NEW

DISCOVER THE WORLD OF RASPBERRY PI!

PiUser

YOUR NEW SOURCE FOR RASPBERRY PI

JOIN THE PI REVOLUTION

Issue 04 // Autumn 2017

Secrets of Interfacing

116 PAGES OF ADVICE INSIDE

Ranges from set up tips for beginners to advanced hacks

25 Maker Projects

Create a media centre, arcade cabinet, and more

Learn to design simple electronics to connect your Raspberry Pi to real-world devices

Future

ISSUE 04 2017

Build a VR setup

Make virtual worlds with Pi3D and Python

Self-flying drone

Program a Pi-powered autopilot mode

Run a VPN through your Pi

Wirelessly connect to your own private network

Future Publishing Ltd
Richmond House
33 Richmond Hill, Bournemouth, BH2 6EZ
Tel +44 (0)1202 586200
Email piuser@futurenet.com

EDITORIAL

Creative Director Aaron Asadi
Art & Design Director Ross Andrews
Editor-in-Chief Jon White
Editor Jack Parsons
Senior Art Editor Andy Downes
Designer Neo Phoenix

ADVERTISING

Commercial Sales Director Clare Dove
clare.dove@futurenet.com
Senior Advertising Manager Lara Jaggon
lara.jaggon@futurenet.com
Advertising Manager Michael Pyatt
michael.pyatt@futurenet.com
Director of Agency Sales Matt Downs
matt.downs@futurenet.com
Ad Director – Technology John Burke
john.burke@futurenet.com
Head of Strategic Partnerships Clare Jonik
clare.jonik@futurenet.com

MARKETING

Marketing Director Richard Stephens
richard.stephens@futurenet.com

PRODUCTION AND DISTRIBUTION

Production Controller Nola Cokely
Head of Production UK & US Mark Constance
Distributed by Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU **Tel** 0203 787 9001
Overseas Distribution by Seymour International

LICENSING

Senior Licensing & Syndication Manager Matt Ellis
matt.ellis@futurenet.com **Tel** + 44 (0)1225 442244

CIRCULATION

Trade Marketing Manager Juliette Winyard
Tel 07551 150 984

Printed in the UK by William Gibbons on behalf of Future.

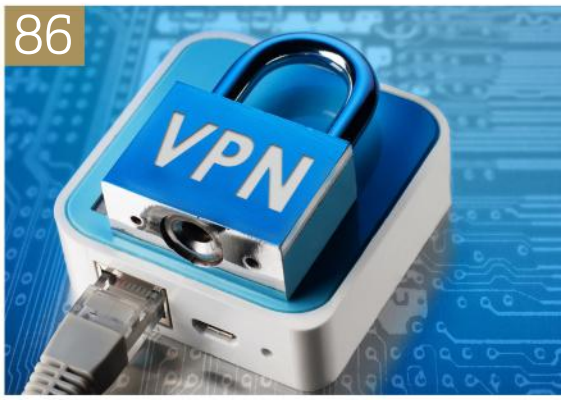
Distribution in the UK, Eire & the Rest of the World by Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU
Tel 0203 787 9001 **Web** marketforce.co.uk
Distributed in Australia by Gordon & Gotch Australia Pty Ltd, 26 Rodborough Road, Frenchs Forest, NSW, 2086 Australia
Tel +61 2 9972 8800 **Web** gordongotch.com.au

Contents

Discover amazing projects to make with Pi from **page 26**

06

78



86

"LEARN TO MASTER YOUR MICROCOMPUTER"

WORLD OF Pi

Discover what a Pi can do!

GET STARTED

Learn the Basics for you Raspberry Pi

Pi PROJECTS

Hack hardware and put your Pi to use

Pi REVIEWS

Latest add-ons tested and rated

Adding printing functions to the case: Thanks to the personalised casing, Pierre could simply slide a cheap thermal printer into the top of the unit. It's attached with just four miniscule screws, which have their own cut-out groove in the unit

Printing on-the-go: One of the biggest issues Pierre faced with this project was the original printing speed. However, thanks to a series of software tweaks he implemented, he could get his taken images fully printed in just a matter of seconds

3D-printed case: Pierre took the time to design the entire casing of the PolaPi Zero in Autodesk. He stripped back the complexity traditionally found in 3D printing, choosing to print just three plastic components for the PolaPi Zero

Tinkering the camera settings: To best maximise the capabilities of the Pi camera module used here, Pierre used its onboard Python library. Within here, he could tinker with the finer points of the camera set up, and toggle how it would work in tandem with the printer

PolaPi Zero

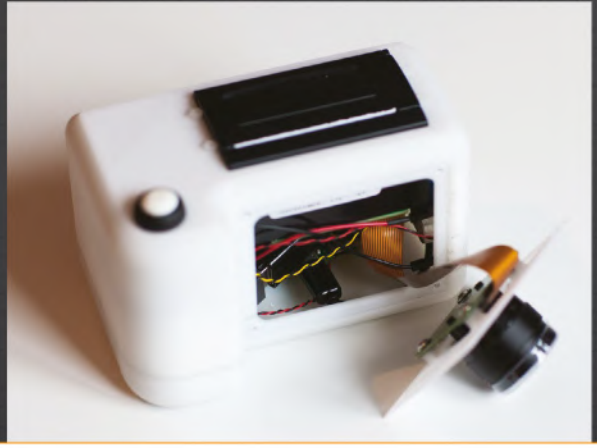
Maker Pierre Muth's camera project is a creative mix of nostalgic Polaroid printing and high-tech digital capabilities

RECIPE

- Raspberry Pi Zero
- Thermal receipt printer
- Sharp memory LCD
- Official Pi camera module
- 3D-printed case
- 3A voltage regulator
- 7.2V battery

How did you come up with the idea behind designing the original PolaPi and the PolaPi Zero?
I was first inspired by a couple of other camera-based development projects, named the PrintSnap and Polatherm Camera. These gave me some ideas for when I developed the original PolaPi model. I like the idea of the Polaroid camera being able to not only show the picture you just took to your friends, but to physically print the picture as well.

After the original version was produced, I sat down with fellow developer Vit Hasek and discussed the possibilities of making the device smaller and more convenient. This is what started the development of the PolaPi Zero. He came with a very clever idea that I didn't have enough time to implement in the first version.



The nano printer used here prints on regular receipt paper, which means printing quality isn't the best

Would you say it's a blend between classic Polaroid cameras and newer digital cameras?
Yes, I would say so, but there are compromises in both areas. The print quality is obviously bad compared to Polaroid cameras, and the picture files you get are not as great as the ones you can obtain with a recent smartphone. However, in addition to the satisfaction of building something and not only using something, the prints are a lot cheaper and it is fun to use with friends. It opens the door to creative modes and is accessible enough to be rebuilt by a lot of people.

Due to the intricate details of this project, has it been difficult forming a finished product?
I had difficulties on both versions, mainly due to my hardware choices and my lack of knowledge on certain domains. On both versions, the main issue was the internal buffer of the printer. The print speed changes with the darkness of the image. It means if the print is bright, less pixels have to be added and thus the paper advance is quicker, and the

CREATOR

PIERRE MUTH

is a long-time Raspberry Pi tinkerer and engineer, who likes to explore how best to integrate the Pi into everyday items

other way around. The screen I've chosen also gave me some head-scratching. I was close to choosing a more standard way, such as the PiTFT of the first version. These Sharp 'memory LCDs' are nice to look at, though, so I decided to continue. They are perfectly visible in sunlight as they are reflective LCD and are – as the printer – pure monochrome.

At first, I started by generating the image data to be sent on the SPI port in the program, but I faced some problems when sending the 12 Kbytes of the image in one block. Fortunately, I found the wrobell library for this LCD: **https://github.com/wrobell/smemlcd**. I spent a bit of time figuring out how to compile the library with Autotools and deal with the error messages it returns.

The case itself looks fantastic. Did you use anything in particular to design and print it?
I started from scratch with the Autodesk 123D design free software. If you're a bit familiar with CAD software, it seems quite limited at first. On the other hand, it is possible to achieve a lot of things within an hour, including the learning time. I tried to limit the number of printed pieces. For tactile buttons, it is a small strip and a square part to hold the button in place. The front and back plates are fixed with four screws, and the hole threads are just made with a tap in the case plastic. The case can then be easily opened.

What sort of printer did you use for the project? Was it easy to implement into the small frame of the PolaPi Zero?
I think I used one of the cheapest thermal printers out there – you can easily get it from Adafruit. It is the Cashinotech CSN-A3. It is meant to be slot-in, so just four small screws will hold it well in the case. It uses a special kind of paper, the very cheap receipt thermal paper, and you can find much of it from recycling outlets. There is a big debate about the waste of this

"IT'S CHEAP AND FUN FOR FRIENDS"

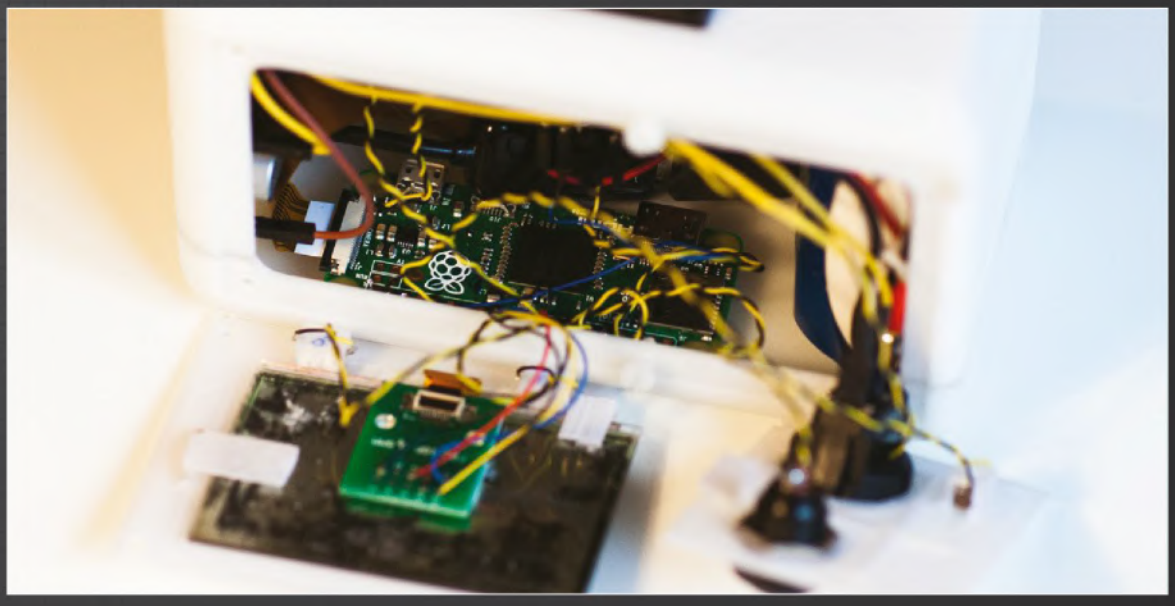
paper type, as the receipts are usually just thrown away. Nevertheless, here the aim is different. You will never print as many tickets as a supermarket, and in most cases the prints are meant to be kept.

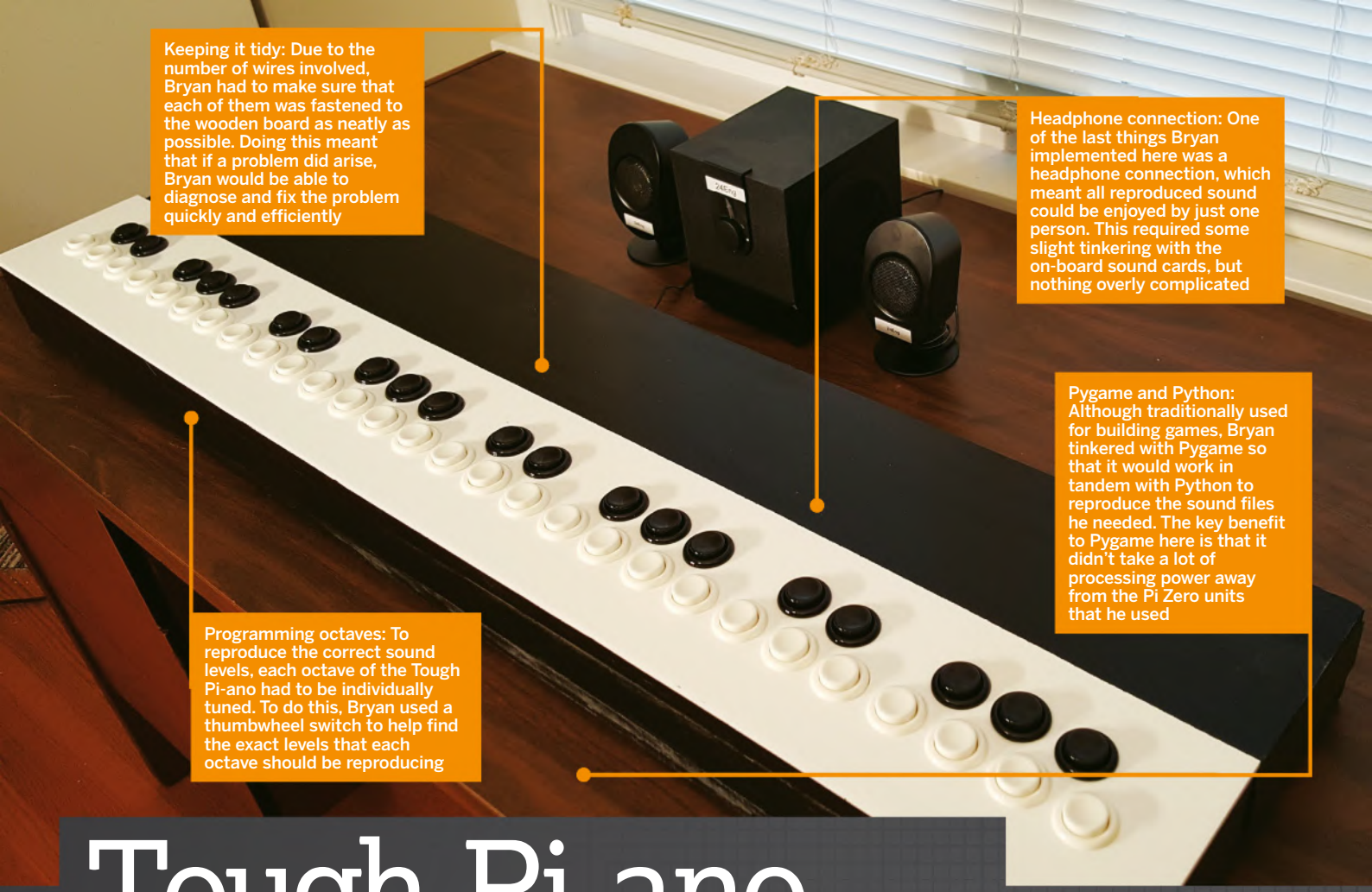
How effective is the Raspberry Pi in this project? Did you find it easy to work with?
Starting with the Raspberry Pi leads you into a vast world of knowledge to acquire, especially if you're not confident with Linux. I started playing with them from their first version and I still play with RPIs because of the enormous community. All the installation process of Raspbian became very easy and the libraries development have aided me in my progress. If you still encounter a problem, there is a large probability that someone else has both faced it and provided a solution.

The Raspberry Pi Zero is a bit more 'closed' than B models, as there is only one USB port. I liked the facility that copies the Wi-Fi network setup file (wpa_supplicant.conf) from the FAT32 partition to the right place at boot time. I then could work on it by connecting a USB Wi-Fi dongle; so, no screen, keyboard or mouse.

After spending some time with it, how do the end photos look? Are you happy with the results?
Do not expect good prints, of course. It is low-resolution (384x640 pixels) and only black or white in its current form. The grey gradients are the result of the error-dithering algorithm, so don't look at them too closely. However, they're unique prints, and a lot of fun to produce. In the future, I'll be looking to improve the quality of them further.

The LCD screen at the rear of the unit is used to manage prints, as well as see on-the-spot previews of taken images





Tough Pi-ano

Engineer Bryan McEvoy's tough-as-nails piano takes an all-new approach to music exploration

RECIPE

- Raspberry Pi units
- Charging and OTG cable
- Dual sound cards
- Thumbwheel switch
- Headphone cable
- Powered speakers
- Headphone splitter
- Power adaptor
- Hardwood casing
- Buttons

Where did your inspiration for the Tough Pi-ano come from?
I started the Tough Pi-ano after talking with my aunt. My cousin loves music therapy, particularly playing the piano, but he has Down's syndrome and his excitable therapy sessions can destroy instruments meant purely for gentle fingers. While I thought of a solution, he kept smashing every garage sale piano they were able to pick up and buy. I knew that each of those pianos was really just a series of inputs and a rather basic sound generator.

Has it been easy getting the project from concept to working device?
One year ago I started working on a version which used hardwood keys with the same proportions as a standard piano, but they were difficult to make in my local hackspace – so I had to rethink this idea. A couple months after the piano had been left aside, I decided it would be better to use commercial parts for the keys even if they didn't look like a piano. After all, I was building this for kids who wanted to smash buttons, not concert pianists. I ordered a ton of arcade buttons in black and white, which obviously give that natural piano look.

Replaceable octaves didn't make it to the final version, but using one computer for each octave stayed. Even if someone managed to destroy one of the computers, three-quarters of the piano would keep running. It would also be possible to have spares on hand until a proper repair could be made. All the arcade switches were connected with slide-on terminals so they could be replaced if someone wore it out. Redundancy was probably the strongest quality of the Pi-ano, but you wouldn't see it unless you looked under the hood.

Designing a musical instrument with the explicit intention of absorbing abuse was a different kind of brainstorming for me. Many of my previous projects were only meant to be strong enough to withstand ordinary wear and tear. The Tough Pi-ano had to take punishment all day long without hurting anyone and have lots of parts that were easy to service. That's why there's a plastic sheet over the keys: no slivers. It'll be wall mounted so it can't fall on anyone. There is no exposed metal if knuckles start scraping. It was a whole different way to approach the user interface.

How accurately does the Tough Pi-ano reproduce sound?
The sound files for this project were legitimate piano samples and the keys corresponded to the correct notes, so anyone who played the piano could sit

CREATOR

BRYAN MCEVOY

is an automation engineer by trade, which has given him the chance to design and build some pretty incredible things.

down and perform a song, but they would have to reach more since the keys were spaced out so much further than a standard piano. Of course, certain limitations are noticeable, but nothing major.

We've seen some mention of using the piano tool within Pygame on your Tough Pi-ano. Did this prove to be a big help?
Pygame was a huge help with this project. The community surrounding Pygame was wonderful, I didn't have to ask them a single question, I was able to find everything I needed in their guides and forums. That kind of support is the same reason I program with the Raspberry Pi. If someone wanted to copy my project, they could do it inexpensively and if they had trouble, there's help available. In fact, someone has already started to build their own Tough Pi-ano for a school and together we discovered that my code won't work on a Pi Zero W.

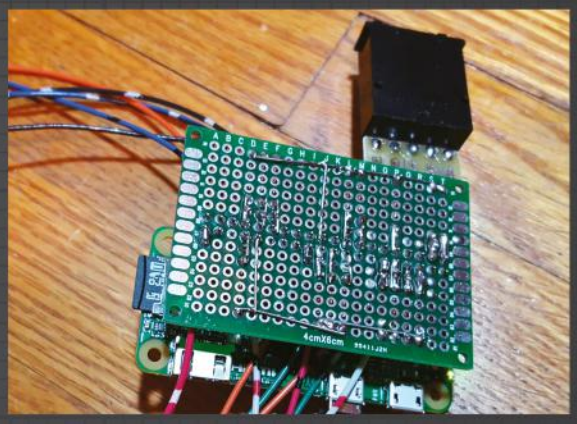
How much of a learning curve has this project been for you in terms of developing with the Pi and Python?
I knew almost nothing about Python when I started this project and I learned a lot. It was honestly a very good project for a beginner, since you build the same switch circuit 12 times for each octave and then build a total of four octaves. One of my rookie mistakes was building a circuit board to hold pull-up resistors for each input. Later, I learned this could have been done in software. I won't repeat that mistake in the future.

How happy are you with the finished product? What would you say is its best feature?
The Tough Pi-ano's best quality, in my opinion, was the simplicity. You put power on it and then you have 48 buttons that make sounds. Nothing extraneous, no settings that might interrupt a jam session, no knobs to break off, no hinges to pinch fingers. One change I would welcome would be a solid case made of heavy-duty plastic.

My aunt and uncle are opening a facility on their property where other kids with Down's, or on the autism spectrum, can come with their families to spend time in a safe environment and be themselves. I'm pleased that they will have a piano they don't have to be gentle with.

Looking past the Tough Pi-ano, will you be using the Pi to build any other projects in the future?

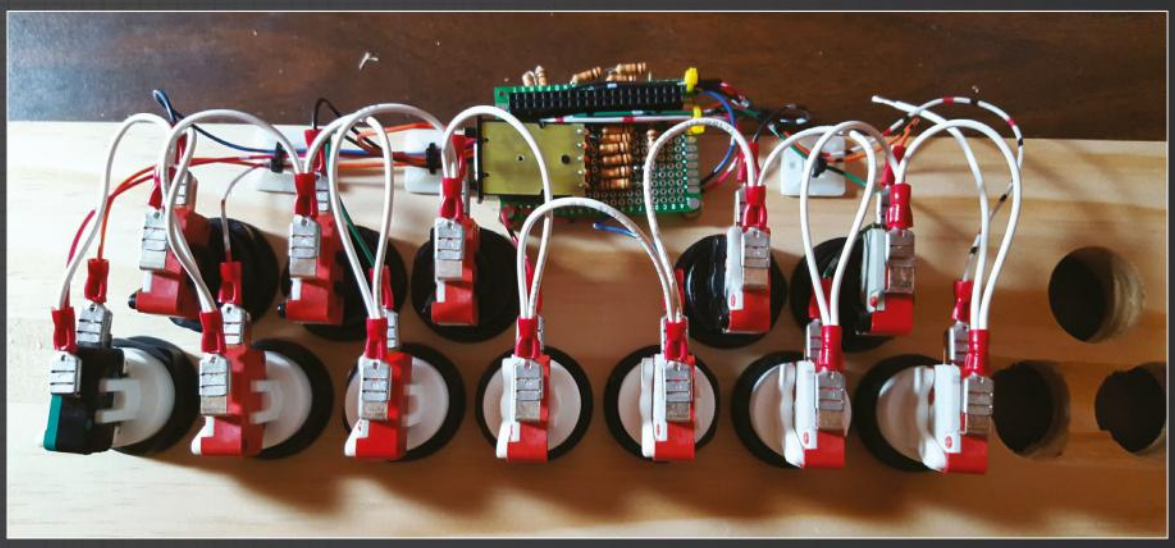
I think the Raspberry Pi line computers are great pieces of hardware. In the past, I've built a MAME box and a wearable computer with a head-mounted display; it runs on just a USB battery pack. Right now though, I'm using them to build a laser tag game with Wi-Fi connectivity. I like to detail all my projects, including how I build them, on my personal blog (**www.24hourengineer.com**), as I find it helps me to keep motivated and keep on developing.

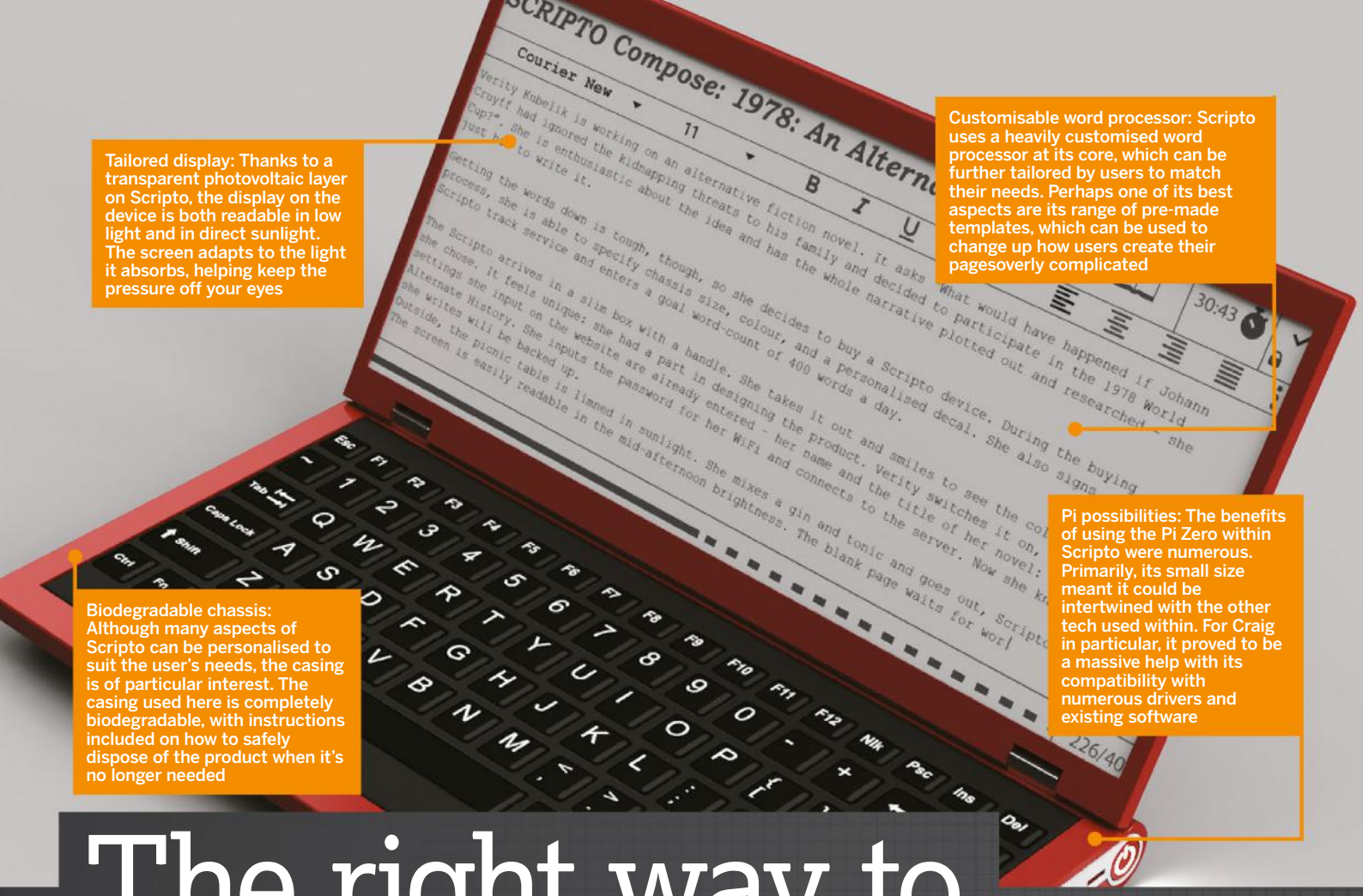


"THE TOUGH PI-ANO HAD TO TAKE PUNISHMENT ALL DAY LONG"

Due to the number of octaves and buttons required, Bryan needed to use multiple Raspberry Pi units to play all necessary sound files

Bryan had some initial static problems from the default USB power supply he used. Switching to a coaxial connector seemed to fix the issue, however





The right way to write with Pi

Scripto is like a miniature laptop that offers a distraction-free way to type on the go

RECIPE

- Raspberry Pi Zero (with enclosure)
- Bio-degradable Arboblend chassis
- LCD touchscreen
- Sunpartner transparent solar panel
- Swappable side panels
- Wi-Fi antenna
- 4 x AA batteries
- SD card reader
- USB power input

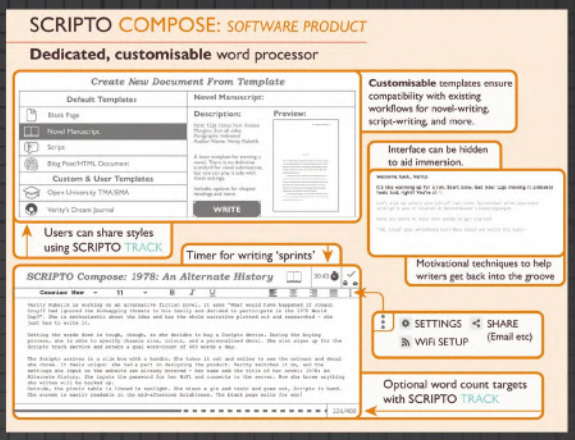
Where did the original idea for Scripto stem from?
The Scripto started as a final project as part of my Computing and Design degree at the Open University. The concept of a distraction-free device for writers is not new, but it is currently under-served in the market. The Raspberry Pi was also inspirational in providing an accessible computing unit for makers who aren't necessarily hardware experts.

Talk us through the design and build phase of Scripto, did you encounter any major issues?
The Scripto was designed with three major goals in mind: it should be distraction-free, it should be convenient versus other solutions, and it should be cheap. Balancing the other factors with cost was a continual challenge. As the device is currently designed, I think the price would be too high and a major goal for the next few months is to redefine the budget to cut features that are nice-to-have but not strictly necessary, such as solar charging. For myself, using sustainable and innovative materials in the chassis was an important aspect of the design, but actual users may not care that their device is green. Another challenging aspect of the design is that the targeted demographic of 'writers' is extremely broad in terms of needs and disciplines. Script writers for theatre and film have different needs to poets and journalists. At a basic level, they all write words and should be able to replicate their chosen formats manually, however, making things convenient and seamless was a design goal, so

CREATOR

CRAIG LAM

Craig has a first class honours degree in Computing and IT and eight years of experience in CAD.



Each panel of Scripto can be customised. Different colours and materials are available to make it your own

While many core word processing features are present, the customised software used here is great for prolonging your focus

the inclusion of templates and formatting requires an assessment of many competing formats. Every author seems to have a different workflow. It may be that the Scripto reverts to a pure text generation device, leaving it up to the writer to copy the basic text into their chosen format. Support for Markdown will be included at the least, however. A major challenge was deciding on the keyboard. Writers usually have preferences about their keyboard. Some prefer mechanical switches, others like the short-travel switches on modern Thinkpads and Macbooks. One thing is for sure: the keyboard must be no smaller than that found on 12" laptops. Writing is a physical act and getting the feel of the device correct will probably require many iterations. A challenge will be to find the resources and partners to help develop affordable prototypes.

For our readers who may not be familiar with Scripto, what are its key features?
The Scripto supports word processing with optional cloud backups. In essence: an old-school word processor like in the 80s and 90s, but with the convenience of automatic backups and progress tracking. It's a small, light device made of sustainable plastics using low powered hardware. A long-term goal would have it integrate with an app and a secure cloud storage service to track and backup users' words.

How does the Scripto help you deal with distractions?
The Scripto doesn't have any distractions: it should be a one purpose device. Beyond selecting which project you are writing in, the barriers between you and typing your thoughts should be minimal. Laptops, desktops, and smartphones are all capable of word processing, and they're also capable of running distraction-free environments – one could customise a basic Linux distro to boot into a word processor, for instance, but I think there is a benefit to having a very portable and dedicated device.

Fostering a positive physio-psychological association with the act of writing is important, and specific hardware is a good way to achieve the mindset that "I'm sitting down with my Scripto to write, and nothing else".

Can Scripto be personalised in any way?
The intent is to make the Scripto a highly personal device. I go into tea shops and cafes and see rows of silver Macbooks – I have nothing against that, personally, but I think we're about to enter an age of mass customisation, and this should extend to our gadgets. Writing is individual and personal, so the tools we use should be as well. If the chassis can be 3D printed as intended, the long term goal is to offer diverse colour options and decals. However, this becomes a challenge of pricing. Early versions may only come in one style and colour.

What sort of role does the Raspberry Pi play here? What were the benefits of using it in this project?
The Pi brings numerous benefits: primarily that it is a whole system on a single board and that it is compatible with existing drivers, software and accessories. The amount of documentation available is second-to-none. There may be more appropriate solutions – an Android based device may be better in terms of power management, but that requires research. At present, the project will use a Pi Zero. The Pi being UK-manufactured fits with the ethical and ecological aims of the project too.

Do you think you'll look to utilise Ras Pi in future projects?
I am always on the lookout for new technologies, but at the moment I could see myself using the Pi again. The Zero has an inspiring form factor – it could fit into all sorts of small interactive products; it's just finding time to explore!

What's next for you? Any other big projects on the horizon?
I recently graduated and I'm looking for work. The Scripto is an ongoing project, however, and I intend to have working prototypes available this year – my wife keeps nagging me! Crowdfunding may be an option if I wanted to scale up, but I'd prefer to have proof-of-concept before embarking on that journey. I also expect to be focusing on smaller projects, one of them being an online interactive guide to tea. I have a preoccupation with cats and so I'm investigating ways to make 'smart' cat toys.

The DIY robot arm

Getting started in the world of Linux can be difficult, but the beginner-friendly MeArm Pi aims to give users an all-in-one development lesson. **Ben Pirt** explains how this robotic creation came to fruition

"THE MEARM PI IS A ROBOT ARM THAT A CHILD CAN BUILD. YOU CAN RUN IT FROM A PI"

The world of robotics is notoriously hard to get to grips with, especially if you aren't coming from a development background. It's a problem that thousands of people face every year, but also something that Mime Industries has looked to quash. The company has introduced a range of open-source robotic kits, which strip back the complexity of building robots and lay the foundations for new users.

At the helm of Mime is Ben Pirt, who has a varied history in working on both energy monitoring hardware and renewable technologies for the developing world. While his original work in the world of robotics was solely done in his spare time to help educate his children, their love for it has spurred him on to make the kits more widely available. Since then, himself and fellow co-founder, Benjamin Gray, have had two successful Kickstarter campaigns, and their highly regarded Mirobot is now used in more than 50 countries. Here he talks about his latest project, the MeArm Pi and how it's taking basic robotics to the next level.

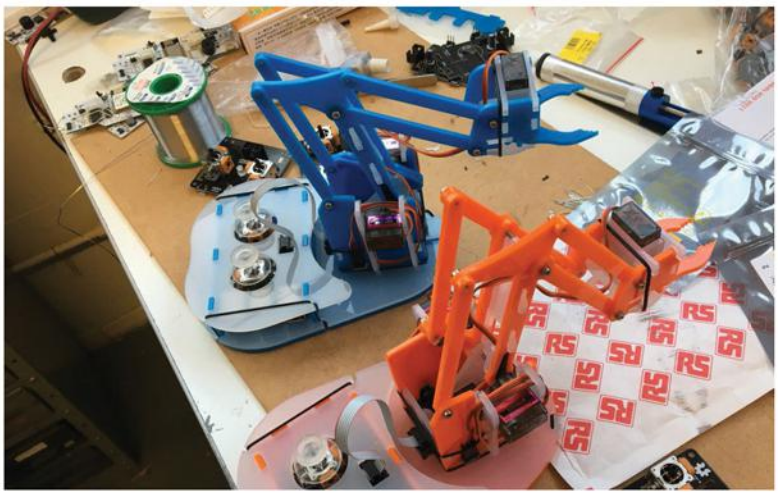
A lot of our readers will have heard about the original MeArm, but for those who haven't, could you give us an idea on what the MeArm Pi is?
The MeArm Pi is a robot arm kit that's easy enough for a child to build. It attaches to a Raspberry Pi so that you can control it either direct from the Pi or using apps that run in your browser. It's designed to be a low-cost way of learning about technology while having a lot of fun in the process. We've taken a lot of the elements you see in developing in Linux as a whole and aimed to make the process far more accessible. It's important to bring in new people to the world of developing for the Raspberry Pi and Linux as a whole, and we'd like to think that we've managed to do that with this product.

What are some of the key differences between the original MeArm and this MeArm Pi?
Perhaps the most fundamental element that's changed between our original MeArm and the Pi version is the original MeArm could be connected to a Pi, but it was down to you to do some wiring of the pins on the Pi to the motors on the arm. It was far more complicated, but we still outlined the build process for new users. With the MeArm Pi, we've tightly integrated the two so that there is now a HAT that sits directly on the Pi, which has a couple of joysticks on board for direct control. In use, this has proved to be a big improvement on the original control system we were looking to implement here.

The other major difference is subtler: while it looks superficially the same as the previous version, the new model uses only around a third of the screws which makes it far easier to build with the instructions we include. Of course, there's still a good level of DIY involved, but the complexity has been removed.

How easy would you say it is to build the arm? Is it particularly tailored to introduce core development concepts to new users?
It's designed so that children of age 11+ can build it themselves, which is something we're particularly proud of. We've gone to great lengths to reduce a lot of the more complex parts of the build from the earlier version, which as I mentioned earlier, was a surprisingly time-consuming process. For example, you used to

Mini-Me Arm? The new Pi version is much smaller than Mime Industries oversized original



You will only need a hex key to assemble your MeArm, so you should be able to put it together in around 30 minutes

PROGRAMMING WITH MEARM PI

Getting to grips with the development process behind the MeArm couldn't be simpler, and there's a few ways for users to get stuck into it. For one, the MeArm can be controlled directly from the GPIO pins on the Raspberry Pi, via a custom HAT that cleverly interlinks the two. This means that nearly any programming language can be used here, and it comes down to personal preference on the one you choose to use. As an alternative, the MeArm team have also built a clever app in Node.js that allows for in-browser programming. For end users, this enables the MeArm to be controlled via a web API, which in turn can help prevent children from needing to learn the basics of the command line before getting stuck in. The choice is in the user's hands, so there's no real reason why this project can't be tailored to you.

have to calibrate all of the motors before attaching them, but because of the new design we are now able to supply these pre-calibrated and streamline the overall build process. On the Raspberry Pi side of things, we're trying to cater to absolute beginners. We provide a ready-to-go image which will start working out-of-the-box and have also put a lot of effort into being able to run the whole system headless, so users don't also have to buy a monitor, keyboard and mouse. Not only does this add to the simplicity, it helps keep budgets to an absolute minimum. The build time is around 30 minutes (compared to a few hours for the previous model). Our goal is that a complete novice who has never used a Raspberry Pi before should be able to build and use the MeArm Pi in 30 minutes, and for those with some expertise, this could be used as a project to refresh your skillset.

How integral has the Raspberry Pi been in the development of the product?

The Raspberry Pi is central to the running of the MeArm Pi and all of its core functions. All of the motors are controlled directly from the GPIO pins and the joysticks communicate directly with the Pi as well. The software is a Node.js service that lets you either use the joysticks or control it through a web API, which is what all of the built-in apps use. We love the Node.js service, as it's once again, a relatively simple piece of kit for new users to learn and develop with.

Is there scope to connect other pieces of hardware to the arm?

Definitely! You can see a demo I did earlier where I hooked up the camera to some facial recognition

The Me Arm Pi is designed to be child's play to use

software so that the arm could follow you around the room through our website. We've also got plans to create some external hardware that the arm can interact with that will have sensors that the Pi can talk to. This is very much in the early stages, as there's a wide variety of options for how we can do this and use them to help educate people further.

What can you tell us about the software that comes with the MeArm Pi?
The big benefit of being able to control the MeArm directly from the GPIO pins of the Raspberry Pi is how many options it opens up for controlling it. Basically, any programming language that can control the pins can control the arm. We've already had it working with Python, Node.js, Ruby and Java but there are many more opportunities.
In terms of the software that we provide, it's a Node.js service that hooks up the joysticks to the arm as well as making a WebSocket API available to control it.

This means that we've been able to make browser-based apps to control the arm that work in any web browser. So far we have Blockly (a block-based visual programming language), Python, Javascript and Snap! (a JS Scratch clone) but there are lots of others on the way. The big benefit of running these in the web browser is that you could set up the arm in a classroom, for example, and then kids can use any browser to control it without needing a monitor and keyboard attached.

So far in its development, what are some of the best ways that you've seen the MeArm Pi used in everyday situations?
The best ways for me are when it's used in workshops or classrooms to teach kids about technology. Ben Gray ran a workshop at the London Science Museum, for example, that helped get kids enthused about not only MeArm, but what a screwdriver and a computer board are capable of. Aside from that, there's been a lot of fun projects built through the MeArm; someone even sent us a concept of a toothbrush helmet they made!

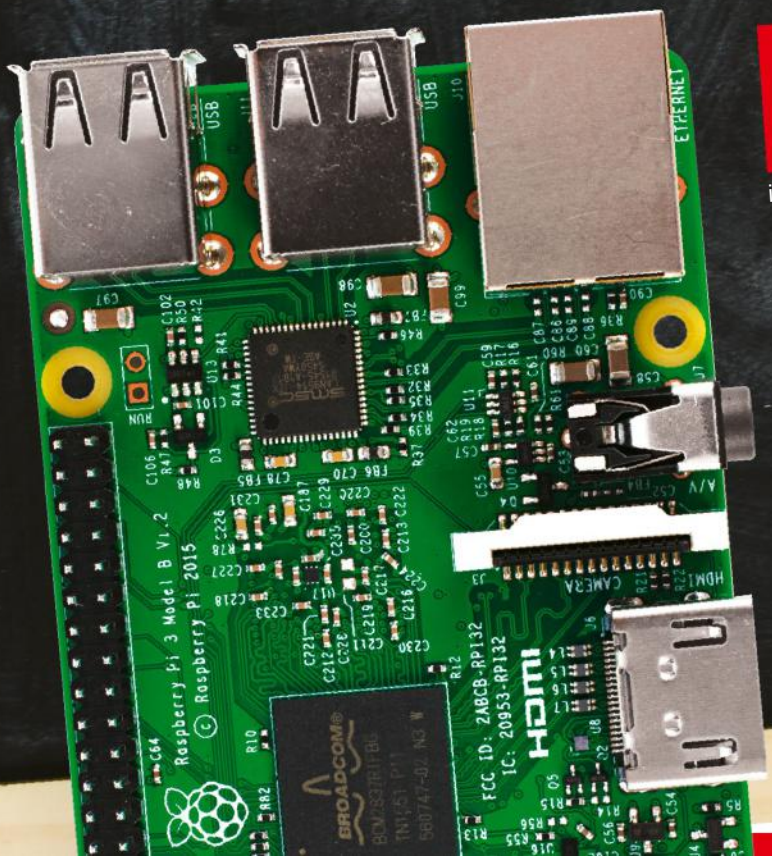
To learn more about Mime Industries and the MeArm Pi, visit www.mime.co.uk.

BUILDING BLOCKS

When purchasing the MeArm Pi kit, the team aren't wrong when they say you'll get everything you could possibly need. The entire structure of the arm is included, with every piece cut out in a reinforced plastic mould. On the side, there's also socket head screws, a hex key, four metal gear servos and a Raspberry Pi HAT with the necessary joystick controls already implemented. Plus, if you don't already own a Pi unit, there's no need to panic. Each package comes with its own Raspberry Pi 3 unit, the official power supply and a software-preloaded 8GB SD card. Perhaps even more important than all of those things, is the included build instructions that guide you through building your own MeArm from start to finish.

SECRETS of PI INTERFACING

Learn how to design simple electronic circuits to interface your Raspberry Pi to real-world devices



Using a single-board computer such as a Raspberry Pi to control real-world devices requires two quite different skills. First, you need to be able to churn out code, and second, you need to be able to interface the Pi to external devices.

Here we look at the second of those areas and, in particular, investigate how to go beyond using off-the-shelf interfaces like HATs, or even building circuits that others have designed, by designing your own electronic circuits. Using this hands-on guide, you'll soon be able to connect switches, LEDs and so much more. Here we'll present circuit diagrams, but if you're not familiar with them, see our earlier guide, which explained how to turn a circuit diagram into a working circuit.

Our main emphasis is interfacing to the Raspberry Pi, but most of this can also be applied to other small computers such as the Arduino, which are also popular for control applications.

Understand the Pi's GPIO hardware

The Pi's GPIO pins are a direct connection the real world

Even if you've never connected external devices to your Raspberry Pi, you can't fail to have noticed the double row of 40 pins at the edge of the board (26 pins on the early Pis). This is the GPIO header, otherwise known as the general-purpose input/output connector, and it provides a means of interfacing to real-world devices. Before starting to think about designing circuits to interface to the Pi, therefore, it's important to understand the basics of the GPIO hardware.

Power and ground pins

Although referred to as the GPIO header, not all the pins connect to the GPIO hardware. Some of the other pins provide power and ground connections that are also used by hardware that's connected to the header.

The Pi's GPIO header has eight ground pins (GND), which you can identify from the documentation. Ground is equivalent to the negative side of the power supply and is often referred to as 0V (ie zero volts).

The GPIO header also has four power supply pins: two that provide +3.3V and two that provide +5V. Using the ground and power supply pins allows you to obtain power from the Pi for your external interface circuitry so you don't need a separate power supply.

GPIO Pins

With two exceptions, the remainder of the pins on the GPIO header are GPIO pins although some also have secondary functions that we're not going to get embroiled in here.

As the phrase 'general-purpose input/output' suggests, these pins can be configured in the software to act either as inputs or outputs. When programmed as inputs, these pins could be connected to a switch, for example, and the software would be able to read whether the switch was open or closed, i.e. on or off. Alternatively, when programmed as outputs, these pins could be connected to an LED, and the software would be able to turn it on or off.

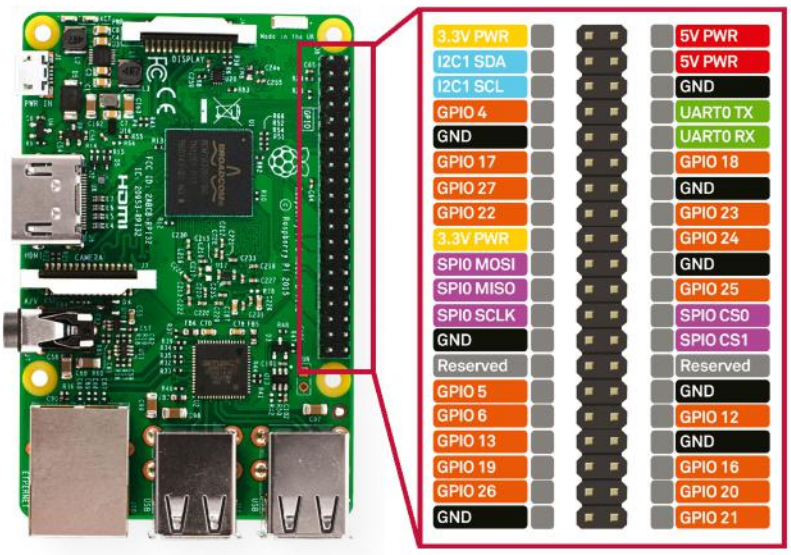
GPIO PIN NUMBERING

There are two numbering schemes for GPIO pins. First there's the physical numbering. This reflects each pin's position on the header, so it runs from 1 and 2 at one end to 39 and 40 at the other. Then, for the actual GPIO pins (as opposed to power supplies), there are GPIO numbers. You can choose to use either scheme in the software.

Maximum ratings

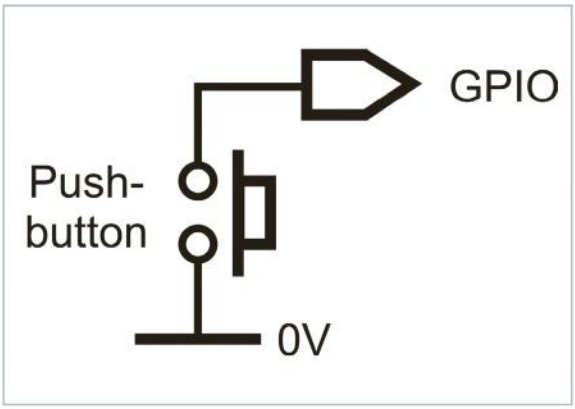
The Raspberry Pi's GPIO operates from a supply of 3.3 volts, so you should never, ever present a higher voltage to any of the pins. Doing so will probably destroy your Raspberry Pi. However, there are ways of interfacing to devices that require higher voltages, as we'll see later in the 'Exceed limits safely' section (on page 24). In fact, there are even ways of interfacing to mains-powered equipment and this is also something we'll discuss.

The maximum voltage isn't the only way of exceeding the GPIO's maximum rating; you should also adhere to its maximum current of 16mA (and a total of 50mA for all GPIO pins). In practice, this means that you'll easily be able to drive an LED, which doesn't require much current, but driving a higher-powered device such as an electric motor requires a bit more attention. Again, this is covered in the 'Exceed limits safely' section.



How to connect a switch and an LED to the Pi

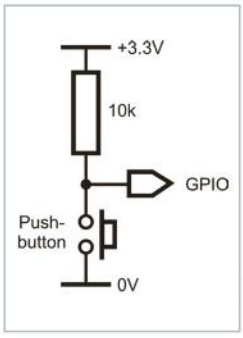
Interfacing a switch or an LED to the GPIO header really couldn't be simpler



01 Wire in the switch

The first job in interfacing a switch to the Pi is to connect one of the switch's two terminals to a GPIO pin (which will be configured as an input in the software) and connect the other of its terminals to 0V (GND). Having done this, the GPIO pin will be connected to 0V, a condition that the software will see as a logic 0, whenever the switch is closed, ie held down in the case of a push button or in its 'on' state with a mechanically latching toggle switch.

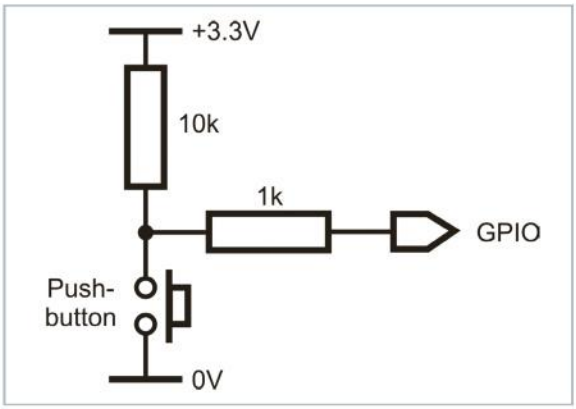
02 Add a pull-up resistor



Although a GPIO pin wired to a switch and 0V will be at logic 0 when the switch is closed, it will be 'floating' when it's open. In other words, it wouldn't be certain whether it would be seen as a 0 or a 1. To overcome this, it must be wired to +3.3V via a resistor, which is referred to as a pull-up resistor. Now, the GPIO pin will be logic 1 when the switch is open. The resistor value isn't critical, but 10k is a good choice.

03 Use built-in pull-ups

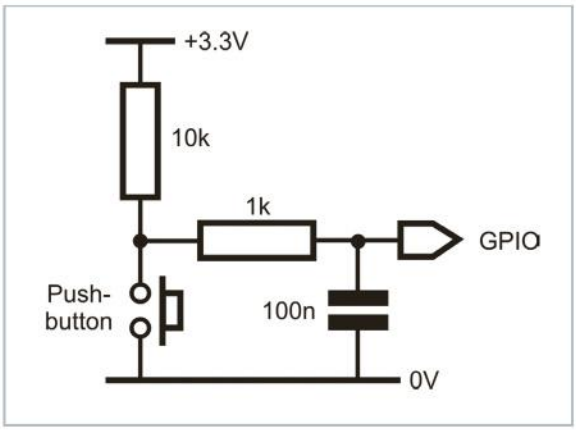
If you're wiring the switch to some types of other single-board computer or to the Pi's GPIO via some logic circuitry, an external pull-up resistor is the only solution. However, if you're interfacing directly to a GPIO pin, you can, as an option, enable an internal pull-up resistor in the Pi's circuitry. The bit of code reproduced here shows how this is done using the RPi.GPIO Python library. The circuit diagrams in the remaining steps assume an external pull-up, but, if you're using an internal pull-up, just omit the 10k resistor.




```
GPIO.setup (2, GPIO.IN, pull_up_down=GPIO.
PUD_UP)
# set GPIO 2 as input with pull-up
```

04 Limit the current

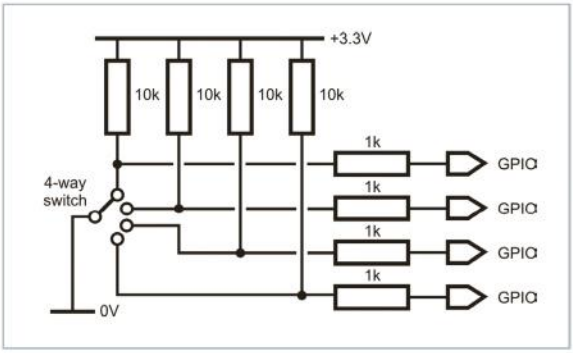
Because GPIO pins are bidirectional, there's a potential problem if a pin that's attached to a switch is accidentally configured as an output and set to a logic 1. This will put 3.3V on the pin which, if the switch is then closed, would be connected directly to 0V. This would cause a high current to flow and, potentially, damage the Pi. Putting a resistor in series with the switch will prevent this, and 1k is the recommended value. The series resistor is also integral to the circuit in the next step, so don't omit it if you want to add de-bounce circuitry.



05 De-bounce the switch

When a switch is operated, the contacts often open and close several times very quickly for a short time. This is called bounce, and it might cause problems. Perhaps pressing a push button is supposed to turn a LED on or off. Now, if the LED is off and you press and release the push button but it switches closed-

open-closed-open instead of closed-open, the LED will switch on and off again very quickly, but it would appear that nothing has happened. This is remedied by adding a capacitor as shown – a value of 100n is typical if you're using a 1k current-limiting resistor. Alternatively, software de-bounce can be selected in the RPi.GPIO library.



06 Use multi-way switches

A multi-way rotary switch is handled in just the same way as a single-way switch except that it connects to several GPIO pins and requires the interface circuitry described earlier for each of those pins. The circuit diagram shows how this would be done for a four-way switch. De-bounce capacitors aren't included because bounce isn't as much a problem with a multi-way or toggle switch as it is with a push button, because of their mechanically latching nature.

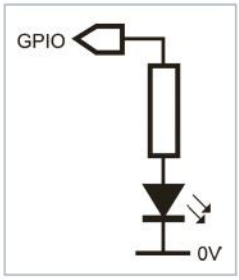
OPTICAL AND ELECTRICAL CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)

PARAMETER	TEST CONDITION	SYMBOL	MIN.	TYP.	MAX.	UNIT
Luminous intensity	I_F = 10 mA	I_V	1	4	-	mcd
Dominant wavelength	I_F = 10 mA	λ_d	612	-	625	nm
Peak wavelength	I_F = 10 mA	λ_p	-	635	-	nm
Angle of half intensity	I_F = 10 mA	φ	-	± 60	-	deg
Forward voltage	I_F = 20 mA	V_F	-	2	3	V
Reverse voltage	I_R = 10 µA	V_R	6	15	-	V
Junction capacitance	V_R = 0 V, f = 1 MHz	C_j	-	50	-	pF

07 Understand LED basics

An LED (light-emitting diode) is a component that produces light when a current flows through it. It's a polarised device, so its anode must always be connected to the positive supply and its cathode to the negative supply. A fundamental property of an LED is its forward voltage, which is usually between 1.8V and 3.3V depending on its colour and type. An LED requires this voltage in order to illuminate. Also important is the recommended current, at which its brightness and so on will be quoted. All these parameters are shown in the LED's specification sheet.

08 Limit the current



Turning on a LED from a GPIO pin involves configuring the pin as an output and then outputting a logic 1. This puts 3.3V on the pin, but this will probably exceed the LED's forward voltage, thereby causing its maximum current to be exceeded, destroying the LED and possibly damaging the Pi. This is prevented by adding a series resistor which drops the excess voltage. The resistor must drop the difference between 3.3V and the LED's forward voltage. The value is worked out using Ohm's law, as described in the next step.

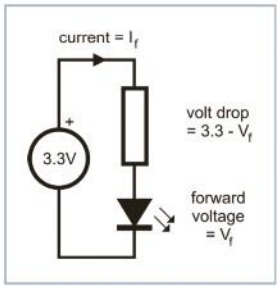
RESISTOR AND CAPACITOR VALUES

Often, the values of resistors and capacitors are not critical, and typical values can be used. However, when interfacing with an LED, you must work out the value of the current-limiting resistor or the filament might blue. Often you'll find that you can't buy a resistor of the value you've calculated because resistors (and capacitors) only come in certain preferred values.

In the common E-12 series, these values are 1.0, 1.2, 1.5, 1.8, 2.2, 2.7, 3.3, 3.9, 4.7, 5.6, 6.8 and 8.2. However, you may come across additional values from other series. These values can be multiplied by powers of ten. So, in the case of resistors, in addition to 4.7, for example, you'll find 47, 470, 4.7k, 47k, 470k and 4.7M.

If a resistor isn't available in a value that you calculated, the general rule is to play it safe. In the case of an LED current-limiting resistor, this means picking the closest larger value.

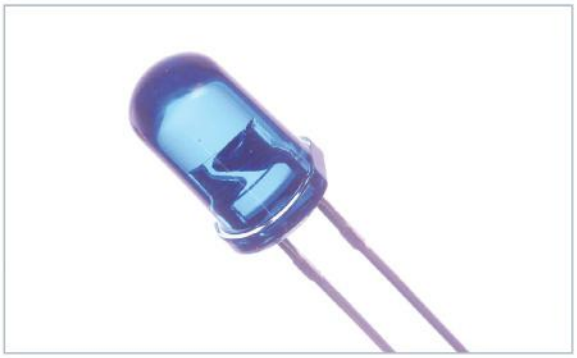
09 Make use of Ohm's law



First you need to decide the drive current for the LED. This must be less than the LED's recommended current and less than the GPIO pin's maximum of 16mA. Also, the total current drawn from all GPIO pins must be less than 50mA. 10mA will often give enough light, even if the recommended current is greater. Ohm's law is summarised as V = I R. This can be rearranged as R = V / I to give the value of the resistor, R, where V is the voltage that needs to be dropped (ie 3.3V minus the LED's forward voltage), and I is the drive current. For example, a forward voltage of 2.0V and a current of 10mA (0.01A) will require a value of (3.3 – 2.0) / 0.01 = 130 ohms.

10 White and blue LEDs

Some LEDs – mainly blue, white and 'pure green' – have forward voltages higher than 3.3V, so they can't be driven directly from a GPIO pin. Even if the value is specified as 3.3V, driving it directly from a GPIO pin would not be safe or reliable. The solution is to drive it from a higher voltage, as discussed later in the 'Exceed limits safely' section (page 24).



Logic circuitry explained

Understand logic circuitry to add extra functionality to your interface

Since the processor in the Raspberry Pi can carry out any imaginable logic operation, it might be reasonable to assume that there's no benefit to be gained from using external logic circuitry. While this would be true if the Pi has sufficient GPIO pins for your application, if you're getting close to the limit then by using external hardware logic, you can reduce the number of pins needed. Our step-by-step guide provides some examples of how to do this; here we provide an introduction to logic circuitry.

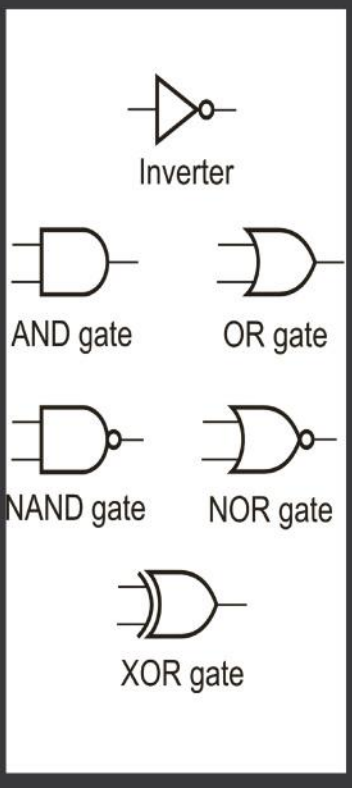
Logic levels

Logic components operate on two voltages that represent the binary values of 0 and 1 although, for some applications, it might be more appropriate to think of them as off and on respectively. In the case of the Pi's GPIO pins, 0 is represented by 0V (GND) while 1 is represented by 3.3V. With other single-board computers that use a 5V supply, 0 is still represented by 0V, but 1 is represented by 5V. You should choose a family of logic chips (see 'IC logic families') to match the supply voltage of your computer.

UNDERSTANDING LOGIC SYMBOLS

The diagram shows the standard symbols for the various logic gates. Each of these has one or more inputs at the left and a single output at the right. You'll notice that some symbols have little circles on their outputs. These are gates that have inverted outputs. So, for example, the symbol for a NAND gate (which means Not AND) is the same as that for an AND gate except for its inverted output.

Most other logic devices are just shown as boxes, again usually with the inputs on the left and the outputs on the right. Because so many devices would otherwise look the same when shown as boxes, these symbols are usually annotated with their part number, while their inputs are labelled with their function and usually their pin numbers. Logic devices connect to 0V and a power supply and while these connections usually appear on more complicated logic devices, they aren't normally shown on gates.



Inverter

The simplest logic component is the inverter, which has one input and one output – see the diagrams for symbols of all logic gates. As the name suggests, its function is to invert the value on its input; so, if the input is 0 then the output will be 1 and if the input is 1 then the output will be 0.

The operation of a logic component is often defined by a truth table and, while it's barely necessary in this simple case, the truth table for an inverter appears here:

Input	Output
0	1
1	0

AND and OR Gates

Next up after the inverter is a group of logic components referred to as gates. Gates can have any number of inputs (although two is the most common), and one output.

To set the ball rolling we'll look at the 2-input AND gate, the function of which can be summed up as follows. If input 1 is 1 AND input 2 is 1 then the output is 1; all other combinations of the inputs result in an output of 0. Using a similar statement, we can sum up the function of the OR gate as follows. If input 1 is 1 OR input 2 is 1 then the output is 1; all other combinations of input (actually there is only one other combination) result in an output of 0. Truth tables for the 2-input AND gate and the 2-input OR gate appear here.

Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

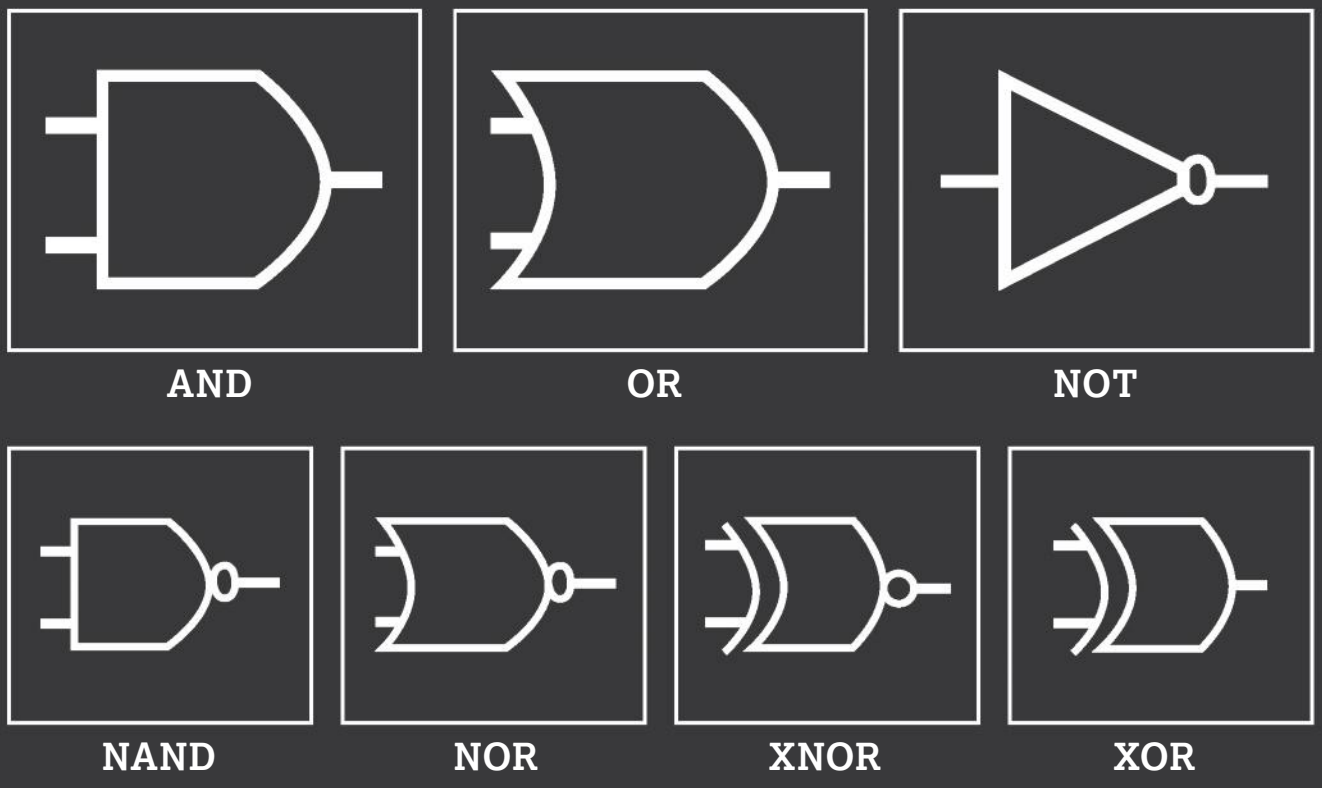
Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

NAND, NOR and XOR gates

The phrases NAND gate and NOR gate might sound odd at first. However, if we point out that NAND means Not AND, and that NOR means Not OR, then the pieces start to fall into place.

A NAND gate is effectively an AND gate with an inverter connected to its output and its truth table is that same as that for the AND gate but with the 0s in the output column changed to 1s and vice versa. Similarly, a NOR gate is an OR gate with an inverter connected to its output, so its truth table

IC LOGIC FAMILIES



One of the most common type of logic devices is the 7400 series. These have part numbers of the form 74<family><id>, where <family> is the 7400 series family and <id> is a number that defines the function. There might also be letters at the start and end, but you can ignore those. SN74LS00N, for example, is a low-power Schottky family device (LS) and its function (00) is a quad 2-input NAND gate (ie each chip contains four 2-input NAND gates). There are several 74 series families and most are not suitable for wiring directly to the Pi's GPIO pins. Some won't work with 3.3V inputs and outputs, and several are only available in surface-mounting packages, which are difficult for amateurs to wire up. Our recommendation is the 74HC family, which is 3.3V compatible and is available in through-hole packages.

is the same as that for the OR gate, again with the 0s and 1s swapped in the output column.

The one remaining type of gate is the XOR gate, which stands for 'eXclusive OR gate', and the truth table of which appears here.

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

You'll notice that it's the same as the truth table for the OR gate, but differs in that the output is 0 when both inputs are 1. Another way of looking at its function is that its output is 1 when the inputs are different, otherwise the output is 0.

Other logic functions

The inverter and the various types of gate are the most fundamental logic components, but they're just the tip of the iceberg. Dozens of other components are available, although in reality, nearly all of their functionality could be duplicated by some combination of the basic logic components. Most of these components have symbols that are just rectangular boxes with their inputs and outputs labelled, so you'd need to consult the truth tables, which appear in their specification sheets, to understand their function.

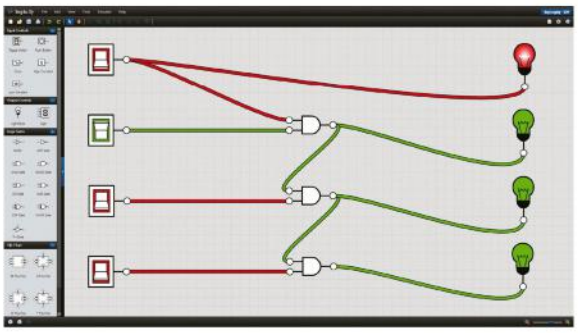
Because we're going to use it later in the step-by-step, we'll look at just one example which goes by the name 2-to-4 decoder. The truth table of one of the two the 2-to-4 decoders in the 74HC139 chip appears here (X means 'either 0 or 1').

Input 1			Output			
E	A1	A0	Y3	Y2	Y1	Y0
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0

First, the device has a so-called enable input, E. The device is only enabled if this input is at logic 0. If the device isn't enabled, all its outputs will be high. Once enabled, one of the four outputs will go to logic 0 depending on the binary number on the inputs A0 and A1. So, for example, inputs of 1 and 0 (binary 10 = decimal 2) will result in a logic 0 on output Y2.

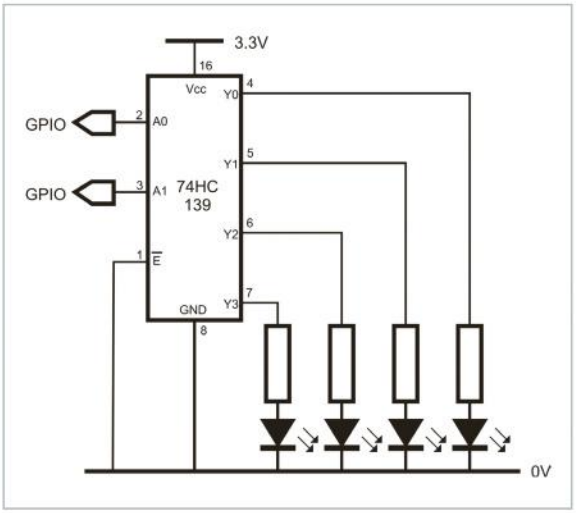
Work with logic gates to use fewer GPIO pins

Logic circuitry can allow you to connect more devices to the Pi's limited GPIO pins



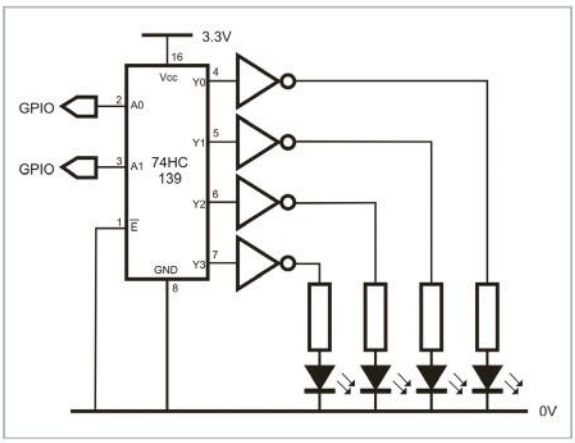
01 Use a logic simulator

Before looking at some examples of logic circuitry, here's a tip to help you check your ideas out without wiring anything up at all. If you're not quite sure which logic devices you need, this will allow you to be sure before placing an order for components. The secret is to use a logic simulator and there are lots to choose from, some that run locally under various operating systems and some that run online. Here we're testing the circuit from Step 6 at **logic.ly/demo**.



02 Use a 2-to-4 decoder

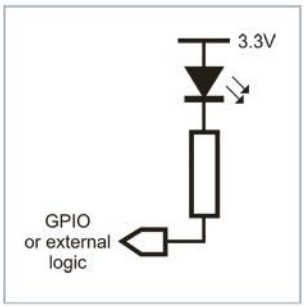
Driving four LEDs usually requires four GPIO pins. However, if we have an application where only one of them needs to be illuminated at any one time, as might be the case if the LEDs were indicating a status, we can make do with just two GPIO pins. This is achieved using a 2-to-4 decoder. As we've already seen, this outputs a logic 0 on one of its four outputs depending on the binary value on its inputs. A 74HC139 chip contains two 2-to-4 decoders and the diagram shows one driving four LEDs.



03 Invert the outputs

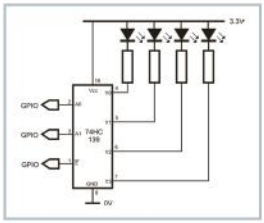
The previous diagram shows how a 2-to-4 decoder can be connected to two GPIO pins and provides four signals. However, the outputs are 'active low', so if LEDs were wired directly to the outputs, all would be lit except for the one represented by the binary input value. One way to have just the one LED lit at any one time is to use inverters, as shown in the diagram. A 74HC04 chip contains six inverters.

04 Connect LEDs to 3.3V



As a simpler alternative to inverting the outputs of logic devices with active low outputs, LEDs can be connected to 3.3V instead of 0V. We've already seen how driving a LED connected to 0V with a 3.3V signal will illuminate it and exactly the opposite is also true. The diagram shows an LED wired in this configuration and it will be lit if a logic 0 is applied to it, either directly from a GPIO pin or from an external logic device.

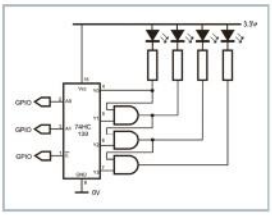
05 Implement a bar display (1)



A bar display, like those sometimes seen on audio equipment, demonstrates the flexibility of external logic. We need to light no LEDs, one LED, two LEDs and so on up to four LEDs depending on a 2-bit binary

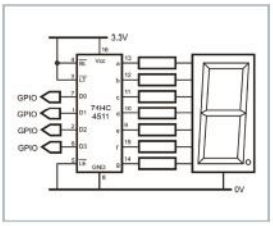
value output on GPIO pins. This uses a 74HC139 again. The diagram is the first stage in providing this functionality and you'll notice that, although similar to the circuit in Step 3, a third GPIO pin is also used to drive the enable pin so that all the LEDs can be turned off.

06 Implement a bar display (2)

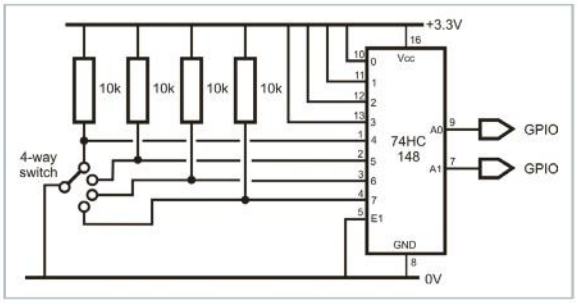


So far we have a circuit much like that in step 3, but with a means of turning off all the LEDs. The circuit in this step shows how adding three AND gates causes binary 00 to drive LED 1; binary 01 to drive LEDs 1 and 2; binary 10 to drive LEDs 1, 2 and 3; and binary 11 to drive LEDs 1, 2, 3 and 4. Looking back at the truth table for the AND gate, it should be fairly clear how this circuit achieves this.

07 Drive a seven-segment LED

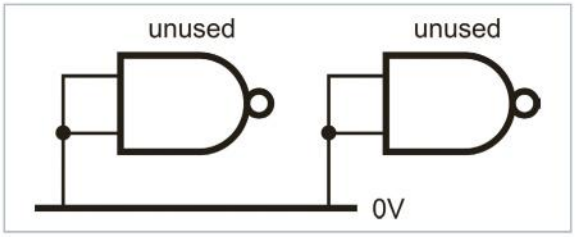


Driving a seven-segment LED without external logic requires seven GPIO pins plus an eighth if you want to drive the decimal point. This can be reduced to four or five respectively by using a 74HC4511. This seven-segment encoder turns on the appropriate LEDs in the seven-segment display depending on the 4-bit binary value on its inputs. An interesting exercise is to work out how to drive a 4-digit seven-segment display – it doesn't need four times as many GPIO pins if you multiplex them.



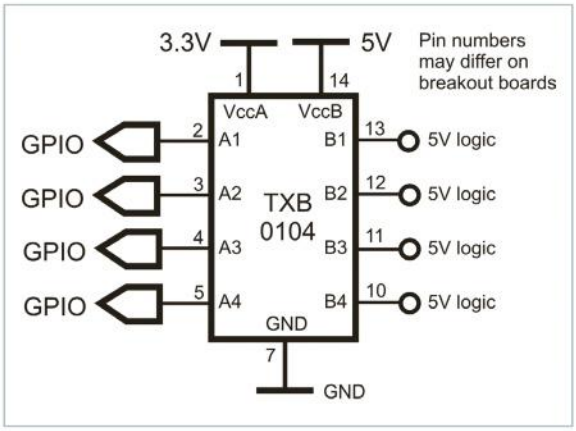
08 Encode a switch

It's not just outputs driving LEDs that can benefit from logic. By definition, a multi-way switch can only have one of its positions closed at once, so the output from a 4-way switch can be encoded as a 2-bit binary number. This is the opposite of decoding that we saw in Step 2. A chip that can do this is the 74HC148, which is actually an 8-to-3 line encoder, but we can use it as a 4-to-2 bit encoder by wiring its eight unused inputs to 3.3V.



09 Connect unused inputs

Very often when you use logic chips, only part of the chip will be used. For example, if you use a 74HC00 quad NAND gate and you use only two of the four gates, two of them will be left unused. This is not a problem, but you shouldn't leave unused inputs unconnected as they can oscillate, causing the chip to draw an excessive current and overheat. The solution is to wire any unused inputs to either 3.3V or 0V. Unused output is OK and should be left unconnected.



10 Use 5V logic

Occasionally you'll want to interface 5V logic circuitry, which you can't connect directly to your Pi. There are several level converter chips, but most are unidirectional. However, Texas Instruments' TXB0102, TXB0104 and TXB0108 chips (2, 4 and 8 channels, respectively) are bidirectional, which means that they'll work if the GPIO pins are configured as input or outputs. The Texas devices are fiddly surface-mount chips, but several companies offer the TXB0104 and TXB0108 on breakout boards which are much easier to handle.

POWER SUPPLIES

If your interface circuitry and external devices require a supply of 3.3V or 5V, it can be provided by pins on the GPIO header. Even so, if you have a significant amount of external circuitry, it would be good practice to wire capacitors between the supply and ground to avoid fluctuations to the supply that could cause the circuit to malfunction. Use a single 100µF capacitor plus several 100nF ceramic capacitors, one per IC in your circuit, and wired close to those ICs. If you need a supply other than 3.3V and 5V, the easiest solution is to use a battery or, perhaps, a pack of several 1.5V AA batteries. If you need a voltage that can't easily be obtained with batteries, or you want to create several supply voltages from one battery, you can use a component called a voltage regulator.

Exceed limits safely

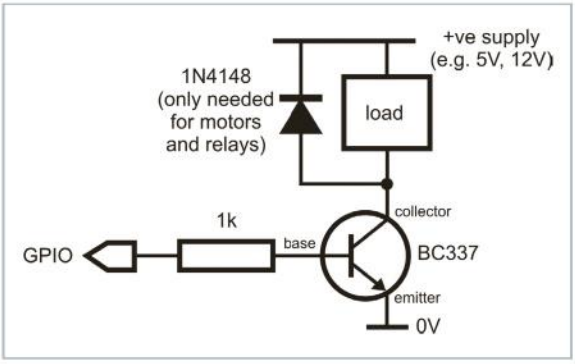
Interface devices that exceed the GPIO's maximum voltage or current rating

Interfacing an output device such as an LED to a GPIO pin requires that the device doesn't require more than 3.3V and it doesn't draw more than a GPIO pin's maximum 16mA current. Devices such as electric motors and blue or white LEDs, that require a higher voltage supply and/or draw a higher current, need special treatment.

The solution is to use a transistor, which can be thought of as an electronic switch. A small current from a GPIO pin turns the transistor on or off, thereby turning on or off a separate circuit involving a higher voltage and often a higher current than a GPIO pin can supply. This secondary circuit might use the 5V on the GPIO header as its supply but, if a higher voltage is required or the current will be higher than the GPIO header can supply, an external power supply will be needed.

The circuit diagram shows the general configuration. When 3.3V is applied to the transistor's base via the resistor, it will turn on, a condition that you can think of as the transistor's emitter being connected directly to its collector. This means that a current can flow from the high voltage supply, through the load (ie motor, blue LED etc) to 0V and so the motor will spin or the LED will illuminate.

Choosing the type of transistor and the value of the resistor is quite an involved process, especially since there is such a staggering choice of different



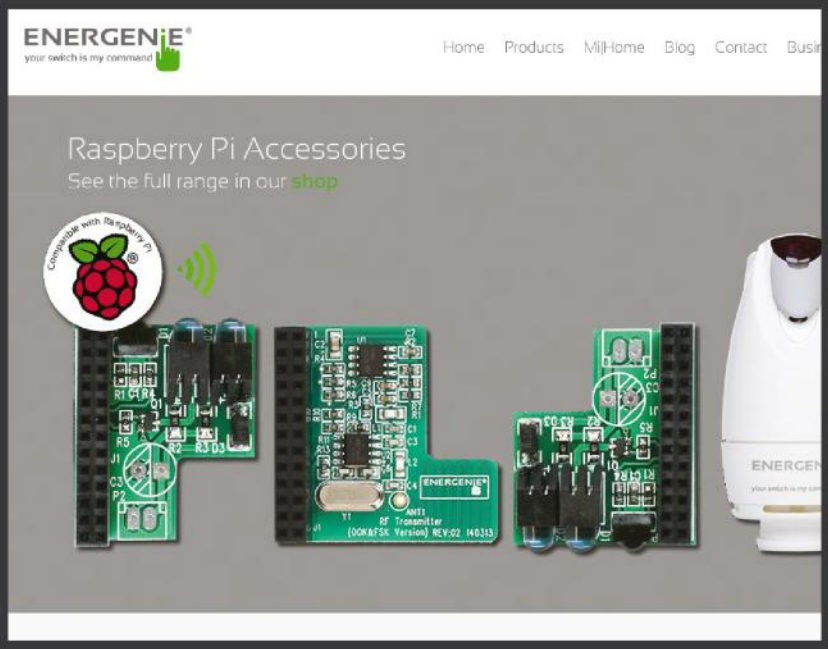
transistors. While we can't fully cover this topic here, we can give some pointers. First of all, in the circuit shown, the transistor must be of the type referred to as an NPN bipolar transistor. Transistors of this type will also be defined by their gain and the maximum current and voltage you can use on the emitter-collector circuit. Here is where it starts to get involved, but just let's say that for voltages up to 24V and currents up to 250mA (conservative limits), a BC337 would be ideal. With this type of transistor and for this current, a 1k resistor would be suitable.

Finally, if your load is a motor or a relay, it is important to wire a diode in parallel with the load (cathode to the supply) to suppress reverse currents which these components can generate and which could destroy the transistor. A 1N4148 would be suitable.

INTERFACE TO MAINS EQUIPMENT

Designing circuitry to control mains equipment isn't difficult but, if something goes wrong, you could destroy your Pi or electrocute yourself. There are HATs that you can buy for this purpose, but we don't recommend these either since the mains terminals are close to the lower-voltage circuitry. If a wire comes loose, therefore, you could set your Pi on fire, blow up a component, firing shrapnel into your face, or leave high voltages precariously close to your fingers.

For this reason, we recommend that you use off-the-shelf interfaces in which the only connection to the mains-powered devices is through a domestic-style 13A socket. Energenie (energenie4u.co.uk) offers 13A remote-controlled sockets that can be used with a radio-controlled handset rather like a TV remote control. Alternatively, they can be used with a radio transmitter module, designed for the Pi, which can also be obtained from Energenie. A starter kit comprising two sockets and one Pi interface costs £21.99 including VAT and delivery.



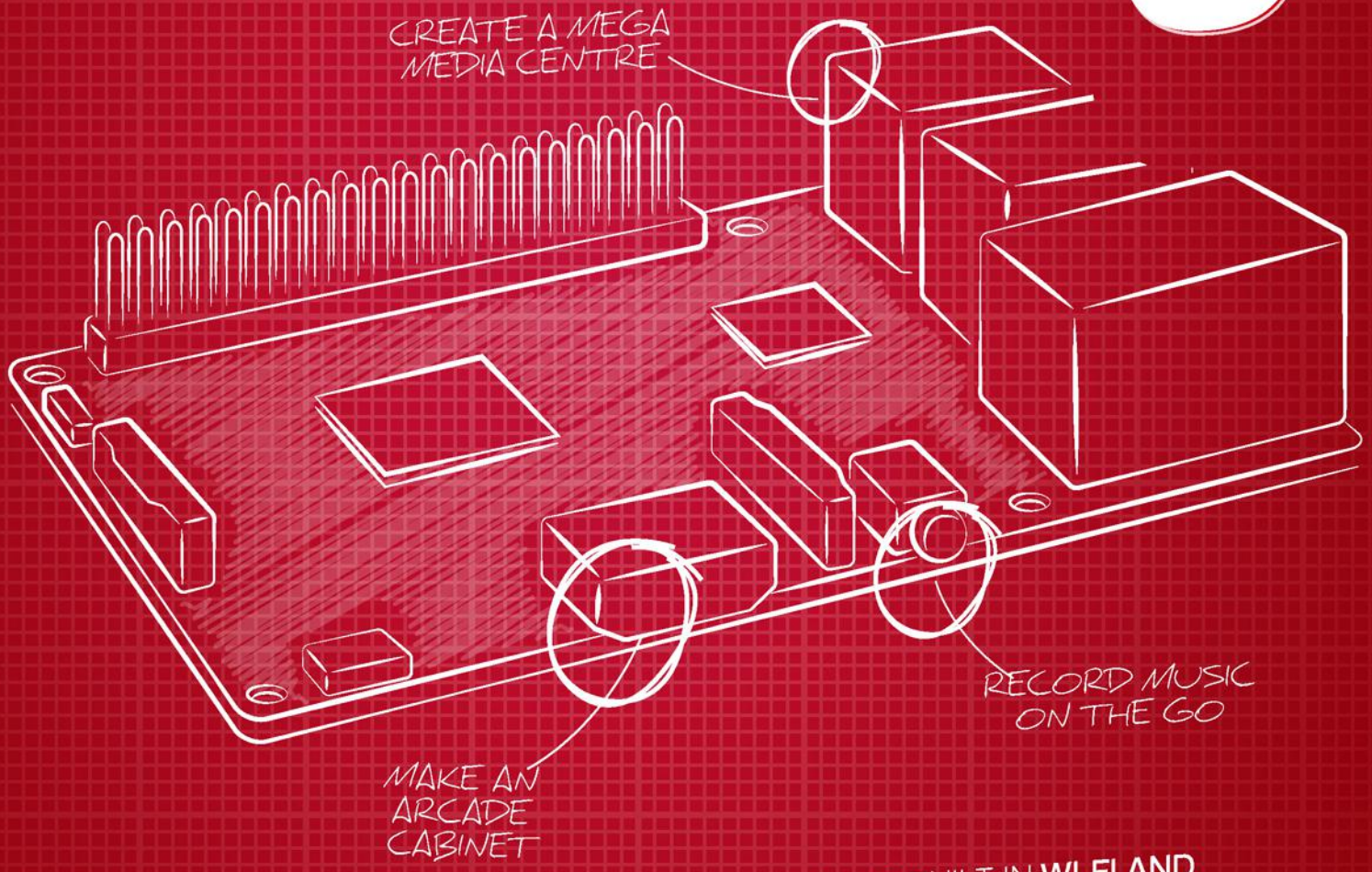
Get the UK's best-selling Linux magazine

DELIVERED DIRECT TO YOUR DOOR

Order online at **www.myfavouritemagazines.co.uk**

or find us in your nearest supermarket, newsagent or bookstore!

25 MAKER PROJECTS FOR RASPBERRY PI 3



THE PI 3 IS **50% FASTER** THAN THE PI 2 AND HAS BUILT-IN **WI-FI AND BLUETOOTH**. WE SPOKE TO THE FOUNDATION'S DIRECTOR OF HARDWARE, **JAMES ADAMS**, TO LEARN MORE, PLUS WE'VE COME UP WITH **25 EXCITING NEW PROJECTS** USING THE PI 3'S WIRELESS TECH

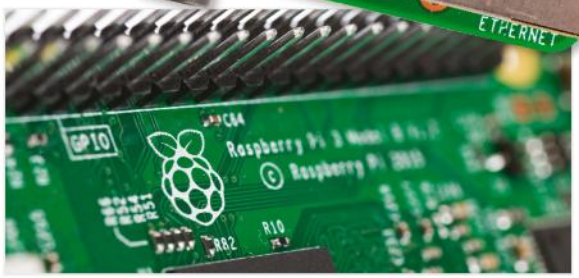
What are the big changes?

A lot of the Raspberry Pi 3 is the same as the Raspberry Pi 2. Our strategy for the Pis from the B+ has been to try and keep the same form factor. The B+ was really the first Pi that I worked on when I came to Raspberry Pi in 2013, and the idea was that we've got this nice form factor and we want to keep it, so to build the Pi 3 I took the Pi 2 design and literally just added the extra stuff.

So, we've got the new processor courtesy of Broadcom – they've done a nice uplift. It's similar to the BCM2836; the Cortex A7 was taken out and the A53s were put in, and everything else has actually stayed the same so you've got backwards compatibility. The clock speed on the GPU has been uplifted slightly from 250 to 400 MHz, and to the PCB itself we've added the Wi-Fi and Bluetooth to the top-left corner. We had to move the LEDs, unfortunately, to the bottom-left corner of the board. There have been a few other little minor tweaks, but largely it is the same as the Pi 2 with Wi-Fi and Bluetooth. That doesn't mean it was a trivial thing to get going – actually, the Bluetooth and Wi-Fi was a big engineering challenge.

How much has the speed improved with the new BCM2837?

We're saying about 50 per cent [from the Raspberry Pi 2]. In reality, you can usually see a bit more improvement than that – it really depends on what features of the processor things are taking advantage of. In everyday use, you'll get about 50 per cent faster, so about 33 per cent clock-to-clock speed increase on the core plus the uplift from 900 to 1.2 GHz. But then the NEON unit in the A53 is, I think, double the width of the one in the A7, so if you have things that take advantage of ARM NEON then they'll go quite a lot faster. I think there are some other branch-prediction cache improvements in the core that, in things like web browsing, can make more of a difference – it really depends on the workload. The A53 is just a much newer and slightly fatter core.



Left The new 1.2GHz BCM2837 contains four ARM Cortex-A53 cores

You've got 64-bit, which we're not using at the moment, but we have had people come to us with preliminary Linux kernels running in 64-bit mode. I think Eben said publicly that we're just going to monitor that, see what we think once we see that running, whether we actually do our own 64-bit OS version, which would be completely separate from the current 32-bit, or if we keep 32-bit going forward, which is nicely backwards compatible with all previous Pis.

Why was 64-bit functionality added to the new Pi?

At the point we were putting a new core into the 2837, we were considering the options. All the new ARM cores basically have 64-bit support anyway – it's a nice [thing to have as a potential]

James Adams is the director of hardware at the Raspberry Pi Foundation, where he manages the current production hardware and develops new Raspberry Pi products. He first joined the team back in 2013 and since then has led the design of the Raspberry Pi Model B+, the 2B and the new 3B. It has been whispered that James is also a demon welder and brewer of beer.

Right James is also a part of the FiveNinjas team that made a Compute Module-based media player

feature for the future – but we're very, very committed to keeping form factors and software backwards-compatible, at least as far as possible. So currently there are no plans to do a 64-bit version of Raspbian. It will require two completely separate OSs that we'll then have to manage. We'd have to recompile everything in userland to be 64-bit, so it's an awful lot of work, and then you've got the support burden as well.

Technically, it's possible, and we'll see what performance gains 64-bit Linux has, and then we'll take a view in the future as to whether it's worth it or not. Obviously, 64-bit ARMv8 is the way that ARM cores are going, and all future ARM cores will support it – so it's a thing, but we don't have an official position. We'll see what happens; I think it's a fine way to be.

A few people have noticed that the Pi 3 runs hotter than its predecessors – should people restrict its uptime or is it still fine for extended use?

The heating issue is something we can reproduce in a special case. One thing I will say is that the Raspberry Pi 2 is similar – it just takes a little bit longer to heat up. In terms of what's actually changed, the frequency has gone up a bit and we've got a slightly bigger ARM core, so that core does generate more heat. Now, the Pi 2 core still generated quite a lot of heat compared to the Pi 1, if you run it flat-out in the kind of use cases that Gareth has been playing with. But will it kill your Pi? Absolutely not. The 2837 SoC is qualified for 125°C temperature running for, I think, ten years of life – that's where they cut it off and say 'That's fine' – so we're not really worried about the hardware falling over.

The Pi will throttle the CPU speed when it reaches a hot temperature – this is set to 85°C in the firmware. We're still talking to Gareth to see why he seems to get his Pi up to a slightly hotter temperature. There is a bit of a temperature gradient across the chip – the on-die temperature sensor is in one corner of the chip – but in any case, we're not worried about it and we may update our temperature algorithms to be more accurate, given Gareth's data.

Having said that, though, these chips can get hot – especially when pushed hard, and they will throttle; they'll drop their frequency and voltage. So a Pi 2 or 3 would benefit from a heat sink to reduce the amount of throttling if you are using it in a very heavy workload continuously, whereas the Pi 1, you could run it flat-out all day and it would barely get warm. There are now four significantly fatter cores running at a much faster rate, so we're just bumping up against physics.

Your mobile phones work in the same way – they do this sprint performance, where they'll ramp the core right up to as fast as possible, your phone will get warm, and as soon as it's getting to the point where it's too hot, they'll turn the voltage down.

But for the standard 'bursty' workloads, like web browsing, the chip on a Raspberry Pi cools down really quickly if you stop doing a lot of work on it, so the board itself is quite a good heatsink. Heat goes through the bottom of the solder balls and into the ground planes of the board, and the

THERE IS NO FM

"The FM is effectively disabled," confirms James. "The Wi-Fi chip is a BGA device, so underneath you've got tiny bumps of solder and they have to solder onto the PCB. Now, on that chip they are pitched 0.4mm apart, which is actually designed for higher-tech mobile phone PCBs – higher-tech than the board that the Pi uses. One of the reasons we got the cost down is because we use low-tech PCB design, so actually, through some clever tricks, we've managed to use that higher-tech part on the lower-tech board, but at the expense of FM – too many signals to route out, and you have to resort to a higher-tech PCB."

"IT WAS A LITTLE BIT MORE OF A CHALLENGE FOR US THAN USUAL"

board's got quite a lot of metal things soldered onto it – USB connectors and such. It is important to stress that [the heat issue] is not going to affect the lifetime of the Pi at all.

The Foundation used said that implenting Wi-Fi and Bluetooth would be too too expensive. What has changed?

There are a few factors. We've always wanted to put Bluetooth and Wi-Fi on the Pi because it's a natural fit – most people who buy a laptop now will have Wi-Fi; wired Ethernet is sometimes not even supported on laptops, if you buy from Apple. So it's an obvious fit. Now, the challenge with that is obviously the price. We worked very hard on the Pi 2 building materials to work with suppliers and get some cost out of it, so we could afford to put the device on. We've managed to make a space in the costing to fit the Wi-Fi, and Broadcom's Wi-Fi solution is really great because it's quite low-cost, doesn't use a lot of external components.

The other side of it is actually that the radio design, development and conformance testing is a very big engineering task. As soon as your product starts radiating, you run into a whole big pile of conformance issues – essentially, every country wants you to do testing to be within their radio spectrum rules, and it just puts a lot more burden on the tests. So we had to work very closely with Broadcom to both set the chip into all the required test modes for the test houses, which was an awful lot of effort. And also just general RF design is harder than standard design, because if you have things like electrical crosstalk or your power supply is next to something noisy, it can really affect the radio performance; they're much, much more sensitive than general digital designs.

It was a little bit more of a challenge than usual – all in all, just quite a lot of work to put RF on something! Most of the time was spent on that – despite the fact that we've changed very little on the Pi, there is probably as much engineering effort going from Pi 2 to Pi 3 as there was from B+ to Pi 2. The nice thing is that now we've done it, we've gone quite a long way up the learning curve on how to do it again – as Raspberry Pi, we now have those engineering skills. »

Left Among the minor changes is a return to the old slide-lock microSD card slot – no more push-to-eject

U.FL ANTENNA

"Rather than pumping it into the aerial and spraying the RF into the air," James explains to us, "you plug a cable into the tester. For that, you need to have some kind of connector on it. There's a tiny shorting resistor you connect to the U.FL connector, and you can then connect an antenna or, in our case, all the test equipment we needed. I would say that if you want to use the U.FL antenna, don't – the FCC don't like it. But, theoretically, you could scrape the soldering mask off, solder a U.FL on, and connect an antenna."

25 MAKER PROJECTS

TAKE ADVANTAGE OF YOUR ON-BOARD WI-FI AND BLUETOOTH WITH THESE NEW AND UPGRADED PROJECT IDEAS

1

Wildlife camera

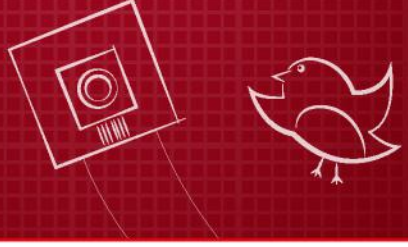
While you can pick up a kit for this, like the one from Naturebytes (**bit.ly/1PdUNue**), it's easy to put your own wildlife camera together. Make sure that you have a waterproof case to protect your Pi, the portable power supply and your other electrical components against any damage from rainwater; there are 3D models (STL files) available, which you can easily tweak to fit the Pi 3 using a simple program like 123D Design. Then you just need to use waterproof sealing on any joints and on the hole through which the camera module points – check out this guide: **bit.ly/1mgUA2X.**

Parts

- Waterproof case with a camera mount
- Portable power pack
- Camera module
- PIR sensor
- Real-time clock

You can set your Pi up to shoot during particular times by triggering a Python script with a cron job – just add a real-time clock module via your GPIO header so the Pi knows what time it is while offline. If your Pi is within reach of your Wi-Fi, it can automatically upload or tweet photos for you. If you would prefer your camera to only shoot when there is an animal to capture, set up a PIR (passive infrared sensor) to monitor for changes to the infrared picture it is seeing, which would indicate a moving source of (body) heat.

With the Pi 3, you can hide your camera enclosure away in some really tricky hiding places (buried in bushes, tucked up inside trees...) and still have easy access to it via the on-board Wi-Fi and SSH if you need to tweak your scripts.



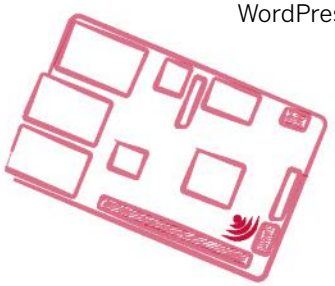
2

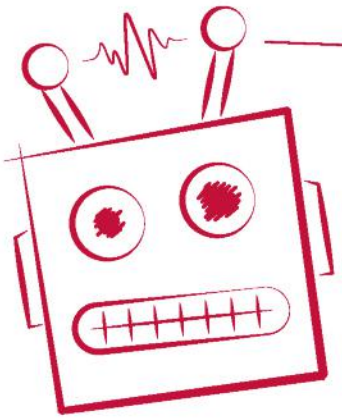
Weather station

Parts

- DHT22 sensor
- BMP085 sensor
- TGS 2600
- UV1-01 sensor
- LDR (light dependent resistor)
- ADC (analogue-to-digital convertor)
- Wind vane
- Anemometer

While a waterproof case is also a good idea here, some of the sensors would benefit from being outside the case, like the temperature sensor – there are waterproofed variants available for some of them. You could always use a mix of sensors, so you can take an average of the readings and also have some built-in redundancy. The idea here is to connect up lots of sensors via the GPIOs in order to get a reading of the weather state in your area. With the Wi-Fi and portable power pack, you can have the weather station in your garden connected to your home router (or a Wi-Fi repeater!) for an internet connection, and you can post all of the data onto an easily-accessed WordPress site, hosted on the Pi.



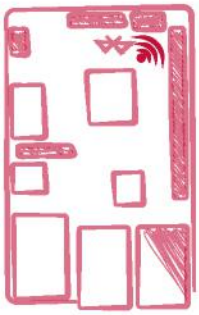


3

Personal robot butler

Parts

- Wi-Fi dongle
- Bluetooth speakers
- Bluetooth headset



Now, we've all been building Raspberry Pi robots for a while now – but with on-board Bluetooth and Wi-Fi? For a start, you don't need a Wi-Fi dongle in order to access your robot and make any changes to the control script on the fly, meaning that you can reduce the footprint of your design. And if you do use a dongle then you could even drive your robot around to use as a mobile Wi-Fi signal repeater! By setting up the BlueZ stack, you could also use your robot to control Bluetooth LE devices – perhaps mounting a pair of speakers on the back of your bot – as well as use Bluetooth devices such as the Wiimote to control movement (see **bit.ly/1RRh4Qu**). Get yourself a Bluetooth headset and you could even set up a voice-controlled application using something like Jasper (**http://jasperproject.github.io**), which listens out for your spoken commands.

Borrow a screen

Connect to a PC via Ethernet and use Remote Desktop. You still have internet access via the Wi-Fi.

"YOU COULD ALSO USE YOUR ROBOT TO CONTROL YOUR DEVICES"

4 3D printer

You can control a 3D printer using the Pi 3, and the extra horsepower means that you can handle the slicing on the Pi as well. First of all, you'll need to either set up a bought kit or look into the RepRap project – RepRaps are 3D-printable 3D printers, so you just need to find a friend with a machine or a high street 3D printer in order to run off the parts, then pick up the non-printable components (start here: **http://reprap.org/wiki/RepRapPro_Huxley**).

With your 3D printer set up, install the OctoPi image onto your SD card and then use the included software, such as OctoPrint and CuraEngine, to handle your G-Code and slicing, driving it all via Wi-Fi. You could even set up the camera module inside the 3D printer and record time-lapse footage of your prints, though you'll want a powered external hard drive as well to store the video files.

Parts

- 3D printer
- Camera module
- Powered external hard drive

5 Wi-Fi hotspot

Set up your Pi as a wireless access point for other devices, which assigns IP addresses. Connect it to the internet via ethernet, and it will act as a bridge for the devices connecting via its Wi-Fi module. If you're trying to get a connection down into your garden, use a portable power pack and plug in a USB Wi-Fi dongle. You then use the on-board Wi-Fi to connect to your home router and the plug-in Wi-Fi to broadcast into your shed.

Parts

- Portable power pack
- Wi-Fi dongle

6 Super server

The Raspberry Pi 3 can keep up with huge amounts of page requests – the Pi Foundation ran an experiment during the Pi 3 launch to see if it could keep up with the huge number of visitors (**http://bit.ly/22mCjRR**), and over 12 hours it served 1.5 million requests. So, even if your website is huge, you can serve it from your Pi, and now it's got Wi-Fi you can configure it remotely while it's tucked away in a drawer somewhere.

7 Mega media centre

Get your monitor, speakers, keyboard and mouse plugged in, then install OSMC – this is your main home theatre PC. Configure a Bluetooth remote control, or a remote control app on your smartphone or tablet. Connect a powered external hard drive to your Raspberry Pi, loaded up with all your favourite music, movies and TV shows. Set up your OSMC media centre by adding the content stored on your external hard drive, and add new files from USB flash drives as and when you plug them in. Set your Pi up as a media server for your mobile devices. Finally, browse through OSMC's App Store for things like a torrent client and TV tuner.

Parts

- Home theatre peripherals
- Bluetooth remote
- Powered external hard drive

PRACTICAL GADGETS

8 PiBook

Fit a display into a 3D-printed laptop case, along with a keyboard and trackpad. Use tiny Bluetooth speakers for audio and connect to the internet via Wi-Fi. Switch to a portable power pack if you want to go mobile. Run something like Ubuntu MATE if you want to complete the look.

9 PiPad

Mount your Pi on the back of the official touchscreen display and wire it all up. Use a regular power supply, and if you want to be untethered then switch to a portable power pack. Be sure to pair it with a Bluetooth keyboard so you can get to the command line.

10 Smart TV

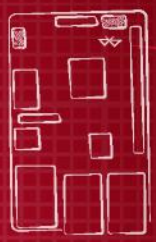
Install Kodi, then mount your Pi onto the back of your TV. Connect your TV to the Pi via HDMI, using adaptors if your TV is DVI/SVGA. Plug in powered external storage to add videos, and set yourself up with some streaming websites. Configure a Bluetooth remote or grab the Kodi app.

11 Speaker

Connect your speakers to the Pi via a Class D amplifier. Fit the setup inside a case, leaving a space for the power cable or including a portable power pack. Set up a Logitech Media Server on your Pi and control playback using the Squeezer app. Owners of a Bluetooth speaker can use BlueZ.

SMART HOME TECHNOLOGY

12 Auto-lights



Many Bluetooth LE light bulbs can be reverse-engineered to work with BlueZ – find out which library to download to work with it at **bit.ly/1UjxGq8**. You could also choose to use Bluetooth home automation switches – plug your non-smart lights into those and then control the switch itself.

13 Plant monitor

Use an SHT10 soil monitor to sense the temperature and moisture of the earth around the roots, then feed that data to a web server on your Pi that you can check via your phone. Add a portable power pack and a waterproof enclosure for the Pi, and you can leave it in the garden.

14 Temperature control

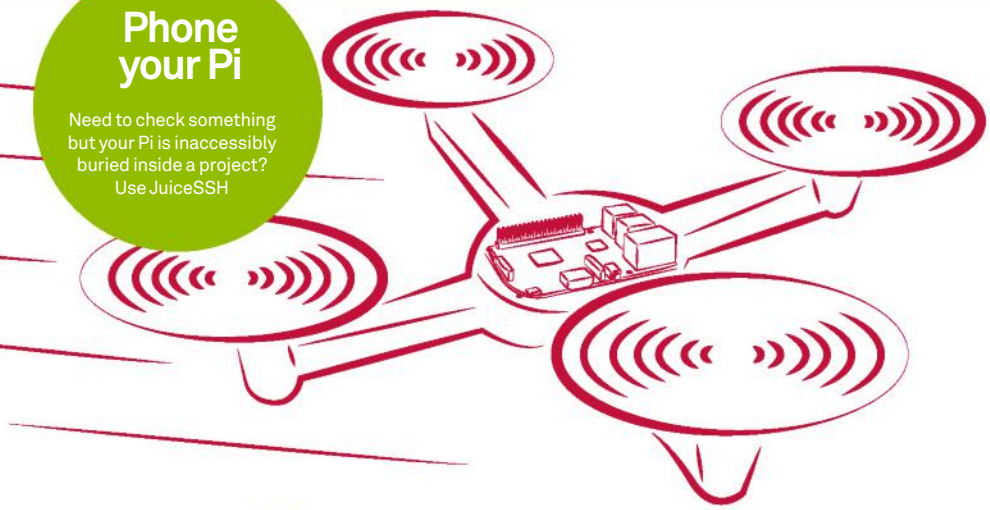
Wire sensors like the DHT22 to your Pi, stick everything inside a case, then move the unit around the house to get readings. Upload all the data to a database on your Pi's web server to access it from a web page, and then find out where the hot and cold spots are in your home.

15 RC sockets

WeMo switches plug straight into your wall socket. You can then plug your non-smart appliance into that socket, and can remotely switch it on and off by sending Bluetooth commands from your Pi over the BlueZ stack. You can then combine this with cronjobs, or a mic and PocketSphinx...

Phone your Pi

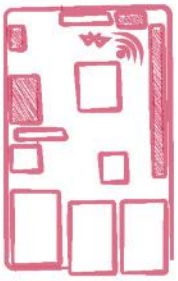
Need to check something but your Pi is inaccessibly buried inside a project? Use JuiceSSH



16 Drone

Parts

- Drone kit
- PXFmini autopilot shield



While you could build a drone from scratch, first timers might find it easier to purchase a complete drone kit. To actually power and drive the drone, you can use your Pi 3 with the new PXFmini autopilot shield. It's an excellent device designed for the Pi Zero but is perfectly compatible with the Raspberry Pi 3, and its extra CPU power gives it a significant performance boost over a Pi Zero or Pi 2 drone. The PXFmini includes a gyro and compass, accelerometer, magnetometer, pressure and temperature sensor, plus an ADC – a very handy, compact bit of kit! And with the Pi 3's on-board Wi-Fi and Bluetooth, you won't need any extra modules in order to control it while it's in the air.

17 On-location recording studio

Parts

- Cirrus Logic Audio Card
- Microphones
- Instruments

Add-ons like the early Wolfson Audio Card and the Pi-3 compatible Cirrus Logic Audio Card enable you to handle audio far better on your Pi. With an audio card, you can take advantage of tiny on-board mics or plug in your own to capture audio. There are also jacks for stereo line input, into which you can plug your instruments to capture high quality audio directly. You can also plug in external amplifiers or powered speakers. The power of the Pi 3 means that you'll be able to handle the recording without slowdown, and if you want to fit a display as well (or use a VNC app) then you can use Audacity to edit your audio on-location.

18 Performance kit live audio

Parts

- USB sound card
- Button input
- Class D amplifier
- Speakers

Set up your speakers using the Class D amplifier and you've got your audio output sorted. Connect your instrument via the USB sound card, and then set up the Guitarix: this is a virtual guitar amplifier that can take your raw input, add an effect, then deliver a processed stereo signal out to your speakers. If you pre-set a few different effects, you can set them up to be triggered by your GPIO-connected push-buttons, or a button panel add-on. Again, with the souped-up Pi 3, you can now handle this level of audio processing comfortably. Alternatively, you could even have the Pi automatically upload recorded tracks to your Dropbox via wireless connectivity.

19 Arcade cabinet

Parts

- Cabinet (bought/made)
- Monitor
- Speakers
- Joystick & buttons
- Gamepads

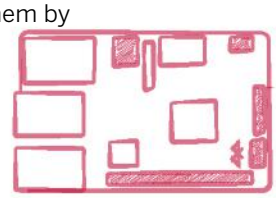
Grab an HDMI monitor (or DVI with an adaptor) and build it into a chassis for the cabinet, along with speakers. Fit the joystick and buttons, then wire them up to the GPIO pins. Install RetroPie on the Pi, add the ROMs that you own, configure the joystick and buttons, then configure any Bluetooth or USB gamepads you have for additional players. Fit the Pi into the cabinet – if you need to adjust your setup, you can either use SSH, a Bluetooth keyboard/mouse, or leave access to two USB ports for a keyboard and mouse.

20 Arcade touch table

Parts

- Coffee table (bought/made)
- Official 7" touchscreen display
- Speakers
- 6-8 Capacitive touch sensors (Pi Desk ones)

Wire up the official touchscreen via the GPIOs and configure the display for touch input. Build it into the surface of a table. Add capacitive touch sensors in a D-pad plus buttons layout and install speakers. Now follow the method used by Frederick Vandenbosch for the surface (see **bit.ly/1lwEMJ8**) – lay down a sheet of clear Plexi over your capacitive touch sensors, add a couple of layers of paper to obscure the components, then another layer of Plexi. The capacitive touch sensors should be sensitive enough for you to trigger them by placing your hands over the surface, and you can even shine LEDs through the paper. Now set up RetroPie.



21 Car entertainment

Parts

- Touchscreen display
- Back seat displays
- Speakers
- Bluetooth keyboard/mouse combo

Set up a Kodi media centre on your Raspberry Pi and add a touchscreen display to make controlling it easy while driving. Install the unit into your car inside the radio slot, and add wired or wireless speakers, since you have your on-board Wi-Fi. Check out **bit.ly/1jdL8rQ** for a little inspiration. Put a Bluetooth keyboard somewhere handy and also grab any media-loaded flash drives you have. Install a Kodi remote app onto your smartphone, set it up and then and use it to control the car stereo. You can also set up a portable wireless hotspot on your mobile, with your 3G connection, and then access your hotspot using the Pi's Wi-Fi module. For the finishing touch, run Navit with TTS output for a true sat-nav system.

PRACTICAL GADGETS

22 Smart Toys

Find a medium-to-large toy and break into it so that you can hide the Pi. Now wire up your LEDs, camera module and mini loudspeaker, then seal the toy back up again. If you run a web server using the Pi's Wi-Fi then you can write a simple control interface to activate lights and sounds.

23 Nerf shuffle

Write a simple script to shuffle your music playlist on a given input, which in this case can be a shot from a Nerf gun. Wire up a medium vibration sensor so that it can detect a hit on your target, then pass that input to your script. Fire gun, hit target, change song. Give this a read: **bit.ly/1RR9YLN**.

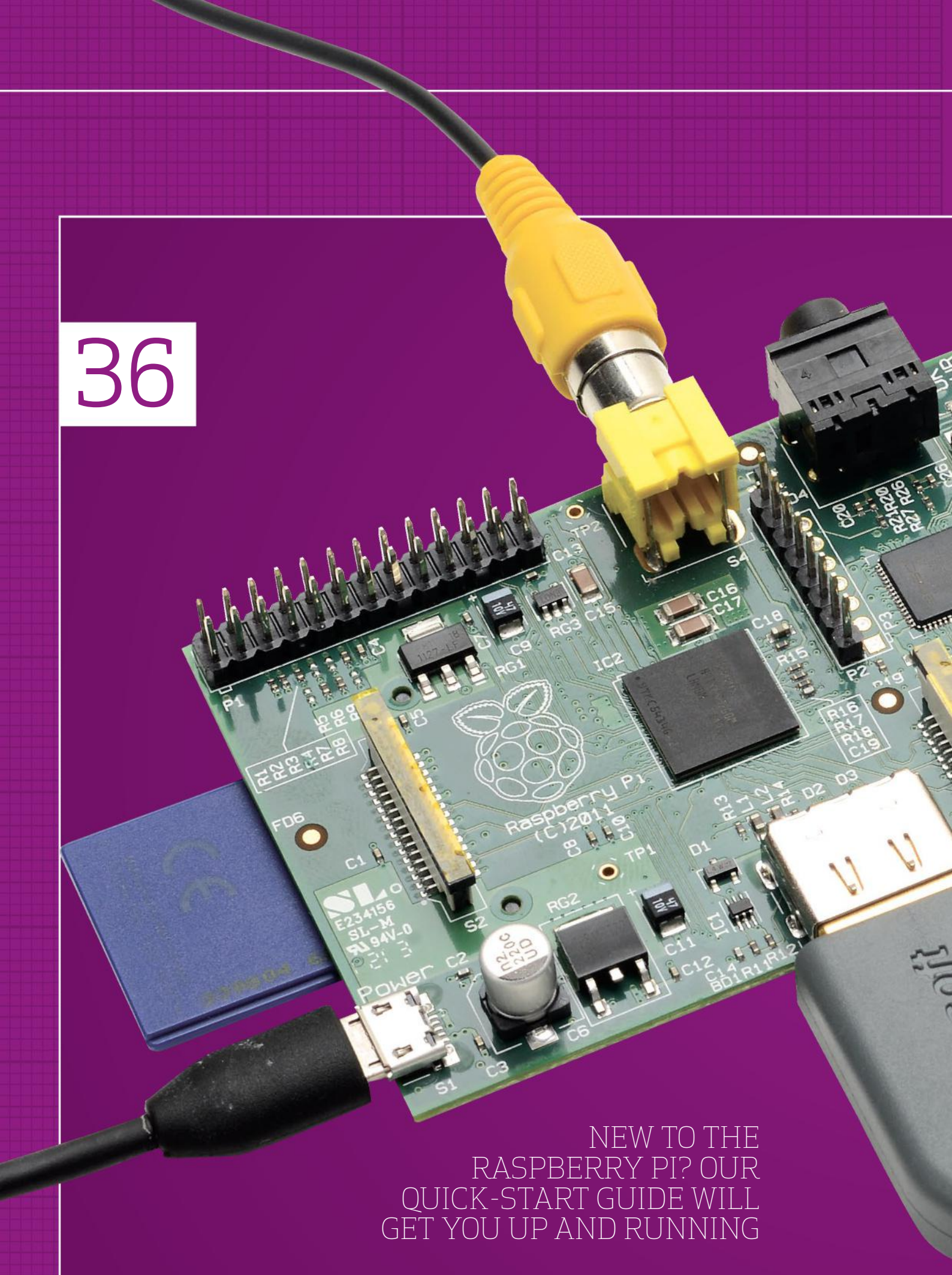
24 Wireless projector

Get something like the Brookstone Pocket Projector and connect it to your Pi, stick the setup inside an enclosure and reposition a projected display on the fly. If the resolution isn't quite there, the Pi 3 can handle a smooth VNC connection to your computer over Wi-Fi.

25 Minecraft magic wand

Connect your Pi to the Wiimote via Bluetooth and then configure it with the CWiid Python module (**bit.ly/1Wtzb2Z**). With the Wiimote set up, you can now use the Wiimote buttons to trigger pre-made Python scripts to hack your Minecraft world.

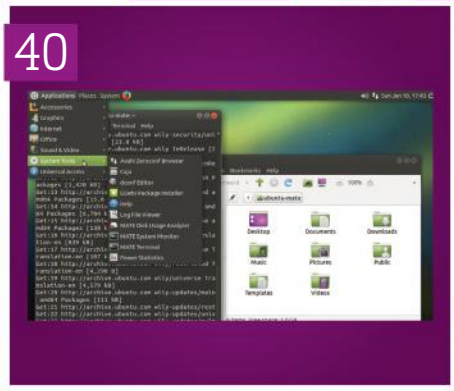
36

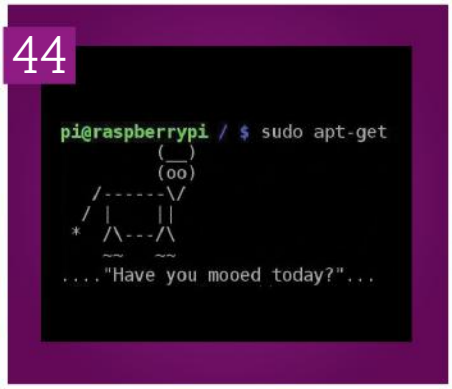


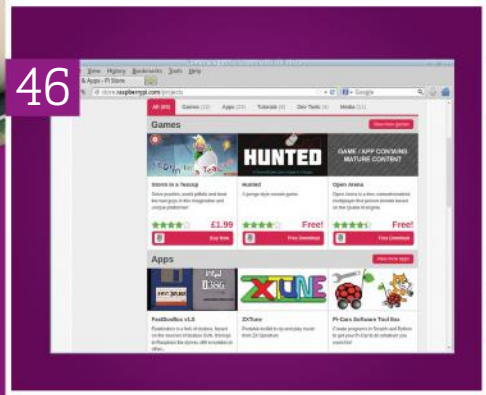
NEW TO THE RASPBERRY PI? OUR QUICK-START GUIDE WILL GET YOU UP AND RUNNING

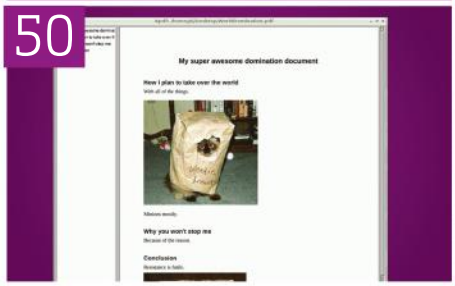
GET STARTED

Take your first steps with a Raspberry Pi...









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 electronics, such as 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

Analogue output

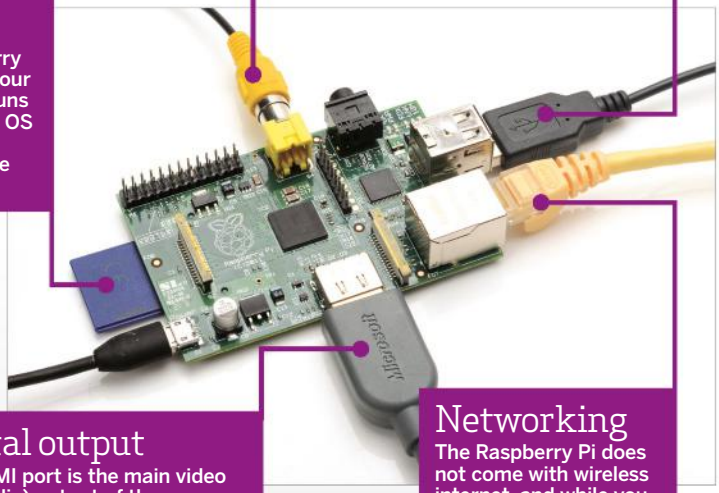
For set-ups that don't use HDMI, the yellow video out port is available. To use this with sound, you'll need to use the small black port next to it, with headphones, or an auxiliary cable to pipe out the audio

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

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



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

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

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

Connect Pi to a network

Get the Raspberry Pi up, running and connected to your computer network

Before we begin, let's make sure we've ticked off everything on our checklist. You'll need: a Raspberry Pi, 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 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.

Once finished, you'll have a fully functional 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,

1 Download software

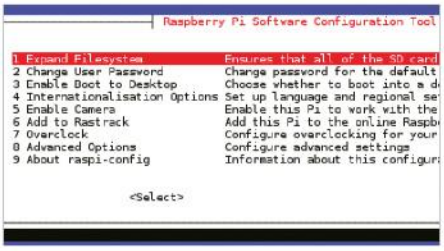
Download the latest Raspbian image, if you download the compressed zip, unzip it to a location of your choosing. Once you done this, head to **bit.ly/VOUamj** and download Win32 Disk Imager. Unzip and run Win32DiskImager.exe.

2 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.

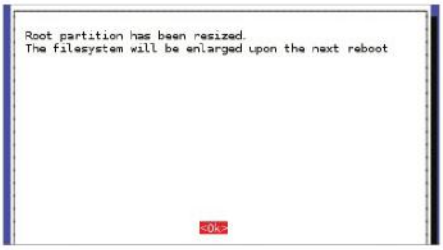
3 Connecting your Pi

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.



4 The Config tool

On boot up, 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.

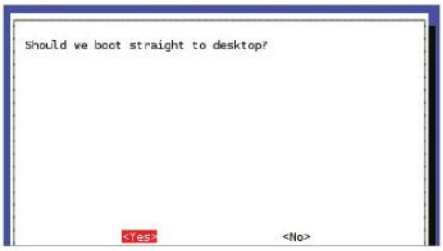


5 Expand root partition

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.

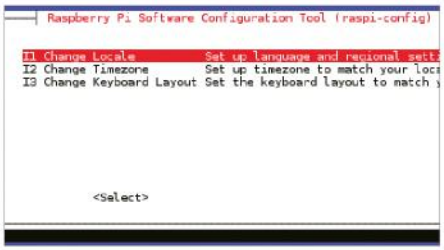
6 Change the password

The default username and password for the Pi is "pi" and "raspberry" respectively. 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.



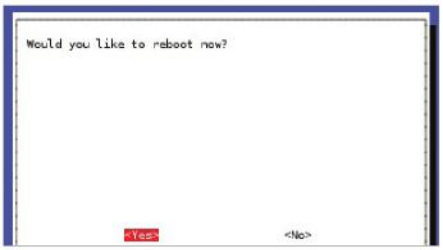
7 Boot to desktop

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.



8 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.

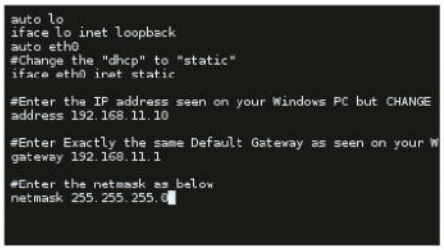


9 Finalise changes

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 this is done you'll be taken to your shiny new Raspberry Pi desktop.

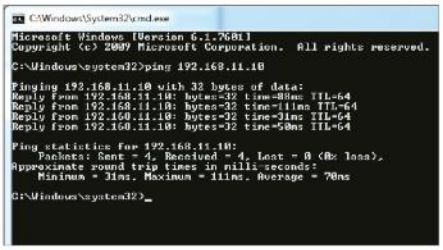
10 Finding network details

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.



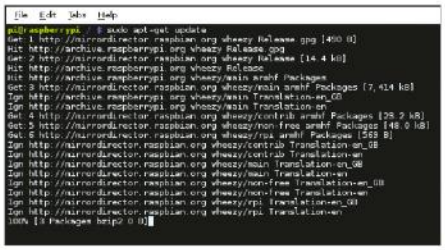
11 Network settings

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.



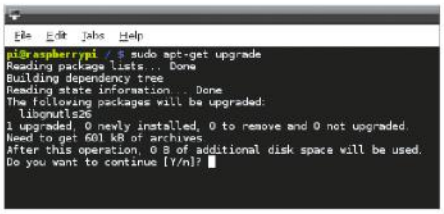
12 Test your connection

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 Update 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.

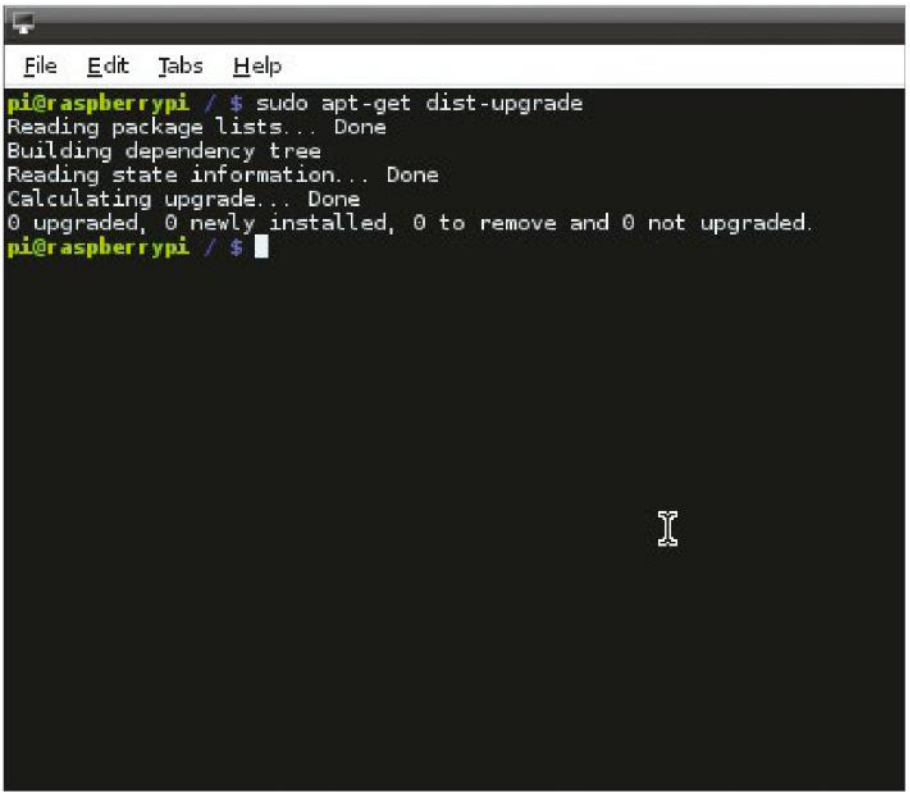


14 Update your software

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

15 Update distribution

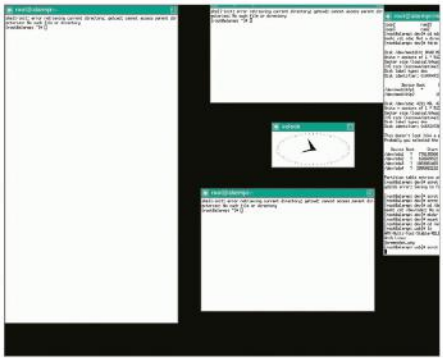
Finally, we need to make sure that our distribution is up-to-date. This will include the base operating system updates such as kernel improvements, driver updates or even some distribution specific application. Use the command sudo apt-get dist-upgrade. Again answer "y" to any installation questions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



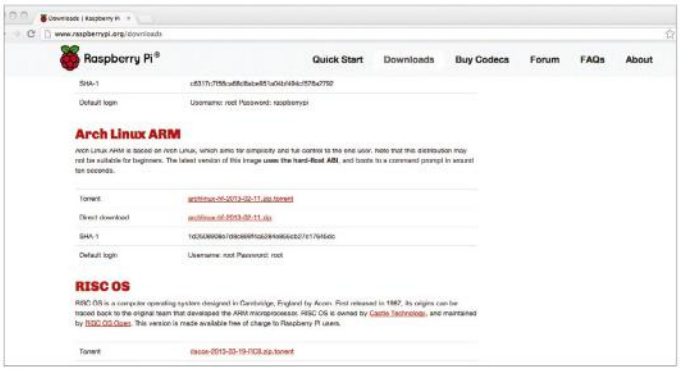
Install ArchLinux

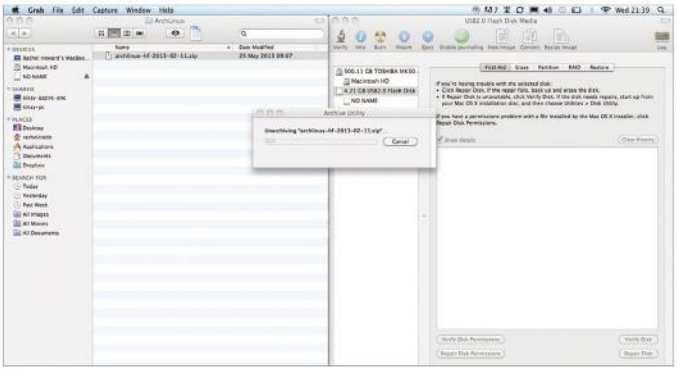
Set upl the powerful but lightweight Linux distro ArchLinux on your Pi

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.



The stripped back gui of ArchLinux allows for improved performance



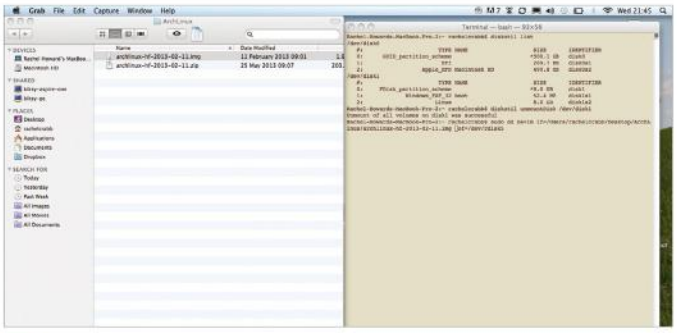


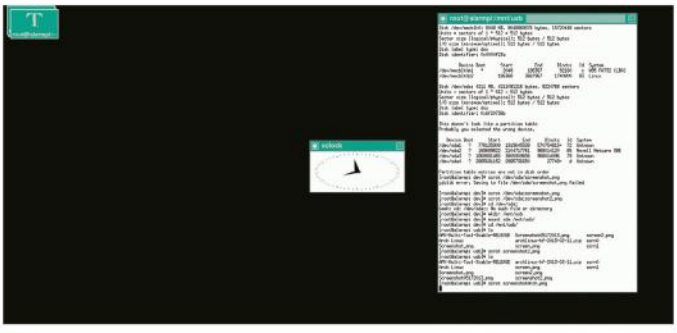
1 Download the image

The first step is the most straightforward: you need to download the latest ArchLinux operating system image from **www.raspberrypi.org/downloads**, where it will be listed under 'third party options'. The file should be around 200MB in size, so its advisable to have at least a 1GB SD card to boot from, but bigger is always better.

2 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.





3 Copy image

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

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

Once this is done, add:

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

4 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'
```

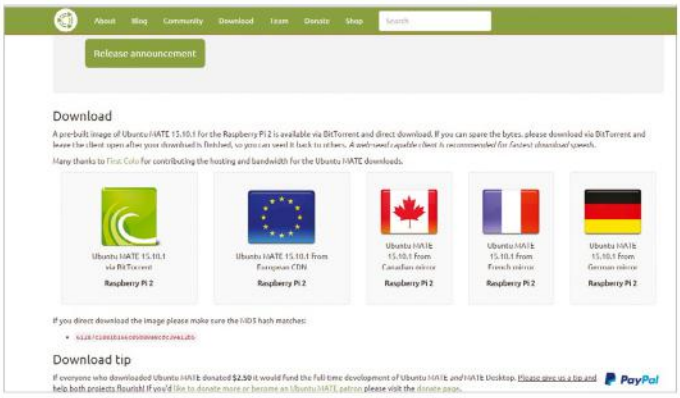
Install Ubuntu MATE

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

With the 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.

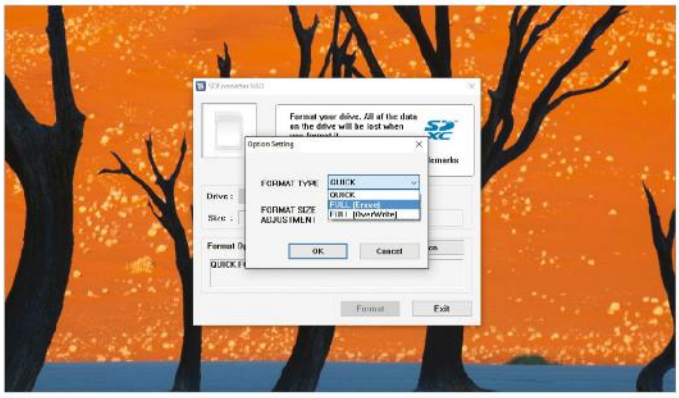
The Ubuntu MATE developers worked hard to port their Ubuntu spin to the Raspberry Pi 2, and now you can enjoy a desktop-class operating system on your single-board computer. Programs like LibreOffice are blazing fast in this distro, and it is a pleasure to use. If you're looking to explore life outside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

GrafX2 is an art package that combines immediacy with some powerful features



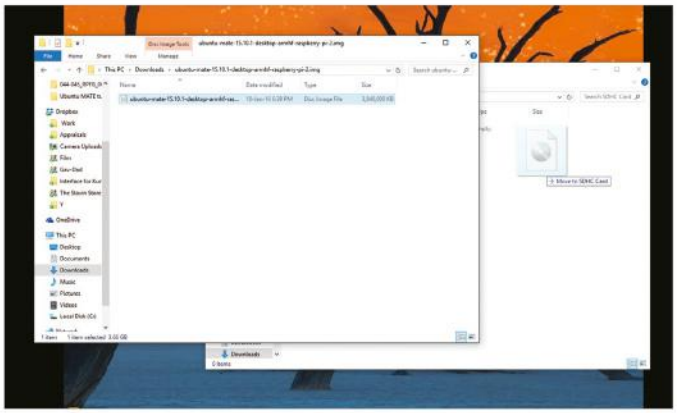
1 Download Ubuntu MATE

First, go the official project website and 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.



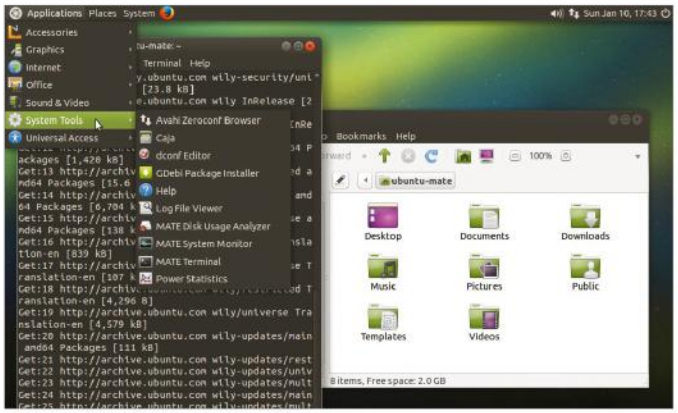
2 Prepare your SD card

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.



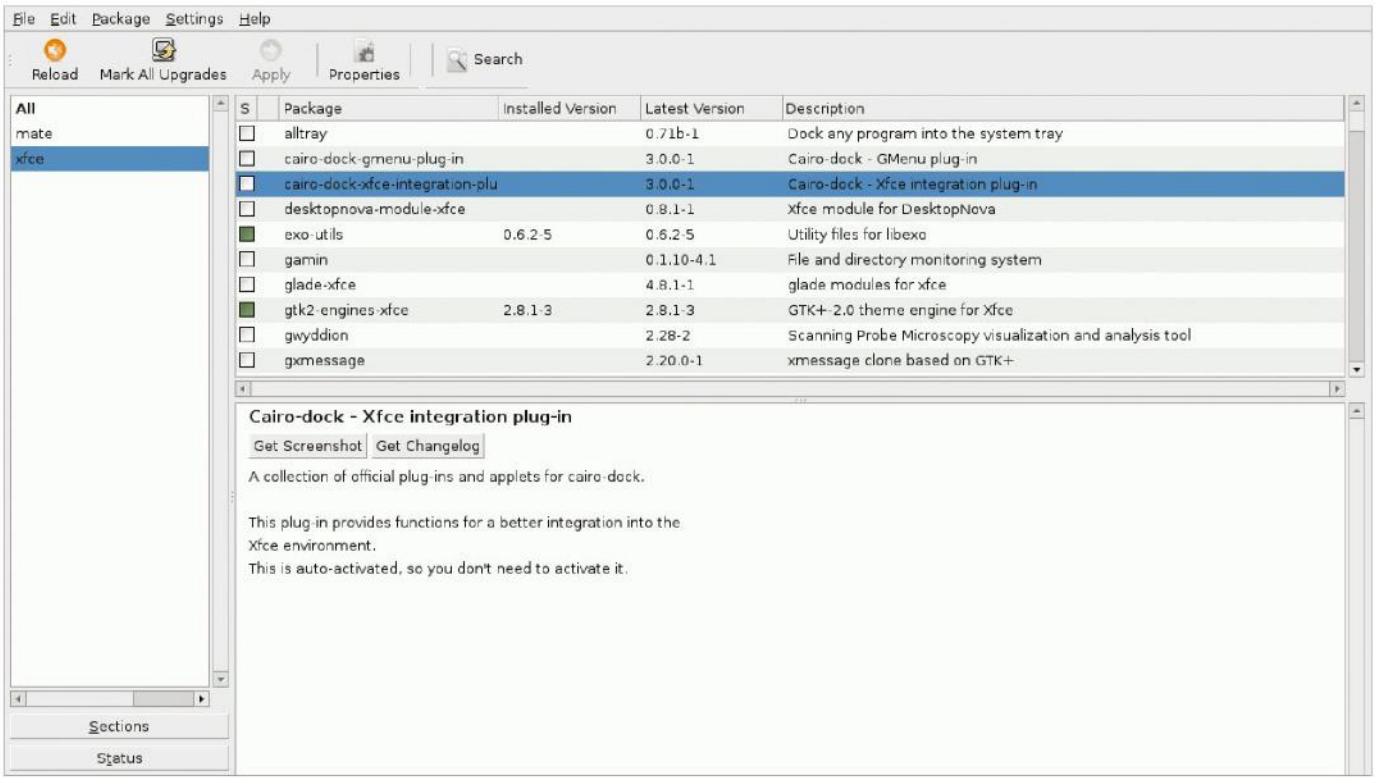
3 Copy the image

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.



4 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 power it up. The install process will automatically begin, and you'll be guided through a graphical interface. Upon completion, you'll be taken to your new desktop.



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.

1 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 improves Debian, that upgrade is also ported over to the Raspbian OS.

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 atarchive.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.

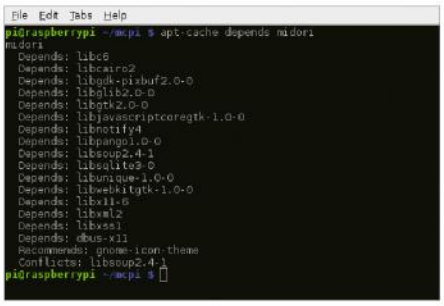


Fig 1: These are the dependencies of the package that provides the Midori web browser

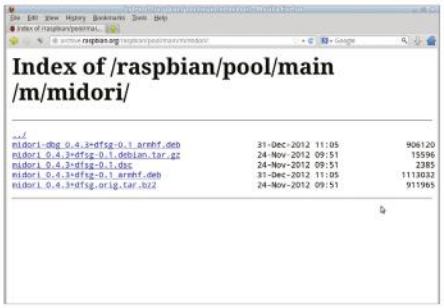


Fig 2: Browsing the Raspbian repository

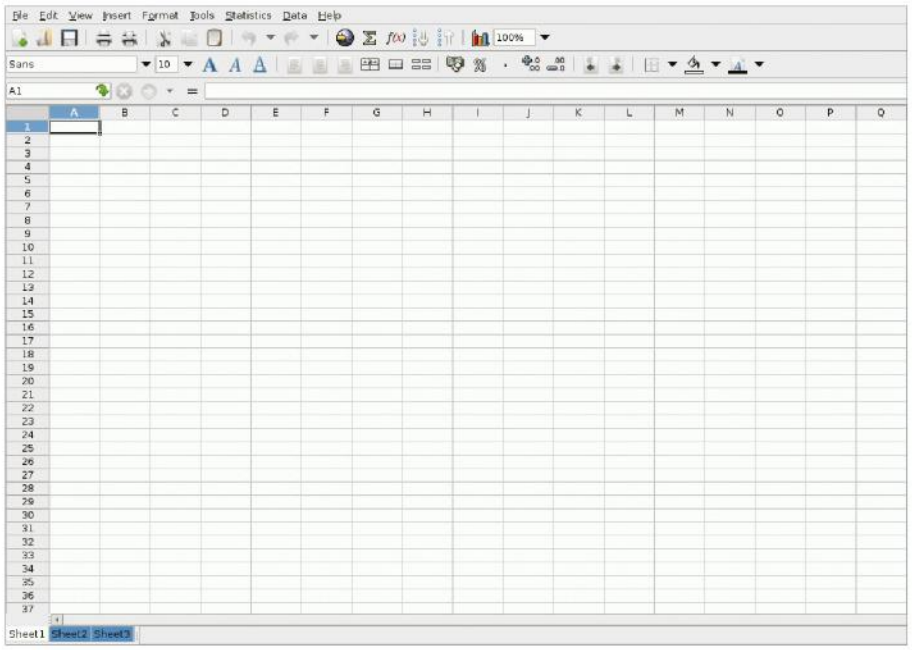


Fig 3: Gnumeric is an excellent spreadsheet application that is both comprehensive and light on resource usage

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.

2 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 repository. There are a few choices, but we're going to work with a package called Synaptic. Type sudo apt-get install synaptic to install it. Once the process is complete, launch Synaptic by typing sudo synaptic.

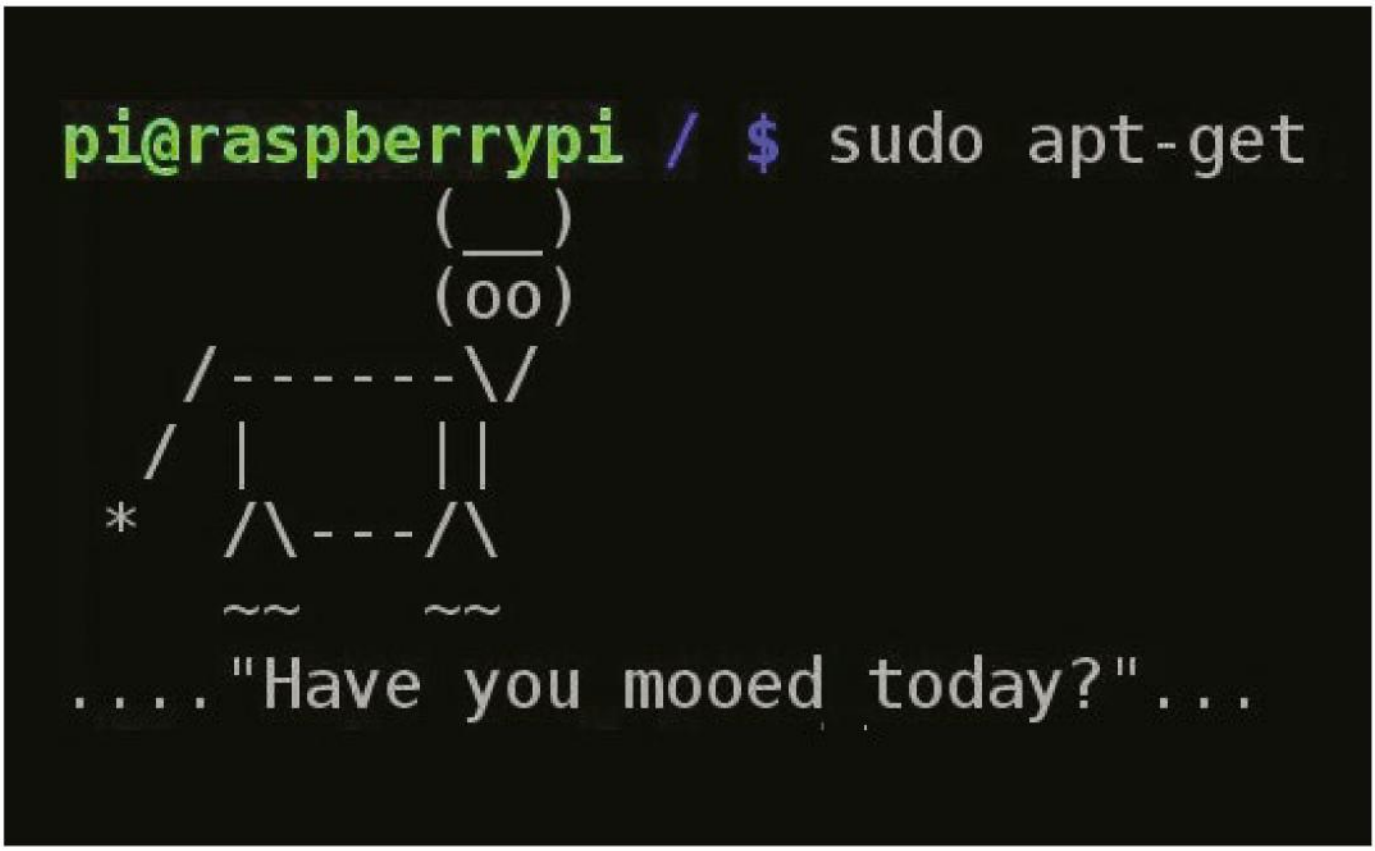
From the main image you can see that Synaptic employs a three-pane layout. The top section shows a list of packages, the sidebar has a list of categories and the main window shows a description of the currently selected package. Click on the Search icon at the top of the window and type 'word processor'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

3 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 4: The Xfce desktop is a good alternative to the default LXDE



Install and use packages on Pi

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

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.

WHAT YOU'LL LEARN

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

System update
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 group, type and even beginning with a certain letter.

Complete removal
Apt will allow you to completely remove an installed package, along with all configuration files.

1 Update and upgrade

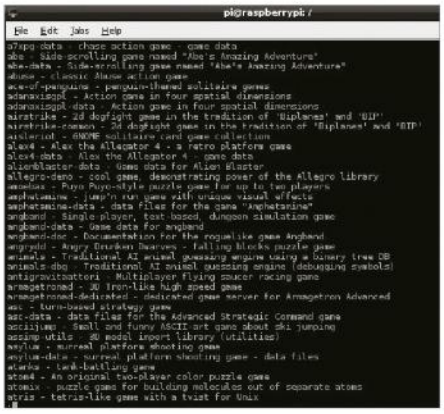
Getting hold of a package on the Pi involves dropping into the command-line terminal, via the LX Terminal icon on the desktop, and entering a few commands. But before we do that, we need to make sure the system is up to date. Enter the following into the terminal:

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

Alternatively, you can try:

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

2 Search for a package



The apt-get command (Advanced Package Tool) is the key to downloading and installing packages on the 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 games:

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

3 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.

4 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
```

5 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
```

6 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
```

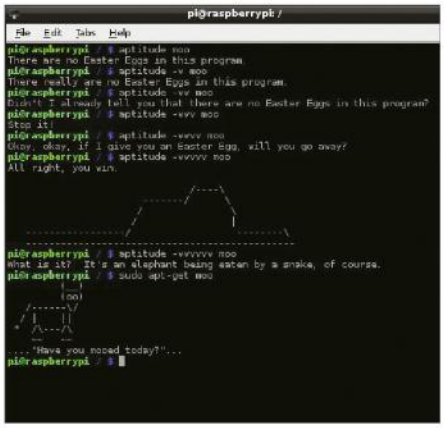
Enter 'Y' to accept the removal.

7 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
```

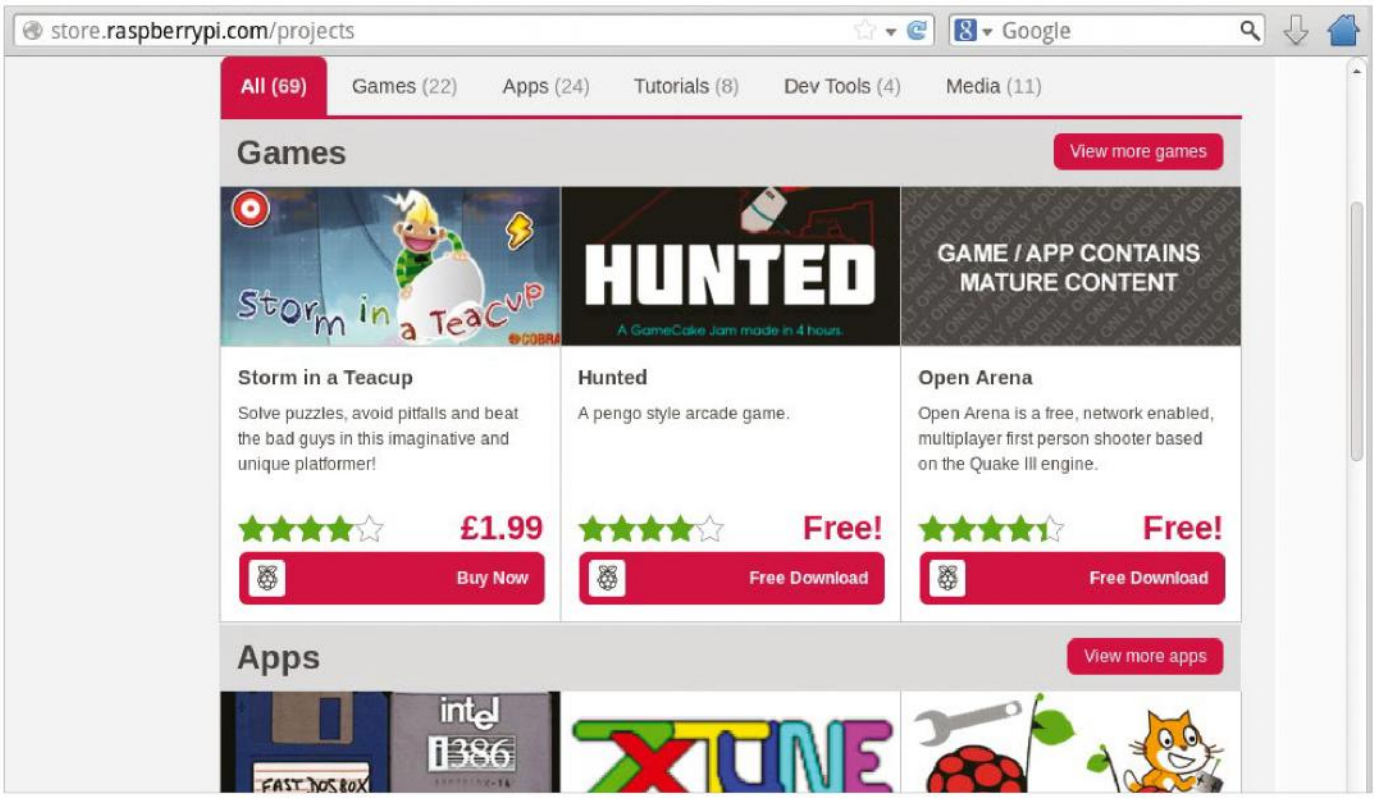
8 Man the Apt command



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:

```
man apt
```

You now know everything there is to know about Raspberry Pi packages and what you can do with them!



Discover the best apps

The apps that exemplify the best of what the system has to offer

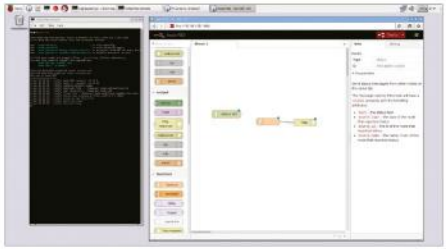
As we've said a few times, the Raspberry Pi is a Linux-based computer. From a software point of view, this means that you can run Linux programs on your Raspberry Pi – you don't need special Pi-only versions; all you need to do is check the software requirements and make sure that you meet the minimum 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 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.

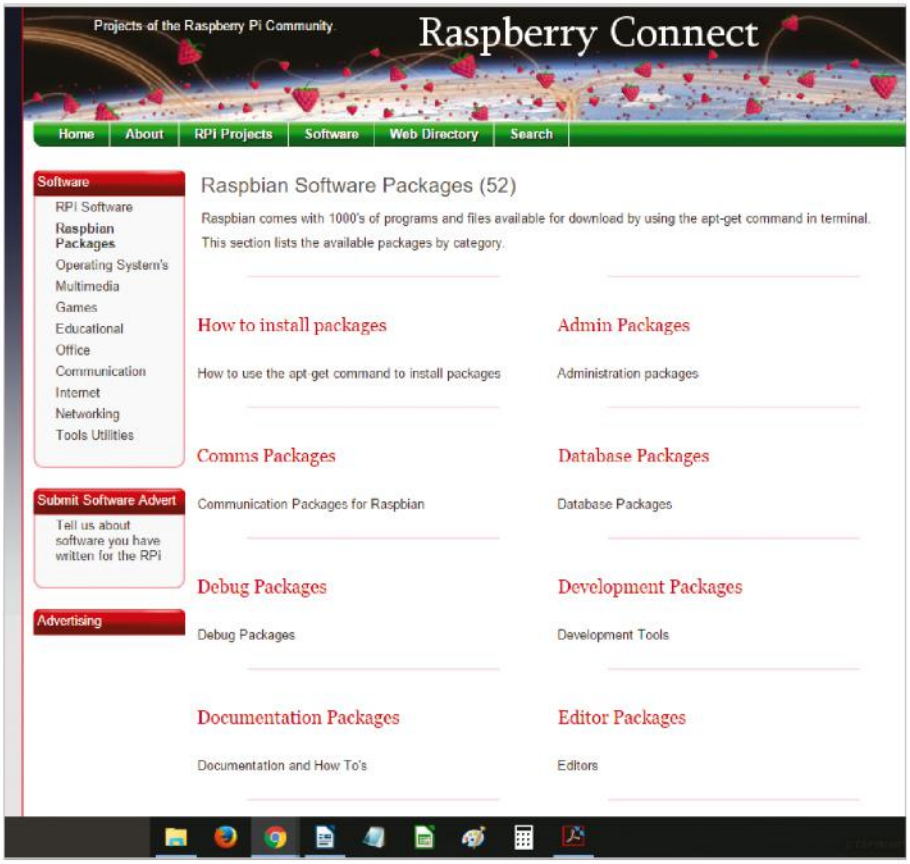


Node-RED

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 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!



FIND PACKAGES

Find out what's inside the official Raspbian repositories

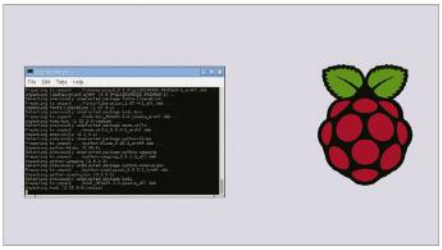
It's all well and good knowing that there are thousands of software packages available to download from the Raspbian repositories, but how do you know what to download in the first place? Well, often you'll hear about a piece of software and it's then a simple case of searching for it online – almost 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.

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

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.

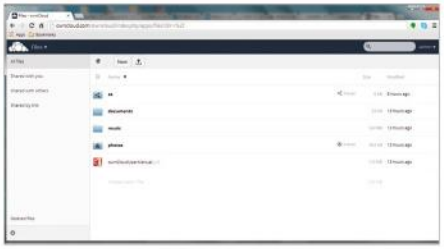


TightVNC

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 and then access the graphical interface on a PC with TightVNC client installed.

Transmission

Some people like to set up their 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 and is also highly customisable.



ownCloud

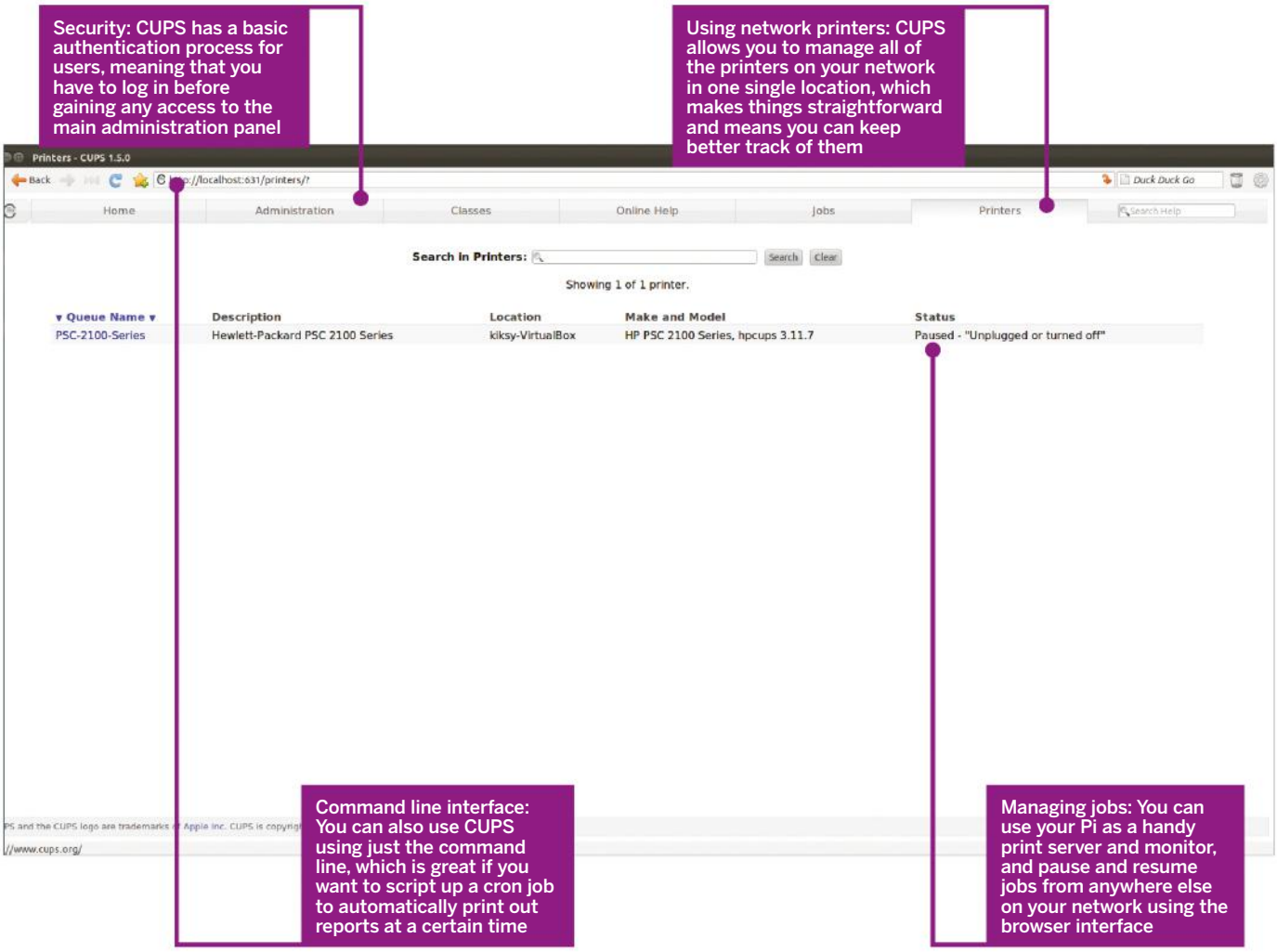
Don't trust Dropbox with your data? Give ownCloud a shot. Once you set it up, your Pi can monitor a particular folder and, when it sees any updates to the files within, it relays those changes around to all the connected devices, just like Dropbox, iCloud or OneDrive.

NagiosPi

The Pi can 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 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 is easily accessible and controllable through a web interface that you can go to in a browser.



Set up a printer on your Pi

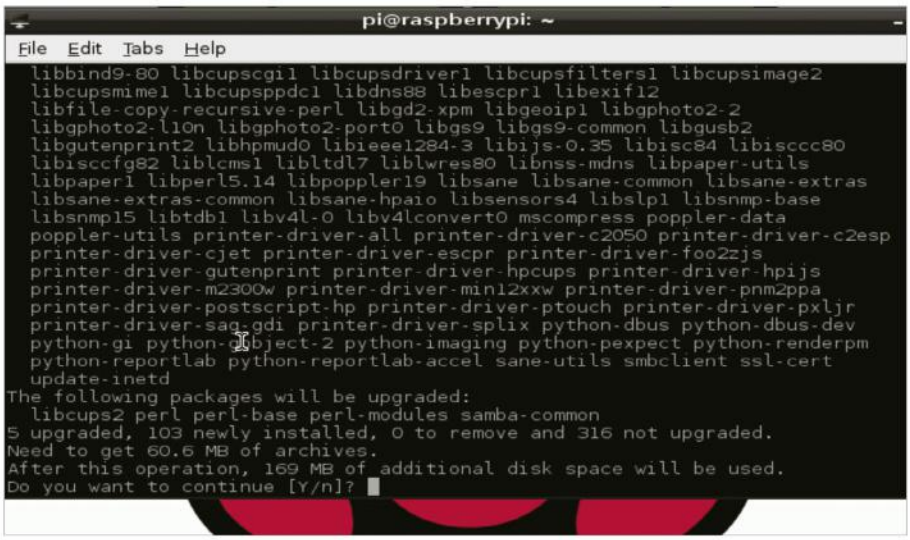
Learn how to print out documents and photos using your Raspberry Pi running Raspbian OS

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.



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

1 Get Pi ready

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

```
sudo apt-get update
```

2 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
```

3 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. This easy-to-use text editor that runs from within a terminal window. Installation is again done using APT:

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

4 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
```

5 Make accessible

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

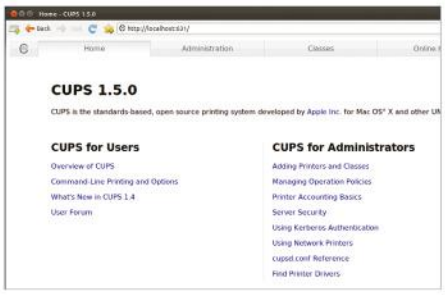
```
<Location/>
Order allow,deny
Allow localhost
Allow 172.20.22.*
</Location>
```

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

6 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. For example:

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



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

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

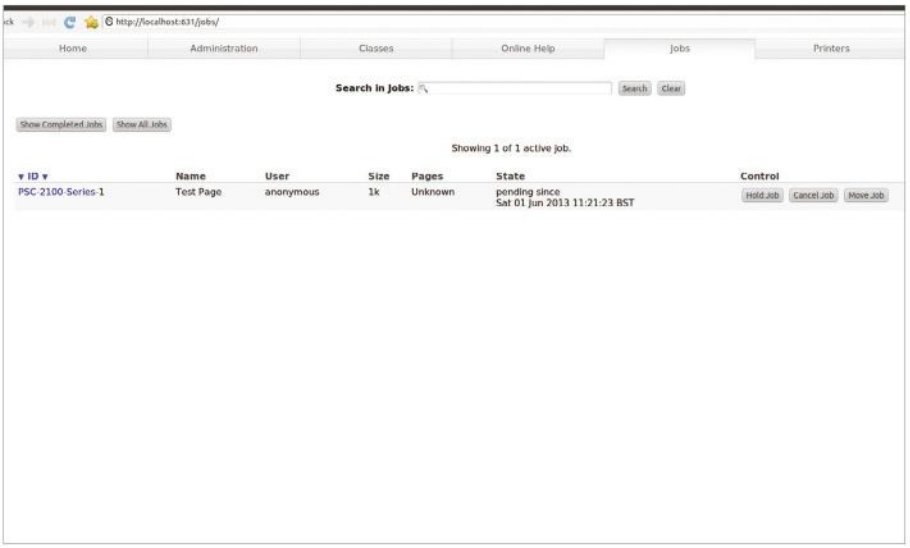
7 Test the printer

To print a test page, click on 'Administration' and then 'Manage Printers'. Click on the name of your printer to see more information about it. From the drop-down menu you should then be able to select 'Print test page'.

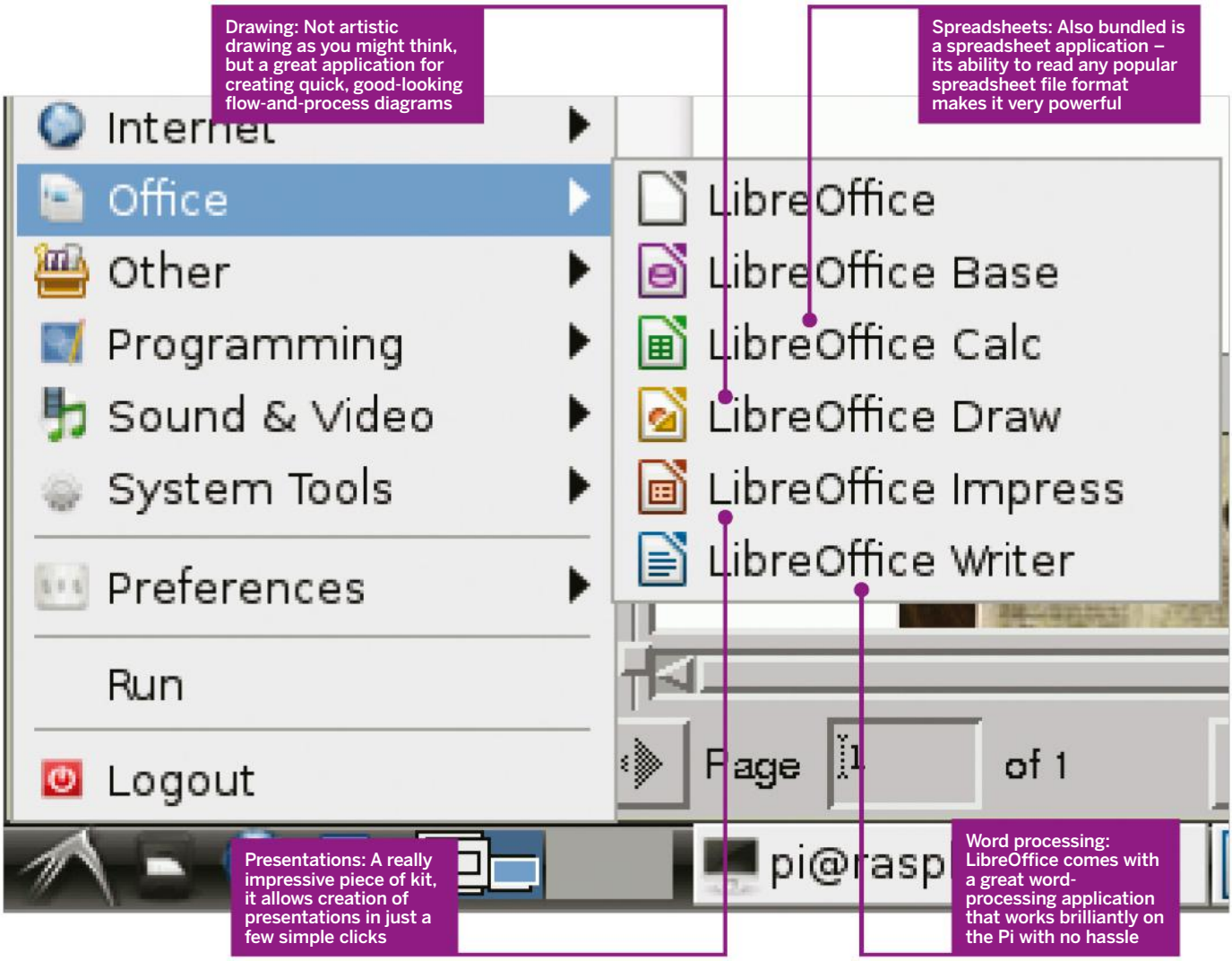
8 View jobs

After the trial, 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



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 Raspberry 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.

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

Installation and set-uo time can vary depending on your home set-up, 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!

1 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. To install the LibreOffice office suit by using the package manager, type the following into Terminal:

```
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.

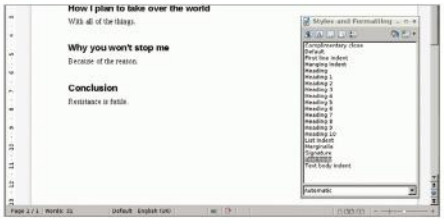
2 Explore the suite

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.

3 Add some style



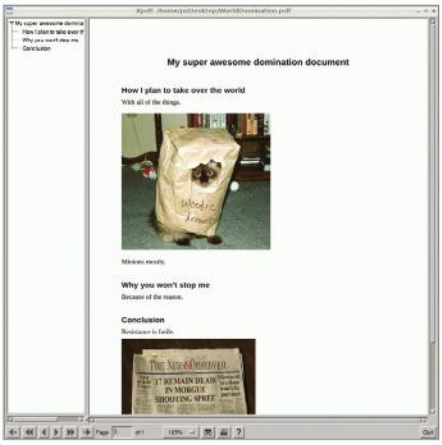
We're focusing on some of the word-processing features on the Pi specifically here, so go on ahead and open up LibreOffice Writer. When you're writing documents, it helps to put some formatting in. It helps with reading the document, and it looks prettier too. When you start writing, to the left of the 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.

4 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.

5 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. This gives you an open dialog that allows you to do exactly the same thing.

6 Export as a PDF

The PDF format is a very common way of exchanging documents because 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. It's also a great way of sharing documents without worrying that someone will accidentally make changes to a documents. It's even possible to set passwords on PDFs to stop purposeful modifications to them.

In LibreOffice, you can save your document as a PDF using the File>Export to PDF menu option. Choose your desired options, click Export and save it in the appropriate place.

7 Add comments

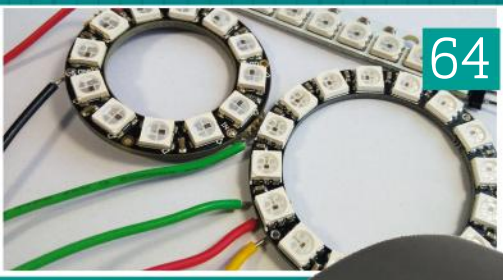
When writing a long document you'll often end up thinking of things that you should really check up on or add to the document later. When you think of these things, you can make notes about them – but physical notes aren't always that ideal. There's a little-known function in most word processors now 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.

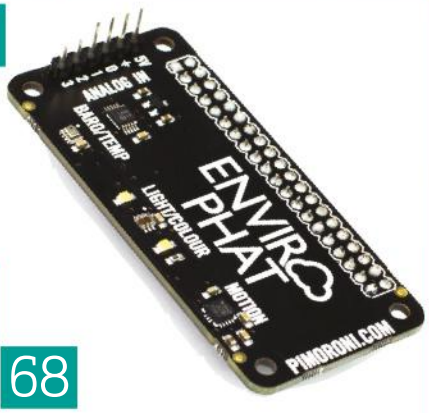
8 Add links

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 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 it the web. 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.

Pi PROJECTS

Put your Pi to use – we show you how, step-by-step!





Create a Pi-powered virtual reality setup

Combine the Raspberry Pi, Python-VRZero and 3D graphics module Pi3D to edit or make virtual reality environments

RECIPE

- Github repository
- 8GB SD card
- Xbox 360 controller
- Oculus Rift Developer Kit (optional)

Virtual reality is the next big tech revolution. It has come a long way since the concepts and CGI visuals of Stephen King's *Lawnmower Man*. In fact, VR is one of the fastest growing areas of technology and you can now design models, explore new places and play games all within a virtual environment.

But a professional VR hardware package like Oculus Rift or HTC Vive is expensive and will set you back several hundred pounds – thousands if you factor in the high-end PC sold seperately. However, it's possible to emulate the VR setup up using a Raspberry Pi, Python-VRZero and a 3D graphics module, Pi3D. Now, this is purely for fun and learning, so don't expect huge gaming PC frame rates, although some of the demos do peak at around 25-30 FPS on a Raspberry Pi 3. This tutorial shows you how you create a VR setup using the Raspberry Pi 3, a Xbox 360 controller and some Python code. This will walk you through how to install the required software and modules. We'll then cover configuring the hardware and drivers to enable you to control movement within the VR environment. The final steps look at the program code structure, where you can develop your own versions of the VR demo or design and build your own virtual worlds.

"VR IS ONE OF THE FASTEST GROWING AREAS OF TECHNOLOGY"

1 Python-VRZero

Using Python-VRZero is a frictionless way to get started creating your own virtual reality worlds in Python on a Raspberry Pi and combine an Oculus Rift. The program adds a number of features on top of Pi3D and solves the headaches of configuring a Pi 3 for VR development in Python. It includes default input event handling for keyboard, mouse, controller and the Rift for moving and altering the view of the player. It also supports Xbox 360 controller configuration and uses OpenHMD to read the rotational sensor reading from the Rift.

2 Fresh SD card install

Before you get started, it is recommended that you use a new SD card with a fresh installed image of the Raspberry Pi's official operating system, Raspbian. You can download the operating system directly from the Raspberry Pi website at **https://www.raspberrypi.org/downloads**. Install using your normal preferred method. This project setup was tested using the Pixel OS image.

3 Update the Pi

Boot up your Raspberry Pi. At this stage you do not need to connect the Xbox 360 controller or Oculus Rift. When loaded, open the LXTerminal and update and upgrade the OS typing the two lines below. This may take some time.

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

4 Install the Xbox 360 controller drivers

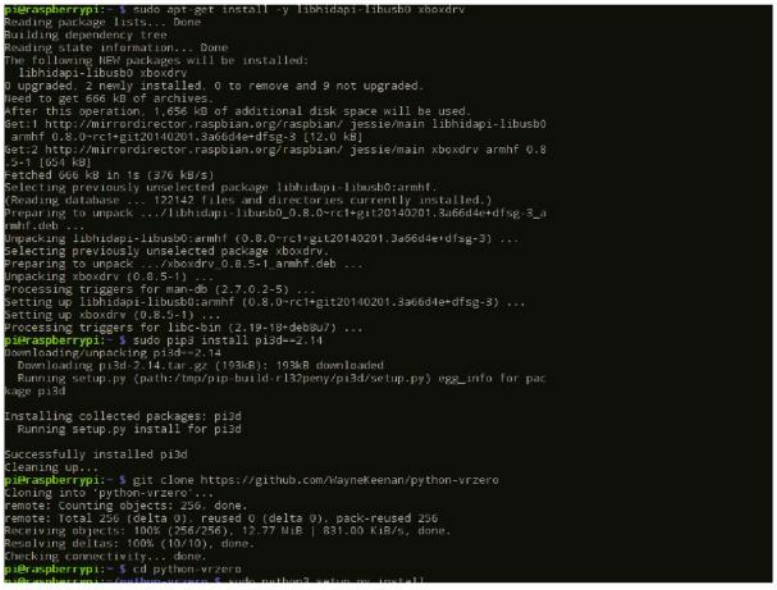
Now install the package dependencies for the Xbox 360 controller. You can keep the library up to date by adding the code --upgrade to the end of the first line. Then install Pi3D software which will render the images. Once that's complete, type the lines below and press Enter:

```
sudo apt-get install -y libhidapi-libusb0 xboxdrv
sudo pip3 install pi3d==2.14
```

5 Install the VR software – part 1

The Python-VRzero is available from the GitHub website and is easy downloaded using the git clone command, line one. Type this into your LXTerminal. Then move to the python-vrzero folder (line two) and install the program (line three). This deploys the software to interact between the hardware, 3D software and VR environment.

```
sudo git clone https://github.com/WayneKeenan/
python-vrzero
cd python-vrzero
sudo python3 setup.py install
```



6 Install the VR software – part 2

Once the installation completes, select and install the OpenHMD (line one) which enables the data from the Oculus Rift sensors to be read and worked with. Type line one into the LXTerminal and press Enter, then line two to install the required module. Enter line three and press Enter to link together all the required libraries:

```
sudo dpkg -i install/openhmd_0.0.1-1_armhf.deb
sudo apt-get install -f
sudo ldconfig
```

7 Copy over the configuration files – part 1

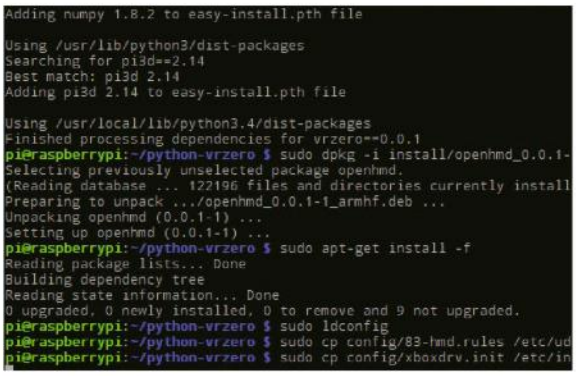
To interact with the Oculus Rift's rotation sensor and the Xbox 360 controller, you'll need to copy over the configuration files. This enables you to orientate and look around the environments. As you turn your head to the left the VR environment

Pi bites

You may be interested in trying out some other demonstrations which are availble at **https://github.com/pi3d/pi3d_demos**. Perhaps try riding a Roller Coaster or driving a tank! This resource also provides a guide how to create your own models using the Pi3D python library and code.

will adjust as if you are looking to the left. Type both of the lines below into the LXTerminal and press Enter after each line:

```
sudo cp config/83-hmd.rules /etc/udev/rules.d/
sudo cp config/xboxdrv.init /etc/init.d/xboxdrv
```



8 Copy over the configuration files part 2

The Xbox 360 controller setup requires an additional command line to copy the default configuration file to the folder which contains the Xbox Drivers. This file contains all the mapping for the buttons, paddles and joysticks. Type in the line as shown below and press Enter:

```
sudo cp config/xboxdrv.defaults /etc/default/xboxdrv
```

9 Rift Development Kit 1

If you do not have an Oculus Rift, for the purposes of testing this code you can still use a HDMI monitor to display the output. Move to Step 11. If you do own or have access to the Oculus Rift Development Kit, you will need to copy over the configuration file into the config.txt file. This file contains the configuration settings for the operating system which are loaded when you Raspberry Pi boots up. Type the line below into the LXTerminal and press Enter.

```
sudo cp config/config_DK1.txt /boot/config.txt
```

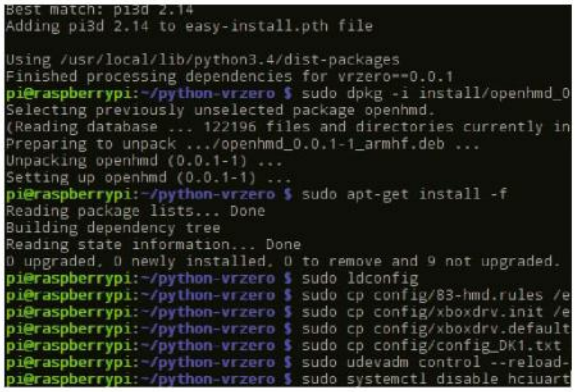
10 Rift Development Kit 2

If you have access the Oculus Rift development kit version 2, then you will again be required to copy over a configuration file. Except this time select the config_DK2.txt file and copy the contents to the boot/config file. In the LXTerminal type the line below and press enter. Ensure that you select the correct configuration for the kit version which you have.

```
sudo cp config/config_DK2.txt /boot/config.txt
```

11 Complete the set up

Finally run two commands. The first command (line one) enables the root-less USB udev config setup



which was set up earlier in Step 7. The second command (line two) disables BluetoothLE. This is required as it stops the OpenGL ES, from hanging. Ensure that each line is typed in as printed, and press Enter after each line to enable the command to run:

```
sudo udevadm control --reload-rules
sudo systemctl disable hciuart
```

12 Restart and plug in

This completes the installation and project setup. From the LXTerminal, shutdown the Raspberry Pi (line one). Attach the Xbox 360 controller and if you have one, the Oculus Rift. Turn the Rift on first before starting the Pi to ensure that it registers the hardware when the Pi boots up:

```
sudo shutdown
```

13 Hardware controls and setup

Python-VRzero sets up sensible defaults for handling input events from attached devices. The keyboard controls movement and the default mappings are: WSAD, SPACE for jump and ENTER for action. The mouse controls looking (and direction of travel when moving). Mouse button 1 is action and mouse button 2 is jump. An Xbox 360 controller controls movement and view using the left and right stick respectively. The 'A' button is 'action' and the 'B' button is jump. The OpenHMD library is used to read the HMD sensor data. VR Zero automatically rotates the Pi3D Steroscopic camera in response to HMD readings.

14 Running a demo

Now for the fun part, which is to run a demo program and immerse yourself in a VR world. There are several to choose from, and each one demonstrates the features of the Python–VRZero program. The demos need to be run using Python 3 and executed as a script from the demos folder. (If you are using a Oculus Rift you will need to navigate to the folder via the display inside the Rift headset.) Open the LX Terminal, and move to the python-vrzero/demos folder, line one. To list the available demos type ls, this will list the file names of all the demos. To run a demo type ./ followed by the name of the demo, for example to run the abbey demo type, ./abbey.py (line 2). You will

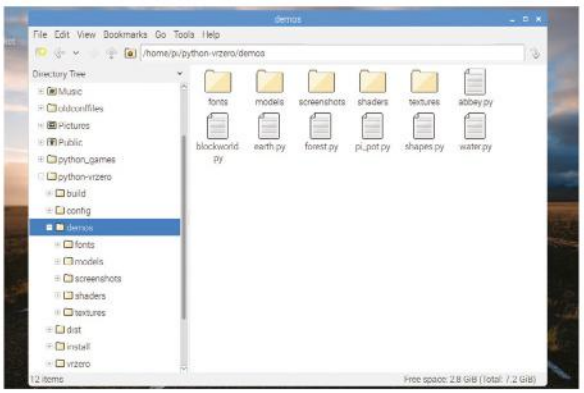
Pi bites

Python-VRzero is an ongoing development, and updates and improvement are always being added. Refer to the GitHub repository for the latest updates: **https://github.com/WayneKeenan/python-vrzero**

```
import pi3d
from vrzero import engine

# VR Zero init, must be done *before* Pi3D setup.
engine.init()

# Pi3D scene setup...

shader = pi3d.Shader("uv_light")
shinesh = pi3d.Shader("uv_reflect")
flatsh = pi3d.Shader("uv_flat")
matsh = pi3d.Shader("mat_reflect")
#########################################
# Load textures
patimg = pi3d.Texture("textures/PATRN.PNG")
coffimg = pi3d.Texture("textures/COFFEE.PNG")
shapebump = pi3d.Texture("textures/floor_nm.jpg")
shapeshine = pi3d.Texture("textures/stars.jpg")
light = pi3d.Light(lightpos=(-1.0, 0.0, 10.0), lightcol=(3.0,

#Create inbuilt shapes
```

be presented with a VR render of Buckfast Abbey, to end the environment press Escape on the keyboard.

```
cd python-vrzero/demos
./abbey.py
```



15 Editing the textures

If you have used Pi3D before you can access the program template to set up your own models. If not, then change the textures in the Shape demo program. First thing that you need to do: open the LX Terminal and type sudo idle 3 to load Python 3. Select File and open,then once this is done navigate to the following folder /home/pi/python-vrzero/demos and select the shapes.py program. Locate line 14 which begins patimg = pi3d (pictured, top of the page). This is the first line of code which tells the program which textures to load for each shape. There are several after which can also be edited.

"EXPERIMENT WITH THE PROGRAM FILES FOR EACH DEMO, EDITING THE TEXTURES"

16 Change a texture

Using the folder explorer, click the file icon to navigate to the textures in the texture folder / home/pi/python-vrzero/demos/textures. You will see the files that are used for the shapes demo. Replace the image file with one of your own and then on line 14 of the program change the file name to match your selected image file choice. If you don't want to use your own texture file then you can change the file name to one of the other image files listed in the folder. Press F5 to save and run the demo. You will notice that the shape textures have changed.

17 Alter the message

Return to your Python editor and locate line 71. This holds the message which is displayed in the shapes VR demo. Change the text to a sentence of your choice. Once you've happy with the phrasing, save and run the program. Congratulations you have now begun to modify your own VR demos! Experiment with the program files for each of the demos editing the textures. For example, how about creating a church made out of chocolate? If you're not sure where to get started and want to try other demos, you can find additional details at **https://github.com/pi3d/pi3d_demos.**

Use the Pi to take control of your drone

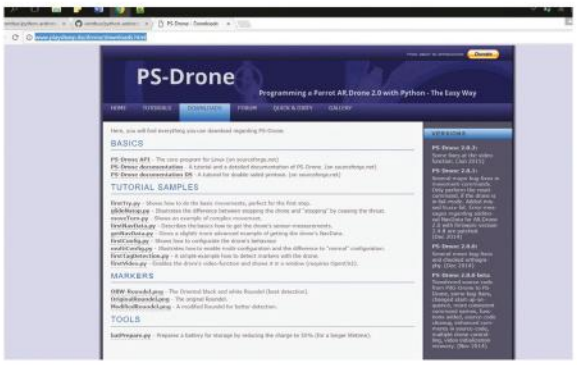
Create code which enables you to customise and automate drone flights, retrieve flight data and respond to the drone tag

RECIPE

- Wi-Fi enabled Raspberry Pi
- Parrot Drone 2.0 (or other)

Drones are becoming ever more popular and mainstream. You may already be aware that Amazon is currently working on using drones to deliver packages directly to your house within hours of you placing an order, creating a new requirement for them to be autonomous and programmable.

This tutorial offers a little taster of a Python module which enables you to write and deploy programs to your drone. The API used is now a few years old (2014) and supports operation with Python 2.7. However, it offers a number of simple methods for beginners to create their own programs and spark your curiosity.



The PS-Drone API is used to control the drone from the Raspberry Pi

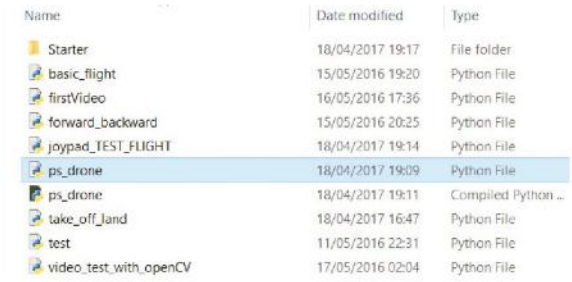
The tutorial begins with a quick walkthrough on installing the required software and libraries. Then jump straight in and create a simple but inspiring program which, when run, automatically launches the drone and then lands it. The tutorial covers the instructions to connect to the drone via the Raspberry Pi's Wi-Fi and then deploy your programs. You will then learn how to automate the drone and program a predetermined set of flight actions; for example, fly forward, turn left, turn right and then land. You can experiment with the various movements and create your own versions and flight plans.

The final section of the tutorial introduces the use of the 'tag' symbol which can be recognised by the drone's front-facing camera. Once recognised, flight data is sent back to the drone and you can use these values to trigger an action or response such as flying forward towards the tag or landing the drone.

1 Update Pi and install drone API

This project uses a Python API which enables you to access and control the drone directly with Python code. To get started, update your Raspberry Pi using the standard update and upgrade commands (sudo apt-get update, sudo apt-get upgrade). Then, using the web browser on your Pi or other computer, head over to www.playsheep.de/drone/downloads.html, right-click on the PS-Drone API program and save it.

2 Test program – part 1



The PS-Drone API file requires no installation, but it must be placed into the same folder as the Python files that you create to control your drone. Open the LXTerminal and type sudo idle to open the Python 2 programming environment. Begin by creating a simple test program which will launch the drone and then land it. Start a new file and import the time and PS-Drone module.

```
import time
import ps_drone
```

3 Test Program part 2

The next step is to initialise the drone, setting up and opening a communication connection between it and the Raspberry Pi. Begin by initialising the API, then connect to the drone via the software, starting the subprocesses. Next, add the command to launch the drone; this uses the code drone.takeoff().

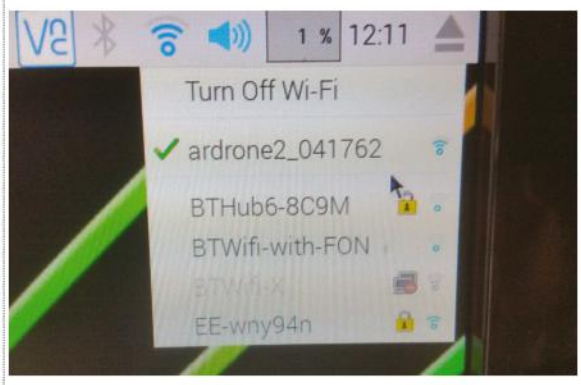
```
drone = ps_drone.Drone()
drone.startup()
drone.takeoff()
```

4 Test program – part 3

The last stage is to use the time library to add a short delay between the drone taking off and then landing. This can be used to check the connections are working by setting the delay to two seconds, which is enough time to start the rotors and stop them without the drone leaving the ground.
To land the drone, use drone.land(). Add the following two lines to your program and save it, ensuring that the file is in the same folder as the PS-Drone API file.

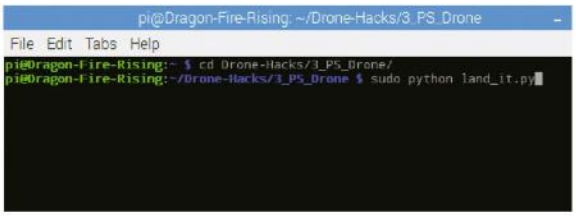
```
time.sleep(2)
drone.land()
```

5 Connect to the drone via Wi-Fi



Before the program will interact with the drone, you need to connect to it via the wireless network that it creates. Each drone has an on-board router which creates and broadcasts an open network. Power up the drone and wait for it to run the preflight checks. Success is usually indicated by four green lights. Then use the network finder on your Pi to search for the drone. Double-click to connect. Note that it requires no key or password to connect.

6 Run the program



To run the program, you'll need to execute it from the LXTerminal window. Open the Terminal and use cd to navigate to the folder where your test program is saved. To run the program, type sudo python name_of_the_file.py and press Enter. (In this tutorial, the file is named take_off_land). Ensure you have enough space to safely launch the drone. If not, then set the time value to one.

```
sudo python name_of_the_file.py
```

7 Automate the drone

Now that you have a working connection and program, adapt it to move or fly the drone. Start a new Python file and import the ps_drone. Then, as before, initialise the API and connect to the drone. Set the drone to take off, but this time set the time delay to 7.5 seconds. This provides sufficient time for the drone to take off, stabilise itself and wait for the next commands.

```
import time
import ps_drone
drone = ps_drone.Drone()
```

Pi bites

You may be interested in trying out some other demonstrations which are availble at **https://github.com/pi3d/pi3d_demos**. Perhaps try riding a Roller Coaster or driving a tank! This resource also provides a guide how to create your own models using the Pi3D python library and code.

Pi bites

You may have a different model or make of drone and there will be compatibility issues with the interfacing. However, a quick search of the internet or GitHub throws up a wide range of software for other makes and models. For example, this project page website (**http://bit.ly/DroneNav**) supports the Parrot BeBop drone and mini copter drones. It covers some exciting hacks such as image recognition, following a particular colour and streaming video.

```
drone.startup()
drone.takeoff()
time.sleep(7.5)
```

8 Fly forward

Add the code to fly the drone forward; this is simply drone.moveForward(). Then add a pause which represents the duration that you want the drone to fly forward for. In this example, the drone flies forward for two seconds. Next, stop the drone. Like a car, this requires a stopping time, so add a short time delay; you can match the delay value used when flying forward. You can test this program before moving onto the next steps.

```
drone.moveForward()
time.sleep(2)
drone.stop()
time.sleep(2)
```

9 Fly backwards

Once the drone has stopped, it will hover until it receives another command. On the next line down, add the code to fly the drone backwards. Again, you will need to add a short time delay, then stop the drone. Finally, set the code to land the drone.

```
drone.moveBackward()
time.sleep(1.5)
drone.stop()
time.sleep(2)
drone.land()
```

10 Run the program

Ensure that your Pi is connected to the drone's network, as shown in Step 5. Remember that the program needs to be executed from the Terminal window. As before, open it and use cd to navigate to the folder where the program is saved. To run the program, type sudo python name_of_the_file.py and press Enter.

11 Turning left or right

```
File  Edit  Format  Run  Options  Window  Help

import time
import ps_drone                 # Imports the PS-Drone-API

drone = ps_drone.Drone()        # Initializes the PS-Drone-API
drone.startup()                 # Connects to the drone and starts subp

drone.takeoff()                 # Drone starts
time.sleep(7.5)                 # Gives the drone time to start

drone.moveForward()             # Drone flies forward...
time.sleep(2)                   # ... for two seconds
drone.stop()                    # Drone stops...
time.sleep(2)                   # ... needs, like a car, time to stop

drone.moveBackward(0.25)        # Drone flies backward with a quarter s
time.sleep(1.5)                 # ... for one and a half seconds
drone.stop()                    # Drone stops
time.sleep(2)

drone.land()                    # Drone lands
```

Modify the same program to alter the flight direction of the drone. This uses the code lines, drone.turnLeft() and drone.turnRight(). After each call, remember to state the duration or time that the action runs for. For example, using time.sleep(2) will turn the drone left or right for two seconds. After the required time, stop the turning action using the code drone.stop(). This returns the drone to the hovering state. Add the relevant code to your program and then save it. Connect and run as we've previously demonstrated in Steps 5 and 6.

```
drone.turnLeft()
time.sleep(2)
drone.stop()
time.sleep(2)
```

12 Tag detection

The drone software has the capability to use the forward-facing camera to identify and read a 'tag'. (Downloadable from the website or available in the FileSilo.) Once detected, this can then be used to trigger an event such as landing the drone or flying it towards the tag. Begin the program by starting a new Python file; import the time and sys modules. Next, import ps_drone. As before, add the startup and connect to the subprocesses.

```
import time, sys
import ps_drone
drone = ps_drone.Drone()
drone.startup()
```

13 Tag settings – part 1

There are four configuration settings to set. First, reset the drone (line one) so that the status is set to 'good', as indicated by four green LEDs. Use the code drone.useDemoMode(True) to use a dataset of 15 when transferring the data from the camera. Now set which packets will be decoded. Finally, set a small time delay to enable the drone to fully awake after the reset.

```
drone.reset()
drone.useDemoMode(True)
drone.getNDpackage(["demo","vision_detect"])
time.sleep(0.5)
```

14 Tag settings – part 2

In this second set of configurations, start by enabling the universal detection by setting the detect type to a value of 3. This triggers the drone to look for the specific tag. Since you are using the front camera only, disable detection from the ground camera. Then set the drone configuration count.

```
drone.setConfig("detect:detect_type", "3")
drone.setConfig("detect:detections_select_h",
"128")
drone.setConfig("detect:detections_select_v", "0")
CDC = drone.ConfigDataCount
while CDC == drone.ConfigDataCount:
time.sleep(0.01)
```

15 Takeoff and taking a reading

The drone is now configured to recognise and read the tag; add a small time delay, line one. This is useful if you need to move the drone outside before it launches. Add lines two and three to launch the drone; once deployed, it will continue to hover. Next, create a loop to continually check for

If the distance to the tag is greater than 300, the drone keeps flying forward

the tag and take the readings. On line nine, tagNum returns the number of tags found; tagX returns the horizontal position of Drone in relation to the tag. The vertical position of the drone is collected with the code on line 11 and stored in a variable called tagY. The distance of the drone from the tag, tagZ, is on the penultimate line and orientation of the drone is stored in tagRot. Remember to include the indentations when adding the lines of code.

```
time.sleep(10)

###take off###
drone.takeoff()
time.sleep(7.5)

# Get detections
stop = False
while not stop:
    NDC = drone.NavDataCount
    while NDC == drone.NavDataCount:
time.sleep(0.01)
    if drone.getKey():
stop = True
    # Loop ends when key was pressed
    tagNum = drone.NavData[“vision_detect”][0]
    tagX =  drone.NavData[“vision_detect”][2]
    tagY =  drone.NavData[“vision_detect”][3]
    tagZ =  drone.NavData[“vision_detect”][6]
    tagRot = drone.NavData[“vision_detect”][7]
```

16 Responding to the data

Now set up a conditional, an if statement to display the data and take action. Line three prints out the collected data; note that it is converted into a string as the original data format is returned as a float, ie a decimal. Create a new variable called distance to store the physical measurement of the distance of the drone from the tag. This is stored as an integer, line four. Check if this distance is greater than 300, line six; if it is then trigger an event. For example, set the drone to fly towards the tag for two seconds, lines seven and eight.

```
    if tagNum:
        for i in range (0,tagNum):
            print “Tag no “+str(i)+” : X=
 “+str(tagX[i])+”  Y= “+str(tagY[i])+”  Dist=
“+str(tagZ[i])+”  Orientation= “+str(tagRot[i])

            distance = int(tagZ[i])
            print (distance)

            if distance > 300:
                print (“Moving Forward”)
                drone.moveForward()
                time.sleep(2)
```

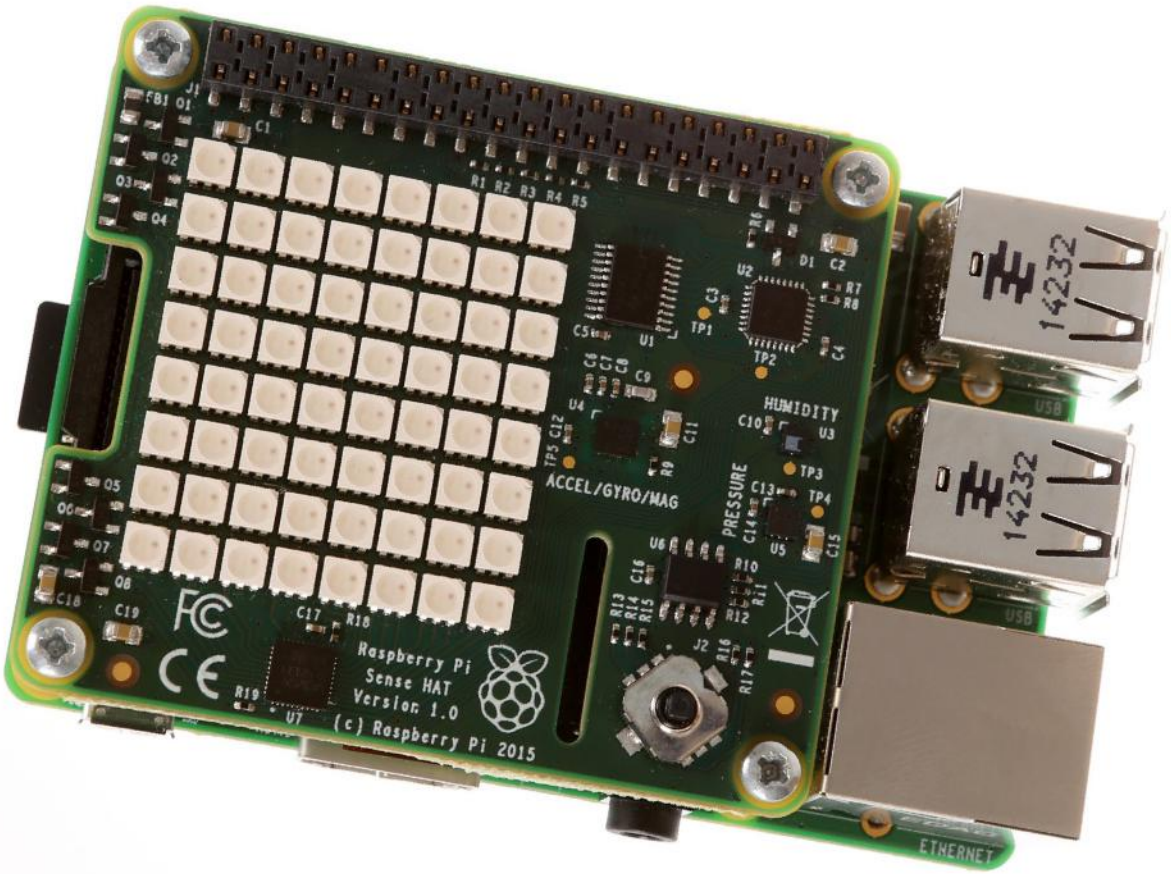
17 Close enough

Once the drone is close enough to the tag, (in this program less than 300), then it stops flying forward, line one. Add a short time delay to allow it to stop, line two. Add the two other conditions; if the drone is already less than a distance of 300 from the tag then print ‘close enough’.

Finally, respond if the tag is not detected, lines five and six. Save your program code and connect to the drone. (Ensure that the indentation levels are correct; if not, this will cause errors when you run the code.) Execute the program as before, following the method described in Steps 5 and 6.

```
                drone.stop()
                #time.sleep(2)
            else:
                print (“close enough”)
    else:
        print “No tag detected”
        #drone.stop()
```

Once youre all set up, check whether you need to register your drone: **http://bit.ly/2szwN5Q.**



Create a real-time LED humidity display

Use the Astro Pi (aka SenseHAT) to take humidity readings and immediately display the results

RECIPE

- SenseHAT

Humidity is a measure of how much moisture or water there is in the air. In this tutorial you will use Python to build a real-time humidity display. The program takes and stores a humidity reading using the SenseHAT's on-board sensor. Then it calculates a simple ratio to determine how many LEDs to turn on. In essence, the higher the humidity the more LEDs are turned on. Then the number of LEDs that need to be turned 'off' is calculated and these are added to the list. Finally, you'll set the program to display and 'turn on' the required number of LEDs. The program continually loops so both the reading and LED display is updated as the humidity changes.

1 Attach the SenseHAT

Ensuring that your Raspberry Pi is off, take the SenseHAT and attach it to the GPIO pins. Slot it firmly into place with the SenseHAT covering the main body of the Pi. Add the power supply and boot up the Pi. Open the LXTerminal Window and type sudo idle3 to open the Python 3 code editor. From the File menu, select New File to create a new program.

2 A test program

Next create a simple program to take a temperature reading and test that your SenseHAT is working correctly. The SenseHAT has a built-in heat sensor that can be used to read and return the current temperature. The sensor is close to the CPU and will pick up some of the residual heat. However, on the whole the reading is sound. Import the SenseHAT module and set the sense variable (lines 1 and 2). Take a temperature reading and store it in a variable called temp (line 3). Finally, print out the current reading (line 4). Save your program and run it; you can do this by pressing F5 on the keyboard.

```
from sense_hat import SenseHat
sense = SenseHat()
temp = sense.get_temperature()
print("Temperature: %s C" % temp)
```

3 Create a live humidity reader

```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()

on_pix = [255,0,0]
off_pix = [0,0,0]
```

Now that you have tested the SenseHAT, you can begin to create the main program. First, import the SenseHat library (line 1). Then use the clear code to turn off any LEDs that have previously been on (line 3). This ensures that the LED matrix is reset each time the program starts. Create two variables, one for the colour of the LEDs when they are on and another for when they are off. In this example the 'on' colour is set to red but you can change the values to change the colours to your own preference.

```
From sense_hat import SenseHat
sense = SenseHat()
sense.clear()
on_pix = [255,0,0]
off_pix = [0,0,0]
```

4 Take a humidity reading

```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()

on_pix = [255,0,0]
off_pix = [0,0,0]

while True:
    hum = sense.humidity
    hum = round(hum,1)
```

Now it's time to take a humidity reading and round it to one decimal place. Begin by creating a while True loop (line 1), which means the program will continually take a reading and display the results. This ensures that the LED display is always updated and the readings are 'live'. On line 2 create a variable to store the humidity reading.
The reading will be very accurate and contain several values after the decimal point. These are not required, so round up the reading to one decimal place (line 3).

```
while True:
    hum = sense.humidity
    hum = round(hum,1)
```

5 What about larger readings?

Depending on the time of year or where you are located, the humidity reading may be very high. However, above a certain value the readings become insignificant. Set a conditional to check if the reading is greater than 100. If it is then change the reading value to 100 (line one and two). This means that the top humidity reading is set to a maximum of 100 and makes the calculation of which LEDs to turn on easier. On the next line create a list called leds which will store the number of LEDs to turn on or off. The SenseHAT has 64 LEDs so calculate the value of humidity which each LED represents (line four). (Each LED represents a value of 0.64 humidity.) Note that the first two lines are indented in line with the previous step.

```
    if hum > 100:
    hum = 100
  leds = []
    ratio =  64 / 100.0
```

6 LEDs On or Off

You now have the components to work out the number of LEDs to turn on to represent the humidity. The formula ratio*hum takes the humidity reading and multiples it by the ratio from Step 5. For example, if the humidity reading is 50, then 50 multiplied by the ratio equals 32, half the LEDs. Convert this to an integer using int and store the value in a variable called on_count (line 1).
To calculate the number of 'off' LEDs, subtract the number of 'on' LEDs from 64, since there are 64 LEDs in total (line 2).

```
on_count = int (ratio*hum)
off_count = 64 - on_count
```

7 Add the 'on' LEDs to the list

Now you have the total number of LEDs that need turning on or off. Write these values back to the list that you created in (Step 5), combining the colour and the total number of LEDs. First, take the colour of the 'on' pixels set in Step 1 and multiply this by the number of LEDs to turn on. Then use the extend list function to write these values into the list created in Step 2. For example, if the humidity is 50, then the total 'on LEDs' will be 32 and this step will add 32 red-coloured LEDs to the led list.

```
leds.extend([on_pix]*on_count)
```

8 Add the 'off' pixels to the list and turn on the LEDs

In Step 6 you calculated the number of LEDs that needed to be turned off using the code: off_count = 64 - on_count. Use this value and multiply it by the colour that you set for the LEDs when they are off, in Step 1. (In this tutorial, (0, 0, 0).) Use the code, 'extend' to create a list which holds the 64 LEDs and record how many of these are 'on' and how many are 'off'. Now read the list and plot it onto the SenseHAT LED matrix using the code: sense.set_pixels(leds) line 2.

```
leds.extend([off_pix]*off_count)
sense.set_pixels(leds)
```

6 Run the program

That completes the program. Press F5 to save and run the program. Watch the LED display, which will show the current humidity as a ratio of the 64 LEDs. Try huffing onto the humidity sensor to see what happens. If it works, the number of LEDs displayed should increase.

GPIO: Work with analogue signals

Les Pounder shows us how we can use analogue sensors and inputs with our Raspberry Pi to control Neopixels

All models of Raspberry Pi come with the GPIO, But no model of Raspberry Pi can interact with analogue components, as the Pi does not come with an Analogue to Digital Converter (ADC). Step forward the MCP3008 ADC. In this tutorial we shall use it with three potentiometers to control a Neopixel ring.

Hardware setup

Please refer to the wiring diagram below. We start the hardware build by inserting the MCP3008 into our breadboard so that it is over the central channel. The notch on the MCP3008 should be facing the top of the breadboard. The MCP3008 has multiple connections to the Pi, for power and for a hardware connection to the SPI bus. These connections are made on the one side of the chip (pins 9 to 16 according to the datasheet). Pins 1 to 8 are reserved for the eight channels that are available for us to use with analogue devices.

Potentiometers, also known as "pots", are three-pinned variable resistors. By turning these we can vary the voltage output from the centre pin. Pots' three pins are voltage, output and ground. They can come in many forms, but we're using single-turn potentiometers similar to those used as volume controls in amplifiers. Others include linear potentiometers, such as those used on a mixing desk, and there also "trimpots" used on circuit boards where infrequent adjustments are needed.

Neopixels are a brand name, created by Adafruit, for the WS2811/12 series LEDs. Each pixel in a series can be individually controlled: its colour, brightness and whether it is on or off. Neopixels need precise timing to control them, and to do that we need to use a GPIO pin on our Pi that can be used with Pulse Width Modulation, PWM. From experience we know that pin 18 can provide this. To power the Neopixels we can also use the 3V and GND pins, which have been broken out to the breadboard as per the diagram.

Software setup

No matter what version of Pi you are using, our first task is to turn off the audio output as this will interfere with our Neopixels. To do this we need to alter the config.txt file in the boot directory, so open a terminal and type the following:

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

At the end of the file, on a new blank line, add the following two lines to comment the change for later reference, and to turn off the 3.5mm audio jack on your Pi. To save and exit the editor, press Ctrl + O and then Enter, then press Ctrl + X. Reboot your Pi to enact the change. Once rebooted and back to the desktop, we can install the Python library that will enable us to control the Neopixels. In a terminal type the following:

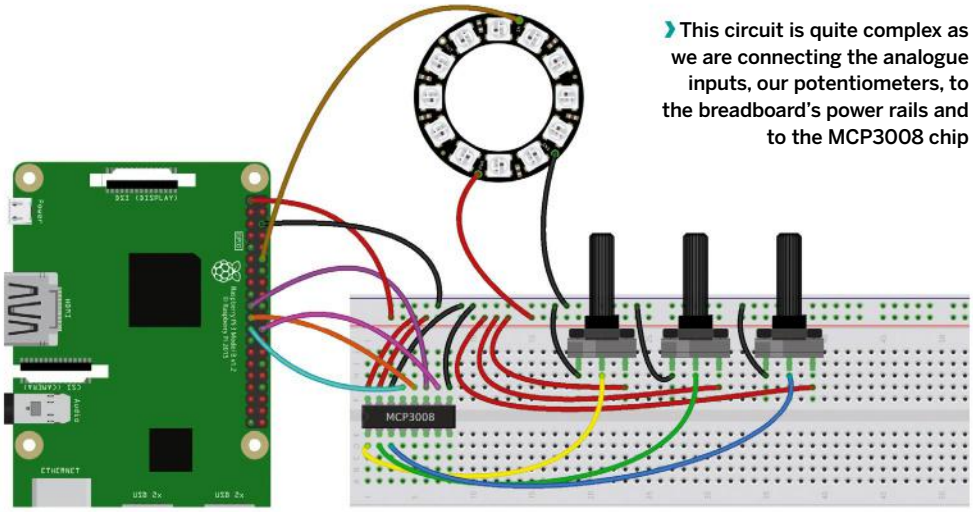
```
$ sudo pip3 install rpi_ws281x
```

After a few moments the library will be installed. The MCP3008 does not need any special software installed as we shall be using it with GPIO Zero, which has a Python class to utilise the chip.

YOU'LL NEED:

- Any model of Raspberry Pi
- The latest Raspbian Pixel Release
- An internet connection for your Pi
- An MCP3008 ADC
- Any Neopixels, we used a 12 pixel LED ring from Adafruit
- Soldering kit
- A breadboard
- 3x 10k potentiometers
- 6x Male to female jumper wires
- 13x Male to male jumper wires

All the code for this project and a diagram can be found at **http://bit.ly/2p5oN6A**
To see the project in action take a look at the YouTube video: **https://youtu.be/0WaW8un23CI**



This circuit is quite complex as we are connecting the analogue inputs, our potentiometers, to the breadboard's power rails and to the MCP3008 chip

NEOPIXELS, APA102 AND OTHER LEDS

The term LED, is quite ambiguous. There is the humble LED, used to signify that a device is on and to say "hello world" when building your first circuit, but there are many others. We used Neopixels, a brand name for WS2811/12 LEDs. These super bright LEDs run from a 3.3V or 5V power supply and use carefully timed pulses to communicate the state of LEDs. Now as we learned in the tutorial, this causes issues as we have to use the GPIO pins, specifically a pin that can perform PWM, and this forced us to disable analogue audio output. But there are other LEDs that we can use instead of the Neopixel, and this is in the form of the APA102 series. These LEDs use the SPI interface, a hardware interface that is capable of sending data much faster to the LEDs, enabling them to be used in projects such as those using Persistence of Vision. APA102 LEDs do not require any additional configuration changes as they do not interfere with audio output. Lots of Pi related companies are using the APA102 instead of the WS2811/12 as it removes the sticky issue of audio configuration changes and provides an easier way to use super bright, individually controllable LEDs.

Coding the project

We start by opening the Python 3 editor, but we need to open it with sudo to ensure that we can use the Neopixels. Open a terminal and type the following to open the editor.

```
$ sudo idle3 &
```

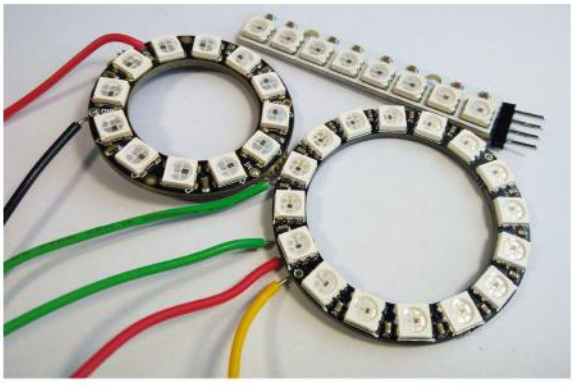
Once it opens, click on File > New to create a new blank file, then immediately click on File > Save and call the file **analogue-inputs.py**. Subsequent saves will now happen much quicker.

Our first section of Python code is our imports. First we import the MCP3008 class from GPIO Zero, enabling us to use the chip. Then we import the sleep function from the Time library, this will help us to control the pace of the project. Finally we import the Adafruit_Neopixel class from the neopixel library, enabling control of the neopixel.

```
from gpiozero import MCP3008
from time import sleep
from neopixel import Adafruit_NeoPixel
```

Our next block of code instructs Python as to which potentiometer is connected to which channel of our MCP3008. We also store the raw values output by the MCP3008 as the variables, r, g, b. These refer to the colours that we can use with our Neopixels. The raw values are between 0.0 and 1.0.

```
r = MCP3008(channel=0)
g = MCP3008(channel=1)
b = MCP3008(channel=2)
```



Neopixels come in many forms, here we can see two rings wired up and ready for use, and an 8 pixel stick with pins

We next create two variables, LEDS, which refers to the number of Neopixels in our string, and PIN, which refers to the GPIO used to control the Neopixels.

```
LEDS = 12
PIN = 18
```

In order to use our Neopixels we need to tell Python where they are connected, and this is done by creating the "strip" object. Then we instruct Python to begin using them.

```
strip = Adafruit_NeoPixel(LEDS, PIN)
strip.begin()
```

A while True loop is used to continually run the code contained within, and in this case the first few lines of code are variables, red, green and blue. These are used to contain the values created by turning the potentiometers, but remember these values are between 0.0 and 1.0. In order to use them with Neopixels we need them to be within 0 and 255, 0 being off, and 255 being full brightness. So we take the initial value and multiply it by 255, and this is enclosed in a round function that will round the number to the nearest integer. This gives us values that can be passed to the Neopixels functions later.

```
while True:
    red = round(r.value * 255)
    green= round(g.value * 255)
    blue = round(b.value * 255
```

For debug purposes we also print the values to the Python shell, this helps us to identify any issues.

```
    print(red,green,blue)
```

A for loop is used to set the colour of each LED in our strip of Neopixels. We use a range that reuses the LEDS variable as the number of times the loop should iterate.

Inside the loop we set the colour of each "pixel" by instructing it as to where it is in the strip, i , and then we pass the red, green and blue values that we created by turning the potentiometers. Lastly in the loop we instruct the Neopixel library to show the results. If we didn't do this, then we wouldn't see any changes.

Pi bites

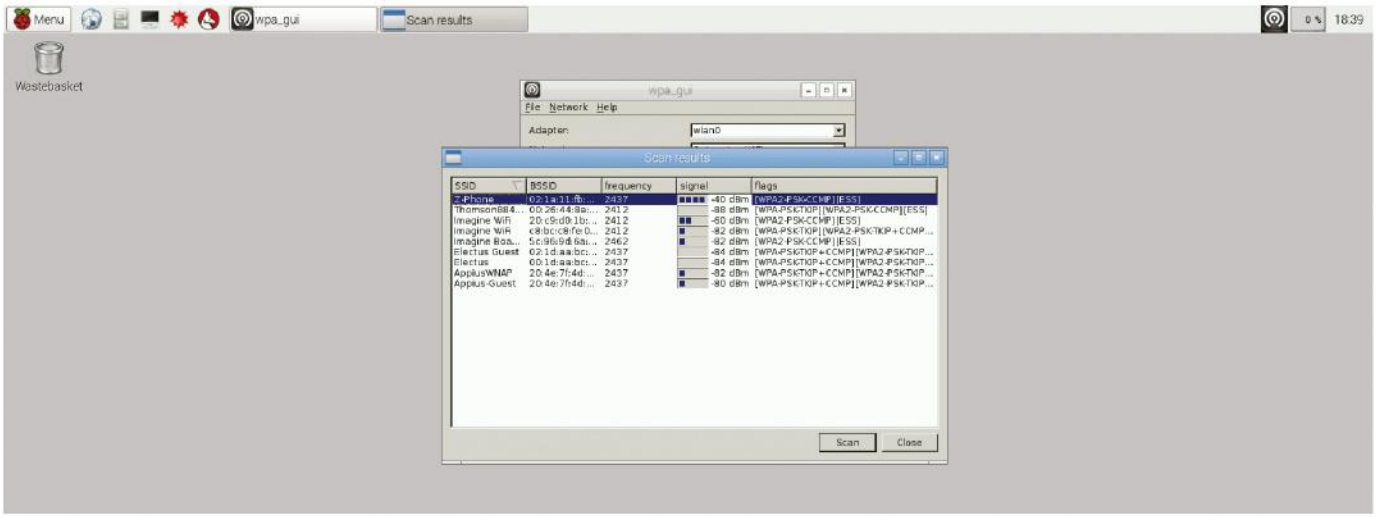
When using any integrated circuit (IC) chip you need to understand what each pin does. Look for the number etched into the chip then use your favourite search engine to find the datasheet.

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

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.



1 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.

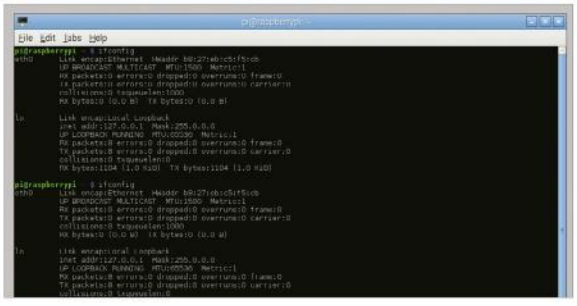
2 Scan for device

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.

3 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.

4 Check connection



Your Android will create an interface known as eth0 on the 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
```

5 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.

6 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.

7 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
```

8 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
```

9 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.

Pi bites

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.

Get to grips with the Enviro pHAT

Take weather-related readings such as temperature and pressure, track movement, measure light and even identify the colour of an object

Pimoroni makes a wide range of HAT (Hardware Attached on Top) boards for the Raspberry Pi. These boards are designed to be fully compatible with all Raspberry Pi models. For the Pi Zero, Pimoroni has created a range of mini HATs called pHATs. This tutorial focuses on the Enviro pHAT, a nifty little board that packs in four different sensors that let you measure temperature, pressure, light levels, colour, acceleration, take a compass heading and retrieve readings from analog inputs. This creates up to ten different variables, making it ideal for monitoring conditions in your house, garage or garden. This tutorial will walk you through the initial install and setup of the hardware and software. Then we'll dive straight in with some starter programs to take readings such as temperature, pressure and movement, and write a simple program to take readings to see how dark or light it is and determine the colour of an object.

1 Install the Enviro pHAT

Attach the Enviro pHAT onto the GPIO pins and secure it in place. Then start up your Pi and install the required libraries. Pimoroni makes this very easy, open the LX Terminal and type:

```
sudo apt-get update
sudo apt-get upgrade
curl -sS https://get.pimoroni.com/
envirophat | bash
```

This installs the required software, which includes a folder that contains example programs to help you get to grips with the Enviro pHAT.

2 Turn LEDS on

Open Terminal and type the following on line 3:

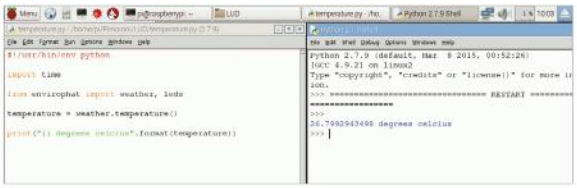
```
sudo idle3
```

Add a short pause, before turning the LEDs off:

```
import time
from envirophat import led
leds.on()
time.sleep(5)
leds.off()
```

Save and run your program. If you've done it correctly, the LEDs should blink once.

3 Take a temperature reading



To create a program to read the current temperature, start a new Python window and import the weather module from the Python library. Then type the following on line 1:

```
from envirophat import weather
```

Next, create a variable called temperature to store the temperature reading. To gather this, use the code weather.temperature() on line 2. This takes the current temperature and then saves this value into the variable. Finally, print out the reading using a print statement, line 3. Save the program and run it; the temperature will be displayed on your screen.

```
from envirophat import weather,
temperature = weather.temperature()
print("{} degrees celsius".format(temperature))
```

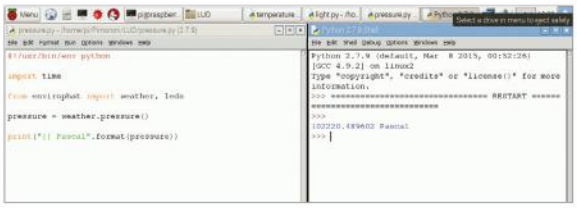
4 Read the pressure

The Envrio pHAT has a digital barometer to take pressure readings. As with the temperature, this is easily achieved with a few lines of code. In your program, import the weather module, which includes the lines of code required for reading the pressure, line 1. On line 2, create a new variable called pressure. Use the code weather.pressure to

take the reading. Finally, on line 3, print out the reading. Save and run your program:

```
from envirophat import weather,
pressure = weather.pressure()
print("{} Pascal".format(pressure))
```

5 Sense movement: part one



The Enviro pHAT's accelerometer can be used to detect motion, such as shaking. But it can also be used to detect motion across three axes to plot a compass reading. Start a new program by importing the time and motion modules, lines 1 and 2.

Next, create a threshold value that holds the amount of motion that will trigger the detection, line 2. This means that you can detect anything from small movements such as a wobble or larger ones such as it falling off a table!

Now create a list called 'reading' to store the motion readings, line 3. Finally, create a variable called last_z which holds the position of the z axis, line 4. Set the z axis to an initial value of zero to denote that no movement has yet been detected.

```
import time
from envirophat import motion
threshold = 0.2
readings = []
last_z = 0
```

6 Sense movement: part two

On the next line, create a loop to keep checking for movement, then take a reading of the position of the z axis and add it to the 'readings list' using the code readings.append, on line 2.

On line 4, take an average of the readings, and then assign this value back the variable 'z'; this is used to compare the new value with the previous value. If it has changed, your Enviro pHAT has moved and motion has been detected. Note that lines 3, 4 and 5 are indented.

```
while True:
readings.append(motion.accelerometer().z)
readings = readings[-4:]
z = sum(readings) / len(readings)
```

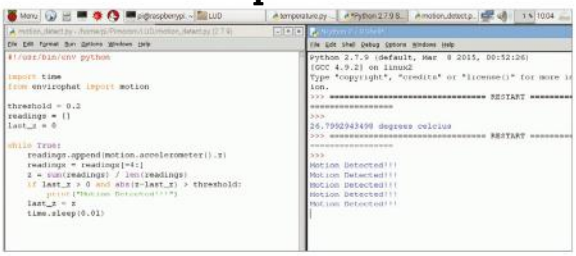
7 Sense movement: part three

Create a condition using an if statement to compare if 'the last value of z is greater than zero' and that the 'absolute value of the new value of z minus the previous value is greater than the threshold value'. Basically, if the value meets these criteria, then the Enviro pHAT has moved to a new position. Inform the user with a simple print statement, line 2. On line 3, update the variable last_z with the current z reading.

Finally, take a small pause before checking again for movement. The program then checks again to see if the z value has changed again and is within the threshold tolerance. Save your program and run it. Wiggle, shake or drop your Raspberry Pi to trigger a movement reading:

```
if last_z > 0 and abs(z-last_z) > threshold:
print("Motion Detected!!!")
last_z = z
time.sleep(0.01)
```

8 Take a light reading and sense colour: part one



This last program takes a reading of the amount of light that is in the room or surrounding environment. Start a new window and import the time and the light module for the program, lines 1 and 2. Next, create a variable in this example called 'amount_of_light'; this stores the Enviro pHAT's light reading. To take the reading use the code light.light().

Then store this in the variable, line 3. Then, on the next line down, create three variables to store the amount of red, green and blue light that is sensed. To take a colour reading of the light use this simple code light.rgb() on line 4:

```
import time
from envirophat import light
amount_of_light = light.light()
r, g, b = light.rgb()
```

9 Take a light reading and sense colour: part two

The final step is to print out the values of light and colour that have been read and stored in the four variables. Display these readings using the line:

```
print("Amount of light", amount_of_light)
```

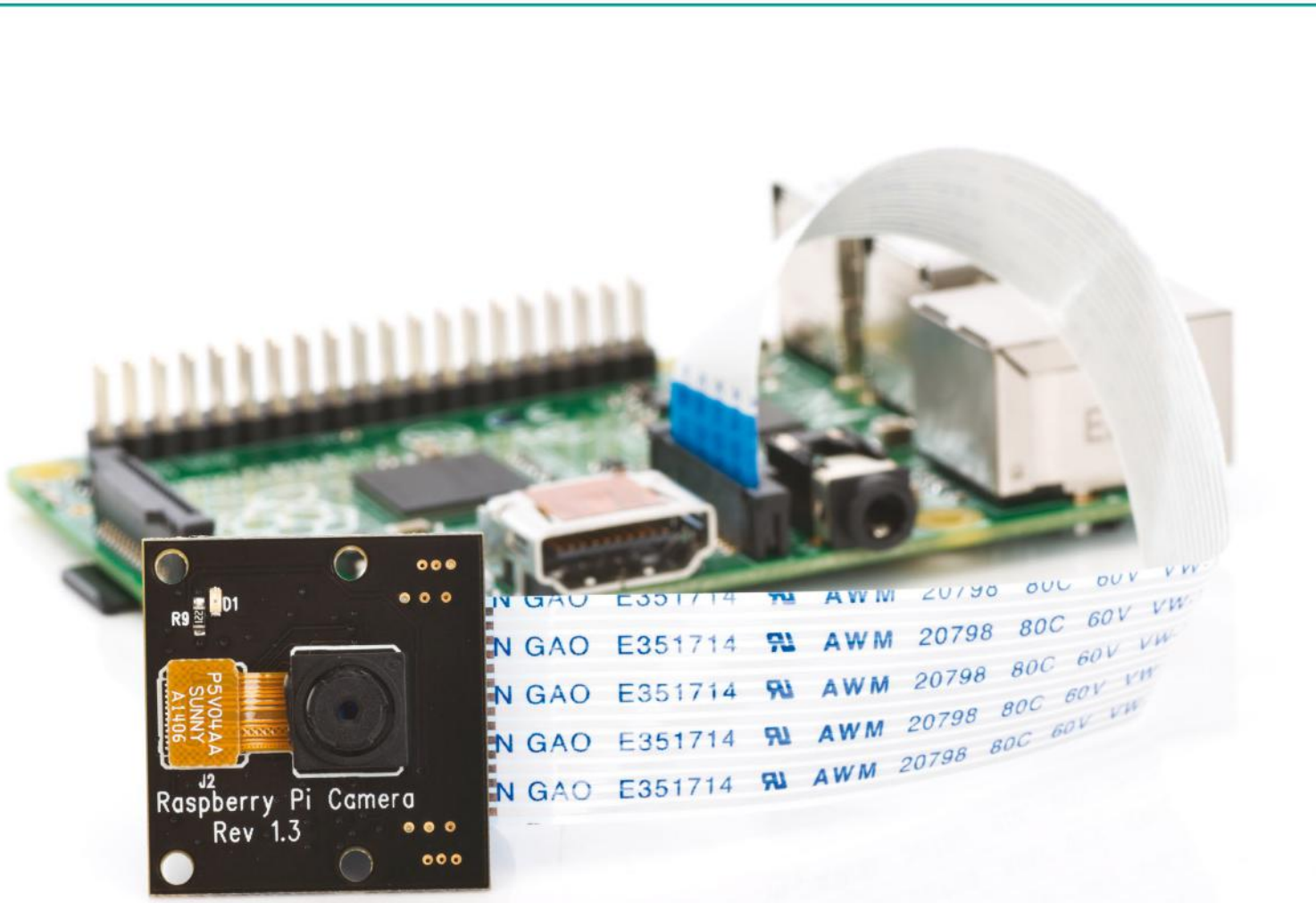
This prints out a small message followed by the reading. The values shown are the amount of white light and the amount of red, green and blue light (RGB) between a value of zero and 255, line 2.

Try running the program while moving your hand over the sensors to make it darker or lighter. You could also try placing a coloured object over the sensors to change the colour reading. Just in case you're wondering, this program is very similar to the paint-matching software that is used in hardware stores to match up a paint colour to another object (e.g. a favourite T-shirt).

```
print("Amount of light", amount_of_light))
print ("Colour", r, g, b)
```

Pi bites

The Enviro pHAT can be combined with a range of other HATs and Pimoroni pHATs, creating almost endless possibilities. For example, display sensor values on a Unicorn pHAT, use a Display-O-Tron HAT to display sensor data, display a graph of values with Scroll pHAT, or how about writing the values directly to a website for others to view and analyse?



Take better night photos with the NoIR camera

Install the NoIR camera to enjoy some improved low light photographs and videos from your Raspberry Pi

RECIPE

- **Raspberry Pi NoIR Camera V2** Element14.com
- **Raspberry Pi case with mount or slot for camera module**
- **Portable power supply**

Want to get more out of your Raspberry Pi's photographic abilities? You might have already connected a basic webcam or the official camera module, but found that certain activities – particularly shooting in low light – have been unsuccessful.

The solution is a new camera module, the Raspberry Pi NoIR Camera Board, which has no infrared filter. This makes the camera ideal for taking infrared photographs or photographing objects in low light (you still won't be able to take photos in the dark without a light source). Capable of delivering 5MP still images, or recording video at 1080p and 30fps, the NoIR camera is easy to install - but for the best results, use with a case and portable power supply.

1 Check your contents

When you receive the Raspberry Pi NoIR Camera module, you'll find two items in the box. The camera module itself should be easy to spot, with its four mounting holes, on in each corner of the PCB. Look out also for the blue gel, a small square colour filter – see the Pi Bites sidebar for how to use that.

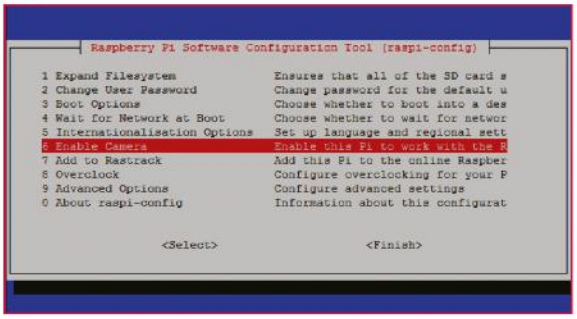
2 Connect the NoIR Camera module

First, making sure to take anti-static precautions, connect the NoIR Camera module to your Raspberry Pi using the ribbon cable, which slots into the ribbon connector with the silver side facing the nearby HDMI port. Don't forget to lift the catch before inserting, and to press down firmly (without excessive pressure) to secure it.

3 Mount your camera module

If your Raspberry Pi case has a mounting point for the camera module, this will almost certainly have four small poles onto which the NoIR camera can be mounted. Simply line up the camera, lens facing the adequately-sized hole, and press the module into place to attach it.

4 Update your Pi

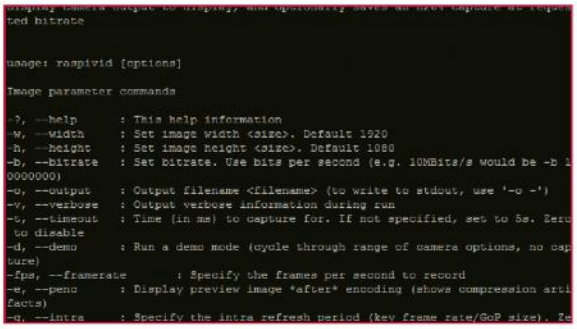


With the camera module safely attached, it's time to boot your Raspberry Pi and run the latest updates:

```
sudo apt install update
sudo apt dist-upgrade
```

This will ensure that the necessary software is installed. You should also open sudo raspi-config and Enable Camera, before restarting your Pi.

5 Take photos and videos



Two tools, raspistill and raspivid, are used for capturing stills and videos with the NoIR camera. Basic usage is as follows:

```
raspistill -o photo1.jpg
raspivid -o video1.h264
```

Use the help command with each to find the full set of commands, which include specifying image size and rotation (see the tip on the right).

6 Take some test photos

To get the most out of the NoIR camera, you'll need to set it up to take some photos in low light. If you can't wait until the evening, you might use other low-light scenarios, such as an attic, basement, or cupboard. For a test shot, try:

```
raspistill -o test.jpg
```

7 Prepare for an evening shoot

If you want to take it outdoors, you'll need a rechargeable battery pack to power the Pi and the camera. You should also have selected a relatively robust case, with good coverage to protect from the elements. For the best results, use an all-weather case, or employ a plastic bag!

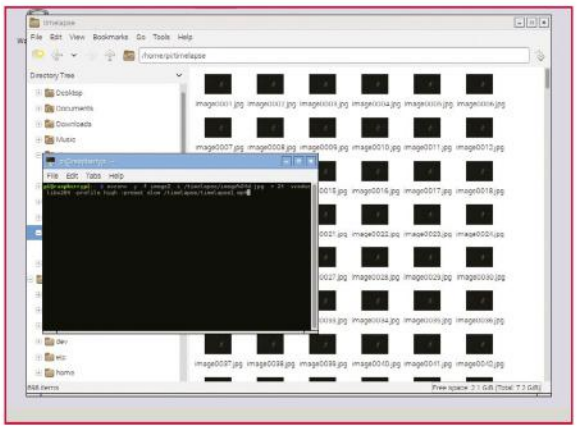
8 Try time-lapse

Time-lapse photography is a good option for capturing images remotely (over SSH) without draining the battery:

```
raspistill -t 480000 -tl 5000 -o /timelapse/
image%04d.jpg
```

Using millisecond values, this command snaps photos every 5 seconds for 8 minutes. Install libav-tools to convert your images into a movie.

9 Compile your time-lapse

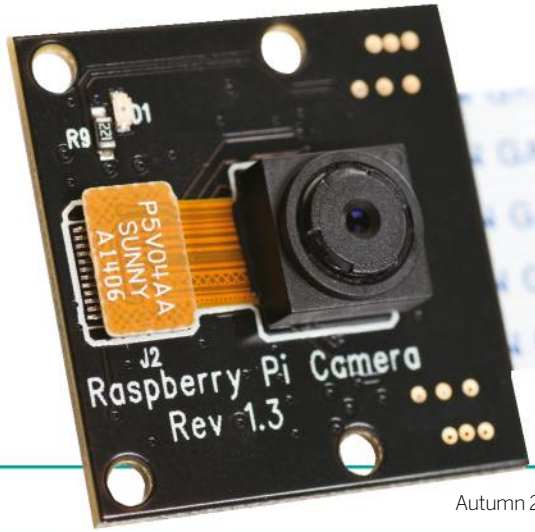


To compile the time-lapse images into a single video file, use:

```
avconv -y -f image2 -i /timelapse/image%04d.jpg
-r 24 -vcodec libx264 -profile high -preset slow /
timelapse/timelapse1.mp4
```

This will take a while to compile on your Pi. When done, use omxplayer to view the results:

```
omxplayer /home/pi/timelapse/timelapse1.mp4
```



Pi bites

Should you find that the results don't quite look right, it might be time to use the blue gel trick. This is around the same size as the NoIR Camera module, and can be placed in front of the lens to deliver a blueish tint to your photos and videos.
Of course, this same technique can be used for standard Raspberry Pi photography - small gels as filters can produce some striking results!

Explorer: Operate LEDs and motors

Les Pounder shows us how we can safely experiment and tinker with the Explorer HAT Pro, a "one for all" board that's easy and cheap to use

Parts...

- Any 40-pin Pi, Raspbian Pixel
- Explorer HAT Pro, 1x LED, 1x 220 Ohm resistor, 1x pushbutton switch, a DC motor, 6x male-to-male jumper wires, 10k potentiometer
- Get the code: **github.com/lesp/LXF225-ExplorerHAT/archive/master.zip**

Master the Explorer HAT Pro with these four projects. We'll be using the Analog inputs with potentiometers, digital Input/Outputs with LEDs and buttons, and the H Bridge controller for operating DC motors with the board. Before we power on the Pi we need to attach the Explorer HAT Pro to all 40 of the GPIO pins. The board should sit atop the Raspberry Pi, matching its outline. To start, enter this command in Terminal to install the Explorer HAT Pro libraries:

```
$ curl https://get.pimoroni.com/explorerhat | bash
```

Follow the install instructions, reboot, and return the Pixel desktop. Now go to the main menu and navigate to the Programming menu. Click Python 3 to open the editor, then click File>New.

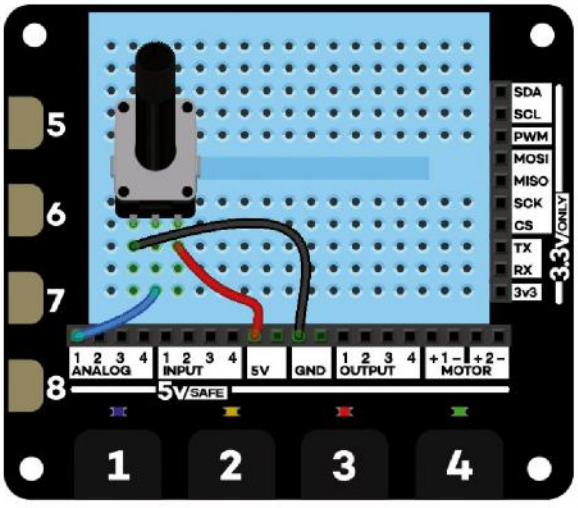
Project 1: toggle the LED

Here, we'll use the capacitive touch buttons (1-4) on the Explorer HAT Pro to trigger the built-in LEDs. Save a blank document as Project1-LED-Toggle.py, then start the code by importing the Explorer HAT Python3 library, and then renaming it to "eh" so that it is easier to work with. We also import the sleep function from the time library:

```
import explorerhat as eh
from time import sleep
```

Next, you should create an infinite loop to hold the code's main body:

```
while True:
```



Potentiometers send a voltage to the Analog input that's varied by turning the dial to alter resistance

Next, set up a series of conditional tests to see if the capacitive touch buttons (1-4) have been pressed. For each button, assign a colour of LED that will be toggled on/off by pressing the button. Add a delay to prevent an accidental double tap of the button:

```
    if eh.touch.one.is_pressed():
        eh.light.blue.toggle()
        sleep(0.1)
```

The rest of the code follows the same pattern, but using elif to represent the other conditions to test:

```
    elif eh.touch.two.is_pressed():
        eh.light.yellow.toggle()
        sleep(0.1)
    elif eh.touch.three.is_pressed():
        eh.light.red.toggle()
        sleep(0.1)
    elif eh.touch.four.is_pressed():
        eh.light.green.toggle()
        sleep(0.1)
```

Click Run>Run Module, then press the 1-4 keys on the Explorer HAT Pro to trigger the LEDs.

Project 2: the blinking obvious

For this project you'll need the potentiometer and three jumper wires. The potentiometer is connected to the ANALOG1 pin and sends the voltage passing through the variable resistor as the dial is turned. At full voltage it'll be just over 5V. Create a new blank file called Project2-Analog-Blink.py and import the Explorer HAT library:

```
import explorerhat as eh
```

Inside of a while True loop,create two variables to read the voltage reading on ANALOG1 and store it:

```
while True:
    on = eh.analog.one.read()
    off = eh.analog.one.read()
```

Next, create a hack that prevents the code from crashing due to a divide by zero error. If the value of our variables "on" and "off" ever reaches zero, then we change their values to be 0.01:

```
    if on == 0 and off == 0:
        on = 0.01
        off = 0.01
```

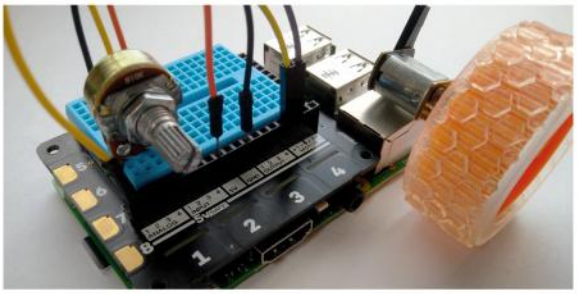
EXPLORERS COME IN MANY PACKAGES…

The Explorer HAT Pro is a top-of-the-line board. It comes with eight capacitive touch pads, four analog inputs, four digital inputs and four outputs. We have a DRV8833 motor controller and pins for I2C/SPI and Serial connections. And we get a breadboard to prototype with.

The capacitive inputs 1-4 are best used with your hands, but for those of you who wish to test the conductivity of a banana, or use either tin foil or copper tape to create larger circuits, then you can use 5-8 with crocodile clips and turn anything into a fun form of input. This is in much the same manner as the Makey Makey, a device that emulates a keyboard/mouse and can turn conductive objects into forms of input.

But the Pro isn't for everyone. The next option is the Explorer pHAT, which is designed for the miniature Pi Zero, but will work with all 40-pin GPIO Pis. The Explorer pHAT omits the capacitive touch interface, breadboard, LEDs and the breakout for I2C. But it's half the price of the Pro version.

So, if you need an all-in-one solution for robots and analog sensors then the pHAT board is all you need. But if you want all the features for whatever your next project maybe, then perhaps the Pro board is more suited to you.



We used a micro gear metal motor because it's 5V compliant, yet can handle the requirements of robotics

The last line of code flashes the built-in LEDs with an on/off time that matches the voltage read at ANALOG1. This means a low voltage will make the LEDs flash faster, a high voltage, slower.

```
eh.light.blink(on,off)
```

Click Run>Run Module, then rotate the dial to change how fast the LEDs will blink.

Project 3: Gain motor controls

Keep the potentiometer connected, add a DC motor connected to MOTOR1 + and –, plus a simple speed controller for a DC motor using the potentiometer as an input. Create a file called Project3-Analog-Motor.py and import the Explorer HAT library:

```
import explorerhat as eh
```

Create an infinite loop:

```
while True:
```

Inside the loop, create a variable called speed, and in there store the reading from ANALOG1, but round the value to one decimal place for ease of use. Multiply the value by 20 to give us a percentage value, which is then passed to the final line of code that controls the motor forwards, and takes a percentage value to represent its speed:

```
    speed = (round(eh.analog.one.read()) * 20)
    eh.motor.one.forward(speed)
```

Finally, click Run>Run Module. Rotate the dial of your potentiometer to change how fast the motor will spin.

Project 4: push the button

Our last project shows how the Explorer HAT's input and output features work with external components, using a push button to toggle the built-in LEDs and a standard LED. This project requires the push button, an LED, a 220 Ohm resistor and four jumper wires. The circuit is built on the breadboard with the button connected to 5V and INPUT1, the long leg of the LED connected to the 5V pin, and the short leg connected to OUTPUT1 via the resistor. The Explorer HAT OUTPUT pins are a path to Ground, and is only activated when turned on. To begin, create a new file called Project4-Input-Output.py, import the Explorer HAT library, then import the sleep function from the time library:

```
import explorerhat as eh
from time import sleep
```

Create a loop, and in there a conditional test, that will check to see if the capacitive buttons (1-4) have been pressed. Button 1 looks like this:

```
while True:
    if eh.touch.one.is_pressed():
```

If that button has been pressed, toggle the Blue LED under button 1 to turn on/off. Also add a short sleep to prevent accidental double taps:

```
        eh.light.blue.toggle()
        sleep(0.1)
```

Create further conditional "else if" tests that will check to see if buttons 2-4 have been pressed, and toggle the LED under each button to turn on:

```
    elif eh.touch.two.is_pressed():
        eh.light.yellow.toggle()
        sleep(0.1)
    elif eh.touch.three.is_pressed():
        eh.light.red.toggle()
        sleep(0.1)
    elif eh.touch.four.is_pressed():
        eh.light.green.toggle()
        sleep(0.1)
```

Click Run>Run Module. Press the buttons 1-4 to toggle the LEDs on and off.

Pi bites

The Explorer HAT Pro and the Explorer pHAT can both power two motors. But you'll need to use micro gear metal motors and not the large yellow DC motors found in many robot kits.

Preserve your Pi's Micro SD card

Use a few tweaks and USB-booting to prolong your SD card and avoid losing important Raspberry Pi projects

RECIPE

- Github repository
- Pimoroni Mote-pHat and accessories
- Soldering iron, flux and solder

At some point during the lifetime of your Pi, you are likely to encounter a problem with your SD/microSD card. If you're lucky, this will be minimal; perhaps a reboot will fix it. If you're unlucky, it could mean the end of your SD card, and the loss of all data on your Pi, including your latest project. The simple fact is that SD cards do not last forever. Flash storage, by design, has a limited number of read/write cycles. While error-correction software is built-in, and cards ship with larger-than-described storage to cover damaged blocks, eventually corruption will cause a problem.

SD card corruption occurs in various ways. It might be due to a sudden variation in voltage during a read/write cycle or from being removed from the Raspberry Pi. Flash storage is also susceptible to extreme temperatures and physical damage. Cheap SD cards, meanwhile, are usually unreliable; whatever the situation, you should rely on the more expensive cards from SanDisk or Kingston.

While it might be useful to regularly back up your flash storage to enable quick recovery, a more proactive option is to bypass the SD card entirely by relying on other storage mediums for booting the Raspberry Pi, but you should also be aware of various tools that can be used to protect your microSD card.

1 Don't flash a fresh OS for every project

```
pi@raspberrypi:~ $ sudo apt-get remove dnsmasq
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  dns-root-data dnsmasq-base libmnl0 libnetfilter-conntrack3
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
  dnsmasq
0 upgraded, 0 newly installed, 1 to remove and 71 not upgraded.
After this operation, 117 kB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 33366 files and directories currently installed.)
Removing dnsmasq (2.72-3+deb8u1) ...
pi@raspberrypi:~ $
```

The read/write cycle limit is a physical hardware restriction preventing infinite reuse of an SD card. So look after it! While 'swappiness' (the kernel parameter defining how much and often RAM is copied to storage) is apparently set low in Raspbian, there is more you can do.

One way to extend the lifespan of the device is to avoid flashing a fresh version of Raspbian (or whatever your preferred OS is) every single time you start a new project.Doing so applies a card-wide reduction in the remaining read-write cycles. So, by maintaining a working microSD card throughout several projects, you avoid the effect that regular flashing has on the card.

This is not a perfect solution, but it will help you get projects started without the initial flashing and configuration that is typically required. Need to keep things tidy? Remove software with sudo apt-

get remove APPNAME. This will uninstall the software you no longer need, but may be time-consuming. In short, only flash Raspbian when you really must.

2 Maintain a constant power supply

It may sound overly simple, but a reliable power supply is actually a major aspect in the preservation of your Raspberry Pi's SD storage. If there's much variation, data can be lost, ultimately causing corruption of the card. At this point, your Raspberry Pi probably won't boot, and a new OS will need flashing.

The Pi requires a constant voltage of 5V. This is available via the approved mains adaptors, but keep in mind that these devices cannot account for failings in the sockets they're plugged into. Don't us3e cheap extension leads. Instead, ensure your mains adaptor is connected either to the wall (if you have modern surge protection built in) or otherwise to an extension with surge protection. It's common to attempt to squeeze as much power out of a Raspberry Pi through overclocking, but this too is a possible cause of disk corruption. Rather than go through the rigmarole – and card-degrading – act of flashing a new ROM, it's safer to avoid overclocking the Pi's CPU if you want to maintain a stable system.

3 Write to RAM, not storage

A great way to reduce the number of read/write cycles on your SD card is to not write to it in the first place. No, this doesn't mean leaving your Pi powered off! Instead, you can write everything to the computer's RAM.

As such, nothing will be written to the microSD card, thereby extending its life just a wee bit longer. Better still, this is easy to set up using tempfs. Begin by opening the fstab in nano with sudo nano /etc/fstab. At the bottom of the file, add this line:

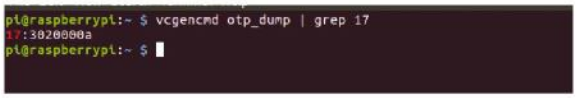
```
tmpfs /var/log tmpfs defaults,noatime,nosuid,
oe=0755,size=100m 0 0
```

Next, press Ctrl+X to exit and save. This will move the /var/log directory to RAM, reducing the microSD card's read/write cycles. Other locations can be safely moved to RAM too:

```
tmpfs /tmp tmpfs defaults,noatime,nosuid,siz
=100m 0 0
tmpfs /var/tmp tmpfs defaults,noatime,nosuid
size=30m 0 0
tmpfs /var/log tmpfsdefaults,noatime,nosuid,mo
de=0755,size=100m 0 0
tmpfs /var/run tmpfs defaults,noatime,nosuid,
mode=0755,size=2m 0 0 tmpfs /var/spool/mqueue
tmpfs defaults,noatime,nosuid,mode=0700,gid=12,s
ize=30m 0 0
```

However, please note: doing this will only last until you reboot your Pi, at which point everything is cleared from RAM.

4 Boot your Raspberry Pi from a USB stick



Recent updates to Raspbian enable you to boot a Raspberry Pi 3 via an attached USB device (a flash drive, standard HDD or even an SSD), bypassing the microSD card entirely. Via SSH, begin with:

```
sudo apt-get update
sudo BRANCH=next rpi-update
```

Then enable USB boot mode:

```
echo program_usb_boot_mode=1 | sudo tee -a
boot/config.txt
```

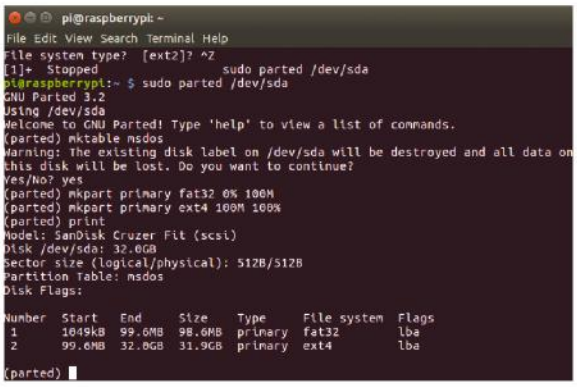
With the program_usb_boot_mode=1 instruction added to the end of the config.txt file, reboot with sudo reboot. When the Pi restarts, you'll need to check if the one-time programmable (OTP) memory has been changed.

```
vcgencmd otp_dump | grep 17:
```

If the previous steps have been successful, you should see something like '0x3020000a' (pictured, above). Your Raspberry Pi is now ready to boot from a USB device, so connect the one that you want to use; but note that it will be reformatted. You can Identify your USB device with lsblk, which will list all block devices.
Connected USB devices are usually called 'sda'. Enter the following to unmount the device and run the Parted tool to take you to the Parted prompt:

```
sudo umount /dev/sda
sudo parted /dev/sda
```

5 Copy Raspbian to your USB drive



At the prompt, enter:

```
mkpart primary fat32 0% 100M
mkpart primary ext4 100M 100%
print
```

Then use Ctrl+C to exit. Back at the command prompt, you will need to create both a new boot

Pi bites

It is possible to run checks to give you advance warning of impending SD card corruption. Begin with this command to check the filesystem every x number of reboots. Change [x] for your preferred number: tune2fs -c [x] / dev/mmcblkp02. You might also enable autocorrection, to help with accurate data storage with nano /etc/default/ rcS --> FSCKFIX=yes.

filesystem and root filesystem:

```
sudo mkfs.vfat -n BOOT -F 32 /dev/sda1
sudo mkfs.ext4 /dev/sda2
```

Next, mount the target filesystems:

```
sudo mkdir /mnt/target
sudo mount /dev/sda2 /mnt/target/
sudo mkdir /mnt/target/boot
sudo mount /dev/sda1 /mnt/target/boot/
sudo apt-get update; sudo apt-get install
rsync
```

To copy Raspbian to your USB device use:

```
sudo rsync -ax --progress / /boot /mnt
target
```

This will take a while to complete, so leave it to finish. Once done, you'll need to copy the SSH host keys from the microSD card to the USB device to maintain the connection via SSH. Enter the following a line at a time.

```
cd /mnt/target
sudo mount --bind /dev dev
sudo mount --bind /sys sys
sudo mount --bind /proc proc
sudo chroot /mnt/target
rm /etc/ssh/ssh_host*
dpkg-reconfigure openssh-server
exit
sudo umount dev
sudo umount sys
sudo umount proc
```

6 Prepare to boot from USB

Before you reboot your Pi from the USB device, edit the cmdline.txt file again in the terminal:

```
sudo sed -i "s,root=/dev/mmcblk0p2,root=/dev
sda2," /mnt/target/boot/cmdline.txt
```

A similar change must also be made to /etc/fstab:

```
sudo sed -i "s,/dev/mmcblk0p,/dev/sda," /mnt
target/etc/fstab
```

You're now ready to unmount the filesystem:

```
cd ~
sudo umount /mnt/target/boot
sudo umount /mnt/target
```

At this stage, you can enter the poweroff command to shut down your Pi. Once the lights are off, disconnect the power supply and remove the microSD card. A few minutes later, you can reconnect the power and boot your Pi – from the USB device!

Your microSD card is now guaranteed a much longer lifespan. You might retain the Raspbian image, however, for preparing other Pis for USB booting in future.

Better still, this means that you can have as much storage as possible for your Raspberry Pi. USB flash usually stops around the 512GB capacity (although there are larger devices) while mechanical hard disk drives can potentially add terabytes of storage to your Pi. Alternatively, an SSD device will speed things up considerably.

7 Boot Raspbian across your network

Booting from USB can be taken to the next level – network boot. Using a Pi as a server, you can set up a Raspberry Pi 3 with Raspbian Lite and set it to initially boot from USB. With the server and new client configured correctly, the Pi can boot from the network, again reducing the impact on the SD card.

On your intended client, follow the previous steps up to the point of removing program_usb_boot_mode from /boot/config.txt, then running the poweroff command.

Next, remove the SD card and insert it into the Pi you'll be using as a server. Boot this device, then run sudo raspi-config. This will open the configuration options. Select the Expand Filesystem option. Then create a copy of the root filesystem:

```
sudo mkdir -p /nfs/client1
sudo apt-get install rsync
sudo rsync -xa --progress --exclude /nfs
/nfs/client1
```

This will take a while to complete, so be patient!

8 Find the addresses

Continue by maintaining your connection via SSH, by regenerating the SSH host keys:

```
cd /nfs/client1
sudo mount --bind /dev dev
sudo mount --bind /sys sys
sudo mount --bind /proc proc
sudo chroot .
rm /etc/ssh/ssh_host_*
dpkg-reconfigure openssh-server
exit
sudo umount dev
sudo umount sys
sudo umount proc
```

Then find the address of your router. If you don't know this, run:

```
ip route | grep default | awk '{print $3}'
```

Pi bites

Low-cost SD and micro SD cards are generally unreliable. But how can you tell what's good quality or not? First, check the brand. Buy cards from SanDisk or Kingston as a rule. Second, look at the Speed Class rating. SDHC and SDXC cards are available, fast, reliable media. The higher the speed rating (upwards from 2MB/sec), the better the card.

Check your Pi's own IP address with:

```
ip -4 addr show dev eth0 | grep inet
```

Then use cat /etc/resolv.conf to find the address of your DNS server. You should now have your device's IP, broadcast and DNS server addresses, so note these down. Run sudo nano /etc/network/interfaces and edit the line iface eth0 inet manual so that it reads:

```
auto eth0
iface eth0 inet static
address [YOUR_IP_ADDRESS]
netmask 255.255.255.0
gateway [YOUR_BROADCAST_ADDRESS]
```

Press Ctrl+X to save and exit. To make this work, you'll need to disable DHCP networking, so run the following and then reboot your Pi:

```
sudo systemctl disable dhcpcd
sudo systemctl enable networking
```

9 Configure your server's network settings

Enter the following with the DNS IP address you noted earlier.

```
echo "nameserver [YOUR_NAMESERVER_IP]" |
sudo tee -a /etc/resolv.conf
```

Prevent changes to this with:

```
sudo chattr +i /etc/resolv.conf
```

You then need to install some software:

```
sudo apt-get update
sudo apt-get install dnsmasq tcpdump
```

The dnsmasq tool can cause problems, so prevent this with:

```
sudo rm /etc/resolvconf/update.d/dnsmasq
```

Reboot again, then run tcpdump to detect DHCP from the client Pi.

```
sudo tcpdump -i eth0 port bootpc -v
```

At this stage, connect the client Pi to the network via Ethernet, and connect the power cable. After ten seconds, the LEDs should light, and a packet from the client will be received by the server and displayed in the tcpdump tool. Press Ctrl+C to exit, then enter

```
sudo echo | sudo tee /etc/dnsmasq.conf
sudo nano /etc/dnsmasq.conf
```

Delete everything in the file and add:

```
port=0
dhcp-range=[YOUR_BROADCAST_IP],proxy
log-dhcp
enable-tftp
tftp-root=/tftpboot
pxe-service=0,"Raspberry Pi Boot"
```

Next, create a new directory, tftpboot:

```
sudo mkdir /tftpboot
sudo chmod 777 /tftpboot
sudo systemctl enable dnsmasq.service
sudo systemctl restart dnsmasq.service
```

10 Prepare for network boot

Continue by monitoring the dnsmasq log with:

```
tail -f /var/log/daemon.log
```

If working, a 'not found' message will be displayed. Copy the necessary files with cp -r /boot/* /tftpboot.
Then restart dnsmasq with sudo systemctl restart dnsmasq and the client Pi is ready to boot the root filesystem and then boot from the network. The /nfs/client1 filesystem must now be exported:

```
sudo apt-get install nfs-kernel-server
echo "/nfs/client1 *(rw,sync,no_subtree
check,no_root_squash)" | sudo tee -a /etc/
exports
sudo systemctl enable rpcbind
sudo systemctl restart rpcbind
sudo systemctl enable nfs-kernel-server
sudo systemctl restart nfs-kernel-server
```

Next, edit /tftpboot/cmdline.txt, changing the line beginning 'root=' to:

```
root=/dev/nfs nfsroot=[YOUR_DEVICE_IP]:/nfs
client1 rw ip=dhcp rootwait elevator=deadline
```

Open fstab (with sudo nano /nfs/client1/etc/fstab) and remove the /dev/mmcblkp1 and /dev/mmcblkp2 lines. You're done! Now start the client and wait for it to boot from the network. This may be a little slower than what you're used to (depending on your network speed), but will extend the life of your Pi's microSD card considerably.

"THE SIMPLE FACT IS THAT YOUR AVERAGE RASPBERY PI SD CARD DOES NOT LAST FOREVER. FLASH STORAGE, BY DESIGN, HAS A LIMITED NUMBER OF READ/WRITE CYCLES"

Build a smart calendar

With a little cunning, you can transform your Pi into the perfect date-checker

Pi bites

If you've previously installed OwnCloud, this project will work perfectly with the built-in OwnCloud Calendar. Just make sure that everyone who needs access has the LDAP link.

One staple in films such as *Back to the Future* and *Time Cop* is that everyone has a handy digital-planner panel in their living room showing their appointments for the day. While we already have calendars on our smartphones and tablets, but now, thanks to the Raspberry Pi, it's possible for you to have an economic wall-mounted calendar in your home or office too.

Given that we have just tacitly admitted we could use the calendar app on our phones instead, is this project merely a novelty, or are there any advantages to it? Well, there's always the look of the thing. A calendar mounted on a wall can certainly be more aesthetically pleasing than many mobile phone displays. Some people on the internet have gone to great lengths in this direction, such as mounting it tastefully in a wooden frame or building it into a mirror.

The main advantage, however, is that it enables you to share a calendar with other people, by putting it in a public place, such as your living room. Your family can see your own appointments, and you can make sure to schedule your commitments around theirs. In the workplace, you can use calendar views, such as Agenda in Google Calendars, to organise meetings and assign tasks to your colleagues.

Crafting your calendar

For this project, you need a Raspberry Pi with internet access. In the interests of saving on cabling and space, it's best to use the Raspberry Pi 3, which has integrated Wi-Fi. You also need to choose a monitor. One excellent option is the Official Raspberry Pi Touchscreen Display *(*see Choosing a Monitor, over the page*)* but ultimately any compatible monitor will do. This isn't a DIY tutorial, so please only attempt to mount the Pi and display if you are comfortable with using a drill and installing brackets. If the display comes with a stand, there's no reason it can't be placed on a desk or table. This is also a good time to start measuring cable lengths, so you can be sure both the monitor and the Pi will have power wherever they're mounted. Once your equipment is in order, you need to consider the type of calendar you wish to use. If you and your family or colleagues already have a calendar you share, you can start following the tutorial right away.

Serving suggestion: this monitor is mounted on a wooden board, then layered with cork for a rustic feel.

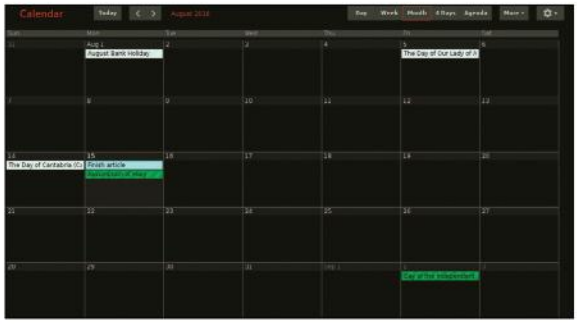
If that's not the case, you may wish to create a single calendar for this purpose. If you're using Google Calendars, follow the steps at **http://bit.ly/1HKeoCL** to do this. For Mac users, visit **http://apple.co/2nz2w1v** to create a new iCloud Calendar. Outlook users can also create a calendar by visiting **http://calendar.live.com**.

It's not that important which calendar service you use, provided it can be displayed in Mozilla Firefox, which we're using for this project. Try to give the calendar a distinctive name, such as Smith Family Calendar, so everyone using it knows it's distinct from their personal calendar.

Importing calendars

If you do have an existing calendar, you may wish to import your personal appointments, birthdays and so on into the new one. It may not be necessary, because providers such as Google and iCloud allow multiple calendars. Events are colour-coded to show which calendar they belong to.

However, if one of the people using your new calendar previously used a different platform – for example, you have decided you'll all use a Google Calendar and one person used an iCloud one on



If you choose to subscribe to other calendars, public holidays in other countries and other useful information will be highlighted

CHOOSING A MONITOR

There's no shortage of suitable small monitors to connect to your Pi. If you feel comfortable with a small amount of wiring, the Official Raspberry Pi 7-inch Touchscreen Display is the ideal size to display a calendar, as well as having a handy slot at the back to place your Pi. The screen, along with assembly instructions, is available from the Pi Hut website for £55 (**https://thepihut.com**). If you don't like messy wires, the Pi Hut also sells a short micro USB power cable for £2 to allow the Pi to draw power from the monitor's USB port.

The Raspberry Pi Touchscreen Display has the added advantage of enabling you to scroll through appointments with a click of a finger. If this is not important to you, or the display is out of your budget, Amazon and eBay also sell Pi-compatible displays. As the Pi has an HDMI port, any HDMI compatible monitor will do, but some monitors come with a driver board to allow you to connect it to the Pi's own DSI port.

If you are very comfortable with electronics and want to save money, find a broken-down laptop with a working LCD. If you can remove the screen safely and buy a compatible controller board online, it can be made to work with the Pi. Visit **www.instructables.com/id/Old-laptop-screen-into-Monitor/?ALLSTEPS** for some tips.

A controller board magically turns a laptop screen into a monitor (get one configured to your specific screen)

their iPhone – you need to import it.

To import events from an iCloud Calendar into Google, first export them into an ICS file by following the steps at **http://apple.co/1l0wS0p**. Then import the file by following step 2 at **http://bit.ly/1mMXSIA**. To export a Microsoft Outlook Calendar to Google Calendar, follow the steps at **http://bit.ly/2cl17IN**.

Customisation

Once you have a single, shared calendar, take some time to set it to a format with which you're comfortable. Most providers have the option of a daily, weekly or monthly view. Next, feel free to fine-tune the calendar's appearance. You can make changes to the iCloud Calendar – for example, to change the viewable time period – by following the instructions at **http://apple.co/2oeuCm2**.

Google Calendar's default look and feel is somewhat spartan. If you would like to experiment with different themes, there are a number available at **http://bit.ly/ 2oeD7O6**. You need the Stylish Firefox extension in order to install them. Visit **https://mzl.la/1fe2Nwr**, then click Add to Firefox to install this.

Full-screen ahead

As you'll be using a much smaller screen than you're used to, space will be at a premium, so consider installing the Real Kiosk add-on for Mozilla Firefox. This does what it says on the tin: it's designed to turn your browser into the equivalent of an internet kiosk. This means the menus, toolbars and even the right-click function are disabled. The chief advantage of this is that Firefox always opens in full-screen mode, making your calendar much easier to see. This also makes sure your device can only be used as a calendar, as people trying to view other websites are bounced back. If you do need to close down Firefox for any reason, you can do this by connecting a keyboard, holding down the Alt key, then pressing F4.

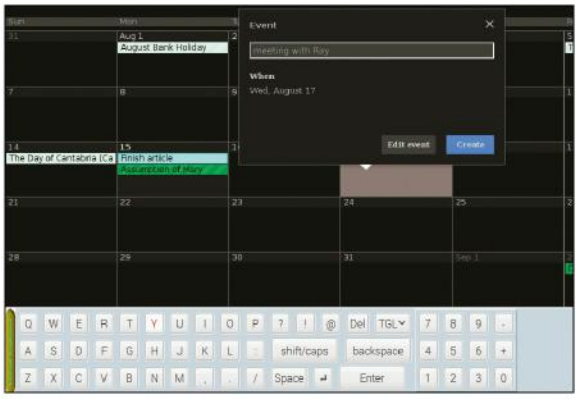
Editing your calendars

Reading this project so far, it would seem that viewing the calendar in the web browser is passive. But if you have a central calendar on your wall, wouldn't it be ideal to let people add and edit appointments as well? If you are using the official Raspberry Pi Touchscreen Display, you can use this to edit the time of events and even create new ones.

Problems may arise when you want to edit the text of events or create names for new ones. Naturally, you could connect a small wireless keyboard and leave it near the wall-mounted calendar in case data needs to be entered. A much less clumsy solution, however, would be to have the keyboard built into the browser itself. The Mozilla Firefox extension VKeyBoard is designed for kiosk browsers, and pops up when clicked to allow users to enter text. Simply visit **https://mzl.la/2njGePN** inside the browser and click Add to Firefox to install. If you have already installed the r-kiosk add-on and can't change your web page, restart Firefox in safe mode, as outlined above.

Sharing the dates

If you want to use any device besides the Raspberry Pi to add or change appointments in Google Calendar, you either need to sign into your Google or iCloud account on that device, or share your calendar with others. To share your Google Calendar, follow the steps at **http://bit.ly/2nzfqwG**. You can send a link to only certain email addresses or make the calendar viewable to anyone with the link. You can do the same for iCloud Calendars by following the steps at **http://apple.co/2bfWHk8**. If you use Outlook 2010, it's also



Use the VKeyBoard plugin to type with a touchscreen. You can maximise and minimise the keyboard by tapping on the yellow button

Pi bites

If you've already installed the Real Kiosk add-on, you won't be able to visit the Add-ons site. Open Terminal and run **firefox-esr -safe-mode** to open Firefox with add-ons disabled.

Pi bites

If you have a personal appointment, you can create a private event. Other people accessing your calendar simply see you're busy, with no further details.

possible to publish a calendar to **Outlook.com**. For more information, follow the section entitled 'Share a Calendar by Publishing it Online' at **http://bit.ly/2oCrLzn** .

Once your calendar has been shared online, those people who wish to edit it will need to be able to access it from their own devices. On desktop, this is a simple matter of visiting the link as you would on the Pi, by using the browser. On mobile, there are a variety of apps that can help. If the shared calendar is with Google, Android users can access it directly from their own calendar app, even if they have a different Google account, by following the instructions at **http://bit.ly/2nP8vBB**. There is an official Outlook app for Android, which allows for easy viewing and editing of Outlook calendars.

Sadly, iCloud Calendars aren't so easy to make friends with, but there are a number of third-party apps, such as SmoothSync, in the Google Play store, which enable you to synchronise calendars. On iOS, there's official Google Calendar and Microsoft Outlook, which you can sign in and view your calendars.

Calendar conundrums

If you create or change an appointment and it doesn't appear right away on everyone's device, wait for five to ten minutes before attempting troubleshooting, to let it percolate through the various layers of software. If the changes are visible on the wall calendar – that is, on the website – the issue is most likely to do with the device, not the Pi.

The rear view of the Raspberry Pi Touchscreen Display, where the Pi can be neatly housed. The stand is optional

Google Calendar allows you to set up multiple calendars, which can be colour-coded to differentiate between them

The software used to view the calendar are very easy to install, so the most problematic part of this project is likely to be when it comes to adding the monitor and fixing it to your wall.

You can make life much easier for yourself by buying a monitor specifically designed for the Raspberry Pi, so you have somewhere to put the computer itself – in other words, tucked away tidly behind the screen.

If the place you want to install the wall calendar is hard to reach, you may be able to buy a longer micro USB cable, but bear in mind that the voltage drops as cable length grows. Consider using shorter cables and/or a powered USB hub.

If the Pi crashes for any reason, Firefox will attempt to restore all open web pages once it reboots, which may mean you have to plug in a mouse or keyboard to close down any extra tabs. You can reduce the chance of this happening by starting Firefox in safe mode, and then entering about:config in the address bar. Press Return to be taken to the settings screen for Firefox. Once you're there, scroll down to the setting marked "Browser.sessionstore.resume_from_crash" and double-click to change from True to False.

Be aware, if you're using Google Calendars, anyone who scrolls to the top of the screen will be able to switch from your calendar to your other Google services, such as Gmail or YouTube. They can also use the search bar to view documents stored in your Google Drive. If this concerns you, consider setting up a dedicated Google account, just for the calendar. You can still access and edit the calendar from your own account.

CALENDAR TWEAKS AND TIPS

When the calendar is on your wall, the cursor can look untidy, especially if you are using a touchscreen. A handy app named Unclutter can hide the cursor except when it's being moved or you're touching the screen. Open Terminal on your Pi (or connect via SSH) and run the command: sudo apt-get install unclutter.

In case the Pi crashes and you're forced to reboot, it's also best to have Firefox programmed to open automatically, saving you the trouble of reconnecting a keyboard and mouse. Open Terminal on your Pi (or connect via SSH) and run the command sudo nano /etc/xdg/lxsession/LXDE-pi/ autostart . Scroll to the bottom of the window and add the line @firefox-esr . Press Ctrl+X, then Y, then Return to save your changes.

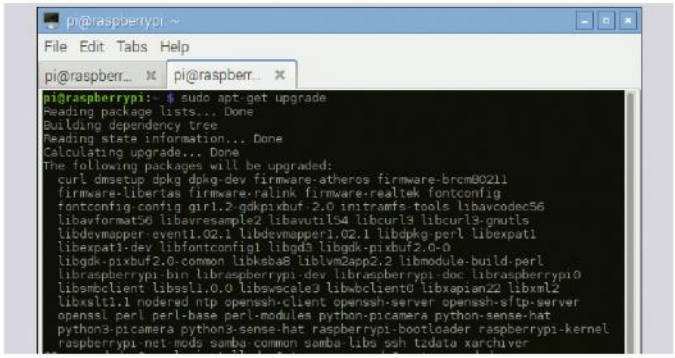
Finally, to make sure the display doesn't sleep after a few minutes, open Terminal or connect via SSH once again and run the command sudo nano /etc/lightdm/lightdm.conf . Scroll down to where it says #xserver-command=X and remove the hash at the start of the line. Next, put a space after the letter X and type -s 0 –dpms .

Press Ctrl+X, then Y, then Return to save your changes, then reboot your Raspberry Pi.

If you are using Google Calendars, click the arrow beside Other Calendars, then Browse Interesting Calendars to see a list of calendars to which you can subscribe – for example, Public Holidays in the United Kingdom. Click Subscribe to have them appear on your own calendar.

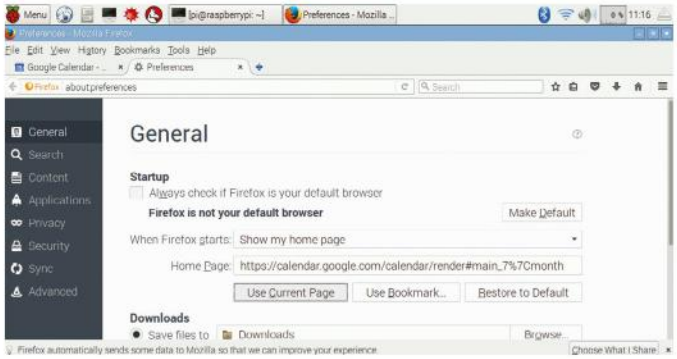
Google lists any calendars to which you subscribe

SET UP YOUR WALL CALENDAR



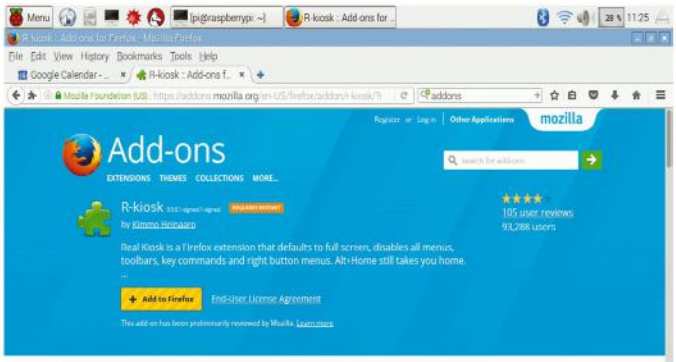
1 Update Raspbian and install Firefox

Before you can physically transform your Raspberry Pi into a wall calendar, you need to connect the device to the internet and open the Terminal app. Run sudo apt-get update and then sudo apt-get upgrade to bring your Pi up to date. Next, enter sudo apt-get install iceweasel to install Firefox Extended Support Release on to your mini computer.



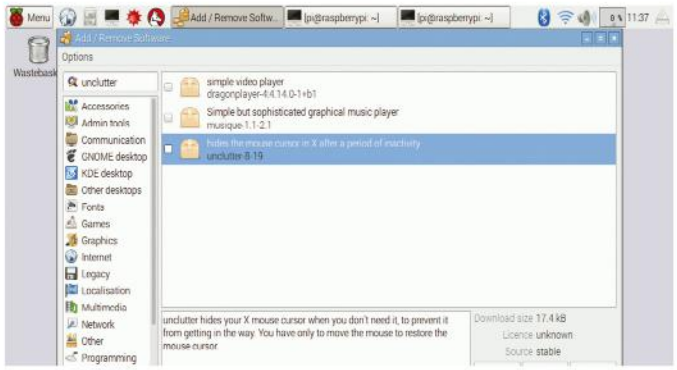
2 Set Firefox preferences

Go to Menu > Internet > Firefox ESR to open Firefox. Visit your calendar address – **http://calendar.google.com**, for example – and sign in if necessary. If Firefox prompts you to remember your preferences, say Yes. Once you can see your calendar, go to Edit > Preferences, and click the Set to Current Page button to make sure Firefox always displays the calendar.



3 Set Firefox for full-screen

This step is optional but recommended. Visit **https://addons.mozilla.org/en-US/firefox/addon/r-kiosk** to install the Real Kiosk add-on – this disables menus and toolbars. Firefox needs to restart for it to work. Remember you can still close the window by connecting a keyboard to the Pi and pressing Alt+F4.

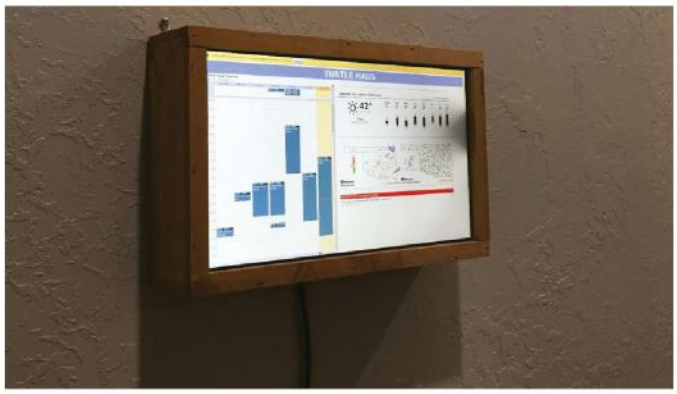


4 Perform tweaks

Make sure the calendar is in the view you want – for example, Monthly. Next, follow the steps in the Tweaks and Tips box (opposite) to hide the mouse when not in use, disable the Pi's sleep function and make Firefox start every time you switch on the machine if you wish. Restart the Pi to ensure your changes have taken effect.

5 Connect it up

Now for the hardware-dependent bit: connecting everything. Obviously, the specific steps required to connect your monitor are going to vary from device to device. If you are using the official Raspberry Pi Touchscreen Display, assembly instructions are available from **http://bit.ly/2nR9qRs**.



6 Finishing touches

Once you've connected your Raspberry Pi up to a screen, you'll want to position it somewhere that everyone can access. If it's in your home, you'll probably also want to make it look as good as possible. You can let your imagination run wild here – take a look online to see what other people have achieved.

Build your own networked hi-fi

Put the Pimoroni pHAT DAC together with a Pi Zero to create a networked hi-fi

Pi bites

Auto-starting on Raspbian. In Raspbian/Jessie the controversial systemd software was added, giving a highly modular way of managing start-up scripts amongst other things. While systemd configuration files are now best practice, they can take time to fully understand. For that reason we would suggest using cron to start the script on reboot as a temporary measure.

```
crontab -e
@reboot /usr/bin/python /home/pi/pyPlaylist/app.py
```

We will show you how to create a high-quality networked music player that takes advantage of the UK's online radio stations, Linux's popular Music Player Daemon, and a responsive web-server to control it all. The full-sized Raspberry Pis have two built-in audio outputs: audio over HDMI cable and a 3.5mm headphone jack that can suffer interference and noise. The Pi Zero itself has no audio jacks but Pimoroni has come to the rescue and built a high-quality DAC (digital audio converter) using the same chip as the Hi-Fi berry (PCM5102A).

1 Soldering the headers

The pHAT DAC comes with a 40-pin header, which you will need to solder. We consider a flux pen, work-lamp and thin gauge 60/40 solder essential for this. An optional RCA jack can also be bought fairly easily to give a phono-lead output for older stereos.

2 Install drivers

The DAC relies on I2C, so we have to load some additional kernel modules. If you are running Raspbian then you can type in the following for a one-script installation over secure HTTP:

```
curl -sS https://get.pimoroni.com/phatdac | bash
```

While HTTPS provides a secure download, curious types may want to review the script before running it.

3 Installing Music Player Daemon (MPD)

Now install the MPD package and enable it to start on boot. MPD will be the backbone of the project providing playback of MP3s and internet radio stations. The MPC (client) software is also installed for debugging and setting up your initial music playlists:

```
sudo apt-get install mpd mpc
sudo systemctl enable mpd
```

4 Clone and install pyPlaylist web-server

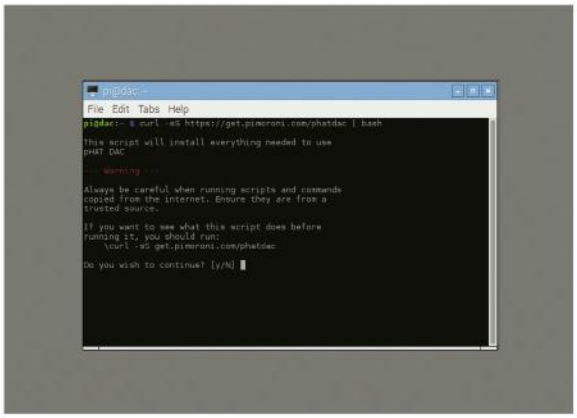
pyPlaylist is a responsive (mobile-ready) web-server written with Python & Flask web framework. Once configured it will give us a way of controlling our Hi-Fi through a web-browser. The following will install pyPlaylist on Raspbian:

```
sudo pip install flask python-mpd2
cd ~
git clone https://github.com/alexellis/pyPlaylist
cd pyPlaylist
./raspbian_install.sh
```

5 Choosing the radio stations

We have put together a list of popular radio stations in the UK which can be run into MPD with the add_stations.sh file. Edit this file or find your own from **radiofeeds.co.uk**.

```
cd ~/pyPlaylist
./add_stations.sh
```



Installing drivers is surprising simple

6 Reviewing the stations

Each station is added into its own playlist – the mpc ls command shows you which playlists are available to play:

```
$ mpc ls
BBC6Music
BBCRadio1
BBCRadio2
BBCRadio4
CapitalXtra
KissFM
```

If you want to remove one of the stations then type in the following:

```
mpc rm BBC6Music
```

7 Starting the web-server

Now that we have some stations, we can run the web-server from the pyPlaylist directory. Then open up a web browser to start playing a radio station. The following command reveals your IP address on Raspbian:

```
$ ./raspbian_get_ip.sh
192.168.0.20
```

Once you know the IP address, connect to the URL in a web-browser on port 5000, for example:

```
http://192.168.0.20:5000/
```

8 Add a custom music playlist

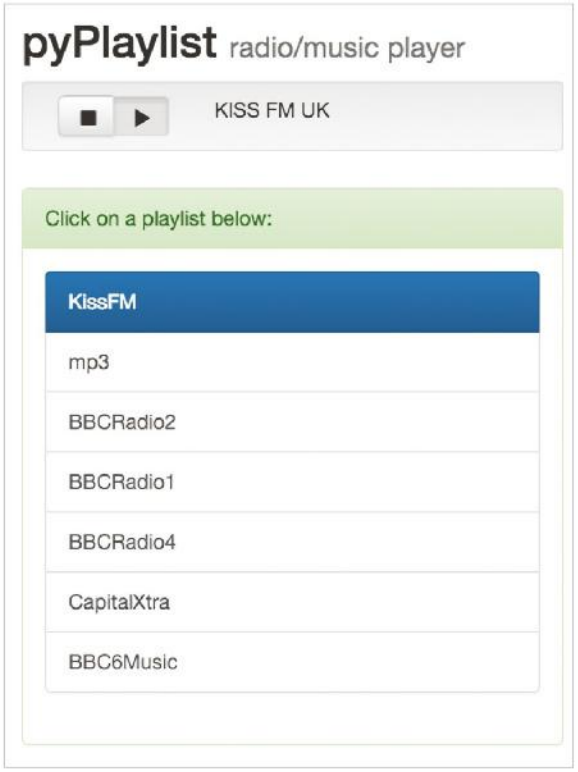
Now put together a sub-directory with your music files under /var/lib/mpd/music/ and ensure that mpd:audio has access to read it. Then we: update mpd's database, clear out the current playlist and add in all the tracks from the new directory (ambient) finally saving it as a brand new playlist.

```
mpc update
mpc clear
mpc ls ambient | mpc add
mpc save ambient
```

pyPlaylist is a responsive and mobile-ready web server

9 Finishing up

Now your music player is functioning, all that's left to do is to add some speakers, obviously! Almost anything with a RCA or 3.5mm input source will work just fine for this purpose. That part we will leave up to you. Now get ready to turn up the volume and enjoy the tunes!



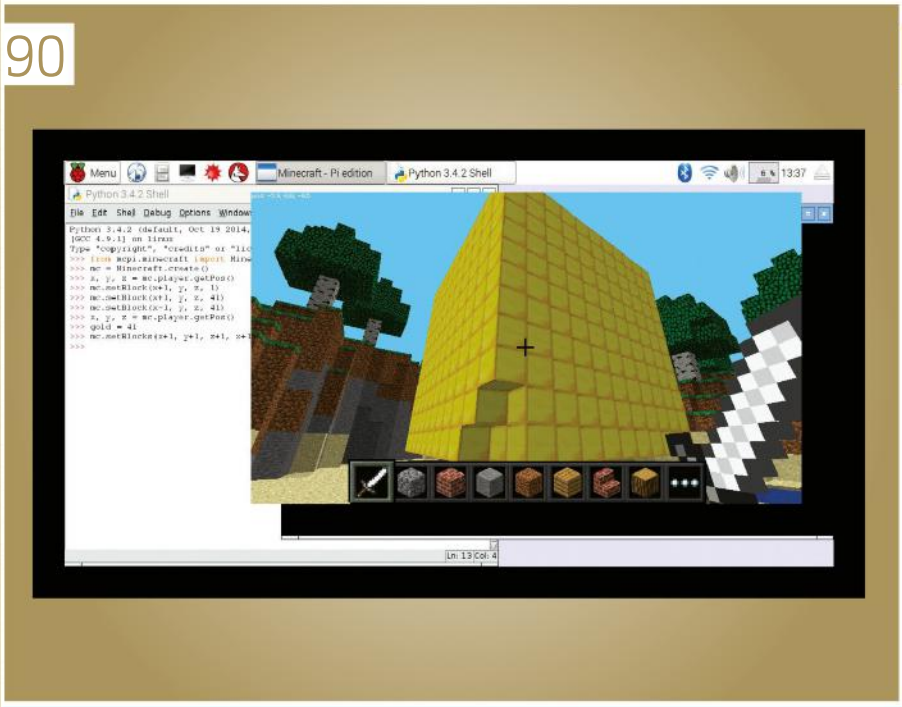
Once you're complete, it's easy to stream your favourite radio stations wirelessly to your hi-fi system

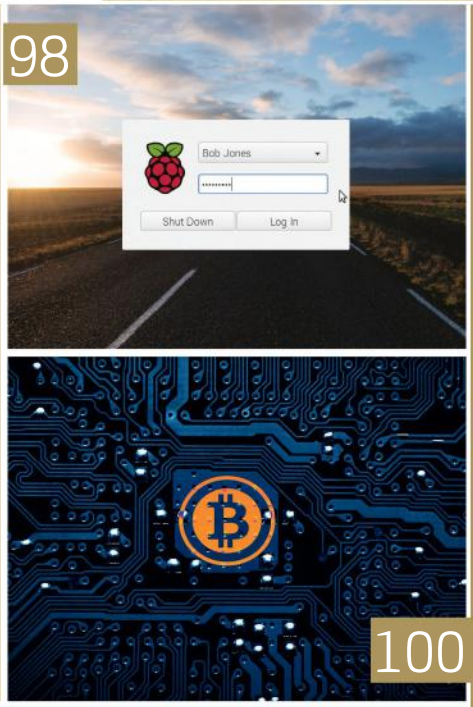
Pi bites

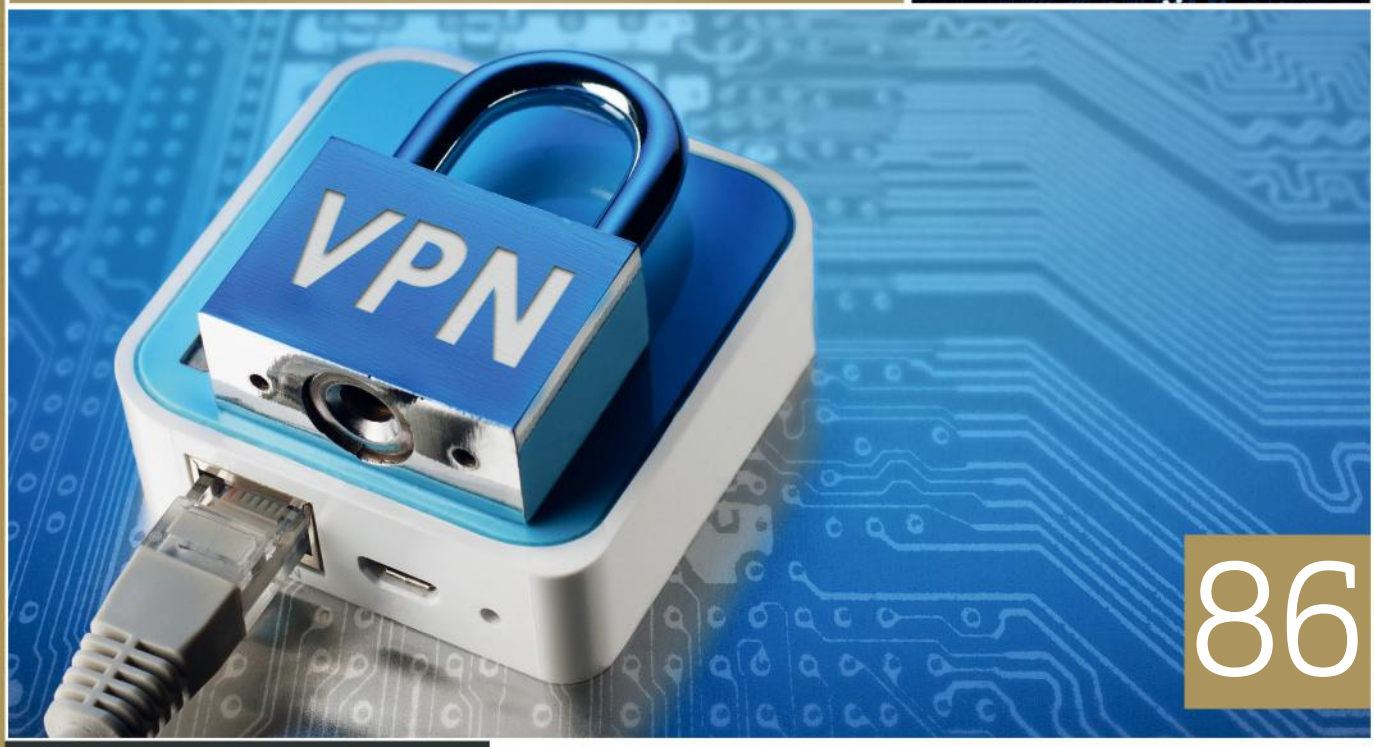
We wrote pyPlaylist with the Python flask framework which is an ideal starting-point for simple RESTful websites. The front-end code saves the screen from completely reloading by using jQuery to update the song or radio information. Bootstrap has been employed to make the pages responsive (compatible with your PC, phone and tablet). The code has been released under GPL, so why not fork the code and tweak it to your own needs?

Pi CODE

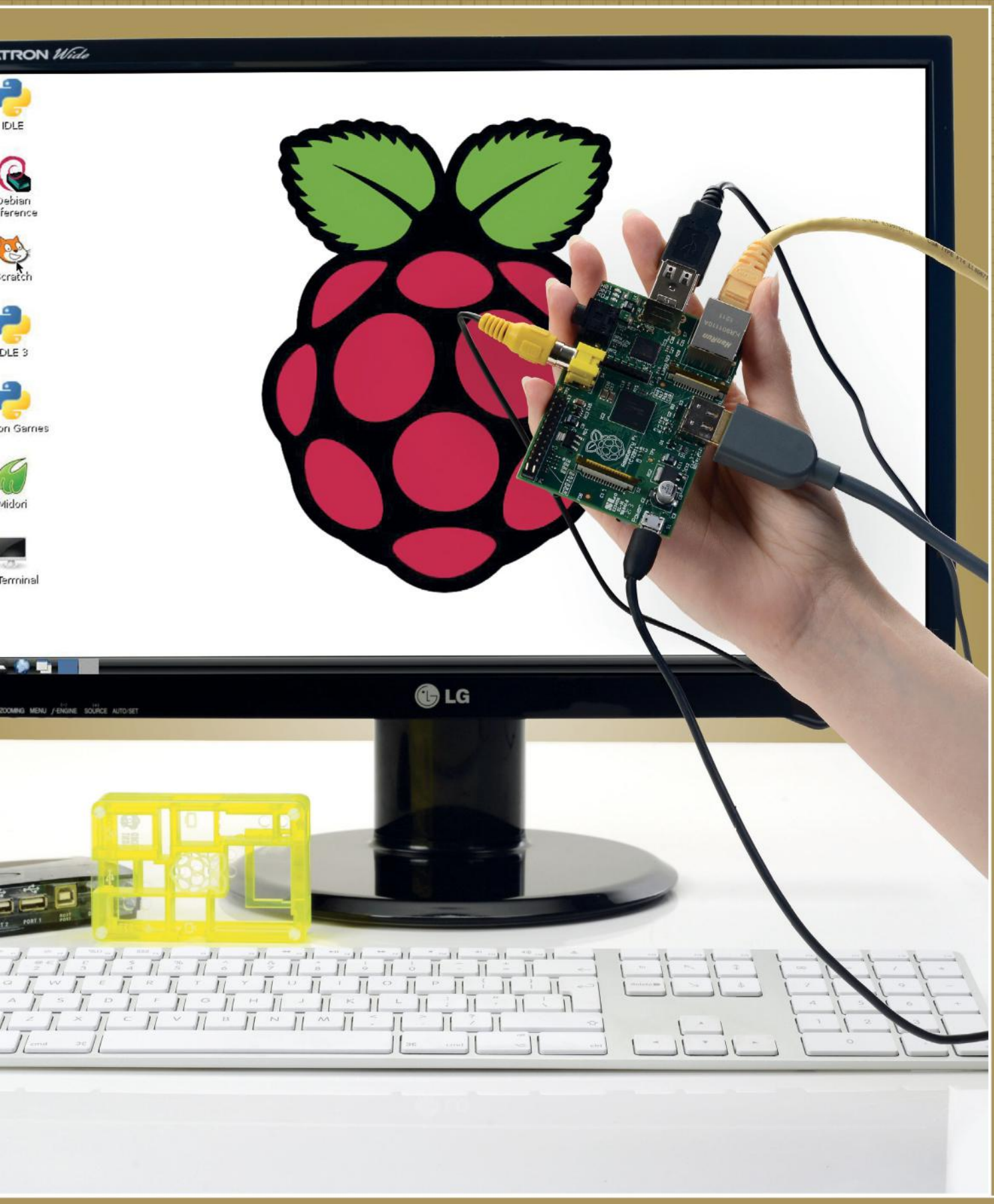
Learn how to code in your language of choice

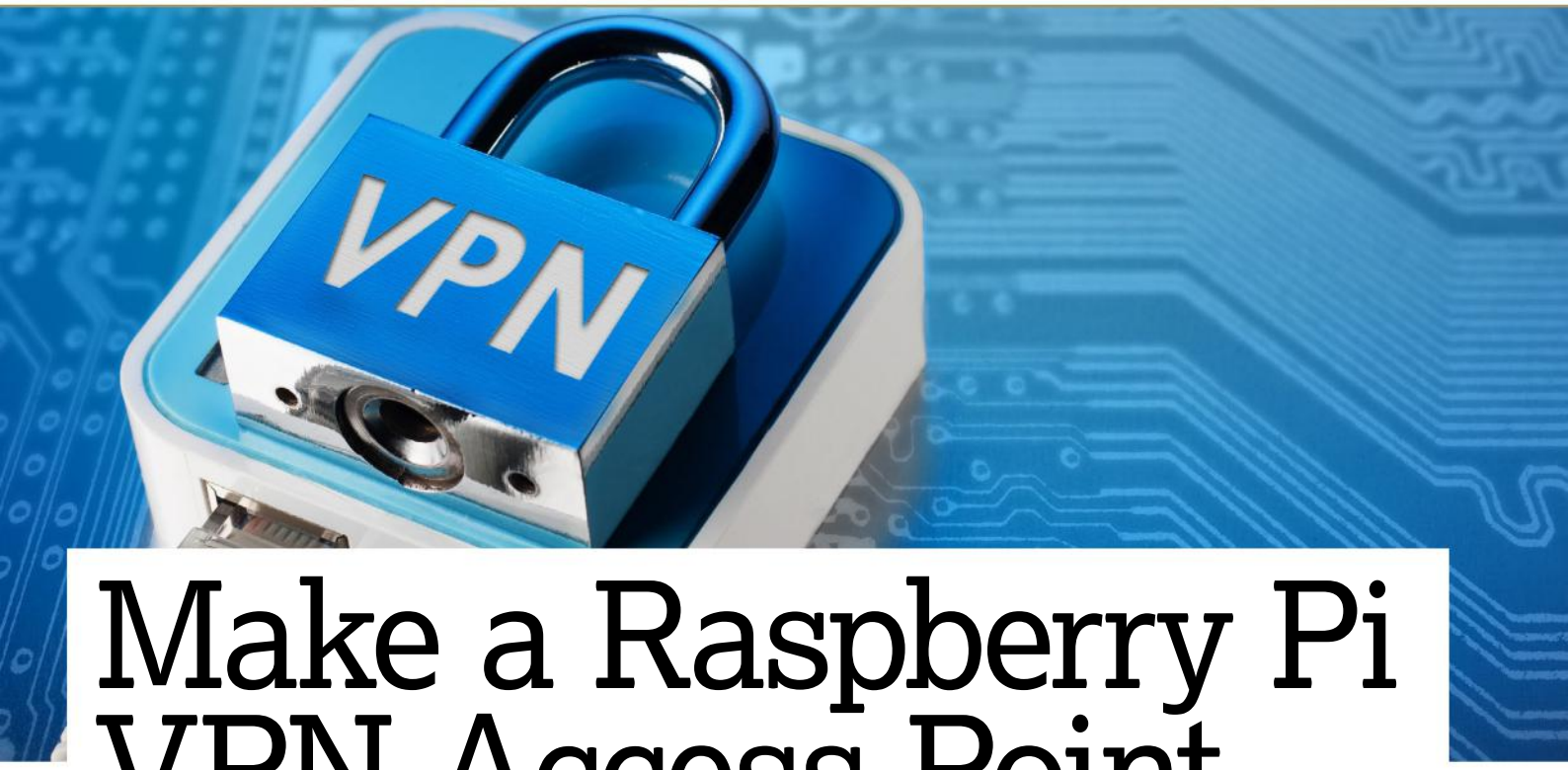






GET MORE FROM YOUR PI WITH THESE PYTHON PROJECTS


IDLE
Scratch
Midori
Terminal
LG



Make a Raspberry Pi VPN Access Point

Use your Pi as a Wireless AP that's connected to your VPN 24/7

RECIPE

- Raspberry Pi with Wi-Fi support
- Ethernet cable
- An active VPN subscription that supports OpenVPN
- Secondary device such as a laptop to test your AP works

Anyone who has been abroad will know it's a constant pain to have to deal with content that has been switched to the local language, not to mention certain states that censor internet content. Fortunately there's a way to evade these restrictions using the Raspberry Pi. In a few simple steps you can configure your Pi to generate its own wireless AP (Access Point) and keep it permanently connected to a VPN (Virtual Private Network) service. All you need to do when travelling is bring your Pi and connect it to a working router and you'll have your own private wireless network and connection.

VPNs were originally designed to allow office workers to connect to their corporate intranet while away, over an encrypted connection. These days they're more commonly used both to protect your connection and make it seem as if your computer is located in another country.

In order to proceed with this project, you will require an active VPN subscription and the client configuration (.conf) file to automatically connect.

1 Choose your VPN

In order to carry out this project, you need an account with a VPN provider. Find a provider that supports the OpenVPN protocol, as this connection is generally considered to be the most secure. Free providers won't require any billing information but they are not as fast or reliable as paid services. If you choose a paid provider, try to find one that accepts anonymous payment methods such as Bitcoin. The website www.weusecoins.com has a list of these.

2 Download configuration

Attach the Pi to your router. Open Terminal or connect via SSH. If your VPN provider supports the secure OpenVPN standard, then they will have provided a configuration file with the extension .conf or .ovpn. For this tutorial a VPN configuration file from free provider VPNBook was used. Download the file to your Pi either by clicking the link or using wget in Terminal:

```
wget http://www.vpnbook.com/free-openvpn-account/VPNBook.com-OpenVPN-Euro1.zip
```

3 Install OpenVPN

The Raspbian repositories contain the OpenVPN software but not the most current version. Use the command su to switch to the root user then run these commands:

```
wget -O - https://swupdate.openvpn.net/repos/repo-public.gpg|apt-key add -
echo "deb http://swupdate.openvpn.net/apt jessie main" > /etc/apt/sources.list.d/swupdate.openvpn.net.list
apt-get update
apt-get install openvpn
```

Run openvpn –version to double-check you have the most up-to-date version of the software. At the time of writing this is 2.3.14.

4 Configure and run OpenVPN

Use the mv command to move the configuration file into the openvpn folder /etc/openvpn, amending the extension if necessary, for instance:

```
sudo mv vpngate_vpn151111650.opengw.net_
udp_1344.ovpn /etc/openvpn/vpn1.conf
```

Next, start the OpenVPN service with the command sudo service openvpn start. Start OpenVPN using your .conf file with the openvpn –-config command, for instance:

```
sudo openvpn -config /etc/openvpn/vpn1.conf
```

Next, run the command:

```
sudo service openvpn start
```

5 Test OpenVPN connection

Once the OpenVPN service is running, open a new tab in your Terminal or start a new SSH service and run the command ifconfig to list your network interfaces. Usually the VPN connection will appear as tun0. You can check your apparent location with the command:

```
curl --interface tun0 freegeoip.net/json/
```

Next, make sure the OpenVPN service starts each time you log in. Then run

```
sudo nano /etc/default/openvpn
```

Remove the # at the start of the line reading "#AUTOSTART="all".

6 Install prerequisites

Now we'll install the necessary software to set up a Wireless AP. Do this by running:

```
sudo apt-get install dnsmasq hostapd
```

Next, run:

```
sudo nano /etc/dhcpcd.conf
```

You you've done that, add these lines to the very bottom of the file:

```
interface wlan0
static ip_address=172.24.1.1/24
```

Press Ctrl+X, Y then Return to save and exit.

7 Set static IP

The next step is to open your network interfaces configuration with:

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

Change the line "iface wlan0 inet static" to "iface wlan0 inet manual". Press Return to start a new line, and then paste:

```
address 172.24.1.1
netmask 255.255.255.0
network 172.24.1.0
broadcast 172.24.1.255
```

Place a '#' at the start of the line beginning "wpa-conf". Save and exit in the same way as before. Restart the dhcpcd service with:

```
sudo service dhcpcd restart
```

8 Configure AP

Run the command:

```
sudo nano /etc/hostapd/hostapd.conf'
```

Paste the following:

```
interface=wlan0
driver=nl80211
ssid=piVPN
hw_mode=g
channel=1
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=raspberry231
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Feel free to change the name of the network from 'PiVPN' to one that is meaningful to you. Similarly change the password 'raspberry231' to something more secure. Next run:

```
nano /etc/default/hostapd
```

Find the line starting #DAEMON_CONF="" and change to DAEMON_CONF="/etc/hostapd/hostapd.conf. Note the '#' at the start of the line must be removed.

9 Configure dnsmasq

Move the old dnsmasq configuration file with:

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.
orig
```

Then create a new one by running:

```
sudo nano /etc/dnsmasq.conf
```

Paste in the following text:

```
interface=wlan0
listen-address=172.24.1.1
bind-interfaces
server=8.8.8.8
domain-needed
bogus-priv
dhcp-range=172.24.1.50,172.24.1.150,12h
```

Note we are using Google's DNS server (8.8.8.8) for now; change this if you wish, then save and exit. Run:

```
sudo nano /etc/sysctl.conf
```

»

"PROTECT YOUR CONNECTION AND MAKE IT SEEM AS IF YOUR IS IN ANOTHER COUNTRY."

Find the line starting "net.ipv4.ip_forward=1" and remove the '#' at the start. Now reboot the Pi.

10 Set up IPV4 Forwarding

For the next step, you need to run each of these commands individually:

```
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
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Next, run the command:

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

Paste the following right above the line reading "exit 0":

```
iptables-restore < /etc/iptables.ipv4.nat
/usr/sbin/hostapd /etc/hostapd/hostapd.conf
```

11 Test Access Point

Run the commands:

```
sudo update-rc.d hostapd enable
```

And:

```
sudo update-rc.d dnsmasq enable
```

Reboot the Pi. You'll need a second device at this stage to see if you can access the Wireless AP. Search for it in your network menu and enter the password you created earlier on. If you can't remember this, run the following on the Raspberry Pi to view it again:

```
sudo nano /etc/hostapd/hostapd.conf
```

Once connected, visit **www.whatismyipaddress.com** to check you're behind the VPN.

12 Fix DNS Leaks

Certain VPN providers use their own DNS servers. Other VPN Providers are less cautious. Visit **https://www.dnsleaktest.com/** and click Extended Test to check you're safe. If any of the DNS servers match your regular ISP, your connection is not fully secure.To resolve this, edit your VPN configuration file, for example:

```
sudo nano /etc/openvpn/vpn2.conf
```

Once you've done that, add these lines immediately above "<ca>":

```
script-security 2
up /etc/openvpn/update-resolv-conf
down /etc/openvpn/update-resolv-conf
```

Save and exit, then restart your Pi. Check the DNS leak website once again. If this fails to resolve the issue, try using another VPN provider.

13 Block unsolicited connections

As your Pi is sitting between your computer and the internet, it can potentially be accessed by other devices. Prevent unsolicited incoming connections from other devices with the following commands:

```
sudo iptables -A INPUT -i tun0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A INPUT -i tun0 -j DROP
```

Make sure to save your changes so that they'll apply on reboot:

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

14 Route all traffic through OpenVPN

When you boot the Pi initially, certain applications may try to connect directly to the internet, which can undermine your anonymity. To channel all network traffic through the VPN, you need to edit your configuration file in /etc/openvpn, for instance by running:

```
sudo nano /etc/openvpn vpn2.conf
```

Make sure the line "redirect-gateway" reads "redirect gateway def1". DNS queries will also be routed through the VPN also so make sure your provider supports this.

15 Set up Firewall

Although you may have previously configured iptables to prevent unsolicited incoming connections, to be on the safe side, consider installing ufw (Uncomplicated Firewall) with

```
sudo apt-get install ufw
```

Run the command:

```
sudo ufw enable
```

To fire it up, then open the default OpenVPN port 1194 with:

```
sudo ufw allow 1194
```

Once you've done this, you may also want to enable Port 22 to allow connecting via SSH. Remember that this doesn't change which ports are open and closed on the router.

Pi bites

If you need a username and password for your VPN, you can save these so OpenVPN will connect automatically. First run:

sudo nano auth.txt

On the first line put the username and on the second put your password. Save and exit. Next edit your OpenVPN config file, for instance:

sudo nano /etc/openvpn/vpn2.conf'

Scroll down to the line with the text "auth-user-pass". Leave a space and enter the path auth.txt, for example:

auth-user-pass /home/pi/auth.txt

Save and exit once again.

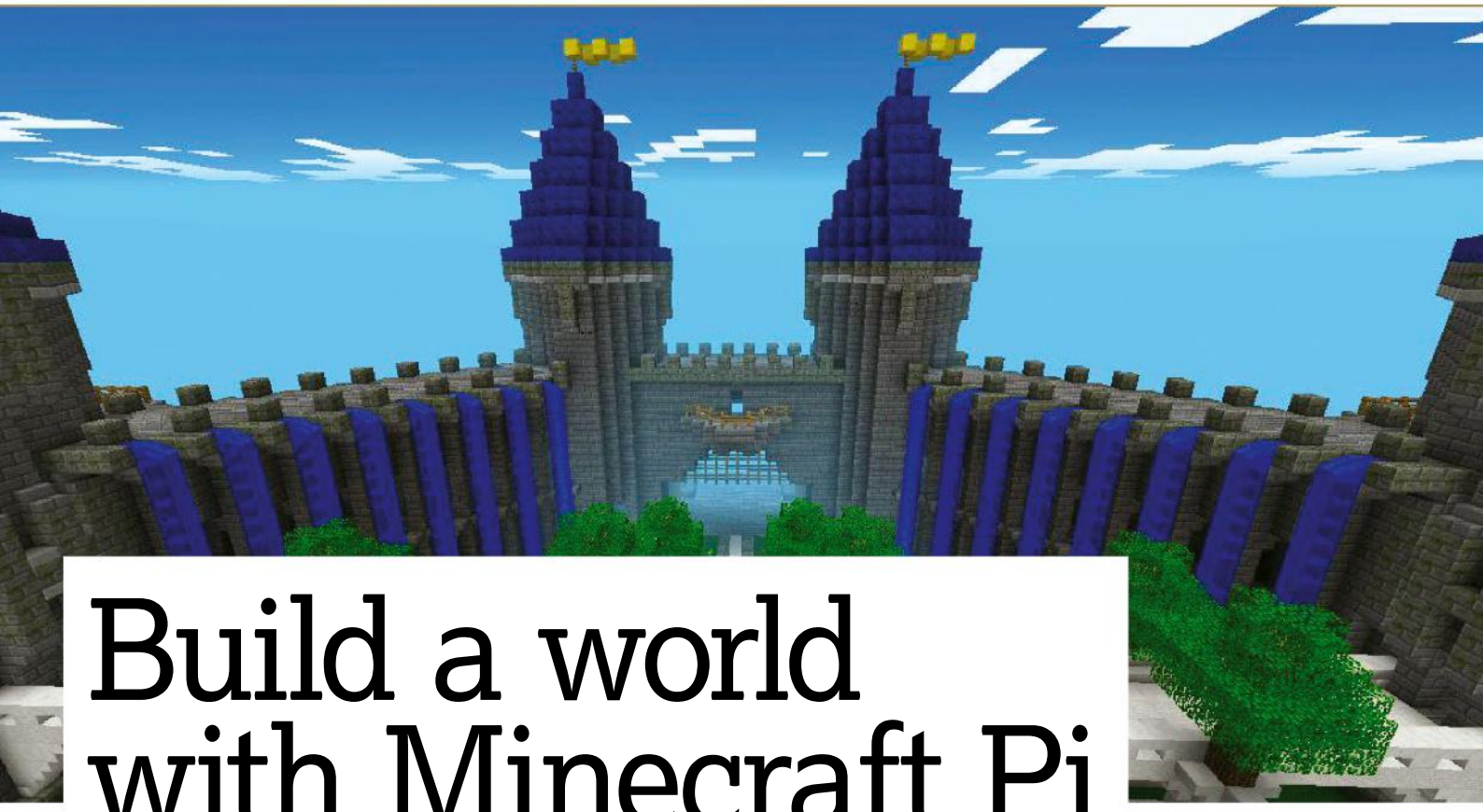
THE TOP TECH IGNITING YOUR WORLD TODAY!

DID YOU MISS OUR HOT 100 ISSUE?

DOWNLOAD THE DIGITAL MAGAZINE NOW

Available on the App Store

FIND IT IN THE T3 MAGAZINE APP



Build a world with Minecraft Pi

Jonni Bidwell invites you into the amazing and creative world of Minecraft. Then reveals how you can host your own server to keep out the riff raff.

Besides empowering a new generation of makers, the Raspberry Pi has also established its merits as a gaming platform. Projects like RetroPie, combined with the Pi's cheapness, mean that it's great for playing old classics. And even some new ones. Like Minecraft.

In case you've been living under a rock since 2011 Minecraft is the best-selling computer game of all time. It's an open-world, sandbox affair in which our hero, Steve, roams around a landscape made of voxels (blocks). Resources can be mined and crafted into other resources, so that Steve can build houses, farm crops, cook food and fight enemies (all rendered in low-res splendour).

A special version, Minecraft: Pi Edition, was released in 2013 and has been bundled with Raspbian since 2014. This version is in some ways cut down, in the sense that there isn't any crafting or baddies, and nor is there any limit on the resources available for building. But it does have some features that aren't present in the full game, most notably a Python API for manipulating the world while a game is being played. Not having to worry about creatures trying to kill you or where your next meal is coming from makes it great for younger players, emphasising the creative aspects of the game. The Python API makes it an ideal platform for aspiring coders, young and old.

To get started, first head over to the Menu, then go to Games>Minecraft Pi. Choose Single Player and then New Game. You'll see that the game window is slightly offset from the application window behind it. Make sure your mouse arrow is parallel with the top of the app window, and hold down to reposition it.

Click Start Game, followed by Create New. Your new world now loads. Feel free to explore and build. The controls for the game are pretty simple, and are as follows:

Key	Action
W	Forward
A	Left
S	Backward
D	Right
E	Inventory
Space	Jump
Double-space	Fly/fall
Esc	Pause/Game menu
Tab	Release mouse cursor

You can use the mouse to look around you, too. It can also be used to select items from the inventory. By default, you're holding a sword. Click blocks to destroy them. The sword can also be used to dig. If you select a block from your inventory, you can begin building. Use the right mouse button to place down the block, or the left to destroy it.

Coding blocks

To start coding in Minecraft: Pi Edition, press the Tab key while it's running. Go to Menu>Programming. Click Python 3. Try to place the text boxes so they're next to one another. You can type commands here to change the in-game world. Over time, you can also write scripts to automate tasks for you – for example, a script that places a stone block wherever you walk. So, without further ado, let's reprogram the game to display a simple message. Type in the following commands, pressing Return after each one:

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
mc.postToChat("Hello world, I'm playing Minecraft Pi!!!")
```

Press Return to display your first in-game message. Next, look at the top-left of the Minecraft Pi screen. You can see your X, Y and Z coordinates. This is an excellent way to work out your location, as well as place blocks precisely.

You can use Python to summon any number of blocks of various materials. First, let's try placing a single gold block immediately behind you. Type in the following command:

```
x, y, z = mc.player.getPos()
```

You'll need to run the above command each time you move, to give the game your updated coordinates. Press Return, then type:

```
mc.setBlock(x+1, y, z, 41)
```

The number 41 at the end of the last command represents the ID number for gold. Each block or item in the game has a unique ID. If you know the ID of a particular type of block, assign it a name to make it easier to remember. For example:

```
gold=41
```

Press Return, then run these commands to create a giant 10x10x10 cube of solid gold:

```
x, y, z = mc.player.getPos()
mc.setBlocks(x+1, y+1, z+1, x+11, y+11, z+11, gold)
```

This is just a small selection of commands that can be run on Minecraft Pi. It's also possible to form complex shapes, teleport the player, and even drop blocks as you move around. For a full list of commands that you can run in Minecraft Pi, as well as the ID numbers for each type of block, head to **http://bit.ly/2uAAbwU**.

Avoiding the minefields

As we've discussed, Minecraft Pi is very similar to the Pocket edition of Minecraft, which can be played on mobile handheld devices such as Android phones and iPhones. It can't interact with people using the official Minecraft client. According to reports on the Mojang website, it appears the client hasn't been updated in a while, but if you simply want to make simple structures and learn to code, without the extra features of the full Minecraft client, then it's ideal.Although we've seen it's possible to get the full version of Minecraft 1.8.9 running on a Raspberry Pi 3 (see

When you get tired of your Midas touch, try repeating the cube command with a different block ID to change the building material

Install the full Minecraft game on the Pi 3

Open Terminal on your Pi and begin by installing some tools:

```
sudo apt-get -y install xcompmgr libgl1-mesa-dri && sudo apt-get -y install libalut0 libalut-dev && sudo apt-get -y install mesa-utils
```

Next, we'll need to enable an experimental graphics driver. Run sudo raspi-config and scroll down to Advanced Options. Press Return. Then choose the AB – GL Driver option and enable it. Now download the official Minecraft client:

```
mkdir ~/Minecraft; mkdir ~/Minecraft/Natives;
cd ~/Minecraft && wget https://s3.amazonaws.com/Minecraft.Download/launcher/Minecraft.jar
```

Launch it by running:

```
$ java -jar Minecraft.jar
```

Log in with your username and password. Next, click the Profile Editor tab, then click the box that's underneath Version to open up a new window. Find the drop-down menu named Use Version and choose 1.8.9. Next click Save Profile. Click the Play button to begin downloading the files. When the downloads are finished, you need to close the launcher and re-open Terminal. Run the following commands:

```
$ cd ~/Minecraft/Natives && wget https://www.dropbox.com/s/4oxcvz3ky7a3x6f/liblwjgl.so
$ wget https://www.dropbox.com/s/m0r8e01jg2og36z/libopenal.so
$ cd /home/pi/.minecraft/libraries/org/lwjgl/lwjgl/lwjgl/2.9.4-nightly-20150209 && rm lwjgl-2.9.4-nightly-20150209.jar
$ wget https://www.dropbox.com/s/mj15sz3bub4dmr6/lwjgl-2.9.4-nightly-20150209.jar
$ cd ~/Minecraft/
$ wget https://www.dropbox.com/s/jkhr58apwa7pt1w/run.sh
$ sudo chmod +x run.sh
```

Next, we need to make some changes to one of the files we've downloaded with the following:

```
$ sudo nano ~/Minecraft/run.sh
```

Scroll down and after the = enter your login details: email, username and password. Press Ctrl+X when you're done, then press Y and Enter to confirm. To run Minecraft at any time, just enter the following command: cd ~/Minecraft && ./run.sh.

the boxout, on previous page), the method isn't officially supported by Mojang. This means that you're likely to see reduced performance over a regular desktop machine, such as glitches and crashes some of the time.

The experimental graphics driver that has to be enabled on the Raspberry Pi to use the full version of Minecraft stops the official Minecraft Pi client from working, but you can disable it again by running sudo raspi-config and then rebooting the machine.Neither version of Minecraft plays particularly well if you're accessing the Pi over VNC, so it's best to play it directly.

If you wish to play the full version of Minecraft, you do need to purchase an account with Mojang (currently priced at £17.95). If you now have the Minecraft bug and want to take it further, read on for details of how to set up your own Minecraft server on the Pi, and build your own digital world block by block.

Pi bites

If you need extra space, Minecraft can run on an external drive. Simply close the game and drag the Minecraft folder across to the drive.

Minecraft server

On the previous pages, we discussed how to play Minecraft directly on the Pi, partly for fun and partly as an introduction to programming with Python. This part of the tutorial concerns installing the Minecraft server software on the Raspberry Pi 3. This enables others running the Minecraft client on their computers to connect to your own online world and play free from thieves and 'griefers'.

Minecraft runs in Java, which is pre-installed in the latest version of Raspbian, meaning it's easier than ever to set up a server. The server has fairly low requirements, but for best performance, we recommend using a Raspberry Pi 2 or 3. With your Pi in hand and a fresh install of Raspbian, connect to it via SSH. We explained the process in LXF224. Note that the default username and password is raspberry and pi, respectively. Make a note of your Pi's IP address, because you'll need it later. Next, create a directory for Minecraft and open it with the following command:

```
$ mkdir minecraft && cd minecraft
```

Then download the latest version of SpigotMC, a highly customisable and lightweight version of Minecraft Server:

```
$ wget https://hub.spigotmc.org/jenkins/job/
BuildTools/lastSuccessfulBuild/artifact/target/
BuildTools.jar
```

Next, tell the Pi to start building the server tools:

```
$ java -jar BuildTools.jar
```

Once this process is 100 per cent complete, you will see a message stating 'Saved as spigot-1.x.x.jar', by '1.x.x' we mean the current version number of Spigot that was downloaded (this was 1.11.2 at the time of writing). Next, start the Minecraft server by running the following code:

```
$ java -jar -Xms512M -Xmx1008M spigot-1.x.x.jar
nogui
```

Again, substitute 1.x.x with the version number of Spigot. The program tells you that you need to agree to the EULA in order to run the server. To do this, type:

```
$ nano eula.txt
```

Scroll down with your arrow keys and delete the word
false . Replace it with TRUE . Press Ctrl+X, then Y and Return to save your changes.Start the server software again with the same command as beforehand:

```
$ java -jar -Xms512M -Xmx1008M spigot-1.x.x.jar
nogui
```

You'll see a message stating 'Loading Libraries, please wait...' some basic information about the game displays. Note that the default game mode is Survival, and you then see the spawn area loading slowly as a percentage. This may restart a couple of times as various levels are spawned. You'll see a message stating 'Done!' once the process is complete.

Now it's time to test your server. Go to a computer with the Minecraft client installed and start the program. On the main screen, choose Multiplayer. Next, click Add Server. Click the box marked Server Name and type a name of your choice – for example, Pi Minecraft Server. Click the box marked Server Address and enter the IP address of your Pi. Click Done when complete. You should now see the Minecraft server listed.

Hover your mouse over it and click the blue Play button. When the game loads, feel free to wander around a little and even work through your anger issues by smashing a few blocks to make sure the game responds well. Once you're satisfied, close the Minecraft window and return to your SSH client, which is still connected to the Pi. Type stop for the time being to stop the server.

You can type stop at any time to halt the server if, for instance, you need to make any changes. In future, if you wish to run the server software, connect to the Pi via SSH, then use the following command:

```
$ cd ~/minecraft && java -jar -Xms512M
-Xmx1008M spigot-1.x.x.jar nogui
```

Again, replace 1.x.x with the actual version number of Spigot you currently have installed. If you're unsure of this, type the command cd ~/minecraft && ls to view the contents of the minecraft folder – you can then see the version of spigot-1.x.x.jar in there.

Basic settings

Once the server is up and running, it's time to look at how you can tweak some of the settings. The basic settings are contained in a file named server.

properties. You can view and edit this file by running the command:

```
$ cd ~/minecraft && nano server.properties
```

If you're intent on fine-tuning your Minecraft world, visit **http://bit.ly/2w2Z16L** to view all values for all potential settings. For now, we're just making some basic changes. Scroll to the very bottom of the file to the value motd= using your cursor keys. This is simply the message that appears when you first connect to the Minecraft server. By default, this simply says 'A Minecraft server'. Delete the line of text and replace with the following cheery message:

```
motd=\u00A74 \u00A7l Welcome to my Pi
Minecraft Server\!
```

Another change you may want to make to the server – especially for people new to Minecraft – is to change the game from Survival Mode to Creative instead. Creative Mode allows for an infinite amount of building materials. Players also don't become hungry or lose energy, so are free to start making their constructs right away. Scroll up to gamemode=0 and change the 0 to 1 to do this. Make sure also to change force-gamemode=false to true to be certain that everyone who logs on in the future will be in Creative Mode.

If you're determined to stay in Survival Mode but want to start off slowly, consider changing spawn-monsters=true to false . You can also change the game's difficulty by changing difficulty=1 to a value between 0 and 3, with zero being the easiest difficulty level. When you are satisfied with the changes that you've made, hit Ctrl+X, then Y, then Return to save your changes and boot the server.

Advanced tweaks

As you become more comfortable with playing Minecraft, or if you're already an experienced player, you may find the basic configuration options in server.properties limiting.This is where SpigotMC really comes into its own, with an advanced configuration file. There's a number of tweaks to improve your Minecraft experience. Stop the server then run the following command:

```
$ cd ~/minecraft && nano spigot.yml
```

By way of an example of how finely tuned Spigot is, scroll down to where it says zombie-aggressive-towards-villager: true and change it to false . This setting determines whether or not zombies will attempt to kill Minecraft villagers. The Pi requires fewer AI resources for 'friendly' zombies, so this may speed up your server. A complete list of the various settings that can be changed in Spigot, along with full descriptions of what they do, can be found at **http://bit.ly/2uFP4wC.**

Server slowdown

Your configuration may now be tweaked to your heart's content, but as more players join, and more complex structures are created, you may find Minecraft slowing down. As we've previously discussed, one way to make sure there is plenty of free space is to move the Minecraft folder to an external drive. The syntax of the command is

```
$ sudo mv ~/minecraft /media/pi/MYUSB
```

where MYUSB is the name of your USB stick. Bear in mind that the above commands for running the software include the cd command, which refers to the minecraft directory in the home folder, so make sure to modify the commands accordingly, for example:

```
cd ~/minecraft becomes cd /media/pi/MYUSB/
minecraft
```

If you're still having trouble, try starting the server with the following command which, without going into technical details, instructs Java to make better use of your CPU:

```
$ cd ~/minecraft && java -jar -Xms512M
-Xmx1008M -XX:+UseConcMarkSweepGC
-XX:+UseParNewGC -XX:+CMSIncrementalPacing
-XX:ParallelGCThreads=2 -XX:+AggressiveOpts
-jar spigot-1.x.x.jar nogui
```

For even better performance, consider editing the server.properties file once again, this time reducing the view-distance from 10 to 5 . This determines how much of the map has to load as you move around. You can also alter max-players to change the number of people who can be logged in at the same time. Now you're all set up and can start playing Minecraft properly. If you decide you want to share your world, read the boxout below.

Pi bites

The strings `\u00A74 and` `\u00A7l` before the actual greeting simply change the colour and formatting of the text. For a full list of colour codes, **visit http://minecraft.gamepedia.com/Formatting_codes#Color_codes.**

Going global with Minecraft

So far in this tutorial we've explained how to set up Minecraft on your local network, which means anyone who wants to play will need to be connected to it. However, you may wish to allow others to join your server over the internet.

First, you'll need to assign your Pi a permanent IP address on your network by editing /etc/network/interfaces . Next, use a feature known as port forwarding on your router to forward incoming traffic from the outside internet to the Pi itself.

Because each router is different, it's best to check the specific steps to do this with your router manufacturer, or enlist a more technically able friend. In either case, say that you need to open port 25565 for both TCP and UDP. You also need your router's external IP address, which you can get by visiting **www. whatismyip.com**. What's more, the Port Forward website has multiple guides for setting up port forwarding: **http://bit.ly/2tG3fQG**.

You should be aware that your ISP may change the IP address of your router from time to time, which could cause difficulties for people playing. If this begins to get on your nerves, you can pay to rent a static IP from your ISP or get a free one from a service such as **www.noip.com**.

Plot your data on a Raspberry Pi

Plotly is a great framework to use when you need to see what's happening to the data that is being collected by your Raspberry Pi

There are several ways that you could use your Raspberry Pi for data collection. This might be a scientific experiment, or you may be running a weather station, or some other monitoring system. In any case, we had focused on how to do the actual data collection and had not really looked at how to make it useful or interesting for any humans who might be checking in on the system. This month, we will look at one option available to handle the visualisation of all of this incoming data, namely the Plotly module. Plotly provides a very robust set of functions and classes to generate visualisations of data. We will assume that you have some type of display or monitor attached to your Raspberry Pi, and that you have an X11 desktop set up. This way, we can focus on how to actually generate the data displays for your project. You also need the Plotly module installed. If using Raspbian, or a variant, you can install Plotly with the command:

```
sudo apt-get install python-plotly
```

If you are coding for Python 3, you can install the python3-plotly package instead. Or, if you absolutely need the latest and greatest version, you can use the following command:

```
sudo pip install plotly
```

This will install Plotly into the system Python library location.So, now that you are ready, how do you get started with Plotly? Plotly comes in two flavours: online and offline. Since we are going to use Plotly as part of a monitoring project, we will focus on using Plotly in offline mode. To use it in this fashion, you need to be sure that you are using version 1.9.x or later. You can check this in an interactive Python session with the following code:

```
from plotly import __version__print(__version__)
```

Since we will be using Plotly in an offline mode, we will need to use the offline versions of the main functions. You can import them with the following code:

```
from plotly.offline import download_plotlyjs, init_
notebook_mode, plot, iplot
```

This import statement loads the core functions that you will need. Plots are all handled by the main function, named plot. All graphs are created using this function, where the details are handled by parameters within the call. But, what can you hand in as parameters? There are several other classes that are used to create specific types of graphs. For example, if you wanted to create a scatter plot, you would need to import the relevant class, as shown below:

```
from plotly.graph_objs import Scatter
```

As a test, you can directly create and display a graph with the following code:

```
plot([Scatter(x=[1,2,3], y=[4,5,6])])
```

As you can see, the plot function takes a list of objects to graph out. In this case, we are handing in a Scatter object; this is instantiated with a list of values for the x-coordinate and a list of values for the y-coordinate. Once the plot function is done, it will try to display the generated graph as a webpage, using the default web browser as the displaying program. Assuming everything is installed correctly, you should see a fresh plot in your browser. This plot is an interactive one, enabling you to do things like zoom in on the plot, select individual data points and also either save off a static image file of your data.

But this is not the way you will likely use Plotly. Assuming that you are using your Pi to collect data, you will likely have large amounts that you'll be wanting to plot. This means that we need another method of loading data. There are several different options. If you are doing data analysis anyway, you could load the data using pandas and creating a data object that can get handed in to one of the graph objects. For example, if you had your data in a CSV file, the following code would load your data and generate a scatter plot:

```
import pandas as pd
df = pd.read_csv('my_data.csv')
plot([Scatter(x=df['x'],y=['y'])])
```

This assumes that the columns in the CSV file are actually labelled as 'x' and 'y'. You can even connect to a database to pull your data for generating your graphs.

Pi bites

It's the official language of the Raspberry Pi. Read the docs at **python.org/doc.**

You will likely also want to do updates to your plots over time as new data comes in. You can do this through either IPython or Jupyter worksheets. In this case, you first need to initialise the system so that the appropriate JavaScript code is imported into your worksheet. This is handled with the boilerplate code below:

```
init_notebook_mode()
```

Once this call is finished, you can use the function iplot(), rather than plot(), to generate and display your plots within your worksheet. If you want to be able to update this plot, you will need to include the parameter filename, and hand in a unique filename to store the graph image for display. This way, when you want to update the graph, you can regenerate the plot with the updated data, using the same filename; this new graph is then refreshed within the worksheet.

These plots are also interactive, similar to the interactivity you get in the browser-based display. The default display gives you a set of axes, labelled with the value ranges for each axis. There are several other options available to add more information to your plots. These are handled as parameters to the graphics objects, such as title, mode, markers, hover text and several others. It is well worth taking a look at the full list of parameters available when you start adding details to your plots. When you start looking at all the options, youll also see that there are a huge number of plots available. There are a series of basic plots, statistical plots, scientific charts, financial charts, maps and 3D charts. There are also custom controls that can add more interactivity to your plots.

Plotly also interacts with other Python modules rather well. In this way, you could actually have some type of numerical analysis happening within your monitoring project. As a longer example, say you had data for a pendulum. You could do a numerical integration of that data, where the time index is your x variable and y is the distance from the centre. That code would look like:

```
import numpy as np
import plotly.graphics_objs as go
trace1 = go.Scatter(x=x, y=y, mode='lines', )
dy = np.trapz(y, x)
annotation = go.Annotation(x=4.5, y=1.25,
 text='Numerical Integration of sin(x) is
approximately %s' % (dy), showarrow=False)
layout = go.Layout(annotations=[annotation] )
trace_data = [trace1]
fig = Figure(data=trace_data, layout=layout)
iplot(fig, filename='1d-numerical-integration')
```

There are a few new items in this example. We are using an Annotation object that contains the results of the numerical integration. There is also a Layout object, which is used to put together multiple graphics objects to obtain a more complicated plot image. These other graphics objects could be annotations, images, labels and even rendered LaTeX code, which would enable you to include pretty-printed equations.

The last item we'll introduce is the ability to do animations with Plotly. In order to handle animations, you will need to use a new object, called a Frame, to store a list of images to use in the animation. You hand in this Frame to either plot or iplot to generate the animation display itself.

By default, you get a simple Play button that lets you start the animation, but this is a great place to use custom controls. If you use a Layout object, you can add control items, such as a slider, to give you control over the animation playback.

As you can see, just from this very short introduction, there is a lot you can add to any Pi projects that manage data. If you are building a house-monitoring system, for example, you could add Plotly graphs to show temperature trends or power usage. Or maybe you might want to track your network usage and have a simple graphic of the average bandwidth used.

With a bit of research, you will be able to find many different plots and data visualisation methods that will be useful for your projects.

Going online with Plotly

So far we've focused on using Ploty offline, but it was actually designed to handle data plotting on its servers. To do this, connect to the internet, then you can use the plot and iplot functions from the main Plotly module. In this way, you can have your monitoring project set up as a headless machine and be able to check on the status from anywhere in the world. You will need to have an account at the main Plotly site (**https://plot.ly**). A free account enables you create plots, but they will be public. For private plots, you'll need to look into the paid options. You can then have your Python program authenticate with the Plotly site with the following code:

```
import plotly
plotly.tools.set_credentials_file(username='my_name', api_
key='my_key')
```

Your API key can be found from the account settings page on the Plotly site. Now, when you call the plot function, you get a URL back pointed at the location of the rendered plot. By default, it will try to open this URL within the default browser on your Pi. If you are creating a headless monitoring system, you can use the parameter auto_open=False to turn off that behaviour. If new data comes in, but you want to use the same plot URL, you have three options to do the update by using the fileopt parameter. If it is set to overwrite, a completely new plot will be generated and be available at the same URL. If you use the option extend, then the additional data is added to the already existing data. The final option is append, which adds a new data set as a separate plotted set on the same graph.

Using Ploty online means you can also stream data sources. If your project has your Pi serving up a stream of measurements out to the internet, you could use the streaming functionality within Plotly to be able to render visualisations of this data.

Encrypt your Pi home folder

Easily keep your data safe with military-grade security on your personal Raspberry Pi

RECIPE

- This tutorial is compatible with all models of Pi running Raspbian Jessie

With the advent of the dazzling Pixel desktop on the Pi, with its glitzy icons and crisp windows, some readers may be considering using a Pi as their home computer. This however is not especially secure given that the Pi automatically logs in the default user without requiring a password. Even if the password is required, anyone can mount the SD card to view your documents, pictures, videos and other data.

1 Back up your data

If you have not done so already, log onto your Pi and transfer the contents of your Desktop, Documents, Pictures etc onto an external medium like a USB stick. This will make transferring your data to your encrypted home folder much easier. Use Ctrl+H to show hidden folders if you want to back up your application settings such as your browser bookmarks.

2 Enable login screen

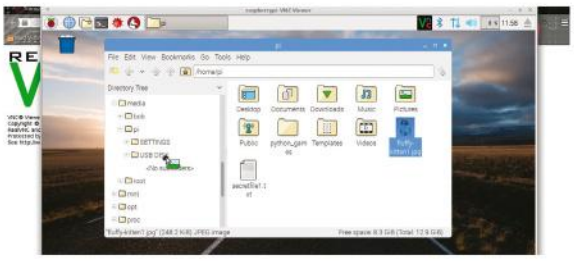
If you have not already done so, open Terminal on your Pi or connect via SSH and run the command:

```
sudo nano /etc/lightdm/lightdm.conf
```

to open your login options. Scroll down to the line autologin-user=pi and put a hash (#) at the start to comment it out. Press Ctrl+X, then Y, and then Return to save and exit. Remember the default password is 'raspberry'.

3 Reboot Pi and log in

Reboot the Raspberry Pi and log in via the login screen. The username pi should already be selected and the password should be 'raspberry'.



Backup all of your files to a USB stick for easy transfer before you begin

Moving forward we recommend that you work on the Pi's desktop directly as you will need to switch between users, which is difficult over SSH. If you do not have a monitor for your Pi, try connecting via VNC as the 'pi' user.

4 Install eCryptfs and related files

Open Terminal on the Pi and run the command:

```
sudo apt-get install ecryptfs-utils lsof cryptsetup
```

You will need to press Y to confirm that you do indeed want to install the software. This will allow you to encrypt the home directory and access it each time you log in.

5 Create new user

If you are serious about wanting to use the Pi, you most probably will want an account in your own name in any case. Use the command:

```
sudo adduser (username)
```

e.g sudo adduser bob to create your account. Type your password twice, then press Return to accept default values for the other options such as location.

6 Encrypt new user's home directory

The eCryptfs software comes with a handy built-in utility to encrypt existing home folders. Simply run the command:

```
sudo ecryptfs-migrate-home -u bob
```

where 'bob' is your new username. You'll be asked to enter a login passphrase twice Make sure to read the Important Notes section once this is done. Do not reboot at this stage.

7 Log into new encrypted user account

Use Menu>Shutdown>Logout to log out of the Pi user. In the dropdown menu select your new username e.g 'bob' and log in. Before bringing any data into this new account, open Terminal and simply type the command mount. This will show a flurry of information; you should see a

reference to 'ecryptfs', which shows the encryption has been successful.

8 Back up your passphrase

If you're unable to log in because of a system problem or you want to move your data to a new computer, by default your files won't be accessible. Fortunately there's an eCryptfs utility that can generate your mount passphrase, which can be used to access your files from another device. Open Terminal and run the command to view the passphrase:

```
ecryptfs-unwrap-passphrase
```

Once you've unlocked it, write it down. Store the passphrase in a safe place.

9 Migrate your data

Use Menu>Shutdown>Reboot to restart your Pi and then log back in as the new user. At this stage you should connect your external drive and begin copying your documents and data back to the right places. You'll see that a new home folder has been created in /home, e.g /home/bob. No other users on the Pi will be able to access the files inside your home folder.

10 Give your new user admin privileges

Log out of your new user for now and back into 'pi'. Open Terminal and run the command:

```
sudo visudo
```

Scroll down to the line reading root ALL=(ALL:ALL) ALL and on a new line immediately after this add:

```
bob    ALL=(ALL:ALL) ALL
```

Where 'bob' is your new username. Press Ctrl+X, Y, then Return to save and exit. Restart the Pi again to effect your changes.

11 Remove backup home folder

When encrypting your new home directory, the eCryptfs software places a backup in the home folder in case anything goes wrong. As this was a new account, the folder is empty, but to keep things simple, open Terminal and run the command to find out its exact name, e.g bob.zyxxc:

```
ls /home
```

Then run the following command to remove it:

```
sudo rm -r -f <directoryname>
```

12 Delete data in Pi home folder

If you had personal data already in the existing 'pi' folder, it is still unencrypted and can be accessed by anyone who can obtain your Pi's MicroSD card. Log onto the 'pi' user and use the shred command to securely delete any files you want e.g shred -zu bank-statement.pdf. You can use the following to erase folders but do not do this with any of the main folders in your 'pi' home folder, e.g Desktop:

```
sudo rm -r -f <directoryname>
```

> "ONCE YOU'VE DONE THIS, YOUR RASPBERRY PI SHOULD BE AS SECURE AS FORT KNOX!"

13 Disable swap space

There is a portion of the Pi's SD card (located in /var/swap) that is used in a way similar to RAM when the Pi is low on resources. For more modern Pi models this is very rarely used and can present a security risk as your data may be written there unencrypted. Log in to your new user account, open Terminal and run:

```
sudo swapoff -a -v
```

You should see a confirmation that it has been disabled.

14 Fix permissions errors

Depending on the method you used to back up and transfer your data from your previous account, there may be permissions issues with certain files and folders (Usually you will see a padlock on files you're unable to edit). You can make sure your new account is the owner of all files within folders with the command:

```
sudo chmod 0750 -R foldername
```

For example, where 'bob' is your new username:

```
sudo chmod 0750 -R /home/bob/Pictures
```

15 Double check your files are safe

This step is optional. Reboot the Pi and log into the 'pi' user. Open Terminal and if you have not previously done so, enter the command:

```
sudo passwd root
```

to set a password for the root user. Next enter the command su to switch into root. Run:

```
cd /home/bob
```

(where 'bob' is your new username) and then use ls to show the contents of the directory. There's now only a shortcut and a 'ReadME' file saying your data is protected. Once you've done this, your Raspberry Pi should be as secure as Fort Knox!

Pi bites

Even if your home folder is encrypted it can be brute-forced by another computer trying various passwords until it hits on yours. Visit the website **https://howsecureismypassword.net/** to see how long it would take to crack your own password. If the result is anything less than a century or so, consider using a stronger password. The DiceWare website (**http://world.std.com/~reinhold/diceware.html**) provides a quick and easy way to generate strong and easy to remember passwords through randomly selecting several dictionary words.

Building a Raspberry Pi Python cluster

Learn to use Python to get some serious work done on your Raspberry Pi cluster

In a previous article, we looked at how to combine multiple Raspberry Pis together to make a cluster of machines. But how can you make use of this small monster that you have created? Luckily, IPython is a great way to put all of these machines to work, getting some serious work done. IPython started out as an enhanced interface for Python, replacing the usual interactive interface that is available. The initial design involved creating a client/server system for IPython, which quickly allowed for some very interesting functionality. One of these extra capabilities is the ability to run multiple IPython servers that a single IPython client can connect to. You can then connect to these IPython servers and run processes in parallel. This first step is to install IPython on your Raspberry Pis. The following command will take care of that for you, and will need to be run on each node of your Pi cluster.

```
sudo apt-get install ipython ipython-notebook
```

This will install the version of IPython which uses Python 2.X. If you are planning on using Python 3.X, you will need to install the packages ipython3 and ipython3-notebook instead. Starting with version 4.0 of IPython, the parallel functionality has been pulled out into its own package called ipyparallel. Most distributions do not have it available within their package management systems. This means that you will need to install it with pip, as shown:

```
sudo pip install ipyparallel
```

Depending on the versions of the dependencies on your system, pip may update some of these packages. Ipyparallel is structured in four sections: the engine, the hub, the scheduler and the client. The engines do the actual running of the code from your program. The client is where your code runs and where the parallel calls are made.

The hub and scheduler are the parts that let the client and server communicate with each other. Once everything is installed, you can start up the various parts of the parallel parts of IPython. Starting a set of engines on your Raspberry Pi can be done with the command ipcluster, just run the following code:

```
ipcluster start -n 4
```

This will start up a controller and a set of four engines. An IPython controller is made up of one hub and a series of schedulers. When you run this command, it stays attached to the starting shell process so that it can write out messages from the controller. If you want to just start up the cluster and leave again, don't forget to put an ampersand (&) at the end of the command so that it ends up being run in the background. As a first test, you can create a client object and connect to these engines with code like this:

```
>>> import ipyparallel as ipp
>>> c = ipp.Client()
>>> c.ids
[0, 1, 2, 3]
>>> c[:].apply_sync(lambda : "Hello  World")
['Hello World', 'Hello World', 'Hello World', 'Hello World']
```

When you instantiate a new Client object with ipp.Client(), with no input parameters, it will look in the directory ~/.ipython/profile_default/security to see where to connect to in order to run your parallel code. The next statement gives you a list of the IDs for the available engines. The final statement uses the method apply_sync to run the given lambda function on some subset of the available engines. In the previous example, we actually ran the code on the entire list of available engines. This all works fine on a single Raspberry Pi, but that is not what we are interested in. How do you use that entire cluster that you have set up and waiting to work for you? There are a few options available on how to organise all of the different processes so that they can all communicate with each other. The simplest method is if you keep the controller and the engines together on the same machine and use ipcluster to run everything. By default, ipcluster sets things up so that only local connections will be accepted. In order to accept connections from external machines, you will want to use IPython to create a new profile, as shown:

```
ipython profile create --parallel --profile=myprofile
```

This will create a new profile directory in ~/.ipython/profile_myprofile with a set of

configuration files. In order to accept incoming connections, you need to tell the controller what interface to listen on. You will need to edit the file ipcontroller_config.py in the profile directory and add the following line for the HubFactory.

```
c.HubFactory.ip='*'
```

Now when you start your cluster, you can hand in a profile name to use the altered configuration options, as shown.

```
ipcluster start --profile=myprofile
```

Within your code, you need to tell it where the controller is running. If the two machines are sharing a filesystem (maybe over NFS, for example), you can use the profile parameter when you instantiate a new client object.

```
>>> c = ipp.Client(profile='myprofile')
```

If they don't share a filesystem, you need to configure the client before making a connection. On the Raspberry Pi that is hosting the controller, you will find a file named ipcontroller-client.json in the directory ~/.ipython/profile_myprofile/security, which will contain all of the connection details that the client will need. If you copy it to the client machine that will run your code and place it in the directory ~/.ipython/profile_default/security, you can again instantiate new client objects without any parameters and it will read the connection details from this JSON file. We are still only using a single Raspberry Pi to run engines on.

What about the rest of your cluster? In order to pull these in, you will need to use the individual ipcontroller and ipengine commands, rather than the all-in-one ipcluster command. Since the controller needs to accept connections from clients and engines that are running on remote machines, we will continue to use the IP configuration option that we set earlier. You can then just run the command ipcontroller to start it up.

The next step is to run the ipengine command on each of the Raspberry Pis that are going to form the pool of computational engines. In the directory ~/.ipython/profile_myprofile/security you will find a second file named ipcontroller-engine.json. This file needs to copied to each of the Raspberry Pis within your cluster so that they will have all of the connection details necessary to be able to communicate with the controller. On each of the Pis, you can start the ipengine process, pointing it to the ipcontroller-engine.json file that you just copied over.

```
ipengine --file='ipcontroller-engine.json'
```

If you want each Pi to host multiple engines, then you will need to start a separate ipengine process for each one. So, now you have IPython engines running on a cluster of Raspberry Pis, with one of them hosting a controller. You can get your Python code to connect to this cluster to parallelise your work, but how can we use all of this power? One of the first steps is to grab a view of some, or all, of the available IPython engines to work with. This is done using the same syntax as that used when slicing lists. Once you have a view, you can use one of the apply methods to get a function to run on the selected engines:

```
>>> v = c[0:2]
>>> v.apply_sync(lambda : 42)
[42, 42, 42]
```

This is a blocking version of the apply method. If the function handed in is going to take time, you can use the method apply_async(), which will run your function on the engine pool asynchronously and return an AsyncResult object that you can use to retrieve the results once they're finished. If you have data that needs to be processed, you can use the map method to distribute the data to the IPython engines and run the given function on each chunk.

```
>>> v.map_sync(lambda x : x*2, range(10))
```

"IPYTHON IS A GREAT WAY TO PUT ALL OF THESE MACHINES TO WORK, GETTING SOME SERIOUS WORK DONE"

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

As with the apply methods, there is an asynchronous version of the map method, too. You can also move data around with methods from the view object. If all of the engines need a complete set of the data being used, you can use the push method to move data out to the engines, and pull to read the data off the engines again. If you want to use the dictionary interface to verify the data transfer, running the following code:

```
>>> dict1 = dict(a='foo', b='bar')
>>> v2 = c[2:]
>>> v2.push(dict1)
>>> v2['a']
['foo', 'foo']
```

You can use scatter and gather to partition your data giving each engine a chunk. Hopefully, this article has inspired you to dig out all of those Raspberry Pis that have been sitting around collecting dust and create your own cluster for big-time calculations. The nice thing about using IPython is that you can easily add or remove machines to the pool that is available for running Python engines.

Make a Bitcoin cold store

Put your Bitcoins on ice and use a Raspberry Pi as a cold wallet to store all your digital currency safely offline

RECIPE

- **Suitable for all models of Pi (we recommend a Pi Zero). For security reasons, you need a dedicated Pi for this project.**

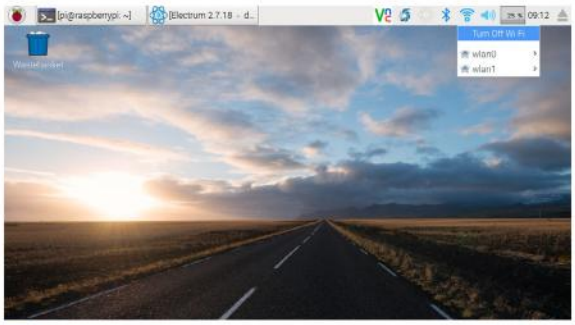
As you're probably aware, the cryptocurrency Bitcoin can be used to make anonymous and irreversible online payments, similar to hard cash. Bitcoin is a huge leap forward for privacy-minded people, but the fact that it's a digital currency means that hackers often target owners of Bitcoin software wallets. In this tutorial, we'll explore a solution using the lightweight Bitcoin Wallet Electrum on the Raspberry Pi. The Pi will host a 'cold' wallet which is kept offline at all times. You'll also set up an online 'watching' wallet on another computer so you can see payments as they come in. Payments are only sent by generating a transaction file with your watching wallet, which you transfer manually via a USB stick to your 'cold' wallet to be digitally signed. As your cold wallet is offline, only people with physical access to it can make payments, which enables you to store Bitcoins safely. This guide assumes that you are familiar with the basics of Bitcoin. If not, then visit this useful introduction page **http://bit.ly/2m7Ct2X for a rundown.**

1 Install software

Connect the Pi to a screen and keyboard. Open Terminal and run these two commands:

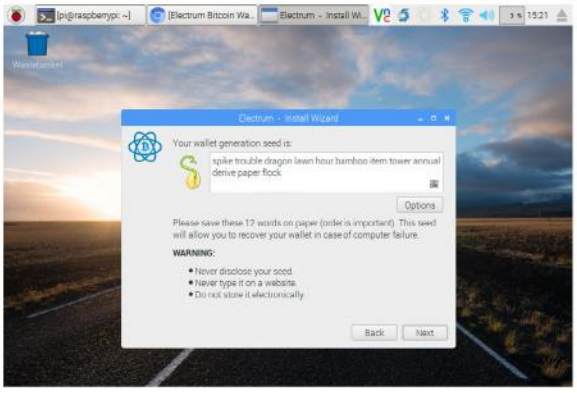
```
sudo apt-get update
sudo apt-get upgrade
```

For security reasons, you should only use this Pi as a wallet for future purchases. The Raspbian repositories (repos) have an old version of the Electrum wallet, but it is best to download the latest version manually. Next, run the following two commands to install the latest wallet software with sudo apt-get install python-qt4 then sudo pip install **http://download.electrum.org/2.7.18/Electrum-2.7.18.tar.gz.**



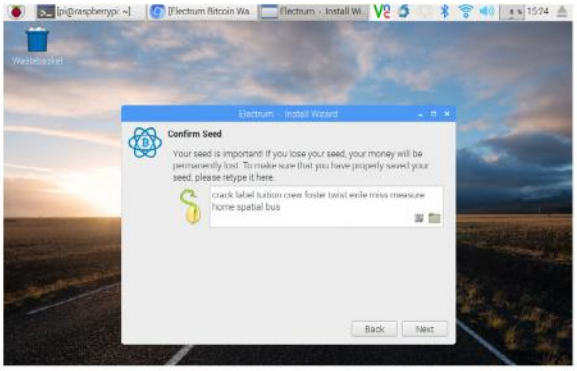
2 Disconnect from the internet

This step is essential for ensuring your wallet is safe. Remove any network cables and/or Wi-Fi adaptors connected to your Pi. If you use a Pi 3, simply click on both the wireless and Bluetooth icons on the Pi and choose 'Turn off Wifi' and 'Turn off Bluetooth' respectively. Keep your Pi connected to a monitor and keyboard, as you'll need this for sending payments later. Once you're offline, run the command electrum in Terminal to start the wallet.



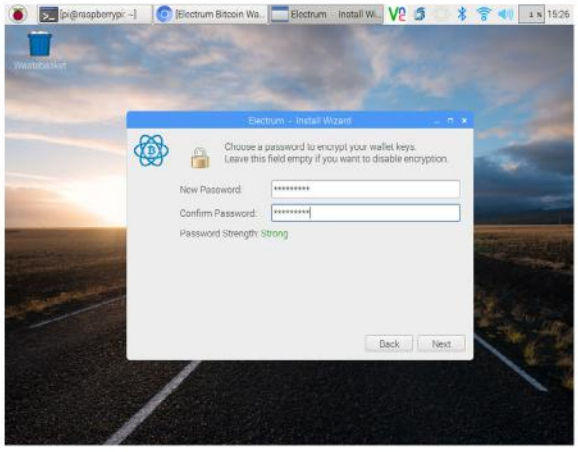
3 Generate wallet seed

Choose 'Auto Connect' on the first Electrum window, then click 'Next'. On the next screen choose a 'standard' wallet and 'Create a New Seed'. Electrum generates 12 random words for wallet seed and do as the program requests and write these down on paper. These random words will enable you to restore your wallet if anything happens to your Pi. Click 'Next' to continue.



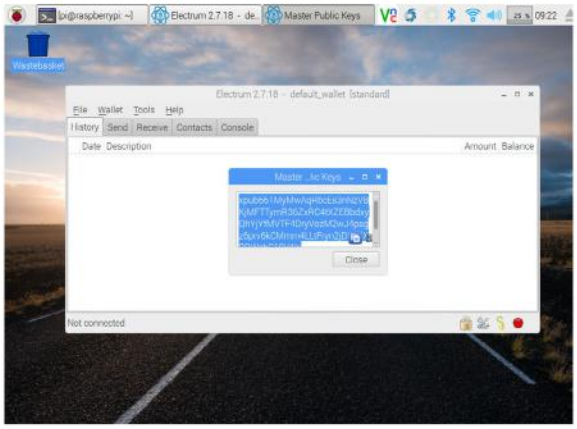
4 Confirm and store wallet seed

Use the words you wrote down earlier to retype your wallet seed. Remember that in the wrong hands, the seed can be used to steal from you, so store the paper in a safe place or better yet memorise the words. Try using the Loci method to memorise the words by associating them with places in a familiar location such as your home.



5 Choose your wallet key

After you have confirmed the wallet seed you can optionally set a password to protect this copy of the wallet on the Pi. You'll need to re-enter this password when sending Bitcoins. Ideally choose a password that's at least 10 characters long with a mixture of letters, numbers and symbols. Electrum alerts you if a password is weak, moderate or strong. Do not use any of the words in your wallet seed for the password.



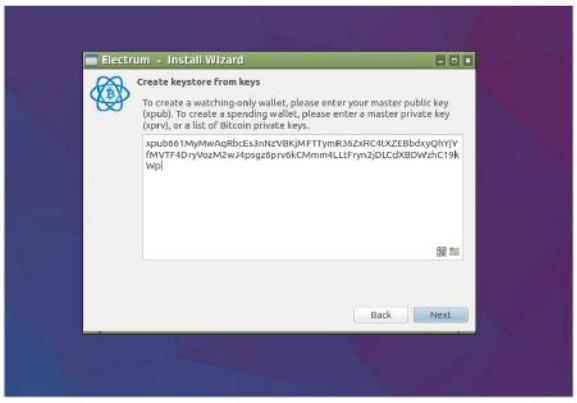
6 Export master public keys

In the Electrum Wallet, click the 'Wallet' menu option then 'Master Public Keys'. This will generate a string of text that you'll use on a separate computer for your 'watching wallet' so you can receive payments. Click the blue icon to copy the key then go to Applications > Accessories > Text Editor to paste it into a text file. Insert a USB stick and save the new text file to it.

7 Install watching wallet

Safely eject the USB stick from the Pi and insert the stick into the computer that you want to use for your 'Watching Wallet'. Electrum works on most platforms but for security reasons, and because we'd always recommend GNU/Linux over any other platform, place your watching wallet on a spare Linux system, if possible.

If you're using a Debian-based distro like Ubuntu, repeat Step 1 but make sure that you install python-pip as well as python-qt4. For other platforms follow the instructions at **https://electrum.org/#download**, but we'll ignore those and move on.



8 Restore master keys

Launch the Electrum client on your online machine. In Linux, you can simply run electrum from the Terminal. Choose 'Auto Connect' once again on the first screen and click 'Next'. Click on the 'standard wallet' option once again and then on the 'Keystore' screen choose 'From Public or Private Keys'. On the next screen, paste in the master key in the text file from your USB stick. Read the warning and click 'OK'. »

Pi bites

By default, anyone snooping on your connection can see your IP address connecting to the Electrum servers. Your IP could be used to link your identity to specific addresses in your 'watching' wallet. If you have Tor running on your system, you can route Electrum's connection through the Darknet by going to Tools > Network. Select 'SOCKS5' as your 'Proxy'. The rest of the information is filled in already, so just click 'OK'.

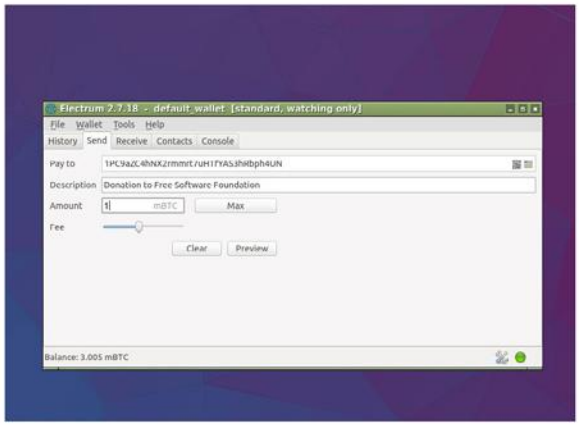
9 Set up addresses

Electrum will now generate addresses for you and when the main window opens click on the 'Receive' tab to view your address. You'll see both a text string and a handy QR code for your new address. Type a description for your receiving address if you wish, then click 'Save'. You can generate more addresses later.

10 Test wallet address

Using your current Bitcoin wallet, try to send some funds to your new wallet address. If possible, use a client which supports scanning QR codes, so there's no danger of getting the address wrong.

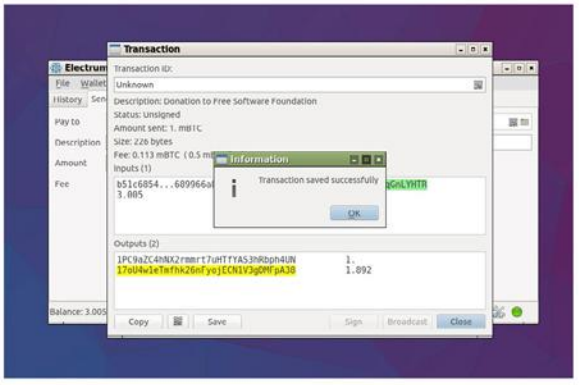
Once the payment is sent, the unconfirmed new balance should show in the bottom-left corner of the Electrum Window. Now, you'll need to wait until the balance is confirmed before proceeding.



11 Set up sending

Now that you can receive funds, it is time to test if you can send them. Find your recipient's wallet address. For this tutorial, we are using the Bitcoin Address of the Electronic Frontier Foundation (EFF) to send a donation.

Click on the 'Send' tab in Electrum and paste the address into the 'Pay to' field. Add a description if you want to and then click 'Preview'.

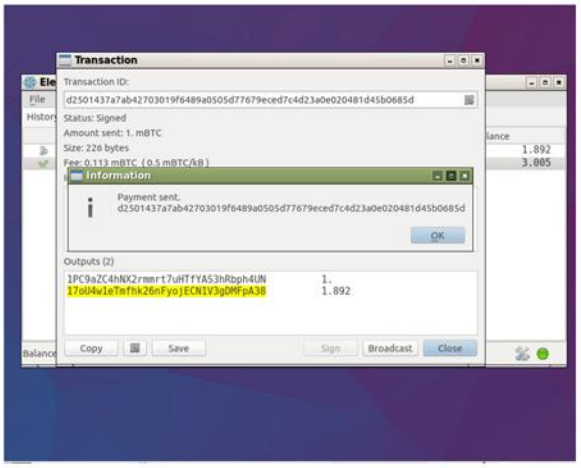


12 Save transaction

Click 'Save' in the new window to save the TXN (transaction) file and navigate to your USB stick and place it there. Electrum confirms that the transaction is saved successfully. You can click 'Close' to shut down the window. A transaction needs to be digitally signed by the cold wallet on your Pi to send the funds. Safely eject the USB stick from your online computer and connect it to the Pi.

13 Sign transaction

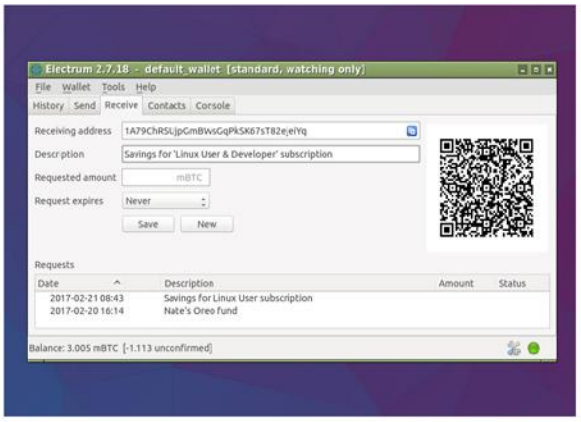
Open the Electrum client on your Pi and click on the 'Tools Menu'. Select Load Transaction > From File and navigate to the transaction file on the USB stick.Details of the transaction will load. Check everything is in order then click 'Sign'. Enter your wallet password. Click 'Save' to store the newly signed transaction file to the USB stick. Click 'OK', followed by 'Close' and safely eject the USB stick from the Pi.



14 Broadcast transaction

Next, you'll need to insert the USB stick to your computer with the 'watching' wallet. Open Electrum and click Tools > Load Transaction > From File. Choose the signed file on the USB stick. A new window opens with the transaction details. Double-check these and click 'Broadcast' to send them to the Bitcoin network.

You will see a confirmation message. Click 'Close' to see the unconfirmed amount in the Electrum window. The processing times for a transaction will vary based on the fee that you paid.



15 Generate new addresss

To prevent your Bitcoin transactions from being traced, you should generate a new address for receiving each payment. In the 'Receive' tab of Electrum, click 'New' to create a new one. Enter a description if you like and click 'Save'.

You can move between different wallet addresses from the 'Requests' pane.

Once this is complete, repeat steps 10-14 to receive and send funds from this new address. Having worked you way through this tutorial, you'll now have a 'cold' wallet set up, a method for making payments and an online 'watching' wallet for tracking payments. And that's it – you're officially all set up with a Bitcoin bank account!

GET TO GRIPS WITH NETWORKING WITH THE HELP OF OUR ULTIMATE HANDBOOK

Discover everything that you need to know about all of your devices and create the best home set-up!

ON SALE NOW

Future

Ordering is easy. Go online at:

www.myfavouritemagazines.co.uk

Or get it from selected supermarkets & newsagents

Parallel programming with Dask

When you need more power, you usually need to move to a parallel machine. Dask tries to simplify that for your Python programs

Once you do reach the limits of your machine, your only option is to spread the computational work around. But, traditional parallel programming can be messy. Because of this, there have been several frameworks developed to try and wrap the complexity. This month, we will look at Dask and how to use it on your own Raspberry Pi cluster.

Dask is a pure Python library that provides two major components. The first is a scheduling system that handles the management of all of the computing nodes of your cluster and takes care of what portions of your work run on which individual machines. The second is a set of new collection classes that will use the parallel cluster implicitly behind the scenes. Using these new classes is likely the easiest way to introduce parallel programming to your own code.

The first step is install the relevant Python module. You can install the scheduler portion of the Dask framework only, which is useful if you don't need the additional new collection classes, or install each of the classes individually. But for the purposes of this article, we will be looking at everything that Dask can do, so you may want to install everything:

```
sudo pip install dask[complete]
```

If you don't want to 'pollute' the Python library directories, you can use the --user option to install it within the Python library owned by your account.

On traditional desktops, Dask's new collections are used to handle large data sets. This might still be the case if you are using a Pi to do low-power processing, so we will start here. The three new collections are called arrays, bags and dataframes.

Dask arrays are a subset of the ndarray provided by the numpy module. The basic structure is a matrix of smaller ndarrays. These individual ndarrays can be backed by either storage in RAM or on disk, both on a single machine or remote machines. This way, you can efficiently manage huge data sets. To create a new array object, you can use the following code:

```
import dask.array as da
  import h5py
  file = h5py.File('myfile.hdf5')
  data = file['/file/path']
  array1 = da.from_array(data, chunks=(1000,1000))
```

This loads the data from an HDF file, creates ndarrays from it and creates a new array object from that data.

If your data either doesn't map to the standard use case for an array, or is a collection of user defined objects, you may find the bag collection useful. This class enables you to group several other lists or iterable objects into a single object and be able to iterate over the entire collection. You can then break this object over several cores and have each of them iterating over their own section of the bag. This naturally fits with operations like maps or filters. The problem is that it isn't as useful if you are using your Raspberry Pi cluster. Still, they may be useful in simplifying your code in certain cases, or if you only need the four processing cores available on the Raspberry Pi 3. As with arrays, you can create bags out of other Python objects:

```
  import dask.bag as db
bag1 = db.from_sequence([1,2,3,4,5])
```

The code (on the previous column) creates a new bag out of an existing iterable object.

The big caveats here are that you are limited to what you can handle within a single Pi. Also, bags are immutable, so they are more useful as a way of organising other objects in your code.

The last new class provided by Dask is the dataframe. Dask's dataframes are subsets of the pandas dataframe. Just as arrays were built out of a series of smaller ndarrays, the Dask dataframe is built up out of smaller pandas dataframes. While there are methods to create a new Dask dataframe from other building blocks, such as Dask arrays or bags, you will likely be loading new dataframes directly from files. Dataframes are meant to be used with very large datasets. The following code is an example of how to load your data from a CSV file:

```
import dask.dataframe as dd
dataframe1 = d.read_csv('myfile.csv')
```

With all three of the new classes, there are methods to write your results in several different file formats. If you have done any amount of programming, you will know that not every

Pi bites

Why Python? It's the official language of the Raspberry Pi. Read the docs at **python.org/doc.**

problem fits within any particular framework. Dask is no exception. To this end, Dask has included a section named delayed. This adds a new keyword that can be used to identify chunks of code that are to be parallelised. For example, let's say that you had a collection of data that you wanted to process through a for loop. Within this for loop, you will be executing the functions 'inc', 'double' and 'add', then summing the results from the loop. The following code shows what this would look like when you use the delayed functionality:

```
from dask import delayed
output = []
for x in data:
    a = delayed(inc)(x)
    b = delayed(double)(x)
    c = delayed(add)(a, b)
    output.append(c)
total = delayed(sum)(output)
total.compute()
```

When Python goes through the for loop, nothing gets executed. The functions are added to a task graph of the whole calculation so that it can be analysed for possible parallelisation. When you finally call the compute() method of the 'total' object, those tasks get executed. This compute method also exists as part of the methods provided by the collections. This is because any functions on the collections are executed lazily. For example, the following code would find the sum of a given array collection:

```
sum_total = array1.sum()
sum_total.compute()
```

This is because the work gets broken into chunks, and the scheduler tries to figure out the minimal amount that will get result you are looking for.

In all of these cases, Dask uses a scheduler to manage the computations in parallel. The four available schedulers are threaded, multiprocessing, async and distributed. Each of the collections in Dask use one of these schedulers by default. Array and dataframe use the threaded scheduler and the bag collection uses the multiprocessor scheduler. Usually, these defaults are appropriate, but you can explicitly use a different scheduler. There are two different ways to approach this: The first is to explicitly set the scheduler to use with the 'get' keyword when you call the compute() method of the given collection. For example, the code, below, uses the multiprocessing scheduler:

```
sum_total = array1.sum()
sum_total.compute(get=dask.multiprocessing.get)
```

The other way that you change the scheduler is to use the 'set_options' keyword. The code, which you can see top-right, shows how you could either set the scheduler within a given context, or how to set it globally:

```
#within context
with dask.set_options(get=dask.multiprocessing.
get):
    sum_total.compute()
#used globally
dask.set_options(get=dask.multiprocessing.get)
sum_total.compute()
```

This allows you to maximise the multiple cores within a single Pi, but that is still a small pool.

The distributed scheduler can really put your Pi cluster to work and is a separate package, but it will get installed if you used the complete option when installing Dask.

There are several binary programs that get installed as well, including dask-scheduler and dask-worker. The overall structure is that a single scheduler runs and manages multiple workers. The scheduler distributes the work among the available workers. When a worker starts up, you need to give it the location of the scheduler you want it to communicate with. On the Pi that will act as the central scheduler, you can run the binary executable dask-scheduler on the command line. You will get output telling you what IP address and port the scheduler will be listening on. To set up each worker node for the distributed scheduler use, for example:

```
dask-worker 192.168.0.1:8786
```

Using Anaconda projects

Install the port of Anaconda for the Raspberry Pi in order to have a bit more control over the Python environment. This is so we can take on more complex projects and still have a relatively clean environment. This means we need to create new projects and manage them. To create a new project, use the following command:

```
conda create --name project1
```

This will create a new environment, named 'project1', with the core you need in order to run Python programs. If you already know that you will need certain modules, you can include them during the creation. For example, if you are doing a project on bioinformatics, you may want to have biopython available, as below:

```
conda create --name project1
biopython
```

You can now switch to the new environment by sourcing the relevant scripts. For example, the following command activates the new project:

```
source activate project1
```

Now, whenever you install new Python modules, they only get installed within this environment. When you are done, you can run the deactivate script to reset the environment to the original state:

```
source deactivate
```

Once you get a number of projects on the go, you will need to manage them effectively. You can get a list of all of the projects with the following command:

```
conda info --envs
```

The currently activated environment is the one that has an asterisk beside the name. The last thing you may want to do with this project is to share it out with others. When you are ready, you can use this command to generate an environment YML file:

```
conda env export > project1.yml
```

Once you've done this,The other person can then create a copy of your project with the following command:

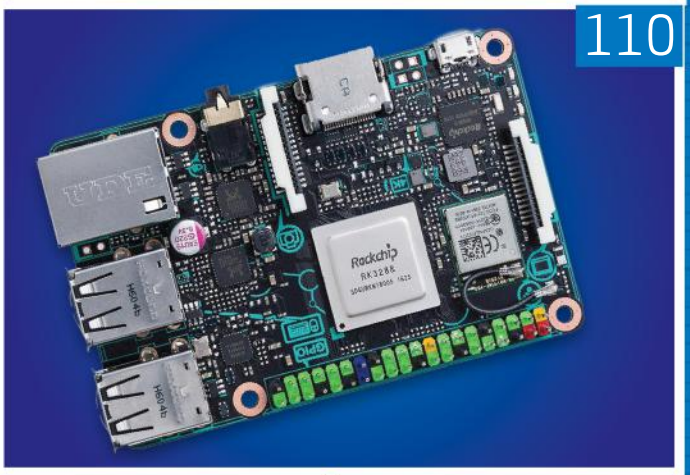
```
conda env create -f project1.yml
```

Pi REVIEWS

Tested and rated: the latest add-ons for your Pi

HANDS-ON REVIEWS WITH THE BEST ACCESSORIES FOR AND ALTERNATIVES TO YOUR PI

108



110

112

Pimoroni Scroll Bot - Pi Zero W Project Kit

Still recovering from another slice of Pi, we settled down on a wet spring afternoon to build a cheery robot face. Here's how we got on...

In brief...

This is a kit comprising the new Raspberry Pi Zero W, some laser-cut acrylic pieces and the Scroll pHAT HD board. The kit itself is a gentle introduction to the skills needed to build physical projects based around the Raspberry Pi Zero W. All you need to provide are an SD card, power and keyboard/mouse.

The Raspberry Pi Zero W has now been with us for just a few months, but in this short time it's caused quite a stir. £10 for a computer with Wi-Fi and Bluetooth is hard to resist, but what if you're new to the scene? Well, fear not because retailers such as Pimoroni have brought out a range of products to introduce you to the world of the Raspberry Pi.

Pimroni's Scroll Bot kit is a self-assembly pack that comes with everything you need to build a Raspberry Pi Zero W-powered light-up gadget. It includes a Pi Zero W, the latest Scroll pHAT HD add-on board (more on that later), and adapters to enable your Pi Zero W to be used with an HDMI screen and USB devices.

In lieu of a USB power supply the kit comes with a USB A to micro USB lead. Given the meagre power requirements of the Pi Zero W you can easily find a power supply in your home. The kit to build the robot frame are laser cut using the same equipment as used to build the famous Pibow cases, and all of the fixings are included in the kit.

Simple assembly

Putting the kit of laser-cut parts together is very easy. The Pi Zero W and the Scroll pHAT HD require header pins to be soldered, or you can use hammer headers if you don't fancy turning up the heat. With the kit assembled, it's then simply a matter of installing Raspbian – but you'll need to source your own SD card because one isn't included, which feels a little short-sighted.

> "THE SCROLL PHAT HD CAN BE USED TO SCROLL TEXT [...] OR SIMPLE IMAGES"

Once Raspbian is installed and booted (if you're not sure how to do this, there was a step-by-step for installing operating systems in **Pi User** #3), you'll need to use the software installer to install the Python libraries for the Scroll pHAT HD board, and then run the included examples to see what the board can do.

The Scroll pHAT HD replaces the earlier Scroll pHAT board and offers a grid of 17x7 LEDs, a total of 119 white LEDs. Much improved over the Scroll pHAT 11x5, 55 LEDs.

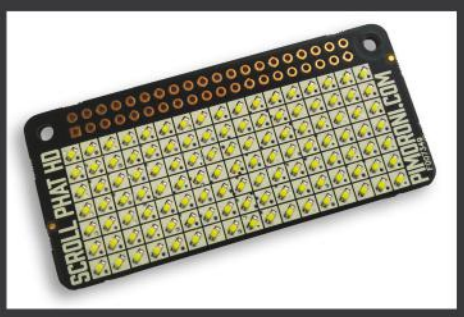
The Scroll pHAT HD can be used to scroll text, animations or simple images across the matrix of LEDs. In our review we made an RSS feed reader that scrolled the top five news items from various websites, but you could easily use the Scroll pHAT HD to show server status information, weather or tweets using a particular hashtag. The included examples range from simple scrolling text, to impressive plasma effects that use complex mathematics to show off the board's features.

Affordable package

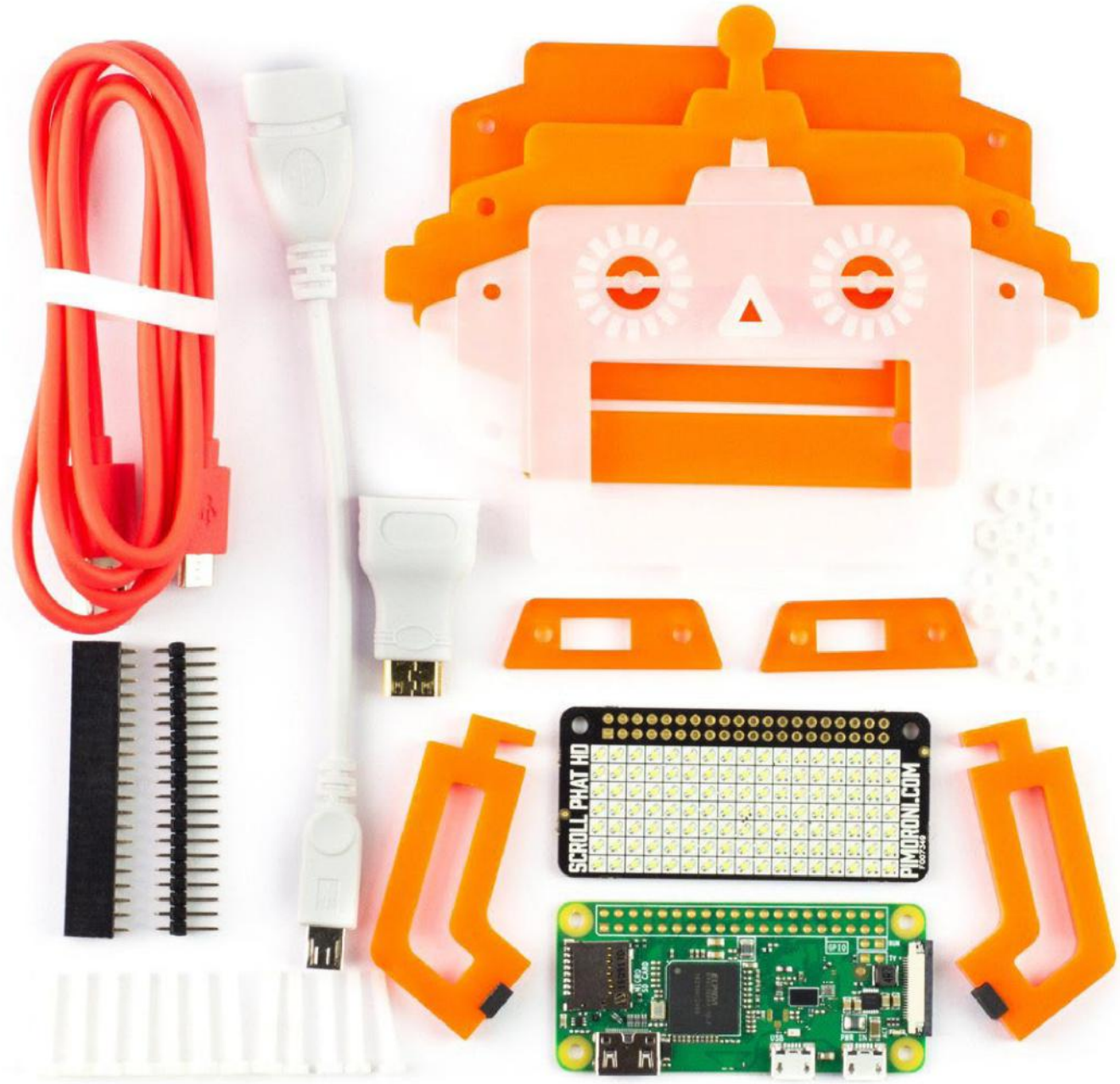
The Scroll Bot kit is a great introduction to the Raspberry Pi. The £35 price tag may look expensive but the Pi Zero W with adapters and Scroll pHAT HD

Features at a glance

Laser-cut parts
Pimoroni is well known for its laser cut parts, and the Scroll Bot Kit's are well made and easy to assemble.



Scroll pHAT HD
This Zero W pHAT is good for beginners: it can scroll text, images and even animations without much of a learning curve.



come in at £26 on their own, so for the extra £9 we get the laser-cut kit, USB lead and a rather nice box that can be reused to store your ever-growing Raspberry Pi or Maker accessories.

The kit is easy to build, although some people may struggle with soldering the Pi Zero W and the Scroll pHAT HD, but you could always ask your local Makerspace for help. Apart from the SD card you need to supply, the Pi Zero W is ready to go and can be used either as an embedded device or as a typical computer.

The Scroll pHAT HD is a fun board. Sure it doesn't have as many uses as other boards on the market, but what it does do, it does well. Scrolling text name badges, news tickers or a lovely smile for a robot are only a few lines of Python away!

This cute kit is easy to build and will look great on the shelf of your office, desk or library

Pi User Verdict

Pimoroni Scroll Bot - Pi Zero W Project Kit

Developer: Pimoroni
Web: pimoroni.com
Price: £35

Rating 8/10

Asus Tinker Board

Is it a board? Is it a Raspberry Pi? No, it's another contender for the microcomputer crown

In brief...

A Raspberry Pi 3-sized board that offers a faster CPU, more memory and Gigabit Ethernet. It's targeted at the maker community, specifically those who require more processing power than is currently on the market. It has plenty of that, roughly twice the computational power of the Pi 3, but lacks the mature software to get the best from the board.

The Asus Tinker Board arrived on the scene with very little fanfare, and seemed to catch everyone by surprise. Unfortunately, this was reflected by the lack of software when the board was released: for the first few days there was no operating system publicly available to run it. But let's put that behind us and take a look.

Powered as it is by an RK3288 System on a Chip, featuring a quad-core ARM Cortex A17 running at 1.8GHz and 2GB of LPDDR3 RAM, we can instantly assume that this board is meant to be a 'Pi Killer' and computationally it is. Running the sysbench prime number test for a single core, it took only two minutes and two seconds to compute all the prime numbers up to 10,000, versus the Pi 3 time of three minutes two seconds. A full minute quicker! We repeated the test utilising all four cores and the Tinker Board completed it in 31.34 seconds and the Pi 3 in 45.7. So we can see there is plenty of power in the Tinker Board's CPU.

Smart specs

The board provides four USB 2.0 ports along with HDMI, micro USB power, and a 40-pin GPIO, which is not fully compatible with boards produced for the Raspberry Pi, but can be used with electronic components (LEDs, buttons, etc) to build your own projects. The software to control the GPIO has to be downloaded separately, quite why it can't be included ready for use we don't know. It is called ASUS.GPIO and you've guessed it, it is a fork of the RPI.GPIO library which has powered thousands of projects. It works in the same manner, but you won't be able to connect any SPI/I2C devices just yet as the software is not ready.

Networking comes in the form of the built-in 802.11(b/g/n) Wi-Fi and Bluetooth 4.0, which uses a PCB antenna for reception, but you can replace the antenna with an externally mounted option to boost your signal. There is also "Gigabit" Ethernet, but when we tested the bandwidth using iperf we only managed to record 35.3Mbits/s. However this is still far higher than the Pi 3 which only manages 11Mbits/s as it comes via a USB 2 interface. The Tinker Board does not share the Ethernet bus with USB, enabling the higher bandwidth, but still short of Gigabit speeds.

> "THE ASUS TINKER BOARD IS UNDISPUTEDLY A POWERFUL PLATFORM FOR MAKERS"

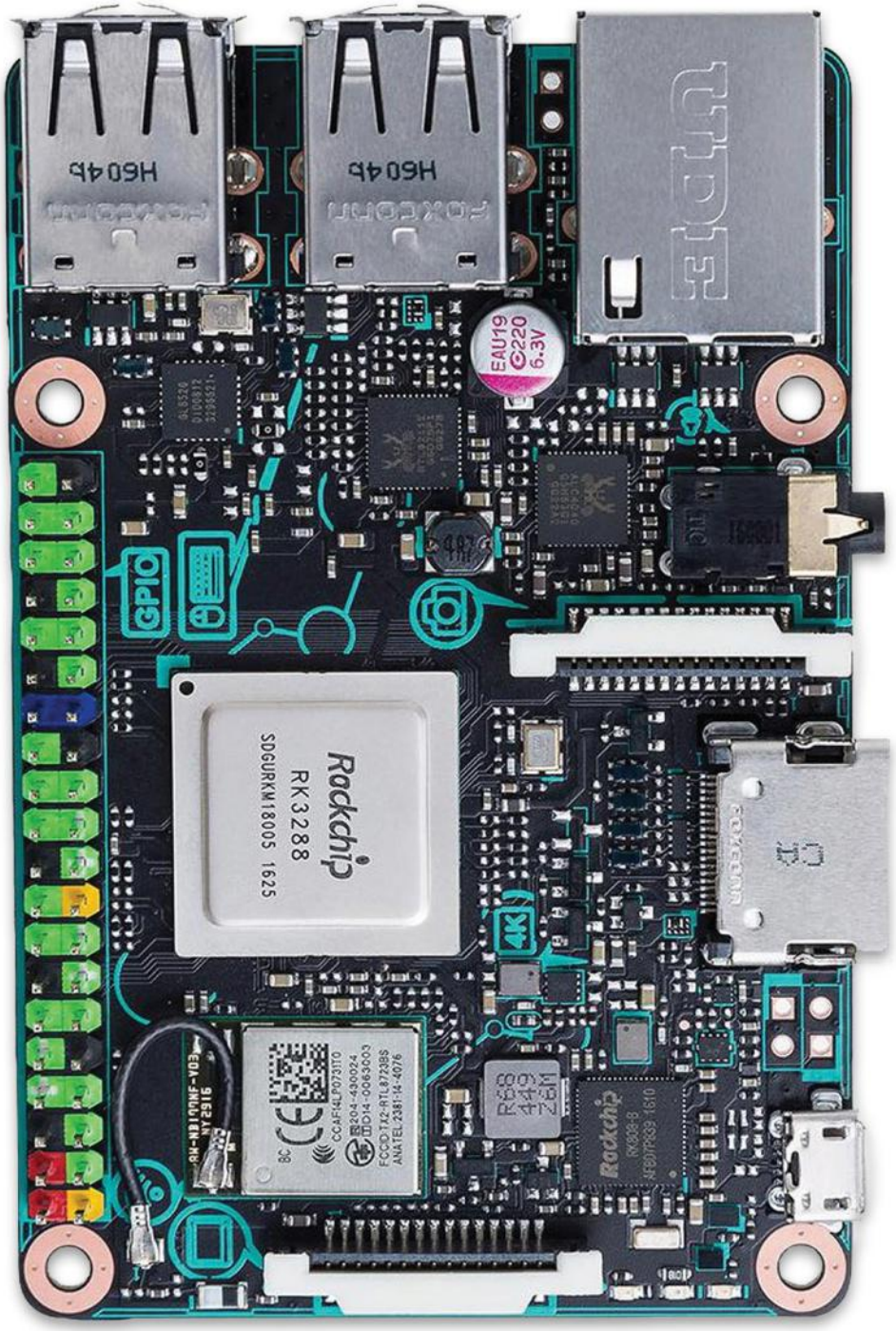
Admirable OS

The Asus Tinker Board runs a version of a Debian-based distribution, called TinkerOS. It is lightweight and works really well as a desktop, giving the user access to a traditional menu and widgets to control wireless connectivity. You will find the Chromium web

Features at a glance

Kodi media player
It provides plenty of power for Kodi streaming and can display 1080p video. All from a small board!

External antenna
The Wi-Fi antenna can be disconnected, enabling the user to add an external one for increased range.



browser present, and it did an admirable job with everything we threw at it – except for YouTube. This board is able to play video at 1080p but YouTube videos ran poorly, even after installing a patch from Asus. Our test of the Star Wars Rogue One trailer at 1080p crawled along in windowed and fullscreen mode.

We installed Kodi on our test unit and we were able watch HD movies and stream HD content to our device. It worked flawlessly and means that the issues observed are only with streaming content via the web browser, which can be fixed with a future software update.

The Asus Tinker Board is undisputedly a powerful platform for makers, but no matter what power it may have, it has not managed to claim the crown from the Raspberry Pi, which offers greater documentation and support for those wanting to learn more. This is a board for experienced hardware hackers only.

The Tinker Board shares the same dimensions as the Raspberry Pi 3, and even fits inside the official case

Pi User Verdict

Asus Tinker Board

Developer: Asus
Web: asus.com
Price: £55

Rating 7/10

Google Wifi

The simplest and most value-packed home Wi-Fi mesh system that's been made yet, but do you actually need it?

In brief...

Google has jumped on the mesh Wi-Fi system bandwagon in an effort to become the centre of your smart home. Offering Simultaneous dual-band Wi-Fi (2.4GHz / 5GHz) supporting IEEE 802.11a/b/g/n/ac with a modular system that its easy and affordable to expand, Google may be onto a winner.

Mesh networking isn't new technology – it was original developed for the military and used in hospitals and large commercial applications for many years. But Google and other manufacturers, such as BT, Linksys and Netgear, have launched whole-home mesh Wi-Fi networking systems. These use a router and several nodes to spread your Wi-Fi signal throughout your house to combat dead zones, create greater coverage, faster speeds and a more consistent connection.

Naturally, the smart home-obsessed Google is all over this technology and as it turns out, Google may very well have crafted the best Wi-Fi mesh system to date. Google Wifi offers more mesh units than competitors for far less, with a focus on simple setup and management. Google Wifi costs £129 for a one-pack and £229 for the two-pack, which Google classes as it's 'medium' setup for 1,500 to 3,000 square foot. (The US has a $299 three-unit 'large' pack.) That's one primary 'Wifi Point' (the one you hook up to the modem or gateway) and however many secondary Wifi Points you choose to buy. Contrast this with the likes of the Netgear Orbi and others that cost $400 (or £320) and Google offers more units for less money than any competitor.

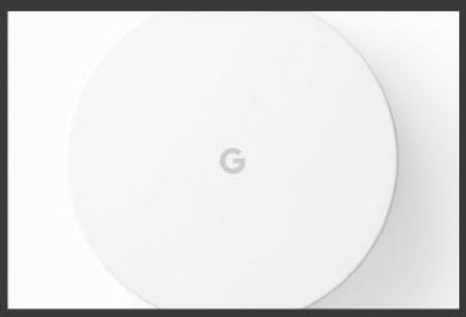
Each Google Wifi unit is a tiny, unassuming cylinder with a simple white LED band in its centre and capable of the same functionality. This means that any of the three units can function as the 'router' of the system, while the others can bestow wired internet (which is beamed to the unit wirelessly) with their included Ethernet ports. All three units are powered via USB-C.

Google's hardware design is gorgeous, while the setup is equally sublime. This works using a free iOS and Android app to facilitate the process. The app configures your Wi-Fi network by first scanning the QR code on the Wi-Fi Points. From there, the app tells you to name your network and set a password, then pair the additional Wifi Points and label them in the app for reference. Again, it takes seconds for the 'router' to recognise the Wifi Points and for them to begin broadcasting. You're not going to get the same depth of access, so no band switching for you. However, Google Wifi handles this in the background automatically.

"GOOGLE'S HARDWARE DESIGN IS GORGEOUS, WHILE THE SETUP IS EQUALLY SUBLIME"

The app offers plenty more useful features, like constant monitoring of your network, your Wifi Points, and the devices connected to it. The app has an included internet speed test similar to that of Ookla's, a mesh test that measures the health of your Points' connections as well as a Wi-Fi

Features at a glance



Sophisticated design
Dare we say it in a pro-open source mag like ours, but Google Wifi's clean white finish reminds us of Apple – and we love it!

Modular mesh
If you have a medium to large size home you'll likely only need two Wifi P=oints, but you can always add more if you like.

Google Wifi benchmarks

Here is how the Google Wifi fared in our brief suite of tests (which we conducted on a 100Mbps service):

Test	Test details	Download | Upload
Ookla Speed Test 5GHz	Within 5 feet/1.52 metres; no obstructions	101.41 | 117.83
Ookla Speed Test 5GHz	Within 13 feet/3.96 metres; three plaster walls	97.05 | 118.67Mbps
Ookla Speed Test 2.4GHz	Within 5 feet/1.52 metres; no obstructions	47.53 | 96.72Mbps
Ookla Speed Test 2.4GHz	Within 13 feet/3.96 metres; three plaster walls	50.95 | 82.98Mbps

Test	Test details	Peak speed
1.5GB Steam download 5GHz (peak speed):	Within 5 feet/1.52 metres; no obstructions	101.41 | 117.83Mbps
1.5GB Steam download 5GHz (peak speed):	Within 13 feet/3.96 meters; three plaster walls	97.05 | 118.67Mbps
Ookla Speed Test 2.4GHz	Within 5 feet/1.52 meters; no obstructions	47.53 | 96.72Mbps
Ookla Speed Test 2.4GHz	Within 13 feet/3.96 meters; three plaster walls	50.95 | 82.98Mbps

test that measures your connection strength from within the network. You can also prioritise bandwidth to one device for a time, control smart home devices and pause internet access to certain devices in a family setting – all from within this app.

In terms of performance, Google Wifi draws the absolute most from our 100Mbps Wi-Fi service, but can do so from every room of our, albeit small, house (see table, above). Wi-Fi mesh systems like Google's aren't focused so much on throughput as they are coverage, but this product delivers regardless.

The traffic prioritisation feature also ensures you get more of that crucial bandwidth than the other devices in your house. Plus, the network can automatically repair itself should one or more of the Wifi Points be unplugged or otherwise lose power.

While we know that Google Wifi operates its mesh system over existing Wi-Fi bands (2.4GHz and 5GHz) using 802.11s mesh protocol rather than Netgear Orbi's 5GHz tri-band system, we haven't found a big difference between either's performance. We do see slightly faster download speeds in MB/s on the 2.4GHz band from the Orbi over the Google Wifi, but that could be an anomaly.

Ultimately, Google Wifi offers excellent coverage over a wide area for little cost.

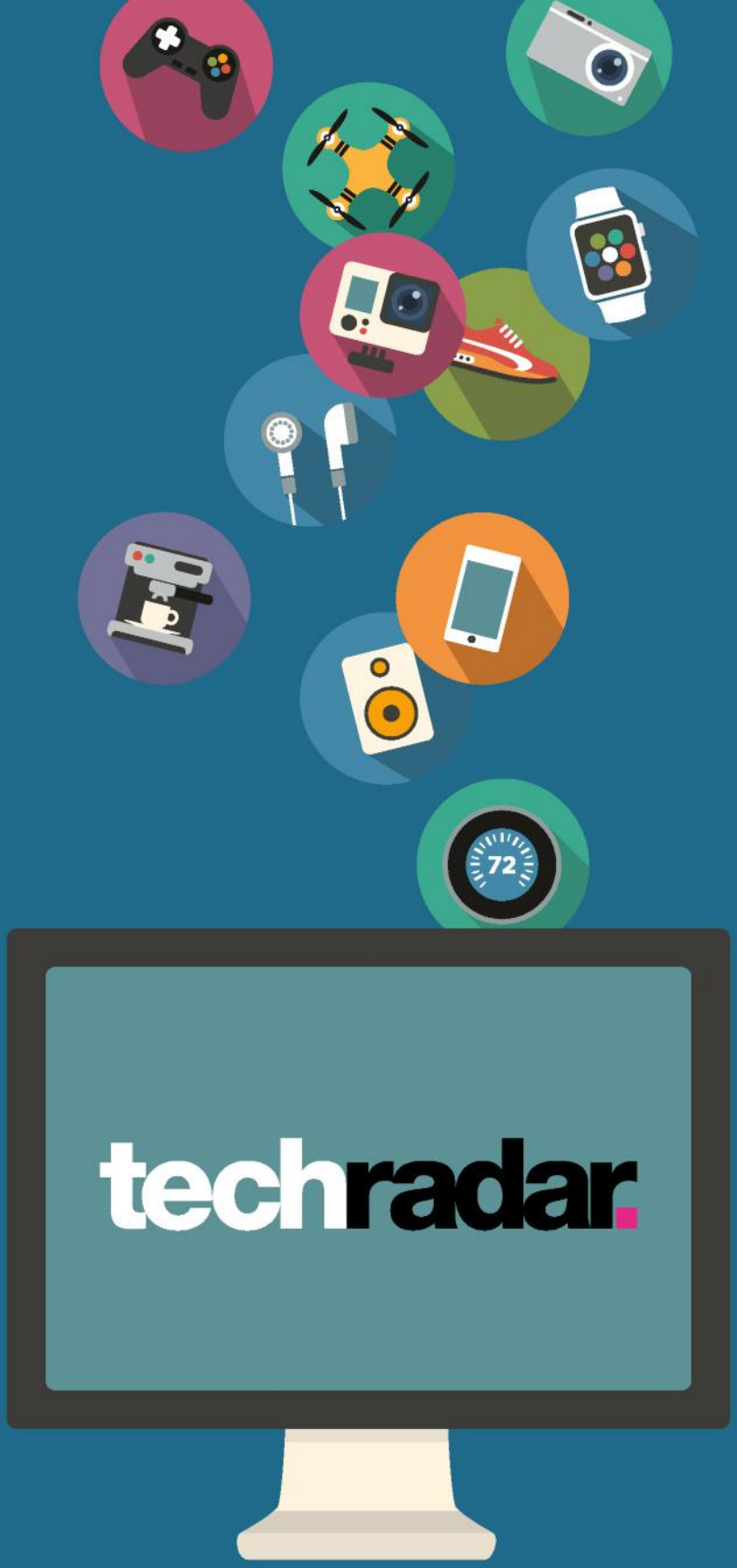
Google Wifi comes with a clever app (available for both iOS and Android) for setting up your internet

Pi User Verdict

Google Wifi

Developer: Google
Web: madeby.google.com
Price: £129–£229

Rating 9/10



The home of technology

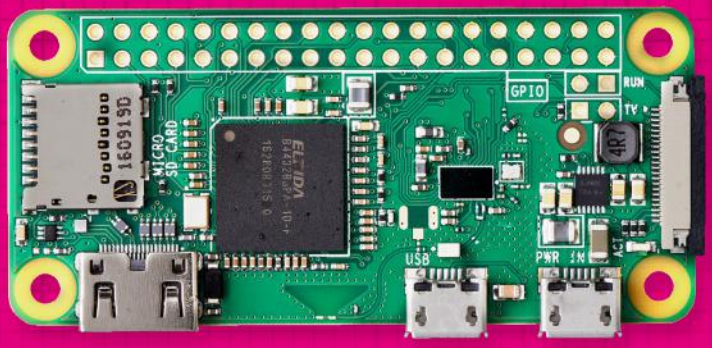
techradar.com

PiUser

YOUR NEW SOURCE FOR RASPBERRY PI
JOIN THE PI REVOLUTION

Discover the world of Raspberry Pi

The latest Pi projects, news and reviews



Perfect for owners of all models of the amazing Raspberry Pi and the fantastic Pi Zero

116 PAGES OF Pi GOODNESS INSIDE!

LEARN HOW TO...