YOUR OFFICIAL RASPBERRY PI MAGAZINE

The MagPi

The official Raspberry Pi magazine

Issue 46 June 2016

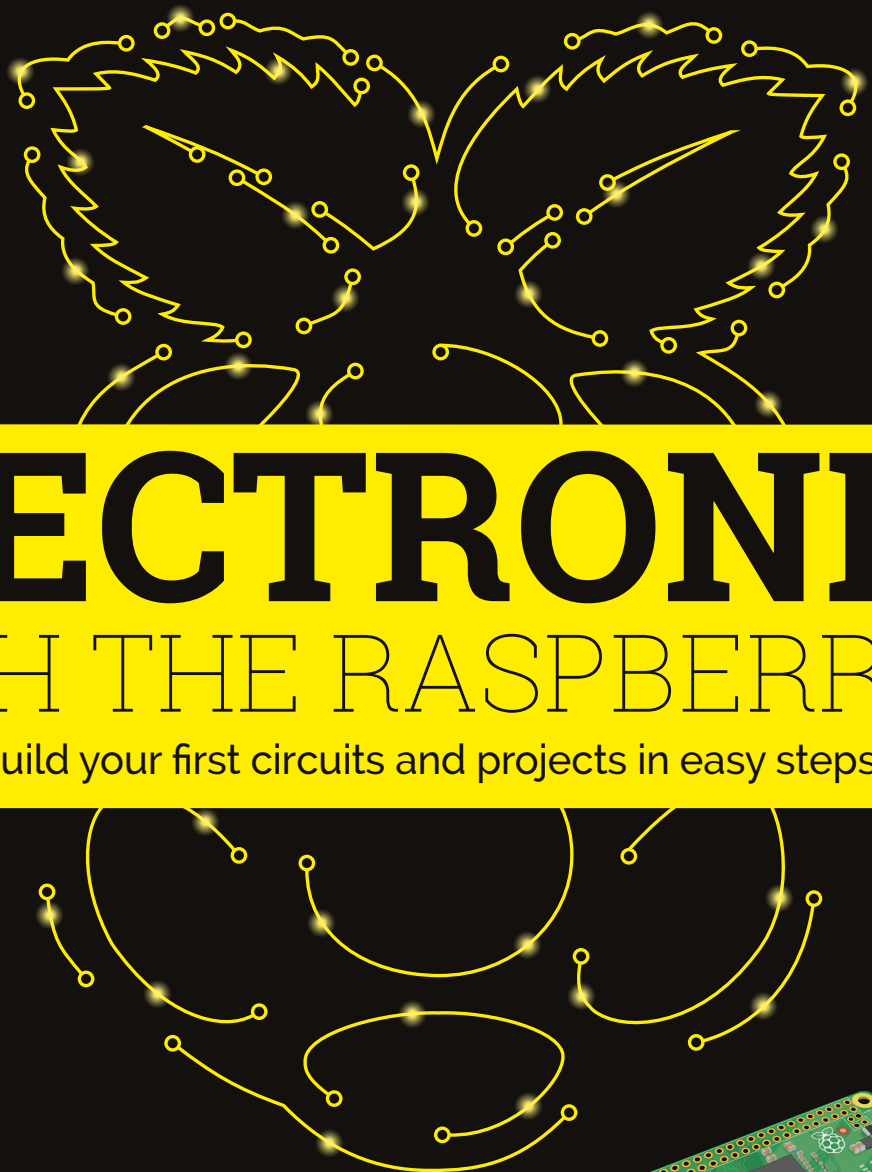
raspberrypi.org/magpi

HACK AN RC CONTROLLER WITH PI ZERO

Upgrade your robot project today

CODE AN ASTEROIDS CLONE

Put the finishing touches to our FUZE Basic game



ELECTRONICS

WITH THE RASPBERRY PI

Build your first circuits and projects in easy steps

ASTRO PI RESULTS!

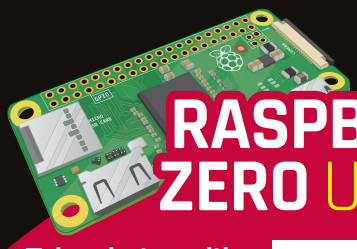
Analyse the results of Astro Pi experiments in your browser

BUILD AN IOT SCALE

Text your weight using Raspberry Pi

Also inside:

- CREATE A NIGHT-VISION CAMERA
- MEASURE GRAVITY WITH A SENSE HAT
- THE LATEST UBUNTU FOR PI REVIEWED
- YEAR 10 KIDS TRACK RISING SEA LEVELS



RASPBERRY PI ZERO UPGRADE

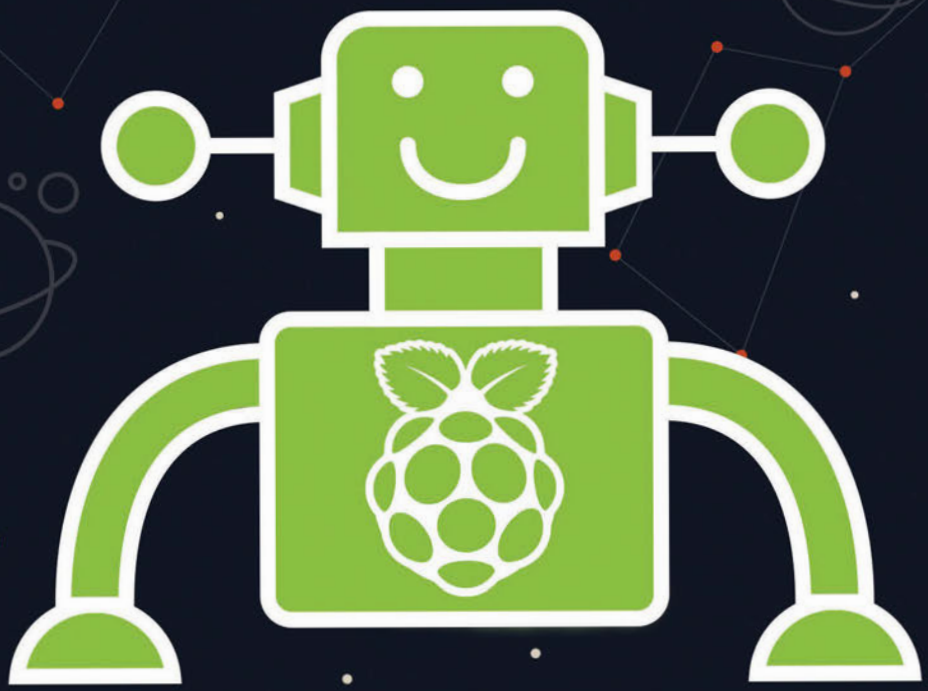
Take photos with the latest version of the Pi Zero!

Issue 46 • Jun 2016 • £5.99

9 772051 998001 06

THE ONLY PI MAGAZINE WRITTEN BY THE READERS, FOR THE READERS

VNC SDK now available for RASPBERRY PI!



With VNC SDK you can connect VNC Viewers with VNC Servers easily over the Internet. Using VNC Cloud, no complicated network configuration at either end is required!

What will you create with VNC SDK?
Check it out here: https://developer.realvnc.com/

REAL VNC

VNC DEVELOPER

Getting connected: www.realvnc.com/products/vnc/raspberrypi/
For more information contact vncdeveloper@realvnc.com

WELCOME TO THE OFFICIAL PI MAGAZINE!

When the Raspberry Pi Zero launched towards the end of last year, it turned the world of tiny computers on its head. Cheap enough to be given away on the cover of this very magazine, yet more powerful than the original Raspberry Pi, the Zero took off quicker than Tim Peake's rocket. While we've all been busy making consoles out of games controllers and robots out of matchboxes (seriously!), there's always been one type of project that's been sorely unavailable – anything using the Raspberry Pi Camera Module. Amazingly, that's no longer an issue: the Pi Zero 1.3 is here and it comes complete with a specially designed camera connector that works with both the old and new Camera Modules. Learn more about it on pages 6 and 7, or better yet, subscribe to the magazine for six or 12 months to get one sent to you completely FREE (see pages 34 and 35 for details).

Of course, you can't get very far with your Raspberry Pi projects without being able to control the General Purpose Input/Output (GPIO) pins. If you'd like to get stuck in, but don't know where to start, look no further than this month's cover feature starting on page 18. Electronics with the Raspberry Pi couldn't be easier!

Russell Barnes
Managing Editor

THIS MONTH:

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
Features Editor: **Rob Zwetsloot**
Sub Editors: **Laura Clay, Phil King, Lorna Lynch**

DESIGN
Critical Media: **criticalmedia.co.uk**
Head of Design: **Dougal Matthews**
Designers: **Lee Allen, Mike Kay**
Cover photography: **Gareth Halfacree**

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

DISTRIBUTION
Seymour Distribution Ltd
2 East Poultry Ave
London
EC1A 9PT | **+44 (0)207 429 4000**

SUBSCRIPTIONS
Select Publisher Services Ltd
PO Box 6337
Bournemouth
BH1 9EH | **+44 (0)1202 586 848**

CONTRIBUTORS
Wesley Archer, Jamie Bailey, Mike Cook, Bram Driesen, Iker Garca, Lucy Hattersley, Richard Hayler, Phil King, Matt Richardson, Jon Silvera & Richard Smedley

recycle
When you have finished with this magazine, please recycle it.

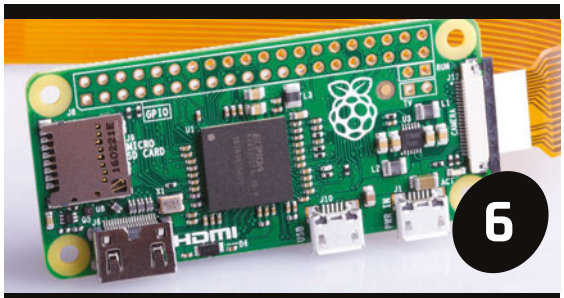
This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.

Contents

Issue 46 June 2016 raspberrypi.org/magpi

TUTORIALS

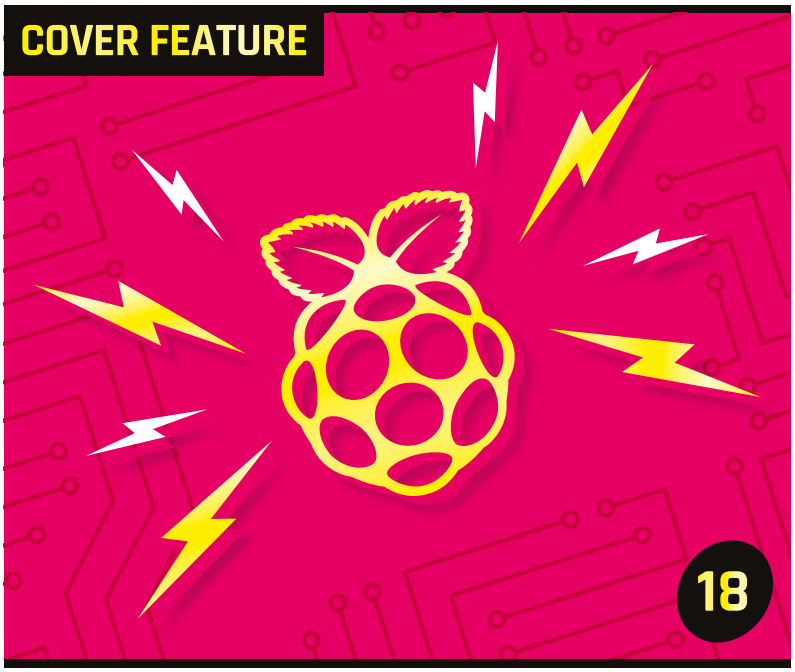
IN THE NEWS



Pi Zero 1.3 arrives

There's a brand new Pi Zero that lets you build even more amazing camera projects with Pi

COVER FEATURE



PI ELECTRONICS

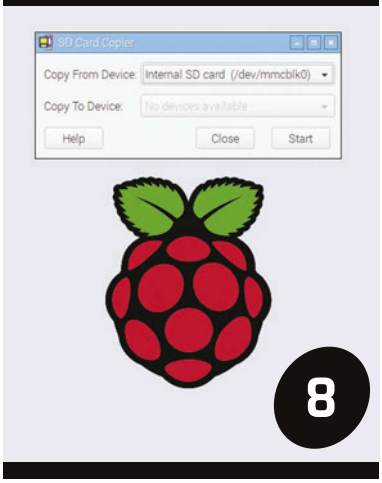
Get to grips with basic electronics projects on your Raspberry Pi

ASTRO PI 12

The results are in! Get your hands on the amazing data collected from the winning Astro Pi projects...

Raspbian

There's another new build of Raspbian available. Learn all about it on pages 8-9



Contents

THE BIG FEATURE

72

THE PI WEATHER STATION

Learn how to set up and use the amazing Oracle-powered Raspberry Pi weather station, and get a few ideas for what to do with it

5 RasPiO ANALOG ZEROS MUST BE WON!

We have five RasPiO Analog Zero boards to give away, and you can grab another today on Kickstarter!

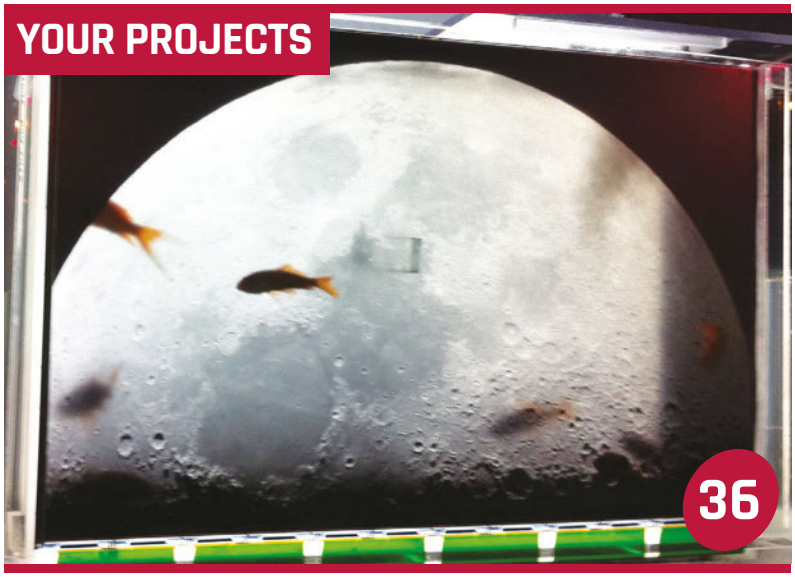
95

RISING SEA LEVELS

Find out how Cornish students watch rising sea levels with the aid of the Pi

10

YOUR PROJECTS



36

FISH-EYE

There's something decidedly fishy about Thomas Hudson's computer monitor

REGULARS

COMMUNITY

REVIEWS

PI ZERO REVISION

The Pi Zero is back in production and now comes with a camera connector

PI EYE IN THE SKY

To celebrate the launch of the new Pi Zero on 16 May, Dave Akerman launched one of them – with Camera Module attached – in a high-altitude balloon. As it soared from a Herefordshire field to the edge of space at an altitude of over 34km, some spectacular shots were captured (**magpi.cc/1XeuQCO**).

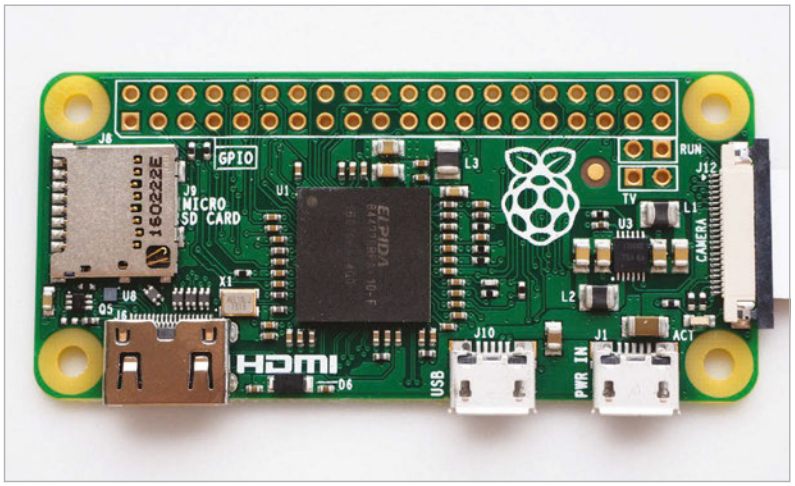
CAMERA ADAPTER

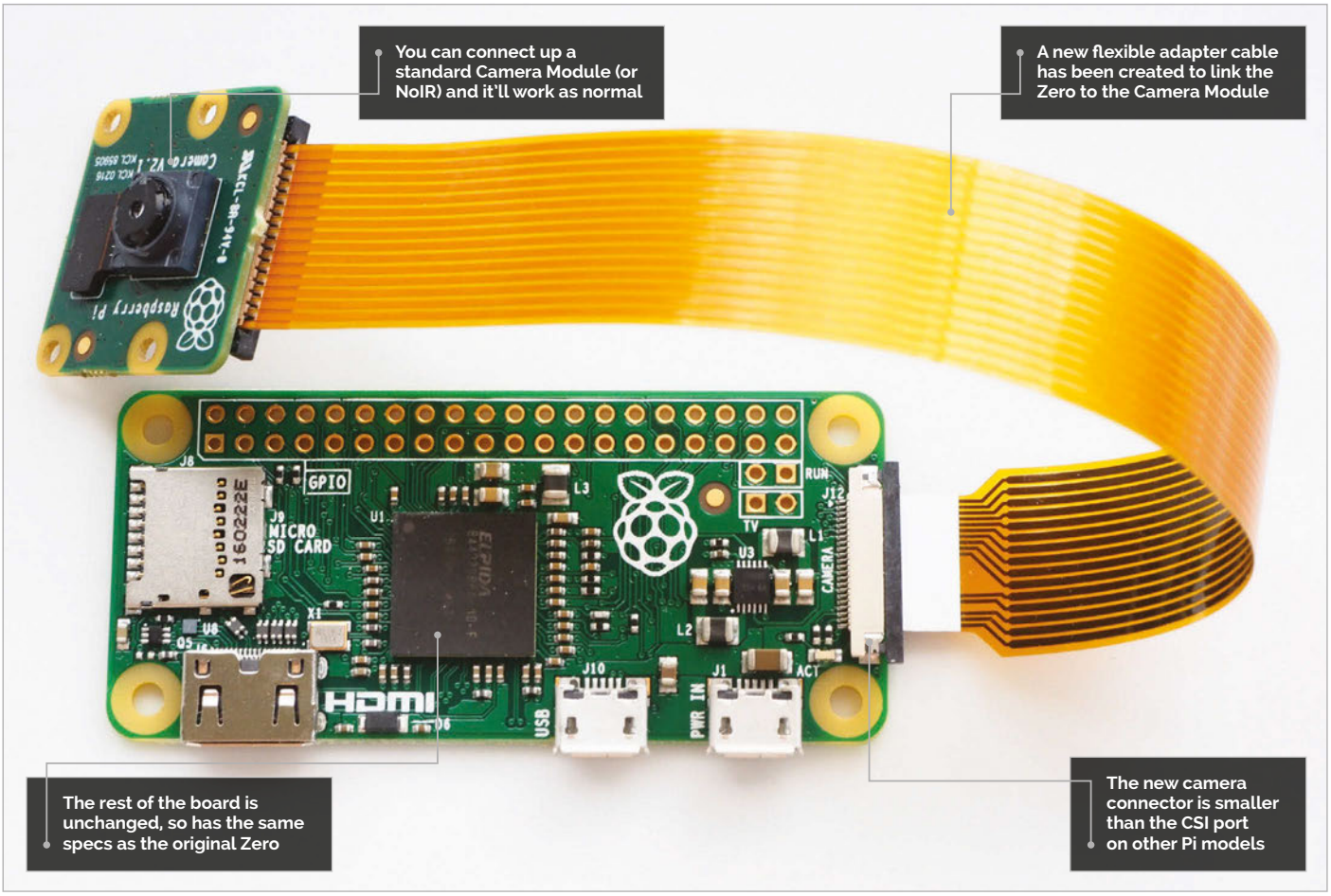
Through "dumb luck", it was discovered that the same fine-pitch FPC connector used on the Compute Module IO Board just fits onto the right-hand edge of the Pi Zero board. Instead of using the Compute Module's optional PCB adapter to connect to a Camera Module, however, a flexible custom cable has been designed. As well as being sleeker and more practical for miniature projects, it's less costly at £4/$5. Like the new Zero itself, the cable is available now from the usual Pi retailers.

For anyone trying to get their hands on a Raspberry Pi Zero, there's doubly good news. Not only is the smallest Pi model back in stock, due to a new production run of 250,000 units, but it comes with an extra feature: a new port to connect the Raspberry Pi Camera Module. This opens up a whole range of new possibilities for miniature projects using a camera.

To fit the diminutive form factor of the Zero, the camera port is slightly smaller than the CSI one on other Raspberry Pi models. Therefore, a special adapter cable is required to connect it to the wider connector on the standard Raspberry Pi Camera Module (including the NoIR). The new custom cable is flexible, 15cm (6 inches) long, and costs £4/$5 – the same price as the Zero itself.

While stressing that this is a revised version of the Pi Zero and not a brand new model, since the rest of the board is unchanged, Raspberry Pi founder Eben Upton is pleased that a recent hiatus in the production for the miniature model, during the run-up to the Pi 3 launch in February, enabled the team to add the new camera connector. Asked whether this was something they wanted to include originally, Eben admits: "No… It's a thing we didn't think we could add." More recently, however, following feedback from users requesting the feature, they realised they could fit a small camera connector to the right-hand edge of the board.





Meeting demand

Eben tells us that they were taken aback by the huge demand for the Pi Zero, following its launch last November. "It's one of the things with Raspberry Pi: on some level, we're always surprised." He tells us that they take a cautious approach to production numbers: "I have this nightmare that the one time I actually build a load, speculatively, I'll find that it's the one product that people don't like." He reveals that the original Model A didn't sell that well, since most users preferred the Model B, which had more features and was only slightly more expensive. Eben says if they'd ramped up production for the Model A at the time, "That really would have sunk us. So I think it's always better to be cautious."

Extra caution was required in the case of the Pi Zero, since its production is handled in-house, unlike other Raspberry Pi models whose manufacture is licensed to distributors RS Components and CPC/Farnell. While Eben insists that this licensing model "is and always will be the heart of the business", the Zero is not a natural fit for it due to not being a high-margin product.

> "They were taken aback by the huge demand for the Pi Zero"

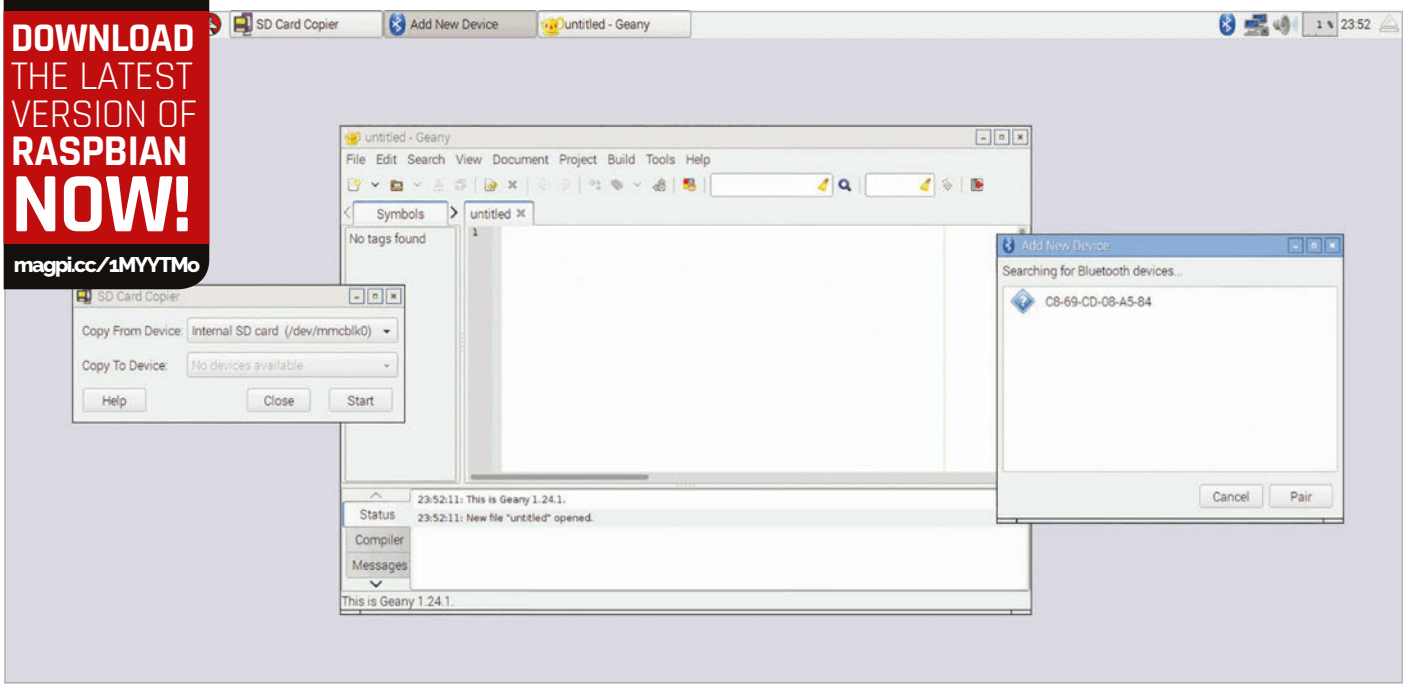
Now knowing that the Pi Zero is in such great demand has enabled Eben to confidently order an initial production run of 250,000. "We're launching with 30-40,000 units… We're going to push the rest of that stock through quite quickly." He reckons an annual production figure of "about a million" is achievable, which means that there should now be enough Zeros to meet demand.

PI ZERO PROJECTS

Eben has been delighted by the wide range of projects created so far using the Pi Zero. "It's been weird seeing how prevalent it's been," considering that the Zero represents only a small fraction of total Pi sales. "The small form factor [and self-contained nature] obviously enables a lot of stuff… build it into a mouse, build it into a joystick/game controller." With the camera connector now offering even more possibilities, we can't wait to see what users create with the new Zero.

DOWNLOAD THE LATEST VERSION OF **RASPBIAN NOW!**

magpi.cc/1MYYTMo



Above A slightly tweaked interface and some new software for Raspbian

RASPBIAN UPDATED

The latest version of Raspbian has some great additions, such as improved Bluetooth support and an internal backup tool

Raspbian, the official Raspberry Pi operating system, is always in development. There's always something new coming, with a desired features list a mile long that Raspbian crafter Simon Long works with to try to improve your Raspberry Pi experience. Recently, a big update to the operating system was released with a number of additions and changes that crossed off some of these features, while addressing support for recent hardware updates.

Improving Bluetooth

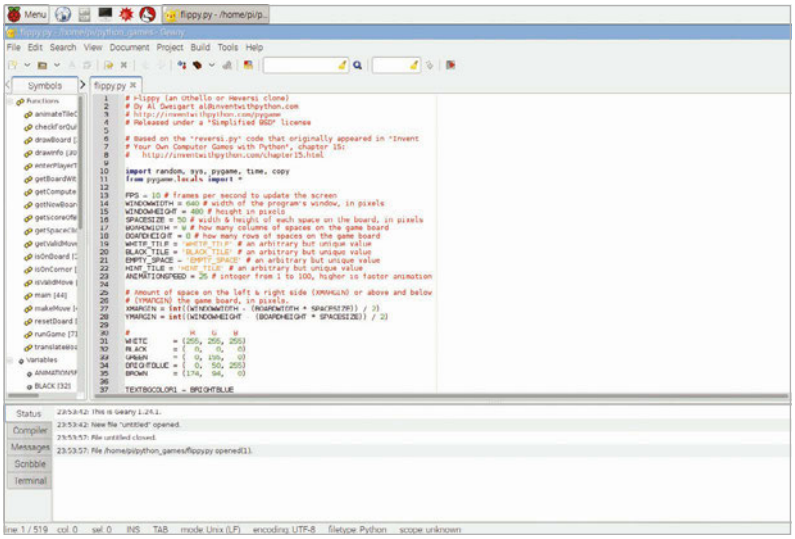
While support was added for the Bluetooth and Wireless LAN chip of the Raspberry Pi 3 during its initial release, the Bluetooth support was a little barebones. It was easy to upgrade using something like Blueman to add a graphical interface, though. In this release, Simon has added a custom Bluetooth interface.

"I'd hoped to be able to use one of the existing Linux Bluetooth UIs," Simon explains in the release blog. "[After] trying them all, none were really what I was looking for in terms of usability and integration with the look and feel of the desktop. I really didn't want to write one from scratch, but that was eventually what I did."

The interface is quite straightforward: by clicking on the Bluetooth icon in the taskbar, you can make the Raspberry Pi discoverable to other Bluetooth devices or add a Bluetooth device. The 'Add New Device' window will fill up with local Bluetooth devices which you can then pair with. Right now, there's only full support for keyboards, mice, and audio devices. Bluetooth audio now works slightly better than before, due to the inclusion of PulseAudio tools. If you recall from a tutorial back in issue 44, you needed to install the latter to get the audio output working.

SD card backup

A common question in the Pi community is how to back up the SD card. There are several existing



Above Geany also works with Python and may be preferable to IDLE for some people

methods, although none of them are native to Raspbian. A new tool has now been added that allows you to copy the SD card while the Raspberry Pi is running.

The SD Card Copier requires you to have a USB card reader for it to work. It's fairly simple: you just need to plug an SD card into the reader, plug the reader into the Raspberry Pi, and tell the SD Card Copier to copy the data to that card.

"The program doesn't restrict you to only copying to a card the same size as the source," Simon mentions. "You can copy to a larger card if you're running out of space on your existing one, or even to a smaller card (as long as it has enough space to store all your files – the program will warn you if there isn't enough space)."

Programming tools

The new version of Raspbian also comes with a new GPIO library, pigpio, which provides two benefits. First, it unifies access to the GPIO pins from Python, C, and other languages. Secondly, as a result of this, you won't need to sudo programs that use the GPIO pins. This includes Scratch.

The popular text editor/light IDE Geany has also been added to Raspbian. It allows for programming in multiple languages; if you're more of a seasoned coder, you might get on better with this than some of the other tools included in Raspbian.

There's your usual series of minor application updates and UI tweaks, as well. It's a great update and you can read up on the full rundown of changes on the release blog: **magpi.cc/1Va2aei**

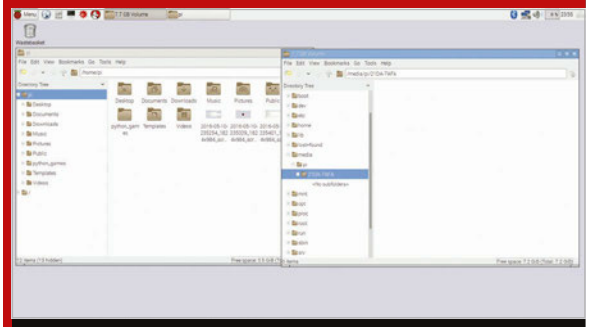
UPDATE NOW

There are two main ways you can get the new update. The quickest way, if you're already running a copy of Raspbian Jessie, is to open the terminal and use the following commands:

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get install piclone geany usb-modeswitch pi-bluetooth
sudo apt-get install python-pigpio python3-pigpio
```

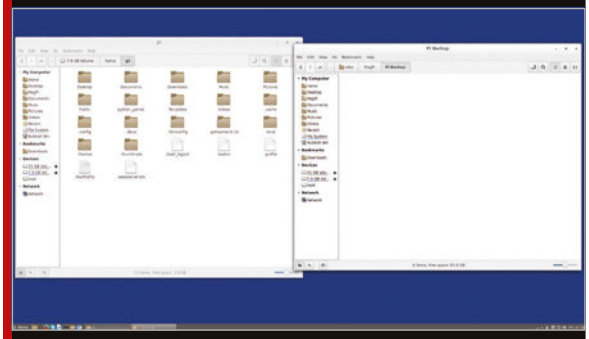
You can also reinstall Raspbian to the card. If you have the time and can easily transfer anything you're working on with it, then we recommend doing this. Find instructions here: **magpi.cc/1MYYTMo**

OTHER WAYS TO BACK UP



Transfer files to USB

A very quick and simple way to perform a backup is literally just to make a copy of any important files to a USB stick plugged into the Pi. This won't remember any settings or extra software you have installed, though.



Transfer files to computer

Similar to USB; turn off your Raspberry Pi and plug your SD card into another computer. From there, you can copy the files onto the other system without much trouble, as well as more easily being able to copy any settings or profile files.

Card copy

In the same way you can write an image to an SD card, you can do the reverse and copy an SD card to create an image. This is similar to the SD Card Copier utility. However, you can save it to anywhere on your computer, which likely has more space than a USB stick.

FLOODING WORKSHOP

Cornish students watch rising sea levels sink their school using a Raspberry Pi supercomputer

GRAPPOLO

Photo credit: Matt Jessop

Students used Grappolo (**magpi.cc/1TUNHxi**), a supercomputer cluster made from three Raspberry Pi 2s. Grappolo is used to teach cluster computing using a grid engine. The micro cluster computer replicates the functioning of the biggest cluster computer facility in South West England. Grappolo is similar to a Raspberry Pi cluster developed at Southampton University, but aimed towards teaching big data processing for geographic information systems methods (rather than raw computation). It costs around £140 to build and is a portable and perfect replica of an operating system running on an actual high-performance cluster computer.

Cornish students spent the day modelling climate change on a Raspberry Pi supercomputer at their annual Environment & Sustainability Day.

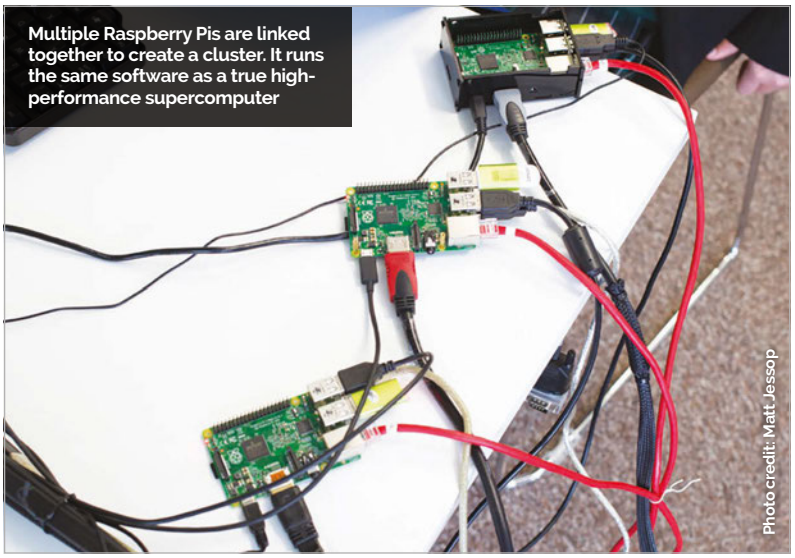
The University of Exeter event enabled students to learn Big Geo Data processing methods. Students used Grappolo, a cluster computer built by Spatial Ecology (**spatial-ecology.net**) to model rising sea levels.

"This passionate community of young future scientists met researchers and took part in a range of activities and workshops," says Dr Andrew Cowley, computing development officer at the University of Exeter. "The aim was to identify the single greatest challenge the local environment faces."

Andrew ran the Flooding Risk Workshop along with Dr Stefano Casalegno, associate research fellow at the University of Exeter.

"We investigated probability and risk," says Andrew, "and modelled impacts of sea level rise under climate change scenarios using the open-source software QGIS (a geographic information system). Students used Grappolo and QGIS software to model how much sea levels would have to rise before their school becomes submerged."

The Raspberry Pi cluster "simulates the functions of the biggest cluster computer facility in the South West," says Andrew. "Grappolo is similar to another Raspberry Pi cluster developed at Southampton University, but



Multiple Raspberry Pis are linked together to create a cluster. It runs the same software as a true high-performance supercomputer

Photo credit: Matt Jessop


Photo credit: Matt Jessop

Cornish students modelled the impact of sea level rise under climate change scenarios

is aimed towards teaching and demonstrating 'big geo data' processing methods rather than raw computation.

"Within the workshops, students visualised impacts of global sea-level rise using a 1km resolution digital elevation model (DEM) derived from the NASA's Shuttle Radar Topography Mission (SRTM)."

SRTM data was captured by the Space Shuttle Endeavour in 2000, and released globally in 2014. The radar data was used to create the first near-global set of land elevations and is now available from the US Geological Survey website (**magpi.cc/1TEyEIj**).

"The public availability of enhanced global SRTM topographic data will greatly benefit international efforts to understand better natural processes that shape our planet, prepare for and respond to natural hazards, and anticipate and prepare for the impacts of global change," commented NASA chief scientist Ellen Stofan. "NASA is proud to have played a critical role in creating these data that will benefit society through open data sharing."

Using a small, Raspberry Pi-powered supercomputer to crunch NASA data sounds cool, but it was locally sourced data that captured the students' imagination. "What pupils enjoyed most," says Andrew, "was the use of a local digital elevation model (DEM) from the Tellus project [**www.tellusgb.ac.uk**]."

Tellus South West is a project funded by the Natural Environment Research Council (NERC). They used an airborne lidar survey to create maps and data. According to Tellus: "The maps and data produced show the soils, rocks, landscape, and ecology of Devon and Cornwall in unprecedented depth and detail, and augment existing data to provide the two counties with among the best and most comprehensive environmental datasets anywhere in the world."

This incredibly accurate data of the surrounding area enabled students from Cornwall to simulate sea-level rise in Cornwall, to flood their schools and other places familiar to them.

"The most rewarding part of the day for Stefano and for me was showing the students about the Raspberry Pi, Linux, and FOSS doing great things together," Andrew tells us. "It was also great that so many of them asked further questions about the Pi, the Grappolo, high-performance computing, and also environmental issues facing Cornwall."

Students elected the Flooding Risk workshops, run by Andrew and Stefano, as the winner of the Environment & Sustainability Day event.

BIG GEO DATA


Photo credit: NASA

Students used two sets of data in the Flooding Risk workshop.

The first is data collected from NASA's Shuttle Radar Topography Mission (**magpi.cc/1Yqd3aL**). SRTM flew aboard the Space Shuttle Endeavour in February 2000, mapping Earth's topography between 56 degrees south and 60 degrees north of the equator. During the 11-day mission, SRTM used an imaging radar to map the surface of Earth numerous times from different perspectives.

Local data of Cornwall from the Tellus Project (**tellusgb.ac.uk**) was also used. This local geophysical survey measured traces of natural magnetism and radioactivity to make maps of the properties of the rocks, soils, and fluids under the skin of the Earth.


Photo credit: Matt Jessop

The Grappolo is a small Raspberry Pi supercomputer used to teach cluster computing using a grid engine

The Astro Pi units have been gathering data aboard the ISS and have returned their sensor readings to Earth

ASTRO PI RESULTS ARE IN!

Get your hands on the sensor data collected from the Astro Pis on the ISS and apply for a special schools conference with Principia

Data from the two Astro Pi units, orbiting the Earth aboard the ISS (International Space Station), is now publicly available. These special Raspberry Pis, nicknamed Ed and Izzy, blasted into space along with UK astronaut Tim Peake in December 2015.

Part of their payload contained seven winning programs, created by school-age students. After these student experiments had completed, the Astro Pis set about a long-term ISS environmental monitoring program.

This mission has produced data files chock-full of ISS sensor readings. These files have now been made available to the public, allowing budding scientists to analyse the data collected.

Ed and Izzy have returned the captured data to Earth, in the form of CSV (comma-separated value) files. The data is ready for download and analysis by the Raspberry Pi community.

The sensor readings can be downloaded directly from the Raspberry Pi website; the data is included in the Flight Data Learning Resource (**magpi.cc/1ZNxUVE**). This resource walks students through the sensor readings and helps them understand the data.

Jamie Bailey, CEO and founder of Initial State, has already taken the Astro Pi data and placed it into a web visualisation. "We took the CSV data collected from the ISS, uploaded it to Initial State's tools, and shared each dataset publicly for anyone to use," says James. "You can also bring data

into Initial State's data analytics tools through file uploads. The data delivered from the ISS is in CSV, a supported file format."

You can view the Astro Pi data visualisation in your browser on the Initial State blog (**magpi.cc/1ZNyoMY**).

> "Ed and Izzy have returned the captured data to Earth, in the form of CSV files"

Holding an Astro Pi

To celebrate Tim Peake's return from his mission aboard the ISS, the UK Space Agency is inviting young adults to attend two special Principia School Conference events in November (**magpi.cc/25b6eRD** and **magpi.cc/25b63WF**). Attendees will also be able to view some flight-equivalent Astro Pi units. These are the same devices as Ed and Izzy, the augmented Raspberry Pis that have been collecting sensor measurements aboard the ISS.

Dave Honess, the education resource engineer in charge of the Astro Pi project, will be at both conferences with a selection of the incredibly rare Raspberry Pi devices. Only eight flight-equivalent Astro Pis were ever made, constructed from aerospace-grade aluminium, and two of these went on to become Ed and Izzy aboard the ISS. "They're kind of like holy artefacts to Raspberry Pi fans," says Dave.

Students presenting at the Principia School Conference will be able to use the flight-equivalent Astro Pi devices as part of their presentation.

ASTRO PI DATA DOWNLOAD

Data from Ed and Izzy, the Astro Pi units aboard the ISS, can now be downloaded in the standard CSV (comma-separated value) format. Three separate files are available.

The first two sensor readings were collected in the Columbus Module, and the third in the Node 2/ Unity module.

Unit	Location on ISS	Length
Astro Pi Vis (Ed)	Columbus	2 weeks
Astro Pi Vis (Ed)	Columbus	4 weeks
Astro Pi IR (Izzy)	Node 2	2 weeks

You can download the data directly from the Astro Pi Flight Data Analysis worksheet on the Raspberry Pi website (**magpi.cc/1ZNxUVE**). This sheet also explains how to understand and use the data.

You can work directly with the data, using Python or a program like Mathematica or MATLAB (both of which are free on the Raspberry Pi).

If you don't want to produce your own graphs, then James Bailey from Initial State has already created some visualisations that you can view online. See Jamie's blog at Initial State for more information (**magpi.cc/1ZNyoMY**).

Tim's mission aboard the ISS has been extended by two weeks, giving him more time with Ed and Izzy

Tim has inspired UK schoolchildren with his mission aboard the ISS

"This conference isn't just for the winners of the various Principia competitions: it's for anyone who's been affected by Tim's mission. I bet there are some really interesting stories out there," says Dave. "It's going to be brilliant!"

The Principia Schools Conference gives students the chance to demonstrate a science project to a panel of space experts, possibly including Tim himself.

Welcome home, Tim

Tim has received a two-week extension to his stay aboard the ISS. The ESA (European Space Agency) has decided to keep the ISS operating at full capacity with six astronauts.

He is now due to land in the steppe of Kazakhstan on 18 June with two fellow astronauts. Tim will return to the European Astronaut Centre in Cologne, Germany, for check-ups and research into how humans adapt to living in space.

His schedule when he returns to Earth will be extremely busy, but Tim hopes to be at the Principia School events and wants to meet people at the conferences. Everybody is working hard to achieve this goal.

Tim has mixed thoughts about returning: "Although I am looking forward to being back on Earth and seeing friends and family again, each day spent living in space is a huge privilege and there is much work to do on the Station.

"This extension will keep the Station at a full crew of six for several days longer, enabling us to accomplish more scientific research.

"And, of course, I get to enjoy the beautiful view of planet Earth for a little while longer!"

Each crew member flies as a trio to the ISS and back in a Soyuz spacecraft. About every three months, a crew returns to Earth shortly before a new one arrives, often leaving a few days when only three astronauts look after the Station.

Izzy, our Astro Pi unit with an IR sensor, is attached to a window of the ISS

All images courtesy of ESA/NASA

Joining the Conference

The Principia School Conference events will take place in November 2016, at universities in York and Portsmouth. Attendance at each event is free, and the UK Space Agency will offer travel bursaries to help with the cost of attending.

Places are limited and potential attendees will need to submit an application. "The judging panel would love to see reports of things such as scientific experiments, art projects, whole school projects, special events, and community outreach," explains the Principia website. "The applications should communicate what work has been done, what was learnt by doing the work, and why this should be shared with an audience at the conferences."

The Principia mission has a tremendous range of educational activities linked to it, one of which is the Astro Pi. The conferences will be attended by students from across all of the Principia missions.

There are up to 500 places available at both of the conferences; the teams and individuals who present the strongest applications will be invited to take part.

It would be great to see a strong Astro Pi contingent at both conferences, so we strongly encourage anyone who engaged with Astro Pi to apply. The deadline for applications is Thursday 15 September 2016 at 12 noon. See the box on the right for details on how to get started.

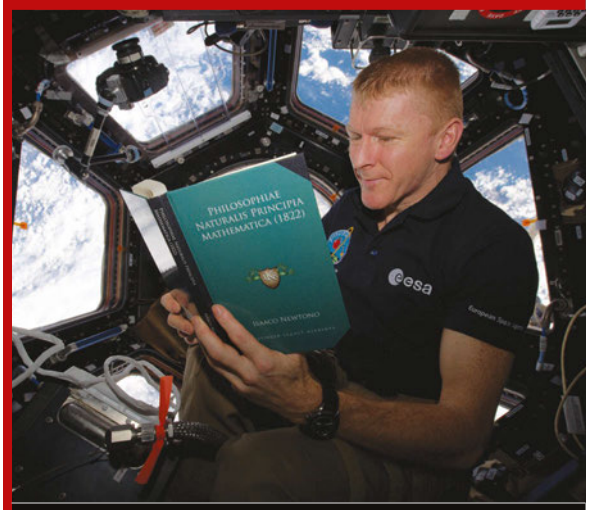
Applications to the conferences are not limited to previous winners of Principia or Astro Pi competitions, so anybody can enter. The UK Space Agency wants your entry to tell the story of your project, describe what you've learned, and what long-term effects the work has had on your school.

> "It would be great to see a strong Astro Pi contingent"

To illustrate the kind of applications Principia is hoping to see, it's worth mentioning the Astro Pi testimonial that maths teacher Gillian Greig, from The Priory School in Hitchin, wrote last year (**magpi.cc/1ZNxNcT**). Gillian's testimonial is a great example of the kind of story the organisers will enjoy seeing.

The best applications will become presentations, viewed by a team of space experts – hopefully including Tim Peake.

APPLY FOR THE SPACE CONFERENCE



Individuals and groups of young people (of school age) can apply to the Principia Schools Conference events. There is no limit on the size of the group that can apply, but a maximum of six children along with two adults will be invited to attend.

The two conferences are:

University of Portsmouth	Wed 2 Nov 2016
University of York	Sat 5 Nov 2016

Overseas applications are welcomed, though the judges will be giving first priority to applications from the United Kingdom, Channel Islands, and the Isle of Man. International applications of high calibre may be successful.

The deadline for applications to both events is Thursday 15 September 2016 at 12 noon.

To learn more about the Principia School Conference and how to apply, visit: **magpi.cc/1ZNy9A2**

pi-top

Worldwide shipping available in green or grey at **www.pi-top.com**

LEARN
PLAY
CREATE

pi-top

- 10 Hour Battery Life
- 13.3" HD Screen
- Modular Components

$299 excluding VAT

pi-topCEED

- Adjustable Viewing Angles
- 14" HD Screen
- Modular Components

$99 excluding VAT

pi-top | Check out our education game **CEED**universe at pi-top.com/learn
Stay up to date with our latest news by following our social media.

LEARN

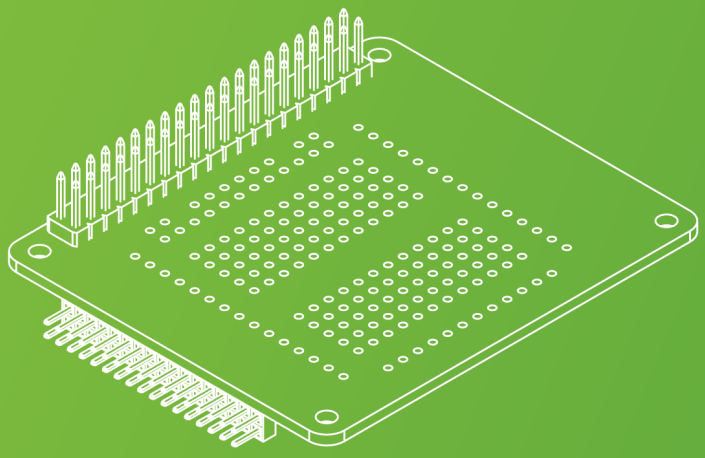
PLAY

CREATE

---

pi-topPROTO

pi-topPROTO is an Add-on Board for your pi-top and **pi-top**CEED. It allows you to prototype circuit boards, create amazing IoT devices and more. Slide the **pi-top**PROTO into the modular rail to build any functionality into your device.



A Beginner's GUIDE TO Electronics ON THE Raspberry Pi

onics

Use the GPIO pins, some electronic components, and code to do cool stuff with the Raspberry Pi

Every month in *The MagPi*, we have projects filling the pages that hook up directly to Raspberry Pi using the GPIO pins running down the side of the board. They're pretty easy to use and, with some patience, you can figure out roughly how they work by reading all the different tutorials.

With the creation of the GPIO Zero Python library, it's never been easier to start coding electronics. With this in mind, we decided to remove the step between idea and project by starting you off with a proper guide to getting electronics working with your Raspberry Pi.

In this feature, we're going to explain to you exactly how the electronic components work and how you wire them up, along with a bit of a briefing on how to program them with GPIO Zero, before showing you some excellent examples of the kind of things you can do with Pi electronics when you know how.

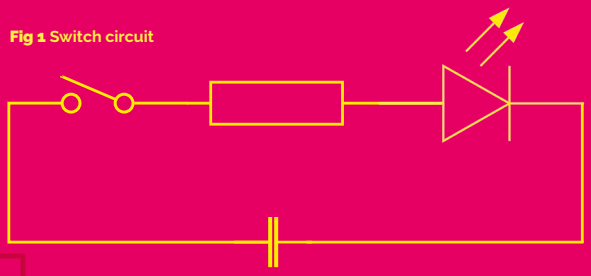
All the code in this tutorial is Python 3

Getting started with RASPBERRY PI Electronics

Reading circuit diagrams and wiring them up to your Raspberry Pi

The Raspberry Pi is great for learning computing. Whether that's coding or more advanced uses of a computer, the Raspberry Pi has lots of tools to aid you in learning about them. It's also very good at physical computing, which in this context means programming and interacting with the real world through electronics. In simple terms, physical computing with the Pi is something like programming it to turn on an LED, an electronic component in an electronic circuit.

Fig 1 Switch circuit



The electronic circuits are the physical part of a physical computing project connected to the Raspberry Pi. These circuits can be simple or very complex, and are made up of electronic components such as LEDs, buzzers, buttons, resistors, capacitors, and even integrated circuit (IC) chips.

At its simplest, an electronic circuit lets you route electricity to certain components in a specific order, from the positive end of a circuit to the negative (or ground) end. Think of a light in your house: the electricity passes through it, so it lights up. You can add a switch that breaks the circuit so it only lights up when you press the switch. That is an electronic circuit.

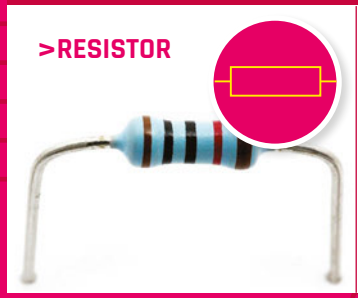
Reading circuit diagrams

Building a circuit can be easy if you know what you're doing, but if you're making a new circuit or are new to electronics in general, you'll most likely have to refer to a circuit diagram. This is a common way you'll see a circuit represented, and these diagrams are much easier to read and understand than a photo of a circuit. However, components are represented with symbols which you'll need to learn or look up.

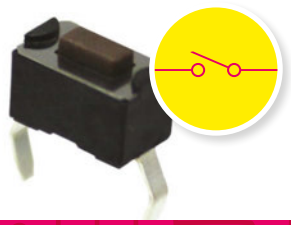
Fig 1 is an example of the light circuit we talked about before. Here we have a power source (a battery in this circuit), a switch, a resistor, and an LED. The lines represent how the circuits are connected together,

COMMON COMPONENT SYMBOLS

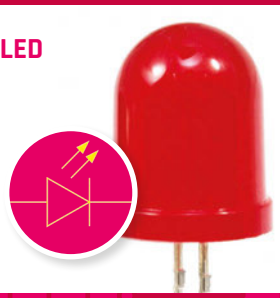
>RESISTOR



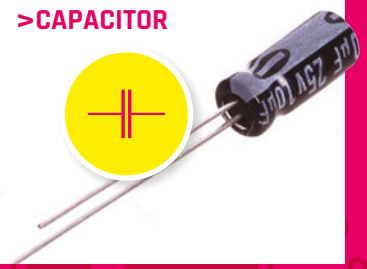
>BUTTON/SWITCH



>LED

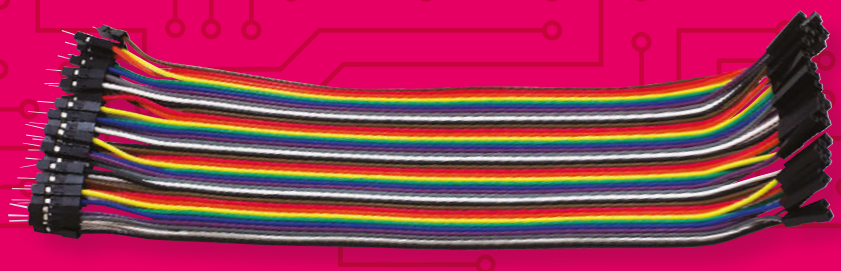


>CAPACITOR



Credit: oomlout

Jumper wires are useful for connecting to the GPIO pins



either via wire or other means. Some components can be used any way round, such as the resistor or switch. However, others have a specific orientation, such as the LED. Diodes only let electricity flow from positive to negative; luckily real-life LEDs have markers such as longer legs or a flat edge to indicate which side is positive, making them easier to wire up.

The Raspberry Pi and electronic circuits

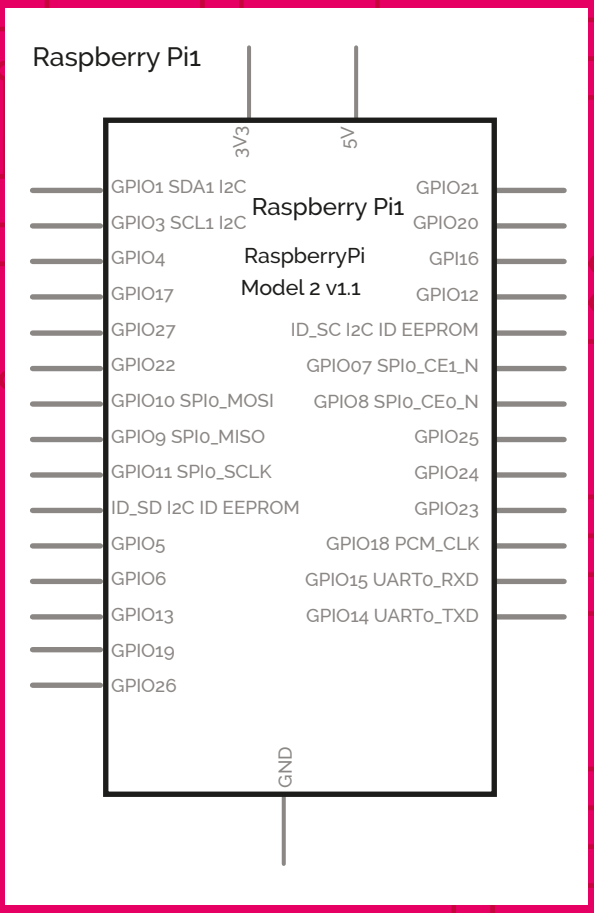
Making a Raspberry Pi part of the circuit is quite easy. At its most basic, it can provide power to a circuit, as well as a negative or ground end through the GPIO pins. Some pins are specifically always powered, mostly by 3.3V, and always go to ground. Most of them can be programmed to create or recognise a HIGH or LOW signal, though; in the case of the Raspberry Pi, a HIGH signal is 3.3V and a LOW signal is ground or 0V.

In an LED example, you can wire up an LED directly to a 3.3V pin and a ground pin and it will turn on. If you instead put the positive end of the LED onto a programmable GPIO pin, you can have it turn on by making that pin go to HIGH.

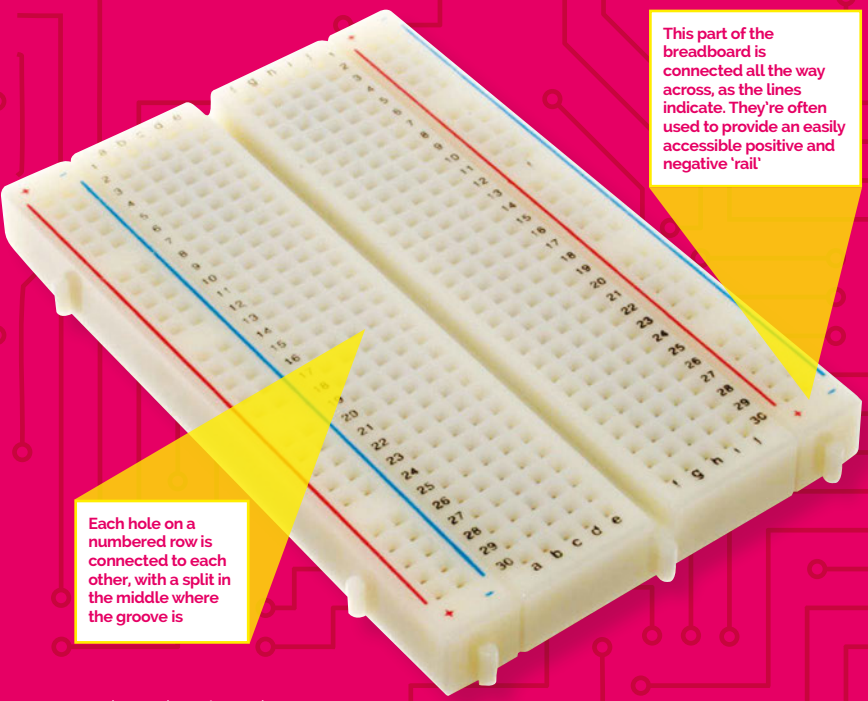
Wiring up a circuit to a Raspberry Pi is fairly simple. To create the physical circuits throughout this tutorial, we're using prototyping breadboards. These allow you to insert components and wires to connect them all together, without having to fix them permanently. You can modify and completely reuse your components because of this.

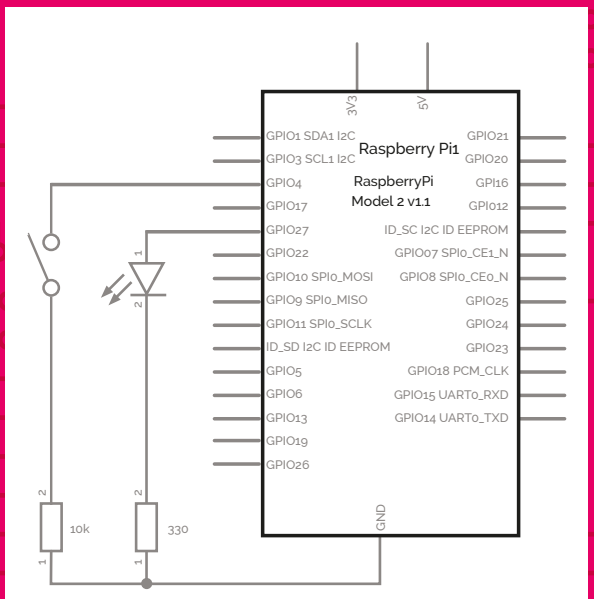
Making a light switch circuit

For our first circuit, we'll create a light switch on some breadboard with the Raspberry Pi. Below, you can see the circuit diagram of what you need to build; look over the page for an illustrated version to refer to when building the circuit. The resistors are required to limit the current on the LED and 'pull up' the circuit on the button. This makes the GPIO go to LOW when the button is pushed.



The rundown of which GPIO pin is which is explained on the next page





A Beginner's GUIDE TO GPIO Zero

GPIO ZERO

read all the docs here:
magpi.cc/1Od1xtB

What GPIO Zero is and how you can use it to program electronics connected to your Raspberry Pi

Once the components are all hooked up to your Raspberry Pi, you need to be able to control them. The Raspberry Pi is set up to allow you to program it with the Python language. This has also been made simpler recently with the addition of GPIO Zero.

GPIO Zero was created to simplify the process of physical computing, helping new coders to learn. It's a Python library which builds upon the existing GPIO libraries **RPi.GPIO**, **RPIO**, and **pigpio**. However, while those libraries provide an interface to the GPIO pins themselves, GPIO Zero sits above them and provides a way to interface to the devices that you connect to those pins.

This change simplifies thinking about physical computing. Consider wiring a simple push button to GPIO 4 and ground pins. In order to react to this button, we need to know that the pin should be configured with a pull up resistor, and that the pin state when the button is pushed will be 0. Here's what this would look like in the classic RPi.GPIO library:

```
from RPi import GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(4, GPIO.IN,
GPIO.PUD_UP)
while GPIO.input(4):
    pass
print("Button pushed!")
```

To complete beginners, there's quite a lot going on there, which gets in the way of actually experimenting with it and even teaching the simple logic required. Here's the equivalent code in GPIO Zero:

```
from gpiozero import
Button

btn = Button(4)
while not btn.is_pressed:
    pass
print("Button pushed!")
```

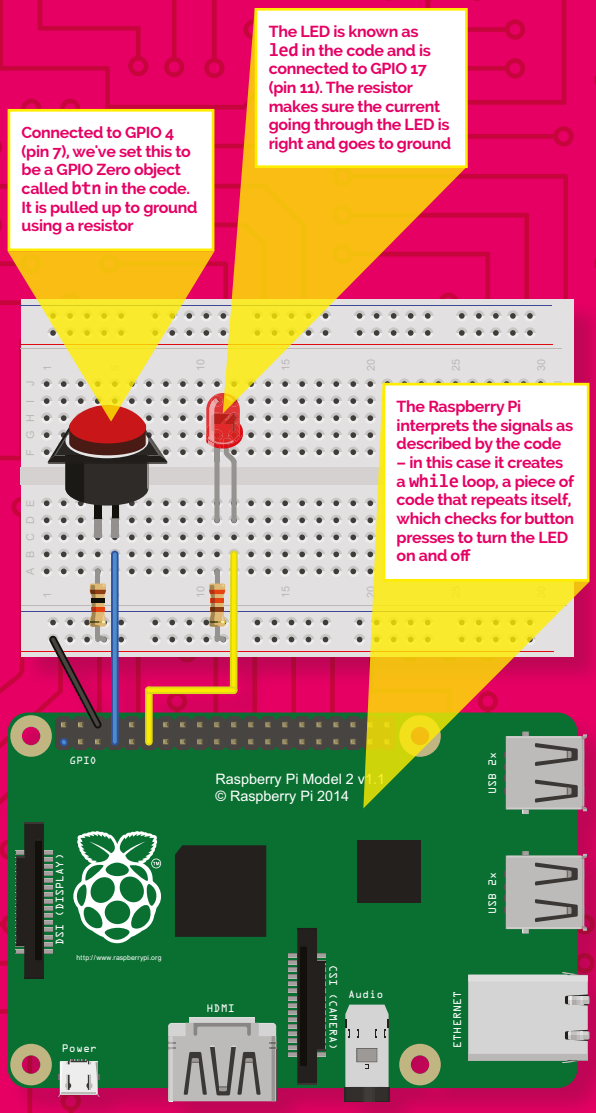
The boilerplate, stuff you have to blindly enter without understanding why you're entering it, is reduced to the barest minimum that we can manage. The name 'GPIO Zero' derives from

GPIO NUMBERS

40-pin GPIO header key for Raspberry Pi 3, 2, B+, and A+

3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GNV	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPI19	35	36	GPIO16
GPI26	37	38	GPIO20
GND	39	40	GPIO21

GPIO Zero uses the BCM GPIO numbering rather than the pin number – use this handy table to remember which is which



this 'zero boilerplate' philosophy, which was first espoused by Daniel Pope's Pygame Zero library.

The logic is also straightforward, with no curious inversion of the input value.

Ways to turn on an LED

GPIO Zero also tries to provide several means of attacking problems, as some coders may find particular methods easier to get their head around. Here, we've extended our script to control an LED attached to GPIO 17 with our button. Instead of looping to wait for the button, we use some of the available `wait_for` methods:

```
from gpiozero import
Button, LED

btn = Button(4)
led = LED(17)
while True:
    btn.wait_for_press()
    led.on()
    btn.wait_for_release()
    led.off()
```

You can test this code with the circuit we created on the previous page. This method works fine for one button, but starts to get tricky when we want to use two. Say we've got buttons wired to GPIO pins 3 and 4, and we want the LED to light when either is pushed. This is quite hard with the wait methods above, so it makes sense to switch to 'event-driven' programming:

```
from gpiozero import
Button, LED
from signal import pause

btn1 = Button(3)
btn2 = Button(4)
led = LED(17)
btn1.when_pressed = led.on
btn1.when_released = led.
off
btn2.when_pressed = led.on
btn2.when_released = led.
off
pause()
```

Simple! But there's a subtle bug. Hold down one button, then press and release the other; the LED goes off despite the fact a button is still being held; think about how the events are being fired to see why!

Fortunately, GPIO Zero provides another programming style which solves this by allowing us to tie the state of the LED to the states of the buttons:

```
from gpiozero import
Button, LED
from gpiozero.tools import
any_values
from signal import pause

btn1 = Button(3)
btn2 = Button(4)
led = LED(17)
led.source = any_
values(btn1.values, btn2.
values)
pause()
```

This uses the new `tools` module from GPIO Zero 1.2, which is designed to work with the `source` and `values` properties used above. Basically, GPIO Zero is excellent for getting started with physical computing, yet is complex enough that you can do more advanced things with it than just wait for a button press.

WHAT'S NEW IN GPIO ZERO?

GPIO Zero can use buttons, LEDs, buzzers, and lots of other components. The library is ever expanding.

>SERIAL PERIPHERAL INTERFACE

Released in 1.2.0, there is now an SPI implementation for specific compatible devices to talk to the Pi. This allows for analogue inputs, analogue-to-digital converters, and other pretty advanced stuff, but it makes using them much more simple.

>HOLD EVENTS

These are variables in something like button code that allow you to set a length of time a button should be pressed before being recognised as a press. This can be useful if your button is very twitchy in a project you're using.

>SOURCE TOOLS

The tools library for the `source` and `values` properties enables you to tweak and play with the way GPIO Zero handles specific components and functions. We won't be covering them this issue, but they're important for advanced uses.

PHIL KING

When not subediting *The MagPi* and writing articles, Phil loves to work on Pi projects, including his new two-wheeled robot.

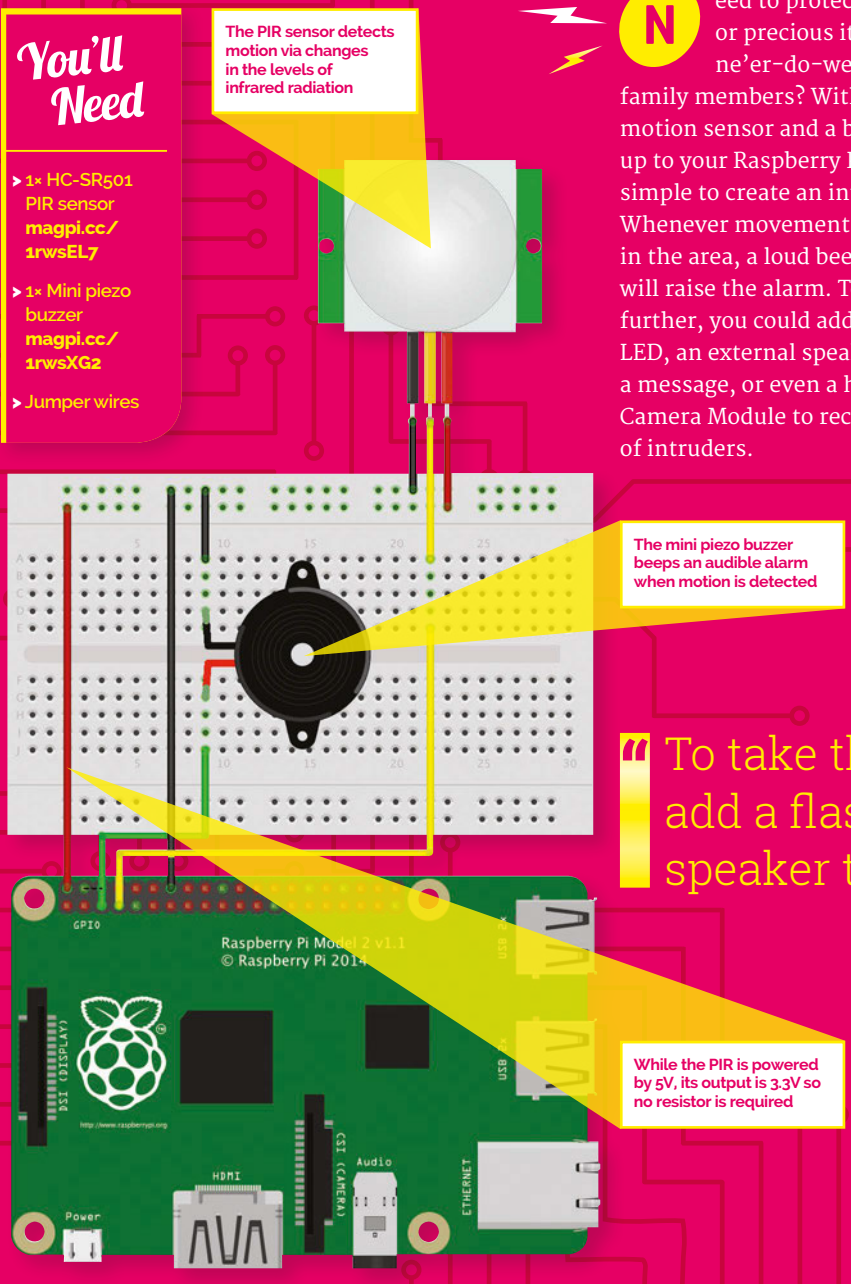
@philking68

Motion-sensing alarm

Stop people from sneaking up on your stuff by creating a motion-sensing alarm that buzzes when it detects someone

You'll Need

- 1× HC-SR501 PIR sensor **magpi.cc/1rwsEL7**
- 1× Mini piezo buzzer **magpi.cc/1rwsXG2**
- Jumper wires



Need to protect your room or precious items from ne'er-do-wells or nosy family members? With just a PIR motion sensor and a buzzer wired up to your Raspberry Pi, it's very simple to create an intruder alert. Whenever movement is detected in the area, a loud beeping noise will raise the alarm. To take things further, you could add a flashing LED, an external speaker to play a message, or even a hidden Camera Module to record footage of intruders.

>STEP-01

Attach PIR motion sensor

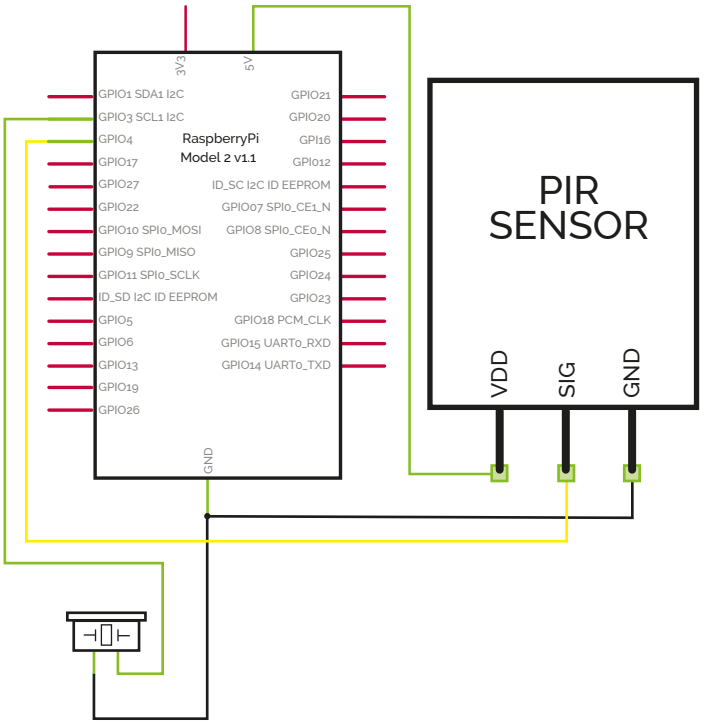
First, we need to wire the PIR (passive infrared) sensor to the Pi. While it could be hooked to the GPIO pins directly using female-to-female jumper wires, we're doing it via a breadboard. The sensor has three pins: VCC (voltage supply), OUT (output), and GND (ground). Use female-to-male jumpers to connect VCC to the '+' rail of the breadboard, and GND to the '–' rail. Wire OUT to a numbered row, then use another jumper to connect that row to GPIO pin 4.

>STEP-02

Wire up the buzzer

Next, we'll hook up the mini buzzer. Place its two legs across the central groove in the breadboard. Note that the longer leg is the positive pin; wire its numbered row to GPIO pin 3 on the Pi to connect it. Wire the row of the buzzer's shorter leg to the '–' rail, then connect the latter to a GND pin on the Pi. Finally, connect the '+' rail to the Pi's 5V pin to power the PIR sensor.

> "To take things further, you could add a flashing LED or an external speaker to play a message"



PIRalarm.py

Language
>PYTHON 3

DOWNLOAD:
magpi.cc/1Ynsd0p

```
from gpiozero import
MotionSensor, Buzzer
import time

pir = MotionSensor(4)
bz = Buzzer(3)

print("Waiting for PIR to settle")
pir.wait_for_no_motion()

while True:
    print("Ready")
    pir.wait_for_motion()
    print("Motion detected!")
    bz.beep(0.5, 0.25, 8)
    time.sleep(3)
```

>STEP-03
Work on the code

At the start of the program, we import the handy **MotionSensor** and **Buzzer** modules from the GPIO Zero library, each of which contains numerous useful functions; we'll need a few of them for our intruder alarm. We also import the **time** library so that we can add a delay to the detection loop. Next, we assign the relevant GPIO pins for the PIR sensor and buzzer; we've used GPIO 4 and 3 respectively in this example, but you could use alternatives if you prefer.

>STEP-04
Setting things up

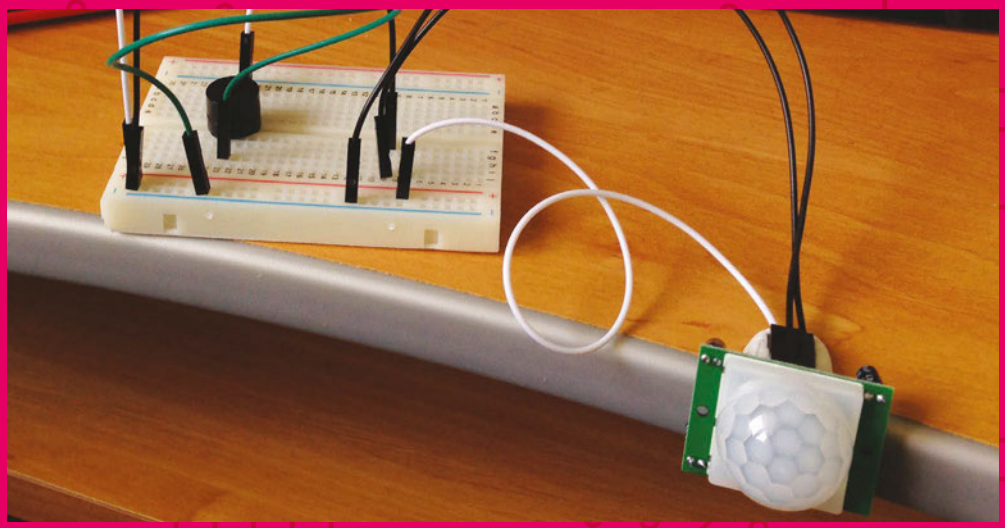
Before starting our motion detection **while** loop, we make use of the GPIO Zero library's **wait_for_no_motion** function to wait for the PIR to sense no motion. This gives you time to leave the area so that it doesn't immediately sense your presence and raise the alarm when you run the code! Once the PIR has sensed no motion in its field of view, it will print 'Ready' on the screen and the motion detection loop can then commence.

>STEP-05
Motion detection loop

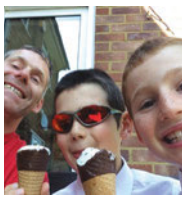
Using **while True:** means this is an infinite loop that will run continually, until you stop the program by clicking the 'X' icon of its window or pressing **CTRL+C** on the keyboard. Whenever motion is detected by the PIR sensor, we get the buzzer to beep repeatedly eight times: 0.5 seconds on, 0.25 seconds off, but you can alter the timings. We then use **time.sleep(3)** to wait 3 seconds before restarting the loop.

>STEP-06
Adjust the sensitivity

If you find that the alarm is going off too easily, or not at all, you may need to adjust the sensitivity of the PIR sensor. This is achieved by using a small screwdriver to adjust the plastic screw of the left potentiometer, labelled Sx; turn it anticlockwise to increase sensitivity. The other potentiometer, Tx, alters the length of time the signal is sent after detection; we found it best to turn it fully anticlockwise, for the shortest delay of 1 second.



The program uses an infinite loop to detect motion, making the buzzer beep whenever that occurs



THE HAYLER-GOODALLS

Ozzy, Jasper, and Richard are mentors at CoderDojo Ham and gave a talk at the Raspberry Pi birthday party about their Astro Pi adventures.

richardhayler.blogspot.co.uk / @rdhayler / coderdojoham.org

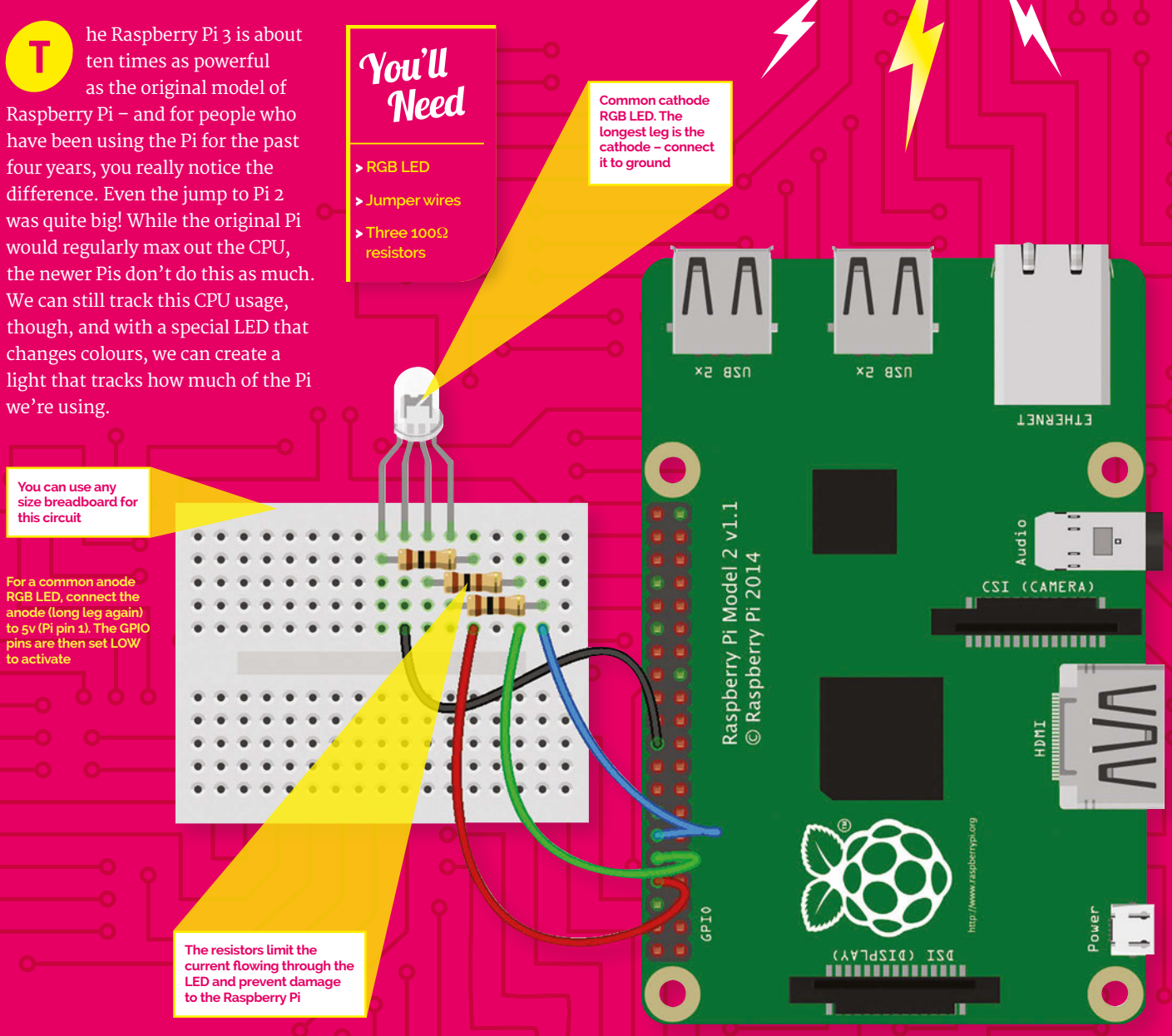
CPU CREATE A monitor

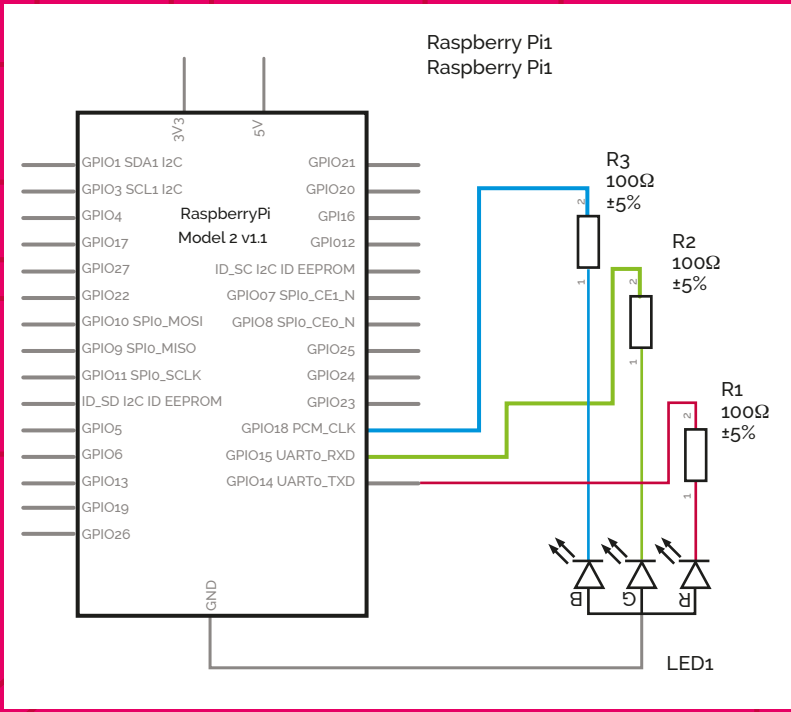
Learn how to use an RGB LED with GPIO Zero and have it light up depending on how much the CPU is being used

The Raspberry Pi 3 is about ten times as powerful as the original model of Raspberry Pi – and for people who have been using the Pi for the past four years, you really notice the difference. Even the jump to Pi 2 was quite big! While the original Pi would regularly max out the CPU, the newer Pis don't do this as much. We can still track this CPU usage, though, and with a special LED that changes colours, we can create a light that tracks how much of the Pi we're using.

You'll Need

- RGB LED
- Jumper wires
- Three 100Ω resistors





rgb_cpumon.py

Language
>PYTHON 3

DOWNLOAD:
magpi.cc/1sy4eC2

```
from gpiozero import
RGBLED
import psutil, colorsys,
time

myled = RGBLED(14,18,15)

while True:
    cpu = psutil.cpu_percent()
    r = cpu / 100.0
    g = (100 - cpu)/100.0
    b = 0
    myled.color = (r,g,b)
    time.sleep(0.1)
```

>STEP-01
Update your Pi

Update your Pi to the latest version of Raspbian and make sure you have the psutil libraries installed:

```
sudo pip3 install psutil--
upgrade
```

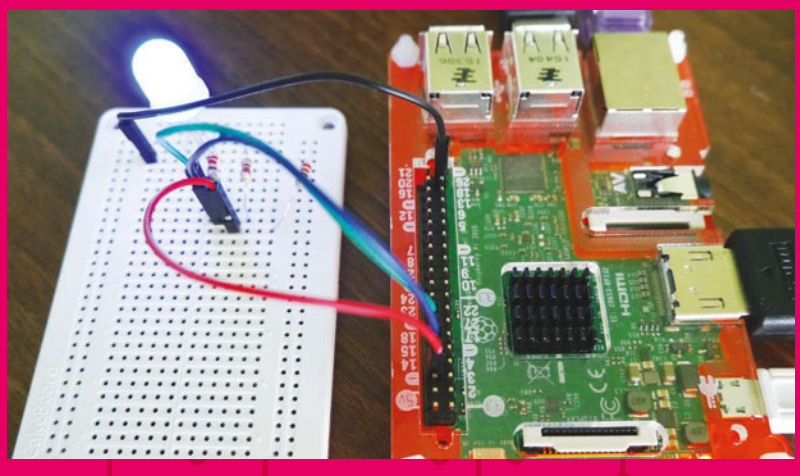
This will let us look up the CPU usage of the Raspberry Pi as a percentage number, which can then be used in our code.

>STEP-02
Select your RGB LED

Light-emitting diodes (LEDs) are cool. Literally. Unlike a normal incandescent bulb, which has a hot filament, LEDs produce light solely by the movement of electrons in a semiconductor material. An RGB LED has three single-colour LEDs combined in one package. By varying the brightness of each component, you can produce a range of colours just like mixing paint. There are two main types of RGB LEDs: common anode and common cathode. We're going to use common cathode.

>STEP-03
Connect up the RGB LED

LEDs need to be connected the right way round. For a common cathode RGB LED, you have a single ground wire and three anodes, one for each colour. To drive these from a Raspberry Pi, connect each anode to a GPIO pin via a current-limiting resistor. When one or more of these pins is set to HIGH (3.3V), the LED will light up the corresponding colour. Connect everything as shown in the diagrams.



The CPU monitor in action. Chunky 10mm LEDs make a big, bright indicator

>STEP-04
Run the code

Download or type up the code from the listing. The LED should light up: its colour will indicate how hard your Pi's CPU is working. Green means less busy, turning redder as the CPU becomes more heavily loaded. Start up some other applications to test it. If you have an original Model B, you'll probably find that just running *Minecraft* is enough to turn the LED red. If you have a Pi 3, you may need to start lots of things running in order to have any impact!

>STEP-05
Customise your project

The example code only uses the red and green components of the LED; the blue value is always set to zero. You could swap things around and create a different colour gradient (e.g. blue to red) or put together a fancy function that maps a percentage value onto all three colours. Have fun with the colours and maybe even have it take into account other resources...

ROB ZWETSLOOT

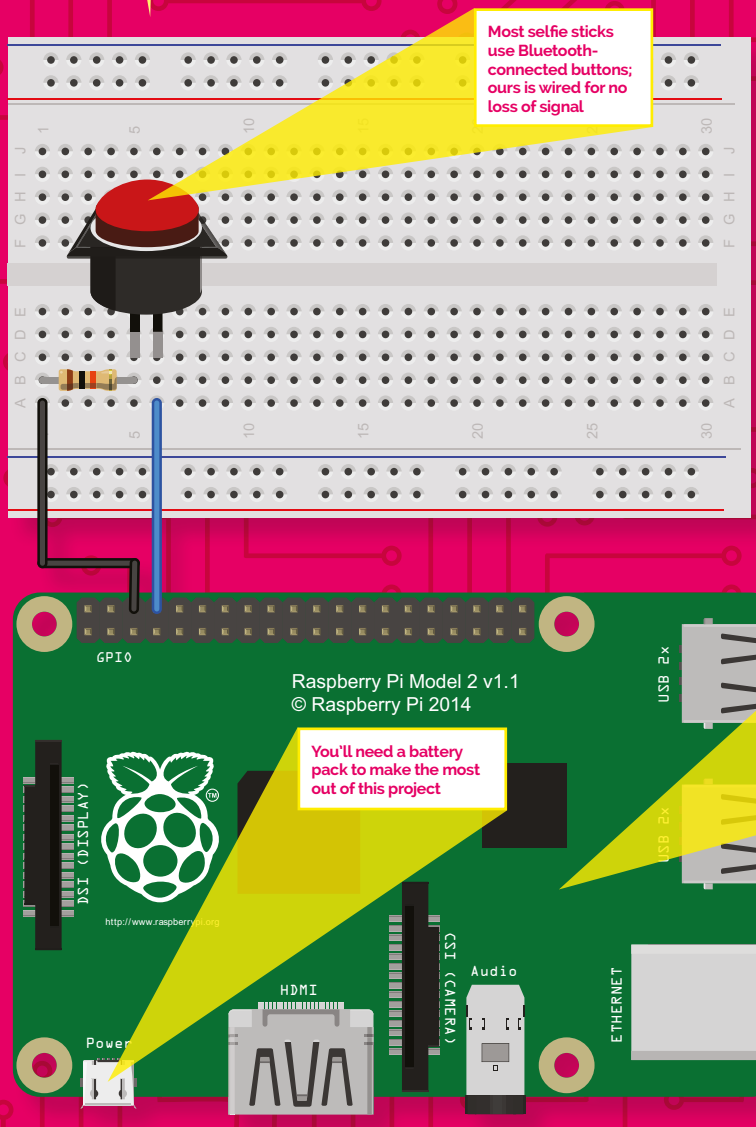
Tinkerer, sometime maker, other-times cosplayer, and all-the-time features editor of *The MagPi*.

magpi.cc

MAKE A Selfie stick camera

Use the Pi Camera Module, some GPIO Zero code, and a very long wire to create a selfie stick with a twist

A lot of people roll their eyes and complain about vanity when it comes to the art of the selfie, but we all know it's nothing like that. New outfit? New glasses? Eyeliner wings symmetrical today? Why not chronicle it? It's a great confidence boost. Taking what we've done so far with GPIO Zero, we're going to make use of a button with a Pi Camera and construct our own custom, Pi-powered selfie stick.



You'll Need

- A Raspberry Pi case with a camera mount
- Raspberry Pi Camera Module **magpi.cc/1UaFuuW**
- A push-button and some long wires
- A stick, slim metal pole, or anything to mount a Pi case on

>STEP-01

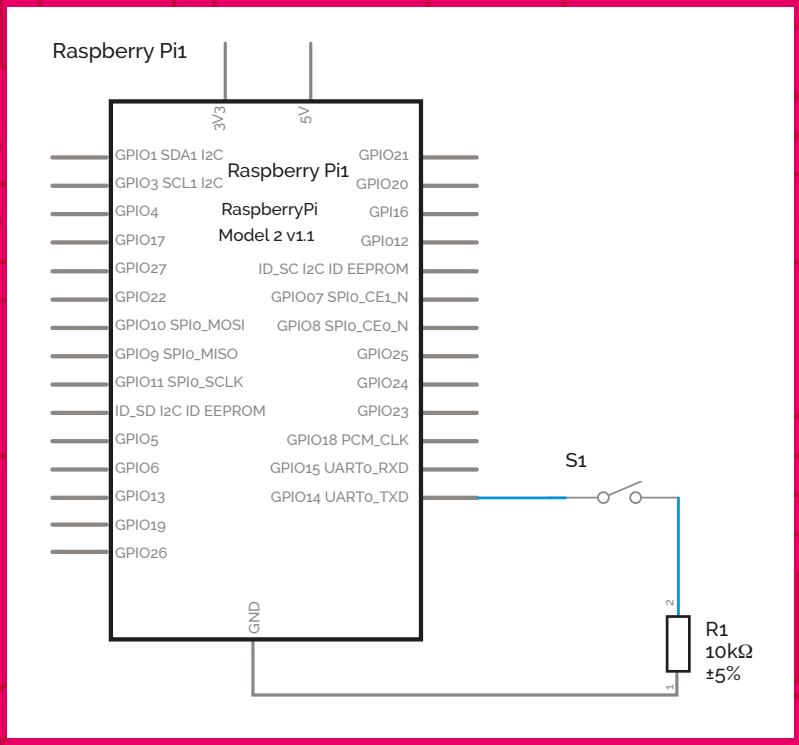
Add the camera

One thing we didn't cover in the intro to electronics was how to install the Raspberry Pi Camera Module onto the Pi. Make sure yours is off and then find the camera connector; it's the white rectangle between the HDMI and audio ports on the Pi 2 and 3. Lift it up gently and then place the Pi Camera ribbon cable in the slot with the silver connector facing the HDMI port. Push the slider back down.

>STEP-02

Camera software

To get the camera working on the Raspberry Pi and in Python, we



selfie.py

Language
>PYTHON 3

DOWNLOAD:
magpi.cc/Selfie-Camera

```
from gpiozero import
Button
from datetime import
datetime
import picamera
import time

btn = Button(14)
pc = picamera.PiCamera()

def picture():
    timestamp = datetime.now()
    pc.capture('pic'+str(
timestamp)+'.jpg')
    time.sleep(1)
while True:
    btn.wait_for_press
    picture()
```

need to do a couple of things. First, go to the Program menu and open the Raspberry Pi Configuration interface. Go to the Interfaces tab and enable the Camera if it's not been done already. Next, open a terminal window and install the picamera Python module with:

```
sudo apt-get install
python3-picamera
```

>STEP-03
Get constructing

The Pi needs to be with the camera, as it's tricky to get a long enough ribbon cable for the Camera Module. Attach the Pi in a case to one end of the stick with whatever means you see fit (glue, Blu-Tack, string, etc.) and then attach the button. We used the same jumper leads as in the electronics intro, but added a long length of wire between the end of them and the button we used.

>STEP-04
Add the code

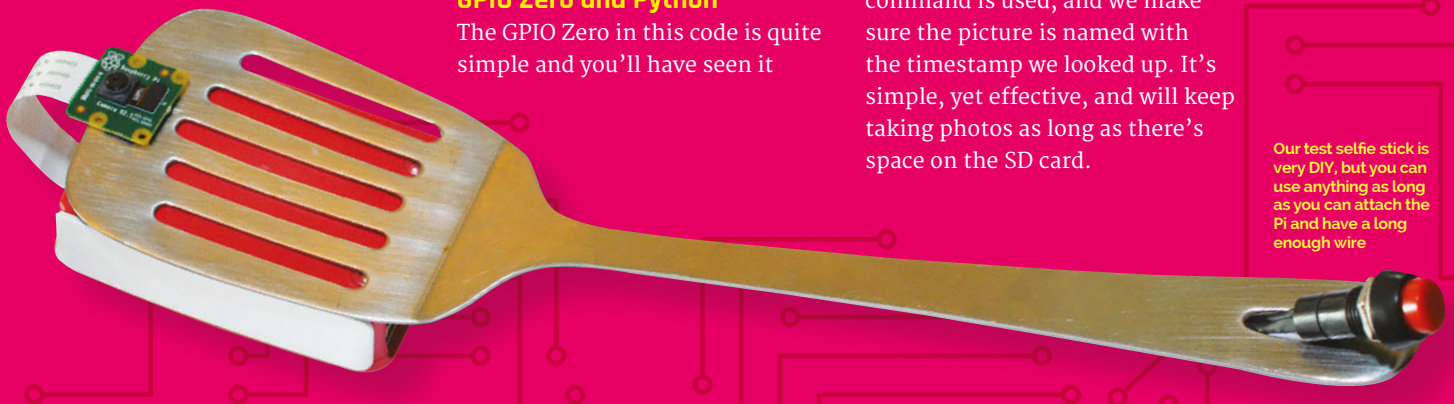
Download or type up the code listing. You can give it a quick test by running the script and then unplugging the HDMI and USB cables from the Pi. Pressing the button will start taking pictures, but you'll need to practise your aim so you can get yourself in. Plug it all back in and have a look at the code so we can explain what you're doing with it.

>STEP-05
GPIO Zero and Python

The GPIO Zero in this code is quite simple and you'll have seen it before in the previous pages. We set the button to GPIO 14 (pin 8) and all we use it for is to interrupt a **while** loop that keeps checking if the button is pressed. We also use **datetime** and **time** to create a timestamp on the picture and cause a slight pause in the code respectively. This makes it easier to organise the photos and not take too many pictures with a slip of the button.

>STEP-06
Picamera library

There are many ways to use this library, but we're doing the simplest. We've set **pc** to be the Pi Camera variable, much like how we set a GPIO Zero variable. When we want to take a picture the **capture** command is used, and we make sure the picture is named with the timestamp we looked up. It's simple, yet effective, and will keep taking photos as long as there's space on the SD card.



Our test selfie stick is very DIY, but you can use anything as long as you can attach the Pi and have a long enough wire

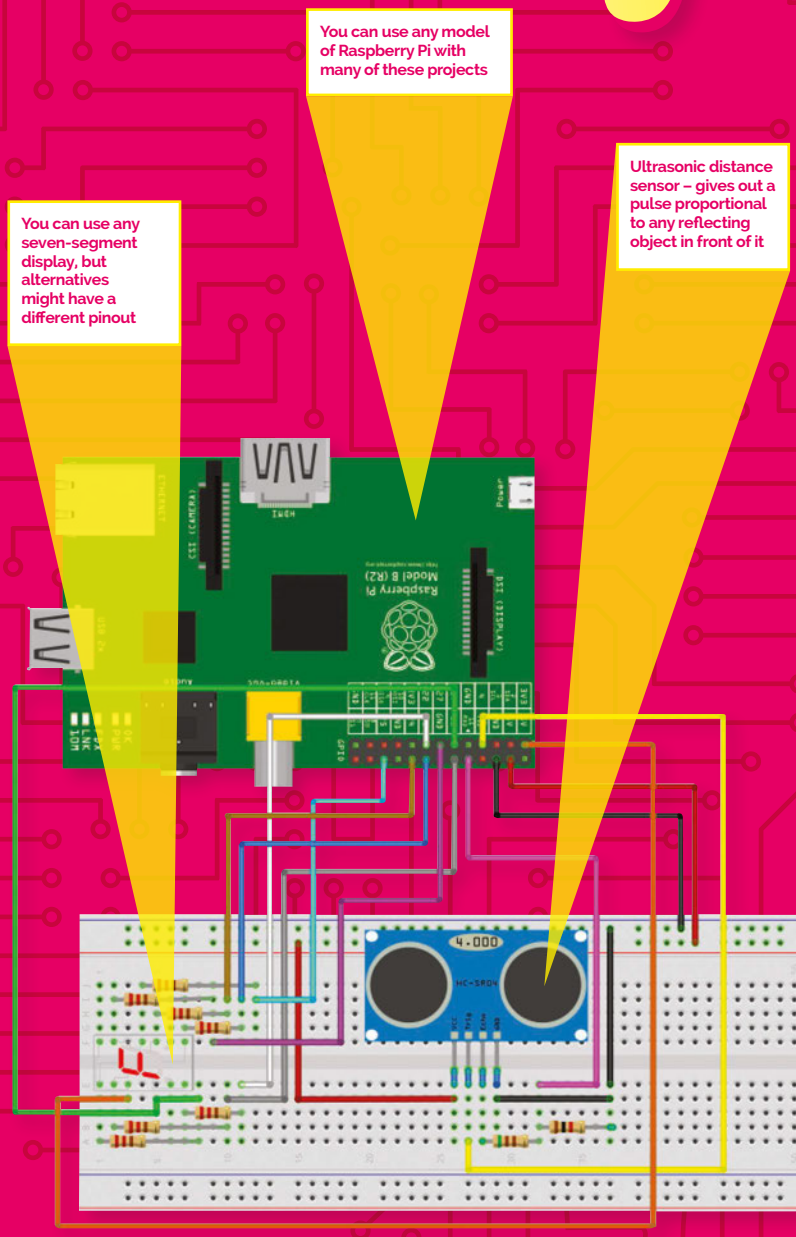
MIKE COOK

Veteran magazine author from the old days, writer of the Body Build series, and co-author of three Raspberry Pi books.

thebox.myzen.co.uk

Make a rangefinder

Link together an ultrasonic distance sensor and seven-segment display to measure distances



You'll Need

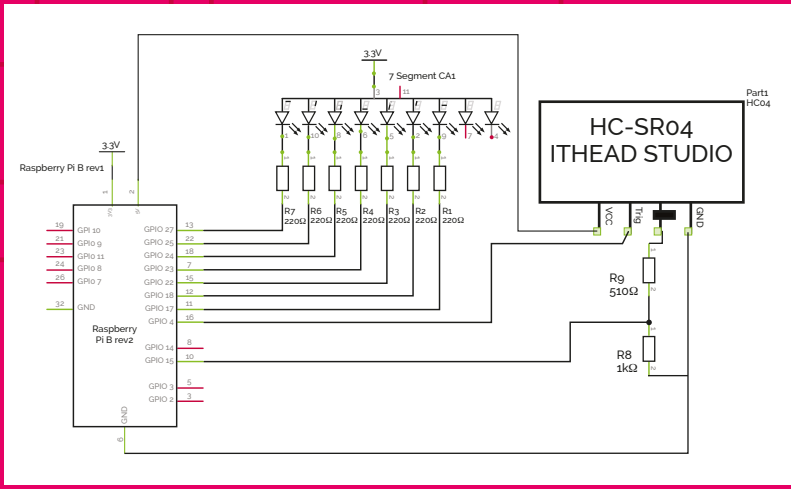
- HC-SR04 ultrasonic distance sensor **magpi.cc/1YnqNCZ**
- Broadcom 5082-7650 7-segment display **magpi.cc/1YnqQPl**
- 9× resistors (refer to circuit diagram for values) **magpi.cc/1YnqU1r**

The HC-SR04 ultrasonic distance sensor is a great favourite with Pi robot makers. It works by bouncing ultrasonic sound off an object and timing how long it takes for the echo to return. This time is then converted into a distance which can be displayed on a single-digit, seven-segment display. Here you can acquire the skills to handle inputs and outputs. You also get to use seven-segment displays, which are quite cool in a retro kind of way.

>STEP-01

Lighting the display

The seven-segment display is a collection of LEDs, with one LED corresponding to one of the segments. All the anodes (positive ends) are connected together. This should be connected to the 3V3 supply. The cathodes (negative ends) should be connected to a resistor to limit the LED current, and the other end of the resistor to a GPIO pin. To turn the LED on, all you have to do is set the GPIO output to be LOW (0V) and it will complete the circuit for the current to flow.



>STEP-02

Generating a seven-segment pattern

The display consists of four bars or segments that can be lit up. By choosing the segments to light up, you can display a number from 0 to 15, although you have to resort to letters (also known as hexadecimal numbers) for this. There are, in fact, 128 different patterns you can make, but most are meaningless. A list called **seg** defines what pins are connected to what segments, and another list called **segmentPattern** defines the LED pattern for each number.

>STEP-03

Displaying numbers

The display function sets up the segments to display any single-digit number passed to it. First, it sets all the segments to off, and then if the number is less than 16, it goes through the entries in the **segmentPattern** list for that number and turns on the appropriate segments. Note that we can still use on and off even though they're not powered by individual GPIO pins, because the LEDs were declared to GPIO Zero as **active_high = False**.

The project in action; the Pi is measuring how far it is to the Raspberry Pi 3 box.

>STEP-04

The distance sensor

The HC-SR04 distance sensor signals its reading by producing an output pulse that the Pi tries to measure. The GPIO Zero library measures this pulse and converts it into a distance by returning a floating-point number that maxes out at 1 metre. We then multiply this number by 10 to give decimetres. Next, we convert it to an integer, to get rid of the fractional part of the measurement, so we can show it on our single-digit display.

>STEP-05

Building the project

For our build, we used a dinky little breadboard shield from Dtronixs. This allowed for a much more compact arrangement than a conventional breadboard, although you can of course still use one. As the HC-SR04 uses a 5V power supply, the pulse we have to measure is also nominally 5V. Therefore, this has to be cut down to 3V3 by using a 512Ω and 1kΩ resistor voltage divider.

>STEP-06

Using the sensor

The distance to the reflective object is updated every 0.8 seconds. If this is greater than a metre, then the display will be blank. A display of 0 indicates that the object is less than 10cm away. Don't touch the sensor, otherwise its readings will be wrong. Also, as it has quite a wide beam, you can get reflections from the side. If several objects are in the field of view, then the distance to the closest one is returned.

displayDistance.py

Language
>PYTHON 3

DOWNLOAD:
magpi.cc/1NqJjmV

```
# displays the distance
in decimetres on 7
segment display
from gpiozero import LED
from gpiozero import DistanceSensor
import time

seg = [LED(27,active_
high=False),LED(25,active_
high=False),LED(24,active_high=False),
      LED(23,active_
high=False),LED(22,active_
high=False),LED(18,active_high=False),
      LED(17,active_high=False)]

segmentPattern = [[0,1,2,3,4,5],[1,2],[0,
1,6,4,3],[0,1,2,3,6],[1,2,5,6],[0,2,3,5,6
], #0 to 5
                  [0,2,3,4,5,6],[0,1,2],[
0,1,2,3,4,5,6],[0,1,2,5,6],[0,1,2,4,5,6]
, #6 to A
                  [2,3,4,5,6],[0,3,4,5],[1
,2,3,4,6],[0,3,4,5,6],[0,4,5,6] ] #B to F

sensor = DistanceSensor(15,4)

def main() :
   print(
"Display distance on a 7 seg display")
   while 1:
       distance = sensor.distance * 10 #
distance in decimeters
       print("distance",distance)
       if distance >= 10.0:
          distance = 16.0
       display(int(distance))
       time.sleep(0.8)

def display(number):
    for i in range(0,7):
       seg[i].off()
    if number < 16:
       for i in range(0,len(
segmentPattern[number])):
           seg[segmentPattern[
number][i]].on()

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

ROB ZWETSLOOT

Tinkerer, sometime maker, other-times cosplayer, and all-the-time features editor of *The MagPi*.

magpi.cc

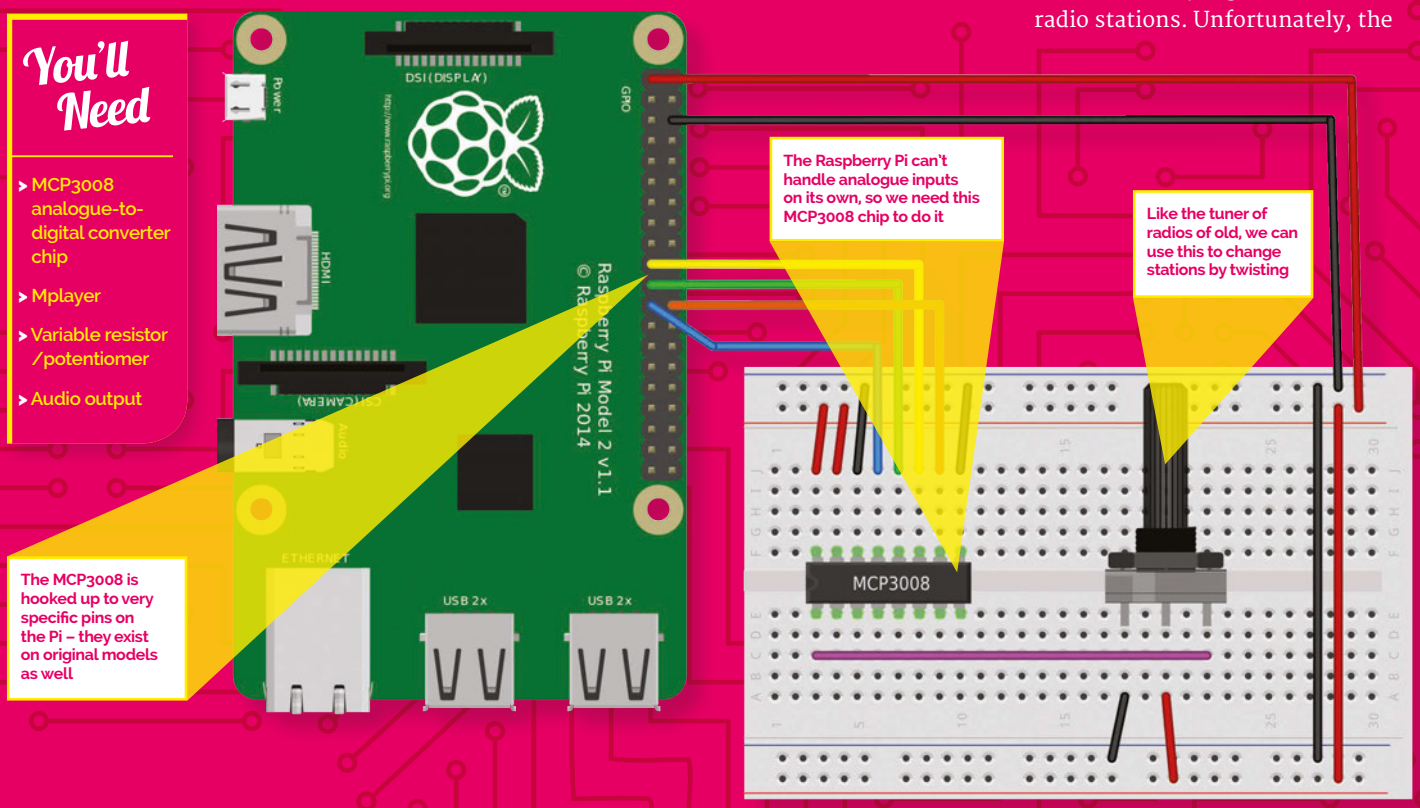
MAKE AN Internet radio

Using advanced parts of GPIO Zero, create an internet radio that uses a dial to switch between stations, just like an old-school radio

This one is a bit cool. The original code for this was developed by *The MagPi* boss Russell Barnes long before GPIO Zero was a thing. The code was over 100 lines long and allowed you to change the volume and select between five different stations using variable resistors/potentiometers. Now, with the latest version of GPIO Zero, we've got it down to 21 lines, although that does leave out volume control for the moment. We'll show you how we did it.

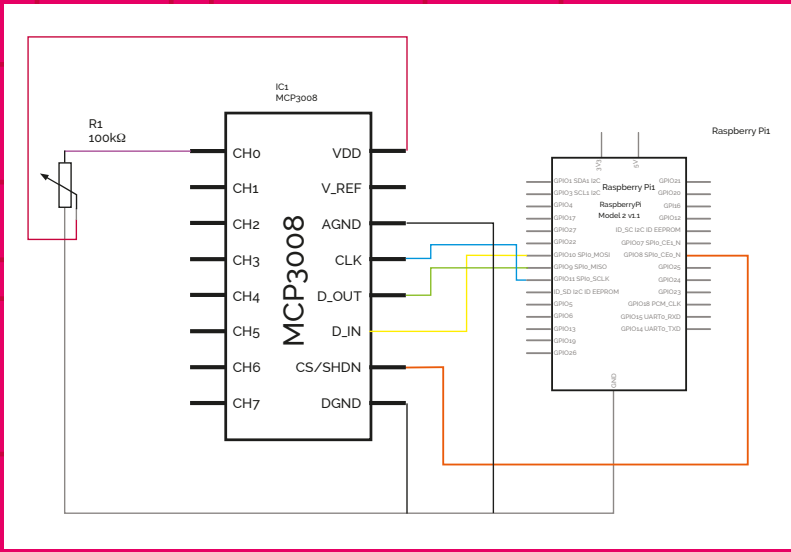
You'll Need

- MCP3008 analogue-to-digital converter chip
- Mplayer
- Variable resistor /potentiomer
- Audio output



>STEP-01
Media software

Before we do anything with the code, we need to make sure we can actually play the radio station. We'll be using the online M3U streams to do so, which can be found for many digital or online radio stations. Unfortunately, the



excellent omxplayer – which is the default Raspberry Pi media player – can't do this, so we need to install mplayer in the terminal with:

```
sudo apt-get install mplayer
```

>STEP-02

Construct the radio

The radio circuit is made up of two main parts: the MCP3008 chip and the variable resistor. The chip lets us read the output of the variable resistor on the Pi. Make sure that when you place it on the breadboard, you do so over the little gap running down the centre; this way the pins won't interact with each other. Be careful when wiring it up: make sure the little indent at the top is the right way round, as shown in the diagram.

>STEP-03

Wiring up a potentiometer

You'll need to identify which pin is which on your variable resistor. It needs a ground, an input, and an output, which are connected to pins 1, 2, and 3 in that order. In our circuit, the input is just the positive 3V3 from the Raspberry Pi. The output is then connected to one of the MCP3008's inputs, which in this case is called channel 0. Twisting the potentiometer will change the resistance and therefore the current at channel 0.

>STEP-04

Add the code

Type up or download the code for the radio. The magic part with this code is that previously you'd have needed many lines of code dedicated to ensuring the MCP3008 chip was being read properly. Now, as long as the chip is wired up correctly, we just tell GPIO Zero which channel we're interested in with the line **`station_dial = MCP3008(0)`**. This is one of the newer additions to GPIO Zero, as part of the v1.2 update.

>STEP-05

Code points

The value reported by the chip for the potentiometer will be a number between 0 and 1. As we only have two stations, we've made it so either side of the turn is a different station. You can add more stations and more selections as you see fit. We're also using the module **`os`** in this code; this allows us to send commands directly to the terminal, and in this case allows us to start mplayer.

>STEP-06

Improvements

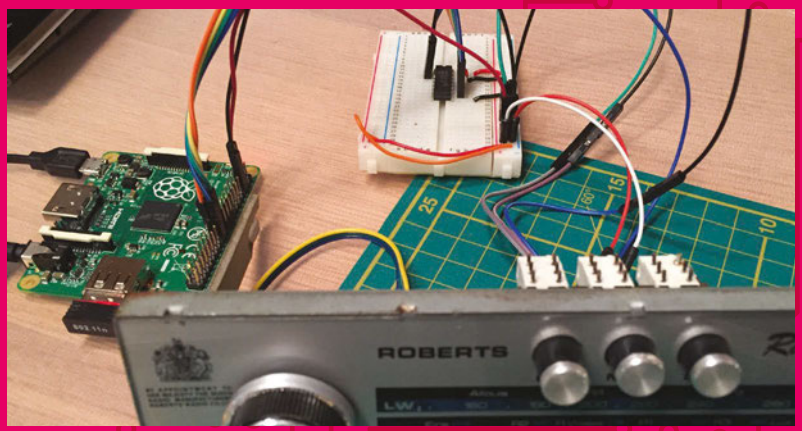
This code is very adaptable if you want to upgrade it. Different or extra radio stations are very easy to add, and it could also be good to include some form of different or better audio output. You can also add on another variable resistor and restore the volume functions, making it a true radio.

internetradio.py

Language
>PYTHON 3

DOWNLOAD:
magpi.cc/10G2Ygi

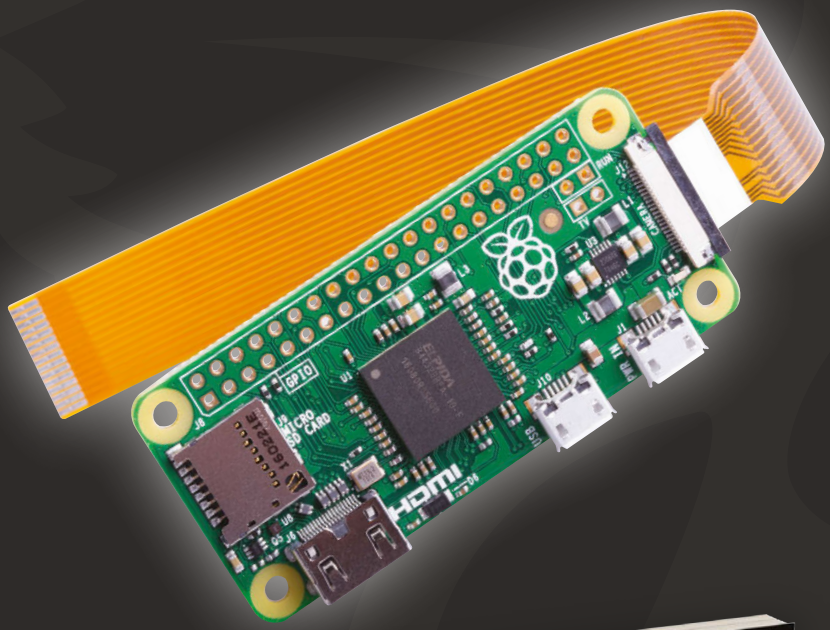
```
from gpiozero import MCP3008
import time
import os

station_dial = MCP3008(0)

Magic = "http://tx.whatson.com/icecast.php?i=magic1054.mp3.m3u"
Radio1 = "http://www.listenlive.eu/bbcradio1.m3u"

def change_station(station):
    os.system("killall mplayer")
    os.system("mplayer -playlist " + station + " &")

while True:
    if station_dial.value >= 0.5:
        station = Magic
        change_station(station)
    elif station_dial.value < 0.5:
        station = Radio1
        change_station(station)
    time.sleep(0.1)
```



Why not try putting the final product inside an old radio case? The dials on the front work similarly

FREE PI ZERO!

Subscribe for six months or a year to receive this stunning free gift today



Subscribe today & receive:

- A free Pi Zero v1.3 (the latest model)
- A free Camera Module connector
- A free USB & HDMI cable bundle

Delivered with your first issue!

Other benefits:

- Save up to 25% on the price
- Free delivery to your door
- Exclusive Pi offers & discounts
- Get every issue first (before stores)

SAVE UP TO 25%

Pricing

Get six issues:

£30 (UK)
£45 (EU)
$69 (USA)
£50 (Rest of World)

FREE PI ZERO!

Subscribe for a year:

£55 (UK)
£80 (EU)
$129 (USA)
£90 (Rest of World)

FREE PI ZERO!

Direct Debit

£12.99 (UK) (quarterly)

How to subscribe:

- **magpi.cc/Subs1** (UK / ROW)
- **imsnews.com/magpi** (USA)
- Call +44(0)1202 586848 (UK/ROW)
- Call 800 428 3003 (USA)

Available on the App Store

GET IT ON Google Play

SUBSCRIPTION FORM

YES! I'd like to subscribe to The MagPi magazine & save money

This subscription is: ☐ For me ☐ A gift for someone*

Mag#46

YOUR DETAILS Mr ☐ Mrs ☐ Miss ☐ Ms ☐

First name ..... Surname .....

Address .....

.....

Postcode ..... Email .....

Daytime phone ..... Mobile .....

**If giving* The MagPi *as a gift, please complete both your own details (above) and the recipient's (below).*

GIFT RECIPIENT'S DETAILS ONLY Mr ☐ Mrs ☐ Miss ☐ Ms ☐

First name ..... Surname .....

Address .....

Postcode ..... Email .....

PAYMENT OPTIONS

1 DIRECT DEBIT PAYMENT £12.99 every 3 issues (UK only)
Instruction to your bank or building society to pay by Direct Debit

DIRECT Debit

Please fill in the form and send to:
The MagPi, Select Publisher Services Ltd,
PO Box 6337, Bournemouth BH1 9EH

Service user number 8 3 8 7 7 3

Name and full postal address of your bank or building society:

To: The Manager Bank/building society .....

Address .....

.....

..... Postcode .....

Name(s) of account holder(s) .....

Branch sort code ☐☐☐☐☐☐ Account number ☐☐☐☐☐☐☐☐

Reference ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ (Official use only)

Instruction to your bank or building society
Please pay Select Publisher Services Ltd Direct Debits from the account detailed in this instruction subject to the safeguards assured by the Direct Debit Guarantee. I understand that this instruction may remain with Select Publisher Services Ltd and, if so, details will be passed electronically to my bank/building society.

Signature ..... Date ☐☐/☐☐/☐☐

Banks and building societies may not accept Direct Debit instructions for some types of account.

SUBSCRIPTION PRICING WHEN PAYING BY CHEQUE OR CREDIT/DEBIT CARD

6 ISSUES ☐ UK £30 ☐ Europe £45 ☐ Rest of world £50

12 ISSUES ☐ UK £55 ☐ Europe £80 ☐ Rest of world £90

2 CHEQUE

I enclose a cheque for ..... (made payable to Select Publisher Services Ltd)

3 CREDIT/DEBIT CARD ☐ Visa ☐ MasterCard ☐ Maestro ☐ Switch

Card number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Expiry date ☐☐ ☐☐ Valid from ☐☐ ☐☐ (if shown)

Issue number ☐☐ (if shown) Security number ☐☐☐
(last 3 digits on the back of the card)

Signature ..... Date ☐☐/☐☐/☐☐

I would like my subscription to begin from issue ..... (month + year)

RETURN THIS FORM TO:
MagPi Magazine Subscriptions, Select Publisher Services Ltd, PO Box 6337, Bournemouth BH1 9EH

☐ Please tick this box if you DO NOT want to receive any other information from Select Publisher Services Ltd.

☐ Please tick this box if you DO NOT want to receive any other information from other companies.

☐ Please tick this box if you DO NOT want to subscribe to The MagPi newsletter.

THOMAS HUDSON

Thomas Hudson is the electronics developer for the Oregon Museum of Science and Technology. He designs exhibits that move across the country.
thomashudson.org

FISH-EYE

There's something fishy about **Thomas Hudson**'s new monitor. We investigate the display that's also an aquarium

Quick Facts

- The Fish-eye was an art installation in Oregon
- The tank was made from acrylic and acrylic glue
- Five common goldfish lived in the tank
- The tank cost around $50 (£35) to build
- It took 12 hours for the fish tank glue to hold

We adore this hybridised fish-tank and LCD screen. Known as the 'Fish-eye', the project was build by Thomas Hudson, a developer at the Oregon Museum of Science & Technology.

The Fish-eye is a fascinating creature. Unexpectedly, the LCD monitor is situated at the front of the tank, and the Raspberry Pi software superimposes the fish onto the display. "A Raspberry Pi Camera Module sits on top of the fish tank, looking down," explains Thomas.

Fish have a tranquil, calming effect, and watching them glide serenely around a tank while you work is awesome. And if you want to watch space fish, then the Earth or Moon make great fishy backdrops.

"I built the tank out of acrylic using acrylic glue," Thomas tells us. "It is amazing stuff: fairly toxic, but it welds the acrylic together so it is watertight.

"The monitors were free," he adds. "One from a free box on the sidewalk and another from a friend. Everyone is getting rid of 19-inch monitors right now."

With the custom tank constructed, Thomas stripped down the LCD screen and fixed it to the front of the fish tank.

"The camera captures live video of the fish from the top view and streams it live to the LCD on the front of the tank. So when you're looking at the front of the tank, you are looking at both live video of the fish from the top view while watching the 'real' fish as seen through the LCD screen."

You don't actually see the fish. The camera records them and superimposes them on your desktop

> "This intersection of moving life and still pixels gave the project its artistic message"

The Fish-eye uses quite a bit of bit of software to achieve its unique effect. It runs a picture program called Feh (**feh.finalrewind.org**) to flash full-screen images of space fish on the Moon. Oxmplayer (**elinux.org/Omxplayer**) is used to show the live stream and also recorded video of the fish. Finally, the Raspberry Pi runs a surveillance script based on Motion MMAL (**magpi.cc/23MBxfH**) to identify moving fish and attach bubbles to those fish displayed on the live video feed.

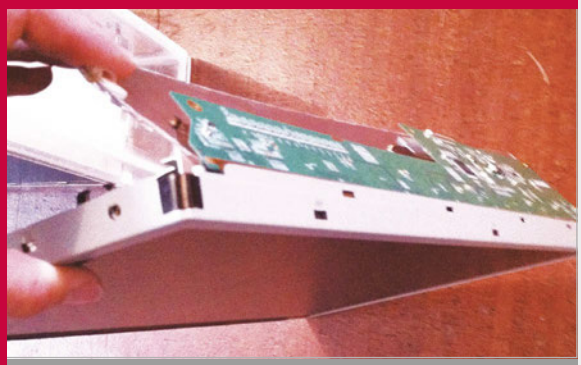
"I worked on the project for about a month," reveals Thomas. "Two shows were happening, so I made two Fish-eyes. One [was] for an annual electronics art exhibition called ByteMe! 5.0; the second was for the Portland Winter Light Festival."

This intersection of moving life and still pixels gave the project its artistic message. "It was important for something to [be] living in the box," explains Thomas. "This is juxtaposed against how much time we spend staring at a 'box' that is not living, and is very much dead, with the exception of all those electrons buzzing through them and the cold emitting light."

The Fish-eye was retired after the Portland Winter Light Festival was over, and the goldfish returned to a regular tank. "It was a sculptural piece," says Thomas. "I feel that kids got it right away. They loved the idea of 'fish on the Moon' and 'fish in space'.

"I think there is something very beautiful about having depth to your monitor. You can still focus on your work, as the screen is crystal clear." Indeed, we think the Fish-eye is a fabulous display.

BUILDING A FISH-EYE

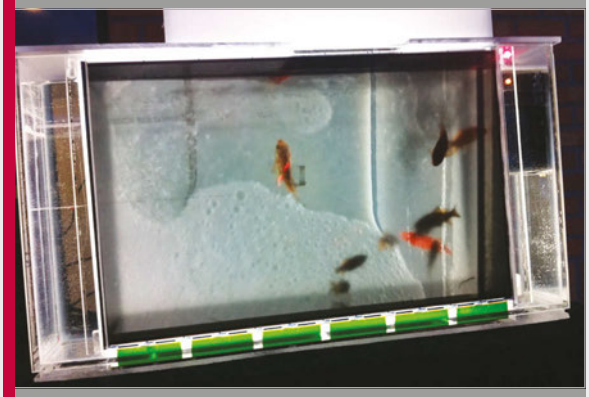


>STEP-01
Stripped display

A 19-inch LCD is recycled from an unloved monitor. Once stripped of its surrounding case, the display will form the front of the Fish-eye screen.

>STEP-02
Tanks a lot

A container is built from sheets of clear acrylic glued together. After the glue has finished drying, the display is attached to the front of the tank.



>STEP-03
Fish Pi

Water fills the tank, and the fish swim inside. The Raspberry Pi and Camera Module blend the display and fish together.

MARK SANDERS

A software developer who loves great coffee, so much so he built his own roaster.

magpi.cc/228QtW9

Quick Facts

- Mark likes to brew coffee in a cafetière, also known as a French press
- Colombian, Ethiopian, and Mexican beans have roasted well
- This is Mark's first Raspberry Pi project
- The whole thing took between two and three months to build
- The roaster is controlled by a custom web interface

RASPBERRY PI COFFEE ROASTER

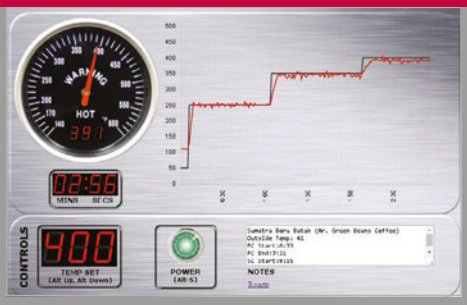
Want to up your coffee-making game even further? Make a cheap yet accurate coffee roaster, like **Mark Sanders** did, for a better cup

Right The build effects on the case add an extra level of class to this custom coffee roaster

Making a truly good cup of coffee takes a lot of effort. Essential oils evaporate from ground coffee within minutes, so pre-ground coffee is out. Boiling water will burn the coffee as it's brewed, so you need it at about 80–85°C. Grind size, exact weights, precise timing, brewing method: all of it is important when making coffee. Possibly the most extreme step you can take is to roast your own beans to make sure they're fresh and to your own requirements, and this is what Mark Sanders has found himself doing.

"While in search of quality coffee, I stumbled upon a webpage

GETTING THE PERFECT ROAST



>STEP-01
Temperature set
The temperature is set in 25°F (~14°C) increments, starting at about 300-350°F (150-175°C) and is manually increased over 5-6 minutes to 450°F (230°C).

>STEP-02
Maintaining temperature
The Raspberry Pi checks the temperature every 0.1 seconds and turns the heating element on or off, depending on whether the roaster needs to warm up or cool down respectively.

>STEP-03
Manual inspection
The roaster won't know when the beans are done, so Mark keeps an eye on the coffee and listens for any cracking in the beans to determine whether or not they're ready.

that suggested home roasting as a source of tasty coffee," Mark tells us. "Getting started is as simple as purchasing a used popcorn popper from the local thrift store for $4-$8."

Popcorn poppers are popular among home-roasters, as they stir the beans while very hot air blows on them – very similar to commercial coffee roasters. There was a small problem, though, according to Mark:

"It roasted the coffee too fast, as I frequently finished roasts in under five minutes [when they should take 7-12]. In order to slow the roast down, I would unplug the popper for 30-second intervals. This grew tiresome and I started to ponder how I could add temperature control to the popper. This was the beginning of my Raspberry Pi project."

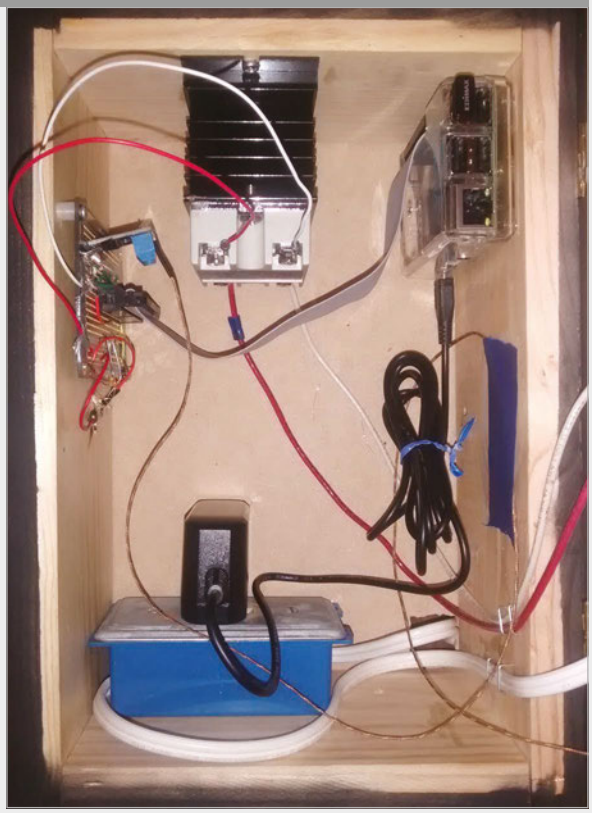
The solution is familiar to those who have tried (or at least seen) sous vide projects: controlling the heating apparatus. In this case, Mark cannibalised the parts from the popper and used them in another vessel.

"I took the heater and fan from a popper and added several other components to create a temperature-controlled coffee roaster," Mark explains. "A thermocouple was added to measure the temperature inside the roasting chamber. A GPIO pin was connected to an AC relay that allowed the Raspberry Pi to control the popper's heater. I developed a web interface to set the temperature and save roast data for archiving. The web interface shows the current temperature using an analogue dial gauge, as well as a chart that graphs the temperature for the entire roast. For some extra flair, four LEDs were added to the front of the roaster to simulate a flame when the heater is turned on."

> "I took the heater and fan from a popper and added several other components"

A full list of parts and instructions to build a roaster yourself is available online: **magpi.cc/228QtW9**. The build is quite fiddly, with a lot of soldering and patience required, according to Mark. However, the results speak for themselves:

"[The roaster] has completed over 20 roasts and it has worked very well. I'm able to control the temperature throughout the entire roast process, log notes about the roast, and save the roast data for reference later. It's much better than unplugging and plugging in the popper by hand."



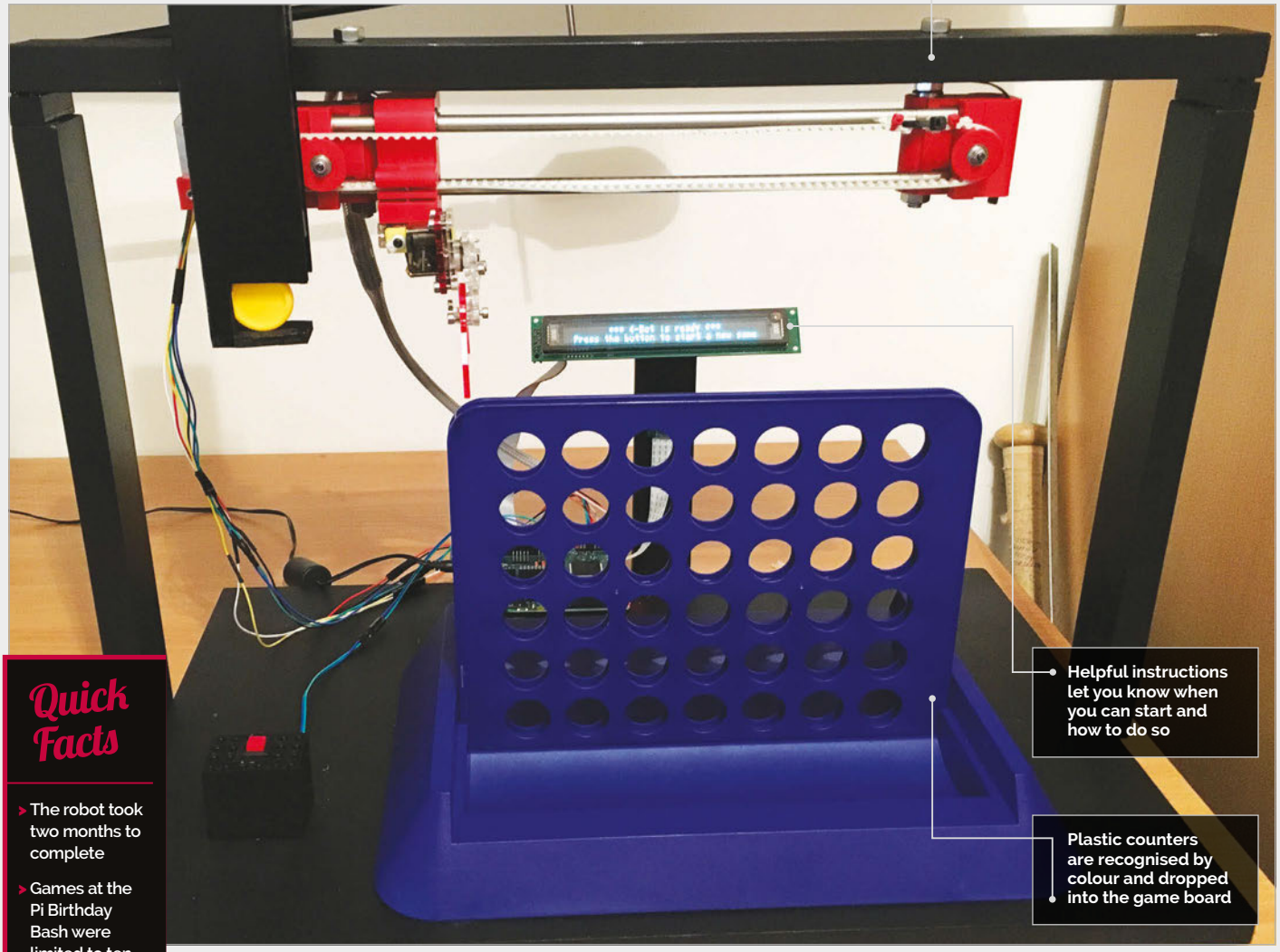
Above The coffee roaster is controlled by a number of electronic components to get it working just right

DAVID PRIDE

Returning to school 30 years later, David became an MSc student thanks to taking up the Pi as a hobby.
magpi.cc/1XrC3zU

4BOT

A robot that plays Connect 4 against a human opponent thanks to a bit of maths and code



Quick Facts

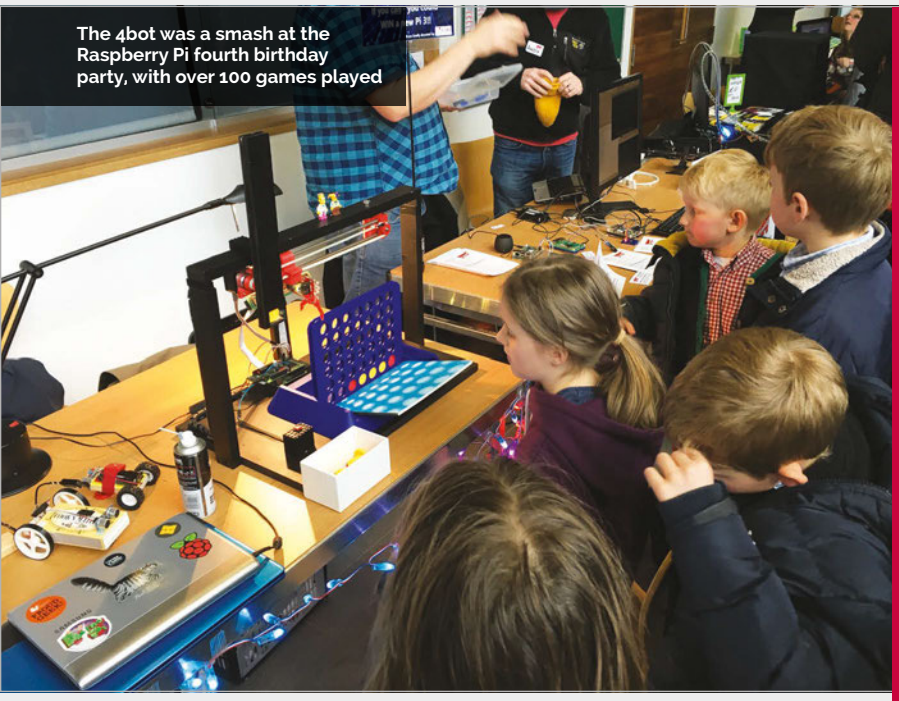
- The robot took two months to complete
- Games at the Pi Birthday Bash were limited to ten moves each
- Only a few people won the game in ten moves
- The add-on board that works the servos will soon be available as a HAT
- David is currently converting an Ohbot to run on Raspberry Pi

You may have seen computers that can defeat grandmasters at chess, win at trivia game shows or, more recently, beat a human at Go, but have you ever played Connect 4 against a robot? Humour us by answering no (statistically speaking, you probably haven't) and then prepare to be amazed at the 4bot created David Pride.

The 4bot has humble beginnings, as David explains its origins to us: "My wife bought me the brilliant MeArm kit and I used it to build a Lego block sorter, as you do. This used the Pi Camera Module and a colour recognition script I wrote in Python to identify the different coloured blocks, and then used the arm to drop them in the correct 'buckets'." A video of this can be seen here: **youtu.be/FJ8WV1uLhFA**.

"Based on this, I was then looking for other uses for the colour-capture code," David continues. "Connect 4 seemed like a really good choice. Research soon led me to find that the game, and the logic behind it, is far from simple. There is good information online; however, whilst I found many versions of Connect 4 for Python, few of them ran successfully on the Pi."

The robot works by taking a picture of the game board, processing the colours, and then giving the game program the state of the board to calculate the next move.



The 4bot was a smash at the Raspberry Pi fourth birthday party, with over 100 games played

"In terms of how well it plays, Connect 4 is a 'perfect' game in mathematical terms," David reveals. "There are a huge but finite number of solutions, and they can all be calculated with enough processing power. The trade-off is in the depth of search and therefore the time taken to calculate each move... If you increase the search depth, this massively increases the calculation time. So I selected a middle ground where the bot plays a pretty mean game, but the total time per move is still acceptable. With capturing and processing the image, calculating the next move, and delivering the counter, the total time per turn is around 25 seconds."

> "In terms of how well it plays, Connect 4 is a 'perfect' game in mathematical terms"

Recognising the colours currently in play is not easy, though, and took a bit of trial and error to get right.

"The trickiest part was undoubtedly capturing the game board accurately every time," David tells us. "It is extremely light-dependent, as the Python module works by capturing the RGB value of the 42 spaces on the board. These values, however, do change dramatically depending on the lighting. I wrote a 'testcard' script that can be run with counters in known locations. This script then reports back what it thinks it sees, and the tolerances for the RGB components can then be adjusted until the result matches what is actually there on the real game board. This made the game more portable, as it can be adjusted to its surroundings each time."

Current upgrades for the robot involve a stronger frame, as it came in for heavy use at the fourth Pi birthday party. Maybe it's time to bring robot Connect 4 battles to the next Pi Wars?...

Above The Pi Factory Lego block sorter

MAKING A MOVE

>STEP-01
Begin game

The 4bot lets you know when it's ready to go. Read the instructions on the display and press the button to start the match.

>STEP-02
Red goes first

It's time to make your first move. Once you're done, the Pi camera takes a photo of the game area and processes it to find out the state of the board.

>STEP-03
Beat the computer

The Raspberry Pi calculates its next move, although it's given a time limit on how much it can think about it. The arm then places the counter in the desired column.

The interface designed by Alain displays six cartoons. Scott uses the buttons to choose which video to play

ALAIN MAUER

Alain lives with his daughter Stacy and son Scott in a small village in the north of Luxembourg.
awallelectronic.blogspot.com

SCOTT TV

One MagPi reader has built a customised television that his autistic son can use unaided

Quick Facts

- The build took two weeks to complete
- Arch Linux is used for the interface
- Videos are played using Omxplayer
- The scripts were written using Python 3.5
- Scott has Kanner syndrome, a severe form of autism

Having a child with autism isn't easy. Alain Mauer's son, Scott, can't make eye contact and doesn't talk. He also requires constant supervision.

"Communication with Scott is very difficult," says Alain. "He understands us, but can't tell us what he wants. You can't leave him alone for a single second.

"But from time to time he has to stay in his room," adds Alain. The answer was to make Scott's bedroom "Scottcompatible! You can lock everything from the wardrobe to the window, and we've installed a camera in the wardrobe too."

But this isn't much fun for Scott. He doesn't like staying alone in his room and finds it boring. "He doesn't play with toys or use his imagination to play. But we know that he likes cartoons with music," reveals Alain.

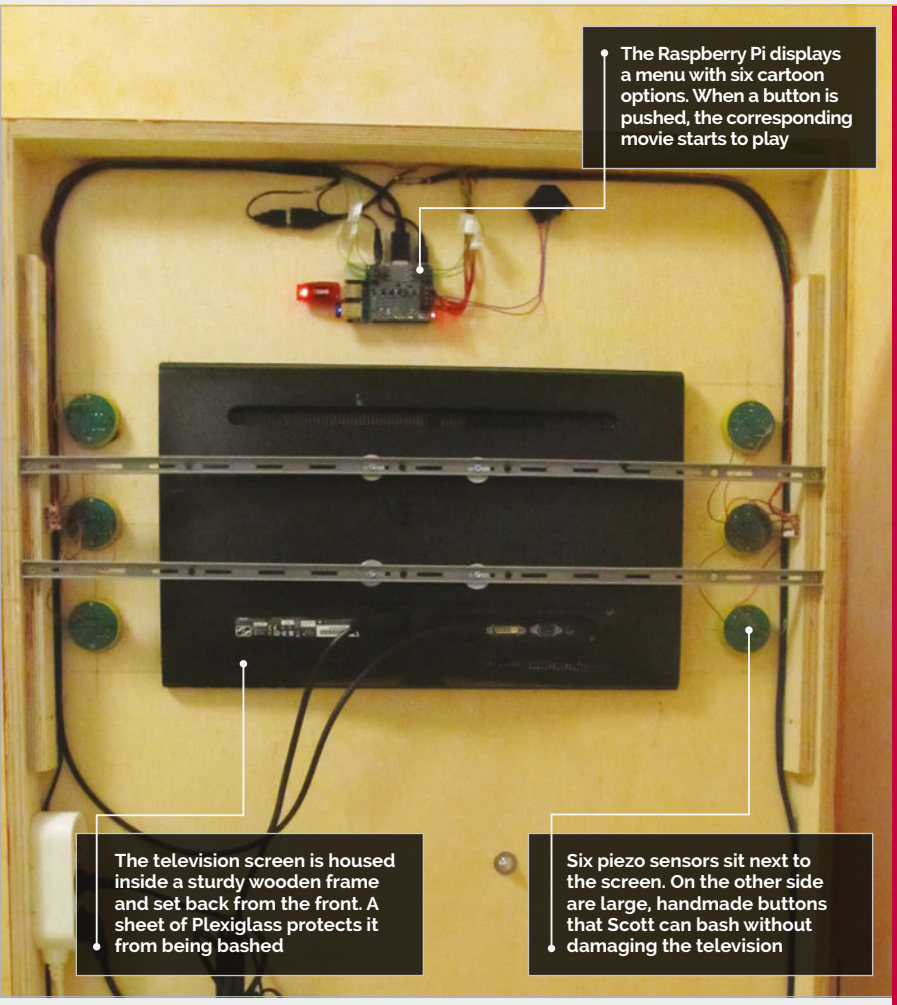
Right Scott enjoying his new TV. He can bash the buttons (and screen) all day long without damaging it

"Scott had a 32-inch TV in his room because he loves to watch cartoons, but one day he destroyed it. So we tried another one with a Plexiglass sheet in front of it, but he tried to destroy it too."

Alain started to wonder if the problem wasn't the television, but what was playing. "He has no ability to tell us, or to stop it, on his own," explains Alain. "So he gets frustrated, and tapping against the noisy thing is his only way of stopping it."

The answer was to build a television that was Scott-proof, and the result is Scott TV: an unbreakable television with the screen hidden inside. Six large, easily bashable buttons start and stop cartoons playing. A Raspberry Pi – tucked safely inside the wooden case – powers the whole project.

Built out of 18mm multiplex wood, the Scott TV case houses the television, but it still needs protection. A sheet of 8mm



Plexiglass in front of the display is the solution. "It wasn't complicated to build," says Alain. "With a jigsaw, a drill machine, and a little table saw, all is possible.

"The Raspberry Pi is the main part of the project. Normally, I tell everybody that the Pi is optimised for controlling stuff and not as a media player. But I was wrong: it plays full HD videos."

The Scott TV has six large buttons, handbuilt by Alain with piezo switches on the inside. "The buttons were the challenge," says Alain, "but you can use any kind of buttons."

The menu displays previews from six cartoons (one next to each button). If Scott pushes a button, the cartoon starts playing. If he pushes any button when a movie is playing, it returns to the menu screen.

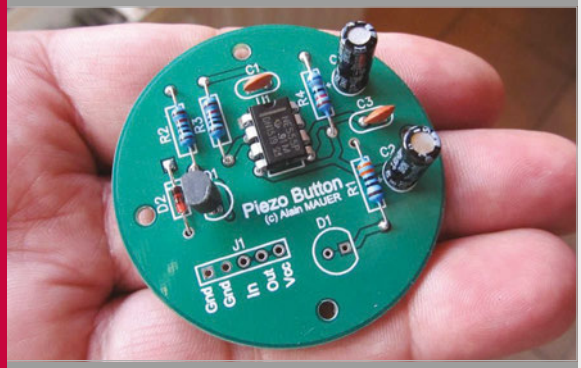
"I used NOOBS to install Arch Linux. For [a] console-based media player, I used Omxplayer. I then used Python 3.5 to write the scripts. I'm not a Linux expert or a programmer, so Google was my friend. The scripts are available on GitHub (**magpi.cc/22amFIG**).

"Since it was set up in his room, Scott likes to stay longer and push the buttons," Alain tells us. "Sometimes he sits in front of the media player and just looks at the animated menu, or he plays the Twinkle Twinkle Little Star video ten times and laughs. To see him so happy was the biggest thank you from him to me."

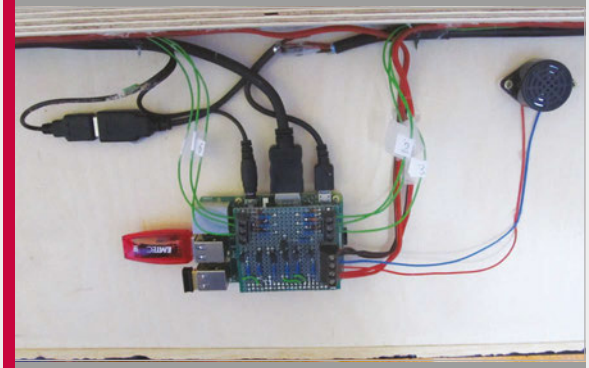
> "To see him so happy was the biggest thank you from him to me"

CREATING SCOTT TV

>STEP-01
Building the case
Multiplex wood was cut, using a jigsaw, to form the case. The large rectangle is for the screen, while the circles form holes for the buttons.



>STEP-02
Button bashing
Scott TV requires six large and robust buttons. These were handmade and proved the most challenging part of the build. Piezo switches are used inside the circular button case to detect pushes.



>STEP-03
Playing media
The buttons and display are wired up to a Raspberry Pi. A Python script detects button pushes, and plays (or stops) one of six cartoons.

The MagPi ESSENTIALS

LEARN | CODE | MAKE

AVAILABLE NOW:

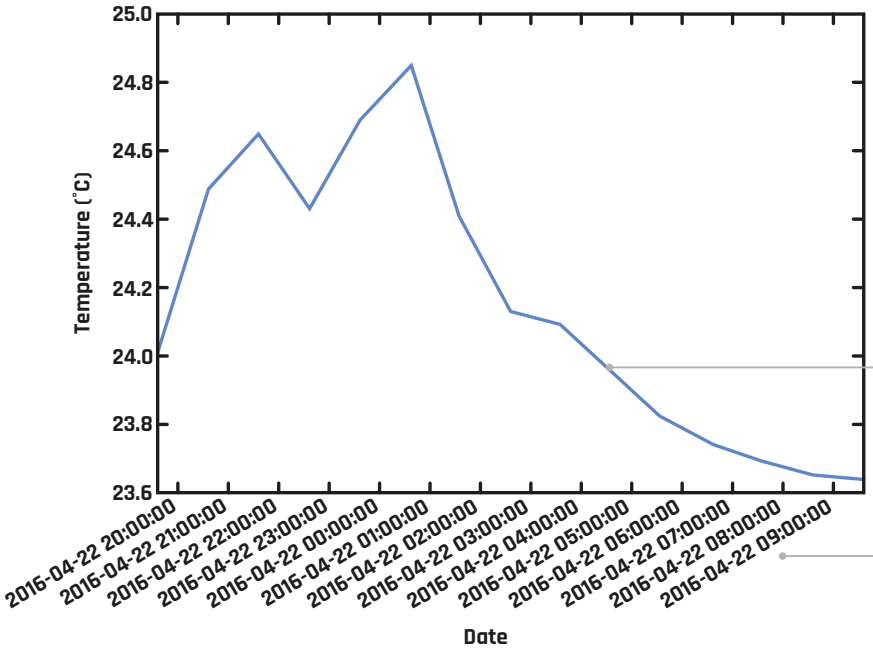
- > CONQUER THE COMMAND LINE
- > EXPERIMENT WITH SENSE HAT
- > MAKE GAMES WITH PYTHON
- > CODE MUSIC WITH SONIC PI

The MagPi ESSENTIALS

From the makers of the official Raspberry Pi magazine

IKER GARCÍA

Born and raised in Eibar (Basque Country), Iker is a PhD student in chemical engineering trying to add a Raspberry Pi to almost everything!
twitter.com/hoopdreams1



Data taken from the Sense HAT can be easily analysed, once plotted out like this

Taking advantage of Matplotlib's powers, the date is shown in a correct format

You'll Need

- Sense HAT **magpi.cc/1TGGUt5**
- Dynamic DNS service **magpi.cc/1TSwg3g**
- Matplotlib **magpi.cc/1TSwgAt**
- Running web server **magpi.cc/1TSwis8**
- Virtual host **magpi.cc/1TSwiYU**

MAKE A SENSE HAT TEMPERATURE RECORDER WITH PYTHON

Data obtained from the Sense HAT can be plotted directly with Matplotlib. With a running web server, this data can be tracked from anywhere!

The Sense HAT enables us to measure different parameters of our environment. However, the raw data obtained from the sensors can be unmanageable. Plotting the data can help us analyse it much more easily; with Python and the powerful Matplotlib library, we can both obtain data from the sensors and create a wide variety of graphs to turn boring numbers into intuitive results. We can also enable a web server and create a virtual host to host the plots so we can access them over the network, creating a beautiful dashboard. While there are services which already provide this, we can build our own for free.

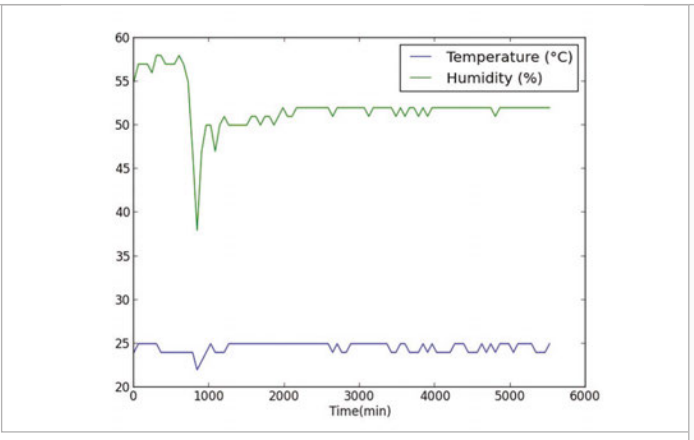
Sense HAT is probably one of the bests HATs you can get for the Raspberry Pi, with plenty of sensors to play with. There are some services that let you take advantage of these sensors and create interactive dashboards, allowing you to see plots of the data from the sensors with just a network connection. However, these services are usually too limited unless you pay for a subscription.

Matplotlib lets you create graphs from Sense HAT data, making it the perfect partner for the HAT. It can make a lot of different graphs, but in this tutorial only the basic one-line plot is going to be covered.

Setting up the web server

With a dynamic DNS (DDNS) service, a running web server, and a virtual host, you can create your own dashboard. While the virtual host, web server, and DDNS service are not essential, without them it would only be possible to track the results locally. Let's suppose that while you are reading this, you have already set up everything you need.

The first step is to create a folder called **dashboard** in **/var/www**. In this folder we will save the CSV file generated by the program, with all the data, and also the HTML or PHP files to create a beautiful dashboard. There are a lot of tutorials out there on how to make your own personalised, custom dashboard. After that's done, create a folder called **images** in **/var/www/dashboard**, in order to save the created graphs there.



Above Different plots can be made: for example, one with two lines that measures temperature and humidity over time

Importing required libraries

In the Python code, we need to import different libraries: **csv** in order to write data to a file, **datetime** in order to get the date, and **matplotlib** to create the plots. After importing these, we first need to be sure that the program can run without an X server running; that's why the **matplotlib.use("Agg")** function is used. Now we also need to import **matplotlib.dates** to format the date for the plot, **matplotlib.pyplot** to draw the plot, **SenseHat** to get data from the Sense HAT, and finally **time**, since we're going to execute all the code inside a **while** loop, with a value for **time.sleep** of 3600 seconds.

Matplotlib needs a list of two values to plot them; that's why we create **x** and **y** empty lists. Now we can start taking temperature data from the Sense HAT, and also the date that corresponds to the value of the temperature.

Once the data is taken, a CSV file is opened in append mode to add the data. This way, we can also have the raw data and not only the plot.

Creating the graph

Instead of taking the data from the CSV file, Matplotlib directly plots the lists that were previously created. To add data to the list, we just append the read temperature and date values to them. With the data on the lists, we can plot the graph and afterwards format both axes and write the proper labels. The final plot-clearing section in the code is required: if not included, the new plot would be drawn over the old one and make a mess.

The code is only an example, but can be used as the basis for amazing programs. As mentioned, Matplotlib is so powerful that we can create other kinds of plots as histograms or animated plots. While the Sense HAT is used for this tutorial, any other sensors can be used. Also, data could be directly taken from raw data files rather than sensors. All this will be covered at the author's GitHub repo (**magpi.cc/1WlQZ35**) soon.

Matplotlib was created by John D Hunter. He passed away in 2012, and this tutorial and all the programs made with his amazing library should be considered in memory of him.

DashboardPi.py

Language
>PYTHON 3

DOWNLOAD:
magpi.cc/1WlQZ35

```
#!/usr/bin/env python
#-*-coding:utf-8-*-
# Create graphs with temperature and data.
# Created by Iker García.

import csv
import datetime
import matplotlib
matplotlib.use("Agg") # Added to plot graphs without running X server.
import matplotlib.dates as md
import matplotlib.pyplot as plt
from sense_hat import SenseHat
import time

x = [] # Define list needed for the plot.
y = []

while True:
  try:
    sense = SenseHat()
    temp = sense.get_temperature() # Reads temperature from Sense HAT.
    temp = round(temp, 2)
    now = datetime.datetime.now()
    file = open("/var/www/dashboard/temperature.csv", "a")
    # Opens file to save data.
    writer = csv.writer(file, delimiter = ",")
    data = [now,temp]
    writer.writerow(data) # Writes data on the csv file.
    x.append(now) # Updates list for the plot.
    y.append(temp)

    plt.plot(x,y) # Plots data set.
    plt.gca().xaxis.set_major_formatter(
md.DateFormatter("%Y-%m-%d %H:%M:%S")) # Format the date.
    plt.gca().get_yaxis().get_major_formatter().set_useOffset(False)
    # Format y axis, in order not to get scientific units.
    plt.xlabel("Date") # x axis label.
    plt.ylabel(u"Temperature (\u00B0C)") # y axis label.
    plt.gcf().autofmt_xdate() # Correct date format on the axis.
    plt.savefig(
"/var/www/dashboard/images/1.png", bbox_inches = "tight")
    # Save the plot with the optimal size.
    plt.clf() # Clears the plot, in order to get a tidy plot.
    time.sleep(3600) # Code is executed each hour.

  except KeyboardInterrupt:
    break
```

JON SILVERA

CEO and founder of FUZE Technologies, Jon Silvera is keen to help kids learn how to code.
fuze.co.uk

MAKE AN ASTEROIDS CLONE IN FUZE BASIC

PART 02

You'll Need

- Raspberry Pi with updated Raspbian
- FUZE BASIC **magpi.cc/1SbgYsY**
- Sound files
- Last month's code

Finish making an incredible Asteroids clone in FUZE BASIC with part two of our tutorial

Last issue, our '4573R01D5' game started to take shape with movement, firing and, of course, our menacing rocks floating around. With that, the basic mechanics of the game are there, but there's no gameplay to speak of yet. This final tutorial adds collision, spaceships, scoring, sound, hyperspace, an attract screen, and the obligatory high score table.

You'll need FUZE BASIC. Visit: **www.fuze.co.uk/getfuzebasic/** to get the latest version.

Continue?

To start, grab the files needed: **magpi.cc/1WBCqIG**. Start up FUZE BASIC and load 4573R01D5 into the editor. To do this, after starting FB you'll be presented with Direct mode, so press **F8** to load a file or **F2** to go to the editor and then load a file. You'll then see the full program listing. There's a lot of code that's in the final version; we'll go through how it all works in this tutorial.

The program consists of a main **LOOP**, then a whole series of **PROC**edures to perform all the main functions.

It's important to note that the **WAIT** command sets a maximum delay minus the amount of time taken in milliseconds. On a fast system, you should set this to `WAIT ( 0.016 - (timeTaken / 1000))` to maintain a constant 60 frames per second. On a Raspberry Pi 2, a delay of 0.010 is better – and remove it altogether if running on an original Model B or B+. Play around to see what you like.

Language
>FUZE BASIC

DOWNLOAD:
magpi.cc/1WBCqIG

Explaining the player code

Every time the player is initially placed on the screen, we use **DEF PROC initialisePlayer** to make sure that there are no rocks nearby. Each asteroid has its distance to the player checked and if it's too close, we update the asteroids and try again. Note that this is not **UPDATE**d, so isn't seen.

The **DEF PROC newExplosion(exX, eyY)** routine is called every time a bullet hits a rock, a rock hits you or an enemy, or if an enemy hits anything. A new explosion is created and put in motion to be drawn by the next routine. **DEF PROC drawExplosions** takes us through the active explosions, and each one of its variables is updated so that each bit of debris (a pixel!) moves in the given direction and at the designated speed.

When a player loses a life, all the variables required to start a new life are set up once again using **DEF PROC resetPlayer**. Basically, everything is reset and repositioned as though the game were just beginning, but score, level, and other gameplay data is left alone.

Drawing the player ship is handled by **DEF PROC drawPlayerShip** and is the simplest routine in the entire program. It just runs through the x and y coordinates held in the **playerData** array and plots the line. Note that if **pThrust** (player thrust) is active, then a couple of extra lines are drawn to show it. There's also the **DEF PROC explodePlayer** routine, which draws a few tiny lines and then expands them over a few frames to simulate the ship exploding.

Coding the gameplay

Let's start with **DEF PROC gameOver**. This simple bit of code displays the 'GAME OVER' text briefly when you lose. **DEF PROC newLevel** resets the asteroids and positions the player in the centre of the screen, ready to begin a new level. **DEF PROC resetSilvoids** resets the asteroids, based on the current level plus a minimum of 4; it starts at 5 on level one, then increases by one for each level. Each asteroid is given a shape type, position, speed, and direction.

For moving the ship, the code uses **DEF PROC playerControls**. Here, **scanKeyboard(scankey)** checks for multiple keys at the same time. **Z** and **X** rotate the ship. Next, we check for thrust (**RIGHT SHIFT**). If thrust is active, i.e. **TRUE**, then thrust/10 is added to the player's x and y position. Finally, we add a decelerator using **pThrustX = pThrustX * 0.995**, so the ship slows down of its own accord.

We're not allowed to hold down the fire button and we're only allowed a maximum of five bullets on screen at once. So, if the **RETURN** key is pressed and isn't being held down, and the number of bullets is less than **maxPlayerBullets**, then we're good to go! Finally, we check to see if our ship has left the screen and if so, reposition it on the opposite side.

Display the action

In **DEF PROC silvoids**, a **LOOP** goes through each asteroid – **0 TO maxSilvoids LOOP**. We check for off-screen and wraparound if needed; next, we figure out what kind of asteroid it is (the type, size, and number of lines), then draw the rock. The screen itself is drawn with **DEF PROC drawScreen**, which also displays the score, any text, and the number of player ships remaining on the screen.

The statement **DistCheck = FN distance (silvoid (count, sX), silvoid(count, sY), pX, pY)**, in **DEF PROC collision**, works out the

> This final tutorial adds collision, spaceships, scoring, sound, hyperspace, an attract screen, and the obligatory high score table

FUZE BASIC

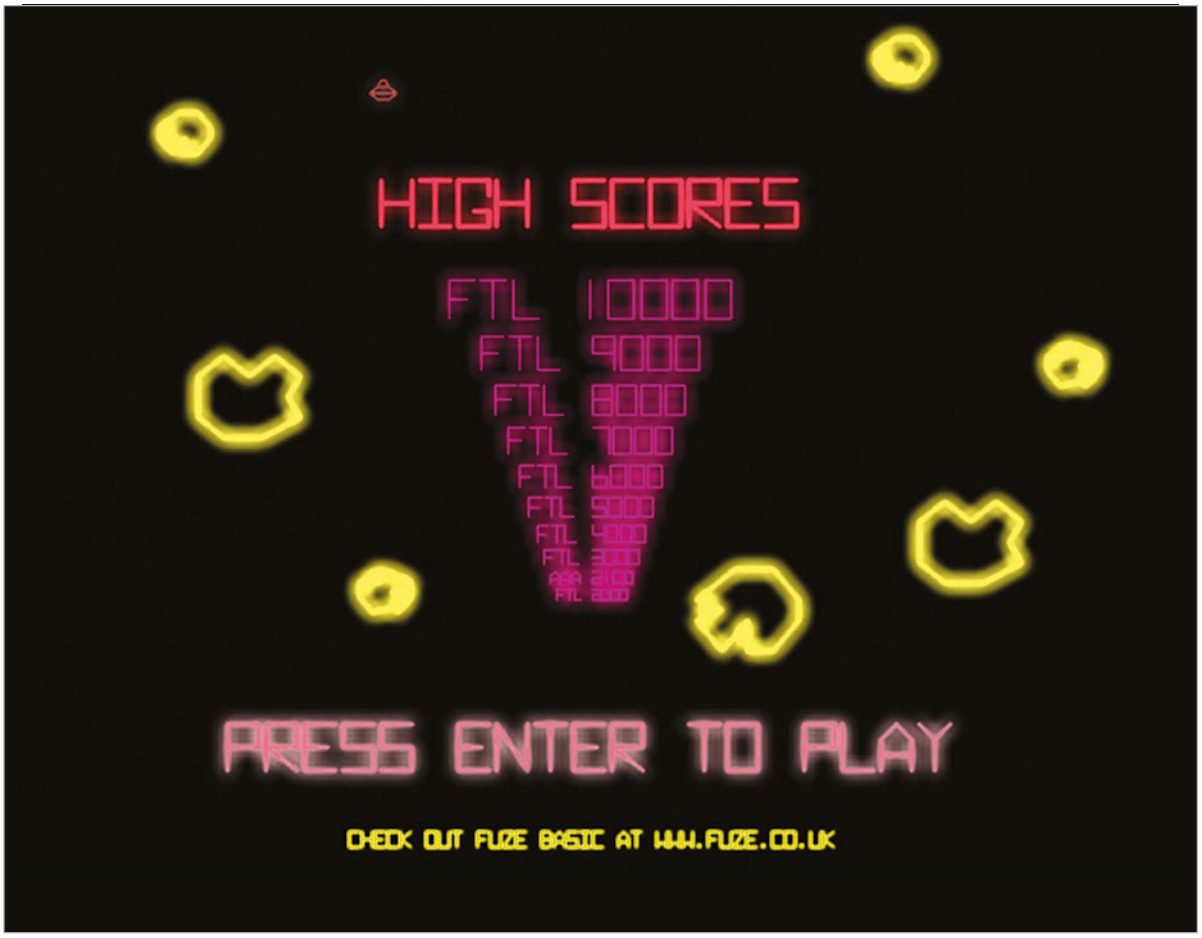
As it's based on the BASIC programming language, if you still have BASIC skills left over from the eighties, they'll still mostly apply today.

FUZE

As well as FUZE BASIC, which brings the BASIC language up to date, you can also get FUZE hardware that's reminiscent of the BBC Micros from the eighties. They're also powered by Raspberry Pi and you can do some GPIO electronics on them as well. Playing this custom version of *Asteroids* on here is sure to trigger nostalgia in certain people. You can have a look at the available models on the FUZE website: **fuze.co.uk/products**

TRUE EXPERIENCE

Wire up some physical buttons to the GPIO ports and modify the code for a true arcade experience.



Right The game remembers high scores and displays them on the attract screen

distance between each asteroid and the player ship. If it's less than the radius of the ship and the rock, i.e. touching, then it's a crash! When we hit a big rock, we want it to break into two, then again for a medium one; if it's a small one, there's just the explosion, so we use **DEF PROC silvoidSmash** to do this.

DEF PROC newPlayerBullet scans through the maximum number of bullets and if it finds an empty slot, creates a new one. **DEF PROC drawPlayerBullets** checks to see if we need to wrap them around the screen, calculate their updated positions, and draw a little circle in place.

Setting up the game

All the initial setting up takes place in **DEF PROC setup**. If you want to try some early hacking, this is where you begin. **DEF PROC hiScoreFile** creates an initial high score list.

DEF FN distance(fnx1, fny1, fnx2, fny2) is without a doubt the most important five lines of code in the entire program. This function works out the distance between two points, which we then use to check if any two objects collide.

Then we have **DEF PROC text(textText$, textX, textY, tR, tG, tB, textSize, bold)** and **DEF PROC fontData**. The first routine draws our text to the screen, but using vector lines and not system fonts. We can also draw symbols, but a simple tiny circle is used for a full stop. The second routine, **fontData**, reads all the character data. The data is represented as 'A', the actual character, and then a number specifying how many coordinate points are required for that character.

The score routine, **DEF PROC score(points)**, is separate so that we can keep a close eye on the score itself. We need to add a new life for every ten thousand points, so we have a second **scoreCheck** to keep track.

DEF PROC initialiseSounds could be included in the main setup, but it can be good to keep things separate where possible. It's where we load in and define our sound effects and music.

Create a demonstration

The routine **DEF PROC attractScreen** uses an array to store numerous copies of the '4573R01D5' logo, each with a slightly different x and y position, size, and colour fade. These are calculated using **COS** and **SIN** to create the circular and wavy effects you see on the title screen. A rather fine tune is played in the background, too. Hats off to David Silvera for that one!

The attract screen also uses the **DEF PROC colourChange** procedure to select new colours for the logo, and a high score table is created with **DEF PROC enterHiScore**. An arcade game is never

LAST OF ITS KIND

There are some ridiculous high scores for the original *Asteroids*. However, the cabinets are slowly dying, so soon you won't be able to challenge them.

HISTORY OF ASTEROIDS

RELEASE DATE:	NOVEMBER 1979
DESIGNERS:	LYLE RAINS, ED LOGG, AND DOMINIC WALSH
PUBLISHER:	ATARI, INC
HARDWARE:	MOS TECHNOLOGY 6502, QUADRASCAN
HIGH SCORE RECORD:	41,338,740, JOHN MCALLISTER, 6 APRIL 2010

The original *Asteroids* is one of the first major hits of what is considered the golden age of arcade games, during the 1970s and early 1980s, when video games were just starting out. Games in this era are typified by classics such as *Pong*, *Space Invaders*, and even *Pac-Man*. *Asteroids* wasn't wholly original, though, being based on concepts from a few previous games such as *Spacewar!* and *Computer Space*. However, the final product was different from the handful of shoot-'em-ups that preceded it.

Asteroids existed in a time when vector graphics and (now considered traditional) pixel-based raster graphics were still both used, depending on what type of game you were making. Vector was used for *Asteroids* to allow for greater precision from the shots fired by the ship, resulting in its iconic triangle ship and asteroids, and other minor graphical effects.

Atari began to port *Asteroids* to its various home gaming console models during the Eighties, strengthening its legacy as one of the all-time greats of video games.

complete without a high score table where you can add your own initials. While this is showing, the game plays in the background, albeit without the player. Use **Z** and **X** to select letters and **#** to confirm.

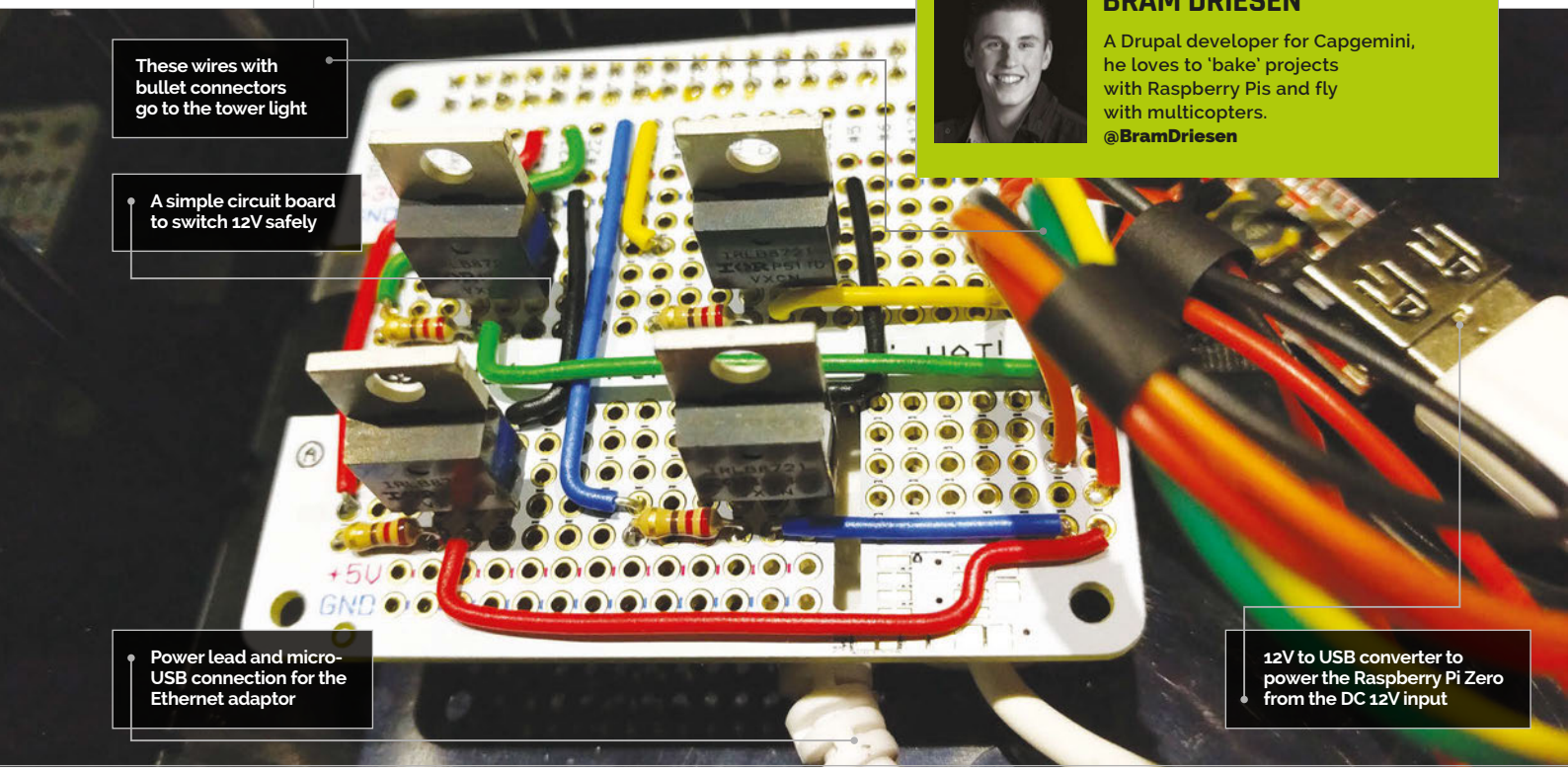
When we start a new game from within a game, we only need to set a few variables rather than setting everything all over again. `DEF PROC gameVariables` handles that. If no enemy ship is in play, then `DEF PROC enemyShip` creates a new one, introducing it based on a random timer which is shortened by the level number. Also, bullet frequency, ship speed, and size of the enemy are all influenced by the level.

The enemy bullet routine `DEF PROC drawEnemyBullets` counts through all the active ones in the array and if `TRUE`, plots a tiny circle in place. Finally, `DEF PROC newEnemyBullet` is the last procedure but also quite a cool one. `ATAN (ABS (ty3 / tx3))` is used to determine the angle of the player ship in relation to the enemy ship. We then modify this with `eBullet(n, eBaccuracy)`, calculated using a random angle divided by the level. We start with a potential angle of up to ±180 degrees divided by the level (divided by two to be kind). By level 20, this comes down to a ±9 degree accuracy range. The result is that enemy bullets become more precise as the levels progress. It also has the bonus effect of making the enemy seem intelligent, as they will sometimes fire just ahead of you. Very nasty!

At around 1,100 lines, 4573R01D5 has become quite the chunk of code. It is, however, incredibly small, as the whole thing is just under 39,000 bytes, or 38kB. It would compress down quite a lot if we took all the comments out, and it could be optimised to bring it down much further. Quite simply, with a bit of tweaking, it would comfortably fit onto a ZX Spectrum!

ARCADE PERFECT

Ports to systems that don't live in the arcade is a common thing for *Asteroids*. It's been on many home systems over the past 30 years!



BRAM DRIESEN

A Drupal developer for Capgemini, he loves to 'bake' projects with Raspberry Pis and fly with multicopters.
@BramDriesen

You'll Need

- IRLB8721 MOSFETs
- 220-ohm resistors
- 12V DC tower light **magpi.cc/1rGNbwB**
- 12V power supply and jack
- 12V to USB converter
- Perma-Proto HAT **magpi.cc/1TQQcn8**
- Micro-USB to Ethernet adaptor
- Miscellaneous other parts

CREATE A JENKINS BUILD STATUS LIGHT

Jenkins is a great tool to monitor code quality. For greater visibility of the build status, you can build an awesome tower light that uses it

Right Using some neodymium magnets, you can mount the tower to any metal object, such as a closet or fridge door

Imagine being able to show the status of your Jenkins projects to anyone in the same room without the hassle of logging in to the dashboard. With some LEDs, a few other cheap components, and some soldering skills, you can complete this project. You can take it to the next level, however, by using an industrial tower light instead of regular LEDs. Employing an existing API, you can retrieve almost any data from Jenkins and get a nice visual indicator of the status of your project.

>STEP-01

Make a HAT

First, we need to create the HAT. We used a Perma-Proto board from Adafruit, but you can also use a breadboard if you don't want a permanent solution. In this version, the following GPIO pins were used, since they best fit the layout:

PIN 18: Red
PIN 23: Buzzer
PIN 24: Yellow
PIN 27: Green

You can use the GPIO output voltage in combination with a MOSFET and a resistor to switch 12V safely to the LEDs. There's also a 12V to USB converter to power the Raspberry Pi from the barrel jack, and a small power switch.

>STEP-02

The Python Jenkins-API

You need to start off with a clean Raspbian Jessie image (not the Lite version) and a Raspberry Pi Zero (make sure to solder the GPIO header pins!). After you have burned the image and completed the basic setup, make sure to set up an internet connection. You can either use a standard WiFi dongle or a micro-USB to Ethernet adaptor. Once you are ready, proceed with the installation of the Jenkins API. In the terminal window, enter the following command:

```
sudo apt-get install python-jenkinsapi
```

>STEP-03

Create the Python script

Most of the coding work has already been done for you and is available on GitHub. To get everything you need for this project, clone the project from GitHub on your Raspberry Pi by using the following command in your terminal:

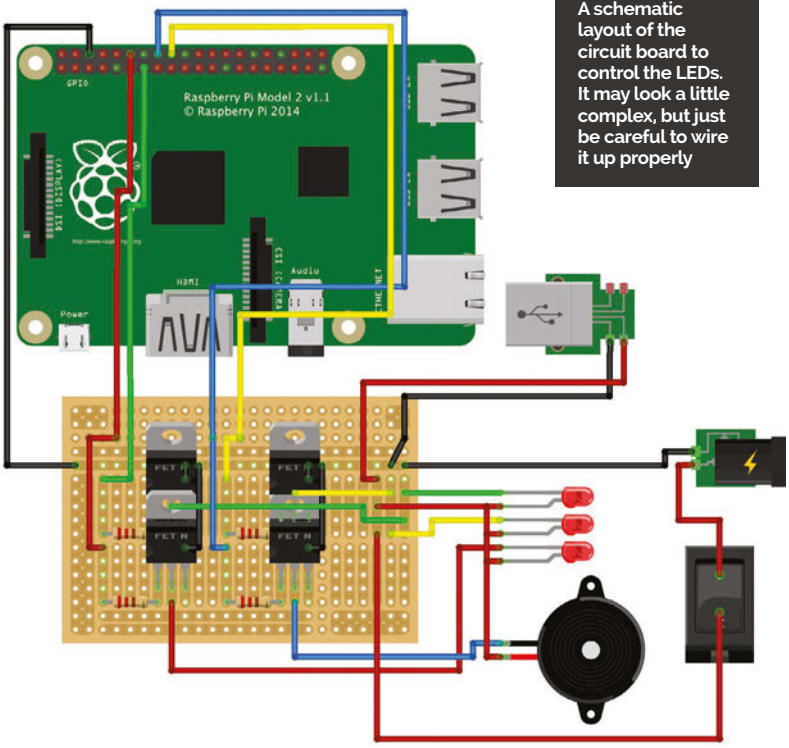
```
git clone https://github.com/BramDriesen/rpi-jenkins-tower-light.git
```

Make sure to clone the project in a directory where you can easily find it afterwards, such as the home directory. You can use **`cd <folder>/<name>`** and **`cd ../`** commands to navigate through folders in your terminal. You can also download a ZIP file of the source code from the versions page: **magpi.cc/1rGOjAt**.

>STEP-04

Configure the code

Now we have cloned the project, we'll browse into the directory to configure the Python script. Copy the default configuration file **default-config.py** and rename it to **config.py**. You can use the command **`cp default-config.py config.py`** to do this all in one go. After you have copied the file, edit it with your favorite text editor; we like to use nano, so in this case we'd issue the command **`nano settings.py`**. Now change the default configuration lines so it contains your credentials, job name(s), and correct GPIO pins according to your situation. If you have only one job, make sure the jobs parameter remains an array with one item inside.



A schematic layout of the circuit board to control the LEDs. It may look a little complex, but just be careful to wire it up properly

Default-config.py

Language >PYTHON

```
# Default configuration file for Jenkins
# Copy this file and name it config.py
jenkinsurl = "http://example-url.com:8080"
username = "your-username"
password = "your-password"
jobs = ['job-name-1', 'job-name-2']
gpios = {
    'red': 18,
    'buzzer': 23,
    'yellow': 24,
    'green': 27,
}
```

>STEP-05

Auto-run on boot

Since we're working on a permanent solution, we want to ensure that our script launches immediately on boot. We can edit our **rc.local** file to launch the script when the Raspberry Pi is booted up. Edit your rc.local file with **`sudo nano/etc/rc.local`** and add the following line to direct the script to auto-start upon boot:

```
python /path/to/the/script/rpi-jenkins-tower-light/jenkinslight.py &
```

Reboot your Pi and you should be good to go! For maintenance, you can always SSH into the Raspberry Pi in the future.

RICHARD HAYLER

Richard is a mentor at CoderDojo Ham, and his school CodeClub was one of the winning teams in the Primary Astro Pi competition whose code ran on the International Space Station in February.

richardhayler.blogspot.co.uk
@rdhayler / coderdojoham.org

SENSE HAT SCIENCE: **PART 04**

FREE-FALLING WITH SENSE HAT

You'll Need

- A Sense HAT **magpi.cc/1TGGUt5**
- Python 3 sense_hat, evdev & select libraries
- Something to provide a soft landing
- A charged power bank

Why do astronauts on the ISS float?

Ask most people why astronauts like Tim Peake float around on the International Space Station and there is a good chance they'll say, "Because there is no gravity." If you think about it, this can't be the case. The ISS orbits the planet at 400km, but the Moon is nearly 100 times further away and is still subject to Earth's gravitational pull. So why are astronauts weightless? It's because the ISS is constantly in free fall, and so the astronauts living inside it experience weightlessness.

This can be difficult to wrap your head around, so we'll use the Sense HAT to explain it.

If you've been following the previous Sense HAT Science articles, this should all be familiar now: attach the Sense HAT to your Pi and power it up. Open IDLE3, or just type **python3** in a terminal window, and then import the **SenseHat** library and connect to the board:

```
from sense_hat import SenseHat
sh = SenseHat()
```

There are a number of ways we can measure motion, forces, and orientation with the Sense HAT: it has a gyroscope, magnetometer, and an accelerometer. We're going to use the accelerometer, and the line of code we need probably won't be a surprise:

```
sh.get_accelerometer_raw()
```

We're using **_raw()** as it returns results in Gs. One G is equal to the force of gravity on the Earth's surface (9.8m/s), and you often hear jet planes, spacecraft, and even roller-coasters described in terms of how many Gs their riders experience.

The three values measured by the Sense HAT – x, y, and z – correspond to the pitch, roll, and yaw axes. If we run the following Python snippet:

```
while True:
    print(sh.get_accelerometer_raw())
```

…then we should see a continuous stream of the x, y, and z values displayed on screen. Jiggle and wiggle the Pi and Sense HAT around and see how the numbers change. Can you carefully move the Pi so that just one of the axis values changes?

Taking readings

For our experiment, we need to able to record x, y, and z readings constantly and save them to a file so that we can analyse them later. To make things easier (and to make our data file smaller), we will use the joystick to start and stop recording. There are several ways of doing this with Python, but the code uses the **evdev** library to register joystick presses.

To run the experiment, power the Pi up from a battery pack and secure the two together using rubber bands or cable ties. Start the code and then disconnect the keyboard, mouse, and display cables.

Now find somewhere from where you can drop the Pi. The greater the distance it falls, the easier it is to see the free-fall period. However, don't forget that you need a big, soft landing zone!

When you're ready, press the joystick and you should see the 8×8 LED matrix light up all green. That means data is being recorded, so bombs away! Once the Pi has landed, press the joystick again to end the logging (the LEDs should briefly flash red, then turn off completely).

Analysing the data

For the data analysis, we can also use the Pi. Plug all the cables back in and start up LibreOffice Calc. The code conveniently writes the data into a CSV format that can easily be ingested by this (or any other) spreadsheet program, which can be used to plot graphs of our data.

From the graph, you should see that before the drop, the Sense HAT accelerometer measures approximately +/-1G, then falls/rises to 0 while it is falling. You'll probably also see the 'bump' corresponding to the landing (hopefully not too big).

So, now we've shown that a falling object will experience zero G, we can see why astronauts on the ISS appear to float. However, if the ISS is constantly falling, why doesn't it just crash into the Earth? This is because the Space Station is also moving 'forward' at a speed fast enough to ensure that it keeps missing the Earth and stays (falling) in orbit.

You can, of course, use the Sense HAT to measure forces greater than 1G, too. Try taking your portable Pi and Sense HAT on a playground ride or, for some serious Gs, a theme park attraction. If you're going to take your Pi and battery pack on a roller-coaster, though, make sure it is securely stowed away in a zippable pocket!

gravity.py

Language >PYTHON 3

DOWNLOAD: magpi.cc/1XoqOOu

```
from sense_hat import SenseHat
from datetime import datetime
import sys
from time import sleep
from evdev import InputDevice, ecodes, list_devices
from select import select
import logging

# Use the logging library to handle all our data recording
logfile = "gravity-"+str(datetime.now().strftime("%Y%m%d-%H%M"))+".csv"
# Set the format for the timestamp to be used in the log filename
logging.basicConfig(filename=logfile, level=logging.DEBUG,
    format='%(asctime)s %(message)s')

def gentle_close(): # A function to end the program gracefully
    sh.clear(255,0,0) # Turn on the LEDs red
    sleep(0.5) # Wait half a second
    sh.clear(0,0,0) # Turn all the LEDs off
    sys.exit() # Quit the program

sh = SenseHat() # Connect to SenseHAT
sh.clear() # Turn all the LEDs off
# Find all the input devices connect to the Pi
devices=[InputDevice(fn) for fn in list_devices()]
for dev in devices:
    # Look for the SenseHAT Joystick
    if dev.name=="Raspberry Pi Sense HAT Joystick":
        js=dev
# Create a variable to store whether or not we're logging data
running = False # No data being recorded (yet)
try:
    print('Press the Joystick button to start recording.')
    print('Press it again to stop.')
    while True:
        # capture all the relevant events from joystick
        r,w,x=select([js.fd],[],[],0.01)
        for fd in r:
            for event in js.read():
                # If the event is a key press...
                if event.type==ecodes.EV_KEY and event.value==1:
                    # If we're not logging data, start now
                    if event.code==ecodes.KEY_ENTER and not running:
                        running = True # Start recording data
                        sh.clear(0,255,0) # Light up LEDs green
                    # If we were already logging data, stop now
                    elif event.code==ecodes.KEY_ENTER and running:
                        running = False
                        gentle_close()
        # If we're logging data...
        if running:
            # Read from acceleromter
            acc_x,acc_y,acc_z = [sh.get_accelerometer_raw()[key] for key in [
'x','y','z']]
            # Format the results and log to file
            logging.info(
'{:12.10f}, {:12.10f}, {:12.10f}'.format(acc_x,acc_y,acc_z))
            print(acc_x,acc_y,acc_z) # Also write to screen
except: # If something goes wrong, quit
    gentle_close()
```

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

COMMAND LINE PI

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

As close to perfect as Raspbian is, things can go wrong. In this tutorial, we learn that there's no need to turn the Raspberry Pi off and on again: just kill the process!

Ever lost the 'off switch' for a program? Sometimes software you're running seems to have no inclination to stop: either you can't find out how to quit, or the app has a problem and won't respond to your **q**, **CTRL+C**, or whatever command should close it down.

There's no need to panic, and certainly no need to reboot: just identify the process and quietly kill it. We'll show you how, and look at what else can be done with knowledge of processes.

Processes

Find the many processes running on your Pi with the **ps** command. On Raspbian, it's usually called with the **a** and **x** switches which give all processes, rather than just those belonging to a user, the **u** switch shows processes by user, attaching it to a **tty**. **w** adds wider output, and **ww** will wrap over the line end to display information without truncating.

Type **ps auxww** to see, then try with just **a** or other combinations. You'll notice that these options work without the leading dash seen for other commands. Both the lack of dashes, and the letters **a** and **x**, date back to the original Unix ps of the early 1970s; this was maintained through various revisions by one of Unix's two family branches, **BSD**, and baked into the first GNU/Linux ps. Unix's other branch, System V, extended and changed ps with new options and new abbreviations for command switches, so for **ps ax** you may also see **ps -e** (or **-ef** or **-ely** to show in long format).

KEEP ON RUNNING

nohup is useful for a program that will be running for some time in the background – perhaps a sensor project you're working on – until you feel happy enough to add it to Raspbian's startup processes.

The **ps aux** listing has various headers, including the **USER** which owns the process, and the **PID** (process identification number). This starts with 1 for init, the parent process of everything that happens in userspace after the Linux kernel starts up when you switch the Pi on. Knowing the **PID** makes it easy to kill a process, if it's the easiest way of shutting it down. For example, to kill a program with a **PID** of 3012, simply enter **kill 3012**. To quickly find the process in the first place, use **grep** on the ps list. For example, locating vi processes:

```
ps aux | grep -i  vi
```

The **-i** (ignore case) isn't usually necessary, but occasionally a program may break convention and contain upper-case letters in its file name. You can also use **killall** to kill by program name; for example, with **killall firefox**.

Piping commands

Naturally, you can pipe ps's output to select the PID and feed directly to the kill command:

```
kill $(ps aux | grep '[f]irefox' | awk
'{print $2}')
```

We don't have space for an in-depth look at **awk** (we're using it here to print the second field of grep's output, the **PID**), but the **[f]** trick at the beginning

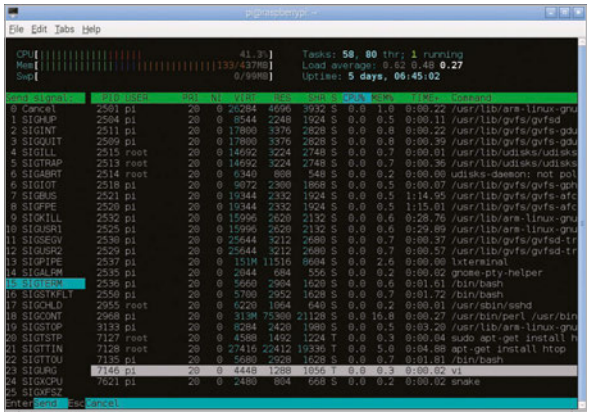


Fig 1 `htop` tells you what's running, what resources it's using, and lets you interact with the process, even killing htop from within htop

of Firefox (or whatever named process you want to kill) singles out the Firefox process. In the vi example above, grep found the grep process itself as well as vi (and anything with the letter sequence vi in its name).

The output of ps also shows you useful information like percentage of memory and CPU time used, but it's more useful to see these changing in real time. For this, use **top**, which also shows total CPU and memory use in the header lines, the latter in the format that you can also find by issuing the command **free**. For an improved top:

```
apt-get install htop
```

htop is scrollable, both horizontally and vertically, and allows you to issue commands (such as **k** for kill) to highlighted processes. When you've finished, exit both top and htop with **q**, although in htop you may care to practise by highlighting the htop process and killing it from there (see **Fig 1**). htop also shows load over the separate cores of the processor if you have a Pi 2 or 3.

Background job

Placing an ampersand after a command in the shell places the program in the background – try with: **man top &** and you'll get an output like **[1] 12768**.

The first number is a job number, assigned by the shell, and the second the PID we've been working with above. man top is now running in the background, and you can use the job control number to work with the process in the shell. Start some other processes in the background if you wish (by appending &), then bring the first – man top – to the foreground with **fg 1**. Now you should see man running again.

You can place a running shell program in the background by 'suspending' it with **CTRL+Z**. fg will always bring back the most recently suspended or backgrounded job, unless a job number is specified. Note that these job numbers apply only within the shell where the process started. Type **jobs** to see background processes; **jobs -l** adds in process IDs (PIDs) to the listing.

Signals

When we sent a kill signal from htop, we were given a choice of signal to send. The most important are **SIGTERM**, **SIGINT**, and **SIGKILL**. The first was the default when we killed from htop, and is the signal that kill sends if not called with a modifier. It tells a process to stop, and most programs will respond by catching the signal, saving any data they need to save, and releasing system resources before quitting.

kill -2 sends **SIGINT**, the equivalent to stopping a program from the terminal with **CTRL+C**: you could lose data. Most drastic is **kill -9** to send **SIGKILL**, telling the kernel to let the process go with no warning. Save this one for when nothing else works.

Mildest of all is the Hang Up (**HUP**) signal, called with **kill -1**, which many daemons are programmed to treat as a call to simply reread their configuration files and carry on running. It's certainly the safest signal to send on a critical machine.

Staying on

nohup will run a program which will continue after the terminal from which it's started has closed, ignoring the consequent **SIGHUP** (**hangup**) signal. As the process is detached from the terminal, error messages and output are sent to the file **nohup.out** in whichever directory you were in when you started the process. You can redirect it, as we did in part 4 (issue 34), with **1>** for **stdout** and **2>** for **stderr**; **&>** is a special case for redirecting both **stdout** and **stderr**:

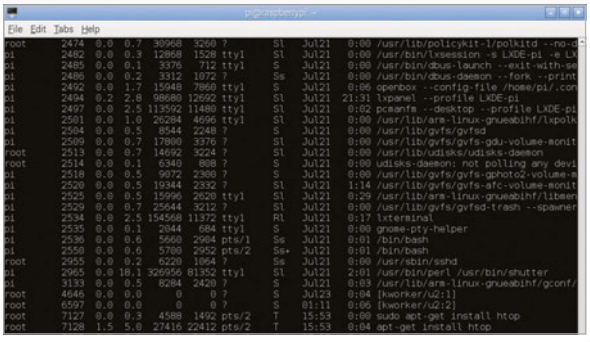
```
nohup myprog &>backgroundoutput.txt &
```

One use of **NOHUP** for Pi users is to be able to set something in motion from an SSH session, that will continue after an interruption to that session. For example, restarting the network connection to which you are connected:

```
sudo nohup sh -c "ifdown eth0 && ifup eth0"
```

Note that the **nohup.out** log file created here will need sudo privileges to read – or reassign with:

```
sudo chown pi:pi nohup.out
```



Above Everything running has a process ID (PID) that can be used to control that program; find them all with `ps aux`

QUICKER BOOT

The startup process of Raspbian is controlled by SysVinit, but like other GNU/Linux distributions will eventually change to the new, faster SystemD. This will change startup processes, but instructions here will still be relevant.

KEEP ON TOP

Using a virtual console, it can be worth keeping htop running so that if there are any problems, you can CTRL+ALT+Fn there for a quick look – even if the GUI freezes.

WESLEY ARCHER (AKA RASPBERRY COULIS)

Self-taught Raspberry Pi enthusiast, guide writer for Pi Supply and creator of raspberrycoulis.co.uk. Enjoys trying out new Pi projects and sharing them!
raspberrycoulis.co.uk / @RaspberryCoulis

You'll Need

- Lisiparoi LED light ring (infrared version) **magpi.cc/1SQVFrW**
- Cyntech Raspberry Pi case **magpi.cc/1SQVGvW**
- Raspberry Pi NoIR Camera Module
- USB WiFi adapter
- MotionEyeOS **magpi.cc/1UCw1Jk**
- Drill and small drill bit (1mm or 2mm ideal sizes)
- Small file (we used a metal nail file)
- Pencil and a sharp knife (craft or Stanley knife)

RASPBERRY PI NIGHT VISION CAMERA HACK

Have a spare Raspberry Pi Model B lying about? Turn it into a night-vision CCTV camera with this simple case hack...

Many people have a number of Raspberry Pis at home. Some might say those folks are obsessed, but Pis are so versatile that trying out new projects where possible is great fun! Putting them to good use is always exciting, so why not turn a spare Model B into a night-vision CCTV camera, using an inexpensive case, the Camera Module, and a nifty little accessory called the Lisiparoi to provide infrared lighting? If you saw the previous guide on adding push notifications to MotionEyeOS (issue 43), then combining this with night vision would be a great addition!

>STEP-01

Mark where you will cut

Before cutting into the case, it makes things easier to mark it out first. Grab your pencil, place your Camera Module and Lisiparoi on the case, and then draw around them. You should be able to see the pencil line, and it's easy to remove if you need to. Make sure you allow enough room around the outside so that it's not too close to the edge of the case, though! You'll also need to cut out space for the pins on the Lisiparoi, holes to mount the Camera Module board, and a hole for the camera lens itself.

>STEP-02

Drill the mounting holes

Using a small drill bit (1mm / 2mm is ideal), drill two small holes in the case. If you are very careful, you can use the Lisiparoi as a template. You should now have two small holes in the case and a pencilled outline of the Lisiparoi. It's now time to drill holes for the header pins, which are a few millimetres away from the bottom edge of the Lisiparoi. It's best to start small and increase the size of the holes as you go; it doesn't have to be exact, as the Lisiparoi will hide everything once in place.

>STEP-03

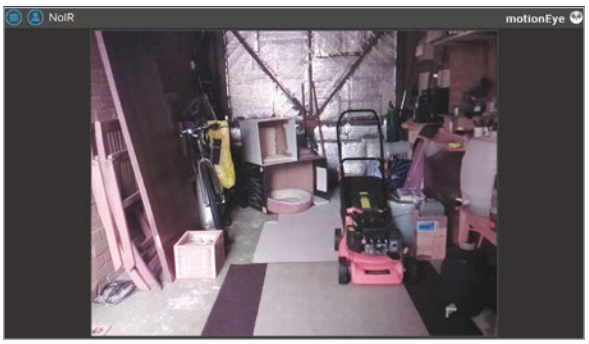
Drill, file, test, and repeat

Whilst doing this, remember – you can always cut more off! To make sure our header pins on the Lisiparoi fit nicely, we drilled the holes to the approximate width of the header and then used a small file to square off the holes. We then placed the Lisiparoi in place to see if it fit; if it didn't, a little more was filed off until it did. It doesn't have to be surgically accurate, since the Lisiparoi sits on top of the holes; even so, take your time, as you can't undo a hole if it's too big!

>STEP-04

Raspberry Pi NoIR Camera Module

It's now time to mark and cut a hole for the Camera Module. Again, we drew around the module with a pencil and then drilled small holes along the inside edge of the outline (otherwise the hole will be too big). Next, very carefully use a sharp knife to cut out the hole; the drilled holes along the outline should help with this step. Make sure this is done on a tough, steady surface (we used an old chopping board). Fine-tune the fit by filing the hole and testing that the Camera Module fits. Repeat until it does.



Left Once it's complete, you should hopefully see something like this from your Raspberry Pi night-vision CCTV camera!

>STEP-05

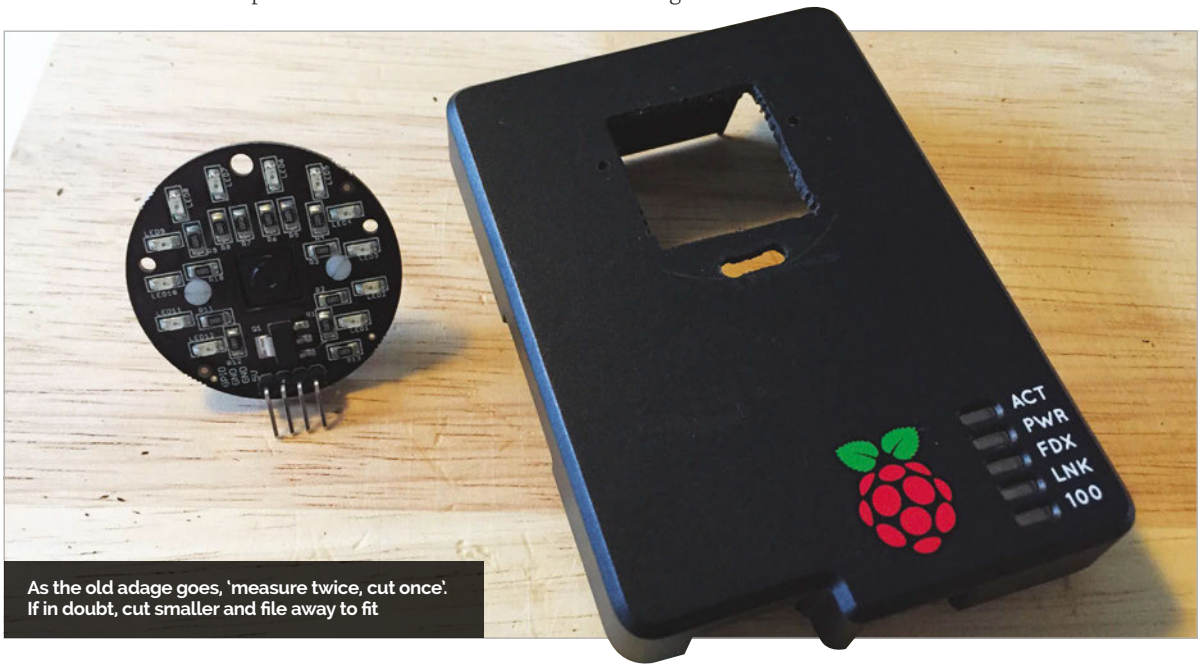
Put everything together

Now that the holes are all cut and fit nicely, it's time to put everything together. Make sure you've soldered the header onto your Lisiparoi first, though! The header needs to be soldered so the pins are on the inside of the case and pointing downwards, so that the jumper cables can be fitted to the Pi's GPIO once fully assembled. Screw your nuts and bolts together and connect up your cables, including the ribbon cable for the Camera Module. For more information on using the Lisiparoi, including wiring diagrams, check out the official site here: **magpi.cc/1SQWvoo**.

>STEP-06

Install MotionEyeOS and test everything

If you have connected the Lisiparoi so that it turns on whenever the Pi does, then you just need a CCTV operating system. MotionEyeOS is perfect for this, because it's simple to use and works very well! Download the relevant image from **magpi.cc/1UCw1Jk**, then write it to your SD card. You won't be able to see if the infrared LEDs are on, but test it out in the dark and you should see more than you would normally without them! As the LEDs are small, they're not very powerful, but should provide enough illumination to see better in the dark!



As the old adage goes, 'measure twice, cut once'. If in doubt, cut smaller and file away to fit

MEASURE, THEN MEASURE AGAIN!

They always say 'measure twice, cut once', and so do we! It is better to be safe than sorry.

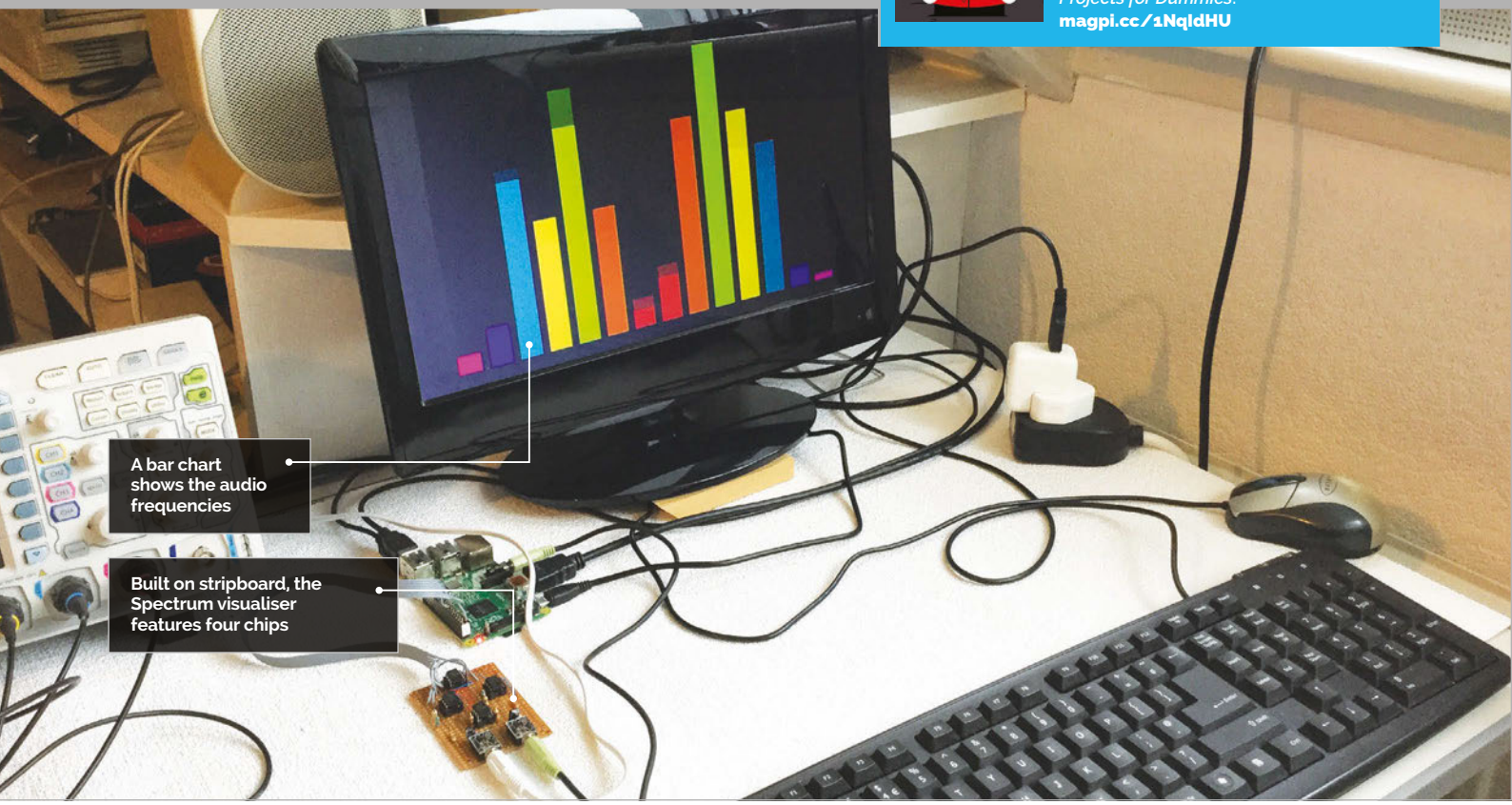
LESS IS MORE

You can always cut more, but you cannot replace what is gone. Think about this when drilling and filing!

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*.
magpi.cc/1NqIdHU



You'll Need

- 2× MSGEQ7 equaliser chip
- 1× MCP602 operational amplifier
- 1× MCP3002 2-channel, 10-bit A/D converter
- 4× 8-pin DIL sockets
- 2× 3.5mm board mounting stereo jack sockets
- 1× 28-hole by 19-hole stripboard
- 1× 26-way GPIO header socket
- Assorted resistors, capacitors, and wires

SPECTRUM SHOW

A STEREO SOUND VISUALISER

Turn any sound source into a stunning graphics display

Converting sound to lights has always been a popular project, although there is little in the way of Raspberry Pi versions, so this month we thought we would remedy that. This project uses the popular MSGEQ7 chip, which is officially obsolete but still widely available. This chip has seven second-order, switched capacitor, analogue filters built into it that cover the audio spectrum. These filters peak at 65Hz, 160Hz, 400Hz, 1kHz, 2.5kHz, 6.25kHz and, for those youngsters still lucky enough to hear that high, 16kHz. The frequency response is shown in **Fig 1** (page 64); note that both the frequency and the output are plotted on a log scale. This ensures that the response looks like it sounds, because of our logarithmic sensing of both frequency and loudness.

The actual peak frequency of each filter depends on the built-in oscillator that is set with an external resistor and capacitor, 200kΩ and 33pF respectively. The chip takes in an audio signal and passes it through these seven filters. However, the chip has only one output: it's an analogue signal that gives the output of each filter in turn, as you feed it clock pulses. The fact that the Pi has no built-in analogue input is probably why there's not many Pi projects using this chip. To get round this, we used the MCP3002 chip that featured in last month's project.

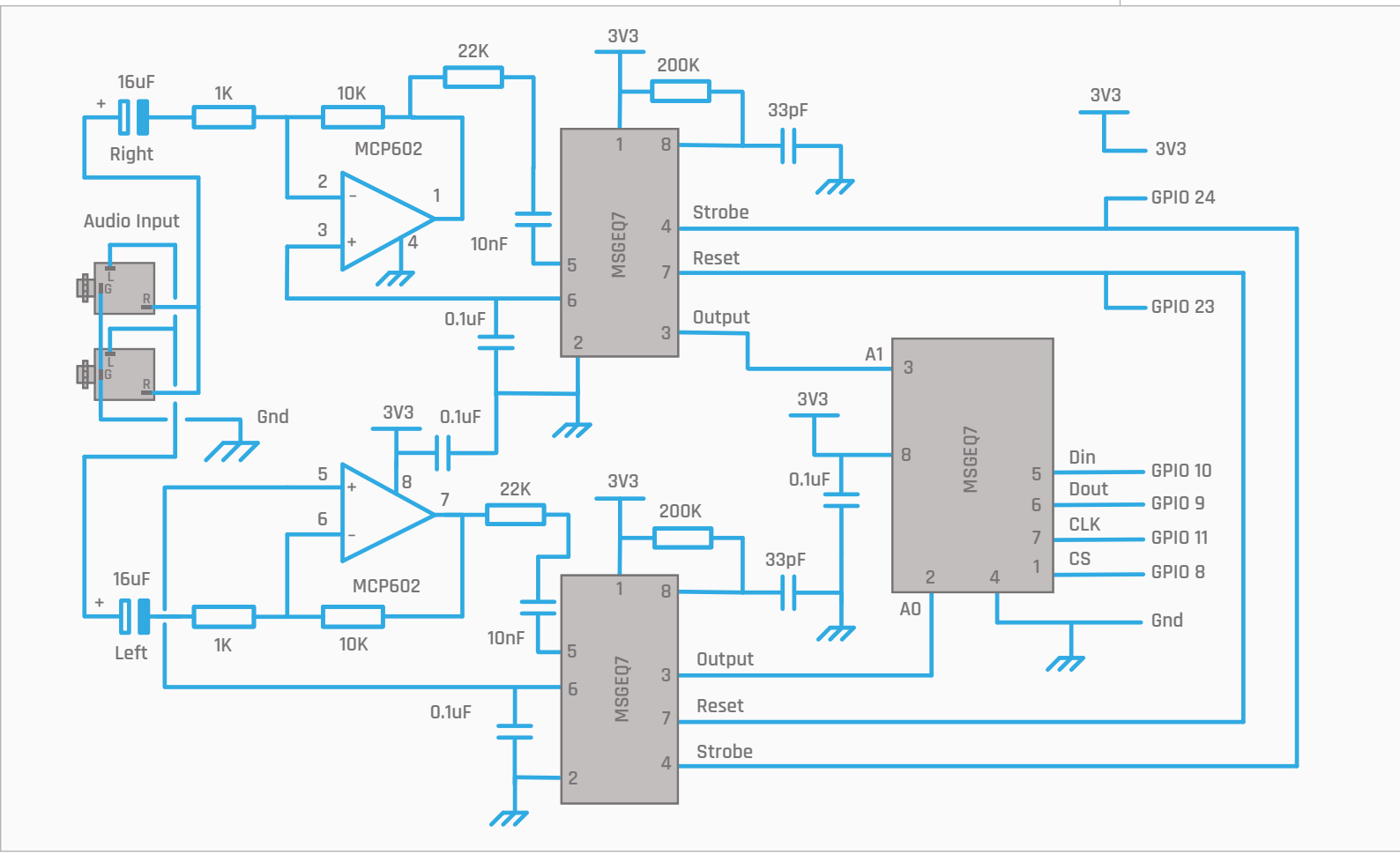


Fig 2 The schematic of the Spectrum board

The hardware

As we wanted a stereo display, we used two of these MSGEQ7 chips, one for left and the other for right. We found that the MSGEQ7 chip required an audio input of about 0.6V PTP (peak-to-peak) to get the full output signal, and while that could be provided by an iPad going absolutely full blast, the audio signal from the Pi itself was not so large. Therefore we decided to put a ×10 amplifier between the audio source and the MSGEQ7. We had two audio jack sockets wired in parallel so the spectrum board could be used in pass-through mode. That is, the audio input was connected to one socket and the other socket could be connected to your headphones or input to your audio amplifier / speaker system. As the MSGEQ7 is capable of being run at 3V, this made the interfacing of the digital clocking signals easy. Each MSGEQ7 is fed into a separate input channel of the MCP3002 A/D converter. The full schematic is shown in **Fig 2**.

The MCP602 has, conveniently, two amplifiers in it with a close to rail-to-rail output, and will work off a single 3V rail. If you are substituting other operational amplifiers here, make sure that they meet that specification. The gain of 10 is given by the 10kΩ resistor between pins 1 & 2 for the right side and pins 6 & 7 for the left side. If you need to increase the gain, just increase this resistor value; for example, a 47kΩ resistor will give a gain of 47 times. As the amplifiers are running off a single rail, you need to generate a virtual ground; luckily this is generated on pin 6 of the MSGEQ7. The 16µF input capacitor value is not very critical, and you could use anything between 10µF and 100µF here.

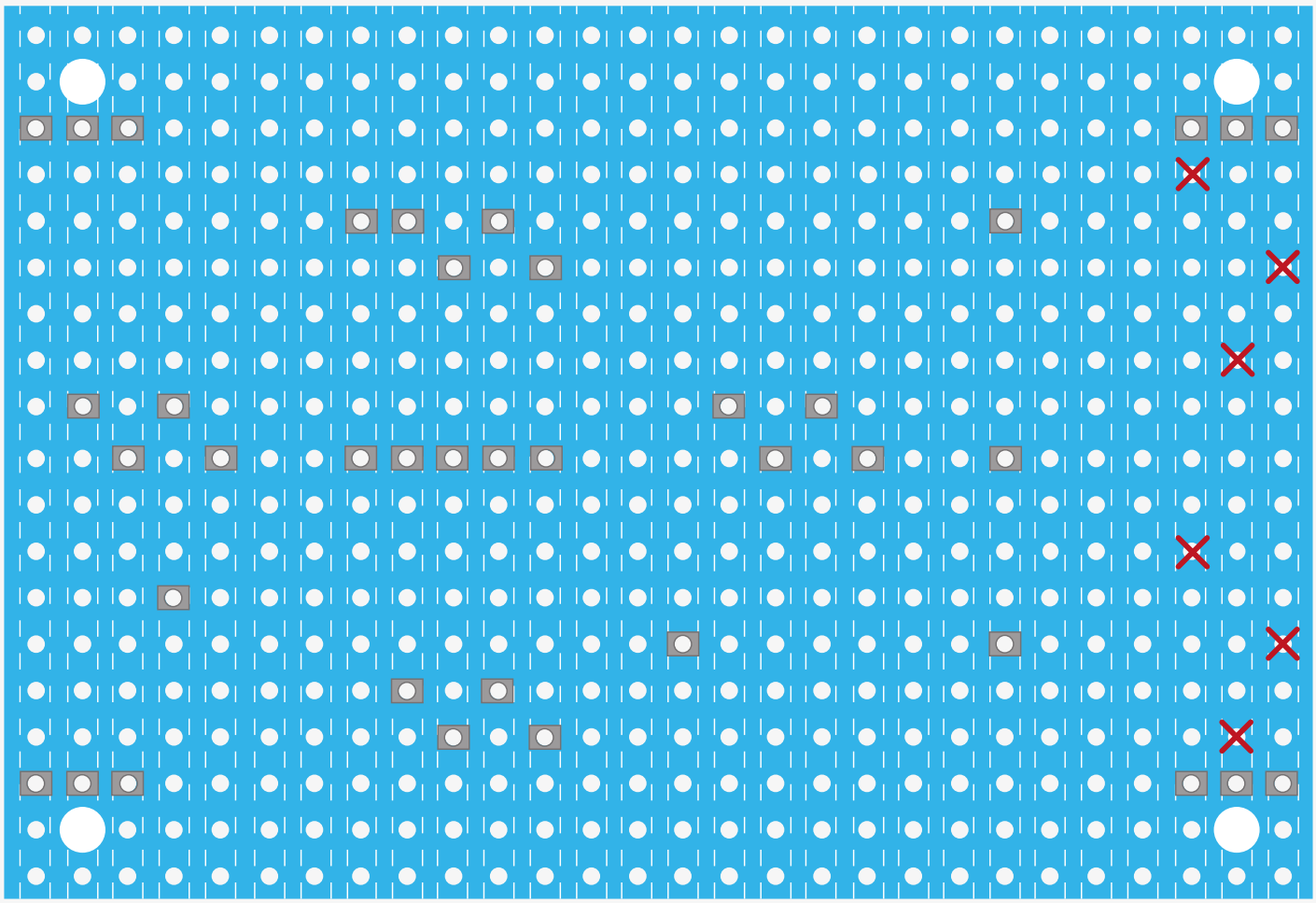
The strobe and reset pins for producing the different filters' output from the chip are directly connected

> As we wanted a stereo display, we used two of these MSGEQ7 chips

to the GPIO pins. The MCP3002 is connected to the SPI bus, just like last month's project. Full construction details are given in the step-by-step section (pages 62–63). Note that if you're trying to adjust the 200kΩ resistors and 33pF capacitors to give you a 165kHz oscillator exactly, you can't just put an oscilloscope on pin 8. This is because the capacitance of most affordable oscilloscope probes will become part of the oscillator circuit and change the frequency. You need to measure that the frequency of a channel peak is correct, to make sure the oscillator is at the correct frequency, but this is not too critical. We would advise you to study the data sheet of this chip.

BUILDING THE SPECTRUM BOARD

Underside of Board



Drill 1mm hole

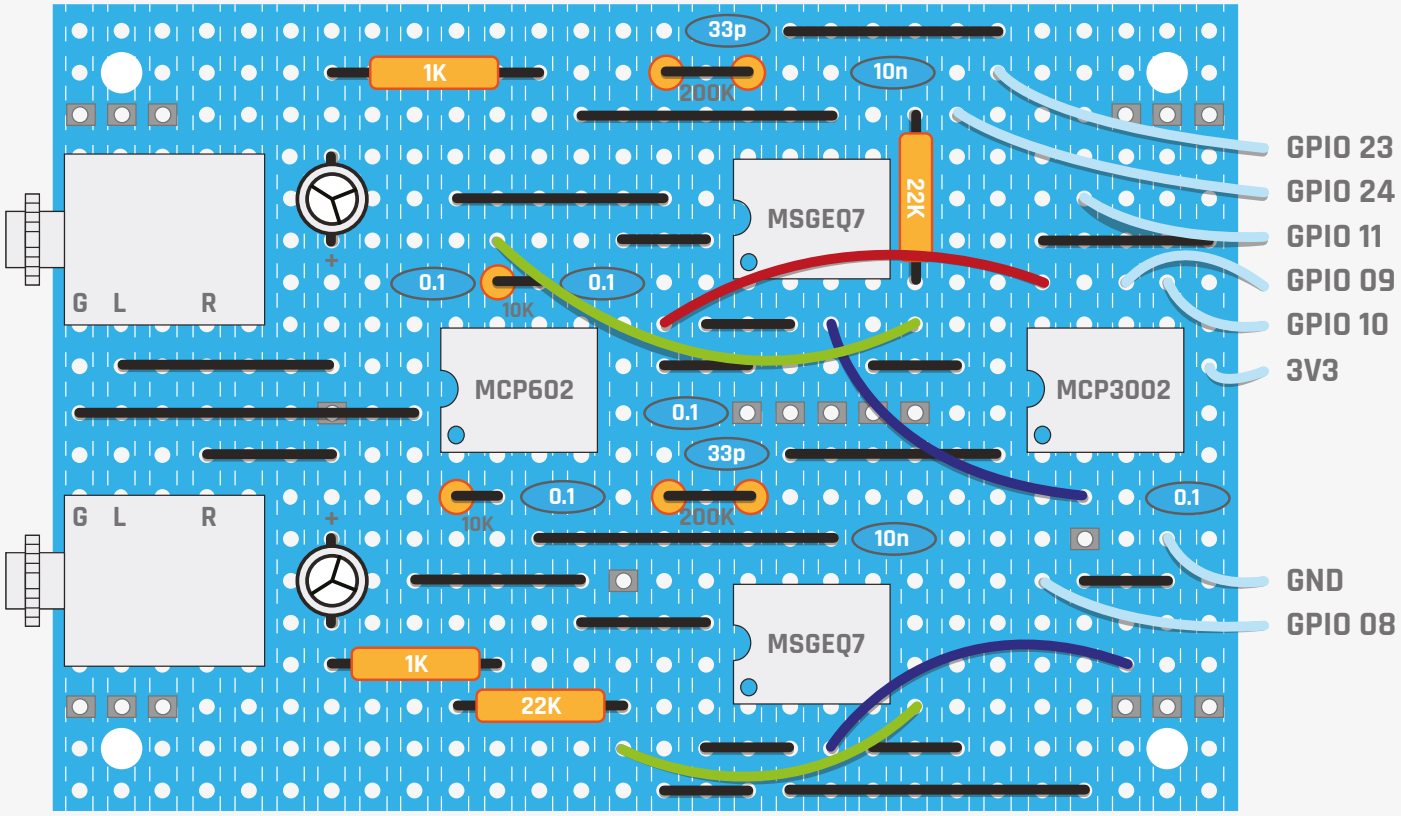
The software

As well as having WiringPi installed, you need to enable the SPI drivers – see last month's project for details if you haven't already done this. The software in the **Spectrum.py** listing (pages 64–5) is written in Python using the Pygame framework. It can run in debug/full-screen mode or in whole-screen mode. The full-screen mode still has the window bar and icon bar visible, whereas whole-screen has nothing but the display. However, any Python error that occurs in the full-screen mode could lock up your system, so set it to debug mode to begin with. When not in the debug mode, the display can be toggled between full-screen and whole-screen with the space bar. This is useful if you're using the Pi as a source of the audio, to allow you to change tracks and make a playlist. We used VLC for this, but there are other audio players available.

>STEP-01

Preparing the stripboard

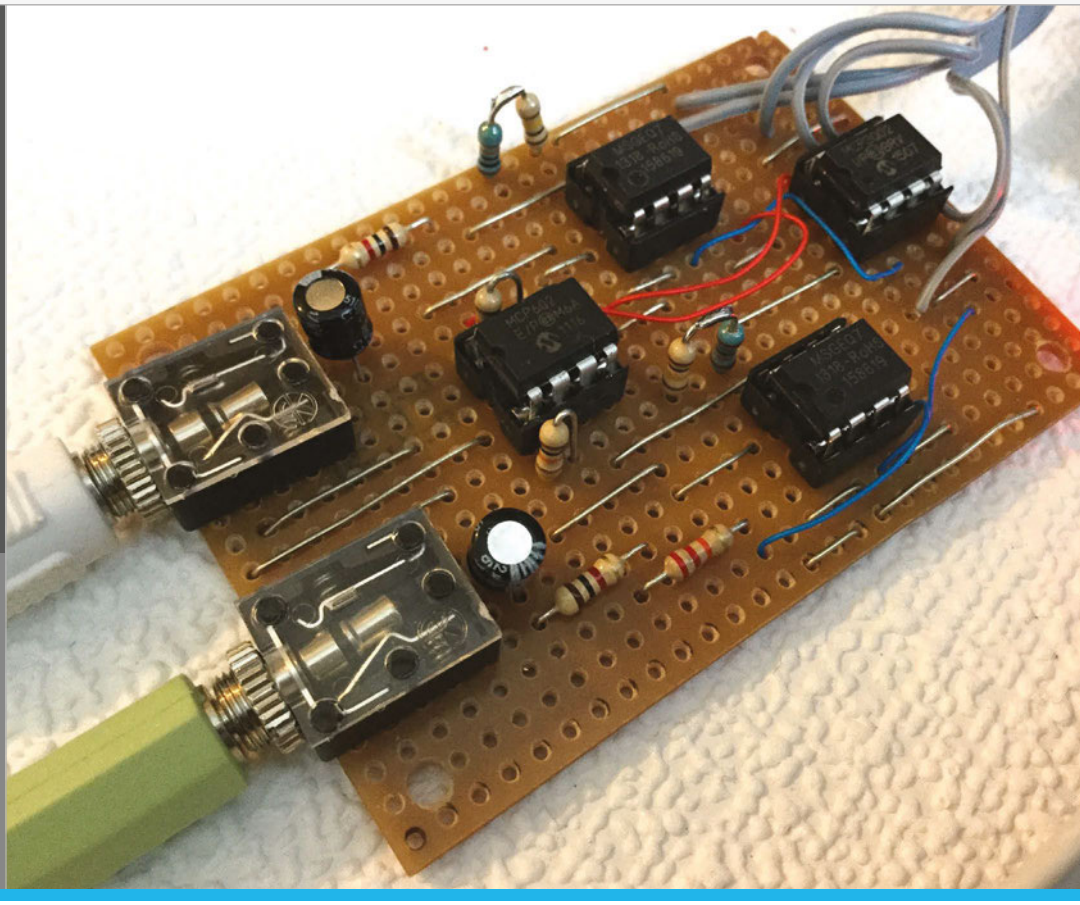
The diagram shows the strip side of the board. Take a piece of 28- by 19-hole stripboard and drill four 3mm holes in each corner, one hole down and in from each corner. Next, drill six 1mm holes where the black holes are, to accommodate the two stereo jack sockets. The position might change if you haven't got the same sort of socket that we used. Finally, cut the strips in the position indicated by the grey rectangles.



>STEP-02

Building the stripboard

This diagram shows the component side of the boards with the strips shown as hidden details. Use solid copper links for the straight jumpers and then add the IC sockets. Fit the resistors and capacitors. Note the ceramic capacitors all fit on adjacent tracks, so you can use surface-mount capacitors on the underside of the board if you want. The 200kΩ resistor was formed by standing up two 100kΩ resistors and connecting the tops; this is because we didn't have any 200kΩ resistors to hand, although you can get away with one here. Finally, place the flexible wires on the board.



>STEP-03

Attaching to the Pi

The photograph shows the finished board. All that remains is to attach it to the GPIO pins of the Pi. There are many ways to do this; we used a 26-way header so that it would fit on any Pi. We used 30cm of eight-way ribbon cable to attach the header to the Spectrum board.

Spectrum.py

```
# Spectrum - Sound visualiser
# By Mike Cook - April 2016

import pygame, time, os, random
from pygame.locals import *
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)]

debug = False
# debug with window bar, display numbers and raw data
screen = pygame.display.set_mode((0, 0))
# with window bar - use for debugging
xs, ys = screen.get_size()
fullScreen = False
if not debug :
    pygame.display.toggle_fullscreen()
    fullScreen = True
else :
    ys -= 66
pygame.display.set_caption("Spectrum Sound Show")

posX = xs / 16
back1 = pygame.Surface(screen.get_size())
errase = pygame.Surface(screen.get_size())

pygame.event.set_allowed(None)
pygame.event.set_allowed([pygame.KEYDOWN,pygame.QUIT])

pinReset = 23
pinClock = 24
random.seed()
leftData = [ 6,7,8,90,80,70,60 ]
rightData = [ 60,70,80,90,80,70,60 ]
colour = [(255,0,0),(255,64,0),(128,255,0),(255,255,0),(
0,114,141),(18,0,237),(255,0,255)]
if debug :
    noiseFloor = [[0,0,0,0,0,0,0],[0,0,0,0,0,0,0]]
else :
    noiseFloor = [[57, 55, 58, 60, 71, 103, 124], [
47, 72, 85, 86, 101, 136, 201]]
rawData = [[0,0,0,0,0,0,0],[0,0,0,0,0,0,0]]

textHeight = 36
font = pygame.font.Font(None, textHeight)

def main():
   initGPIO()
   while True:
     checkForEvent()
     getSpectrumData()
     plotData()

def getSpectrumData():
    # set up the chip to read in the data
    global leftData, rightData
    io.digitalWrite(pinReset,1)
    io.digitalWrite(pinClock,1)
    time.sleep(0.001)
    io.digitalWrite(pinClock,0)
    time.sleep(0.001)
    io.digitalWrite(pinReset,0)
    io.digitalWrite(pinClock,1)
    time.sleep(0.004)
    # now read in each channel in turn
    for s in range(0,7):
       io.digitalWrite(pinClock,0)
       time.sleep(0.004) # allow output to settle
       leftData[s] = scaleReading(io.analogRead(70),s,0)
       rightData[s]= scaleReading(io.analogRead(71),s,1)
```

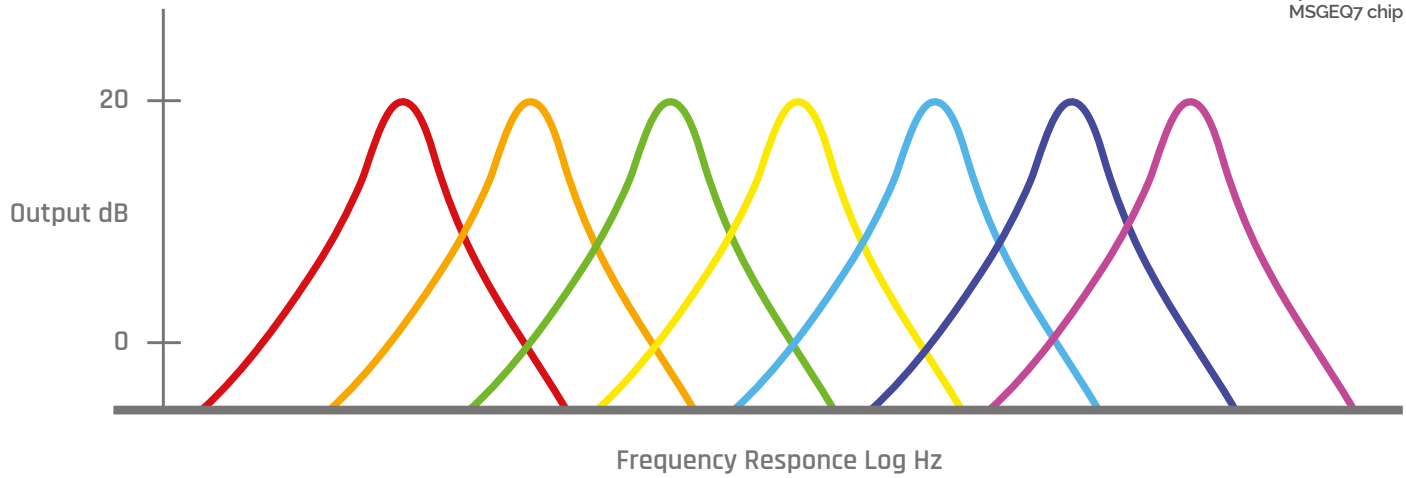


Fig 1 The frequency response of the MSGEQ7 chip

Language
>PYTHON 2.7

DOWNLOAD:
magpi.cc/1NqJjmV

PROJECT VIDEOS
Check out Mike's Bakery videos at:
magpi.cc/1NqJnTz

```
        io.digitalWrite(pinClock,1)

def scaleReading(reading,band,side):
    # adjust for screen size and noise floor
    global rawData
    if debug :
       rawData[side][band] = reading
    reading -= noiseFloor[side][band]
    if reading <0 :
       reading = 0
    scaled = (float(reading) / (1024.0 - float(noiseFloor[side][band]))) * ys
    return int(scaled)

def plotData():
    # display volume bars
    global leftData, rightData
    pygame.draw.rect(screen,(0,0,0),(0,0,xs,ys),0)
    for band in range(0,7):
       pygame.draw.rect(screen,colour[6-band],((band+1)*posX,ys-leftData[6-band],posX-20,ys),0)
       pygame.draw.rect(screen,colour[band],((band+8)*posX,ys-rightData[band],posX-20,ys),0)
       if debug :
          drawWords(str(leftData[6-band]),(band+1)*posX,0)
          drawWords(str(rightData[band]),(band+8)*posX,0)
    pygame.display.update()

def drawWords(words,x,y) :
        textSurface = pygame.Surface((len(words)*12,textHeight))
        textRect = textSurface.get_rect()
        textRect.left = x
        textRect.top = y
        pygame.draw.rect(screen,(81,133,133), (x,y,len(words)*12,textHeight-10), 0)
        textSurface = font.render(words, True, (180,180,180), (81,133,133))
        screen.blit(textSurface, textRect)


def initGPIO():
    try :
       io.wiringPiSetupGpio()
    except :
       print"start IDLE with 'gksudo idle' from command line"
       os._exit(1)
    io.pinMode(pinReset,1)
    io.pinMode(pinClock,1)
    io.mcp3002Setup(70,0)

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

def checkForEvent(): # see if we need to quit
    global fullScreen
    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_SPACE and not debug:
          pygame.display.toggle_fullscreen()
          fullScreen = not fullScreen

# Main program logic:
if __name__ == '__main__':
  try:
     main()
  except:
    if fullScreen :
        pygame.display.toggle_fullscreen()
```

The display we chose to implement is a simple bar display with the left channel as the mirror image of the right one. This means that the lowest frequency is at the centre of the screen, and the highest at the edges. The colour of each bar is set by the colour list of RGB tuples. As the program can be run on any size of window, everything is worked out from the screen size once the full-size window has been created.

With any analogue system there's always noise; the display looks better if no sound equals no display. To do this, there is a **noiseFloor** list of values for each side that's subtracted from the MSGEQ7's readings before display. This is initially set to all zero values, but when you quit from the debug mode, a list of the output values is printed. If you quit with no audio, these values can be pasted directly in the code that sets the **noiseFloor** list for the non-debug mode.

Taking it further

We found it best to use an external sound source such as an iPad, so that you can easily trim the volume to compensate for quiet or noisy music. We found that the best type of music was any with a good dynamic range and beat.

With regards to altering things, the simplest thing you can do is to change the colours of the bars and the way they're displayed. For example, the colours could change as the bars get higher. However, you might want to take it further and change the way the amplitude values are displayed. They could be used to control the drawing of an abstract figure or an animation. The display or the colours could be made to change after a set time. Alternatively, you could drive a string of NeoPixels to generate a bright external display.

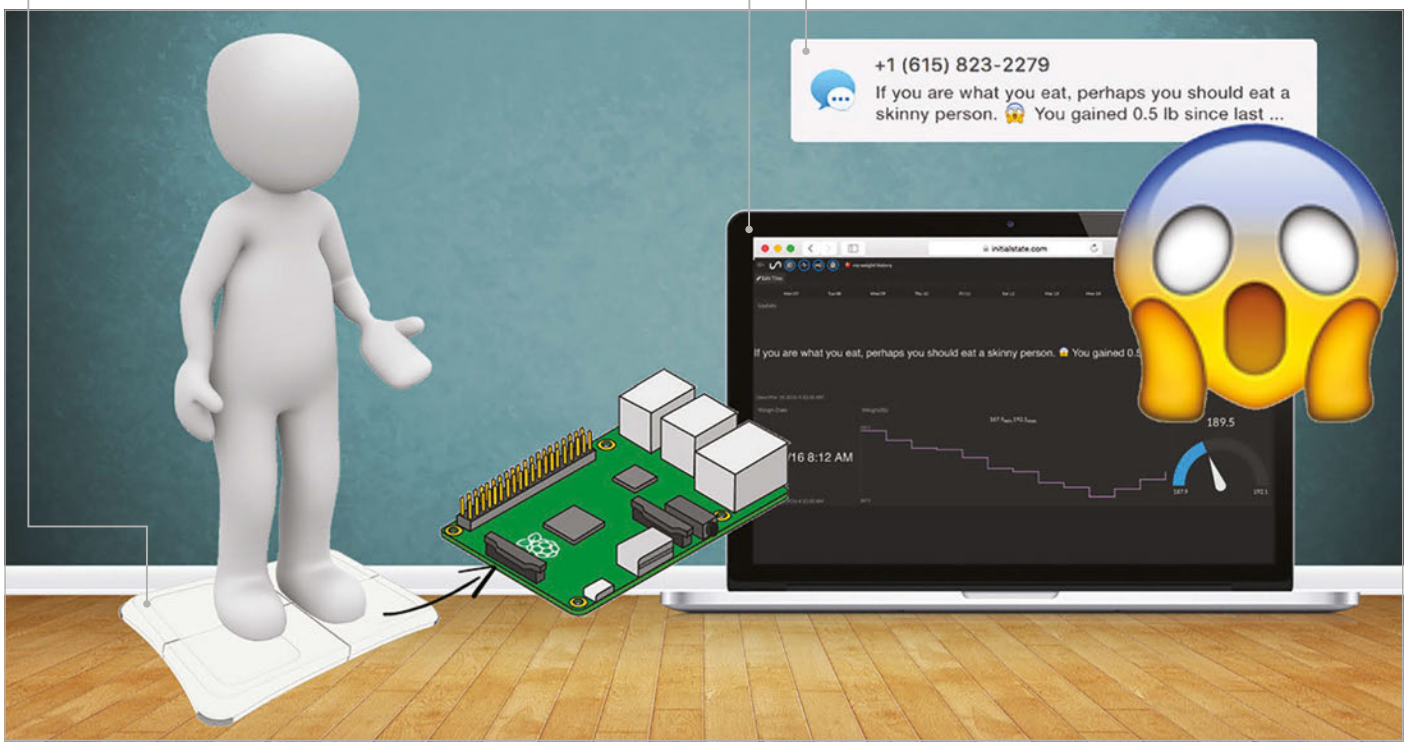
JAMIE BAILEY

Jamie is an electrical/system engineer, integrated circuit designer, and CEO/founder of Initial State, a data analytics service for Internet of Things devices.
initialstate.com

The Wii Balance Board is a Bluetooth scale that can be controlled from a Python script

Weight measurements are sent to your dashboard on Initial State's website

A snarky yet informative SMS is sent every time you weigh yourself



BUILD A WEIGHT-TRACKING WISECRACKING SCALE

You'll Need

- Wii Balance Board
- Wii Fit rechargeable battery pack
- ⅜" Felt pads
- Pencil

Connect a set of scales to the web that tracks your weight and sends you text message updates with an attitude

Oh, that boring, soulless bathroom scale. We love to hate you when you don't show us that number we want. We swear at you, as if you'd care. Why hasn't anyone made a scale that's actually fun to use? It's time to create a scale that's not only smart, but has a bit more personality to brighten your day. We're going to build our very own hackable, weight-tracking, text-messaging bathroom scale that comes with a built-in sense of humour.

Right The Wii Balance Board pairs with your Pi through Bluetooth, making it easy to read your weight with a script

Setting up the Wii Balance Board

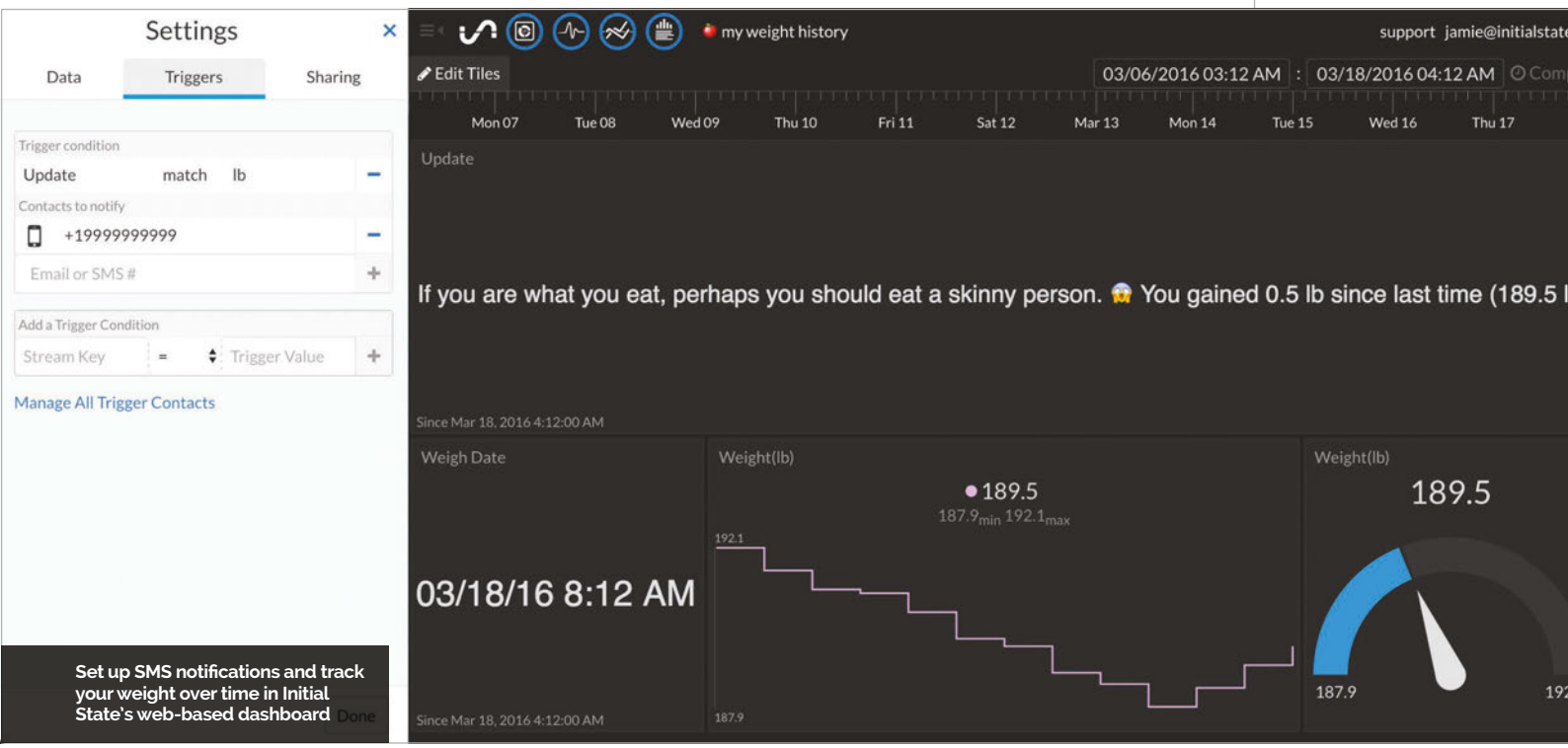
As the Raspberry Pi 3 comes with Bluetooth built in, it makes it very easy to communicate with the Wii Balance Board. If you have a Raspberry Pi 1 or 2, you'll need to use a USB adapter such as the inexpensive iAnder USB Bluetooth 4.0 Low Energy adapter (**magpi.cc/1NqI6mj**).

Power on your Pi and open up a terminal window. You can see the address of your Bluetooth dongle by entering the following command:

```
hcitool dev
```

Install the Bluetooth modules that we will be using in our Python scripts:

```
sudo apt-get install python-bluetooth
```



Set up SMS notifications and track your weight over time in Initial State's web-based dashboard

After installation completes, we're ready to connect and communicate with the Wii Balance Board. We won't be permanently pairing our Board with our Pi, like we do with most of our Bluetooth devices. The Wii Balance Board was never intended to be paired with anything other than a Wii. Pairing will happen every time we run our Python script.

It's time to connect our Wii Balance Board to our Raspberry Pi. We'll do this by running a Python script. To get the scripts we'll use for this project, clone the GitHub repo:

```
cd ~
git clone https://github.com/InitialState/
smart-scale.git
cd smart-scale
ls
```

You should see two Python files in the new **smart-scale** directory: **smartscale.py** and **wiiboard_test.py**. Run the wiiboard_test.py script to test communication and take weight readings from the Wii Balance Board:

```
sudo python wiiboard_test.py
```

You'll see the following response:

```
Discovering board...
Press the red sync button on the board now
```

Remove the battery cover underneath the Wii Balance Board to locate the red sync button. Make sure that you press the button within a few seconds of running the script or a timeout will occur. Once successful, you'll see something similar to the following on rhe screen:

```
  Found Wiiboard at address
00:23:CC:2E:E1:44
  Trying to connect...
  Connected to Wiiboard at address
00:23:CC:2E:E1:44
  Wiiboard connected
  ACK to data write received
  84.9185297 lbs
  84.8826412 lbs
  84.9275927 lbs
```

You have now successfully converted your Wii Balance Board into a Raspberry Pi-connected scale. Now, let's make it a cool scale.

Create a hacky lever to access the sync button on the bottom of the Board, using a pencil and a few ⅜" felt pads

INITIAL STATE

Initial State is an easy-to-use platform for collecting data from connected devices and turning that data into dashboards, waveforms, notifications, and more. **initialstate.com**

Above The Wii Fit rechargeable battery pack allows you to constantly power your scale from a wall outlet, to avoid having to do a Bluetooth sync each time you weigh

Hardware tweaks

Nintendo assumed you would always power your Wii Balance Board with four AA batteries and so included no AC power adapter. Having only battery power would be inconvenient because we cannot permanently pair our Wii Board to our Pi through Bluetooth. We need to sync it, then allow it to remain synced without draining the batteries, so that we can simply step on the scale and weigh. Luckily, there are several third-party adapters made for the Wii Balance Board that we can use to provide constant power from a wall outlet. The Wii Fit Rechargeable Battery Pack (**magpi.cc/1NqIEIT**) is a perfect solution to our power problem. Replace the batteries with this battery pack, and plug the AC adapter into a wall outlet.

Having to pair the Wii Balance Board and Raspberry Pi every time we run our Python script presents another inconvenience due to the location of the sync button. The sync button is at the bottom of the Wii Board, which means we would have to flip it over every time we sync. We can fix this by making a hacky little lever using a pencil and three ⅜″ felt pads. The rechargeable battery pack exposes the sync button to the underneath surface of the Board. Tape a pencil (or something similar) that runs from the sync button to the outside front of the Board. Stack three ⅜″ felt pads (or something similar) on the centre(ish) of the pencil to create a stationary pivot. Flip the Board over and you can press the sync button by simply pressing down on the lever. Hacky but effective.

ADD YOUR OWN HACKS

You can modify the messages that get sent, the units used for measurement, and even add your progress toward your own weight loss goals by modifying the existing Python script.

Initial State

We want to stream our weight/data to a cloud service and have that service turn our data into a nice dashboard that we can access from our laptop or mobile device. Our data needs a destination; we'll use Initial State as that destination.

Go to **magpi.cc/1TFlhaz** and create a new account, then install the Initial State Python module onto your Pi:

```
cd ~
\curl -sSL https://get.initialstate.com/
python -o - | sudo bash
```

When prompted to automatically get an example script, type **Y**. This will create a test script that we can run to ensure that we can stream data to Initial State from our Pi. Run the test script to make sure we can create a data stream to your Initial State account:

```
python is_example.py
```

Go back to your Initial State account in your web browser. A new data bucket called 'Python Stream Example' should have shown up on the left in your bucket shelf. Click on this bucket and view it in the visualisation apps Tiles, Waves, and Lines.

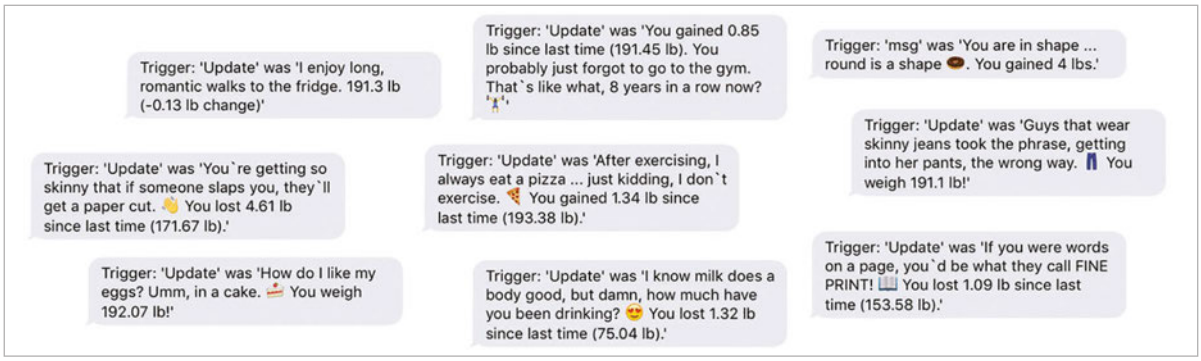
You are now ready to start streaming real data from your scale.

The final script

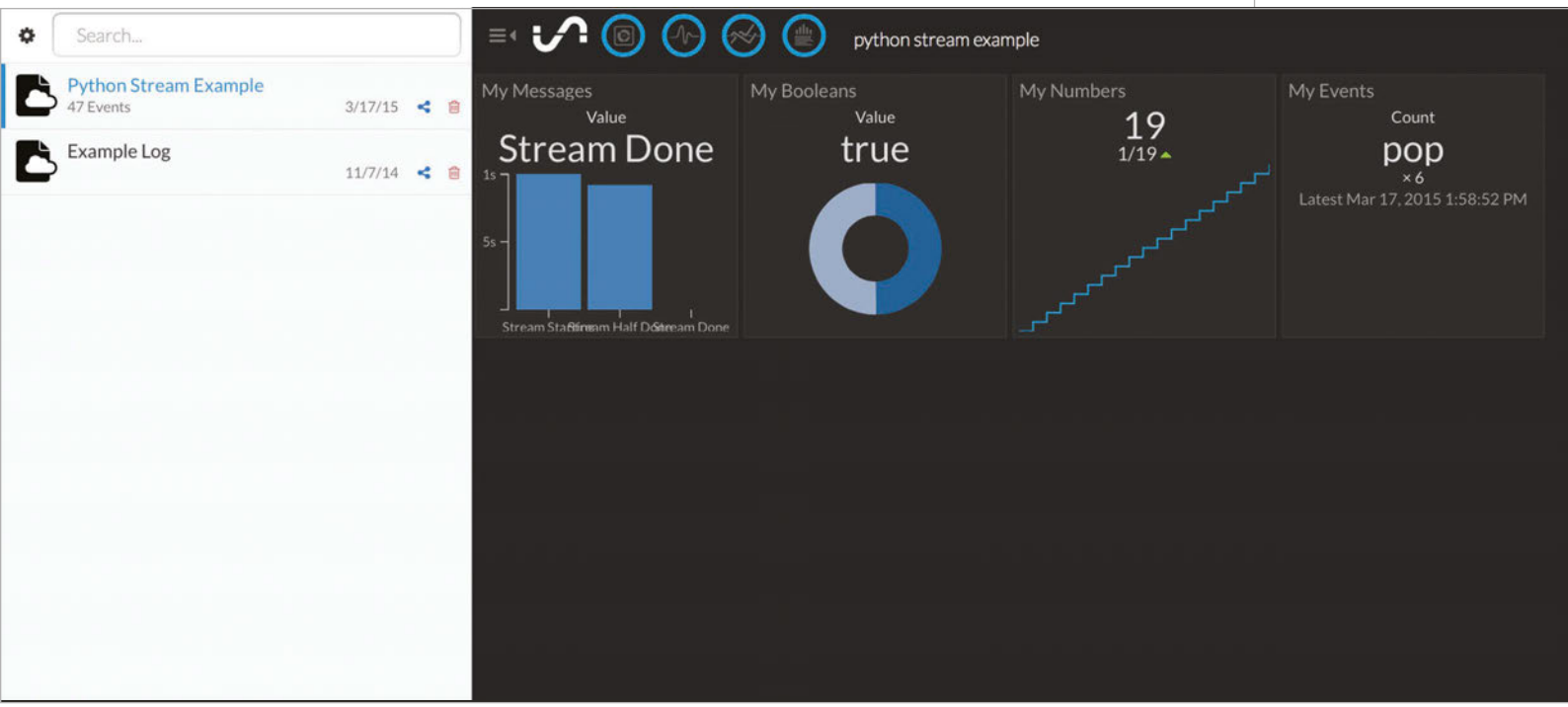
The final script that puts everything together is called **smartscale.py**, in your **~/smart-scale** directory. A few settings need to be set in the script before you can run it. Open up **smartscale.py** in your favorite text editor, such as nano:

```
cd ~
cd smart-scale
nano smartscale.py
```

Near the top of this file, there is a User Settings section. These are all of the settings that you can tweak to your taste. The only setting you have to set is your ACCESS_KEY, which is your Initial State account key. If you don't put your ACCESS_KEY in this field, your data won't show up in your account. Go to your



Right Receive a variety of funny, inspiring, and insulting SMS messages from your smart scale



Above Run a test script from your Pi to make sure you can create a data stream to your Initial State account

> Luckily, there are several third-party adapters made for the Wii Balance Board that we can use to provide constant power from a wall outlet

Initial State account in your web browser, click on your user name in the top-right, then go to 'My Account'. You'll find your access key at the bottom of the page, under 'Streaming Access Keys'.

Once you have specified each parameter in this section and saved your changes, you're ready to run the final script:

```
sudo python smartscale.py
```

Step on the scale to take a measurement. Go to your Initial State account and click on the new data bucket with the name corresponding to the BUCKET_NAME parameter (i.e. My Weight History). Click on Tiles to view your weight history dashboard. You can customise your dashboard by resizing and moving tiles, as well as changing view types and even adding tiles.

SMS setup

Let's create a SMS alert whenever the scale takes a weight measurement. First, make sure your weight history data bucket is loaded, then click on the bucket's settings in the data shelf (under its name). Click on the Triggers tab; if you don't see a Triggers tab, make sure you're subscribed to an Initial State account plan that allows Triggers.

Select the data stream to trigger on; you can use the drop-down list to select from existing streams once a data bucket has loaded, or you can type in the stream name/key manually. Select the 'Update' data stream and then select the conditional operator, in this case 'match'. Select the Trigger value that will trigger an action (manually type in the desired value). Type in **lb** if you are using imperial units, or type in **kg** if you are using metric units. Whenever the Update stream contains 'lb' (or 'kg'), you will get a text message notification.

Next, click the '+' button to add the Trigger condition. Select the action ('notify by SMS') and then click the '+' button to add the action. Input any verification code, if adding a new phone number, to complete setup.

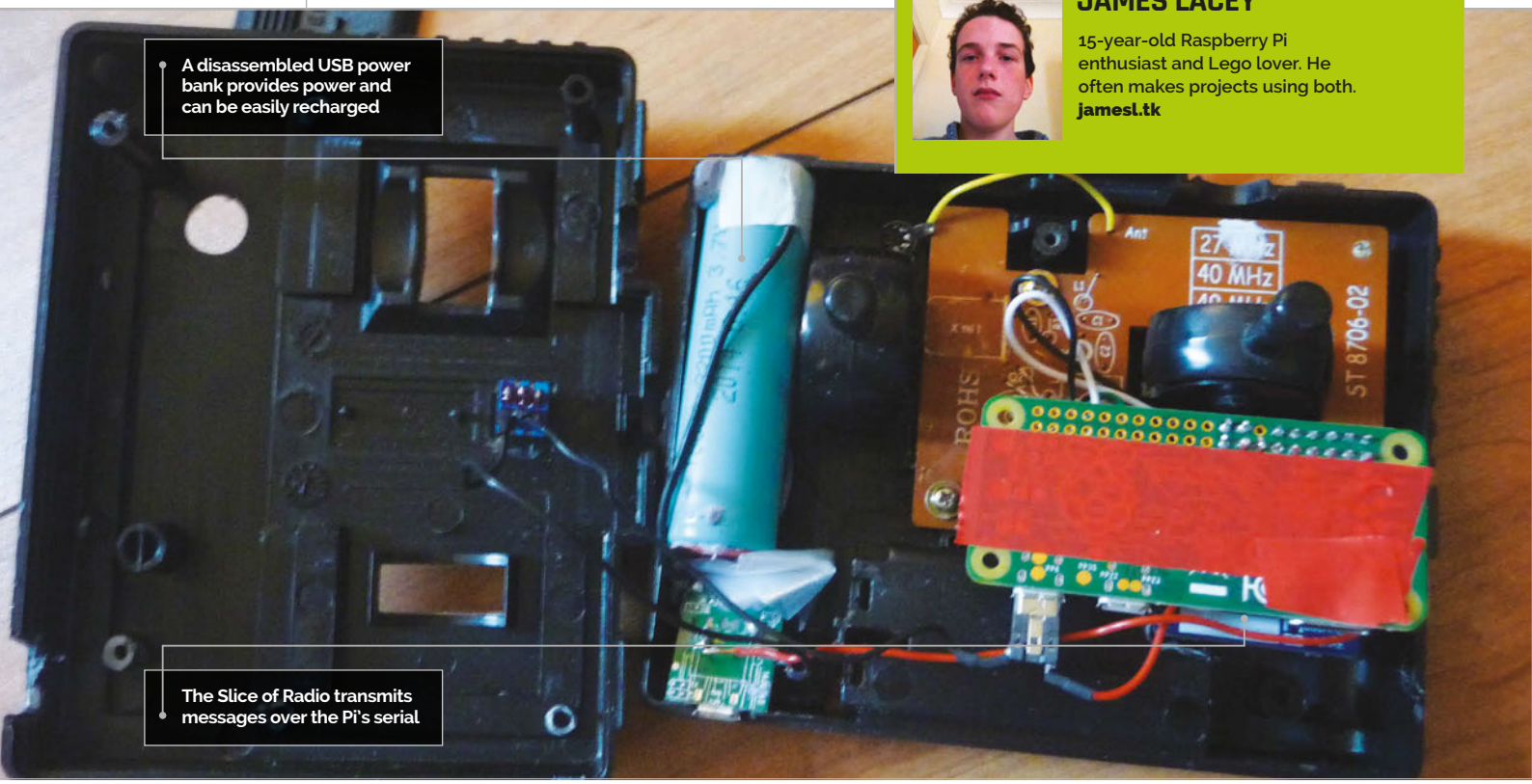
Your trigger is now live and will fire when the condition is met. Click Done to return to the main screen. Every time you weigh yourself, you will get an SMS that contains your weight, how much it has changed since the last measurement, and a random joke, insult or compliment.

Customisation

Not a fan of the humour built into this project? You can change the jokes in the `messageWeighFirst`, `messageWeighLess`, `messageWeighMore`, and `messageWeighSame` functions to whatever wacky brand of humour you have. You can also stream data from other sources into the same weight history dashboard to create your own personal health dashboard. Hack away.

ADD MORE DATA

You can stream data from other sources into the same weight-tracking dashboard to create your own personal health dashboard. Learn more at **magpi.cc/1NqK8mp**



JAMES LACEY

15-year-old Raspberry Pi enthusiast and Lego lover. He often makes projects using both.
jamesl.tk

HACK A REMOTE CONTROL WITH PI ZERO

Control an Arduino-powered custom Lego car with a simple remote control and a Raspberry Pi Zero

You'll Need

- Large RC car controller
- Slice of Radio **magpi.cc/1W7Sphx**
- XinoRF **magpi.cc/1QSdoyo**
- USB power bank **magpi.cc/1W7UZE6**
- Toggle switch **magpi.cc/1W7VPkl**
- Hook-up wire
- Micro USB cable

Ever wanted to use a Raspberry Pi to remotely control an Arduino, but using something smaller and more portable than a keyboard? Here we'll show you how to add a Pi Zero and wireless radio to a controller from an RC toy, turning it into a versatile and customisable project controller!

>STEP-01

Prepare the controller

Open up the controller by removing any screws and plastic retention clips. Locate and remove the main PCB of the controller. You will see contacts on the PCB, which the joysticks or buttons on the controller touch to send signals to the integrated circuit (IC) on the board. Using a multimeter, you should be able to follow each contact to a pin on the main IC. On this controller there are four contacts, so four IC pins should be found and labelled.

>STEP-02

Remove the main IC

The IC will probably be soldered to the PCB, so cut off all the legs with pliers and desolder all the legs from the board. Now is also the time to remove other large components from the PCB that may interfere with the placement of the Pi Zero. Remembering which pin hole connects to which IC, solder differently coloured wires to each hole and the other half of the contacts (which are usually connected via a common trace). It's now time to prepare the Pi Zero.

>STEP-03

Preparing the Raspberry Pi

Solder on only the first ten pins of a male GPIO header to the Pi Zero, so the serial pins are populated with connectors. Leave the rest of the GPIO holes unpopulated so you can solder on the pin connectors.

PREVENT SHORT CIRCUITS

Cover the Pi Zero and any exposed metal of the controller in tape to prevent short circuits.

Prepare a microSD card with Raspbian Jessie Lite on it, then connect to an HDMI monitor, mains power, and a keyboard to run raspi-config. Next, get the code from the GitHub repository (**magpi.cc/1QSep84**) and place it in your home directory.

>STEP-04

Wiring up the Pi

If you are using the code without alterations, solder the contact for the forward position to GPIO 27, the backward contact to GPIO 22, the right contact to GPIO 23, and the left contact to GPIO 24. Connect the other half of each contact to a 3.3V pin. Bend over the wires so they are as flush with the PCB as possible, so you can still attach the Slice of Radio transceiver. Turn on the Pi and run the code; the HDMI monitor should display no signal, and the LEDs on the Slice of Radio should flash when you move the levers.

>STEP-05

Running the code at boot

Edit the **/boot/cmdline.txt** file to include the line **"init=/home/pi/filename-of-code.sh"**. Verify that the code is correct and **chmod +x** the script. This is your last chance to change anything – the Pi will not boot to a terminal again without manual editing of **cmdline.txt**. Disconnect your HDMI monitor and keyboard, then reboot the Pi Zero. The LEDs on the Slice of Radio should flash to show that data is being transmitted.

>STEP-06

Fitting the Pi in the controller

You may have to use a Dremel tool or wirecutters/pliers to remove parts of the internal structure of the controller, so that the Pi fits. We had to remove most of the battery compartment. After fitting, check that all the wires are still securely attached to the Pi and the controller. Test the controller again, ensuring the LEDs flash. You've completed most of the controller, but if you have sufficient space available, continue to the next step to get rid of that annoying power cable.

>STEP-07

Fitting the battery

Prise the two halves of the power bank apart with a knife or screwdriver. You should see a cylindrical battery and a small PCB with a USB and micro USB socket. Remove these from the housing and desolder the large USB socket from the board. Using a multimeter, find and label the positive pad that the USB socket was soldered to. Cut off the large USB plug from the micro USB cable, then solder the red wire to the positive pad and the black wire to a terminal of the switch. Connect the other terminal to the negative pad on the PCB. Make a hole in the controller for the micro USB connector and glue it in place. You're done!

control.sh

Language
>BASH

DOWNLOAD:
magpi.cc/1QSep84

```
#!/bin/bash
# put this code in /etc/rc.local to execute at boot
mount /dev/mmcblk0p2 / -o remount,ro # prevent SD
card corruption by mounting read-only
mount proc /proc -t proc # as this is executed just
after the kernel has finished booting, /proc doesn't exist. This creates it.
tvservice -o # Turn off HDMI
stty -F /dev/ttyAMA0 9600 # set the baudrate of the Slice of Radio
gpio -g mode 27 down # configure internal pull ups
gpio -g mode 22 down
gpio -g mode 23 down
gpio -g mode 24 down
TTY=/dev/ttyAMA0
while true; do
A=0
B=0
C=0
D=0
[[ $(gpio -g read 27) == 1 ]] && A=1 # forward
[[ $(gpio -g read 22) == 1 ]] && B=2 # backward
[[ $(gpio -g read 23) == 1 ]] && C=4 # right
[[ $(gpio -g read 24) == 1 ]] && D=8 # left
SWSTATE=$(($A+$B+$C+$D))
case $SWSTATE in
0)
echo "aAASTOP-----" > $TTY # stop and center, as there is no input on any switch
echo "aAACENTER---" > $TTY
sleep 0.5
;;
1)
echo "aAAFORWARD--" > $TTY
echo "aAACENTER---" > $TTY
;;
2)
echo "aAABACKWARD-" > $TTY
echo "aAACENTER---" > $TTY
;;
4)
echo "aAASTOP-----" > $TTY
echo "aAARIGHT----" > $TTY
;;
5)
echo "aAAFORWARD--" > $TTY
echo "aAARIGHT----" > $TTY
;;
6)
echo "aAABACKWARD-" > $TTY
echo "aAARIGHT----" > $TTY
;;
8)
echo "aAASTOP-----" > $TTY
echo "aAALEFT-----" > $TTY
;;
9)
echo "aAAFORWARD--" > $TTY
echo "aAALEFT-----" > $TTY
;;
10)
echo "aAABACKWARD-" > $TTY
echo "aAALEFT-----" > $TTY
;;
*)
echo "DEBUG: $SWSTATE" # print to a debug terminal if
there is an unknown option
;;
esac
done
```

STRENGTHEN WEAK SOLDER JOINTS

Use glue to strengthen weak solder joints that may break off; use the low setting on a hot glue gun.

SET UP & USE THE RASPBERRY PI ORACLE WEATHER STATION

The new weather station kit is making its way to schools – here's how to set it up, plus a taste of what you can do with it...

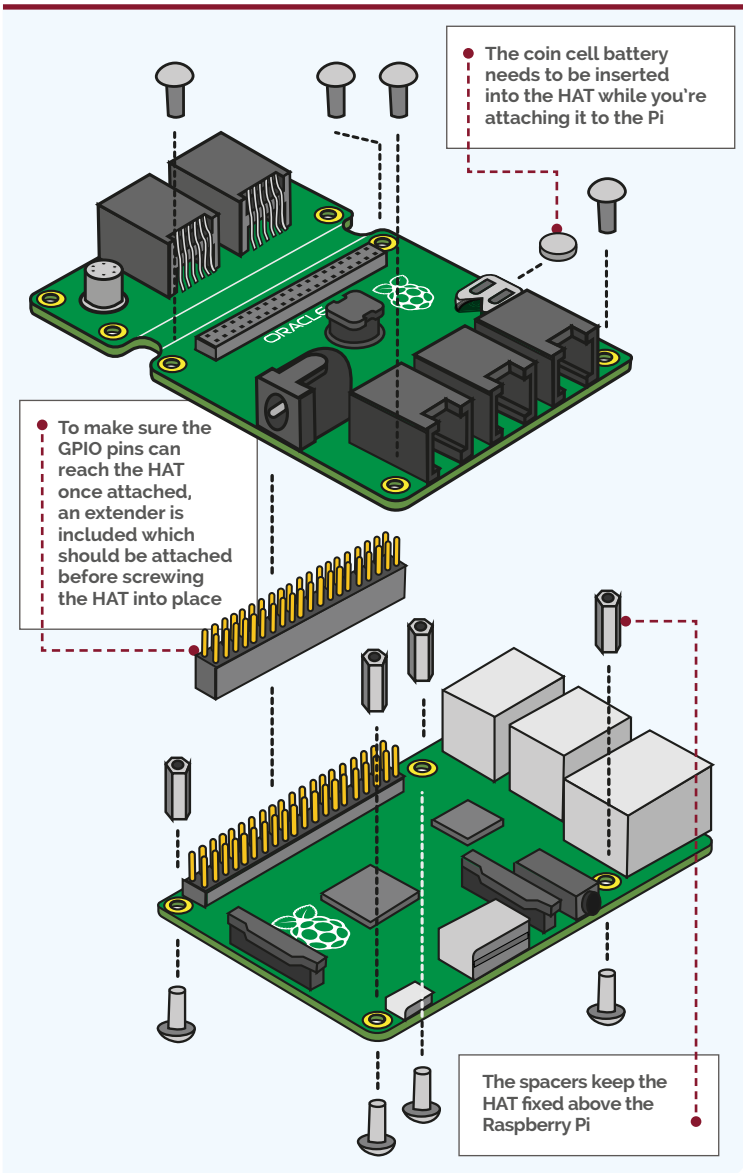
Last month in *The MagPi*, we reported on how the long-awaited Raspberry Pi Oracle weather station was making its way to schools, with an initial shipment going out to a select few in the UK. Since then, more have been sent out and more schools have been receiving them, ready to do some weather science using the power of a little Python code.

We've put together a little guide based on the information freely available from the Raspberry Pi resources website (**magpi.cc/1qEg9Nh**), to help you build your station and then use it to take some measurements. In the future, hopefully, more people – not just schools – will be able to take advantage of the educational potential of the weather station kit.

BUILD YOUR WEATHER STATION

ALTERNATE BUILD INSTRUCTIONS can be found online: magpi.cc/1YprQSO

Grab a screwdriver and let's get constructing!

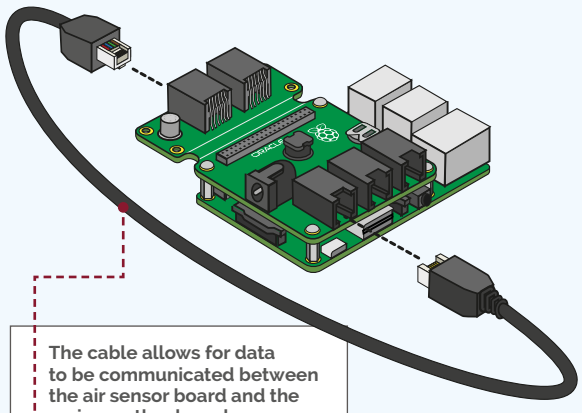


01 - ATTACH THE HAT

The weather station is made up primarily of a HAT (which is two boards that can split up if needed) and sensors. We begin by attaching the HAT to the Raspberry Pi; first, add the GPIO extender and then screw in the spacers on the Raspberry Pi, as shown in the image. Once they're secure, align the HAT on top in the orientation shown and then screw it to the spacers. Finally, insert the battery.

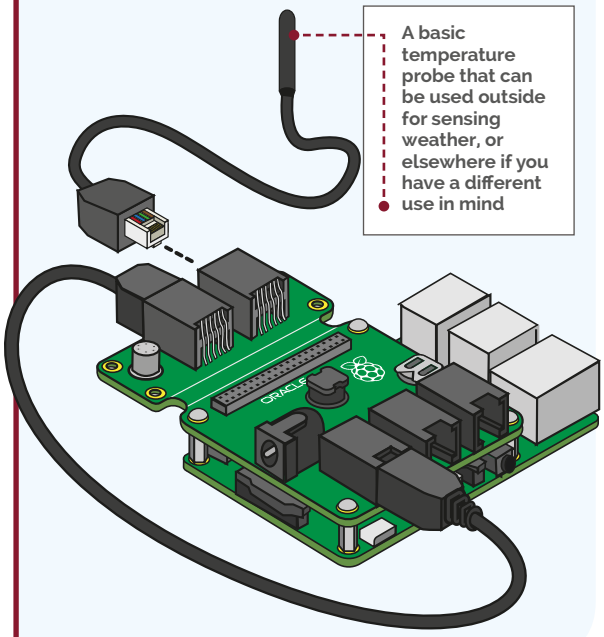
02 - CONNECTING THE BOARDS

The weather station HAT is made up of two pieces: the weather board, which is the main HAT, and the air sensor board. They're not actually connected by any circuits, because you can have them separate for specific uses. So, to get them to communicate, you need to use an RJ11 cable and insert it into the ports shown in the image.



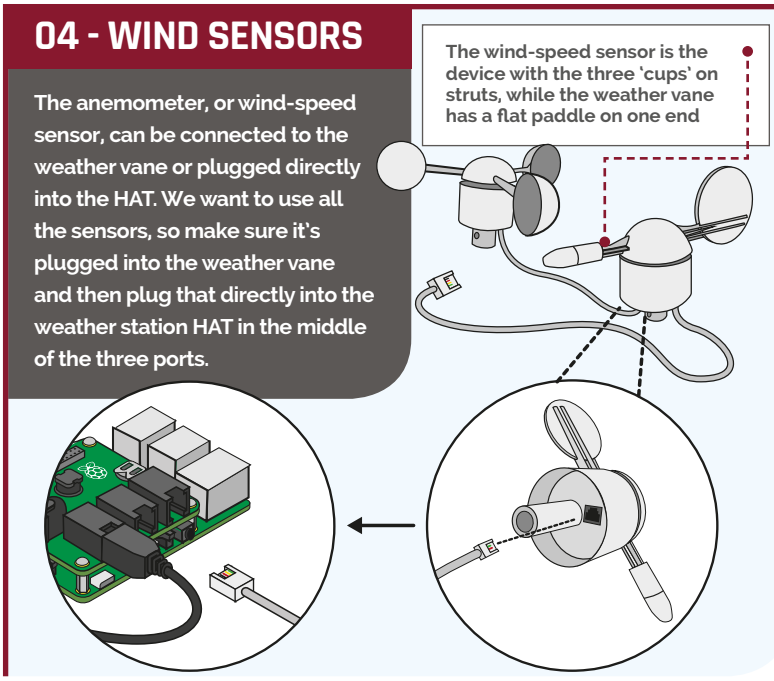
03 - SENSING TEMPERATURE

The temperature sensor is attached to the air sensor portion of the board via the remaining port. The cable allows it to be placed away from the Raspberry Pi.



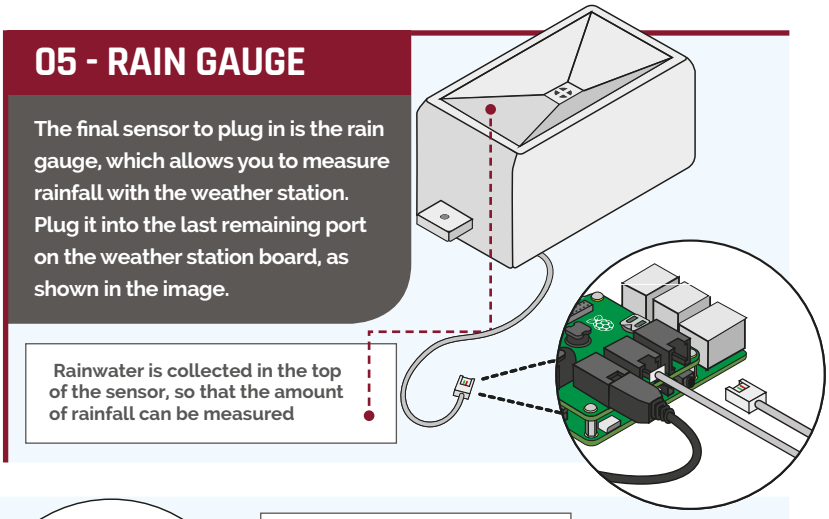
04 - WIND SENSORS

The anemometer, or wind-speed sensor, can be connected to the weather vane or plugged directly into the HAT. We want to use all the sensors, so make sure it's plugged into the weather vane and then plug that directly into the weather station HAT in the middle of the three ports.



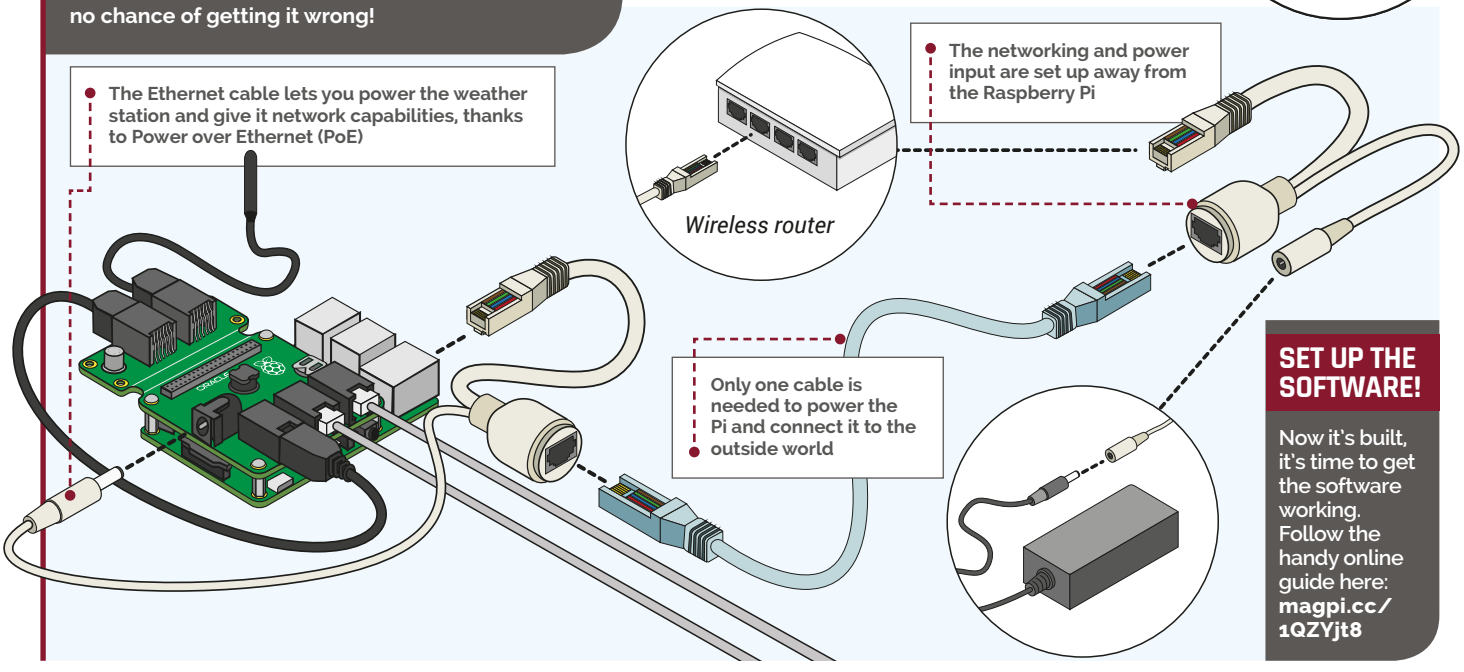
05 - RAIN GAUGE

The final sensor to plug in is the rain gauge, which allows you to measure rainfall with the weather station. Plug it into the last remaining port on the weather station board, as shown in the image.



06 - POWER AND NETWORKING

Now that the weather station is constructed, you can provide it power and networking; find a spot where the cables will reach both power and a router and attach the final cables as shown in the diagram. The splitter cables can only go one way round, so there's no chance of getting it wrong!



SET UP THE SOFTWARE!

Now it's built, it's time to get the software working. Follow the handy online guide here: **magpi.cc/1QZYjt8**

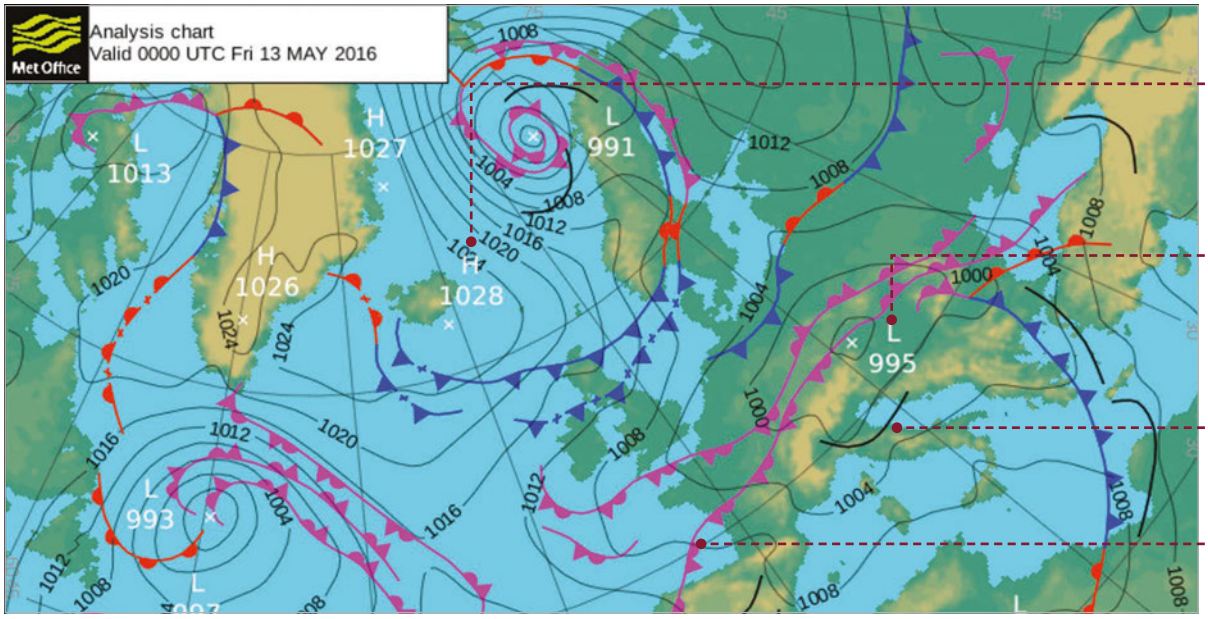


Image from the Met Office

FORECASTING THE WEATHER

How exactly can the readings help figure out upcoming weather?

All of us at some point have seen a TV weather forecast, but what does it all mean? The BBC have an excellent video that helps to explain the terminology of weather forecasts, and how the forecasts are made in the first place, which you can see here: **youtu.be/lITCF3UPVu4**.

As you can see from the video, the weather forecast is determined by high pressure, low pressure, isobars, and fronts. High pressure is generally associated with good weather, which is defined as the air getting heavier and denser, creating higher air pressure. The air dries under high pressure; this usually occurs in the summer.

Low pressure is the opposite: the air rises and draws in water vapour which then creates clouds. Clouds mean rain, eventually, and clouds also block the sun, meaning its heating effect is slightly negated.

Pressure maps for the weather are usually also represented with lines; these are the isobars and they determine the wind strength. If isobars are close together, it means strong winds; the further apart they are, the gentler the breeze is.

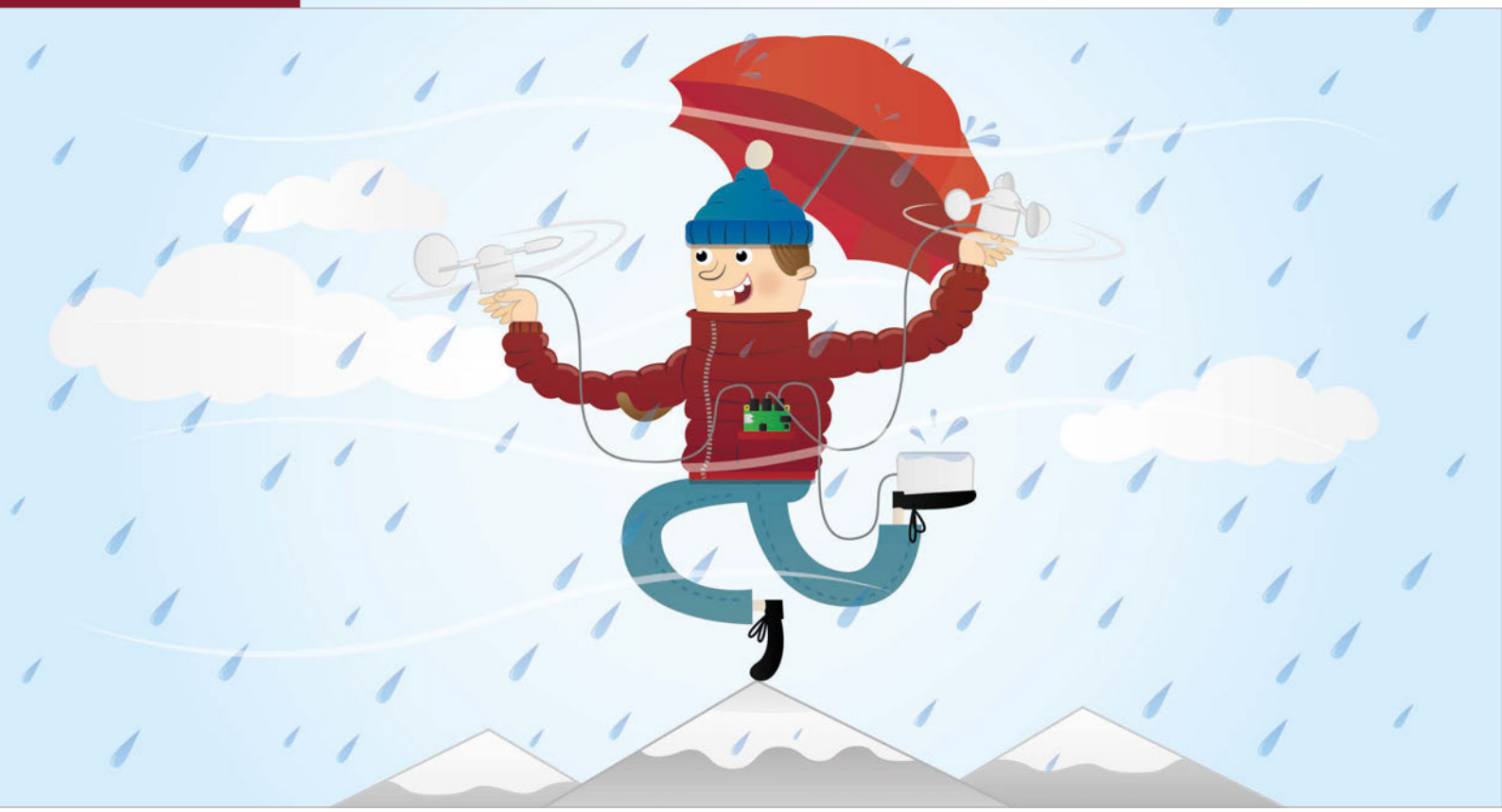
Finally, fronts are the invisible lines that determine when two different masses of air meet together. The air can have different densities on each side, and be cooler or warmer. Fronts usually result in rain or storms, depending on their intensity.

How to make predictions

A weather forecast is a useful prediction of future conditions, but how are these predictions made? There are two key steps to predicting the weather.

First is data collection. Vast quantities of data are collected about current conditions all over the world. This is done using satellites, ground-based observers, and weather stations like the kit you're using. They gather data on temperature, pressure, wind speed, cloud movement, and other little bits of information.

Once data is collected, it's then modelled. This computer model, which knows the state of the weather at various points, then predicts the path of weather fronts and systems. These models are not always completely accurate, especially when you've decided to have a picnic or go to the beach. The further ahead in time the prediction is, the less reliable it becomes. However, prediction models are becoming ever more sophisticated and, with the help of your experiments using the weather station, may become more accurate still.



MEASURING WIND SPEED

Get started using the weather station by finding out the current wind speed

Let's use the anemometer, or wind speed sensor, to collect data about wind speed. The sensor has three arms with buckets on the end which 'catch' the wind, causing the arms to spin. If you were to dismantle the sensor, you would find a small magnet attached to the underside.

At two points within the magnet's rotation, it triggers a reed switch which produces a LOW signal we can detect. So, for each full rotation of the arms, the sensor will produce two detectable signals.

Detecting the signal

Before we begin calculating wind speed, we need to be able to count the signals coming from the anemometer. Set up your Raspberry Pi and ensure you are in desktop mode, then launch a terminal window.

Create a weather station directory by typing **mkdir weather_station** and pressing **ENTER**. Once that's done, open up Python by typing **sudo idle3**, and add the following code...

```
#!/usr/bin/python3
import RPi.GPIO as GPIO

pin = 5
count = 0

def spin(channel):
    global count
    count = count + 1
    print (count)

    GPIO.setmode(GPIO.BCM)
    GPIO.setup(pin, GPIO.IN, GPIO.PUD_UP)
    GPIO.add_event_detect(
pin, GPIO.FALLING, callback=spin)

    input("Press enter to stop logging\n")
```

Now you can test your code! Press **F5** and save when prompted. Your code should display the number of half-rotations counted. Press **CTRL+C** to stop the program.

We can now count the signals from the anemometer; next, we need to calculate the wind speed.

Calculating wind speed

We can count the number of rotations of the sensor by doubling the detected inputs. But how do we change that into a speed figure? Let's start by considering the formula for calculating speed:

Speed = Distance / Time

Imagine we counted the number of signals over the course of 5 seconds. We now have the time, but we also need distance. The distance travelled by a cup will be equal to the number of revolutions multiplied by the distance around the edge of the circle (circumference). So we could write:

Speed = Revolutions Circumference / Time

The circumference can be calculated if we know either the radius or diameter of the circle. We can measure the radius of the circle made by the anemometer by measuring the distance from the centre to the edge. Knowing the radius, we can find the circumference with $2 \times \pi \times r$. We also know that the revolutions are half the number of signals detected, so our formula becomes:

Speed = (Signals/2) × (2 × π × r) / Time

This formula should enable us to calulate the speed of the wind in cm/s (centimetres per second).

Updating the code

Now that we're able to calculate the wind speed from the information we can collect, we need to add the code to make this work. We need to measure the radius (in cm) of the anemometer for use in the program, and decide on the time interval for calculating average wind speed (at least 5 seconds).

Add to the code using the listing (on the right) as a guide – or download it – and save it as **windspeed.py**.

Measurement units

Currently, the program we have created will measure the wind speed in cm per second. However, this isn't particularly useful. A more practical unit would be km per hour. In order to convert our units, we'll need to convert cm to km and convert the seconds into hours. Work out how many cms are in a km, and seconds in an hour, and change the code accordingly.

Calibration

The program should now display the wind speed in km/h, but is it accurate? The info for the anemometer says that if it rotates once a second, that should equate to 2.4 km/h. So, in the example interval of 5 seconds, 5 spins (10 signals) should equal 2.4 km/h.

Run your program and spin the anemometer 5 times within the first 5 seconds. What wind speed value is reported? You should see something like:

```
1
2
3
4
5
6
7
8
9
10
2.03575203953 kph
```

That's not quite right! This loss of accuracy is due to something called the anemometer factor, and is a result of some of the wind energy being lost in turning the arms. To compensate for this, we are going to have to multiply the reading generated by our program by a factor of 1.18, which should correct this error. Update the final line in the **calculate_speed** function to read:

```
return km_per_hour * 1.18
```

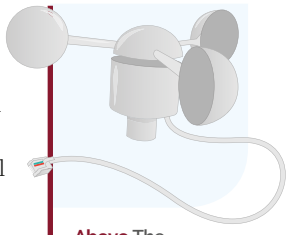
Rerun the code and this time you should get a value closer to 2.4. If you're having a bit of trouble getting the code right, download the complete version of it from here: **magpi.cc/1WuoIVF**.

windspeed.py

Language >PYTHON

DOWNLOAD: magpi.cc/1WuoKwK

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
import time, math

pin = 5
count = 0

def calculate_speed(r_cm, time_sec):
    global count
    circ_cm = (2 * math.pi) * r_cm
    rot = count / 2.0
    dist_km = (circ_cm * rot) / 100000.0
    # convert to kilometres
    km_per_sec = dist_km / time_sec
    km_per_hour = km_per_sec * 3600
    # convert to distance per hour
    return km_per_hour

def spin(channel):
    global count
    count += 1
    print (count)

GPIO.setmode(GPIO.BCM)
GPIO.setup(pin, GPIO.IN, GPIO.PUD_UP)
GPIO.add_event_detect(pin, GPIO.FALLING,
callback=spin)

interval = 5

while True:
    count = 0
    time.sleep(interval)
    print (calculate_speed(9.0, interval), "kph")
```



Above The anemometer, or wind speed sensor, is a very simple device

FREQUENTLY ASKED QUESTIONS

Your technical hardware and software problems solved...

NEED A PROBLEM SOLVED?

Email **magpi@raspberrypi.org** or find us on **raspberrypi.org/forums** to feature in a future issue.

PI ZERO 1.3

HOW CAN I CONNECT A CAMERA?

New cable

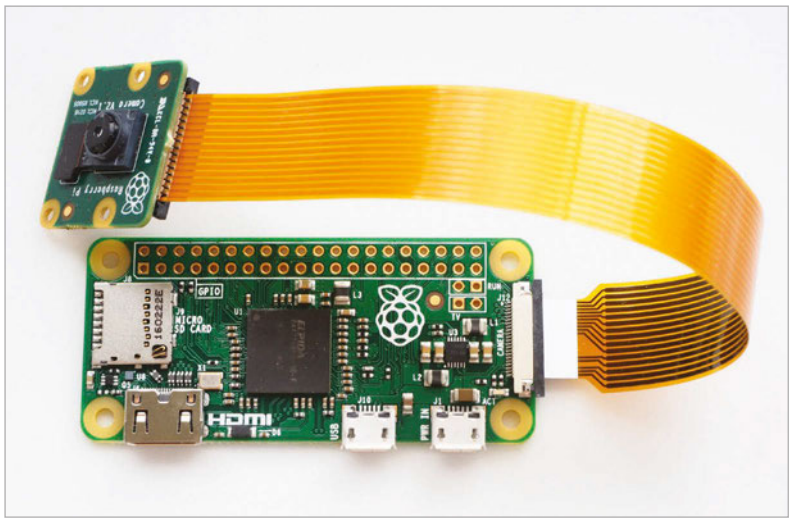
The standard cable that comes with the Camera Module is too big for the connector on the Zero (the normal one couldn't fit), so you'll need to replace it with a different one. They're available online from places like the Pi Hut: **magpi.cc/1V5N9dh**.

Insert the cables

The connectors on both the camera board and the Pi Zero open by pulling back a tab that secures the cable. Make sure the silver strip on the new ribbon is the same orientation as the old one on the board, secure it, and then connect it in the same way to the Pi Zero.

Enable the camera

Once the camera is connected up, turn on the Pi Zero. After it loads up the desktop, go to the Program menu and select Preferences>Raspberry Pi Configuration. In the Interfaces tab, you'll find an option to enable the Camera Module. Do this and then reboot.



MY PI ZERO ISN'T BOOTING

Check the power

Make sure you have enough power going to the Pi Zero, either by using an official Raspberry Pi power supply or checking the amps on your adaptor. If you have too much plugged in, it may not run well off a PC's USB port.

Manual SD card creation

If NOOBS isn't working, then you'll need to burn the SD card manually. Download the image for the latest Raspbian to your computer and then follow the guide to writing an SD card: **magpi.cc/1V5Oj8E**.

New SD card

Your SD card might be shot from too much use, so it may be worth getting a new one and then following the guide to write the image manually to the SD card. 8GB cards are generally best for the Pi Zero, although you can use slightly larger cards.

USING THE CAMERA ON PI ZERO

Taking photos

The Pi camera on the Zero will work the same way it does on a normal Raspberry Pi. Once the camera is enabled, you can use **`raspistill`** in the terminal to start taking photos from the camera. You can read a full guide to how it works here: **magpi.cc/1YuuU03**.

Taking video

Again, the **`raspivid`** command in the terminal will work in the same way. The Pi Zero being so slimline means you can make a great little camera for quick videos or a Go Pro-style experience. Read more about it here: **magpi.cc/1V5ODEy**.

Programming with Python

With the correct libraries, you can program the camera with Python, which you can do slightly more easily through the terminal. We have a pretty thorough guide to doing different things with the camera in Python in issue 45 of *The MagPi*.

FROM THE RASPBERRY PI FAQ
RASPBERRYPI.ORG/HELP

Does the Raspberry Pi support networking?
The Model B, Model B+, and Model 2B/3B versions of the device have built in 10/100 wired Ethernet. There is no Ethernet on the Model A, Model A+, and Zero versions.

Does the Raspberry Pi support any form of netbooting or PXE?
The Raspberry Pi does not support PXE booting or network booting without an SD card. If you want to network-boot multiple Raspberry Pis, you could use PiNet. This is a free and open-source community-based project initially designed for schools. Each Pi boots off a small set of startup files on an SD card and fetches the rest of the data it needs from the PiNet server, thereby allowing you to maintain a single operating system image for all the Pis. PiNet also adds network user accounts, shared folders, and automated backups.

Does the Raspberry Pi have built-in wireless networking?
Only the Pi 3 has built in wireless LAN, but all other models can support a USB WiFi dongle. The Foundation offers its own branded WiFi dongle which has been fully tested for use with the Raspberry Pi. It is available through the Swag store. You can, of course, use a dongle from another provider if you wish.

Why is there no Gigabit Ethernet?
The Ethernet is attached via the USB 2.0 bus, so the upstream bandwidth would not support Gigabit.

Does the Raspberry Pi have built-in Bluetooth?
Only on the Raspberry Pi 3.

How do I connect more USB devices?
Use a USB hub to increase the number of ports. Some keyboards have USB hubs built in, which would work well. It is highly recommended that you use a powered USB hub.

THE MAGPI APP

Having trouble with *The MagPi* on the App Store or Google Play? Here are your most common questions answered:

How do I find *The MagPi* on Google Play or the App Store?
All you have to do is go to the search bar and type 'The MagPi' or 'Raspberry Pi' to find us.

I've subscribed to the digital edition and I can't sign in to restore my purchases. Please help!
Since your *The MagPi* purchases are linked to your Google or Apple accounts, there's no need to sign in at all. If you'd like to re-download your purchases on your current device, or make your purchases available on other devices, all you need to do is press 'Subscribe' on the home screen, then 'Restore Purchases' on the next screen.

How can I search the digital magazine for keywords?
Finding direct references is really easy with *The MagPi* app: all you have to do is tap the screen to get the app's GUI to show, and then press the small magnifying glass icon in the top-right corner of the screen. Just type in your search term to find the relevant results.

Maker Says

A community-developed, Ubuntu-based operating system that beautifully integrates the MATE desktop

Ubuntu

UBUNTU MATE

Ubuntu 16.04 LTS runs really well on the Raspberry Pi using the lightweight MATE desktop environment

With its 1.2GHz processor, plus WiFi and Bluetooth, the Raspberry Pi 3 has the chops to match some laptops. This speed boost has thrown new light on alternatives to Raspbian, the officially supported operating system.

On Intel-based desktops, Ubuntu is one of the biggest names in Linux. Canonical, the maker of Ubuntu, has just released version 16.04 LTS of the Ubuntu operating system. Surely this would be a splendid time to get Ubuntu on the Raspberry Pi?

This is where Ubuntu MATE steps in. Ubuntu MATE isn't quite the full edition of Ubuntu: it swaps out the Unity desktop interface for a lighter version known as 'MATE'.

Ubuntu MATE for the Raspberry Pi was developed by Martin Wimpress and Rohith Madhavan, and it intersects nicely with the recently released Raspberry Pi 3. Ubuntu MATE is built on the brand new 16.04 LTS edition, but with an older and less demanding interface. Sounds perfect, right?

We found the installation to be a relative breeze, although a 16GB microSD card was required (the instructions hint that you might want to use one). When you start up Ubuntu MATE, a welcome screen offers specific Raspberry Pi information. Here you can resize the file system to use the full space on the SD card (rather than the default 3.9GB image size).

A familiar MATE

MATE is a slicker interface than LXDE (the desktop included with Raspbian). While the two are functionally similar in many ways, MATE has rounded windows and more colourful buttons. It's also vastly more customisable, with a range of preferences to change how it looks and works. A MATE Tweak setting enables you to determine the appearance and included elements for the desktop, interface, and windows.

Ubuntu MATE comes packaged with a comprehensive set of default programs. Caja is the default file management app, which is on a par with PCMan, although not as feature-rich as Nautilus (included with the desktop version of Ubuntu). Pluma is the default text editor; again, we find this slightly more stylish than LeafPad, although both are basic as regards functionality.

Beyond this, you get the LibreOffice suite and Firefox web browser. Programming is covered with Scratch, IDLE, and Sonic Pi. One area where it excels is multimedia, with big-name apps like Rhythmbox, VLC media player, and Shotwell working out of the box.

Although you get access to the Ubuntu repository in Ubuntu MATE, you don't get access to the Raspbian repository by default (and so we needed to add it from **magpi.cc/24IM2D8**).

Related

RASPBIAN

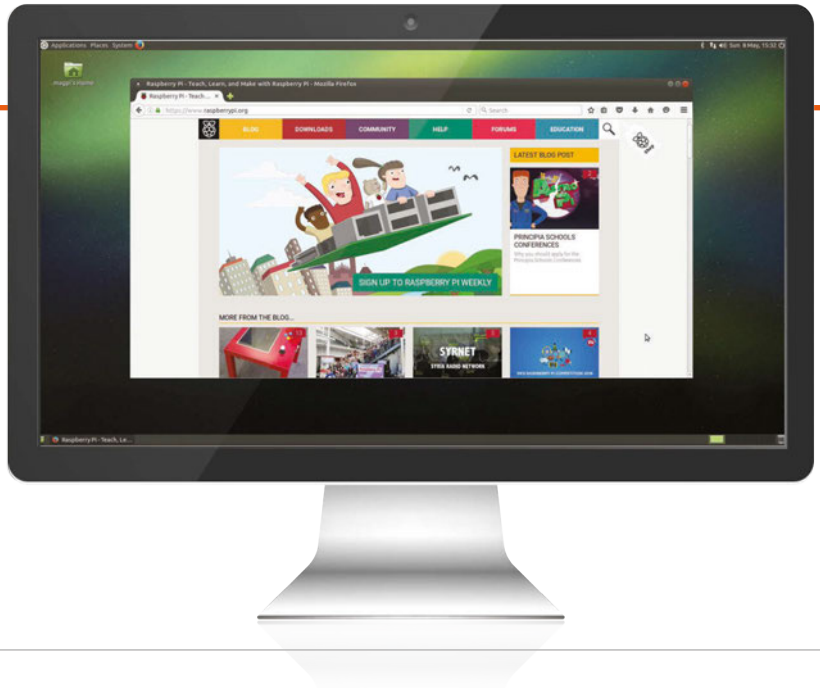
Raspbian is the officially supported operating system of the Raspberry Pi, and is the best all-round operating system for learning and general use.

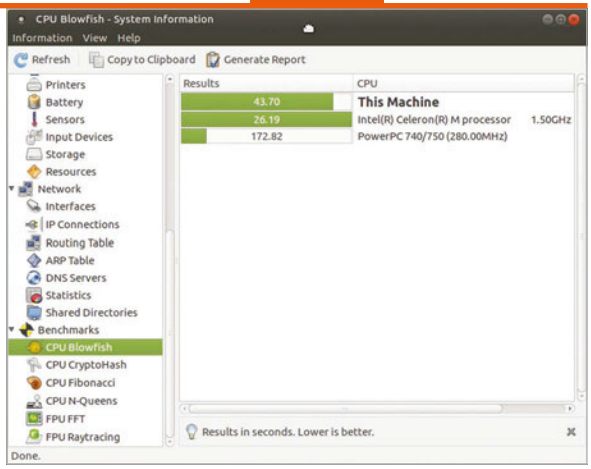
Free

raspbian.org

ubuntu-mate.org

Free





Left Firefox is the web browser of choice for Ubuntu MATE
Above Get great system monitoring tools through the OS

It's a good idea to manage your expectations on software. You still only get access to AArch32 (ARMv7) software, not Intel i86 packages. Programs like VirtualBox, Minecraft Java, and Adobe Flash remain out of reach.

While Ubuntu has a lot of consumer software, Raspbian has better programming tools in the box: Blue J Java, Node-RED, and a free version of Mathematica (software that costs £190 on other Linux operating systems).

An educational OS

Raspbian remains a much better choice for learning the nuts and bolts of programming, thanks to its wider range of coding programs. This is also because Raspbian features in so many tutorials, while Ubuntu MATE leaves you to figure things out.

It's not a complete loss. Ubuntu MATE comes with RPI.GPIO and gpiozero installed, and we found it just as easy to get an LED blinking in Ubuntu as Raspbian.

Even so, with most tutorials aimed at Raspbian users, you'd be better off sticking with the official distro for education and learning.

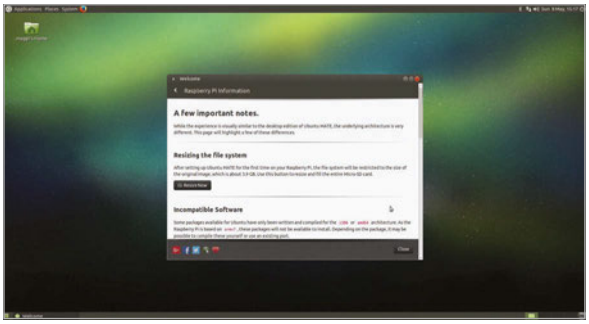
We put both Raspbian and Ubuntu MATE through a series of speed tests to determine which has the edge for performance.

Our first test was a stopwatch for startup (from the time the power was connected to the point you can click and get a response from the menu). We disabled the welcome screen on Ubuntu MATE and turned on automatic login. Ubuntu MATE started up in 39.59 seconds while Raspbian was ready in 20.91 (almost twice as fast).

The slower performance extended to opening apps. LibreOffice took 9.18 seconds to start in Raspbian, and 16.25 seconds in MATE.

We installed HardInfo on both operating systems and ran a series of system benchmarks with the winners in bold:

- **CPU BLOWFISH** (HIGHER BETTER)
  43.70 – **Raspbian**
  40.87 – Ubuntu MATE
- **CPU CRYPTOHASH** (HIGHER BETTER)
  24.63 – Raspbian
  24.94 – **Ubuntu MATE**
- **CPU FIBONACCI** (LOWER BETTER)
  13.47 – **Raspbian**
  14.03 – Ubuntu MATE
- **FPU FFT** (HIGHER BETTER)
  48.73 – **Raspbian**
  47.95 – Ubuntu MATE
- **FPU RAYTRACING** (LOWER BETTER)
  43.89 – **Raspbian**
  54.25 – Ubuntu MATE



Above The presentation is very much like a standard Linux/ Ubuntu release

Raspbian was faster in every test apart from CPU CryptoHash, where having a higher score is better (and the 0.3-second difference seems negligible to us).

Having said that, Ubuntu MATE is perfectly usable on the Raspberry Pi, and we had no problem running multiple apps and programs. If your focus is to run a Raspberry Pi as a regular computer, then Ubuntu MATE has a lot to offer over Raspbian.

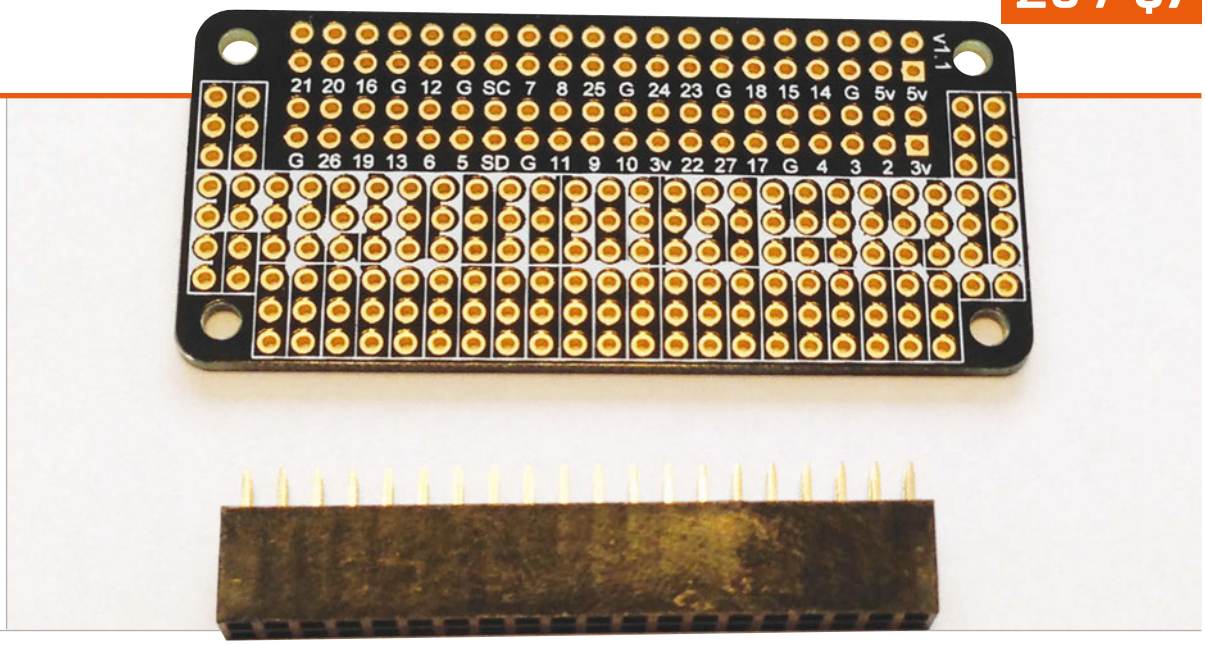
Last word

While Raspbian remains a better choice for project building and education, we have oodles of admiration for this fully rounded desktop operating system running on the Raspberry Pi 3 microcomputer.

★★★★☆

magpi.cc/1TzsY1V

£5 / $7



Maker Says

"Makes it easy to move your messy Raspberry Pi breadboard projects to a PCB

ProtoBoards.co.uk

PROTOZERO

An easy-to-use prototyping board with perfect Pi Zero proportions

Breadboards are great for testing out electronics projects, but the end result may well look messy, often featuring a spaghetti-like tangle of jumper wires. Not only that, but such a setup isn't all that portable, as some loose wires or resistors could easily fall out if you dare to move it. If you're looking to make your project more permanent and practical, you'll want to solder the components to a prototyping board like this one.

The subject of a successful KickStarter campaign, the ProtoZero is the brainchild or Richard Saville, author of the popular **averagemanvsraspberrypi.com** blog. Resembling the little brother of his previous ProtoPal prototyping board, it's made of perfboard and is designed to match the dimensions of the Pi Zero perfectly, although it can also be used on any other 40-pin model, including the Pi 3, 2, A+, and B+. While it is can not technically be described as a HAT, as it lacks an EEPROM, the ProtoZero does sit neatly on top of the Pi's GPIO pins, for which it has two corresponding rows of connector holes with helpful labels. The remaining 154 holes – likewise featuring high-quality ENIG (electroless nickel immersion gold) PCB plating – are arranged in printed lanes of three or four. Components may be soldered to the front or (similarly labelled) rear, if you want to hide unsightly wires and suchlike. Example project ideas include an LED array, temperature monitor, and four-character digital display, but the only limit is your imagination.

Note that since the ProtoZero comes in kit form, you'll first need to solder the supplied 40-pin GPIO header to the board, but this is no more difficult than doing so for the Pi Zero itself – and you'll be soldering components to it anyhow. One advantage, as well as reduced cost, is that you may elect to use a stacking header (not supplied) instead, enabling you to stack multiple ProtoZeros and/or HATs.

While some veteran electronics project makers may prefer to create their own custom-made prototyping boards from inexpensive perfboard, the ProtoZero is ideal for less-experienced users who want to take their projects one step up from the breadboard. With its perfectly rounded edges and matching form factor, it also gives a professional look to Pi Zero projects.

Last word

The ProtoZero offers electronics dabblers an easy, inexpensive way to make their projects more permanent, while neatly matching the dimensions of the Pi Zero for a polished finish. Using high-quality PCB plating, it features a labelled breakout of the 40 GPIO pins, along with a decent-size prototyping area.

★★★★☆

Related

PROTOPAL

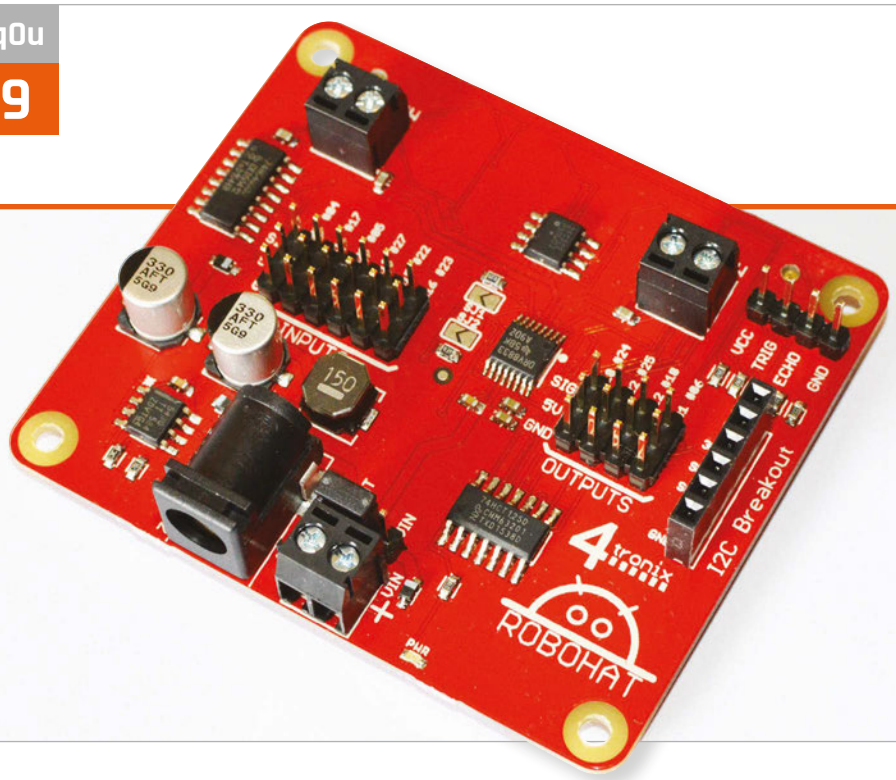
Possibly a better choice for larger Pi models, it features a larger 288-hole prototyping area and two (user-connected) power rails

£5 / $7

magpi.cc/1TzsZ69

magpi.cc/1TPyq0u

£20 / $29



Maker Says

"Complete robotics controller for Raspberry Pi"

4tronix

ROBOHAT

A full-sized Raspberry Pi robotics controller and the bigger sibling of the Picon Zero. Should this be your controller of choice?

Another month, another robot board for your Raspberry Pi; this time we're looking at the RoboHAT from 4tronix, basically a bigger version of the Picon Zero we looked at last issue. This board is designed to work with all 40-pin Raspberry Pis (so no original Model A or B, sadly) and as it's a HAT, it attaches neatly on top of a compatible Raspberry Pi without hanging off the sides.

Like the Picon Zero, the RoboHAT comes with an amazing number of inputs, outputs, and general robot connections to make use of. It all comes pre-soldered on a very sturdy piece of PCB, so you're ready to start using it right away. On the board are the requisite DC motor screw terminals; the RoboHAT comes with two, enabling it to control two motors. There's a set of pins for connecting the popular HC-SR04 ultrasonic distance sensor, and a mixture of ten input and output pins, along with a series of I²C breakout pins. It's powered by a DC jack that can also handily power the Raspberry Pi while in use, cutting down on power inputs.

It's all more spread out than on the Picon Zero, so the board becomes a bit less of a mess when everything is connected up and in use. There are also custom Python scripts and libraries to get it all working, making programming very easy. For example, importing the robohat module allows for `robohat.forward()` to move the robot forward. There are many examples included to get your head around how the library works, and they're all very easily installed using the instructions on the 4tronix blog: **magpi.cc/1TPBtWJ**.

It's a good board for novices and also some advanced users, allowing you to take the skills you may have learnt from a very simple kit and apply them to a custom project of your own design quite easily. It's easy to find out which GPIO pin controls which aspect, so you can create further custom code if that's your thing, making it fairly open and versatile. It's also a pretty good price at only £20.

Last word

A well-designed robot HAT that works very well with the Raspberry Pi and many common robotics components. Definitely worth a look if you're planning to advance your Raspberry Pi robotics.

★★★★★

Related

PICON ZERO

A much more compact version of the RoboHAT, designed for the Pi Zero, although it can also be used on a standard Raspberry Pi.

£13 / $19

magpi.cc/1p9wGaA

Maker Says

"Turn your Raspberry Pi into a modern indoor air quality monitoring device

LiV Pi

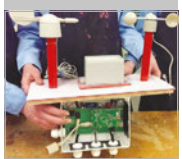
LIV PI

A powerful air quality monitor made for the modern and slightly polluted world, powered by Raspberry Pi

Related

RASPBERRY PI ORACLE WEATHER STATION

It's not quite the same, but you can measure the weather with the Raspberry Pi Oracle Weather Station, albeit without precise pollution sensors.



£N/A

magpi.cc/1Va4d23

Coming from Hong Kong, the LiV Pi is a curious, Raspberry Pi-compatible product. Marketed somewhat as a device for businesses to keep an eye on the air quality inside their buildings, it's a product that's been created mainly to suit the situation in Hong Kong. The city has some serious air pollution problems, which you can read more about elsewhere. So, with that and general building safety in mind, it does start to make more sense.

What the LiV Pi boils down to is a large, colourful acrylic case with a two-line LCD display on the front. Inside, you're supposed to install a Raspberry Pi along with the specially provided add-on board that functions almost like a HAT. This HAT allows you to plug in some supplied sensors and a real-time clock, and then you're basically ready to go.

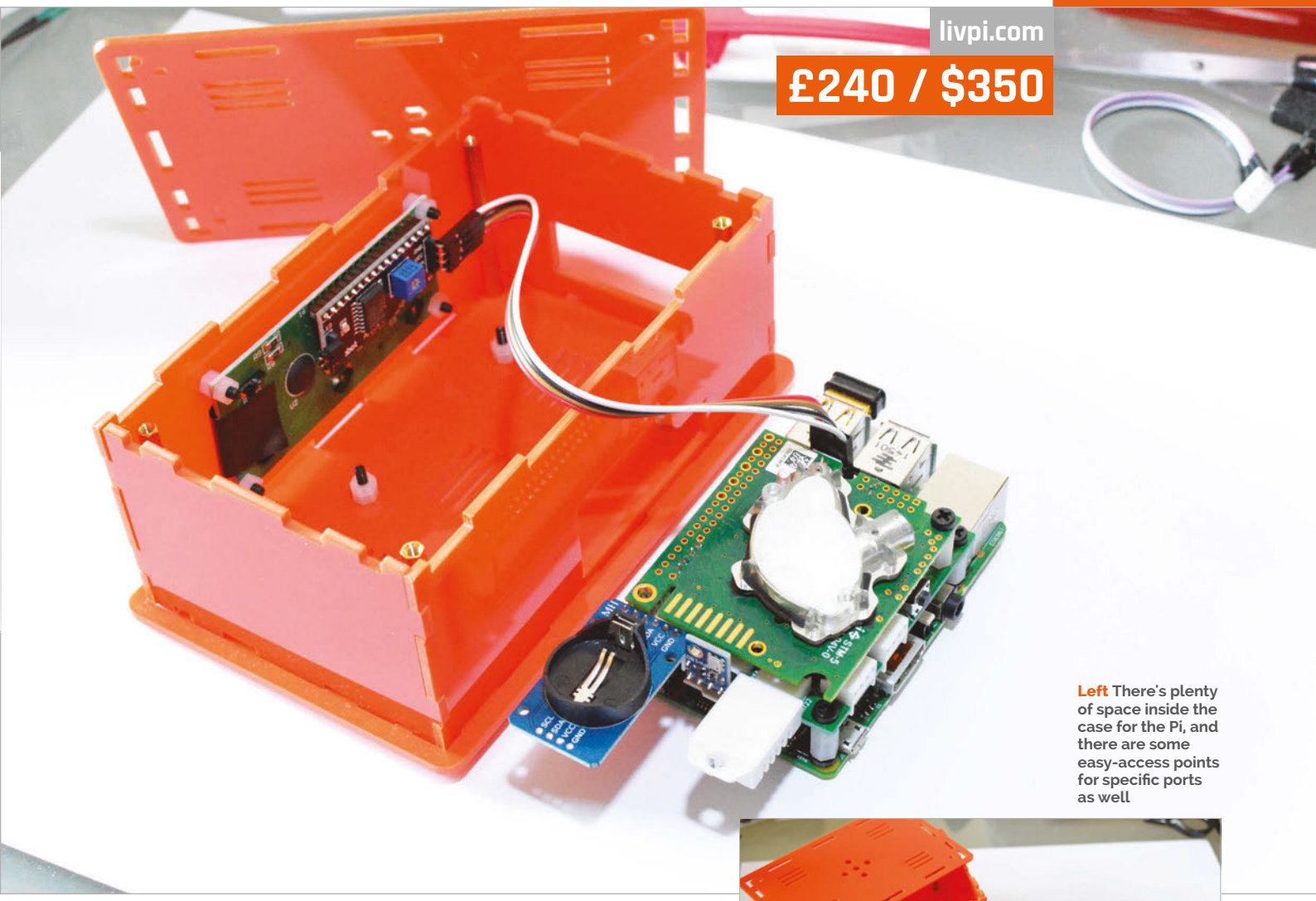
LiV Pi ships with three main sensors: a carbon dioxide (CO_2) sensor, a temperature/humidity sensor, and an air pressure sensor. The CO_2 sensor is the main component here, and the LiV Pi team explain on their website that a rise in CO_2 levels in the air is a good indication of air quality, since it's likely to have bought much nastier air with it. The humidity and temperature sensor allows you to keep track of the environment indoors in other ways, and it sounds like the barometric pressure sensor was thrown into the system just because they could.

There's some fun to be had constructing the full LiV Pi. On the website, you can buy a bare-bones kit of the sensors and board (pre-soldered or not). Alternatively, you can get the full version with a case, which is what we're looking at here. The price does seem quite steep; $350 for the full LiV Pi is not an easy number to swallow, although it's not 'targeted' towards consumers per se.

The construction is quite simple. With the full kit you get the acrylic case, which almost snaps together and is made up of six pieces. The snapping isn't very firm, so there are screws and spacers to keep it together, making it fairly easy to assemble and disassemble when needed. This version also comes

livpi.com

£240 / $350



Left There's plenty of space inside the case for the Pi, and there are some easy-access points for specific ports as well

with everything already soldered up on the boards. Then again, there's not a huge amount of soldering to be done with the basic kit version (from $35): the odd resistor and connector is all that's required, with the trickiest bit being the 40-pin connector for the GPIO.

Once the case is together, it's just a matter of plugging the sensors into the correct ports, putting a battery in the RTC, and hooking it up to the LCD screen. The guides online and the markings on the boards show more than enough to do this accurately. Finally, you'll need to install an SD card with the custom LiV Pi image before packing it all in the case. With the full pre-soldered kit this is all quite quick; in fact, writing the SD card will take longer than constructing the rest, so you should probably get that started beforehand.

The software requires a bit of setup to get working at full capacity. You need to make sure it has a network connection and that the hardware clock is working, and you can also set up email and other message alerts. This does allow you to keep the LiV Pi in its box most of the time, though; you can dial in with a browser to view data and get email updates about the current readings instead of manually plugging it back into a monitor. The whole system is pretty complete and is continually getting updates; at the time of writing, they had recently added Raspberry Pi 3 support and have even begun selling the full kit with a Pi 3. So, for the moment, it's well supported and with upgrades to come in the future.

However good it is, with decent sensors and good software to keep an eye on them, we do wonder how useful a device this will be. We can definitely report that it does what it sets out to do: it monitors the air, and it does it well. If this is something you or a business have been looking for, then it's definitely worth a look due to its extra network connectivity and customisation over other solutions.

Above The full range of components is not very high when it comes in a full kit, and they're all quick to assemble

Last word

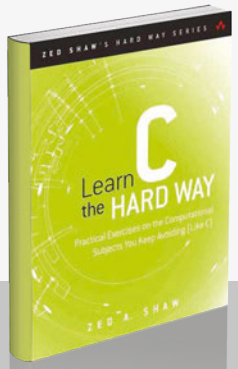
It has very limited uses and is quite expensive, but the design is good and it does what it sets out to do very well. Building it is pretty fun, too.

★★★★☆

RASPBERRY PI BESTSELLERS

C A new generation of C tutorials for a new generation of programmers

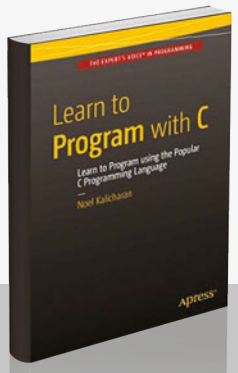
LEARN C THE HARD WAY



Author: Zed Shaw
Publisher: Addison Wesley
Price: £24.99
ISBN: 978-0321884923
magpi.cc/1TQkpD1

Zed Shaw's famous 'type all of this in yourself' method of instilling learning, brought to C. Strongly opinionated but, if you get on with it, a practical introduction to C basics.

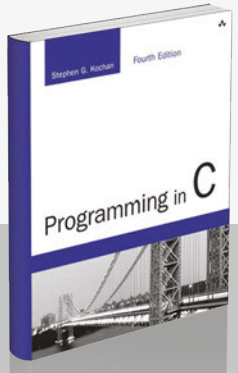
LEARN TO PROGRAM WITH C



Authors: Noel Kalicharan
Publisher: Apress
Price: £17.50
ISBN: 978-1484213728
magpi.cc/1TQkJBu

A comprehensive C tutorial for non-programmers that – despite missing appendices on setting up a compiler – wins friends through the excellent clarity of the author's explanations.

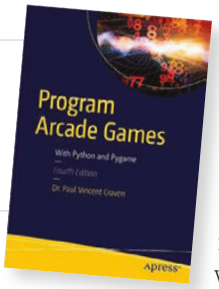
PROGRAMMING IN C, 4TH EDITION



Authors: Stephen G Kochan
Publisher: Addison Wesley
Price: £30.99
ISBN: 978-0321776419
magpi.cc/1TQkIxs

Comprehensive, a classic (now on its fourth edition), and very detailed: serious but readable. If you struggle with pointers, this could be the book to give you that 'aha' moment.

PROGRAM ARCADE GAMES, WITH PYTHON AND PYGAME



Author: Dr Paul Vincent Craven
Publisher: Apress
Price: £29.99
ISBN: 978-1484217894
magpi.cc/1W02yNA

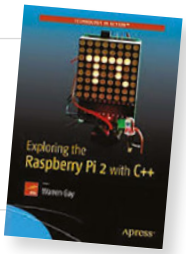
Based on Craven's popular **programarcadegames.com** website, and now in its fourth edition, this book does a superb job of fitting programming concepts – and Python learning – to building several games with the ever useful Pygame library. Programming lessons are carefully introduced throughout each game or game-related project – starting with type correctness in the opening calculator example.

Aimed at younger readers, but accessible to all, Craven's teaching experience shows through in both tone and pace: this is a book that you will not finish quickly, but should keep you motivated through the journey. Along the way, everything from functions and lists, through searching and sorting, to sprites and array-backed grids will be gradually absorbed, until the reader has the skills to write their own games.

There is plenty of code in this book – more than we'd normally expect to see – but Apress's production is excellent here, with pages looking well balanced, and beginners unlikely to be unnerved by the sight of so much Python! Plentiful exercises, including a whole chapter at the end revisiting every project in the book, drive the lessons deep. Well written, well developed, and – despite eschewing the self-conscious whimsy of some other tutorials – very enjoyable to work through.

Score ★★★★★

EXPLORING THE RASPBERRY PI 2 WITH C++



Author: Warren Gay
Publisher: Apress
Price: £29.99
ISBN: 978-1484217382
magpi.cc/1TQkeaS

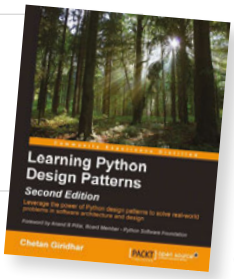
C++ is a powerful and complex beast, which does little to protect you from your mistakes, and is not something we often see in a Pi project. Nevertheless, if you've learnt the basics, you can build your skills while experimenting with your favourite single-board computer, thanks to this interesting tour of the Pi, its hardware, and some useful C++ programs.

Download the code from **magpi.cc/1TQkoyY** and you'll get useful utilities like gp, which manipulates and reads the GPIO interface, and pipwm, for handling PWM (pulse-width modulation) signals. The powerful pispy captures GPIO data and in the book is used to build an oscilloscope. Practical electronics feature in circuit designs for debouncing input signals.

A C++ primer feels slightly misplaced three quarters of the way in, but you can skip forward to that point at any time. Lastly, a multicore web server highlights the Pi 2's abilities, then leaves the challenge of improving and finishing the project to the reader. Appendices on GPIO classes, and other classes – such as Matrix, which document those used earlier in the book – round off a book which may be essential, if you want to work on your Pi with C++.

Score ★★★★½

LEARNING PYTHON DESIGN PATTERNS



Author: Chetan Giridhar
Publisher: Packt
Price: £25.99
ISBN: 978-1785888038
magpi.cc/1TQkVRj

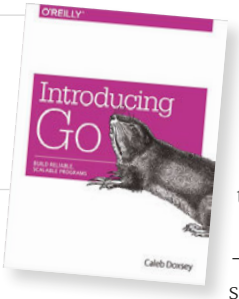
When Kent Beck and Ward Cunningham began applying the concept of design patterns, espoused by architect Christopher Alexander, to programming, OOP (object-oriented programming) was still a relatively tiny niche. As it grew over the next decade, design pattern use spread, and basic patterns were collected in the famous Gang of Four book. These proven development paradigms speed up programming, and well-defined components and interfaces make for scalable and maintainable code.

With many of today's intermediate Python programmers not coming from a formal Java or C++ background – and many patterns being workaraounds for the limitations of those two languages – what's needed is an introduction from the perspective of dynamic scripting languages like Python. Giridhar provides this with a practical, Python-based look at several well-known patterns, from Observer to MVC patterns.

Following a useful introduction to OOP concepts and design patterns, we start with the Singleton design pattern, then progress through several patterns – all with Python code, and practical examples, as well as a discussion weighing up the advantages and disadvantages. Lastly, AntiPatterns tells the reader "what we shouldn't do as architects or software engineers." Concise, thoughtful, and a useful stepping stone on your Python journey.

Score ★★★★☆

INTRODUCING GO: BUILD RELIABLE, SCALABLE PROGRAMS



Author: Caleb Doxsey
Publisher: O'Reilly
Price: £16.50
ISBN: 978-1491941959
magpi.cc/1TQkZk9

Go may be a difficult word to Google for, but the language has many positives – bringing 40 years of advances to a C-style language, from concurrency to memory safety. The key feature is simplicity, even at the cost of missing out on features taken for granted in other languages.

To match the Go language's simplicity, this is a concise, efficient introduction to a concise and efficient language, and very much in the tradition of K&R. Indeed, you'll find yourself taking that classic off your shelf to check half-remembered references as you spot them in Doxsey's Go introduction. Types, variables, and control structures make for a gentle start, with the opening chapters accessible for relatively inexperienced programmers. Exercises at the end of each chapter ensure that the reader thinks through each topic covered.

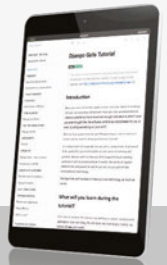
'Arrays, Slices, and Maps' – we were very pleased to see an Oxford comma in the chapter title – and 'Functions' manage to cram a lot of idiomatic Go into explanations which, like the language, know what to leave out. Concentration is required for the pared-back coverage of further Go features, then chapters on packages, testing, and concurrency leave the reader well prepared to tackle some real-world challenges in Go.

Score ★★★★☆

ESSENTIAL READING: DJANGO

The Django web framework is the comprehensive, and Pythonic, answer to your website needs

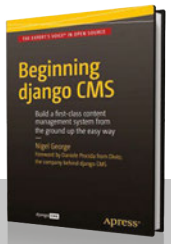
Django Girls Tutorial



Author: Django Girls
Publisher: Online tutorial
Price: Free
ISBN: N/A
magpi.cc/1WptxT1

From total beginner to fully working blog site with Django – an online version of the Django Girls workshop.

Beginning Django CMS



Author: Nigel George
Publisher: Apress
Price: £22.99
ISBN: 978-1484216705
magpi.cc/1WptBlT

George assumes no Django knowledge, and gets you up and running – a welcome alternative to the occasionally opaque official docs.

Django By Example

Author: Antonio Melé
Publisher: Packt
Price: £28.99
ISBN: 978-1784391911
magpi.cc/1WptIOn

Useful collection of real-world websites, from social sites to online shops. Good integration of other technologies.

Two Scoops of Django

Author: Daniel Roy Greenfeld & Audrey Roy Greenfeld
Publisher: Two Scoops Press
Price: £31.12
ISBN: 978-0981467344
magpi.cc/1WptMh0

High-level, low-level, practical, philosophical, opinionated, essential – a reference that makes "Python and Django as fun as ice cream."

High Performance Django

Author: Peter Baumgartner & Yann Malet
Publisher: CreateSpace
Price: £28.23
ISBN: 978-1508748120
magpi.cc/1WptNl3

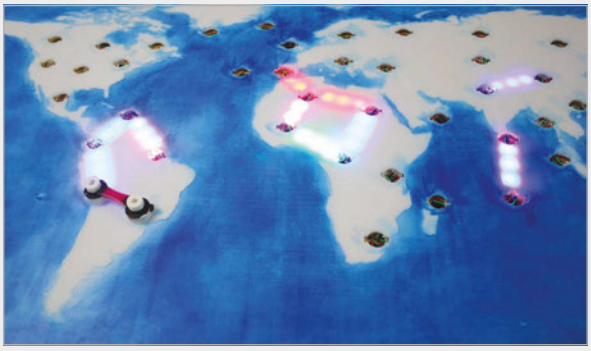
"A repeatable blueprint for building and deploying fast, scalable Django sites" with lessons beyond Django sites.

THE MONTH IN RASPBERRY PI

Everything else that happened this month in the world of Raspberry Pi

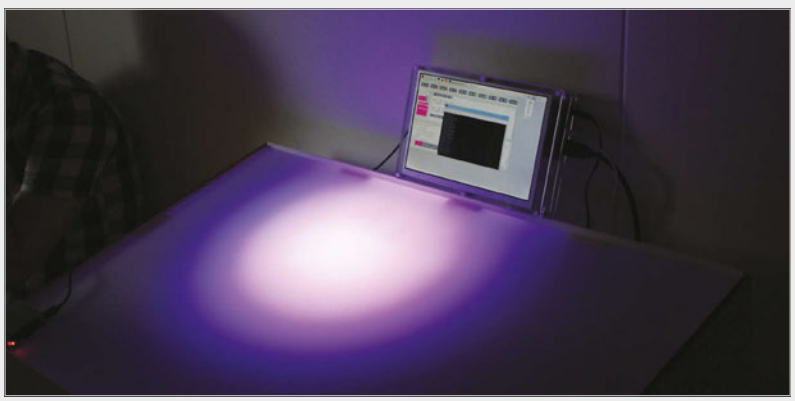
CREATIVE TECHNOLOGISTS EXHIBITION

In early 2015, the Raspberry Pi Foundation set up a new programme called the Raspberry Pi Creative Technologists (**rpct.io**). Nine young creative people between the ages of 16 and 21 were put through a year-long programme of activities, to help them enhance their current creative projects through technology such as the Raspberry Pi. After a year of studying and making, it was finally time for the Creative Technologists to show off what they'd learnt at a special exhibition put on at Raspberry Pi HQ in Cambridge during April. With such projects as 'Who is Terror' (**youtu.be/w3VA7nCg9rI**) and an interactive pop-up book, there were plenty of amazing things to see at the exhibition. If you weren't able to make it, we've put together a little gallery for you to enjoy.



Above The 'blinkenlights' project from Andrew Mulholland creates beautiful illuminated pictures with LEDs and overlays

Below This interactive pop-up book by Maddy Pendergast almost stole the show, with the book projected onto paper to make the effect



Below Using several strips of LEDs and a gesture control HAT, Connor Ballard was able to create a light that moves with your fingers. It's more complicated than it sounds!

5000 PI-TOPCEEDS SHIPPED!

The Raspberry Pi crowdfunded computer is currently making its first shipment to backers

It's been a busy 18 months for the pi-top crew. No sooner had they launched their long-awaited pi-top Raspberry Pi laptop than they began crowdfunding a new project: the pi-topCEED, a $99 Raspberry Pi desktop computer. Coming as a kit with a screen and Pi-powered innards, it's a stripped back version of the original pi-top that came out at the end of last year, allowing the price to be reduced to hit that attractive sub-$100 mark.

Things have progressed fairly well with the project, with only a slight delay to the original delivery date of April. That's pretty good for a crowdfunded project, and the delay was to accommodate an upgraded screen anyway. The pi-topCEEDs are now finally making their way to backers as 5,000 are shipped out around the world.

We've been keeping an eye on what the pi-top team get up to for a while now, due to their inventive view of a highly customisable, upgradable Raspberry Pi computer that's ideal for learning programming and physical computing. Hopefully, the backers are happy with the final product and many kids and adults will get to learn some useful computing skills.

CEED UNIVERSE

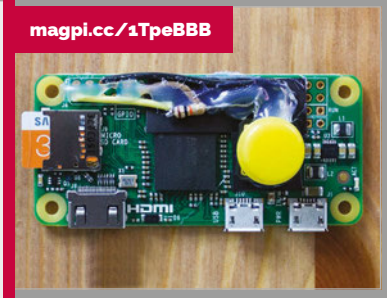
The educational game from the folks at CEED is a very retro-looking open-world puzzle game that requires you to do actual coding to survive. It can also interact with physical computing, so you can create and learn about circuits as you play. All of your progress is saved to the cloud, so you can continue wherever you have access to the game. It's still in active development, but it has come on in leaps and bounds since we first covered it in issue 40 of the magazine.

BEST OF THE REST

Here are some other great things we saw this month

SIMPSONS SHUFFLER

magpi.cc/1TpeBBB



We heartily endorse this event or product. With a press of a button, a random episode of *The Simpsons* is played on-screen, admittedly using a strict selection of episodes from seasons three through ten. Maybe some from 11. Check out Stephen Coyle's blog for info on how to make your own.

INSTANT PI CAMERA

magpi.cc/1quGXil

Another neat idea from the whizzes at Adafruit is this tutorial on how to create an instant Pi camera with a thermal printer attached, much like the old Game Boy Printer of yore. Maybe it's something you can miniaturise with one of the new Raspberry Pi Zeros?

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 your get-together? List it here:
raspberrypi.org/jam/add

1 RASPBERRY JAM LEEDS

When: Wednesday 8 June
Where: Swallow Hill Community College, Leeds, UK
magpi.cc/1Ovi5gl
The aim of this Raspberry Jam event is to bring people together from across a wide area, to discover the exciting potential of the Pi.

2 STAFFORD RASPBERRY JAM

When: Wednesday 8 June
Where: King Edward VI High School, Stafford, UK
magpi.cc/1TcysWv
A big meetup where Pi enthusiasts get together, share ideas, help each other, and most of all have fun!

3 PI AND MORE 9

When: Saturday 11 June
Where: Universität Trier, Trier, Germany
magpi.cc/1TcyAp4
At Pi and More, beginners and experts meet for talks, workshops, and projects focusing on the Raspberry Pi.

4 RASPBERRY JAM: SLICE OF PI CLUB

When: Tuesday 21 June
Where: Heart of Worcestershire College, Redditch, UK
magpi.cc/1TcyC03
Spring season of the Slice of Pi Club where people show off projects, learn skills, and make robots.

5 RASPBERRY JAM HULL

When: Saturday 25 June
Where: Malet Lambert School, Hull, UK
magpi.cc/1TcyH3X
At the Hull Jam, you'll be able to get hands-on with digital making activities, through workshops and a hackspace area to share projects.

6 1ST WIMBLEDON RASPBERRY JAM

When: Sunday 26 June
Where: Wimbledon Library, Wimbledon, UK
magpi.cc/1OviCPj
For people interested in making, creating, programming, electronics, robots, and computing in education.

7 ISTE RASPBERRY PI JAM

When: Monday 27 June
Where: Colorado Convention Center, Denver, CO, USA
magpi.cc/1OviJul

This open session will allow people to learn all about the Pi from Raspberry Pi Certified Educators.

8 RASPBERRY JAM PRESTON

When: Monday 4 July
Where: Media Factory Building, Preston, UK
magpi.cc/1OviZcF

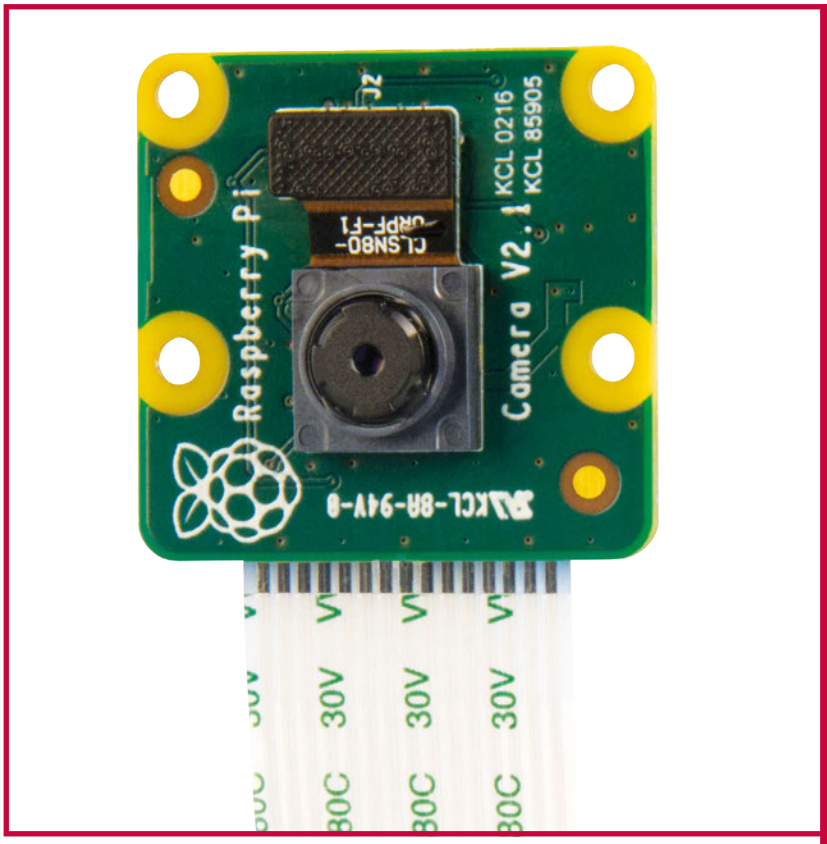
Join the Preston Pi community as they learn, create, and share the potential of the Raspberry Pi computer at an open, family-friendly event.

DON'T MISS: ISTE RASPBERRY PI JAM

When: Monday 27 June **Where:** Colorado Convention Center, Denver, CO, USA

Held during the four-day ISTE technology conference, this will still act like a normal Jam, albeit in a much larger space. That means you can come and show off your project, and also watch demos and presentations on how to use the Raspberry Pi to its full potential. For more details, check out the event page: **magpi.cc/1OviJul**

YOUR LETTERS



Above With a slightly higher resolution and a NoIR version, motion activation can be optimised further with the new camera

Subscription gift

The new camera connector on the Raspberry Pi Zero is brill. It was actually something I'd been really wanting on there! I have a question, though: I don't have a subscription, but I've been thinking of picking one up. I know you get a Raspberry Pi Zero with a subscription – has this been changed to one of the new Raspberry Pi Zeros with the camera connector? Will it also come with the cable adapter? It would probably persuade me to get one!
James Daid

As of this issue, any new subscriptions can get a bundle with a new Pi Zero with the camera port, the cable bundle we previously gave out to adapt the micro USB and mini HDMI ports, and the camera cable! The only downside is you won't be getting issue 40 with it, although it's still available as a free PDF download from our website. You can get this with 6- and 12-month subscriptions, and you can find out about them at **magpi.cc/Subs1**

In motion

With the release of the new Pi Camera Module, I was reminded that I was planning on doing a little project with one a while back, but never researched it: just a simple motion-sensing video shooter for my garden. Is it possible to use the Raspberry Pi as a motion-activated video camera? If that's possible, can I also record sound with it?
Thanks,
Ralph

This is absolutely possible. There are a number of ways to do motion-activated video recording with a Raspberry Pi and a Pi Camera Module; possibly the most reliable method is to use a PIR motion sensor and a Python script to activate a short burst of video recording from the camera. The benefit of this is that it works in low-light situations, whereas doing motion-sensing with the camera itself (which is perfectly feasible) can be a little more temperamental.

As for sound recording, the camera doesn't have a microphone attached to it, so you'd need to hook one up yourself. There aren't many tutorials on doing this, but it's also perfectly feasible and programmable; you may need to edit the video and audio together afterwards, though. We hope this all helps!

Camera compatibility

There's both a new camera and a new Raspberry Pi Zero out, so I was wondering what the compatibility between the two was? Does the original camera work with the Pi Zero? Is it a different cable for either camera?
Roy T

It's somewhat of a coincidence with the timing of the two releases; however, you should be pleased to know that all versions of the Pi Camera Module (NoIR or normal, original or version 2) work on the Raspberry Pi Zero. Not only that, but the new cable works on all versions as well; you'll need that cable to plug it into the Pi Zero, though. We've already seen some people upgrade the camera on their main Pi and use the old camera on their new Pi Zero.

Riscy business

I love the magazine, but I've noticed a trend to focus on Raspbian working with the Raspberry Pi. I know there are more operating systems out there, especially other Linux-based ones; however, I'm a big fan of RISC OS, which is also on Pi. In fact, these days it's one of the last remaining places you can use it. With the Pi being ARM-based as well, its roots are with RISC. It also comes with BASIC, which was a big part of people learning to code in the Eighties! Will you ever have more coverage of RISC OS in the magazine?
Matt Ault

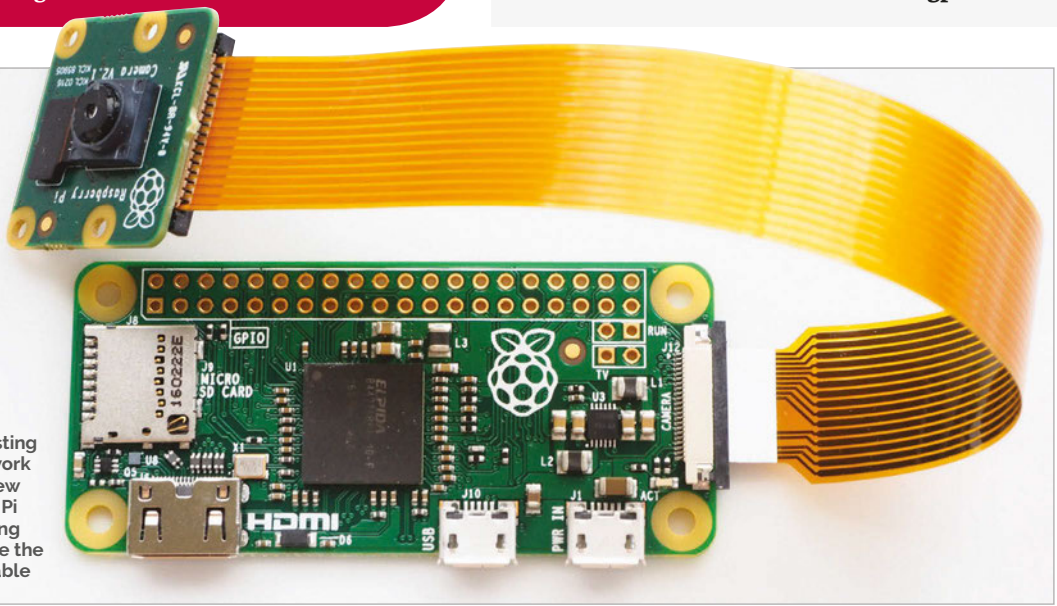
You're right: we do focus a lot on Raspbian. It is the official Raspberry Pi operating system after all, and a lot of the educational tools and techniques are developed for it: *Minecraft* Pi, GPIO Zero, Scratch, etc. It makes it slightly more easier to work with for newbies and relates better to a modern computing lifestyle. A lot of it transfers over to other Linux distros like Arch, Ubuntu, and Fedora on the Pi as well. There is a benefit to RISC OS, though, as it allows for a different approach to computing, and it has its own educational stuff. So, if people let us know they want more RISC OS content, we'll definitely consider it and try to get some more into the magazine.

FROM THE FORUM: FOR ALL AGES

The Raspberry Pi Forum is a hotbed of conversations and problem-solving for the community – join in via **raspberrypi.org/forums**

As a new boy of 88 years of age, who is endeavouring to repel the onslaught of galloping senility by swotting up on the Raspberry Pi, may I ask for some advice on which back issues would help me in my attempt at absorbing the necessary knowledge to move forward?
atemunro

Well, we have a few issues and books that have a beginner's slant in some way. This very issue, in fact, has the beginner's guide to electronics. Other issues that may interest you are issue 36 with our Getting Started with Raspberry Pi feature, our Pi Zero issue (issue 40) which will introduce you to the Pi Zero, and the camera issue from last month. All our Essentials books also start very simple and get more advanced, to teach you along the way, and there's bound to be something to pique your interest in the *Official Raspberry Pi Projects Book*. They're all available for free on our website: **magpi.cc**



All the existing cameras work with the new Raspberry Pi Zero, as long as you have the adapter cable

WRITE TO US

Have you got something you'd like to say?
Get in touch via **magpi@raspberrypi.org** or on The MagPi section of the forum at **raspberrypi.org/forums**

READ US **ANYWHERE**

FREE: DOWNLOAD ALL 30 ORIGINAL ISSUES

The MagPi Magazine

Available now

for smartphones & tablets

Available on the App Store

GET IT ON Google Play

Subscribe from

£2.29 rolling subscription or **£26.99** full year subscription

Download it today – it's free!

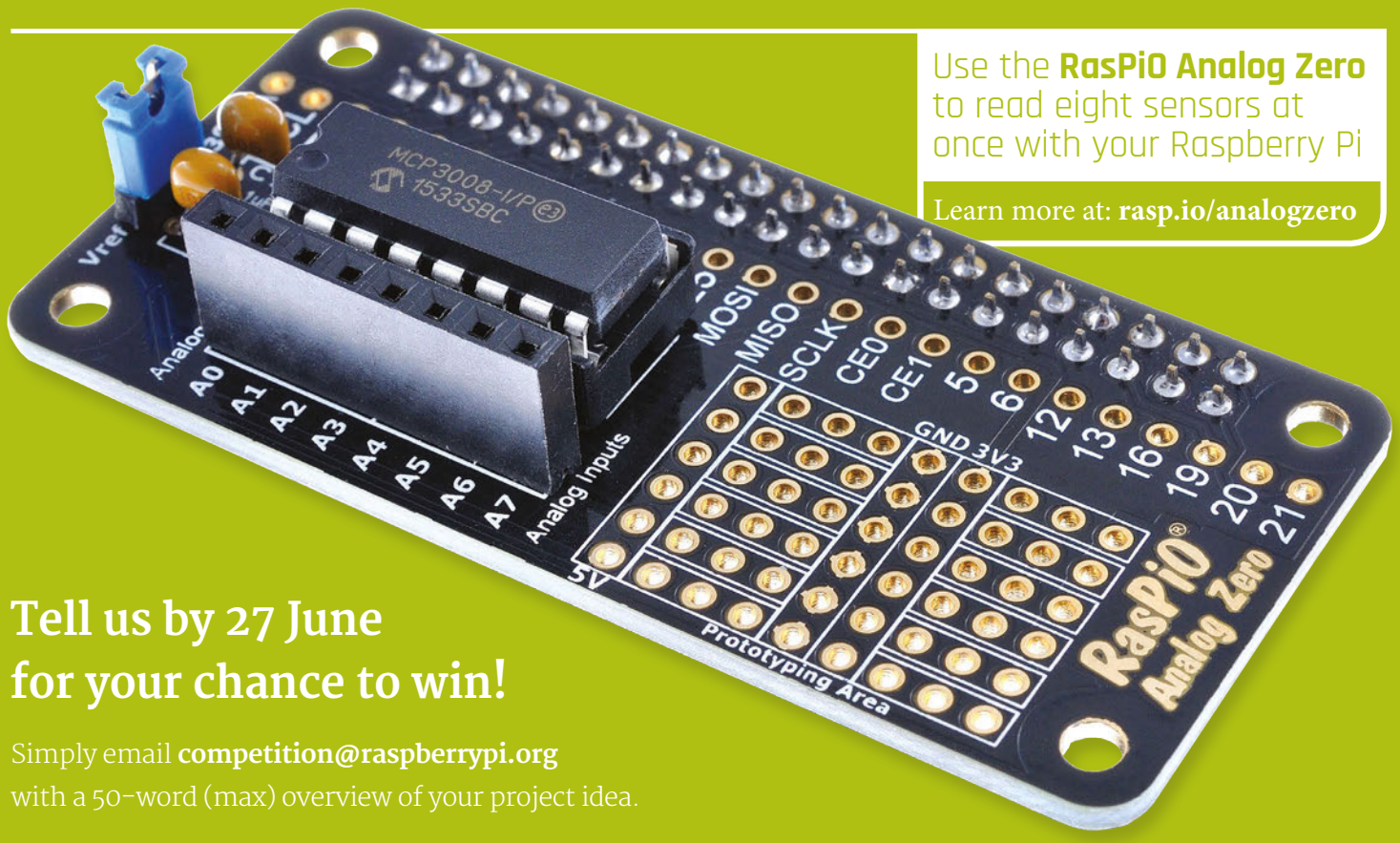
- Get all 30 legacy issues free
- Instant downloads every month
- Fast rendering performance
- Live links & interactivity

In association with

RasPiO®

5 RasPiO ANALOG ZEROS MUST BE WON!

WHAT ANALOGUE PROJECT WOULD YOU BUILD WITH THE RASPIO ANALOG ZERO?



Use the **RasPiO Analog Zero** to read eight sensors at once with your Raspberry Pi

Learn more at: **rasp.io/analogzero**

Tell us by 27 June for your chance to win!

Simply email **competition@raspberrypi.org** with a 50-word (max) overview of your project idea.

Competition closes 27 June 2016. 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.

CURBING CONSUMERISM

The power of maker technology helps cure **Matt Richardson**'s technolust

Quite a few years ago, I once lined up early in the morning to buy a new smartphone on the day of its release. I'm not proud of that. In fact, I find myself cringing just thinking about it. At the time, I was closely following all the latest developments in technology and I had a severe case of consumer technolust. I almost always had to own the latest and greatest gadget, even if my wallet couldn't quite accommodate my acquisitions. In the years that followed, I've cooled off quite a bit. I only rarely find myself pining for new consumer tech.

Don't get me wrong, I love all of the amazing advancements in consumer technology such as phones, tablets, laptops, wearables, and VR headsets. I don't take for granted how far we've come in the last few years. For example, whenever I pull out my phone to search a map or send a photo to my friends, I think how lucky we are to have such powerful tools at our disposal. What's changed is my interest in technology. I became more excited about how maker technology like Raspberry Pi can empower people to create the things they want instead of waiting for a large electronics company to make it for them.

As an example, lately I've seen a lot of makers using Raspberry Pis to create 'magic mirrors', which are mirrors that can display information. These devices let you see news headlines as you primp in the morning. To create them, makers use a two-way mirror and mount a display behind it. The display is driven by a Raspberry Pi to show traffic conditions, weather, your agenda, the news, or anything else that you want.

It's quite possible that one day you'll be able to find mass-produced consumer magic mirrors on retail store shelves. But maker technology like Raspberry Pi enables people to create it now instead of waiting for large electronics companies to make one for them. While they may not be easy enough for a complete novice to set up, maker-made magic mirrors also have the benefit of being incredibly customisable and versatile. And instead of becoming completely obsolete, the parts can be repurposed for other projects when the time comes.

Custom-made technology

I've been finding many other examples of makers using Raspberry Pis to create the products that large companies don't yet make widely available. Makers with diabetes have used technology to create their own closed-loop systems between their glucometer and their insulin pump in order to micromanage their blood glucose levels. Another maker recently used a pair of Raspberry Pi cameras on a robot to create a remote telepresence vehicle for virtual reality headsets.

Instead of consumer products, my technolust is for now for maker products, but I don't feel the same kind of guilt. Maker technology is versatile and can be used for so many more things than could even be imagined by the products' creators. And since maker products can be used for so many possible things, they have a much longer shelf life. Even though we've made so many improvements to the Raspberry Pi in the last four years, there are still many original Raspberry Pi Model Bs out in the world earning their keep, and I suspect that they'll be chugging away for years to come.

Although consumer technology is empowering, I think there's one key thing about maker technology like Raspberry Pi that's worth getting really excited about: what people create with maker technology today will become the consumer technology of tomorrow.

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 accessories.

Breakout Pi Plus

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

ADC Differential Pi

8 channel 18 bit analogue to digital converter. I²C address selection allows you to add up to 32 analogue inputs to your Raspberry Pi.

IO Pi Plus

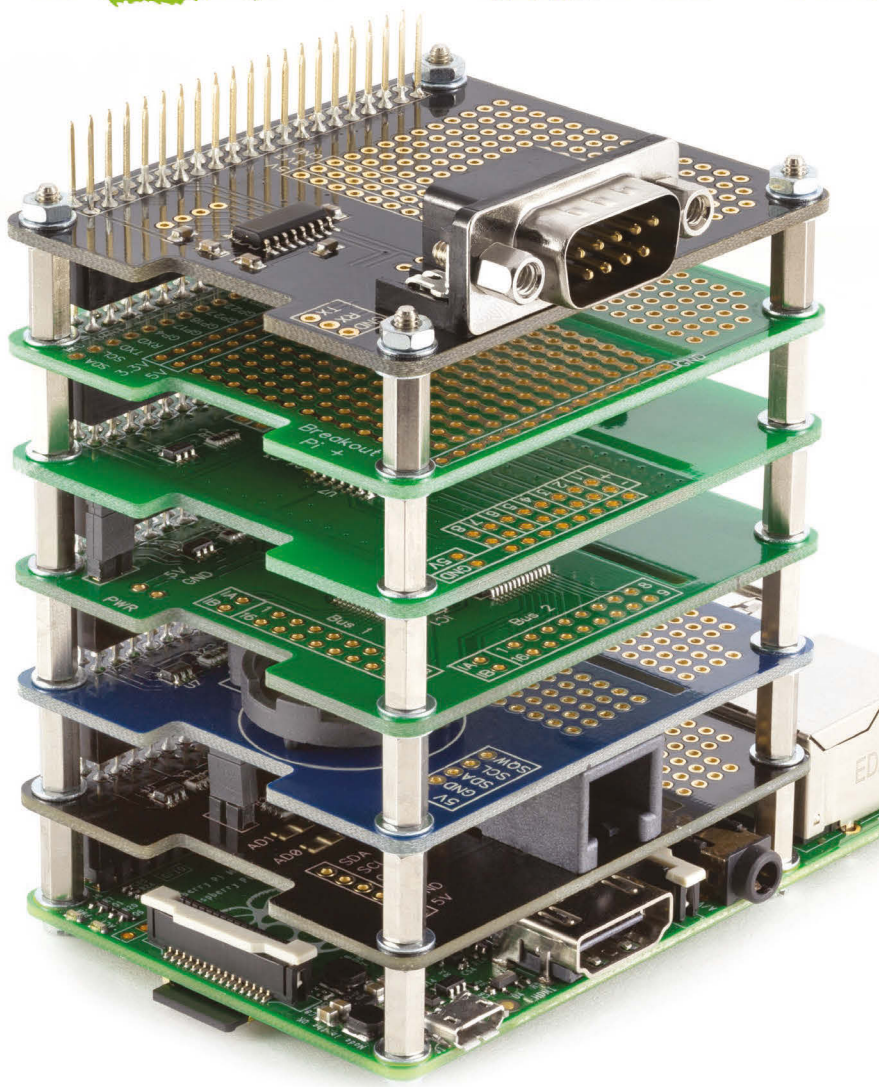
32 digital 5V inputs or outputs. I²C address selection allows you to stack up to 4 IO Pi Plus boards on your Raspberry Pi giving you 128 digital inputs or outputs.

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.



Now Available for the **Pi Zero**

ABelectronics UK

www.abelectronics.co.uk