70 PAGES OF EXPERT ADVICE IN THE OFFICIAL RASPBERRY PI MAGAZINE

The MagPi

The official Raspberry Pi magazine

Issue 34 June 2015 | raspberrypi.org/magpi

Win! £300 of PaPiRus ePaper HATs FROM PI-SUPPLY.COM

BUILD A MULTIPLE-CHOICE QUIZ USING SCRATCH

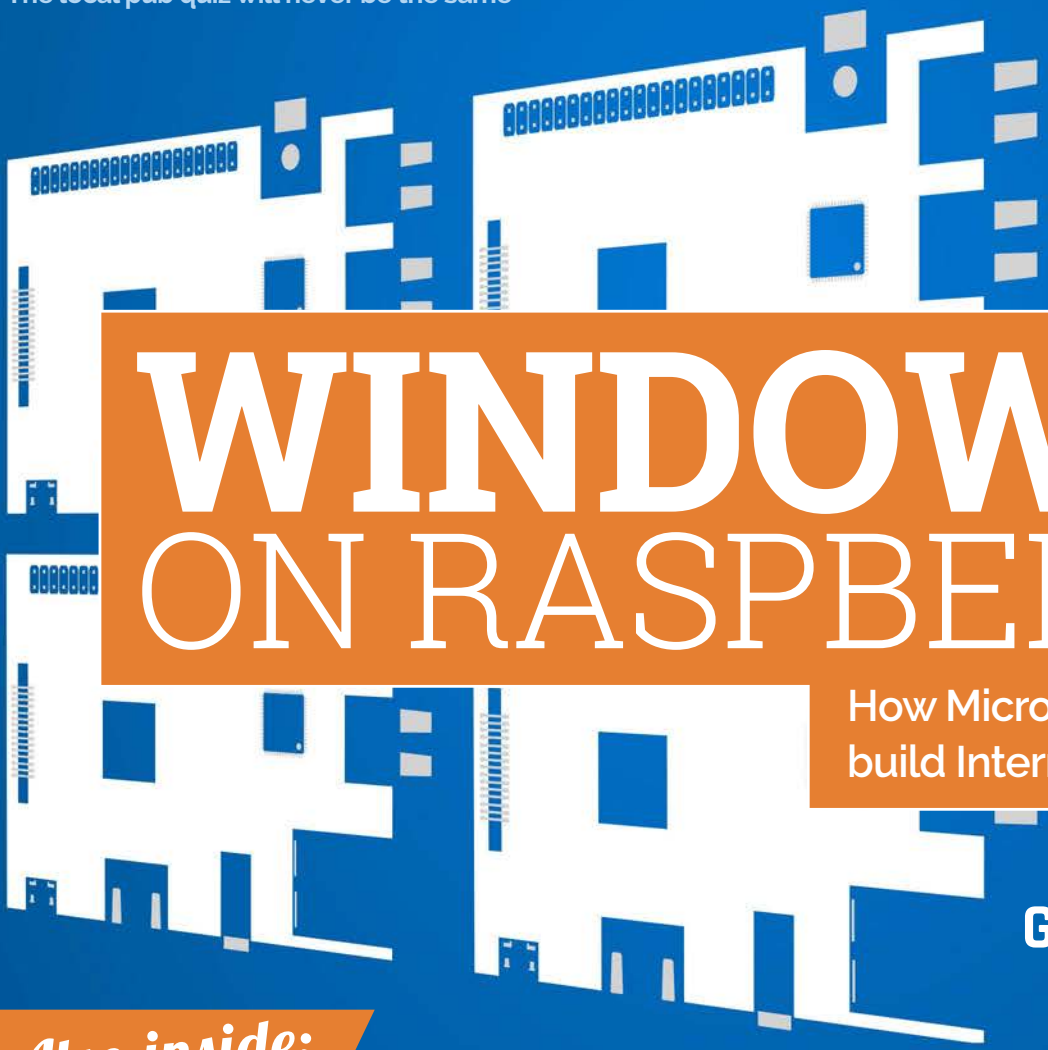
The local pub quiz will never be the same

USE YOUR RASPBERRY PI REMOTELY WITH VNC

Access your Raspberry Pi desktop on your tablet or smartphone

PUT YOUR CODE IN SPACE!

The latest on the Astro Pi project. There's still time to enter



WINDOWS 10 ON RASPBERRY PI

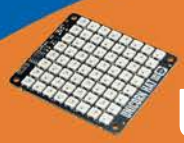
How Microsoft plans to help you build Internet of Things devices

MAKE A PLATFORM GAME WITH PYTHON

Put all your game-making skills to work on your first full game

Also inside:

- HACK A CLOCK WITH YOUR RASPBERRY PI
- GET CREATIVE WITH CONDUCTIVE PAINT
- MORE AMAZING PROJECTS REVEALED
- 4TRONIX AGOBO ROBOT REVIEWED



SCROLL TEXT OVER THE UNICORN HAT

Put your text up in lights with our expert step-by-step guide

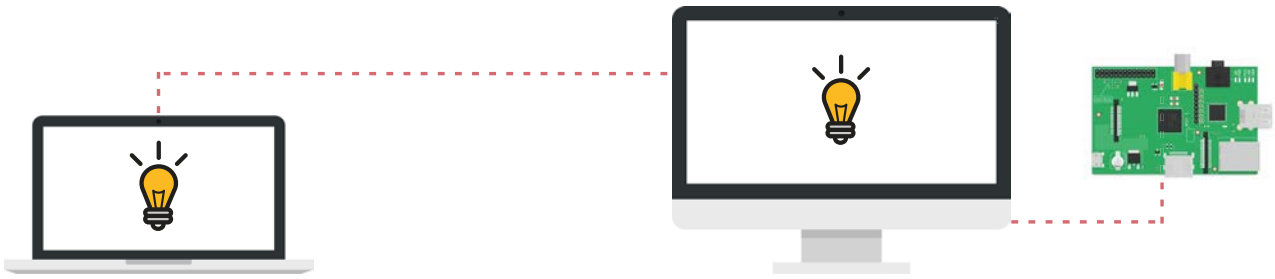
Plus HOW TO CONNECT A PRINTER TO YOUR RASPBERRY PI

VNC® now available for RASPBERRY PI!

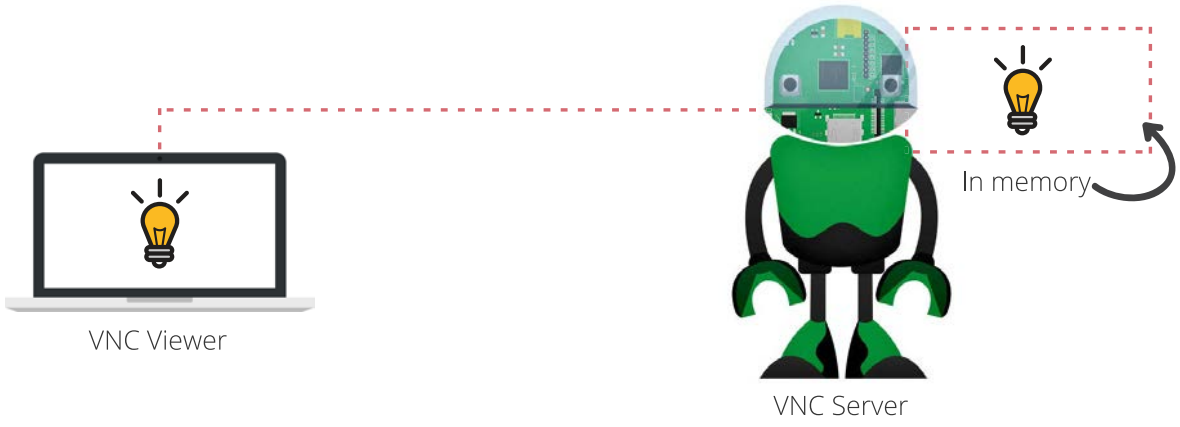
RealVNC has released VNC for Raspberry Pi, which means you can now connect to your Pi from any Windows, Mac or Linux computer, or iPhone, iPad or Android device!

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/

REAL VNC

For more information contact sales@realvnc.com

WELCOME TO ISSUE 34!

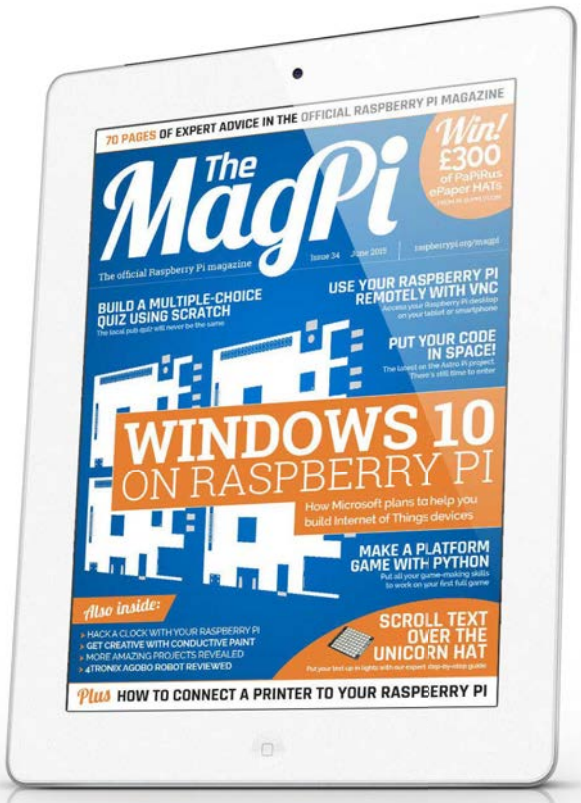
Around the time of its launch, I was lucky enough to visit Pi Towers to interview Eben Upton about the release of the Raspberry Pi 2. It was immediately clear to me that the Broadcom BCM2836 at the heart of Pi 2 would change the way we use the Raspberry Pi forever. That's not just because the extra processing power and memory would transform it from a credit-card-sized PC into a fully fledged personal computer, capable of all the day-to-day tasks we otherwise spend hundreds of pounds to carry out. From the outset, it was clear that the change in architecture would open many new doors in terms of all-important software support.

Upon learning about the new architecture, like many I realised that Ubuntu, the Linux world's most popular distribution (and said to be installed on some 20 million computers), would almost certainly find its way onto the Raspberry Pi and it has subsequently done just that. You can find our review of Ubuntu MATE 15.04 on page 59. While it currently lacks several key Pi-specific features (like GPIO and Camera Module support), it's already proving its worth.

What I didn't bank on, though, was the arrival of Windows 10. Whether it was short-sightedness or downright shyness, I didn't mention it to Eben at the time. Of course, it's not the same Windows 10 some of you will be using on your laptops or smartphones; Windows 10 IoT Core is a very different offering, and you can learn all about it in our cover feature starting on page 14.

Enjoy the issue!

Russell Barnes



THIS MONTH:

08 THE ROBOTS INVADE!
See how Raspberry Pi robots are being used in the classroom

14 WINDOWS 10 COMES TO PI 2
Find out how Microsoft wants to help you make IoT devices

26 GO FROM 0-100 IN THREE SECONDS
McMaster University turns to the Pi for some speedy calculations

50 MAKE A PYTHON PLATFORM GAME
In part 4 of our game-making series we build our first full game!

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

The MagPi

Available on the App Store

Get it on Google play

EDITORIAL
Managing Editor: **Russell Barnes**
russell@raspberrypi.org +44 (0)7904 766523
Technical Editor: **David Whale**
Sub Editors: **Laura Clay, Phil King, Lorna Lynch**

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

PUBLISHING
For advertising & licensing:
russell@raspberrypi.org +44 (0)7904 766523
Publisher: **Liz Upton**
CEO: **Eben Upton**

With thanks to this month's contributors: **Dave Bower, Mike Cook, Gareth Halfacree, Lucy Hattersley, Phil King, Simon Long, Sean McManus, Simon Monk, Les Pounder, Matt Richardson, Richard Smedley, Sean M Tracey** and **Robin Withers.**

Contents

Issue 34 June 2015 raspberrypi.org/magpi

TUTORIALS

> EVERYDAY ENGINEERING (PT 4) **30**
Hack an analogue clock with Dr Simon Monk

> COMMAND LINE PI (PT 4) **34**
Richard Smedley continues his essential series

> MAKE A QUIZ IN SCRATCH **36**
Dazzle your friends with this multiple-choice game

> HACK RASPBIAN (PT 3) **38**
This month Simon shows us how to customise Openbox

> REMOTE CONTROL WITH VNC **40**
Access your Pi's desktop from your tablet in easy steps!

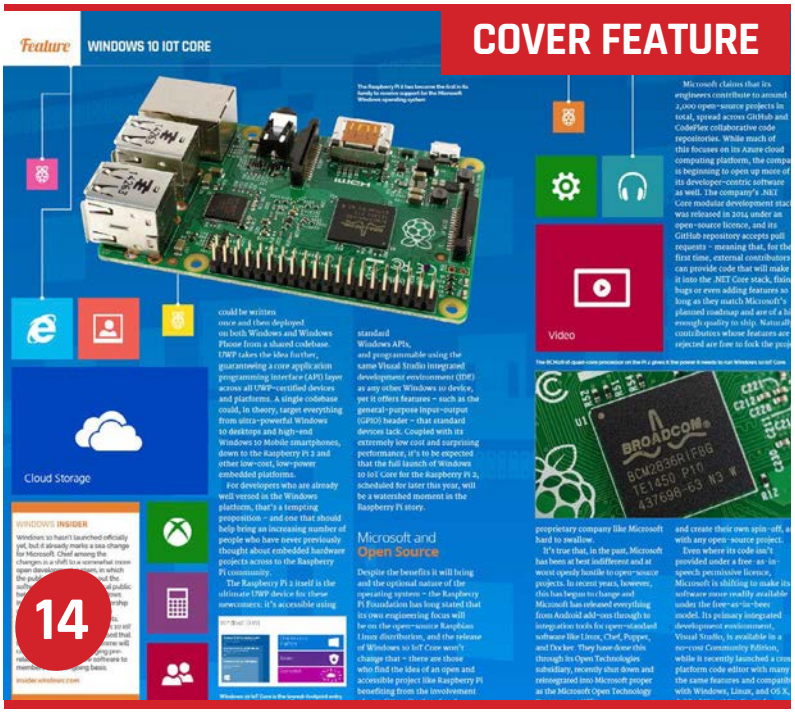
> TEXT ON THE UNICORN HAT **42**
Find out how Sean McManus can put your text up in lights

> PRINT WITH THE PI **44**
It's a dark art, but it's not impossible! Get started here...

> MIKE'S PI BAKERY (PT 2) **46**
This month Mike Cook builds a Safari Adventure game

> MAKE A PLATFORM GAME **50**
In part 4 Sean puts all the skills you've learnt to the test

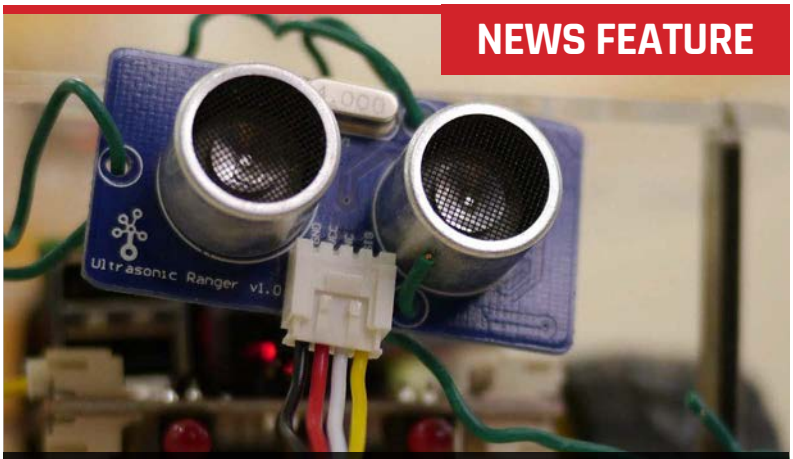
COVER FEATURE



Windows 10 for Raspberry Pi

Gareth Halfacree talks us through Microsoft's plans to help you build Internet of Things devices, and how a Pi 2 robot was the star of its recent HoloLens reveal

NEWS FEATURE



Robot invasion!

We speak to associate professor George Dimitoglou about how a Raspberry Pi robot is helping him teach

8

First Astro Pi winners announced!

10

While two primary school winners will already be getting their projects sent to space, it's not too late for you to enter

Contents

REGULARS

> NEWS **6**
Keep up-to-date with the biggest stories in the Pi community

> BOOK REVIEWS **62**
The latest computer books reviewed and rated

> COMMUNITY EVENTS **64**
Find a community gathering near you in the coming weeks

> THE FINAL WORD **68**
Matt Richardson tells us of his experiences at Build

REVIEWS

> 4TRONIX AGOBO **56**
The affordable Model A+ exclusive robot goes on test

> KANO OS 2.0.0 **57**
How does the latest release of this free OS stack up?

> MEARM **58**
An affordable and open-source robot arm from Phenoptix

> UBUNTU MATE 15.04 **59**
Les Pounder test-drives a brilliant new distro for the Pi 2

YOUR PROJECTS

#HIUTMUSIC JUKEBOX

Find out how a creative agency helped customers reach out to their favourite jeans brand with a Twitter-powered Raspberry Pi music player

Pocket PiGRRL **22**

Adafruit's Ruiz brothers refresh their Nintendo Game Boy homage with a Model A+ twist

CandyPi **24**

Get a fix of tasty candy with the aid of this Raspberry Pi-powered sweet dispenser

McMaster EcoCAR **26**

It's quarter the size of an F1 car, but still reaches speeds of 150km/h!

ANNOUNCEMENT

The MagPi IS MOVING TO PRINT!

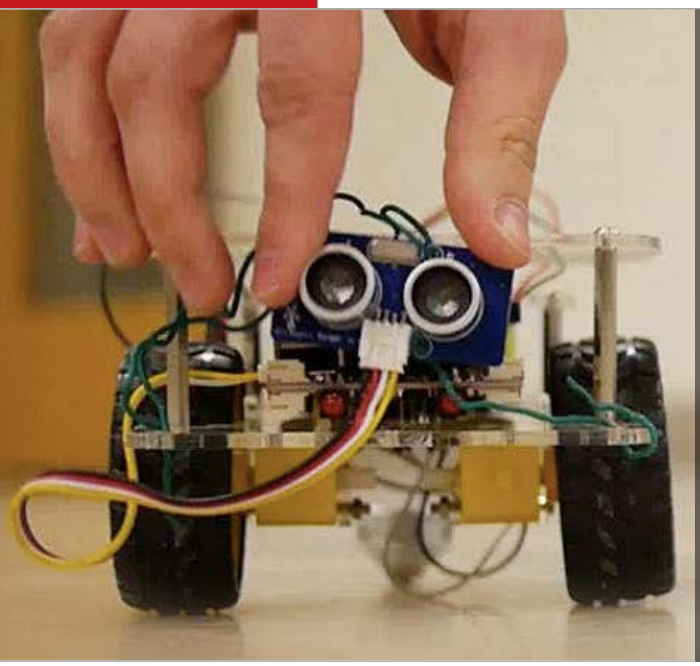
We're celebrating the magazine's third birthday with an exciting development: from issue 36 you'll be able to buy The MagPi in print from all good newsagents!

You read it right – from issue 36 you'll be able to stroll into a shop and buy your very own copy of *The MagPi* magazine! As well as ordering a print copy online, UK readers will be able to pick it up from most WH Smiths and Smiths Travel stores, not to mention selected Tesco stores and independent newsagents. Readers in the USA, however, will be able to buy it from selected Barnes & Noble stores. As you might expect, you will also have the option to subscribe for up to 12 months so you can receive the magazine direct to your door every month.

In time, we'll also be giving you the option to purchase heavily discounted print and digital versions in one handy package, meaning you'll have every kind available – a physical copy for stroking, a PDF copy for your big-screen PC, and a digital edition for your tablet or smartphone.

If that's not enough, we'll also be increasing the size of the magazine to 100 pages. That means more news, more projects, and even more tutorials. We'll be introducing new features too, including your chance to ask or answer questions and comment on current affairs. We hope you're as excited as we are!

THE MAGPI IS MOVING TO PRINT



ROBOTS INVADE OUR COLLEGE CLASSROOMS

How a Raspberry Pi robot is helping to teach computer science at Hood College in the USA...

A swarm of mini robots whizz around the classroom floor as the teacher steps through the door. Far from being irate, however, he's delighted that his students have done their homework. For this is the Robotics & Intelligent Systems class at Hood College, Maryland, an upper-level elective for students majoring in Computer Science. Associate professor George Dimitoglou tells us it's a popular course: "I get the students who can't wait to work on robots; I don't have to convince anyone it is interesting or cool..."

Having taught robotics for a number of years, George was looking for a platform to support the use of a modern programming environment, such as Python, that the students already knew or could quickly learn, so they could focus on learning how to develop autonomous, intelligent robotic behaviours.

Dexter Industries' Pi-powered GoPiGo robot fitted the bill, meeting a number of key criteria. "First, it allowed us to use a modern programming environment, which meant we could start doing robotics immediately after kit assembly," explains George. "Second, it gave us the ability to have the full package of everything we need to exercise autonomous behaviours (microcontroller, servos, sensors). Third, it is an expandable platform; we can add many more sensors and modules to increase its capabilities."

Another advantage is the affordable price: "Having their own robots, rather than depending on a lab-based shared pool of hardware, meant that students can work on their homework and projects anywhere, at home, the library, or at the cafeteria, without worrying about lab hours of operation and access privileges. Plus, they also liked the idea of getting to keep the robot after the course is over."

Parallel parking

It's not just goofing around with tech toys, however. Far from it. Students are expected to implement algorithms for mobile and intelligent behaviours. "We start with basic motion and sensor exercises and quickly program the robots to do obstacle avoidance and different manoeuvres such as a figure-eight around two obstacles, and parallel parking," reveals George. "We then move to more intelligent activities such as incorporating a camera for mapping, browser streaming, and smart navigation to do 'surveillance'."

MEET YOUR MAKER

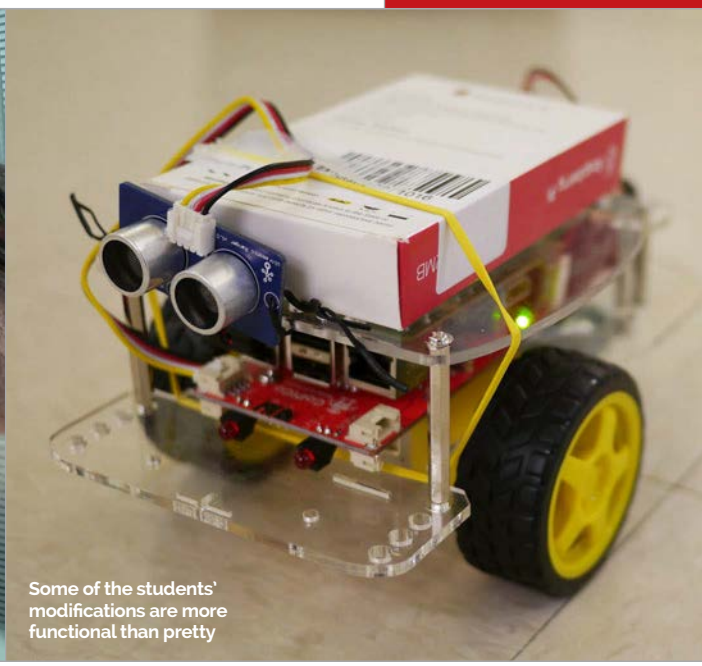
The Hood College robotics students were delighted by a surprise visit from GoPiGo creator John Cole, of Dexter Industries, back in March. "Having John in class was a wonderful opportunity for the students to ask questions, work out any kinks, and better understand the rationale behind certain design aspects and features of the platform," enthuses associate professor George Dimitoglou.

John tells us, "It was a great experience to see what the students were doing with the robot, and how they were learning real-life engineering with the GoPiGo and the Raspberry Pi." He was impressed by what the students were doing with the robot: "They were really inventive and had some great feedback for the design. They had designed different ways to mount sensors [and] add traction for wheels on slippery surfaces, and suggestions about where we needed to make our assembly instructions better.

"Where they were most innovative was in the software... I was really impressed by the level at which they were working with Python. What was even better was that since our software is all open source, we could take their suggestions and improve [it] immediately."

Video highlights of the visit can be found at: **youtu.be/i1PIORq0ILs**.

A selfie of George (second from left) and students, taken from a GoPiGo



Some of the students' modifications are more functional than pretty

So far, the focus is on trying to see what they can do with the unmodified GoPiGo platform. However, one of the students outfitted his robot with a portable USB rechargeable battery, while another created a rotating ultrasonic sensor base using a bottle cap and a screw. "They are not pretty, but I like seeing students being innovative and I encourage it." George and his students have also started looking into integrating more sensors into the GoPiGo: "We are currently experimenting with a low-cost LIDAR to perform distance calculations for mapping using a near-infrared laser beam."

Crash course

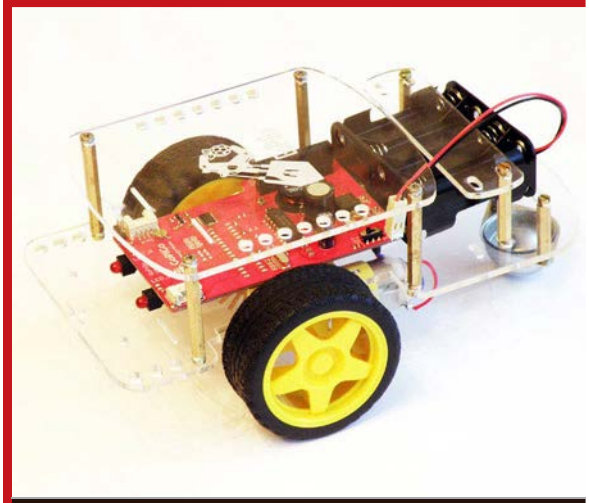
George says there are both technical and non-technical educational benefits to the course. As well as being programmable using Python, the GoPiGo has separate hardware components and "[students] learn about the physical 'anatomy' of a robot and what each part does."

As for the non-technical benefits, George believes that robotics in general is an eye-opener for computer science students. "Robotics is the physical manifestation of computation and for many students – not just my own – it is the first time while programming they get a real emotional connection with their work product. When software fails, it gives a bunch of warnings and errors on the screen. When a robot fails, it runs off the table, crashes on a wall, hits an obstacle, or completely misses a door. The failure of the computation is suddenly physical, unquestionable, and obvious. Suddenly they realise their code has ramifications and they pay more attention; they even test their programs more."

> "When a robot fails, it runs off the table, crashes on a wall, hits an obstacle..."

Eventually, George would also like to do some distributed coordination activity with multiple robots working together on completing a task, using the WiFi dongle and the GoPiGo's ad hoc networking capabilities. "This type of exercise is what got me into robotics," divulges George. "I got into it while looking for an environment to exercise my theoretical distributed coordination algorithms and I got... sucked in."

GOPIGO



Launched via a successful Kickstarter crowdfunding campaign last year, the GoPiGo turns your Raspberry Pi into a fully functional two-wheeled robot. The $90 Base Kit includes everything you need – the GoPiGo board, chassis, wheels, motors, encoders, and power battery pack – and is easy to assemble.

According to its creator John Cole, of Dexter Industries, the GoPiGo was originally designed to make robotics accessible to everyone, especially in education. "We've got software for it in a lot of languages at this point, including Scratch and Python. Our hope was to give students a place to start with robotics and leave the upside or potential as wide as possible. We have a few things we're working on right now that are specifically geared towards education (which we will be announcing soon!) to help make it even easier to use in the classroom." Learn more at dexterindustries.com/GoPiGo.

FIRST ASTRO PI WINNERS ANNOUNCED

Children from two UK entries have succeeded in having their Raspberry Pi coding ideas accepted to travel to the ISS in November, but it's not too late to submit your own code…

TEACHING PI

Want to see inspiring Pi-based activities in your children's school? Tell the teachers about Picademy: it's a free professional development experience for primary and secondary teachers, open to individuals around the world: **raspberrypi.org/picademy**

In **The MagPi 31**, we reported on the fabulous competition to enable UK schoolchildren to send their code up to the International Space Station (ISS) with British astronaut, Major Tim Peake. The competition was divided into age groups, with promising entrants to the three 11–18 age bands already awarded free Astro Pis – a Raspberry Pi with Astro Pi HAT (add-on board) – to work out their code on.

Meanwhile, primary-age children (under 11s) have submitted their ideas to the Astro Pi judges, and winners have been announced. The standard of the nearly 200 entries in the primary-age category was, we are told, "incredibly high," and Major Tim, announcing the results, noted that the judges had a "really tough time choosing the overall winners."

Doug Liddle of British Satellite manufacturer SSTL, and a member of the Astro Pi judging team, said: "Ultimately, the winning teams had to propose ideas that were creative, practical, and useful to stand a chance of winning. I hope that most of these talented primary school teams also decide to get involved in the next stage of the competition and give the secondary schools a run for their money." Read on to find out how you can still enter the competition.

During our coverage of the launch of the Astro Pi competition, Libby Jackson, the UK Space Agency's astronaut flight education programme manager, told us to expect "amazing ideas [and things] that we haven't even thought of" from the entrants. Jackson has been proven right already, with just the primary school entries in.

This is certainly the case with the two winners in the category, to whom **The MagPi** extends congratulations. Hannah Belshaw from Cumnor House Girls' School in Croydon won top place with her idea to represent data from the Astro Pi in the world of *Minecraft*. According to the judges, "This really ticked all of the boxes of practicality, creativity, and usefulness," using the Astro Pi to record data about the ISS's environment, then turn it into a *Minecraft* world for everyone to explore.

Not only does this use the popularity of *Minecraft* to get children further involved in STEM (Science, Technology, Engineering and Maths) subjects, but by

Above Judging a competition like this can be intense – but there's a lot of fun along the way, and judges were extremely impressed by the high standard of entries

> "This really ticked all of the boxes of practicality, creativity, and usefulness"
>
> **Major Tim Peake, British astronaut**

representing abstract sensor data that the Astro Pi captures in an understandable way, it's a great piece of data visualisation. The 'digital flyby' created in *Minecraft*, with data downloaded from the International Space Station, can be run by anyone with a Raspberry Pi.

Sweaty astronauts

Crocodiles team from the Cranmere Code Club at Cranmere Primary School, Esher, were also winners with their idea to investigate whether the Astro Pi can detect the presence of astronauts in the module using the temperature and humidity sensors of the HAT, taking a picture to see what caused the changed readings.

"The Cranmere entry was very clearly and comprehensively presented," said CGI's Pat Norris from the judging panel. "It included a statement of the objective of what is effectively a scientific experiment and of the approach proposed to achieve that objective, and complemented this with logic flowcharts and a diagram. Part of the activity takes place on the ISS and part on the ground after the data has been collected, giving the Cranmere Code Club an opportunity to participate directly in the experiment. The judging panel was impressed by the sophistication of the entry, demonstrating an appreciation of the scientific method (hypothesis tested by experiment) and a thorough analysis of the logic involved."

Since both schools will be receiving a class set of Astro Pi kits, they'll be able to get involved in the data-logging activities once Major Tim starts his mission, as will a number of other primary entries who were highly commended by the competition judges after two days of judging.

Coding up

The two winning ideas will now be coded up for the primary schools by a team from the Raspberry Pi Foundation, whose Jonathan Bell said of the *Minecraft* idea: "We anticipate that we will have as much fun programming (and testing) this entry as children will have exploring a game world created from data captured in space."

The judges have reviewed all of the ideas sent in by entrants from the 11- to 18-year-old age bands, and have sent out complete Astro Pi kits to applicants. These, along with any others who'd like to get involved [see 'It's not too late!' boxout below], can now get busy developing their code for judging after the 20 June cut-off date.

Six schools from the 11–18 groups were highly commended, winning a whole class set of Astro Pi kits. Along with any primary schools that wish to code up their ideas, the older children now have until 29 June to submit their code, which could be sent into space along with the winning primary entries.

We'll be bringing you further coverage of Astro Pi after the final winners are announced, and will let you know what happens when the code gets carried up to the ISS with Major Tim's cargo of Raspberry Pi boards and Astro Pi HATs.

The Astro Pi aluminium flight case that British ESA astronaut Major Tim Peake will be using on the ISS

IT'S NOT TOO LATE!

For phase two of the Astro Pi competition, secondary schools must code their ideas up, submitting them via the competition website by 29 June. Primary students, even if they haven't already entered the primary category, can submit coding in the youngest age bracket of this competition.

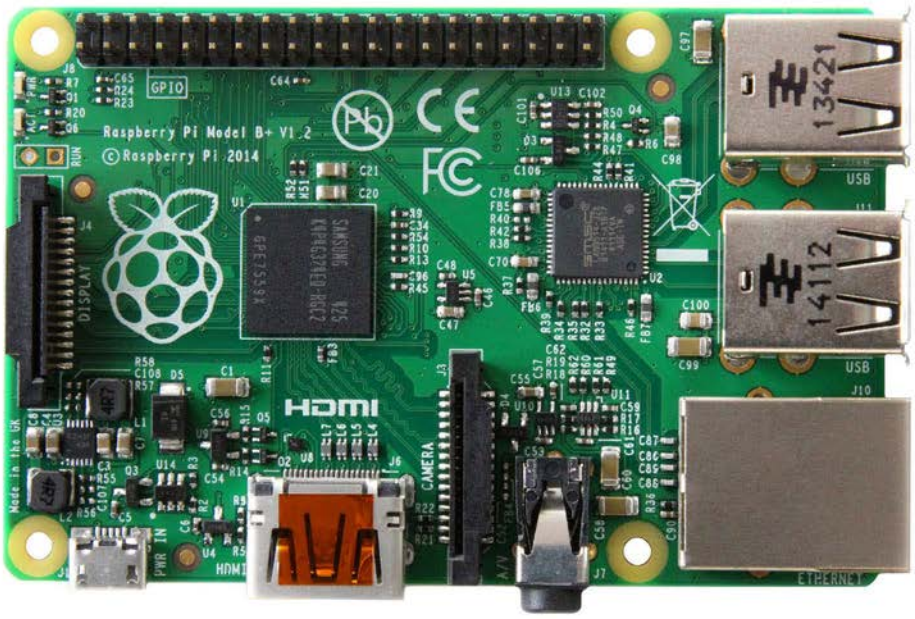
If you missed phase one, you can still enter – right up until the last day, but that wouldn't leave much time for testing! See the website for competition rules, and how to get your hands on an Astro Pi HAT if you didn't apply earlier:

astro-pi.org/secondary-school-competition

Your group will need to submit:

- A short document providing an overview of the experiment (or app), its objectives, and methods
- The Python source code and an executable file
- Any additional information you like (in PDF format)

RASPBERRY PI MODEL B+ NOW ONLY $25



The mid-range Model B+ is perfect for projects that need good connectivity, but don't require much CPU load

SB COMPONENTS UPDATES ITS MEDIAPI+ CHASSIS FOR PI 2

A second generation of the popular MEDIAPI set-top box-style case has been released by SB Components (**sb-components.co.uk**). The MEDIAPI+ features a 3A power supply (with UK, EU, and US plugs) to power the case's 4-port integrated USB hub that routes connectivity to the front and rear. It features IR support and an accompanying IR remote control, as well as space for a USB hard drive. Learn more at: **sb-components.co.uk/products/mediapiplus**

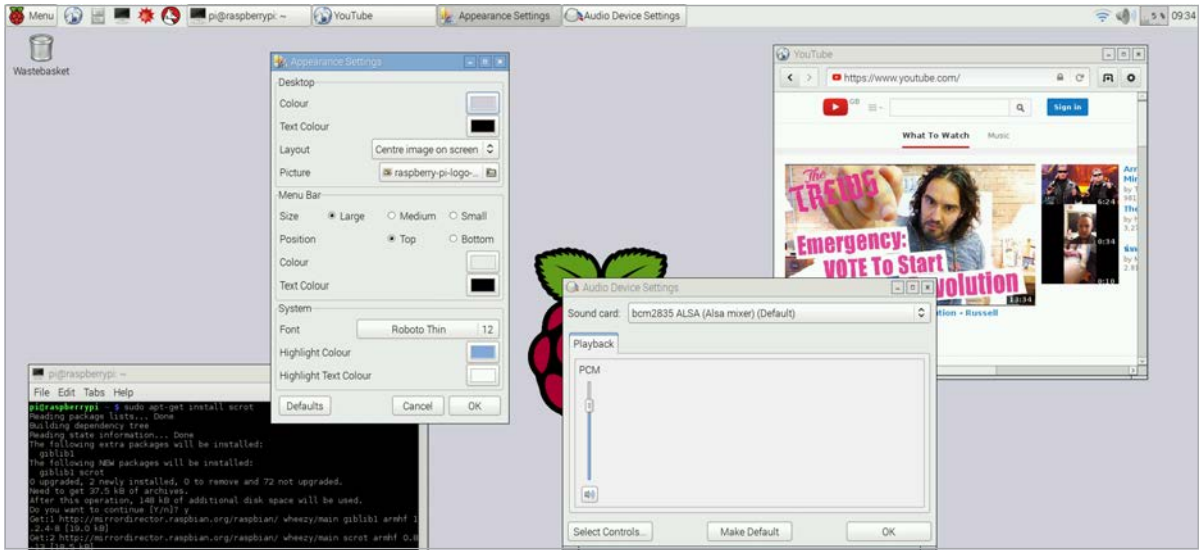
The mid-range Raspberry Pi drops in price, creating a clearer price structure across the range...

The Raspberry Pi Model B+ has dropped in price to £16 / $25. In his blog post announcing the price cut on **raspberrypi.org**, Eben Upton put the reduction down to simple manufacturing savings: "A side effect of the production optimisations that allowed us to hit the $35 target price for Raspberry Pi 2 is that the Model B+ is now much cheaper to manufacture than it was when it was introduced. With this in mind, we've decided to drop its list price to $25."

The Raspberry Pi 2 Model B went on sale in February and has proved a resounding success, improving performance sixfold in the same form factor and price bracket. At the start of 2015, it was announced that the Raspberry Pi platform had sold over five million since its 2012 debut, though the Pi 2 is said to have already sold in the region of one million units within just a few months.

With the price drop of the Model B+, the consumer Raspberry Pis available offer a logical price range: $20 for the Model A+, $25 for the mid-range B+, and $35 for the high-end Raspberry Pi 2 Model B.

"If you're looking for a Raspberry Pi with networking and multiple USB ports, and don't need the extra performance or memory that the Raspberry Pi 2 brings, you might want to check it out," Eben concludes.



Left New quality-of-life updates to Raspbian bring it more in line with other world-leading operating systems

RASPBIAN GETS ANOTHER INTERFACE UPDATE

The Raspberry Pi Foundation's UX engineer Simon Long has released yet more quality-of-life and usability changes for the Pi's default distro...

Following on from the successful batch of user interface changes over Christmas 2014 for Raspbian, the Raspberry Pi's officially supported operating system, a new set of desktop changes has recently landed. "The overall look and feel hasn't changed, but there have been a few major new features added, and a lot of small tweaks and bug fixes," explains the Raspberry Pi Foundation's UX engineer, Simon Long.

Among the key updates is a much-needed revamp of the Wi-Fi interface. "If you look towards the right-hand end of the menu bar, there is now a network icon," says Simon. "This shows the current state of your network connection – if a Wi-Fi connection is in use, it shows the signal strength; if not, it shows the state of your Ethernet connection." This brings Raspbian more in line with other leading operating systems and therefore should provide a far more familiar experience for most users.

In much the same way, updates have been made to the audio interface, as Simon explains:

"This works exactly the same way as on Windows or Mac OS [X] – left-click it to drop down the volume control, which you can then slide with the mouse. There's a mute checkbox at the bottom to quickly mute or unmute the audio output."

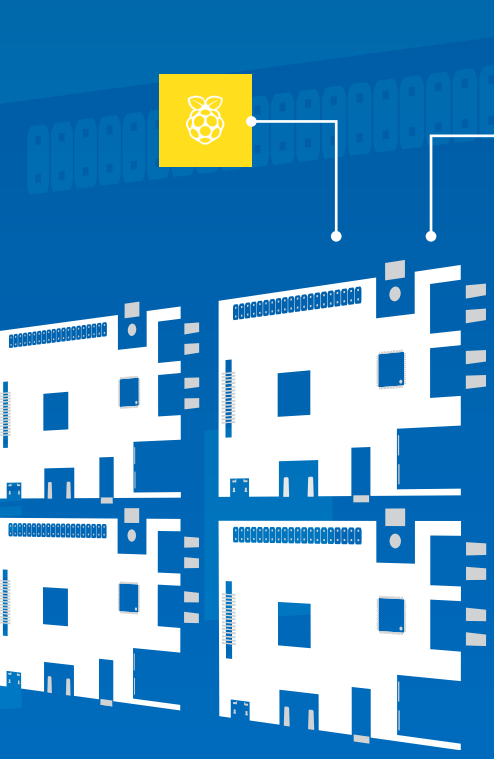
If you right-click the volume icon, a pop-up menu appears to enable you to select which audio output is used – on a standard Raspberry Pi, you have the choice of HDMI or analogue. If you have installed a USB audio interface, or an external sound card, these will also appear as options in this menu.

You'll need to read Simon's article on **raspberrypi.org** for a full rundown of other changes and fixes, but if you want to upgrade your current installation of Raspbian, just type these two commands separately in an LXTerminal window:

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

If you'd like to use the new network interface, you'll need to install it (it's a separate package) by typing the command below in the terminal and following the instructions. Don't worry – your original config files can be found in **/home/pi/oldconffiles** if you want them:

```
sudo apt-get install
raspberrypi-net-mods
```



Windows 10 on Raspberry Pi

How **Microsoft** wants to help you build your own **IoT** devices

When the Raspberry Pi 2 launched back in February, the shift to the new Broadcom BCM2836 processor delivered more than just a sixfold boost in performance; it brought the promise of wider compatibility with other operating systems. While some announcements, such as Canonical's Ubuntu Core being made available for the Raspberry Pi 2, were unsurprising, one caught the community's imagination in a major way: Microsoft confirming that a version of Windows 10 would be available for the Raspberry Pi 2, and that it would be released free of charge to all makers and tinkerers around the world.

For many, it seemed unbelievable – even unwelcome. Knee-jerk comments on the blog post announcing the partnership claimed that the Raspberry Pi Foundation had turned 'to the dark side', expressed disappointment that closed-source Windows 10 would arrive on the platform before a long-awaited build of Google's Android, and even raised concerns about its performance and the impact the release could have on the Foundation's educational aims.

Others were more optimistic, pleased to see Microsoft confirming what the community has long known: that the Raspberry Pi is something special, transformative and revolutionary, and that with the proper support it can easily please both hobbyists and tinkerers, while also providing the low-cost platform required for commercial and large-scale Internet of Things (IoT) projects.

While commentators were divided on the effect the move would have on the Raspberry Pi project itself, all could agree that the announcement was badly covered by the mainstream press. Numerous headlines trumpeting that 'Windows 10 will be free on the Raspberry Pi' missed the biggest part of the story: Microsoft was developing an entirely new version of the operating system, designed exclusively for embedded use, dubbed the Windows 10 Internet of Things (IoT) Core.

Those who had read the headlines and expected to receive a fully functional desktop operating system for their low-cost microcomputers would be disappointed; however, for those who understood what Microsoft was trying to do, things were about to get very exciting indeed.

What is Windows 10 IoT Core?

Due to launch this summer, Windows 10 – previously known by the codename Windows Threshold – represents Microsoft's vision for the future of computing. The main version, Windows 10, will be available for desktops, laptops, and tablet computers; Windows 10 Mobile will bring the same features and functionality to smartphones, while the two versions will work together in a variety of ways. While Windows 10 could be said to be a simple update to Windows 8.1, and Windows 10 Mobile to Windows Phone 8.1, Windows 10 IoT Core is something entirely new.

Born from the company's previous work on Windows Embedded, and Windows CE before it, Windows 10 IoT Core is exactly what the name implies: an attempt by Microsoft to create a core platform for the Internet of Things, the buzzword-of-the-day, given over to everything from mesh sensor networks monitoring floodwaters and radiation levels, to thermostats that connect to your smartphone, and doorbells that send you social network alerts.

> "Microsoft was developing an entirely new version of the operating system, designed exclusively for embedded use"

The operating system itself is split into three versions: Windows 10 IoT for small devices, the version available for the Raspberry Pi 2, does not come with the user interface – 'shell,' in Microsoft's parlance – of the desktop operating system; Windows 10 IoT for mobile devices comes with the Modern user interface, as used on Windows 10 Mobile; Windows 10 IoT for industry devices includes the full Desktop shell of Windows 10 along with support for running legacy Win32 applications – something the other two builds can't offer.

Although, technically speaking, Windows 10 IoT Core for small devices is a fully fledged operating system in its own right, with no shell it's a very limited one. The reason for that is that Microsoft has aimed the software at Windows developers who want to get involved in the Internet of Things.

Rather than running a shell through which a user chooses applications, Windows 10 IoT Core is designed to run a single application – turning the Raspberry Pi, or whichever device the system is running on, into a dedicated machine that can be easily programmed and interacted with, using standard Windows application programming interfaces (APIs) and languages.

That's not to say the application running on a Pi can't be interactive, of course: on first boot, the operating system loads an example application which displays the IP address of the Pi via the HDMI video output. Video, even 3D and accelerated 2D, is entirely possible if your application requires it.

Universal Windows Apps

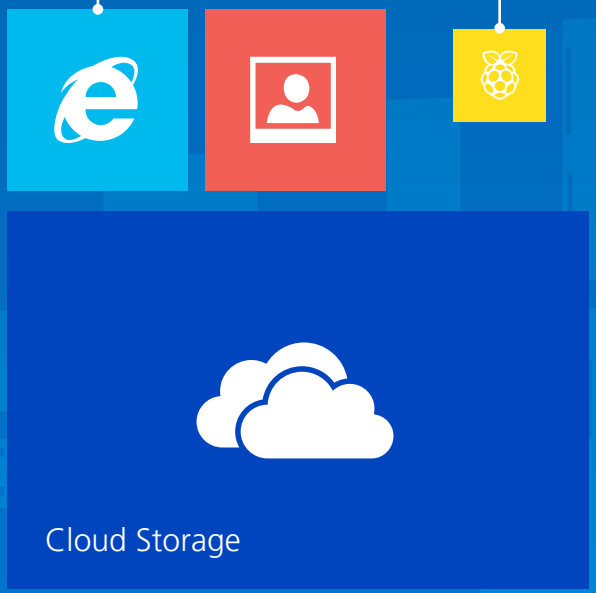
These applications are what Microsoft calls Universal Windows Platform (UWP) apps, the idea being that developers can create software designed for the UWP and deploy it to any Windows 10 device – a desktop, tablet, smartphone, or even Raspberry Pi. There are caveats to the platform's universality – console-based apps will run on a Windows 10 IoT Core device, but won't connect to any on-device console and thus be unavailable for interaction in a way that doesn't apply to other Windows 10 builds. Nevertheless, it's a shift which will, in theory, make development easier for those who are new to Internet of Things and hardware projects.

The concept was first introduced with Windows 8, which had Universal Windows 8 Apps: software that

SUPPORTED PLATFORMS

Microsoft is positioning Windows 10 IoT Core as the one-stop operating system for Internet of Things projects, and while Raspberry Pi support was a clear must-have, the company has also announced compatibility with other devices. The Intel MinnowBoard Max is already supported, while industrial developers can make use of Microsoft's own Sharks Cove board or Qualcomm's upcoming DragonBoard 410C. Oddly, the Intel Galileo, which supports a version of Windows 8.1, is not compatible with Windows 10 IoT Core, and Microsoft has confirmed it has no plans to change that fact. It will also be possible to use Arduino microcontrollers with Windows 10 IoT Core, via Bluetooth or USB connections.

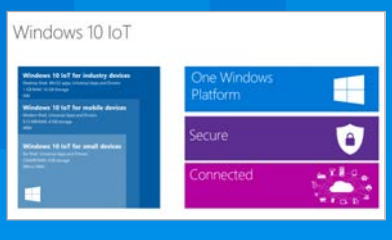
The Raspberry Pi 2 has become the first in its family to receive support for the Microsoft Windows operating system



could be written once and then deployed on both Windows and Windows Phone from a shared codebase. UWP takes the idea further, guaranteeing a core application programming interface (API) layer across all UWP-certified devices and platforms. A single codebase could, in theory, target everything from ultra-powerful Windows 10 desktops and high-end Windows 10 Mobile smartphones, down to the Raspberry Pi 2 and other low-cost, low-power embedded platforms.

For developers who are already well versed in the Windows platform, that's a tempting proposition – and one that should help bring an increasing number of people who have never previously thought about embedded hardware projects across to the Raspberry Pi community.

The Raspberry Pi 2 itself is the ultimate UWP device for these newcomers: it's accessible using standard Windows APIs, and programmable using the same Visual Studio integrated development environment (IDE) as any other Windows 10 device, yet it offers features – such as the general-purpose input-output (GPIO) header – that standard devices lack. Coupled with its extremely low cost and surprising performance, it's to be expected that the full launch of Windows 10 IoT Core for the Raspberry Pi 2, scheduled for later this year, will be a watershed moment in the Raspberry Pi story.



Windows 10 IoT Core is the lowest-footprint entry on a three-tier Windows 10 IoT release map

Microsoft and Open Source

Despite the benefits it will bring and the optional nature of the operating system – the Raspberry Pi Foundation has long stated that its own engineering focus will be on the open-source Raspbian Linux distribution, and the release of Windows 10 IoT Core won't change that – there are those who find the idea of an open and accessible project like Raspberry Pi benefiting from the involvement of a traditionally closed and

WINDOWS INSIDER

Windows 10 hasn't launched officially yet, but it already marks a sea change for Microsoft. Chief among the changes is a shift to a somewhat more open development system, in which the public are invited to try out the software well ahead of a typical public beta release. Dubbed the Windows Insider programme, free membership gives you access to Windows 10 Technical Preview for PCs, tablets, and phones, as well as Windows 10 IoT Core. Microsoft has also promised that the Windows Insiders programme will continue post-launch, bringing pre-release updates and new software to members on an ongoing basis.

insider.windows.com

Video

Microsoft claims that its engineers contribute to around 2,000 open-source projects in total, spread across GitHub and CodePlex collaborative code repositories. While much of this focuses on its Azure cloud computing platform, the company is beginning to open up more of its developer-centric software as well. The company's .NET Core modular development stack was released in 2014 under an open-source licence, and its GitHub repository accepts pull requests – meaning that, for the first time, external contributors can provide code that will make it into the .NET Core stack, fixing bugs or even adding features so long as they match Microsoft's planned roadmap and are of a high enough quality to ship. Naturally, contributors whose features are rejected are free to fork the project and create their own spin-off, as with any open-source project.

The BCM2836 quad-core processor on the Pi 2 gives it the power it needs to run Windows 10 IoT Core.

proprietary company like Microsoft hard to swallow.

It's true that, in the past, Microsoft has been at best indifferent and at worst openly hostile to open-source projects. In recent years, however, this has begun to change and Microsoft has released everything from Android add-ons through to integration tools for open-standard software like Linux, Chef, Puppet, and Docker. They have done this through its Open Technologies subsidiary, recently shut down and reintegrated into Microsoft proper as the Microsoft Open Technology Programmes Office.

Even where its code isn't provided under a free-as-in-speech permissive licence, Microsoft is shifting to make its software more readily available under the free-as-in-beer model. Its primary integrated development environment, Visual Studio, is available in a no-cost Community Edition, while it recently launched a cross-platform code editor with many of the same features and compatible with Windows, Linux, and OS X, dubbed Visual Studio Code.

THE B15 DEMO

B15, shown off at Microsoft's Build conference, is an accomplished wheeled robot powered by a Raspberry Pi 2 and the company's Windows 10 IoT Core operating system, complete with camera 'eye.'

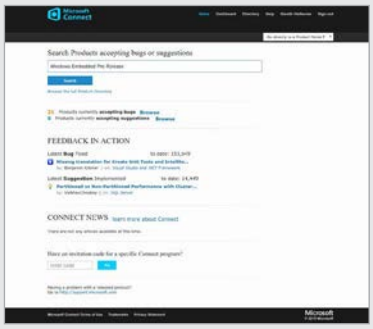
B15's real power is unveiled when coupled with HoloLens, Microsoft's upcoming Windows 10-based augmented reality headset. Responding to a verbal command, a virtual robot – tethered to the real robot – appears, along with a floating user interface.

B15 responds to gestures and voice commands, but the virtual portion is only visible through the HoloLens headset. Anyone without a headset will see nothing but the wheeled physical robot body.

GETTING STARTED WITH WINDOWS 10 IOT CORE

STEP 1: REGISTER

The beta test for Windows 10 IoT Core is public, but requires registration. To get on the guest list, visit **connect.microsoft.com** and sign in with or create a Microsoft Account. Search for 'Windows Embedded Pre-Release', choose the Join option, and read and accept the end-user licence agreements which appear.

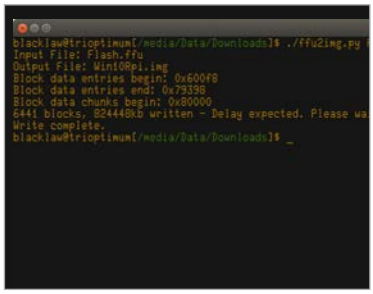


STEP 2: DOWNLOAD

To download the Windows 10 IoT Core SD card image, visit **connect.microsoft.com/windowsembeddedIoT** and click Downloads. Choose 'Windows 10 IoT Core Insider Preview Image for Raspberry Pi 2', click Download, and extract the FFU file from the Zip archive.

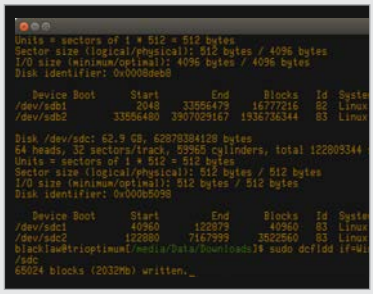
STEP 3: CONVERT

Officially, you need to be running Windows 10 Technical Preview to make use of the IoT Core FFU image. An easier method is to convert the file using the ffu2img.py Python script, downloadable from **github.com/t0x0/random**. Copy the script into a text file, save it, and run it via Python with the name of the image: Flash.FFU.



STEP 4: FLASH

The converted file, Flash.IMG, can be flashed to an 8GB or larger microSD using dd on Linux or OS X, or Win32ImageWriter on Windows, just like any other operating system image. Once flashed, insert the card into the Raspberry Pi 2 – remember, Windows 10 won't work with older models – power up, and follow the on-screen instructions.

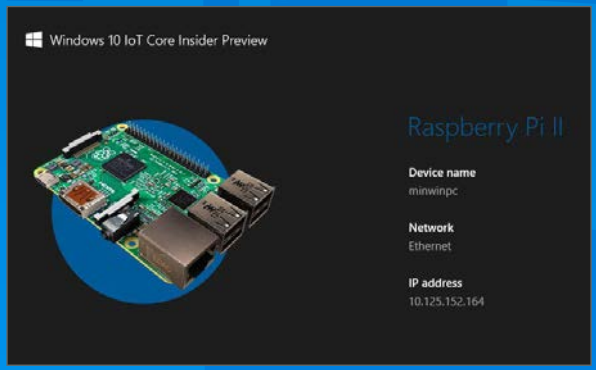


HoloLens and B15

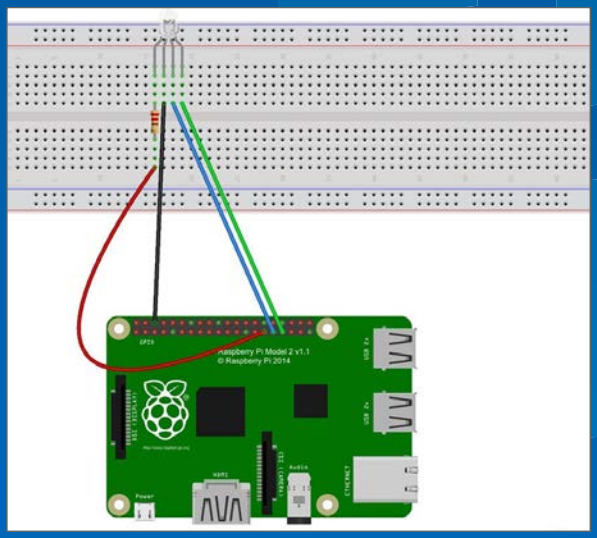
While the publicly available beta test build of Windows 10 IoT is ready for developers to begin experimenting with Raspberry Pi 2-powered Internet of Things projects, there's a second string to Microsoft's bow that won't be readily available for a little while: HoloLens.

First demonstrated during the formal unveiling of the mainstream Windows 10 operating system, HoloLens is a holographic user interface system designed to blend the best of virtual and augmented reality. Wearing a headset with an embedded Windows 10-powered computing system, a HoloLens user can overlay everything from a user interface for music playback to solid-seeming representations of 3D models, and interact with them through simple voice and gesture commands.

The HoloLens system isn't true holography, of course; we're still a little way away from having a Star Trek-style holodeck in the office or living room. The system works by having the wearer peer through translucent display panels located over the eyes. These panels are clear enough to present an unimpeded view of the world when the HoloLens system is inactive, but at any moment can display full-colour and full-motion images. These images do not float over the view of the world, as with traditional head-up displays, but lock themselves in reality using clever position-sensing technology – giving the impression that the objects are truly there.



Microsoft is going out of its way to make Windows 10 IoT attractive to makers



Microsoft's supporting documentation for Windows 10 IoT even includes wiring diagrams created in the popular Fritzing package

SmartGlass

At its recent Build event, Microsoft demonstrated that HoloLens and Windows 10 IoT can work hand-in-hand. A robot, dubbed B15, trundled onto the stage. Battery-powered, and featuring a camera system mounted on an articulated arm, the robot was based on a Raspberry Pi 2 running the pre-release Windows 10 IoT Core operating system. At a command from its HoloLens-wearing user, the robot and the headset synchronised – resulting in a cute, futuristic virtual robot springing into existence above the real-world and rather more prosaic B15.

This virtual robot is tethered to its real-world chassis by the HoloLens positioning system, and can be commanded to perform a variety of tasks – including offering up a gesture-based user interface showing everything from the wearer's emails to weather information. For those not wearing a linked HoloLens system, the real-world robot provides a frame of reference while allowing the system as a whole to interact physically with the world around it.

The Future

The B15 demo gives a taste of where Microsoft is heading in its thinking regarding the Internet of Things as a whole. While there will be plenty of whizz-bang applications like B15, building on work already ongoing regarding robotics and their ability to interact with people in a natural and attractive manner, its vision for Windows 10 IoT is more broad-reaching.

Microsoft's roadmap shows Windows 10 IoT infiltrating the market at every level, from the hobbyist who can download Windows 10 IoT Core for free and install it on their Raspberry Pi, to commercial system builders making interactive kiosk systems, Internet of Things gateways and routers, hand-held IoT hardware, and everything else in between.

> "The B15 demo gives us a taste of where Microsoft is heading regarding IoT"

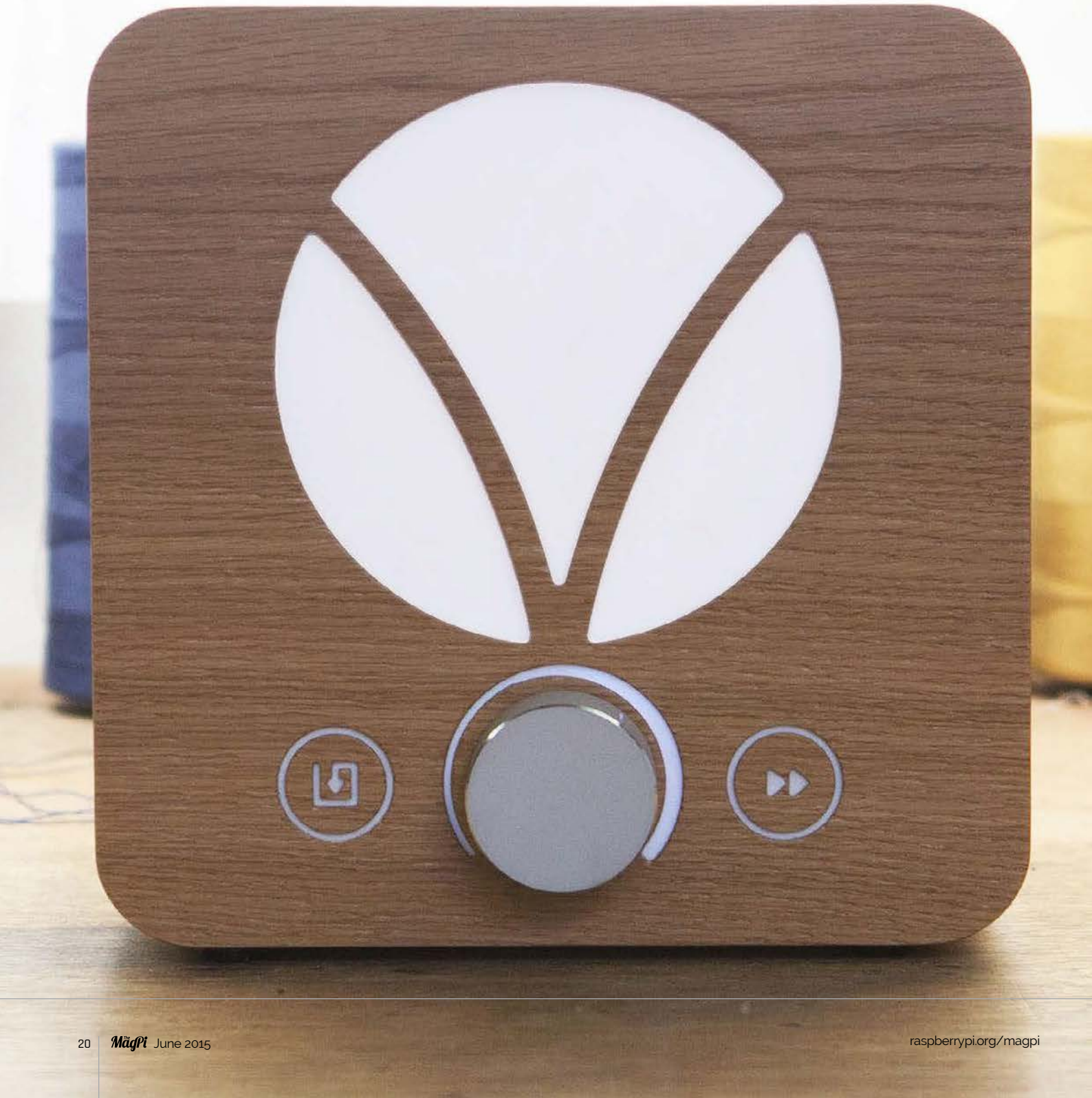
While the fact that Microsoft is not changing its decades-old tune and releasing Windows 10 IoT under an open-source licence means that it will be unsuitable for many projects, its ability to open the IoT market to Windows-centric developers who have never before considered hardware hacking is to be welcomed. Its decision to make the Windows 10 IoT Core platform free for both makers and commercial device builders means that a barrier to entry – the previous requirement to ditch years of experience in learning how to write programs for Windows in order to make use of devices like the Raspberry Pi – has been soundly smashed down.

Having spent much of its corporate life chasing the enterprise customer, Microsoft's sudden love for the maker is surprising but welcome. As well as Windows 10 IoT Core for the Raspberry Pi 2 and other low-cost single-board computer platforms, the company has announced a partnership with open-hardware microcontroller project Arduino to make Windows 10 the first Arduino Certified operating system in history. Should Microsoft's promise of continued support and increasing functionality prove true, it stands to make Windows 10 – and the releases which follow over the coming years – a popular platform in a burgeoning new market.

B15, which combines a Raspberry Pi 2 robot with Microsoft's HoloLens system, offers a glimpse of the future

#HIUTMUSIC

Creative technology agency Knit built an internet radio with a difference to entertain Hiut Denim's workforce...



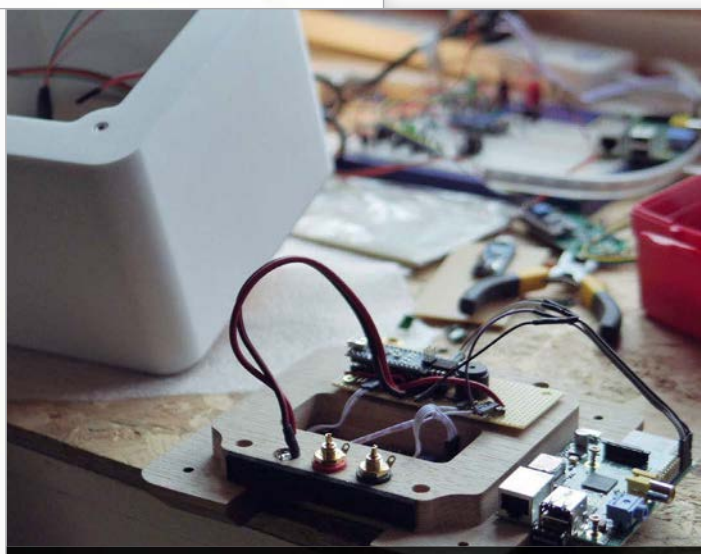
The #HiutMusic jukebox is a rather beautiful Twitter-powered music player that takes pride of place in the Hiut Denim Factory on the west coast of Wales, where 'the music is loud and the coffee is strong'. It was created by Knit, a creative technology agency that approached Hiut Denim with an idea to help customers connect with the boutique jeans company.

"We wanted to facilitate a dialogue between Hiut and their fans through the emotion of music, creating an opportunity for customers to have an impact on the people that make their jeans," explains Jack Chalkley, head creative technologist at Knit.

It's powered by an internet-connected Raspberry Pi, which uses the Spotify and Twitter APIs in a rather novel way. "It plugs into the existing sound system on the factory floor and fans can request a track by posting a tweet that includes #HiutMusic, the artist, and track title," says Jack. The tweet is detected and the song is queued up and played, but the project doesn't end there.

"Hiut's jean makers can skip, save, and adjust the volume of tracks on the face of the radio," continues Jack. "The 'skip' button and volume knob do exactly what you would expect, but the 'save' button saves the current track to a favourites playlist and a tweet is sent from the **@HiutMusic** Twitter account, sharing the request." What's more, the backlit display on the front of the #HiutMusic jukebox changes colour based on how far away the track request was sent. The further the sender, the warmer the colour displayed. "For example, a request from Wales would turn the display light yellow, whilst a request from New Zealand would illuminate deep red," explains Jack.

You can learn more about the #HiutMusic jukebox at **weareknit.co.uk**.



Above The project also uses an Arduino Nano, partly to help reduce the CPU load on the Raspberry Pi, which uses up to 70% on decrypting Spotify tracks

Above A ring of NeoPixels is used to create the geographically controlled light effect

Above Capacitive touch sensors are used for the 'skip' and 'save' buttons

POCKET PIGRRL

Adafruit's Ruiz brothers are back with a 2015 refresh on their brilliant Nintendo Game Boy project that's half the size and twice the fun of its predecessor...

Last year it was the 25th anniversary of the legendary Nintendo Game Boy handheld console; to celebrate, **Adafruit.com** came up with a great project for Raspberry Pi emulation fans, called the PiGRRL.

Suffice it to say that the project was a resounding success and **Adafruit.com** has returned with this 2015 refresh, named the Pocket PiGRRL. It uses the Raspberry Pi Model A+ and a 2.4" PiTFT HAT (with a resolution of 320×240 pixels), making for a much smaller and lighter project than its predecessor. According to its makers, it's about half the size overall, measuring 118mm tall and 69mm wide.

To create the controls, instead of using a SNES controller as before, the makers have opted to use cheap and easily sourced tactile switches soldered to a cut-down Perma-Proto PCB, which is wired to a ribbon cable and connected to the Raspberry Pi's GPIO pins. While some readers might be wondering how you can play *Super Mario World* with just two buttons, you'll also find instructions and 3D printing

files for a four-button version of the Pocket PiGRRL.

To power the project, the Ruiz brothers are using the PowerBoost 1000C, which features a built-in load-sharing battery charger circuit, meaning you can power your Raspberry Pi while it charges the project's ample 2000mAh Li-Po battery. The Pocket PiGRRL even features an audio amplifier and a tiny 1W mono speaker.

While you're free to set up the software side of the project in any way you like, the Ruiz brothers have opted to use RetroPie (**bit.ly/1lYZkDg**), a great emulation package for the Raspberry Pi that won our retro gaming group test back in issue 31.

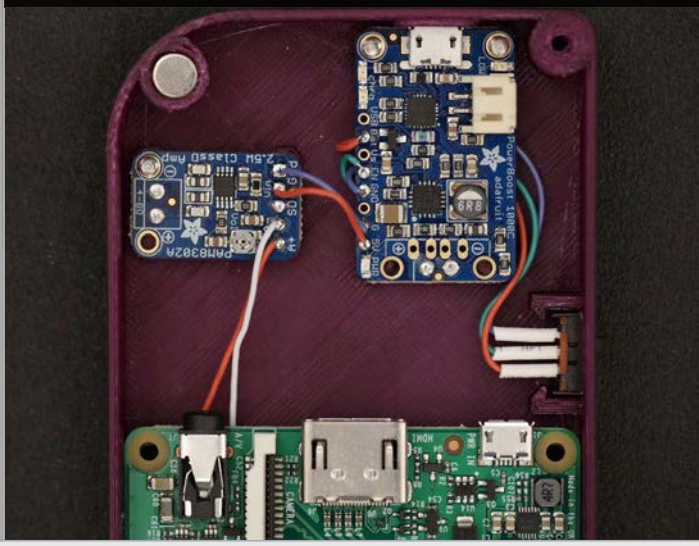
It's fair to say the project isn't particularly taxing to build, though you will at least come out of it an expert solderer. You're definitely going to need helping hands with a large magnifier - some of those joins need precision work!

You can learn more about the project, and find out how to build your own Pocket PiGRRL, on the Adafruit Learning System via **learn.adafruit.com/pocket-pigrrl**.

Below There aren't a great deal of parts for the Pocket PiGRRL, making it a great introduction to more advanced Pi projects

Below The files you need to 3D-print the chassis are available for download on the project's webpage. If you don't have a printer, you can order 3D printed parts online

Below Here you can see the audio amp (top left) and the power charging unit (top right). The three wires leading to the side of the chassis control the power switch



LAURI HAKKARAINEN

Lauri and his friends founded Esmes Digital, a web and mobile application business that sometimes has spare time to make crazy projects.
sneek.co/blog/candypi

CANDYPI

Use your Raspberry Pi to satisfy your sweet tooth with an old-school candy dispenser and new-school technology...

Quick Facts

- The project uses a Raspberry Pi Model A+
- It took the team a day to build the whole thing
- The USB port was removed and attached via wires
- The project's components are wrapped in duct tape to protect against power surges
- Their next project is a Pi-powered 3D printer

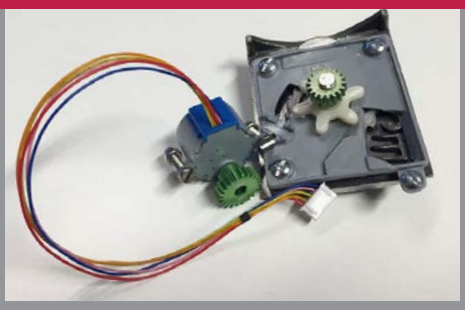
We remember going to the shops as kids and looking at the sweet-dispensing machines. We only needed ten pence to try to get some! Unfortunately, our mother had our health in mind (and probably better sweets in a hidden location anyway), so it was a very rare occasion to actually get any sweets from one of these dispensers. While these machines seem to have all but disappeared, mini versions of them are now popular, offering gifts and trinkets. Inserting coins, though, is old-hat, which is where Lauri and his team at Esmes come in:

"We are huge fans of Jelly Bellys, and a while back we ordered ourselves a small candy machine. We needed to use coins to give us candy, and putting the coin in the slot became boring after a while. So we decided to modify the machine so that we could use a mobile phone to trigger the mechanism, since using coins is so 2014. Complete overkill, but why not! With Raspberry Pi, we could host the mobile front-end on the device itself and interface with a stepper motor controller."

With that, the CandyPi was born: the mobile-phone controlled candy dispenser, with no need to nag your mum. They thought about the concept for a while, but it wasn't until they came across the right gears and screws and other components needed that they decided to actually give it a go.

HOW TO GET A JELLY BEAN

>STEP-01
Find Candy Pi
With CandyPi set up and connected, all you need to do is navigate to the browser interface for the actual machine itself: it makes sense to do this via your smartphone.



>STEP-02
Push the button
No complicated controls or levers to push: all the machine can do is dispense random jelly beans when asked. So there's a single button to press to get to your sweets.

>STEP-03
Eat your sweets
The Raspberry Pi and technological part is done. Before you lies a bounty of sugared and/or sweetened goods. Grasp them with your hand and feast on your victory.

> "When we run the motor, it makes the original dial rotate. Rotating that dial moves the mechanism, which drops the candies to the 'chute'"

Lauri explains how the CandyPi works: "The motor is attached to the rotary dial via gears. When we run the motor, it makes the original dial rotate. Rotating that dial moves the mechanism, which drops the candies to the 'chute'. We decided not to reinvent the wheel and it seems that it was a good idea, since the mechanism is quite stable."

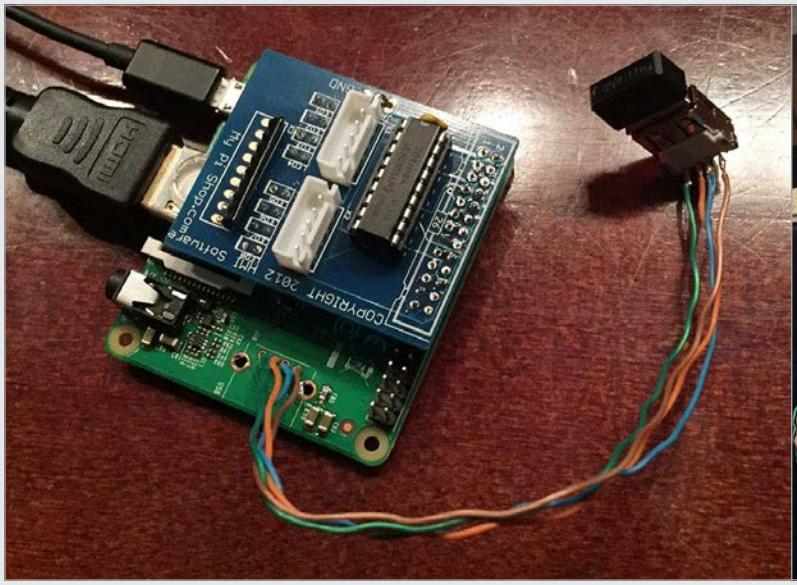
The setup includes the original machine, a stepper motor, some specific gears to get the mechanism working, and a connected Pi with WiFi connectivity to power the whole thing and provide the web interface. Why, specifically, did they use a Pi?

"Raspberry Pi is the favourite embedded platform for us," Lauri says. "Many available tutorials, GPIO, small form factor, cheap price, [and the] possibility to run Linux are huge pros. With Linux, we could host the web front-end easily on the device itself (nginx), and the stepper motor interface was easy to do in Python."

While Esmes won't be selling the CandyPi, pre-assembled or as a kit (it was just a fun little project, after all) the build process is well documented on their website (**sneek.co/blog/candypi**) if you want to give it a go. As for the future of the project, they might add a more powerful motor to make the CandyPi work a little better, but otherwise they'll probably just use it to get sweets whenever they're a bit peckish in the office.

Below left Everything is such a tight fit that the protruding USB port had to be removed and reattached via wire

Below A slight tear-down of the device reveals that it's mainly the candy dispenser and the Pi



MCMASTER FORMULA HYBRID

Members of the McMaster Formula Hybrid team worked together for over 18 months to build a racing car capable of competing in the competition. **formulahybrid.ca**

Instead of using traditional dials, McMaster's Pi-powered racing car uses an LED display to give the driver information about his speed, revs, and lap time

The Raspberry Pi 2 is housed behind the dashboard, where it is sealed in to protect it from the elements. Telemetric data is sent wirelessly to the team in the pits

The front wheels are powered by 15kW electric motors; the rear wheels by a 250cc petrol engine. Sensors throughout the car provide data on how well the hybrid engine is performing

Quick Facts

- The car is quarter the size of a regular Formula 1 car
- The front wheels are powered by an in-hub 15kW electric motor
- The rear wheels are powered by a 250cc KTM SFX motorbike engine
- It can do 0-100 km/h in three seconds flat
- It has a top speed of 150km/h

MCMASTER FORMULA HYBRID

McMaster University needed a smart telemetric system to get its hybrid racing car onto the winner's podium, and the Raspberry Pi 2 provided everything the team could ask for

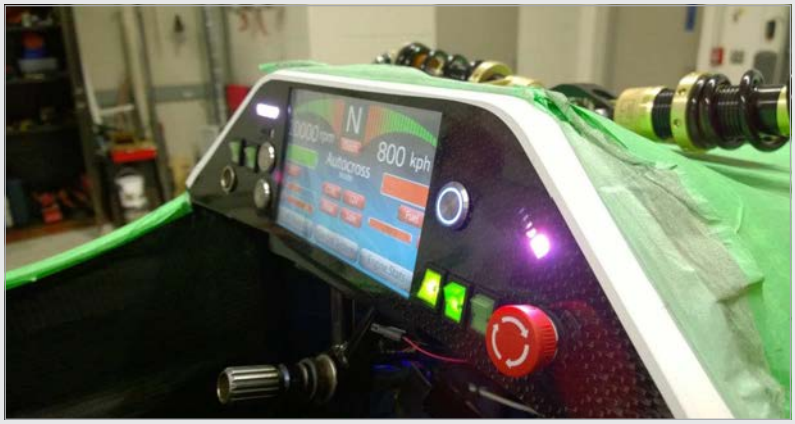
When the engineering students from McMaster University in Canada started working on their entry for the 2015 Formula Hybrid and EcoCAR 3 competitions, they knew they'd need more than raw muscle to get a podium finish. Not that their race car lacks muscle: it packs a 15KW in-hub motor for the front wheels, and a 250cc motorcycle engine for the rear wheels.

The challenge was bringing all the technology together and keeping the car going. McMaster's secret weapon was a Raspberry Pi 2, used to gather telemetry data and send it to the team on the trackside.

We caught up with Jonathan Moscardini, LV lead for the electrical division of McMaster University's Formula Hybrid and EcoCAR 3 teams. He tells us that the Raspberry Pi has been crucial to the car's success. "We're a pretty big team, between 70 and 90 people. But our project is entirely student designed, built, and tested. The students do everything."

McMaster has been entering the Formula Hybrid and EcoCAR 3 competitions for several years now and had pretty much perfected the physical car: "It has a very clever monobox," asserts Jonathan. "It's one big fibre tub."

Physically the car was fine, but that's not enough to win a race. "We do a lot of electronics in the car," says Jonathan, "a lot more than we need to" and that's where the Raspberry Pi steps in. "What the Pi does for us is handle all of our communications. Essentially, it's



Above The Raspberry Pi 2 sits behind the dashboard and powers the display. When the car is racing, it automatically sends telemetric data to the team in the pits

both the dashboard computer and our team radio. It also gives us a few new features along the way, simply because it's so powerful."

Good telemetric data is essential when building a race car. "We have live up-to-date information about everything that's happening in the car," says Jonathan. "We use a wireless adapter known as a Bullet [BULLET WirelessHART Adapter], which is built for a variety of outdoor uses. We have one at either end: one in the car and one in the pits."

The telemetry data enabled McMaster students to analyse the car as it raced around the track, which helped them fix a variety of engineering challenges. "We were having reliability issues," divulges Jonathan. The team used the Raspberry Pi with various sensors to analyse parts of the car as it went around the track. "When we installed a Raspberry Pi 2 it made life a lot easier."

While they initially installed a Raspberry Pi 2 to gather telemetry data, its use soon expanded to other aspects of the car. McMaster engineering students quickly realised they had enough power to build an electronic dashboard. "We also installed the PiCam [Raspberry Pi Camera module]," says Jonathan, to get first-person video recording from the car. The Raspberry Pi was sealed behind the steering wheel, away from the elements. "You really need to be able to seal and waterproof the computer," Jonathan tells us. "You need to keep it away from the elements".

So what if you want to start integrating your Raspberry Pi with a track car? "Planning and testing is one of the biggest things," advises Jonathan. "Race cars are not the place to be trying things out. As far as implementing goes, planning and testing is really important because it's a lot easier to work on things when you're not in the car. So you really need to have everything installed before you start."

Above The car is much smaller than a Formula 1 car, but with a top speed of 150km/h it is much more than a standard kart

INSTALLING A RACE-CAR COMPUTER

>STEP-01

Raspberry Pi

The Raspberry Pi is connected to custom PCBs built by Advanced Circuits (**4pcb.com**), and a stock video display (picked up from eBay). The Raspberry Pi Camera module is also added (to provide video recording of the race circuit).

>STEP-02

Sealed in

The Raspberry Pi electronics kit is located behind the steering wheel and is sealed in using Loctite 243 and Henkel (henkel.com) adhesives, sealants, and functional coatings. This protects it from the elements. The display is mounted on the dashboard behind the (detachable) steering wheel.

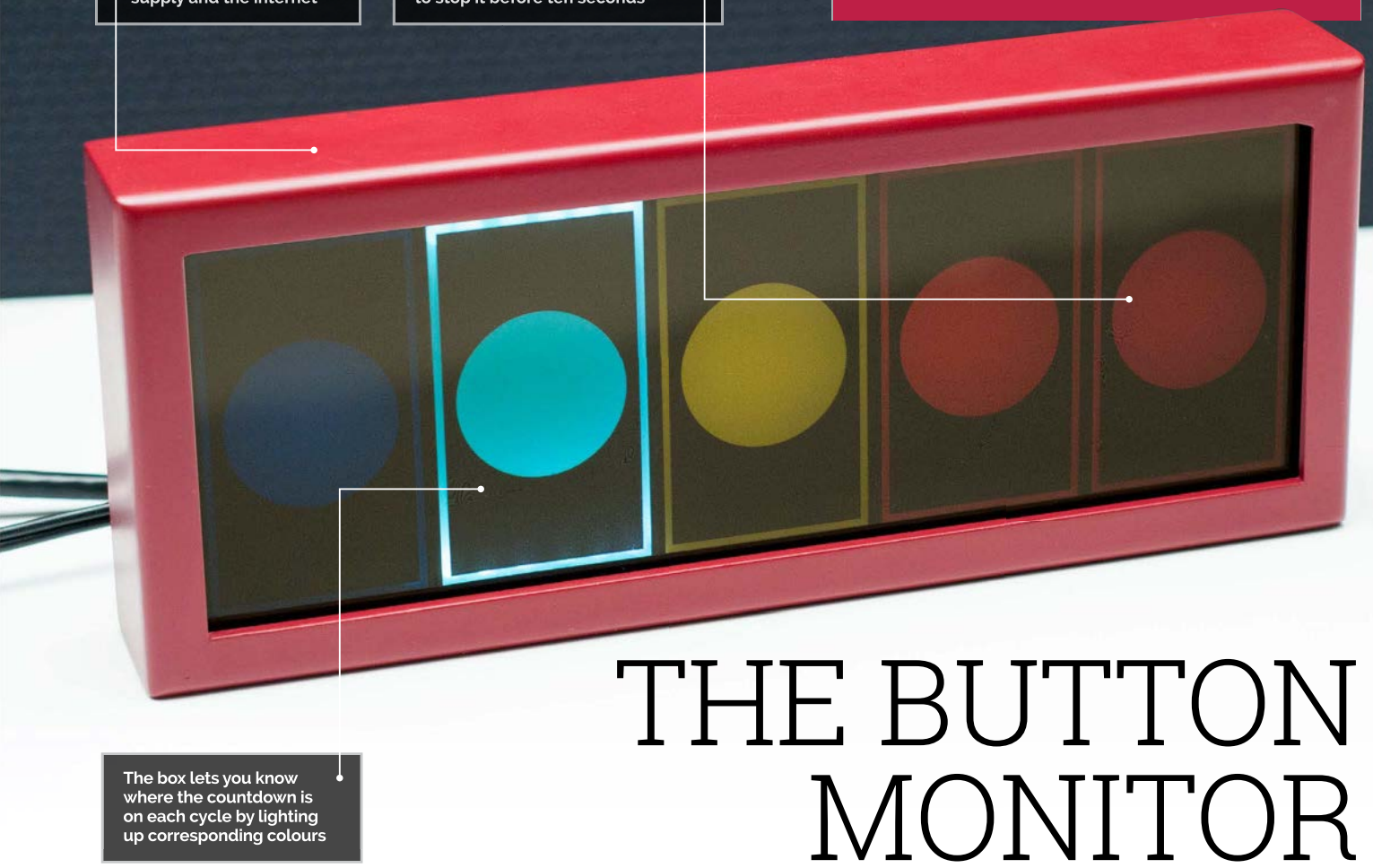
>STEP-03

Trackside telemetry

The Raspberry Pi is hooked up to a BULLET WirelessHART Adapter, which communicates with another Bullet at the trackside. Data captured by the Raspberry Pi is sent wirelessly to the team working on the car.

TRAVIS BROWN

A front-end web developer from Florida who used to work as a Linux server admin, Travis currently works in the advertising sector.
bit.ly/1KAz6US



THE BUTTON MONITOR

Quick Facts

- Nearly a million people have pressed the button
- Reddit has decided that the winner will be called the 'Pressiah'
- Travis has also made the Pi-Pocket Game Boy
- The project took minimal time due to it being a refit
- Travis has not pressed the button yet

Have you been taken in by Reddit's button? The bizarre social experiment has become prime fodder for a fun Raspberry Pi project...

Have you heard of the button? It's a countdown timer on a subreddit on the aggregate website **reddit.com**. The object is very simple: it counts down from 60 and you've got to press the button when the timer hits zero. Sounds very easy, right? Only, everyone sees the same countdown and if they press the button too early, it will reset.

You also only have one chance to press the button on your account. You can't create a new account to try again, as only accounts from before April 2015 are permitted to press the button. It's an incredible social experiment, with not even a prize announced for whoever wins the game. With trolls, the uninformed, and people vying for just a good 'score', it has predictably not been won yet and very rarely gets below 30 seconds.

Travis has created a little system to track the countdown of the button in real-time. "The Button Monitor connects to the Reddit WebSocket powering the button itself," he tells us. "It displays a physical real-world indication of the current status of the button. It's a rework or rebranding of a previous project where there were only five positions for each of the DEFCON (Defense Condition) levels. The purple flair colour is represented by the counting down of all five lights until only one particular circle is lit, indicating that should I press the button. The blue flair is what I would receive."

The 'flairs' are a little graphic that appears next to your username in a subreddit. For the button subreddit, these indicate when you pressed the button. The project itself is very simple, containing only the Pi, frame, some heavyweight posterboard, transistors, a handful of LEDs, and a few sheets of transparency film.

"It seems to work really well," Travis says. "I've never had issues with the Pi itself and overall it is incredibly stable. The Pi has ample resources for a project like

this and responds very quickly to web requests... The Pi is brilliant in that, at least in my case, there is no compiling, firmware burning, or rebooting necessary for each change that needs to be made. This makes it great for rapid prototyping; I can edit files right over SSH, or [you can] use a program like Notepad++ [if] you need a more IDE-like experience.

Above The Pi and the circuitry fit neatly in the case, making it compact and mobile

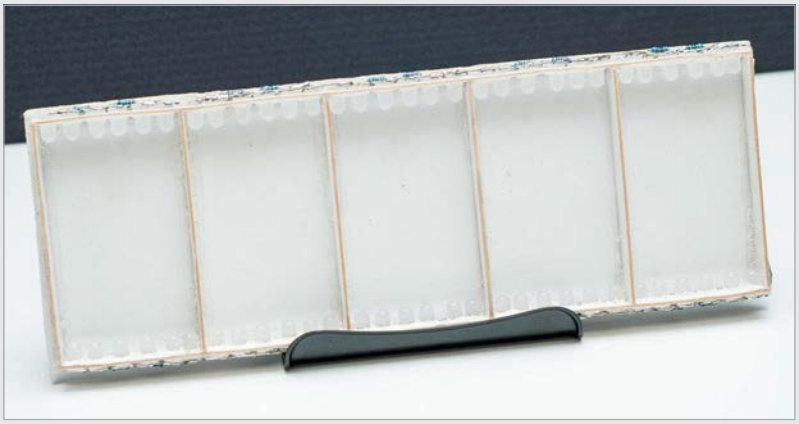
box will continue to be a Pi project of some kind.

"The project was originally a DEFCON Sign and this is what I will probably return it to once the button has finally named the 'Pressiah', though there are several things I could tie it to in the future. For instance, we have several servers at our office and in the cloud that we use to host

> "Travis has created a little system to track the countdown of the button in real-time"

Most importantly, with things increasingly becoming web enabled, having a full Linux distro and the power of its network stack is unparalleled by something like an Arduino and WiFi Shield."

What's the future of the monitor then? Travis sees it as only a temporary thing, but the client websites and applications, so it could be turned into a server health meter. Another option would be to tie into a service like Google Analytics to gauge the flow of traffic for a particular website or trending topics. Anything really that can be quantified or represented by a five-position light box."



Left The basic frame is interchangeable and usable in multiple projects

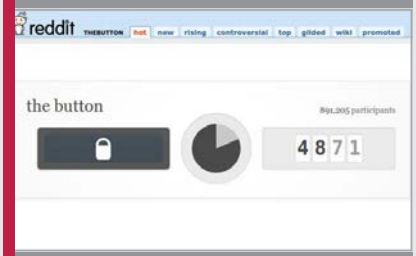
MONITOR THE BUTTON

>STEP-01
Turn on the system

Turning on may seem like it's obvious, but when the system is set up, it tweets out an IP address to an anonymous account that you need to use.

>STEP-02
The right mode

Currently, the project is set up in multiple ways so Travis can select a different mode each time, such as DEFCON and the button.



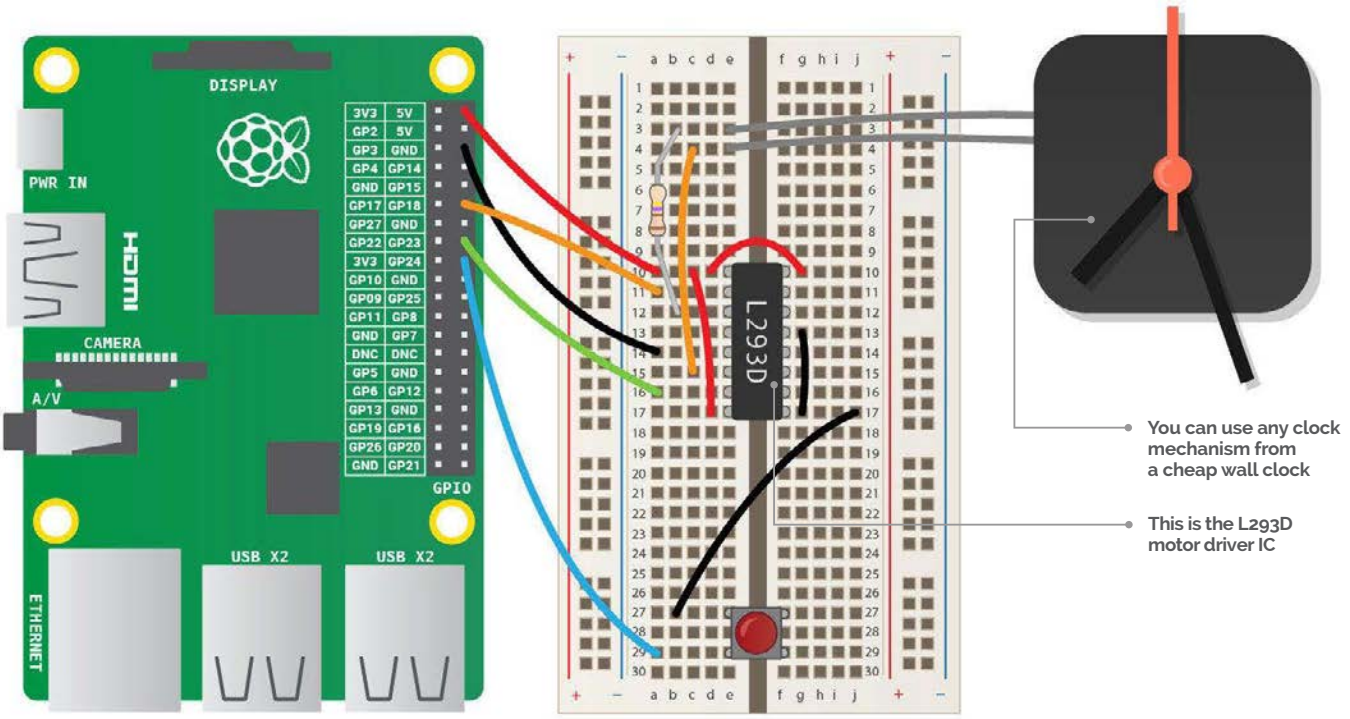
>STEP-03
Sit back

Use the box to monitor the button and see what flair you could get by actually pressing it. Do you dare sacrifice your one press?

EVERYDAY ENGINEERING PART 4

SIMON MONK

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



HACKING AN ANALOGUE CLOCK

You'll Need

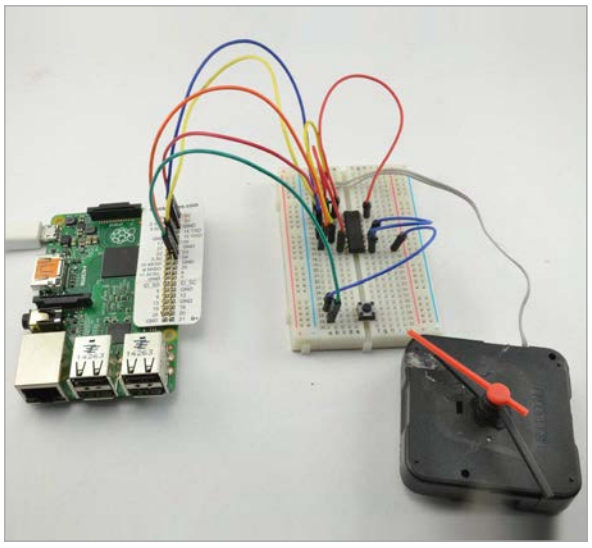
- Half-size breadboard
- Quartz wall clock
- L293D motor driver IC (Adafruit: 807, CPC: SC10241)
- Tactile push switch
- 5 male-to-female jumper wires
- 5 male-to-male jumper wires
- 470 ohm resistor
- Thin hook-up wire

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

Analogue wall clocks use a special motor that advances the second hand by one second each time the current through the motor is reversed. Normally, the motor is controlled by a chip inside the clock mechanism. With a little bit of hacking, you can disconnect this chip and have the Raspberry Pi control the speed of the clock motor. As a simple example, this project has a button that, when pressed, increases the speed of the clock to five ticks per second. You can see a video of the project in action at **youtu.be/m8aqpq9AINw**.

As you'll see from the list of required components, this project uses a breadboard, an L293D motor controller chip, and a wall clock.

When it comes to the wall clock, the cheaper the better. Ours came from Asda supermarket and cost about £3 ($5). To make this kit, you are going to be taking the clock mechanism off the clock itself,



Above Tick tock... everything is wired up

Above A low-cost quartz clock

opening the mechanism up, and then modifying it for use with the Pi. There is a significant risk that all the cogs will fall out and be hard to put back together. The mods to the mechanism also mean that the clock will not then work without a Raspberry Pi after you have modified it, so do not use an expensive clock.

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.

The L293D chip is widely available and can be bought from Adafruit, CPC, and indeed most component suppliers.

You need thin wires to connect to the clock mechanism's coil, as they will need to exit the mechanism's case. We used a short length of two wire cores cut off from an old computer IDE cable that we happened to have lying around, but any fairly thin wires will do.

Quartz clock mechanisms

The motor used in quartz clocks is a kind of stepper motor that uses only a single coil. To make it move, you have to supply it with alternating positive and negative pulses of around 30 milliseconds. So on the first tick, A will be positive and B negative, but then after a second, the polarity must be reversed – with A negative and B positive – for the motor to move on a step.

Each tick causes the second hand of the clock to move forward one second. Gears cause the minute and hour hands to also advance at their own rates. So, you cannot control each hand separately.

If you supply pulses faster than one per second, you can get the second hand to whizz round quite quickly. Similarly, if you leave a longer gap between pulses, you can slow the clock right down.

Hacking your analogue clock

There are really two parts to this project: hacking the analogue clock mechanism and building a motor controller on the breadboard.

BUILDING THE PROJECT

Taking the clock mechanism apart is easy, but preventing all the little cogs falling out is harder. A good trick is to photograph each step of the process, so that if things do go wrong, you can look at the photo when putting it back together.

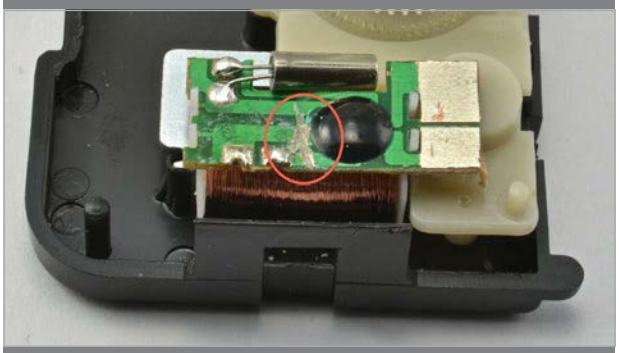
When building the motor controller, follow the annotated breadboard diagram, taking care to ensure that all the components and jumper wires fit into the correct rows on the breadboard.

>STEP-01

Disassemble the clock mechanism

Carefully remove the hands from the clock, then remove the battery, and detach the mechanism from the clock body. You then need to carefully take the mechanism's case apart. Usually, these are just held together with clips. Pull it apart gently at first and try to keep all the cogs in place.

You can see the tiny green PCB containing the clock's electronics. The next step is to modify this.

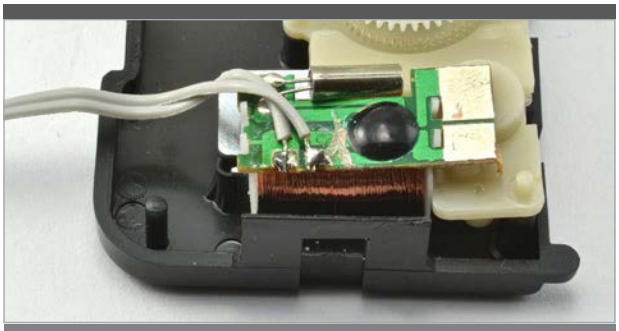


>STEP-02

Isolate the coil

This project does not use the clock's electronics: it drives the clock coil directly. If you look carefully at your clock mechanism's PCB, you will see that the very thin coil wires lead back to two solder pads. PCB tracks then lead back to the controller chip on the PCB (black blob).

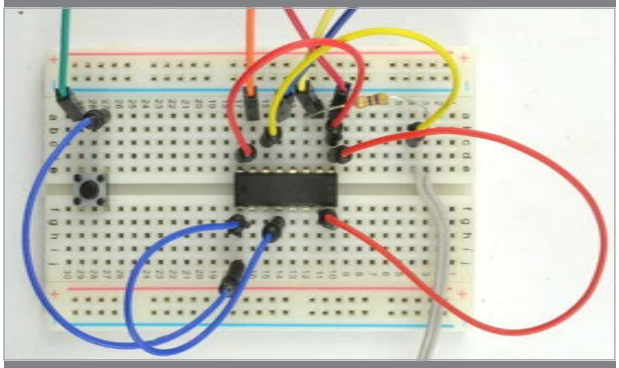
To prevent any possibility of the clock electronics interfering with the motor driver chip on the breadboard, isolate the coil by using a screwdriver to scrape a gap in the tracks that lead to the two coil pads on the PCB. The cuts in the PCB tracks have been circled in the photo.



>STEP-03

Solder extension wires to the coil

Carefully solder a few inches of thin insulated wire onto the coil pads. You can now carefully reassemble the clock mechanism. The wires need to leave the mechanism case and should be thin enough that the case just clips over them.

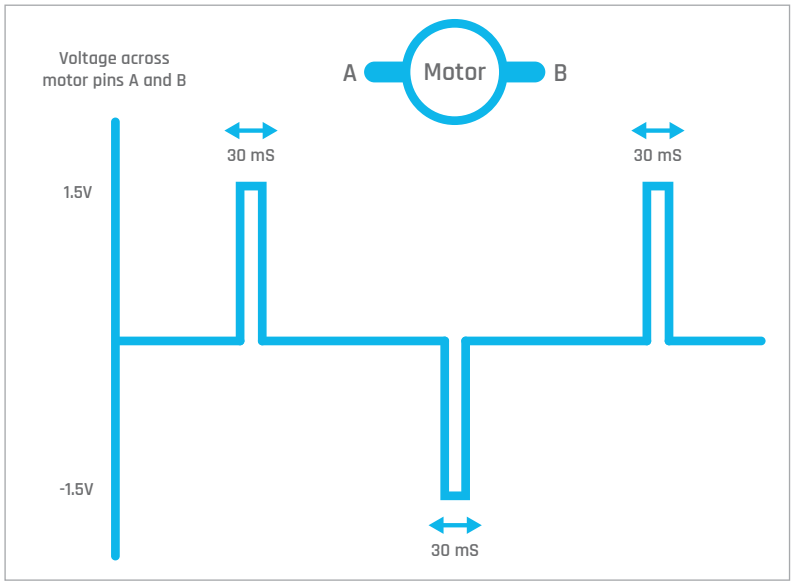


>STEP-04

Build the breadboard

Build the breadboard; start with the IC, which will have a notch on one end. Position the notch end on row 10 of the breadboard. It does not matter which way around the resistor goes, and the switch will only fit across the central gap the correct way around.

Finally, link the breadboard to your Raspberry Pi using the female-to-male jumper wires.



Below Driving the motor of a quartz clock

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 the **RPi.GPIO** library that is pre-installed on Raspbian.

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

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

This command will actually bring down the code for all the projects in Simon's MagPi series, so if you have already issued this command for one of his earlier articles, change directory to **pi_magazine** and run the following command to update your directory with this project (**04_analog_clock**).

```
git pull
```

How the code works

The Python code for this program is well commented. You will probably find it handy to have the code up in an editor while we go through it.

The program starts by importing the **RPi.GPIO** and **time** libraries that it needs. Some constants are then defined. The only one that you may need to alter is **PULSE_LEN**. This provides the duration of the pulses sent to the motor coil. If the coil tries to move, but doesn't quite tick, then try altering this value up and down a bit.

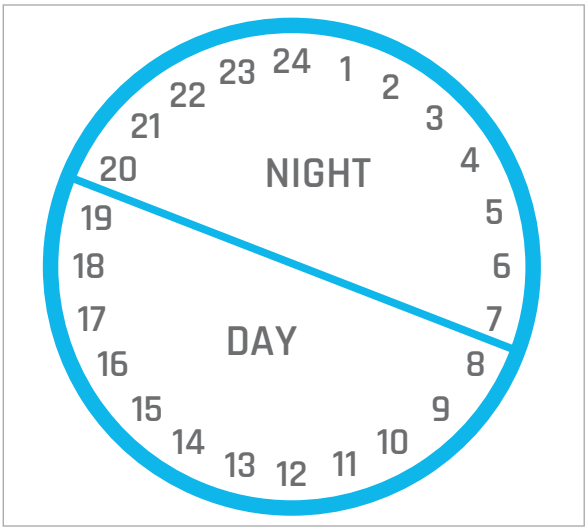
The global variable **positive_polarity** is used to keep track of whether a positive or negative pulse is due next tick (remember that they have to alternate). The variable **period** is the delay between each tick; it defaults to one second. **last_tick_time** is used to record the last time there was a tick. If you were to just use **time.sleep()** to add delays between each tick, then the clock would be very inaccurate as the time taken to do the motor driving would not be accounted for. Later on in the code, you will see how this variable is used to work out whether another tick is due without using **time.sleep()**.

The **tick** function advances the second hand by one second. It does so by calling **pulse** to generate the actual pulse of the right duration and polarity.

The main program loop starts by finding the current time (**t**). If this is **period** seconds after the last time a tick occurred (**last_tick_time**), then **tick** is called and **last_tick_time** set to **t**, ready to wait until another **period** of time has passed.

The main loop also checks to see if the switch button is pressed. If it is, then it sets **period** to 0.2 (1/5 second), otherwise it sets it to 1 second. This loop will cycle round much faster than once a second, but remember that a tick only happens if enough time has elapsed according to the Raspberry Pi's internal clock and so the timing is accurate.

The **try** / **finally** clause is used to set the GPIO pins back to inputs when the program is quit using **CTRL+C**.



Using your hacked analogue clock

Before running the program, you may want to set the position of the hands to the correct time. The hands should just reposition without turning the cogs, apart from the second hand.

To run the program for the clock, change directory to **04_analog_clock** and enter the command:

```
sudo python analog_clock.py
```

The hands of the clock should start ticking round as normal. When you press the button, the hands should speed up. If the second hand judders rather than moves properly, try altering the **PULSE_LEN** constant.

You will find a couple of other programs in the **04_analog_clock** directory. The program called **analog_clock_set_time.py** will advance the hands to the correct time when the button is pressed. To use it, set all the hands to 12 o'clock and then press the button. The clock will then tick at ten times real time until the hands show the Raspberry Pi's system time.

The program **analog_clock_24.py** only ticks every 2 seconds and is designed to make a 24-hour clock, where there are 24 numbers on the dial rather than 12. To use this, you need to make a new clock face.

There are lots of other things that you could make with this project. For instance, you could get the clock to display the next available train departure, automatically advancing to the next train time as the current one is due to depart.

The project only uses one half of the L293D motor controller, so you could use the other half to control a second clock mechanism and build a world clock showing the time in two time zones.

You do not have to use all the clock hands, so you could just use the second hand as an indicator that shows the weather.

NEXT MONTH

In the next project in this series, you will learn how to turn your Raspberry Pi into a simple internet radio.

Analog_clock.py

Language >PYTHON

```
import time
import RPi.GPIO as GPIO

# Constants
PULSE_LEN = 0.03 # length of clock motor pulse
A_PIN = 18       # one motor drive pin
B_PIN = 23       # second motor drive pin
BUTTON_PIN = 24

# Configure the GPIO pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(A_PIN, GPIO.OUT)
GPIO.setup(B_PIN, GPIO.OUT)

# Global variables
positive_polarity = True
period = 1.0          # default tick period of 1 second
last_tick_time = 0    # the time at which last tick occurred

def tick():
        # Alternate positive and negative pulses
        global positive_polarity
        if positive_polarity:
                pulse(A_PIN, B_PIN)
        else:
                pulse(B_PIN, A_PIN)
        # Flip the polarity ready for the next tick
        positive_polarity = not positive_polarity

def pulse(pos_pin, neg_pin):
        # Turn on the pulse
        GPIO.output(pos_pin, True)
        GPIO.output(neg_pin, False)
        time.sleep(PULSE_LEN)
        # Turn the power off until the next tick
        GPIO.output(pos_pin, False)

try:
        while True:
                t = time.time()
                if t > last_tick_time + period:
                        # its time for the next tick
                        tick()
                        last_tick_time = t
                if GPIO.input(BUTTON_PIN) == False:
                        # the button is pressed
                        period = 0.2
                else:
                        period = 1.0
finally:
    print('Cleaning up GPIO')
    GPIO.cleanup()
```

RICHARD SMEDLEY

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

COMMAND LINE PI PART 4: MANIPULATING TEXT

Richard Smedley presents your cut-out-and-keep guide to using the command line on the Raspberry Pi. In part 4 we discover pipes, and connect together multiple simple commands for more powerful text processing

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.

The Unix family of operating systems – which includes other flavours of GNU/Linux and also Apple's Mac OS X – deals with data from commands as streams of text. This means that commands can be chained together in countless useful ways. For now, though, we'll focus on giving you a firm foundation to building your own custom commands.

Getting our feet wet

When a command is called at the terminal, it is given three streams, known as standard input (stdin), standard output (stdout), and standard error (stderr). These streams are plain text, and treated by the Pi as special files – 'everything is a file', as we mentioned in part 3; that's what gives the Pi and other Unix family systems the ability to put together simple commands and programs to build complex but reliable systems. Normally, stdin is what you enter into the terminal, while stdout (command output) and stderr (any error messages) appear together. The reason the last two have a separate existence is that you may want to redirect one of them – error messages for example – somewhere apart from the regular output your commands produce. We'll look at separate error messages later, but first we need to know how to redirect and connect our output to other commands or files.

Connecting commands together are pipes, the '|' symbol found above the backslash on both GB and US keyboards (although the two keyboards for English speakers place the \ respectively to the left of Z, and at the far right of the home row). When you type a command such as **`ls -l`**, the output is sent by Raspbian to the stdout stream, which by default is shown in your terminal. Adding a pipe connects that output to the input (stdin stream) of the next command you type. So…

```
ls -l /usr/bin | wc -l
```

…will pass the long listing of the **/usr/bin** directory to the wordcount (**`wc`**) program which, called with the **`-l`** (line) option, will tell you how many lines of output **`ls`** has. In other words, it's a way of counting how many files and folders are in a particular directory.

Search with grep

One of the most useful commands to pass output to is **`grep`**, which searches for words (or Regular Expressions, which are powerful search patterns understood by a number of commands and languages), like so:

```
grep if ~/python_games/catanimation.py
```

This displays every line in the **catanimation.py** file containing the character sequence 'if' (**Fig 1**)– in other words not just the word 'if', but words like 'elif' (Python's **`else if`**), and words like 'gift' if they were

ABSOLUTE PATH

We're using **`~/mylisting4.txt`** with **`~`** short for **`/home/pi`**. If you **`cd`** to **`~`** then you can just use the filename without the **`~/`**

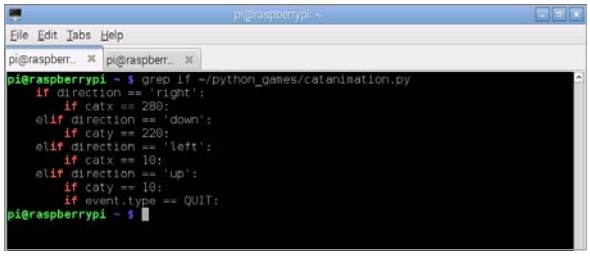


Fig 1 No matter how long the file, `grep` will dig out the lines you need . It's also handy for finding the results you want from a multi-page output

present. You can use Regular Expressions to just find lines with 'if', or lines beginning with 'if', for example.

Piping search results and listings to **grep** is the way we find a needle in one of Pi's haystacks. Remember **dpkg** from the last part, to see what was installed? Try…

```
dpkg -l | grep -i game
```

…to remind yourself which games you've installed (or are already installed). The **-i** switch makes the search case insensitive, as the program may be a 'Game' or 'game' in the description column. A simple **dpkg -l | more** lets you see output a page at a time.

sort will, as the name suggests, sort a listing into order, with various tweaks available such as **-f** to bring upper and lower case together. One way to collect unsorted data is to combine lists. **sort** will put the combined listing back in alphabetical order:

```
ls ~ ~/python_games | sort -f
```

Suppose you copied one of the games to your home directory to modify: you know it's there, but you don't want to see the same name twice in the listings. **uniq** will omit the duplicated lines or, with the **-d** switch, show only those duplicates.

```
ls ~ ~/python_games | sort  -f| uniq
```

File it away

Pipes are not the only form of redirection. **>** (the greater than symbol) sends the output of a program into a text file, either creating that text file in the process, or writing over the contents of an existing one.

```
ls /usr/bin > ~/mylisting4.txt
```

Now look in **mylisting4.txt** and you'll see the output of **ls /usr/bin**. Note that each item is on a separate line (**Fig 2**). Your terminal displays multiple listings per line for space efficiency; however, for easy compatibility between commands, one listing per line is used. Most commands operate on lines of text: for example, **grep** showed you in which lines it found 'if'. Note that some commands need a dash as a placeholder for the stdin stream being piped to them:

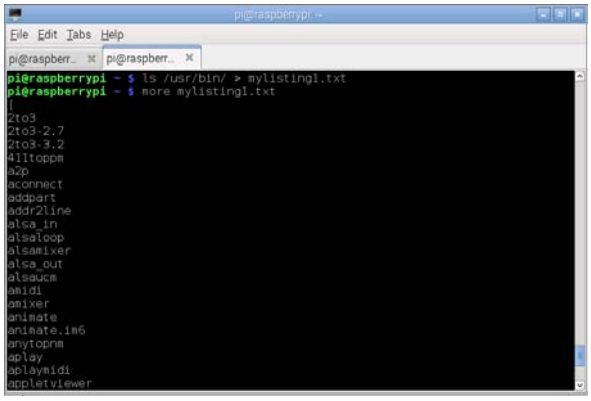


Fig 2 With redirection, you can get all of the output from a command saved straight into a text file. Save your error messages to ask about them on the forums!

```
echo "zzzz is not a real program here" |
cat mylisting4.txt -
```

Appending

If you want to add something to the end of a file without overwriting the contents, you need **>>**.

```
echo "& one more for luck!" >> ~/
mylisting4.txt
```

echo simply displays whatever is in the quote marks to stdout; the **-e** switch lets you add in special characters, like \n for newline (see below). Remember, you can look at the last few lines of a file with **tail ~/mylisting4.txt**. **<** will link a program's input stream to the contents of a file or stream. Make an unsorted list to work on, and **sort** it:

```
echo -e "aardvark\nplatypus\njellyfish\
naardvark" > list1
sort < list1
```

You can also combine **<** and **>**:

```
head -n 2 < list1 > list2
```

…will read from **list1**, passing it to **head** to take the first two lines, then putting these in a file called **list2**. Add in a pipe:

```
sort < list1 | uniq > list3
```

Lastly, let's separate that stderr stream: it has file descriptor 2 (don't worry too much about this), and **2>** sends the error messages to any file you choose:

```
cat list1 list2 list3 list42 2>errors.txt
```

The screen will display the 'list' files you do have, and the 'No such file or directory' message(s) will end up in **errors.txt** – **2>>** will append the messages to the file without overwriting previous contents.

COMMAND HISTORY

Forgotten that clever thing you did last time, or where you got that download? Invoke history. **history | grep wget**, for example, will find all recent uses of **wget**.

FILING HOMEWORK

There are many more commands beyond **grep**, **sort** and **uniq** that can be chained together. Take a look at **cut** if you're feeling adventurous.

SEAN MCMANUS

Sean McManus is a Code Club volunteer, and wrote the book *Scratch Programming in Easy Steps*. He also co-wrote *Raspberry Pi For Dummies*.
sean.co.uk
twitter.com/musicandwords

MAKE A MULTIPLE-CHOICE QUIZ IN SCRATCH

Dazzle your friends with your own quiz game, containing hundreds of questions! How many can they get right in 30 seconds?

You'll Need

- LibreOffice – Install it from the command line with: `sudo apt-get install libreoffice`
- List of capitals by size – **wki.pe/List_of_national_capitals_by_population**
- Internet access

Lists are used to remember lots of information, but adding items to them block by block can take a lot of time and Scratch code. In this project, you'll see how you can import (or bring in) large lists from other places, so you can easily make a quiz game with hundreds of questions. As you create this game, use your own favourite background and sprites, and arrange them with enough space for the answers to appear. Perhaps you can add your own question list? Anything works, as long as each answer only applies to one question.

>STEP-01

Gather your data

For this game, you'll need two text files: one for the questions and one for the answers. We're going to make a quiz about capital cities, so one file will contain a list of capitals, and the other will contain the countries they are in, in the same order. Start by finding the list of capital cities by population on Wikipedia. Click and drag over the table to highlight it and then press **CTRL+C** to copy it. It's easier if you highlight from the bottom up. Be patient when the screen scrolls!

Click to answer. The answer data comes from a list on Wikipedia

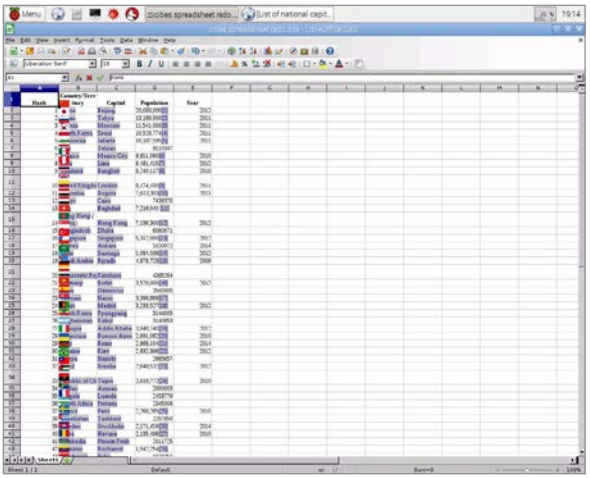
The game runs for 30 seconds before it ends

>STEP-02

Create your question files

Start LibreOffice Calc and paste in the table using **CTRL+V**. Click OK. This might take a minute or two to work. Click above your cities column to highlight it.

Above You can get a well-organised list of capital cities from this page on Wikipedia



Above Copy the capital cities table from Wikipedia into LibreOffice Calc to make it easy to extract the columns separately

Press **CTRL+C** to copy the column. Open your text editor, Leafpad, which is in the Accessories menu. Press **CTRL+V** to paste. You should now have a text file containing just capital cities, each one on a new line. If you have a heading at the top (the word 'Capital'), delete it, and remove any blank lines at the end too. Save this file as **`cities.txt`**. Open a new file in Leafpad and repeat the process with the countries column in LibreOffice Calc. This time, save your Leafpad file as **`countries.txt`**.

>STEP-03

Importing your data into Scratch

Start Scratch. Click the Variables button and make a list. Call it **`cities`** and make sure it's for all sprites. When the empty list appears on the Stage, right-click it and click **`import`** in the menu. Browse to the files you just created, and double-click your cities text file. The list on the Stage will be filled with the cities from your file. Repeat the process to make a list called **`countries`** and fill it with your countries file. Your list files should be the same length. Right-click the list boxes on the Stage and choose **`hide`**.

>STEP-04

Set up your variables

Through the Variables part of the Blocks Palette, make variables called **`question number`** (used to remember which question/answer pair we're asking), **`score`**, **`shuffle choice`** and **`temporary storage`** (used for shuffling the list of options), and **`wrong answer`** (used when making the list of wrong options). You also need to make a variable called **`player guessed`** to remember which answer the player chooses, and a list called **`possible answers`**. Make all these variables and the list "for all sprites".

>STEP-05

Make the main game code

The main game code uses three scripts (Listing 1). Add them all to the cat sprite. The game uses broadcasts to pass control to the various parts of the program, including on the same sprite. The 'ask a question' section picks a random question number from the list of countries and makes a list of possible answers. It includes the correct answer, and two wrong answers which must be different from the correct answer. The code then shuffles this list to put the answers in a random order, before using a broadcast to make the answer sprites appear and show their answers.

>STEP-06

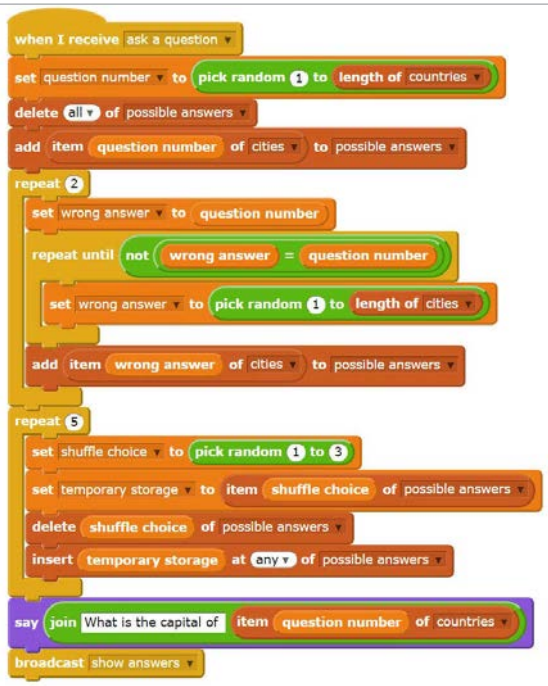
Make the answer sprites

Import a new sprite to use for showing the answer. We're using Gobo. This sprite has five short scripts (Listing 2). Make the variable **`answer choice`**, but click the button to make it for this sprite only. If the game shows all the same answers when you run it, you probably made a mistake here! When you've finished this sprite, right-click it and duplicate it twice. In the copies, change the value of the **`answer choice`** variable at the top to 2 for the first one and 3 for the second one. Happy quizzing!

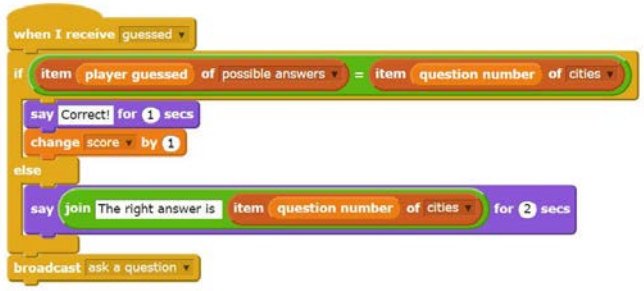
LISTING 1

A

B



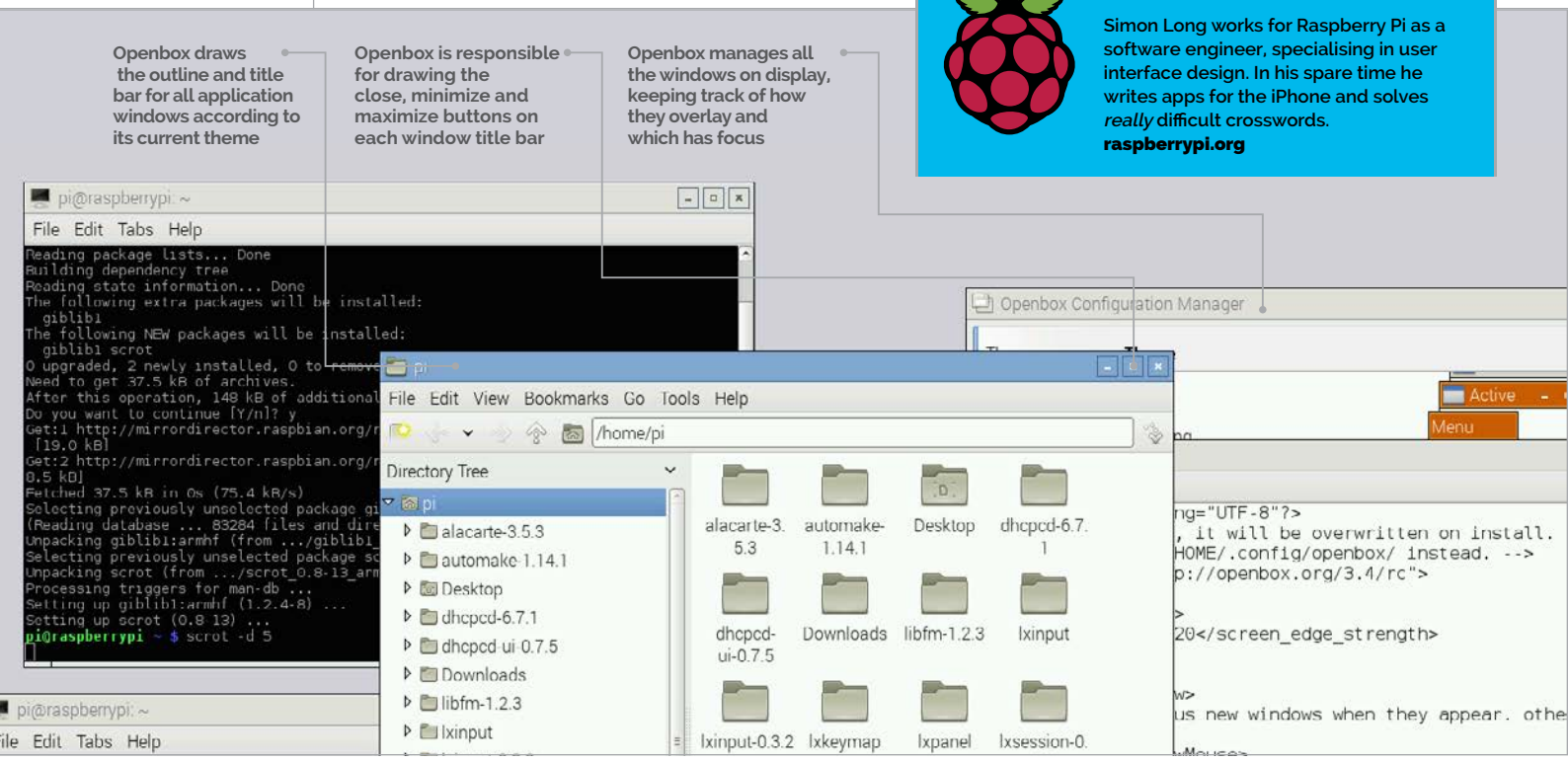
C



LISTING 2

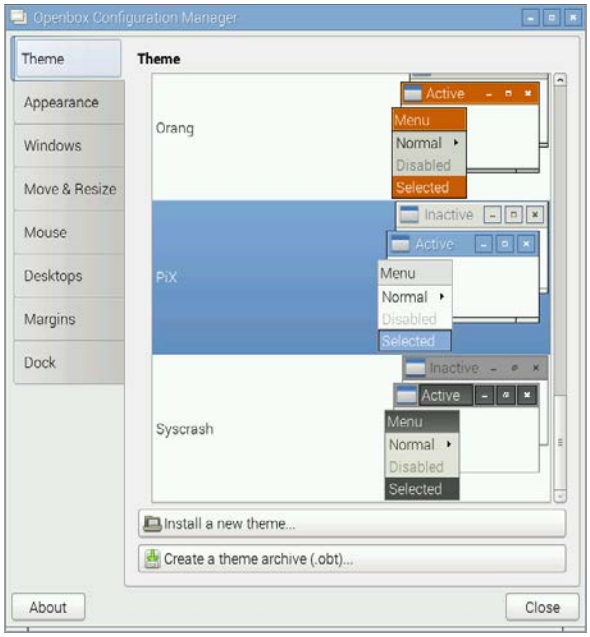
SIMON LONG

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



HACKING RASPBIAN'S DESKTOP PART 3: CUSTOMISING OPENBOX

In the third part of his series, **Simon Long** talks us through how to customise Raspbian's window manager, Openbox...



Right The Openbox Configuration Manager allows various aspects of the way a window is displayed to be modified

Openbox is a component of LXDE (Lightweight X11 Desktop Environment), the desktop user interface included as part of Raspbian. Openbox is the window manager – the software which is responsible for drawing the windows in which applications display their contents.

When an application opens, it requests a window from the window manager of the size required for what the application needs to display. The window manager creates the window, and draws the title bar and border. The window is then passed to the application, which draws its contents.

This mechanism may seem complicated, but it means that all windows created in the system will have the same overall visual appearance. The alternative would be for each application to create its own title bar for each window it requires, and the result would be a less consistent appearance, particularly when multiple applications run simultaneously.

The window manager is also responsible for managing the ability to move windows around the screen, for controlling what happens when a window is put on top of or underneath another, and for managing which window receives keyboard and mouse input.

What are themes?

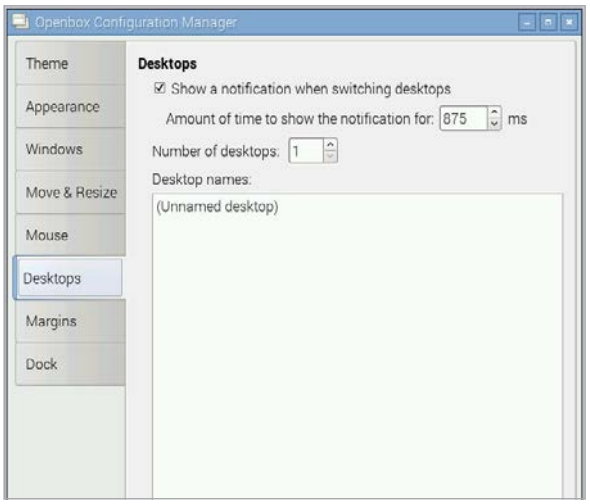
Before looking at how to customise Openbox, it is necessary to understand the idea of theming. LXDE allows the use of theme files for some aspects of operation. An Openbox theme file contains information on all aspects of the visual appearance of a window, such as the title bar colour, what the title bar buttons look like, and how the appearance changes when the window is active or inactive.

Several themes are available in Raspbian – they are stored in **/usr/share/themes**. Each theme has a directory in here, but not all themes include Openbox settings. If a theme does include Openbox settings, its directory includes a subdirectory called **openbox-3**, and in this subdirectory is the Openbox theme file itself, called **themerc**.

It is possible to have themes which are only available to one particular user; these are the same files, but stored in the **.themes** directory in the user's home directory. If a theme file in this directory has the same name as one in the global **/usr/share/themes** directory, the one in **.themes** has priority.

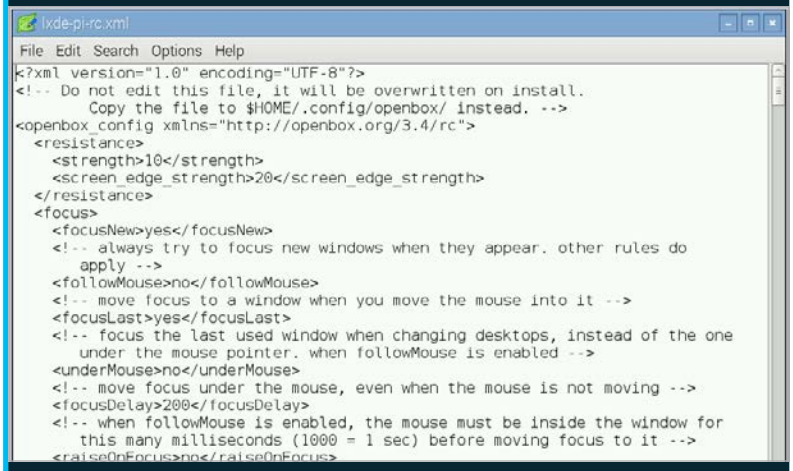
You can install new themes if you don't like any of those installed; they are available for free from numerous sites. You can also create your own, but doing so is outside the scope of this article – by all means open a **themerc** file with your favourite editor and have a play!

(Note that the default theme is called PiX – this is used to allow the Pi-specific Appearance Settings application to work properly. If you change to a different theme, running Appearance Settings will revert to the PiX theme – be aware of this if you want to customise your desktop in detail.)



Above The Desktops tab of the Configuration Manager allows multiple desktops to be managed

THE OPENBOX CONFIGURATION FILE



The settings file that controls the appearance of Openbox, including setting the theme used, is found in the **.config/openbox** subdirectory of a user's home directory, and has the extension .xml. If there are multiple files in here, the file used has the same filename as the current session profile name. On a default Raspbian install, this is LXDE-pi, and so the file used by Openbox is called **lxde-pi-rc.xml**.

Making changes

The Openbox XML file is quite large and complicated – you can view and modify it with a text or XML editor, but it is easier to use the Openbox Configuration Manager application. To launch this, type **obconf** in a terminal window.

The configuration manager has a number of tabbed pages. The first page, Theme, shows all Openbox themes installed on the Pi, and you can select from them by clicking in the list – each theme name is shown with examples of the window features it affects, and how they appear when using that particular theme.

The second page, Appearance, sets the fonts used for the window title bar. It also determines which buttons are shown on the title bar, and their order.

These first two pages contain most of the options that affect the appearance of the desktop; the remaining pages are more about adjusting the detailed behaviour of the window manager, and are really for expert users only – by all means play with the settings, but the effect of most of them is fairly subtle.

One tab worthy of particular mention is Desktops. Openbox supports multiple desktops, which can be useful if you want to have many applications open at once; an application's windows will appear on the desktop from which it was opened. If you want to use more than one desktop, simply increase the 'Number of desktops' value on this screen. Clicking the middle button or scroll wheel on the mouse when the pointer is on the desktop brings up a menu which allows you to switch desktops. It is also possible to switch desktops by adding a desktop switcher plug-in to LXPanel – see the previous article for details of how to do this.

NEXT MONTH

The final part of this series looks at using themes to customise the appearance of applications.

EMILE COLE

Fresh out of a Creative Writing degree in Plymouth, Emile is a junior technical author at RealVNC.
realvnc.com/products/vnc/raspberrypi

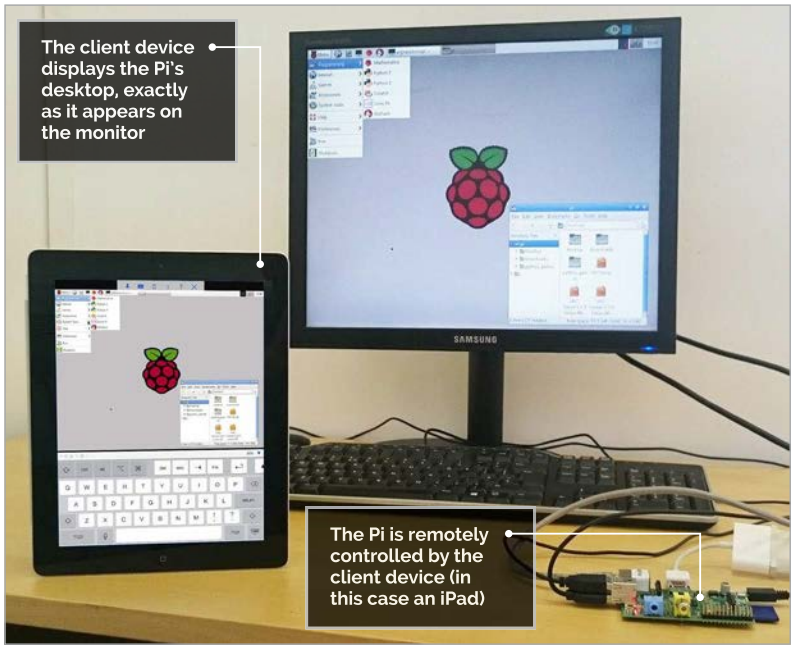
CONTROL YOUR PI REMOTELY WITH VNC SOFTWARE

You'll Need

- VNC Server software
- VNC Viewer software
- A VNC licence (free or paid)
- A client device (such as a mobile phone)

Learn how to control your Raspberry Pi from afar, whether you're at the office or stuck using 4G on the train home…

As pocket-friendly as the Pi is, you can't take it everywhere. At some point, you'll find yourself with nothing but a smartphone or computer for company, and it just won't be the same. Fortunately, thanks to RealVNC, there's a way to remotely access your Pi's desktop – even if you're running it headless. This tutorial will talk you through the basics of setting up VNC Server on your Pi and connecting to it from a client device using VNC Viewer. We'll also look briefly at Virtual Mode, and how you can use it to gain visual access to a headless Raspberry Pi.



>STEP-01

Installing VNC Server

Start by opening LXTerminal and running:

```
curl -L -o VNC.tar.gz http://bit.ly/1ILmo8p
tar xvf VNC.tar.gz
```

Once everything's downloaded, navigate to its location and run:

```
sudo dpkg -i <VNC-Server-package-name>.deb <VNC-Viewer-package-name>.deb
```

Note: You'll need to replace the angle brackets and their contents with your own package names, e.g. `VNC-Server-5.2.3-Linux-ARM.deb`; this will also install VNC Viewer, meaning your Pi can take control of other computers if you wish.

Next, download VNC Viewer to your client device, such as a mobile phone. This can be done for free through either **realvnc.com/download** or the iOS or Android app store.

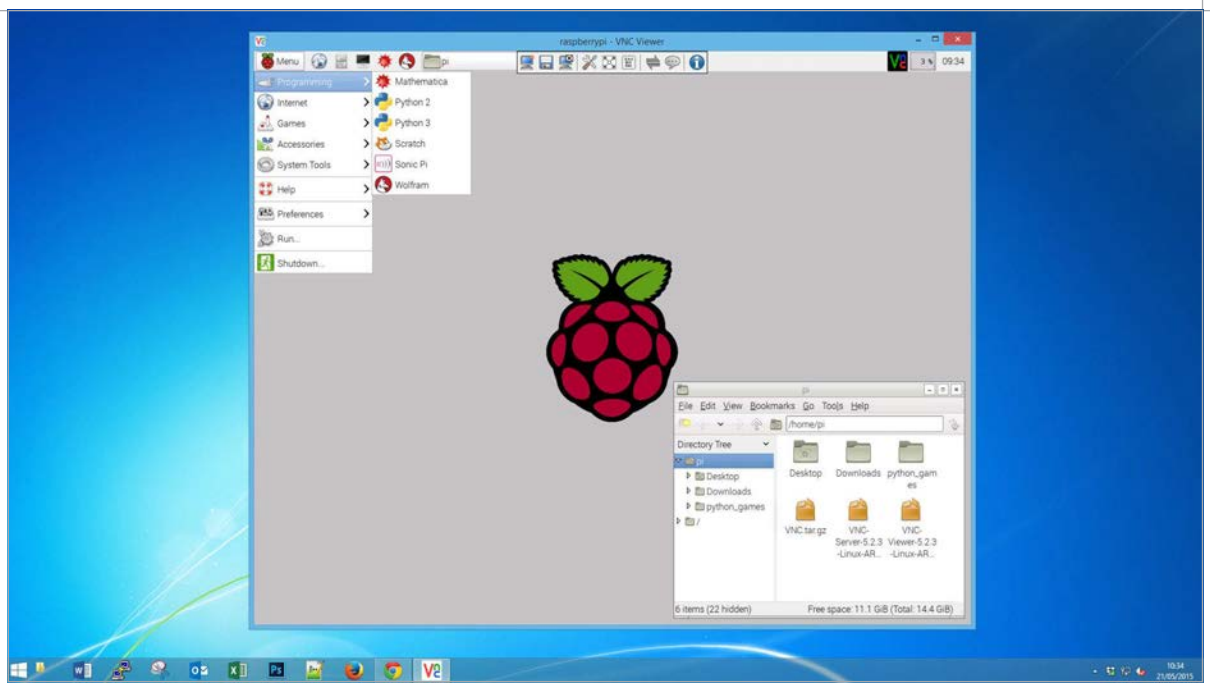
>STEP-02

Licensing VNC Server

VNC Server must be licensed with a Free, Personal or Enterprise licence. To obtain one, visit **realvnc.com/products/vnc** and choose the right option for you; while a Free licence will provide you with basic remote control, you'll miss out on features such as encryption, system authentication, file transfer, chat, and dedicated product support. For the purposes of this tutorial, however, a Free licence will be fine. Once you have a key, apply it at the command line with:

```
sudo vnclicense -add <license-key>
```

Note that VNC Viewer will not need to be licensed.



Left Controlling your Pi from a Windows desktop

>STEP-03

Controlling your Pi locally

VNC Server is usually run in Service Mode or Virtual Mode; if your Pi is headless, only the latter will work, so skip to step 6. Create a new password for Service Mode with **`sudo vncpasswd /root/.vnc/config.d/vncserver-x11`**, then type in a password. Service Mode gives full access to your Pi's desktop. Run **`sudo /etc/init.d/vncserver-x11-serviced start`**. Now open VNC Viewer on your client device. Type your Pi's private IP address (see VNC Server's dialog box, or run **`ifconfig`**) and connect. You'll now have visual access to your Pi. For detailed setup instructions, refer to **realvnc.com/products/vnc/raspberrypi**.

>STEP-04

Controlling your Pi over the internet

If your Raspbery Pi's at home and you're not, you'll obviously have to connect to it over the internet. Here, you'll need to configure port forwarding. In a modern router's settings, you should be able to forward the 'VNC' service to port 5900, then select your Pi from the device list.

VNC Viewer can now connect to your Pi from anywhere; just input its public IP address (visit **whatismyip.com** from your Pi). This may only work until your ISP reboots your router if they do not provide you with a static public IP. To ensure you'll always have the right connection information, apply for a hostname using a service such as **noip.com** or **dyndns.com**.

>STEP-05

Navigating your Pi on a smartphone

When connecting from a smartphone for the first time, controlling your cursor may feel strange. Instead of navigating your Pi like you would your phone's web browser, you should try to see your phone as a laptop touchpad controlling a cursor. This provides accurate control of your Pi's high-resolution desktop, despite the small screen of your device.

Drag your finger across your screen to move the cursor. The visible desktop automatically scrolls with you. To left-click, tap anywhere once; double-tap to double-click. More gestures are available by clicking the '?' in the app toolbar.

> "Instead of navigating your Pi like you would your phone's web browser, you should try to see your phone as a laptop touchpad controlling a cursor"

>STEP-06

Creating a virtual desktop

VNC Server can be run in Virtual Mode to create and remotely control as many virtual desktops as your licence allows. In Virtual Mode, the user does not see what they would if sitting in front of the Pi. Instead, they see a virtual desktop that is visible to only the VNC Viewer user. This can provide visual access to a headless Raspberry Pi computer.

Type **`vncserver`** at a command prompt, then create a password. Note down VNC Server's IP address (including the colon and subsequent display number). Next, you need to input this information into VNC Viewer; you should now be connected to your very own private virtual desktop. To learn more about VNC, visit **realvnc.com**.

STARTING VNC SERVER IN SERVICE MODE

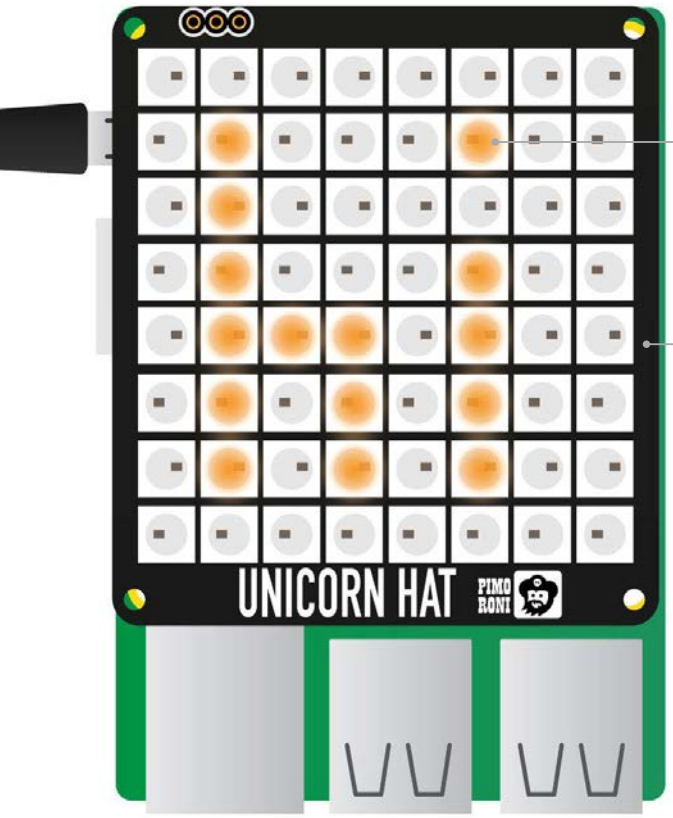
To automatically start VNC Sever in Service Mode when your Pi is turned on, run: **`sudo update-rc.d vncserver-x11-serviced defaults`**

STOPPING VNC SERVER

To stop VNC Server in Service Mode, run: **`sudo /etc/init.d/vncserver-x11-serviced stop`**. To stop it in Virtual Mode, run: **`vncserver -kill :<display-number>`**

SEAN MCMANUS

Sean McManus is the author of *Raspberry Pi For Dummies* (with Mike Cook), and *Scratch Programming in Easy Steps*.
sean.co.uk
twitter.com/musicandwords



The code displays multicoloured red, purple and blue text

A diffuser layer makes it easier to read, and protects your eyes from the bright LEDs!

MAKE TEXT SCROLL ON THE UNICORN HAT

You'll Need

- Unicorn Hat
- Frosted or Ninja diffuser
- Pibow case (optional)
- Internet access

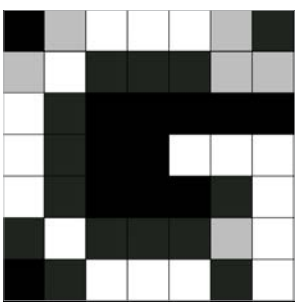
The Unicorn HAT provides a compact and colourful way to display scrolling messages on your Raspberry Pi. Here's how it's done...

It's easy to get colourful special effects from the Unicorn HAT, an 8×8 matrix of RGB LEDs. But did you know you can also use it to output text? This project shows you how to scroll messages across it, and could form the basis of any project that needs to display information, such as a robot or Twitter display. Nobody wants to hand-design (or even hand-code) a whole new font for this, so this project uses Pygame to scan a font on screen instead, capturing that font data in a Python dictionary.

>STEP-01

Create your font

The key to this project is to use one of the fonts already on your Pi, **FreeSans.ttf**, and convert it to a useful form for the Unicorn HAT scroller program. Open Python 2 and enter **Listing 1**. It first displays each character in turn and scans it to check the brightness of each pixel. Next, it builds a list of values for each pixel in the letter, and compiles a dictionary of those list values. Finally, it trims the character data back to the minimum width required.



Right The font creator program uses three different colour shades. On the Unicorn HAT, these shades are multiplied by the colour numbers to create the letters

>STEP-02

Manually modify selected characters

This step is optional, depending on your application. Some of the punctuation symbols (including @) don't render clearly at the size we need to display them for scanning. If you want, you can define them manually. Each character has a list that contains another list of data for each row. Use 1 to plot a point and 0 for an empty point, as shown for @ in Listing 1. Here's a shortcut to save hand-designing the characters first: the Amstrad CPC6128 manual (available at **bit.ly/1AgpY7J**) shows the design for that classic computer's font, which also uses an 8×8 grid (see Chapter 7).

>STEP-03

Enter the scroller code

Listing 2 is your scroller code. Enter it into a new window in Python 2 and save it as `scroller.py`. The code assumes you have your Raspberry Pi with the USB ports on the left, so you can stand it up on the side that has no cables going into it. Change the orientation at the start if your Pi is a different way around, to 0, 90 or 270.

>STEP-04

Paste in your font dictionary

Run the font creator program (`python fontmaker.py`), highlight the font dictionary when it's shown

Language
>PYTHON

fontmaker.py (Listing 1)

```
# -*- coding: utf-8 -*-
import pygame
pygame.init()
canvas=pygame.display.set_mode((100,100))
pygame.mouse.set_visible(0)
char_set = "QWERTYUIOP ASDFGHJKL ZXCVBNM \
1234567890-= !$%^&*()_+"
char_set += "[]{} ;'#:@~ ,./<>?\"\\"
font_dictionary = dict()

for letter in char_set: #main dictionary creation loop
    canvas.fill((0,0,0)) #clear the canvas
    fontObj = pygame.font.Font(\
'/usr/share/fonts/truetype/freefont/FreeSans.ttf',9)
    textSurface = fontObj.render(
    letter,True,(255,255,255),(0,0,0))
    textRectObj = textSurface.get_rect()
    canvas.blit(textSurface, textRectObj)
    pygame.display.update() #display the letter

    letter_data = []
    for y in range(8): #check each row of
    # the letter on canvas
        letter_row=[]
        for x in range(8):
            #each x position of letter on canvas
            colour = canvas.get_at((x,y+2))
            if colour[0]>200:
                letter_row.append(1)
            elif colour[0]>100:
                letter_row.append(0.75)
            elif colour[0]:
                letter_row.append(0.15)
            else:
                letter_row.append(0)
        letter_data.append(letter_row)

    for x in range(7,-1,-1): # Trim excess space on right of letter
        column=[letter_data[y][x] for y in range(8)]
        if max(column)==0:
            for i in range(8):
                del letter_data[i][x]

    font_dictionary[letter]=letter_data
font_dictionary[' ']=[[0]*4]*8 #space gets trimmed to empty otherwise
font_dictionary['@']=[[0,1,1,1,1,1,0],[1,1,0,0,0,1,1],[1,1,0,1,1,1,1],\
[1,1,0,1,0,0,1],[1,1,0,1,1,1,1],[1,1,0,0,0,0,0],[0,1,1,1,1,1,0],[0]*7]

pygame.quit()
print font_dictionary
```

in the shell, and use SHIFT+**CTRL+C** to copy it all. It starts and ends with a curly bracket and might span more than one screen. Paste it in place of the curly brackets, where **font_dictionary** is defined near the top of Listing 2. Now your scroller program has the font data it needs. Save your program.

>STEP-05

Restart and install software

There appears to be a conflict between Pygame and the Unicorn HAT, so you can't use them both in the same session. If you see random flashing on the Unicorn HAT when you run the scroller program, this is probably the reason why. Restart your Raspberry Pi now. If you haven't already installed the Unicorn HAT drivers, go into the command line and issue the command:

\curl -sS get.pimoroni.com/unicornhat | bash

>STEP-06

Run your scroller

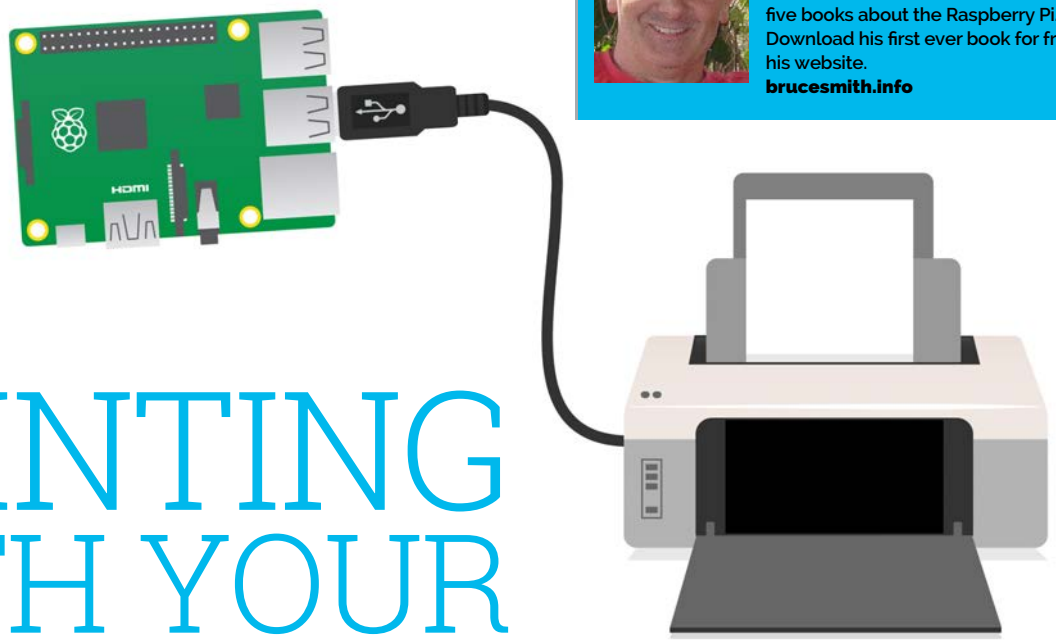
You should view your Unicorn HAT through a diffuser layer. You can buy one from Pimoroni, designed for use as a lid on the full-size Pibow case. To run your scroller, open the command-line and go to the folder containing your code in. Enter **sudo python scroller.py**. The program will ask you for text to scroll and then scroll it across the display. When you build this code into other applications, put the message you want to display into the variable **string_to_show**.

scroller.py (Listing 2)

```
import unicornhat as unicorn
import time
unicorn.rotation(180) #adjust for your Pi's orientation
unicorn.brightness(0.4)
#warning: Altering this value can make LED VERY bright!
font_dictionary={} # paste in your font dictionary here
string_to_show=raw_input("Enter the text to scroll: ")
scroll_rows=[[0]*8]*8 #blank space at start of message

for character in string_to_show:
   if character.upper() in font_dictionary:
       character_rows = font_dictionary[character.upper()]
   else:
       character_rows = font_dictionary['-']
   for i in range(8):
       scroll_rows[i] = scroll_rows[i]+character_rows[i]
       scroll_rows[i] += [0] #gap between letters

for i in range(8):
   scroll_rows[i]+=[0]*8 #blank space at end of message

for scroll_position in range(len(scroll_rows[0])-8):
   for y in range(8):
      thisrow = scroll_rows[y]
      for x in range(8):
          pixel_shade=thisrow[x+scroll_position]
          unicorn.set_pixel(x,y,int((95+x*20)*pixel_shade),\
              int(100*pixel_shade),int((95+y*20)*pixel_shade))
   unicorn.show()
   time.sleep(0.04)
```

BRUCE SMITH

Bruce Smith is an award-winning author of over 100 books. He has written five books about the Raspberry Pi. Download his first ever book for free from his website.
brucesmith.info



PRINTING WITH YOUR RASPBERRY PI

In an excerpt from **Bruce Smith's** *Raspberry Pi Insider Guide*, we learn how to install and configure a printer on the Pi...

If you intend to use your Raspberry Pi in a home office or to learn programming, you're sure to want to print out your results. In this excerpt from the book *Raspberry Pi Insider Guide* (**brucesmith.info/?p=14**), we look at the software to use and how to configure it.

One of the few areas where the Raspberry Pi has been a little disappointing is in the use of printers and the associated printer support. Most operating systems attack the issue of printers head-on, but Raspbian is not one of these. Installing and configuring printers is a bit of a black art, as it has been for many years on Linux, not least because there are so many makes and models of printer. Often, the real trick is picking the correct printer from the lists that are presented by the system. It is important that you know the exact make and model of your printer, as the software that translates the information from the page you are printing to ensure the correct output onto paper is dependent on this. This software is called the printer driver.

Fig 1 The CUPS configuration screen, as seen through the web interface

Printing with CUPS

Before starting out with your printer installation, make sure your package lists are fully up-to-date by using:

```
sudo apt-get update
```

Then ensure that your printer is switched on and available. There are three possible connection types you could have: through the USB or on the local network, either wireless or cabled. The setup process is the same in each case. In the worked example that follows, we've used a Samsung ML-2580N printer connected via a router on a home network.

You will need to download and install the CUPS (Common Unix Printing System) software. The command to do this is:

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

The process can take a while. CUPS uses the group **lpadmin** to determine who is authorised to administer printers. You will need to add the **lpadmin** group to your user profile to enable you to administer the printers. This can be done by issuing:

```
sudo usermod -a -G lpadmin pi
```

This assumes that your username is still the default **pi**; replace **pi** with your own username if you have changed it.

The rest of the setup can be done through a web browser. In your desktop environment, open your preferred web browser. In the URL bar, enter **http://127.0.0.1:631** and, after a moment or two, the 'CUPS' screen should appear, looking similar to that shown in **Fig 1**.

As this is also the screen that you will need to navigate to if you want to change settings or add new printers in the future, it makes good sense to bookmark it at this point.

There are a number of tabs running across the top. Click on Administration (**Fig 2**) and then select Add Printer. At this stage, you will be prompted for your username and password – enter your normal Raspbian login name and password.

CUPS will then search for printers that are locally connected, along with any it can see on the network. This may take a few minutes. Select your printer from the results, then click Continue (if you have a printer attached via USB, then it should be listed under the 'Local Printers' option).

The next window allows you to edit the name and location of the printer. If you have only one printer, the fields can usually be left at their defaults, but it's nice to personalise things.

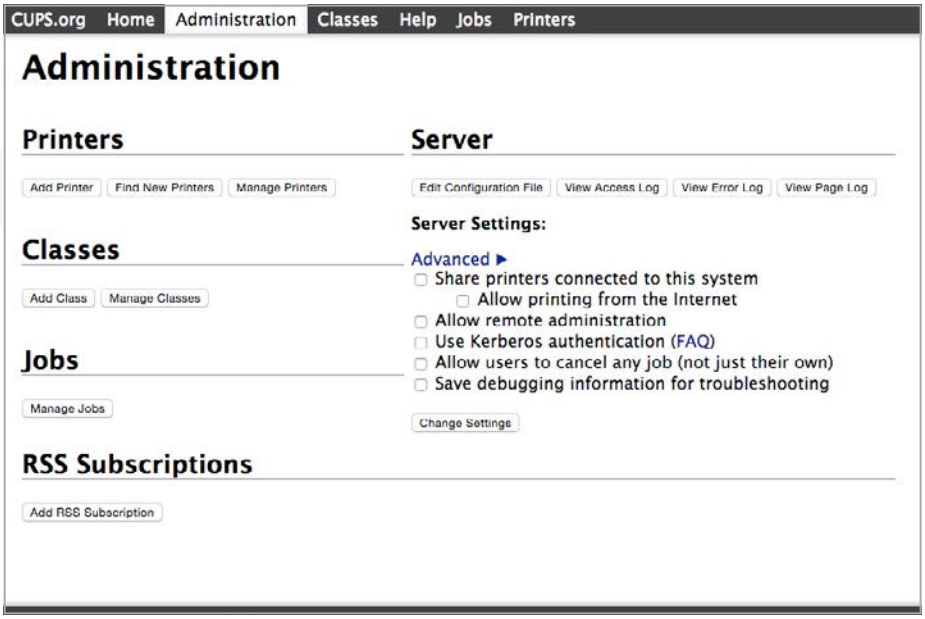


Fig 2 The Administration screen. You'll need to enter your username and password to make any changes

Printer sharing

One field that you may consider if you have a local printer is the 'Sharing' option. This is not normally required for a network printer, where you can connect direct from the computer to a printer, but here it would allow you to share a USB printer across the network if required.

Once you select Continue, CUPS will search through its database and come up with a list of potential drivers for your printer. You should scroll through these options until you find the one for your printer – i.e. the one that exactly matches the name and model number of your printer.

If there is not an exact match, then there are a couple of options. Firstly, if there is a make and model number that is very similar, try this; or if there is a generic driver for your make of printer, this would probably work as well – even if it doesn't support all the printer features. Alternatively, try searching the internet with something like 'Linux printer driver XXX', where 'XXX' is the make and model of your printer.

Once you have selected your printer, click Add Printer. The next page allows you to set the default options for your printer. It isn't really necessary to do this at this exact point, as you'll tend to do this from the application when printing.

Now that you've followed these steps, in any program that you use in future, you should be able to go to the **File>Print** option and select the page or pages you want to produce!

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

SAFARI PARK CHALLENGE

Learn how to detect toy cars on a board and create the Safari Park challenge – it makes for a great individual project or a whole class one, ideal for open days

You'll Need

- 12 all-metal paper clips
- 12 small cable ties
- Stranded wire
- A2 thick cardboard
- Wiring Pi
- Colour printer A4 or bigger

One of the great things about physical computing is the ability to sense objects and get the computer to react to them. You can sense objects in a number of different ways, but perhaps the simplest is getting them to trigger a switch. Here we show you one way of detecting a toy car and how to build a Safari Park Challenge with this method.

The game

In our game, the roads through the safari park's animal enclosures connect four pedestrian areas. Here you can get out of your car or drive through the next enclosure. The challenge is to tour the whole park by driving through each enclosure only once. In each pedestrian area, you have a choice of several roads, but you're not allowed to use any of them twice. On each road there is a sensor switch to detect your car; when triggered, the computer marks off that you've visited that area, makes a sound, and shows a picture of the animal in that area. You can start from any one of the four pedestrian areas.

Detecting the car

We need to detect the cars as they drive down the road. This can be done in a number of ways, but here we'll show you perhaps the simplest method, where the object itself forms part of the switch. For this, it has to be conductive. Most toy cars are made of plastic; however, with the use of Bare Conductive paint, almost anything can be made conductive. We took a small toy car and made the front wheels conductive by painting them. We found that the paint from the tube was a bit thick to apply a smooth coat, so we added a


Fig 1 Using paint to make the tracks

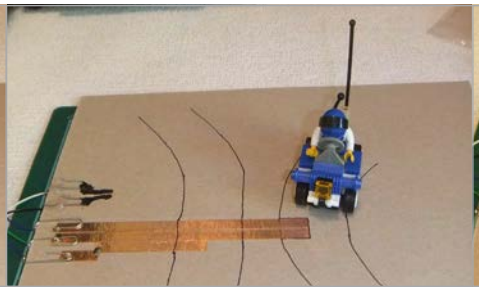


Fig 2 Using copper foil to make the tracks

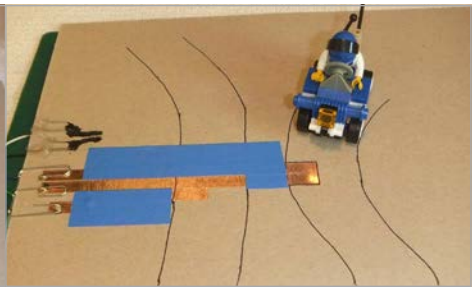


Fig 3 Using insulation tape to mask out the contacts

tiny bit of water and mixed it thoroughly until we had a thick but brushable consistency. Then we gave the front wheels two coats, allowing 24 hours drying time between them. An ohmmeter showed that we had obtained a resistance of about 250 ohms across the wheel. Do not do this for the rear wheels.

The track

The wheels are only half the switch; the other is the track that they'll bridge. We looked at two ways to make this. **Fig 1** shows a short track made by conductive paint, attached to two paper clips. Wires were attached to the clips underneath the board using ready-crimped header sockets, although you could solder them – if so, do this first before painting the tracks. Finally, a small blob of hot melt glue secured the paper clips. This worked well, although there was a bit of a speed hump for the car to manoeuvre over. We didn't try smoother ink, as we felt it would not be conductive enough over long lengths.

The alternative way to make the tracks is with strips of self-adhesive copper foil, available from hobby shops and electronics suppliers. **Fig 2** shows three copper strips making up two sensors. The centre strip is connected to ground, and the two outer strips to two GPIO pins. **Fig 3** shows how, by using insulation tape, you can mask off areas to give two independent contacts. Tests showed that this worked, but better results were obtained when the wheel cut across the strips at an angle, as it straddled each side of the track instead of contacting at a single point.

Both types of track work in the same way. The GPIO pin is made into an input with the pull-up resistor enabled, thus normally reading a logic **1**. When the wheel shorts this track to the ground track, the GPIO pin will read a logic **0**.

The code

The game code depends on two directories: **sounds**, containing sounds in the WAV format, and **images**, containing JPEG pictures. Placed in the same directory as the Python code, these are used when the switches are triggered. They are tied together by the **areaName** list; if you change the names of the pictures and sounds, you must also change this list. The program runs under the Pygame system with a window size of 600×400 pixels – you should make sure your animal pictures are no bigger than this, although they can be smaller. The **pinList** list ties the GPIO numbers to the **areaName** list; this is shuffled at the start of each turn to put the animals in different enclosures, but you can comment this line out (with a **#** at the start) if you like.

The **visited** list keeps track of where you've been. If you haven't visited an area, then the list elements are all set to be **-1**; if you have, they're set to **0**. The main loop of the program calls the **checkForEvent** function, followed by the **scanSwitches** one. It is this latter function that looks at the track and sees if the car is over any switch; if it is, then it triggers the appropriate sound and shows the matching picture. Then it checks if that area has been visited, prints out a message if it has been, then that area is ticked off the list. Finally, the **finished** function is called to check if there are still places to visit and returns a **True** if there are not. This triggers the end sequence, which consists of resetting the visited list, shuffling the pin list to move the animal areas, and displaying the final picture.

> "We need to detect the cars as they drive down the road"

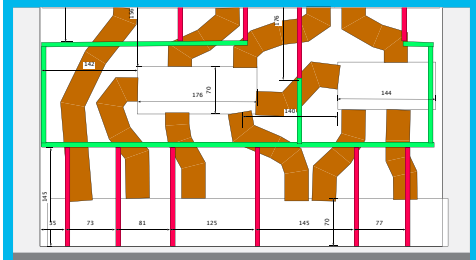
The game's background

This game is based on the famous logic problem, the Seven Bridges of Königsberg (**bit.ly/1z0Ofxa**), which Leonhard Euler proved was impossible in 1735, thus inventing a whole new branch of mathematics we now call topology. Basically, the problem cannot be solved if there is an odd number of nodes; in our game terms, this translates into roads from a pedestrian area, unless that place is the start or end point of the tour. Our variation is possible to complete from any starting location because there is an even number of nodes (roads) from each pedestrian area.

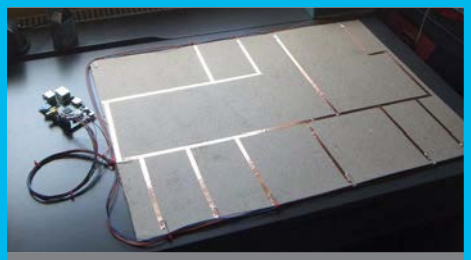
You have a go

You could comment out the shuffle routine and have the animals in a fixed place every time, allowing you to place their pictures on the board. You could also record your own introduction and ending sound sample to go with the relevant pictures. You might want to keep track of where the car has been and prevent it being picked up to place in another area. To do so, you need a list of permissible roads from each area you are in.

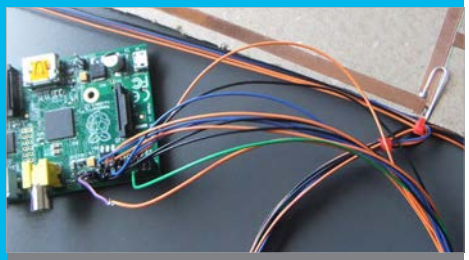
BUILDING THE GAME



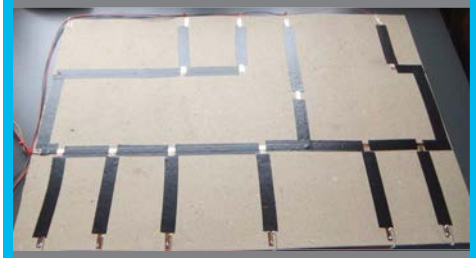
>STEP-01
Planning the board
This is planned to be built on an A2 piece of cardboard. We used the backing sheet of an A2 drawing pad from a well-known chain of remaindered bookshops. The copper foil strips are shown in green and red, with the green ones being connected to the ground and the red to individual GPIO pins. The dimensions are shown in mm and are approximate. We made the roads 35mm wide, as our car's width was 30mm. You can change this to suit the width of the car you use. Either way, the exposed foil sensors should be placed at one side of the road so the left or right wheel makes contact with it, depending on the direction the car is going down the road.



>STEP-02
Lay out the tracks
Lay out the foil strips in the position of the plan. Where the ground strips (green) make a right-angled turn, lay two strips at right angles on top of each other. Then put a blob of solder on the join to electrically connect them together. Try to use as little solder as possible and avoid getting a hump over the join. Connect a paper clip to each of the foil strips and solder it to the foil. You will need to heat up just the metal of the paper clip to get it hot enough to melt the solder, which will then flow down and stick to the copper. Do not try to apply the iron tip to the copper, as not enough heat will be transferred into the paper clip. Then secure the paper clip to the cardboard with hot melt glue.



>STEP-03
Attach the wires
Solder wires from the paper clip to the GPIO pins of the Raspberry Pi. Connect the green strip from the plan to a GPIO ground, and the red strips from the plan to GPIO pins 2, 3, 4, 15, 17, 18, 22, 23, 24, and 27. It does not matter what GPIO pin is connected to what strip. Alternatively, you can cold-solder all connections by putting some Bare Conductive paint over the paper clip, foil and wire joints. Then the wires should be attached to the paper clips by means of small cable ties to make a tidy job. Note that there is a paper clip in the top-left corner of the board for added tidiness. Without some sort of cable management, it will look a mess and detract greatly from the game's overall appearance.



>STEP-04
Cover the foil
Make sure the foil is firmly pressed down; use the back of your thumbnail or a plastic fold-and-score tool to remove wrinkles and get the self-adhesive glue to stick firmly. Then cover the foil strips with plastic insulation tape, all except a 5×10mm square around where the two foil contacts come close to each other. This makes the foil more secure by adding to the adhesiveness – the glue on the back of the foil is not the strongest and if the cardboard is rough, it will not stick very well. This also adds protection for the areas of copper you don't want to contact, and will protect the foil if you want to remove the 'grass' we add in the next step.

>STEP-05
Cover the board
Print out about four A4 sheets of a plain pale green colour. Use this to cover all the board, but leave holes for the foil contacts. You don't have to be too accurate here. Also, a blocked ink-jet nozzle on our printer added some white stripes to the green colour. Now, create some drawings for the top and bottom pedestrian areas, print them out, and stick them to the top and bottom of the board. We used solid white stick glue for this, to minimise any wrinkling in the paper, but be careful not to get any on the printed side of the paper as it will make the ink smudge. Alternatively, the PDF files we used are available on GitHub. Note that the top and bottom areas are split up in these files, to allow you to print them using an A4 printer.

>STEP-06
Add roads and pedestrian areas
Print out the roads and central pedestrian areas and cut them out. Lay the road down, making sure that the contact foil is just covered by the side of the road. Mark where the foil contacts are and then cut out a hole in the road to let the wheel through. Align the road carefully with the hole and hold onto one end of the road so it will not move, then peel the other end back. Apply glue to the grass layer and smooth the road down, then lift up the other end and apply glue to the base again. Try not to get glue on the foil contacts. These files are available on GitHub or you can draw them yourself, making full use of clip art to enhance the graphics. Give the front wheel two coats of paint and allow it to dry for at least 24 hours between coats.

Language
>PYTHON

DOWNLOAD:
github.com/Grumpy-Mike/Mikes-Pi-Bakery

Safari.py

```
# Safari - Car game
# By Mike Cook - May 2015

import pygame, time, os, random
import wiringpi2 as io

pygame.init() # initialise graphics interface
pygame.mixer.quit()
pygame.mixer.init(frequency=22050, size=-16, channels=2,
buffer=512)
enclosureSound= [ pygame.mixer.Sound(
"sounds/"+str(c)+".ogg") for c in range(0,10)]

os.environ['SDL_VIDEO_WINDOW_POS'] = 'center'
pygame.display.set_caption("Wildlife Safari Park")
pygame.event.set_allowed(None)
pygame.event.set_allowed([pygame.KEYDOWN,pygame.QUIT])
screen = pygame.display.set_mode([600,400],0,32)

try :
   io.wiringPiSetupGpio()
except :
   print"start IDLE with 'gksudo idle' from command line"
   os._exit(1)

pinList= [2,3,4,15,17,18,27,22,23,24] # pins for switches
random.seed()
random.shuffle(pinList)
visited =[-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]
duplicate = False
areaName =["Lion","Camel","Monkeys","Elephant",
"Zebra","Tiger", "Rhinoceros","Ostrich","Donkey",
"Kangaroo","intro","final"]
animalPicture = [ pygame.image.load(
"images/"+areaName[frame]+".jpg").convert_alpha()
                  for frame in range(0,12)]
enclosureSound = [ pygame.mixer.Sound("sounds/"+areaName[
frame]+".wav")
                  for frame in range(0,10)]
def main():
   global visited, duplicate
   print"Wildlife Safari Park"
   print"press return to restart a tour"
   showPicture(10) # introduction picture
   initGPIO()
   while True:
     checkForEvent()
     scanSwitches()
     if finished() :
        visited =[-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]
        random.shuffle(pinList)
        print"Tour complete"
        if duplicate :
           print"...but you visited an area more than once"
           duplicate = False
        time.sleep(6.0)
        showPicture(11) #final picture
        time.sleep(6.0)
        showPicture(10) # intro pic

def initGPIO():
   #for pin in range (0,pinList.len):
   for pin in range (0,10):
      io.pinMode(pinList[pin],0)
      io.pullUpDnControl(
pinList[pin],2) # input enable pull up

def scanSwitches():
   global visited, duplicate
   for pin in range (0,10):
      if io.digitalRead(pinList[pin]) == 0:
         print"You have now visited the",
areaName[pin],"area"
         enclosureSound[pin].play()
         showPicture(pin)
         if visited[pin] == 0:
            duplicate = True
            print"you have already been here"
         visited[pin] = 0
   return

def showPicture(animal):
   pygame.draw.rezct(screen, (0,0,0), (0,0,600,400),0)
   screen.blit(animalPicture[animal],[0,0])
   pygame.display.update()
   time.sleep(0.4)

def finished(): # have we visited all the animals?
   finished = True
   for place in range(0,10):
      if visited[place] == -1:
         finished = False
   return finished

def terminate():
    # close down the program
    print ("Closing down please wait")
    pygame.mixer.quit()
    pygame.quit() # close pygame
    os._exit(1)

def checkForEvent(): # see if we need to quit
    global visited, duplicate
    event = pygame.event.poll()
    if event.type == pygame.QUIT :
         terminate()
    if event.type == pygame.KEYDOWN :
       if event.key == pygame.K_ESCAPE :
          terminate()
       if event.key == pygame.K_RETURN :
          visited =[
-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]
          duplicate = False
          print"Start your tour again"
          showPicture(10) # intro pic

# Main program logic:
if __name__ == '__main__':
    main()
```

PART 4

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

YOUR FIRST GAME

In the fourth part of this series, we make an exciting platform game using the knowledge we've learned so far...

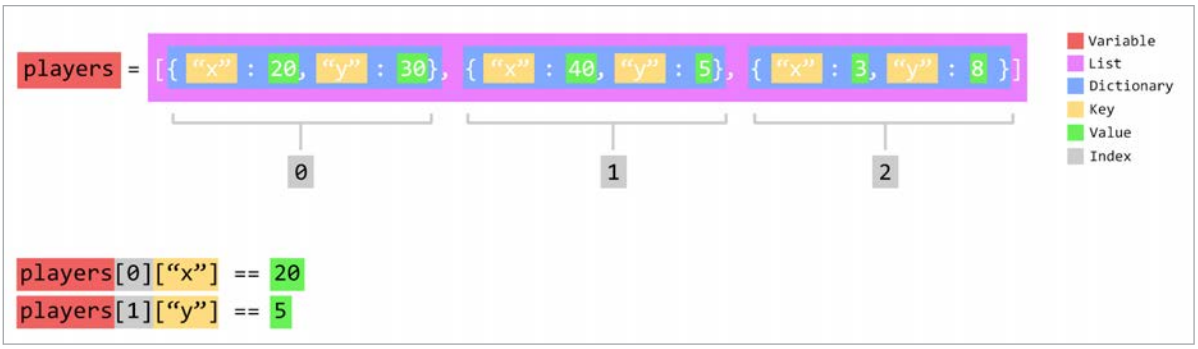
Now that we've covered making shapes, animating them, and control mechanisms, we have everything we need to make our first proper game. We're going to make an old-school drop-down game where platforms rise up from the floor and try to crush our player against the roof; the only way to survive is by dropping through the gaps in the platforms. Unlike our previous tutorials, we're not going to write a program that just runs – we're going to also make a simple start screen and a game over screen. With these new elements we still have a couple of new things we're going to learn about along the way, like loading images and timing events. This is by far the largest piece of code we will have written, but don't worry: if you've followed along so far, you'll recognise much of the code already!

How does our game work?

Before we write any code, though, it's important to have a solid understanding of how our game is going to work. When the game starts, our avatar (a red rectangle) will drop down from the top of the screen. Every two seconds, a white platform will start to rise from the bottom of the screen; if our character lands on one of these platforms, it will start to rise along with it. If we go off the top of the game screen, it's game over. Doom is not assured, however! We can move our character with the left and right arrow keys so we can drop down through the randomly positioned gaps in the platforms. The aim of the game is to stay alive as long as possible. It sounds easy, but things get tougher as time goes on, because the platforms will start to appear after a shorter delay.

Variables and prerequisites

Lines 1–39 of our code (see page 52) contain the `import` statements and variables we're going to need to get our game off the ground. By now, much of this should look pretty familiar. At the top we have our `import` statements, which let us include modules to help us with our game development. Take note of the `GAME_TIME` import, as this will be used quite a bit later on. Lines 8 and 9 are loading images that we'll be using for our start and game over screens. We could draw the graphical user interface (GUI) with code, but by using images we're saving ourselves time and effort at the cost of just a few kilobytes (1 megabyte = 1,024 kilobytes).



Above A handy breakdown of the various components of a dictionary

We've seen lines 11–15 before; these are the variables that we'll use to control how the game window looks. Lines 20–38 are the variables that we'll use to keep track of our game state, like where things are, how big they are, and how they should move. Two of these might jump out as a little bit different from variables we've used in the past: **`gamePlatforms`** and **`player`**; one looks empty and the other has multiple values. **`gamePlatforms`** is a variable type known as a list, while **`player`** is a type known as a dictionary, and they don't work like other variables do. Let's take a little time to understand how they work.

> In our past tutorials, we've almost always used variables that have one value, but there are other variables that can contain multiple values

Dictionaries and lists

In our past tutorials, we've almost always used variables that have one value, but there are other variables that can contain multiple values – like tuples, for example – and they are very useful to us as we start to make bigger and more powerful programs. When we write small programs, having variables with a single value is great, because we can see which variables are doing what. However, as programs grow, it can get harder to name variables in a way that relates to what we're trying to do. Let's imagine a game where there's more than one player, like an MMO; if we wrote code like we've done before, we'd need to create multiple sets of variables for each player. It doesn't take a genius to realise the code is going to get unmanageably long, very quickly.

What if we wanted to handle four or 100 or 1,000 players at the same time? Do we hand-write variables for every single one? No. We can use dictionaries and lists instead.

The **`player`** variable on lines 32–38 is a dictionary. A dictionary is a variable with multiple keys that have values. You can think of a dictionary like you would its real-world counterpart: if you want to know what something is, you search through until you find the definition. So, let's say we want to know what the value of **`x`** in the **`player`** dictionary is; all we have to do is request **`player["x"]`**. We can do the same with any other value that is stored in it, and we can also save or change values.

If the value **`player["y"]`** is 20 and we wanted to change it to 25, we'd enter **`player["y"] = 25`**, just like setting any other variable. Dictionaries are really useful, because they let us bunch together values in a way that's easy to read and easy to access with code. If we revisit our MMO game thought exercise, we'd quickly realise that we'd still need 100 variables to handle 100 players, even though we've made things tidier and more usable. What's the best way to keep track of dictionaries? That's where lists come in.

Lists are variables that can store groups of other variables. If we wanted to keep track of the players in our game, we wouldn't need to make a variable for each player – we could just add a player dictionary to a list and work through them whenever we need to. If, for example, we wanted to get the information for the second player in our imaginary MMO, we'd enter something like **`players[1]`** and that would return a dictionary for our second player which we could then get values from, like so: **`players[1]["x"]`**. The **`1`** in this example is called an index. It's important to notice that list indexes start counting from 0, so if we want to access the first item in a list, we use the index 0; if we want to get the fourth item from a list, we use the index 3.

In our game, we're not using lists and dictionaries to track players, but to track the platforms that we'll be moving along and dropping through. We'll have a look at that in a little while, once we've examined the game's logic.

Just_drop.py

```
01. import pygame, sys, random
02. import pygame.locals as GAME_GLOBALS
03. import pygame.event as GAME_EVENTS
04. import pygame.time as GAME_TIME
05.
06. pygame.init()
07.
08. title_image = pygame.image.load("assets/title.jpg")
09. game_over_image = pygame.image.load("assets/game_over.jpg")
10.
11. windowWidth = 400
12. windowHeight = 600
13.
14. surface = pygame.display.set_mode((windowWidth,
    windowHeight))
15. pygame.display.set_caption("Drop!")
16.
17. leftDown = False
18. rightDown = False
19.
20. gameStarted = False
21. gameEnded = False
22. gamePlatforms = []
23. platformSpeed = 3
24. platformDelay = 2000
25. lastPlatform = 0
26. platformsDroppedThrough = -1
27. dropping = False
28.
29. gameBeganAt = 0
30. timer = 0
31.
32. player = {
33.   "x" : windowWidth / 2,
34.   "y" : 0,
35.   "height" : 25,
36.   "width" : 10,
37.   "vy" : 5
38. }
39.
40. def drawPlayer():
41.
42.   pygame.draw.rect(surface, (255,0,0), (player["x"],
    player["y"], player["width"], player["height"]))
43.
44. def movePlayer():
45.
46.   global platformsDroppedThrough, dropping
47.
48.   leftOfPlayerOnPlatform = True
49.   rightOfPlayerOnPlatform = True
50.
51.   if surface.get_at((player["x"], player["y"] +
    player["height"])) == (0,0,0,255):
52.     leftOfPlayerOnPlatform = False
53.
54.   if surface.get_at((player["x"] + player["width"],
    player["y"] + player["height"])) == (0,0,0,255):
55.     rightOfPlayerOnPlatform = False
56.
57.   if leftOfPlayerOnPlatform is False and
    rightOfPlayerOnPlatform is False and (player["y"] +
    player["height"]) + player["vy"] < windowHeight:
58.     player["y"] += player["vy"]
59.
60.     if dropping is False:
61.       dropping = True
```

Language
>PYTHON

CODE & PICS:
github.com/sean-mtracey/Games-with-Pygame

Right We could code our title screen, but using an image is much simpler

The 'main' game loop

Lines 40–146 are where the logic for our game lives, but the state of our game is controlled in our main loop between lines 149 and 199. Just like our previous programs, on lines 153–176 we listen for various in-game events in our main loop and effect changes based on the events dispatched by our Raspberry Pi (keyboard, exit events, etc). Lines 178–196 are where the state of our game is determined and functions are called accordingly.

Each function between lines 40 and 146 is coded to handle a single aspect of the gameplay independently of the other functions, but they need to be called in a certain order to make sure that our game runs as expected. In order to understand each function and aspect of our game, we're going to work through them in the order that our main loop calls them. When our game first runs, we want to display our welcome screen telling people to press space to start; when the user presses the space bar, we want to start the game and when the user is pushed off the screen, we want to show the game over screen and let them restart. All of this is handled in the main loop. Let's break it down.

The start game screen

When our game starts, we're presented with the start game screen. This screen is a simple image that we loaded on line 8 of our code listing. At the other end of our listing, on line 189, we draw that image onto our surface. It's in the final **`if-elif`** statement of our main loop. On line 178, our script checks whether or not a game is already underway; if it is, it will check the game and render the screen as required. If there isn't a game underway, the loop then checks whether or not a game has ended on line 187, in which case we want to display the player's score on the game over screen and offer them the option to play again. If a game has neither been started nor finished, we can infer that we've just started the game and so we can render the start screen.

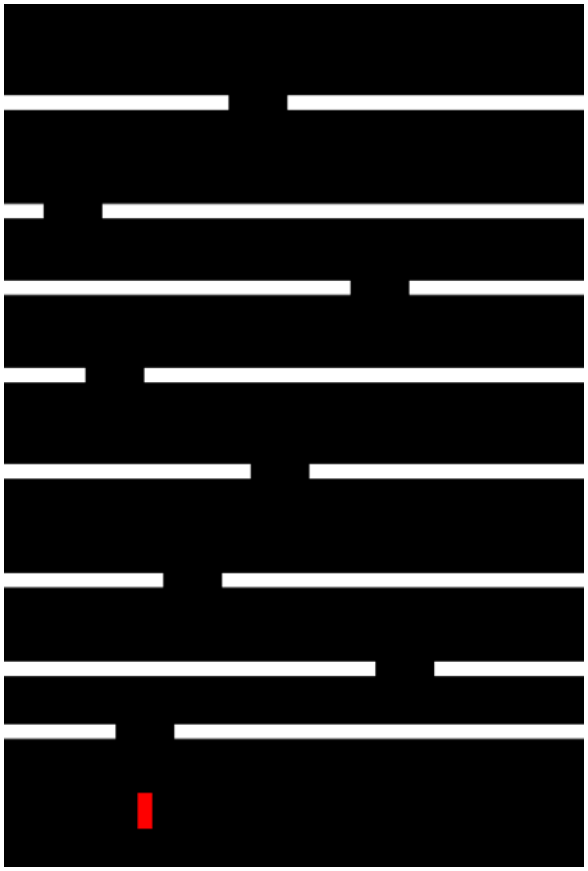
```
62.       platformsDroppedThrough += 1
63.
64.   else :
65.
66.     foundPlatformTop = False
67.     yOffset = 0
68.     dropping = False
69.
70.     while foundPlatformTop is False:
71.
72.       if surface.get_at((player["x"], (player["y"] +
    player["height"]) - yOffset )) == (0,0,0,255):
73.         player["y"] -= yOffset
74.         foundPlatformTop = True
75.       elif (player["y"] + player["height"]) - yOffset > 0:
76.         yOffset += 1
77.       else :
78.
79.         gameOver()
80.         break
81.
82.   if leftDown is True:
83.     if player["x"] > 0 and player["x"] - 5 > 0:
84.       player["x"] -= 5
85.     elif player["x"] > 0 and player["x"] - 5 < 0:
86.       player["x"] = 0
87.
88.   if rightDown is True:
89.     if player["x"] + player["width"] < windowWidth and
    (player["x"] + player["width"]) + 5 < windowWidth:
90.       player["x"] += 5
91.     elif player["x"] + player["width"] < windowWidth and
    (player["x"] + player["width"]) + 5 > windowWidth:
92.       player["x"] = windowWidth - player["width"]
93.
```

```
94.  def createPlatform():
95.
96.    global lastPlatform, platformDelay
97.
98.    platformY = windowHeight
99.    gapPosition = random.randint(0, windowWidth - 40)
100.
101.   gamePlatforms.append({"pos" : [0, platformY],
    "gap" : gapPosition})
102.   lastPlatform = GAME_TIME.get_ticks()
103.
104.   if platformDelay > 800:
105.     platformDelay -= 50
106.
107. def movePlatforms():
108.   # print("Platforms")
109.
110.   for idx, platform in enumerate(gamePlatforms):
111.
112.     platform["pos"][1] -= platformSpeed
113.
114.     if platform["pos"][1] < -10:
115.       gamePlatforms.pop(idx)
116.
117.
118. def drawPlatforms():
119.
120.   for platform in gamePlatforms:
121.
122.     pygame.draw.rect(surface, (255,255,255),
    (platform["pos"][0], platform["pos"][1], windowWidth, 10))
123.     pygame.draw.rect(surface, (0,0,0), (platform["gap"],
    platform["pos"][1], 40, 10) )
124.
125.
```

In order to start the game, we're checking for a space bar keyboard press on line 170. If a game hasn't been started (or has just finished), the **restartGame** function on lines 133–142 is called. All this function does is reset all of the variables for a new game. On the next loop of our main loop, the settings will be in place for a new game and one will be started.

The game platforms

Once the space bar has been pressed, a new game will begin. All of the game logic is between lines 40 and 146, but we call the methods to generate the game on lines 182–185 in a particular order. First, we call **movePlatforms()**, which works through every platform in the game and moves it up the screen at the speed set with the variable **platformSpeed**. **movePlatforms** also checks whether or not the platform has reached the top of our game window; if it has, it will remove that platform from our **gamePlatforms** list. You may notice that the **for** loop on line 110 is a little different from those we've used in the past. Unlike most **for** loops in Python, this one passes the index through to the loop with the **idx** value. We need this index so we can remove the right platform from the **gamePlatforms** list, otherwise we'd have to work through the list and try to figure out which one needs to go each time – and that wouldn't



Left Our 'Drop' game starts off easy, but gets harder as it progresses, creating platforms more quickly than before

```
126. def gameOver():
127.   global gameStarted, gameEnded
128.
129.   platformSpeed = 0
130.   gameStarted = False
131.   gameEnded = True
132.
133. def restartGame():
134.
135.   global gamePlatforms, player, gameBeganAt,
    platformsDroppedThrough, platformDelay
136.
137.   gamePlatforms = []
138.   player["x"] = windowWidth / 2
139.   player["y"] = 0
140.   gameBeganAt = GAME_TIME.get_ticks()
141.   platformsDroppedThrough = -1
142.   platformDelay = 2000
143.
144. def quitGame():
145.   pygame.quit()
146.   sys.exit()
147.
148. # 'main' loop
149. while True:
150.
151.   surface.fill((0,0,0))
152.
153.   for event in GAME_EVENTS.get():
154.
155.     if event.type == pygame.KEYDOWN:
156.
157.       if event.key == pygame.K_LEFT:
158.         leftDown = True
159.       if event.key == pygame.K_RIGHT:
160.         rightDown = True
161.       if event.key == pygame.K_ESCAPE:
162.         quitGame()
163.
164.   if event.type == pygame.KEYUP:
165.     if event.key == pygame.K_LEFT:
166.       leftDown = False
167.     if event.key == pygame.K_RIGHT:
168.       rightDown = False
169.
170.     if event.key == pygame.K_SPACE:
171.       if gameStarted == False:
172.         restartGame()
173.         gameStarted = True
174.
175.   if event.type == GAME_GLOBALS.QUIT:
176.     quitGame()
177.
178.   if gameStarted is True: # Play game
179.
180.     timer = GAME_TIME.get_ticks() - gameBeganAt
181.
182.     movePlatforms()
183.     drawPlatforms()
184.     movePlayer()
185.     drawPlayer()
186.
187.   elif gameEnded is True:
188.     # Draw game over screen
189.     surface.blit(game_over_image, (0, 150))
190.
191.   else :
192.     # Welcome Screen
193.     surface.blit(title_image, (0, 150))
194.
195.   if GAME_TIME.get_ticks() - lastPlatform > platformDelay:
196.     createPlatform()
197.
198.   pygame.display.update()
199.
```

be good for the frame rate. The function **pop** removes an item from a list at a given point; if we wanted to remove the second platform in the list, for example, we'd pass **gamePlatforms.pop(1)** – remember, lists begin at 0, so 1 is the second item in our list.

Once we've worked out where the platforms need to go and which ones need to go away, we can draw them. We do this with **drawPlatforms** on lines 118–123. Nothing fancy here; we're just drawing a white rectangle which is the width of the screen, and then a black rectangle for the gap that the character can drop through to the next platform.

But where do these platforms come from? On line 196, we find the answer. Pygame keeps track of how long the game has been running for with its **get_ticks** function. We want to release a platform around every 2 seconds; so, on each loop, we check to see how long it's been since we created a new platform by subtracting the time that we last created a platform from the current game time, which we access with **GAME_TIME.get_ticks()**. The game time is recorded in milliseconds, so if 2,000 milliseconds (1,000 milliseconds = 1 second) have passed since we generated a platform, it's time to create a new one; we do that with **createPlatform** on line 94.

On line 96, we use **global** to tell our code that we want to use variables that exist in the global scope, not create new ones with the same name in the function scope. On lines 98–99 we're creating variables that will define the position of the platform (**platformY**) and the location of the gap through which to drop along with it (**gapPosition**). We want our platform to always rise from the bottom of the window, but the gap can be at any point along the length of the platform.

Just as tracking players becomes difficult when we have lots of them to deal with, the same is true of our platforms here. We're generating a platform every 2 seconds – and that delay gets smaller each time. If you've been playing for more than a minute, you'll have jumped on something like 100 platforms! We can't pre-program all of those and even if we could, our game would become very bland after a couple of plays. Line 101 is where we create our new platforms. Like any list, we can add new items to it; in Python, we can do this with **.append()**. In this case, we're creating a dictionary with everything we need to create a platform, the position of the platform (stored with the **pos** key), and the location of the gap (stored with the **gap** key).

```
gamePlatforms.append({"pos" : [0, platformY], "gap" : gapPosition})
```

List
Dictionary
Key
Value

Above Here's a handy reference to help with appending a dictionary item to a list

Moving our avatar

Our avatar isn't a complicated construct – he's a Capricorn, likes long walks on the beach, and is a cat person – at its simplest, it's a red rectangle. After our platforms are drawn, we want to work out where our avatar can go. As you'll see, the **`movePlayer()`** function on lines 44–92 is home to most of our game's logic. Before we look at the code, let's talk about what's happening: we want our player to fall when there is either a gap in the platform or no platform at all. We also want the avatar to travel up with the platform if there is no gap present. To code this logic, we could check the position of all of the platforms every frame and write some code that would figure out whether or not our avatar is on top of a platform or not, but that's really going to make our Pi work hard – it wouldn't be efficient. Instead, we're doing something simpler: our platforms are always white and our background is always black, so if we can know the colour of the pixel just beneath our avatar, we can work out whether or not we need to drop or not. Simple!

We also need to check that our avatar is completely off the edge of our platform before we drop. To do this, we check the values just beneath our avatar, to both the left and the right. We can get the colour of a pixel at a certain point with **`surface.get_at((X, Y))`**; this will return a tuple with four values **`(RED, GREEN, BLUE, OPACITY)`**, each between 0 and 255, just as if we had set the colours ourselves. On lines 51–52 we check the colour beneath the bottom left of our avatar, and on lines 54–55 we do the same for the bottom right. If the colour values we find at either the bottom left or the bottom right of the avatar are **`(255,255,255,255)`** (white), then we know at least one edge of our avatar is still on a platform. If both are anything but white, then there's a gap in the platform or we're in blank space, so we can let our avatar drop. This all happens on lines 57–68. We also check that we don't let our avatar run off the bottom of our window – we can't have our red rectangle flying off into infinity now, can we?

So, that's the code that handles what to do if we aren't on top of a platform, but what about when we want our avatar to travel with the platform? Well, if our avatar finds itself unable to go down, we need to work out where the platform stops and the blank space starts. We do this on lines 64–80. On lines 66 and 67 we set two variables, **`foundPlatformTop`** and **`yOffset`**; we use these values to help our **`while`** loop on lines 70–80. When we find a pixel that's white beneath either the bottom left or right of our avatar, we need to work backwards to move our avatar up with the platform. Our **`while`** loop subtracts 1 from our **`player["y"]`** value and checks the colour that it finds there. Remember, we haven't drawn our avatar yet, so the only colours on our surface are black (background) or white (platforms). If the coordinates checked are white, 1 is added to the **`yOffset`** and the **`while`** loop continues to search for a black pixel. It'll do this until it finds a black pixel above the x coordinate of our avatar, adding 1 to the **`yOffset`** variable each time. Once a black pixel is found, we've discovered where our platform ends and can subtract the **`yOffset`** from **`player["y"]`** to put our avatar just on top of the platform; this is done on line 73. If we don't find a black pixel before we reach the top of the surface, it's game over: our avatar is trapped off screen.

Moving our character left and right is done on lines 82–92. If the code looks familiar, it's because we used it in our last tutorial to move our squares around. Now that we've worked out where our avatar can go, we can draw it by calling **`drawPlayer()`** on line 203.

Left Just like our start screen, our game over screen is just an image drawn straight onto our surface when we need it

Game over

We're almost done; the last thing we want to handle is what happens when our player loses the game. It's literally game over once our avatar disappears off the top of our screen, and we want to tell the user that. When our avatar disappears, we call the game over function on line 79. All the **`gameOver`** function does is set some variables that our main loop will check to see if the game is underway. Once **`gameEnded`** is **`True`** and **`gameStarted`** is **`False`**, our main loop will draw our game over screen. Just like our welcome screen, we draw our game over image onto the surface on line 193 and give the player the option to restart the game with another space bar press.

And that's it! Using all the skills we've already acquired (and a few new ones), we've built our first fully fledged game. Like all good games, we've got a start, a middle, and an end.

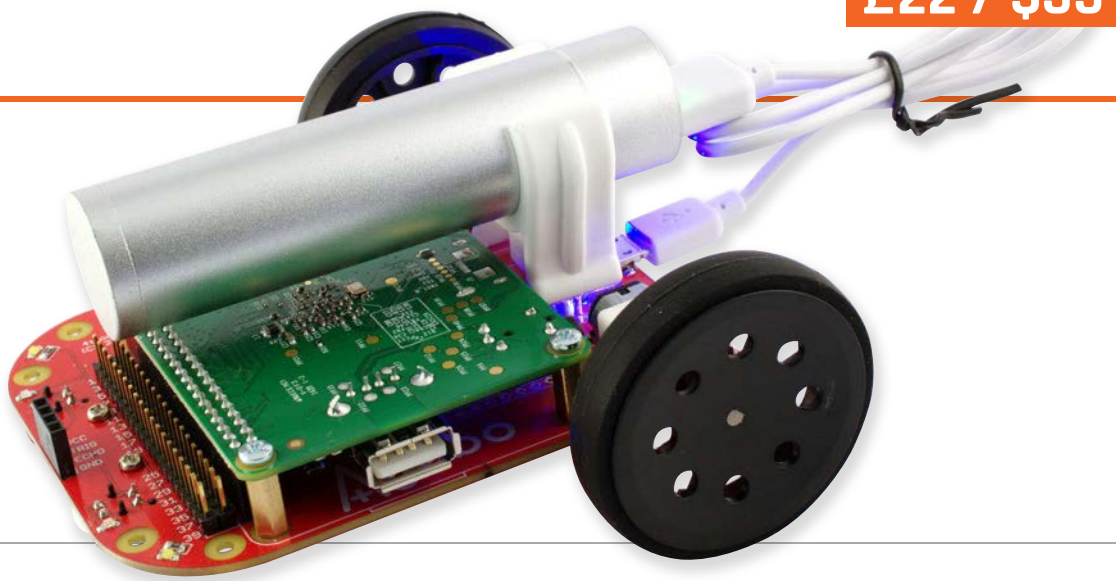
4tronix.co.uk

£22 / $35

Maker Says

"The easy-peasy robot for the Raspberry Pi Model A+"

4tronix



4TRONIX AGOBO

The Agobo is designed to cut a few corners in its pursuit of simplicity. **Gareth Halfacree** puts it to the test...

The Pi2Go and lower-cost Pi2Go-Lite wheeled robots from UK-based 4tronix have proved popular, but there is still a gap in the market for a no-solder and even lower-cost option suitable for educational use. Enter the Agobo.

Unlike the somewhat bulky kit-form Pi2Go design, the Agobo is a single circuit board which arrives with all the components pre-soldered. That's not to say it's ready-to-go: a small bearing assembly needs putting together and screwing to the front, which can be fiddly when the extended screws and spacers are required. You're well advised to do this over a container that will catch the smaller ball bearings when the cover slips from your fingers.

Trickiest part

The bearing assembly is the trickiest part of the process by far, however. Once that's fitted, there's a single bolt to secure the battery holder in place, then four brass pillars to support the Pi itself. Unlike the Pi2Go, the Agobo is compatible exclusively with the Model A+ – a sensible design choice, given its battery-sipping power characteristics. This attaches upside down and connects to a female GPIO header on the Agobo board, and is then secured in place with four screws and an optional but attractive protective acrylic plate.

The Agobo is billed as a hackable robot, and it certainly is. The GPIO header is replicated at the front of the board for the addition of any extra hardware, and a separate I²C breakout makes the connection of sensors very simple if the on-board line-following sensors aren't enough for your needs. An optional add-on dubbed the PlusPlate provides a large prototyping area, programmable RGB LED, and an nRF24L01-compatible socket for the addition of a radio module.

Despite its low price, the Agobo feels solid and robust. The thin wheels don't offer the traction of its full-sized competitors, but the metal-geared N20 motors are surprisingly powerful for their size. The use of an off-the-shelf lipstick-style USB battery is clever, and it can run the device for a considerable time per charge, although the length of the cable and the need to leave it dangling from the back like a tail is somewhat disappointing. The downloadable Python code examples are clear and work well.

Last word

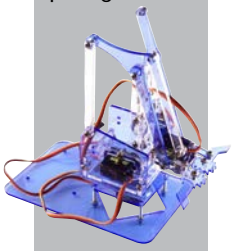
While parts of its assembly can be fiddly and it lacks the features of its more expensive competition, the Agobo is a perfect introduction to simple wheeled robotics for owners of the low-power Model A+.

★★★★☆

Related

PHENOPTIX MEARM

A desktop robotic arm, the MeArm requires a PWM controller add-on for Pi use, but offers amazing control in an open hardware package.

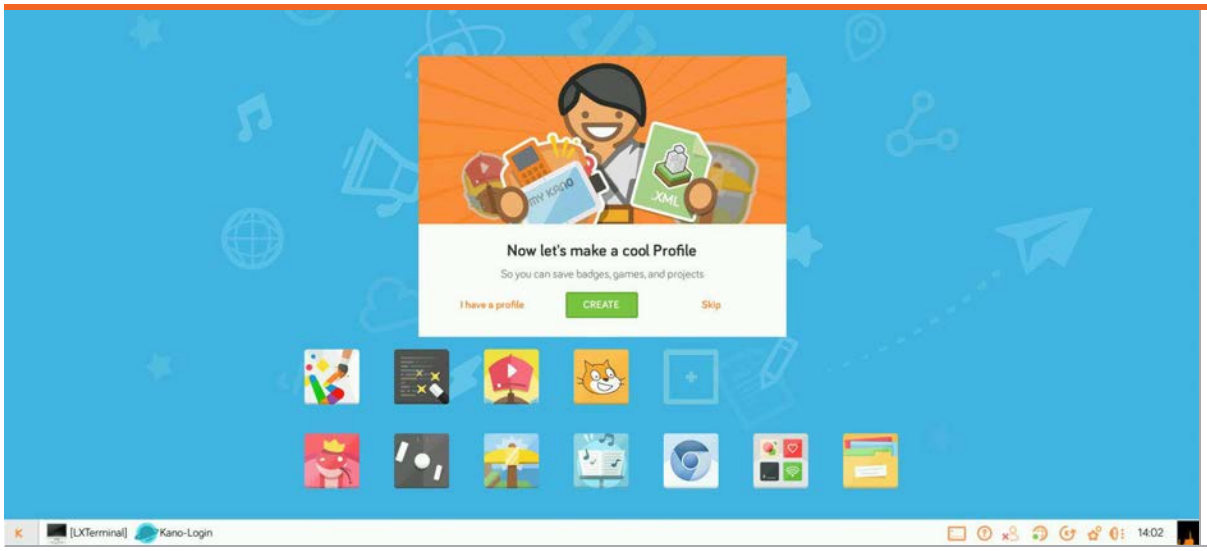


£33 / $55

phenoptix.com

kano.me/downloads

FREE



Maker Says

An open source OS for exploration, creation, and play – free for Raspberry Pi

Kano.me

KANO OS BETA 2.0.0

Les Pounder tests a Raspberry Pi distro for kids, designed to teach important computing concepts via a series of games and challenges...

The Kano first appeared as a crowdfunding campaign in late 2013. The startup sought $100,000 to produce a Raspberry-Pi-powered computer kit that anyone could make. They went on to smash their funding target, raising $1.5m, and have subsequently successfully fulfilled rewards to their backers.

The latest Kano offering ships as a kit consisting of a Raspberry Pi 2, case, speaker, and a rather snazzy wireless keyboard. In this kit you will find a microSD card with a copy of the Kano OS, but you can download the OS for free via the Kano website (**kano.me/downloads**).

Gamification

The latest version of the project's Raspbian-based operating system, Kano OS 2.0.0, provides a slick and child-friendly interface. On your first boot, you are tasked with setting up your Pi via a series of *Matrix*-style challenges. Once set up, you're presented with a fresh interface that offers common applications such as Sonic Pi, Scratch, and *Minecraft*.

What's particularly novel about Kano's approach to learning is the team's use of gamification (learning via a challenge-based system), to encourage the user to stick with it and make more progress. Each of the challenges built into Kano incentivises you to progress via an achievements scheme which tracks your progress and shares your status with the other Kano users via Kano World, an online resource for additional Kano projects created by members of the community. As Kano is based upon Raspbian, it's easy to update your software via the built-in updater, and if you require more applications then you can easily drop into a Terminal and use apt-get to install.

Kano's interface is rather lovely, but at times it really did struggle to catch up with us and there were times, such as when loading the apps menu, where we had to wait for a few seconds for the screen to populate. However, Kano is a good choice for small children who want to experience learning for themselves, and could be used as a stepping stone on their learning pathway.

Last word

Kano OS is a bright and fun way to learn computing and it serves the target demographic rather well. Children and parents can enjoy learning at their own pace thanks to the clever games within.

★★★★☆

Related

RASPBIAN

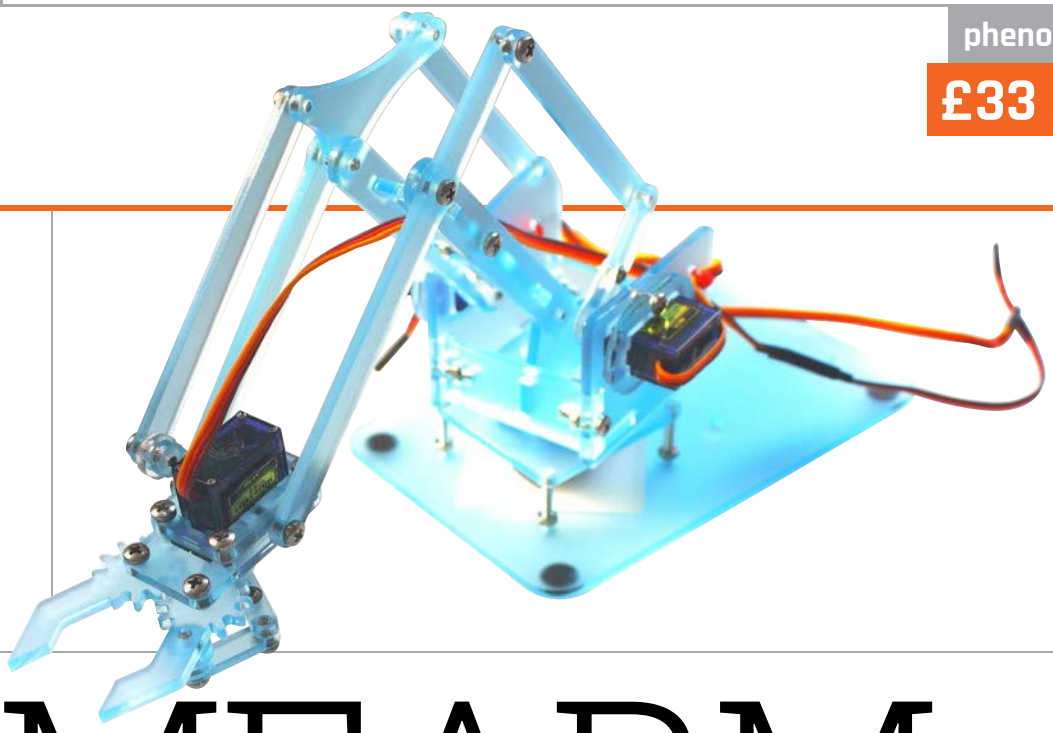
The default operating system for the Raspberry Pi has recently had a user interface overhaul and is now a lovely looking and well-supported platform for projects.

FREE

raspberrypi.org

phenoptix.com

£33 / $55



Maker Says

"The pocket-sized, affordable open-source robot arm"

phenoptix.com

MEARM

Russell Barnes tests an affordable, open source robot arm designed to get learners started with physical computing...

They're at the core of industry and they're soon to be at the forefront of modern medical surgery. Robot arms also make brilliant learning tools, especially for those wanting to utilise the GPIO pins on their Raspberry Pi. Historically, robot arms have been just north of affordable for the average bedroom hobbyist, but all that changed with the arrival of the MeArm.

You don't even have to take it out of the box to appreciate its strongest facets. In fact, you don't even need a box. You can build this open source arm from plans available on Thingiverse (you can 3D-print it, cut it from a single sheet of acrylic, or even whittle it from wood). For just £5 / $8, you can buy all the screws and fixings needed from its maker, **phenoptix.com**, or purchase a matching set of four hobby servos for £12 / $18. If you can't 3D-print, whittle wood or laser cut acrylic, you can buy a full MeArm kit, including everything you need to build a complete MeArm (minus an add-on board to drive the servos), for just £32 / $50.

The design itself is very clever and over the course of 2014 its creators, Ben Gray and Jack Howard, tweaked the design through four iterations, gradually refining the build, its instructions, and some code examples you can use to control it.

The build process is very hands-on. It has the feel of a 3D jigsaw puzzle when you take the parts from the box, turning the process into a fun afternoon project in its own right. The instructions are picture-led and thorough, but it's not entirely devoid of frustration – there's still some room for improvement.

As well as being a pleasure to look at, the MeArm is surprisingly sturdy. If you're expecting a certain grip strength or pinpoint precision, you're entirely missing the point of the MeArm. That said, we were happy with its accuracy and its ability to recreate pre-defined movements.

While there are plenty of options available to drive the four servos (or even the option to buy a fully soldered Adafruit 16-channel servo HAT from **phenoptix.com**), this stands as a missing piece from the MeArm puzzle – meaning it's up to the user to find their way in this regard.

Last word

The creators of the MeArm set out to build an affordable, open source robot arm and the perfect introduction to robotics. They very much succeeded and, despite minor shortcomings, we can't recommend it enough.

★★★★★

Related

4TRONIX AGOBO

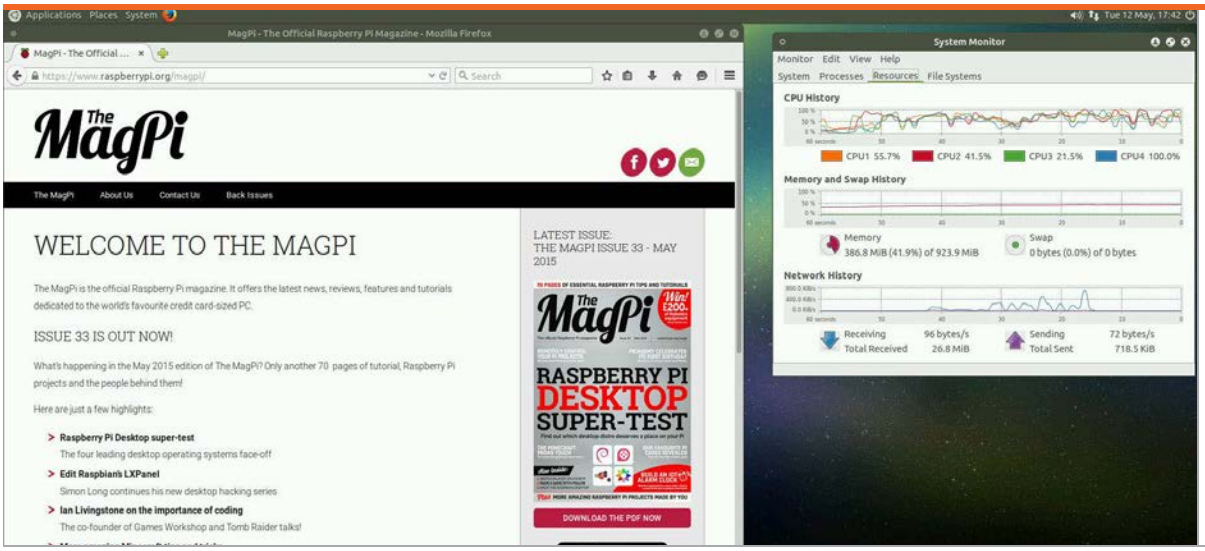
See the review of this tiny and affordable Model A+-specific robot on page 56 this issue.

£33 / $55

4tronix.co.uk

ubuntu-mate.org/raspberry-pi

FREE



Maker Says

"We have done what we can to optimise the build for the Raspberry Pi 2

ubuntu-mate.org

UBUNTU MATE 15.04

A lightweight version of the latest Ubuntu release arrives for the Raspberry Pi 2. **Les Pounder** sees how it stacks up against the competition...

Ubuntu has long been a frontrunner in the Linux community, but it is only recently that it has been able to enter the Raspberry Pi arena. With the Raspberry Pi 2 and – more specifically – its ARM7 processor, we are starting to see multiple instances of Ubuntu on our $25 PC.

Ubuntu MATE is a lightweight distribution based on the popular MATE desktop, which is a fork of the more traditional GNOME desktop. Ubuntu MATE is well equipped to work with limited resources and it copes remarkably well with the Raspberry Pi 2 hardware, but since the I/O throughput on the microSD card is something of a performance bottleneck, its makers recommend a class 6 or 10 microSD card. On first boot you are asked to create a user account, and this takes around five minutes to complete. In our test the installer crashed right at the end, but everything was fine on reboot.

A grown-up distribution

Ubuntu MATE has all the bells and whistles, such as mounting remote file servers, a control centre for system settings, and the Firefox web browser. It also comes with LibreOffice, Thunderbird mail client, and Transmission torrent manager. Media playback is handled via VLC, but for best results we advise using omxplayer in the terminal. On the downside, this first release is missing Pi essentials like GPIO and camera support, which is a shame, but work is progressing to add them via the raspi-config menu system.

Software Centre

Installation of software is handled via the Ubuntu Software Centre or, for those who are comfortable with the terminal, via the Apt package manager (unlike other Ubuntu distros for Pi 2 which use the newer Snappy core). Administration tasks such as backups and user configurations can be handled via the System menu at the top of the screen, giving Ubuntu MATE a lovely professional feel.

Ubuntu MATE is still in the early stages of development, though the team are making great progress. It will be interesting to see if this popular Linux distro is adopted by the community in a similar manner to Raspbian. If you fancy testing Ubuntu on your Raspberry Pi, this is a good place to start.

Last word

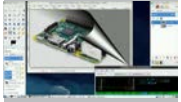
Ubuntu MATE is the best of the latest Ubuntu distributions for the Raspberry Pi 2. It provides a usable desktop, consistent responsive experience, and an easy method to install software.

★★★★☆

Related

PIDORA

A full Fedora distribution for the Raspberry Pi, that has everything you need to use your Pi as a professional workstation.



FREE

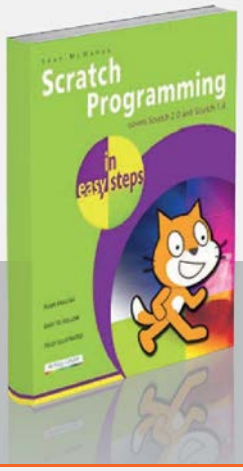
pidora.ca

RASPBERRY PI BESTSELLERS

SCRATCH PROGRAMMING

The visual programming language helping learners to progress, from Code Club to college...

SCRATCH PROGRAMMING IN EASY STEPS



Author: Sean McManus
Publisher: In Easy Steps
Price: £10.99
ISBN: 978-1840786125
bit.ly/1H6lEoz

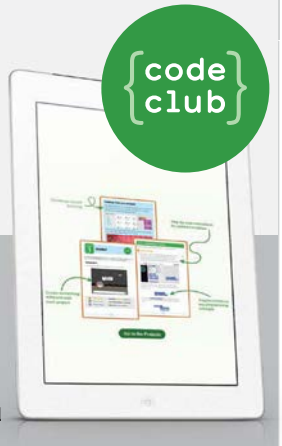
A well-thought-out book for the young and new to coding, which can be confidently given to those without programming parents to support them, and will get the parents programming, too!

SUPER SCRATCH PROGRAMMING ADVENTURE!

Authors: The LEAD Project
Publisher: No Starch
Price: £16.50
ISBN: 978-1593275310
nostarch.com/scratch

The next step after Code Club or the Easy Steps book – children continue to learn by doing as they follow a comic-book story and create games in the classic arcade style.

CODE CLUB



Authors: Code Club
Publisher: Morgan Kaufmann
Price: Free Online
ISBN: N/A
codeclub.org.uk/projects

Term 1 and 2 of Code Club, as successfully taught by hundreds of volunteers to thousands of 10- and 11-year-olds across the UK and beyond. CC-licensed, hosted on GitHub, and now translated into several languages.

RASPBERRY PI BLUEPRINTS

Author: Dan Nixon
Publisher: Packt
Price: £29.99
ISBN: 978-1784392901
bit.ly/1KOJLLI

Ten hardware projects to build up your knowledge of hardware, software, and the Pi. The first project, a 'pirate' radio transmitter, needs only a small piece of wire as an antenna – the Pi does the rest. Other projects gradually add more components but retain the wow factor; these culminate in the 'bottle xylophone' which converts MIDI files into sound with the help of 15 glass bottles, played by 15 Pi-controlled servos.

While some projects, like 'home theatre' and arcade cabinet, may be found in a number of places, they are well executed and accompanied by more exotic projects like the bottle xylophone. Certain projects need traditional construction skills, such as the magic mirror, which combines a regular mirror with a web display. Although involved, the instructions for the web-controlled robotic arm are clear, and following the steps takes you through a streaming server and network control of the Pi.

With its combination of software, electronics, and construction hardware, Nixon's book balances useful learning in all of the areas that make the Pi such a powerful and flexible basis for projects. The Weather Station project even involves reverse engineering and understanding the workings of off-the-shelf Maplin sensors to get them working with the Pi. Imaginative, educational, and fun.

Score ★★★★★

MAKER PRO

Author: Edited by John Baichtal
Publisher: Maker Media
Price: £13.50
ISBN: 978-1457186189
oreil.ly/1PmhOSc

It starts with curiosity, the purchase of a Pi... and maybe some components to connect to the GPIO pins. Soon, a hobby grows into a passion, and the latest project looks like it could be a business in its own right. What next? *Make:* has collected interviews and essays from 17 makers who've followed diverse paths from passion to profession.

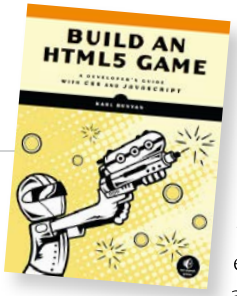
Here we find the artist Susan Solarz, who found the resources and community of her local makerspace gave her what she needed to create the Origami Rocker; the genesis of MakerBot Industries; the excitement of democratising science through DIYbio; the Nintendo guitar; and other projects, familiar and unfamiliar. Not every maker has left their day job behind, and the diversity of routes to professional makerdom exposes a less diverse commonality of background and education, but also shows that the tools are within reach of every reader.

The final, short chapter, Sophi Kravitz's 'Quit Your Day Job – a tale of moving from full-time through part-time work to get time for projects, then striking out on her own – cannily doesn't oversell the dream. A dream it may remain for many, but makerspaces and open source hardware are helping to drive real changes in manufacturing, and small boards like the Pi give the opportunity to join this new world.

Score ★★★★½

BUILD AN HTML5 GAME

Author: Karl Bunyan
Publisher: No Starch
Price: £19.99
ISBN: 978-1-59327-575-4
nostarch.com/html5game



There are some great beginner books for learning coding (see the Scratch Bestsellers sidebar), but we're often asked how to take the next step. Bunyan's excellent new book on web programming sits between beginner guides to JavaScript and the many specialist, advanced titles available for experienced web programmers.

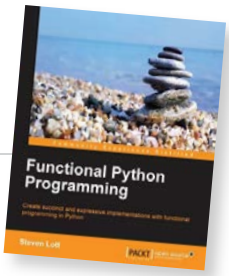
Assuming you have the basics of JavaScript, CSS, and HTML, the author walks readers through the creation of a classic Bubble Shooter game to play in the browser. Along the way, development techniques are demonstrated – chapter 1 includes using the browser's own debugging tools, and the JavaScript libraries Modernizr and jQuery are used.

It's not all shortcuts; the game logic chapter works the learner hard to understand the mechanics of implementing a game, and you'll learn a lot of CSS as you program the animations. After each chapter you'll have another stage of the game complete, feeling some sense of achievement, and ready to learn more. CSS transitions are introduced to speed things up, and HTML5 is used to add missing features like scoring and sound effects. By this point you'll be all ready to develop your own game idea, and maybe produce the next answer to *Angry Birds*.

Score ★★★★★

FUNCTIONAL PYTHON PROGRAMMING

Author: Steven Lott
Publisher: Packt
Price: £30.99
ISBN: 978 1784396992
bit.ly/1JMnhee



If you've wanted to try functional programming, but the thought of learning the strange syntax attached to most FP languages has added to your inertia, here's something for you. Python is a multi-paradigm language, coping well with procedural programming for scripts and many Raspberry Pi hardware projects, and object oriented for programs used at the enterprise scale; it can also be used in a functional style.

Yes, Python lacks unlimited recursion, lazy evaluation of all expressions, and an optimising compiler, but functions are first class objects in Python and the language does have all of the things you already love about it.

Lott shows you how to add to these, using Python to implement functional techniques and design patterns.

If you really want to throw yourself into functional programming, this is not the only book you should read on the subject. But if you want to improve your Python, you'll find in this book the tools to write functions that will help you work far more efficiently with Big Data, for example. Every chapter has something to offer, and you'll finish this book a better programmer.

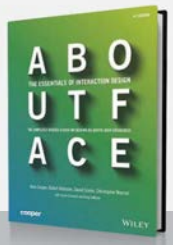
Score ★★★★☆

ESSENTIAL READING: BUILDING A WEBSITE

Skills needed for building websites and apps cover diverse technologies for many ability levels...

The Essentials of Interaction Design, 4th Edition

Author: Alan Cooper et al
Publisher: Wiley
Price: £33.99
ISBN: 978-1118766576
bit.ly/1L64hYV



Understand interaction design and you won't be doomed to join the ranks of frustrating, unusable websites.

Creating Flat Design Websites

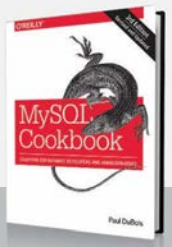
Author: António Pratas
Publisher: Packt
Price: £26.99
ISBN: 978-1783980048
bit.ly/1bTORug

Concise and pragmatic guide to the dos and don'ts of implementing the look *de nos jours*.

MySQL Cookbook, 3rd Edition

Solutions for Database Developers and Administrators

Author: Paul DuBois
Publisher: O'Reilly
Price: £53.50
ISBN: 978-1449374020
oreil.ly/1QOzUJC

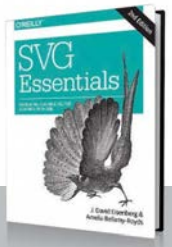


Love it or loathe it, you'll find MySQL everywhere, so try O'Reilly's comprehensive tutorial and reference.

SVG Essentials, 2nd Edition

Producing Scalable Vector Graphics with XML

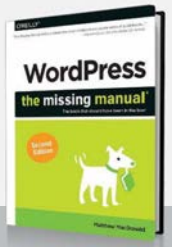
Author: J. David Eisenberg, Amelia Bellamy-Royds
Publisher: O'Reilly
Price: £26.50
ISBN: 978-1449374358
oreil.ly/1HkpsrH



Updated for modern browsers, learn the XML-based 2D vector image format built for transformation.

WordPress: The Missing Manual, 2nd Edition

Author: Matthew MacDonald
Publisher: O'Reilly
Price: £19.99
ISBN: 978-1449341909
oreil.ly/1dh6jdJ



Sometimes you just need to get a site up quickly. WordPress does the hard work, and MacDonald fills in the blanks.

RASPBERRY JAM EVENT CALENDAR

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

PUT YOUR EVENT ON THE MAP

Want to add you get-together? List it here:
raspberrypi.org/jam

1 CAMBRIDGE RASPBERRY JAM

When: Saturday 6 June
Where: Institute of Astronomy, Madingley Rd, Cambridge
camjam.me
One of the biggest regular Jams in the home of Raspberry Pi. Excellent facilities for talks, demos, and workshops

2 PLYMOUTH JAM

When: Saturday 13 June
Where: Fairbairn House, Higher Lane, PL1 2AN
bit.ly/1BqjyoX
MESH is holding the event at @THINQTANQ and entry will cost just £5 per person, or free for under 18s and OAPs.

3 SEATTLE RASPBERRY JAM

When: Wednesday 17 June
Where: Jigsaw Renaissance, Seattle, USA
meetup.com/Jigsaw-Renaissance
Seattle Jam takes place at Jigsaw Renaissance, which is located inside a vibrant art community called Inscape Arts.

4 SILICON VALLEY JAM

When: Saturday 20 June
Where: Computer History Museum, Mountain View, CA
bit.ly/1IKPPZ1
This regular event takes place every third Saturday of the month between 1:30pm and 4:30pm (PDT). Don't miss it!

5 COVENTRY MAKEJAM

When: Saturday 20 June
Where: Coventry Makerspace, Koco Community Building
bit.ly/1G23lGu
Come along and build a project, meet fellow Pi users, look at some cool projects, or even build your own!

6 RECURSION 2015

When: Saturday 4 July
Where: King Edward VI School, Stratford-upon-Avon
recursioncomputerfair.co.uk
A fun-packed day dedicated to computer science and computing, Recursion 2015 is free for all visitors and exhibitors.

7 PRESTON RASPBERRY JAM

When: Monday 6 July
Where: Media Factory Building, Preston, PR1 2HE
bit.ly/1J7KfOu

Come along for lightning talks, demonstrations, support, and hands-on experience with the Raspberry Pi.

8 TORBAY TECH JAM

When: Saturday 11 July
Where: Paignton Library, Paignton, TQ4 5AG
torbaytechjam.org.uk

Bring along your Raspberry Pis, Arduinos, laptops or other devices, and your tech projects for an afternoon of hacking!

DON'T MISS: CAMJAM

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

CamJam, held at the fantastic Institute of Astronomy in the heart of Cambridge, has been a highlight on the Raspberry Pi events calendar since its debut back in 2012. Run by Mike Horne and Tim Richardson, the event includes some of the best workshops for learners – young and old – to get to grips with Raspberry Pi projects, buy equipment, and learn about the latest goings-on in the Raspberry Pi community.

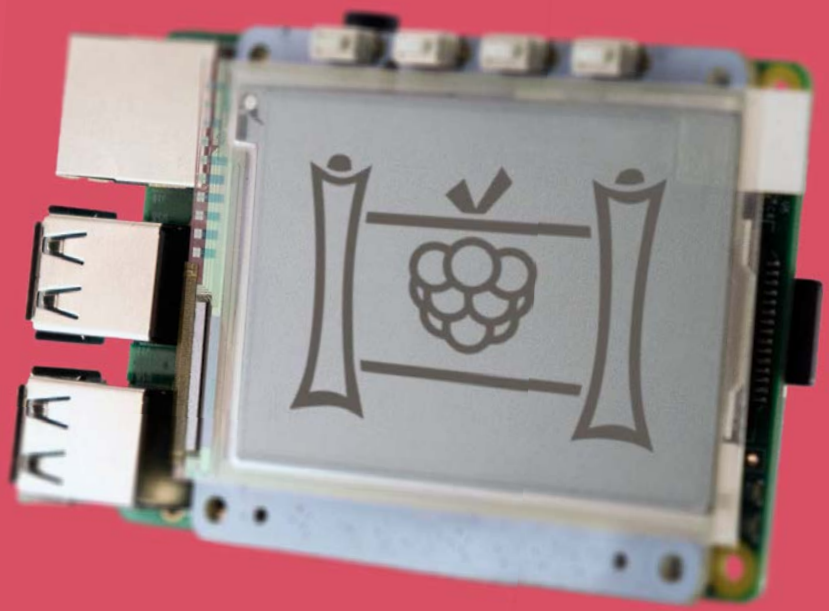
In association with:
Pi Supply
www.pi-supply.com

5 MULTI-SCREEN PAPIRUS EPAPER HAT PACKS MUST BE WON!

WHERE DO YOU READ ABOUT RASPBERRY PI?

(APART FROM The MagPi)

Tell us by 26 June 2015 for a chance to win!



Learn more about the PaPiRus ePaper HAT at papirus.ws

How to enter:

Email **competition@raspberrypi.org** and tell us about your favourite print or online source of Raspberry Pi projects, knowledge, and know-how (not including **raspberrypi.org** and *The MagPi*). Five random entries will win a £60 / $90 PaPiRus ePaper HAT with all three screens!

Terms & Conditions

Competition closes 26 June 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.

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.

WELCOME WINDOWS DEVELOPERS

Matt Richardson meets Windows developers and learns the significance of the platform coming to Raspberry Pi

At the end of April in San Francisco, Microsoft held its Build conference for developers. There, the company would release the first developer's preview of the Windows 10 IoT Core for Raspberry Pi 2. Microsoft invited me to attend in order to connect with its developers and introduce many of them to Raspberry Pi for the first time.

Between sessions, when Build attendees would course through the hallways and lobbies of the Moscone Convention Center, the Raspberry Pi booth was swamped with curious and excited developers. One of them picked up a Pi and examined it closely.

"There's so much I've wanted to do with this," he said to me. "But I just never had the time."

When I talk to people about Raspberry Pi and they tell me that they don't have the time to experiment with it, I'll usually reply with, "Well, we've figured out how to make an inexpensive computer, so now I guess we'll have to figure out how to give people more time." Admittedly, it's not my best joke, but in this case it led to a very meaningful conversation about how Windows 10 IoT Core for Raspberry Pi might be the next best thing to giving this particular developer more time.

He told me that as a Windows developer, the ability to use the programming language, development environment, and debugging tools that he was already using on a daily basis meant that he was much more motivated and excited to start developing for Raspberry Pi.

Like many others, he develops software for Windows platforms as his day job and has a few ideas for side projects around his house that could use a Raspberry Pi. Now that he's able to use his familiar work tools, he could make his project a reality in much less time, a nice runner-up to actually having more time in the day.

In a store near you...

At the conference, I also spoke to a technology lead from a large supermarket chain who had a lot of ideas about how Raspberry Pi could be deployed in its stores, from monitoring the temperature of refrigerator cases to tracking inventory. The company already had a large team of Windows developers and its approach to hardware was transitioning from using fully proprietary solutions to a more do-it-yourself mentality, one in which its own developers made tailored solutions with maker-friendly components. Because of his team's familiarity with Windows development, it's quite possible that the availability of Windows 10 IoT Core for Raspberry Pi would cut down on development time and would be less expensive than proprietary alternatives.

After being face-to-face with the Windows development community for three days, my perspective changed drastically about the significance of Windows coming to Raspberry Pi. Initially, I thought about Windows as another option for the Raspberry Pi community. While that's certainly the case, the more significant aspect of this new release is that it makes Raspberry Pi another hardware option for the vast Windows developer community. They'll be able to work with our inexpensive hardware and yet use the powerful tools they already have and love.

And even though we haven't found a way to give people more time to use Raspberry Pi, whenever a new software platform and development tool becomes available to use, it makes Raspberry Pi a more enticing and efficient tool for those developers. It allows more people to do more, much faster. And in this case, it's a *lot* more people. So, to the large group of Windows developers that are now trying Raspberry Pi for the first time, welcome to our community.

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

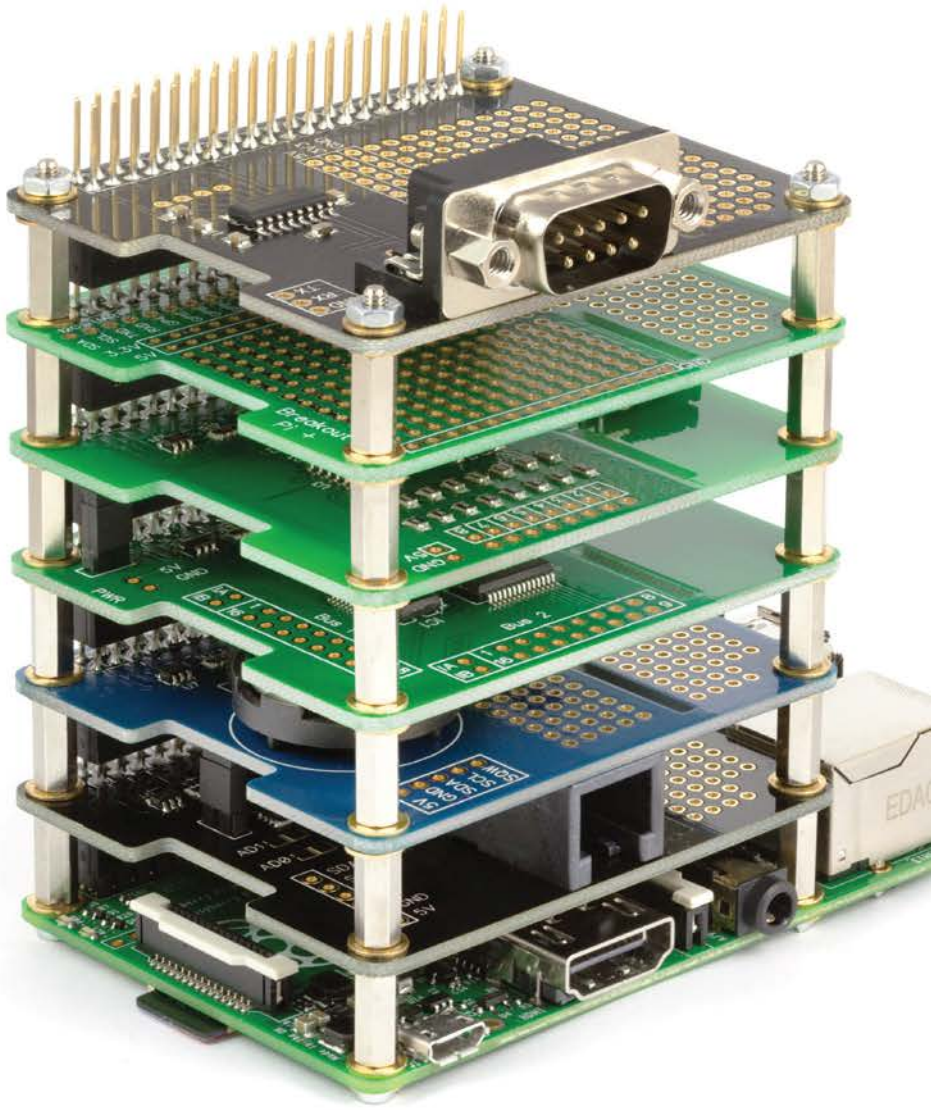
32 digital 5V inputs or outputs. I²C address selection allows you to stack up to 4 IO Pi Plus boards on your Raspberry Pi.

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