70 PAGES OF ESSENTIAL RASPBERRY PI TIPS AND TUTORIALS

The MagPi

Win! £200+ of Robotics equipment FROM PIBORG.ORG

The official Raspberry Pi magazine Issue 33 May 2015 raspberrypi.org/magpi

REMOTELY CONTROL YOUR PI PROJECTS
SSH into your Pi from around the world

PICADEMY CELEBRATES ITS FIRST BIRTHDAY
We talk to its creator Carrie Anne Philbin

RASPBERRY PI DESKTOP SUPER-TEST
Find out which desktop distro deserves a place on your Pi

THE MINECRAFT MIDAS TOUCH
Turn trees into gold (and much more)

OUR FAVOURITE PI CASES REVEALED
Four sub-£15/$20 models rated

Also inside:
- WATCH IPLAYER ON YOUR PI
- MAKE A GAME WITH PIGLOW
- HACK THE RASPBIAN DESKTOP

BUILD AN IOT ALARM CLOCK
Raid the cupboards for a servo motor, a fork & a sweet tin lid (you're going to need them)

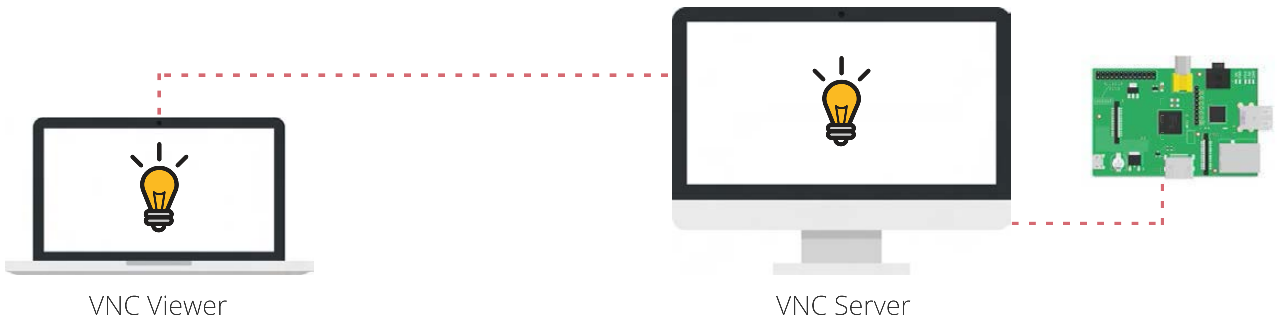
Plus MORE AMAZING RASPBERRY PI PROJECTS MADE BY YOU

VNC® now available for RASPBERRY PI!

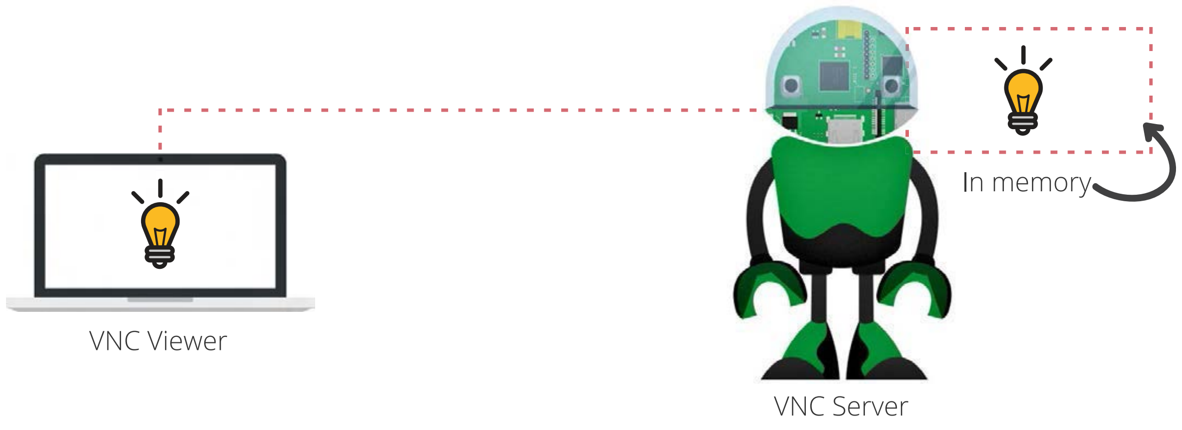
RealVNC has released VNC for Raspberry Pi, which means you can now connect to your Pi from any Windows, Mac or Linux computer, or iPhone, iPad or Android device!

VNC can be used on your Pi in two ways:

1. If the Raspberry Pi is connected to a monitor or TV and is running a graphical desktop, you can see exactly what a user sitting in front of the Pi would see; perfect for generic remote access or sharing a screen with a friend!



2. If your Pi is headless or not running a graphical desktop, then graphical remote access can still be achieved by using VNC to create a virtual desktop instead; ideal for users who have built their Pi into a robot.



Download VNC: www.realvnc.com/download/
Getting connected: www.realvnc.com/products/vnc/raspberrypi/

For more information contact sales@realvnc.com

WELCOME TO ISSUE 33!

It's hard to believe, but this is already our third new-look issue! The level of support the magazine is receiving from the Raspberry Pi and wider technology communities has been utterly fantastic. Believe it or not, The MagPi is one of the most popular technology magazines on the market today - it's seriously exciting stuff!

Of course, it doesn't hurt that the Raspberry Pi has sold more than five million units since its release in 2012, meaning it's already one of the most successful British computers ever made. With the advent of the Raspberry Pi 2 and Windows 10 IoT support from Microsoft, there's every possibility another five million will be sold in the next year or so alone. That makes for a truly massive community!

Before I leave you to dig into the news, features and tutorials we have in store for you this issue, I'd like to remind you that you can buy this and the other new-look MagPi issues in the App Store or on Google Play – just click on the little buttons at the bottom of the page. Some of you have been asking why we sell the magazine digitally when the content is released under a Creative Commons licence, but the answer is simple: it's an easy and convenient way for MagPi readers and Pi enthusiasts to support the magazine and the Raspberry Pi Foundation. It also means you can enjoy the magazine on your favourite Apple and Android devices with all the mod cons like automated uploads, speedy rendering, and clickable links!

Russell Barnes

70 PAGES OF ESSENTIAL RASPBERRY PI TIPS AND TUTORIALS
The MagPi
Win! £200+ of Robotics equipment
RASPBERRY PI DESKTOP SUPER-TEST
Find out which desktop distro deserves a place on your Pi
Plus MORE AMAZING RASPBERRY PI PROJECTS MADE BY YOU

FIND US ONLINE raspberrypi.org/magpi **GET IN TOUCH** magpi@raspberrypi.org

The MagPi

EDITORIAL
Managing Editor: **Russell Barnes**
russell@raspberrypi.org
Technical Editor: **David Whale**
Sub Editors: **Phil King, Laura Clay**

DESIGN
Critical Media: **criticalmedia.co.uk**
+44 (0)1202 399824
Head of Design: **Dougal Matthews**
Designers: **Mike Kay, Lee Allen**

PUBLISHING
For advertising & licensing:
russell@raspberrypi.org
Publisher: **Liz Upton**
CEO: **Eben Upton**

With thanks to this month's contributors: **Mike Cook, David Crookes, Alex Eames, Gareth Halfacree, Lucy Hattersley, Phil King, Simon Long, Simon Monk, Martin O'Hanlon, Les Pounder, Matt Richardson, Richard Smedley, Sean M Tracey, Richard Waterworth** and **Robin Withers.**

Available on the App Store
Get it on Google play
BY NC SA

The MagPi magazine is published by Raspberry Pi (Trading) Ltd., Mount Pleasant House, Cambridge, CB3 0RN. The publisher, editor and contributors accept no responsibility in respect of any omissions or errors relating to goods, products or services referred to or advertised in the magazine. Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0). ISSN 2051-9990.

70 PAGES OF ESSENTIAL RASPBERRY PI TIPS AND TUTORIALS
The MagPi
Win! £200 of Robotics equipment
RASPBERRY PI DESKTOP SUPER-TEST
Find out which desktop distro deserves a place on your Pi
THE MINECRAFT MIDAS TOUCH
OUR FAVOURITE PI CASES REVEALED
BUILD AN IOT ALARM CLOCK
Plus MORE AMAZING RASPBERRY PI PROJECTS MADE BY YOU

Contents

Issue 33 May 2015 raspberrypi.org/magpi

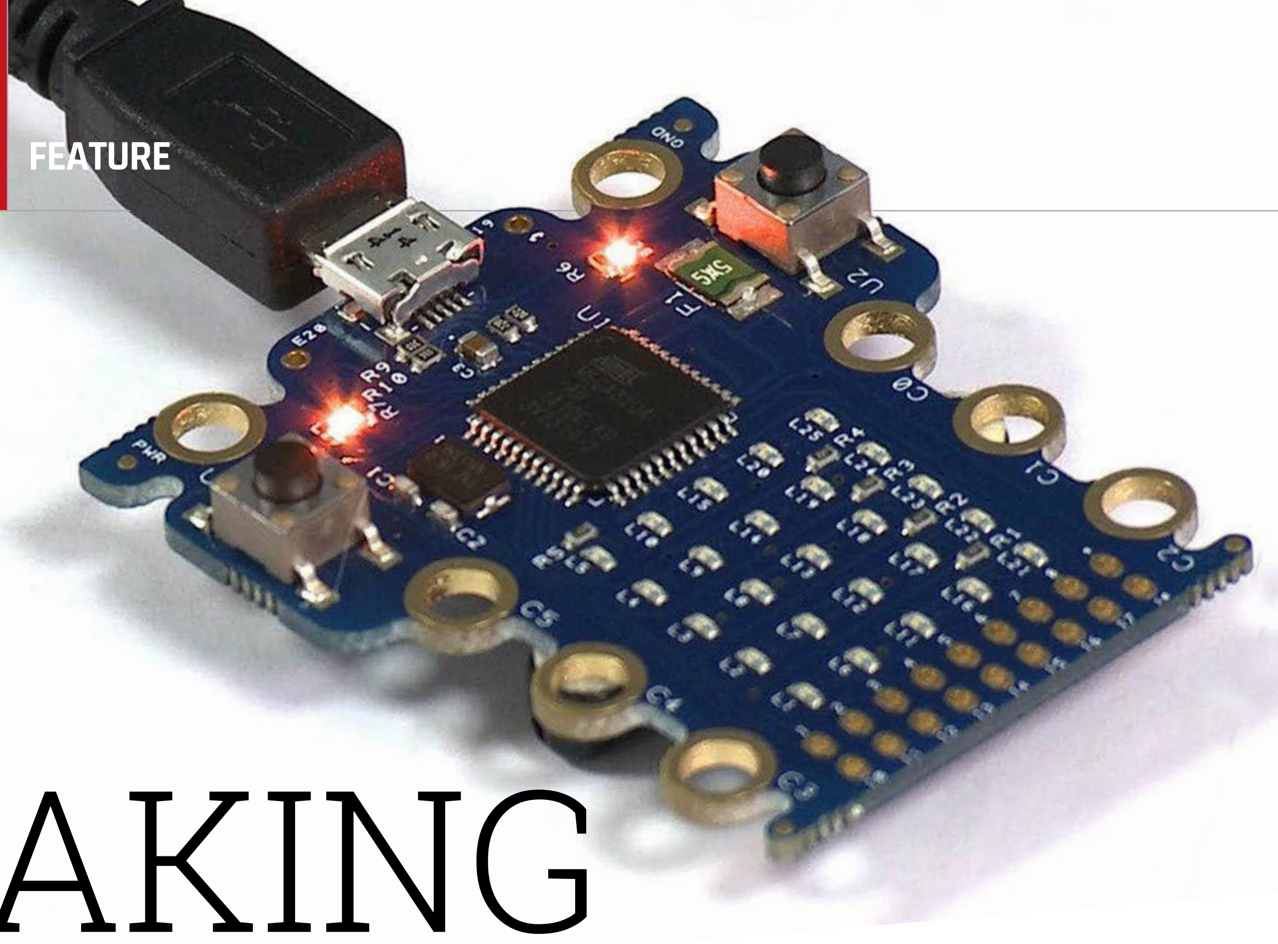
SAVE 45% with a Newsstand subscription (limited time offer)

Available now for smartphones & tablets

Get it on Google play

Available on the App Store

The MagPi



MAKING IT HAPPEN

As the BBC looks to complement the Raspberry Pi with an entry-level programmable device called the Micro Bit, we talk to its advisor Ian Livingstone about the importance of coding...

Digital creativity is in the spotlight like never before. The Raspberry Pi has opened up computing in a way not seen since the 1980s, *Minecraft* has shown that gaming is as much about creation as it is consuming, and there are Code Clubs being set up in towns and cities everywhere to get children 'under the bonnet' of machines and see just how they tick.

Stepping into this scenario is the BBC, which has recently launched a UK-wide initiative called 'Make It Digital' to inspire a new generation to get creative with coding, programming, and technology. At its centre is a new coding device being given to every child in year 7 – a million units in total. It is being nicknamed the Micro Bit, harking back to the days of the BBC Micro. But while that name is subject to change, the idea behind it is not.

The Micro Bit is a tiny, wearable device with a display of 25 embedded LEDs. It has a single micro-USB connector, low-power Bluetooth, digital and analogue I/O, an accelerometer, and magnetometer. Children are able to program simple messages or animation and even connect it to a Pi. It has been designed to be introductory, complementing the strides made by the Raspberry Pi rather than competing with it.

"Raspberry Pi has worked with the BBC on the Micro Bit," says Ian Livingstone, veteran games designer and advisor for Make It Digital. "It's an entry-level device produced in collaboration with many partners and it's about getting people through the funnel of technology. I fully support the Pi because it's a fantastic tool to enable creativity to be manifested, whether it's though building a website, doing some robotics, making an app, or building a game."

Key to Make It Digital – which will be backed by a season of TV programmes and online activity involving top BBC shows such as *Doctor Who* and *Eastenders* – is the desire to recapture the spirit of coding. Ian Livingstone

EARLY LEARNING

Year 7 comprises children aged 11 and 12, but should this initiative not be aimed even younger? "Ideally you want to get all children of all ages in primary schools learning the basics of computer science as a discipline of problem solving, computational thinking, algorithmic thinking, and logic, but if you are going to give each one of them a device, you need to take some responsibility and ensure it's not a wasted opportunity. I think 11 and 12 is the right age," says Ian Livingstone.

laments that the UK has "lost a generation" – "I just think it's a great shame that [from] the mid-80s until last year, children have been taught how to use technology but given no means to create their own" – but he hopes the Micro Bit, as well as the Pi, will go some way to turning things around.

The big problem, he argues, is that while some schools did teach computer science and coding, it wasn't mandatory. Most ICT teachers, he adds, just taught the curriculum as it was, teaching digital literacy without any digital making. Fortunately, that has changed. The new computing curriculum introduced last September marked a fundamental shift towards coding. And while it sparked some nervousness among teachers, Ian makes no apologies.

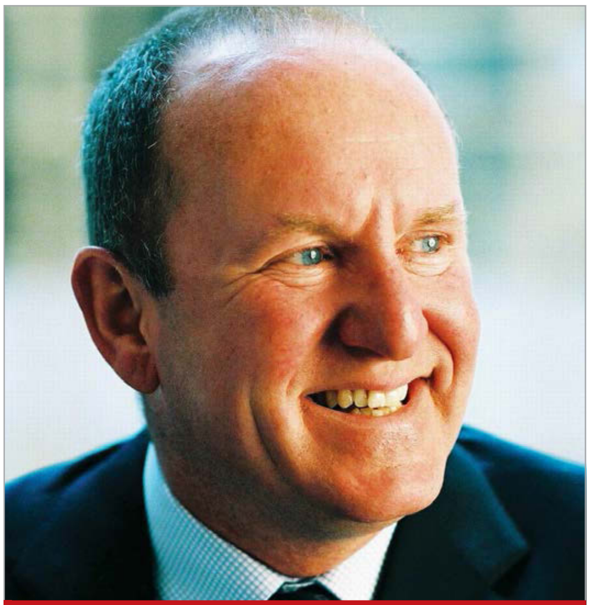
"This can be transformational for the country, but it does come with challenges about how well equipped the teachers are to teach computing science, and coding in particular," Ian explains. "A lot of them aren't, but that's not a reason not to do it. You just have to accept that there is always going to be some child in the class who knows more than the teacher. So teachers should learn alongside the children and there should not be any problems with egos here. The teachers can facilitate a joint learning experience where they learn alongside children naturally, collaborating with peer-to-peer learning."

The Micro Bit, he says, is part of the solution, helping teachers, parents and pupils to adjust and learn. He believes it will be a great enabler of digital creativity and he draws parallels with the BBC Micro's record of helping to give rise to the games industry. His big hope is that the Micro Bit will encourage children to code when they may not have otherwise done so. "A lot of them may be thrown into the river, hit with a hammer, or sold on eBay, but I think the net output of giving one away to every child will be hugely positive," he asserts.

According to the BBC, the Micro Bit is all about encouraging independent learning among pupils. It wants teachers, parents, and children to watch out for supportive programming on the CBBC channel and to head to its website for live BBC Learning lessons and other educational content. The shows will hark back to the likes of *The Computer Programme*, which ran for ten episodes on BBC 2 in 1982.

"The Micro Bit is for children, but clearly teachers and parents need to understand how to use it," notes Ian. "The algorithm is just a set of instructions to this little device and you can type in code to make it do simplistic robotics. It's entry-level stuff, but it's part of the journey and the important thing is learning by doing. You can't learn computer science from a textbook, just like you can't learn to speak a foreign language by memorising 50,000 words. Children need to speak the language of code and engage with it. *Minecraft* and *LittleBigPlanet* have paved the way, showing that you're not just a passive recipient of content, but can actually manipulate it and make it do something. With the Micro Bit and the Raspberry Pi, you can create your own games. It's the next step on the path."

> "I fully support the Pi because it's a fantastic tool to enable creativity to be manifested"



WHO IS IAN LIVINGSTONE?

He is an advisor for Micro Bit, but what about his technology industry background?

Ian Livingstone is a founding father of the UK games industry. He co-founded Games Workshop in 1975, wrote many titles in the *Fighting Fantasy* book series, and launched global gaming franchises including *Tomb Raider*. He also co-wrote *The Livingstone-Hope Skills Review*, a Next Gen report published by Nesta in 2011 which set out how the UK could transform into a leading talent hub for the videogames and visual effects industries.

The report had been commissioned by Ed Vaizey, the Minister for Culture, Communications and the Creative Industries, and it stressed the importance of placing computer science on the curriculum as an essential discipline. Civil servants within the Department for Education were initially reluctant to take the recommendations on board. "But we got to Michael Gove's special advisors and the penny dropped," says Ian Livingstone. "We convinced the Secretary of State and his team. The new Computing curriculum was launched last September."

Above The BBC Micro played a big part in the 1980s coding boom

WHAT WOULD YOU DO WITH A RASPBERRY PI 2?

Back in issue 31 we asked what you'd do if given a shiny Raspberry Pi 2 to play with. This issue we're rewarding our top five favourite ideas, thanks to **Pimoroni.com**...

Below We offered five Raspberry Pi 2s in issue 31 for your project ideas. We were inundated with entries

In association with PIMORONI

5 Pi 2s must be WON!

WHAT WOULD YOU DO WITH A RASPBERRY PI 2?

Let us know for your chance to win!

How to enter:

For your chance to win a Raspberry Pi 2 with a Pimoroni case of your choice, tell us what your dream Raspberry Pi project is. Simply email **competition@raspberrypi.org** with no more than 100 words detailing your perfect project, hack, or make.

Winners can pick a Pibow (pictured) or Coupé case in a colour of their choice. See them at shop.pimoroni.com

Terms & Conditions

Competition closes 30th March 2015. Prize is offered worldwide to participants aged 18 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families or friends. Winners will be notified by email after the draw date. By entering the competition, the winner consents to any publicity generated from the competition in print and online. Participants agree to receive occasional newsletters from The MagPi magazine (unless otherwise stated upon entry). We don't like spam. Participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered.

We relaunched *The MagPi* soon after the Raspberry Pi 2 was released, so we knew immediately that we had to give a few away. We spoke to our friends at **shop.pimoroni.com** and they kindly agreed to sponsor us, offering to give away five Raspberry Pi 2s and a Pibow or Coupé case of the winner's choice.

Our question, in case you need reminding, was simple: what would you do with a Raspberry Pi 2? The response was fantastic: hundreds and hundreds of you got in touch with some brilliant, intelligent, wacky, and downright stupid ideas; we've laid out our winners here. While the competition is over, we still want to know what you're doing with your Pi. Let us know about your innovative hacks and projects via **magpi@raspberrypi.org**, or turn to page 64 and take part in this month's competition to win a Raspberry Pi robot courtesy of our friends at **PiBorg.org**.

Office server

Long-time Linux user Aris Tsitras is going to create an office server with the Pi 2, since it fits the needs of his small office environment. "The Raspberry Pi is perfect for business as well as for pleasure. My office consists of ten people and to have a fully fledged server to handle email, webpages, PBX and DNS would be just too much." A Raspberry Pi 2 or two – assuming you have a sufficiently small load – would do the job just as effectively, but considerably more efficiently. "As for pleasure, we have a media centre setup to chill during our breaks."

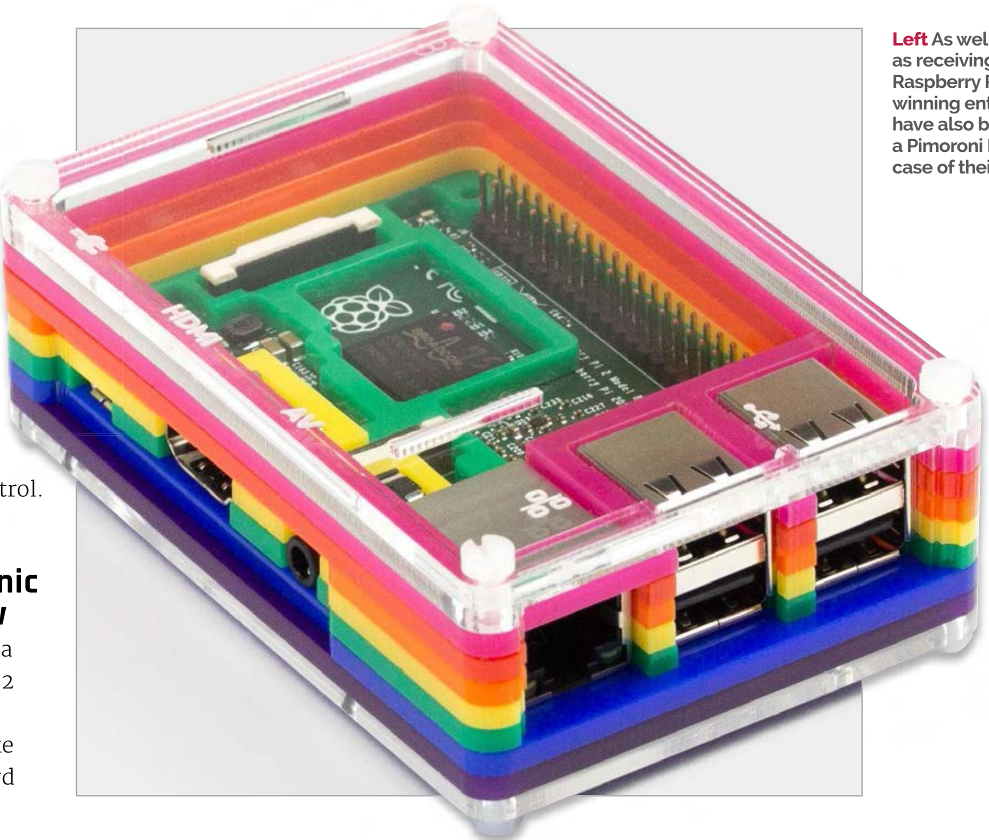
We're looking forward to seeing how the Raspberry Pi 2 performs in its office job!

Custom media centre

Guido Erlinger of Germany told us he's going to use his Raspberry Pi 2 to make a specially adapted media centre player for his physically and mentally disabled son. While he has plenty of media at his disposal (around 250 movies on DVD no less), his son is unable to change DVDs by himself.

"The aim of the project would be to use a small system that consumes little power. The challenge will be to create an application which presents all the category pictures and the DVD cover images, so he can easily choose an DVD."

Media centres are already one of the most popular uses for the Raspberry Pi outside hacking, so Erlinger's son will likely find some independence with one of the many solutions like Kodi or OpenELEC, not least when you consider that – once the movies are backed up on an external hard drive – it's trivial to navigate movies, music and pictures, using a modern HDMI TV's remote control.


Left As well as receiving a Raspberry Pi 2, the winning entrants have also bagged a Pimoroni Pibow case of their choice

> "Media centres are already one of the most popular uses for the Raspberry Pi outside hacking"

Animatronic scarecrow

"I would use a Raspberry Pi 2 to control an animatronic scarecrow to make with my son," explains Richard Mitchell, a graphic designer at Brunel University London. While robotics is one of the most popular hands-on activities with the Raspberry Pi's versatile GPIO pins, we've never encountered an animatronic scarecrow before, especially one designed to interact with passers-by and tell them jokes (something we're pleased to remedy).

Its get better, though: "It would be made of lots of seagulls hidden inside a big coat, which opens up to reveal them at the end." Why? "Because gulls need to scare crows so they can get more chips."

Richard and his son plan to enter it into the annual Hayling Island Scarecrow Competition (**bit.ly/1J9p50p**) and we can't wait to see how it fares!

Fly Fishing

"The Raspberry Pi project I'm working on is designed to help fly fishermen in Norway evaluate the fishing conditions in the Rena river without having to be there," starts Peter Davidse.

"The Pi will be measuring rainfall, air pressure, wind speed, wind direction, water temperature, water flow, water level, and air temperature. All the data will be presented on a webpage for PC and mobile devices, and as an excerpt sent by mail."

Peter goes on to explain that he may later add a camera to monitor a stretch of river, so he'd better watch out for our guide on how to do just that with the Raspberry Pi Camera Module in an upcoming issue. It's surprising just how easy it is to set up.

If you would like to get started with 'physical computing' (basically, sensing the real word with technology) on a slightly smaller and more affordable scale, check out our five-star review of the latest CamJam EduKit on page 59.

Retro gaming cabinet

Andy Jackson's planned project revolves around retro gaming: "My dream Raspberry Pi project is to firstly make a custom cocktail cabinet with arcade controls for my daughter, so she will be able to play old-school arcade games." Well, that dream just got one step closer to reality!

Given that Andy's daughter Faye is only just about to turn one, we have a sneaking suspicion it'll be Andy tapping away at retro classics for a while yet!

"As she gets older, I want to use the Pi to teach her how computers work, and show her what her daddy does for a living (hopefully, doing my part to get the next generation of girls into IT)."

PICADEMY: TEACHING THE TEACHERS

The Picademy initiative has just had its first birthday. To celebrate, we caught up with its creator and former teacher, Carrie Anne Philbin, to find out more about how the project helps educators to teach computing in schools...

WHAT IS PICADEMY?

The Raspberry Picademy is a **free** professional development experience for primary and secondary teachers, open to individuals around the world. Over the course of two days, 24 teachers get hands-on with computing, and discover the many ways in which the Raspberry Pi can be used in the classroom. No experience is necessary; the Foundation's Education Team will help you discover practical ways in which Raspberry Pi can support and further your teaching of the new curriculum. Learn more and apply today at: **raspberrypi.org/picademy**

When the new creative-focused Computing curriculum was announced for September last year, it sought to unlock the secrets hidden behind a myriad glossy operating systems and empower pupils to learn about more than just Word, PowerPoint, and Excel.

But there was a small problem. Some teachers were too poorly equipped to teach the new subject and feared their skills gap would be exposed for their pupils to see. Yet, thanks to a free two-day course for primary and secondary teachers called Picademy, many of them have grown in confidence.

Picademy is the brainchild of Carrie Anne Philbin, a former East London Computing and ICT teacher who left to take on the role of education pioneer at the Raspberry Pi Foundation. She got the idea for Picademy after attending the 2014 BETT Show, a huge education exhibition in London, and noting the changing attitudes among teachers. The initiative has now been running for the past year, during which time it has supported 200 teachers.

"I'd attended in 2013 when I was still a teacher… I was hanging around the Pi guys and the OCR stand, and all of the questions they were fielding concerned the reasons for using the Raspberry Pi in the classroom," she tells us. "In 2014, it was very different. Teachers were asking about our teaching materials and they were convinced the Pi would help them. One thing that struck me over the four days was the

Carrie Anne Philbin is also the author of Wiley's *Adventures in Raspberry Pi*

The Picademy 'Pi-deas' wall invites teachers to share ideas

SIGN UP for Leeds TODAY bit.ly/pi-leeds

number of people asking when we would launch a teacher training programme."

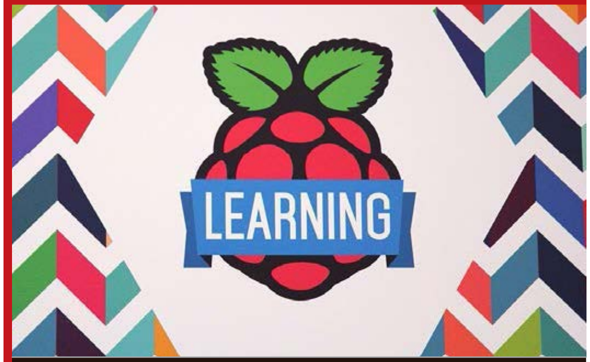
The first Picademy ran on 14 and 15 April in 2014, and Carrie Anne had set down some rules about who would make up the class of 24. She wanted half of the teachers to be from a primary school and the other half from a secondary school. A further four places would be handed to teachers working in other countries. Everyone was asked to fill in an application form.

Smooth operation

Carrie Anne also had a clear idea of how she wanted the Picademy class to operate. "It was always going to be over two days – one for the workshop and one that allowed teachers to really make it their own and get something from it," she reveals. "We wanted to have it in Cambridge, too, because it was easy for us to do. We had a new office in a larger space, and access to kit and people, so it kind of made sense. But above all, we wanted to create the best possible environment so that we could inspire teachers a little bit more, get them started with Pi, and build confidence around computing and computer science in general."

Giving teachers the confidence to approach teaching in a new way has been one of the most crucial aspects of the Picademy. "A lot of teachers would say they were getting started and they were not afraid to have a go, but they did worry that pupils were already ahead of them and they wanted to know how to stay one step ahead,"

A WORLDWIDE FOCUS



So far, Picademy has attracted educators from the USA, the Czech Republic, Lebanon, and Portugal. But Estonian teachers appear to be taking most interest. "Estonian teachers are coming on our courses because Estonia is doing what we are doing – getting children interested in computing," says Carrie Anne Philbin. "A lot of people in Europe are watching what we are doing. It's very exciting."

Carrie Anne explains. "We looked to change that thinking and say it's okay for students to be ahead. It's about how you deal with that as a teacher. For us, it's about creating a community of teachers who can talk and support each other."

> We warn teachers that we will blow their mind with a lot of information, but tell them that it is worth it

The sessions are tailored to facilitate this. The first days are "full of stuff and in your face," says Carrie Anne. "We warn teachers that we will blow their mind with a lot of information, but tell them that it is worth it." The day involves back-to-back workshops that introduce teachers to many aspects of physical computing – including Scratch, Python, Pi Camera, Sonic Pi, and *Minecraft Pi* – and by engaging teachers in fun activities, they are able to see how computing can be cross-curricular. At the end of the first day is a brainstorming session, jotted down on a series of sticky notes, which looks at the projects the teachers would like to work on and take back to the classroom. This continues into the evening over dinner.

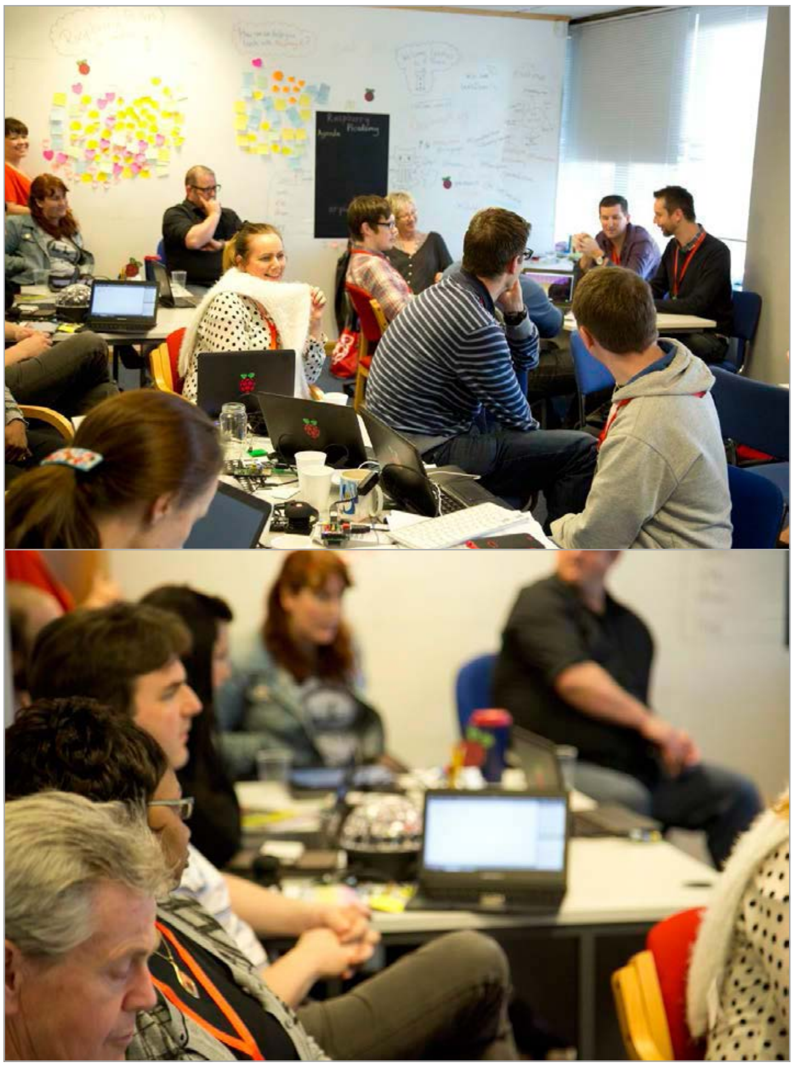
"They are exhausted by this point, so we go for a meal, which is nice because it keeps the conversation going," confides Carrie Anne. "It gears them up for the following day, which starts with Eben [Upton] talking about the history of the Pi, what we're doing, and where we're going. We also have talks from the community. It's important teachers realise there is a good community to tap into."

By mid-morning on the second day, the teams look back at the sticky notes, break into groups, and make something of their own. They present their findings to the group, get a badge and certificate, and are sent on their way. "The second day is what everyone raves about," says Carrie Anne. "Often we can't get rid of people – they want to live here."

INSPIRATION AND SUPPORT

"Picademy allowed me to gain a great network of educators, from across the age ranges, who can inspire and support me. I gained confidence in my own computing abilities, recognising what I could already do, as well as learning new skills and knowledge. I also gained an understanding of the variety of activities and approaches that could be used. Seeing several people in one room given the same resources but creating their own thing is a fantastic way to demonstrate creativity."

Sway Grantham, primary computing teacher and Raspberry Pi certified educator



Above There's a lot to cram into the training, and teachers go hands-on with Pi projects

Mixed abilities

So far there have been seven Picademy sessions and they have attracted a wide range of specialist teachers, from computer science to music and art. "We even had a history teacher," Carrie Anne adds. As such, each session has had a mixed range of abilities. "We have had experts in computer science and people with no programming experience whatsoever, but it's allowed us to demonstrate that computer science can underpin a lot of other subjects. By personalising the sessions for each teacher, we ensure they take away something useful to them and their classes."

Not everyone has stayed the course. Although the sessions are always oversubscribed, some teachers do drop out. In the first four Picademies, some left after the first day. "I think some have an assumption that all of the answers will be handed to them, but it's not about that," she insists. "It's about encouraging teachers to find answers themselves and it's about getting an idea of computer science as a problem solver. It's perhaps not for everyone." Even so, the team says it learns from feedback. It has introduced robotics to the workshops and it now allows teachers to take away the provided SD cards. Outsiders including Martin O'Hanlon are contracted to help with sessions, too.

The Picademy has been such a success that the Raspberry Pi Foundation is thinking of expanding it. Last November, a course ran for Welsh teachers in Wales and, on 26 and 27 May this year, the Picademy North is running at the National STEM Centre in York. Carrie Anne is thinking of creating another in the south-west of England. There is every chance that there will be Picademies in towns and cities across the UK in the future.

"We are looking to scale it [up]," she discloses. "But we do have a problem because we are a small educational team working with each other in one building. If we put on a Picademy in a bunch of cities, we'd have to contract it out and there could be a drop-off in quality. We'd have to rebrand it, create a Picademy X, because I would be concerned with branding, but it's something we are considering and would do if we got the franchise model right and the appropriate team. We'll see how Picademy North goes."

One thing is for certain: Picademy fills a hole. "The government invested £3 million in the re-skilling of teachers, but that only works out as £170 per school, which doesn't cover a teacher for a day," says Carrie Anne. "Our course is free, which is one of the reasons so many apply. Still, it gives us a chance to dispel some myths. The media has presented the curriculum changes as being all about code, but that's a red herring. It's about computational thinking. We're not teaching five-year-olds machine code."

Above Picademy isn't just limited to Pi Towers, or even Britain's borders

MANY USEFUL IDEAS

"The Picademy's Continuing Professional Development was excellent and it gave me many useful ideas to go back to school with, particularly the use of *Minecraft Pi*. It was a great experience to spend time with like-minded people who could see how the Pi can be of benefit in supporting the computing curriculum. I use the Pi in my school, and I teach workshops on *Minecraft Pi* and Python programming."

Sarah Zaman, primary computing teacher and Raspberry Pi certified educator

Hello, World!

SoundIfI

RISC OS specs

MagPi

Snappy Core stuff

MagPi Film

Print Me

BLASTER!

Raspberry Pi

Pidora bits

My Stuff

World Map

Recycle Bin

Raspbian notes

Raspberry Pi Desktop SUPER-TEST

Which operating system should power your Raspberry Pi? We put four top Pi distros to the test...

How we tested

We installed the distros on a Pi Model A and Raspberry Pi 2, using an EMTEC 8GB Class 10 SD card and a SanDisk 8GB Class 10 microSD card. We used a Mac Pro 3,1 with SD Formatter 4 to install each OS. We also used a Microsoft LifeCam VX-7000 and Maplin AD-102 breadboard during the test.

The Raspberry Pi packs an amazing amount of power into its tiny frame. But hardware is only half the story; the other half is the software you choose to control it. When setting up your Raspberry Pi, there are a myriad of operating systems (OS) to choose from, most based on different distributions of Linux ('distros' for short).

Raspberry Pi users will be familiar with Raspbian, the operating system based on Debian Wheezy Linux that is marked as 'Recommended' during NOOBS setup. Raspbian is great to get started with, but it's far from the only OS available.

Every Raspberry Pi owner should take some time out to investigate the other available operating systems. Each has something unique to offer, from the forward-thinking approach to software distribution in Snappy, to the retro-infused alternative approach of RISC OS.

But which OS is the best to use? There's only one way to find out. In this test we looked at four key OS options: Raspbian, Pidora, RISC OS, and Snappy Ubuntu Core. All are free and have versions designed especially for the Pi.

That's not to say they're all the same: Raspbian and Pidora use different default desktop environments and Snappy Ubuntu Core doesn't yet have a desktop at all – software is installed using a brand new package manager called 'snappy'. RISC OS is an entirely different creature from anything you are likely to have used before.

Each OS offers something unique, and all are interesting and worth exploring. But which distro should be your main go-to operating system? That's what this group test is determined to find out.

Pidora

Packing the power of Fedora into the Raspberry Pi

Compatibility: Raspberry Pi 1
URL: pidora.ca

Pidora is a remix of the popular Fedora flavour of Linux. Currently in its fifth version, the Pidora distro has come a long way since 2012, but it still stands in the shadow of Raspbian. Pidora is part of NOOBS, but adoption and support has been dropping ever since Raspbian became the recommended installation.

Key features:

- Based on the popular Fedora version of Linux.
- Optimised for ARMv6 architecture; doesn't work with the Raspberry Pi 2.
- Default Xfce desktop is good-looking but sluggish compared to LXDE.

Pidora looks every bit as powerful as its bigger brother, but badly needs optimisation

Snappy Ubuntu Core

Fast and futuristic, but may be a little too advanced for most users

Compatibility: Raspberry Pi 2
URL: developer.ubuntu.com/en/snappy

Snappy is the most recent OS for the Pi. Users expecting standard Ubuntu with its Unity desktop will be surprised. Snappy is a command-line affair aimed squarely at developers looking to build web servers and Internet of Things (IoT) devices. The apt-get function is replaced with snappy, which offers some great innovations regarding software deployment.

Key features:

- Command-line interface with no desktop installation available.
- More robust 'transactional' software updates that are guaranteed to work.
- Software packages installed using snappy instead of apt-get.

Snappy Ubuntu Core offers a range of new features that are ideal for server and IoT developers

Raspbian

The recommended distro keeps going from strength to strength

Compatibility: Raspberry Pi 1 and 2
URL: raspbian.org

Raspbian is based on Debian Wheezy but optimised for the Raspberry Pi hardware. Technically, Raspbian isn't the official OS and isn't affiliated with the Raspberry Pi Foundation. However, its 'recommended installation' status, along with plenty of documentation and support, make it feel official. Raspbian is definitely the place for newcomers to start.

Key features:

- Largest user base, and most documentation and tutorials reference Raspbian.
- Compatible with all models of Raspberry Pi.
- Lightweight interface is fast and responsive, and it has a built-in Pi Store.

Raspbian is the recommended distro for newcomers

RISC OS

Flashback from the 1980s that's still going strong today

Compatibility: Raspberry Pi 1
URL: RISCosopen.org

Created during the heyday of Acorn Computers, RISC OS was designed explicitly for the ARM chipset and has been kept alive by a small team of dedicated enthusiasts. RISC OS is ultra-fast, but hails from a time before the modern GUI metaphor had settled, and its WIMP (Windows, Icons, Menu and Pointers) interface has more than a few quirks.

Key features:

- Ultra-fast thanks to its ARM focus and small footprint.
- Great for low-level programming and getting close to the ARM instruction set.
- Quirky desktop interface and lack of widespread adoption make it difficult to learn.

RISC OS looks normal on the surface, but is radically different to other operating systems

IMPORTANT!!!

Piano Teach

Untitled

Hello, World!

TEST 1

Installation

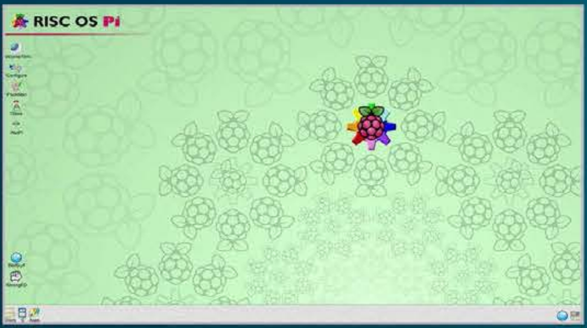
Pidora ★★★★☆	Raspbian ★★★★★
RISC OS ★★★★★	Snappy Ubuntu Core ★★★☆☆

We started our test using SD Formatter 4 to wipe our SD cards and installed NOOBS (New Out Of Box Software). NOOBS enables you to choose from a list of operating systems, including three on test here: Raspbian, RISC OS, and Pidora.

Raspbian starts up with a Software Configuration Tool that enables you to reclaim space on the SD card, change the password, choose boot options, and enable support for the Raspberry Pi Camera. Pidora has a more basic Setup Agent, which walks you through creating the user account. RISC OS boots straight into the GUI. Installation was straightforward in all these operating systems.

Snappy Ubuntu Core is the exception in that it has to be installed manually, using the dd tool to copy files from the image file to the SD card.

Upon startup, you're faced with a command line that informs you that apt-get has been replaced with snappy. (The password and login are both 'ubuntu'). Because the Raspberry Pi doesn't sport a clock backed up by battery, the date in Snappy is set to 1970 on start-up, and this prevents snappy software installation, so part of each startup process is using **`date -s`** to correct the date.



TEST 2

Hardware support

Pidora ★★★★☆	Raspbian ★★★★★
RISC OS ★★★☆☆	Snappy Ubuntu Core ★★☆☆☆

Pidora gets a black mark because it doesn't support the Raspberry Pi 2. Attempting to choose Pidora in NOOBS brings up: 'Warning Incompatible Operating System(s) Detected'. We tried anyway, but got a blank screen for our troubles. So for the rest of this test, Pidora was relegated to our Pi Model A.

Meanwhile, Snappy Ubuntu Core only supports the Raspberry Pi 2, which leaves a considerable amount of Raspberry Pi devices out of the loop.

All of our distros had no trouble recognising the keyboard, mouse, and network connection. Where things became a little more complicated was regarding accessories and GPIO. This is another area where Raspbian is rewarded by being the recommended distro. We had little trouble setting up our webcam and GPIO in both Raspbian and Pidora, but tutorials were easier to source for Raspbian. We couldn't get the webcam to work in RISC OS; GPIO in RISC OS is possible, but it's a confusing setup process.

The RPi.GPIO package isn't present in the snappy list of software, so GPIO access is currently unavailable in Snappy Ubuntu Core.

Print Me

World Map

Recycle Bin

TEST 3

Software support

Pidora	★★★☆☆
Raspbian	★★★★★
RISC OS	★★★☆☆
Snappy Ubuntu Core	★★★★☆

TEST 4

Programming and GPIO

Pidora	★★★★☆
Raspbian	★★★★★
RISC OS	★★★☆☆
Snappy Ubuntu Core	★★★☆☆

Every OS comes with pre-installed software and can download and install additional software via the internet. In Raspbian, software is installed using apt-get or via the Pi Store.

In Pidora, Yum is used to install software, but you can install Apt from Yum if you prefer (and then install the Pi Store manually).

RISC OS has a built-in store where you can download new software created by the RISC OS community. However, a heartbreaking aspect of RISC OS is that you have to use BeebIT and ArcEm emulation software to run the rich library of older BBC Micro and Archimedes software, which you could do just as easily from any Linux distro.

Ubuntu Snappy Core replaces apt-get with snappy: programs are split into frameworks and apps (which are isolated like apps on mobile devices). Snappy installs programs using a transactional update system that downloads components before updating or installing (and can roll back programs to earlier versions).

It also supports WebDM, which enables you to connect to your Pi from a web browser and install or remove software. Snappy is very interesting, but very little software is currently available.

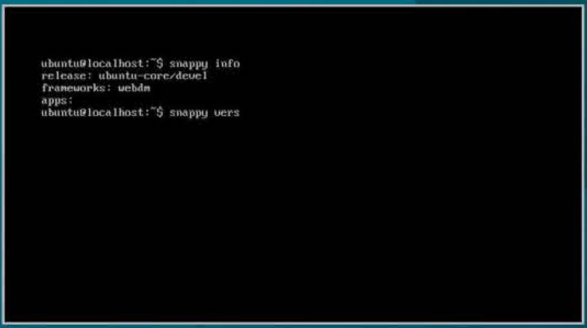
Both Raspbian and Pidora make it pretty easy to cover the basics of programming, and accessing the GPIO is a case of downloading the correct library (such as RPi.GPIO for Python or Scratch GPIO).

Snappy Ubuntu Core comes with Python, but without apt-get there was no way for us to install the RPi.GPIO tools. Snappy Ubuntu Core is tremendously interesting to advanced developers though, and is an ideal platform for setting up your own web server, but beyond that it's far from an ideal environment for learning to code.

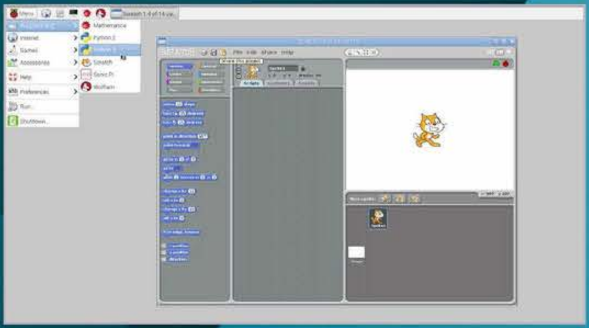
RISC OS is either a fantastic or dreadful programming environment, depending on what you want to learn. Press **CTRL+F12** and enter **`basic`** to access BBC Basic, a fantastic starter language that brings tears to the rose-tinted eyes of many a seasoned developer.

Beyond nostalgia, RISC OS is particularly good for low-level programming and learning assembly language (BBC Basic features an inline assembler). Absent are many modern high-level programming languages. That sais, RISC OS has its charms for many fun projects.

Distro



```
ubuntu@localhost:~$ snappy info
release: ubuntu-core/devel
frameworks: webdm
apps:
ubuntu@localhost:~$ snappy vers
```



Hello, World!

TEST 5
Community and support

Pidora	★★★★☆
Raspbian	★★★★★
RISC OS	★★★★☆
Snappy Ubuntu Core	★★★☆☆

Good community support is vital for getting the most out of your Raspberry Pi. It's unsurprising to learn that Raspbian, as the recommended installation, has the best community support.

All of the projects from the Raspberry Pi Foundation and many other sites (**raspberrypi.org/community**) gravitate around Raspbian.

Having said that, it's worth noting that Fedora, the flavour of Linux behind Pidora, has a large and respectable community. In particular, Linus Torvalds, the initial developer of Linux, is known to use Fedora (even if somewhat begrudgingly).

Of course, numbers and celebrity names aren't everything and the RISC OS community (**RISCosopen.org/forum**) is both dedicated and committed. It's a small clique, though, and outside of the community you'll struggle to find much online documentation. Look instead to decent books like David Bradforth's *First Steps with RISC OS 6* and Bruce Smith's *Raspberry Pi RISC OS System Programming Revealed*.

Snappy Ubuntu Core fares worst in this test with no real community to speak of, but that's only because it is such a newcomer. Give it time.

COMMUNITY
Community projects from the Foundation and its team members
MagPi
Pi Weekly
RASPBERRY JAM
COMMUNITY WEBSITES
A selection of online community projects, blogs, YouTube channels and tutorial sites
RASPI TODAY
RasPi.TV

TEST 6
Unique features

Pidora	★★☆☆☆
Raspbian	★★★★☆
RISC OS	★★★★★
Snappy Ubuntu Core	★★★★★

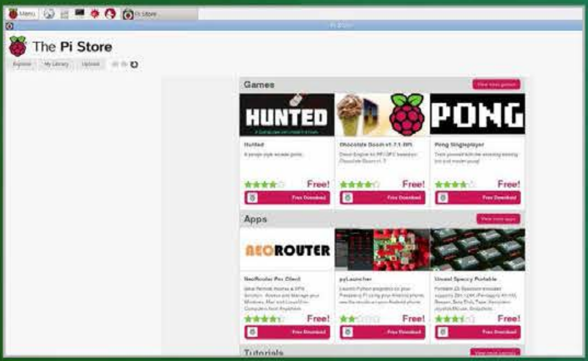
Each of the operating systems has something different to offer. Raspbian has the most all-round support, the widest range of programs, and is the most nurturing environment.

Raspbian is also noticeably faster and less error-prone than Pidora. While Pidora's lack of Pi 2 support makes any meaningful speed test impractical, we'd put our money on Raspbian and Snappy Core's armhf architecture beating Pidora in any speed test.

Pidora is worth investigating if you work on a Fedora-based main computer, own a Raspberry Pi 1, and want consistency between the two machines.

Snappy Ubuntu Core has much to offer those looking to play a part in the next wave of software development. Developing Snappy apps isn't a game for newcomers, but the transactional deployment techniques seem worth learning. And it's a good OS for setting up a web server. Whether you can make anything out of it currently depends on your skill as a programmer, but we are looking forward to seeing Snappy Ubuntu Core develop as an OS.

RISC OS is nothing if not unique. We've made much of its individuality throughout this group test. Suffice to say it's a fantastic way to broaden your horizons.



Print Me

World Map

Recycle Bin

Spec comparison

	Raspbian	RISC OS	Pidora 2014	Snappy Ubuntu Core
First release	01/06/12	05/11/12	01/10/12	02/02/15
Latest release	16/02/15	25/04/15	24/07/14	03/02/15
Default desktop	LXDE	Pinboard	Xfce	None
GPIO access	Yes	Yes	Yes	No
Pi models supported	Pi 1, Pi 2	Pi 1, Pi 2	Pi 1	Pi 2
Included with NOOBS	Yes	Yes	Yes	No
OS family	Linux	RISC OS	Linux	Linux
Base	Debian Wheezy	RISC OS	Fedora	Ubuntu
Developer	Raspbian Team	ROOL	CDOT	Canonical
Contact	raspbian.org/RaspbianTeam	riscosopen.org/content	cdot.senecacollege.ca	developer.ubuntu.com/en/snappy
Architecture	armhf	armhf	armv6hl	armhf

All Settings

All Distros

www.raspberrypi.org

Save

	Raspbian	RISC OS	Pidora	Snappy Ubuntu Core
	BEST ON TEST	RUNNER-UP		
	★★★★★	★★★★☆	★★★★☆	★★★☆☆

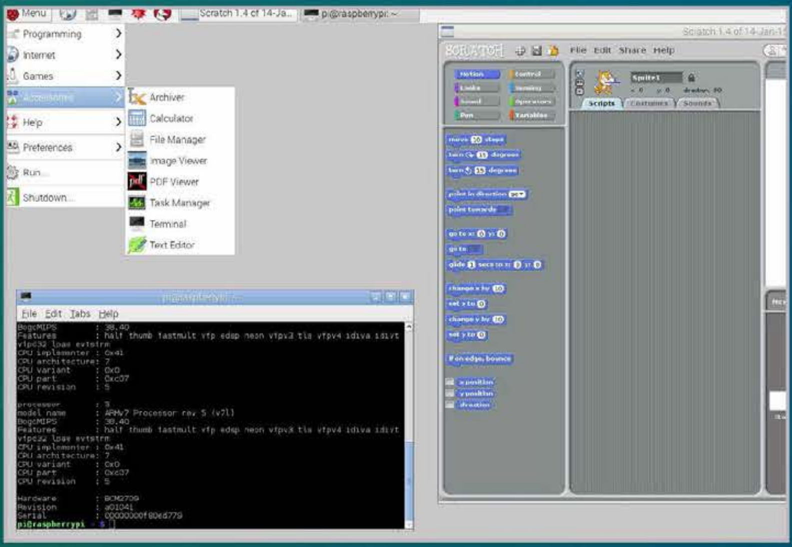
THE LAST WORD

We expected this to be a much closer race, but Raspbian has – if anything – increased its lead over other operating systems, especially Pidora.

One exception may be Ubuntu Snappy Core, which is a wholly different proposition. This innovative distro's snappy package manager and WebDM interface point to an interesting future for software installation, but without the familiar apt-get installing the usual options is much more challenging.

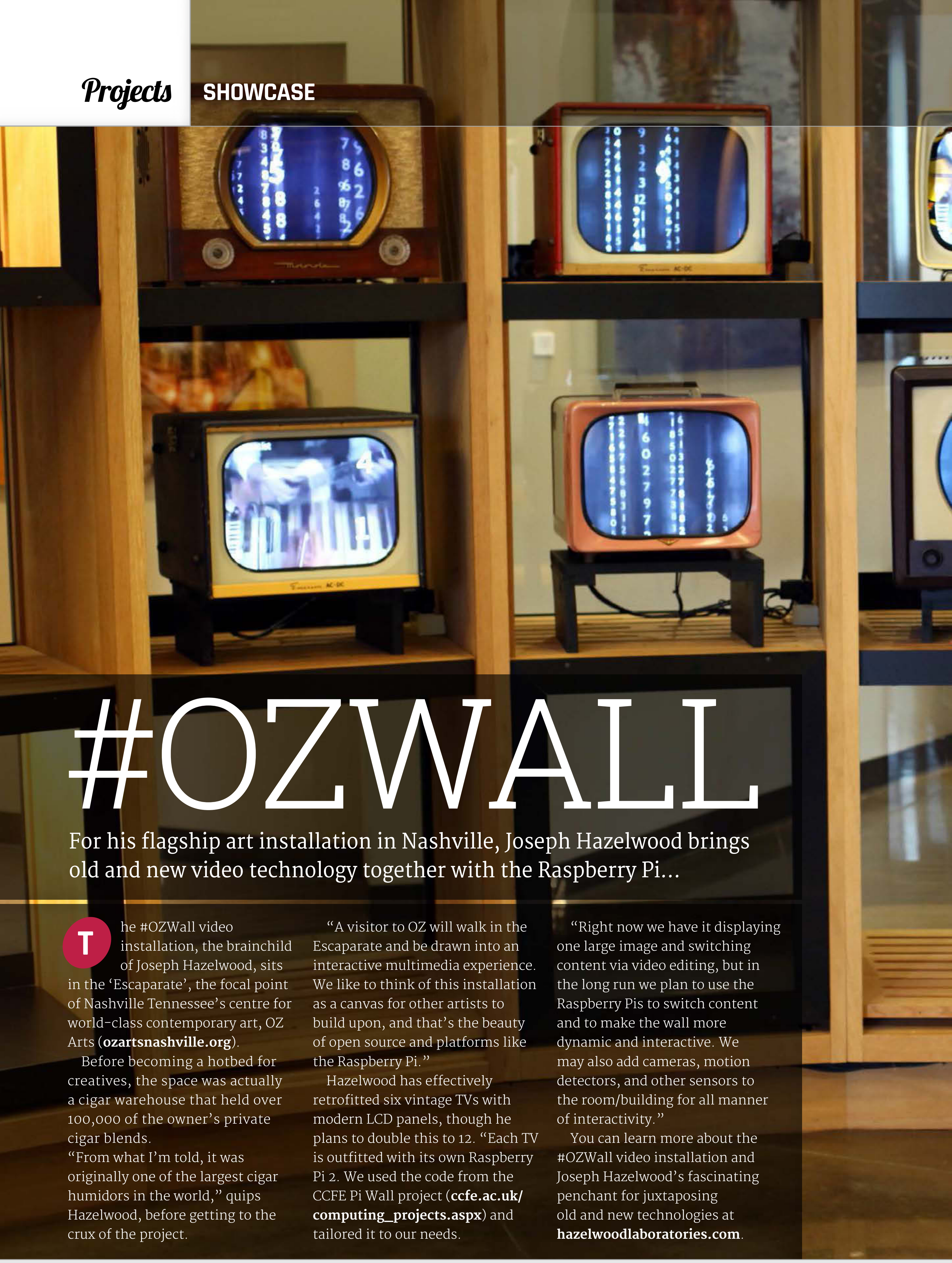
RISC OS remains great fun in a Monty Python-esque 'now for something completely different' way. It's a good tool for learning assembly, but the absence of many modern high-level programming languages prevent it from becoming more than that for many.

We've always thought Raspbian is best for newcomers, but it's hard to credibly recommend any other OS for general use. Our advice? Use all of them to find a good fit for you, but always have Raspbian on hand.



```
BogoMIPS        : 38.40
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5

processor       : 3
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 38.40
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5

Hardware        : BCM2709
Revision        : a01041
Serial          : 00000000f80ed779
pi@raspberrypi ~ $
```

Above Raspbian remains our choice for the best overall OS to install on a Raspberry Pi



#OZWALL

For his flagship art installation in Nashville, Joseph Hazelwood brings old and new video technology together with the Raspberry Pi…

The #OZWall video installation, the brainchild of Joseph Hazelwood, sits in the 'Escaparate', the focal point of Nashville Tennessee's centre for world-class contemporary art, OZ Arts (**ozartsnashville.org**).

Before becoming a hotbed for creatives, the space was actually a cigar warehouse that held over 100,000 of the owner's private cigar blends.
"From what I'm told, it was originally one of the largest cigar humidors in the world," quips Hazelwood, before getting to the crux of the project.

"A visitor to OZ will walk in the Escaparate and be drawn into an interactive multimedia experience. We like to think of this installation as a canvas for other artists to build upon, and that's the beauty of open source and platforms like the Raspberry Pi."

Hazelwood has effectively retrofitted six vintage TVs with modern LCD panels, though he plans to double this to 12. "Each TV is outfitted with its own Raspberry Pi 2. We used the code from the CCFE Pi Wall project (**ccfe.ac.uk/computing_projects.aspx**) and tailored it to our needs.

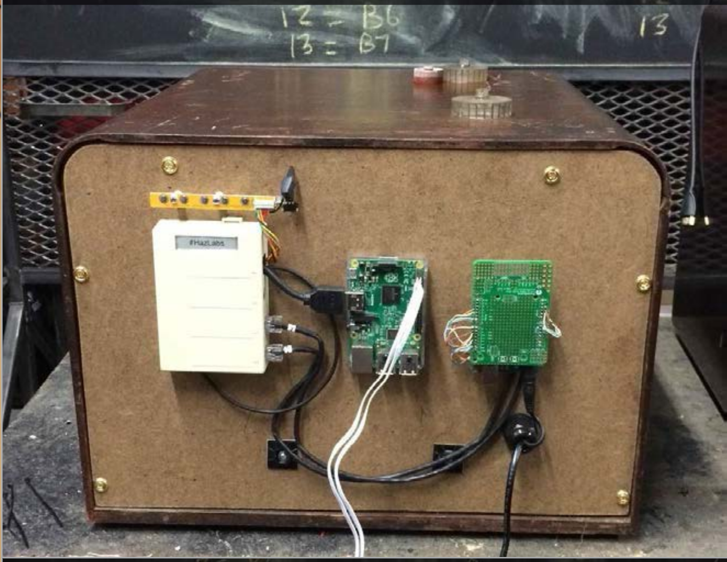
"Right now we have it displaying one large image and switching content via video editing, but in the long run we plan to use the Raspberry Pis to switch content and to make the wall more dynamic and interactive. We may also add cameras, motion detectors, and other sensors to the room/building for all manner of interactivity."

You can learn more about the #OZWall video installation and Joseph Hazelwood's fascinating penchant for juxtaposing old and new technologies at **hazelwoodlaboratories.com**.

Image: Sam Frawley, www.samfrawley.com

Below As shown from the rear, each TV in the #OZWall video installation is powered by its own Raspberry Pi 2

Below The programming aspects of the project were taken care of by developer Phillip Lehner



Below Joseph Hazelwood retrofitted and lovingly restored old TVs with modern flat screens

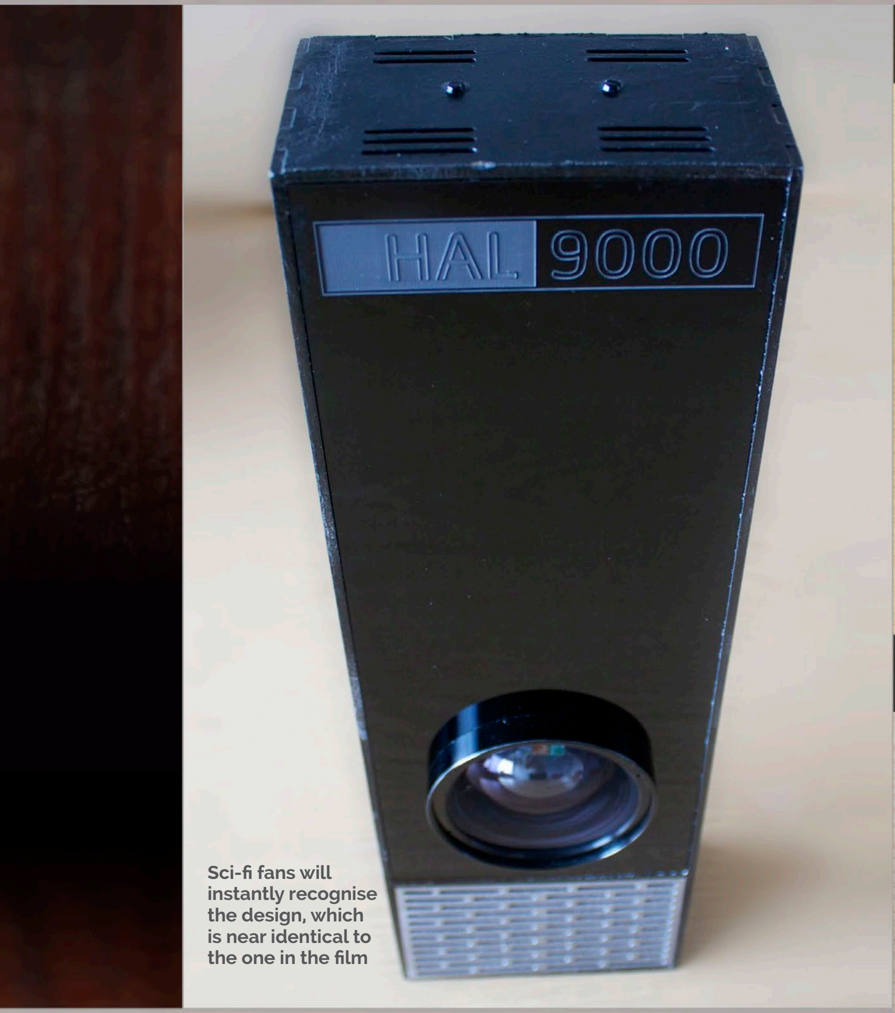
H.A.L. 9000

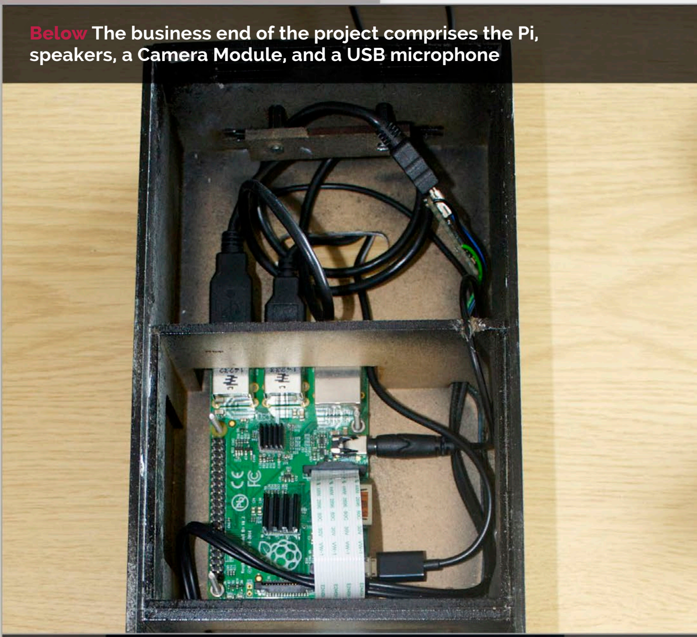
Willem Koopman opens the pod bay doors on his latest Raspberry Pi project: a fully functioning H.A.L. 9000…

Inspired by a BBC Radio 4 documentary celebrating the theatrical rerelease of Stanley Kubrik's 1968 epic *2001: A Space Odyssey* (**bbc.in/1vc3NG8**), Willem Koopman decided to build his own homage to this particular piece of cinematic history – a working model of H.A.L. 9000, the ship's computer.

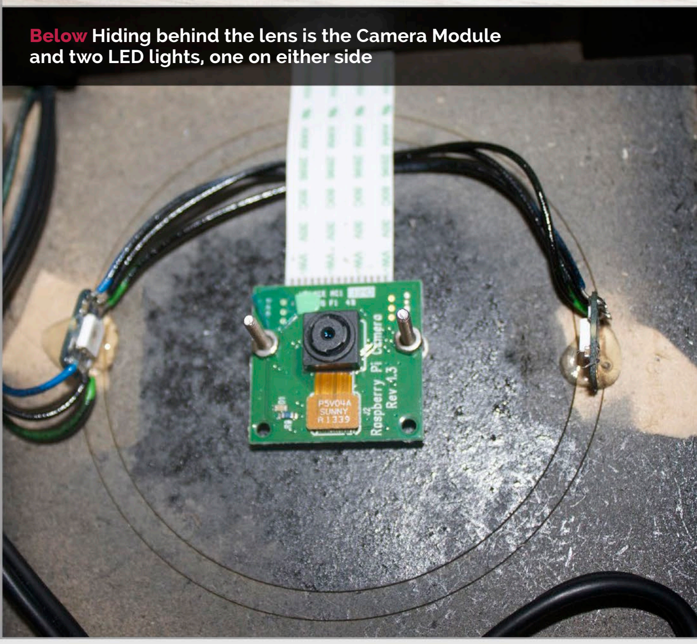
Much like the original H.A.L. who so famously lost his digital marbles part way through the film, Willem's effort still needs code to be added and polished: "The eventual plan is to have it as a work bot, so that when the website breaks or someone make a loud noise, we can take a picture for prosperity and/or automatically shame the perpetrator." But it already performs some suitably clever – and fitting – parlour tricks.

"At the moment, it runs a very lightly modified version of Jasper (**jasperproject.github.io**), which


Sci-fi fans will instantly recognise the design, which is near identical to the one in the film


Below The business end of the project comprises the Pi, speakers, a Camera Module, and a USB microphone

Below Willem used an old wide-angle lens converter to create H.A.L.'s iconic 'eye'


Below Hiding behind the lens is the Camera Module and two LED lights, one on either side

allows a basic Siri-like behaviour. You can ask it to do various things like tell the time, check emails and Facebook – basically, all the marvellous things that Jasper allows you to do. You can also ask it to open the pod bay doors."

Besides the Raspberry Pi-powered elements, there was an awful lot of work involved in getting the case to look just right – not least the eerie fisheye with its chilling red backlighting.

"I was researching how the original was made and found out that the 'eye' was a standard Nikkor 8mm fisheye. Many years ago, I bought a wide-angle lens converter (the sort used to make skateboard videos before many readers were born). I dug it out, and the plan snowballed from there." The lighting is catered for by two well-placed LEDs.

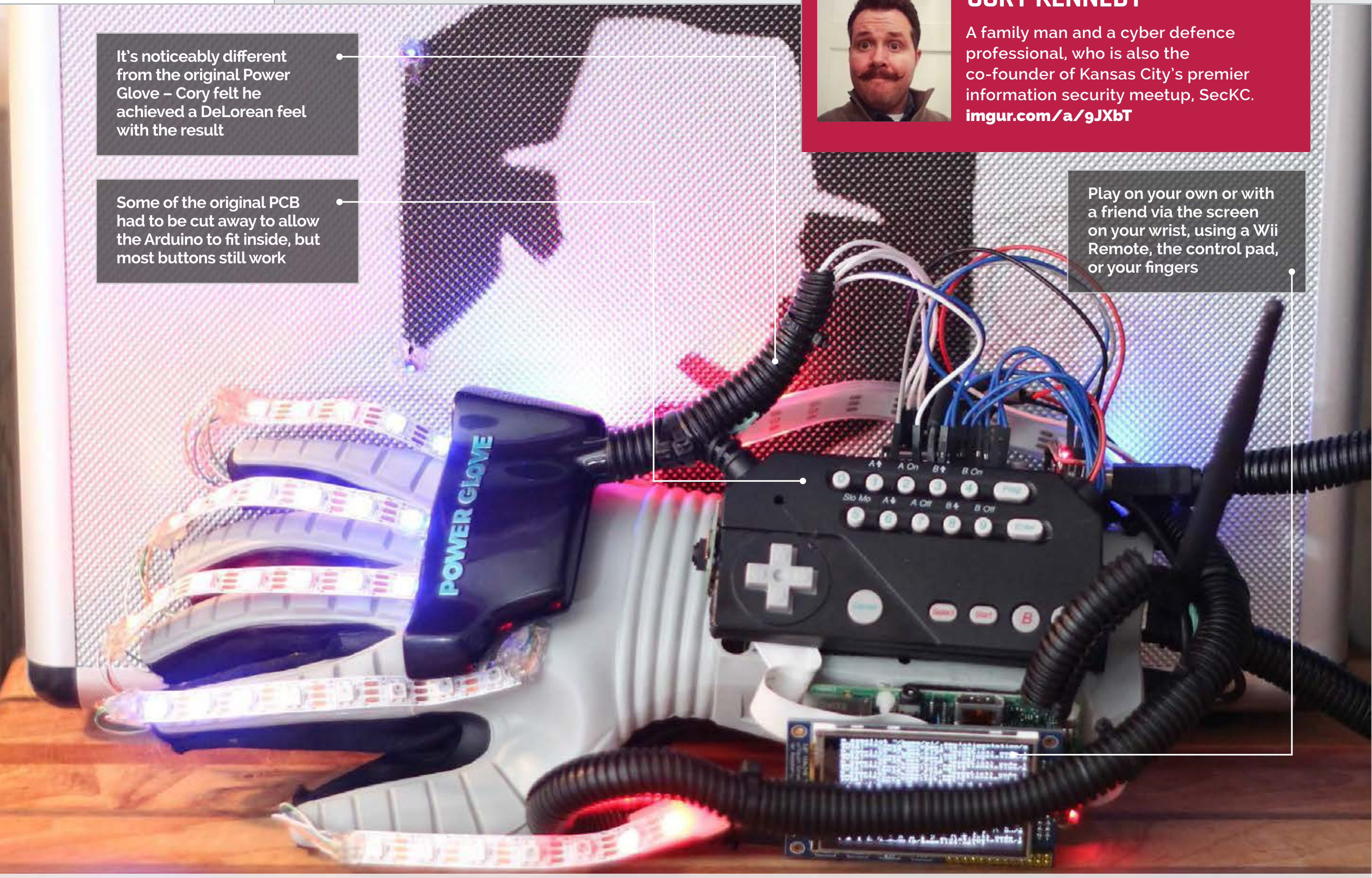
The case itself was created using laser-cut MDF. "Cheap laser cutting is the most marvellous innovation of modern times," enthuses Willem. "Just create an SVG, send it off, and a few days later precisely machined parts arrive in the post. I can design and build straight dovetail joints to within 0.2mm, all without a chisel!"

While the cutting was pain-free, the painting process has been anything but. "I spen about three weeks trying different techniques to get the glossy spray paint to actually stick." Willem says that the moral of the story is to always use a decent undercoat, but stay away from spray primers. "Lightly sand to get rid of the brush strokes. Repeat."

Watch out in the coming months for a tutorial to set up Jasper to create your own personalised Siri with the Pi, but until then you can learn more about Willem Koopman from his blog: **secretbatcave.co.uk**.

CORY KENNEDY

A family man and a cyber defence professional, who is also the co-founder of Kansas City's premier information security meetup, SecKC.
imgur.com/a/9JXbT



It's noticeably different from the original Power Glove – Cory felt he achieved a DeLorean feel with the result

Some of the original PCB had to be cut away to allow the Arduino to fit inside, but most buttons still work

Play on your own or with a friend via the screen on your wrist, using a Wii Remote, the control pad, or your fingers

Quick Facts

- It took seven days from conception to presentation
- Cory's son helped with the soldering
- The Power Glove was an official Nintendo licensed product
- It accepts two Wii Remotes as controllers
- It was created as part of a 'show-off' meetup

PWNGLOVE

The infamous NES peripheral, the Power Glove, has been the butt of many jokes and centre of a lot of nostalgia. What happens when you don't need the rose-tinted glasses any more?

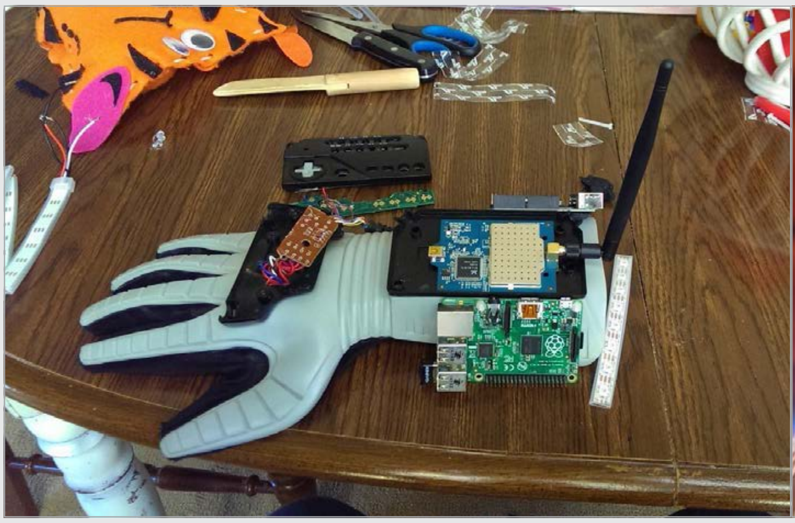
Let's face it, once you take off your rose-tinted glasses and forget the quotes and memes surrounding cult 1980s adver-film *The Wizard*, you're left with a barely functional peripheral that is very much a product of its time. The Power Glove was a novelty and, while it captured the minds of many young children, it never quite lived up to its promise at the time. Apparently, it's this exact reason that got Cory Kennedy motivated.

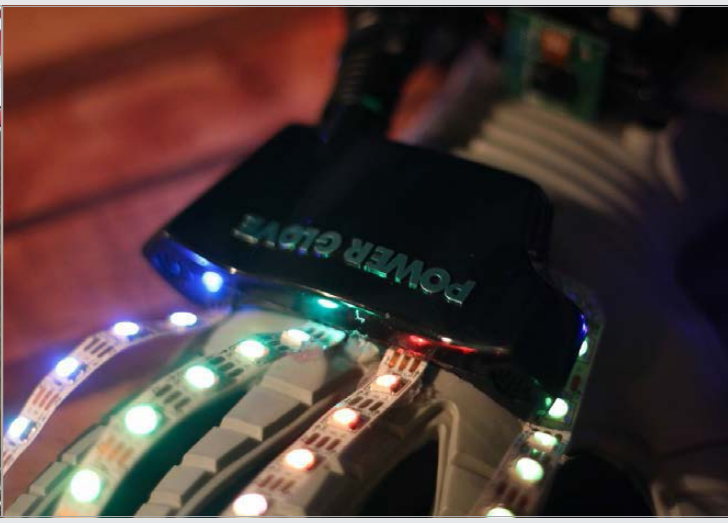
"The PwnGlove was a project rooted in the desire to build the memory my 12-year-old self was deprived of, thanks to the marketing genius behind Power Glove's capabilities," Cory explains to us. "I saw other people doing all sorts of projects with Power Gloves and I thought it would be a perfect match to pair it with retro NES gameplay."

It was originally created for Cory's information security meetup's 'Hacker Show-off Contest', where he had 15 minutes to wow the crowd with his invention. Although the focus was information security, Cory wanted to try something else:

"I wanted to do something different, mainly because where I am today professionally is rooted in tinkering – not only with computers at a young age, but also videogames… I wanted to be the kid from the 'now you're playing with power' ad. Which is exactly why I added the NeoPixel array."

As well as being a pretty good fit for the inside of the Power Glove, the Raspberry Pi came in very handy for enabling





"As well as being a pretty good fit for the inside of the Power Glove, the Raspberry Pi came in very handy for enabling Bluetooth and Arduino interaction"

Above The lights on the glove are supposed to pay homage to advertising material for the Power Glove

Above Left An Arduino can be found in the original circuit compartment, plus a screen and Raspberry Pi on the wrist

Bluetooth, Arduino interaction, and access to the actual games.

"The bodywork was tough, mainly trying to resist the urge to take away from the original look too much," Cory says, regarding actually getting everything into and around the glove. "I say that, but then I take a look at the monstrosity it has become and I have to say… It now reminds me of the *Back to the Future* DeLorean."

He did try to use as much of the original parts as possible, though. "It leverages all the original components, modified physically a bit to cut [out] the non-essential PCB, to allow for room for new components. There are four original bend sensors (thumb, index, middle, and ring), which connect to an analogue multiplexer living in the palm housing, which sends that data back to the Arduino. This is all piped over Bluetooth back to the Raspberry Pi. The wrist pad buttons are mostly intact (programming buttons aren't working), and they are set up now to do things like allowing the Konami code and switching the colour and pattern sequence on the NeoPixel array."

Cory is forever making tweaks to his PwnGlove, improving it in minor ways to solve what he sees as problems, but admits that most people don't even notice.

"To be honest, the reactions it gets from people, especially kids, certainly makes those imperfections disappear."

START PLAYING WITH POWER

>STEP-01
Konami coding

To get into the PwnGlove, you need to use the password. This is entered using a special sequence of keys: Up, Up, Down, Down, Left, Right, Left, Right, B, A, Start. 30 extra lives await.

>STEP-02
Choose your game

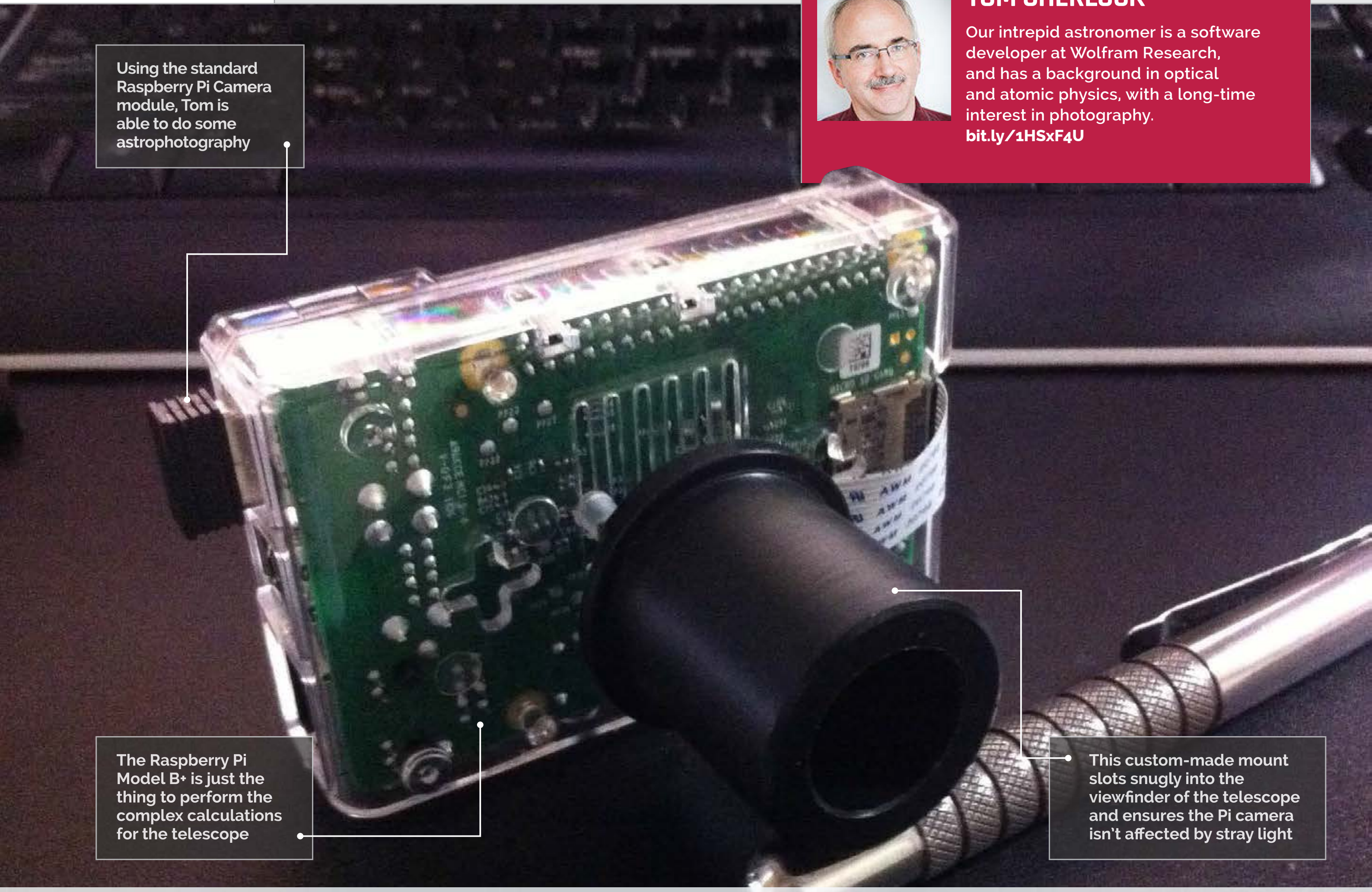
The RetroPie interface comes up, enabling you to select any compatible game from your library. It's best to use the buttons on the pad to select one.

>STEP-03
Team up with a bad dude

Throw a Wii Remote to a friend and you can then rescue the president, save the princess, fight the Red Falcons, or swear at Battletoads together.

TOM SHERLOCK

Our intrepid astronomer is a software developer at Wolfram Research, and has a background in optical and atomic physics, with a long-time interest in photography.
bit.ly/1HSxF4U


Using the standard Raspberry Pi Camera module, Tom is able to do some astrophotography
The Raspberry Pi Model B+ is just the thing to perform the complex calculations for the telescope
This custom-made mount slots snugly into the viewfinder of the telescope and ensures the Pi camera isn't affected by stray light

MATHEMATIC TELESCOPE

The Raspberry Pi has access to a huge amount of mathematical power and knowledge, thanks to Wolfram's Mathematica – here's how one man uses it for stargazing...

Quick Facts

- The whole thing only took a few hours to make
- You don't need a very expensive telescope for this project
- You can see the camera preview on a remote desktop
- Tom wants to have Wolfram control the motors next
- The whole Pi setup weighs less than a normal CCD camera

Space. The final frontier. This is the adventure of Tom Sherlock. His mission: to explore the stars from his backyard, using a telescope to photograph stunning cosmic vistas. And he's managed to make the process much better and easier for himself by using a Raspberry Pi.

He needs more than a Pi, though – it's good, but it's not good enough to photograph close-ups of the Moon. In this particular instance, the Raspberry Pi is connected to a telescope using a custom-made mount to slot it into the eyepiece, so that the Raspberry Pi Camera module can take these excellent photos. On its own, it's impressive enough, but Tom has added a few more tricks to make it a truly unique Pi project – namely, he can control it with Wolfram Language and Mathematica.

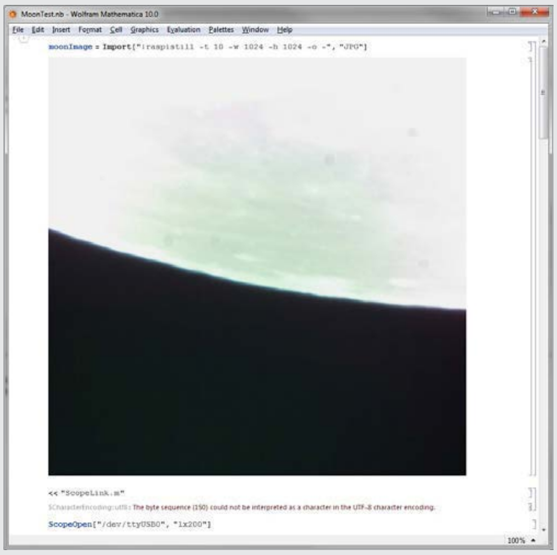
"Mathematica combines a powerful language, device control, and image processing facilities with a great deal of built-in data," Tom explains to us. "This includes astronomical data, so it was not hard to see the advantage of having one piece of software which could manage all these tasks in a unified, scriptable way."

Tom actually works at Wolfram on Mathematica, so is well versed in the ways of the language and Mathematica itself. He's also an amateur astronomer, which is where the idea came from in the first place:

"Over the past decades and especially in recent years, amateur astronomy has become more and more dependent on computers for various tasks, like planning observations, pointing and driving the telescope, capturing and processing images and data, and finally sharing images and data with other astronomers.

"Typically, you have to run several different packages on several different computers to accomplish these tasks. For example, you might plan observations with one package, control the telescope with another package, capture images with still another package, and then process the images with one or more additional packages, sometimes involving specialised plug-ins."

Tom was able to solve all these complications thanks solely to the Pi having Wolfram available with Raspbian. Just having a Pi and some coding skills is only one part of the challenge, however, as telescope selection is also important.


MoonTest.nb - Wolfram Mathematica 10.0
File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
100%

```
moonImage = Import["!raspistill -t 10 -w 1024 -h 1024 -o -", "JPG"]

<< "ScopeLink.m"
$CharacterEncoding::utf8: The byte sequence {150} could not be interpreted as a character in the UTF-8 character encoding.
ScopeOpen["/dev/ttyUSB0", "1x200"]
```

Above The Pi camera might not be of the highest fidelity, but it can take stunning shots when used correctly

"It has to be controllable from a computer," Tom elaborates. "Many scopes these days have computerised mounts, and the computer built into the mount will let you select any object in the mount's internal database and slew the scope to that object. Generally, these mounts will also allow you to connect them to an external computer and then, using the correct protocol, specify an arbitrary set of coordinates. Since Mathematica has the coordinates of hundreds of thousands of different objects, you can easily look things up and have the scope slew to that object."

> On its own, it's impressive enough, but Tom has added a few more tricks to make it a truly unique Pi project

It all sounds good in theory, but just how does it work in practice?

"Remarkably well, considering how much the Pi is doing. The main problems I found were more details than anything else. It takes a bit of time to look up astronomical coordinates on-the-fly, so I found it was better if I did that ahead of time instead of 'in the field'."

Above Just a Raspberry Pi Model B+ is needed to perform the complex calculations needed for the telescope

SEEING STARS

>STEP-01
The setup, according to Tom

Set up the scope and mount, and perform the mount's alignment procedures so that its computer knows how it is oriented, with respect to the Earth and the celestial sphere.

>STEP-02
Attaching the Raspberry Pi

Once the telescope is sorted, then you just need to attach the Raspberry Pi. Tom uses a serial-to-USB adaptor to do this, then boots up the Pi.

>STEP-03
Do the maths

Once the Pi is booted up and in Mathematica, you can start issuing commands using the Wolfram Language to control the telescope – this will change depending on what mount you're using.

EVERYDAY ENGINEERING PART 3

SIMON MONK

Simon Monk is the author of the *Raspberry Pi Cookbook* and *Programming Raspberry Pi: Getting Started with Python*, among others.

simonmonk.org
monkmakes.com

BUILD YOUR OWN ALARM CLOCK

You'll Need

- Half-size breadboard
- Adafruit LED backpack (1270 or 878)
- Standard servo motor
- Tactile push switch
- Jumper wires
- 1k ohm resistor
- DC barrel (CPC: CN19646)
- 6V DC 1A power supply (this just supplies power to the motor)
- Metal lid or baking tray
- Block of wood
- A fork!

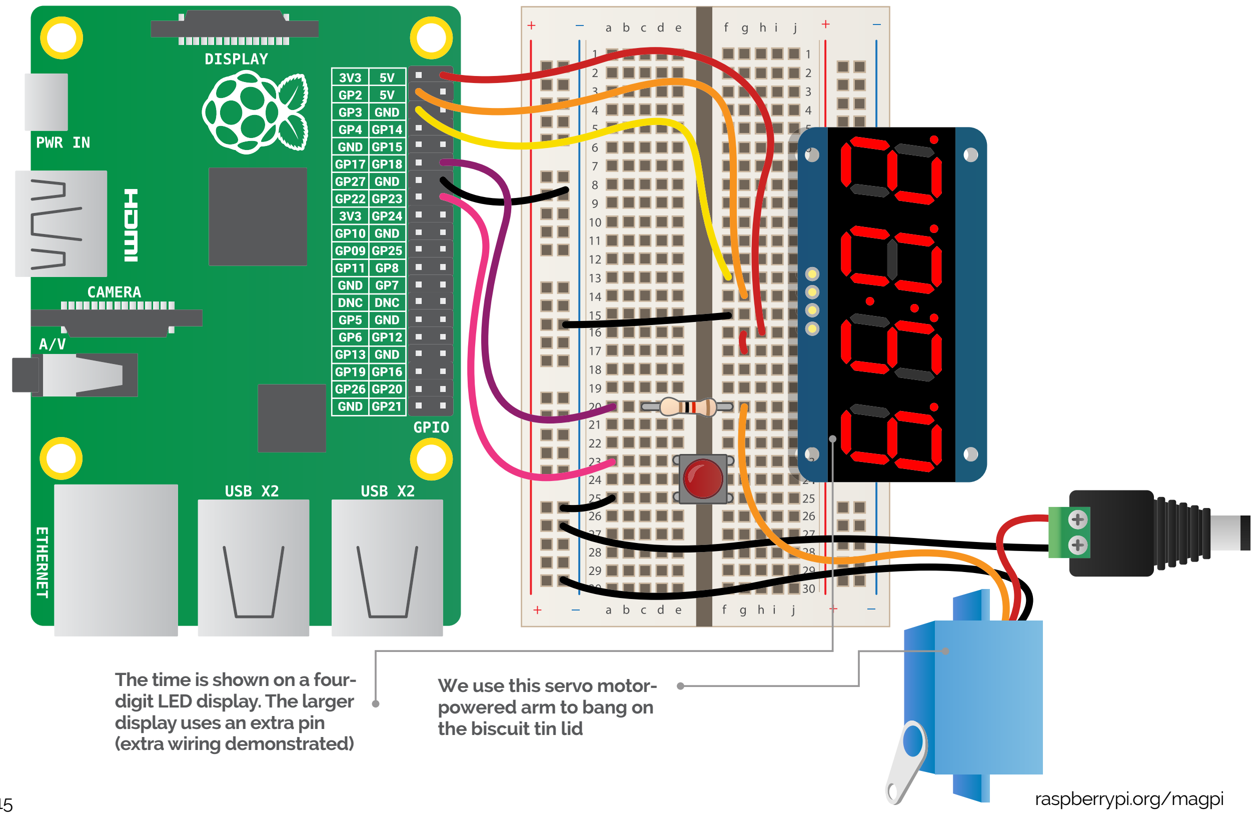
Solve real-world electronic and engineering problems with your Raspberry Pi and the help of renowned technology hacker and author, **Simon Monk**

Everyone with teenage kids knows that they can require considerable encouragement to be extracted from their beds in the morning. This helpful project is an automated variation on banging a cooking pot with a metal implement. The project also has a nice bright 7-segment LED display and an 'Off' button that allows the alarm to be cancelled. You can see a video of the project in action at **youtu.be/zold22xfCqk**.

The alarm time is set using a web interface for the project, so you can even adjust the alarm time for the person to be woken remotely, over your local network or even the internet.

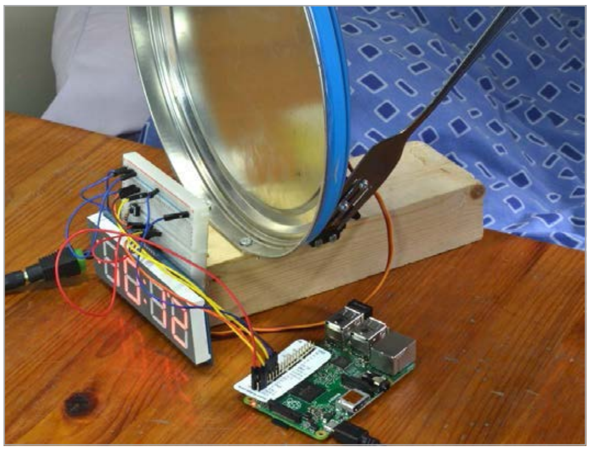
As you'll see from the list of required components, this project uses a breadboard and an Adafruit LED display module. The latter is supplied as a kit which does require some simple soldering.

This project also requires a little mechanical construction. You will need a drill and assorted screws to attach the lid and also to fix the fork to the servo arm. The fork also needs to be hammered flat. So make sure that you have



The time is shown on a four-digit LED display. The larger display uses an extra pin (extra wiring demonstrated)

We use this servo motor-powered arm to bang on the biscuit tin lid



Above BANG! BANG! BANG! BANG!

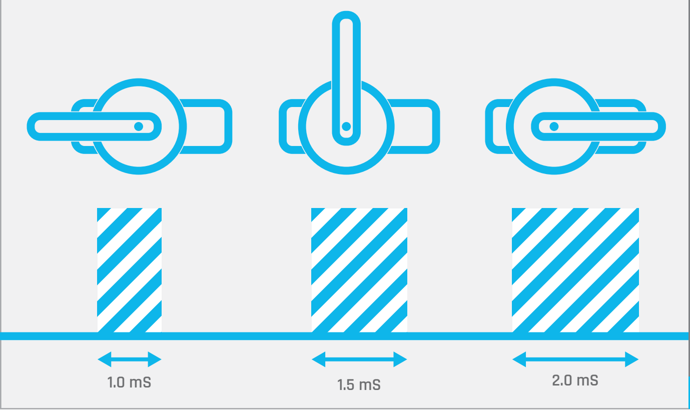
adult supervision (assuming you need it), and eye protectors when hammering and drilling.

The breadboard, jumper wires, switch, and resistor are probably best bought as an electronics starter kit. The Monk Makes Electronic Starter Kit for Raspberry Pi includes these parts. Most starter kits for the Raspberry Pi will include the breadboard, jumper wires, and some resistors.

Servo motors

The key to this project is the servo motor that repeatedly moves to bang the metal lid. Servo motors are different from more conventional motors in that they are not free to rotate round and round. In fact, they can only normally rotate about 180 degrees.

Servo motors have three leads: two power leads (ground and 6V) and a control lead. The control lead is fed a series of pulses of varying length. The length of the pulses will determine the angle to which the servo motor is set. If the pulses are around 1 millisecond, the arm of the servo will be at one end of its range (let's call that 0 degrees). A pulse length of 1.5 milliseconds will position the servo arm right in the middle of its range (90 degrees) and a 2 milliseconds pulse will put the arm at the far end of its range (180 degrees). In practice, servos often don't do the full 180 degrees and the range is sometimes more like 0 to 170 degrees.



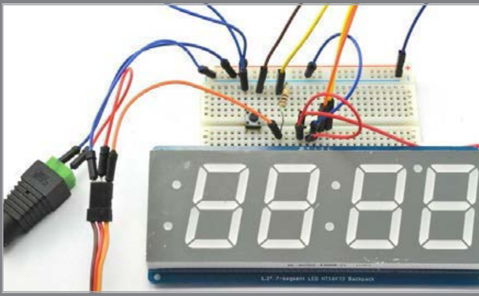
BUILDING THE ALARM CLOCK

Follow the annotated breadboard diagram, taking care to ensure that all the components and jumper wires fit into the correct rows on the breadboard. The difference between the two Adafruit displays listed in the 'You'll Need' box is that one is a lot bigger than the other. The larger display actually has two positive power pins, which is why there are two rows on the breadboard connected to 5V for the display.

>STEP-01

Build the breadboard

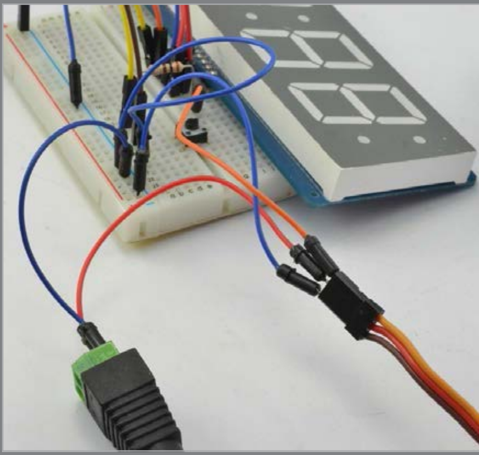
In this project, the breadboard is mostly used as a way of bringing together the connections from the servo motor, power supply, and display. Start by fitting the display onto the breadboard, and then the resistor switch and leads to connect to the Raspberry Pi.



>STEP-02

Connect the breadboard

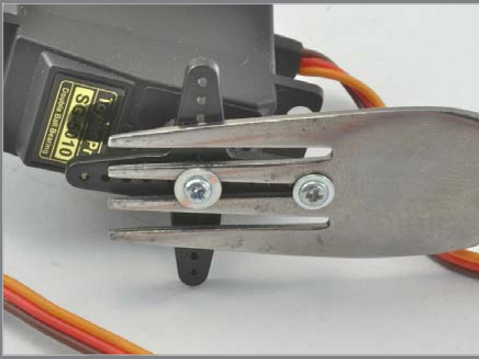
Attach male-to-male jumper wires to the screw terminals of the barrel jack adaptor. Use red and black leads and make sure that you attach the red lead to the screw terminal marked '+'. The red lead plugs directly into the connector on the servo with the servo's red lead. The black lead from the barrel jack goes to the ground bus. The brown servo lead is the ground lead and the orange lead the control lead. Attach the female-to-male leads from the breadboard to your Raspberry Pi.



>STEP-03

Preparing the hardware

We would advise coming back to these final two steps after you have tested out the project software, as lining up the servo arm will depend very much on the exact position of the servo and lid. Hammer the fork flat. Be careful because it may fly up and hit you when you whack it. The fork can then be attached to the servo arm using screws.



>STEP-04

Fix the hardware to a base

Find yourself a small block of wood that can act as a base to hold the servo and the tin. Fit the lip of the tin to one end of the block and the servo to the side of the block, so that the fork will just reach the tin to hit it. This is best done with the software running and the servo moving.

> The program will start a web server on your Pi

Now that the hardware side of the project is complete, we just need to get the software running. The program is written in Python and uses a library called Bottle. If you made the previous project in this series (a web door lock), you should already have Bottle installed. If not, to install Bottle, make sure that your Raspberry Pi has an internet connection and then run the commands:

```
sudo apt-get update
sudo apt-get install python-bottle
```

You can download the program from your Raspberry Pi command line, using the command:

```
git clone https://github.com/simonmonk/
pi_magazine.git
```

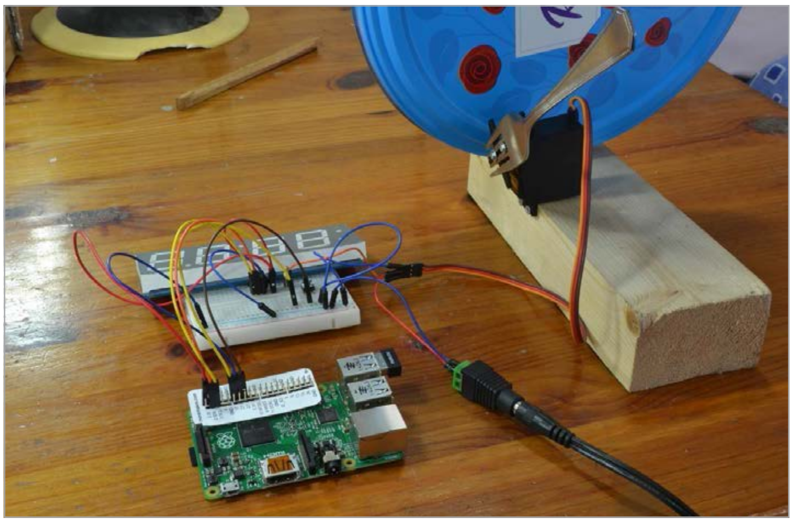
Adafruit has a library of Python code for using its displays (**github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code**). The files needed for this project are copied into the code you just downloaded, so you don't need to install this again. You may need to set up your Pi to be able to use the I²C serial communications mechanism used by the Adafruit display. If you get error messages relating to the display when you try to run the program, follow the instructions here to enable I²C: **raspberrypi-spy.co.uk/2014/11/enabling-the-i2c-interface-on-the-raspberry-pi**.

Before you run the program, open it in an editor such as nano, and change the value of the **IP_ADDRESS** variable to match the IP address of your Raspberry Pi. You will also probably need to change the variables **START_ANGLE** and **END_ANGLE**, which set the range of travel of the servo when it's doing its banging.

To run the program, change to the directory where the code for this project lives and then run the program, using the commands below:

```
cd /home/pi/pi_magazine/03_alarm_clock
sudo python alarm_clock_server.py
```

This will start a web server running on your Raspberry Pi. You can check this by opening a browser window on another computer or your phone and entering the IP address of your Pi. For our Pi, it was 192.168.1.14. If all is well, a webpage will be displayed inviting you to enter a time at which the alarm should be sounded.


Below The complete project including sweet tin lid and clangy fork!

How the code works

Alarm_clock_server.py is a pretty heavily commented Python program. You will probably find it handy to have the code open while we go through it.

The program starts by importing the **Bottle**, **RPi.GPIO**, **datetime**, **time** and **Adafruit_7Segment** libraries it needs. Then there are some constants that you may need to tweak. You have already met **START_ANGLE** and **END_ANGLE**. The **DELAY_IN** and **DELAY_OUT** variables may need altering in your setup to give the servo longer to reach the correct position.

Two global variables are used: **alarm_on** and **alarm_time**. The variable **alarm_on** is a true/false value that records whether the alarm is enabled or not. Pressing the Cancel button will set this to False to stop the banging. The variable **alarm_time** contains the time for the next alarm to be triggered. This is in the format of two numbers for the hour (24-hour clock) followed by a colon, followed by two digits for the minute. Note: you must enter the colon when you type in the alarm time on the web form.

Three utility functions are then defined, which are used by later functions. The **set_angle** function adjusts the pulse width for the PWM signal to set the servo arm to the angle specified. **bong** causes the servo to make one banging motion, and **update_display** updates the Adafruit LED display by splitting the time up into separate digits and writing them to the appropriate digits of the display.

The next section of the code supplies the web interface. Bottle uses the **@route** directive to indicate that the functions that immediately follow it are handlers for web requests. So the default route page of '/' is handled by the function called **index**. The **index** function uses Bottle's templating mechanism to return the HTML contained in the template file, **home.tpl**. You will find all the HTML templates used in the project contained in the code directory for it.

The homepage contains a form with an alarm time entry box and a Set button.

The second handler is for the '/setconfirmation' page, where the form from the index page is posted. This handler function extracts the alarm time from the web request and assigns it to the global variable. It then sets **alarm_on** to be True.

Since this Python program is mostly acting as a web server, the code to check the alarm time and to update the LED display all happens in a separate thread of execution. This thread is started by the command **start_new_thread**, which causes the **update** function to, in effect, run independently of the rest of the program. The **update** function checks the current time against the alarm time and if they are the same, it calls **bong** to move the servo. It also checks for the Cancel button being pressed and updates the LED display.

The last few lines of the program start up the web server on port 80. The **try** / **finally** clause is used to set the GPIO pins back to inputs when the program is quit using **CTRL+C**.

Using your alarm clock

You can use your web browser on a computer, or your smartphone if it is connected to your Wi-Fi network. For this reason, the Raspberry Pi needs to have a network connection. A network connection also allows the Pi to synchronise with an internet time server.

If you find that your Raspberry Pi is an hour out, then you may need to use the **raspi-config** tool to set the time zone.

If you enjoy a bit of webpage design, then the first thing you will probably want to do is to add some styling to the webpages. The place to do this is in the template files. These are:

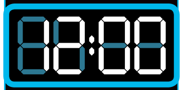
home.tpl - this is the homepage containing the password form.

setconfirmation.tpl - the page opened to confirm the alarm time setting.

This is a fun kind of project that lends itself to experimentation and improvement, so dive into the code and see what else you can make it do.

NEXT MONTH

In the next project in this series, we will continue with the theme of time and hack an analogue wall clock. We can then make it run super-fast or slow or even use it to indicate things other than time, all using your Raspberry Pi.



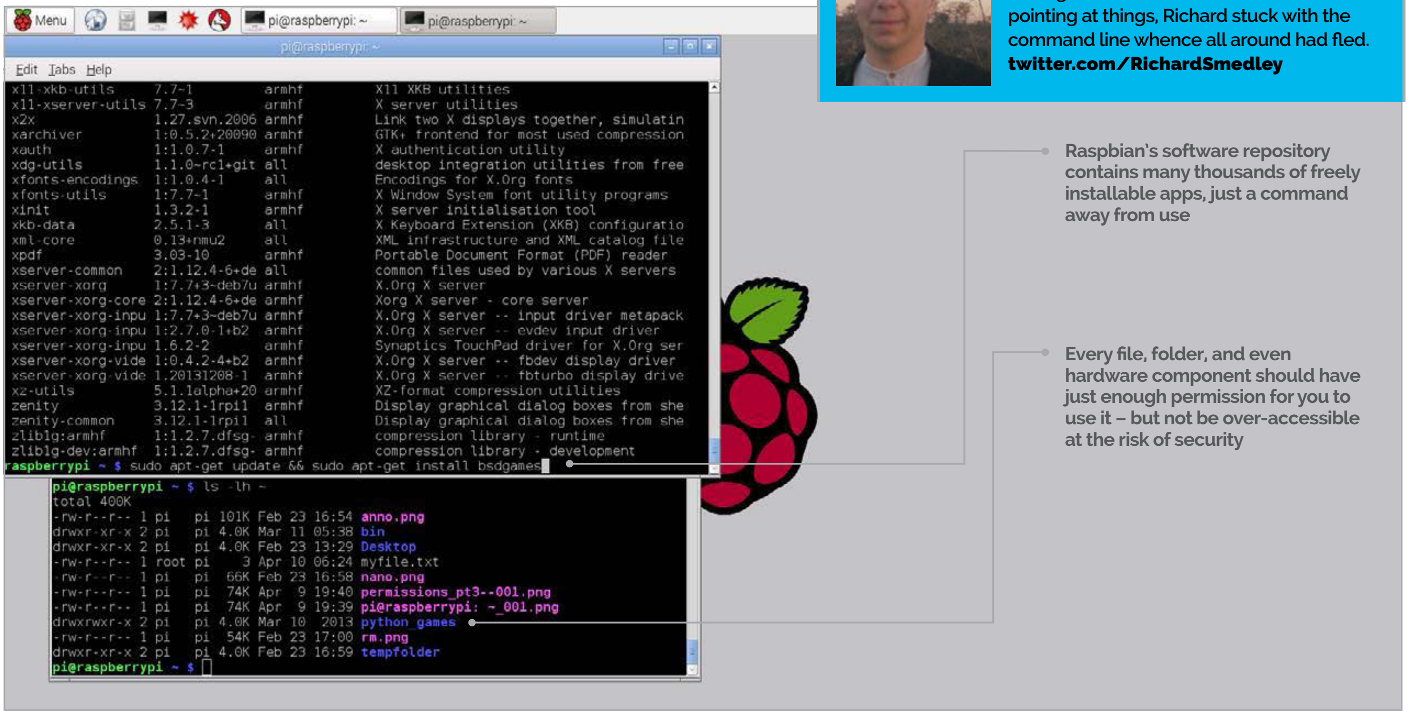
Alarm_clock_server.py

Language
>PYTHON

```
from bottle import route, run, template, request
import thread, time, datetime
import RPi.GPIO as GPIO
from Adafruit_7Segment import SevenSegment

IP_ADDRESS = '192.168.1.13' # of your Pi
START_ANGLE = 50 # start angle before moving to bong the lid
END_ANGLE = 73 # end angle this should just be touching the lid
DELAY_IN = 0.5 # delay to allow servo to get to start position
DELAY_OUT = 0.1 # delay at point of bong before returning to start
SERVO_PIN = 18 # control pin of servo
BUTTON_PIN = 23 # pin connected to Cancel button

segment = SevenSegment(address=0x70)

GPIO.setmode(GPIO.BCM)
GPIO.setup(SERVO_PIN, GPIO.OUT)
pwm = GPIO.PWM(SERVO_PIN, 100) # start PWM at 100 Hz
pwm.start(0)
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

alarm_on = True
alarm_time = ''

def set_angle(angle):
    if angle < 1.0:
        pwm.ChangeDutyCycle(0) # stop servo jitter when not
banging
    else:
        duty = float(angle) / 10.0 + 2.5
        pwm.ChangeDutyCycle(duty)

def bong():
    set_angle(START_ANGLE)
    time.sleep(DELAY_IN)
    set_angle(END_ANGLE)
    time.sleep(DELAY_OUT)

def update_display():
    now = datetime.datetime.now()
    hour = now.hour
    minute = now.minute
    second = now.second
    segment.writeDigit(0, int(hour / 10))      # Tens
    segment.writeDigit(1, hour % 10)           # Ones
    segment.writeDigit(3, int(minute / 10))    # Tens
    segment.writeDigit(4, minute % 10)         # Ones
    segment.setColon(second % 2) # Toggle colon at 1Hz

@route('/')
def index(name='time'):
    return template('home.tpl')

@route('/setconfirmation', method='POST')
def confirm():
    global alarm_time, alarm_on
    alarm_time = request.POST.get('alarm_time', '')
    alarm_on = True
    return template('setconfirmation.tpl', t=alarm_time)

def update(thread_name):
    global alarm_on, alarm_time
    while True:
        current_time = time.strftime("%H:%M")
        if current_time == alarm_time and alarm_on:
            bong()
        if GPIO.input(BUTTON_PIN) == False:
            alarm_on = False
            set_angle(0)
        update_display()
        time.sleep(0.1) # allow other threads to run
thread.start_new_thread(update, ("update_thread",))

try:
    run(host=IP_ADDRESS, port=80)
finally:
    print('Cleaning up GPIO')
    GPIO.cleanup()
```

RICHARD SMEDLEY

Having found words often better than pointing at things, Richard stuck with the command line whence all around had fled.
twitter.com/RichardSmedley



```
x11-xkb-utils      7.7-1          armhf        X11 XKB utilities
x11-xserver-utils 7.7-3           armhf        X server utilities
x2x               1.27.svn.2006 armhf          Link two X displays together, simulatin
xarchiver         1:0.5.2+20090 armhf          GTK+ frontend for most used compression
xauth             1:1.0.7-1      armhf         X authentication utility
xdg-utils         1.1.0~rc1+git all            desktop integration utilities from free
xfonts-encodings  1:1.0.4-1      all           Encodings for X.Org fonts
xfonts-utils      1:7.7-1        armhf         X Window System font utility programs
xinit             1.3.2-1        armhf         X server initialisation tool
xkb-data          2.5.1-3        all           X Keyboard Extension (XKB) configuratio
xml-core          0.13+nmu2      all           XML infrastructure and XML catalog file
xpdf              3.03-10        armhf         Portable Document Format (PDF) reader
xserver-common    2:1.12.4-6+de all            common files used by various X servers
xserver-xorg      1:7.7+3~deb7u armhf          X.Org X server
xserver-xorg-core 2:1.12.4-6+de armhf          Xorg X server - core server
xserver-xorg-inpu 1:7.7+3~deb7u armhf          X.Org X server -- input driver metapack
xserver-xorg-inpu 1:2.7.0-1+b2  armhf          X.Org X server -- evdev input driver
xserver-xorg-inpu 1.6.2-2        armhf         Synaptics TouchPad driver for X.Org ser
xserver-xorg-vide 1:0.4.2-4+b2  armhf          X.Org X server -- fbdev display driver
xserver-xorg-vide 1.20131208-1  armhf          X.Org X server -- fbturbo display drive
xz-utils          5.1.1alpha+20 armhf          XZ-format compression utilities
zenity            3.12.1-1rpi1  armhf          Display graphical dialog boxes from she
zenity-common     3.12.1-1rpi1  all            Display graphical dialog boxes from she
zlib1g:armhf      1:1.2.7.dfsg- armhf          compression library - runtime
zlib1g-dev:armhf  1:1.2.7.dfsg- armhf          compression library - development
raspberrypi ~ $ sudo apt-get update && sudo apt-get install bsdgames
```

```
pi@raspberrypi ~ $ ls -lh ~
total 400K
-rw-r--r-- 1 pi   pi 101K Feb 23 16:54 anno.png
drwxr-xr-x 2 pi   pi 4.0K Mar 11 05:38 bin
drwxr-xr-x 2 pi   pi 4.0K Feb 23 13:29 Desktop
-rw-r--r-- 1 root pi    3 Apr 10 06:24 myfile.txt
-rw-r--r-- 1 pi   pi  66K Feb 23 16:58 nano.png
-rw-r--r-- 1 pi   pi  74K Apr  9 19:40 permissions_pt3--001.png
-rw-r--r-- 1 pi   pi  74K Apr  9 19:39 pi@raspberrypi: ~_001.png
drwxrwxr-x 2 pi   pi 4.0K Mar 10  2013 python_games
-rw-r--r-- 1 pi   pi  54K Feb 23 17:00 rm.png
drwxr-xr-x 2 pi   pi 4.0K Feb 23 16:59 tempfolder
pi@raspberrypi ~ $
```

Raspbian's software repository contains many thousands of freely installable apps, just a command away from use

Every file, folder, and even hardware component should have just enough permission for you to use it – but not be over-accessible at the risk of security

COMMAND LINE Pi PART 3: PERMISSION TO INSTALL

Richard Smedley presents your cut-out-and-keep guide to using the command line on the Raspberry Pi. In part 3, we look at Raspbian's efficient system for installing and updating software, among other things…

You'll Need

- Raspbian **raspberrypi.org/downloads** – though most of the tutorial series will work with the command line running the Linux default Bash shell on any GNU/Linux PC.

SHARED RESPONSIBILITY

If you share your Pi, it's worth reading up on sudo – and the visudo command – to find how to give limited but still useful admin privileges to the other users.

Installing software should be easy, but behind every piece of software is an evolving set of dependencies that also need installing and updating. Keeping them separate reduces unnecessary bloat and duplication, but adds the potential for bugs, missing files, and even totally unresolvable clashes.

Fortunately, Debian GNU/Linux cracked the problem back in the 1990s with the Debian Package Management system and the Advanced Package Tool (APT); Debian-based systems, like Ubuntu and the Pi's Raspbian, inherit all of the benefits. Here we'll show you the basics you need to know to install new software and keep your system up to date from the command line, and then look at the not entirely unrelated field of file ownership and permissions.

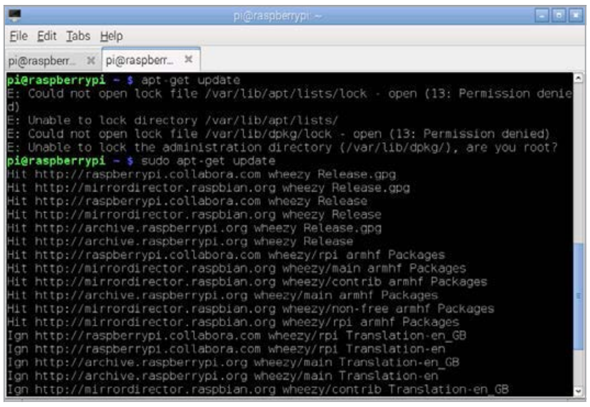
Using the **apt** command to update your system's list of installable software should be as simple as issuing the command like so: **apt-get update** – but try this logged in as user pi and you'll just get error messages. The reason for this is that changing system software on a GNU/Linux (or any type of Unix) system is a task restricted to those with administrative permissions: the godlike superuser, or admin, also known as root.

Pseudo root, su do

We'll get onto permissions properly a bit later, but for now you'll be pleased to know that you can fake it, using the **sudo** command. **sudo** potentially offers a fine-grained choice of permissions for users and groups to access portions of the admin user's powers. However, on the Pi, Raspbian assumes, quite rightly, that the default user will be someone just wanting to get on with things, and **sudo** in front of a command will pretty much let you do anything; you have been warned.

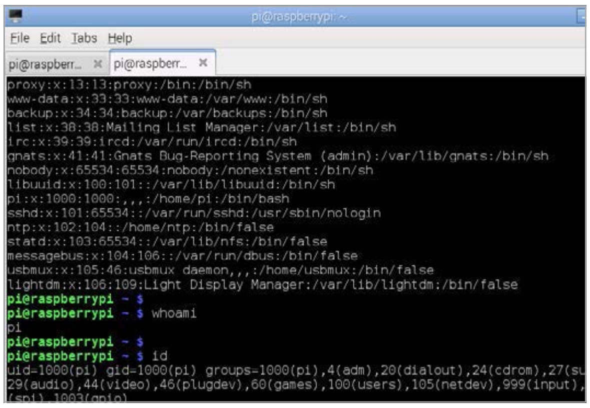
The following two commands will update Raspbian's installed software (**Fig 1**):

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



```
pi@raspberrypi ~ $ apt-get update
E: Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denie
d)
E: Unable to lock directory /var/lib/apt/lists/
E: Could not open lock file /var/lib/dpkg/lock - open (13: Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root?
pi@raspberrypi ~ $ sudo apt-get update
Hit http://raspberrypi.collabora.com wheezy Release.gpg
Hit http://mirrordirector.raspbian.org wheezy Release.gpg
Hit http://raspberrypi.collabora.com wheezy Release
Hit http://mirrordirector.raspbian.org wheezy Release
Hit http://archive.raspberrypi.org wheezy Release.gpg
Hit http://archive.raspberrypi.org wheezy Release
Hit http://raspberrypi.collabora.com wheezy/rpi armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/main armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/contrib armhf Packages
Hit http://archive.raspberrypi.org wheezy/main armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/non-free armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/rpi armhf Packages
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en_GB
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en
Ign http://archive.raspberrypi.org wheezy/main Translation-en_GB
Ign http://archive.raspberrypi.org wheezy/main Translation-en
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en_GB
```

Fig 1 Raspbian updates its listing of thousands of available apps – providing you give it admin permissions



```
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
pi:x:1000:1000:,,,:/home/pi:/bin/bash
sshd:x:101:65534::/var/run/sshd:/usr/sbin/nologin
ntp:x:102:104::/home/ntp:/bin/false
statd:x:103:65534::/var/lib/nfs:/bin/false
messagebus:x:104:106::/var/run/dbus:/bin/false
usbmux:x:105:46:usbmux daemon,,,:/home/usbmux:/bin/false
lightdm:x:106:109:Light Display Manager:/var/lib/lightdm:/bin/false
pi@raspberrypi ~ $
pi@raspberrypi ~ $ whoami
pi
pi@raspberrypi ~ $
pi@raspberrypi ~ $ id
uid=1000(pi) gid=1000(pi) groups=1000(pi),4(adm),20(dialout),24(cdrom),27(su
29(audio),44(video),46(plugdev),60(games),100(users),105(netdev),999(input),
(spi),1003(gpio)
```

Fig 2 Raspbian tells you who you are, and what group access you have, for permission to use and alter files and devices

You can wait for one to finish, check everything is okay, then issue the other command – or you can save waiting and enter both together with:

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

The **&&** is a Boolean (logical) AND – so if the first command doesn't run properly, the second one will not run at all. This is because for a logical AND to be true, both of its conditions must be true.

It's always worth running the **update** command before installing new software too, as minor updates are made in even stable distributions, should a security problem be found in any of Raspbian's software. We've just run an update, so no need to repeat that for now. Sticking with a command-line theme, we're going to install an old suite of terminal games:

```
sudo apt-get install bsdgames
```

Searchable list

You can find particular apps with apt-cache search: **apt-cache search games**

And examine individual packages with apt-cache show: **apt-cache show bsdgames**

Apt is actually a front end to the lower-level **dpkg**, which you can call to see what you have installed on the system: **dpkg -l**. Even on a fresh system, that's a large listing – we'll show you how to get useful information from such listings another time.

Downloaded packages hang around in **/var/cache/apt** and if you find yourself short on disk space, issuing **sudo apt-get clean** will clear out the archive, without affecting the installed software.

Now, remember the extra details that **ls -lh** showed us in part 1? Try **ls -lh /etc/apt**

That **-rw-rw-r--** at the beginning of the listing for **sources.list** comprises file attributes, telling you who may use the file. Other entries in the listing have a **d** at the beginning, indicating they are directories. You'll also see hardware devices have a **c** here, for character device – **ls -l** on **/dev/input**, for example. On Linux, everything is a file, even your mouse! An **a** tells us this is just a regular file; it's the remaining nine characters of the group that cover permissions.

Every file has an owner and a group membership. Files in your home directory belong to you. If you're logged in as user pi and **ls ~ -l**, you'll see **pi pi** in each listing, telling you the owner and the group. Note: we put the switch at the end this time – that's a bad habit under certain circumstances, but we're just showing you what's possible. Owner and group aren't always the same, as **ls -l /dev** will show you.

> "Every file has an owner and a group membership"

Octal version

Those numbers are an octal representation of user, group, and others' permissions: in each case, read is represented by 4, write by 2, and execute by 1, all added together. So here we have 7s for read+write+execute for user and group, and 5 for read+execute for all other users. **ls -l** and you'll see we're back to **-rwxrwxr-x**.

You can use **chown** to change who owns a file and **chgrp** to change which group it belongs to. Make a new text file and **sudo chown root myfile.txt** – now try editing it and you'll find that while you can read the file, you can no longer write to it. You can also make a file that you can write to and run, but not read!

Next time we'll be doing useful things with the output of our commands, but meanwhile have a go at **robots** from the **bsdgames** package we installed – it'll come in handy later for another purpose.

FREE TO USE

Software in the main repository is not just free to use, but freely modifiable and redistributable too. Free software, like Raspbian's Debian base, is built on sharing – for education and for building community.

PROBLEMS?

Fine-grained permissions make for greater security, but can trip you up. Typing **sudo** in front of a command that doesn't work is both a diagnosis and a quick workaround of a permissions problem.

ALEX EAMES

Alex runs RasPi.TV and RasP.iO, and finds himself between a blog and a hardware business. You can find him as @RasPiTV on Twitter.

RasPi.TV

WATCH IPLAYER ON RASPBERRY PI

Download BBC TV programmes as high-definition MP4 files (without any DRM) for offline viewing on your Pi or other devices…

You'll Need

- A decent-sized SD card (minimum 8GB)
- A broadband connection
- A few lines of code: github.com/raspitv/get_iplayer
- A TV, or a monitor and speakers

Get_iplayer is a fabulous, open source utility program which allows you to see what's currently available from the BBC's iPlayer website and download any TV programmes you want. You can choose resolutions between 512×288 and 1280×720, so you can pick one that suits your viewing device and storage capacity. There's no DRM on these files (although the BBC's terms state that you are only allowed to keep them for 30 days), and because they are BBC programmes, there are no adverts either, making it perfect to use for films.

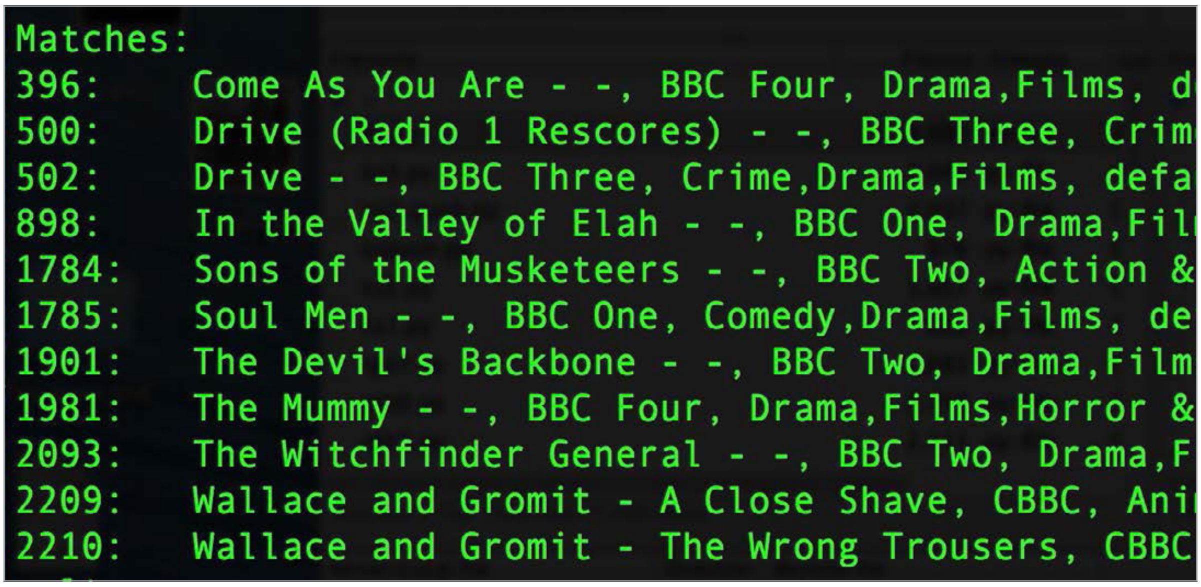
>STEP-01

Update your package lists

We're going to install some software. The first thing you should do when you install software in Raspbian is – at the command prompt – type `sudo apt-get update` (and press **ENTER**), which updates your package lists. Then it's a good idea to `sudo apt-get upgrade` (pressing **ENTER** again) once all your software packages have been updated to their latest versions. It can take upwards of 30 minutes if you haven't done this recently. Then you need to add

Get_iplayer enables you to download films and TV shows from the BBC iPlayer service

As the BBC doesn't use DRM, you can watch shows on your Pi or any other device



```
Matches:
396:    Come As You Are - -, BBC Four, Drama,Films, d
500:    Drive (Radio 1 Rescores) - -, BBC Three, Crim
502:    Drive - -, BBC Three, Crime,Drama,Films, defa
898:    In the Valley of Elah - -, BBC One, Drama,Fil
1784:   Sons of the Musketeers - -, BBC Two, Action &
1785:   Soul Men - -, BBC One, Comedy,Drama,Films, de
1901:   The Devil's Backbone - -, BBC Two, Drama,Film
1981:   The Mummy - -, BBC Four, Drama,Films,Horror &
2093:   The Witchfinder General - -, BBC Two, Drama,F
2209:   Wallace and Gromit - A Close Shave, CBBC, Ani
2210:   Wallace and Gromit - The Wrong Trousers, CBBC
```

Left Here's some example output from the `get_iplayer --cat film` command. Note the programme IDs (PIDs) on the left

Jon Davies's PPA (Personal Package Archive) to your **sources.list**. Copy and paste the five lines of GitHub code from **github.com/raspitv/get_iplayer/blob/master/code.txt** into a terminal window on your Raspberry Pi and press **ENTER**.

>STEP-02

Install the keyring and software

Once you've done this, you need to repeat the **`sudo apt-get update`** command in the terminal. You'll likely get an error message about keyrings, so now you need to install Jon's keyring as well with the following command:

```
  sudo apt-get --allow-unauthenticated -y
install jonhedgerows-keyring
```

Then press **ENTER**.

Next, repeat the **`sudo apt-get update`** command one last time. Now we're ready to go ahead and install the **get_iplayer** program itself (notice the installation name in the command we type is hyphenated, not underscored):

```
  sudo apt-get install get-iplayer
```

>STEP-03

Using get_iplayer

Before we start, it's always good to know where to find help, should you need it. To do this, type: **`get_iplayer --usage`** in the terminal, which should give us a list of the basic options. If you want more options, you can use **`get_iplayer --help`**, or even more using the command **`get_iplayer --longhelp`**.

There are a lot of options, so it can be a bit overwhelming, but most of them are not needed for simple searching and downloading of content. Before we can download a programme, we need to collect the index of all the available content. This is done by using the **`get_iplayer`** command all by itself in the terminal.

>STEP-04

Narrowing down the search

At any given time, there are a couple of thousand items available for download. That's a bit overwhelming, so we need a way to cut it down a bit. You can use categories with the **`get_iplayer`** command – for example, **`get_iplayer --cat film`**. You can choose any category from the main list: Arts, CBBC, CBeebies, Comedy, Documentaries, Drama and Soaps, Entertainment, Films, Food, History, Lifestyle, Music, News, Science & Nature, or Sport. You can also use a keyword; if matched, it'll return possible downloads.

>STEP-05

Downloading content

Looking at the list of films available, each item starts with the programme's ID number (PID). Let's choose *Wallace and Gromit – The Wrong Trousers*. This has a PID of 2210. So, to download this film at the best available resolution (1280×720), you would type:

```
  get_iplayer --get 2210 --modes best
```

After about 10-15 minutes, the file is downloaded and processed into an MP4 file, which we can view, store or delete at will.

>STEP-06

Watching content

As part of the default Raspbian installation, you have a GPU-accelerated media player called Omxplayer. Because it uses the GPU, it's capable of playing HD video, even on a Pi Model A. To watch the film we just downloaded, we would type the following into the terminal:

```
  omxplayer [filenamehere].mp4
```

If you're dealing with long filenames, once you've typed **omxplayer** and the first few letters of the filename, you can press the **TAB** key and it will auto-complete the filename for you (then press **ENTER**). You can see the full list of Omxplayer controls at **elinux.org/Omxplayer**.

DO I HAVE ENOUGH SPACE?

Use the command **`df -h`** to see if you have enough space left on your SD card.

WANT SOUND THROUGH THE AUDIO JACK?

Use **`omxplayer -o local`** to send sound through the Pi's audio jack rather than the (default) HDMI port.

DISCLAIMER

The BBC's T&Cs state that all iPlayer content is for UK playback only. In addition, any downloads must not be kept beyond 30 days, and must not be distributed in other forms. Neither MagPi nor the Raspberry Pi Foundation condones any breach of these rules. For further details, visit **bbc.co.uk/terms**

SIMON LONG

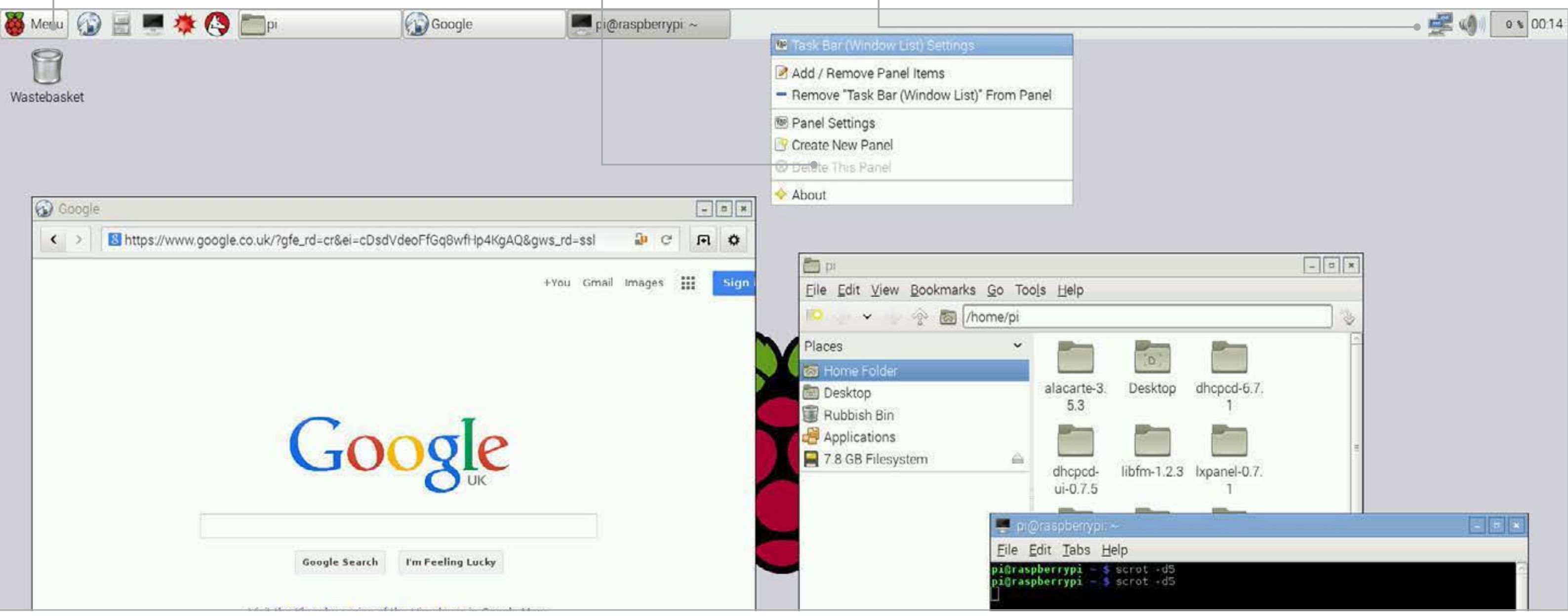
Simon Long works for Raspberry Pi as a software engineer, specialising in user interface design. In his spare time he writes apps for the iPhone and solves *really* difficult crosswords.

raspberrypi.org

All items on the menu bar panel are plug-ins, enabling the layout to be customised – this is the 'menu' plug-in

Right-clicking a plug-in on the panel will usually bring up a menu, which contain options for both the plug-in and the panel itself

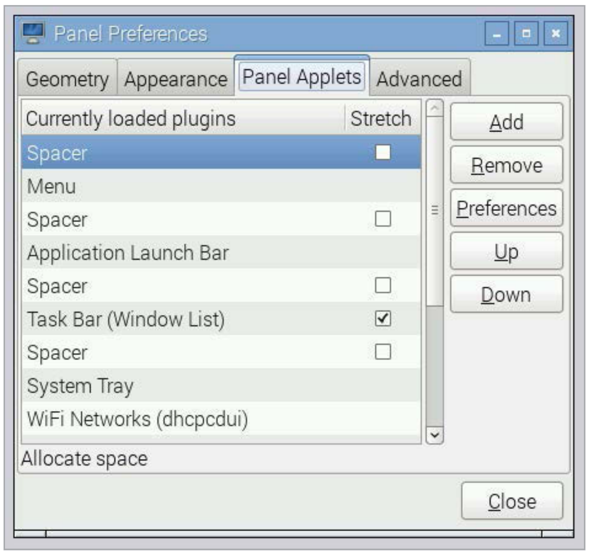
Plug-ins can have icons that change in response to status – the network icon, for example, shows the current connection state



```
pi@raspberrypi ~ $ scrot -d5
pi@raspberrypi ~ $ scrot -d5
```

HACKING RASPBIAN'S DESKTOP PART 2: CUSTOMISING LXPANEL

In part two of his series, **Simon Long** shows us how to change the appearance of the Raspbian desktop by playing with LXPanel...



Currently loaded plugins	Stretch
Spacer	☐
Menu	
Spacer	☐
Application Launch Bar	
Spacer	☐
Task Bar (Window List)	✓
Spacer	☐
System Tray	
WiFi Networks (dhcpcdui)	

Allocate space

Right The Panel Applets tab on the Panel Preferences dialog allows plug-ins to be added, removed, and rearranged

LXPanel is a component of LXDE (Lightweight X11 Desktop Environment), the desktop user interface included as part of Raspbian. As the name suggests, LXPanel is responsible for generating panels – windows that overlay the desktop which can be used for menu bars, application launchers, and other general system functions.

By default, a single panel is displayed on the Pi's desktop, which is used for the menu bar at the top of the screen. LXPanel also allows multiple panels to be used on the same screen, so it is possible to have a second panel displayed at the bottom, to be used as an application launcher, for example. Panels are always attached to one edge of the screen, but the edge can be the top, the bottom, or even the sides.

Each panel can be customised to contain a selection of user interface components. These components are called plug-ins, and there are two kinds. LXPanel includes a number of built-in plug-ins (including the main menu button, the taskbar, and the quick launch icon bar), and standalone plug-ins can also be written from scratch and added to LXPanel. Writing a new plug-in is a fairly complicated programming task and goes beyond the scope of this article, but information on how to get started with writing one can be found at the **lxde.org** site.

In addition to the plug-ins associated with a panel, there are a number of global parameters which control various aspects of the overall display of each panel. Many plug-ins also have parameters which can be set to control their individual appearance and behaviour.

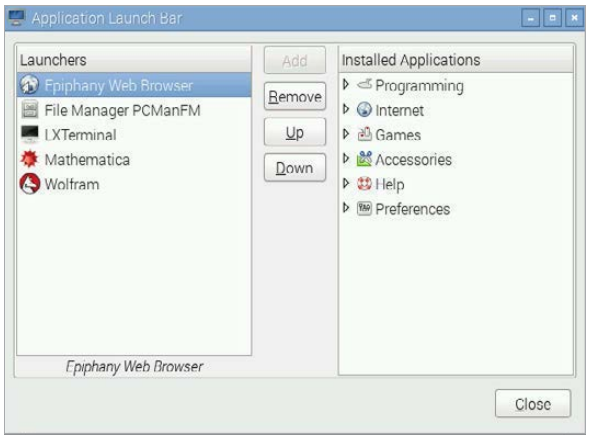
Tweaking the panel

The easiest way to change the configuration of a panel or a plug-in is by using the LXPanel preference dialogs. If you right-click on a panel or one of its plug-ins, a menu pops up. (Note that some plug-ins override this menu, so don't be surprised if this doesn't work in some cases.) The top item on this menu is **<name of plug-in> Settings** – selecting this opens the preferences dialog for that particular plug-in; obviously, the items in that dialog will depend on what customisation is possible for the plug-in.

The menu also contains a number of other menu items for customising the panel itself.

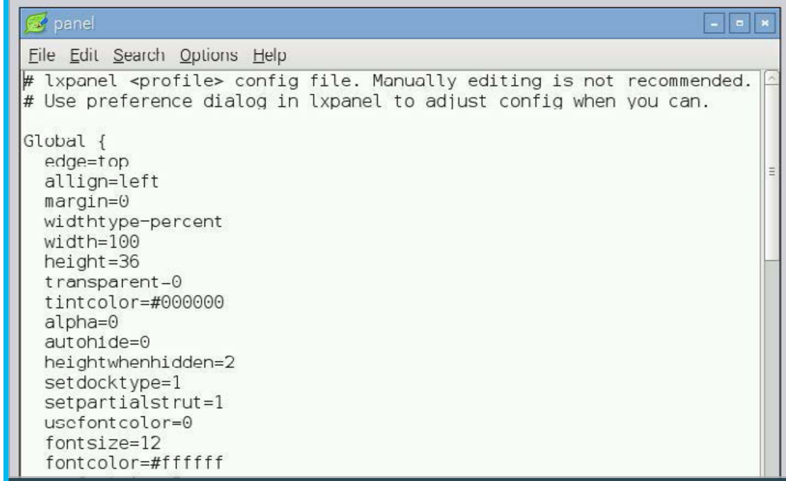
Add/Remove Panel Items opens a dialog box which shows the plug-ins currently on the panel; from here, it is possible to add, remove, or rearrange the plug-ins on the panel.

Also in the menu is **Panel Settings** – this opens a dialog which allows global settings for the panel to be changed, including the edge of the screen to which it is attached, and how much of the screen it takes up. Other tabs on this dialog allow the colour and background of the panel to be changed, and to control whether or not the panel automatically hides when the mouse is moved away, and reappears when it is moved back. Adding or removing panel items can also be performed from a tab on this dialog.



Above Individual plug-ins can be customised – this is the configuration dialog for the application launch bar plug-in

THE PANEL FILE



```
# lxpanel <profile> config file. Manually editing is not recommended.
# Use preference dialog in lxpanel to adjust config when you can.

Global {
  edge=top
  allign=left
  margin=0
  widthtype-percent
  width=100
  height=36
  transparent-0
  tintcolor=#000000
  alpha=0
  autohide=0
  heightwhenhidden=2
  setdocktype=1
  setpartialstrut=1
  usefontcolor=0
  fontsize=12
  fontcolor=#ffffff
```

In general, it should not be necessary to edit this file by hand – all parameters that you want to set should be modifiable via the relevant settings dialogs. If you do edit the file by hand, the changes will only be applied to the current desktop if LXPanel is restarted; you can achieve this by either rebooting the Pi, or by issuing the following command from a terminal window: `lxpanelctl restart`

The menu also includes the option to **Remove <name of plug-in> From Panel** – this removes the plug-in you right-clicked from the panel, and is identical in effect to using the Remove button in the dialog box mentioned above.

Finally, the menu includes the option to **Create New Panel** – this creates a new blank panel on a free edge of the screen. This can then be customised by right-clicking on the new panel to bring up the same menu – but in this case, options selected will affect the newly created panel.

Under the bonnet

Changes made in these dialogs are stored in the LXPanel configuration settings file. This, along with various other settings files, is found in the hidden **.config** directory inside your home directory.

Inside **.config/lxpanel** is a directory with the name of your current lxsession profile – this is called **LXDE-pi** on a default installation of Raspbian – and inside this is a further subdirectory called **panels**. Each file in this directory is the definition of a single panel. By default, there is just the one file called **panel**, but if multiple panels are added, there will be multiple files in this directory – one per panel.

The panel file, which can be viewed or changed with a text editor, includes an initial **Global** section in which general panel parameters are stored, followed by a section for each plug-in. Each **Plugin** section contains a line giving the **type** of the plug-in, followed by a **Config** section containing parameters specific to that plug-in.

NEXT MONTH

In the next part of this series, we look at how to use the window manager to change the appearance of applications.

MARTIN O'HANLON

Martin 'Minecraft' O'Hanlon is an active member of the Raspberry Pi community, co-author of *Adventures in Minecraft*, and keeps an excellent account of his projects on his blog.
stuffaboutcode.com

MORE MINECRAFT CODING TIPS & TRICKS

Have you exhausted the Minecraft: Pi basics available from **raspberrypi.org/resources**? Have you completed our tips from issue 31? Here are another five tips and mini-programs to experiment with...

You'll Need

- Raspbian
- Minecraft: Pi Edition
- Python 2 editor (IDLE)

BACK UP AND RESTORE MINECRAFT WORLDS

Ever accidentally set off a load of TNT and wished you hadn't? It's at times like these it's a good idea to have a backup of your whole Minecraft world so you can restore it back to normal.

Well, you can. *Minecraft: Pi Edition* stores all of your worlds in a directory on your Raspberry Pi's SD card, so by using the terminal and a few commands, you can find your favourite world and back it up to a file. Open a Terminal window with:

Menu > Accessories > Terminal.

Next, change directory to the **minecraftWorlds** directory using the following command:

```
cd ~/.minecraft/games/com.mojang/ ↩
minecraftWorlds
```

Each world is saved in its own directory and named the same as what's displayed in the *Minecraft* 'Select World' screen. Use the **`ls`** terminal command when you're in that directory to see your saved worlds. To make a backup of **world--**, use the **`tar`** command to create a compressed file:

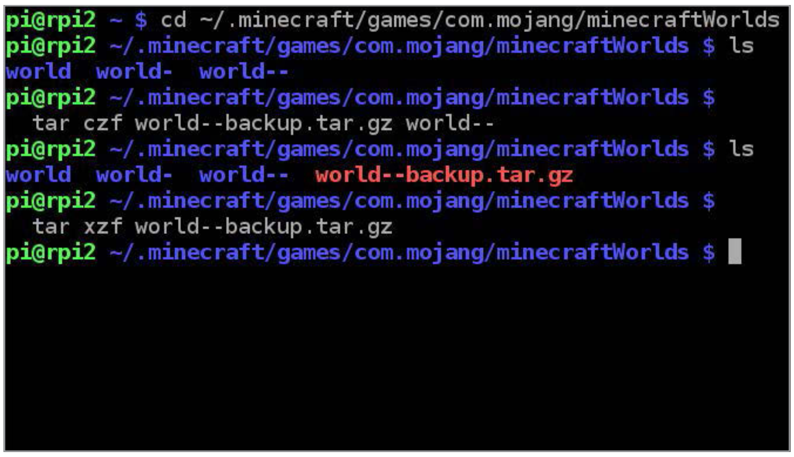
```
tar czf world--backup.tar.gz world--
```

`tar` is the command, **`czf`** tells it to Create a Zipped File, **`world–backup.tar.gz`** is the name of the backup file, and **`world--`** is the directory of the world you want to back up.

Now, the next time you want to restore your world, all you have to do is use the following command:

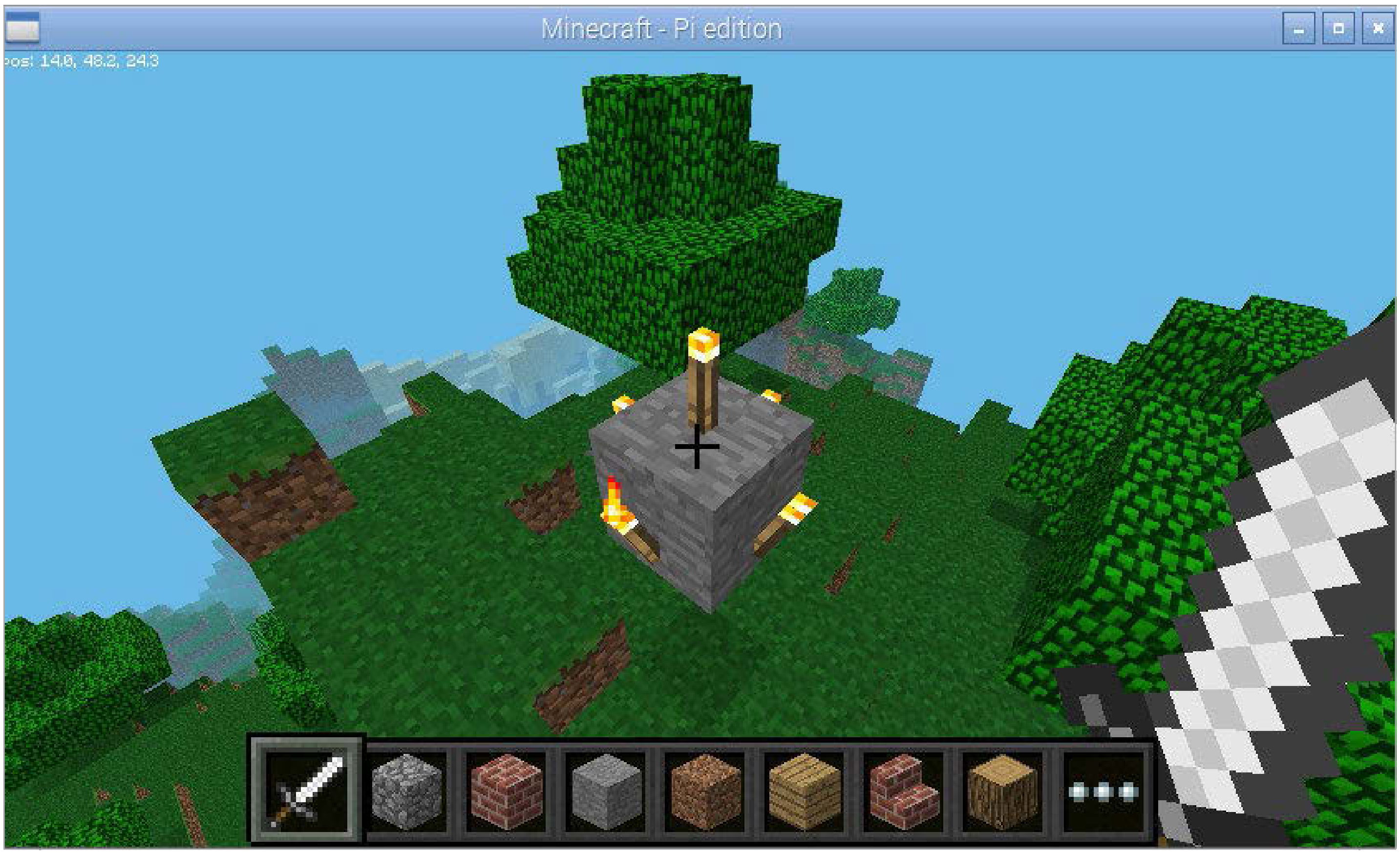
```
tar xzf world--backup.tar.gz
```

Be warned: once entered, there is no going back!



```
pi@rpi2 ~ $ cd ~/.minecraft/games/com.mojang/minecraftWorlds
pi@rpi2 ~/.minecraft/games/com.mojang/minecraftWorlds $ ls
world  world-  world--
pi@rpi2 ~/.minecraft/games/com.mojang/minecraftWorlds $
 tar czf world--backup.tar.gz world--
pi@rpi2 ~/.minecraft/games/com.mojang/minecraftWorlds $ ls
world  world-  world--  world--backup.tar.gz
pi@rpi2 ~/.minecraft/games/com.mojang/minecraftWorlds $
 tar xzf world--backup.tar.gz
pi@rpi2 ~/.minecraft/games/com.mojang/minecraftWorlds $ 
```

Below Back up your *Minecraft* worlds as a compressed file



Above Use torches like a pro, automatically placing them around a block

USING TORCHES

If you want to get some light into your *Minecraft* world, you need to create torches. You can do so using the API, but you need to know how to place the torches around the block you want to attach them to.

Torches have their own block type and take up an entire block space in *Minecraft* (even though it looks like they take up a small amount of it), which is why you can't have more than one torch in a block.

> You could change the code to make a tower of stone

When torches are created using the **`setBlock()`** API function, it automatically connects the torch to the block which is next to it, to the north, south, east, west, or on top.

Use the following code to create a block of stone above the player and place torches all around it, and one on top.

```
from mcpi import minecraft
from mcpi import block

mc = minecraft.Minecraft.create()

pos = mc.player.getTilePos()

mc.setBlock(pos.x, pos.y + 2, pos.z, block.STONE)

# create torches
# on top
mc.setBlock(pos.x, pos.y + 3, pos.z, block.TORCH)
# to the east
mc.setBlock(pos.x + 1, pos.y + 2, pos.z, block.TORCH)
# to the west
mc.setBlock(pos.x - 1, pos.y + 2, pos.z, block.TORCH)
# to the north
mc.setBlock(pos.x, pos.y + 2, pos.z - 1, block.TORCH)
# to the south
mc.setBlock(pos.x, pos.y + 2, pos.z + 1, block.TORCH)
```

You could change the code to make a tower of stone with torches all around, to provide a beacon to help you find your way.

FIND OUT WHEN BLOCKS ARE HIT

When Steve hits a block by right-clicking with a sword in *Minecraft*, it creates a 'hit event'; you can get this using the API and it'll tell you who hit the block, its position, and what face (i.e. top, bottom, left, right) it was hit on.

You use the function **events.pollBlockHits()** to get a list of events that have occurred since it was last called. You can then loop through events using a **for** loop.

```
from mcpi import minecraft

mc = minecraft.Minecraft.create()

while True:
    hitsList = mc.events.pollBlockHits()
    for hit in hitsList:
        mc.postToChat("A block was hit (who, ↵
position, face)")
        mc.postToChat(hit.entityId)
        mc.postToChat(hit.pos)
        mc.postToChat(hit.face)
```

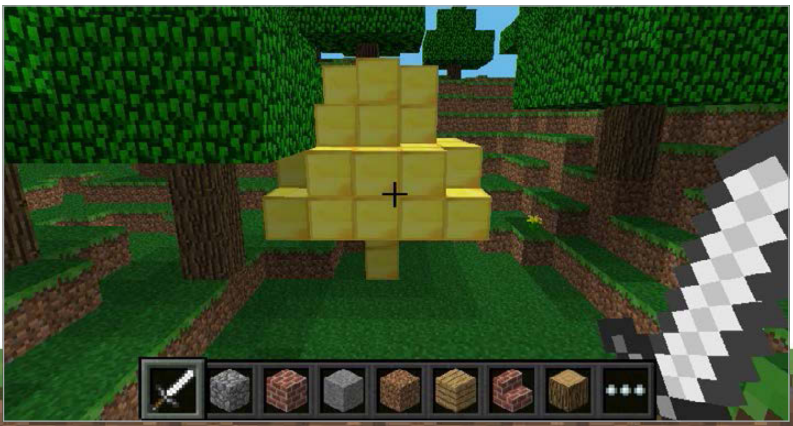
Start up *Minecraft*, run the program above, and experiment with hitting some blocks by holding a sword and right-clicking the block. By using the position and the **getBlock()** function, you can find out the type of block (e.g. stone, dirt, grass) that was hit:

```
        blockType = mc.getBlock(hit.pos)
        mc.postToChat(blockType)
```

Or even better, change the block which was hit, using **setBlock()** to give Steve the Midas touch and make every block he hits turn to gold:

```
        mc.setBlock(hit.pos, 41)
```

Have a think about what other things you can make happen in *Minecraft: Pi* using block hit events.


Below Give Steve the Midas touch and turn blocks into gold!

SAVE AND RESTORE CHECKPOINTS

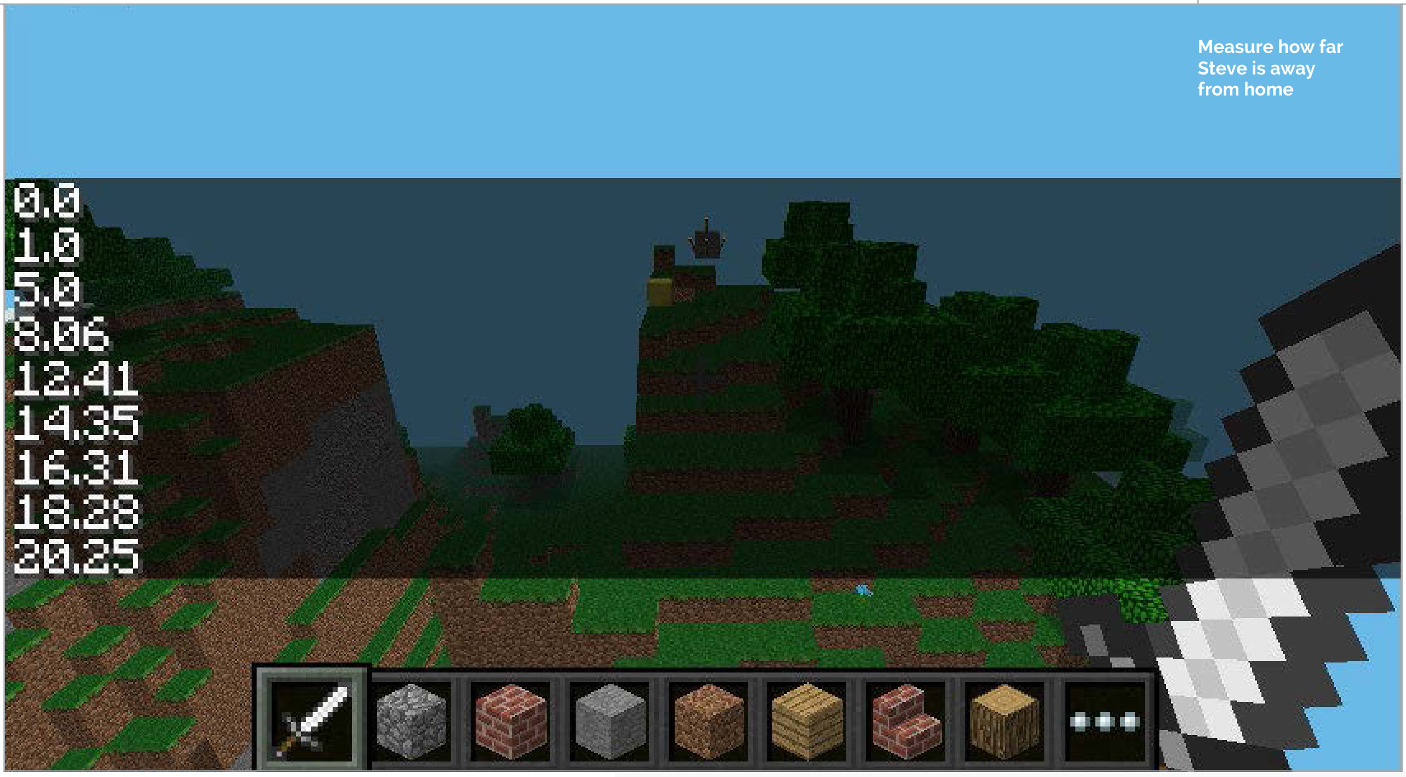
Checkpoints let you create in-game mini-backups so you can undo changes that have been made. You can use the **saveCheckpoint()** API function to make a temporary copy of your world; when you use **restoreCheckpoint()**, this copy is used to put your world back to how it was when you saved the checkpoint.

```
from mcpi import minecraft

mc = minecraft.Minecraft.create()

mc.saveCheckpoint()

mc.restoreCheckpoint()
```

"Every 30 seconds, your program will save a checkpoint"

You can use the checkpoint functions to create a program which will allow you to 'undo' any unwanted changes you make to your *Minecraft* world. Every 30 seconds, your program will save a checkpoint and if you ever want to go back to it, just hit a block.

```
from mcpi import minecraft
from time import sleep

mc = minecraft.Minecraft.create()

count = 0

while True:
    #every 30 secs save a checkpoint
    if count % 30 == 0:
        mc.saveCheckpoint()
        mc.postToChat("Checkpoint saved")
    count = count + 1
    sleep(1)

    #if a block is hit, restore checkpoint
    if mc.events.pollBlockHits():
        mc.restoreCheckpoint()
        mc.postToChat("Restoring checkpoint")
```



Measure how far Steve is away from home

CALCULATING THE DISTANCE BETWEEN TWO BLOCKS

When coding *Minecraft*, it's really useful to know the distance between two blocks – and by using some (fairly) simple maths, we can work it out. This can be used in loads of fun ways, such as a hide and seek game where a diamond block is hidden and Steve is told whether he is getting colder or warmer.
The maths works like this:

01 **Calculate the difference between the x, y & z coordinates of the two positions**

02 **Multiply the difference by itself (its square)**

03 **Add all the squares together**

04 **The distance equals the square root of the total above**

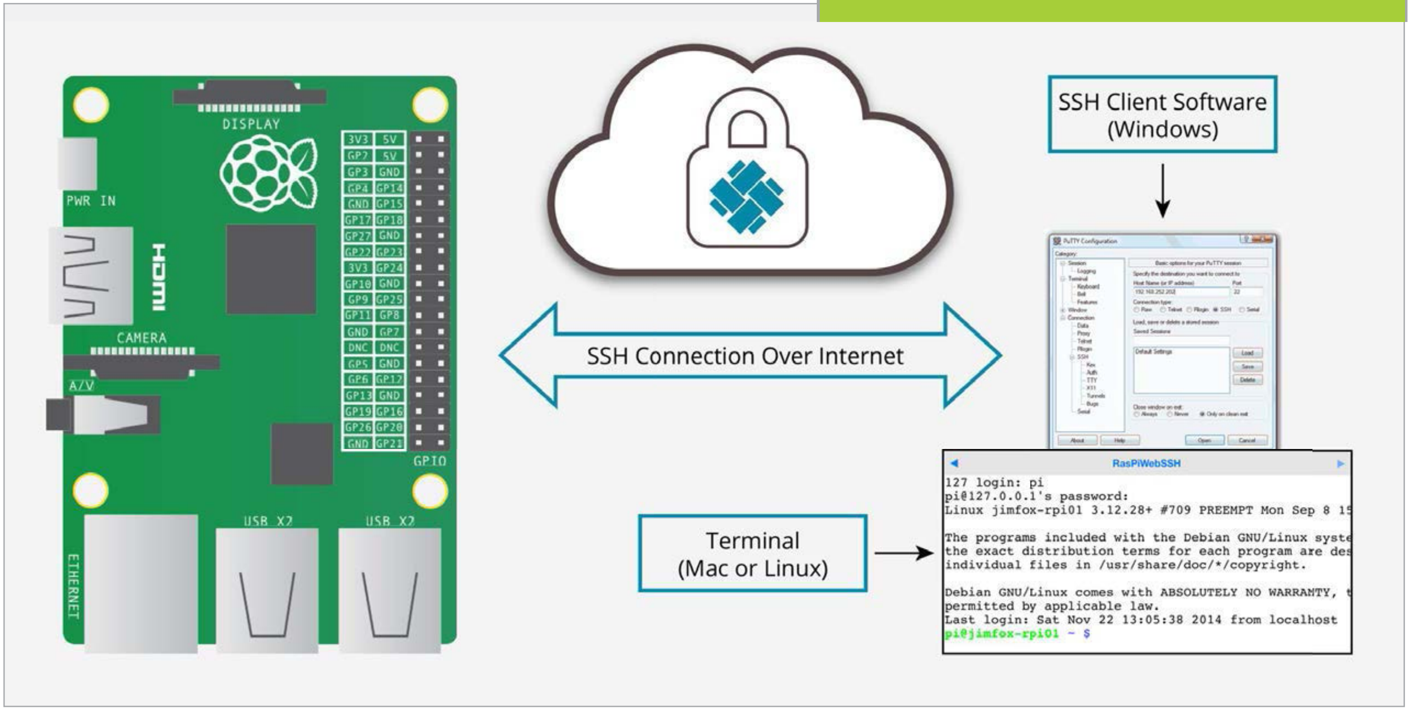
This program uses this calculation to display how far the player is from where they started. So the further they move away, the greater the distance. See how it works by copying the following code example into IDLE or your favourite text editor (don't forget to save it with the **.py** file extension):

```
from mcpi import minecraft
from math import sqrt
from time import sleep

mc = minecraft.Minecraft.create()

startPos = mc.player.getTilePos()

while True:

    posNow = mc.player.getTilePos()

    xDiff = startPos.x - posNow.x
    yDiff = startPos.y - posNow.y
    zDiff = startPos.z - posNow.z

    xSquare = xDiff * xDiff
    ySquare = yDiff * yDiff
    zSquare = zDiff * zDiff

    total = xSquare + ySquare + zSquare

    distance = sqrt(total)

    mc.postToChat(distance)

    sleep(1)
```

Try changing the program to show the distance between the player and a random diamond block you have created.

JAMES FOX

James works at the Weaved HQ in Palo Alto, California. Weaved is a small group of networking geeks who love all things TCP/IP, Linux, basketball, and English Bulldogs.
weaved.com @weavedinc



```
127 login: pi
pi@127.0.0.1's password:
Linux jimfox-rpi01 3.12.28+ #709 PREEMPT Mon Sep 8 15

The programs included with the Debian GNU/Linux syste
the exact distribution terms for each program are des
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRAMTY, t
permitted by applicable law.
Last login: Sat Nov 22 13:05:38 2014 from localhost
pi@jimfox-rpi01 ~ $
```

You'll Need

- Free Weaved account **developer.weaved.com**
- Latest Raspbian image **raspberrypi.org/downloads**
- A PC (Windows, Mac, or Linux)
- Internet connection (for both systems)

SSH ACCESS FROM THE WEB ON RASPBERRY PI

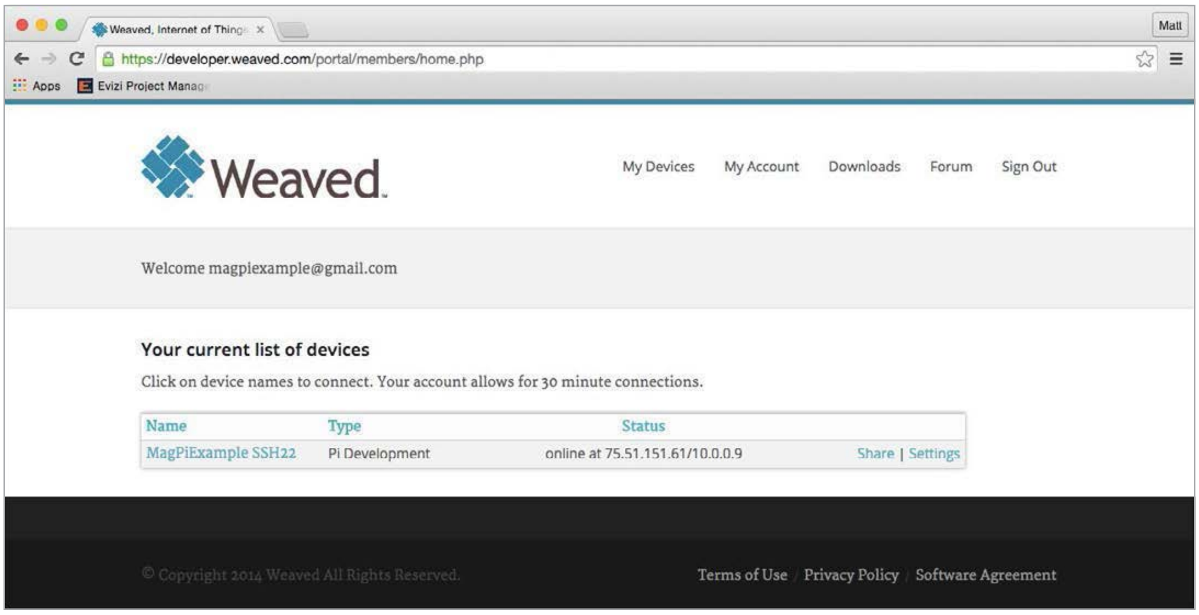
SSH into your Pi from anywhere over the internet using Weaved – no need for a static IP address, dynamic DNS, or using port forwarding

SSH (Secure Shell) is a powerful tool in Raspbian for establishing a secure command-line interface between your Raspberry Pi and another computer. If you've been using a Raspberry Pi for a while, you're probably already familiar with SSH as a handy way to interact with your Pi in headless mode (without an HDMI monitor, keyboard, and mouse). Now, with Weaved, you can SSH into your Pi remotely from anywhere in the world over the internet, using free cloud services and downloadable software. There's no longer any need to reconfigure your router to port forward and expose your home network to the internet.

>STEP-01

Get Weaved

From any browser, go to **developer.weaved.com** to register for a free Weaved user account. Enter your email and create a password for your account, then click 'Sign Up'. You'll use your Weaved account later to access your Pi from anywhere over the internet via SSH. For this tutorial, we created an account using the fictitious email address **MagPiExample@gmail.com**. Next, navigate to the Weaved Downloads page and click on the Raspberry Pi logo. You'll arrive at the Installation Instructions for Raspberry Pi webpage – **developer.weaved.com/portal/members/betapi.php**. Follow the instructions to download and run the automated Weaved installer.



Welcome magpiexample@gmail.com

Your current list of devices

Click on device names to connect. Your account allows for 30 minute connections.

Name	Type	Status	
MagPiExample SSH22	Pi Development	online at 75.51.151.61/10.0.0.9	Share | Settings

© Copyright 2014 Weaved All Rights Reserved.

Left The device list shows your devices and lets you make connections from any web browser or the Weaved app

CONNECTION TIME

The Weaved SSH hostname and port are assigned for 30-minute sessions. Once that time has elapsed, you can reconnect from your device list.

>STEP-02

Installing Weaved

During installation, you will be asked for inputs at the command line. Enter **1** at the Protocol Selection Menu to select SSH on default port 22. Next, confirm your selection and enter your Weaved account email and password when prompted for them. You'll also be asked to enter an alias (a name) for the SSH service on your Pi. Later, you will connect over the internet via SSH by selecting this name from your Weaved device list. In this tutorial, we named the service 'MagPiExample SSH'. When installation is complete, you'll see the CONGRATULATIONS message.

>STEP-03

Log in to connect

Now you're ready to connect to your Pi via SSH over the web. Go to **weaved.com** from any web browser and sign into the account created in step 1. After login, you'll see your device list, which now includes the newly registered SSH service on your Pi. Give your Pi up to 3 minutes to appear as active in your device list, from the time the installer finished running in step 2.

>STEP-04

SSH over the web

Click on the SSH service name in the device list. The Weaved service will generate a pop-up window with a private host name and port number to use for your connection. Copy/paste the hostname and port number generated by the Weaved service into an SSH client like PuTTY (Windows). Or use the SSH command provided in a terminal window (Mac/Linux). If you're typing the SSH command in a terminal window, substitute your Pi's local username and the hostname and port number generated by the Weaved service, as indicated. Then log in with the password from your Raspberry Pi, e.g. user 'pi' and password 'raspberry'. You will then be logged in!

>STEP-05

Share your Pi

Now you can share the remote connection services on your Pi with other users. Grant temporary, or even permanent, access to your Pi to your friends and fellow Pi enthusiasts! It's a convenient way to get 'remote desktop' help from a more experienced Pi user or show off your latest project. Sharing a connection service on your Pi with another Weaved user is easy. Go to your device list and click Share. Just enter your friend's Weaved account email address and click Add. Now, your Pi will appear in their device list until you un-share. It's all under your control.

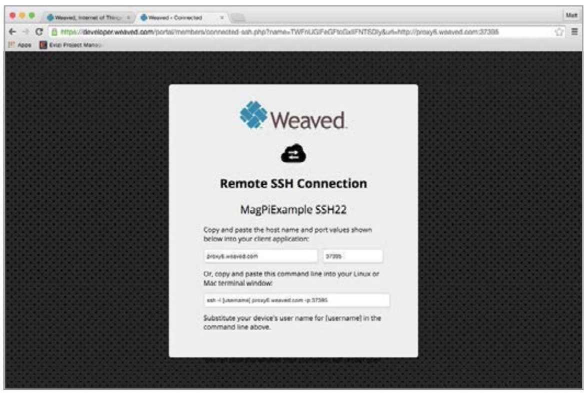
>STEP-06

Get the mobile app

Download the free Weaved mobile app for iOS or Android to enjoy convenient internet access to your Pi from your phone or tablet. Log into the app to see your device list, then click on the SSH service to connect. You'll need a third-party SSH client app for iOS or Android to run the remote client side (phone/tablet) of the SSH connection. Then copy/paste the Weaved-generated hostname and port number into your SSH app, just like you did in step 4.

INSTALL MULTIPLE SERVICES

Run the installer again to enable simultaneous remote access to any combination of TCP services on your Pi (e.g. HTTP, VNC).



Remote SSH Connection

MagPiExample SSH22

Copy and paste the host name and port values shown below into your client application:

proxy6.weaved.com 37395

Or, copy and paste this command line into your Linux or Mac terminal window:

```
ssh -l [username] proxy6.weaved.com -p 37395
```

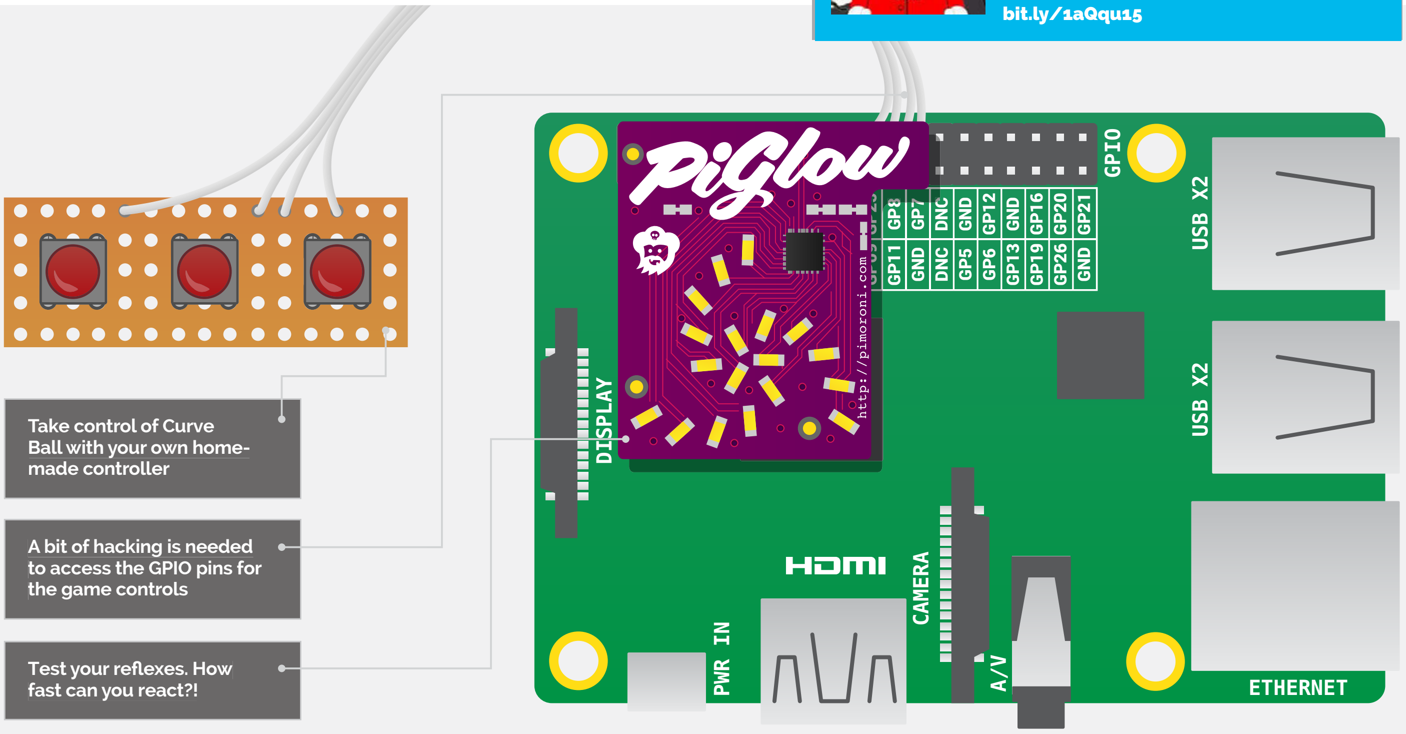
Substitute your device's user name for [username] in the command line above.

Left Copy/paste the hostname and port generated by the service into your SSH client (e.g. PuTTY for Windows). Or enter the full command, substituting your Pi's username

MIKE'S PI BAKERY

MIKE COOK

Veteran magazine author from the old days and writer of the Body Build series. Co-author of *Raspberry Pi for Dummies*, *Raspberry Pi Projects*, and *Raspberry Pi Projects for Dummies*.
bit.ly/1aQqu15


PiGlow
http://pimoroni.com
GP11 GND DNC GP5 GP6 GP13 GP19 GP26 GND
GP8 GP7 DNC GND GP12 GND GP16 GP20 GP21
GPIO
USB X2
USB X2
DISPLAY
HDMI
CAMERA
PWR IN
A/V
ETHERNET

Take control of Curve Ball with your own home-made controller

A bit of hacking is needed to access the GPIO pins for the game controls

Test your reflexes. How fast can you react?!

CURVE BALL

Turn Pimoroni's PiGlow into an interactive game that will test your reactions...

You'll Need

- Pimoroni PiGlow
- Small piece of stripboard (15 × 5 holes)
- 3x tactile switches (or similar)
- Stranded wire
- Soldering iron
- Wire cutters

While the PiGlow board does have a library and some code examples, it is worth looking into how to drive the board directly. These days we do beginners a disservice by turning virtually anything into a library. This discourages them from actually learning how to talk to a device and leaves them trying to use library calls that they do not understand. In short, it all adds to the fog of what they are trying to do.

The PiGlow

The design of the PiGlow board is quite simple: it revolves around an SN3218 which is an I²C, 18-channel LED driver. We'd recommend that you download the data sheet and take a look. Now, data sheets can be quite intimidating and put us in mind of a small child we encountered some years ago, who said, "This book tells me more about penguins than I wish to know." Data sheets are the same: there is a lot you simply do not need to know, especially if you are looking at how to drive a project in a ready-built board. The most useful part of any data sheet is the front page – this is the headline feature, and then the details of those

PREREQUISITES

We need to set up a few requirements from the command line. Press **RETURN** after each of the following commands and follow instructions when prompted:

```
  sudo apt-get update
  sudo apt-get upgrade
  sudo apt-get dist-upgrade
  sudo apt-get install python-smbus
  sudo apt-get install python-dev
python-pip
  sudo pip install wiringpi2
```

Finally, we need to configure I²C:
`sudo raspi-config` (enable I²C, make it on by default, then select reboot)
`sudo nano /etc/modules` (add the line `i2c-dev` to the end of the file, then save it)
`sudo reboot`

features are contained (somewhere) in the rest of the sheet. You will see that it is a constant current driver; that means there is no need for any resistors in line with the LEDs. Also, the brightness of each LED can be individually controlled by means of PWM. This stands for ‘pulse-width modulation’ and involves rapidly turning the LED on and off; the ratio of the on time to the off time determines the LED brightness. The LEDs on the PiGlow board are quite bright anyway and you may not always want them to go on at full blast.

Now, commendably, Pimoroni has included all the PCB source files, but it still took some time to work out where each LED was mounted physically on the board, what colour it was, and what channel number of the SN3218 it was connected to. All this information was extracted and distilled down into **Fig 1**. As the SN3218 is an I^{2}C interfaced device, it consists of a number of registers, or internal address locations, that control the lights. These are shown in the data sheet, starting on page 7, and are well worth taking a look at.

The control registers

The way this chip works is to have a register that sets the duty cycle (the ratio between the on-time and off-time) for each of the LEDs. So, that means that there are 18 of them, called PWM registers; they are numbered **`0x01`** to **`0x12`** (the 0x means what follows is a hexadecimal number). Writing a byte to those registers will set the brightness of that individual LED. However, there will be no change in the LED’s brightness until you write to register **`0x16`**; this is the update register, and transfers the levels you set in the PWM registers to the PWM circuitry. This means that the code can take its time updating the registers, but display them all at once. This is important because, as you all know, Linux has a habit of timing out your programs at unpredictable intervals, and this ‘update all at once’ function stops any staggering caused by seeing only a partial update. The other registers are used to enable the LEDs in three sets of six, set a shutdown, or reset the registers back to the default.

The controls

Now, to control the game, you need three switches attached to the GPIO pins. Tactile switches are best because they are dirt cheap; however, for a luxury game you can use larger, arcade-type momentary push switches, although you’ll have to mount them in a box to make them robust. Basically, the circuit is simple: one end of a switch goes to a GPIO pin, and the other end to ground. In the software, the internal pull-up resistors are enabled to give a solid logic **`1`** when the switch is not pressed, and a logic **`0`** when it is. We’re not great fans of solderless breadboards for making circuits – the problem is that at best it is only a temporary solution, but normally you have more trouble with the stability of the contacts than it is worth.

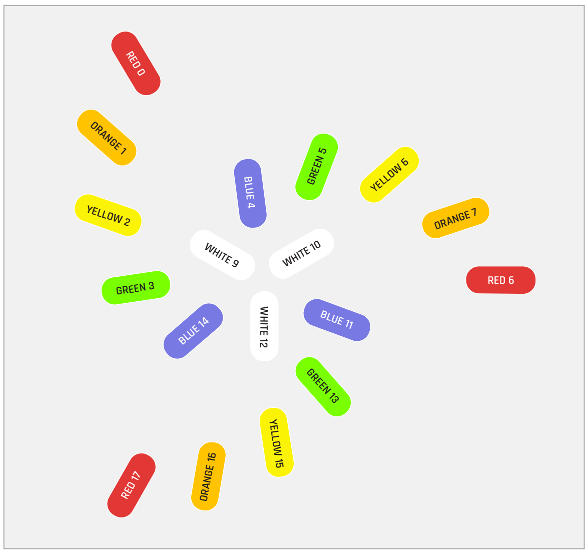


Fig 1 The PiGlow LED colours, positions, and driver numbers

The hack

It is a shame that the PiGlow board only actually uses four connections on the 26-way header – all the rest are connected, but not used. The answer was to single out a small, easy-to-get-at section where we could attach wires to the GPIO pins. For this, one ground and three GPIO pins were required. We decided that the last four connections on the far end of the board provided the best spot. A photograph of this is shown in **Fig 2**, overleaf – see the boxout for a step-by-step guide to building the game. If you are not comfortable, you could always use one of Pimoroni’s Black Hat hacker boards to get at the GPIO pins you need to use.

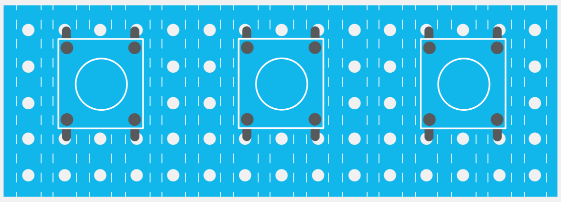
The game

When you play the game, the flight of the ball will be shown as a curved track of the LEDs lighting up one at a time. Your job is to hit it; that is, to press the button that corresponds to the white bat where the spiral path will end up. This is not quite as easy as it sounds. Once one button is pressed, the bat lights up and the buttons have no effect until the next ball. The sooner you press the button, the higher your score will be. If you chose the correct bat, then when the ball reaches the bat, it goes shooting off along the same path as it entered. The trick to the excitement level of the game is the speed of the incoming ball: it should not be too fast that you don’t have time to press the button, but it shouldn’t be much slower than this.

The code

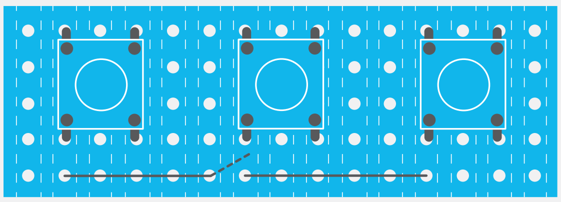
The code listing (overleaf) shows a simple version of the game with plenty of scope for you to change it. As you need direct access to the GPIO pins, you have to start IDLE with **`gksudo idle`** from the command line. The state of the LEDs is held in a list called **`lights`**; the **`updateLEDs`** function transfers this list to the I^{2}C chip so you can see them. There are a number of lists that define pin numbers and LED numbers. The

BUILDING THE GAME CONTROLLER



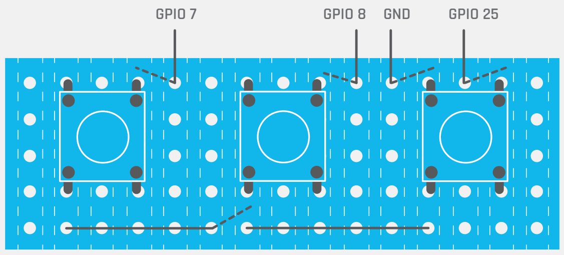
>STEP-01
Building the controls

Take a small piece of stripboard, 15 by 5 holes. Make sure the strips are vertical on the back. The dotted lines show the copper strips on the other side of the board as hidden detail. Mount three tactile switches on the board and solder them up.



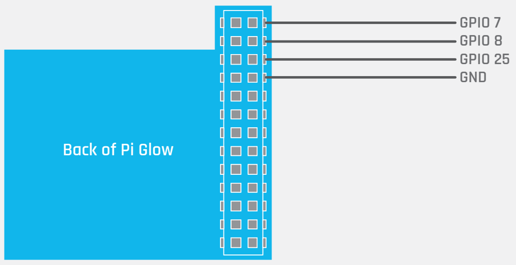
>STEP-02
Wire up the ground

The line along the bottom is the common ground that links one side of all three switches. The centre switch is connected to ground by the wire going down one hole and connecting to the next track along, as shown by the black dotted line in the diagram.



>STEP-03
Attach the wires

The wires to the PiGlow go through the hole and, underneath the board, are bent over to make a connection to the track next to it. This is shown as a black dotted line in the diagram. They are connected to the PiGlow.



>STEP-04
Connecting to the PiGlow board

Solder the ends of the wires to the underside of the PiGlow board. Make sure the minimum amount of insulation is stripped off the wire, to prevent adjacent wires shorting.

triangleIn list is, in fact, a list of lists, giving the order of the LEDs in any one spiral. The variables **speed** and **returnSpeed** control the tempo of the game. There are extensive comments and you should be able to see what is going on by reading the listing. One interesting thing to note is that the main function is controlled by a **try except** structure. This means that when **CTRL+C** is pressed to exit the program, there is an opportunity to turn off all the LEDs.

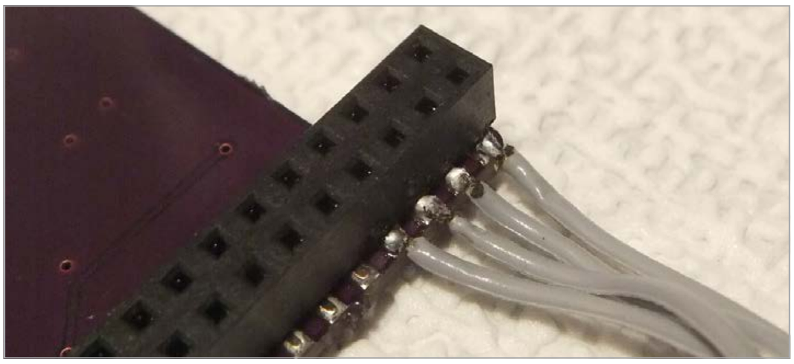


Fig 2 The solder pads on the PiGlow are tiny. Twist the wire, add solder to it, then cut it short to make soldering to the PiGlow easier

You have a go

The real fun part of any project is when you get to change things, to improve it or make it work just the way you want it to. Here, we would suggest you first play about with the speed variables, and also making the time delay before a pitch variable. You can add sound in a number of ways; the simplest is by giving a call to the omxplayer to play a sound when a hit is detected.

You could extend the game to keep a running score and have a 'three strikes and you're out' rule. You change which switch controls which bat by simply shuffling or redefining the **pinList** list. You could keep a high score, and break into some pyrotechnic displays on the LEDs when a new high score is generated.

Another idea is to turn the game on its head and call it 'Down The Plughole'! You have to stop the bath from overflowing by pressing the switch corresponding to a single light on the PiGlow. Get it wrong and all the lights on that ring light up. One final hardware improvement is to simply put a square of paper over the PiGlow board; this stops the glare and makes the colours look richer.

Language
>PYTHON

CurveBall.py

```python
# Curve Ball - a game for the PiGlow board
# By Mike Cook - March 2015
import time, random, sys
from smbus import SMBus
import wiringpi2 as io

# command register addresses for the SN3218 IC used in PiGlow
CMD_ENABLE_OUTPUT = 0x00
CMD_ENABLE_LEDS = 0x13
CMD_SET_PWM_VALUES = 0x01
CMD_UPDATE = 0x16
SN3218 = 0x54 # i2c address of SN3218 IC
bus = None
try :
    io.wiringPiSetupGpio()
except :
    print"start IDLE with 'gksudo idle' from terminal"
    sys.exit()

pinList= [7,8,25] # GPIO pins for switches
lights = [0x00 for i in range(0,18)] #LED brightness list
red    = [0,6,17]  # red LEDs
orange = [1,7,16]  # orange LEDs
yellow = [2,8,15]  # yellow LEDs
green  = [3,5,13]  # green LEDs
blue   = [14,4,11] # blue LEDs
white  = [12,9,10] # white LEDs
triangleIn  = [red,orange,yellow,green,blue,white]
triangleOut = [white,blue,green,yellow,orange,red]
speed = 0.03 # delay is twice this
returnSpeed = 0.1 # for hit back
score = 0

def main():
    initGPIO()
    busInit()
    while True: # repeat forever
        wipe()
        updateLEDs(lights)
        while scanSwitches() != -1: #make sure fingers off
            pass
        pitch()

def pitch(): # throw the ball
    global score
    time.sleep(1.0) # delay before the throw # try making this random
    arm = random.randint(0,2) # direction of curve ball
    bat = False
    push = -1
    for triangle in range(0,5):
        wipe() # clear all LEDs in the list
        if bat:
            lights[white[push]] = 0x20 # turn on bat LED
        lights[triangleIn[triangle][arm]] = 0x80
        updateLEDs(lights)
        time.sleep(speed)
        if not bat: # no switch pressed yet - look for one
            push = scanSwitches() # switch pressed?
            if push != -1:
                bat = True  # no more looking at switches
                score = 6 - triangle # sooner you see # it the higher the score
        else:
            lights[white[push]] = 0x20
        updateLEDs(lights)
        time.sleep(speed)
    if arm == push:
        print "hit - score ",score
        for triangle in range(0,6):  # hit it back
            wipe()
            lights[triangleOut[triangle][arm]] = 0x80
            updateLEDs(lights)
            time.sleep(returnSpeed)
        time.sleep(0.7)

def initGPIO(): # set up the GPIO pins
    for pin in range (0,3):
        io.pinMode(pinList[pin],0) # make pin into an input
        io.pullUpDnControl(pinList[pin],2) # enable pull up

def scanSwitches(): # look at each pin in turn
    down = -1 # default return value means no switch pressed
    for pin in range (0,3):
        if io.digitalRead(pinList[pin]) == 0:
            down = pin
    return down

def busInit(): # start up the I2C bus and enable # the outputs on the SN3218
    global bus
    bus = SMBus(1)
    bus.write_byte_data(SN3218,CMD_ENABLE_OUTPUT, 0x01)
    bus.write_i2c_block_data(SN3218, CMD_ENABLE_LEDS, [0xFF, 0xFF, 0xFF])

def updateLEDs(lights): # update the LEDs to # reflect the lights list
    bus.write_i2c_block_data(SN3218, CMD_SET_PWM_VALUES, lights)
    bus.write_byte_data(SN3218,CMD_UPDATE, 0xFF)

def wipe(): # clear the lights list
    global lights
    for i in range(0,18):
        lights[i] = 0

# Main program logic:
if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        # set all the LEDs to "off" # when Ctrl+C is pressed # before exiting
        wipe()
        updateLEDs(lights)
```

NEXT MONTH

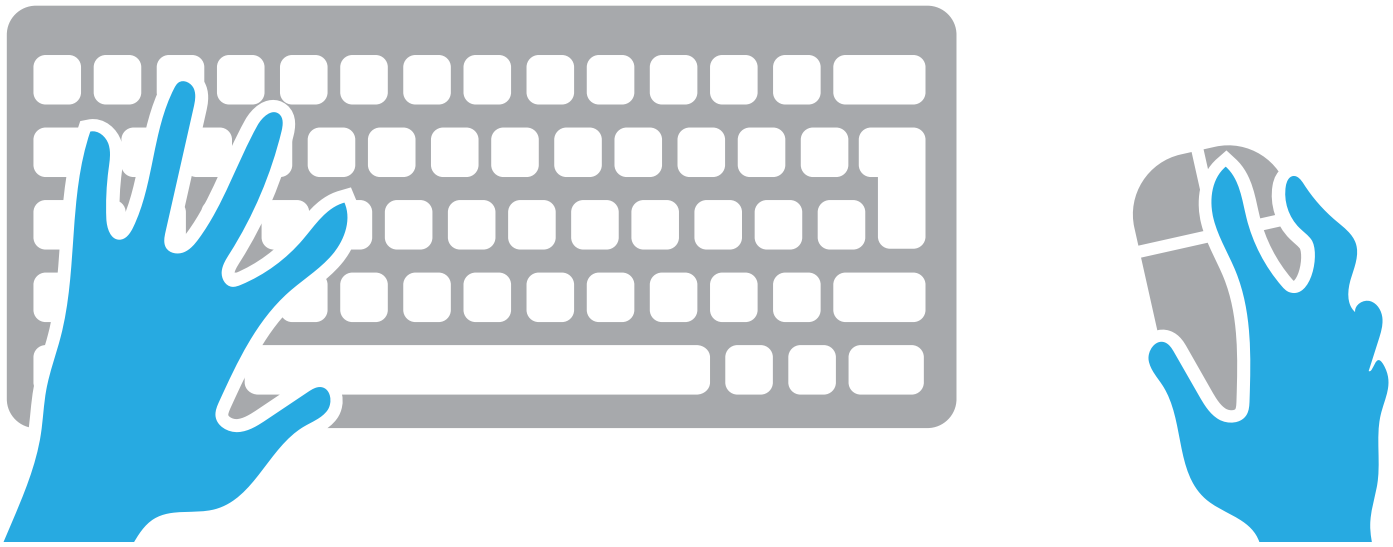
In next month's Pi Bakery, Mike looks at a novel way of detecting toy cars and builds a wildlife Safari Park game.

PART 3

MAKE GAMES WITH PYTHON

SEAN M TRACEY

Sean is a technologist living in the South West of England. He spends most of his time making silly things with technology.
sean.mtracey.org



TAKING CONTROL OF THE KEYBOARD & MOUSE

In the third part of this series, we write some code to get to grips with using our keyboard and mouse with Python and Pygame

QUICK TIP

Pygame has a set of handy built-in variables for checking which keys are pressed. We've only used a couple, but you can find the complete list at pygame.org via: bit.ly/1ycZt2i

In the first couple of tutorials, we got to grips with the core concepts of drawing and moving shapes of all types, sizes and colours with Pygame. Now that we know our way around Pygame, we're going to start making things that we can play with that are a little more interactive. This time, we're going to make two simple programs to learn how to use our keyboard and mouse. Our first program will use the keyboard; with it, we'll draw a red square and give it some code so it can move left and right and jump, which may conjure memories of a certain, heroic plumber. Our second program will use the mouse. Again, we'll create a square that we can pick up, drag around and, when we let go of our mouse button, will drop to the floor with the help of a little Pygame-programmed gravity. We're focusing on game dynamics for the time-being – we'll make prettier things later, we promise.

So, on to our first program - **keyboard.py**. Unlike last time, we're not going to be chopping and changing bits of code to affect the program. If you copy out **keyboard.py** and run it on your Raspberry Pi, it'll run just as we intend it to. This time, we're going to walk through the code line by line to understand exactly what each bit does for the program. Like a lot of things in computing, let's start at the top! The first 12 lines of code on the next spread should look pretty familiar to you by now; these are the variables we've used in the previous two parts to define how our window should look, and how we want to interact with Pygame and its methods. The next dozen or so lines are variables that we'll use to determine how our keyboard-controlled square should look and where it should be. Following that, we have two functions, **`move()`** and **`quitGame()`**, which we'll use to move and quit the game. Finally, just as in the previous tutorial, we have our main loop where we update our game and redraw all of our pixels.

What keys have we pressed?

How do we know what keys are pressed and when? In the last issue, we imported **`pygame.events`** as **`GAME_EVENTS`**; now we get to use it. Every Pygame program we write is one big loop that keeps on running forever or until we exit the program. Every time our loop runs, Pygame creates a list of events that have occurred since the last time the loop ran. This includes system events, like a **QUIT** signal; mouse events, such as a left button click; and keyboard events, like when a button is pressed or released. Once we have the list of events that Pygame received, we can decide how our program should respond to those events. If the user tried to quit, we could save the game progress and close the window rather than just exiting the program, or we could move a character every time a key has been pressed. And that's exactly what **keyboard.py** does.

On line 85, we create a **for** loop that will work through each event in the list that Pygame created for us. The events are arranged in the list in the order that Pygame received them. So, for example, if we wanted to use the keyboard events to type in our player's name, we could trust that we would get all of the letters in the right order and not just a random scramble of letters. Now that we have a list of events, we can work through them and check if certain events that are relevant to our game have happened. In **keyboard.py**, we're primarily looking for keyboard events; we can check whether or not an event is a keyboard event by checking its 'type' property with **event.type**. If our event.type is a **pygame.KEYDOWN** event, we know that a key has been pressed; if our event.type is a **pygame.KEYUP** event, we know that a key has been released. We look for **KEYDOWN** events on line 87 and **KEYUP** events on line 93. We look for **KEYDOWN** events first because logic dictates it – you've got to press a key down before it can pop back up again!

We know have a way of knowing if a key has been pressed, but how do we know which key our player pressed? Every Pygame key event has a 'key' property that describes which key it represents. If we were to print out the **event.key** property, we would see a bunch of numbers, but they aren't the keys that we pressed! The numbers we would see are key codes; they're numbers that are uniquely tied to each key on your keyboard, and programmers can use them to check which keys they represent. For example, the **ESC** key on your keyboard is 27, the **A** key is 97, and the **RETURN** key is 13. Does this mean that we have to remember a seemingly disconnected bunch of numbers when we're writing keyboard code? Fortunately, the answer is no. Pygame has a ton of values for checking key codes, which are easier to read and remember when we're writing code. On lines 89, 91, 93, and 97, we use **pygame.K_LEFT**, **pygame.K_RIGHT**, **pygame.K_UP**, and **pygame.K_ESCAPE** to check whether or not any of the key presses are keys that we're looking for.

Once we know that a key has been pressed and which key it was, we can then write code to affect our program in specific ways. For example, if the left arrow key has been pressed, we can move our player to the left with **playerX -= 5**, but we haven't done that here. Why not? Pygame doesn't emit duplicate events for key presses, so if we hold down a key to keep our square moving to the left, nothing would happen. Our square would move the first time Pygame detected the key press, but then it would stop until we pushed the button again. This is intended to help prevent situations where multiple key presses could glitch our games or give a player an unfair advantage, but it doesn't help us very much when it comes to creating games with smooth movement. So how do we get around this? Every time we detect a key press, instead of taking an action, such as moving our square, we set a variable instead. The variables **leftDown**, **rightDown**, and **haveJumped** are the variables that we can use to describe the key states (up or down) to the rest of our program. Whenever we detect that the left arrow button has been pressed, we set **leftDown** to **True**; if we detect that the left arrow button has been released, we set **leftDown** to **False**. If our player holds down the key, **leftDown** will always be **True**, so we can make our Pygame program keep moving our square smoothly across the screen, even though it's not receiving a constant barrage of events telling it to do so.

> "Once we have the list of events that Pygame received, we can decide how our program should respond to those events"

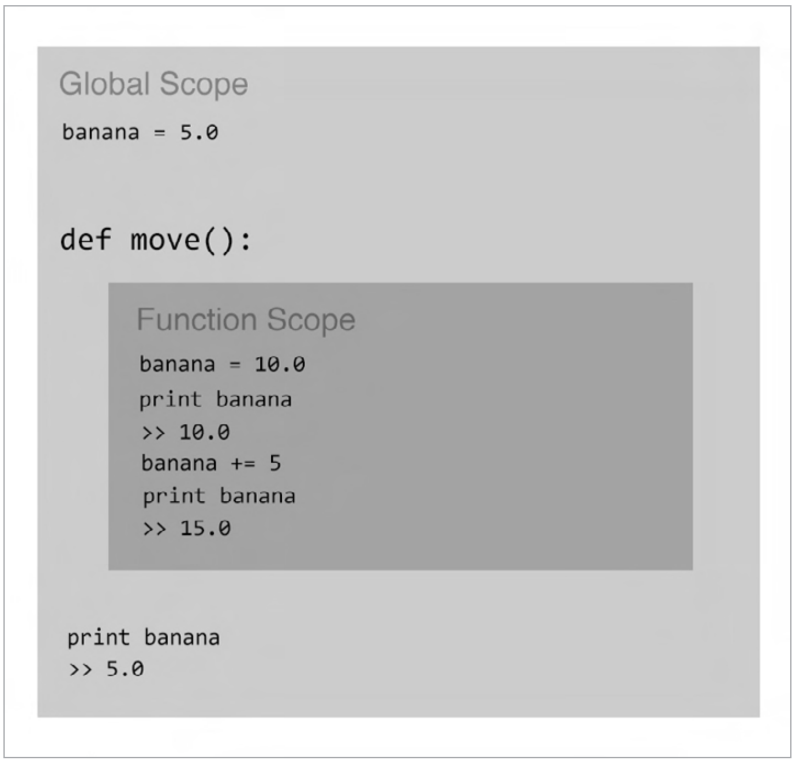
Move()

Just after our key detection code we have line 111, which simply has **move()** on it. This is a function call. Before now, almost all of the code we've written has been inside our main loop. The problem is that after a while, having every single line of code in one big loop can get a little messy and hard to follow. So, to make our lives easier, we've put the code that's responsible for making our character move into its own function, the **move** function. When we call **move()**, a lot of code then runs. So let's take a look at what's going on…

On line 31 we have a **global** statement. Because our code is inside the **move()** function, it no longer has the same scope as our **for** loop. Although we can look at the values of variables outside of our function, we can't set them, unless we include them in the **global** statement. This tells Python that when we call **playerX**, for example, we definitely mean the **playerX** at the top

QUICK TIP

The X and Y coordinates of a mouse are relative to the left and top of the window, not the screen that it's in.



```
Global Scope
banana = 5.0

def move():

    Function Scope
    banana = 10.0
    print banana
    >> 10.0
    banana += 5
    print banana
    >> 15.0

print banana
>> 5.0
```

Above A basic illustration of code scope

of the file – not a new **playerX** that we might create within the function.

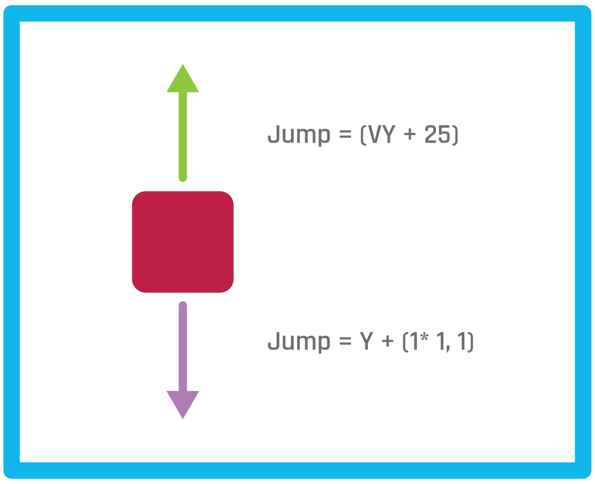
Lines 34 to 50 are where we make our little square move left or right, depending on the buttons that have been pressed. If the left arrow button is down, we want to move the square/character/object to the left. This is what we're doing between lines 36 and 41. To do this convincingly, we first need to check whether or not our square is moving already and the direction it's going in. If our square is already travelling to the right, we need to make it stop and then change direction. Think about it: if you're running in a straight line, you can't turn right around and keep running at the same speed – you need to stop, turn, and then build the speed up again. Line 37 checks whether our square's X velocity is over 0.0 (going to the right). If it's not, then we either don't need to move at all, or we're already moving to the left, so we can just keep on moving. But if we are moving to the right, setting **playerVX** to **moveSpeed** and then inverting it will stop our square and send it in the correct direction. We don't want our square to run off the screen either; lines 40 and 41 stop our square moving if it's at the left edge of our screen. Lines 44-50 do the exact same thing but in reverse.

Lines 52-70 are a little different. It's here that we add gravity to our square's movement. When we hit the up arrow on our keyboard, our box jumps, but what goes up must come down. Just like when we change direction when we run, we need to slow down after jumping before we start to fall back down again. That's what's going on here. First, on line 52, we check to see whether our square is travelling upwards at a speed greater than 1 pixel per frame. If it is, then we multiply that value by 0.9 so it will eventually come to a point where it is travelling less than 1 pixel per second; when that happens, we set the value to 0 so that we can start falling back to the bottom of the screen. Next, our code checks whether our square is in the air – it can't stay up there all day! On lines 59-61, we check that the square is in the air and then start adding the **gravity** value to the **playerVY** value; this will make our square move back down to the bottom of the screen. Each time we add the **gravity** value to the **playerVY** value, we multiply the former by 1.1; this makes the square speed up as it falls back to the bottom of the screen, just like it would if you threw a ball in the air. Lines 63-64 reset the **gravity** and **playerVY** values when the bottom of the square hits the bottom of the screen. Lines 68-70 are fun, in that they stop the square from moving any faster left or right once our square has jumped in the air. You can't change direction after you jump; you can only change direction when you hit the ground again, so that's what our square does too.

Pygame mouse events

That's enough of the keyboard for now; it's time for the mouse to shine. The mouse is a simple bit of kit, so the code for it is far less complicated than our keyboard code. If you copy out **mouse.py** and run it, you'll see a familiar fetching red square sitting at the bottom of the screen. Pressing your keyboard keys will do nothing this time, for this square is different. If you want to move it, you've got to pick it up! Drag your mouse over the square, hold down the left mouse button and drag up. Our square moves with our mouse. If you let go of your mouse button, the square will fall back to the bottom of the window. Nice and simple, but how does it work?

This time, we have hardly any code at all in our main **for** loop. Here we're only checking whether or not the first mouse button has been pressed and then we call three functions: **checkBounds()**, **checkGravity()**, and **drawSquare()**. In our **keyboard.py** code, we put some of our code into functions; this time we're doing it to all of them, but we'll get to those in a bit.



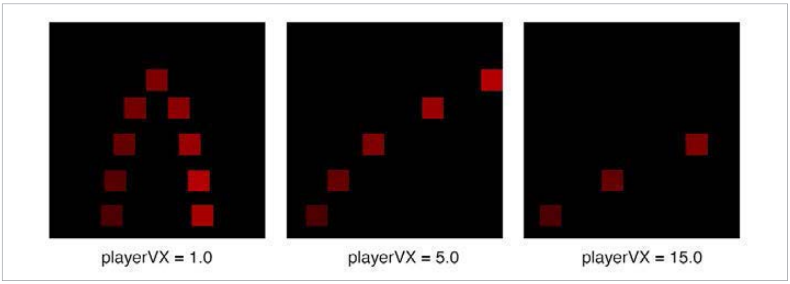
Above An example of gravity working against a Y velocity

Keyboard.py

Language

>PYTHON

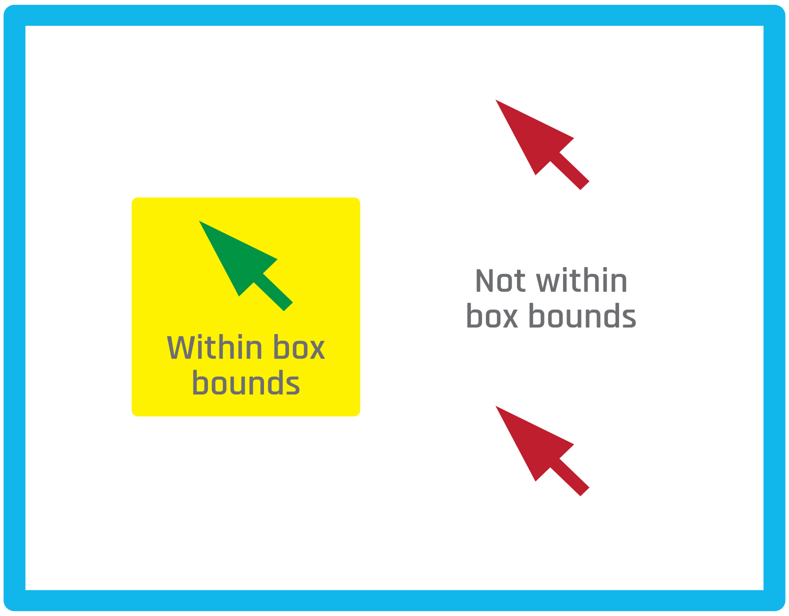
```python
import pygame, sys
import pygame.locals as GAME_GLOBALS
import pygame.event as GAME_EVENTS

# Pygame Variables
pygame.init()

windowWidth = 800
windowHeight = 800

surface = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption('Pygame Keyboard!')

# Square Variables
playerSize = 20
playerX = (windowWidth / 2) - (playerSize / 2)
playerY = windowHeight - playerSize
playerVX = 1.0
playerVY = 0.0
jumpHeight = 25.0
moveSpeed = 1.0
maxSpeed = 10.0
gravity = 1.0

# Keyboard Variables
leftDown = False
rightDown = False
haveJumped = False

def move():
    global playerX, playerY, playerVX, playerVY, haveJumped, gravity

    # Move left
    if leftDown:
        #If we're already moving to the right, reset the
        # moving speed and invert the direction
        if playerVX > 0.0:
            playerVX = moveSpeed
            playerVX = -playerVX
        # Make sure our square doesn't leave our
        # window to the left
        if playerX > 0:
        playerX += playerVX

    # Move right
    if rightDown:
        # If we're already moving to the left reset
        # the moving speed again
        if playerVX < 0.0:
            playerVX = moveSpeed
        # Make sure our square doesn't leave our
        # window to the right
        if playerX + playerSize < windowWidth:
            playerX += playerVX

    if playerVY > 1.0:
        playerVY = playerVY * 0.9
    else:
        playerVY = 0.0
        haveJumped = False

  # Is our square in the air? Better add some gravity
  # to bring it back down!
    if playerY < windowHeight - playerSize:
        playerY += gravity
        gravity = gravity * 1.1
    else:
        playerY = windowHeight - playerSize
        gravity = 1.0

    playerY -= playerVY

    if (playerVX > 0.0 and playerVX < maxSpeed) or (playerVX < 0.0 and playerVX > -maxSpeed):
        if not haveJumped and (leftDown or rightDown)
            playerVX = playerVX * 1.1

# How to quit our program
def quitGame():
    pygame.quit()
    sys.exit()

while True:

    surface.fill((0,0,0))

    pygame.draw.rect(surface, (255,0,0),
(playerX, playerY, playerSize, playerSize))

    # Get a list of all events that happened since
    # the last redraw
    for event in GAME_EVENTS.get():

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K_LEFT:
                leftDown = True
            if event.key == pygame.K_RIGHT:
                rightDown = True
            if event.key == pygame.K_UP:
                if not haveJumped:
                    haveJumped = True
                    playerVY += jumpHeight
            if event.key == pygame.K_ESCAPE:
                quitGame()

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT:
                leftDown = False
                playerVX = moveSpeed
            if event.key == pygame.K_RIGHT:
                rightDown = False
                playerVX = moveSpeed

        if event.type == GAME_GLOBALS.QUIT:
            quitGame()

    move()

    pygame.display.update()
```



Above A demonstration of the varying effects of the X velocity when jumping

The two important things we need to know when using a mouse are where it is and which buttons, if any, have been pressed. Once we know these two things, we can make stuff do stuff, and we all love making stuff do stuff. First, we're going to find out where the mouse is, and we do that on line 76 with **`pygame.mouse.get_pos()`**. Unlike our keyboard, we don't have to work through a list of events and check whether they were mouse events. Instead, when we call **`pygame.mouse.get_pos()`** we get a tuple back with two values: the current X and Y value of the mouse inside the window. Great, we know where the mouse is; all we need to know now is whether or not any of the buttons have been pressed and we do this on line 81. **`pygame.mouse.get_pressed()`** returns a tuple of three values: the first is for the left mouse button, the second for the middle mouse button, and the third for the right mouse button. If the button is pressed down, then the value is **`True`**, otherwise it's **`False`**. We're not doing anything with the middle or right mouse button, so we can simply check the first value (the left mouse button) with **`pygame.mouse.get_pressed()[0]`**. Now, if **`pygame.mouse.get_pressed()[0]`** is **`True`**, then our player has clicked a button and we can do stuff. In this case we set **`mousePressed`** to **`True`**, just as we did with **`leftDown`** and **`rightDown`** in **keyboard.py**, so we can use it throughout our program.

Below An illustration of checking the box bounds against the cursor coordinates



Checking the square

Now that we know where our mouse is and which buttons are pressed, we can do something with that information. Straight after our code that checks our mouse buttons, we call **`checkBounds()`** on line 86. **`checkBounds()`** has one job: to check whether or not our mouse position is within the bounds (edges) of our square. If we were making a fully fledged game, this function would likely check the position of every game object against the mouse coordinates, but in this example we're only looking at our lovely red square. Line 31 checks whether or not our mouse button has been pressed – after all, there's no point in checking where our mouse is if it's not doing anything. If our mouse button has been pressed, on line 33 we look at where the X coordinate of the mouse is and compare it to the X coordinate of our square. If our mouse X is greater than the left of our square and is smaller than the X value of the right of our square (**`squareX + squareSize`**), we know that our mouse is within the X bounds of our square, but that doesn't mean that it's inside our shape. Before we do anything with our mouse, we need to check that the Y coordinate of our mouse is within our square too, which we do on line 35. If the Y value of our mouse is greater than the top of our shape and less than the bottom of it, then we can be certain that our mouse is somewhere inside of our shape. In **mouse.py**, we've checked the X coordinates and Y coordinates on separate lines – we could have done it on one, but it's quite an intimidating line to read, let alone write! Now that we know our mouse is positioned within our square and that we've pressed our mouse button, we can set our **`draggingSquare`** variable to **`True`**.

Once **`checkBounds()`** has done its job, **`checkGravity()`** gets to work. Just like in **keyboard.py**, **`checkGravity()`** looks at where our square is in the window and if it's not on the bottom of our window, it will accelerate our square to the bottom. However, it will only do this if we've let go of our mouse button, because we don't want our shape to fall to the ground when we're holding onto it.

Our final function is **`drawSquare()`** – five Pygame points if you can guess what it does... Based on the adjustments of **`checkBounds()`** and **`checkGravity()`**, **`drawSquare()`** will draw our square (did you manage to guess?). If our square is being dragged around by our mouse, it will draw the square at the mouse coordinates. But if we aren't dragging it around, it will draw a graceful gravity-driven descent back to the bottom of our window. **`drawSquare()`** has one little trick up its sleeve: as well as affecting the position of our square, it also changes its colour – red when not being dragged and green when being dragged. This code could be useful if we had a little character and we wanted to change his graphic to make it look like he was holding onto our cursor.

Mouse.py

Language
>PYTHON

```python
import pygame, sys
import pygame.locals as GAME_GLOBALS
import pygame.event as GAME_EVENTS

# Pygame Variables
pygame.init()

windowWidth = 800
windowHeight = 800

surface = pygame.display.set_mode((windowWidth, windowHeight))

pygame.display.set_caption('Pygame Mouse!')

# Mouse Variables
mousePosition = None
mousePressed = False

# Square Variables
squareSize = 40
squareColor = (255, 0, 0)
squareX = windowWidth / 2
squareY = windowHeight - squareSize
draggingSquare = False
gravity = 5.0

def checkBounds():

    global squareColor, squareX, squareY, draggingSquare

    if mousePressed == True:
        # Is our cursor over our square?
        if mousePosition[0] > squareX and mousePosition[0] < squareX + squareSize:

            if mousePosition[1] > squareY and mousePosition[1] < squareY + squareSize:

                draggingSquare = True
                pygame.mouse.set_visible(0)

    else:
        squareColor = (255,0,0)
        pygame.mouse.set_visible(1)
        draggingSquare = False

def checkGravity():

    global gravity, squareY, squareSize, windowHeight

    # Is our square in the air and have we let go of it?
    if squareY < windowHeight - squareSize and mousePressed == False:
        squareY += gravity
        gravity = gravity * 1.1
    else:
        squareY = windowHeight - squareSize
        gravity = 5.0

def drawSquare():

    global squareColor, squareX, squareY, draggingSquare

    if draggingSquare == True:

        squareColor = (0, 255, 0)
        squareX = mousePosition[0] - squareSize / 2
        squareY = mousePosition[1] - squareSize / 2

    pygame.draw.rect(surface, squareColor, (squareX, squareY, squareSize, squareSize))

# How to quit our program
def quitGame():
    pygame.quit()
    sys.exit()

while True:

    mousePosition = pygame.mouse.get_pos()

    surface.fill((0,0,0))

    # Check whether mouse is pressed down
    if pygame.mouse.get_pressed()[0] == True:
        mousePressed = True
    else:
        mousePressed = False

    checkBounds()
    checkGravity()
    drawSquare()

    pygame.display.update()

    for event in GAME_EVENTS.get():

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                quitGame()

            if event.type == GAME_GLOBALS.QUIT:
                quitGame()
```

WHAT WE'VE LEARNED

We've learned that Pygame creates a list of events that occurred every time the frame is updated, and that we can work through them to check for events that we want to use. We learned that Pygame receives key codes when buttons are pressed, but has a big list of key code events that we can use so we don't have to remember all of the numbers. We learned that we can get mouse events whenever we like, and that we can get coordinates of where the mouse is and which buttons are pressed. We've learned how to simulate gravity and jumping, and we've made ourselves think about how things move in the real world too. Congratulations, we now have the beginnings of a real game.

RICHARD WATERWORTH

Richard Waterworth is a 15-year-old blogger and video maker who loves tinkering with gadgets (and taking really good pictures like these).
richardtech.net

RASPBERRY PI CASE SUPER-TEST

Raspberry Pi blogger **Richard Waterworth** compares four of our favourite B+ and Raspberry Pi 2 Model B-compatible cases to see which one comes out on top...

pi-supply.com

£9 / $14

SHORT CRUST PLUS

The Short Crust Plus is a fairly minimal Raspberry Pi case. It comes in two different colour options for the base (black and white), and this allows you to alter the overall look. The top of the case is a smooth glossy finish, which can turn into a fingerprint magnet, but it's easily cleaned. The main base of the case is a rougher matte finish, which compliments the top perfectly.

The Short Crust Plus offers plenty of ventilation on the bottom of the case, and the release trigger (also found underneath) allows you to easily remove your Pi. In terms of cost, the Short Crust is reasonable considering the high-quality plastic used. It also comes with non-slip rubber feet and screw holes in the base, should you wish to mount it under a desk or on a wall.

If you're looking for a cleanly designed, modern-looking case with all the mod-cons, you should definitely consider the Short Crust Plus.

Score ★★★★★

modmypi.com

£5 / $8

HELIX CASE

The Helix is no ordinary Raspberry Pi case. It's made from MDF and features a flexible 'shell' around its core. The quality of the case is impressive for the price. However, if you are disassembling it you'll need to be careful, as some elements of the case are quite fragile. Once you've installed your Pi in the Helix, though, it looks great, and quite unlike the vast majority of cases on offer elsewhere.

Inside the case, there are some supports that your Raspberry Pi sits on, but there's nothing on the top of the Pi to hold it securely in place. Given the asking price (it's the cheapest on test), we can't be too harsh, but we can't help but think if the pins holding the 'shell' of the case were a bit bigger and there were a few more supports to hold the Pi, this could be a really great case.

The ModMyPi Helix might not be for everyone, but if you want to set yourself apart and don't have a great deal of money to spend, this is definitely the case for you.

Score ★★★☆☆

BEST VALUE

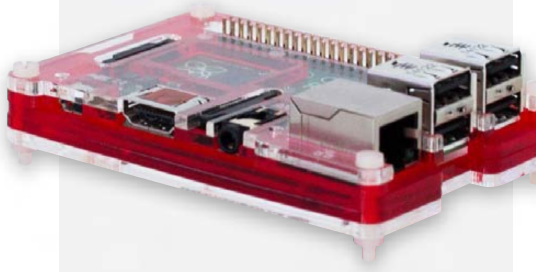
pimoroni.com

£8.50 / $12

PIBOW COUPÉ

The Pibow Coupé offers a very slimline design. Unlike conventional cases, the Coupé does not fully cover the entire Pi, only its sides and bottom, allowing the USB and Ethernet ports to protrude from the top of the case. In typical Pimoroni style, it's constructed from thin layers of plastic and comes in a variety of colours.

While we weren't sure about the look of Coupé at first, it quickly grew on us during testing, though we wouldn't use it for a home theatre setup. For us it'd be much more at home as a second computer or slimline project case, though you might disagree. Most importantly, however, the Coupé provides easy access to all ports, which is especially useful for people who will be using the Camera module or tinkering with components attached to the GPIO ports.



The Pibow Coupé has such a distinctive look, you'll probably love it or hate it. Either way, it's a great case for people who want to use their Raspberry Pi for hardware projects.

Score ★★★★☆

thepihut.com

£13 / $20

FLIRC PI CASE

The Flirc Case has a very solid construction as it's made from a solid aluminium shell with a matte, rubberised plastic top and base. The aluminium finish gives the case a premium feel, too, which is always a bonus. Unlike the other cases on test, it comes with a built-in heatsink. It's easy to put together, using four screws to secure it.

It's certainly not as prone to fingerprints as the Short Crust Plus, but it does smudge, which is only really noticeable up close. The Flirc is primarily advertised as a home theatre case for the Raspberry Pi, but it's likely to be at home in other environments, too. It's the most expensive case on test but if you can afford it, the cost is easily justified considering the materials used in its construction.

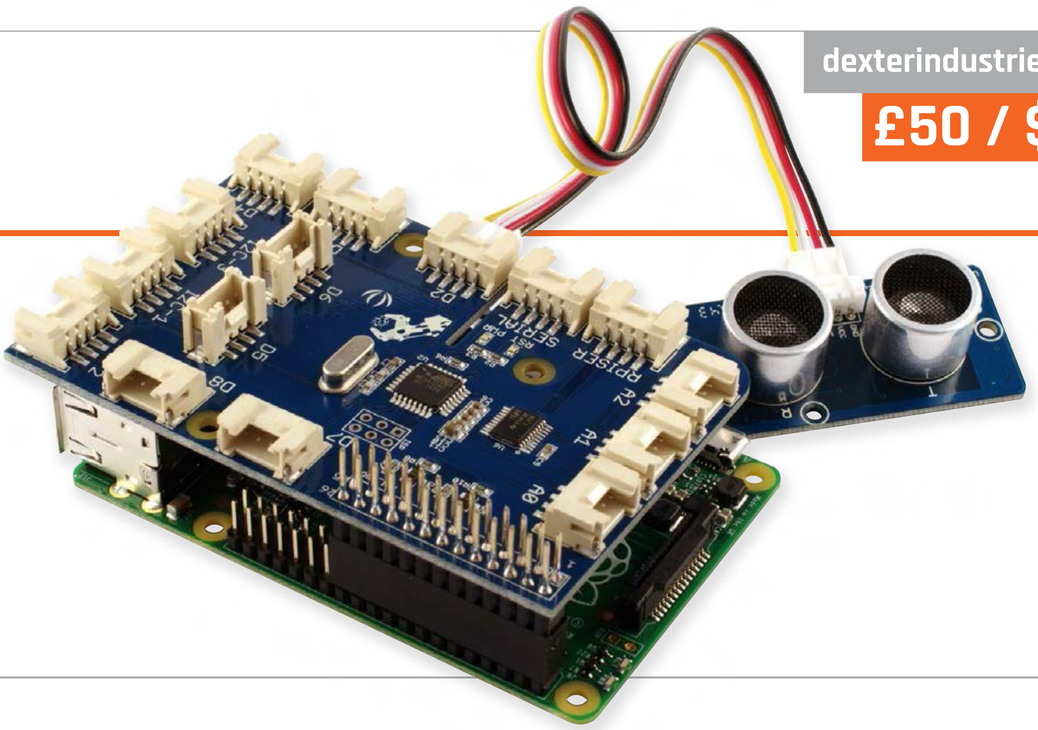
RUNNER-UP

The Flirc is a very clean and simply designed case for your Raspberry Pi. It looks sleek and feels premium, thanks to its aluminium finish.

Score ★★★★☆

dexterindustries.com

£50 / $80



Maker Says

“Bring your Raspberry Pi into the physical world

Seeed

GROVEPI+ STARTER KIT

Gareth Halfacree finds out if the GrovePi+ kit really can make experimentation as simple as plug-and-play for electronics newcomers

The GrovePi+ is positioned as a kit that makes it as easy as possible to get started with electronics projects, and yet offers considerable flexibility and power. Packaged in the same eye-catching green carry-case as Seeed Studio’s other Grove kits, the GrovePi+ includes the main board plus 12 modules: sensors for sound, light, angle, distance, and combined temperature and humidity; red, blue, and green LEDs; buzzer; character-based LCD display with RGB LED backlight; button; and relay.

Fit to burst

It’s pretty packed, in other words, and the small amount of free space in the box is taken up with a handy 30-page guidebook. Printed in full colour, the booklet walks you through getting started and includes several example projects using the bundled hardware. These range from making an LED blink, through to posting multiple sensor readouts to Twitter, and are a good source of inspiration.

The GrovePi+ board itself can be thought of as a customised Arduino, using the same ATmega328 microcontroller to provide capabilities beyond a normal Raspberry Pi: three analogue inputs, seven digital input/outputs, three I²C connections, and serial connectivity to both the microcontroller and the Pi itself. The GPIO pins are even brought through and free for use, although these are limited to the 28 pins of the original design rather than the extended 40-pin header of the Plus or Raspberry Pi 2 models, with which the GrovePi+ is fully compatible.

Clever script

Installation is relatively straightforward, using either a Dexter Industries customised Raspbian download or a two-step customisation process for existing Raspbian installations. A clever script takes care of the more complicated steps – such as enabling I²C communication and disabling the Linux serial console – although during testing, it failed to register the GrovePi Python library correctly, requiring a final manual installation step that the documentation did not mention.

Once installed, a wealth of Python examples are provided. These range from simple programs to make a buzzer sound, through to the more complicated projects detailed in the bundled manual. Sadly, these stand alone: despite claims of support for the C language, no library or examples are yet available, with the project’s maintainers citing a lack of demand.

Related

ARDX ARDUINO STARTER KIT

A fully featured starter kit for the Arduino Uno, complete with numerous well-documented projects, the ARDX can be a handy accessory for any computer – including the Raspberry Pi.

£62 / $85

oomlout.co.uk

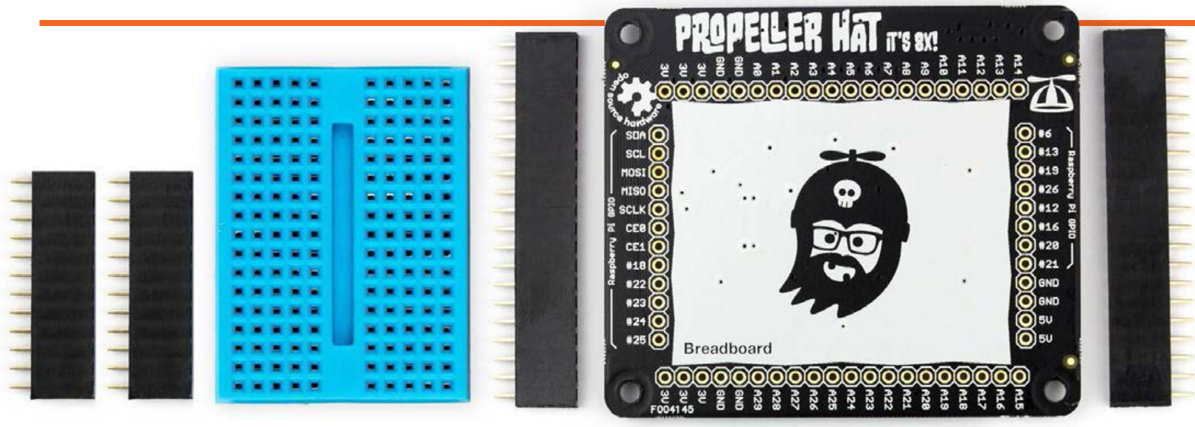
Last word

If you’re a Python fan looking to branch into electronics, the GrovePi+ is a great choice and won’t rob you of all your GPIO headers. The team has, however, failed to deliver on its C promise.

★★★★☆

shop.pimoroni.com

£15 / $23



Maker Says

It will get you flying in the world of micro-controllers

Pimoroni

PROPELLER HAT

Les Pounder delves into the world of multi-core micro controllers, courtesy of Pimoroni's latest board

Quite often, electronics projects require split-second precision. Since this isn't the Raspberry Pi's forte, you'd normally use a small microcontroller, like an Arduino or the Parallax Propeller.

With the Propeller HAT, Pimoroni has placed the Parallax microcontroller on a HAT add-on board, along with a handy 170-point breadboard to help you rapidly prototype projects. The Propeller HAT – a completely open-source hardware and software platform, based upon the Parallax Propeller 1 P8X32A – comes partially built, requiring a little soldering to populate the pin headers that you can see around the board. In total there are 30 pins, labelled A0 to A29. To the left and right of the board, you can see some of the standard Raspberry Pi GPIO pins broken out for use too, including pins for I²C and SPI.

SPIN it

The Parallax Propeller at the heart of the Propeller HAT can be programmed using the PropellerIDE, which creates files in a language called SPIN. Once you have written your project, it is a simple matter of clicking 'Run' in the IDE to create a binary file and then uploading the file to the board. It's a similar process to the Arduino, where code is compiled and then uploaded to be run. Binary files can also be uploaded to the board using the p1load command in the terminal, enabling fast upload of pre-compiled projects. To round off the software, there's also an excellent Python library that enables you to create scripts to control the Propeller HAT with Python.

Elsewhere, the Propeller HAT can emulate the SID chip, commonly found in Commodore 64 and 128 8-bit machines from the 1980s, enabling your projects to output SID music via a 3.5mm jack attached to the breadboard.

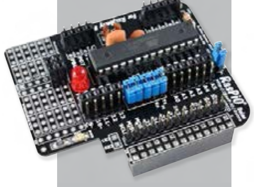
Is it right for you?

You might be wondering who this board is for. If you are just starting your journey into the world of hardware hacking and making, then this is not for you. But if you are an experienced hacker, this is a great place to learn more about multi-core microcontrollers, and generally have a lot of fun hacking and making on your Raspberry Pi.

Related

RASPIO DUINO

We awarded this four stars in issue 31. The Duino merges Arduino and Raspberry Pi worlds into one handy platform and is useful for a multitude of projects.



£12 / $18

rasp.io/duino

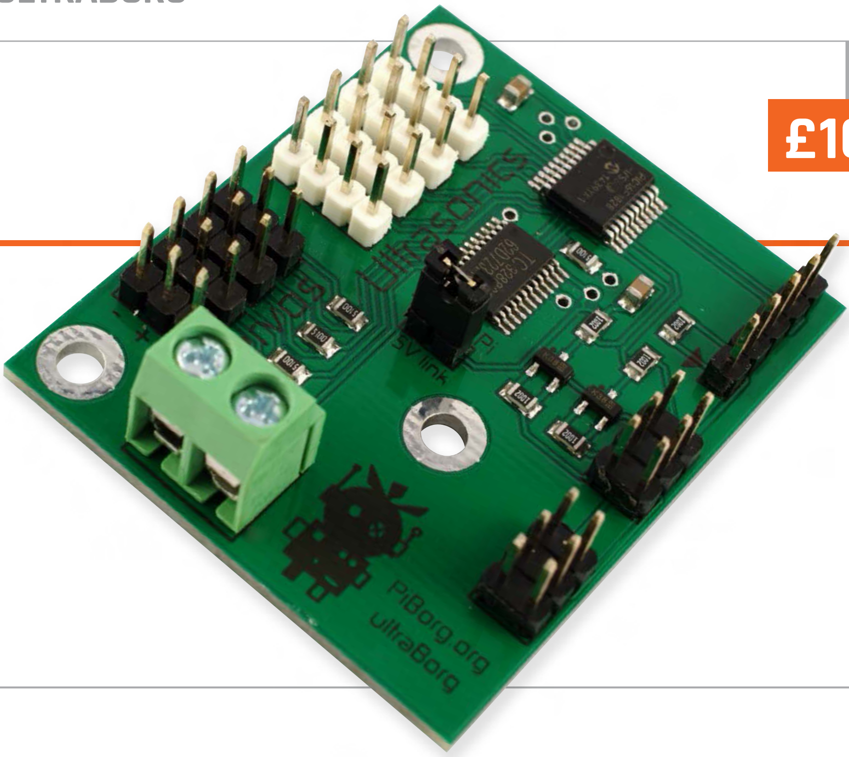
Last word

It's not a board for those new to hardware projects with the Raspberry Pi, but rather a board for those with good experience with electronics and microcontrollers.

★★★★☆

piborg.org

£16 / $25



Maker Says

Precision servo control with ultrasonic module support
PiBorg

PIBORG ULTRABORG

A combined ultrasonic sensor and servo driving board, is the UltraBorg the all-in-one add-on which roboticists have been craving? **Gareth Halfacree** finds out…

Designed with the Raspberry Pi in mind, but compatible with any microcomputer or microcontroller that can talk I²C, the UltraBorg is aimed at robotics enthusiasts. Built to simplify the building of servo-based robots, the diminutive board provides support for two device types: servos and ultrasonic sensors.

For servos, it provides four channels of 16-bit control on a bank of triple-pin headers on its left. This resolution, provided by a Toshiba pulse-width modulation (PWM) chip and considerably higher than the 12-bit found on most rival devices, indicates that PiBorg has really thought its design through; this is confirmed by the ability to save a startup position, as well as maximum and minimum limits in the controller.

The ultrasonic portion of the board is, likewise, four-channel, supporting the four-pin modules common to most robot kits and hobby supply shops. All functionality is controlled through an on-board PIC chip, which takes the pressure off the Raspberry Pi's processor and allows for accurate real-time control.

Easy to install

Installation of the board is straightforward using a simple software installation script available from PiBorg's website, although you'll need to provide your own 5V power supply. The board's mounting holes are designed to allow it to be positioned in a variety of orientations, including sharing the Pi A+/B+ and Pi 2's mounting holes and piggybacking above to save space. The UltraBorg takes up the first six pins on the GPIO header, leaving the rest free – and while the limit of just four servos may seem troubling, it's possible to daisy-chain multiple UltraBorgs together to support as many ultrasonic sensors and servos as your project requires.

PiBorg has even thought to include a simple GUI. This makes tuning the servo limits as simple as possible, with sliders to adjust the positions, and buttons to save a startup position and set max/min rotation limits; these are saved to the PIC processor's EEPROM, which survives power cycles.

The GUI is joined by a basic demonstration program, showing sliders for all four servo channels, along with distance reports from the four ultrasonic channels. Elsewhere, the included Python examples make it easy for new users to get started.

Related

ADAFRUIT 16-CHANNEL PWM/ SERVO HAT MINI KIT

Provided as a simple soldering kit, this Adafruit HAT may not offer an easy way to interface with ultrasonic sensors, but can control up to 16 PWM devices – including servos.

£16 / $17.50

adafruit.com

Last word

The UltraBorg is a great choice for robotics projects, offering features rarely seen on servo control boards, but its four-channel limitation – overcome through daisy-chaining – means it can get expensive for more complex projects.

★★★★☆

thepihut.com

£7 / $10

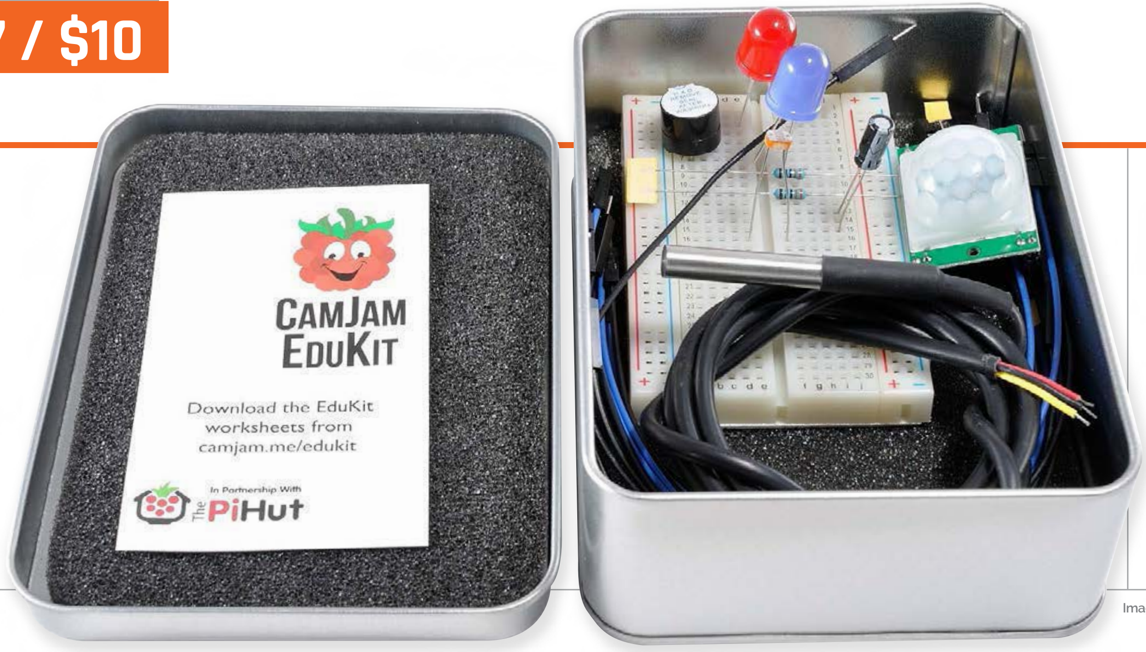

CamJam EduKit
Download the EduKit worksheets from camjam.me/edukit
In Partnership With The PiHut

Image: Alex Eames. www.RasPi.TV

Maker Says

The latest addition to the EduKit family!

CamJam

CAMJAM EDUKIT 2

A tin full of tinkering kit, for less than a tenner. **Les Pounder** investigates CamJam's latest box of tricks...

The EduKit is a pocket-money project box that's compatible with all models of Raspberry Pi. Rather than produce an add-on board, the CamJam team have packaged all the components needed to build a number of physical computing projects into a rather handy tin.

A follow-up to CamJam's original EduKit, the second edition is entitled 'Sensors' and contains a lot of electronic components – enough to make six starter projects. In the kit you'll find LEDs, buzzers, wires, and a breadboard – as we did in the first EduKit – but there's also a light-dependent resistor (LDR), passive infrared sensor (PIR), and a DS18B20 temperature sensor inside.

Worksheets

So what can you make with these components? Using the six downloadable worksheets, you can start by controlling the LED and buzzer with some simple Python code. The worksheets also cover creating an alarm based on the PIR sensor, which can detect movement; you could, for example, trigger a sequence of LED flashes and buzzer sounds. One of the more challenging worksheets uses the DS18B20 sensor to read ambient temperature, but it can also be safely placed into liquid for accurate measurements, perhaps a great cross-curricular activity for introducing the Pi into a science lab. Another worksheet focuses on the LDR sensor to measure the light in a room. Since it produces analogue values (something that the Raspberry Pi can't process), the EduKit 2 includes a capacitor, which will be charged and timed using Python code to get the readings. The charge time is dependent on the flow of energy controlled by the LDR, thus giving the user an estimated analogue value. It's nicely done.

Cost-effective

This kit and its predecessor are very cost-effective and well-supported starting points for anyone interested in getting started with the GPIO pins on the Raspberry Pi. While there are plenty of practical project examples included with the kit, they can be applied to countless other projects and ideas and even used with different electronics platforms. The accompanying online worksheets are excellent quality, and provide solid instructions that get progressively more challenging as you learn.

Last word

A very high-quality, yet cost-effective starting point for many physical computing projects. An essential purchase for home and school learners.

★★★★★

Related

EXPLORER HAT PRO

A more expensive solution than the EduKit. However, with analogue inputs, capacitive touch sensors, and a motor controller featuring an H Bridge, this is a great add-on board for many project types.

£18 / $22.95

shop.pimoroni.com

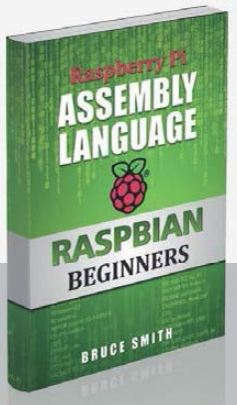
RASPBERRY PI BESTSELLERS

ASSEMBLER PROGRAMMING

ARM is a far better platform for learning assembly language coding than x86 – here are the current top three guides...

RASPBERRY PI ASSEMBLY LANGUAGE

Author: Bruce Smith
Publisher: CreateSpace
Price: £14.99
ISBN: 978-1492135289
tinyurl.com/q6tokqb



A well-paced, example-led introduction to assembly language programming on ARM, from the comfort of Raspbian. The best beginner's book – but read the errata on the website.

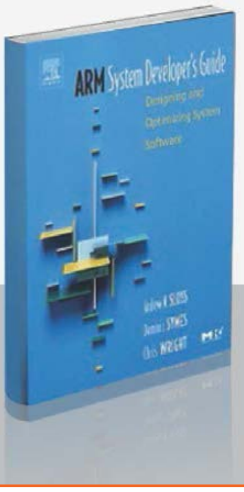
BAKING PI: OPERATING SYSTEMS DEVELOPMENT!

Authors: Alex Chadwick
Publisher: University of Cambridge Computer Lab
Price: free online
ISBN: N/A
tinyurl.com/k4pd38p

Popular online course which "takes you through the basics of operating systems development in assembly code", and starts with controlling the GPIO pins directly. Still mostly works on newer Pis.

ARM SYSTEM DEVELOPER'S GUIDE

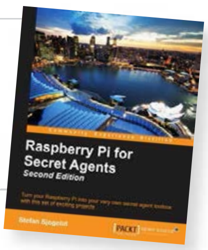
Authors: Andrew Sloss, Dominic Symes & Chris Wright
Publisher: Morgan Kaufmann
Price: £57.99
ISBN: 978-1558608740
tinyurl.com/mk588lq



Detailed guide to the ARM instruction set, popular among embedded ARM developers for more than a decade, and still a great introduction to writing efficient C and assembler code for the architecture.

RASPBERRY PI FOR SECRET AGENTS 2ND EDITION

Author: Stefan Sjogelid
Publisher: Packt
Price: £15.99
ISBN: 978-1784397906
tinyurl.com/oknx38x



Many spy film gadgets that were once out of reach can now be built cheaply with a Raspberry Pi. There's no need for a Pi 2 here, but many of the pranks rely on a camera and/or microphone, while some require a Wi-Fi module, GPS, and battery pack. The reason for all these accessories? A multitude of pranks and secret-agent ways of using your Raspberry Pi around the home and beyond.

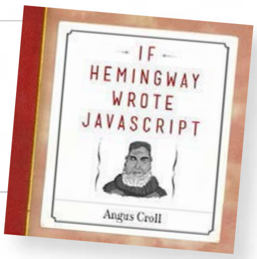
The first chapter introduces SSH, as it's a lot easier to hide a Pi alone than with a monitor. The Audio Antics chapter brings in ALSA and Sound eXchange (SoX), alias, tmux, scripting, and scheduling – while using the Pi for bugs, calls, and voice distortion. Using the camera, there's motion detection and capture, plus getting the Pi to turn on in the middle of the night to scare the unwary. A Networking chapter includes man-in-the-middle attacks and plenty of useful Linux info. 'Taking Your Pi off-road' encompasses battery packs, GPS, and data encryption.

This book of tricks is a great driver to learn new skills – after all, there are few better motivators embedded in the human psyche than the desire to wind people up, or have a laugh at the expense of those near and dear to you. Good fun and, like so many Pi books, good value too.

Score ★★★★★

IF HEMINGWAY WROTE JAVASCRIPT

Author: Angus Croll
Publisher: No Starch
Price: £13.50
ISBN: 978-1593275853
nostarch.com/hemingway



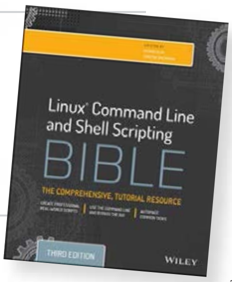
A valuable counterweight to the excellent but prescriptive Crockford guide to JS [see sidebar on next page]. JavaScript is an expressive language, but it's easy to forget how far you can push the limits until you see a set of examples like this. From Joyce to Woolf, 25 authors are given new voices as putative JS developers on five problems – from Fibonacci to finding prime numbers – separated by poetic interludes.

Each snippet is both entertaining and instructive. While languages like Perl are notorious for 'there's always more than one way to do it' unpredictability, JS has enough regularity to cope with someone moving the boundaries about. As pure entertainment, this is a treat to read, but Croll isn't slow to underline the persuasive calls of different approaches to problem solving.

Complemented by Miran Lipovača's illustrations, these portraits of the artist as a coder throw up countless gems, from memorable variable names to insightful snippets into life and literature. The contrast as we leap from Italo Calvino to J K Rowling is a particular delight, but it's hard to imagine this book without any of the 25 entries – even Dan Brown. The only downside is the way you find yourself, after reading other novels, now wondering about how each and every author would approach problems with JavaScript.

Score ★★★★★

LINUX COMMAND LINE & SHELL SCRIPTING BIBLE



Author: Richard Blum & Christine Bresnahan
Publisher: Wiley
Price: £33.99
ISBN: 978-1118983843
tinyurl.com/p7hcmgb

Command line knowledge isn't just about automating tasks for busy Linux systems administrators – a deeper Linux understanding will enable you to get far more out of Raspbian and your Pi. Although comprehensive and detailed, this guide assumes no prior Linux knowledge and explains much of what's happening internally, making a useful tutorial, as well as reference, for the non-GUI world.

The first section, on the command line, covers similar areas to *The MagPi*'s own command line introductory series, but in much greater detail. The remaining three sections cover shell scripts in progressively greater, then more practical, detail. While shell scripting will never match Python, say, for larger programs, here you'll find inspiration for getting Raspbian (and other Linux systems) to create useful utilities, and even to build database-driven code that interacts with your email account and collects web data. So it should prove an inspiration as well as a useful reference.

GNU/Linux doesn't need 750-page books because it's complicated, but because it's powerful. Very powerful. You can live happily with Raspbian without knowing three-quarters of what Blum and Bresnahan examine here, but this is a great collection of Linux knowledge when you do need it.

Score ★★★★☆

PROGRAMMING ELIXIR

Author: Dave Thomas
Publisher: Pragmatic
Price: £23.99
ISBN: 978-1937785581
tinyurl.com/o3r7yrl

Elixir runs on the Erlang VM and is a functional programming language that's great for concurrency. It needs a more recent Erlang version than the one in Raspbian, but installing one is easy enough – or SD card images are readily available from someone who's already done the job. That done, you want a quick way to dive in and learn: enter Dave 'Ruby Pickaxe' Thomas's practical introduction.

Aimed at existing coders, the book dives straight into functional programming ("programming should be about transforming data") and the power of Elixir's concurrency – a benefit of the Erlang virtual machine. Elixir's easy syntax, powerful macros, and standard library will get you up and running so quickly, you may not notice that Thomas's well-structured book is carrying you so far through the learning experience.

The real strength of this work is in shifting how you approach coding, teaching functional thinking.

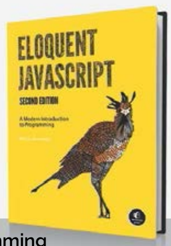
Much shorter than Thomas's famous 'pickaxe' book on Ruby, it's about putting the fun back into programming, in a world of multi-cores and adequate RAM, that begs for functional programming but without the academic trappings. As it says: "It's tomorrow already. Are you ready?"

Score ★★★★★

ESSENTIAL READING: JAVASCRIPT

The web's own language is powerful and eloquent, yet lives in a familiar environment (your browser)...

Eloquent JavaScript Second Edition



Author: Marijn Haverbeke
Publisher: No Starch
Price: £26.50
ISBN: 978-1593275846
nostarch.com/ejs2

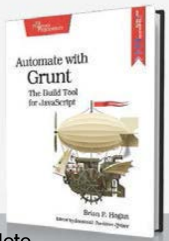
One of the best introductions to programming anywhere, updated for current JavaScript, and expanded with projects and more.

JavaScript & jQuery: Interactive Front-end Web Development

Author: Jon Duckett
Publisher: Wiley
Price: £26.99
ISBN: 978-1118531648
javascriptbook.com

Strikingly presented introduction to both adding interactivity with JS and speeding up your development using jQuery.

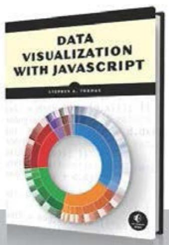
Automate with Grunt: The Build Tool for JavaScript



Author: Brian P Hogan
Publisher: Pragmatic
Price: £11.50
ISBN: 978-1941222119
tinyurl.com/nexephh

Focused guide to understanding the complete JS build environment - use Grunt to convert code, run tests, and produce distributions for production.

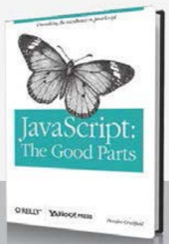
Data Visualization with JavaScript



Author: Stephen A Thomas
Publisher: No Starch
Price: £26.50
ISBN: 978-1593276058
nostarch.com/datavisualization

Communicate! Tree maps, heat maps, network graphs, word clouds, and even pie charts – interactively and in your browser.

JavaScript: The Good Parts



Author: Douglas Crockford
Publisher: O'Reilly
Price: £19.99
ISBN: 978-0596517748
tinyurl.com/3zpqdh5

Small but dense, and very rewarding, but not the first JavaScript book you should read. You'll be rethinking your entire approach to coding in JS.

RASPBERRY JAM EVENT CALENDAR

Find out what community-organised, Raspberry Pi-themed events are happening near you...

4 MILWAUKEE RASPI JAM
707 N 11th Street, 4th Floor, Milwaukee
6 CACHE VALLEY JAM
Wilson Elementary School, Logan
5 SILICON VALLEY JAM
Computer History Museum, Mountain View

PUT YOUR EVENT ON THE MAP

List your forthcoming events at:

raspberrypi.org/jam

1 PARIS PI JAM #2

When: Wednesday 6 May
Where: 135 boulevard de Chanzy, 93100 Montreuil
meetup.com/Paris-Raspberry-Pi-Jam
It's safe to say the first Paris Raspberry Jam must have gone well, because the second get-together is coming soon!

2 TORBAY TECH JAM

When: Saturday 9 May
Where: Paignton Library, Paignton, TQ4 5AG
torbaytechjam.org.uk
This monthly tech jam welcomes all kinds of hacking and making technology (as they all should!).

3 A SLICE OF PI CLUB

When: Tuesday 12 May
Where: Heart of Worcestershire College, Bromsgrove
sliceofpiclub.wordpress.com
Another new Raspberry Jam event with a focus on having fun hacking and making with the Raspberry Pi!

4 MILWAUKEE RASPI JAM

When: Wednesday 13 May
Where: 707 N 11th Street, 4th Floor, Milwaukee
bit.ly/1EEU2K2
Find it at the Kohler Center for Entrepreneurship at Marquette University. Have fun everyone!

5 SILICON VALLEY JAM

When: Saturday 16 May
Where: Computer History Museum, Mountain View
bit.ly/1IKPPZ1
This California-based event happens every third Saturday of the month at the wonderful **computerhistory.org** museum.

6 CACHE VALLEY JAM

When: Tuesday 19 May
Where: Wilson Elementary School, 89 S 500 E, Logan, USA
cachevalleyraspberryjam.com
The awesome Cache Valley maker community will be joined by our own Matt Richardson on the day!

3
A SLICE OF PI CLUB
Heart of Worcestershire College, Bromsgrove
7
SOUTHEND-ON-SEA
Tickfield Centre, Essex
8
CAMBRIDGE RASPBERRY JAM
Institute of Astronomy, Cambridge
2
TORBAY TECH JAM
Paignton Library, Paignton
1
PARIS PI JAM #2
Paris, France

7

SOUTHEND-ON-SEA

When: Saturday 30 May
Where: Tickfield Centre, Essex, SS0 7AB
twitter.com/SouthendRpiJams
Everyone is welcome to this free family tech event focused on coding and education. Make sure you book your tickets in advance.

8

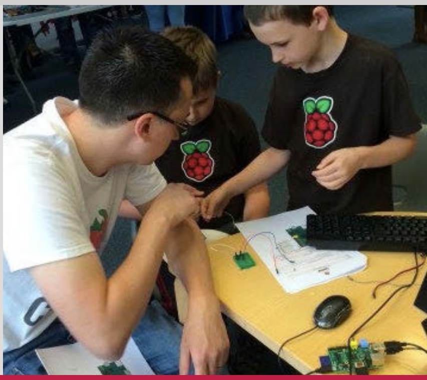
CAMBRIDGE RASPBERRY JAM

When: Saturday 6 June
Where: Institute of Astronomy, Cambridge, CB3 0HA, UK
camjam.me
After a fantastic ‘away’ event, the Cambridge Jam returns to its usual location for the June gathering.

DON’T MISS: CAMJAM

When: **Saturday 6 June** Where: **Cambridge, UK**

For many, the Cambridge Raspberry Jam is the highlight of the events calendar. Held at the brilliant Institute of Astronomy, its participants and organisers have access to a fantastic lecture theatre, a meeting room for workshops, a wide-open space for market stalls, and a mezzanine level which usually plays host to your community creations. Exact details will be posted nearer the time, so keep an eye on the event’s website.
camjam.me



In association with:

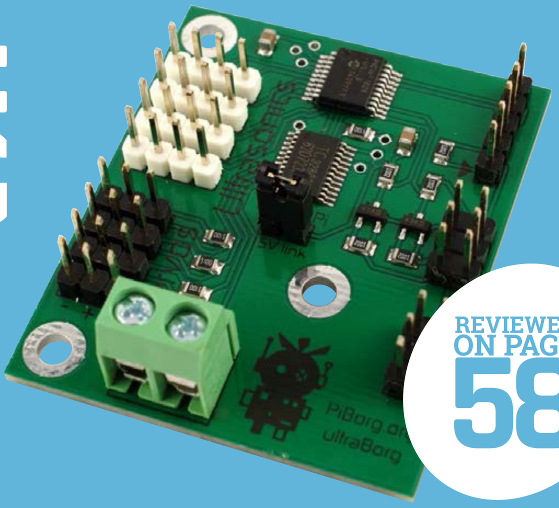
piborg.org

£200/$300 OF piborg GOODIES MUST BE WON!

WHAT DO YOU DO WITH YOUR RASPBERRY PI?

Tell us by 28th May for your chance to win!

1ST PRIZE: DIDDYBORG ROBOT



3 RUNNERS-UP: ULTRABORG

REVIEWED ON PAGE 58

How to enter:

Simply email **competition@raspberrypi.org** using no more than 50 words to tell us what you use your Raspberry Pi for most of the time. Ours is mostly used to write this.

Terms & Conditions

Competition closes 28 May 2015. Prize is offered worldwide to participants aged 18 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families or friends. Winners will be notified by email after the draw date. By entering the competition, the winner consents to any publicity generated from the competition in print and online. Participants agree to receive occasional newsletters from The MagPi magazine (unless otherwise stated upon entry). We don't like spam. Participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered.

Weaved and the Power of Pi.

Add the power of IoT and mobile in 15 minutes.

LEARN MORE

Weaved™
IoT Kit

The Internet of Things for Everyone.

The Weaved IoT Kit for Raspberry Pi gives developers the tools they need to transform any Raspberry Pi application into an IoT application. Here's what you get:

Weaved Embedded Cloud Stack
This super-light, downloadable software package is installed on your Pi to communicate with the Weaved Cloud Services.

Weaved Cloud Services
Weaved operates secure Cloud Services so you can incorporate IoT features into your Pi project like iOS/ Android Push Notifications and remote Mobile Device connections to your Pi over the Internet.

Weaved App for iOS/Android
Download the Weaved App to create a real IoT mobile App experience for your Pi project. The Weaved App receives standard Push Notifications from your Pi based on trigger events you define. Use Weaved to monitor and control your Pi from anywhere through a Web GUI hosted on your Pi.

Weaved Developers Portal
Register as a developer at developer.weaved.com to receive full access to our downloadable embedded software, documentation, how-to videos and sample projects.

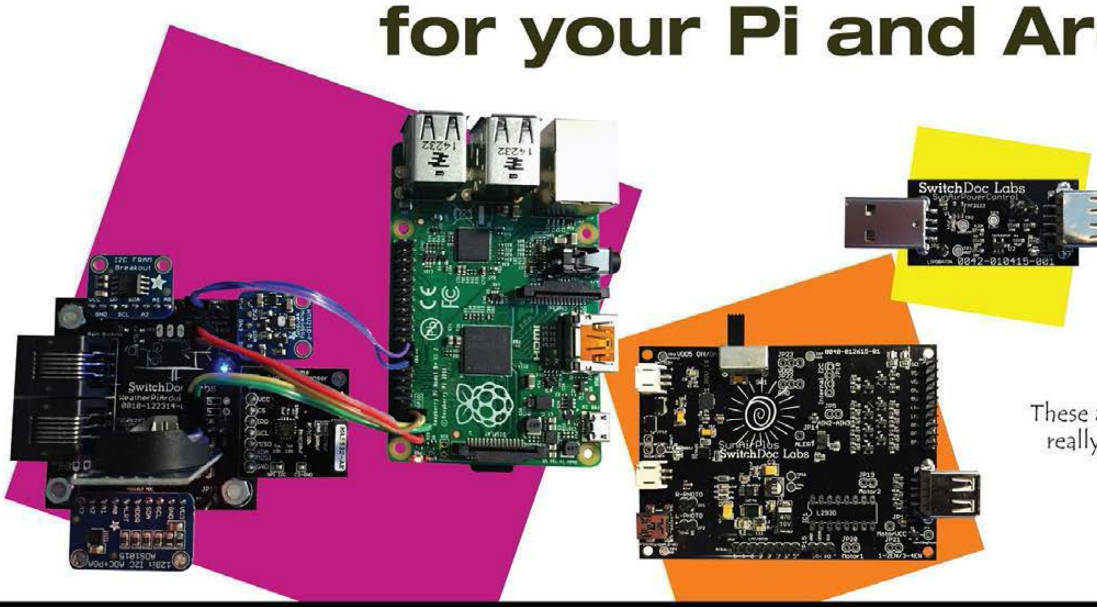
Weaved™

Weaved Available on the App Store Weaved IoT Kit

©2015 Weaved, Inc. All rights reserved. The Weaved App, 'The Internet of Things for Everyone,' Weaved and the Weaved logo are trademarks of Weaved, Inc. All other logos and names are trademarks or registered trademarks of their respective companies. Raspberry Pi is a trademark of the Raspberry Pi Foundation.

341 Hawthorne Avenue, Palo Alto, CA 94301 • Tel 650.262.0320 • www.weaved.com

Solar Power, Weather and Breakout Boards for your Pi and Arduino



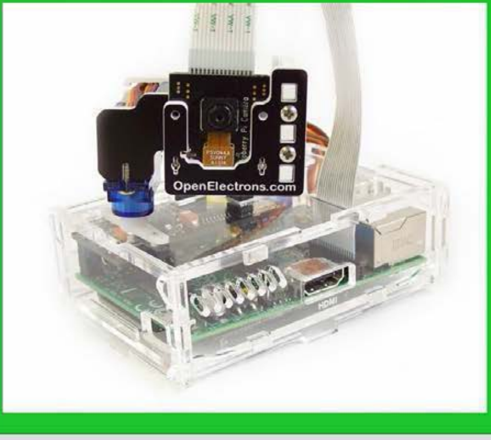
SwitchDoc Labs

switchdoclabs.com

© 2015 SwitchDoc Labs, LLC

PiConsole

A clever solution to Command Raspberry Pi Console from your Android or iPhone!



Pi-Pan

a Complete Pan-Tilt kit for Raspberry Pi Camera

10% OFF !!!

Use Code at checkout: MAGPIQE4WE

OpenElectrons.com

Supplying the full Raspberry Pi range of boards and accessories – worldwide.

Visit RS Components today to view the full range of Raspberry Pi boards and all the accessories you may need, including books, cables, data storage, expansion boards, operating systems, power supplies, prototyping kits and more.

Raspberry Pi

Visit RS Online

NEW Raspberry Pi 2 Model B 832-6274	Rasberry Pi Model A+ 833-2699	Raspberry Pi Model B+ 811-1284	Compute Module Development Kit 813-4164

DESIGNSPARK Help, hints, and tips and more for your Raspberry Pi Click Here

For full pricing and stocking information visit
www.rs-components.com/raspberrypi

RS

WANT TO GET NOTICED?

REACH THE RIGHT AUDIENCE FAST

› World's #1 Pi magazine
› Block booking discounts
› Free to download & share
› Live links on iOS & Android

The MagPi is the most exciting mag in tech today, boasting one of the biggest & most engaged audiences in the industry.

The MagPi

FOR MORE INFO EMAIL **RUSSELL@RASPBERRYPI.ORG**

MATT RICHARDSON

Matt is Raspberry Pi's US-based product evangelist. Before that, he was co-author of *Getting Started with Raspberry Pi* and a contributing editor at *Make:* magazine.

RASPBERRY PI AS A MATERIAL

Full computers have become a common component in projects. **Matt Richardson** discovers that it's not a new idea at all

When I began working with Raspberry Pi in my technology projects, it prompted a change in how I view computers. Because of the small size and low price of the board, I started to think that a computer isn't strictly a tool, but can also be considered a component in a project. Just as though I was reaching for resistors, wires, and microchips from my workshop component bins, I was also starting to reach into a bin of computers whenever I needed one. The Raspberry Pis in this stash were a material like any other.

This was an entirely new idea to me, but it wasn't until I came across the work of Seymour Papert that I found out that this isn't a new concept at all.

You might already be familiar with some of Seymour Papert's work. He was part of the team that created the Logo programming language, which uses a robotic or on-screen 'turtle' to move and draw according to procedural code. In 1988, Papert and George Franz contributed an article to the *Teachers College Record*, entitled 'Computer as Material: Messing About with Time'.

The article describes a junior high-school science lesson in a New York City public school. The students watched the teacher repeatedly place an empty glass jar over a lit candle to see it extinguish. The students were then challenged to measure the amount of time it took for the candle to be snuffed without the use of watches or clocks.

With the materials available to them (test tubes, pulleys, marbles, microscopes, scrap wood and the like) some students built pendulums; others used sand to make crude versions of an hourglass. Computers with Logo were also available to them, just like any of the materials above.

"When the students let their imaginations go, they found a variety of odds and ends for different explorations and investigations," according to Papert and Franz. "The computer was just one more material, alongside candles, crayons, ammeters, and rulers."

The students using Logo wrote code to keep time in a few different ways, from drawing a second hand on screen, to having the computer beep once per second. Even better, their timers became extensible with the addition of other components. For instance, with a light sensor, they could track the speed of LEGO cars on a ramp to iteratively optimise their designs for speed. Or with a temperature sensor, they could ensure that their class pet would be safe from the cold when the school turned off the building's heating system at night.

Constructionism

Papert coined the term 'constructionism' to describe this method of project-based learning and his ideas are being implemented in makerspaces at schools and libraries worldwide. As far back as 1971, Papert foresaw the computer as a central part of educational making. In 'Twenty Things to Do with a Computer', an MIT AI Lab paper Papert wrote with Cynthia Solomon (**hdl.handle.net/1721.1/5836**), they describe a computer system suited for such projects, complete with inputs and outputs.

"In our image of a school computation laboratory, an important role is played by numerous 'controller ports' which allow any student to plug any device into the computer," write Papert and Solomon. "The laboratory will have a supply of motors, solenoids, relays, sense devices of various kinds, etc. Using them, the students will be able to invent and build an endless variety of cybernetic systems."

When reading Papert's articles, it's hard not to think of Raspberry Pi as the perfect computer-as-material. So don't be surprised if one day you discover a bin for computers next to the bins for pipe cleaners and googly eyes.

Expand your Pi

Stackable expansion boards for the Raspberry Pi

Serial Pi Plus

RS232 serial communication board. Control your Raspberry Pi over RS232 or connect to external serial

Breakout Pi Plus

The Breakout Pi Plus is a useful and versatile prototyping expansion board for the Raspberry Pi

ADC Pi Plus

8 channel analogue to digital converter. I²C address selection allows you to add up to 32 analogue channels to your Raspberry Pi.

IO Pi Plus

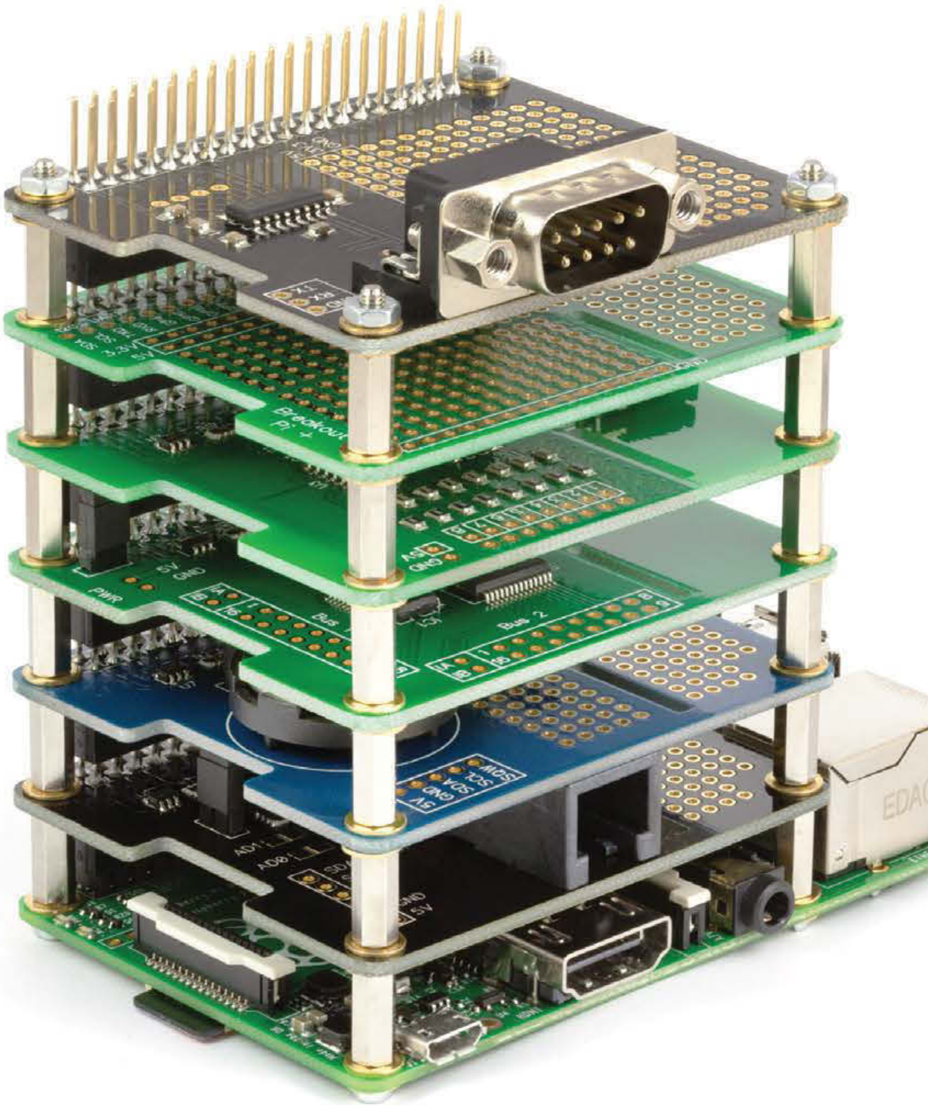
32 digital 5V inputs or outputs. I²C address selection allows you to stack up to 4 IO Pi Plus boards on your Raspberry Pi.

RTC Pi Plus

Real-time clock with battery backup and 5V I²C level converter for adding external 5V I²C devices to your Raspberry Pi.

1 Wire Pi Plus

1-Wire® to I²C host interface with ESD protection diode and I²C address selection.



We also stock a wide range of expansion boards for the original Raspberry Pi models A and B

AB electronics UK

www.abelectronics.co.uk

Wyliodrin

Have you tried Wyliodrin yet? It's free!

Use a browser from any device to program and monitor your Raspberry Pi

Program and monitor your Pi from anywhere in the Internet

Use our graphs to display your sensors' data

Just drag & drop blocks to create your applications, using Visual Programming

www.wyliodrin.com

python C++ node Perl Java ARDUINO php