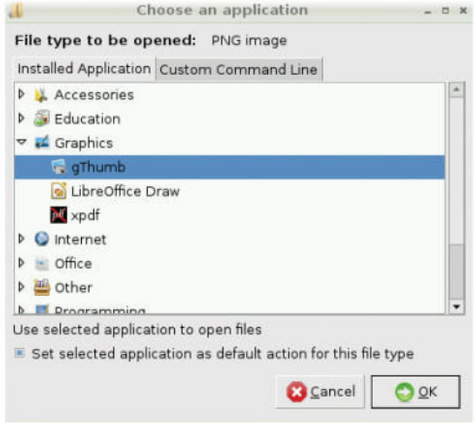
06: Play music

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.

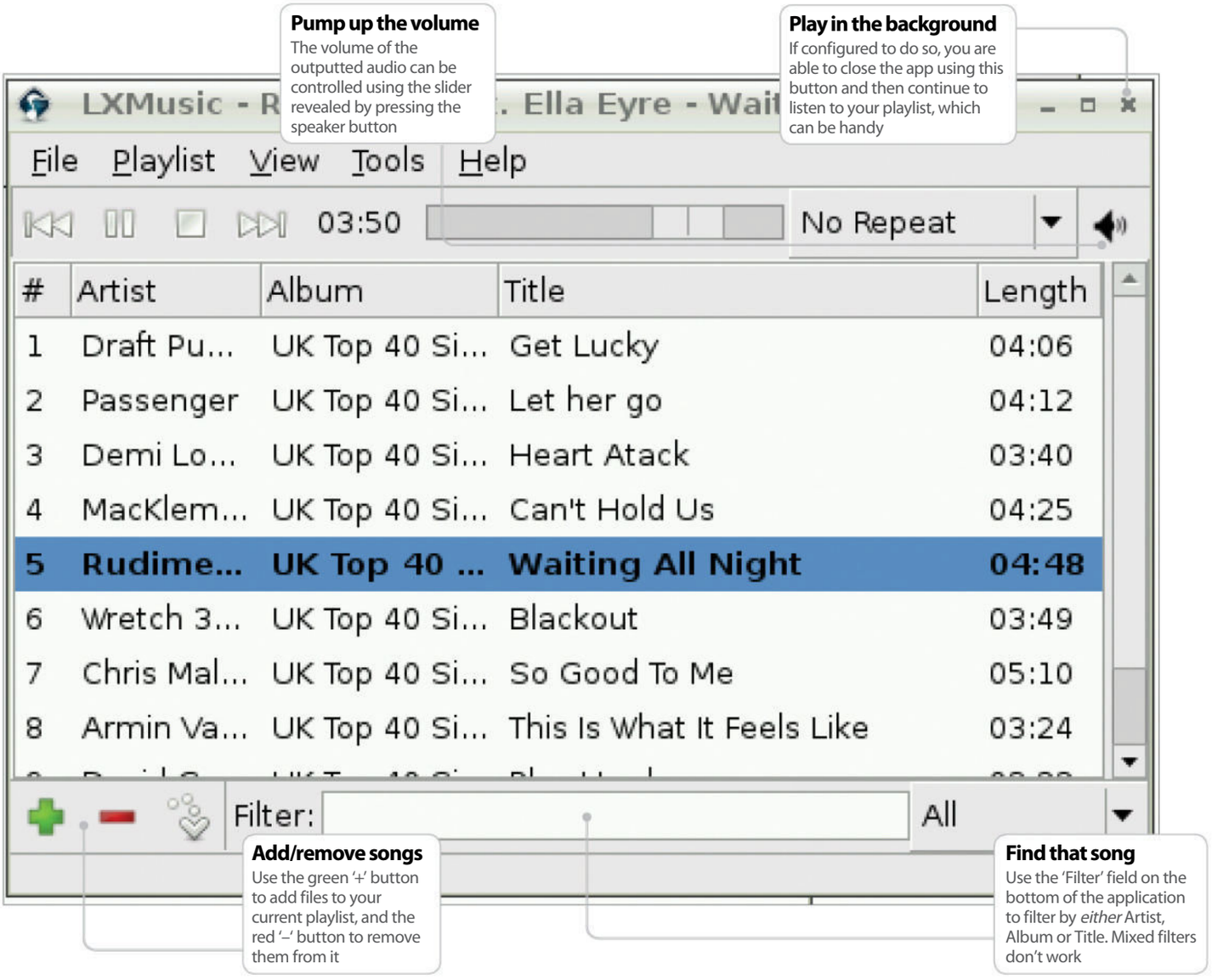
To change your default program, right-click an image and select 'Open With'. Choose either Image Viewer or gThumb, select the 'Set selected application as default' checkbox, and click OK. From now on, the application will load as default. You can experiment with whichever you prefer.



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 MP3 files on your Pi

Discover how to play MP3 files 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.

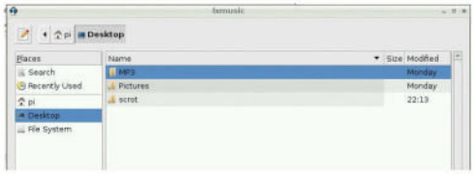
You can now access the Music Player with ease. Proceed by double-clicking the icon and actually opening the application – now we can listen to some music!

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.

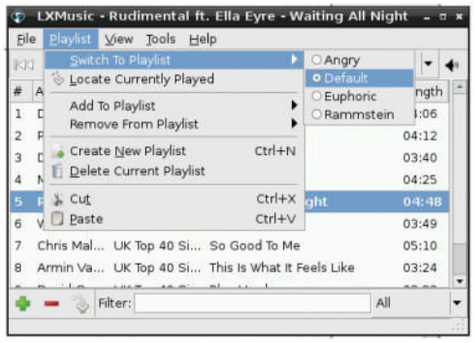
04: Manage your playlist

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

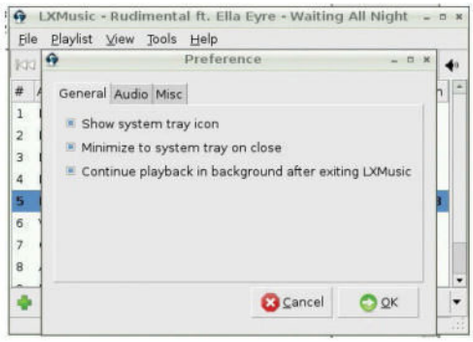
You're probably used to having more than one playlist too, aren't you? Some for different artists? Maybe a mix? Different moods? Well, we're able to manage multiple playlists in LXMusic – although you may not be used to the way that they're presented in this application. To start, create a new playlist 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:

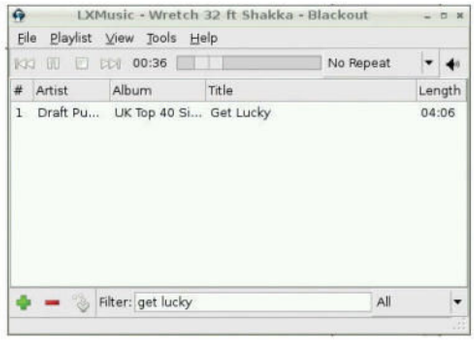
- (OPTIONAL) Show system tray icon – Show an icon in the system tray to control and get back to the music player
- (OPTIONAL) Minimise to the system tray on close – instead of exiting the application, just minimise (allows playing of music without the window)
- (OPTIONAL) Continue playback in background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

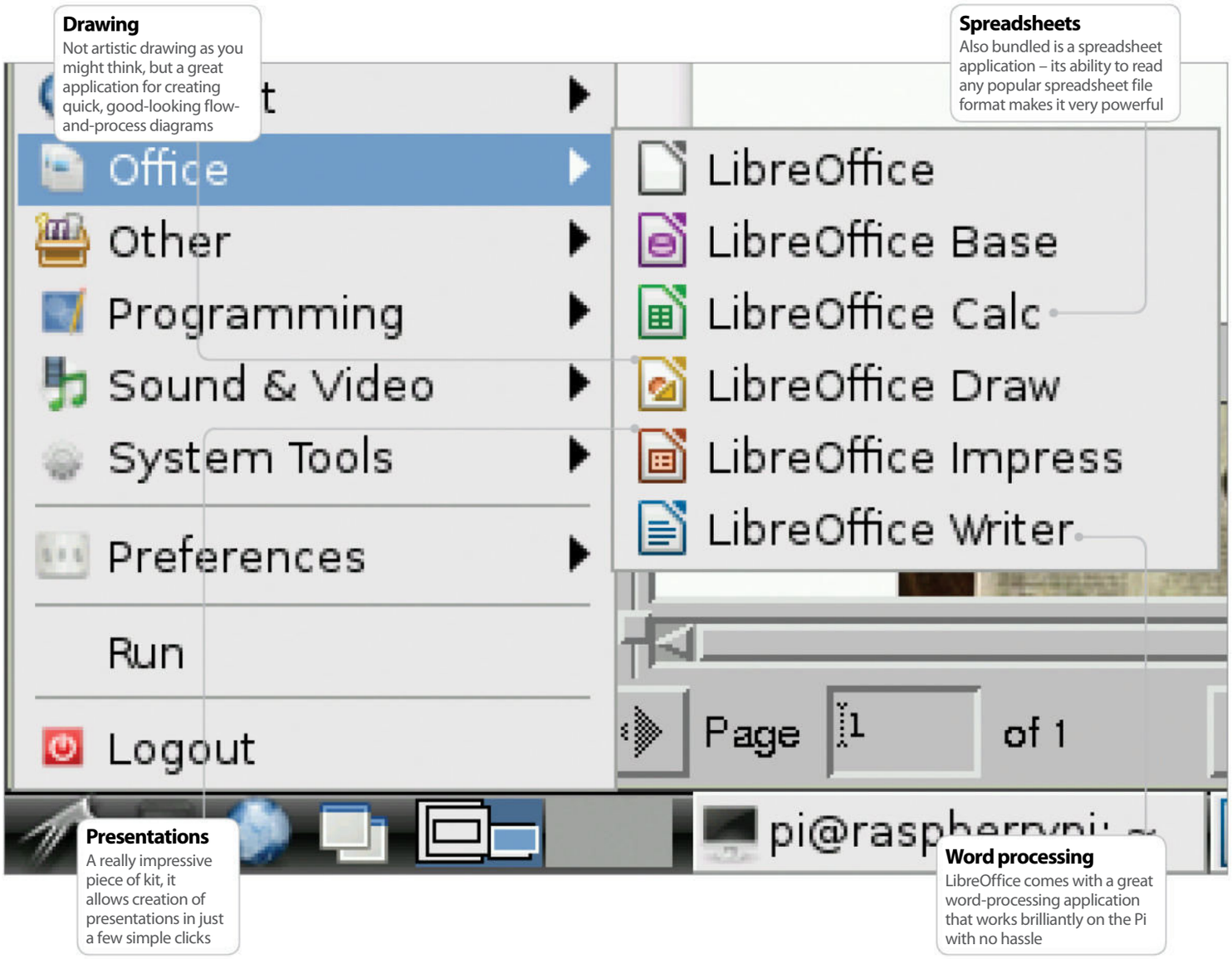


08: Odds and ends

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.

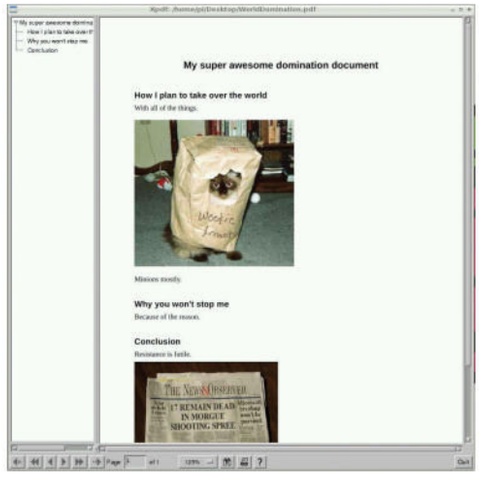
Alternatively, you can use the Insert>Picture>From File option – that gives you an open dialog to do the same thing.

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.

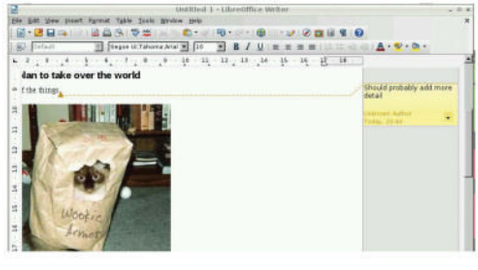
The main advantage of PDFs, however, is that you don't really need to worry about the recipient having a specific application to look at them in – as most modern operating systems will have some sort of PDF viewer built in.

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.



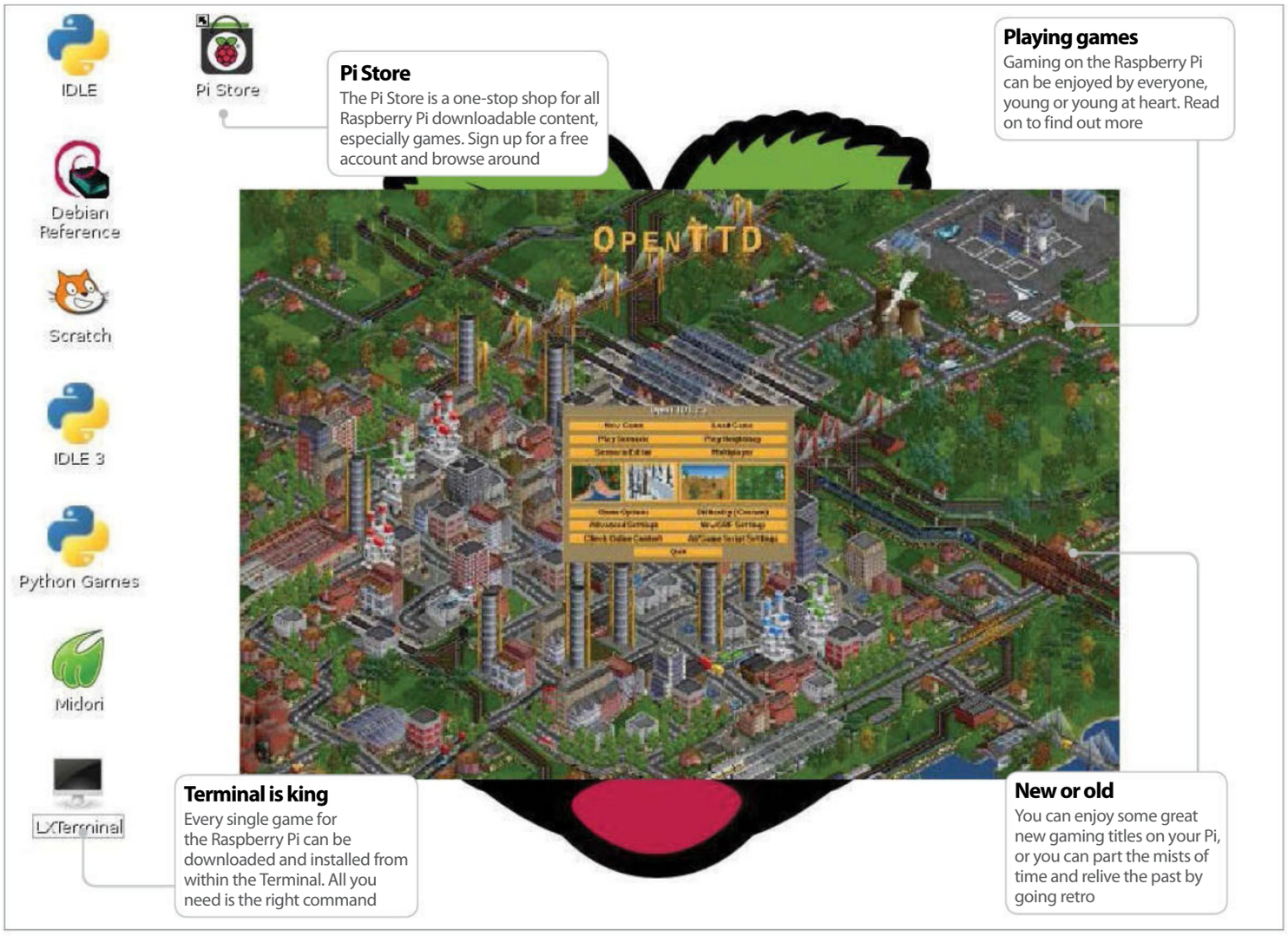
07: Help yourself by using comments

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.

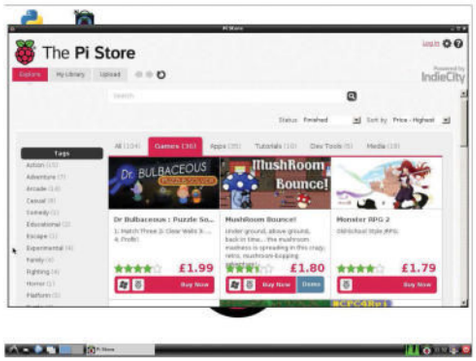
Resources

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.

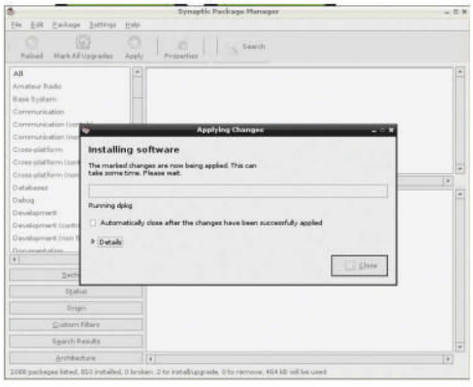
To install Synaptic, simply drop into a Terminal and enter the following command:

```
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'.



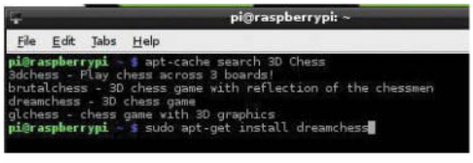

```
Click to 'Mark for Installation'
In the pop-up window, click the 'Mark' button
Click 'Apply' located in the top menu to install
Click the 'Apply button in the pop-up window
Click 'Close' when the operation is complete
```

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

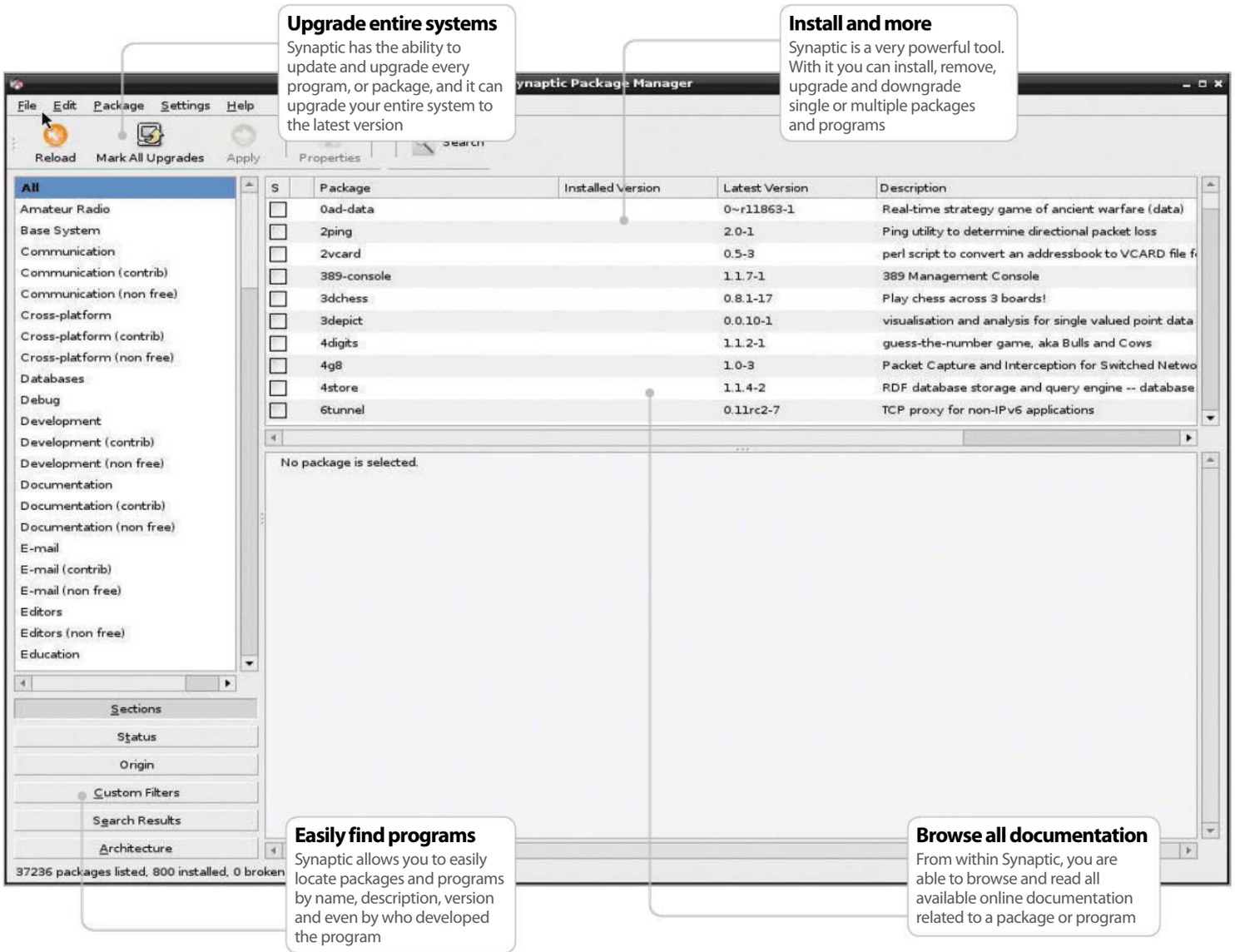
While it would be great to list every game available for the Raspberry Pi, unfortunately we can't do this.

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.



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.

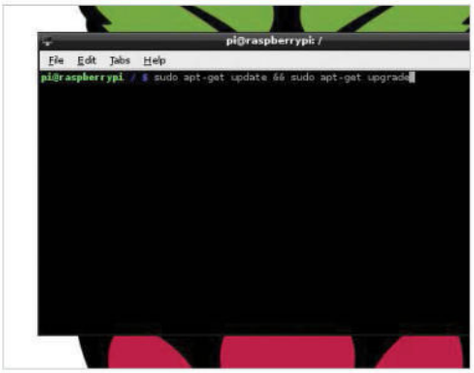
Resources

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:

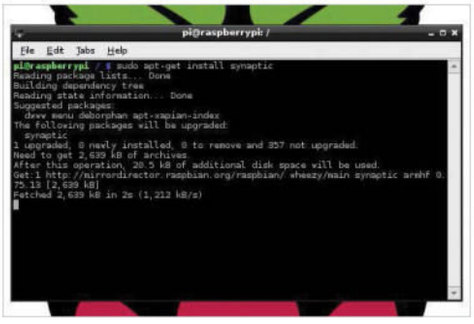
```
sudo apt-get update
sudo apt-get upgrade
```



02: Install 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
```



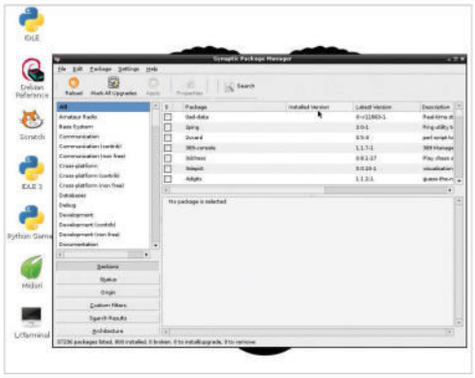
03: Run Synaptic – Part 1

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 Run 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: Run 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 Fix 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: Fix 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…

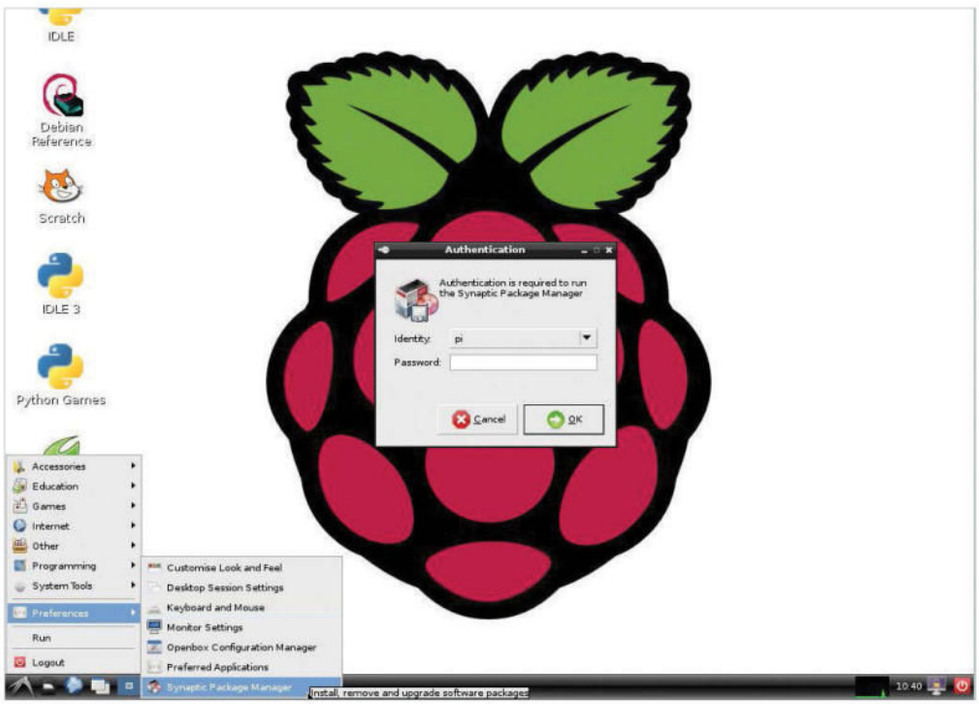
```
Exec=gksudo synaptic-pkexec
```

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:

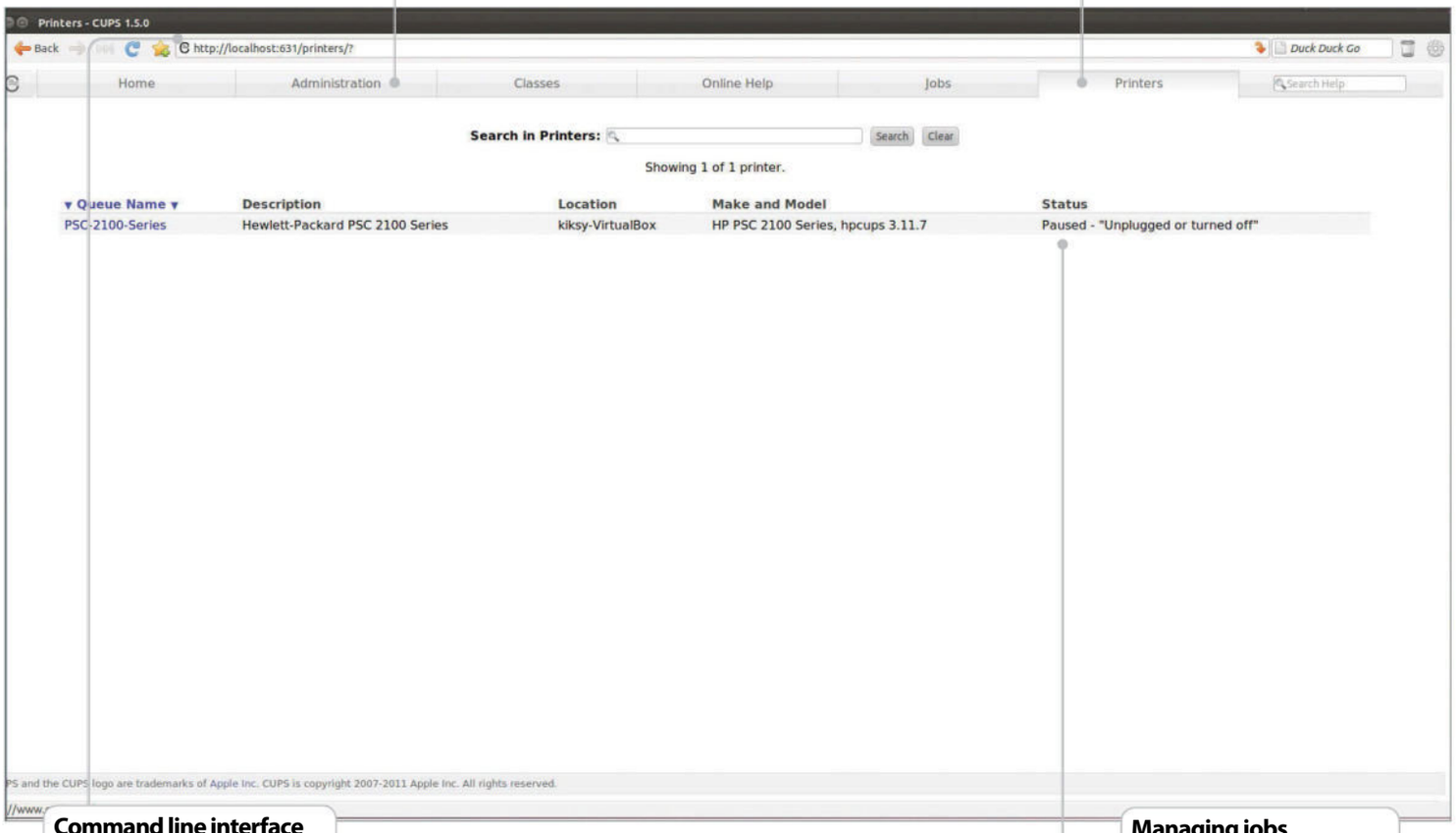
```
gksudo synaptic
```

"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

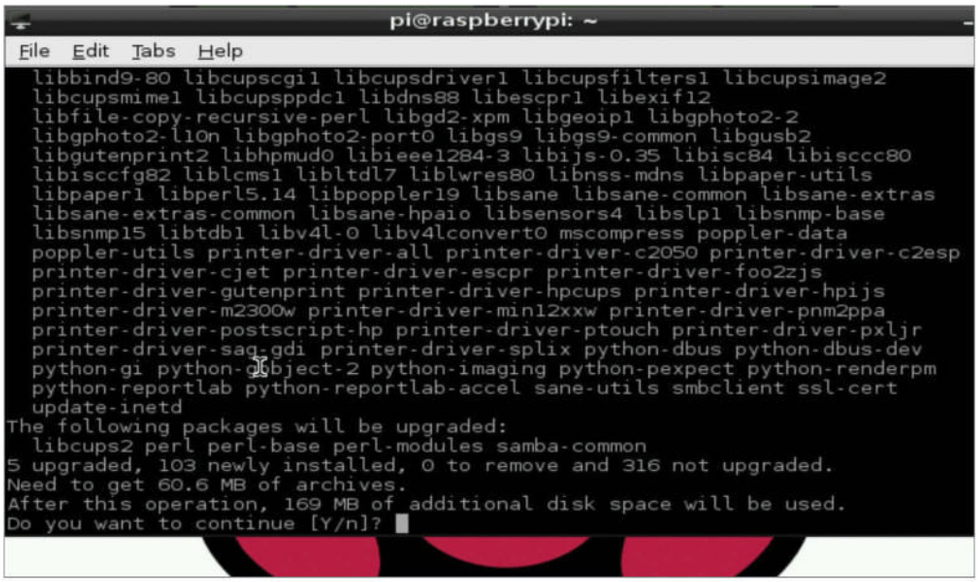
With its diminutive size, the Pi makes a great portable computer; just pop it in a bag and plug it in to any TV or projector with an HDMI port. One reason people carry personal computers around with them is for basic word-processing. Of course, the Pi with Raspbian shines at this, as it has access to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

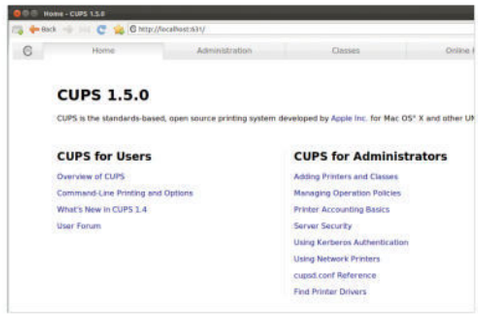
```
<Location/>
Order allow,deny
Allow localhost
Allow 172.20.22.*
</Location>
```

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.

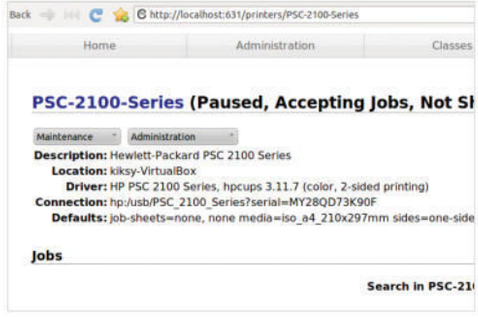
```
http://localhost:631/
```



■ 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'.

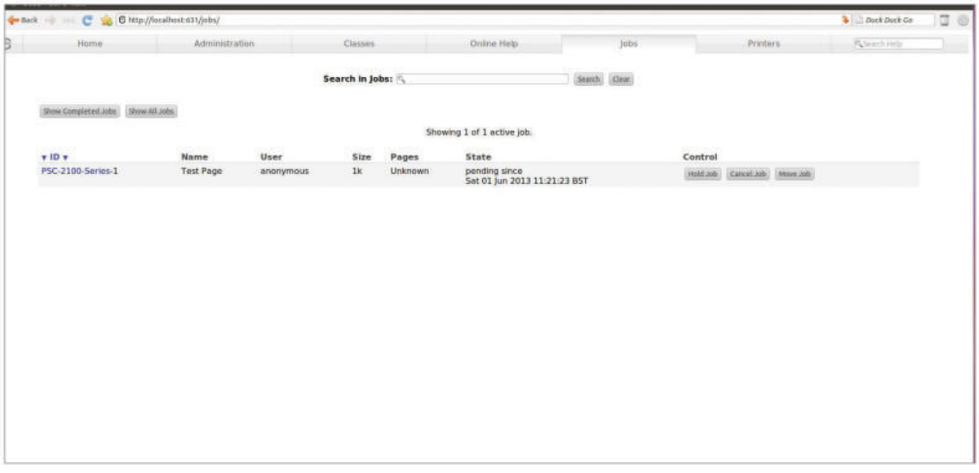


■ Step 7: It's useful to print a test page in order to see if your selected printer is working correctly

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"

"Using this module, you can use the camera to take photos and video footage"

What you'll need for this project

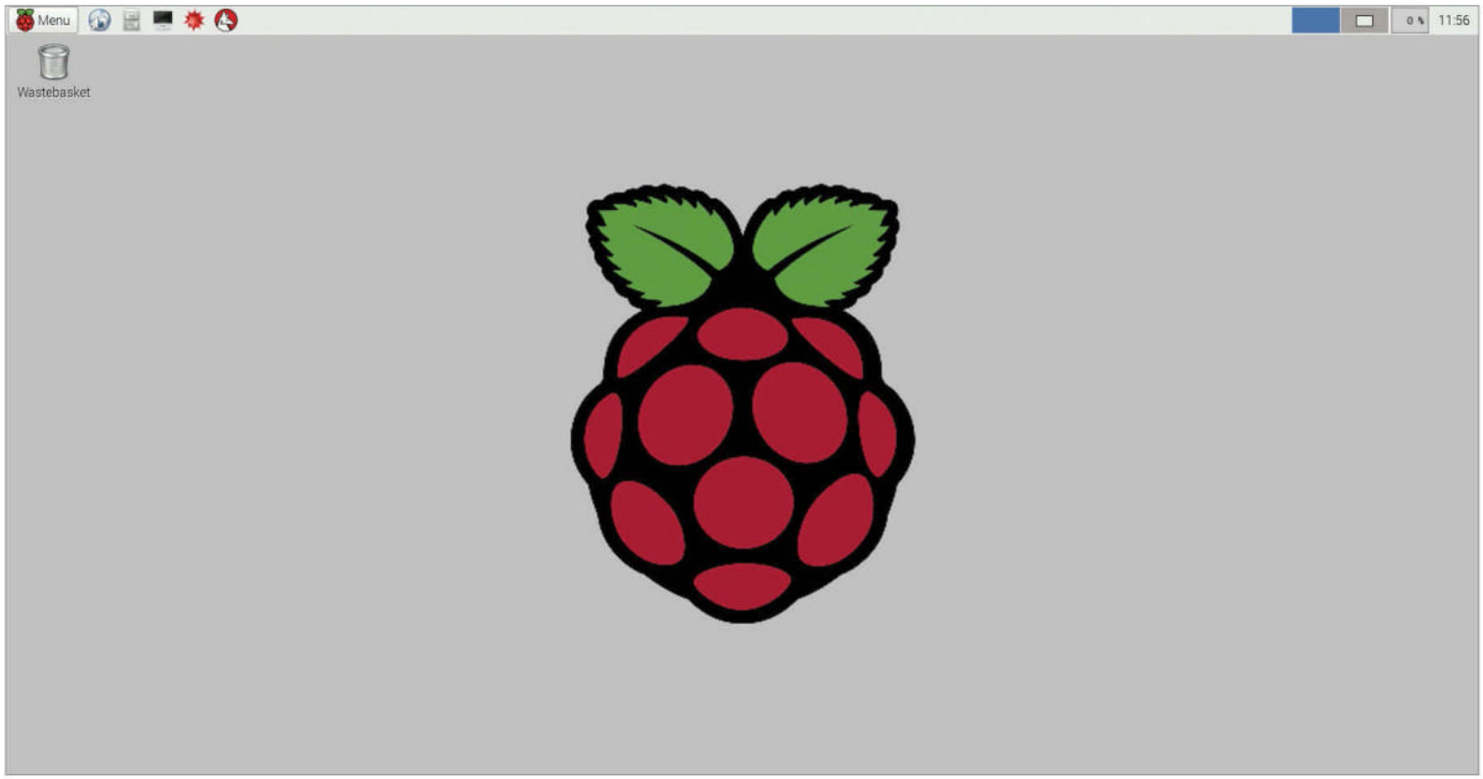
- **Raspberry Pi 2**
- **Raspbian** raspberrypi.org/downloads
- **Keyboard**
- **Mouse**
- **Wireless dongle**
- **Monitor**
- **Case**
- **Powered USB hub** (optional)

Make a Pi 2 desktop PC

Use your Raspberry Pi as a desktop replacement PC thanks to the increased power of the Raspberry Pi 2

The Raspberry Pi 2's increased power over its predecessor is well-documented by now. More CPU cores and more RAM making it six times faster is an impressive number, and you can see the actual changes that it makes to the experience.

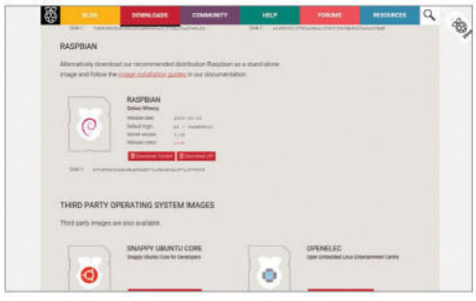
This power actually enables you to conduct a very simple project that was just out of reach for the original Raspberry Pi: a Raspberry Pi desktop PC. All the components for it were available, but the Pi was just a little too slow to properly give a fluid desktop experience. Now, with the improved resources, many of the previous restrictions are gone – enough of them to be able to build a Pi desktop. So grab a Pi 2 and we'll get started.



■ Above: Make sure you set the Pi to boot straight to the desktop, like your main computer

01: Get Raspbian

We will be using Raspbian for our desktop Pi. Not only is it simple to obtain and easy to use, but it is supported by the Pi Foundation and community, which means it's going to be the most flexible operating system with the most choices for a desktop. Download it from: www.raspberrypi.org/downloads.



02: Install Raspbian

Once Raspbian is downloaded, you can install it to your SD card. Put the micro SD into an SD card reader and connect it to your main system (a PC or laptop).

Open up the terminal, cd to the location of the image and use:

```
$ sudo dd bs=1M if=raspbian.img of=/dev/
[location of SD card]
```

03: Setup options

On first boot there will be some setup elements that it is necessary you go through. The most important things to do for this desktop are to first hit 'Enable Boot to Desktop' and then to extend the installation to fill the entire SD card. After you have done that, do anything else that you want to do in these menus and then reboot before moving on to the next step.

04: First boot

You will boot into a fresh version of Raspbian with the newer interface and default apps available to use. From here you can start using it as normal if you wish, but it is worth noting that there are a few extra steps that you should take to make it truly desktop worthy.

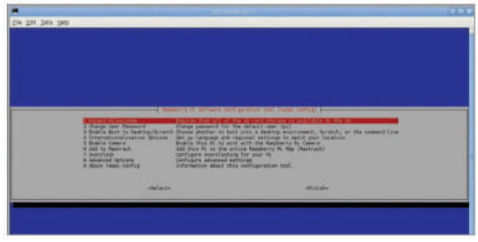
05: Software updates

Our first step is to perform an upgrade on the system to make sure it's all up to date and working properly. To do this, open up the terminal from the menus and use:

```
$ sudo apt-get update
```

… to refresh the software list, followed by the next command to then upgrade to newer software:

```
$ sudo apt-get update
```



06: Firmware upgrade

While we're updating, it's a good idea to upgrade the firmware on the device. Still in the terminal, you'll want to activate the firmware upgrade software with:

```
$ sudo rpi-update
```

07: Extra configuration

At this point, you might want to tweak the Pi a little further. To bring up the initial configuration screen, you'll need to go back into the terminal and launch it with:

```
$ sudo raspi-config
```

08: Advanced options

From here you can activate some extra options that you might need in the future. Enabling the Pi camera driver is a good first step, and you can even have it boot to Scratch if you want to focus on fun game development. Otherwise, there are also some overclocking options that you can consider if the system starts getting slow for you.

09: Accessorise your Pi

Just setting up the operating system on the Raspberry Pi is only a small part of the process – we also have to consider the hardware surrounding it that will actually make it usable as a desktop replacement.

10: Human input devices

Standard USB keyboards and mice are best suited for this task, much more so than a lot of the wireless keyboard and mouse combos that are popular among Pi users.

However, it is important you don't try to save on USB ports by getting a keyboard with USB connections of its own: the Pi cannot power USB hubs, even just two on a keyboard.

11: Monitor to see

The Pi can output a maximum of 1080p, which in normal display terms is 1920 x 1080. While it only outputs in HDMI, a lot of modern monitors do have an HDMI input. If you don't want to get a brand new monitor though, you can always get a HDMI to DVI or HDMI to VGA adapter.

12: Case for protection

The Pi is pretty sturdy and we'd be lying if we said we didn't regularly keep ours out of a case, however, it's not indestructible. While we're doing a lot of different projects involving accessing different components, a desktop Pi doesn't require this level of access. We like the Pimoroni Pibow cases, but there are several other secure, protective alternatives.

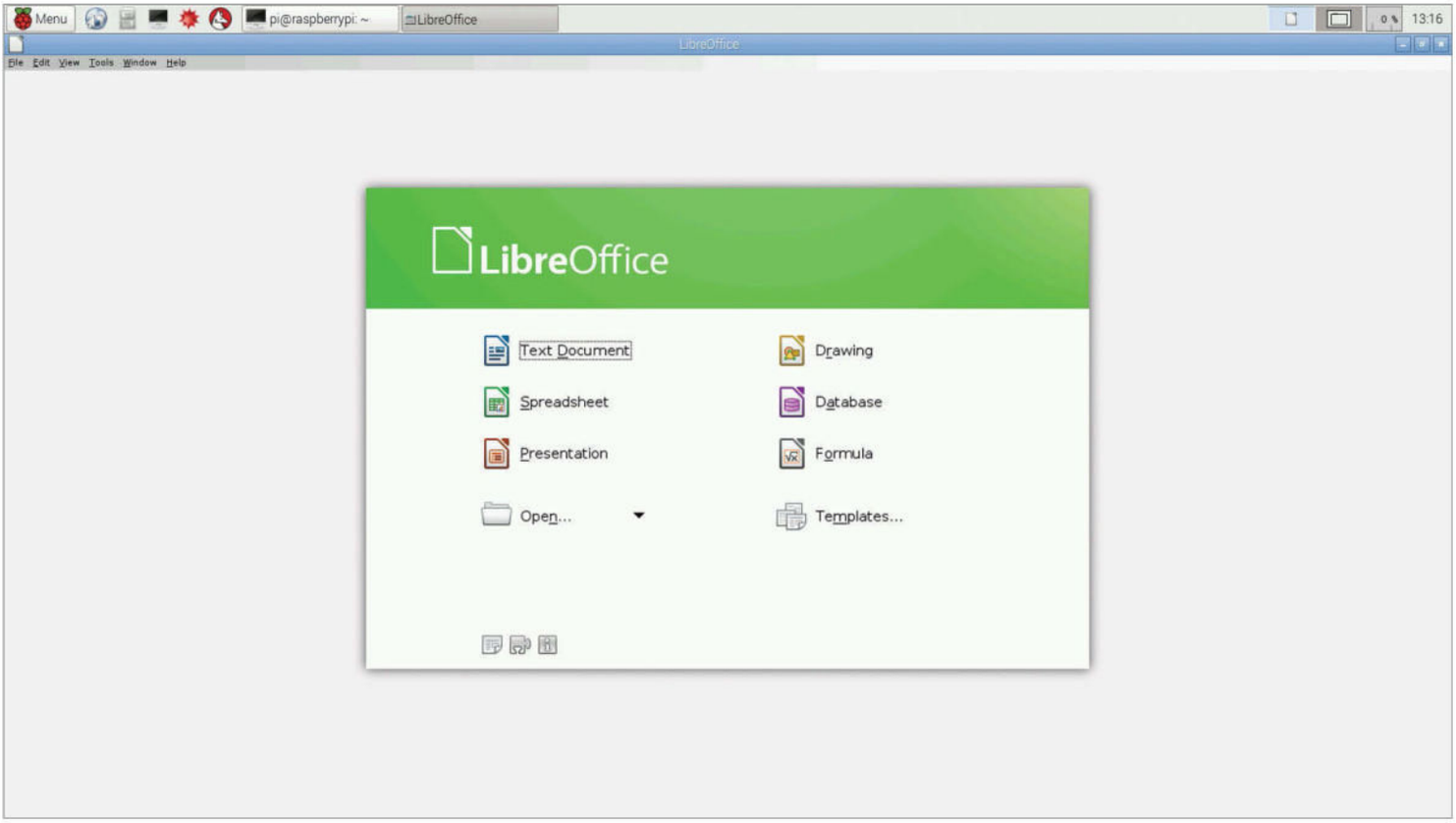
13: Wireless for Internet

While some people are fine using wired connections, not everyone has that luxury. Wireless dongles are a perfect fit for the Pi, especially now they're almost no larger than the USB port themselves. However, not just any dongle will work and you'll have to check against this list to make sure that you get a compatible one: **elinux.org/RPi_USB_Wi-Fi_Adapters**.

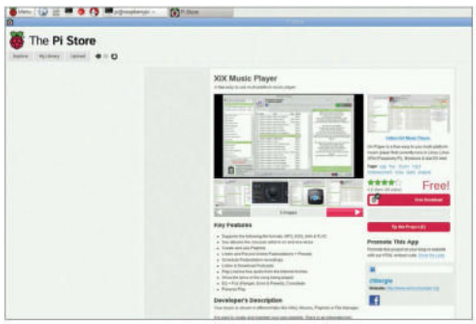
14: Anything else?

Our standard desktop PC setup is complete, with one USB port to spare. You can use that single port for USB sticks or portable storage, or you can invest in a powered USB hub to give yourself more connectivity options. Otherwise, investing in a good, 2A power supply will make sure you're never short on power for anything.

■ Left: You can run a pretty decent Minecraft server for a handful of your friends with the Pi 2



■ Above: With the Pi 2, you can run a full office suite without running into any awkward slowdown



"LibreOffice is not installed by default, but as the premier open source office suite, or Linux office suite in general, it is readily available to be used on Raspbian, which is useful"

15: Add extra software

We're not quite done getting our Raspberry Pi desktop ready just yet. We need to add some extra software to make it feel more like a real desktop. While we already have a browser installed and some of the basics, the first other piece of software we should add is an office suite.

16: Work with LibreOffice

LibreOffice is not installed by default, but as the premier open source office suite, or Linux office suite in general, it is readily available on Raspbian, which is useful. Open up the terminal and install it with the following:

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

17: GIMP for photos

This is the big one – while the original Pi was not quite able to handle LibreOffice, it was useless trying to use GIMP. Now GIMP works just fine, although more complex tasks might make it slow down just a touch. Install it with:

```
$ sudo apt-get install gimp
```

18: XiX for music

If you would like to be able to listen to music while you work, one of the best pieces of software to check out is XiX. It's available to download on the Pi Store. It's a free download and you can find the Pi Store in the Pi menus to install it.

19: Pi for desktop

Now we are set up you can start properly using your Raspberry Pi as a desktop system, while still making use of the educational functions it is capable of as and when you need to use them. The software repository and Pi Store should contain any other software that you would want or need to make the most of your new Pi system on your desktop.

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: Set 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, as well as internet access.

02: Star 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: Update 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: Add 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
```

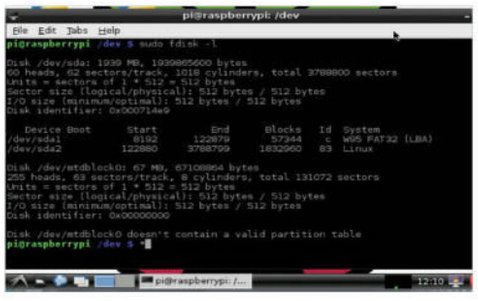
05: Detecting mounted devices

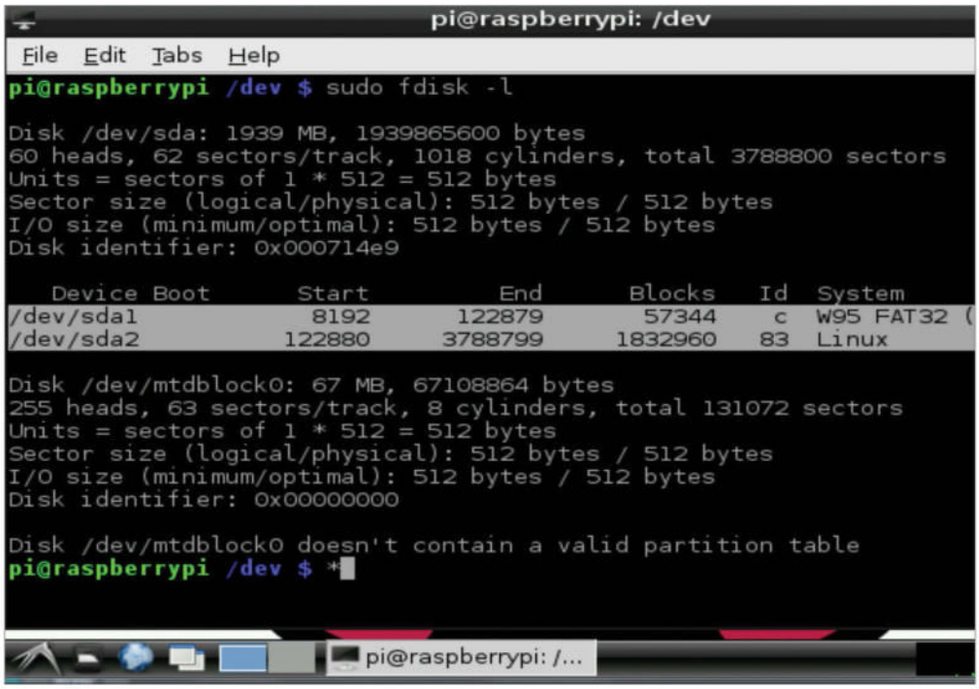
It's necessary to set up all the mount points for each of the hard drives we have attached to the Pi, and that requires a little more work than in Windows or OS X. Firstly we have to list all the attached devices, using fdisk.

```
sudo fdisk -l
```

06: Understand 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
```

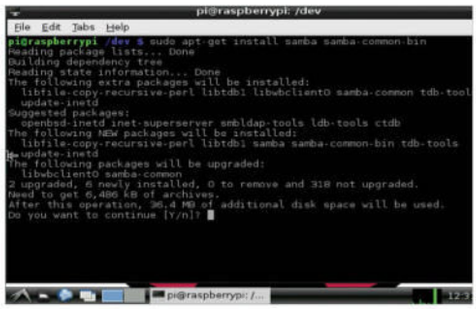
09: Mount drives

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: Install 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

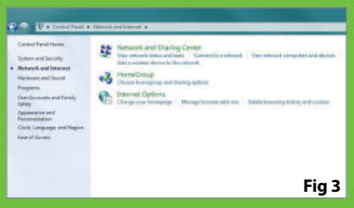


Fig 3

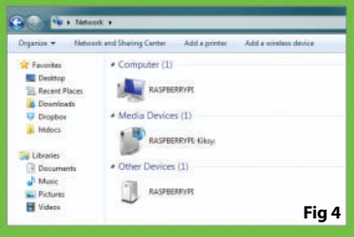


Fig 4

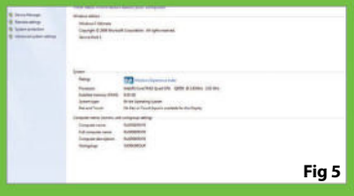


Fig 5

■ 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: Set 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: Set security options

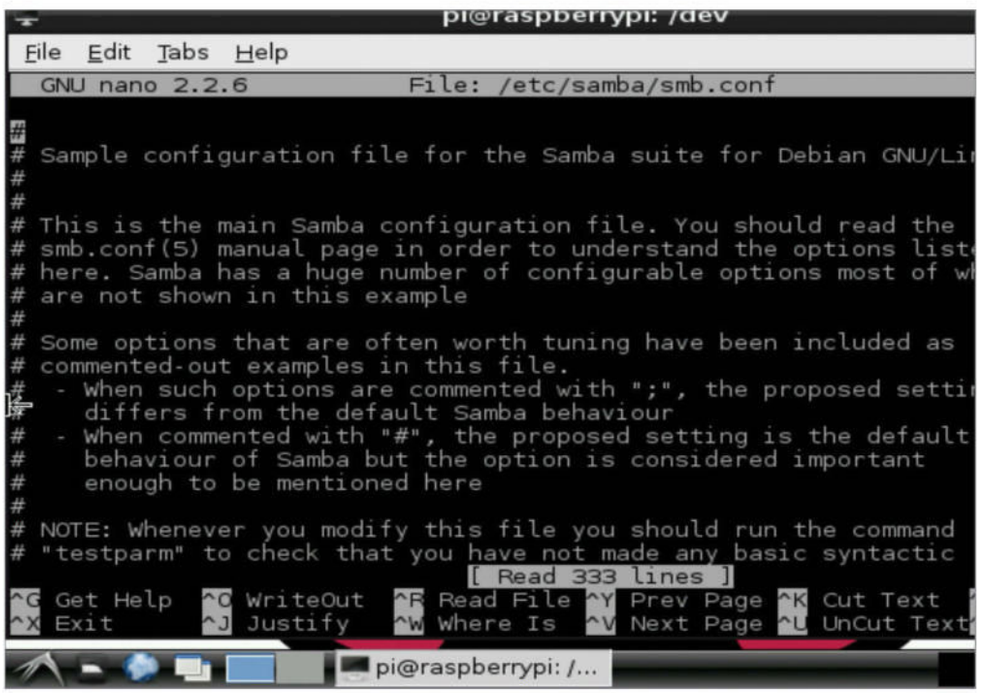
By default there is no security on your NAS shares. If you're okay with this, then you can skip this step. If not, then you need to scroll down and uncomment the 'security = user' line by removing the # symbol. This is found in the #Authentication# section.

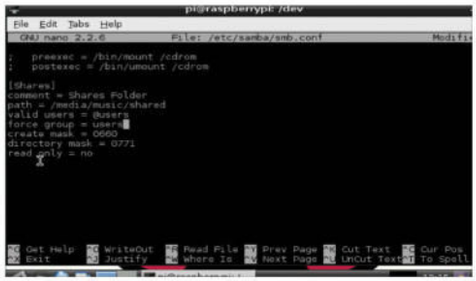
```
security = user
```

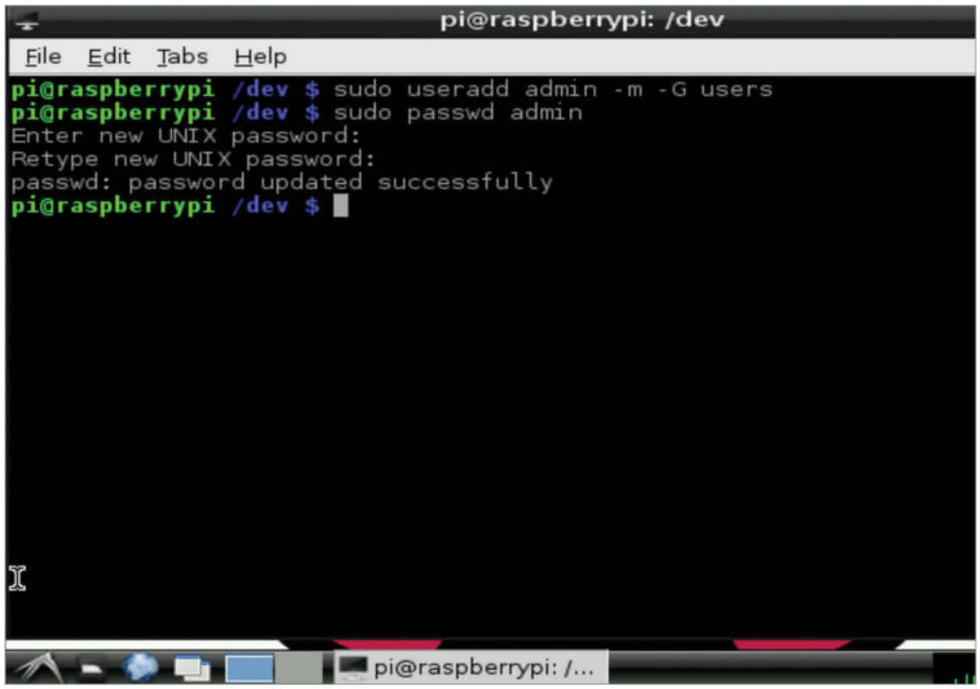
16: Add a share

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: Add 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
```

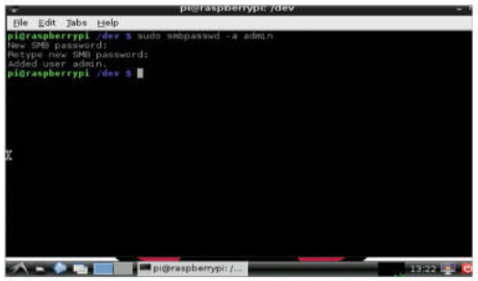
18: Add user to Samba

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
```

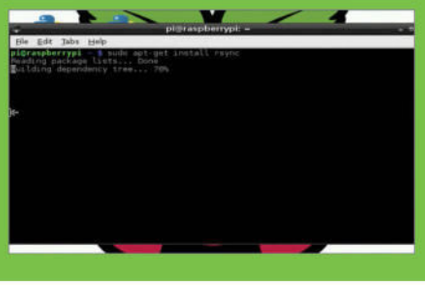
Then we need to add this line into the table. Obviously you'll need to edit the actual location of the directory or device you wish to back up, as well as the target drive.

```
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.



What you'll need for this project

- **OpenELEC** openelec.tv
- **HDMI cable**
- **USB IR receiver**
- **IR remote**
- **Case**
- **Dedicated power supply**
- **Optional USB storage**

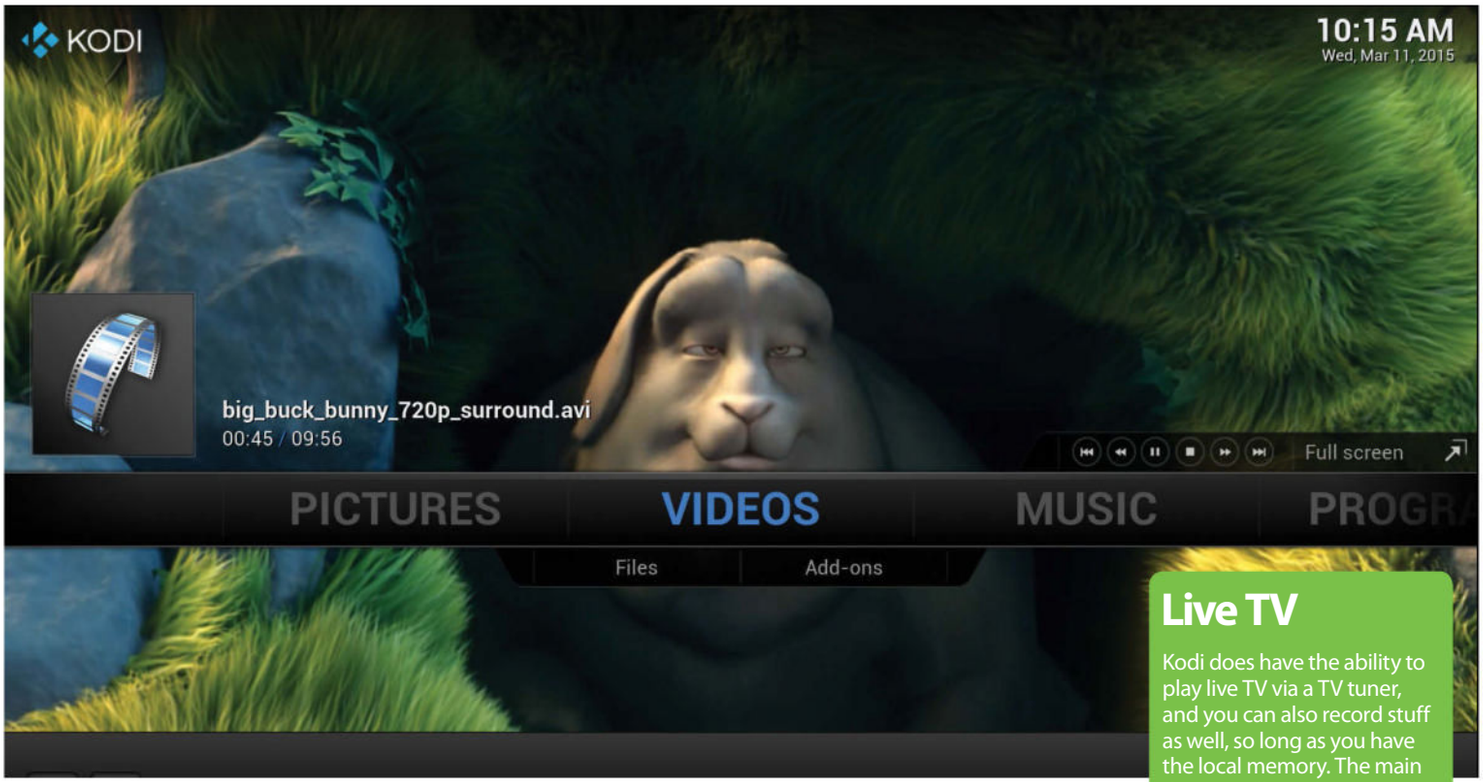
Make a Raspberry Pi 2 HTPC

Create a more powerful and capable HTPC using the Raspberry Pi 2 and the excellent OpenELEC project

We know people who just have a Raspberry Pi for XBMC, now called Kodi. It's a great idea and a great use for the Pi – it works just well enough that you can easily play media locally or over the network.

The biggest issue came with GUI response on the original Model Bs, and a lack of USB ports for connecting up everything that you want. While optimisation over the last few years has helped, the leap to Raspberry Pi 2 has basically solved all of these problems by giving you much more powerful hardware to play with.

This handy guide will help you make the most of your gadget's advanced functionality by teaching you to create the perfect Raspberry Pi 2 HTPC.



■ Above: Kodi really is designed to be used with a remote and there are some great guides to using them on the OpenELEC site: **bit.ly/1B0AERv**

Live TV

Kodi does have the ability to play live TV via a TV tuner, and you can also record stuff as well, so long as you have the local memory. The main thing you'll need to invest in is a compatible TV tuner, a list of these is available here: **bit.ly/1r3mEVj**

01: Choose the software

In the past, Pi HTPCs were just a choice between RaspBMC and OpenELEC. However, RaspBMC is on a bit of a hiatus and OpenELEC is your best bet for getting the most up-to-date software. There's not a massive difference between the two, as they both run XBMC.

02: Get the software

Head over to **openelec.tv** and look for the Download section. There's a specific Raspberry Pi section which is split up into original (ARMv6) Pi and the newer Raspberry Pi 2 (ARMv7). Grab the image file from this page for the Pi 2.

03: Install to card

Open up the terminal and use fdisk -l to determine where your SD card is located on your system. Something like /dev/sdb or /dev/mmcblk0 will be ideal. Navigate to the image using cd and install it with dd using:

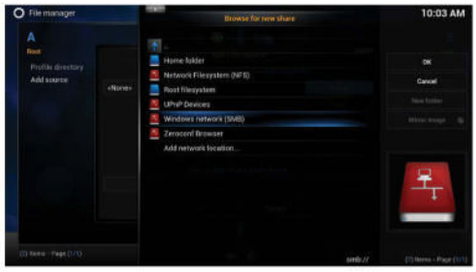
```
$ dd bs=1M if=OpenELEC-RPi2.arm-5.0.5.img
 of=/dev/mmcblk0
```

04: First boot

Plug in your Raspberry Pi, either to your TV or to another screen just to begin with, and turn it on. OpenELEC will resize the SD card partitions and write a few extra programs before finally booting into Kodi.

05: Configure Kodi

Go through the basic wizard to get through the interface – if you are connecting via wireless you will need to go to OpenELEC in the System menu and activate the wireless receiver before selecting your network and then entering your password.



06: Add network shares

You can stick a portable hard drive or USB stick into the Pi for storage, but the best method is really to stream over the network. Go to File manager under System and select Add source. Go to Browse and choose your network protocol to browse the network. Alternatively, you can try adding it manually.

07: Build your media centre

Placement of your Raspberry Pi is important. As it's going to be out all the time, we highly recommend getting a case for it – the Pibow cases from Pimoroni are quite well suited for this type of use as they are sturdy and can be attached to the rear of some TVs.

08: IR sensors and controllers

Kodi can be controlled with a number of different things – including USB game controllers and compatible IR sensors. We've used FLIRC in the past, but if you have your Pi behind the TV, you'll need a sensor on a wire that can stretch to a useful position.

09: Future updates

OpenELEC has the excellent ability to update itself without needing you to reinstall it every few months, meaning you won't need to do much maintenance on it at all. Now you can sit back and enjoy your media much easier than before.

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

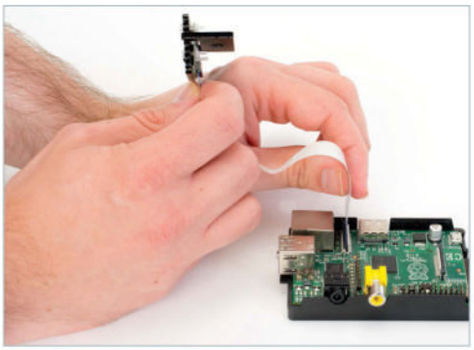
Raspberry Pi Raspbian:
www.raspberrypi.org/downloads

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
```

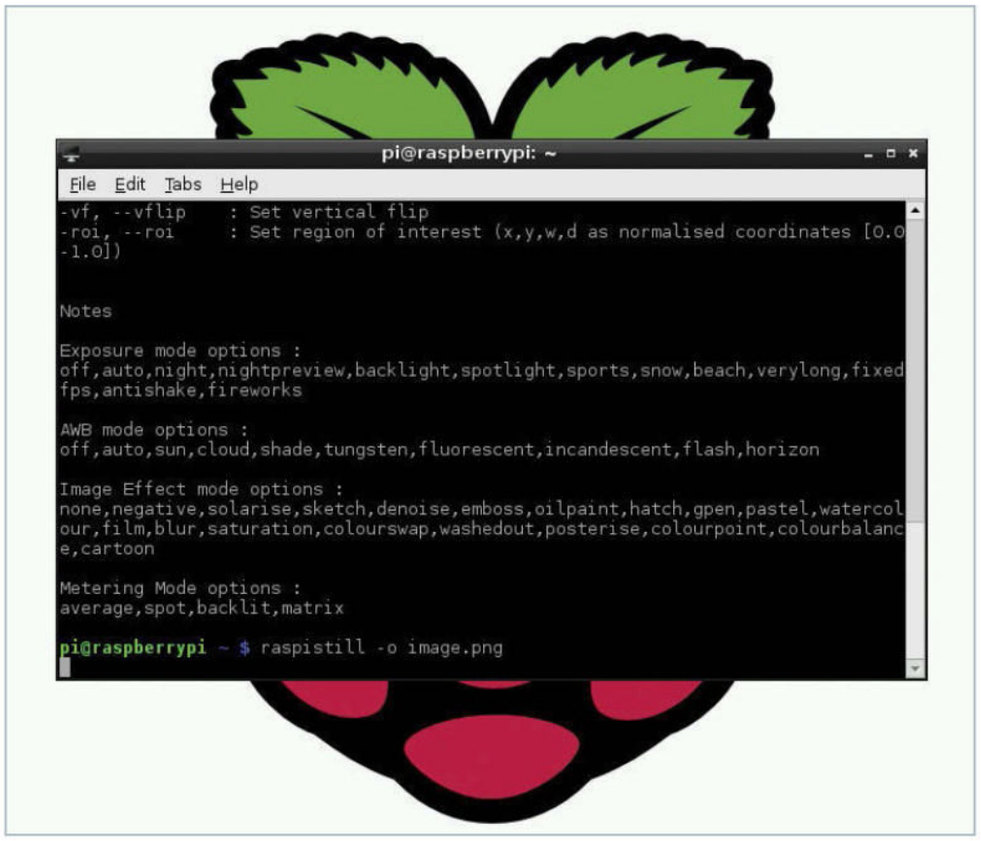
…to start the standard configuration screen. Navigate down to Enable Camera, press Enter and then simply key over to enable and confirm with another press of Enter. Select Finish and then reboot.

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.

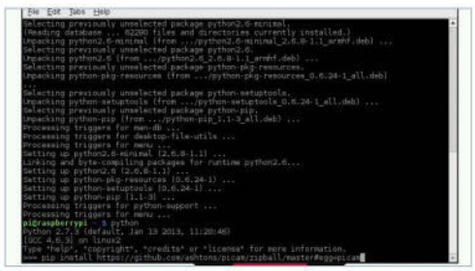


05: Record video

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.

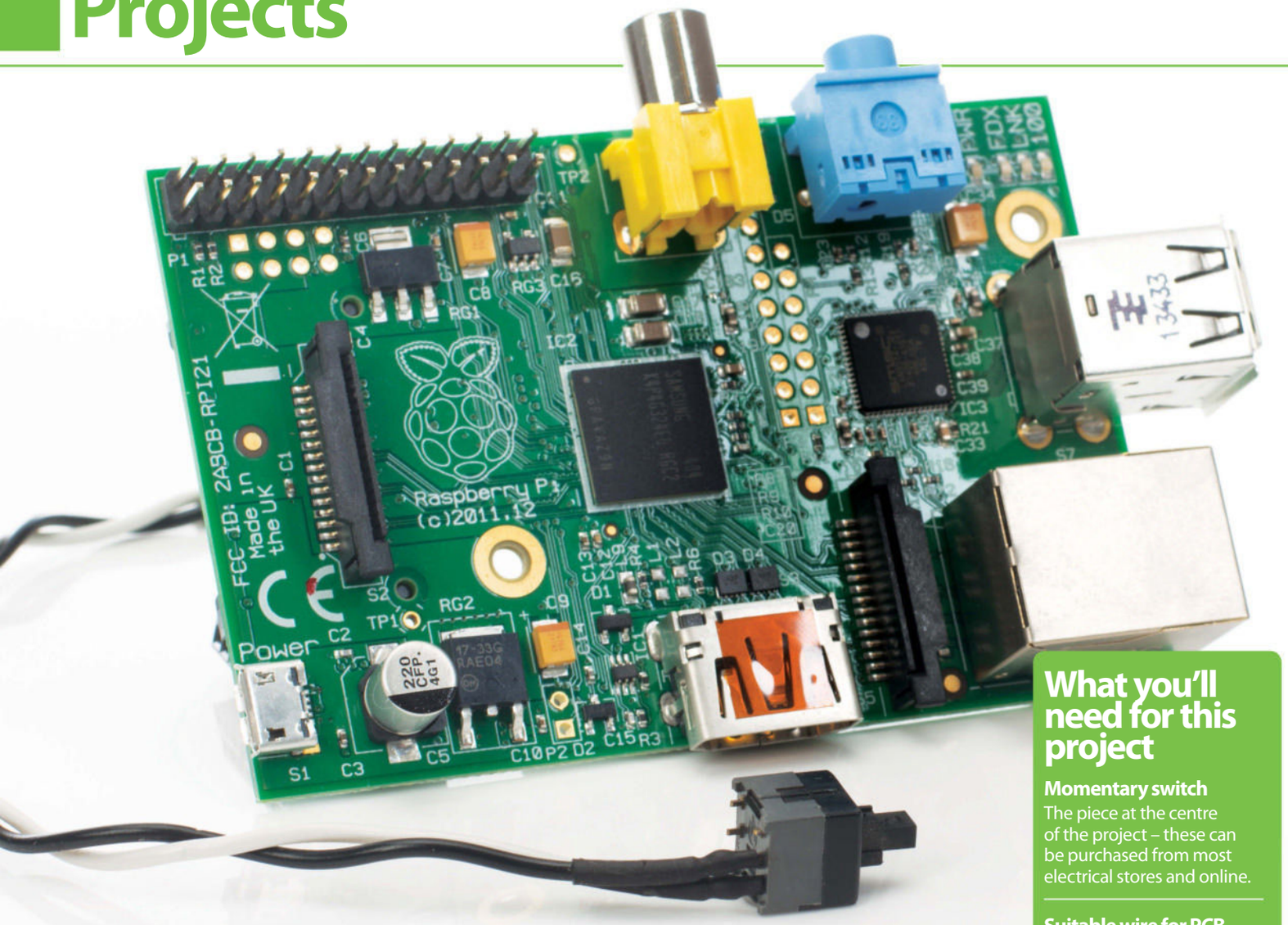
08: Advanced photos

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.



What you'll need for this project

Momentary switch
The piece at the centre of the project – these can be purchased from most electrical stores and online.

Suitable wire for PCB projects

Soldering iron and solder
This may seem a little daunting at first, but a soldering iron is an essential piece of kit for many hardware projects.

Single pin pair header

HDD/motherboard jumper

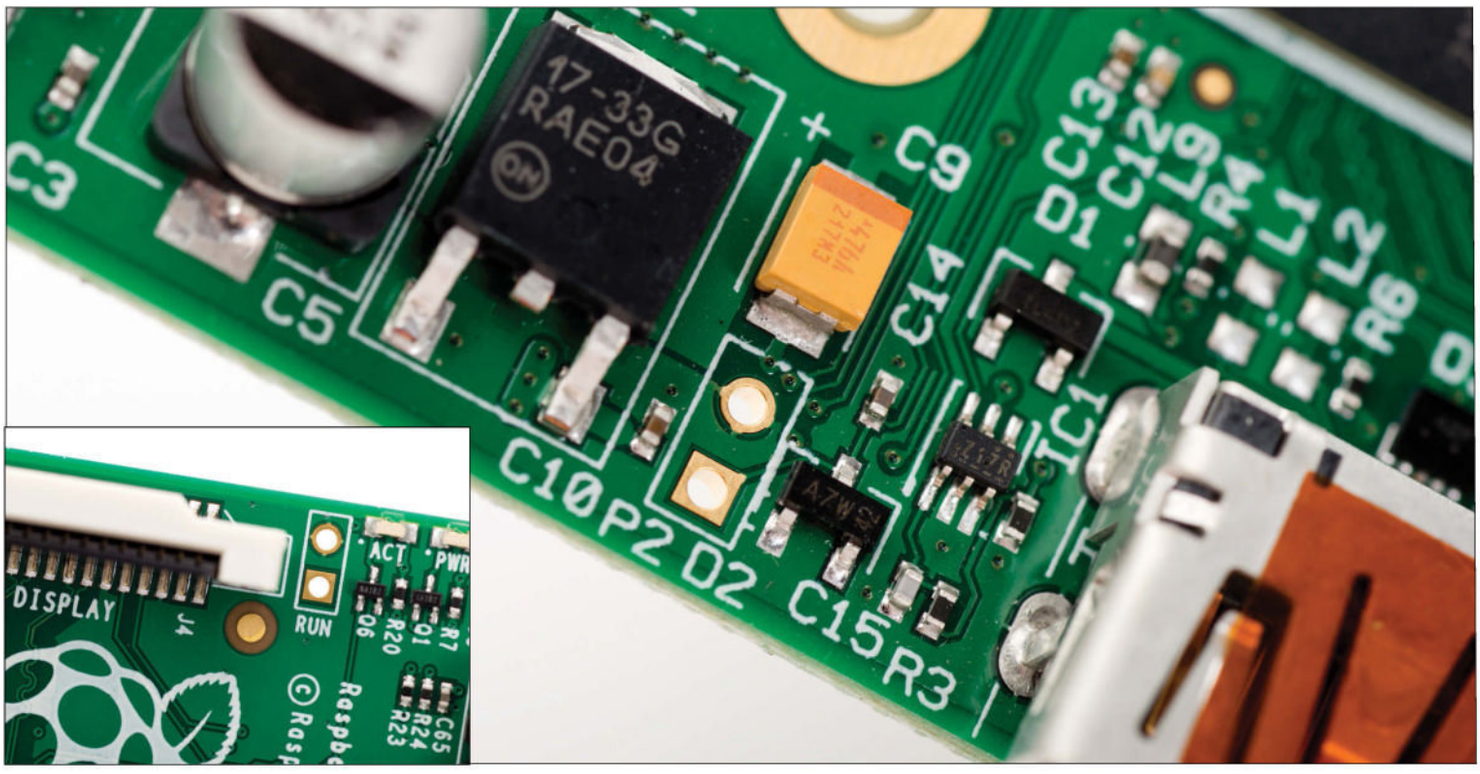
Add a reset switch to your Raspberry Pi

Need to restart your Pi after a system lock-up? Ease strain on the mains connector – install a reset switch!

We all know that shutting down a Raspberry Pi by removing the power cable is risky. Data may be writing to the SD card, leading to corruption, while repeated removal of the power cable can cause problems with the connector port.

Clearly this can cause some problems when faults cause the Raspberry Pi to hang, so the simple fix here is to add a simple reset function to the device. There are three ways this can be done: with a USB reset button, a motherboard jumper on the GPIO bus or with a momentary button connected to newly-soldered pins on the P6 header on the Model B Rev 2 and B+ (this is the most complicated option).

If you have an old PC lying around, retrieving the reset button and cable from this and even the connecting motherboard pins is achievable if you're handy with a soldering iron. Otherwise, we recommend purchasing the parts online, although be aware that you'll probably need to buy more pins than you'll use.



01: Check your Raspberry Pi model

Only two models feature the P6 header: the Raspberry Pi Model B Rev 2 (which you can find next to the HDMI port) and the B+ (to the left of the '© Raspberry Pi 2014' label). You will need to install the pins manually, however, as they are not preinstalled for this function.

02: Find your components

Header pins can be purchased online, although this will invariably result in having to order more than you need.

Alternatively, if you have an old motherboard, remove a pair of pins with a soldering iron. Similarly, you might buy a new reset button, or use one from an old PC.

03: Solder pins to your Pi

To gain stability when soldering, place the Pi upside down on a layer of packaging foam, with the header slotted into the holes.

Using fine solder, secure the pins to the mainboard with your soldering iron. This will require a very steady hand, so get assistance if you require it.

04: Connect your reset switch

Leave the solder to cool for a few minutes before attaching the reset switch connector.

Some cases don't have space for the pins and/or the connector, however, so take the time to plan ahead and make sure everything fits. If not, you may need to make some adjustments to your case.

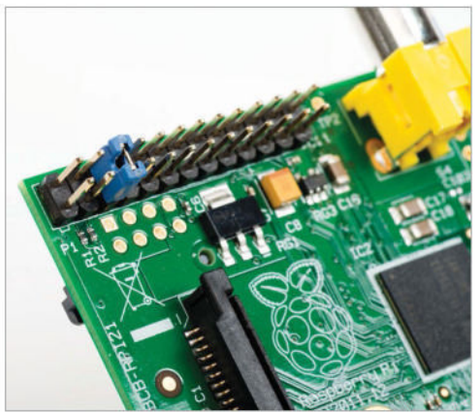
05: Reset Raspberry Pi following crashes

With the switch installed, you'll be able to reset the Raspberry Pi when required. Note, however, that this isn't an option to be used for whenever you feel like restarting. Rather, it should be done only when the system fails to respond within a reasonable time frame.

06: Reset with a HDD jumper

Not keen on soldering new pins to your Raspberry Pi? That is perfectly understandable, but it doesn't mean you cannot reset the computer. We have another solution for you.

Using a motherboard jumper, two GPIO pins and a script to initiate an ordered shutdown is a simple alternative that doesn't involve solder and potential PCB damage.



07: Identify the GPIO pins

This method works on most models. Each has a GPIO array, 26 pins on the A and B (Rev 2) and 40 on the A+ and B+. The jumper should be placed on GPIO3, pins 5 and 6 counting from the left with the board the right way around.

08: Detect jumper with a script

Use the script at **bit.ly/1Ge5n0O** to detect the jumper, making it executable (sudo chmod 755) before running. Within a minute your Pi will shut down. Add this line to /etc/crontab to run the script whenever you boot up.

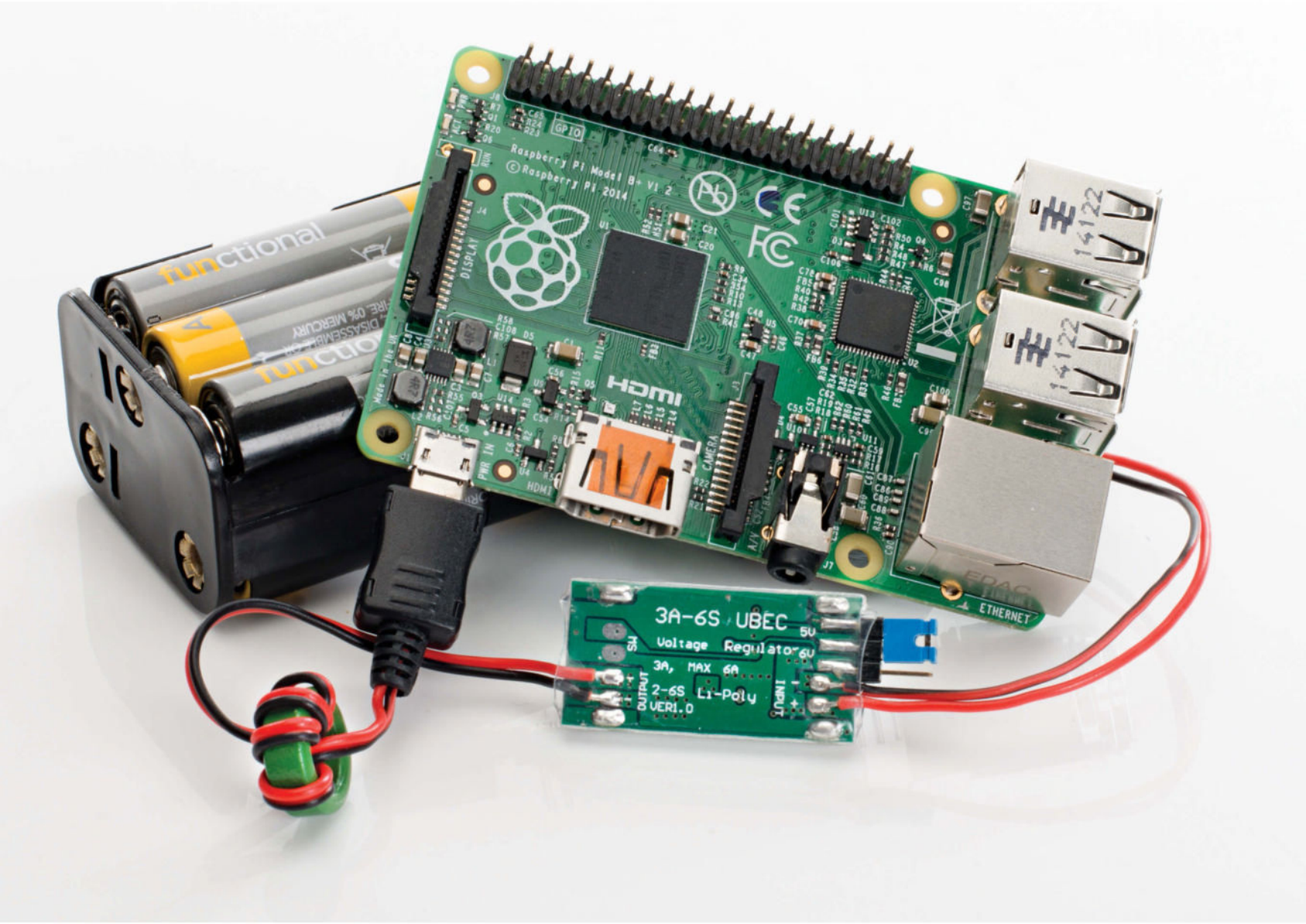
```
@reboot   root   /home/user/scripts/gpio_
actions.sh
```

Remember to remove the jumper before booting up!

09: Try a USB reset button

Specialist online stores offer USB reset buttons that can be connected to your Pi for scenarios when the device needs to be rebooted.

If the idea of using the HDD jumper or doing some minor soldering doesn't suit you, then a USB reset switch might be your best option.



Add a battery pack to your Raspberry Pi

Don't leave your Raspberry Pi behind – incorporate it into mobile projects by powering it with AA batteries

Resources

AA battery box
bit.ly/1FDiJGa

3-Amp UBEC
bit.ly/1HLKih7

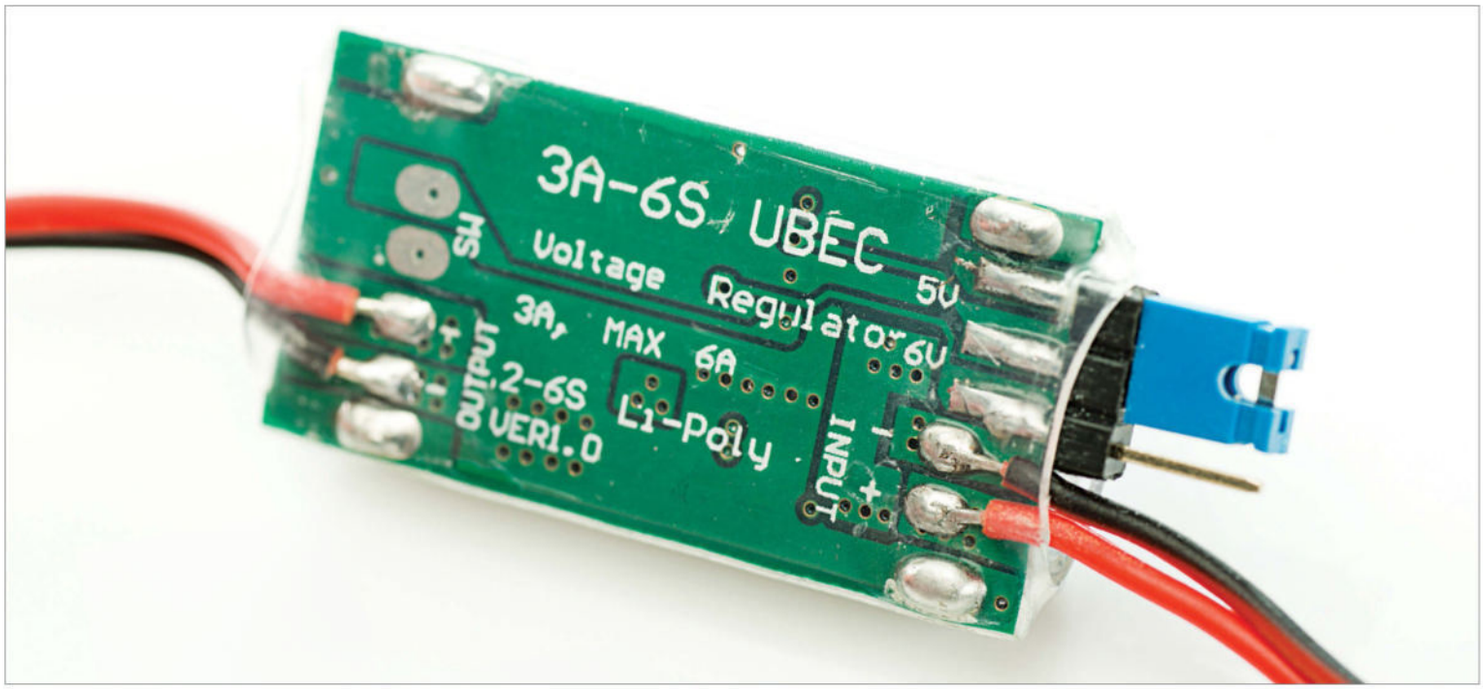
3-Amp terminal strip

6x AA rechargeable batteries

Your Raspberry Pi's mobility is usually restricted by the length of the power lead. Rather than limiting it to your desk or living room, however, you can use it for mobile projects as diverse as launching it into near-Earth orbit or monitoring and automating your garden.

Of course, in order to carry out such functions, you will need to find an alternative power source by adding batteries. Don't worry, adding battery power to your Raspberry Pi is simpler than you might have imagined; the place to start is by attaching a battery pack.

All that is required to complete this renovation are six rechargeable AA batteries (or single-charge alkaline), a battery box with space for the batteries and a UBEC. The latter is a Universal Battery Elimination Circuit, a voltage regulator that will regulate the power supply and prevent damage to the Raspberry Pi, which can be bought for under £10.



■ Above: You can also use a UBEC to charge your smartphone from a battery pack

01: Order your components

If you're buying your components online, you should be able to get them all within five days. However, if you're ordering offline (specifically the UBEC), you should avoid traditional electronics stores and instead visit a model enthusiast store, as these circuits are regularly used in RC devices.

02: Check your UBEC

Two types of UBEC are available. If you used the store that suggested in the box to the left, you'll receive one with a micro USB power connector for easy connection to your Raspberry Pi. However, if you bought one from eBay then there is a strong chance that you will receive one with a 3-pin connector.

03:: Change the UBEC connector pins

To use the UBEC with a 3-pin connector, alter the position of the pins so that they occupy the two outer slots.

Use a small jeweller's screwdriver to lever up the small plastic catch and remove the red wire from the central slot, before sliding into the unoccupied outer slot.

04: Connect the UBEC to the battery box

With five batteries in the battery box, connect it to the UBEC: red-to-red, black-to-black. You might do this by twisting the wires or soldering, or you can employ a 3-amp terminal strip, cut down to two pairs. The terminal strip can be cut to size using a modelling knife.

05: Add a battery to boot

With your Pi ready to use and your Wi-Fi dongle plugged in, connect the UBEC to the micro USB port and insert the sixth battery into the battery box. The Pi's power and status lights should indicate that the computer is booting up, which gives you a fully portable computer.

06: Connect the 3-pin UBEC

If you purchased the UBEC with the now-modified 3-pin connector, you'll need to connect this to the Raspberry Pi's GPIO header. Specifically, connect the positive +5V (red) connector to Pin 2 and the negative 0V connector to Pin 6. Once again, check the status lights to ensure the Pi is booting.

07: Measure uptime

You should have already set up your Pi for SSH use, so connect to the device via Putty after giving it time to boot fully (at least 60 seconds). In the terminal, enter:

```
watch -n 60 uptime
```

This command will display the system uptime and also keep the Wi-Fi connection active.

08: Judge your uptime results

Uptime results depend upon the type of battery you use and the Raspberry Pi model. Single-charge batteries will last a little bit longer, but this is a more expensive option. Meanwhile, newer models have greater power requirements but run for less time. For more power, add more batteries!

09: Power extreme!

If more batteries are added in parallel, this should result in almost double the uptime (at least 16 hours on a 256MB Raspberry Pi Model A), but instead of alkaline or rechargeable batteries you might want to consider a modern lithium-based AA cell, which will last considerably longer than alkaline batteries.

Protect your Pi with a UBEC

It is possible to power your Raspberry Pi directly from four or more batteries, but without the safety fallback that the UBEC provides, you're likely to burn out the computer. Unregulated power can cause considerable damage to your Raspberry Pi, and as you increase the current with more batteries, the risk also increases. Unless you are planning to learn the hard way and you have plenty of cash to burn, always use a UBEC!

Tether your Raspberry Pi to an Android device

Need the internet on your Pi on the go? Try out a physical tether to your Android device for instant online access

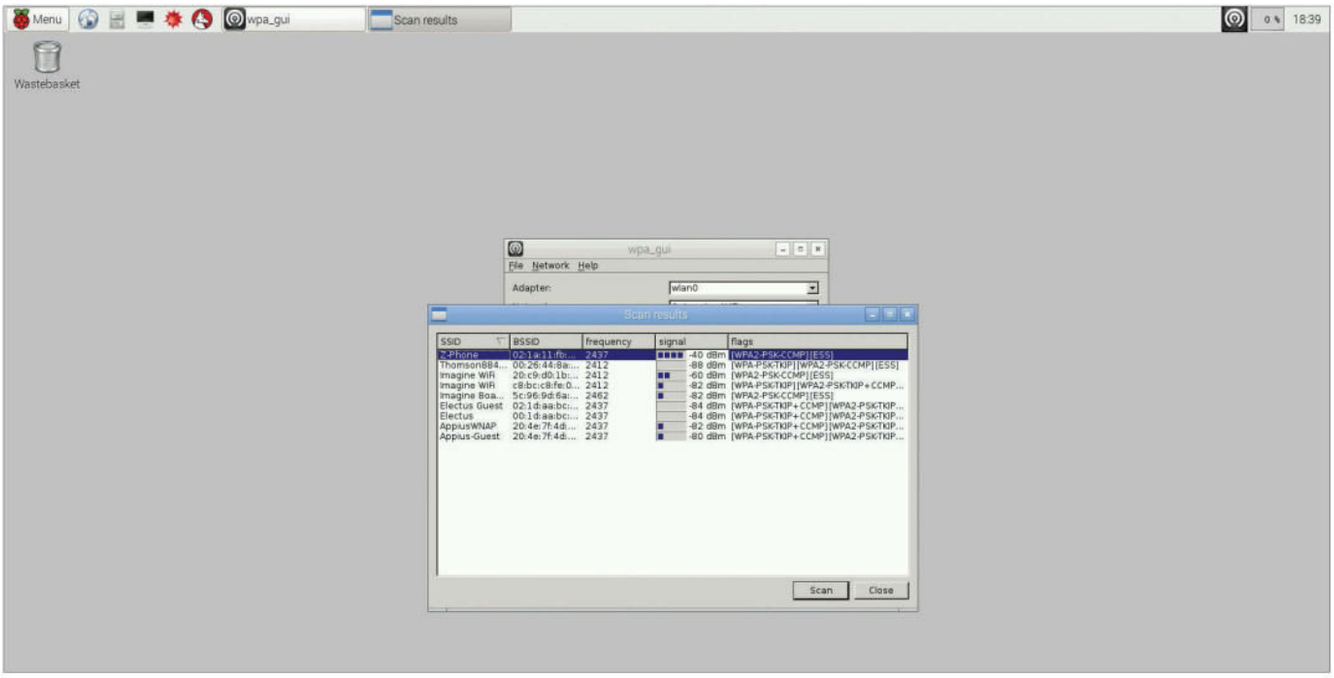
Resources

Android device

USB cable

The portability of the Raspberry Pi is one of its most lauded features and you can get many different accessories to help enhance this key selling point.

Mini screens, mini wireless keyboard and mouse combos, portable batteries and more can get you out and about, but an internet connection is a stumbling block that you can't easily fix with an accessory. What you do also usually have with you is an internet-connected magic pocket box called a smartphone that, with a bit of know-how, you can connect the Pi to and steal some Internet from. Over the next two pages we will impart this know-how to get your Raspberry Pi on the internet when you're on the go.



■ Above: It's usually easy to figure out which device is your phone – set it right next to the Pi and it will be the one with the best signal!

01: The easy way

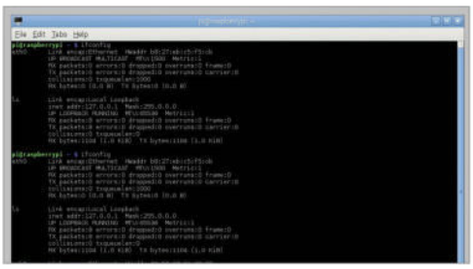
A lot of modern smartphones now have a Wi-Fi hotspot feature, which the Raspberry Pi can easily attach to. First of all, activate the hotspot on your phone, then boot into the Pi. Connect a wireless dongle and open up the wpa_gui in Preferences>Wi-Fi Configuration.

02: Scan for device

Click Scan to open up the scan window and then select Scan again from inside there. It should pick up your device – connect it as you would to any Wi-Fi network and the Pi will remember it for when it needs it next.

03: Set up tether

First connect your phone to your Raspberry Pi via a USB cable – depending on the amount of power your Pi has, it might have trouble charging your phone but it will still allow you to tether. In the tethering menu you can now activate the USB tethering option.



04: Check connection

Your Android device will create an interface known as eth0 on the Raspberry Pi. You can check to make sure this is happening, and that it will let you tether, by opening up a terminal and typing the following:

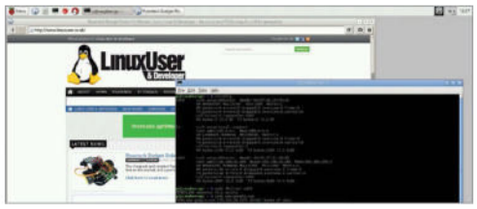
```
$ ifconfig
```

05: Quick connect

You can connect from the terminal right now to access the internet. You should be able to do this by typing the following into the terminal:

```
$ sudo dhclient usb0
```

This will automatically grab any available IP address that your phone will give to it.



06: Test connection

There are a few ways to test your connection. We'd usually stay in the terminal and ping **www.google.com**, which you can do, or you can click on the browser and see if it loads the page.

07: Save the settings

Once you reboot your Pi, it won't remember to automatically connect to the phone's tether. However, we can add an entry to its config so that it will try and do this in the future. From the terminal use:

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

08: Interface settings

Here you'll find all the current network settings – yours might look different from ours depending on whether you have added any fixed wireless settings or passthroughs. Using the same syntax as the eth0 line, add:

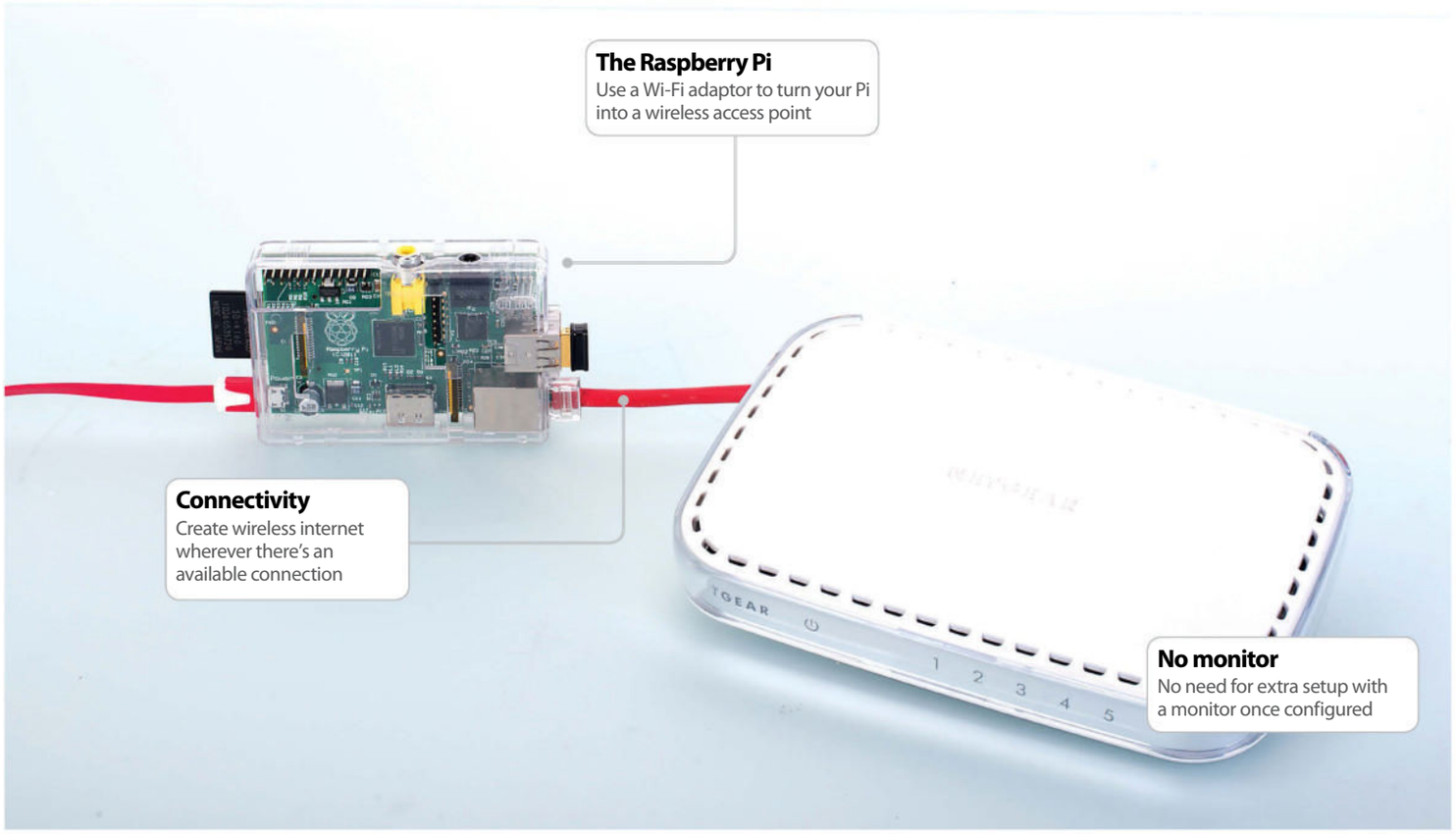
```
iface usb0 inet dhcp
```

09: Tether on the go

After saving and rebooting, your Pi should now automatically connect to your phone, whether it's via Wi-Fi hotspot or a physical connection. Going through this process may draw a little more charge than usual while tethering, so be sure to keep an eye on your battery level to avoid suffering an outage.

Keep an eye on your mobile data usage

Using your Pi on your mobile phone will eat up data much faster than browsing on your phone normally is. We suggest not doing a full software, distribution or firmware update if you don't want to spend a fortune on data. You can also set limits on the amount of data used on your phone to save yourself any problems, and a physical tether will allow you to connect via the phone's Wi-Fi if that's an option.



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

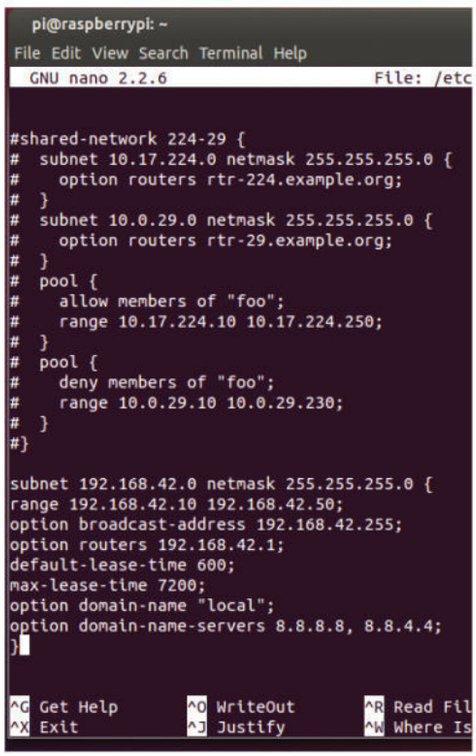
Install a DHCP server to your Pi with:

```
$ 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

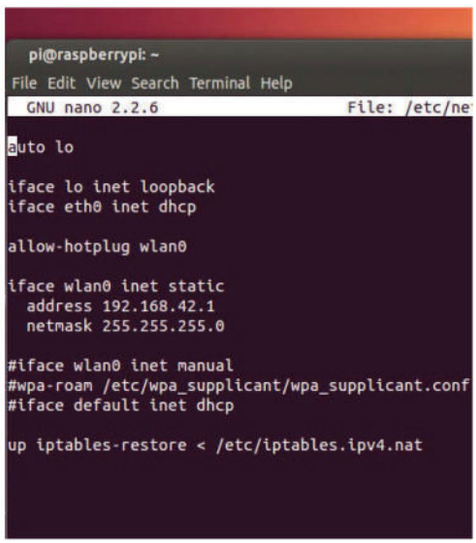
After the line 'allow-hotplug wlan0', you need to enter the following:

```
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

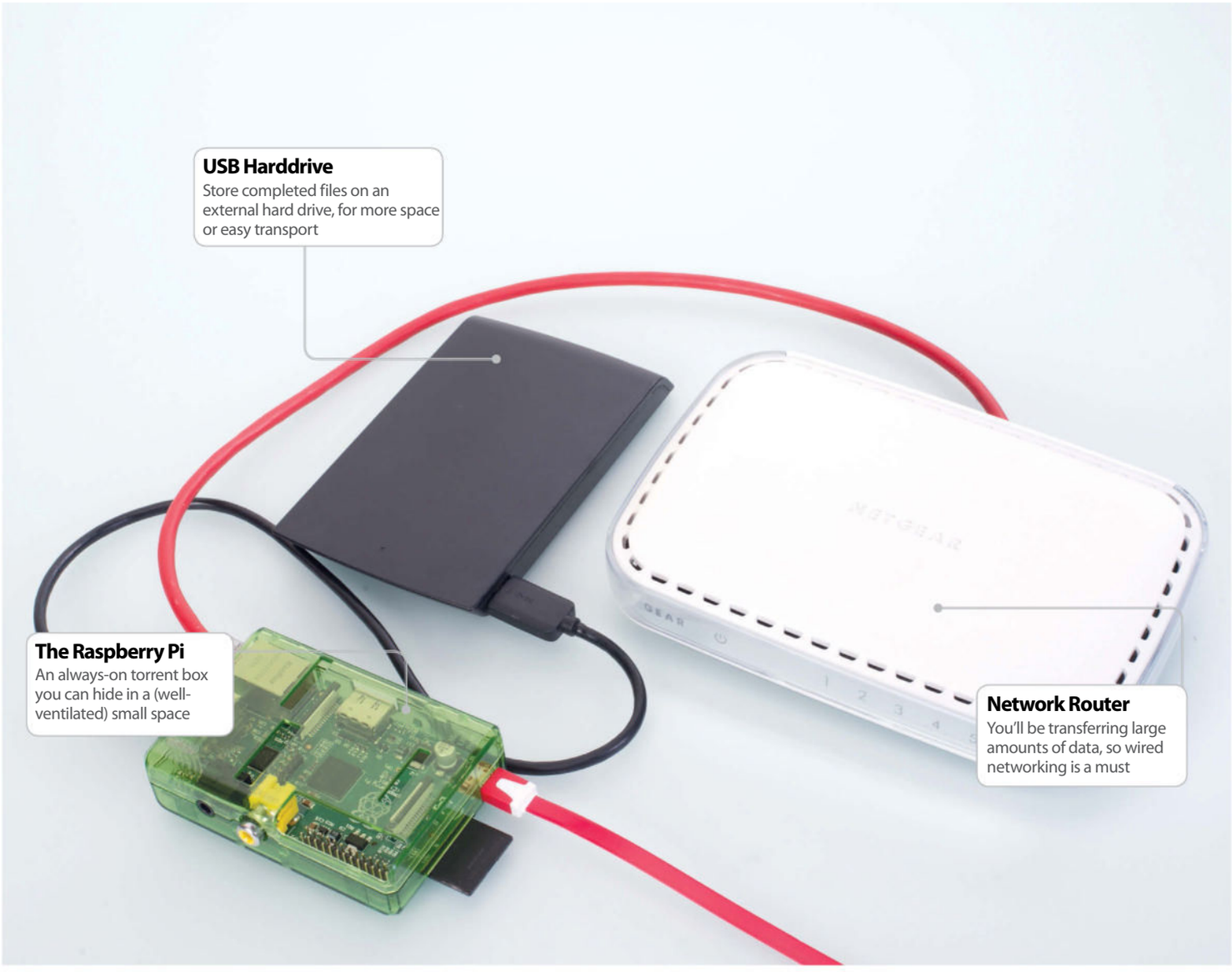
So that this works after a reboot, type:

```
$ 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 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.

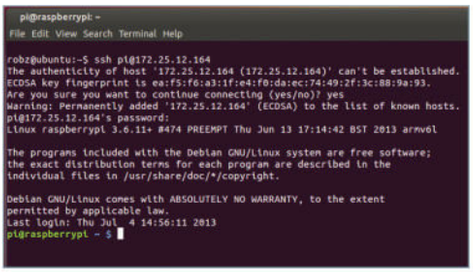
It can't do it on its own though – after all, the average Raspberry Pi SD card will only be between 2GB and 8GB in size. You'll quickly run out of space, which is where an external hard drive comes in.

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

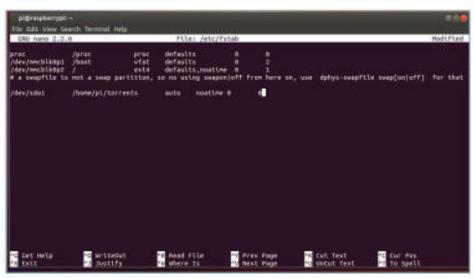
Unless you plan to reformat your portable drive, you'll need to install NTFS support onto your Pi. Type:

```
$ 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:

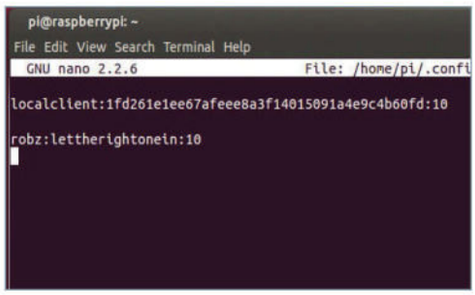
```
$ nano ~/.config/deluge/auth
```

And add to the bottom:

```
[user]:[password]:10
```

…to restrict access.
Now start it up with:

```
$ deluged$ deluge-console
```



06: Remote connection

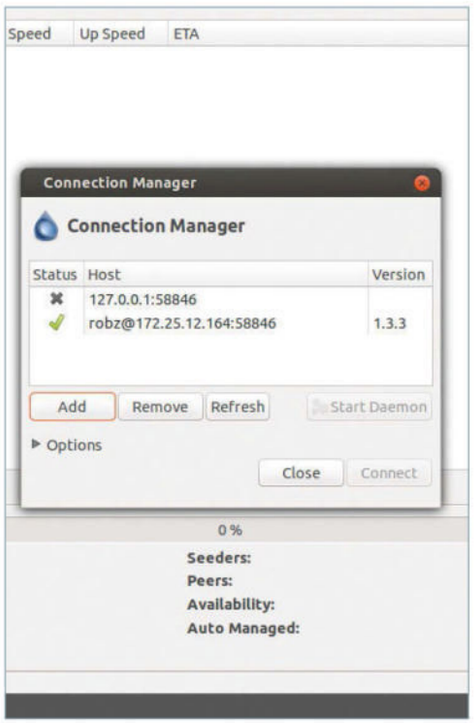
Now you're in the client, type the following three commands:

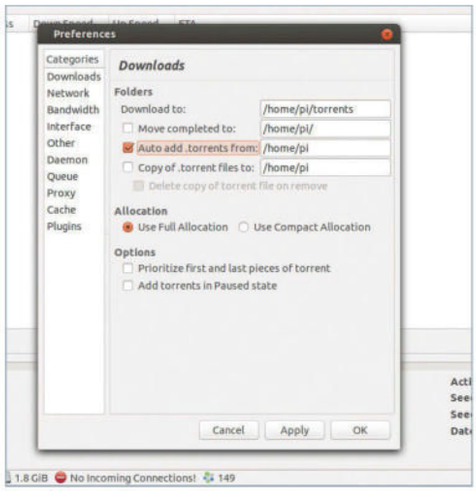
```
config -s allow_remote True
config allow_remote
exit
```

Restart the Deluge daemon with:

```
$ sudo pkill deluged && deluged
```

Now open the graphical client on your Linux PC.



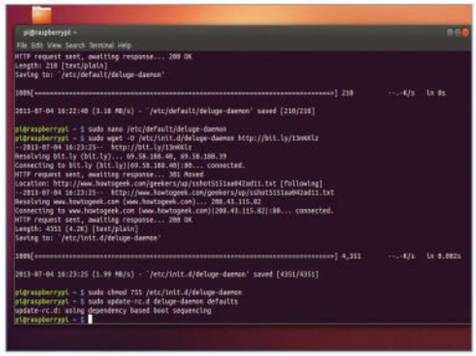


07: Remote interface

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.



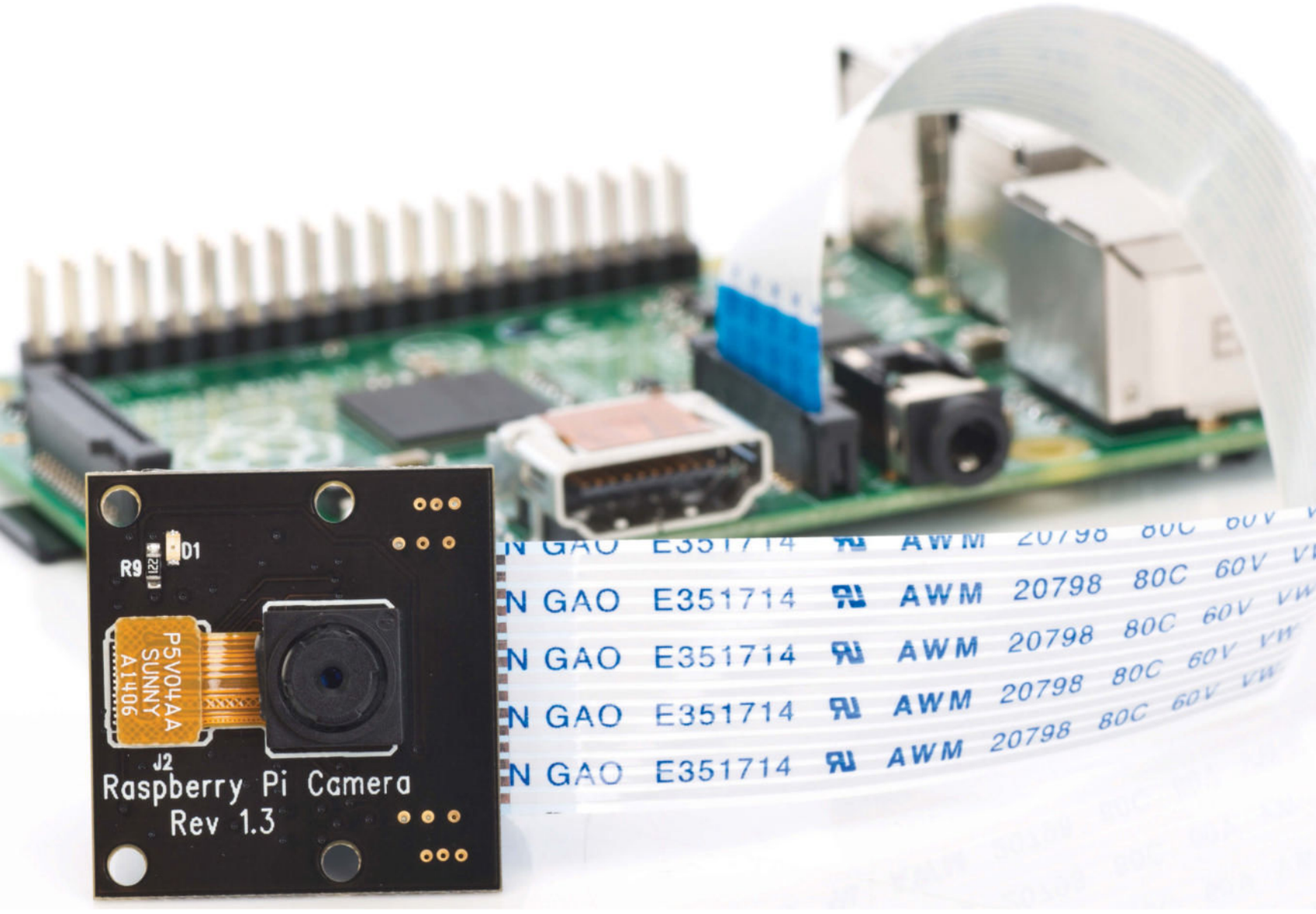
09: Start on boot

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
```



Capture photos at night with the NoIR Pi camera

Set up and deploy a mobile solution to take night-time photos of your garden's hidden wildlife

Resources

NoIR camera module

LISIPAROI LED light ring
bit.ly/1meQGaR

PIR Sensor

Portable battery

You are probably already familiar with the Raspberry Pi camera module. However, it is also available in the NoIR edition – 'NoIR being shorthand for 'no infrared'. Using this module, you can use the camera to take photos and video footage in the dark, similar to a night vision camera. Many of us are fortunate enough to have a wide variety of animals that visits our gardens and homes, and while we obviously see them more during the day, there are also a large number of nocturnal visitors to your garden. This tutorial shows you how to set up your camera, then add and combine an infrared light source and PIR motion sensor to trigger the camera and photograph night-time wildlife. Each photo that is captured is saved to the Pi and a 'time stamp' is also added to monitor and track the time that the animal visited your garden.

01: Attach the NoIR camera module

To get started, add your camera to the Raspberry Pi. It is vital that you ensure that you remove all static electric charge you may have built up by touching, say, a radiator first, as the camera can be damaged or even destroyed by static. The blue-coloured label points away from the HDMI port. Once in place, start up the Raspberry Pi as normal. Access the configuration settings using sudo raspi-config and enable the camera as shown below, then reboot.

It is always advisable to update the software, so type this into the comman d line:

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

■ The NoIR camera module picks up just enough light to enable you to take some fascinating outdoor photography at night

02: Take a picture

Once the Pi is updated, test that the camera is working correctly. In the LXTerminal, enter the following: raspistill -v -o test.jpeg. When run, you will see a brief preview on the screen and a picture will be taken and saved with the file name 'test'. This file is saved in the /home/pi folder. The parameter –o is for output and this is the name of the file you save the image as. For example, raspistill –o keyboard.jpeg saves the image as a file called keyboard.

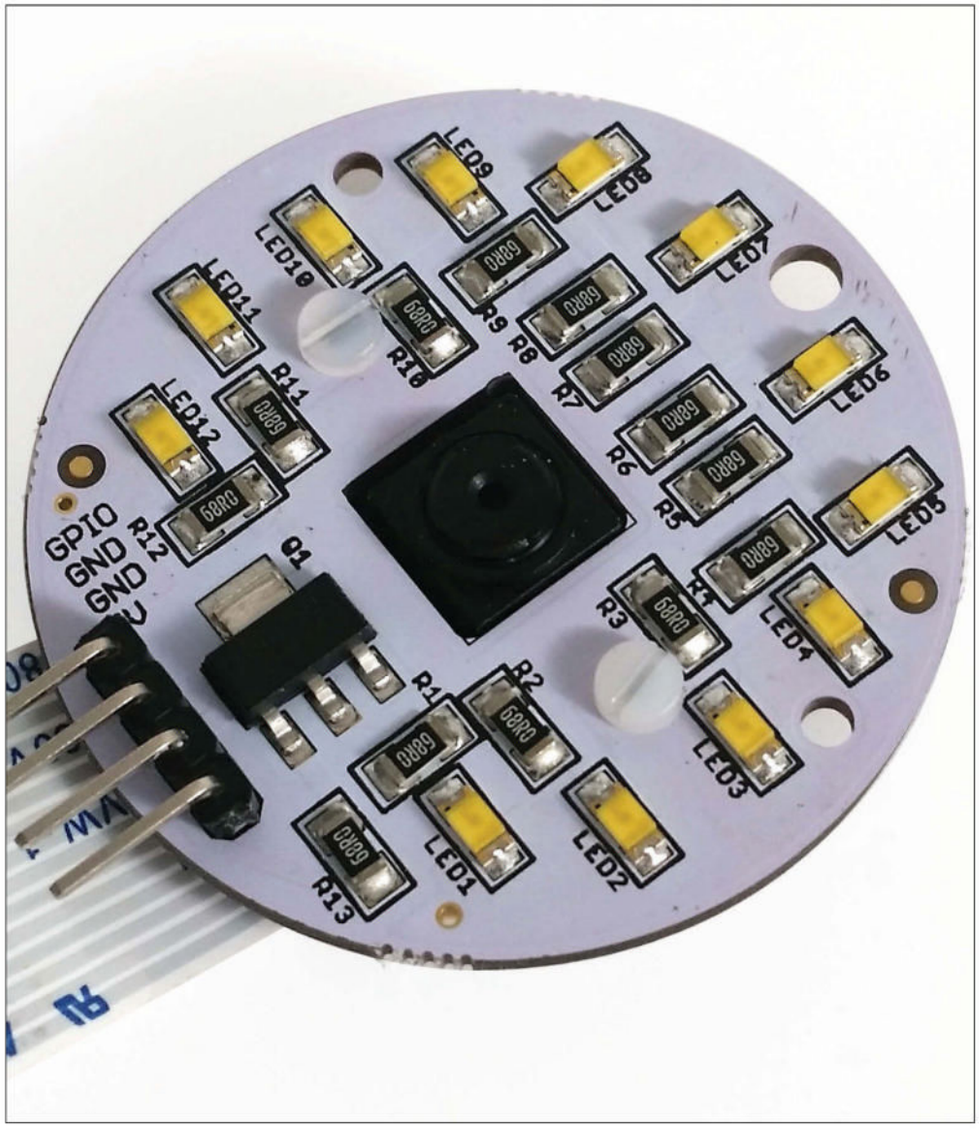
03: Using Python to take pictures

In this project, you will use Python to control the NoIR module and capture the pictures. Taking a picture with Python can be easily achieved with just a few lines of code. Open your Python editor and create the function below, then save and run. The image can be previewed using the code camera.start_preview() (line 6). To take a picture, use camera.capture('nature.jpg') (line 8). Replace 'nature' with the name that you wish to call the image file. This will test that the camera is working correctly with Python.

```
import time
import picamera
def Nature_selfie()
  with picamera.PiCamera() as camera:
      camera.start_preview()
      time.sleep(2)
      camera.capture('nature.jpg')
Nature_selfie()
```

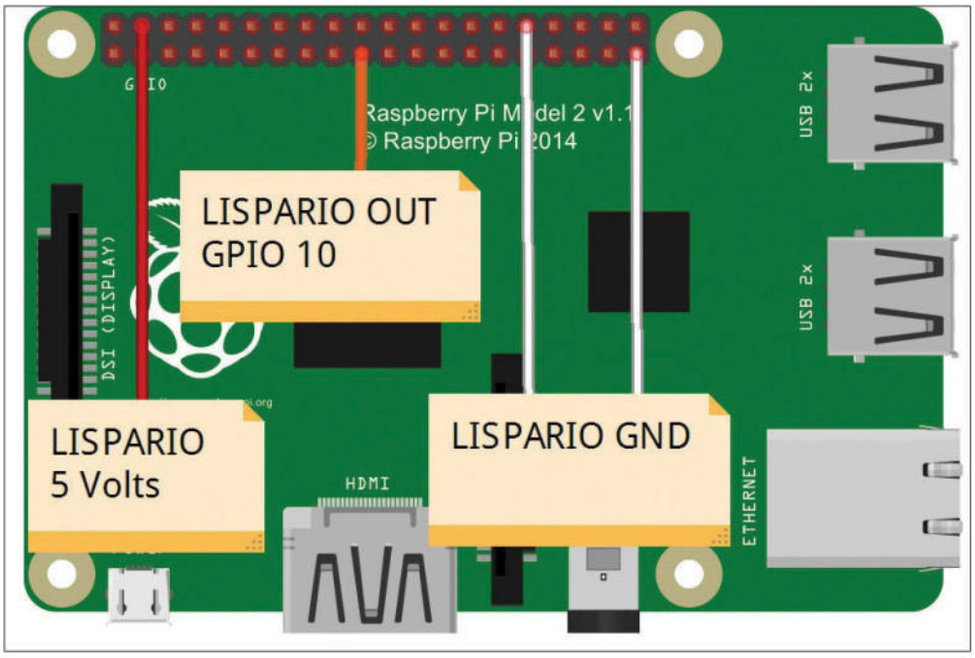
Cron

Cron, which many say stands for Commands Run Over Night, is used as a time-based job scheduler which permits you to schedule jobs (commands or shell scripts) to run periodically at certain times or dates. It is commonly used to automate system maintenance programs used for administration or disk-related tasks.



04: But does it work?

When you take your first photo in the dark, it may appear that the NoIR module does not work. The picture is not a night vision spy-style photo; in fact, it is probably completely black. What you need to change this is an infrared light source. The LISIPAROI is an add-on for the camera module which is designed to provide additional illumination when taking pictures or recording video in the dark. It features extra mounting points, which are perfect for using custom mounts or a gooseneck holder. There are two versions: a standard for the original camera and the NoIR version. The infrared LISIPAROI (noted with clear LEDs) has 12 infrared LEDs arranged around the camera module to offer a wide spread of light when used in low/zero light conditions, making it perfect for capturing night wildlife activity.



05: Connecting up the LISPARIO

Connecting the LISIPAROI is easy: simply take four female-to-female leads and attach them to the pins on the LISIPAROI. Connecting to the Raspberry Pi is easy, too: the pins required are 5V, GND, GND and the GPIO 10 pin. In this project, the code uses the BCM pin numbering system – the physical pin number is provided here for ease. The 5V is attached to physical pin 4, the ground wires go to pins 32 and 39, although there are several GND pins you can use. The final wire is the GPIO 10 pin, which is physical pin number 19 on the board. Now you have your LISIPAROI connected and you are ready to take a picture.

06: Another test

Now it's time to adapt your previous camera code to turn on the LISPARIO and capture a picture in the dark. First, import the GPIO module (line 1, below): import RPi.GPIO as GPIO. Set the GPIO numbering system to BCM with the code GPIO. setmode(GPIO.BCM) on line 4. The LISAPRIO runs on GPIO pin 10, so set this on line 5: GPIO. setup(10, GPIO.OUT). Before the picture is taken, set the output to HIGH on line 6 to turn the LEDs on: GPIO.setup(10, GPIO.HIGH). The picture is then taken and saved. On the last line, the LEDs are turned off. Check your image file – you should now have a night vision-style photo.

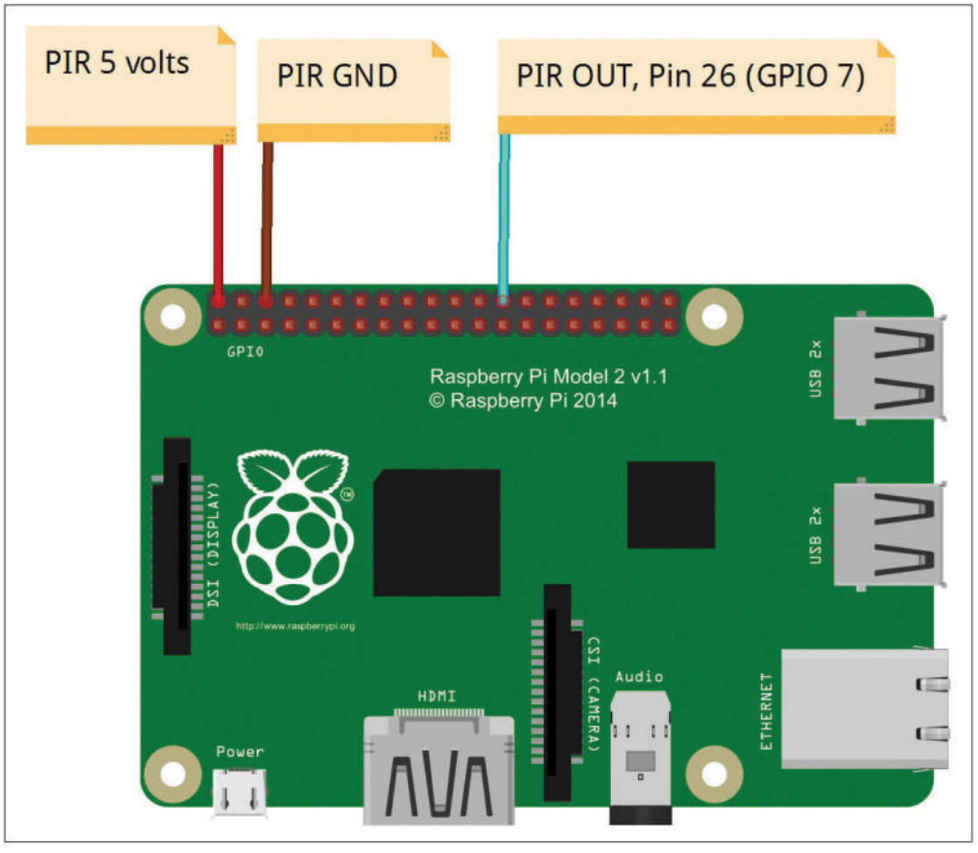
```
import RPi.GPIO as GPIO
import time
import picamera
GPIO.setmode(GPIO.BCM)
GPIO.setup(10, GPIO.OUT)
GPIO.output(10, GPIO.HIGH)
with picamera.PiCamera() as camera:
  camera.start_preview()
    time.sleep(2)
    camera.capture('nature.jpg')
  camera.stop_preview()
GPIO.output(10, GPIO.LOW)
```

07: PIR sensor

A PIR sensor, 'passive infrared sensor', picks up the heat energy that is given off by objects. This radiation is invisible to the human eye because it radiates at infrared wavelengths. However, it can be detected by electronic devices such as the PIR. The sensor can be used to detect when a change in heat has occurred. The PIR has two dials that can be changed to adjust the settings of the PIR. The first is to adjust the 'heat' sensitivity, which will make the PIR trigger with more or less of a heat change. The second dial adjusts the rest time between the sensor stopping and restarting. This is originally set to a delay of around a few seconds.

08: Connecting the PIR sensor

Time to connect the PIR. To check this is working correctly, first remove the LISIPAROI wires. The PIR has three wires: the 5V, a ground and the out wire. Remember you are using the BCM pin numbering system, so the number stated in the code will be the GPIO pin number on the Raspberry Pi. The +5V wire connects to physical pin 2, the out connects to pin 26(GPIO 7) and the Ground connects to physical pin 6.



"Taking a picture with Python can be easily achieved with just a few lines of code"

09: Testing the PIR

Now that the PIR sensor is connected, test that it is working correctly and adjust the settings to ensure that it triggers correctly. Open your Python editor, open a new file and enter the test code below. The important line of code is :

```
GPIO.add_event_detect(PIR, GPIO.RISING,
callback=Motion_Sensing)
```

It is set to detect the GPIO rising as the heat from, say, a badger is detected by the PIR and it triggers the GPIO pin 7. The Raspberry Pi identifies that the voltage is rising and executes the callback. In this example, the callback is a function called Motion_Sensing, which when run will display the phrase "We see you" in the Python console window. When the badger moves, there is another change in heat and the PIR senses this.

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
PIR = 7
GPIO.setup(PIR, GPIO.IN)
print "Ready to find you"
time.sleep(2)
def Motion_Sensing(PIR):
  print "We see you"
try:
  GPIO.add_event_detect(PIR, GPIO.RISING,
callback=Motion_Sensing)
  while 1:
    time.sleep(100)
except KeyboardInterrupt:
  print "Quit"
  GPIO.cleanup()
```

BCM number

GPIO pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off. You can also program your Raspberry Pi to turn them on or off. The GPIO.BCM option means that you are referring to the pins by the "Broadcom SOC channel" number. If you start counting the pins, this is the physical pin number. The GPIO.BOARD option specifies that you are referring to the pins by the plug number, ie the numbers printed on the board.

10: Saving as a new picture

Currently, when the Pi takes a picture using the code in Step 3 it overwrites the existing file because both files have the same filename. This is not suitable for taking multiple pictures over a period of time.

To ensure the files are not over-written, create a variable called File_Number. Each time the camera is triggered this variable is incremented by a value of one. So in the example below, the first file is called Nature1.jpg, then the next files are Nature2.jpg, Nature3.jpg and so on. This saves each new photo with a different file name and won't overwrite existing images.

```
camera.capture("Nature" + str(File_Number) +
".jpg")
      File_Number = File_Number + 1
```

11: Add the time the picture was taken

You may be keen to know the time that the camera was triggered and the pictures taken, especially if the setup has been running over night. To do this, create a new variable called time_of_photo, at the start of the program, to store the current time. To retrieve the time that the photo was taken use time.asctime(time.localtime(time.time())). This will check the local time of your Raspberry Pi and save it to the variable time_of_photo. Ensure that your Pi's clock is set correctly before you start your program.

```
time_of_photo = time.asctime( time.
localtime(time.time()) )
```

12: Add the time to the picture

Recording the time that the picture is taken is only relevant if you are there watching and waiting for wildlife to trigger the camera, and you can see the time value. A better solution is to use the time to add a 'time stamp' to the picture. This means that when you view each image you can see the time that the pictures were taken at the top. The line of code required is: camera.annotate_text = "time_of_photo".

13: Recap

Before you finalise the project, a quick recap on the features and setup: the NoIR camera module is attached to the Pi to capture photos in the dark. This requires an infrared light source, which is provided by the LISIPAROI. A PIR is attached and used to sense changes in heat, which then turns the LEDs on and triggers the camera to capture a picture. The current time is added to the picture for future reference. The diagram on the facing page shows the wiring solution for the LISIPAROI and PIR.

14: Taking a video

You may prefer to take a video when the PIR is triggered; you could adapt the program to record videos of wildlife that may visit your garden or the location where your camera is. Again, Python makes it simple to record video using the code camera.start_recording('/home/pi/Desktop/evidence.jpg') to start the recording, which you can save as a file called evidence. If you want to video for, say, 20 seconds then use time.sleep(20) before stopping the recording with camera.stop_recording('/home/pi/Desktop/evidence.jpg').

```
camera.start_preview()
  camera.start_recording('/home/pi/Desktop/
evidence.h264')
  time.sleep(20)
  camera.stop_recording( )
```

15: Automatically start the program

If you deploy the project outside then you will probably not have a monitor or screen attached, meaning that you cannot see what the program is doing. To sort this, set the program to start automatically when the external power supply is plugged in. First, write down where you have saved the Night Box program – for example, in the /home/pi folder, called Night_Box.py, in which case you'd use /home/pi/Night_Box.py. Double-check you've got the correct path by opening the LXTerminal and typing:

```
  sudo cat /home/pi/name_of_your_script.py
```

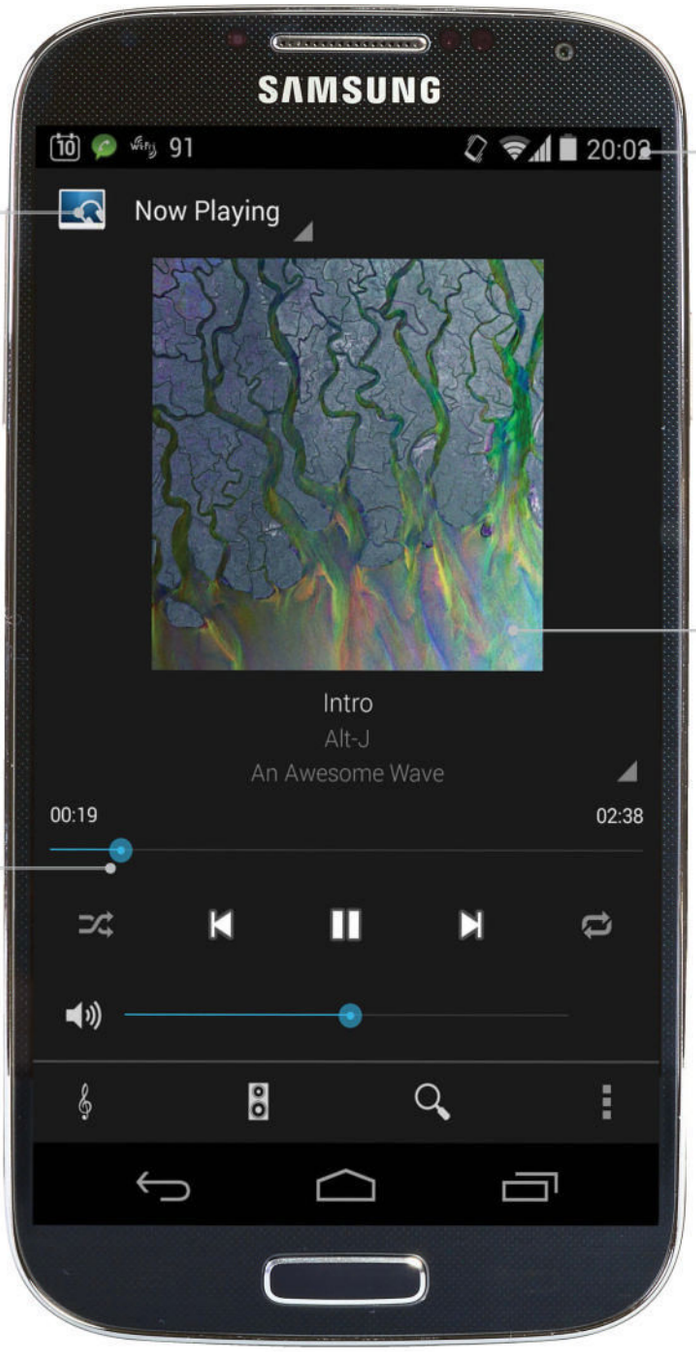
If correct, this will display the contents of your Python code. The next step is to create a Cron job by modifying the 'crontab'. To edit it, type sudo crontab –e (this will run the Cron task for all users) Scroll to the bottom of the window and then add the following line to the file:

```
  @reboot python /home/pi/name_of_your_program.
py &
```

The "&" at the end will run the code in the background and ensure your Raspberry Pi will boot up as normal. Save the file by hitting Ctrl+X followed by Y, and then reboot.

16: Deploy the Night Box

To deploy the setup effectively you will want to create a box or holder for the PIR, LISIPAROI, etc. This will protect them from the weather, especially if it rains or snows. You may want to consider a 3D-printed solution or even an old match box to hold the components. In the previous step you scheduled a Cron task to start the program automatically when the Raspberry Pi is first booted up. Find a suitable location and set up your box. Plug in your portable power supply and the Pi will boot up and start the program. Then check to see what visited your garden.



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**

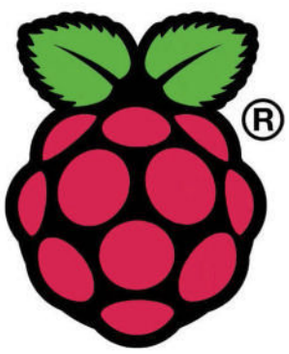
04: Configure the daemon

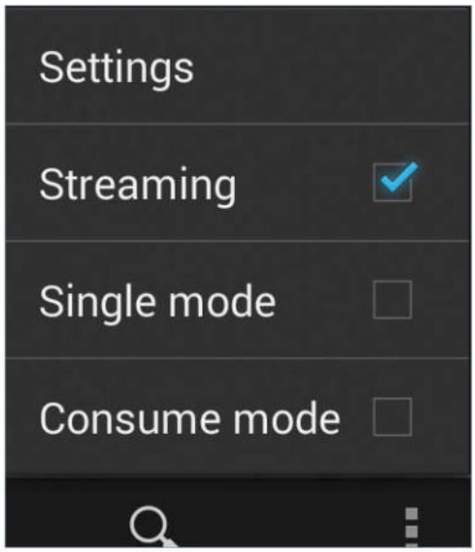
We want to edit /etc/mpd.conf (using sudo). The first change is to make the daemon listen on all interfaces, so we can use MPD clients from other devices. Do this by changing the line:

```
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.

■ 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.

"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"

Hack your TV with Pi

Use the Energenie IR board and a Raspberry Pi as a remote control for your television

In this project you will learn how to emulate your television remote using your Raspberry Pi and a Energenie IR board in order to control your big screen.

Why? So you can change the channel while no one is looking! Infrared, or IR for short, is light with a wavelength greater than the red end of the visible light spectrum, but less than that of microwaves. Infrared radiation can't be seen with the naked eye, but it can be felt as heat energy. Infrared radiation is used to transmit data from device to device, including between remote controls and their televisions, Blu-Ray players or to provide data links over short distances between computers or mobile phones.

This tutorial will show you how to set up the Energenie's Pi-Mote IR board, which will enable your Raspberry Pi to learn infrared remote-control signals and then transmit these same commands in order to control your device. This tutorial will focus on controlling a television in particular, but remember that the board is compatible with a wide range of devices. This means you can turn your Raspberry Pi into a remote for a range of household objects, such as your media system, smartphone dock or even the air conditioning.

01: Edit the config.txt file

Attach your IR board module to your Raspberry Pi and boot it up. It is always advisable to update your SD card software. In the LXTerminal type:

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

Next add code to the /boot/config.txt file to enable the LIRC IR software and IR module to interact. In the LXTerminal type:

```
sudo nano /boot/config.txt
```

This will load the config .txt file. Scroll to the bottom of the text and add the following line:

```
dtoverlay=lirc-rpi-overlay
```

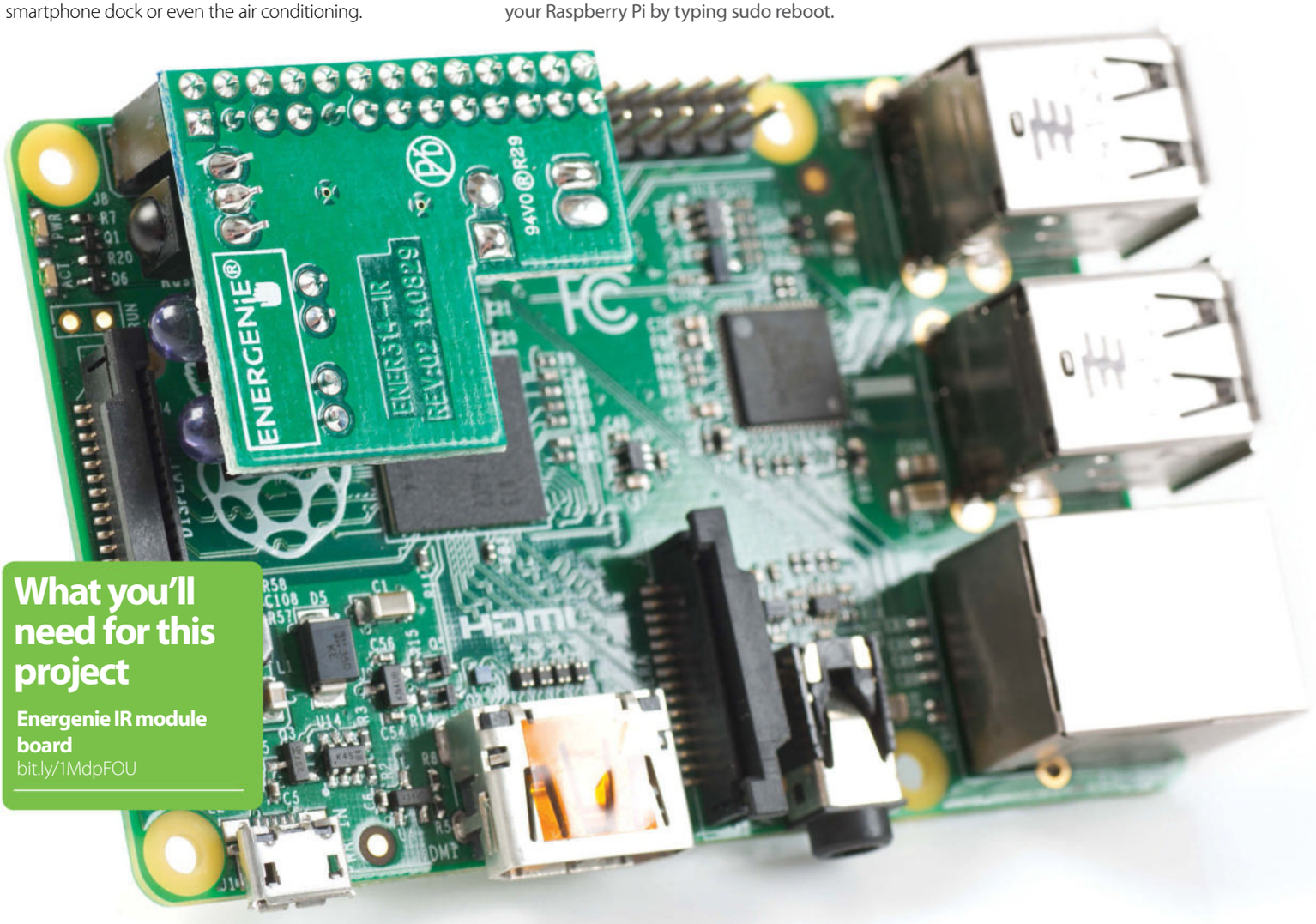
Now press Ctrl+X and save the file, then reboot your Raspberry Pi by typing sudo reboot.

02: Install the software

Next install the LIRC software; this stands for Linux Infrared Remote Control. It is the program that enables you to interact with your television and transmit commands accordingly. In the LXTerminal type:

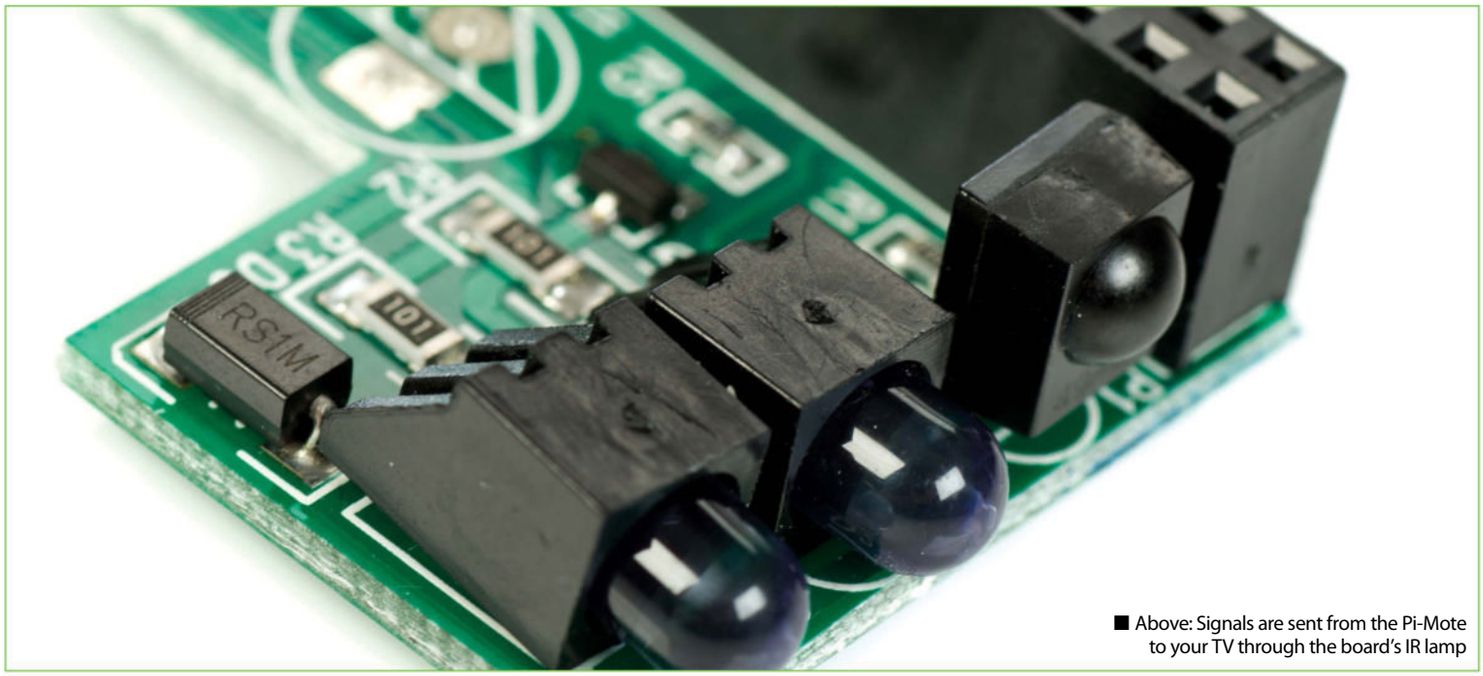
```
sudo apt-get install lirc
sudo apt-get install lirc-x
```

Restart your Pi. The default GPIO pins used when no pins are specified are pin 12/GPIO 18 for input for received infrared signals and pin 11/GPIO 17 for output for transmitted infrared signals.



What you'll need for this project

Energenie IR module board
bit.ly/1MdpFOU



■ Above: Signals are sent from the Pi-Mote to your TV through the board's IR lamp

03: What are all these files?

It's worth clarifying some file names and folders. All of the files used are stored in /etc/lirc, and there are two main ones: the hardware and LIRC configuration files (hardware.conf and lircd.conf). The latter holds all the data about your remote control, such as signal length, names of buttons and header details. This is the file to edit to emulate a remote control.

04: Test the IR receiver is working

To test that the IR receiver will 'pick up' the transmission from your remote, you need to stop the LIRC daemon, enable the test mode and then start the mode 2 testing. This runs a program to output the mark-space of the IR signal. It measures the pulse and space length of signals, returning the values to the terminal. Open LXTerminal and enter the commands below, then grab your control, point it at the IR receiver and press some buttons:

```
sudo /etc/init.d/lirc stop
sudo modprobe lirc_rpi
sudo mode2 -d /dev/lirc0
```

You should see something like this:

```
space 16300
pulse 95
space 28794
pulse 80
space 19395
space 28794
pulse 80
```

05: Transmit a signal

Now your remote is recognised and the IR board is working, you can use the irsend tool to record the signals and use these measurements to send commands to your TV. First alter the hardware.conf file located in the /etc/lirc folder:

```
sudo nano /etc/lirc/hardware.conf
```

Make the following changes:

```
LIRCD_ARGS = "--uinput"
DRIVER = "default"
DEVICE = "/dev/lirc0"
MODULES = "lirc_rpi"
```

Press Ctrl+X to save the file – but don't rename it – then press Y and Return. Now restart the lirc daemon by typing :

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

06: Get a new LIRC file

Next locate a compatible lircd.conf file that contains all the information you need to know about the remote's buttons and each of their specific functions. (Instead of buttons they are referred to as keys.) The good news here is that you do not need to create this file from scratch, because it is possible to use or adapt an existing remote control configuration file. These can be found on LIRC supported remote index websites, such as **lirc.sourceforge.net/remotes**. (Some remote configuration files will not be available on the LIRC index. This is because not all remotes are supported by LIRC. You may find more up-to-date files hosted on the GitHub website.) Download the correct file for your remote and save it in the Pi/Home folder. Note that we're using a Samsung remote in this tutorial.)

07: Use your new LIRC file

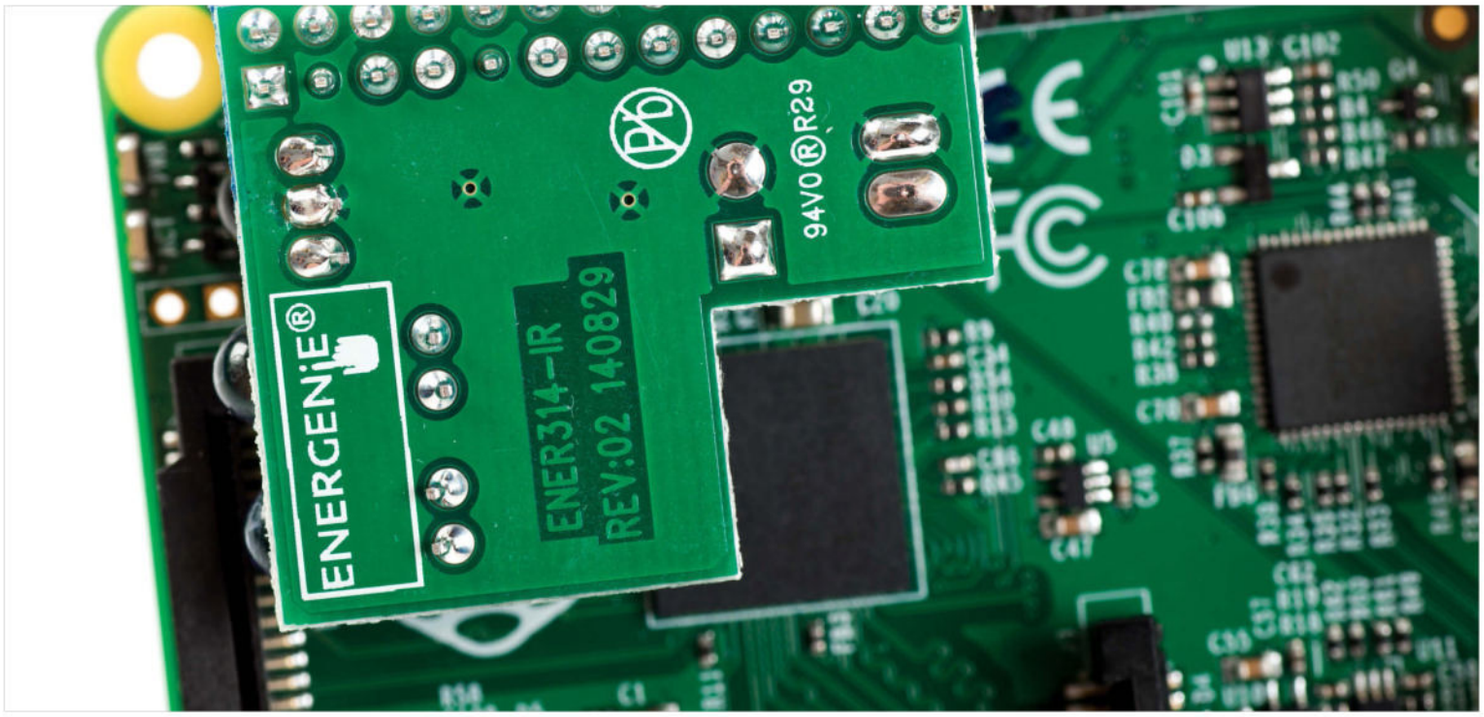
Once you have a suitable LIRC file, copy and paste over the code from the file that you have found into lircd.conf, which is stored in the /etc/lirc folder on your Pi. This will overwrite any current configuration setup that you have, but it is only an issue if you have already set up a configuration file. Open the existing lircd.conf file by typing:

```
sudo nano /etc/lirc/lircd.conf
```

Now copy and paste over the code from the new LIRC file. Press Ctrl+X to save the file, but do not change the name. Then restart the LIRC by typing sudo /etc/init.d/lirc restart. It will say that it failed to stop the daemon, but this is because it is already stopped and therefore it cannot be stopped again!

Lircd daemon

The LIRC enables you to decode and transmit infrared signals. The one used in this project is the lircd daemon that decodes IR signals received by the device drivers and accepts commands for IR signals to be sent if the hardware supports this. You could adapt your project to create a remote control for your media device using a simple Apple remote control as the input.



■ Above: Energenie's Pi-Mote controller board costs £10, and you can get RC plug sockets with it for an extra £10

08: Test the lirc.conf file

Assuming that the lircd.conf file is compatible and your setup is successful, you can now test that it works. In the terminal window, type irsend LIST Samsung " " (replacing Samsung with the name of your television remote, which is stated at the top of the lircd file). This code will list all the KEYS (buttons) that are installed in the lircd.conf, displaying a list of the commands that you can send to your television.

09: Hack your television

Now that your lircd.conf configuration file is recognised, you are ready to control your television. The line of code is very simple and follows the format:

```
irsend SEND_ONCE Remote_Name Remote_Button
```

For example, to control the TV Menu you would type in the LXTerminal: irsend SEND_ONCE Samsung KEY_MENU. This will send the Menu IR signal and the menu will appear on the TV. In order to send a different button signal, alter the KEY_ field. The key prefixes can be found in the lircd.conf file or by listing the keys with: irsend LIST Samsung " ".

10: Make your own lircd.conf file – part one

Sometimes you may not find a compatible lirc. conf file and instead you have to create one yourself. This involves running a program called irrecord, pointing your remote at the IR board and then simply pressing loads of buttons! This will then record the signals from your remote where you can assign KEYS to each of the signals. Stop the LIRC software by typing in the terminal:

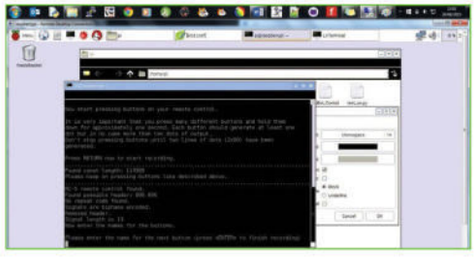
```
sudo /etc/init.d/lirc stop
```

11: Make your own lircd.conf file – part two

Next create a new lircd.conf configuration file and save the output. In the LXTerminal type:

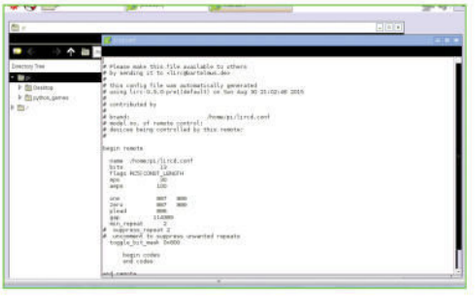
```
irrecord -d /dev/lirc0 ~/lircd.conf
```

This will open the 'create' program and will present you with instructions on how to record and save the signals. There are two stages to this: the first part involves you repeatedly pressing the buttons on the remote until there are two lines of dots on the screen. This measures and records the signals being sent from the remote. Do this in a logical order, starting at the top of the remote and working downwards.



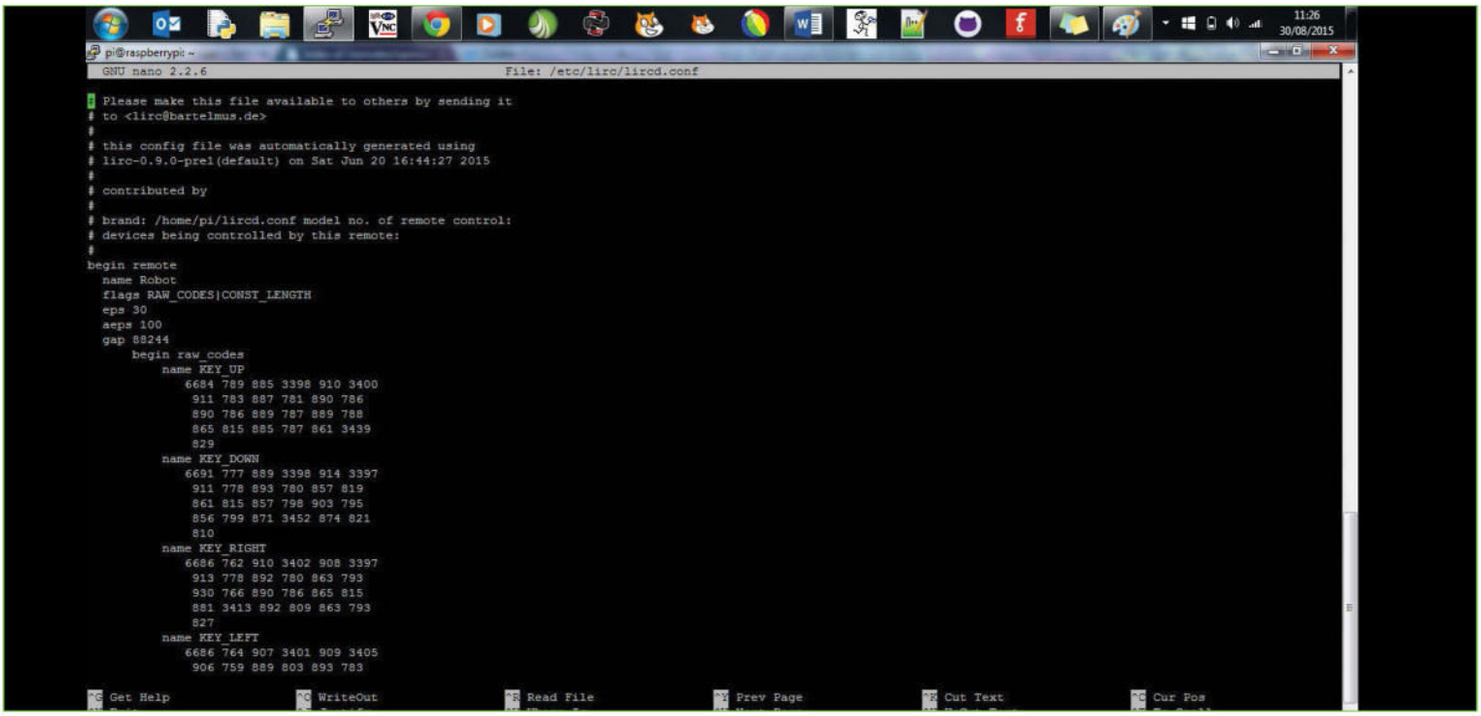
12: Make your own lircd.conf file – part three

Once the two lines of dots have been completed, your remote has now been recognised. The program will ask you to enter the names of the keys for each of the signals it has recorded. Follow each of the on-screen prompts, typing the names for each of the remote buttons/keys. For example, type KEY_UP and then press the corresponding 'up' key on the remote. You will then be prompted to type in the name of the next key, for example KEY_BACK, then press the 'back' key on the remote and so on. Keep doing this until you have entered names for each of the recorded keys.



13: Rename the remote

Unlike the ready-made lircd.conf files, the name of the remote on line 14 will probably be set as "home/pi/lircd.conf". Under the heading begin remote, find the name label and rename "home/pi/lircd.conf" as something else. This makes it easier to refer to in the code line, for example, as "Samsung".



14: Transfer the lircd.conf file

Now your lircd.conf file is ready to transfer to the /etc/lirc folder, as shown previously in Step 7. The simplest method is to copy and paste over the code that you have just created, but this will overwrite any old configuration file setup that you have. If you want to keep a previous configuration then follow Step 15, else jump to Step 16. In the LXTerminal, type:

```
sudo nano /etc/lirc/lircd.conf
```

15: Not overwriting

If you have already set up a lircd.conf file or you want to use a new one and still keep the old one, then you can create a new configuration file. This is automatically saved in the /home/pi folder and can be copied over to the /etc/lirc folder. Firstly, make a backup of the original lircd.conf file, creating a copy of the file and saving it as lircd_orginal.conf. In the LXTerminal, type:

```
sudo /etc/init.d/lirc start
sudo mv /etc/lirc/lircd.conf /etc/lirc/lircd_
original.conf
```

Then copy over your new configuration file:

```
sudo cp ~/lircd.conf /etc/lirc/lircd.conf
```

Your original configuration file will be saved as lircd_original.conf.

16: Restart the LIRC

Now you have a configuration file you can use your remote control. Restart the LIRC by typing sudo /etc/init.d/lirc restart. As before, you can test that the lircd file is working by listing all the registered KEYS stored in the file; just type: irsend LIST the_name_of_your_remote " ". Send some commands to your television or device using the code:

```
irsend SEND_ONCE Remote_Name Remote_Button
```

For example, to control the TV Menu you would type in the LXTerminal: irsend SEND_ONCE Samsung KEY_VOLUME_UP. This will send the 'volume up' signal.

17: Common errors and code recap

Transmission error – this usually means that the lircd.conf file is not correct and contains an error. For example, using a file from 2007 instead of the 2015 version.

Connection refused – generally, this means the LIRC has failed or the hardware changes are not correct. Check the boot.config file and the hardware.conf file, then restart the LIRC by typing:

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

Restart the LIRC program – this one can prove useful after you have changed a file, such as the hardware.conf or lirc.conf:

```
sudp/etc/init.d/lirc restart
```

Stop the LIRC program – useful when testing the program:

```
sudo /etc/init.d/lirc stop
```

Start the LIRC program –
`sudo /etc/init.d/lirc start`

List all the Keys in the file –

```
irsend LIST Samsung "" #
```

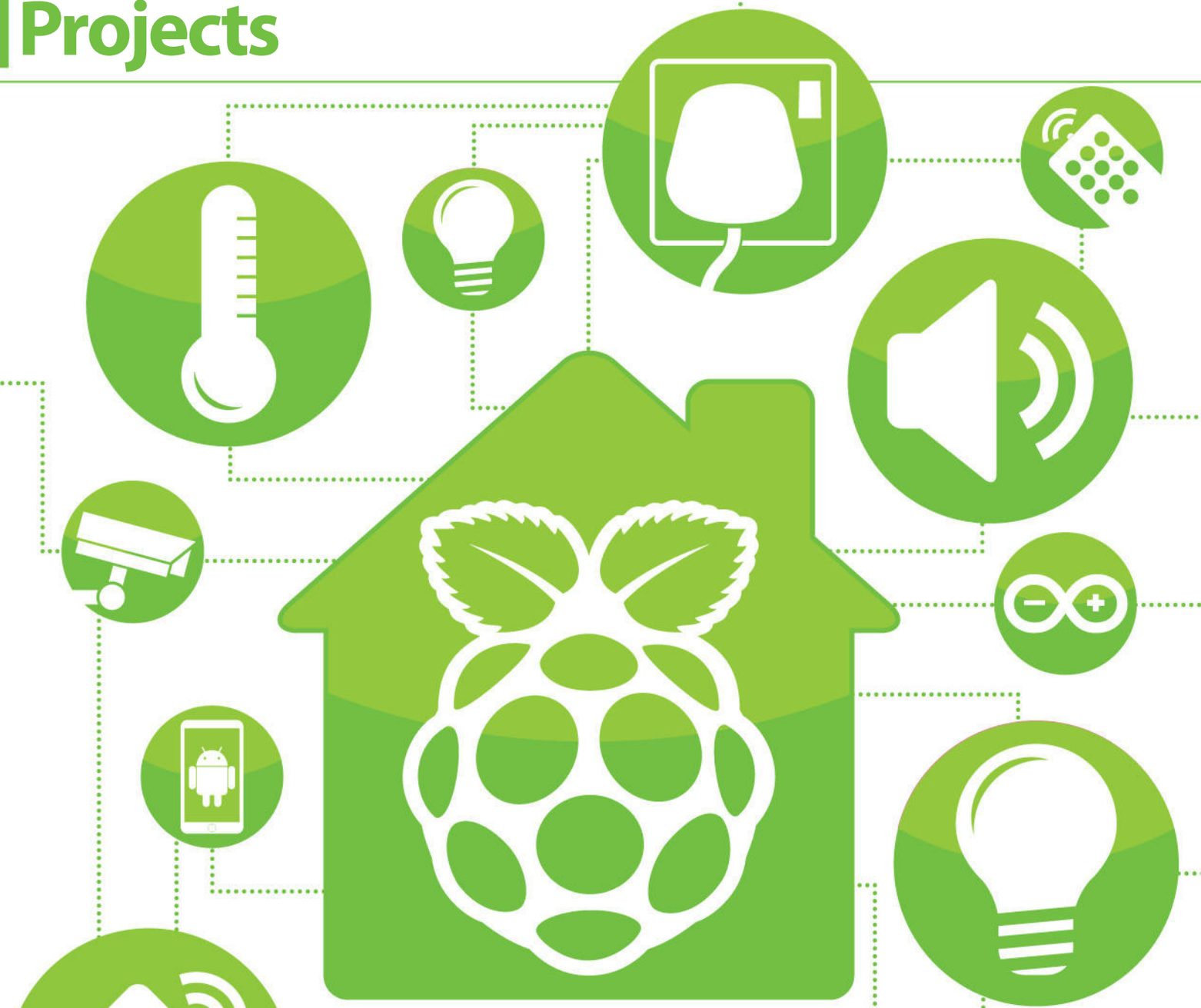
… replacing the word Samsung with the name of your remote.

Create a new bespoke lircd configuration file –

```
irrecord -d /dev/lirc0 ~/lircd.conf
```

Create a UI

It is possible to combine the IR board, the LIRC program and a web interface, which would open up many possibilities for projects. Combining the GPIO pins with a web server means that you can create a user interface that can be used to control your devices. Change the channel from your laptop or phone, turn the volume up or down or turn the TV off. A starter project can be found at **bit.ly/1O4CaMU**.



HOME AUTOMATION WITH YOUR PI

Now you've mastered some basic projects, enhance your home with this advanced tutorial. Add your own smart infrastructure to control your environment, all custom-coded and controlled by your Raspberry Pi

The Raspberry Pi is the perfect little computer to have around the house doing work that requires just enough computing power to keep it running. It's not the purpose of the device, but it is just really good at it. File servers, media centres, etc – its size and flexibility make it an often surprisingly powerful tool.

So, taking advantage of the Raspberry Pi's portability and functionality, why not take it a step further? Instead of handling idle computing tasks around the house, what if we had it control the house? Through modern home automation and a bit of fancy coding, you can easily make the Raspberry Pi do a little bit more and control many aspects of your home.

We're going to run you through not only setting up the controllable lights and thermostats and such, but also how to go about controlling them. Snarky voice interface not required.

Your smart home setup

Remote control sockets

Energy saving and green houses are a big thing right now, and you can buy power strips that will shut down every socket based on the draw from a single socket. This isn't always accurate, though, and being able to manually control the socket is not always easy if it's hidden away or part of a power strip. With the use of remote control sockets, you can control the power of anything from the Pi and a web interface, enabling better control and less use of device standby modes.

Lights

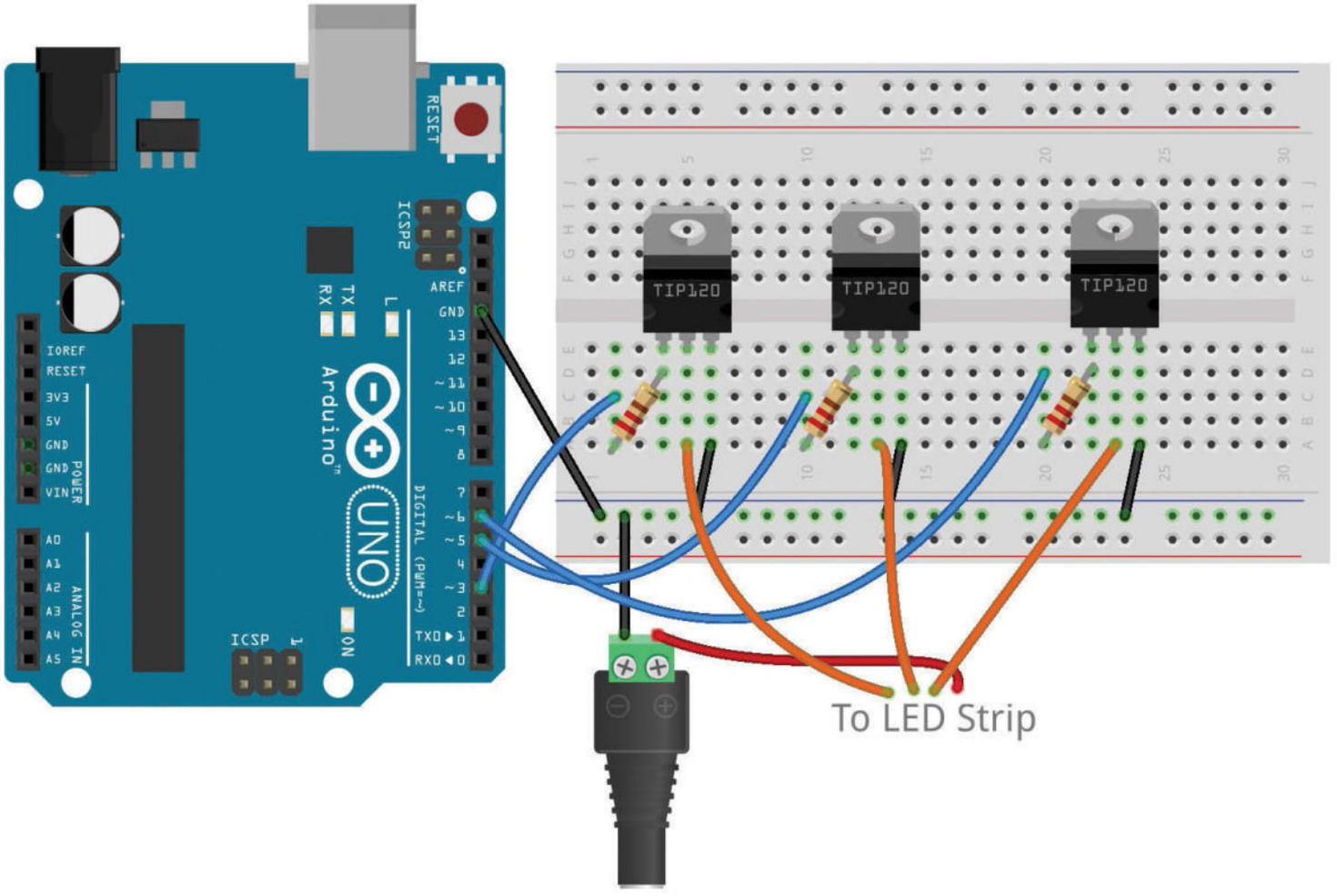
A classic home automation function is controlling the lights in the house depending on the time of day or how dark it is. There are many ways you can do this: the popular method right now is Wi-Fi enabled bulbs, allowing for direct control, but you can also use the remote control sockets or use strips of LEDs that can easily light a room and are much easier to manually interface with. With all of these methods separate from the automated control, you can also remotely control the lights to switch on and off as you please. It's not a good idea to try and spook house guests, though.

Thermostat

Technologies like Nest are becoming extremely popular, but connected thermostats have been around for a long time – longer for those with a soldering iron. While we're not going to be quite creating a thermostat that 'learns', using it for external control and monitoring is easy enough when you have the right equipment. We'll be concentrating on the monitoring part in this tutorial, using a thermistor and a bit of calibration to figure out the optimum temperature.

Security doorbell

It's surprisingly easy to get one of these home security systems set up. Using technology created for security cameras and streaming, you can create a live feed using the Raspberry Pi that can be displayed wherever you want. This can be done quite simply on the Pi using a webcam or even the Pi camera itself, and you can even try and hook it into the doorbell if you want a really cool party trick.



■ Above: The three separate colours used for the RGB sequence are wired manually to the LEDs

Build your own automated system

Discover how to remotely switch sockets, your garage door, control a strip of LED lights and more

Resources

Raspbian:
www.raspberrypi.org

Arduino:
https://www.arduino.cc/en/main/software

Remote control sockets

LED light strips

TIP120 NPN transistor

TMP36 temperature sensor

USB webcam

Remote control sockets

We are going to use a 433MHz receiver and transmitter module (these can be found for a couple of pounds on eBay – simply search for '433MHz' and you'll find what you're looking for) connected to an Arduino to switch a pack of remote control sockets. We used a pack of four remote control sockets from Energenie (£17 on Amazon). Remote control sockets are ideal for items such as floor-standing lamps, and anything else without an on/off switch. In our expert's case, he has an audio mixer that doesn't have an on/off switch.

Once you have the sockets set up to work with the remote, you can use the 433MHz receiver and a simple piece of Arduino software to capture the message sent by the remote so it can be sent later using the transmitter module. The modules have very simple wiring: 5V, Ground and Data. The receiver has two pins for the data line but you only need to connect the Arduino to one of them.

The beauty of sniffing remote control codes is that it can be used for anything you like that works at 433MHz – remote control garage doors or light switches are likely to use 433MHz.

LED light strips

LED light strips are great. They are very easy to find on Amazon and cost about £15 for a length of five metres with a 12V power supply and remote. They have adhesive on the back so they can be stuck to a surface. Alternatively, you can just leave it on the reel, as that will still give off a lot of light.

Each LED on the strip has an individual red, green and blue colour component. These colours can be

set to any brightness level between 0 and 100% using pulse width modulation. This is where the signal for each colour is a square wave that is on a certain amount of the time. For example: if the Red signal had a 50% duty cycle then it would be on for 50% of the time, resulting in reduced brightness. The strip has four connectors on it: +12, Red, Green and Blue.

The colour connectors are the ground pins for each colour. When the pin for a colour is connected to ground, the circuit is completed, allowing 12V to flow through the strip and to ground. We will use a high current transistor for each colour to control the connection. Note that the ground from the Arudino and power supply should be connected together on the breadboard. The 12V supply for the LEDs should not connect to the Arduino – only to the LED strip – or the Arduino will get damaged.

We used a TIP120 NPN transistor, which is capable of switching 5 amps – this is plenty for our application (the power supply that comes with the strip only supplies 2.5 amps, and we can switch 15 amps because there is a 5 amp transistor for each colour). This transistor would also be good for controlling the speed of a fan or other kinds of motors.

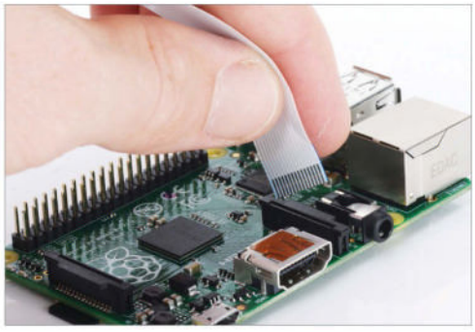
The pins of the TIP120 are Base, Collector and Emitter, from left to right. The base is connected to the PWM signal from the Arduino via a 220-ohm resistor. The collector is connected to one of the colour pins of the LED strip, and the emitter is connected to ground. Note that when passing high amounts of current, these chips get hot.

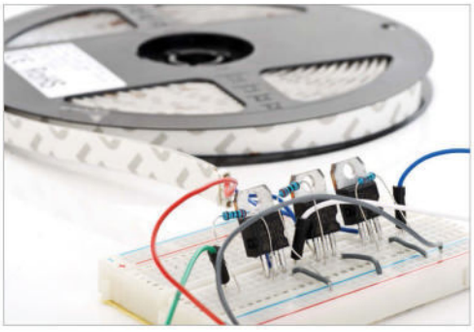
Also, it is wise to use solid core wire from the emitter-to-ground and collector-to-LED strip wires because they can safely handle more current than breadboard jumper wires. You need to ensure that none of the wires connected to the strip can short, otherwise you could damage the LEDs in the strip.

Our setup here is only temporary – it could be worth moving the circuit onto veroboard so that it is more stable once you have tested that it works on a breadboard.

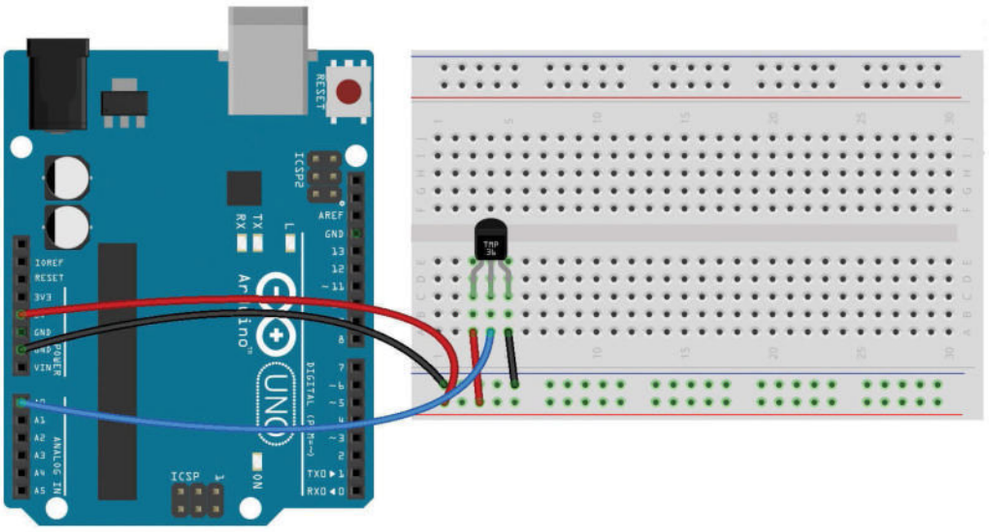
Temperature sensor

The TMP36 is a very simple IC. It has three pins: 5V supply, ground and a voltage out from 0-2V to represent temperature. This variable voltage can be read with the analogue in pins of an Arduino. The formula is: Temp in °C = [(Vout in mV) – 500] / 10, so a voltage of 0.7V would be 20°C.





■ Left: The Raspberry Pi camera plugs directly into the Raspbery Pi itself, not requiring extra wiring
Right: Here's a better look at the colour controllers hooked up to the LED lights



■ Above: The temperature sensor is very simply wired up, connected to positive and ground rails along with a data pin

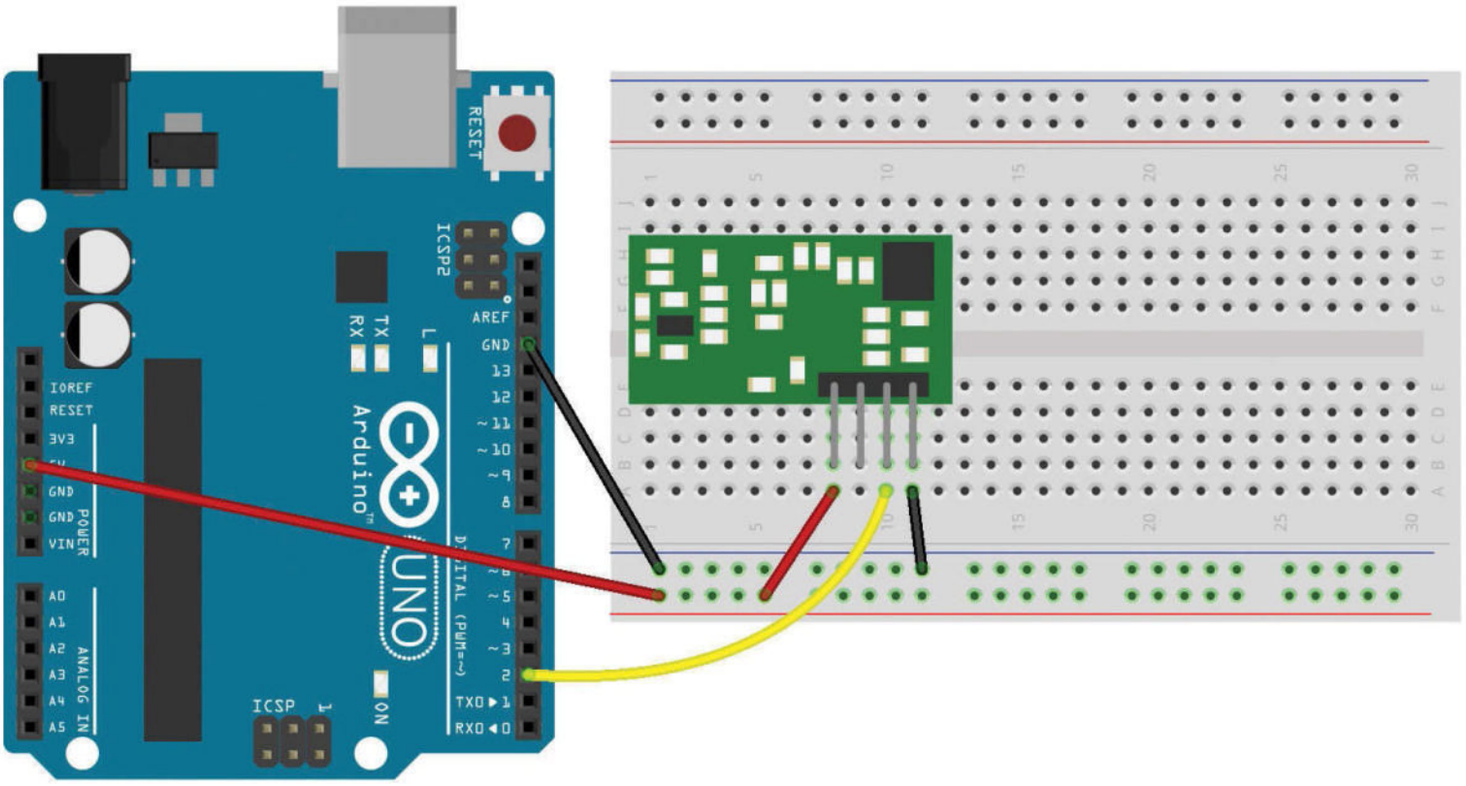
Camera

You can either use a USB webcam or Raspberry Pi camera for the video stream. The Raspberry Pi camera should be connected with the blue plastic facing the Ethernet connector, and the exposed traces of the ribbon cable facing towards the HDMI connector.

"Remote control sockets are ideal for items like floor-standing lamps, and anything else without an on/off switch"

Use more hardware

A more involved option for home automation could be remote control curtains or remote control blinds. These are an excellent option because you can use them on a timer to wake yourself up, and also have them close whenever the sun goes down. These devices are likely to use the same 433MHz frequency that the remote control sockets use, so you could add them to the software we're going to use without much effort. If you wanted to put them on a timer you could roll your own Python script that sends the open and close commands to the Arduino (we'll discuss how this works later on) at the right times. This could be combined with Wi-Fi light bulbs so that the lights can come on as the curtains close.



■ Above: The radio receiver needs power for it to work and then a connection to the data pin

Control your automated system

Configure your Raspberry Pi and Arduino to work in harmony for an automated home

We are going to use heimcontrol.js as the control software for our automated system. This is a home automation web interface written in Node.js that runs on the Raspberry Pi, and sends low level serial commands to an Arduino connected via USB to control various hardware connected to it.

Although some of the things the Arduino does can be done with the Raspberry Pi in theory, the Raspberry Pi does not have the ability to read analogue voltages, so temperature and light sensors would not be possible in this case. Also, the Raspberry Pi only has one hardware pulse width modulation output, compared to the three that are needed for the LED strip.

Raspberry Pi prep

Use the latest Raspbian image as a starting point for this project. Log into the Pi using the default username of "pi" and default password "raspberry". The first thing you'll need to do is use **sudo raspi-config** to enable the camera and resize the root filesystem so the image uses all of the available space on the SD card. Once this is done, we can update our packages to the latest version and then install some extra packages that will be required later on:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install libboost-dev arduino streamer
```

Next, we need to download a precompiled Node.js package and a MongoDB package (both are required by heimcontrol.js). You will be able to find these packages at **http://liamfraser.co.uk/lud/homeautomation**, in case they go missing.

```
wget https://node-arm.herokuapp.com/node_0.10.36_armhf.deb
sudo dpkg -i node_0.10.36_armhf.deb
```

Check it has installed correctly:

```
pi@raspberrypi ~ $ node -v
v0.10.36
```

Time to install MongoDB…

```
wget https://github.com/tjanson/mongodb-armhf-deb/releases/download/v2.1.1-1/mongodb_2.1.1_armhf.deb
```

Resources

Node.js & MongoDB:
http://liamfraser.co.uk/lud/homeautomation

Receiver software:
http://liamfraser.co.uk/lud/homeautomation/ReceiveDemo_Simple.pde

```
sudo dpkg -i mongodb_2.1.1_armhf.deb
# Start service:
sudo /etc/init.d/mongodb start
# Automatically start service at system
startup:
sudo update-rc.d mongodb defaults
```

Now it's time to install heimcontrol.js, which our expert had to fork on GitHub in order to fix a couple of issues.

```
npm config set python python2.7
git clone https://github.com/liamfraser/
heimcontrol.js.git
cd heimcontrol.js
npm install
```

The install process will take a while as there's quite a lot of stuff to compile. Before you can access heimcontrol.js, you will need to know the IP address of your Raspberry Pi. You can find this out using the ip addr command. Our address was 172.17.173.41. Run heimcontrol.js by typing:

```
node heimcontrol.js
```

Note that you have to be in the directory where you cloned the repository for this to work. This is probably /home/pi/heimcontrol.js. Heimcontrol runs on port 8080, so type the IP address of your Pi into your web browser followed by :8080 – in our case the correct URL was: **http://172.17.173.41:8080.**

We have applied a patch that disables authentication by default, because it gets annoying if you have to log in every time you want to use the web interface. You can re-enable authentication by editing config/development.js inside the heimcontrol.js directory.

Now that we know heimcontrol is working, Ctrl+C out of it because we have more work to do before we can start using it. We need to load the video for the Linux camera driver for the Raspberry Pi camera so that it can be used with the streamer software we installed earlier. To do this, you need to edit /etc/modules using sudo and your favourite text editor (use nano if in doubt, so **sudo nano /etc/modules**). Add the line "bcm2835-v4l2" to the end of the file so the driver is loaded at boot. To load it instantly, run **sudo modprobe bcm2835-v4l2.**

Arduino prep and remote control scanning

We need to write some software to the Arduino called duino, which allows the ports to be controlled over serial from heimcontrol.js. This way of working is elegant because it allows more sensors to be added to the Arduino without any need to reprogram anything. We have already installed the Arduino software, so now we need to copy some libraries required by duino to the Arduino installation directory so that the software can be compiled:

```
Received 16738063 / 24bit Protocol: 1
Received 16738062 / 24bit Protocol: 1
Received 16738062 / 24bit Protocol: 1
Received 16738055 / 24bit Protocol: 1
Received 16738054 / 24bit Protocol: 1
Received 16738054 / 24bit Protocol: 1
Received 16738059 / 24bit Protocol: 1
Received 16738059 / 24bit Protocol: 1
Received 16738058 / 24bit Protocol: 1
Received 16738058 / 24bit Protocol: 1
Received 16738051 / 24bit Protocol: 1
Received 16738051 / 24bit Protocol: 1
Received 16738050 / 24bit Protocol: 1
Received 16738050 / 24bit Protocol: 1
Received 16738061 / 24bit Protocol: 1
Received 16738061 / 24bit Protocol: 1
Received 16738061 / 24bit Protocol: 1
Received 16738060 / 24bit Protocol: 1
Received 16738060 / 24bit Protocol: 1
Received 16738060 / 24bit Protocol: 1
```

■ Above: Here are the codes that we sniffed from our remote control socket controller

```
cd /usr/share/arduino/libraries
sudo cp -r /home/pi/heimcontrol.js/node_
modules/
duino/src/libs/* .
```

```
cd ~
```

Before we write the duino software to the Arduino, we want to use the Arduino to sniff the messages sent by the remote for the remote control sockets. To do this, connect the 433MHz receiver module (the wider module of the two modules with four pins instead of three – see diagram to left) to the Arduino. Connect the data pin (either of the middle pins) to pin 2 of the Arduino, VCC to 5V, and GND to GND. Download the receiver software using:

```
wget https://raw.githubusercontent.com/sui77/
rc-switch/master/examples/ReceiveDemo_Simple/
ReceiveDemo_Simple.pde
```

… which is again mirrored over at **http://liamfraser.co.uk/lud/homeautomation/ReceiveDemo_Simple.pde.**

Now we have to start the Arduino software. If you are connecting to the Pi via SSH then you can enable X11 forwarding to your local machine by logging in with:

```
ssh -X pi@172.17.173.41
```

Alternatively, you can type **startx** to start an x session if you have a screen and keyboard connected to the Pi. Once you have logged in with your desired method, type **arduino** into a terminal in order to start the Arduino programming software.

Control software

We have used some existing software in this article because writing an entire web interface and associated control software is quite involved. Also, there is no point in reinventing the wheel. However, you can easily write some of your own software in Python that talks to the Arduino to add functionality. The duino software has a very simple interface over serial: !0113001.

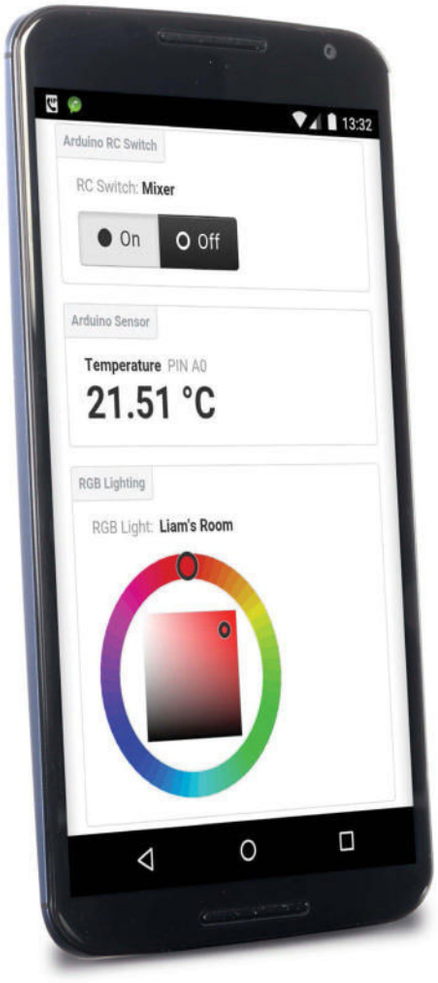
! starts the message, **01** means digitalWrite, **13** is write to pin 13, and **001** is the value to write to the pin (ie set it to high). The protocol is documented at **https://github.com/liamfraser/duino**. You can also add some of your own plugins to heimcontrol.js. If you look at one of the existing plugins and copy how that works, it shouldn't be too difficult. It's also a good excuse to learn some JavaScript in the form of Node.js.

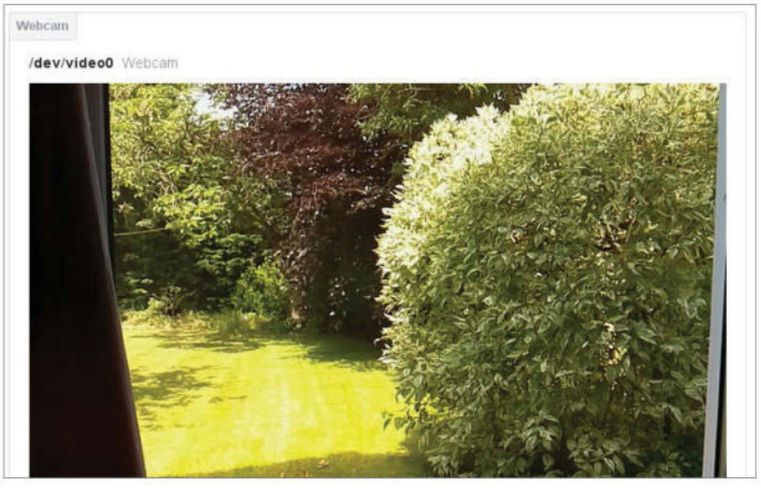
Open the ReceiveDemo_Simple.pde file that you just downloaded and upload it to the Arduino. Then open the serial monitor (Tools>Serial Monitor) and press the reset button on the Arduino. By pressing each button on your remote, you can see the code to switch each socket on and off. Make a note of the codes for each button because you will need to enter them later on. Our output can now be seen in the top-right image.

Once this is done, we can finally write the duino software to the Arduino. This process is the same as what you've just done except the file is located at /home/pi heimcontrol.js/node_modules/duino/src/duino/duino.ino.

The software might take a minute or two to compile. Once it has been uploaded, you can exit the Arduino software and press the reset button on the Arduino. Now we can put everything together and start adding our sensors to heimcontrol.js.

"The Raspberry Pi does not have the ability to read analogue voltages, so temperature and light sensors would not be possible"





■ Above: The heimcontrol.js interface works really well on mobile devices

Set up your control interface

Now is the time to start adding our sensors and devices to heimcontrol.js

01: Start heimcontrol.js on boot

Before we start adding devices to the system, it makes sense to start heimcontrol.js on boot. To do this, we need to add a line to /etc/rc.local, which is a script that gets run at boot by the root user. The file needs to be edited with sudo and your favourite editor, for example with the following text:

```
sudo nano /etc/rc.local
```

Add the following line before the line "exit 0":

```
su pi -c "node /home/pi/heimcontrol.js/
heimcontrol.js" &
```

Heimcontrol.js will be started automatically at boot from now on, but for now you can start it with **node /home/pi/heimcontrol.js/ heimcontrol.js.**

02: Add the camera feed

Go to Settings and select Webcam. Set the method to Streamer and the devices as /dev/video0. Pick an interval; we picked two seconds but you can pick any interval you like. Shorter intervals are more feasible on a Raspberry Pi 2, as it is generally more responsive. Click Save and then go back to the home page.

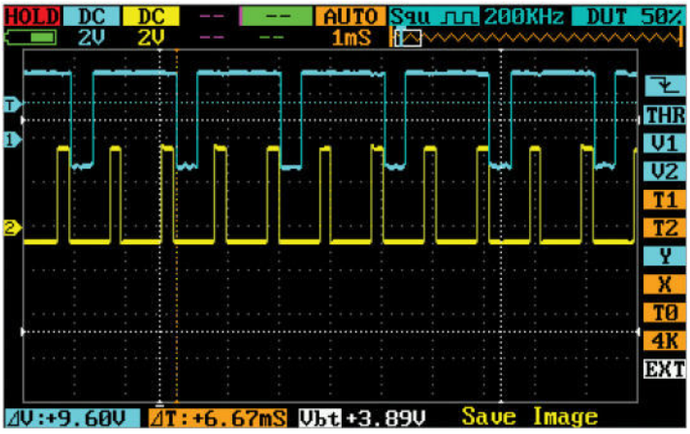
03: Prepare remote control socket codes

This is where you need the codes that you sniffed from the remote earlier on. The heimcontrol.js web interface takes the code as a binary string of 1s and 0s. However, the code we sniffed is in a different format so you'll need to convert it to binary using Python. Type **python2** into the terminal to open a Python interpreter. Format the integer you captured as a binary string like so:

```
>>> "{0:b}".format(16738063)
'111111110110011100001111'
```

Resources

Multimeter



■ Above: The pulse width modulation signals from the Arduino. One wave is on 20 per cent of the time, the other 80 per cent. This controls the brightness of each colour to create any colour from the three primary colours

04: Add the remote control socket

Go to the Settings menu and go to the Arduino section. Click the Add button and set the method to RC Switch. Set the code type to binary. Give the switch a name, enter the pin that the RF transmitter is connected to (in our case, pin 2) and enter the two codes that you just worked out for the on/off buttons. Go back to the home page and test that the switch works. If it doesn't, you might need to add an antenna to the transmitter by making a loop of wire. Check everything is connected correctly.

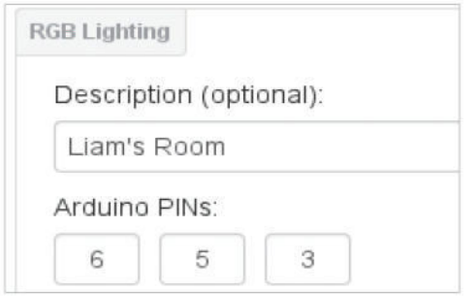
"The temperature sensor can be tricky because it needs calibrating"

Get remote access

If you want to remotely access your home automation web interface then you can use a dynamic DNS provider such as No-IP. This allows you to create a domain name that always points at your home IP address. Then if you port forward SSH on your router to your Raspberry Pi, you will be able to SSH into it from anywhere (obviously you'll want to change the default password if you go this route). From there you can port forward the heimcontrol web interface to your local machine. We have covered this in a previous Raspberry Pi File Server tutorial, which can be viewed here: **bit.ly/1LacazG** (instead of using port 9091, you need to use port 8080).

05: Add the temperature sensor

The temperature sensor can be tricky because it needs calibrating. A multimeter is a good idea so you can accurately read the analogue voltage. Go to the Arduino settings page and add a sensor. The formula for the TMP36 is: [(Vout in mV) - 500] / 10. We read 718mV with a multimeter, which would put the temperature at 21.8°C. Experiment with the formula by seeing what the raw value of x is from your sensor, but ours ended up as: (((x+45) * (5000/1023.0)) - 500) / 10. (5000/1023 is because the Arduino has a 10-bit analogue-to-digital converter, ie 0-1023 to read a voltage up to 5V.) Note that you have to ensure you have perfectly matched brackets, otherwise the software will crash because it simply tries to eval the string you put in.



06: Add the LED strip

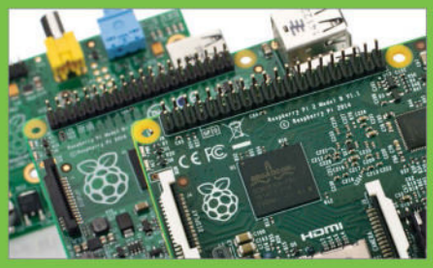
Finally, it's time to add the LED strip. Make sure the signal wires are connected to PWM-capable pins on the Arduino marked with a ~.

We used pins 3, 5 and 6. Go to Settings>RGB Lights. Ensure you have the signal wires connected to the correct colours. We used 6 for red, 5 for green and 3 for blue. Click Save and turn on the 12V supply. Select a colour from the colour picker. Now your strip should light up! Pick Red, Blue and Green to ensure everything is wired up correctly and good to go.

Expand into the future

Set up multiple systems

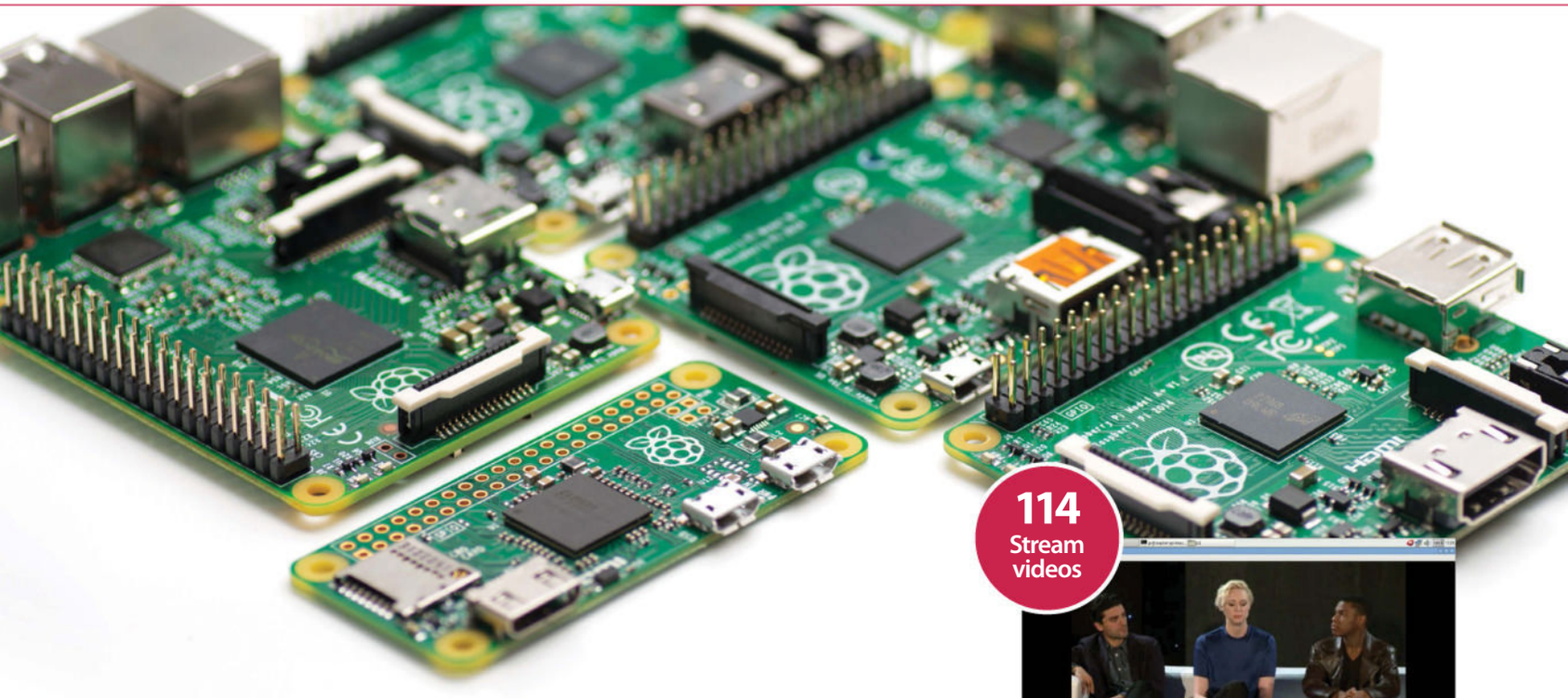
Now that you have heimcontrol.js set up in one room, you could extend the system by setting up multiple Raspberry Pis and have one in each room of the house – potentially a good way of using all of the older models that are gathering dust in your drawers. You could either have a master Raspberry Pi that reverse proxies multiple instances of the web interface depending on the link you give it: /livingroom, /kitchen and so on, or just simply have a bookmark for each room. If you were eager to have as few cables as possible, you could always look at getting a power-over-Ethernet injector/splitter that could send 5V power over an Ethernet cable along with the network connection.



Set up an audio system

A nice addition to home automation would be some kind of multi-room audio system. If there's already a Raspberry Pi in each room, you only need a cheap class-T audio amplifier (£20), a USB sound card for better audio quality and some bookshelf speakers to get this going. A pair of powered PC speakers would also do the trick. If you want just one set of speakers then mopidy (a music player daemon with Spotify support and a web interface) would be fine. Alternatively, you could look into setting up Squeezebox, which is multi-room audio software that was originally developed by Logitech but is now open source.

Programming



Learn the basics of Pi programming using Scratch and Python

"Thanks to the Miro media management software, we can automate all of this"

"Bring your creative ideas to life using simple lines of code"

Use Python and Scratch

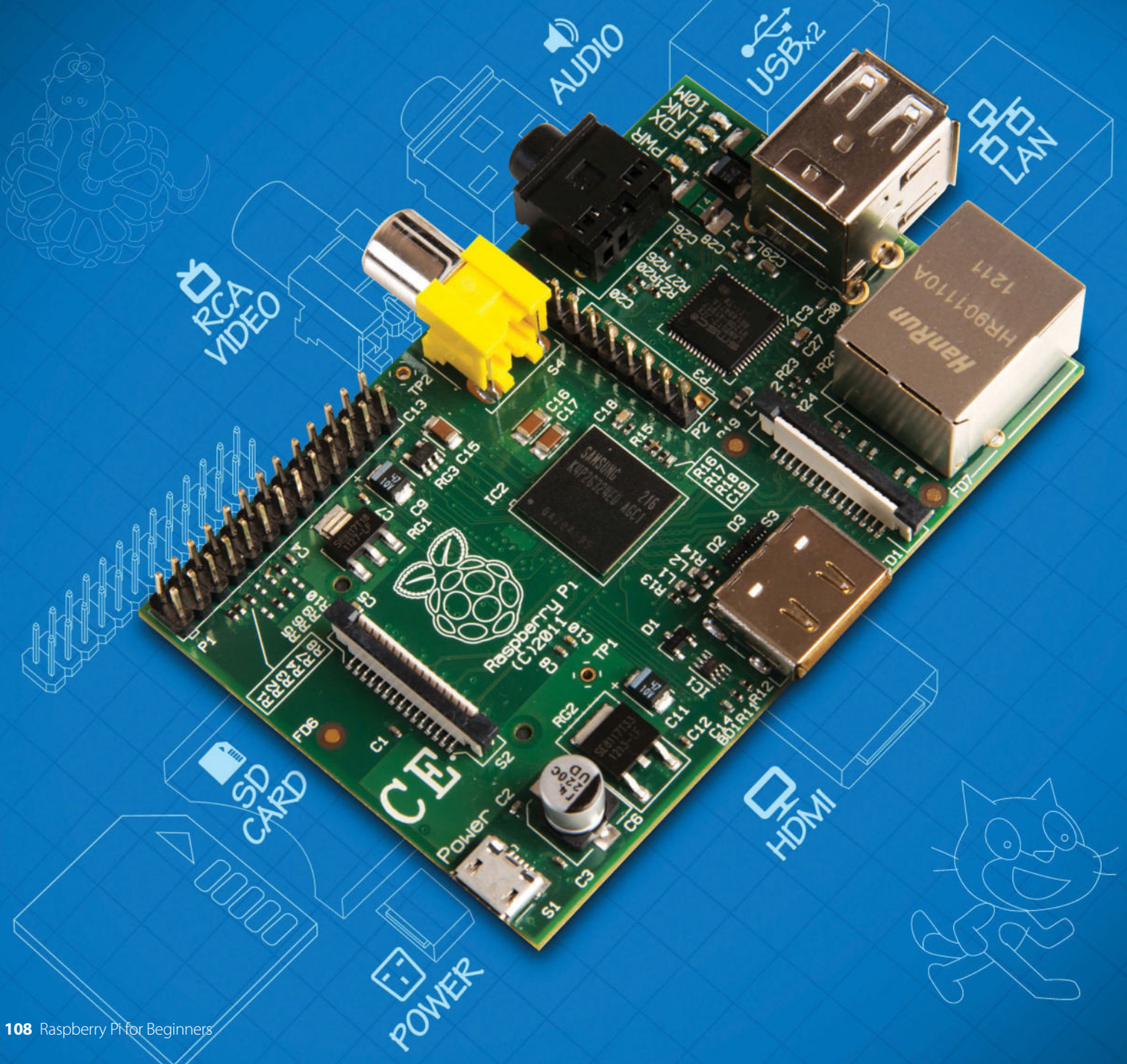
Explore the fun of coding with Python and Scratch

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.

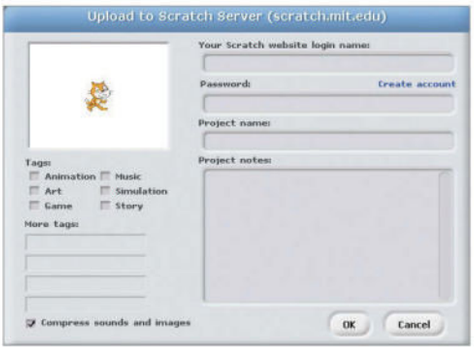
Scratch studio stage

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.

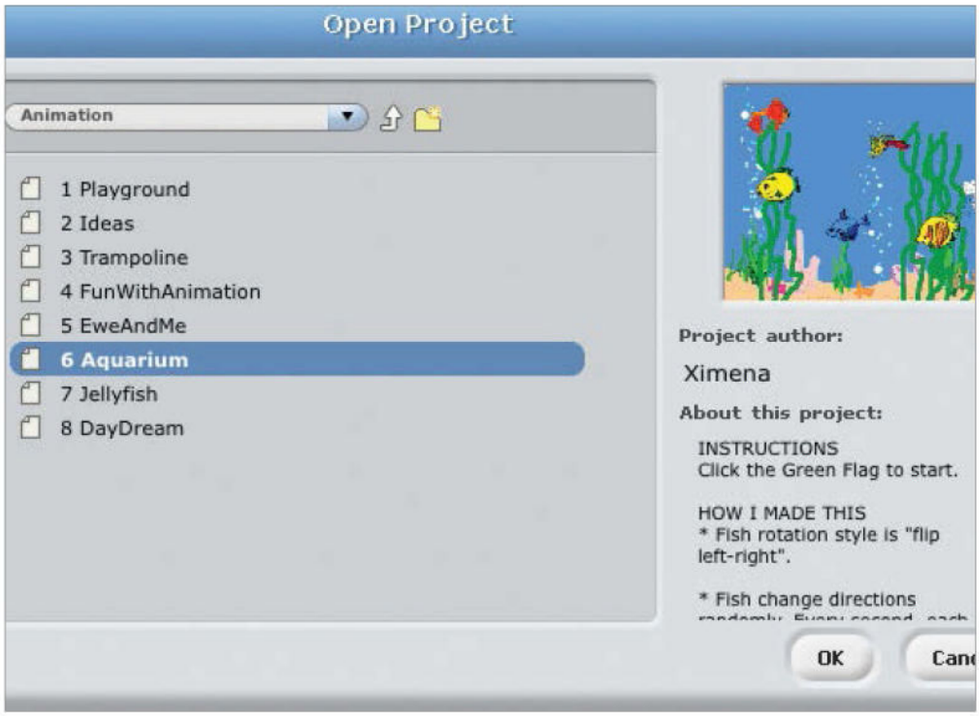
Scratch studio sprites

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.

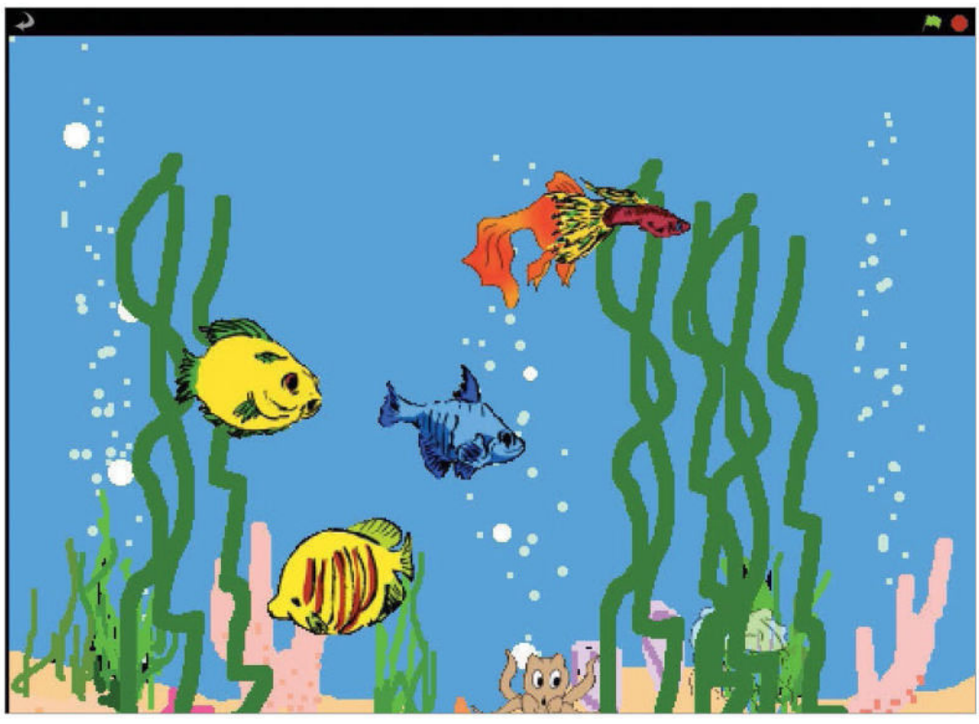
The three buttons across the top of this area offer various ways to create a new sprite. The first button opens up a blank canvas in the Paint Editor. The second button creates a new sprite based in an image file, as selected by the popup file section dialog window. The third will select a random image 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.

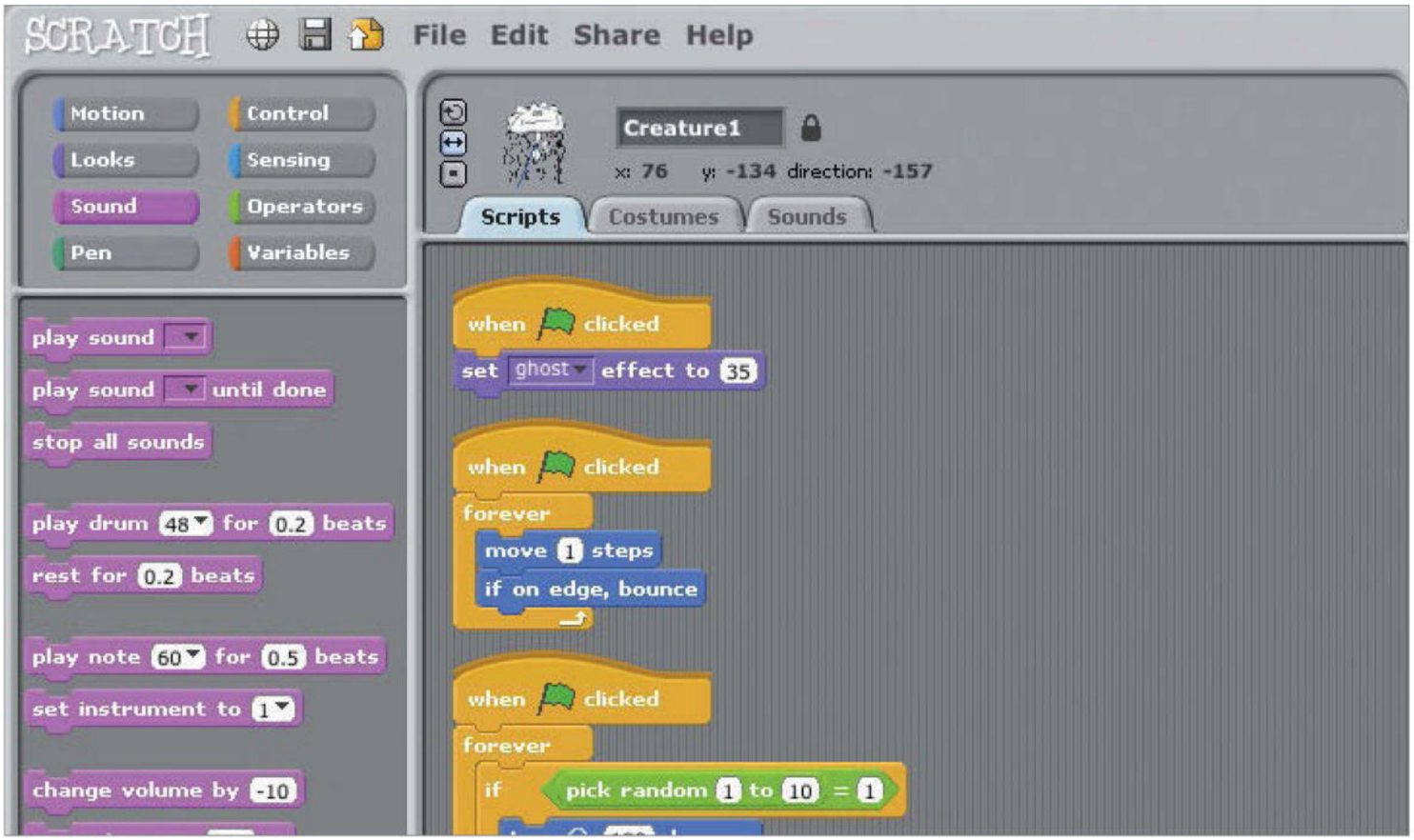


■ Fig 3: File Open Dialog – Use the Studio's File menu and 'Open…' option to load the Aquarium project

"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.

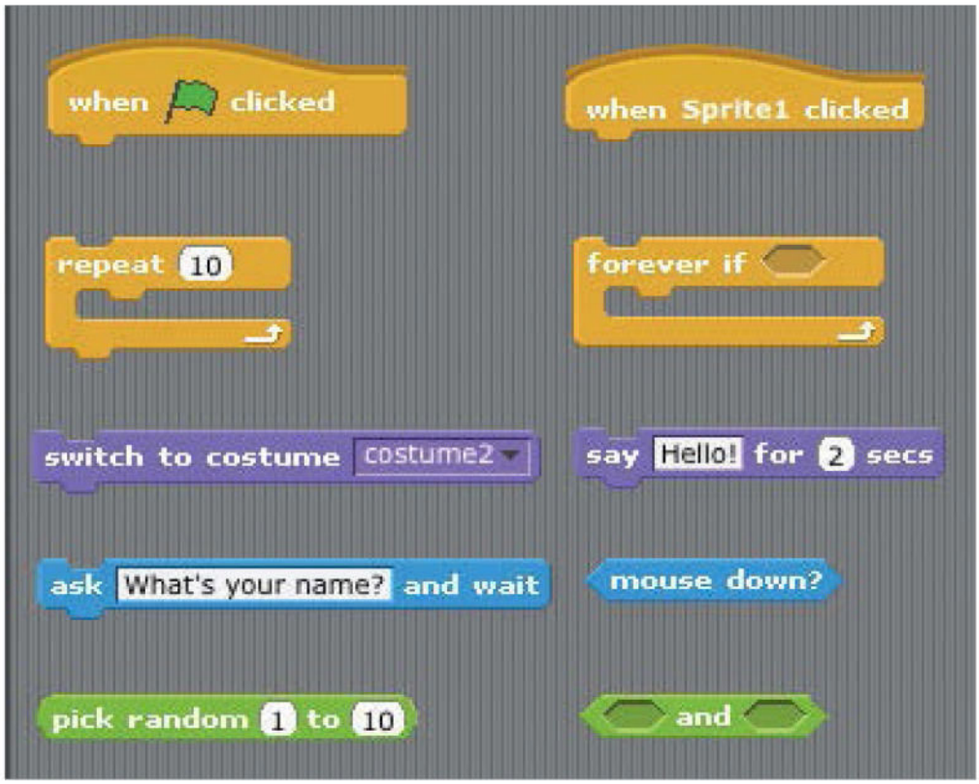
Block Script Walkthrough

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.

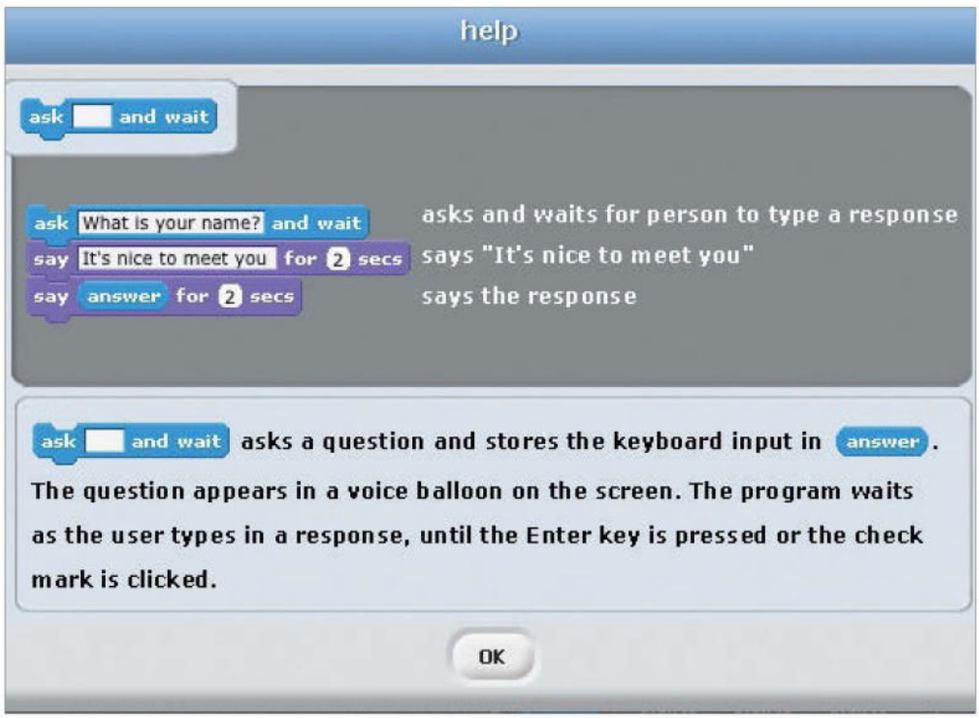
The first inner script block selects the next background image. Click on the 'Backgrounds' tab to view all the stage images. The second inner block simply pauses execution for a number of seconds. Setting the value to '1' means that this script will pause for a second before then performing the action 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

Stream internet TV to your Raspberry Pi

Get your favourite shows and video podcasts streamed automatically to your TV with Miro

What you'll need for this project

Raspbian Wheezy
HDMI cable
Monitor/TV display

Finding the content you're interested in viewing can take a while. Whether you're looking for internet TV stations, video podcasts, audio podcasts or shows syndicated online, taking the time to find and download them can be slow going, particularly if you have a busy lifestyle. You might even have no time to watch after you've waited for the download.

Thanks to the Miro media management software, we can automate all of this, and with the software running on a Raspberry Pi, you can easily build a compact system for downloading and playing back shows that you have an interest in. We're talking targeted TV on demand, which makes this project ideal for staying up to date with particular news and trends on a certain topic.

01: Set up your Pi with Raspbian

Sadly, Miro cannot run on Raspbian Jessie, so make sure you're using Wheezy, available via raspberrypi.org/downloads /raspbian. Ensure your Pi is connected to a TV or display via HDMI.

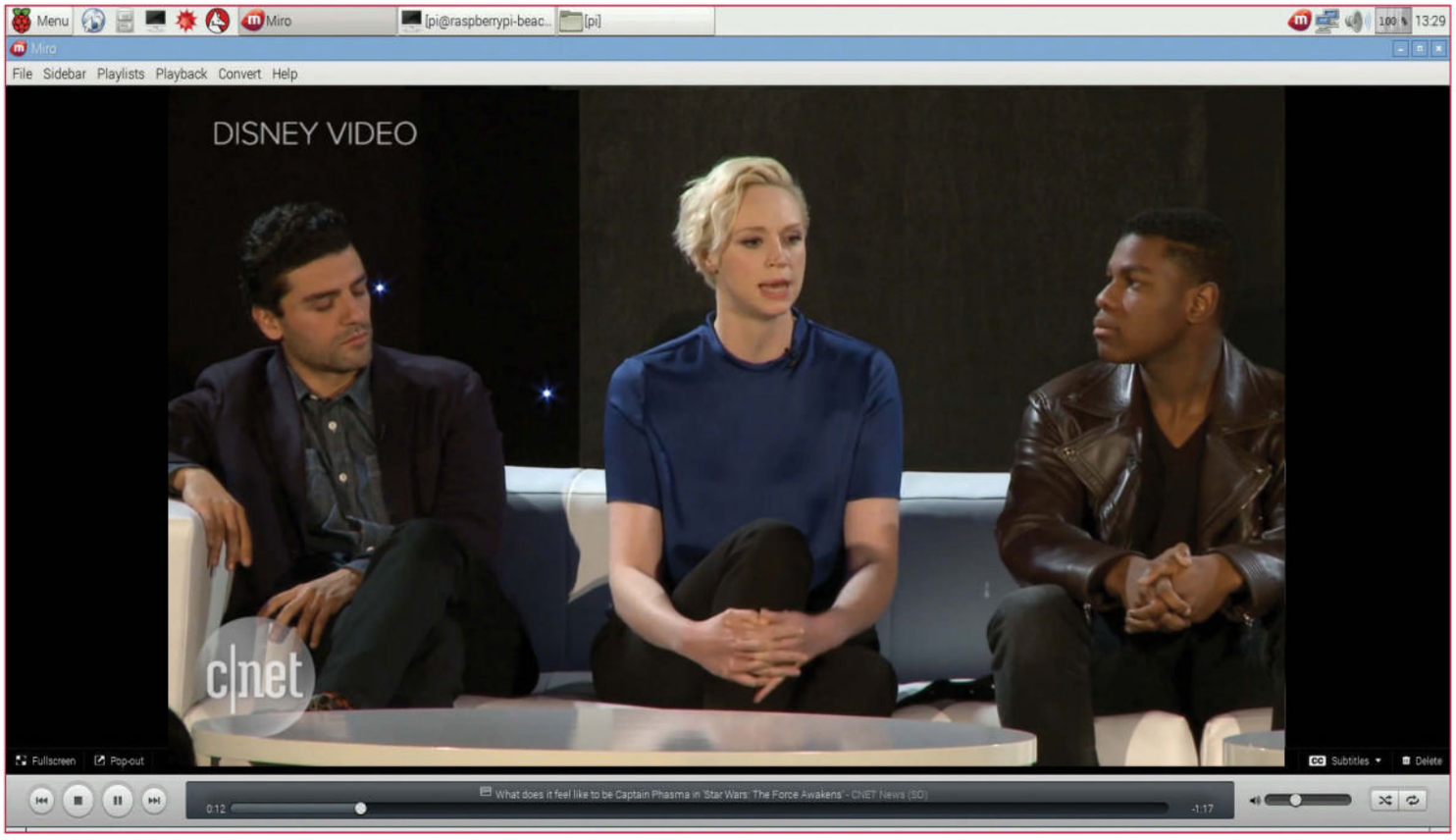
As Miro is a desktop application, you'll need your mouse and keyboard connected to configure it.

02: Install Miro

With Wheezy flashed to your SD card and your Pi booted up, open Terminal and enter:

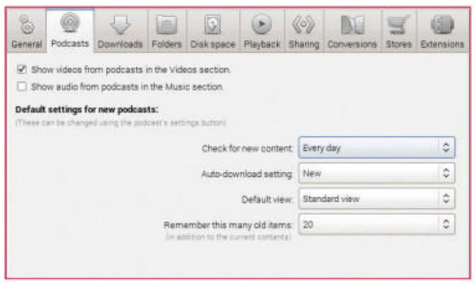
```
sudo apt-get install miro
```

Installation will take a few moments. Once complete, you'll find Miro in Menu>Sound and Video. Click it to get started.



03: Set Miro to launch at startup

Make sure Miro app is configured to launch at startup. Open File>Preferences>General and check 'Automatically run Miro when I log in' and 'When starting up Miro remember what screen I was on when I last quit'. Also set your Pi to boot into X using the raspi-config utility.

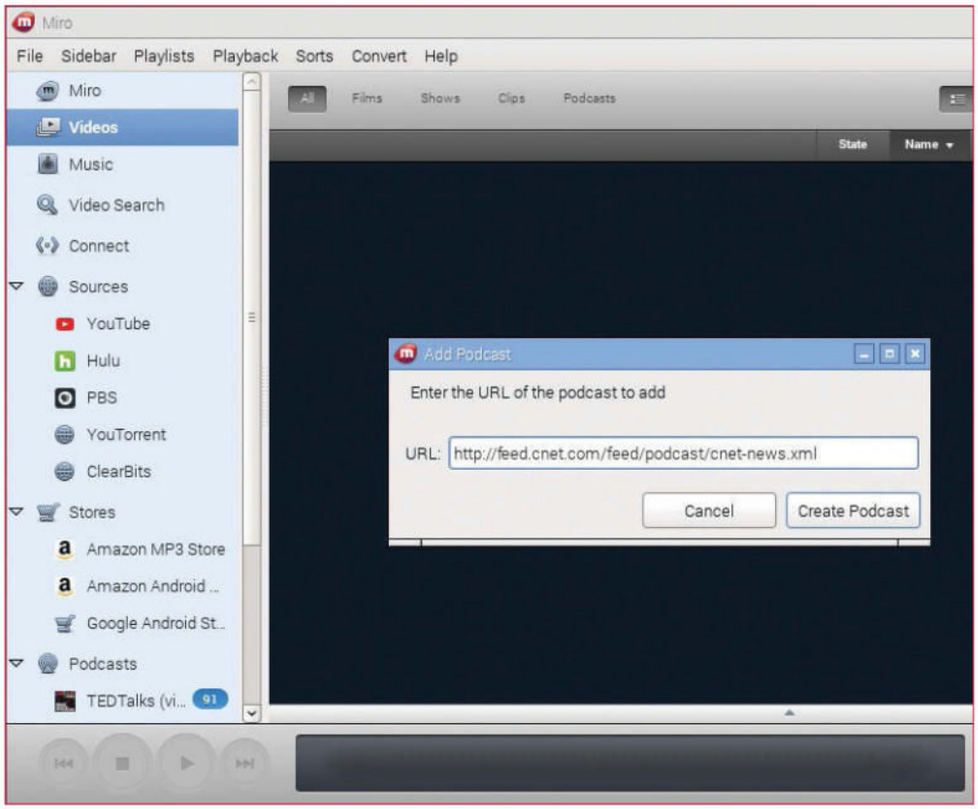


04: Check for content

Switch to the Podcasts tab and place a check in the box labelled 'Show videos from podcasts in the Videos section'. On the right-hand side of the window, set your preferred frequency for checking for new content. Miro will poll your favourite websites and feeds based on this setting.

05: Configure playback settings

Move now to the Playback tab, and check Play media in Miro. This limits reliance on other apps, which may drain resources. You should also click the Play video and audio items one after another radio button, and under Resume Playback, check the first and third items.



■ Fig 1: You can subscribe to all sorts of content, from internet TV channels to news podcasts

06: Source videos and podcasts

With Miro set up to play back the video and audio content you want to enjoy, it's time to find some! The best way to do this is to just check the websites that you regularly use for video and audio podcasts (preferably the former) and then copy the XML link.

07: Add podcast feeds

In Miro, open up File>Add Podcast and then paste the podcast feed URL into the dialog that appears, clicking Create Podcast when you're done.

The more links you add, the more regularly updated content will be downloaded to your Pi-powered Miro media manager, ready to watch on demand.

08: First time use

Remember earlier when we instructed Miro to behave a particular way upon launch? It's time to set that behaviour now, by opening the Videos view in the left-hand pane and playing the first video. Each time you boot your system, Miro will jump to this view and begin playing.

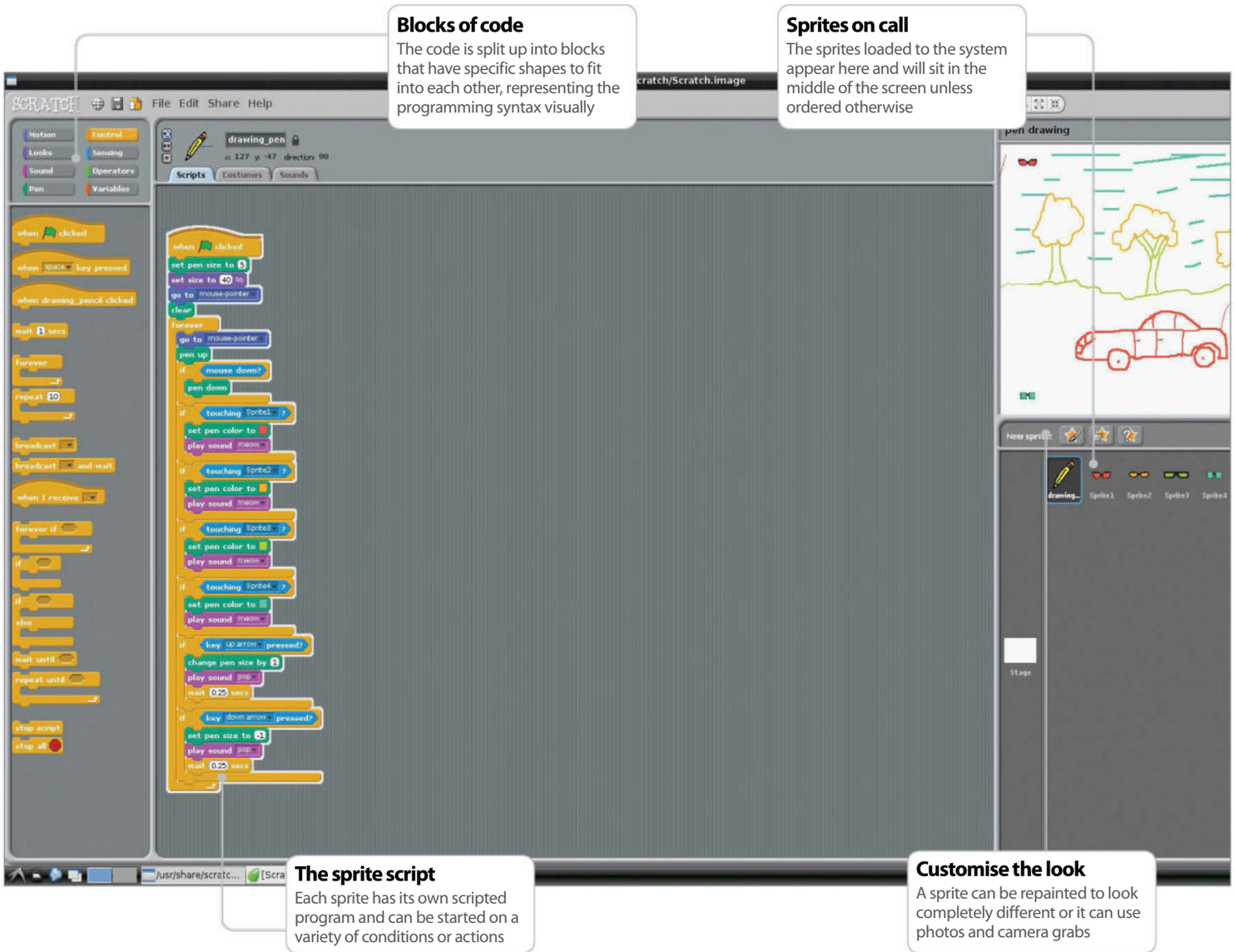
09: Avoid YouTube

As good a solution as Miro is to building a video podcast streaming center, displaying material that you're interested in on demand, it's sadly just no good for videos on YouTube. This doesn't really restrict you too much as there are plenty of other media outlets to cover, but it's worth mentioning if you're a frequent YouTube watcher. This is a shame, but shouldn't impact the way you use it – your Raspberry Pi now downloads focused content on demand!

"The more links you add, the more regularly updated content will be downloaded to your media manager"

Checking for new content

It is tempting to set a regular frequency for your content checking in File> Preferences>Podcasts, but note that checking too regularly is going to result in resources being hogged temporarily, which may result in an interruption if you happen to be actually watching something when Miro checks for new content. Limit polling to hourly or daily checks.

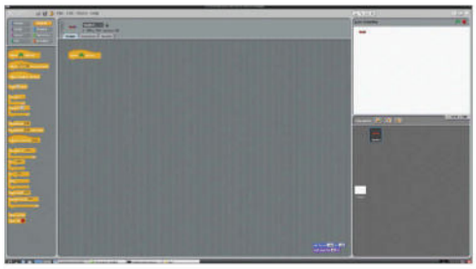


Create a simple drawing application

Here's how to add colour and pencil sizes using the built-in pen tool

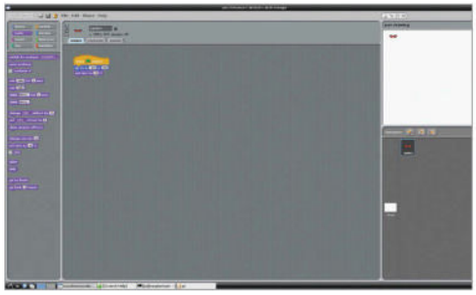
The aim of this tutorial is to use the Pen tool in Scratch to develop a more useful drawing application that you can then expand on yourself. It uses only the sprites that are bundled with the installation on the Pi but you could easily make some custom ones to generate the colour changes. In fact, one 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.



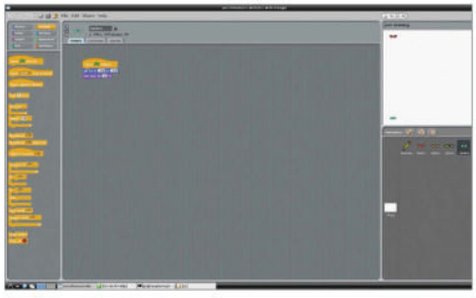
01: Load up sprites

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.

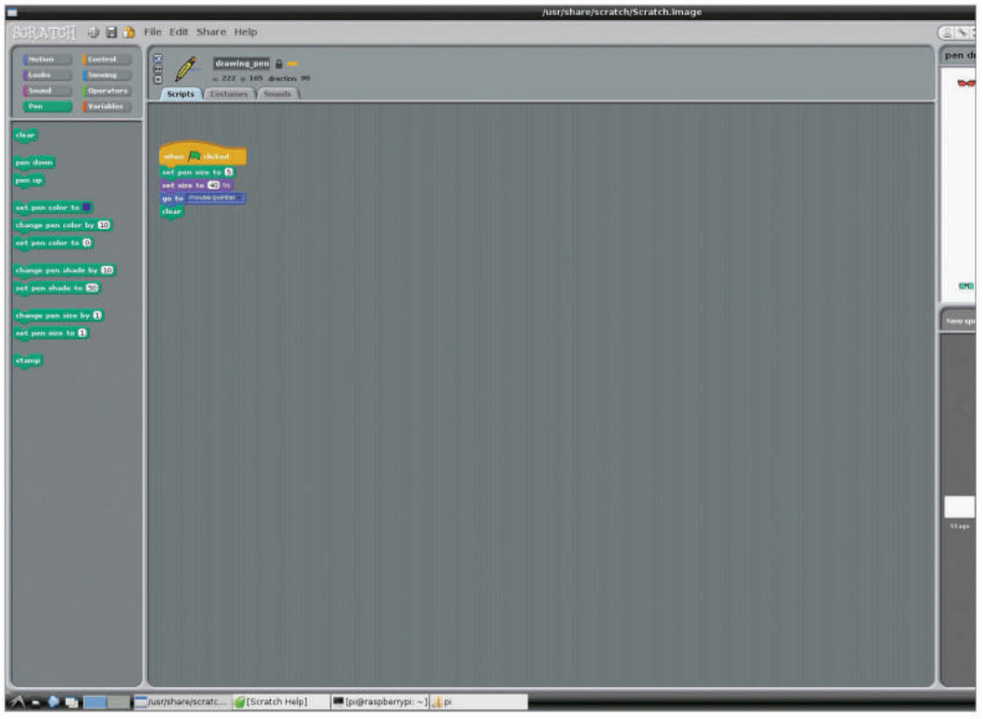


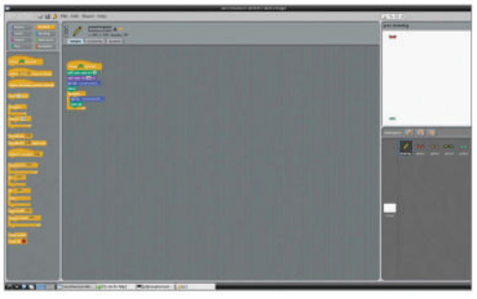
03: More sprites for colour

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

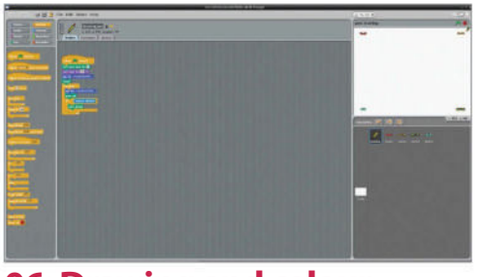
Under Control, drag a 'when clicked' header in. Get a 'set pen size' from under Pen. Click on the variable and enter 5. Go to Motion and get the 'go to' block. Click on the drop down arrow and select 'mouse-pointer'. Go to the Pen group and drag out 'clear'.





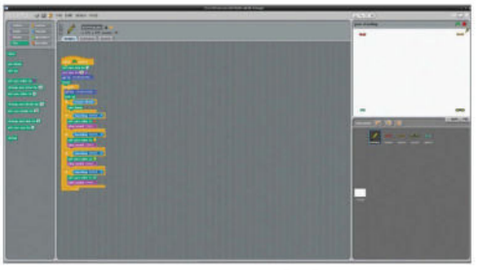
05: Control the pen draw

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: Set 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.

Set up the official 7-inch Pi Display

Assemble the brand new display module and get it up and running with your Pi

What you'll need for this project

Raspberry Pi official 7-inch display

Power supply

Small Phillips screwdriver

The Raspberry Pi Foundation has had plans for an official display since the very beginning – even the first Model B board that was released back in February 2012 had the necessary DSI connector. Development of the display module began in 2013, but as director of engineering Gordon Hollingworth explains in his recent blog post (**bit.ly/1Z75He0**) there were a number of issues along the way, including EMC (electromagnetic compatibility) testing and the sourcing of a high quality yet affordable display (which, from first-hand experience, are not easy hurdles to jump).

However, the output device is exactly what you would expect from the Raspberry Pi Foundation: a beautiful 7-inch multi-touch capacitive screen for the incredible price tag of just $60 (plus local taxes and shipping, which in the UK will come to around £51 including delivery). Like everything Raspberry Pi, it is well worth the money. In this tutorial, we will show you how to put it all together and then get up and running with your Raspberry Pi.

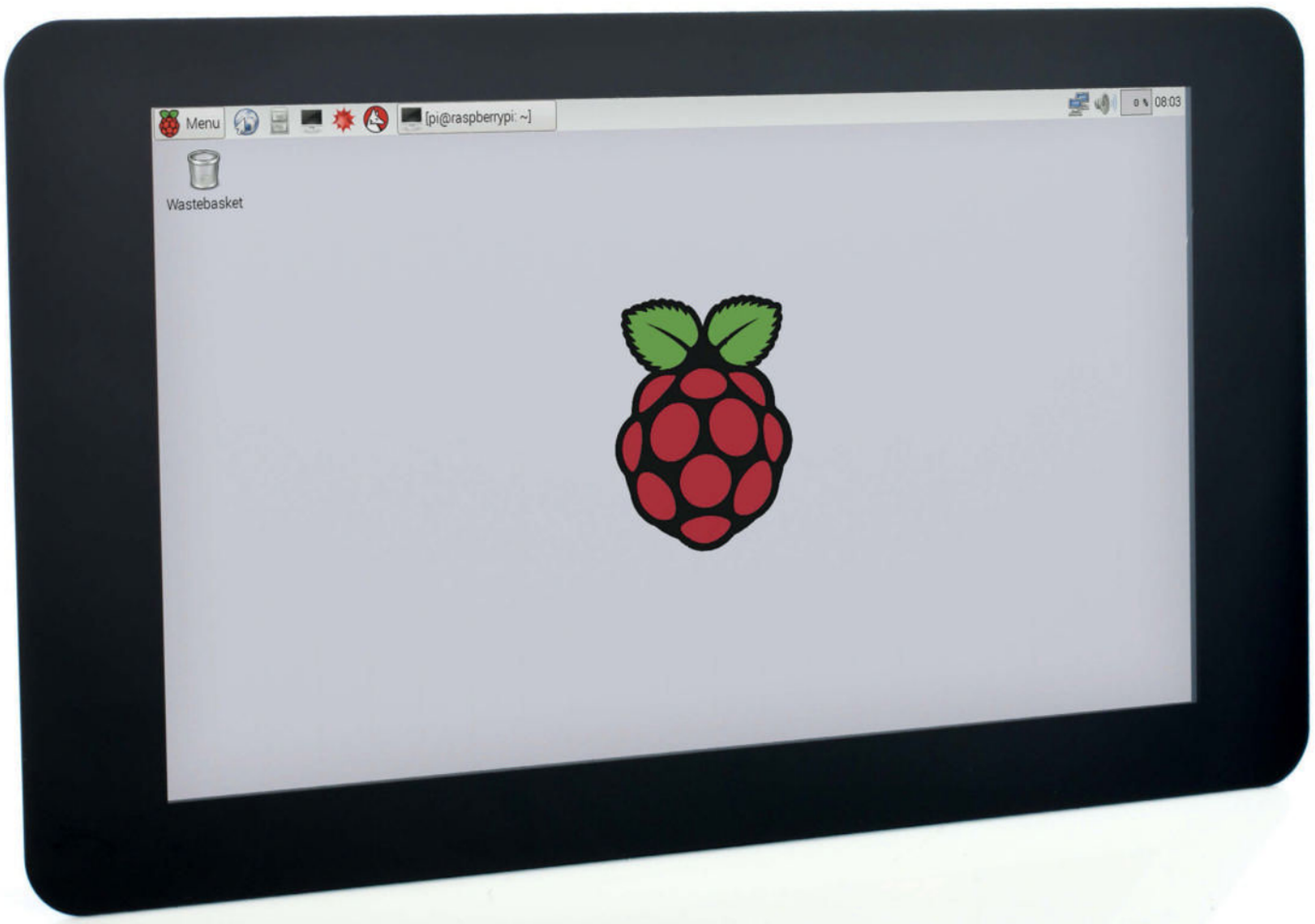
01: Order the parts

You can order the Raspberry Pi display through all of the normal channels. Shop around if you like, but take a look at CPC (cpc.farnell.com/SC13858), RS Components (**bit.ly/1OVz8vW**) and the Raspberry Pi Swag store (**bit.ly/1QGvp3E**), which all seem to have the best prices available at the moment. However, the stock levels are a bit short because there has been a huge demand for the display, so you may need to look around some of the smaller outlets to find somewhere with it still in stock.

02: Prepare your Pi

As with all Raspberry Pi projects, it is important to make sure your device has the most up-to-date version of any software you will be using. For this guide it is even more important as the drivers for the display have only recently been released (they don't even feature in the latest version of NOOBS). Open a terminal window and type:

```
sudo apt-get update
```



```
sudo apt-get upgrade
```

Once this has completed, shut down your RasPi using the command sudo shutdown –h now.

03: Unpack the screen

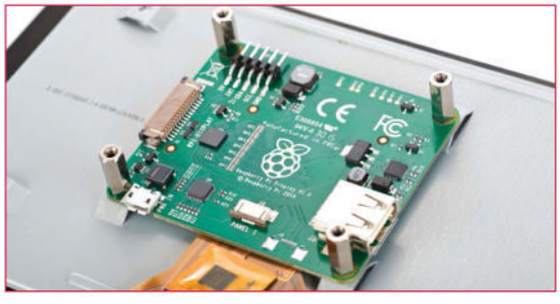
The screen comes as a kit of parts, as can be seen above. Included is the display itself, a driver board, a ribbon cable, mounting hardware and jumper cables. The packaging box should also include some pink anti-static foam and it is recommended to keep the flat part of this to use as a work surface, to avoid scratching the screen. For the time being you should also leave the protective film on the front of the screen, and possibly keep it on until it is installed in its final location.

04: Attach the driver board – part 1

Lay the display face-down on the anti-static foam, and you will see on the back that there is a large ribbon cable with a smaller ribbon cable attached. Grab the driver board, where on the back there should be a large connector. You should also lay this face-down on the anti-static foam.

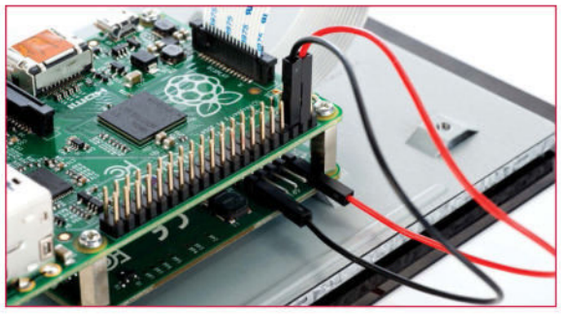
05: Attach the driver board – part 2

Carefully unclip the large connector on the driver board, insert the large ribbon cable and then close the clip again to secure the cable. Turn the driver board over onto the back of the display, where you should now see a small ribbon cable and a small connector on the top side of the board. Connect this smaller ribbon cable in the same way you did the large one.



06: Secure the driver board

Once the ribbon cables are securely fastened, the next step is to secure the driver board to the back of the display. Grab the mounting posts, align the four holes on the board with the holes on the display and then screw the driver board into place with the mounting posts. Be careful not to overtighten.

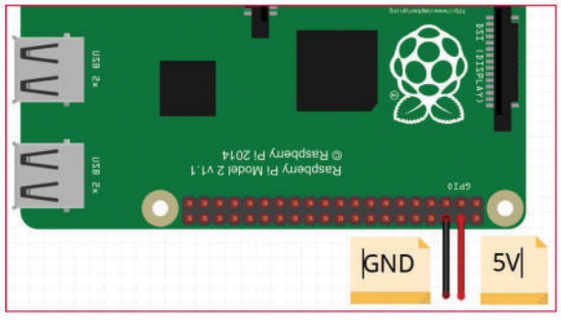


07: Attach the Pi – part 1

On the top of the driver board, on the opposite side to the USB connector, you should see another connector. Grab the white ribbon cable that came with the kit and connect it here with the blue mark facing the back of the screen. Next to this, you should see another connector with five pins. Plug the red jumper cable into the one marked '5V', then plug the black jumper cable into the one marked 'GND' (the colour of each cable doesn't matter, but this is the convention).

08: Attach the Pi – part 2

Place your Raspberry Pi on top of the mounting posts with the USB port on the same side as the USB ports on the display driver board. Screw the Raspberry Pi into place with the provided screws; once again, be careful not to overtighten. Then you can attach the other end of the white ribbon cable into the DSI connector on the Raspberry Pi, which is the one on the same side as the ribbon cable that is already attached. It should loop round easily and plug in.



09: Power your Pi

The display has been designed so that the driver board and Raspberry Pi can run from a single power supply. Take the red jumper cable (or whatever colour you used for 5V) that you plugged into the driver board in Step 7 and connect it to pin four on the Pi GPIO (second down, closest to edge of the RasPi board). Then take the black or GND jumper cable and connect it to pin six on the RasPi GPIO (third one down, closest to edge).

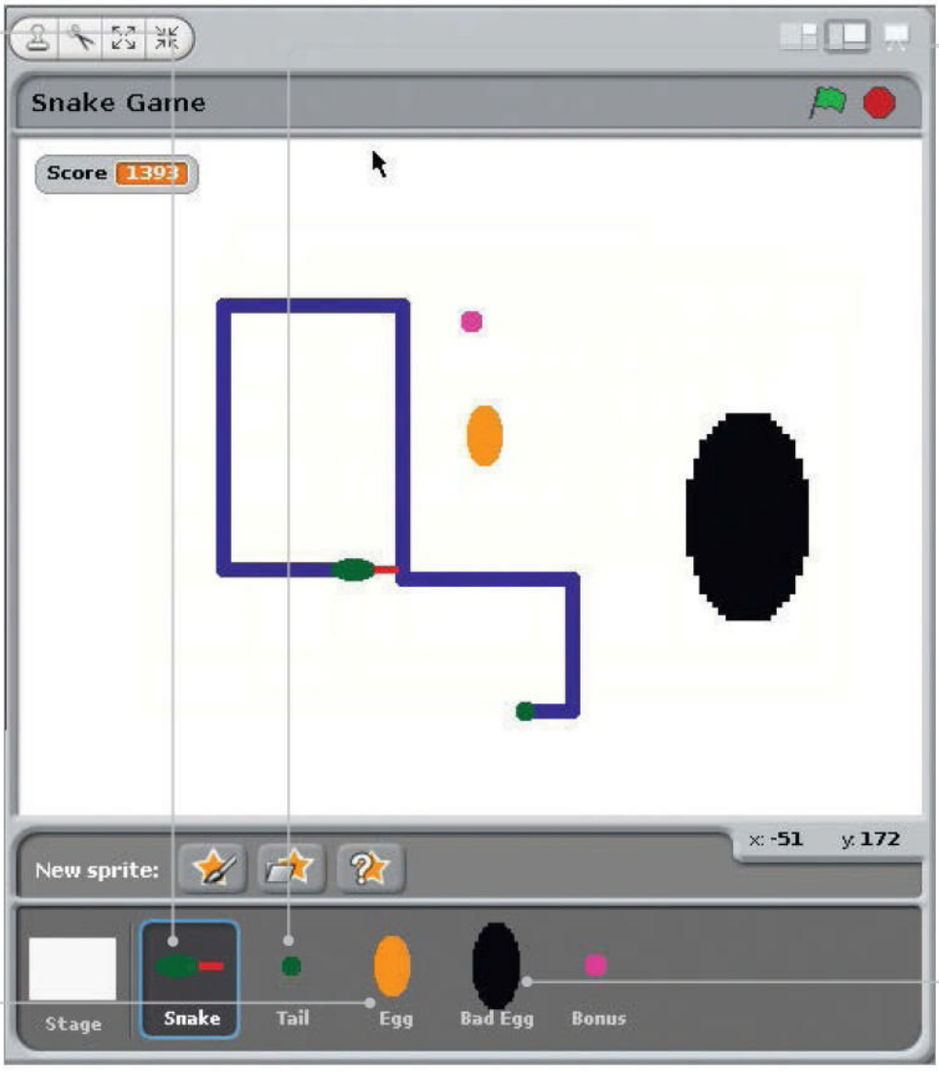
10: Boot your Pi

You should now be able to plug in your 2A micro-USB power supply into the connector on the display driver board, then the Raspberry Pi and display will both boot up. The touch functionality works through the white ribbon cable, so no other connections are necessary and it should work straight away!

Customise the display

Even though the display has only been available for a short period of time, a number of companies, such as Pimoroni and ModMyPi, have already created cases for the Raspberry Pi Display to enable you to customise and protect your new device. They are all reasonably priced at around £10, and come in a variety of colours and styles to suit your needs.

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.

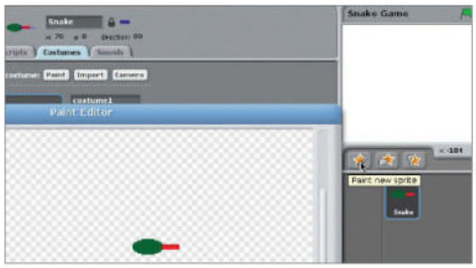
By following this tutorial you will learn to create your own simple sprite graphics, send and receive broadcast events, use a list variable to store data, play sound effects, generate random numbers and use sensing commands to detect when a sprite is touching something.

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
scratch.mit.edu/explore/?date=ever

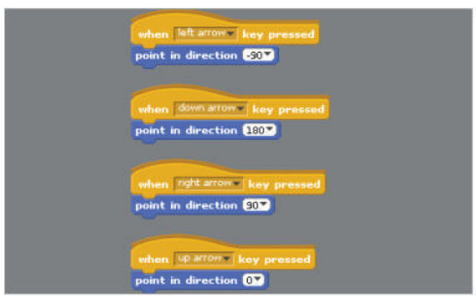


01: Paint the Snake sprite

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.

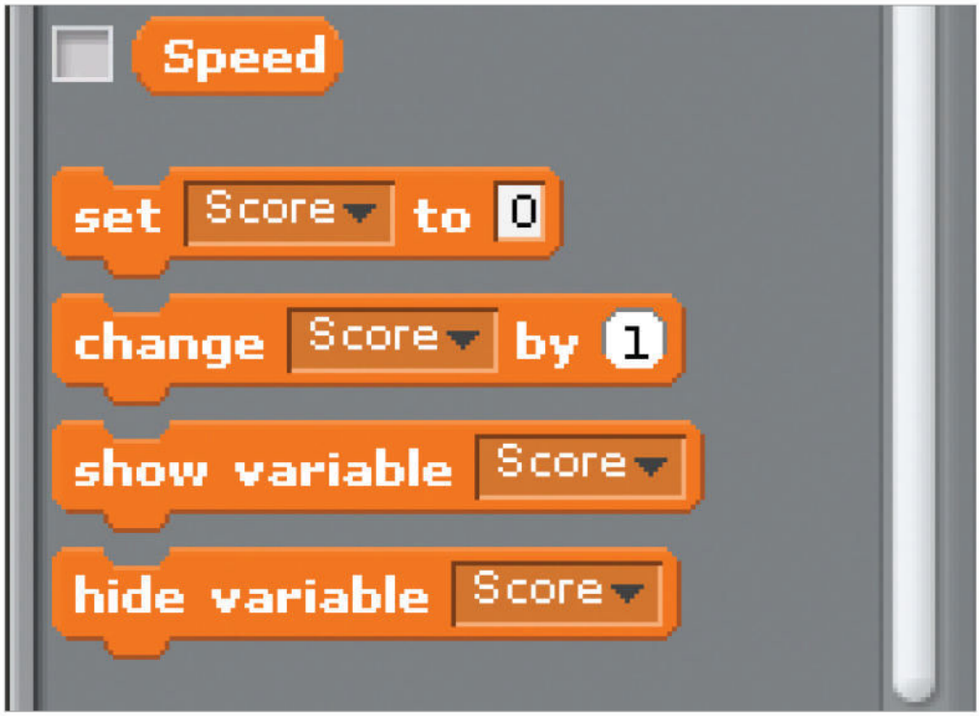


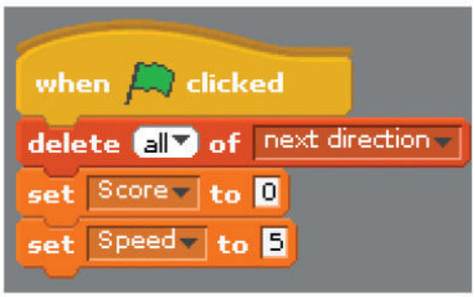
03: Respond to arrow keys

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

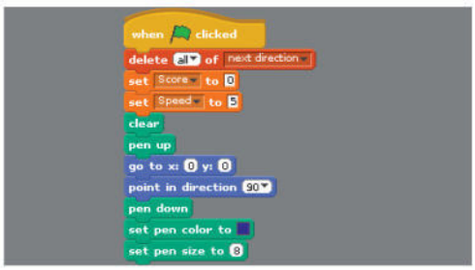
Click on the Variables palette. Make two variables, Score and Speed, that are visible to all sprites. Make a list called Next Direction which is visible to all sprites; it will store the sequence of directions that the head takes. Only have the Score variable checked so it appears on the stage.





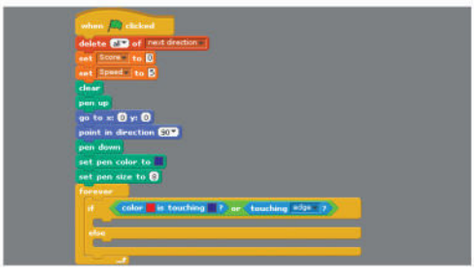
05: Initialise Snake variables

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.



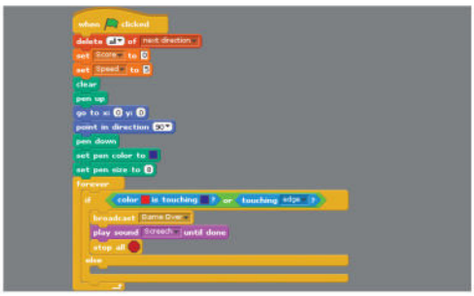
06: Draw Snake body

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

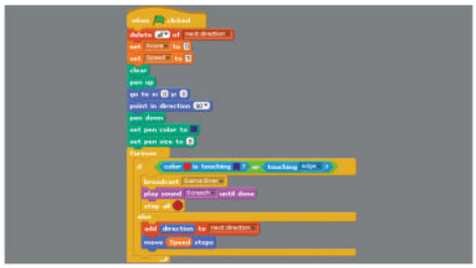
Use a 'forever' command with an 'if-else' command nested inside. We have a collision if the red tongue is touching the blue body (the head is always touching the body) or the Snake sprite is touching the edge. Use the Eyedropper tool to select colours within Scratch.



08: Handle Game Over

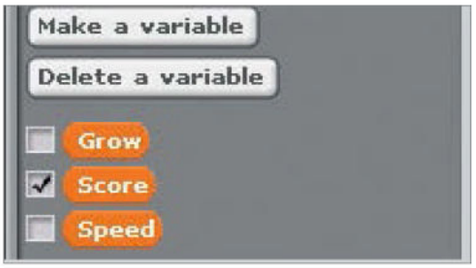
When a collision has been detected, broadcast a Game Over event (you'll need to create a new event) to the other sprites so they can also react. Also, play the Screech sound effect and stop all scripts so that the snake freezes in its current position at the end of the game.

Programming



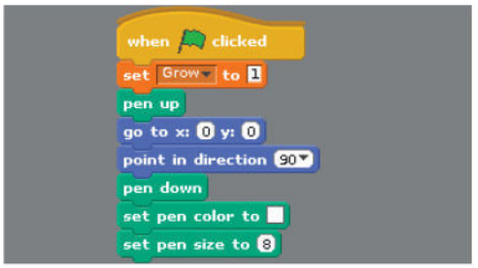
09: Handle movement

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.



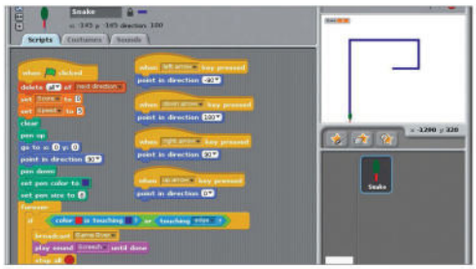
12: Make a Grow variable

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.



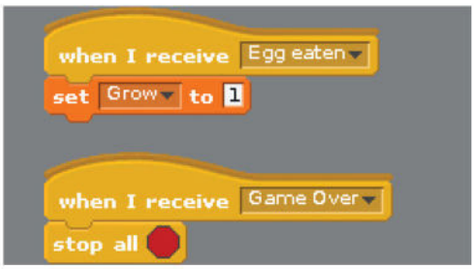
14: Initialise the tail

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.



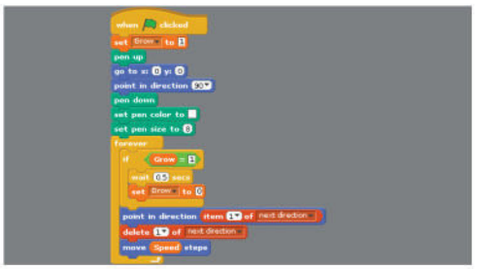
10: Try out the snake

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"

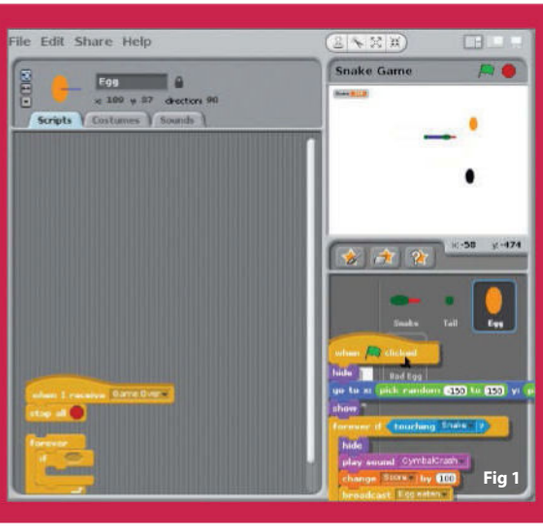


Fig 1

■ 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

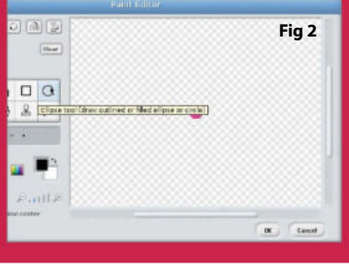


Fig 2

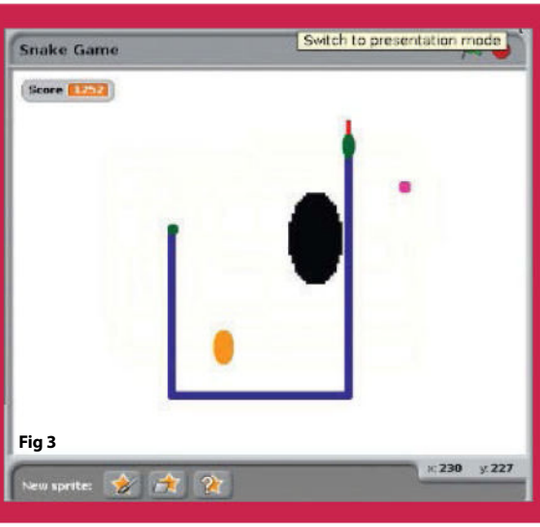
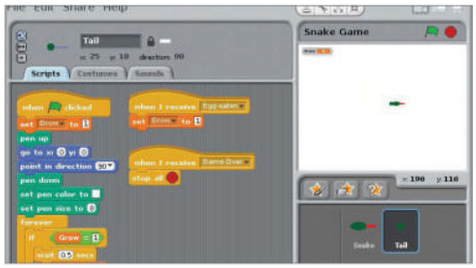


Fig 3



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

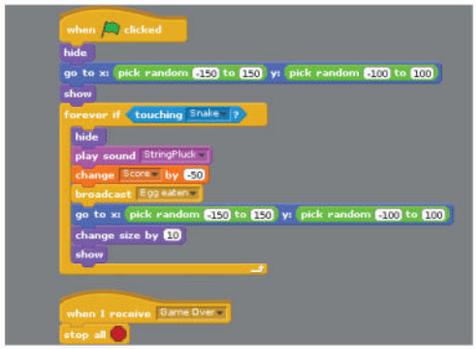
Click the New Sprite: Paintbrush icon to paint a new sprite. In the Paint Editor, draw a small yellow ellipse. Name the sprite Egg. The Egg will appear randomly on the stage and cause the snake to grow and increase its score.

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.

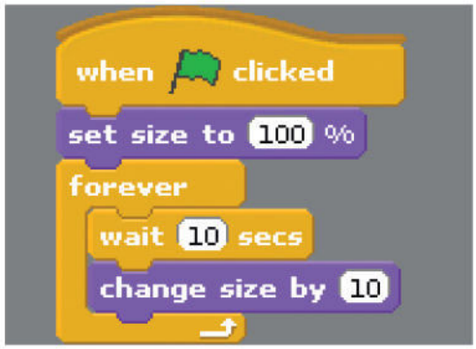
19: Add Egg scripts

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.



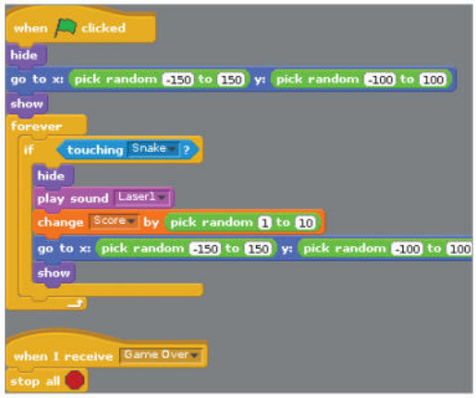
20: Make Bad Egg

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.

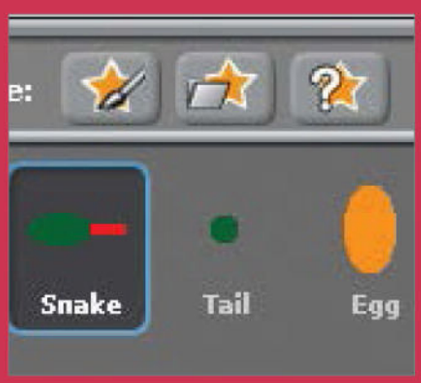
Sprites can have multiple costumes to change their appearance. We only used one costume per sprite, but you could extend the project so that the snake head changes appearance briefly when it eats a Bad Egg.

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.



Get started with Python

Always wanted to have a go at programming? No more excuses because Python is the perfect way to get started!

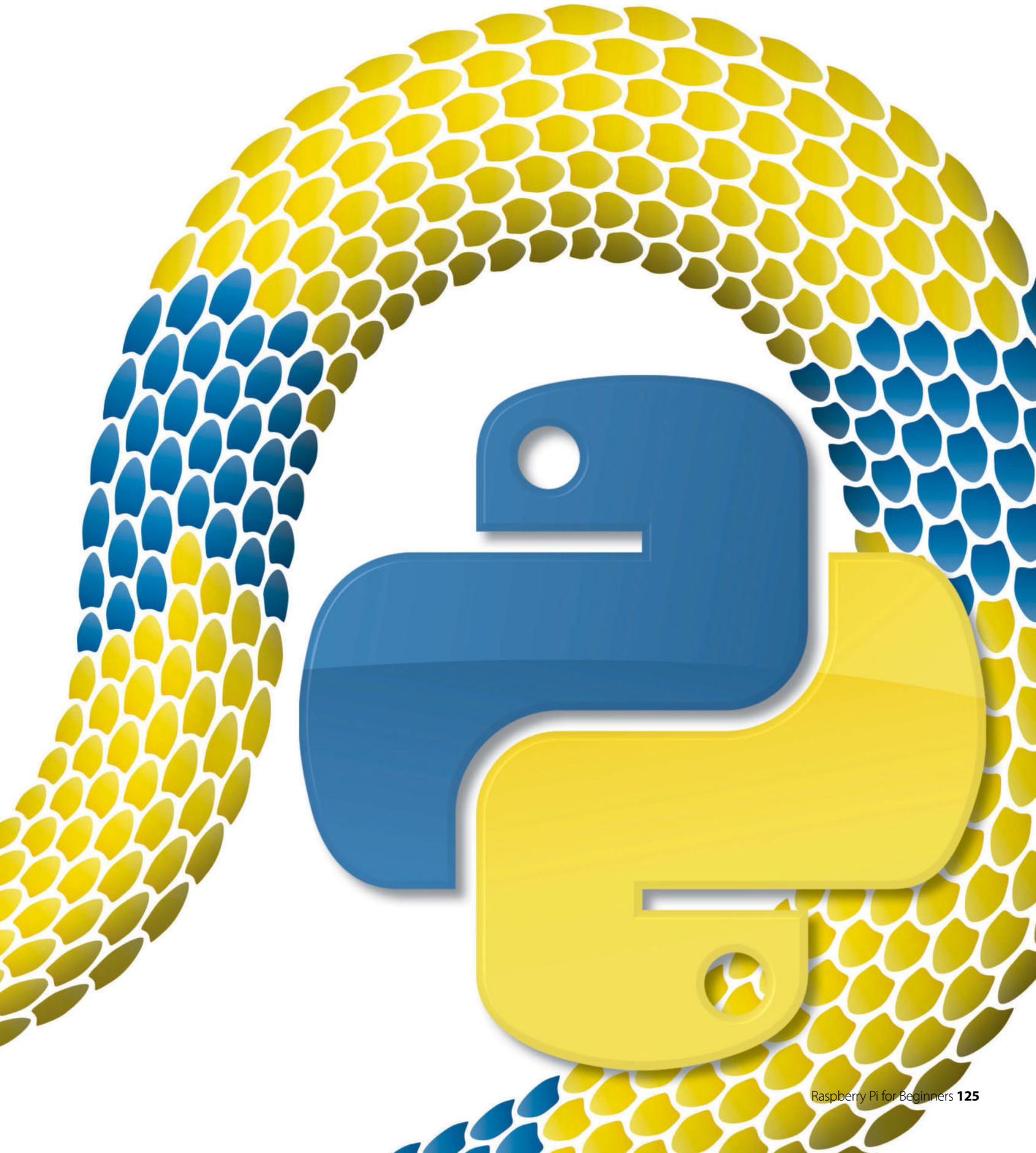
Python is a great programming language for both beginners and experts. It is designed with code readability in mind, making it an excellent choice for beginners who are still getting used to various programming concepts.

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.



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 just made to the file in nano using the key combination `Ctrl+O`, followed by Enter. Use `Ctrl+X` to exit nano.

```
#!/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
```

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.

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 aren't able to 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 structure

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 uses a local copy, so changes in 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

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")`

```
# 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 into smaller chunks"

```
#!/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 important rule.

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 section of 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.

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

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.

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.

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."
```

```
*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
```

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
>>>
```

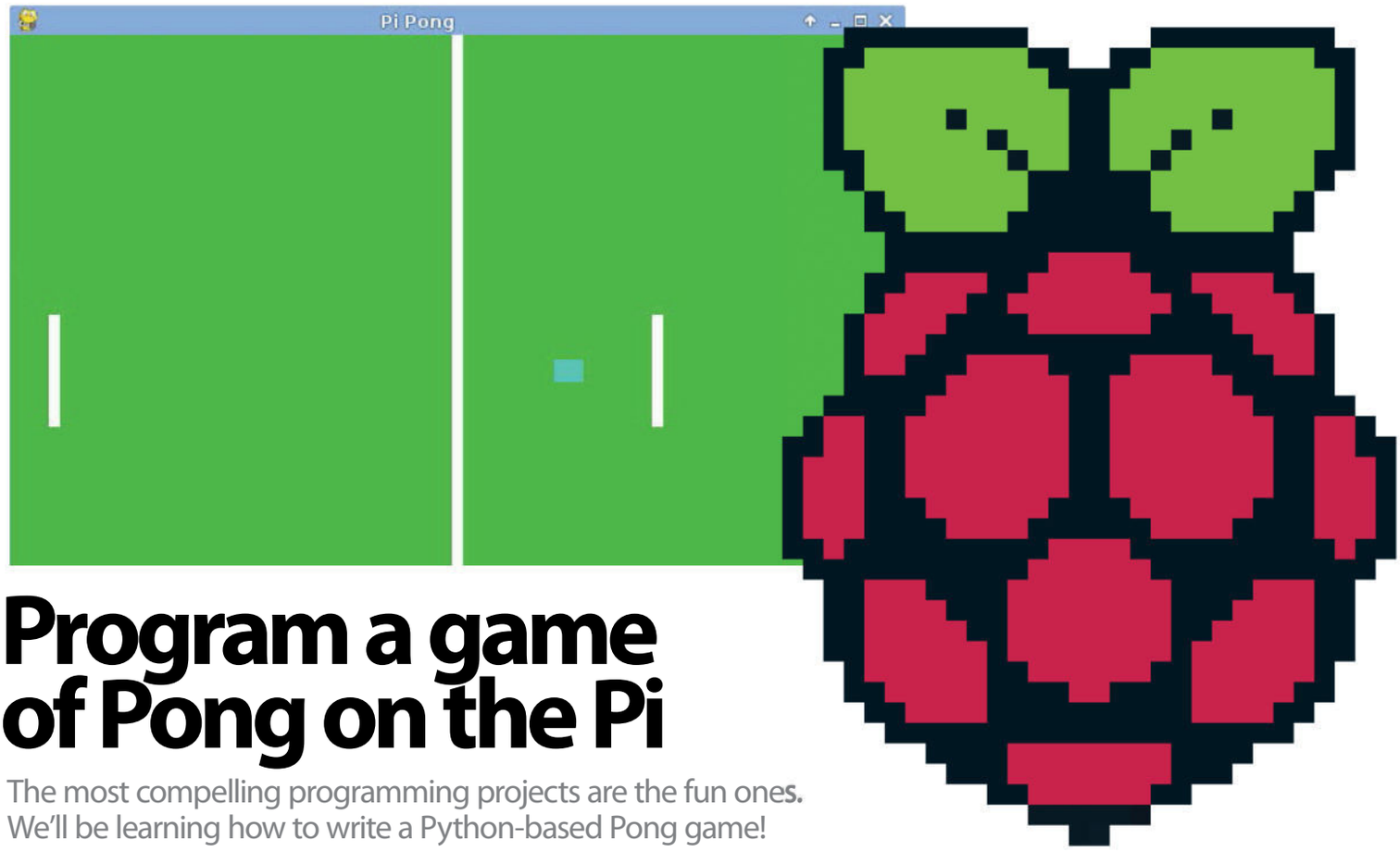
The code in action

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.



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, 480)
```

■ 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

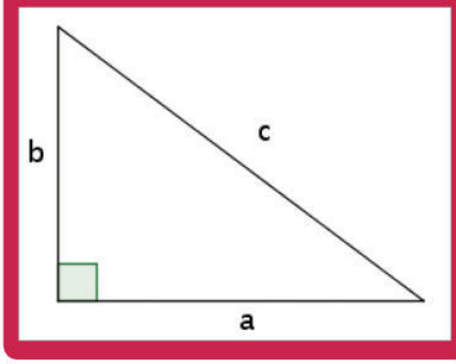
The most complicated part of this game is the ball class, because it involves collisions and then bouncing off things. Don't worry, though; the physics of the ball isn't 100 per cent accurate, so it is much easier to understand. This is just supposed to be a very basic implementation; not a polished 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, because adding a negative becomes subtraction.

```
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

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.

02: And finally

```
# Check for bat collisions
self.ball.batCollisionTest(self.player1Bat)
self.ball.batCollisionTest(self.player2Bat)
```

■ Fig 9

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.

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/.

"We'll need to reset the ball every time that it hits a wall behind the bat"

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. 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.

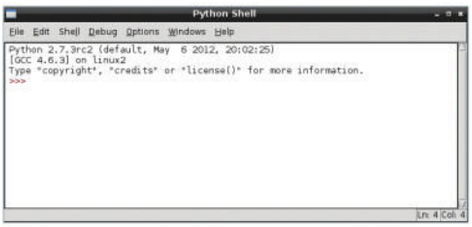
Unfortunately, Geany is missing from the image, but we will walk you through how to use IDLE as it is just as good.

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: Get 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: Introduce 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: Open 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: Add 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

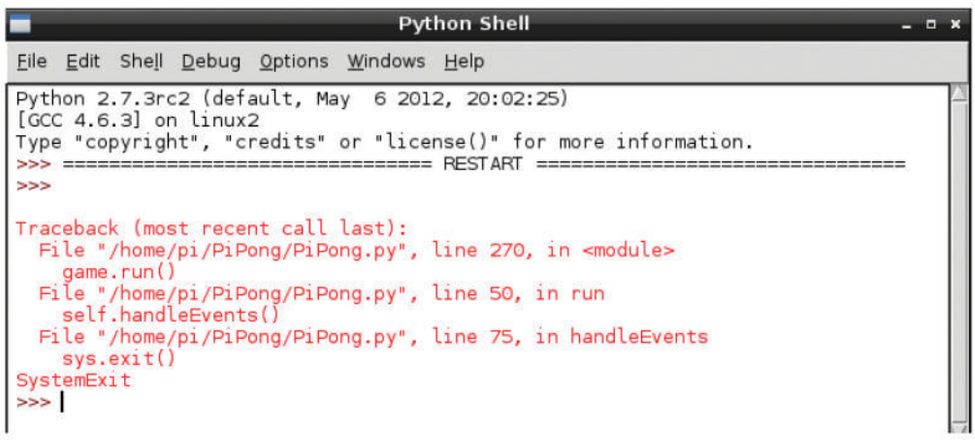
Drawing the menu is easy. We already have it, we just need to draw it to the screen using the blit function (blit stands for block image transfer) at the location of the rectangle that we created before. We'll also need to handle the 1 and 2 key events that will eventually set which type of game to play. 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: Select 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: Choose 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: Play 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: Play 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: Give it a go

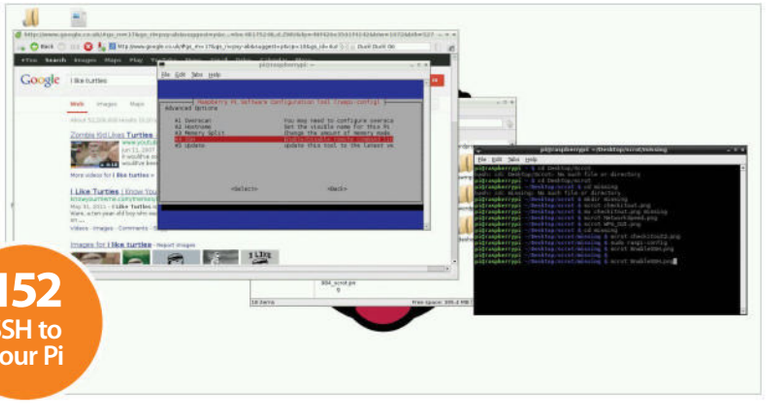
Press F5 in order to run your game and then select 1 or 2 from the menu in order to decide which type of player you would like to play against. You should find that if you select 1, the CPU-controlled bat will move up and down the screen to be in line with the ball. You should also hear sounds. Now everything 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

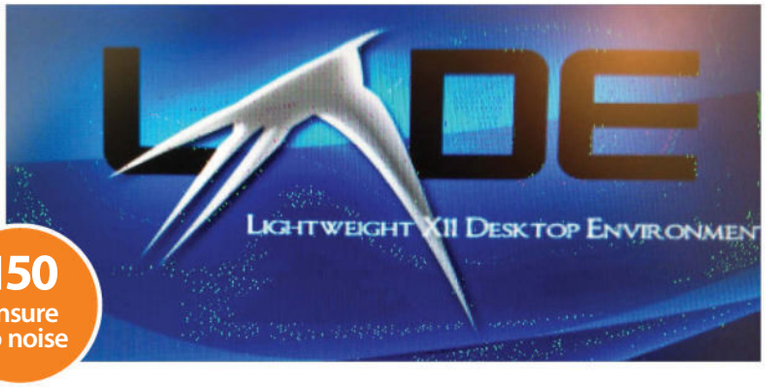
Troubleshooting

Overcome common Raspberry Pi problems

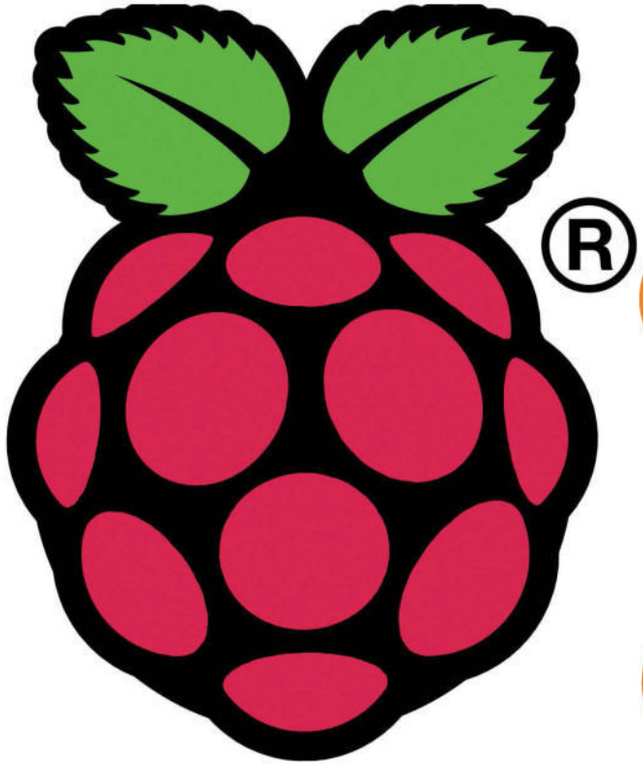
"No matter how simple a device, problems will occur"



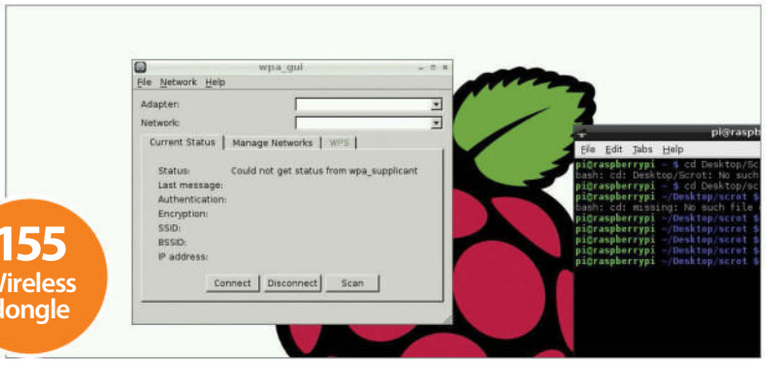
152 SSH to your Pi



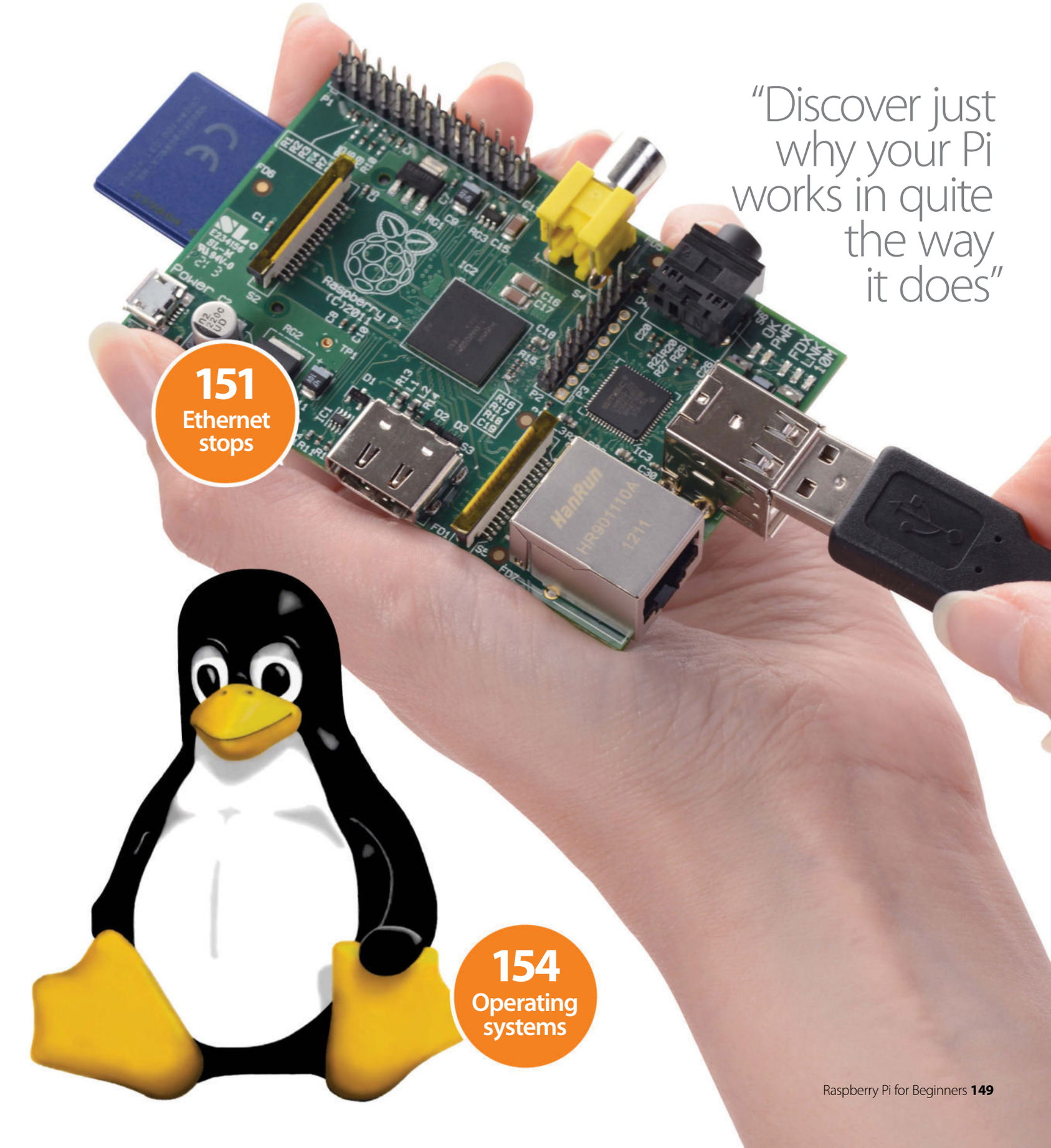
150 Ensure no noise



152 Startx command



155 Wireless dongle


"Discover just why your Pi works in quite the way it does"
151
Ethernet stops
154
Operating systems

Troubleshooting: hardware

A list of common Raspberry Pi hardware problems and what to do when they occur

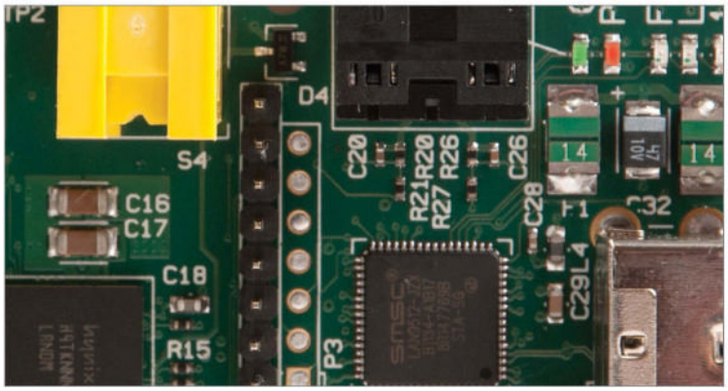
We've all been there. We've come back to something, and for whatever reason, the thing we're trying to do just won't work! We've plugged everything back in where we thought it came from, but there's nothing. So, what do we do when it comes to this? Usually there are a few different solutions to any one 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

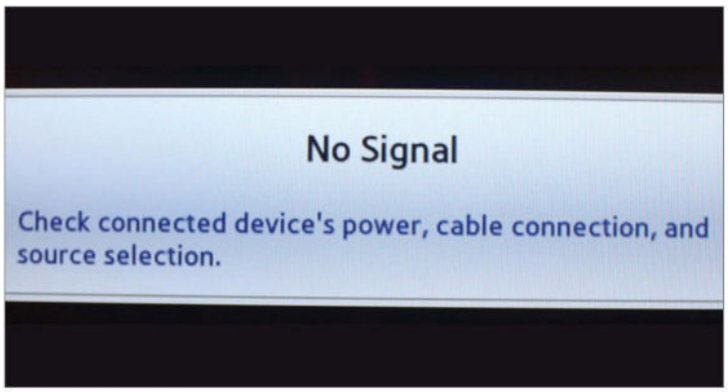
Solution: This is more common than you might think. It's likely down to either a blank SD card, or a lack of correct data on the SD card causing a boot failure. First, check your SD card is fully inserted. With the latest Pi firmware, if you have:

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!

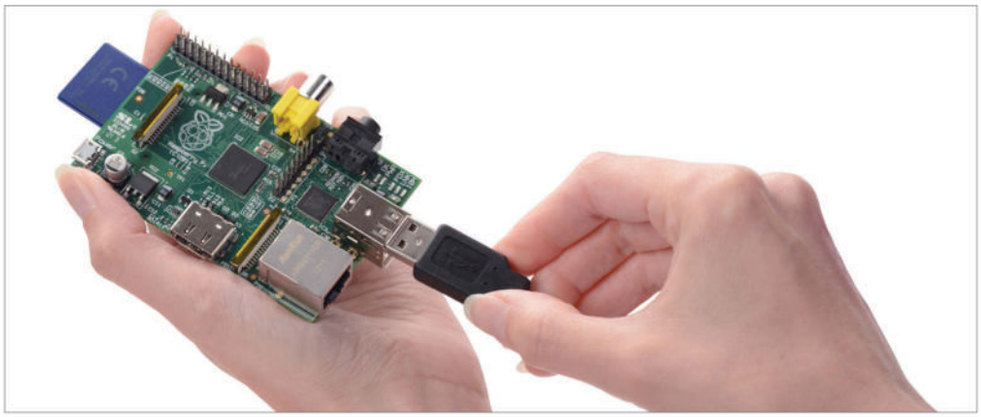
If setting it to default clock speeds helps, increase again gradually until you find something stable.

Problem 4: The HDMI image from the Pi has noise

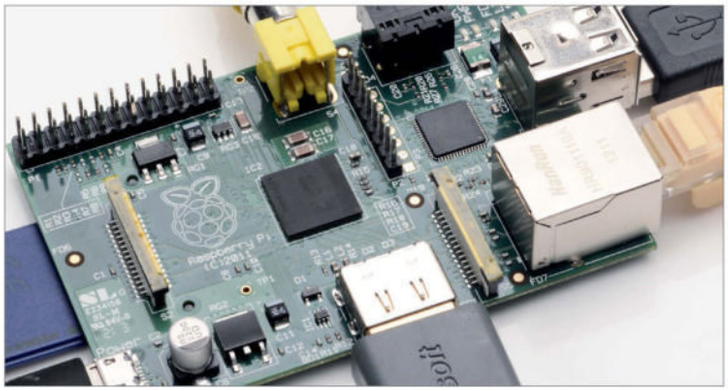
Solution: If the picture from your Pi is noisy (screen is shaking, dots or green lines, etc, moving around) then it's probably a dirty connection. Make sure the HDMI or composite cable is connected properly and free from dust and other contaminants. If you still have a problem, try another HDMI cable.

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

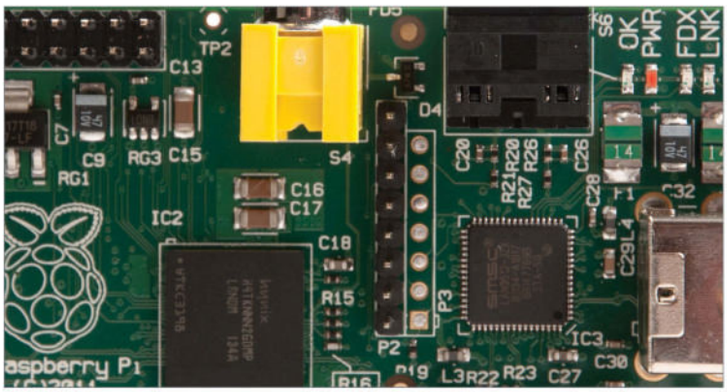
Solution: The first thing that you should do in this situation is check your power supply is working! It has been known for some power supplies to provide an inadequate source of power. First, try changing out your cable (it may just be a poor-quality cable).

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.

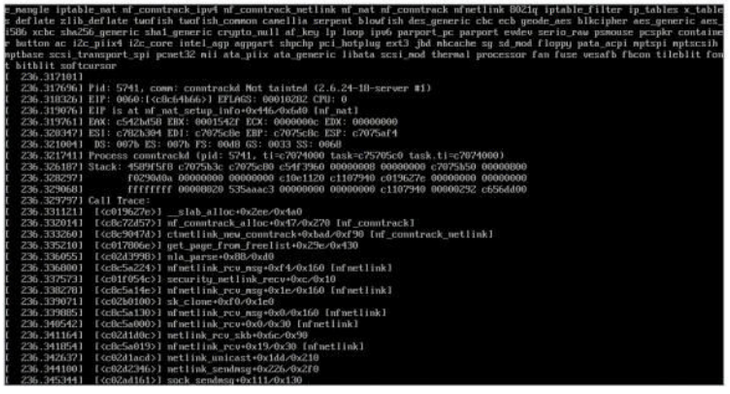
Some problems will seem to have little to no correlation with their answers, but those scratch-your-head moments soon go when you realise that actually the thing is working and doing exactly what you wanted it to again. There's a lot you can do from the command line to help with any of this stuff, but 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.

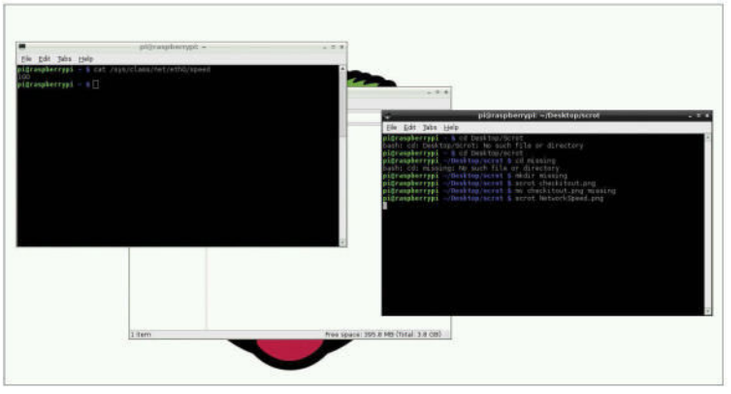
As always, be sure to keep frequent backups of your most precious data, though.



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.

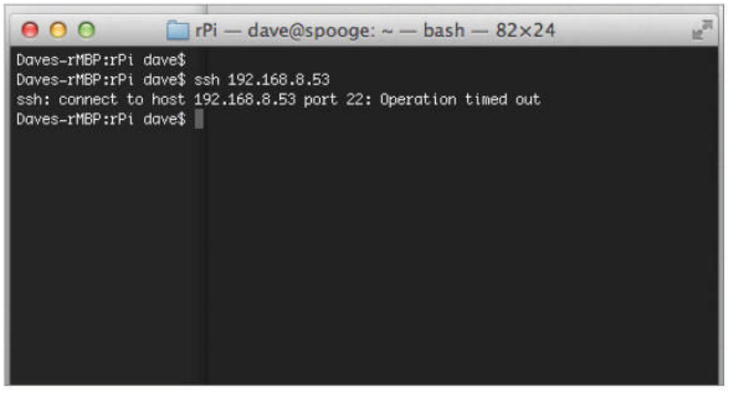
If you're still getting the messages, it's likely that your flash of the SD card was unsuccessful. Reflash the card and try again. Make sure you use admin privileges to do it!



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

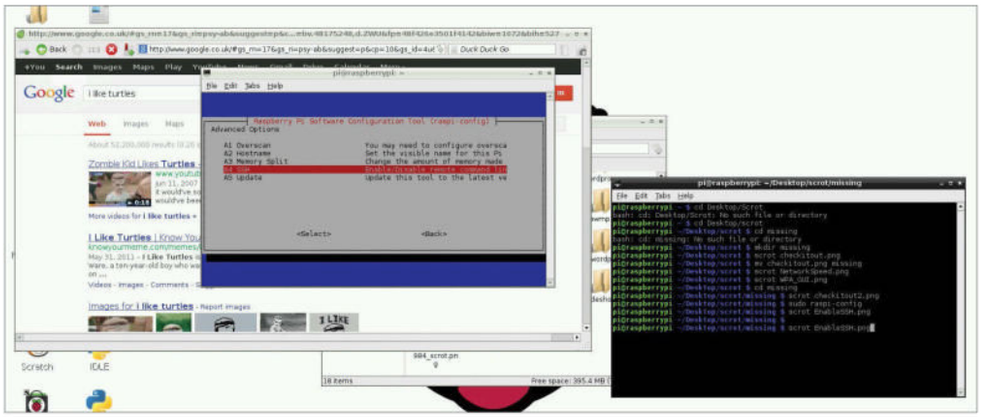
Solution: If you're receiving a connection time-out error when you try to SSH to your Pi from another machine, it probably means that SSH access to it is disabled.

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.



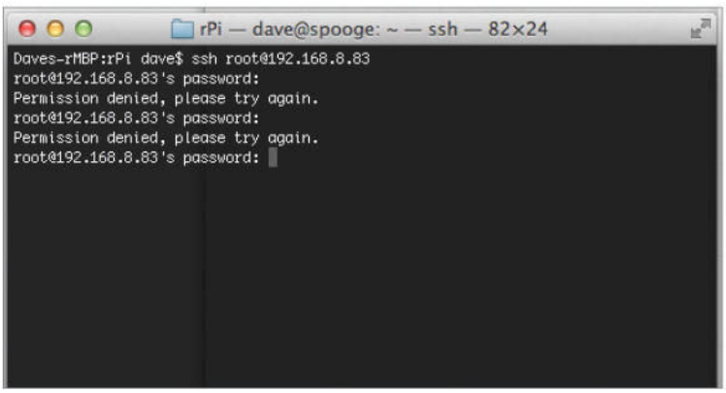
"Open up the raspi-config tool, then use the option 'ssh' to enable SSH. You should now be able to connect to your Pi"

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.

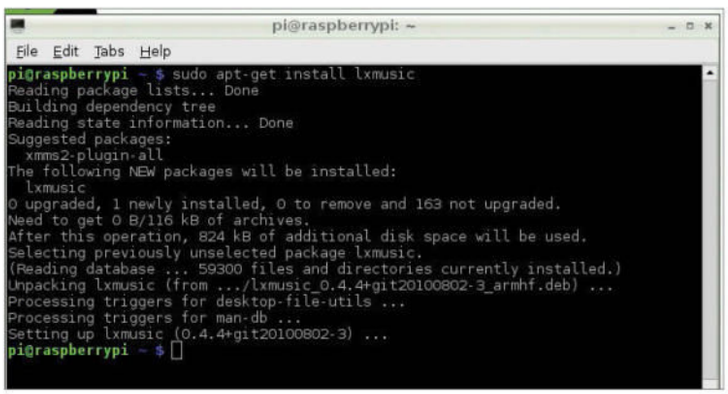


Problem 6: I don't know the root password

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.

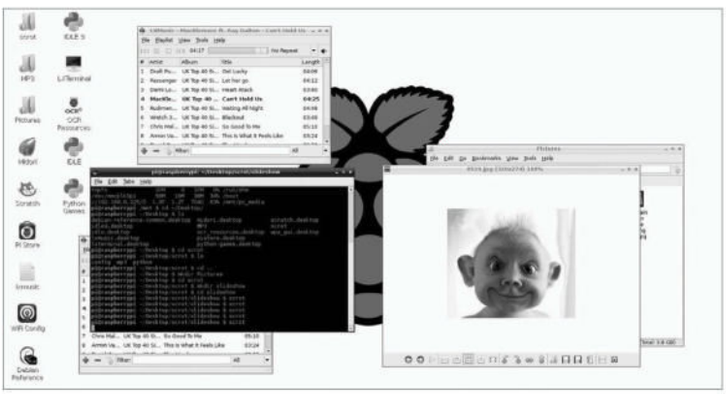


Problem 7: I can't install new software with apt-get

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
```

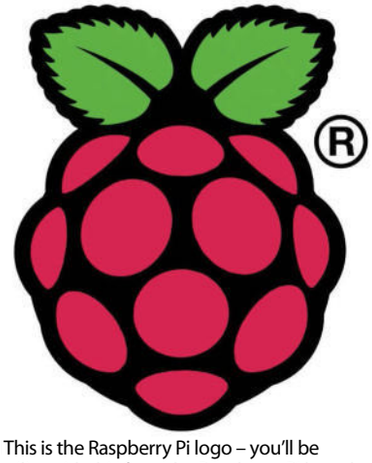
Where x is: 0 – NTSC; 1 – Japanese NTSC
2 – PAL; 3 – Brazilian PAL

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.

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.

There are currently three models on sale; the original Model A and Model B were replaced with Model A+ and B+, while the Raspberry Pi 2 Model B is the newest addition to their lineup. The price variation between each is minimal and relative to their specifications.

What operating systems can I run on my Pi?

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.

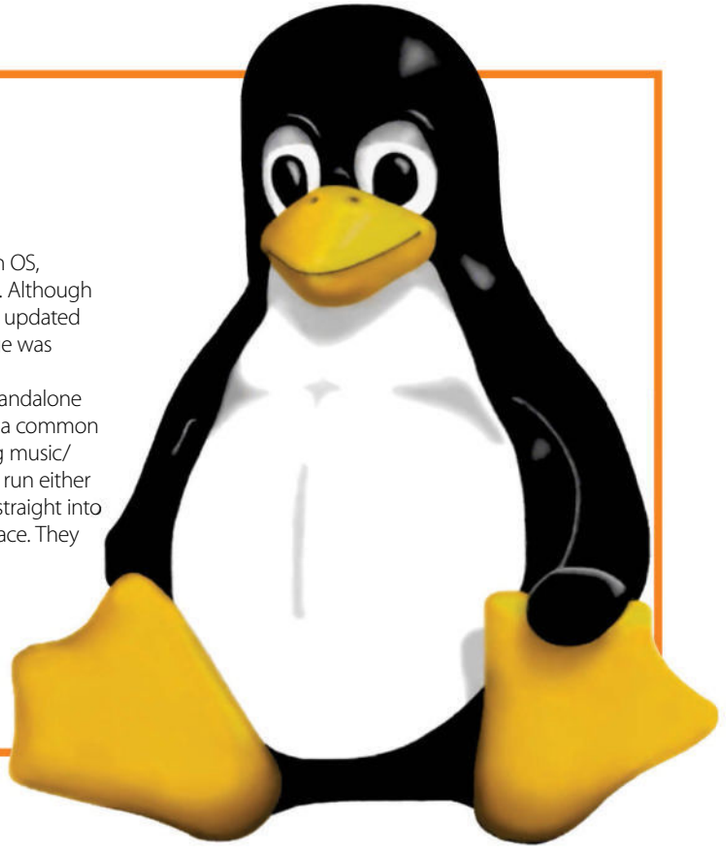
Debian (Squeeze) was the default distro used on the early versions of the Raspberry Pi board. It was the first fully functional OS with a desktop, browser and set of development tools.

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. Although stable releases are infrequent, it was updated recently and the latest SD card image was uploaded on in February 2015.

OpenELEC and OSMC 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.



■ There are a lot of differently purposed operating systems available for the Raspberry Pi. These are just a handful!

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.

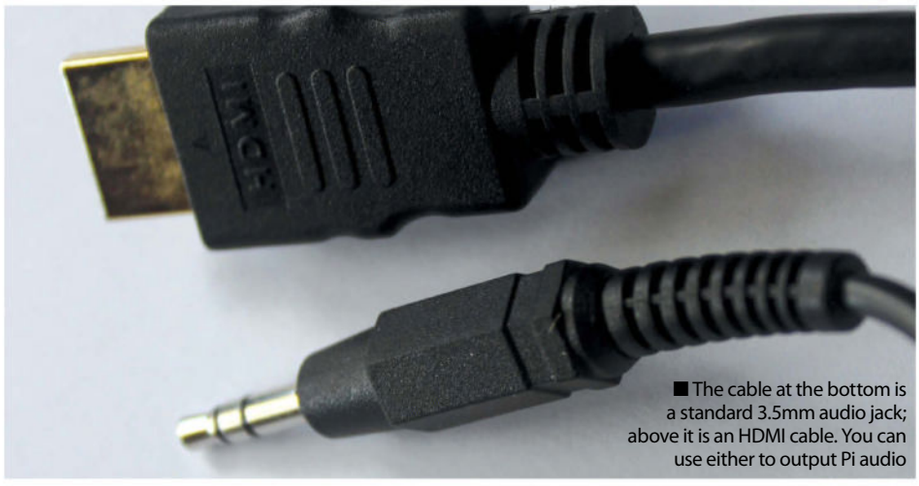
As for choosing an SD card, the best way to select is just to not go for the absolute cheapest one. Go with something of a recognised brand, as the Pi will be slower with a card of a lower class.

There is a list of officially tested cards on the eLinux site – here, you'll be able to look for cards of a specific size and/or class:

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 required for NOOBS installation is 8GB and the minimum for a standard image (e.g. Raspbian) installation is 4GB.

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 in that case the drive would be bigger than the Pi – and as the Pi has limited power available, it requires powering separately.



■ 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 audio jack, but this is also what you'd use to plug in your headphones.

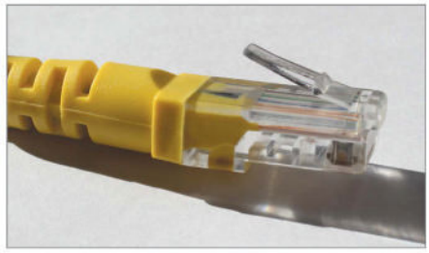
At a slightly higher level, rather than using the analogue audio jack you could use the HDMI port – which means that the audio goes out digitally. As well as giving a higher quality of audio, it also means that you can run this into your TV or other HDMI advice, then out through your TVs optical connection 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>
```

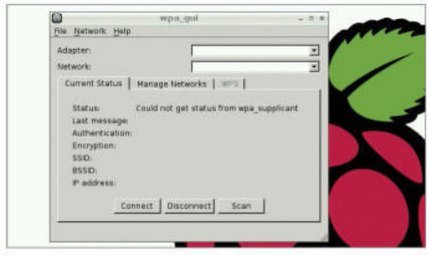
Where n is:
0 – auto
1 – audio jack
2 – HDMI

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. If using Raspbian, use the 'WiFi config' tool on the desktop. Select your adaptor and network, then type in your network key. You'll be ready to go, so long as your dongle is supported by the Pi, such as the official Wi-Fi dongle offered by the RasPi Foundation.

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.

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.

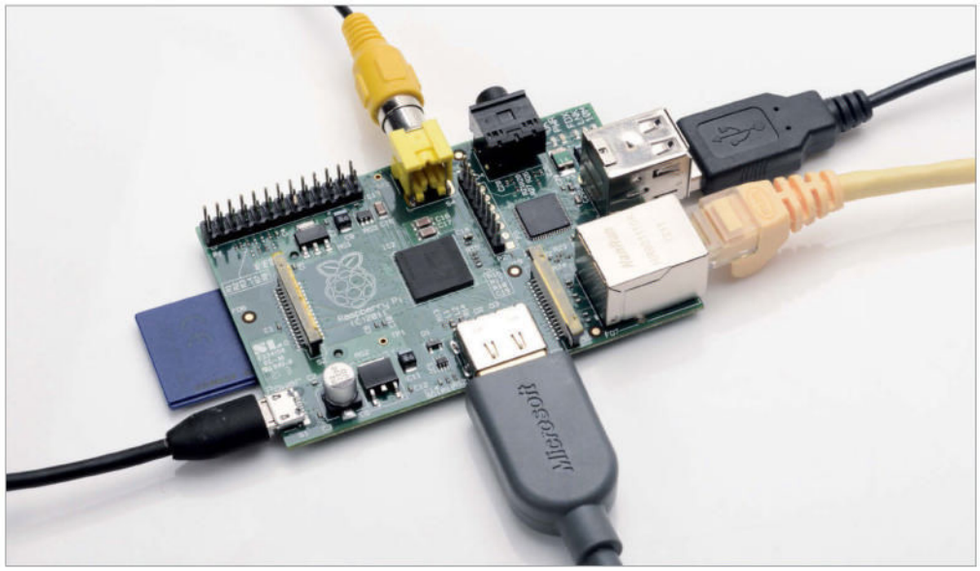
Arduino
A lot of people like to use their Raspberry Pi in conjunction with an Arduino. It's a board that allows you to easily power motors and lights and other physical objects that can make use of computers

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.

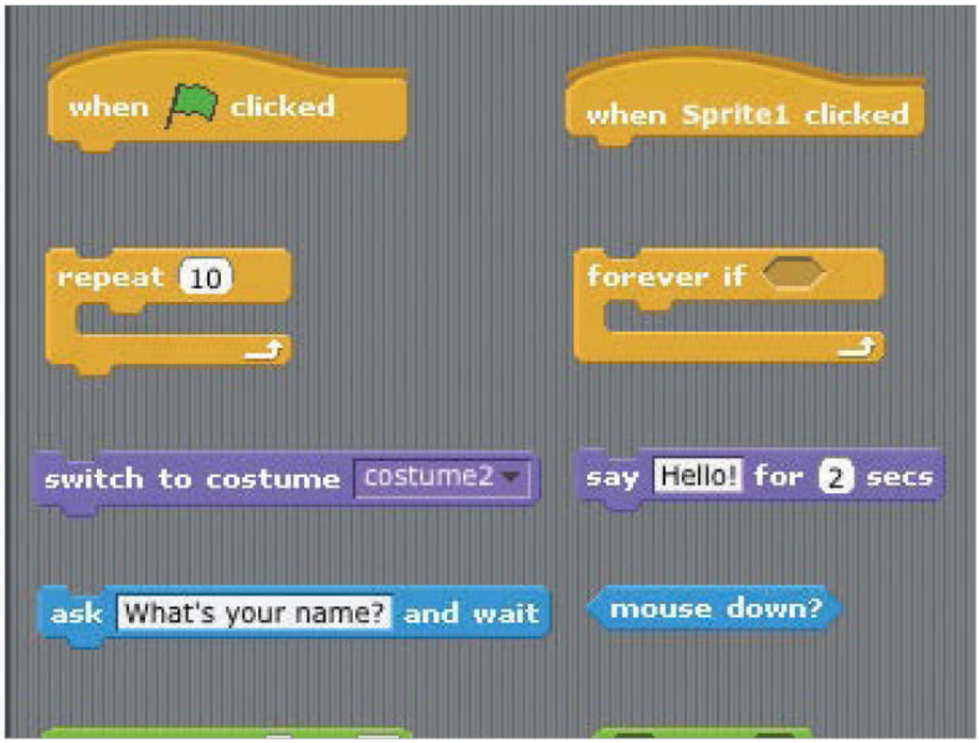
External hard drive
SD cards don't have a lot of space, so the cheapest and best way to add more storage to your Raspberry Pi is to have a USB-powered hard drive. This is great if you plan to keep a lot of files on the Pi

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.

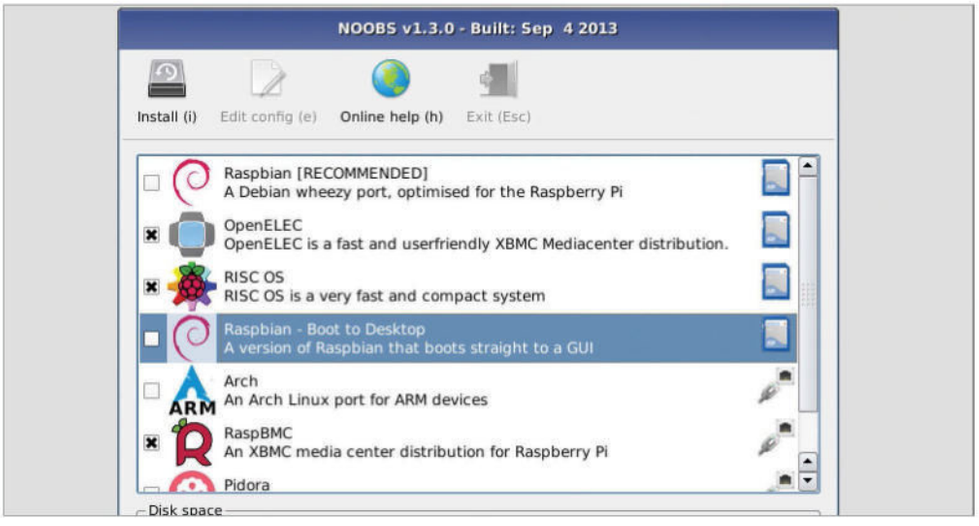
RaspBMC
This is a Raspberry Pi specific media centre operating system, turning your Raspberry Pi into an all-in-one entertainment centre that can be hidden away behind your TV, while making watching or streaming files easy.

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

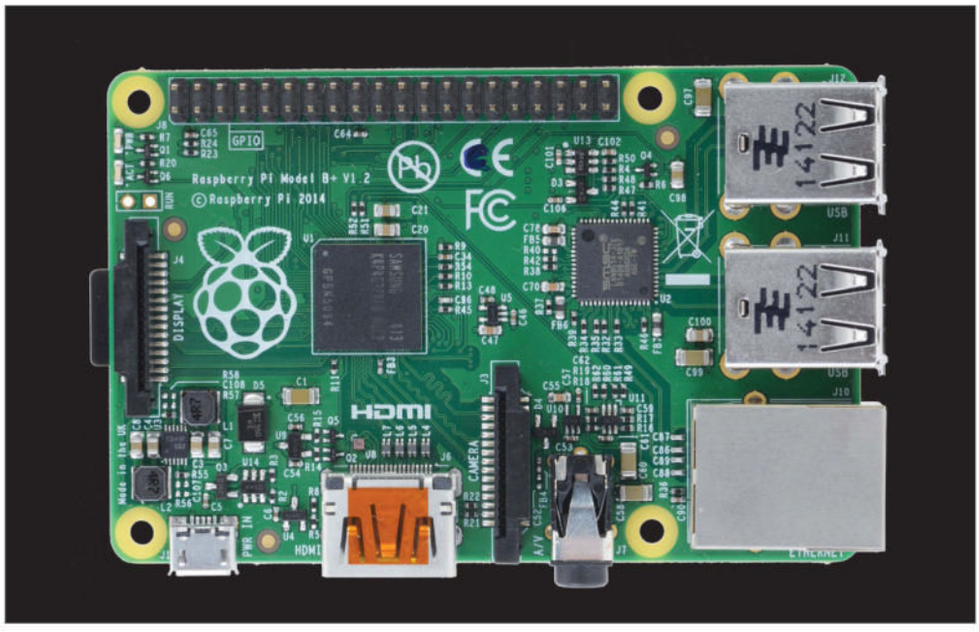
SD card
SD is short for 'Secure Digital' – it's a card used for storing data on. Your Raspberry Pi uses one as its boot device – effectively as a hard drive. You've likely seen one before in your camera.

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'.

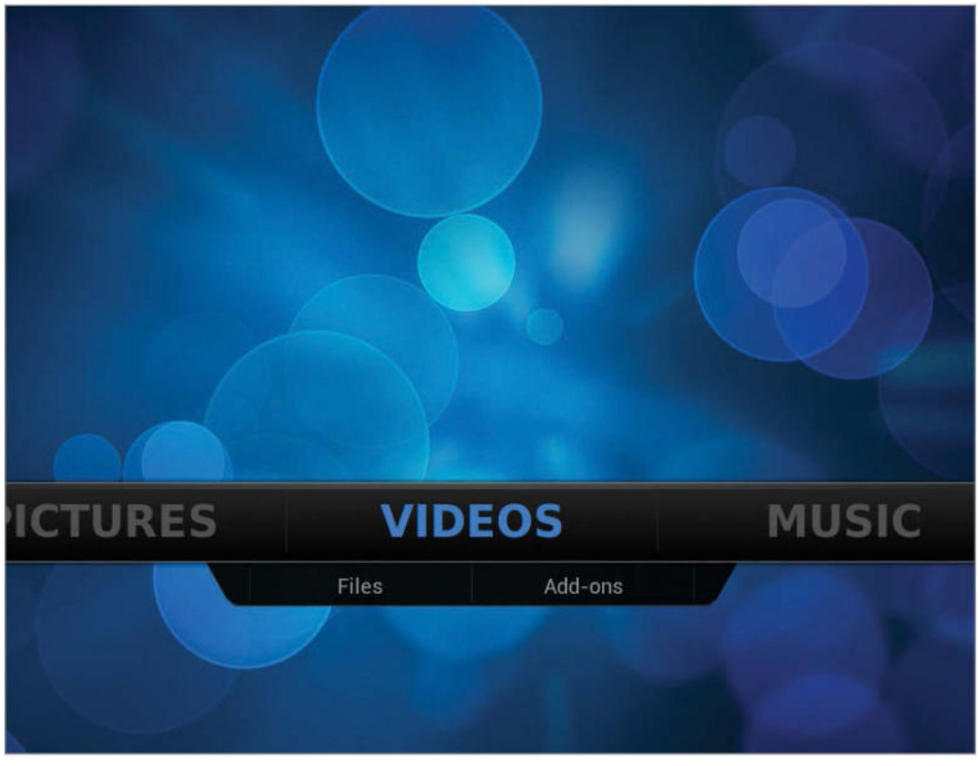
Slideshow
A slideshow is the idea of taking lots of different pictures and displaying them one after the other in an automated way. This means that rather than having to sit by your computer clicking through photos, it'll do it for you.

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

Plus free online resources'

subscribers to...

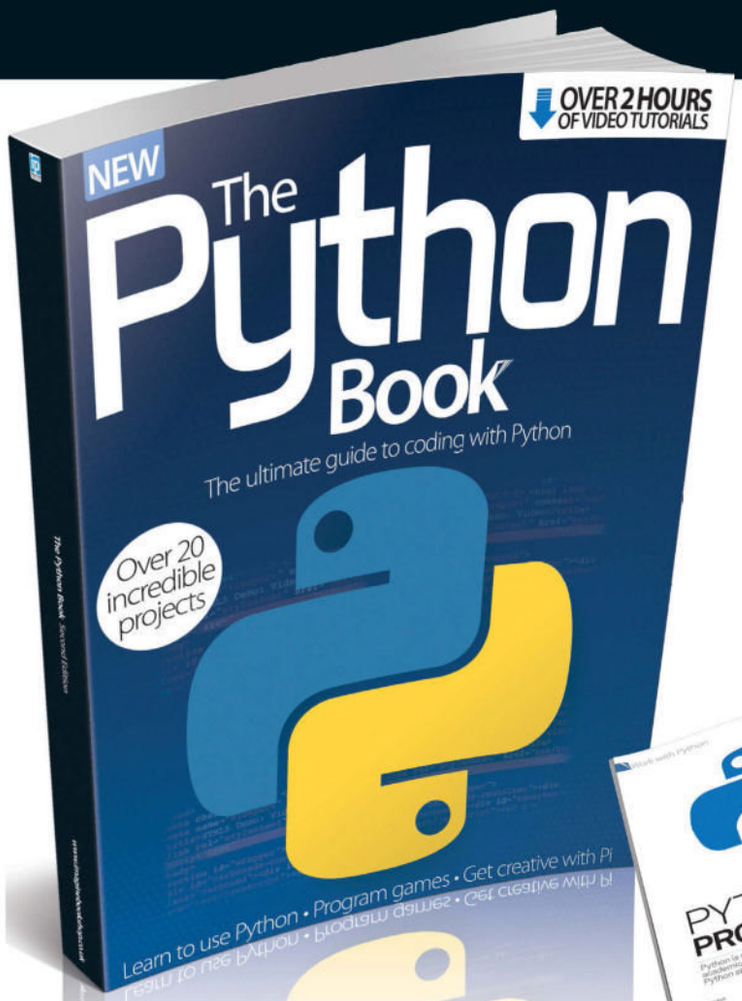
LinuxUser & Developer™

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 ZGGZINE

Or telephone UK 0844 249 0282+ Overseas +44 (0) 1795 418 661

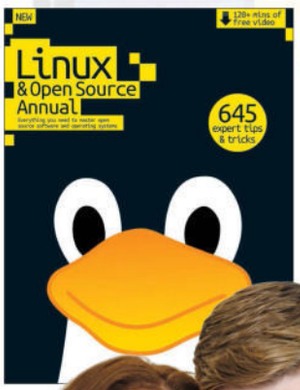
+Calls will cost 7p per minute plus your telephone company's access charge

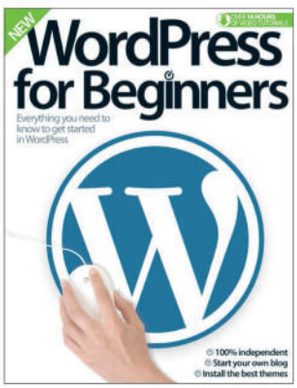


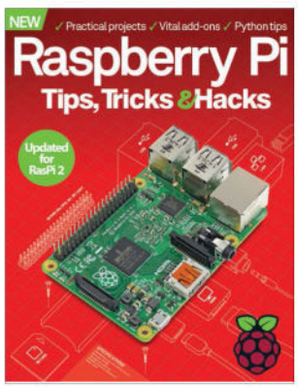
The Python Book

Discover this exciting and versatile programming language with the new edition of The Python Book. You'll find a complete guide for new programmers, great projects designed to build your knowledge and tips on how to use Python with the Raspberry Pi – everything you need to master Python.

Also available…







A world of content at your fingertips

Whether you love gaming, history, animals, photography, Photoshop, sci-fi or anything in between, every magazine and bookazine from Imagine Publishing is packed with expert advice and fascinating facts.

BUY YOUR COPY TODAY

Print edition available at www.imagineshop.co.uk

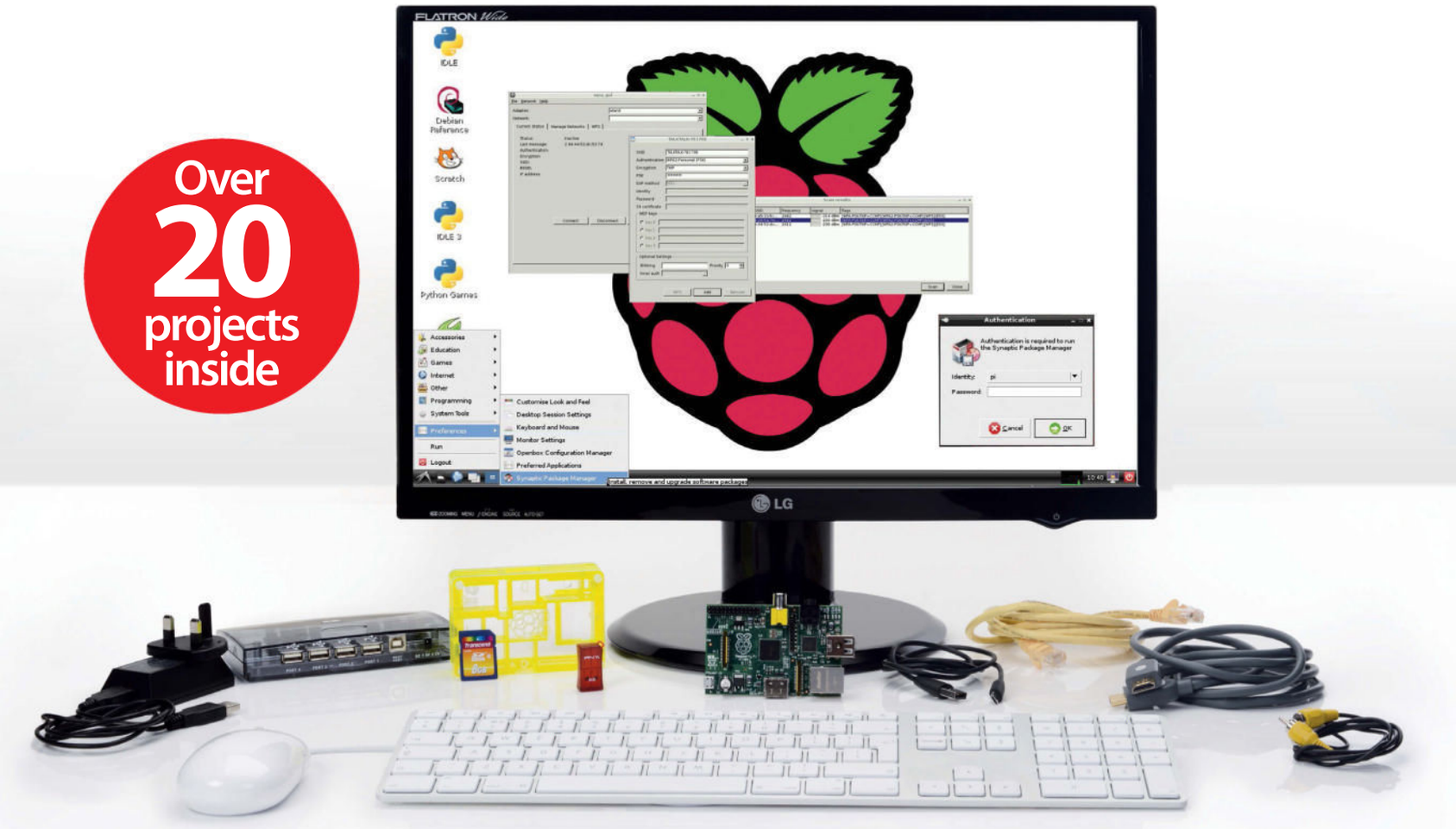
Digital edition available at www.greatdigitalmags.com

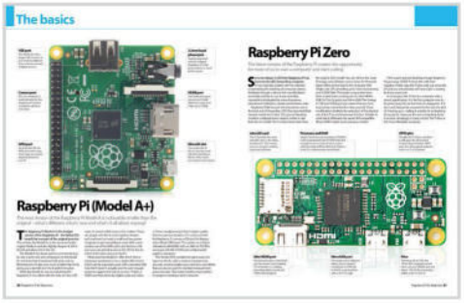
Print Tablet Phone Desktop

Raspberry Pi for Beginners

All you need to know to get started with Raspberry Pi

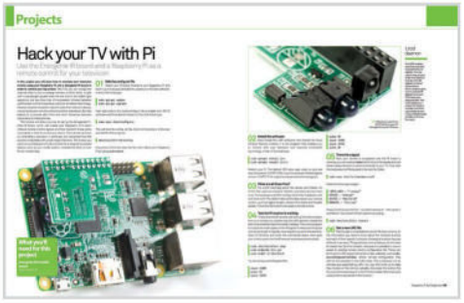
Over 20 projects inside





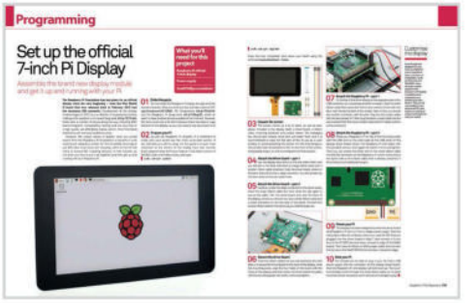
Basics

Navigate your Raspberry Pi and get to grips with all the essentials



Projects

Put your device into use and hone your skills with practical projects



Programming

Realise your device's potential with Scratch and Python programming