BUY IN PRINT WORLDWIDE MAGPI.CC/STORE

The MagPi

Issue 79 | March 2019 | magpi.cc

The official Raspberry Pi magazine

WIN! OFFICIAL KEYBOARD & MOUSE KITS

DEVELOP AN ANDROID APP

Build & install your own apps with Raspberry Pi

MAKER SPECIAL! #MONTHOFMAKING

11 projects to create & share

ADD AI TO YOUR PROJECTS

Use Google's Edge TPU kit to make intelligent machines

SET UP A SMART CAR CAMERA

Keep an eye on your car using an automatic number plate reader

£5.99

TOP 10 Home Automation Projects

GLOBAL DELIVERY magpi.cc/store

Raspberry Pi PRESS

WELCOME to The MagPi 79

I believe it's incredibly important to make things. It doesn't really matter what you make: a chair, a work of art, or an electronic gadget. But you should be able to look around you and think "I made that!"

Lots of folks tell us that they love the Raspberry Pi, the little computer at the heart of the digital maker movement. But too many people buy a Raspberry Pi and never really make anything with it.

Far too much of our world is about buying, rather than making. So this month we're all committing to a #MonthOfMaking. We want everybody to make something this month and share it with other makers. Let's inspire each other to become makers, not buyers.

Obviously, I think it should be something you make with a Raspberry Pi. And we've got eleven projects to help you get started. But truth be told, I'll be happy to see all type of makes this March.

For my own part, I made a Teachable Machine this month using Google's new Coral USB Accelerator kit. The joy I felt when I taught my Raspberry Pi to recognise a banana! I can't tell you…

Anyway, I hope you join us in making something this month. And don't forget to share it with us and the rest of the community.

Lucy Hattersley Editor

EDITOR
Lucy Hattersley

Editor of *The MagPi*. Lucy codes, crafts, and creates wonky robots. She speaks French (badly) and mangles the piano. One day she'll get that pet dog.

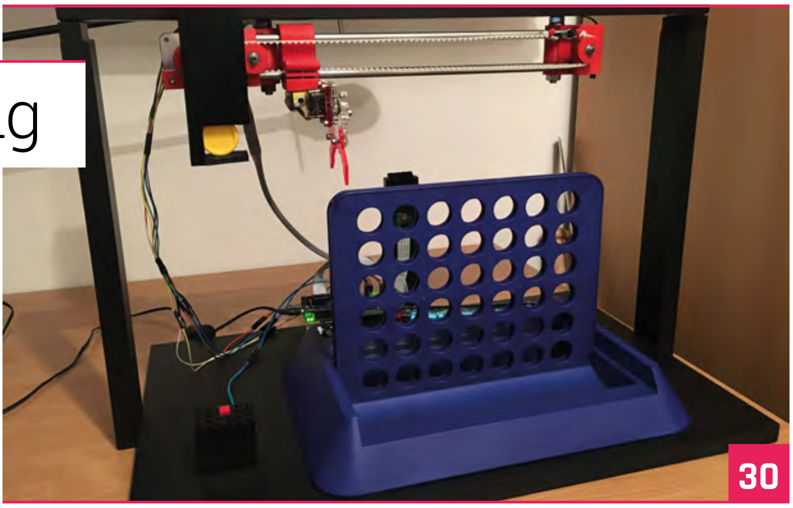
magpi.cc

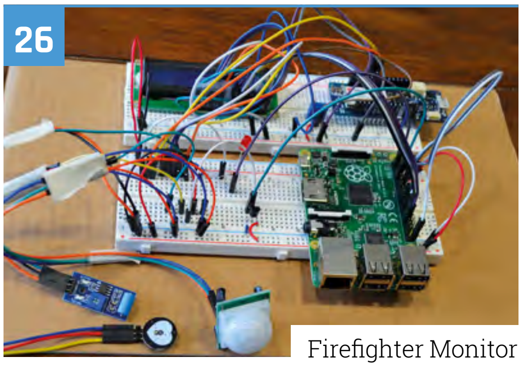
GET A NEW RASPBERRY PI 3A+ WITH A SUBSCRIPTION! PAGE 28

YOUR OFFICIAL RASPBERRY PI MAGAZINE
The MagPi
The official Raspberry Pi magazine
WIN! KEYBOARD & MOUSE KITS
MAKER SPECIAL!
#MONTHOFMAKING
11 projects to create & share
DEVELOP AN ANDROID APP
SET UP A SMART CAR CAMERA
ADD AI TO YOUR PROJECTS
TOP 10 Home Automation Projects

Contents

> Issue 79 > March 2019





Firefighter Monitor

Telescope Mount

Sjoelen Shuffleboard

Teachable Machine

Car Spy Pi

Keybow

This month in Pi

Raspberry Pi opens Cambridge store

Learn all about Raspberry Pi and buy electronics components – **Lucy Hattersley** shops at the all-new official store

Raspberry Pi has opened an official retail store in the Grand Arcade, Cambridge. We caught up with Gordon Hollingworth, Director of Software Engineering at Raspberry Pi – one of the people instrumental in designing the store – to ask why Raspberry Pi was opening a retail environment.

The project booths are "really important," says Gordon. "I wanted them to be inviting and easy to navigate, I didn't want a list of instructions printed on paper, so we use the touchscreen in kiosk mode to display them."

Gordon Hollingworth, Director of Software Engineering at Raspberry Pi

" The store concept is about trying to get closer to a less connected demographic "

Reach out

"The store concept is about trying to get closer to a less connected demographic, people who aren't involved with technology, and show them that coding isn't an inexplicable dark science reserved only for a few. Instead show them that it is possible, with the right instructions and an inquisitive nature, to learn about computers and coding," explains Gordon.

At the end of the store is a 'museum bar'. This glass table contains the first ever Raspberry Pi, a blue Brazilian Pi, red Chinese Pi boards, and a rare copy of *The MagPi* #40 with free Pi Zero. Plus prototype models of Raspberry Pi boards and cases.

A shelf full of Babbage bears!

Behind the scenes

The Raspberry Pi store has been gestating for "over six years," Gordon reveals. But each year Gordon and Eben Upton, Raspberry Pi CEO and co-founder, "decided against it."

Things changed when Maplin closed all its stores in 2018. "With the demise of Maplin, we decided there was the possibility of recruiting just the right person to launch the store for us," explains Gordon.

Raspberry Pi is thinking about what comes next. "Who knows," says Gordon, "maybe there will be a Raspberry Pi store in every city."

▲ Project booths enable newcomers to discover making with Raspberry Pi

▲ The swish façade of the Raspberry Pi store in Cambridge

◀ Visitors are encouraged to try out Raspberry Pi technology in-store coral.withgoogle.com

Google releases Coral

Introducing Coral, a new platform for building devices with local AI

Google has launched Coral, a new platform of hardware components and software tools featuring the Edge TPU to accelerate neural networks on local devices. The Coral USB Accelerator was designed as a pluggable accessory to bring the Edge TPU chip to Raspberry Pi boards.

"Building upon our work with the original AIY kits, we're delighted to introduce our new line of products that bring private, fast, and power-efficient neural network acceleration right into your own projects," says Billy Rutledge, Google's Director of AIY and Coral teams.

> "We can't wait to see what prototypes the community comes up with"

These on-device neural networks will enable makers to integrate fast and efficient AI features into their prototypes and projects while maintaining data privacy. Developers will use TensorFlow to create and train neural networks before compiling them to run on the Edge TPU boards using the software utilities provided. Once the compiled network is installed, all of the inference is performed locally on the Edge TPU and not sent into the cloud, increasing performance without network latency and keeping user data local and under their control.

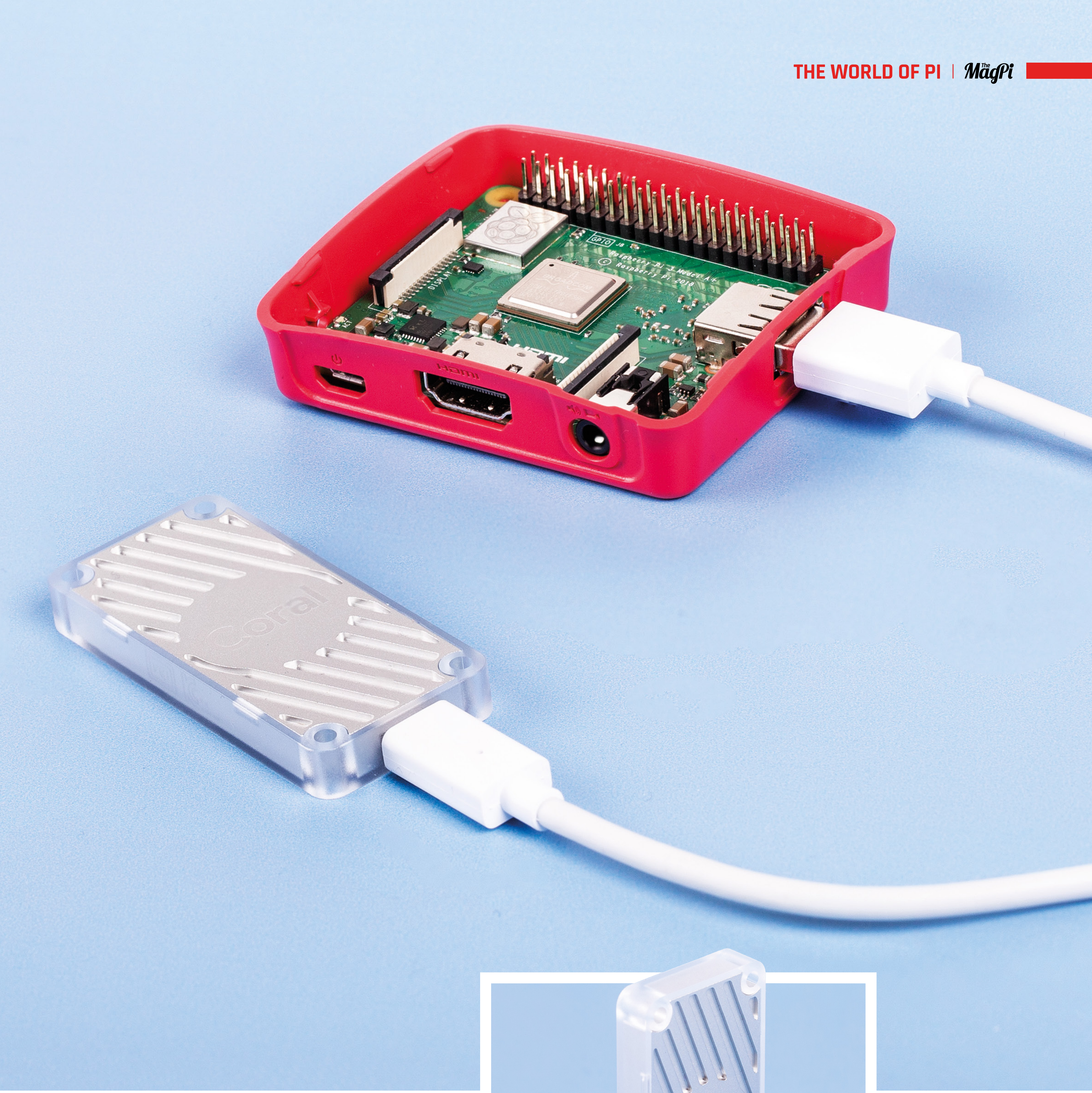
The Coral platform also includes a collection of pre-trained, pre-compiled models to make it easy to get started. These pre-built models have been optimised for the Edge TPU, and software utilities allow developers to repurpose the models for their own use.

Using the USB Accelerator

See our tutorial on using the USB Accelerator to build a Teachable Machine (page 40) for an example of how Coral can bring local AI into your projects.

The new Coral products and AIY kits are available now and can be picked up globally, including the UK and EU countries through Mouser Electronics (**mouser.com**).

Coral products are welcome additions to the Raspberry Pi developer environment. Artificial intelligence remains one of the most creative parts of modern computing, and we can't wait to see what prototypes the community comes up with.



▲ The Coral USB Accelerator connects to a Raspberry Pi via a USB type-A to USB type-C cable

◀ The USB Accelerator contains an Edge TPU chip

Official keyboard and mouse

Raspberry Pi official keyboard and mouse now available. By **Lucy Hattersley**

This month sees the launch of two new products: The Official Raspberry Pi Keyboard & Hub and the Official Raspberry Pi Mouse. Both are designed to work with, and integrate with, the Raspberry Pi computers.

The Official Raspberry Pi Keyboard & Hub is a full-sized 79-key keyboard that attaches to the Raspberry Pi via a USB type-A to USB type-B cable. On the rear of the keyboard are three more USB 2.0 type-A ports, adding a hub aspect to it. This makes it ideal for attaching additional products (including the new mouse).

Red and white

The keyboard is high-quality and ergonomic, but fits neatly onto a desk. It's designed in red and white, to match the Official Case and other Raspberry Pi products.

The Official Mouse is a three-button optical mouse with a scroll wheel (also in fetching red and white). It also connects to the Raspberry Pi via a USB type-A connector. Both devices use wired connections rather than Bluetooth, which we think makes more sense for Raspberry Pi (as it's easier to set them up and you don't need to worry about charging or batteries).

The Official Keyboard & Hub is available now in the new Raspberry Pi Store in Cambridge for £18. The Official Mouse is also available in the Cambridge store for £7. Both devices will be widely available online and at other Raspberry Pi resellers shortly.

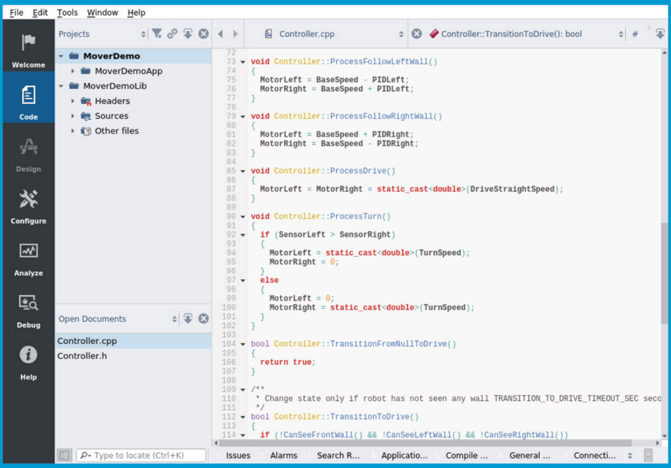
The new Official Mouse looks great sitting next to the keyboard and a Raspberry Pi

" Both devices use wired connections "

Can't make it to the PI STORE?

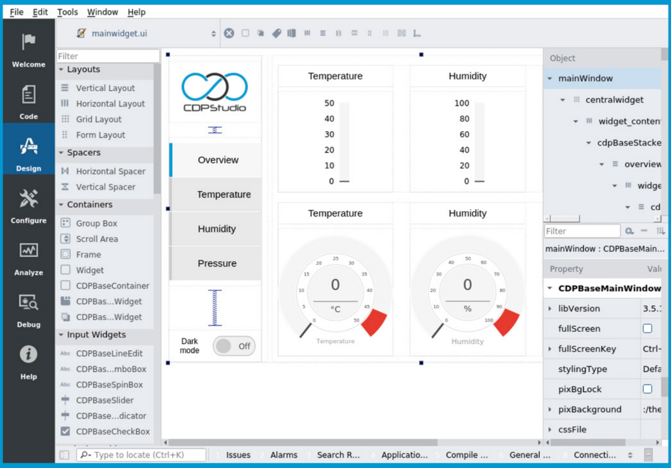
Check out our competition on **page 95!**

▲ The Official Keyboard & Hub is a full-sized 79-key keyboard designed to accompany the Raspberry Pi

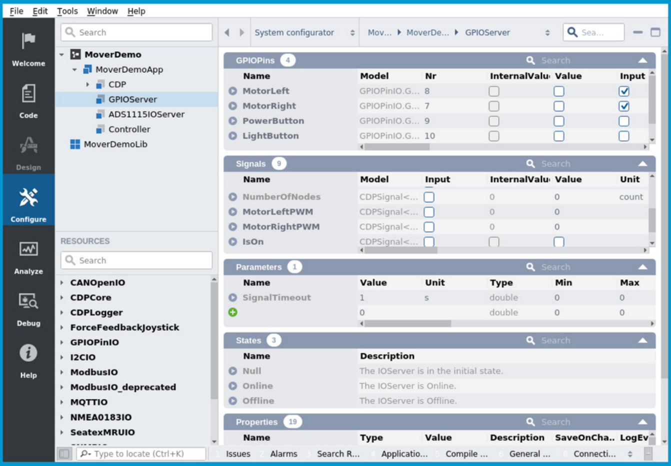


```
void Controller::ProcessFollowLeftWall()
{
  MotorLeft = BaseSpeed - PIDLeft;
  MotorRight = BaseSpeed + PIDLeft;
}

void Controller::ProcessFollowRightWall()
{
  MotorLeft = BaseSpeed + PIDRight;
  MotorRight = BaseSpeed - PIDRight;
}

void Controller::ProcessDrive()
{
  MotorLeft = MotorRight = static_cast<double>(DriveStraightSpeed);
}

void Controller::ProcessTurn()
{
  if (SensorLeft > SensorRight)
  {
    MotorLeft = static_cast<double>(TurnSpeed);
    MotorRight = 0;
  }
  else
  {
    MotorLeft = 0;
    MotorRight = static_cast<double>(TurnSpeed);
  }
}

bool Controller::TransitionFromNullToDrive()
{
  return true;
}

/**
 * Change state only if robot has not seen any wall TRANSITION_TO_DRIVE_TIMEOUT_SEC seco
 */
bool Controller::TransitionToDrive()
{
  if (!CanSeeFrontWall() && !CanSeeLeftWall() && !CanSeeRightWall())
```

Code



Design



GPIOPins 4

Name	Model	Nr	InternalValu	Value	Input
MotorLeft	GPIOPinIO.G...	8	☐	☐	✓
MotorRight	GPIOPinIO.G...	7	☐	☐	✓
PowerButton	GPIOPinIO.G...	9	☐	☐	☐
LightButton	GPIOPinIO.G...	10	☐	☐	☐

Signals 9

Name	Model	Input	InternalValu	Value	Unit
NumberOfNodes	CDPSignal<...	☐	0	0	count
MotorLeftPWM	CDPSignal<...	☐	0	0	
MotorRightPWM	CDPSignal<...	☐	0	0	
IsOn	CDPSignal<...	☐	☐	☐	

Parameters 1

Name	Value	Unit	Type	Min	Max
SignalTimeout	1	s	double	0	0
	0		double	0	0

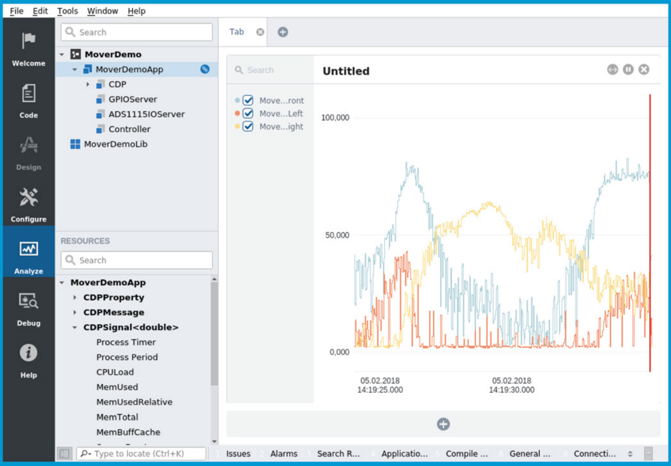
States 3

Name	Description
Null	The IOServer is in the initial state.
Online	The IOServer is Online.
Offline	The IOServer is Offline.

Properties 19

Name	Type	Value	Description	SaveOnCha..	LogEv

Configure



Analyze

Now free for home projects

A professional control system development tool

CDP Studio is a development platform for industrial control systems, now coming with a free version for non-commercial use. The system can run on a Raspberry Pi, supports C++, open source libraries and has a large feature toolbox including GPIO, I2C and MQTT. Its built in GUI design tool and features lets you code less and do more.

Free download on www.cdpstudio.com

CDP Technologies AS
Nedre Strandgate 29
P.O. Box 144
NO-6001 Ålesund, Norway

Tel: +47 990 80 900
info@cdptech.com
www.cdpstudio.com

CDPStudio

Tortoise Fridge

Chilly winters affect us all. Thanks to Pi-controlled heating, Stefan Wollner's tortoise enjoys a cosy hibernation year after year, as **Rosie Hattersley** discovers

MAKER

Stefan Wollner

Stefan works for a large insurance company. He loves technology and taught himself programming.

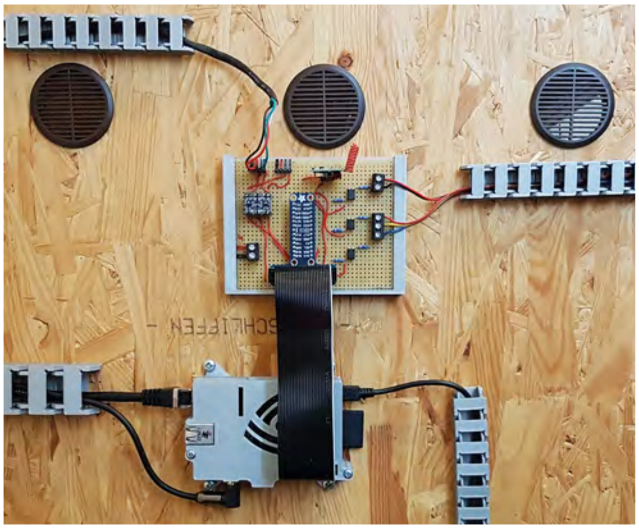
magpi.cc/dioTvm

Seasonal temperature shifts can be a challenge for all living creatures. We humans can simply add or remove a jumper or vest, but our domestic companions don't have that option.

Tech enthusiast Stefan Wollner encountered exactly this issue when he moved home in 2015 and realised the basement was too warm for his pet tortoise, Pumba. Stefan had planned for the tortoise to spend the winter in a refrigerator – a common ploy for tortoise owners. However, the temperature of a standard refrigerator can't be constantly monitored and the internal thermostat only provides a limited degree of temperature control. The warmest and coldest settings of an average fridge vary just three degrees Celsius.

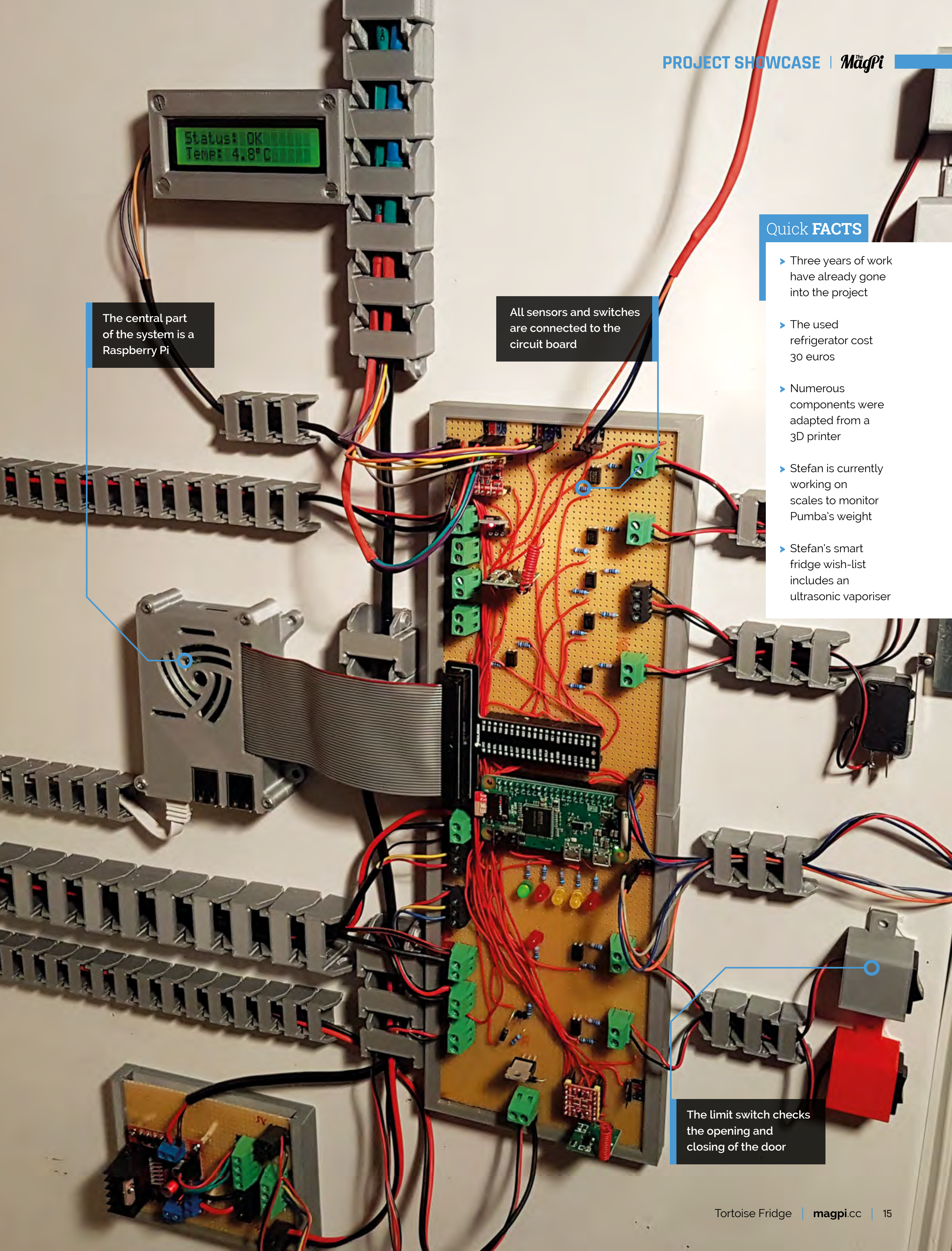
In a flash of inspiration, Stefan realised that he could use a Raspberry Pi and a temperature sensor to keep watch on Pumba's environment. For just a few euros, Stefan bought a DS18B20 temperature sensor to place inside the refrigerator. The system checks the temperature once a minute and records the results in a database, which it then displays as a real-time graph. The sensor's findings are then acted upon by a cooling compressor connected to a 12 V-to-230 V switching relay, operated by the Raspberry Pi.

Two small Pi-powered displays visually confirm everything is working as it should. One two-line display shows the fridge's current temperature and the temperature of the basement; the other displays the time the refrigerator was last opened, and the current status of the cooling compressor.



▶ The heart of the kitchen installation is a first-generation Raspberry Pi B

▼ A house move prompted Stefan Wollner to consider how to adapt the new surroundings for his tortoise's hibernation



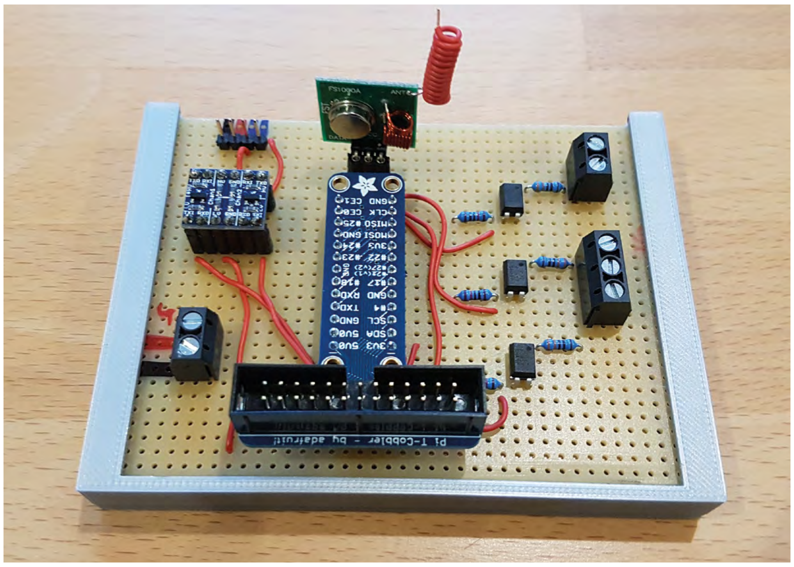
The central part of the system is a Raspberry Pi

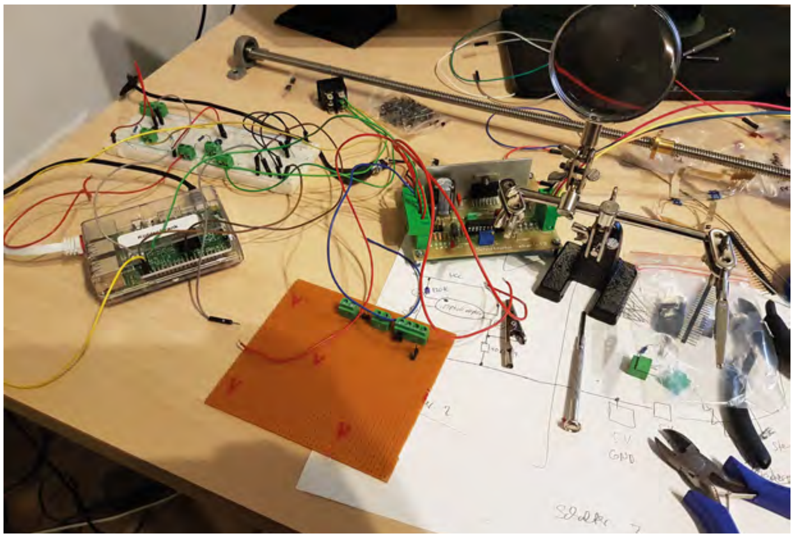
All sensors and switches are connected to the circuit board

The limit switch checks the opening and closing of the door

Quick FACTS

- Three years of work have already gone into the project
- The used refrigerator cost 30 euros
- Numerous components were adapted from a 3D printer
- Stefan is currently working on scales to monitor Pumba's weight
- Stefan's smart fridge wish-list includes an ultrasonic vaporiser





▲ A T-Cobbler Plus breakout board is used to wire the other electronics to the Raspberry Pi

▶ Pumba enjoys chomping on flowers

▼ Prototyping the project on a desk

Whenever someone visits the basement, the backlight of both displays is activated by a PIR motion sensor. There's no need for Stefan to visit the basement to check up, though: another Raspberry Pi shows this data on two additional LCD displays in the kitchen. If Stefan's busy preparing food when he wants an update on Pumba's environment, he simply presses a button on the kitchen display and gets an audible update.

> “Next up is an ultrasonic vaporiser to regulate the air humidity”

A breath of fresh air

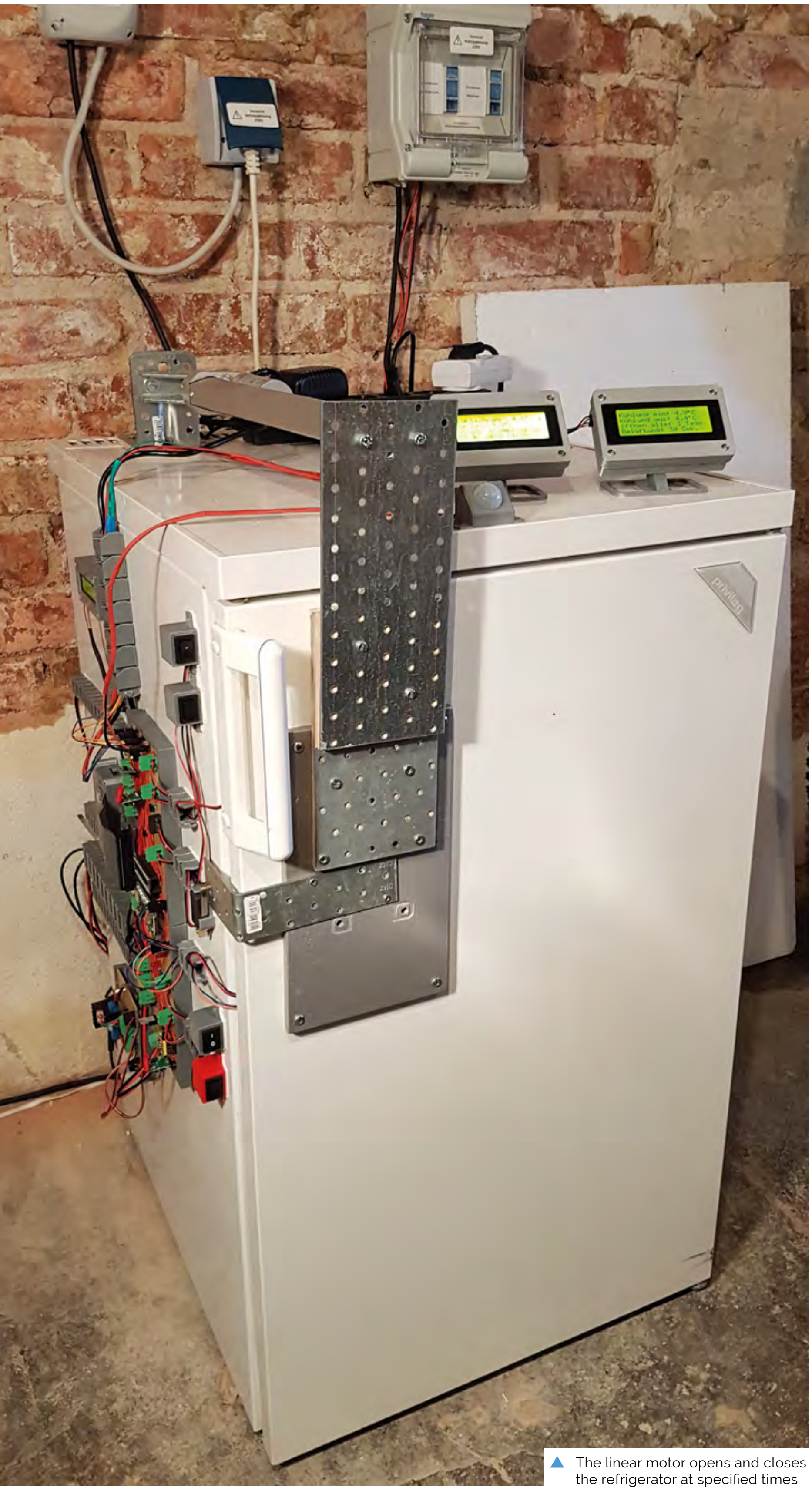
Temperature control wasn't the only issue facing Stefan and his beloved tortoise. He also had to consider its oxygen supply. Whenever Stefan was at home, the fridge door would be opened regularly, refreshing the oxygen supply, but work and socialising often took him away from home. Stefan wasn't prepared to risk the unthinkable and let the oxygen supply run out. Instead, he created an automatic door-opening system with a linear motor that opens and closes the refrigerator door at preset times and checks the door has been closed properly.

The (by now) very smart fridge automatically issues error messages, warnings, and notices, keeping Stefan abreast of any issues that might arise with his carapace-clad companion's living arrangements. Should he need to, he can even remotely adjust the temperature and the timing of the fridge door opening, to allow more or less oxygen into the hibernation centre.

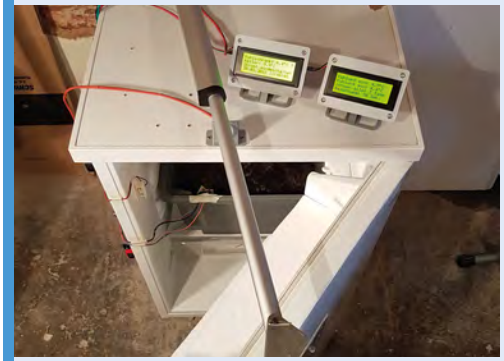
Emergency shutdown procedures

Stefan has spent many hours making the perfect hibernation environment for Pumba. The whole shebang is also safe from intruders and electricity outages. A Raspberry Zero W secures the whole setup. This has its own sensor in the refrigerator which measures the temperature and switches off the system completely when the threshold value has been reached, and sends a notification.

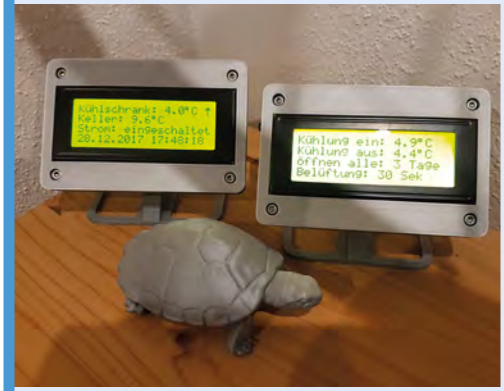
Stefan says, “Since I will probably never complete the work on this project, I am in the process of planning the next enhancements. Among other things I want to install in the near future are an ultrasonic vaporiser to regulate the air humidity and a scale for weight control.” He also plans to automate what the tortoise is fed and when.


The linear motor opens and closes the refrigerator at specified times

Open door policy



01 When no one's at home, the oxygen controller kicks in, automatically opening the fridge door at preset times to refresh the air supply so the resident can breathe easily.



```
Kühlschrank: 4.0°C ↑
Keller: 9.6°C
Strom: eingeschaltet
28.12.2017 17:48:18
```

```
Kühlung ein: 4.9°C
Kühlung aus: 4.4°C
öffnen alle: 3 Tage
Belüftung: 30 Sek
```

02 A range of sensors monitor the temperature in the fridge and the levels of oxygen available. LCD displays in the basement and kitchen show the current status of the fridge and when its door was last opened.

03 Oblivious to his owner's Herculean efforts, Pumba luxuriates in a den of fresh leaves, precisely temperature-controlled and with ample fresh air and food.

Raspberry Pi Tomography Beamline

This project creates 3D models in a very unusual way, and can be controlled by Twitter. **Rob Zwetsloot** tries to figure it out

MAKER

Emma Osbiston

An undergraduate electronic engineering student who came up with the beamline concept while working at Diamond Light Source.

magpi.cc/GGQdcd

Tomography is a process where high-energy electromagnetic radiation (usually X-rays) penetrates an object, and sensors on the other side capture a cross-section of it. These slices are then built up to create a 3D model of the object, complete with its interior. If this sounds familiar, it might be because you've had a CAT scan at some point in your life. It's fairly standard medical technology, and these days you can control them with just a Raspberry Pi.

> It uses a turntable to rotate the object, with the Pi Camera taking a 'reading' of the object under the visible light beamline

Emma Osbiston was entrusted by Diamond Light Source, a facility housing a synchrotron, with creating a way of demonstrating a Pi controlling such a device, albeit safely without a particle accelerator shooting out beams of radiation.

"Essentially, I was tasked with developing a 'mini-beamline', using a Raspberry Pi running Diamond's Data Acquisition software," reveals Emma. "The most significant difference between [this] and a real tomography beamline at Diamond was that instead of high-energy X-ray light, I was tasked to use visible light as my beamline, which meant that instead of an expensive, highly specialised detector, I was able to just use a Raspberry Pi Camera Module as my detector."

A light scan

While a lot of 3D capture rigs use multiple camera setups to take numerous photos at once, this setup uses a turntable to rotate the object, with the Pi Camera Module taking a 'reading' (photos) of the object under the visible light beamline. Using the software, a 3D model is then constructed from the data.

"The idea of the project was to create something that could be useful for outreach and explanation of Diamond's beamlines," explains Emma, "as well as showing off GDA – Diamond's 'Generic Data Acquisition' software for synchrotrons – by demonstrating its flexibility by running it on a Raspberry Pi!"

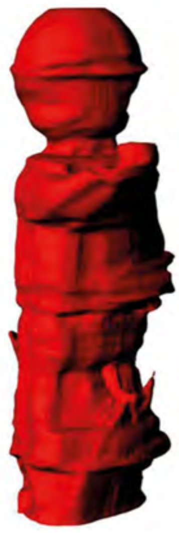
3D with friends

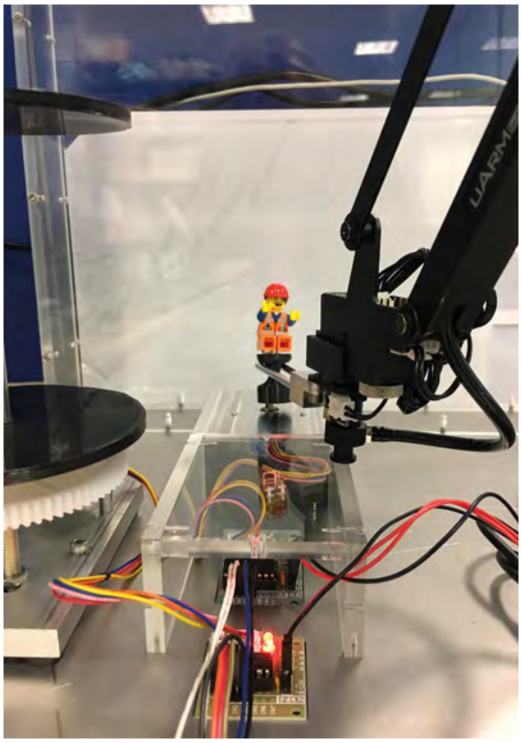
A big part of the project was that it would be operated using Twitter requests, with users tweeting to @DiamondRPi with a number between 1 and 10. A robot arm would then select the sample and put it on the scanning turntable. The scan would then be sent back to the original user.

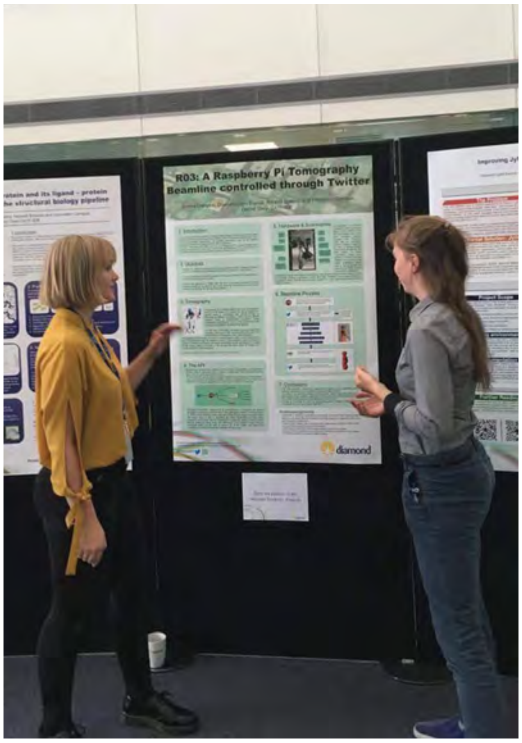
"I would have liked to improve the Twitter response of [this project]," Emma admits. "For example, providing better-quality images or even GIFs, and a progress report from the Pi Camera showing the sample being scanned – this in particular is something being continued at Diamond now!"

The project was completed to the original specifications at least, Emma tells us: "By the end of the summer we had a successfully working 'beamline' that filled all of the initial starting goals and some more – the robot arm had been intended as more of a stretch goal if the project went well, but by the end we had that successfully working too! Essentially, a user could tweet @DiamondRPi with a request, and get scan data back."

▼ What a Lego man looks like as a Lego scan

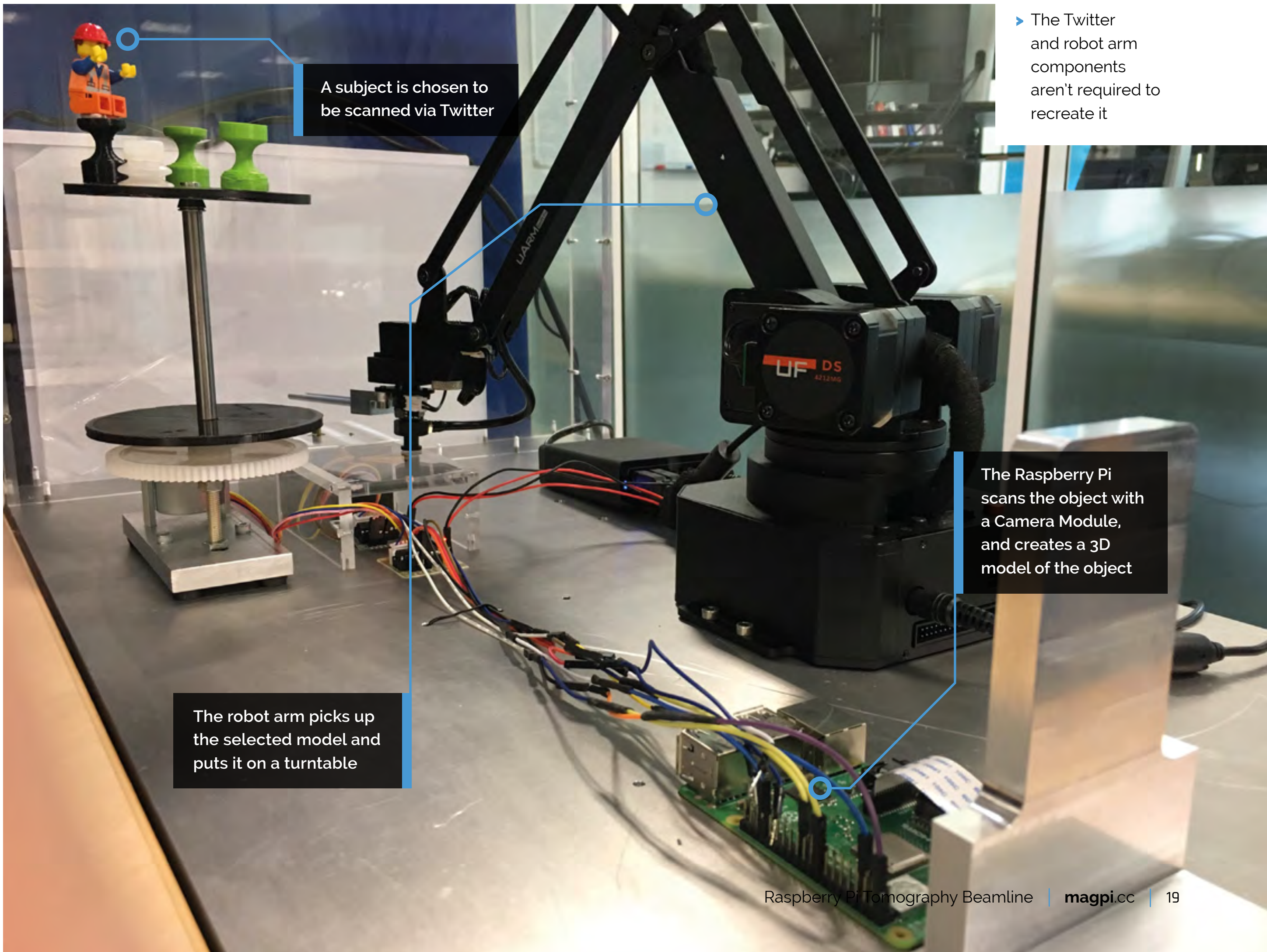





R03: A Raspberry Pi Tomography Beamline controlled through Twitter
diamond

Quick FACTS

- The technical project name is: R03: A Raspberry Pi Tomography Beamline Controlled Through Twitter
- Using a Pi allows for the design to be reproducible
- The Twitter account, @DiamondRPi, is still active
- The project was created during a three-month summer placement
- The Twitter and robot arm components aren't required to recreate it



A subject is chosen to be scanned via Twitter

The Raspberry Pi scans the object with a Camera Module, and creates a 3D model of the object

The robot arm picks up the selected model and puts it on a turntable

Heverlee Sjoelen

Raise a glass to Grant Gibson, whose attempt at simplifying a shuffleboard game has proven refreshingly rewarding, as **David Crookes** discovers

MAKER

Grant Gibson

Grant is interested in physical builds and installations and he has produced a number of interactive games, kiosks, and museum exhibits.

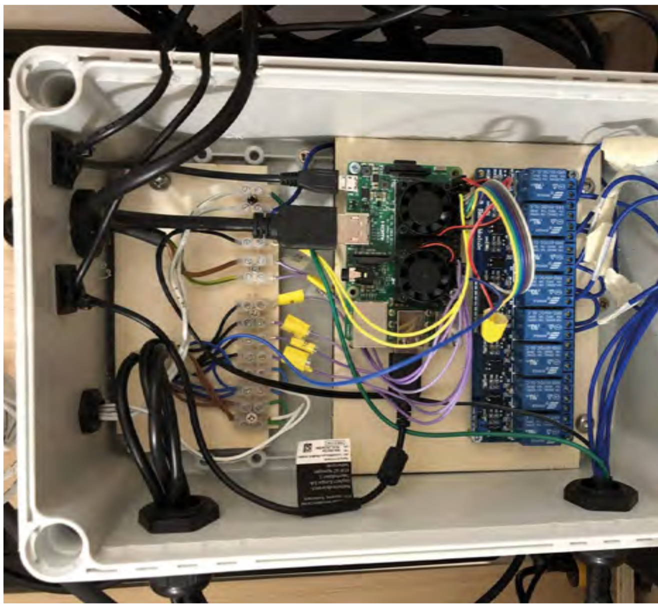
magpi.cc/DQiGwQ

Chances are you've never heard of the Dutch table shuffleboard variant Sjoelen. But if you have, then you'll know it has a basic premise – to slide wooden pucks into a set of four scoring boxes – but some rather complex rules.

It may seem odd that a game which relies so much on hand-eye coordination and keeping score could be deemed a perfect match for a project commissioned by a beer brand. Yet Grant Gibson is toasting success with his refreshing interpretation of Sjoelen, having simplified the rules and incorporated a Raspberry Pi to serve special prizes to the winners.

"Sjoelen's traditional scoring requires lots of addition and multiplication, but our version simply gives players ten pucks and gets them to slide three through any one of the four gates within 30 seconds," Grant explains.

As they do this, the Pi (a Model 3B) keeps track of how many pucks are sliding through each gate, figures how much time the player has left, and displays a winning message on a screen. A Logitech HD webcam is also used so that bystanders can watch the live reaction of players as they veer between frustration and success. "It's a tactile game with a long, interesting history," Grant says.



▲ To operate the can dispenser mechanism, replays were used to control a set of car door locking actuators from the Pi's GPIO pins

" The prototype went out of alignment and started slicing through cans, creating a huge frothy fountain of beer "

Taking the plunge

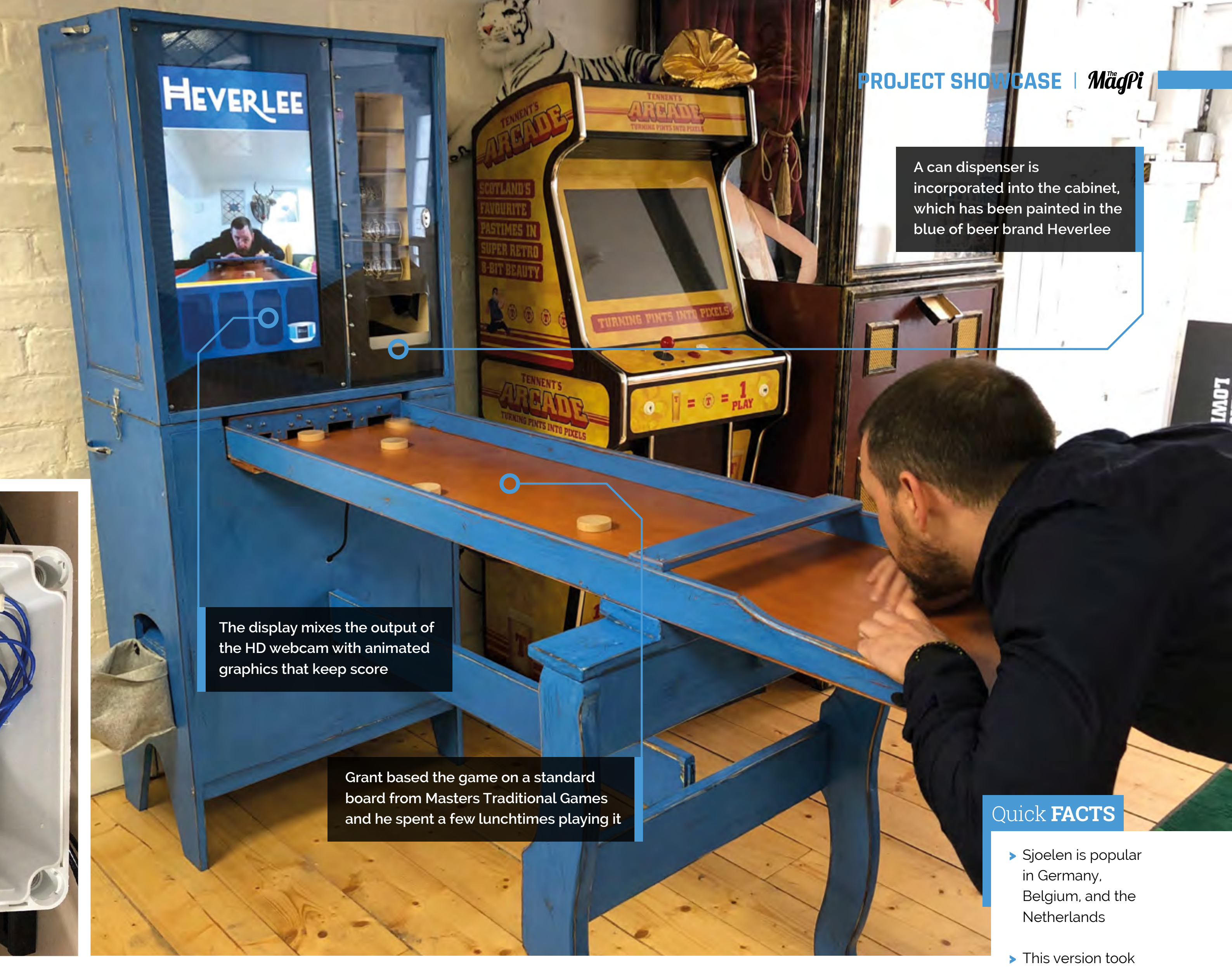
Grant started the project with a few aims in mind: "I wanted something that could be transported in a small van and assembled by a two-person team, and I wanted it to have a vintage look." Inspired by pinball tables, he came up with a three-piece unit that could be flat-packed for transport, then quickly assembled on site. The Pi 3B proved a perfect component.

Although Grant has tended to use full-size PCs, he says the Pi allowed him to reduce software complexity and the need for external hardware to control I/O. Indeed, he was able to simply use Python for the I/O tasks and get the Pi to communicate with a full-screen Chromium browser, via JSON, in order to handle the scoring and display tasks in JavaScript.

"We used infrared (IR) sensors to detect when a puck passed through the gate bar to score a point," Grant adds. "Because of the speed of the pucks, we had to poll each of the four IR sensors at over 100 times per second to ensure that the pucks were always detected. Optimising the Python code to run fast enough, whilst also leaving enough processing power to run a full-screen web browser and HD webcam, was definitely the biggest software challenge on this project."

Bottoms up

The Raspberry Pi's GPIO pins are used to trigger the dispensing of a can of Heverlee beer to the winners. These are stocked inside the machine, but building the vending mechanism was a major headache since it needed to be lightweight, compact, and keep the cans cool.


HEVERLEE
TENNENT'S ARCADE
TURNING PINTS INTO PIXELS
SCOTLAND'S FAVOURITE PASTIMES IN SUPER RETRO 8-BIT BEAUTY
1 PLAY

A can dispenser is incorporated into the cabinet, which has been painted in the blue of beer brand Heverlee

The display mixes the output of the HD webcam with animated graphics that keep score

Grant based the game on a standard board from Masters Traditional Games and he spent a few lunchtimes playing it

Sjoelen has been undergoing a hipster revival of late, making it perfect for bars

Quick FACTS

- Sjoelen is popular in Germany, Belgium, and the Netherlands
- This version took around 150 hours to create
- The total parts cost was less than £400
- A local furniture maker built the cabinet
- Beer is dispensed to the winners

No off-the-shelf vending unit offered a solution, and Grant's initial attempts using stepper motors and clear laser-cut acrylic gears proved disastrous. "After a dozen successful vends, the prototype went out of alignment and started slicing through cans, creating a huge frothy fountain of beer. Impressive to watch, but not a great mix with electronics," Grant laughs.

Instead, he drew up a final design that was laser-cut from poplar plywood. "It uses automotive central locking motors to operate a see-saw mechanism that serve the cans. A custom Peltier-effect heat exchanger, and a couple of salvaged PC fans, keep the cans cool inside the machine," reveals Grant.

"I'd now love to make a lightweight version sometime, perhaps with a folding Sjoelen table and pop-up scoreboard screen, that could be carried by one person," he adds. We'd certainly drink to that.

Raspberry Pi-Driven Telescope Mount

Faced with the astronomical costs of a WiFi-accessible telescope mount, Dane Gardner reached for an affordable star: the Pi Zero W. **David Crookes** looks into it

MAKER

Dane Gardner

Dane is a software engineer by day and an amateur stargazer at night. He lives in Seattle, where there's usually a high chance of cloud and rain.

magpi.cc/JpWWff

Even though you can gaze up at the night sky with your naked eye and see an assortment of stars for free, you'll need to splash some cash on a telescope to get a closer look. Dane Gardner has found a way of incorporating higher-end features into a more affordable scope by making use of the WiFi capabilities of a Raspberry Pi Zero W.

He owns a Celestron NexStar 6 SE, a computerised telescope with a six-inch aperture and a fully automated mount that contains a database of 40,000 celestial objects. It automatically locates and tracks objects, making life easier for a stargazer. One thing it doesn't have, however, is WiFi, nor does it allow Secure Shell access (something no off-the-shelf mount has). "The thought of having absolute control over the telescope via a Linux command line was just too tempting to pass up," Dane tells us.

Pi in the sky

Living in Seattle, a computerised mount is almost a necessity. Not only do the bright skies of the city make it difficult to manually find objects using an astronomy map, the clouds are often quick to roll in. Trouble is, WiFi-enabled mounts are expensive – a Celestron Evolution costs $500 more than the NexStar 6 SE – but thankfully the Pi Zero W is available for a fraction of that and, just as importantly, it also fits perfectly inside the telescope's hand controller.

Indeed, that is where Dane placed it. He'd already spent a couple of months experimenting with a Pi Model B, and looking at ways to use an original Pi Zero and a USB hub for connecting WiFi and RS-232 adapters. Once the Pi Zero W was launched, however, it presented the perfect solution: "I was really lucky that all I had to do was wire up the Pi, configure some software, and write some glue scripts to hold it all together." Powered by the mount's battery pack, it also comes with the bonus of Bluetooth.

Controlling movement

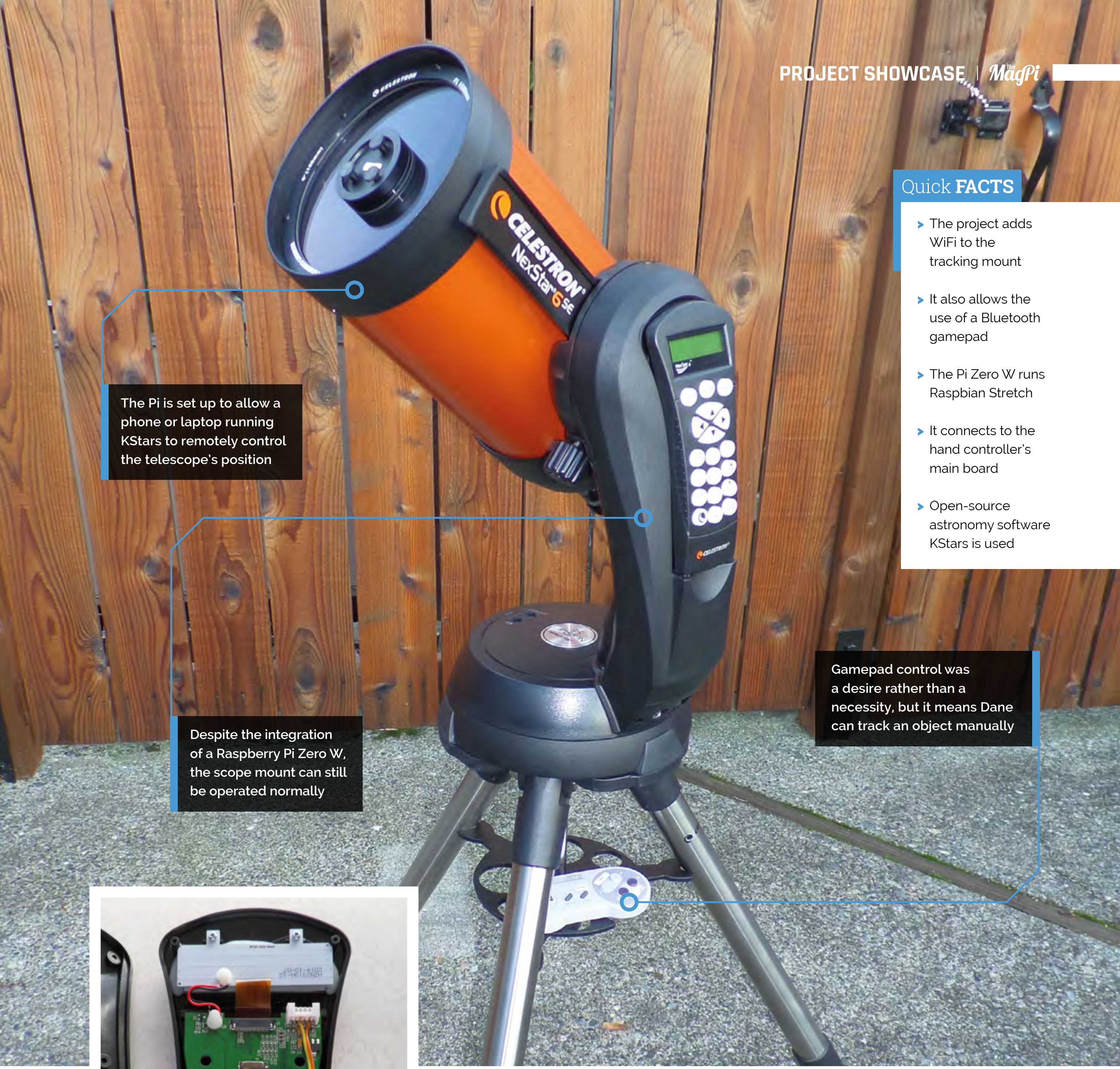
This presented a further opportunity. Although Dane was happy that he could now make use of the both the open INDI protocol designed for controlling astronomical equipment and the open-source astronomy software KStars (which graphically simulates the night sky and makes for a great observation planner), he could also add an alternative controller: a gamepad. This allowed for a better slew interface.

"Wireless slew control allows the scope to settle out faster after movement, so much so that I can track an object manually using the game controller," Dane explains. "I have a great image of Jupiter I captured this way before the sun had even set completely. This is something I couldn't do before with the corded hand controller, which introduces vibrations through the cord and requires me to be very near the scope to control it."

> "The thought of having absolute control over the telescope via a Linux command line was too tempting to pass up"

The result is a telescope mount that better suits Dane's needs, incorporating automatic location and time settings by making use of his external IP address and NTP when connected to a network. "I can run outside, get set up very quickly, and see what I want to see before having to break it back down and head back inside," he says.

What's more, he can also keep both hands in his pockets while he controls the scope. "That's a huge bonus on those cold nights of visual observation," he reveals. "But I'm still surprised at how this just came together so fluidly. I didn't really experience any major problems and none of my projects ever go this well."



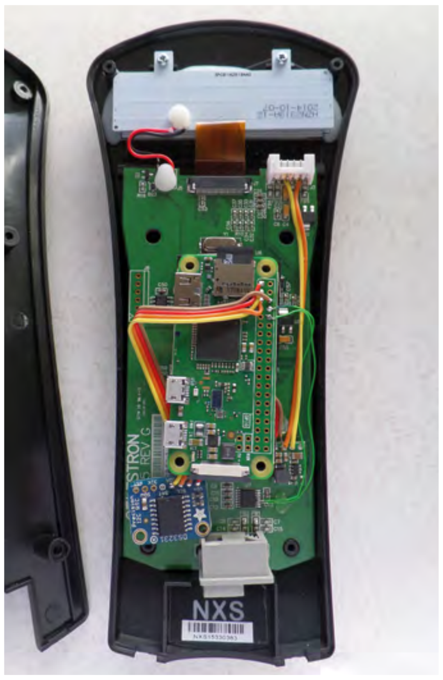
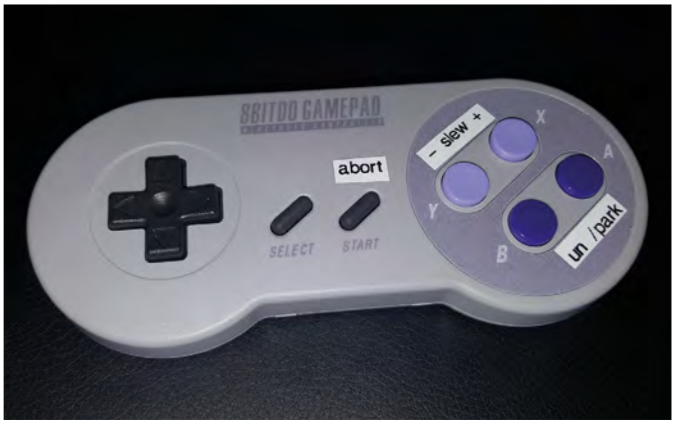
The Pi is set up to allow a phone or laptop running KStars to remotely control the telescope's position

Despite the integration of a Raspberry Pi Zero W, the scope mount can still be operated normally

Gamepad control was a desire rather than a necessity, but it means Dane can track an object manually

Quick FACTS

- The project adds WiFi to the tracking mount
- It also allows the use of a Bluetooth gamepad
- The Pi Zero W runs Raspbian Stretch
- It connects to the hand controller's main board
- Open-source astronomy software KStars is used

Although the 8BitDo SNES30 Bluetooth controller works well, Dane is still having problems with automatic reconnects after it goes into sleep mode

Dane discovered that he could use the Pi Zero W to bypass the RS-232 chip and go pure UART without any level shifting

StereoPi

Powered by a Raspberry Pi Compute Module, this open-source camera board can capture stereoscopic 3D photos and videos. **Phil King** gets double vision

VIRT2REAL

MAKER

virt2real

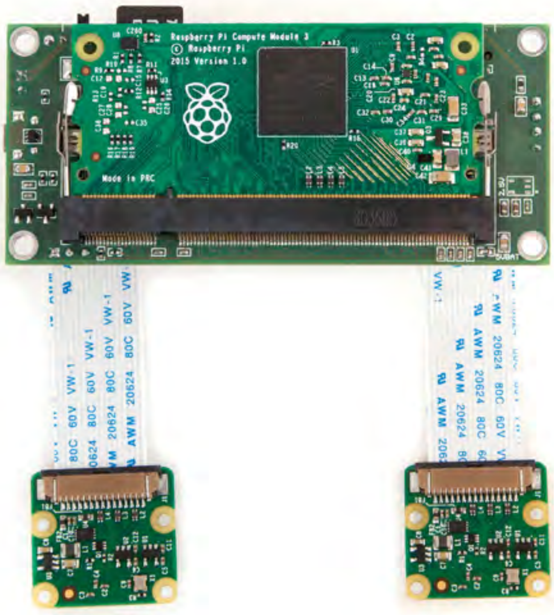
Based in Russia, the team have been making remote-controlled vehicles with real-time video live-streaming since 2010. Projects include controlling a real car and a bulldozer with an iPad.

virt2real.com

Did you know that since 2014, Raspbian has offered built-in support for stereoscopic photography? The only obstacle to making use of it is that standard Raspberry Pi models only have a single Camera port. The Pi Compute Module, on the other hand, supports the connection of two Camera Modules, which is why the Russian virt2real team have used it to power StereoPi – a project that has taken over three years to develop.

"Starting in the middle of 2015, we did our first generation board on Compute Module 1," recalls virt2real's Eugene Pomazov. "We understood that we needed not only to control our drones or robots, but add more autonomous features. The age of AI and computer vision had come.

"We tried to find ready-to-use solutions on the market. But all of them were either expensive industrial-level solutions, or crappy-quality toys. […] That's why we decided to create StereoPi. Two cameras with fully controlled options, friendly Raspberry Pi ecosystem with a lot of ready-to-use examples, and an inexpensive solution with computing power on board which costs about $100-150."



A Raspberry Pi Compute Module is attached to the SODIMM connector on the underside of the StereoPi board

A demo live-streaming video of an aquarium to YouTube; you can view it in 3D on any phone equipped with a 3D viewer, or on an Oculus Go VR headset

Attaching to any Compute Module via its SODIMM connector, the latest version of the StereoPi board features two Camera ports, positioned at the ideal spacing (65 mm) for stereoscopic photography.

Standard Pi ports

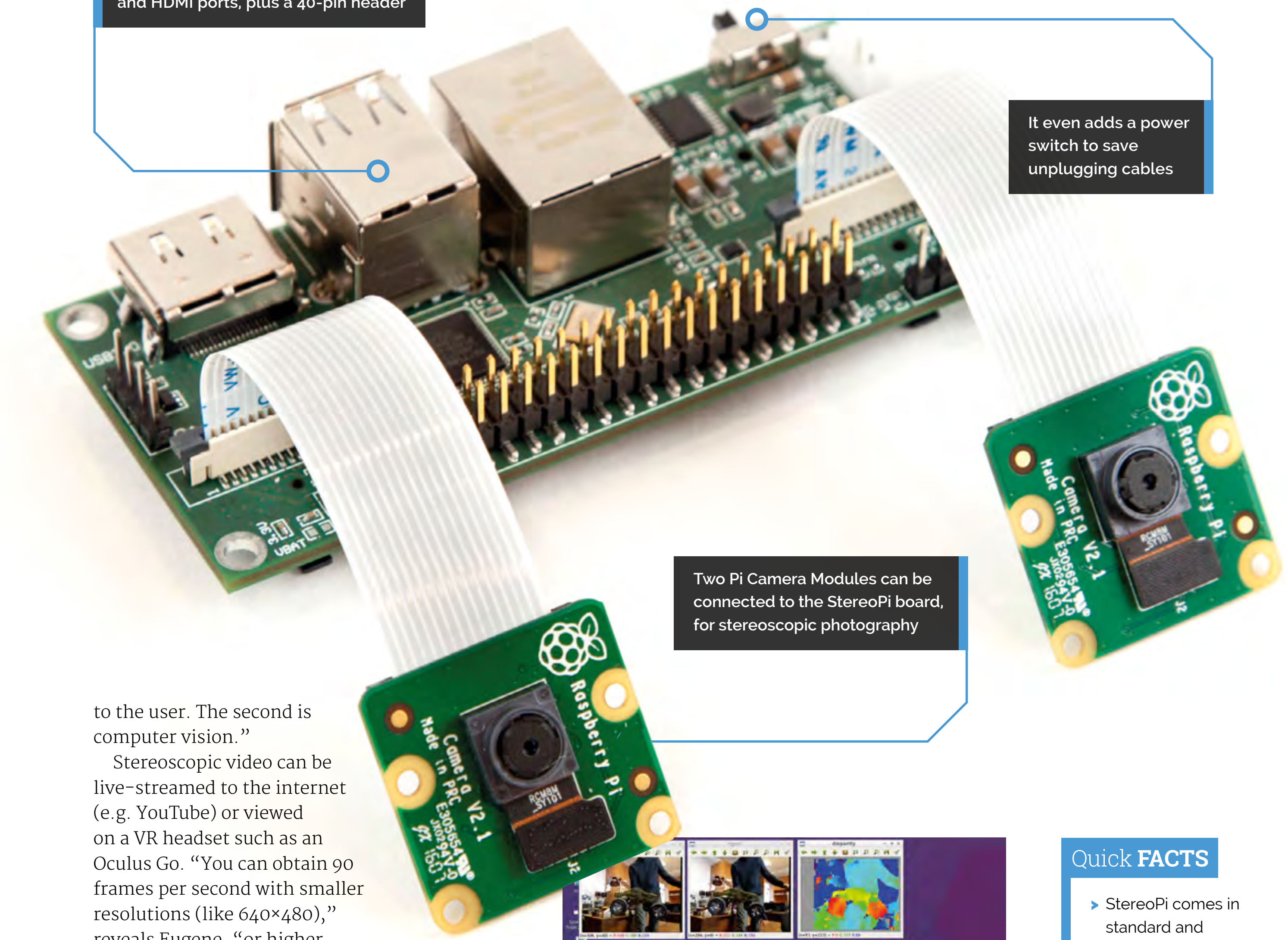
It also features standard Pi model connections such as a 40-pin GPIO header; ports for Ethernet, USB, and HDMI; and a microSD slot. Eugene says the team's favourite addition is a small power switch, enabling you to restart the system without having to unplug the cable or turn off at the mains.

With Raspberry Pi at its core, StereoPi works with the standard Raspbian OS. Since stereoscopic support is already built into the raspistill or raspivid commands, as well as the Python picamera library, it's ready to use out of the box.

Naturally, you'll need some kind of mounting plate for the two cameras, to keep them spaced and aligned correctly. The virt2real team have created a few mounts for different uses, including 3D photography (with cameras side by side) and panoramas (with wide-angle lenses facing in opposite directions). "We plan to share all drawings and ready-to-print STL files so anyone can use and modify them for their needs," says Eugene.

Double the possibilities

Eugene envisages two main application scopes for StereoPi: "The first one is a 3D video live-stream



Powered by a Pi Compute Module, the board features USB, Ethernet, and HDMI ports, plus a 40-pin header

It even adds a power switch to save unplugging cables

Two Pi Camera Modules can be connected to the StereoPi board, for stereoscopic photography

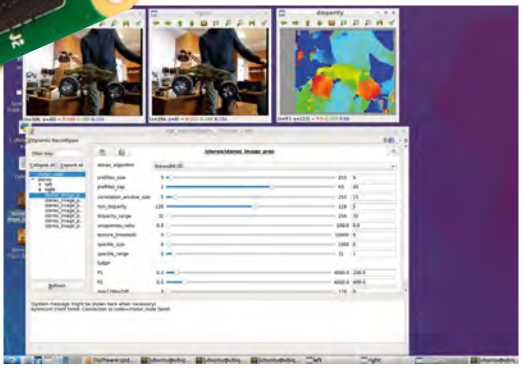
to the user. The second is computer vision."

Stereoscopic video can be live-streamed to the internet (e.g. YouTube) or viewed on a VR headset such as an Oculus Go. "You can obtain 90 frames per second with smaller resolutions (like 640×480)," reveals Eugene, "or higher resolutions at smaller fps (30 with Full HD)."

The compact and light nature of StereoPi makes it particularly useful for attaching to drones and robots. As well as live-streaming video from them, this is where the computer vision aspect comes into play. "Collision avoidance was the first idea that came to our mind when we decided to create StereoPi," explains Eugene. "You can use OpenCV or ROS [Robot Operating System] and on-board computations by means of the Raspberry Pi Compute Module to obtain a depth map or SLAM [simultaneous localisation and mapping]."

He concludes, "In both live-stream and computer vision cases, we think StereoPi will be a great tool for learning and experiments."

If this article has piqued your interest in the possibilities of 3D photography, see StereoPi's crowdfunding campaign at **magpi.cc/JjOTKY**.



▲ A depth map generated in ROS – the StereoPi is ideal for computer vision systems on robots, to aid collision avoidance

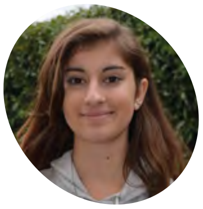
" The compact and light nature of StereoPi makes it particularly useful for attaching to drones and robots "

Quick FACTS

- StereoPi comes in standard and slim editions
- The standard one measures 90×40×23 mm
- The final product will have a black PCB
- It supports v1 and v2 Pi Camera Modules…
- Plus an HDMI video capture module

Firefighter Monitoring System

A switched-on high school student is developing a device to monitor the vital signs of firefighters. **Nicola King** puts her finger on the pulse



MAKER

Parisa Khashayar

A high school freshman, Parisa enjoys teaching younger children how to code while volunteering at her local CoderDojo.

magpi.cc/kGsnNh

When Parisa Khashayar witnessed a firefighter being treated for an injury during a wildfire, she began to ask herself how, or even if, firefighters' health and well-being was monitored while they were out in the field: "Firefighters put themselves in danger to keep us safe, but I asked myself, what keeps them safe? Besides their fire-resistant suit, what other measures are taken to ensure their safety? My research showed that there wasn't a lot of high tech equipment available to monitor their health and warn them of dangerous conditions."

Having confirmed with a local fire station that a device to monitor firefighter health would indeed be useful, Parisa set about designing and building her Monitoring System. "I started planning and designing the device when I was in eighth grade (about a year and a half ago)," she tells us. "I knew some coding, but I had to learn how to connect the hardware pieces together."

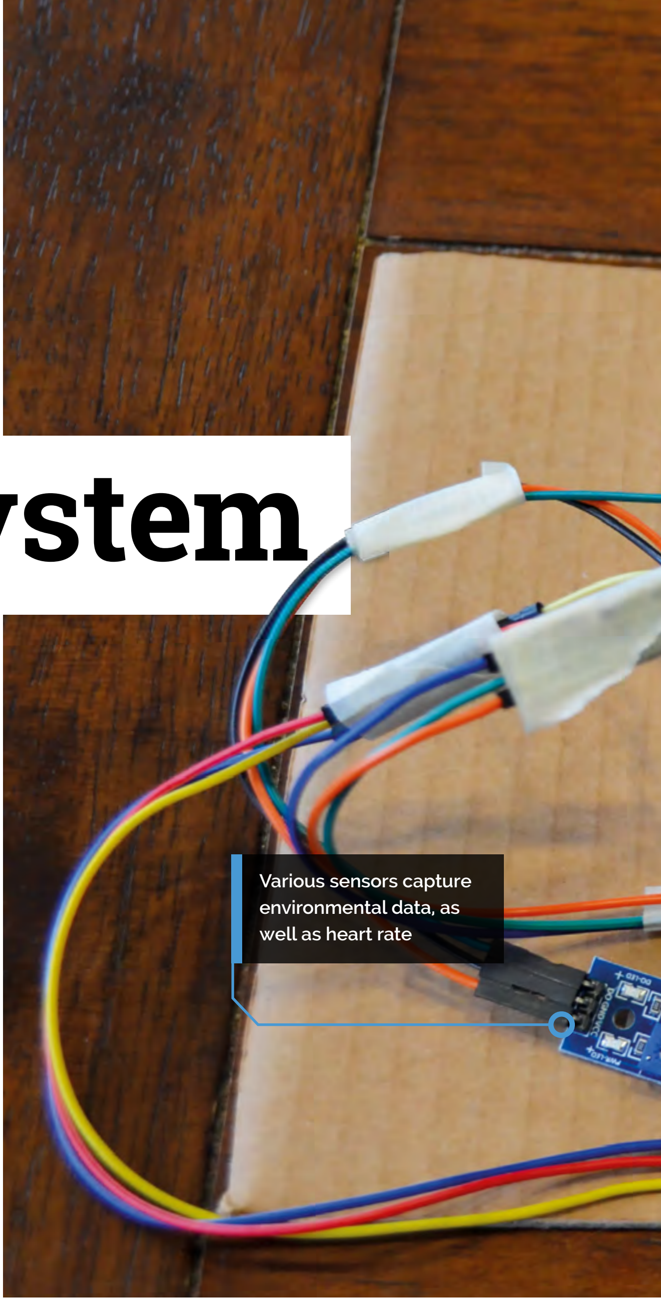
Decisive data

Parisa's device cleverly takes key measurements and compares them to threshold data. "I used a microprocessor and four different sensors to collect

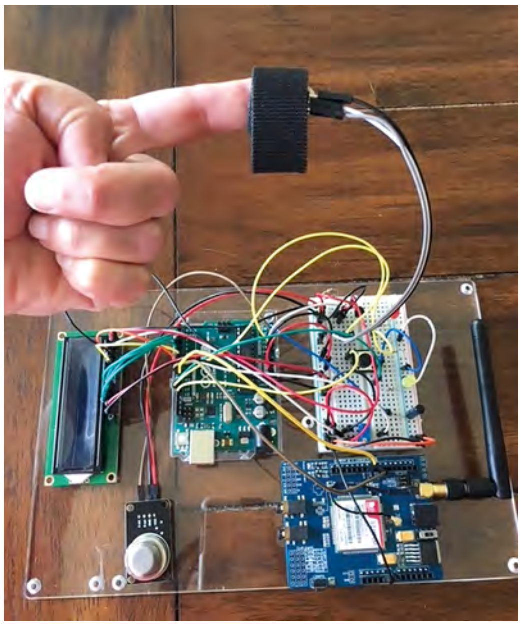
" It is very gratifying to design and build and code a machine that can perform a task and solve a problem "

and measure data about a firefighter's condition as well as their surrounding environment," she explains. "My code takes that information and compares it to a preset threshold; if the data does not fit within that threshold, it will send a message to the nearby command centre through cellular technology."

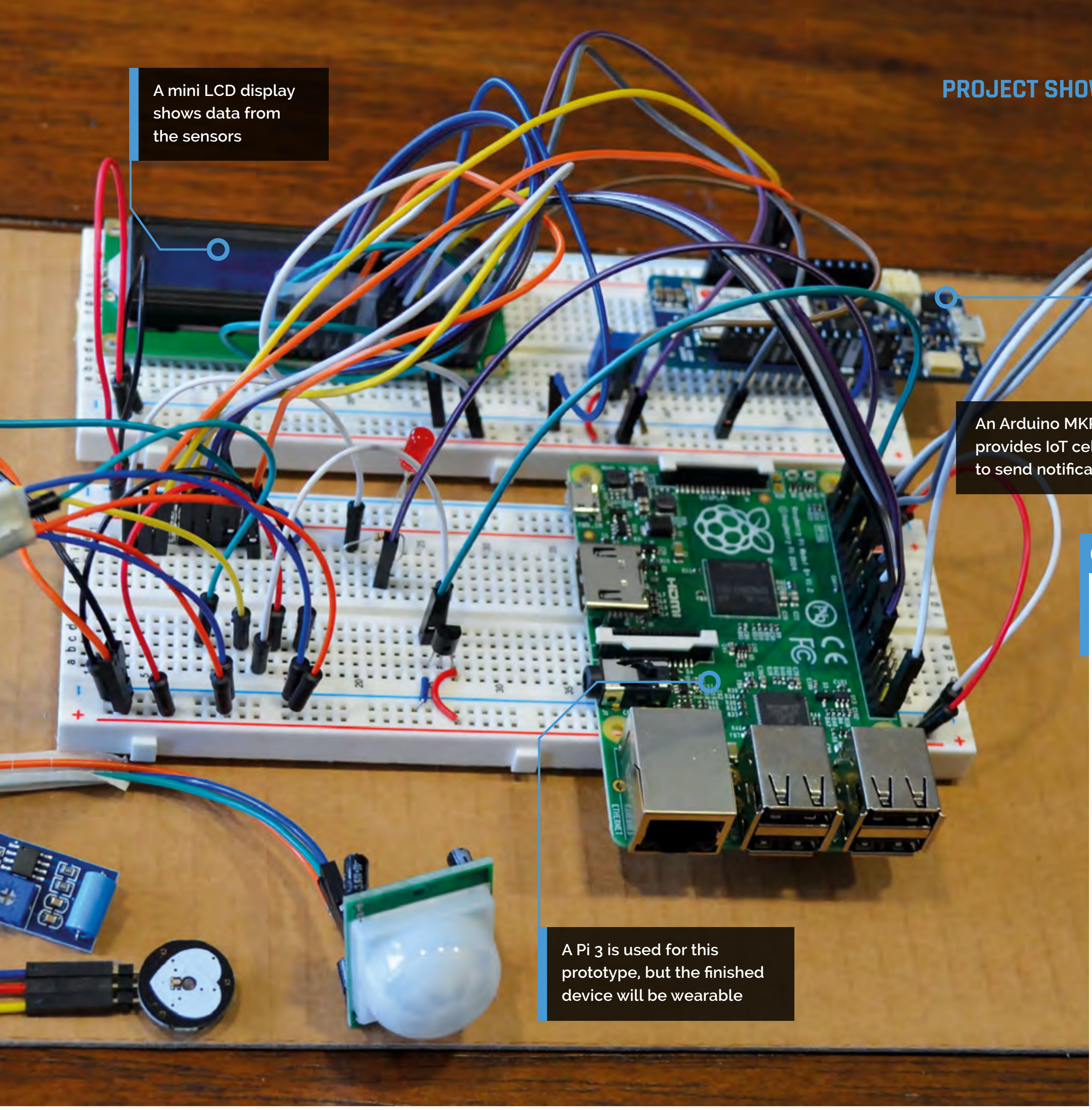
Parisa admits there was a lot of fine-tuning required in order to develop her project. "The hardest part of



Various sensors capture environmental data, as well as heart rate



The original prototype was based on an Arduino


A mini LCD display shows data from the sensors
An Arduino MKR NB 1500 provides IoT cellular connectivity to send notifications
A Pi 3 is used for this prototype, but the finished device will be wearable

Quick FACTS

- Sensors monitor gas, temperature, motion, and heart rate
- She plans to turn it into a Pi-based wearable device
- It won joint first place for hardware at Coolest Projects USA 2018
- Details of Coolest Projects 2019 can be found at **coolestprojects.org**
- Parisa hopes to study biomedical engineering in the future

my first design was definitely coding the heart rate sensor and the GSM. The heart rate sensor needed a lot of calculations to convert the raw data to an actual heart beat and heart rate. The GSM is what I used to send the notification out."

Undeterred, Parisa forged ahead with her design, culminating in the opportunity to showcase her work at Coolest Projects USA in 2018, an exciting technology fair for young people. "We talked about problems we encountered with our projects and exchanged ideas… One of the best parts was meeting the judges; they were incredibly encouraging and excited to find out about everyone's work," she says.

Having used an Arduino Uno for her original prototype, Parisa is now converting the project to Raspberry Pi so that the size of the device can be reduced. Introduced to the Pi at Coolest Projects, she explains that she saw how "powerful, yet simple, it is to program… I plan to get everything into a PCB small enough to fit into a wearable, and test it out with one of the fire stations nearby who agreed to help test out my prototype."

Hot shot

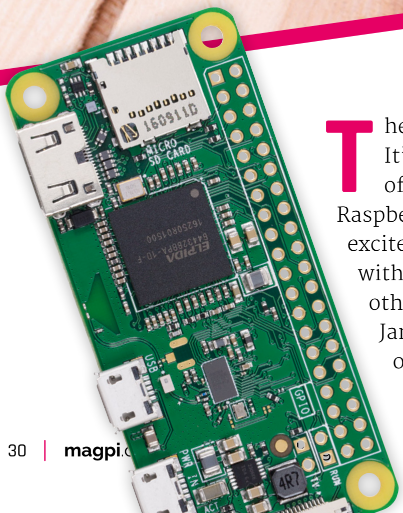
With this exciting, practical, and potentially life-saving innovation, Parisa has proved that coding is her forte, and is something that she thoroughly enjoys, explaining, "It is a way to express myself and [of] connecting myself to the outside world… It is very gratifying to design and build and code a machine that can perform a task and solve a problem."

So, what does the future hold for this very smart student? "I just started high school this year with a full schedule that's kept me busy, so I haven't had much time to get back to the drawing board. I will always be looking for new ways to solve problems but, for now, I'm mostly focused on improving my Firefighter Monitoring System before I move on to another project," says Parisa, who is clearly destined for great things. Watch this space!

▼ Parisa talking about her project at Coolest Projects USA

MAKER SPECIAL! #MONTHOFMAKING

Build your first project or show off your latest work-in-progress in this online community event!



The Raspberry Pi community is huge. It's one of the most amazing parts of everything that makes up the Raspberry Pi – a global collection of people excited to show off what you can do with the Raspberry Pi, and even help others with their projects at Raspberry Jams, Code Clubs, CoderDojos, and other gatherings!

We want to celebrate the community during March with our #MonthOfMaking event: throughout the month, work on a project and share your works-in-progress to social media. That's it! Whether you're new to making or have a Raspberry Pi make you breakfast, we want to see what you can do and flood the internet with creative uses for physical electronics.

Need some ideas of where to start? Read on for some inspiration…

Tools for the job

To make something cool, you might need some tools

Electronics

Circuits can connect your Raspberry Pi and microcontrollers to lights, buttons, speakers, or even a full-on robot. Here are some of the tools of the trade.

- **Wire**
- **Jumper cables**
- **Breadboard**
- **LEDs**
- **Buttons**
- **Motors**
- **Soldering iron**

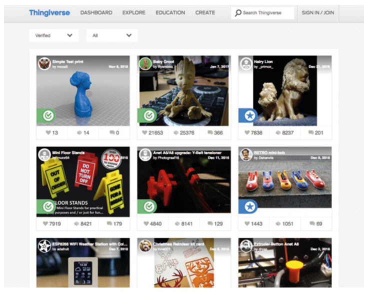
Wood and metal

Old-school building methods can sometimes require some serious hardware. If you're new to this, ask an experienced adult for some tips.

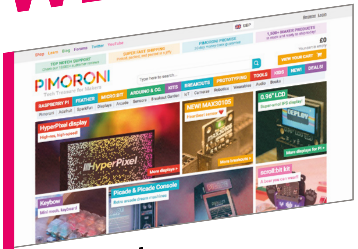
- **Saw**
- **Welder**
- **Vice**
- **Drill**
- **Glue**
- **Sander**
- **Jigsaw**
- **Measuring tools**

3D printing and plastics

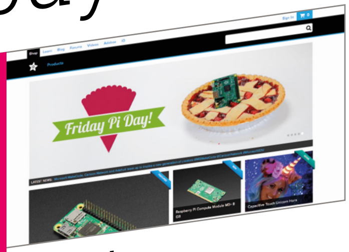
Don't try this at home… unless you have an expensive 3D printer or laser cutter. If you don't, you can find designs for 3D printers on websites such as Thingiverse (**thingiverse.com**) and print them on 3D Hubs (**3dhubs.com**).



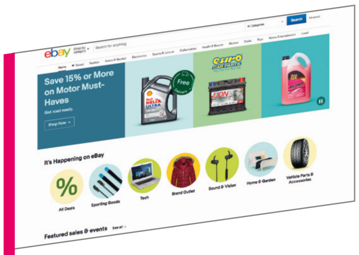
Where to buy



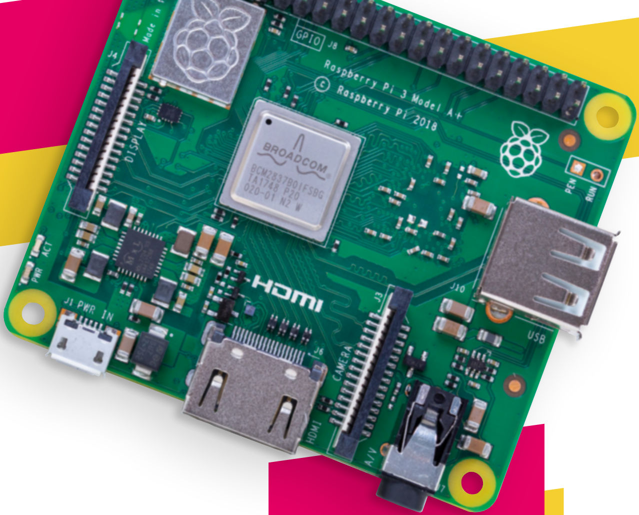
Pimoroni
shop.pimoroni.com
A UK supplier of great Raspberry Pi and maker tech



Adafruit
adafruit.com
A global leader in amazing maker components and hardware



eBay
ebay.co.uk
Purveyors of cheap things for pretty much any build you want to do



Ideas for maker starters

New to making? Here are some ideas to help you get started

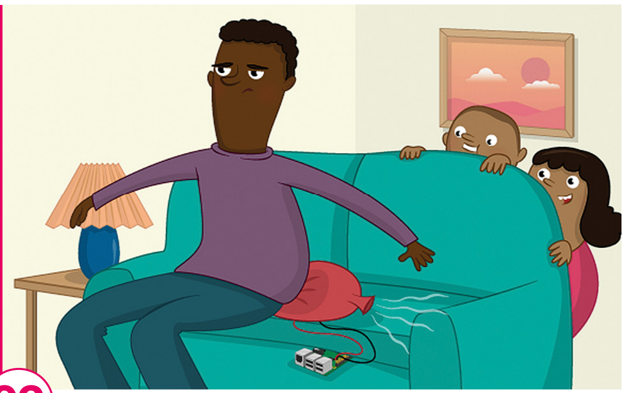
Everyone needs to start somewhere. All the most amazing makers in the world at some point had to figure out 'Hello World', how to control an LED with code, and how to build.

Don't be afraid to share your first forays into the world of making. The rest of us are here to help and appreciate your work! To give you that first boost, here are some beginner projects that you might find useful.

01 Beginner Wearables Sushi Cards

This is a fun wearable project that uses simple circuits and pre-designed parts to create some fun-looking clothing items. Perfect for showing off your new-found making skills!

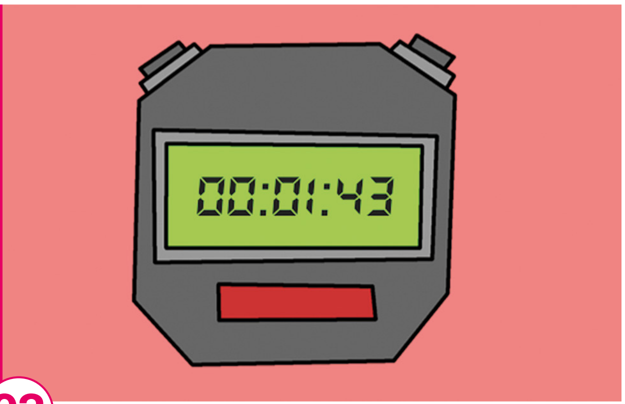
magpi.cc/ZjUjiS



02 Whoopi Cushion

What's the point of making if you're not having fun doing it? We particularly love this project as it's very cheap to make, and includes some real DIY skills to create a big button for the device.

magpi.cc/2AgN6W

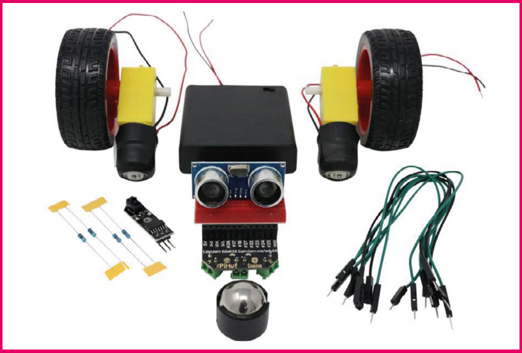


03 Countdown Timer

This simple project shows you how to use a Sense HAT – an environmental sensor with a simple display add-on for the Pi – as a countdown timer. You can probably also turn it into a stop-watch as well.

magpi.cc/KnaWBY

Fun kits



CamJam EduKit 3 | £18

This inexpensive kit lets you use your Raspberry Pi to create a robot that can be fully automated – the box itself may be used as a chassis! It's also supported in GPIO Zero, the short-hand Python programming library for physical computing that comes with Raspbian on the Raspberry Pi.
magpi.cc/RhpjZh

Mood Light | £32

Unlike some other kits, this one comes with a Pi Zero W and pretty much everything else you need to construct it. It teaches you about basic programming, and also has a load of great components you can use in future projects.
magpi.cc/xqySvs

Picade Console | £60

A slightly more expensive kit, but a very fun one: create your own retro gaming console in the form of a classic arcade stick and buttons! It's a great build, and some of the parts are really useful for future projects.
magpi.cc/BSeTDD

Learning resources from Raspberry Pi

Part of the Raspberry Pi Foundation's mission is to bring free computing education to everyone, and the Raspberry Pi projects site has programming, making, and other computer-based tutorials for you to learn from. You can find it at **rpf.io/projects**, where you'll even find more making ideas while perusing tutorials on 'Physical computing with Python' (**magpi.cc/2mjGMJZ**) or 'Getting started with picamera' (**magpi.cc/QrMnng**).

Inspirational makers

Here are some folks to follow on YouTube!

Estefannie Explains It All

Software engineer and maker of fun projects
magpi.cc/vDhJBq

Jabrils

Maker who specialises in AI
magpi.cc/foAxEd

Tinkernut

Making, mending, and hacking
magpi.cc/pQNeif

Junie Genius

Science, making, and everything in between
magpi.cc/oSkvRG

Blitz City DIY

A maker who dabbles a little bit more in the art side
magpi.cc/FDpqSi

Ideas for maker hobbyists

If this isn't your first maker rodeo, here are some fun projects to try out

Choose a theme!

Not sure where to start? Maybe picking a theme for your build(s) will help!

Robots

IoT

Home media

Home automation

Home office

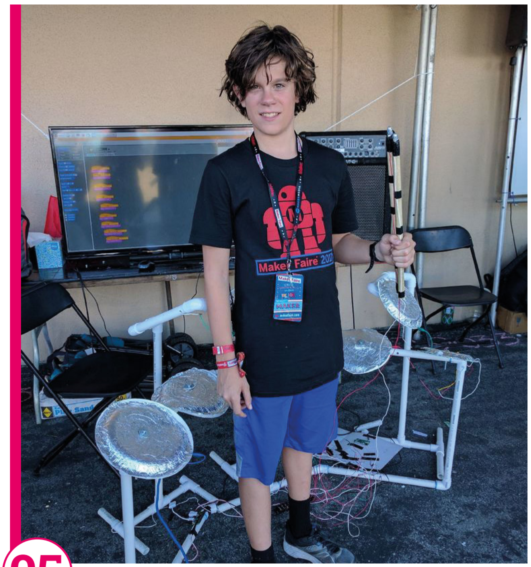
Photo and video

Every day, we see yet another incredible project or make that someone has done. It's almost impossible to run out of ideas, it seems – but sometimes we need a bit of inspiration. Here are some projects to get your brain cells firing.

04 Amazon Echo Controlled Curtains

AI and voice control is widely accessible on Raspberry Pi, thanks to Amazon's Alexa, Google's AIY, and even Microsoft's Cortana. This project marries it with some home automation to open and close curtains in your house. You can adapt it to control other automated devices, such as lights or a garage door.

magpi.cc/AEWhkN



05 Raspberry Pi Drum Kit

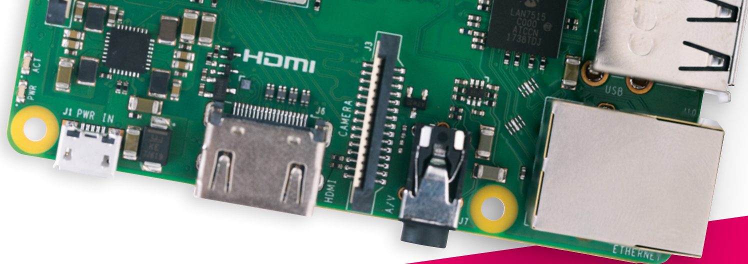
This project is the ultimate in DIY, and very cheap to make as well! It uses a load of PVC pipes, paper plates, wire, and lots of other bits that you can find in a DIY store. Oh, and it also requires two acorn nuts. It's programmed in Scratch, making it very accessible to young makers.

magpi.cc/JzdHuD

06 DiddyBorg v2

How about building a bigger robot? The DiddyBorg v2 is one of our favourite robot kits, and you can customise it very heavily to do some amazing automated things. The best part is: all the parts are top-quality and easily transferable to even bigger and better robots!

magpi.cc/SHBgNc

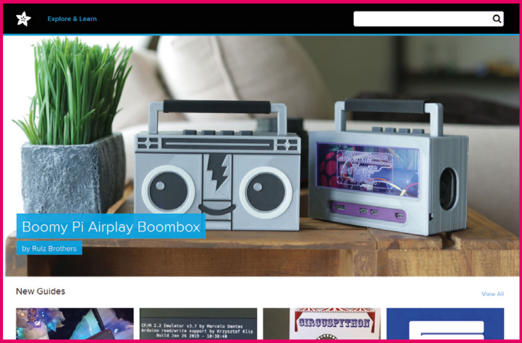


07 Magic Mirror

This classic project is almost a rite of passage for Raspberry Pi owners. It's a lot simpler than it sounds, as well, thanks to Michael Teeuw's excellent software. We did a full big build of this in issue 54 of *The MagPi* (**magpi.cc/54**), and you can now find loads of different features to add to this kind of project.

magicmirror.builders

Maker resources



Adafruit Learning System

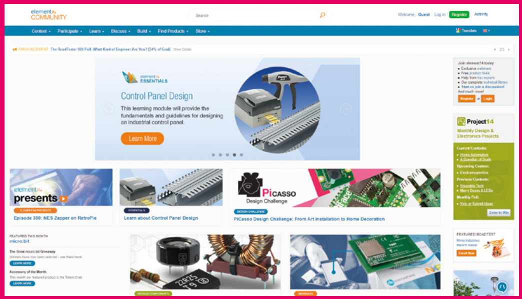
As well as buying great hardware from Adafruit, you can find amazing projects with fantastic, thorough breakdowns of how to build them. There's always something cool and new on the website, covering a wide range of different project types!

learn.adafruit.com

HackSpace magazine

Our sibling publication, HackSpace magazine is an incredible resource for making incredible things, and not just with the Raspberry Pi. Of course, we also have great projects and tutorials, but you've already read every issue of *The MagPi*. Right?

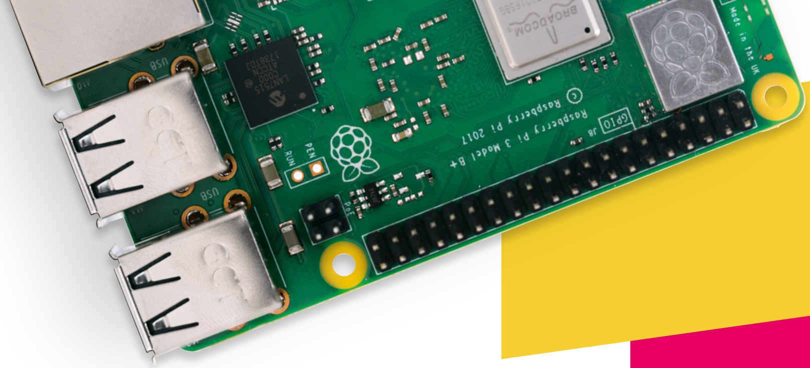
hsmag.cc



element14 community

Not only a Raspberry Pi reseller, but also an amazing community of makers taking part in challenges and showing off what they've made. It's definitely worth a look at what's been made recently, and what the current theme of the month is!

magpi.cc/GRmbXe



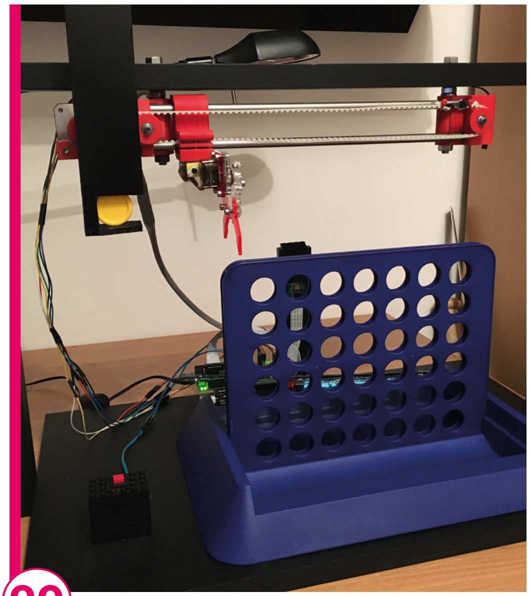
Ideas for expert makers

Feel like a true challenge? Then get your arc welder ready...

Everyone likes a monster build. Something impressive. If you're the kind of maker who builds fire-breathing dragon vans in their sleep, then here are some awesome challenges that you might be up to.

08 Arcade cabinet

Sure, you can use a kit to make a little box that sits on your counter, but to build a full-on arcade cabinet from scratch, complete with working lights and art and everything, now that takes some time and skill. We especially like this version made by I Like To Make Stuff.
magpi.cc/sniEhi

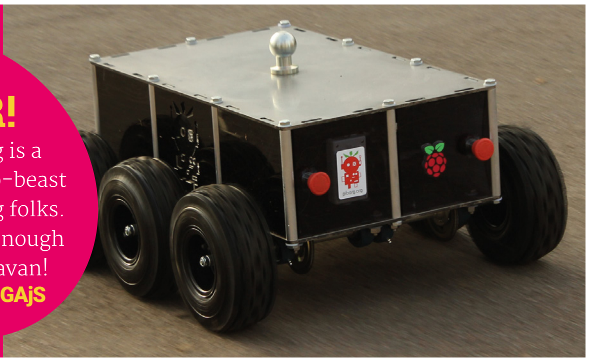


09 4-Bot

Robots aren't all just the little drivable kits you can buy – there are also robot arms, bipedal robots, and other kind of walker bots. We quite like the 4-Bot here: it's a robot you can play a game of Connect 4 with. It uses computer vision to see how you've played and calculates its next move. We've also seen chess-playing robot hands. So as long as you can code it, you can make it.
magpi.cc/1XrC3zU

Go BIGGER!

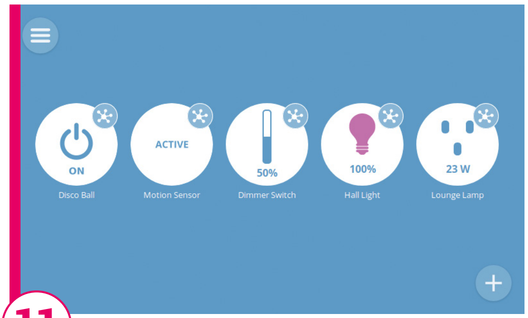
The DoodleBorg is a six-wheeled robo-beast from the PiBorg folks. It's powerful enough to tow a caravan!
magpi.cc/ieGAjS



Go BIGGER!
OpenROV makes amazing Pi-powered underwater drones called Trident.
openrov.com

10 Underwater drone/ROV

Whenever you mix water and electronics, you're always going to have a tough time. The true trick with a submersible ROV is to keep the electronics dry, while still being able to control it remotely. Oh, and did we mention the balancing issues with keeping something underwater? It's quite a precise operation, which makes it great for a big challenge.
magpi.cc/NLAiTh



11 Smart home

What if you could control your own home through a Raspberry Pi? We've seen a few people attempt it with varying success. Mozilla's Project Things has a good post on how you can use its tech to completely automate your home – give it a look!
magpi.cc/avfrFj

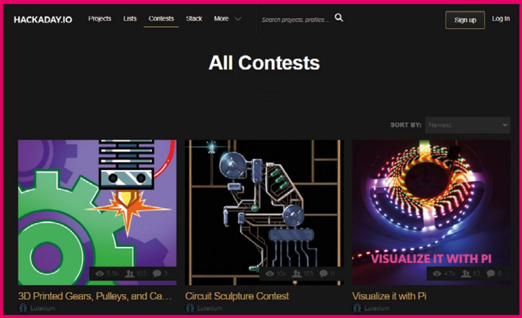
Get competitive

Pi Wars

The yearly competition for Raspberry Pi robot makers involves many non-violent competitions so that your robot gets to go home with all its original paintwork and/or wheels. It's possibly a little too late to apply for this one, but next year's will be here before you know it.
piwars.org

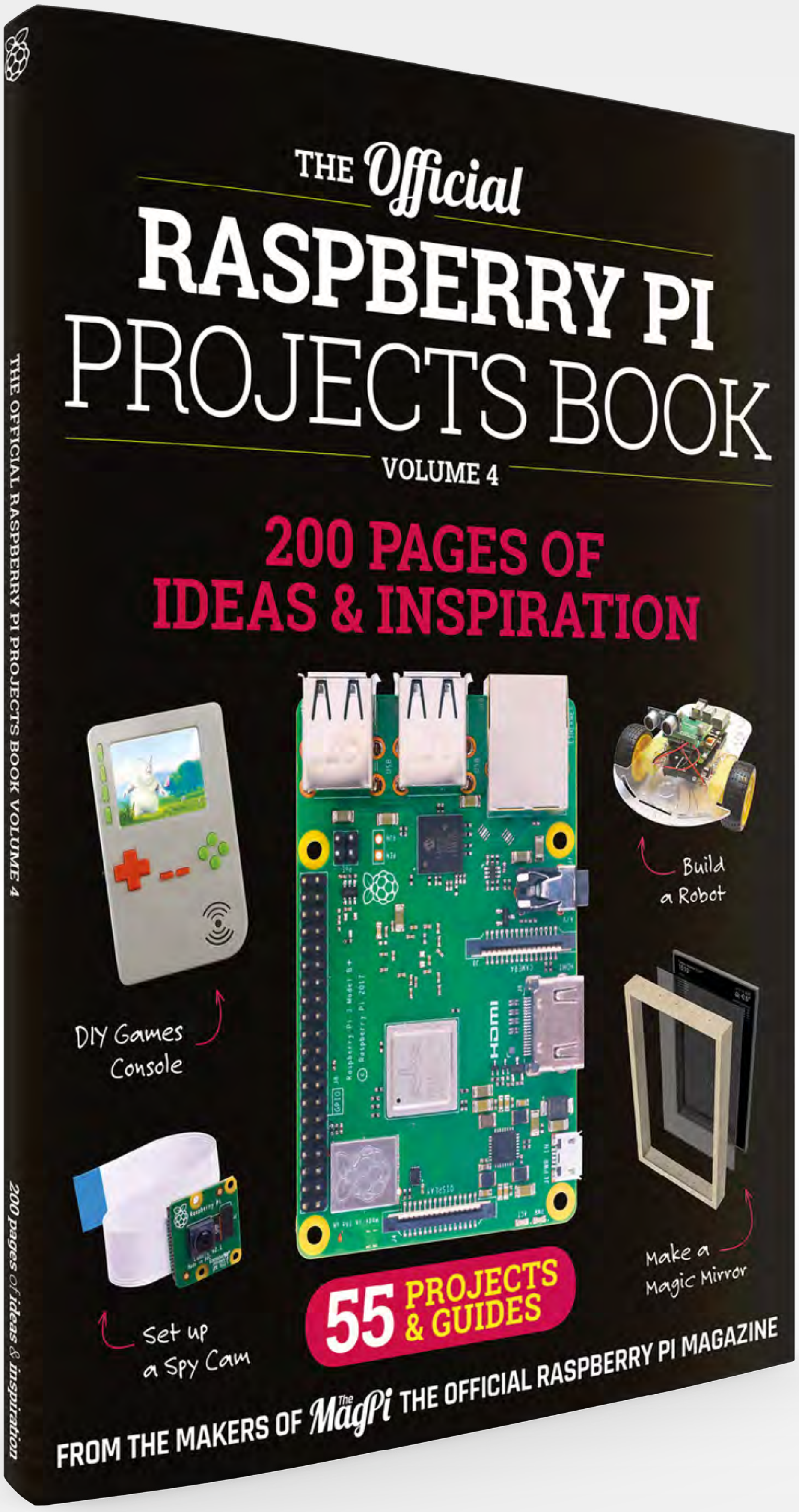
Makevember

#MonthOfMaking isn't the first maker-focused hashtag – if you really fancy a challenge, #makevember in November tasks makers with making something every day! While you probably won't be doing anything near as big as you might make over a month, it's still fun for experimenting and being creative.



Hackaday

As well as being a great repository for cool projects and builds, there are regular contests that anyone can enter. There's even a yearly prize for the best build! If you're making something truly spectacular, don't miss out on putting it up on here.
hackaday.io/contests



THE *Official*

RASPBERRY PI

PROJECTS BOOK

VOLUME 4

200 PAGES OF IDEAS & INSPIRATION

55 PROJECTS & GUIDES

FROM THE MAKERS OF *The MagPi* THE OFFICIAL RASPBERRY PI MAGAZINE

THE OFFICIAL RASPBERRY PI PROJECTS BOOK VOLUME 4

200 pages *of* ideas & inspiration

Build a Teachable Machine with Google's Edge TPU

Use the Coral USB Accelerator and Raspberry Pi Camera to build a device that can be taught to recognise objects. By **Lucy Hattersley**

coral.withgoogle.com

MAKER

Mike Tyka

Mike works with artificial neural networks as an artistic medium and tool. He created some of the first large-scale artworks using Iterative DeepDream and collaborated with Refik Anadol to create pioneering immersive projection installations using Generative Adversarial Networks called Archive Dreaming. Mike currently works on machine learning at Google in Seattle.

miketyka.com

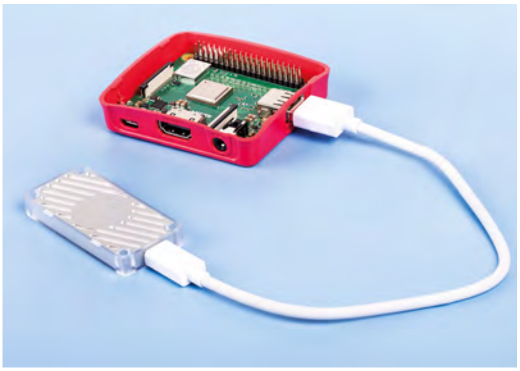
You'll Need

- Raspberry Pi 3B / 3B+
- Coral USB Accelerator ($74.99)
- Raspberry Pi Camera Module
- 8GB microSD card
- Raspbian Stretch with Desktop image
- Breadboard
- 5 × Push-button switches
- 10 × Jumper wires (male to female)
- 9 × Short jumper wires (male to male)
- 4 × Resistors (330 Ω)
- 4 × LEDs

Coral's new USB Accelerator lets you to build AI capabilities into any Raspberry Pi project. The accelerator is built around Google's Edge TPU chip, an ASIC that greatly speeds up neural network performance on-device.

When combined with the Raspberry Pi and Raspberry Pi Camera, you have a complete system that's perfect for executing complex computer vision tasks like object recognition. Since the Accelerator operates locally, you do not need to connect to a cloud service or share secure data over the internet. It also runs with less latency than a cloud connection, performing object detection in near real time.

Using the Raspberry Pi Camera Module, all Raspberry Pi boards can easily gain photographic functionality. And with its built-in GPIO pins, you can prototype circuits and even integrate the Raspberry Pi into projects and industrial environments. Throw a USB Accelerator into the mix and you have a capable AI device, ready to take on all kinds of tasks.



The USB Accelerator attached to a Raspberry Pi

In this tutorial we're going to build a Teachable Machine. This project was designed by Google's Mike Tyka.

The Teachable Machine learns to identify objects held up in front of it. These objects can be anything: keys, fruit, chess pieces, or even fingers or faces.

The user holds up items in front of the Raspberry Pi Camera Module and presses a button on the Teachable Machine. The device then remembers the object being held up – if it sees it again, it will light up the corresponding LED.

The Teachable Machine is a great example of how to add a layer of machine learning technology sparkle to your projects without having to train a whole network from scratch.

The Teachable Machine can distinguish objects quickly and effectively (even from different orientations). This sort of project wasn't possible even a few years ago on a powerful computer with an expensive graphics card. And now we can add it to a small single-board Raspberry Pi computer.

01 Start with Raspbian

Start by flashing a fresh installation of Raspbian Stretch with Desktop to a microSD card (**magpi.cc/quickstart**).

Set up your Pi with the Raspberry Pi Camera Module attached to the camera socket via a 15-way ribbon cable and insert the microSD card. Add power to start up the Raspberry Pi.

The USB Accelerator enables a Raspberry Pi to perform AI tasks in real-time

02 Set up the camera

When the Raspbian OS boots, click the Raspberry Pi menu icon in the top-left and choose Preferences > Raspberry Pi Configuration.

Click Interfaces and set Camera to Enabled and click OK. A 'Reboot needed' alert will appear; click Yes.

Open a Terminal window (**CTRL+ALT+T**) and take a test shot and open it:

```
raspistill -v -o test.jpg
xdg-open /home/pi/test.jpg
```

03 Set up USB Accelerator

Make sure the USB Accelerator device is not connected while you set it up (disconnect the device if you plugged it in). Open a Terminal and enter these commands to download and install the software:

```
 wget http://storage.googleapis.com/cloud-
iot-edge-pretrained-models/edgetpu_api.
tar.gz
 tar xzf edgetpu_api.tar.gz
 cd python-tflite-source
 bash ./install.sh
```

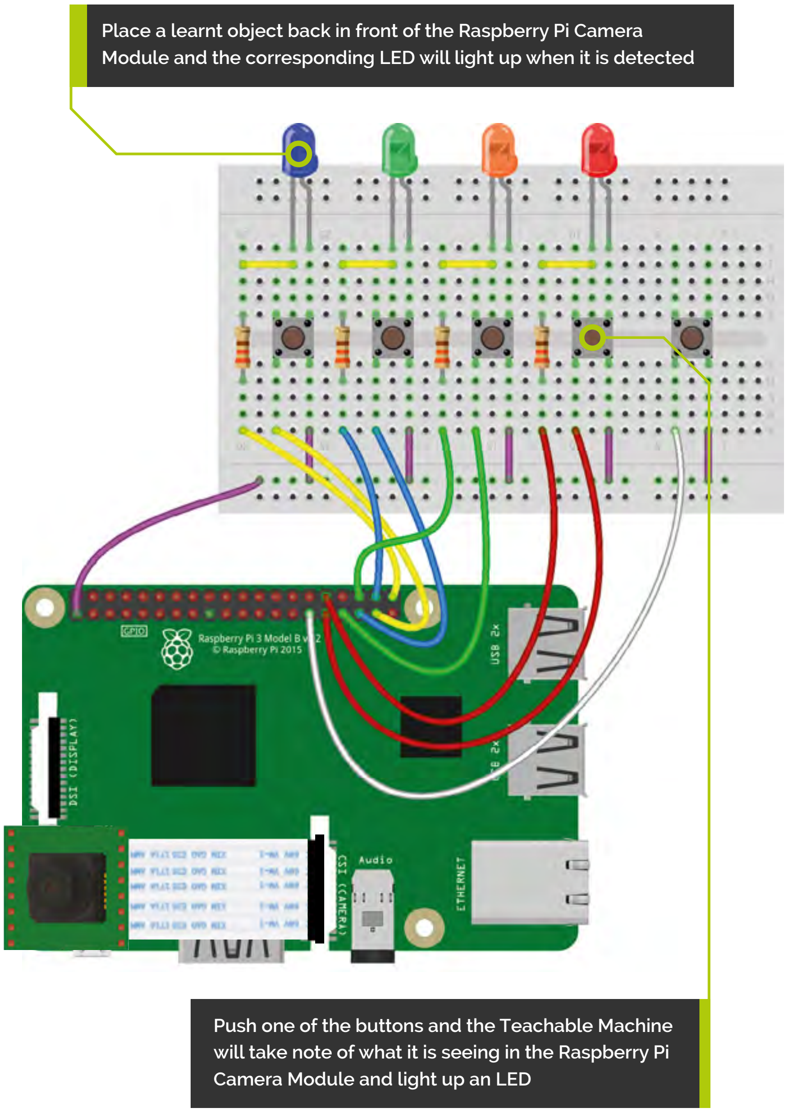
During the installation, it will say: 'Would you like to enable the maximum operating frequency?' Answer 'n' for now. You can enable it later if you want to increase the performance.

04 Test USB Accelerator

Now plug in the USB Accelerator using the supplied USB Type-A to USB Type-C cable. Your USB Accelerator is now set up.

See Coral's 'Get started with the USB Accelerator' document for more information on setting up and testing: **g.co/coral/setup**.

Figure 1


Place a learnt object back in front of the Raspberry Pi Camera Module and the corresponding LED will light up when it is detected
Push one of the buttons and the Teachable Machine will take note of what it is seeing in the Raspberry Pi Camera Module and light up an LED
GPIO
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
USB 2x
USB 2x
DSI (DISPLAY)
CSI (CAMERA)
Audio
ETHERNET

05 Download a model

Let's take a look at the included demo code. The **classify_capture.py** script works with the Raspberry Pi Camera Module to perform live image classification of objects around us.

What's needed is a model trained to detect some commonplace objects. Download this MobileNet model trained to recognise 1000 objects:

```
 wget -P test_data/ https://storage.
googleapis.com/cloud-iot-edge-pretrained-
models/canned_models/mobilenet_v2_1.0_224_
quant_edgetpu.tflite
```

Careful!

The USB Accelerator can become hot during use. Avoid touching it while the project is running.

Top Tip

How it works: transfer learning

The Teachable Machine uses an approach called 'transfer learning'. In this technique, makers start with an already trained model, and customise it for the task in hand.

The Teachable Machine uses a headless MobileNet model, in which the last layer (which makes the final decision on the 1000 training classes) has been removed, exposing the output vector of the layer before. The Teachable Machine treats this output vector as an embedding vector for a given image.

More info on the details of the algorithm and transfer learning can be found here: **magpi.cc/LeKzkc**.

Now get the corresponding labels:

```
 wget -P test_data/ http://storage.
googleapis.com/cloud-iot-edge-pretrained-
models/canned_models/imagenet_labels.txt
```

This model and label set can be found (along with many other models) on the Coral website (**coral.withgoogle.com**).

06 Live camera detection

We now have a pre-trained model for 1000 objects and a corresponding label set, plus a live capture camera script. Put all three together and you can run an object-detecting camera.

```
 python3 demo/classify_capture.py --model
test_data/mobilenet_v2_1.0_224_quant_
edgetpu.tflite --label test_data/imagenet_
labels.txt
```

Move the Raspberry Pi Camera Module around the room and the preview window will identify the objects around you: laptop, mouse, soda can, and so on.

The **classify_capture.py** program uses two options:

```
 --model
 --label
```

And each one links to the model and corresponding labels downloaded in Step 5.

This is a TensorFlow Lite model (hence the .tflite extension) that has been pre-trained to detect 1000 objects. Typically, training will have taken place on a much faster computer, or cloud service, using thousands of train and test images. During training, the model gradually gets better at matching the images to the label list. When it's good enough, we use it on the Raspberry Pi.

Top Tip

Get models

Google has a selection of models available on its Coral website: **magpi.cc/OmyrGC**

07 The Teachable Machine

It's time to bring the technology together to build something. We're going to build the Teachable Machine project. Like our Classify Capture demo, it scans for objects using the Raspberry Pi Camera Module. However, the Teachable Machine learns to detect the objects you are holding up in front of it.

Start by installing the additional dependencies:

```
 sudo apt-get install libgstreamer1.0-0
gstreamer1.0-tools gstreamer1.0-
plugins-base gstreamer1.0-plugins-good
gstreamer1.0-plugins-bad gstreamer1.0-
plugins-ugly python3-gst-1.0 python3-gi
```

Now add bcm2835-v4l2 to the end of **/etc/modules**:

```
 sudo cat bcm2835-v4l2 >> /etc/modules
```

08 Set up the Teachable Machine

Turn off the Raspberry Pi with this Terminal command (or choose Menu > Shutdown and click Shutdown):

```
 sudo shutdown -h now
```

Remove the power from the Raspberry Pi and set up the Teachable Machine breadboard with the switches and LEDs as shown in the circuit diagram (**Figure 1**). We find it best to lay the Raspberry Pi Camera Module flat on a surface so it is pointing upwards.

Reattach the power once the circuit is set up.

09 Install the code

Open the Chromium browser and visit our GitHub page (**magpi.cc/github79**).Download the **teachable_rpi3.tgz** file from there to your home folder. Open a Terminal window and extract the code:

```
 tar xvzf teachable_rpi3.tgz
 cd /home/pi/teachable/
```

embedding.py

DOWNLOAD THE FULL CODE:

magpi.cc/github79

Language: **Python**

```python
"""Detection Engine used for detection tasks."""
from collections import Counter
from collections import defaultdict
from edgetpu.basic.basic_engine import BasicEngine
import numpy as np
from PIL import Image


class EmbeddingEngine(BasicEngine):
  """Obtains embedding from headless mobilenets."""

  def __init__(self, model_path):
    """Creates a EmbeddingEngine with given model.
    Args:
      model_path: Path to TF-Lite Flatbuffer file.
    """
    BasicEngine.__init__(self, model_path)
    output_tensors_sizes = self.get_all_output_
tensors_sizes()
    if output_tensors_sizes.size != 1:
      raise ValueError(
          ('Model should have only 1 output tensor'
           'This model has {}.'.format(
               output_tensors_sizes.size)))

  def DetectWithImage(self, img):
    """Calculates embedding from an image.
    Args:
      img: PIL image object.
    Returns:
      Embedding vector as np.float32
    """
    input_tensor_shape = self.get_input_tensor_
shape()
    if (input_tensor_shape.size != 4 or
        input_tensor_shape[3] != 3 or
        input_tensor_shape[0] != 1):
      raise RuntimeError(
          'Invalid input tensor shape! '
          'Expected: [1, width, height, 3]')
    required_image_size = (input_tensor_shape[1],
                           input_tensor_shape[2])
    with img.resize(required_image_size,
                    Image.NEAREST) as resized:
      input_tensor = np.asarray(resized).flatten()
      # Run model on accelerator
      return self.RunInference(input_tensor)[1]


class kNNEmbeddingEngine(EmbeddingEngine):
  """Extends embedding engine to also provide
     kNearest Neighbor detection. This class
     maintains an in-memory store of embeddings and
     provides functions to query with a new query
     embedding and find nearest neighbors.
  """

  def __init__(self, model_path, kNN=3):
    """Creates a kNNEmbeddingEngine with given model
    Args:
      model_path: path to TF-Lite Flatbuffer file.
      kNN: how many nearest neighbors to consider.
    """
    EmbeddingEngine.__init__(self, model_path)
    self.clear()
    self._kNN = kNN

  def clear(self):
    """Forgets all stored embeddings."""
    self._labels = []
    self._embedding_map = defaultdict(list)
    self._embeddings = None

  def addEmbedding(self, emb, label):
    """Add an embedding vector to the store."""
    # Normalize the vector
    normal = emb/np.sqrt((emb**2).sum())
    # Add to store, under "label"
    self._embedding_map[label].append(normal)

    # Expand labelled blocks of embeddings for when
    # we have less than kNN examples. Otherwise
    # blocks that have more examples unfairly win.
    emb_blocks = []
    self._labels = []
    for label, embeds in self._embedding_map.items():
      emb_block = np.stack(embeds)
      if emb_block.shape[0] < self._kNN:
          pads = [(0,self._kNN - emb_block.shape[0]),
                  (0,0)]
          emb_block = np.pad(emb_block, pads,
                             mode="reflect")
      emb_blocks.append(emb_block)
      self._labels.extend([label]*emb_block.shape[0])
    self._embeddings = np.concatenate(emb_blocks,
                                      axis=0)

  def kNNEmbedding(self, query_emb):
    """Returns the self._kNN nearest neighbors to a
       query embedding."""

    # If we have nothing stored, the answer is None
    if self._embeddings is None: return None

    # Normalize query embedding
    norm = np.sqrt((query_emb**2).sum())
    query_emb = query_emb/norm

    # We want a cosine distance from query to each
    # stored embedding. A matrix multiplication can
    # do this in one step, resulting in a vector of
    # distances.
    dists = np.matmul(self._embeddings, query_emb)

    # If we have less than self._kNN distances we
    # can only return that many.
    kNN = min(len(dists), self._kNN)

    # Get the N smallest distances.
    n_argmax = np.argpartition(dists, -kNN)[-kNN:]

    # Get the corresponding labels associated with
    # each distance.
    labels = [self._labels[i] for i in n_argmax]

    # Return the most common label over all
    # self._kNN nearest neighbors.
    most_common_label = Counter(labels)
    return most_common_label.most_common(1)[0][0]

  def exampleCount(self):
    """Returns the size of the embedding store."""
    return sum(len(v) for v in
               self._embedding_map.values())
```

The USB Accelerator is part of Coral's line of AI-related products

All the code needed to run the Teachable Machine is found (and is run from) this directory. The **embedding.py** code listing demonstrates the most critical parts. Enter this code to test out the circuit:

```
sudo python3 teachable.py --testui
```

The LEDs will flash to indicate that the circuit is working. Press the buttons and the Terminal will display which button is working (between 0 and 4). Press **CTRL+C** to interrupt the program.

10 Run the Teachable Machine

Now it's time to run our Teachable Machine and see what it is capable of doing. Enter:

```
sh run.sh
```

The Teachable Machine will initialise its model and start running. The LED in the USB Accelerator should be glowing and the Terminal will display a frame rate (typically around 30 fps). Hold an item (such as a piece of fruit or a computer mouse) above the camera and press one of the buttons that's paired with an LED (the LED will light up next to the button). Let go of the button and move the item away.

Now, when you bring the item back in front of the camera, the corresponding LED will light up. Keep bringing different items in front of the camera and pressing different buttons to train the machine to detect each one. It's helpful to dedicate one button to just background (with no item held).

If the Teachable Machine seems a little unsure, you can always press the button again to retrain it. It is good practice to press the button multiple times and rotate the object slightly with each press.

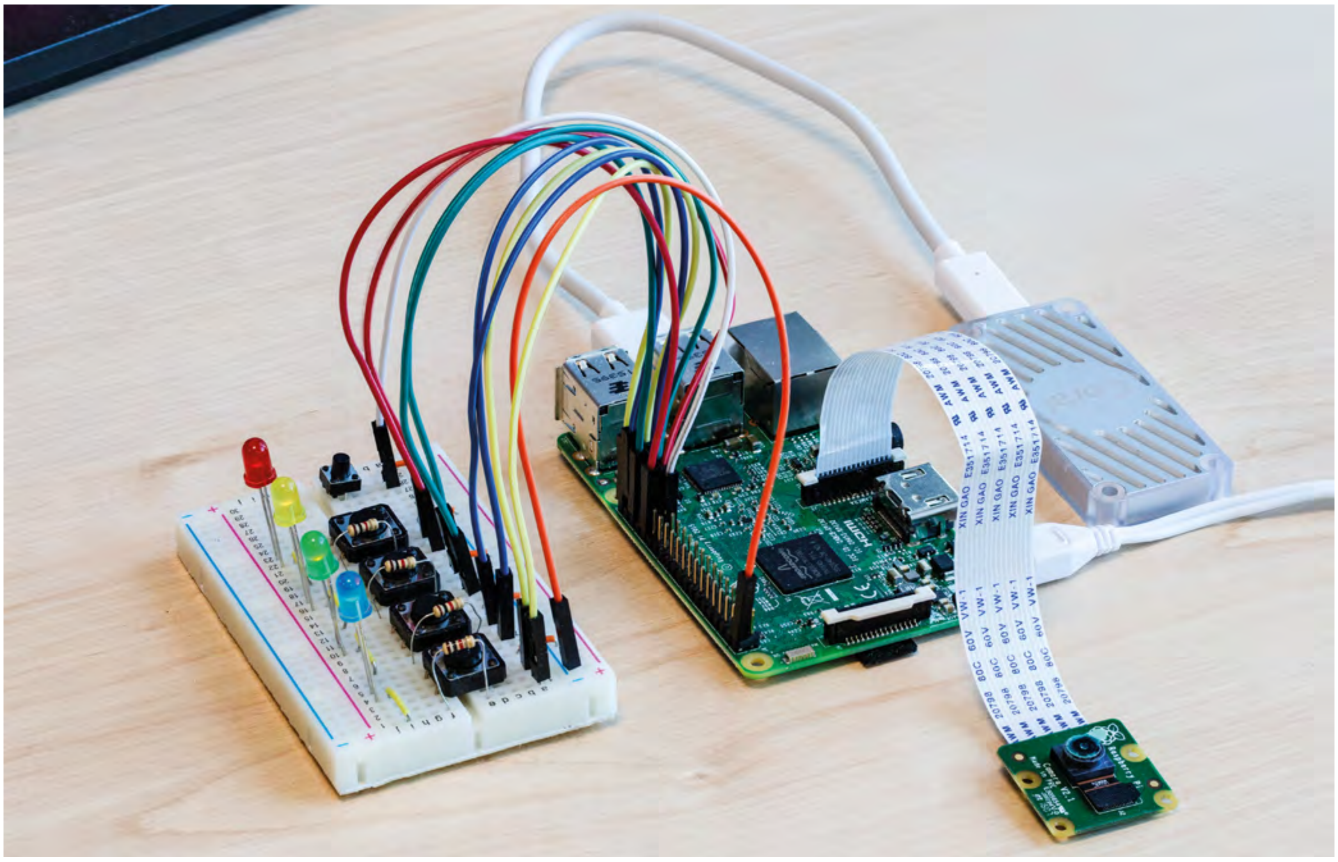
Have fun playing around with different items. Press the fifth button (reset) to clear all the items from memory.

Top Tip

Laser-cut

See the file **rpi3plate.dxf** on our GitHub if you want to laser-cut a stable platform for the components. **magpi.cc/github**

▼ The assembled Teachable Machine project. When a button is pressed, the device observes an object in front of the camera; then it recognises the same object and lights up an LED



MAKER

PJ Evans

PJ is a writer, trainer, and freelance software engineer. His son can't 'borrow' the car as easily any more.

@mrpjevans

Car Spy Pi

Who's that parked on the driveway? Find out automatically using ANPR

Automatic number-plate recognition (ANPR) is becoming more and more commonplace. Once the exclusive realm of the police, the technology used to accurately read car number-plates can now be found in supermarket and airport car parks. It wasn't long ago that this technology was extremely expensive to purchase and implement. Now, even the Raspberry Pi has the ability to read number-plates with high accuracy using the Pi Camera Module and open-source software. Let's demonstrate what's possible by building a system to detect and alert when a car comes onto the driveway.

01 Pick a spot

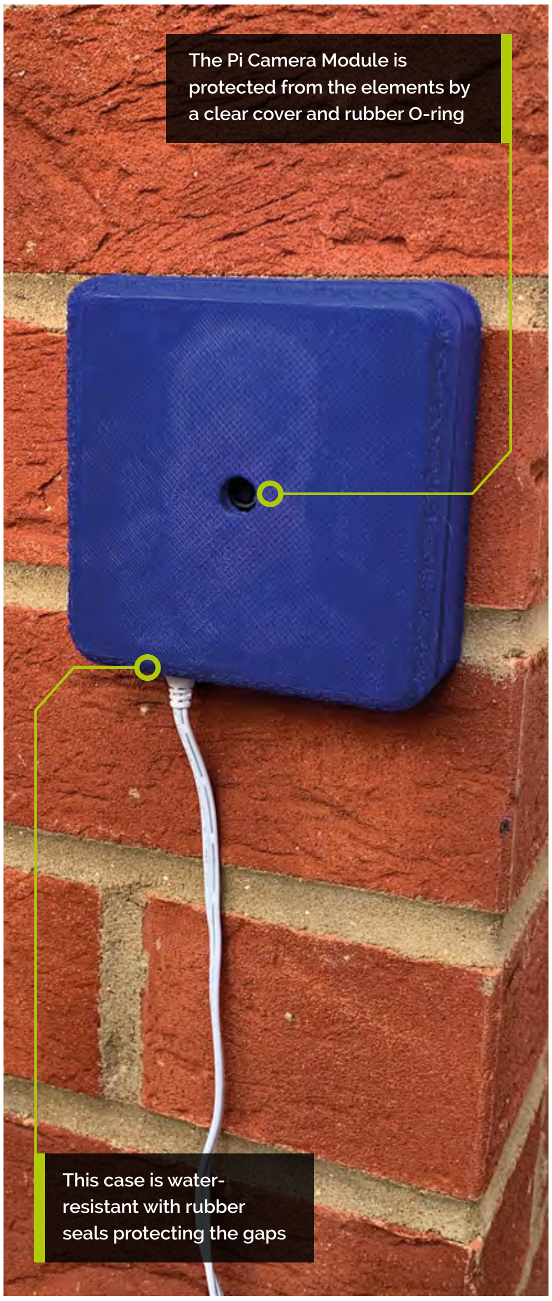
First things first: where are we going to put it? Although this project has lots of applications, we're going to see who's home (or not) by reading number-plates of cars coming and going on a driveway. This means the Raspberry Pi is probably going to live outside; therefore, many environmental constraints come into place. You'll need USB 5V power to the Pi and a mounting position suitable for reading plates, although the software is surprisingly tolerant of angles and heights, so don't worry too much if you can't get it perfectly aligned.

02 Get an enclosure

As our Pi is going to live outside (unless you have a well-placed window), you'll need an appropriate enclosure. For a proper build, get an IP67 waterproof case (e.g. **magpi.cc/epzGcX**). We're opting for homemade, and are using a Raspberry Pi 3 A+ with the RainBerry – a 3D-printable case that, once you add some rubber seals, provides adequate protection. Make sure whatever you choose has a hole for the camera.

You'll Need

- Pi Camera Module **magpi.cc/camera**
- Suitable outdoor enclosure, e.g. **magpi.cc/hOVWBP**
- Pushover account (optional) **pushover.net**



The Pi Camera Module is protected from the elements by a clear cover and rubber O-ring

This case is water-resistant with rubber seals protecting the gaps

03 Prepare your Pi

As we don't need a graphical user interface, Raspbian Stretch Lite is our operating system of choice. Any Pi can handle this task, although if you want the fastest image processing possible, you probably want to avoid the Zero models and get a nice, zippy Model 3B or 3B+. Get your operating system set up, make sure you've done the necessary `sudo apt update && sudo apt -y upgrade` and have configured WiFi if you're not using Ethernet. Finally, make sure your Pi Camera Module is connected and enabled. You can check this by running `sudo raspi-config` and looking under 'Interfacing Options'.

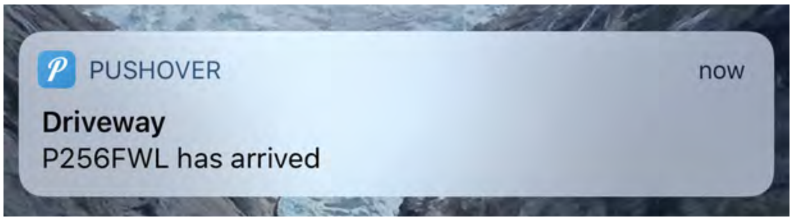
04 Install openALPR

Thankfully, we don't need to be experts in machine learning and image processing to implement ANPR. An open-source project, openALPR provides fast and accurate processing just from a camera image. 'ALPR' is not a mistake: this US project is 'Automatic *License* Plate Recognition'. Thanks to APT, installation is straightforward. At the command line, enter the following:

```
 sudo apt install openalpr openalpr-daemon
openalpr-utils libopenalpr-dev
```

This may take a while, as many supporting packages need to be installed, such as Tesseract, an open-source optical character recognition (OCR) tool. This, coupled with code that identifies a number-plate, is what works the magic.

Protect your Raspberry Pi with a waterproof case


PUSHOVER now
Driveway
P256FWL has arrived

When a car arrives or leaves our driveway, we receive an alert in seconds

05 Time for a test

Once installed, you'll be unceremoniously dropped back to the command prompt. OpenALPR has installed a command-line tool to make testing its capabilities easier and they have also kindly provided a sample image. On the command line, enter the following:

```
cd
wget http://plates.openalpr.com/ea7the.jpg
```

This is a sample USA plate image and a tough one too. Wget grabs the file from the web and places it in your home directory. Let's see if we can recognise it:

```
alpr -c us ea7the.jpg
```

All being well, you'll see a report on screen. The most likely 'match' should be the same as the file name: EA7THE.

06 Install Python libraries

We can use openALPR in Python, too. First, install the libraries using pip. If you don't have pip installed, run:

```
sudo apt install python-pip
```

To install the libraries:

```
 pip install openalpr picamera python-
pushover
```

Now test everything is working by running Python and enter the following code line by line at the >>> prompt:

```
 import json
 from openalpr import Alpr

 alpr = Alpr("us", "/etc/openalpr/openalpr.conf", "/usr/share/
openalpr/runtime_data")
 results = alpr.recognize_file("/home/pi/ea7the.jpg")
 print(json.dumps(results, indent=4))
 alpr.unload()
```

Our target. The software does a great job of recognising number-plates from different heights and angles

If you've not seen JSON-formatted text before, this might seem a bit much, but you should see the correct plate number returned as the first result.

07 Get a Pushover token

So that we can get an alert when a car arrives or leaves, we're using old favourite Pushover (**pushover.net**), which makes sending notifications to mobile phones a breeze. There's a free trial, after which it's a flat fee of $4.99 per device, with no subscription or limits. Once logged in, go to 'Your Applications' and make a note of your User Key. Then click 'Create an Application/API Token'. Call it 'ANPR', leave all the other fields blank, and click 'Create Application'. Now make a note of the API Token; you'll need this and the User Key for your code.

08 Typing time

Now you have everything you need to create your ANPR application. Enter the code listing shown here or download it from **magpi.cc/VEsaCg**. Save it as **anpr.py** in your home directory. Edit the file and enter your User and App tokens where prompted. Save the file, then test by entering:

```
python anpr.py
```

The code makes use of the Pi Camera Module and openALPR in tandem. Every five seconds, the camera takes a picture which is passed to openALPR for analysis. If a licence plate is found, we get the number. If there has been a change, an alert is sent to Pushover, which is then forwarded to any registered mobile devices.

09 Make your list

If you want to have more friendly names rather than just the plate number, try adding a Python dictionary just after the import statements, like this:

```
lookup = {
            "ABC123": "Steve McQueen",
            "ZXY123": "Lewis Hamilton"
            }
```

Then change all instances of `number_plate` in the alert text as follows:

```
lookup[number_plate]
```

Now you'll get a friendly name instead. See if you can handle what happens if the plate isn't recognised.

10 Run on boot

A key part of any 'hands-free' Raspberry Pi installation is ensuring that in the event of a power failure, the required services start up again. There are many ways of doing this; we're going use one of the simpler methods.

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

Find the final line, `exit 0` and enter the following on the line above:

```
#Start ANPR Monitoring
/usr/bin/python /home/pi/anpr.py
```

Press **CTRL+X** then **Y** to save the file. Finally, run the earlier pip command again, using sudo this time to install the libraries for the root user:

```
sudo pip install openalpr picamera
python-pushover
```

On reboot, the code will start up and run in the background.

anpr.py

▶ Language: **Python 3**

DOWNLOAD THE FULL CODE:

magpi.cc/VEsaCg

```python
from openalpr import Alpr
from picamera import PiCamera
from time import sleep
import pushover

# Pushover settings
PUSHOVER_USER_KEY = "<REPLACE WITH USER KEY>"
PUSHOVER_APP_TOKEN = "<REPLACE WITH APP TOKEN>"

# 'gb' means we want to recognise UK plates, many others are available
alpr = Alpr("gb", "/etc/openalpr/openalpr.conf",
            "/usr/share/openalpr/runtime_data")
camera = PiCamera()
pushover.init(PUSHOVER_APP_TOKEN)
last_seen = None

try:
    # Let's loop forever:
    while True:

        # Take a photo
        print('Taking a photo')
        camera.capture('/home/pi/latest.jpg')

        # Ask OpenALPR what it thinks
        analysis = alpr.recognize_file("/home/pi/latest.jpg")

        # If no results, no car!
        if len(analysis['results']) == 0:
            print('No number plate detected')

            # Has a car left?
            if last_seen is not None:
                pushover.Client(PUSHOVER_USER_KEY).send_message(
                    last_seen + " left",
                    title="Driveway")

            last_seen = None

        else:
            number_plate = analysis['results'][0]['plate']
            print('Number plate detected: ' + number_plate)

            # Has there been a change?
            if last_seen is None:
                pushover.Client(PUSHOVER_USER_KEY).send_message(
                    number_plate + " has arrived", title="Driveway")
            elif number_plate != last_seen:
                pushover.Client(PUSHOVER_USER_KEY).send_message(
                    number_plate + " arrived  and " + last_seen + " left",
                    title="Driveway")

            last_seen = number_plate

        # Wait for five seconds
        sleep(5)

except KeyboardInterrupt:
    print('Shutting down')
    alpr.unload()
```

11 Add logging and curfews

One use of this installation is to track the times cars come and go. This can be especially useful for young drivers who have curfew restrictions on their insurance. See if you can augment the code to check whether a registration plate has not been seen after a certain time. For example, if your younger family members have such a restriction, send them an alert to their phone if their car isn't in the driveway 30 minutes beforehand. You might save them an insurance premium increase! Also, why not log all the comings and goings to a file? A bit of data analysis might help reduce car usage or fuel costs.

12 Make it your own

As ever, it's over to you. Now you have the ability to track and record registration plates, there are many different applications for you to explore. Since all the analysis of the image is done locally, no internet connection is required for the system to work. Is there someone 'borrowing' your parking space at work? Catch 'em in the act! Why not take your Car Spy Pi on the road? It could record every vehicle you encounter, which may be useful should something untoward happen. Combine a Raspberry Pi Zero with a ZeroView (**magpi.cc/eBnYrZ**) and you're all set.

Part 02

Code an isometric adventure game: AmazeBalls

Pygame Zero in 3D. Let's make the map bigger in part two of this three-part tutorial

MAKER

Mark Vanstone

Educational software author from the nineties, author of the ArcVenture series, disappeared into the corporate software wasteland. Rescued by the Raspberry Pi!

magpi.cc/YiZnxL

@mindexplorers

In this second part of our tutorial on using Pygame Zero to create a 3D isometric game, we will start from where we left off in the last part and look at ways to make our 3D area larger and easier to edit. This will mean using a map editor called Tiled, which is free to download and use, to make your own 3D maps. We'll learn how to create a simple map and export it from Tiled in a data format called JSON. We will then import it into our game and code our draw function to scroll around the map area when the player moves. Lots to do, so let's get started.

01 Tools for the job

In the previous part of this tutorial, we made our map data by writing a two-dimensional list of zeroes and ones which represented either a floor block or a wall block. The player was able to move onto any block that was a zero in the data. This time we are going to get a map-editing app to do the work for us instead of typing the data in. First, we need to get Tiled installed. You can find the Tiled homepage at **mapeditor.org**, where options are given to support the developer if you like what they are creating.

02 Getting Tiled

Tiled can be used on many different systems, including PCs and Mac computers. Also, more importantly for us, it works really well on Raspbian and Raspberry Pi. It's also super-easy to install. All you need to do is open up a Terminal window, then make sure you're online and are up-to-date by typing `sudo apt-get update`. You may need to type `sudo apt-get upgrade` too, depending on how long you've left your Pi without updates. When those have run, just type `sudo apt-get install tiled`, hit **RETURN**, and you should see Tiled being installed. When it's done, a new Tiled icon will appear in your Graphics submenu.

03 Cartography

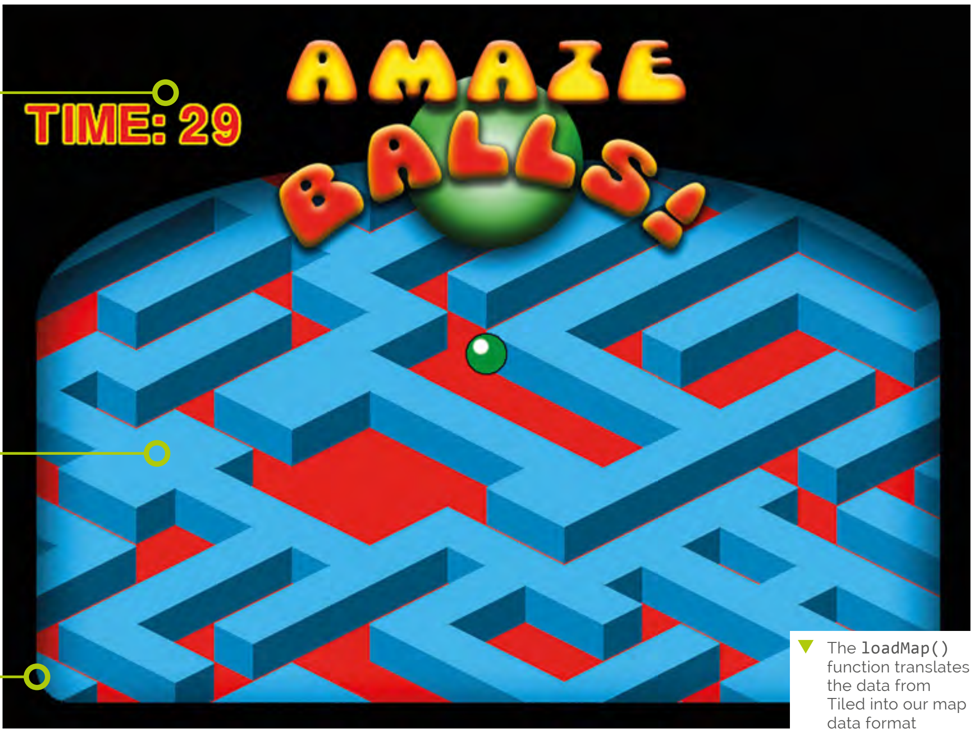
Previously our map was 12 blocks by 12, which all fitted on the screen at once. With our map editor we can make a much bigger map. It could be huge, but for the moment let's stick to a grid of 30 by 30 blocks. You may want to download our ready-made map and blocks from **magpi.cc/fPBrhM**. If you load in the map, you should see a blue and red maze, a bit like in part one but much bigger. This time, though, we have added a new block to indicate the finish point in the maze. You should be able to scroll around to see the whole map.

04 Exporting the data

Have a play around with the map editor; there is some great documentation on the website. When you have familiarised yourself with how it works, we can think about the task of getting the map data into our game. To export the data, go to Export in the File menu and when the dialogue box opens up, asking 'export as…', find a suitable place (perhaps a subdirectory called **maps**) to save the map (perhaps as **map1**), but in the drop-down labelled 'Save as type', select 'Json map files (*.json)'. This type of file (pronounced 'Jay-son', short for JavaScript Object Notation) can be viewed in a text editor.

You'll Need

- Raspbian Jessie or newer
- Tiled (free map editor) **mapeditor.org**
- An image manipulation program such as GIMP, or images available from **magpi.cc/fPBrhM**
- The latest version of Pygame Zero (1.2)

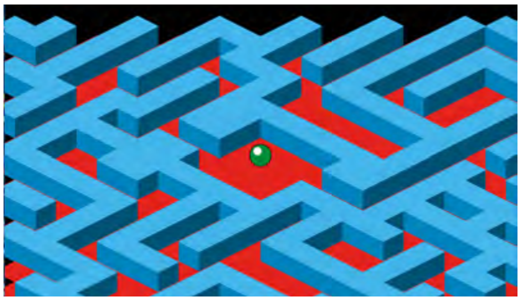


The player must complete the maze in the fastest time possible

An external editor is used to create the maze

The maze is bigger than the game window and scrolls around with the player

The `loadMap()` function translates the data from Tiled into our map data format



When we have rewritten our `drawMap()` function, we will see some jagged edges around the extremities of the drawing area

05 JSON and the Argonauts

If we open the JSON file, we will see a load of curly and square brackets with words and numbers spread all over the place, but before you proclaim 'It's all Greek to me!', let's have a look at some of the elements so that we can understand the structure of the data. If you are familiar with the JavaScript language, you'll recognise that the curly brackets { and } are used to contain blocks of code or data, and square brackets [and] are used for lists of data. Look at the element called 'layers' and you will see data describing our map.

06 Loading the data

For this game we don't actually need all the data that is in the JSON file, but we can load it all in and just use the bits we need. Let's make

figure1.py

```python
import json
import os

def loadmap(mp):
    with open(mp) as json_data:
        d = json.load(json_data)
    mapdata = {"width":d["width"], "height":d["height"]}
    rawdata = d["layers"][0]["data"]
    mapdata["data"] = []
    for x in range(0, mapdata["width"]):
        st = x*mapdata["width"]
        mapdata["data"].append(
rawdata[st:st+mapdata["height"]])

    tileset = "maps/" + d["tilesets"][0]["source"].replace(
".tsx",".json")
    with open(tileset) as json_data:
        t = json.load(json_data)

    mapdata["tiles"] = t["tiles"]
    for tile in range(0,len(mapdata["tiles"])):
        path = mapdata["tiles"][tile]["image"]
        mapdata["tiles"][tile]["image"] =
os.path.basename(path)
        mapdata["tiles"][tile]["id"] =
mapdata["tiles"][tile]["id"]+1
    return mapdata
```

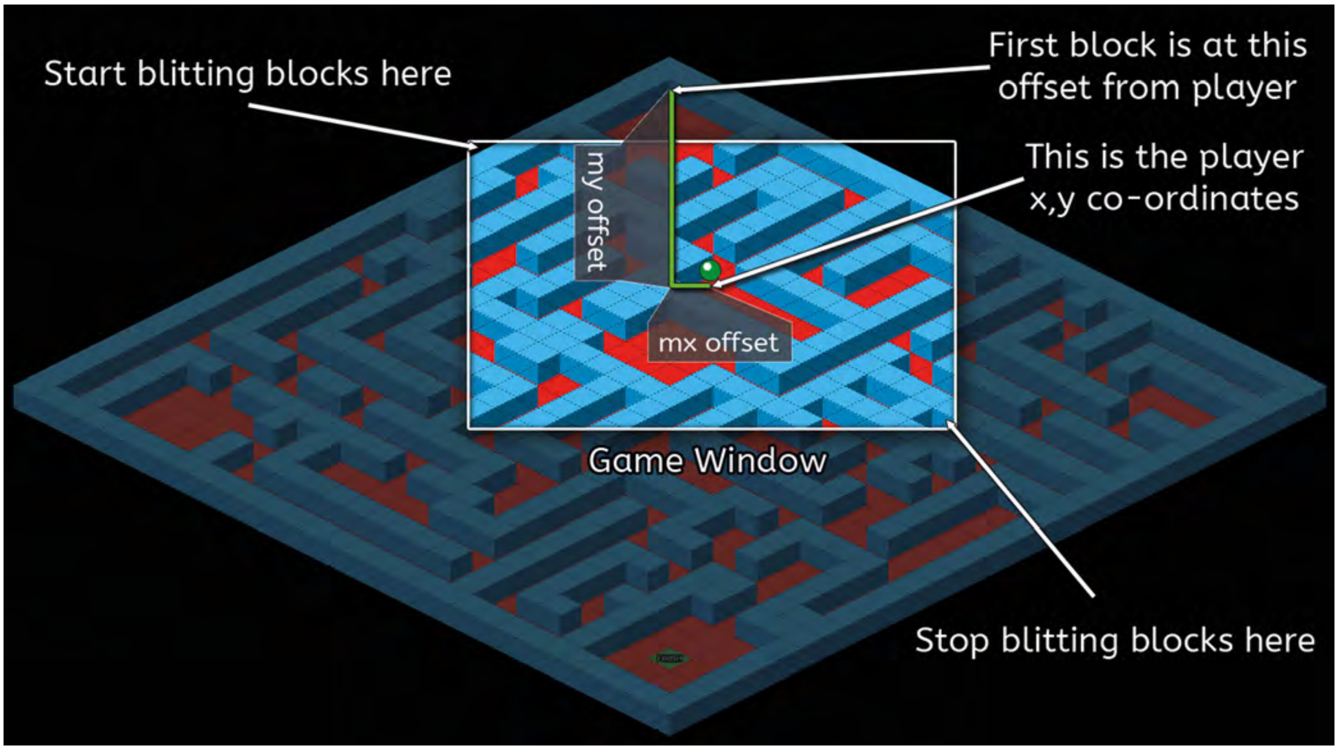

Start blitting blocks here
First block is at this offset from player
This is the player x,y co-ordinates
my offset
mx offset
Game Window
Stop blitting blocks here

Figure 2 The coordinates that the map starts from are calculated as an offset from the player, and the maze blocks are only drawn inside the rectangle of the game window

a new module to deal with map handling, like we have done in previous tutorials (see *The MagPi* #76, **magpi.cc/76**). Let's make a new module called **map3d.py**. Python provides a module to import JSON files, so we can write `import json` at the top of our **map3d.py** file to load the module. We'll also need to use the os module for handling file paths, so import that too. Then we just need to write a function to load our map.

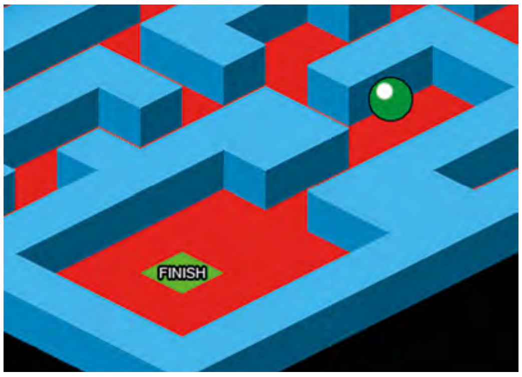
Top Tip

Looking at JSON

You can look at JSON files in any text editor, but a programming one is probably best – perhaps try Geany.

07 Getting what we need

Let's make a function called `loadmap()` and use a parameter called `mp` to pass a file location to the function. Have a look at **figure1.py** to see how we write this. You can see that we load in our map data into a variable `d` using the function `json.load()`. Then we can copy the width and height into a dictionary called `mapdata`. This dictionary will hold all the data we need by the time we get to the end of the function. Having made a temporary copy of the block data (`rawdata`), we can then loop through the values and put them in the format we want in `mapdata`.



We have added a new tile for the finish. When the player moves onto this block, the maze is solved

08 The tileset connection

One bit of information that we need is where to find the details about which images to use on each block. This is included as a value called 'tilesets'. In this case we will assume that we only have one tileset defined, so we can read it in and go and find our block images. Except there is a slight crinkle in the plan. Our map data refers to the tileset file as a .tsx file. What we need to do is go back to Tiled and export our tileset from the tileset editor as a JSON file. Then, when we import it, we just switch the .tsx extension for .json.

09 A night on the tiles

Once we have loaded in our tileset data as a JSON file, we can then loop through the tiles and get the names of each of our block images. You will note that we add one to the id value to match the values that Tiled has exported. When we have all that data in our `mapdata` dictionary, we can pass that back to our main program by writing `return mapdata`. Going back to our main program, we'll need to add an import for our map3d module at the top of the code and then, before our `draw()` function, we can write `mapData = map3d.loadmap("maps/map1.json")` instead of our list of zeroes and ones.

10 Thinking big

In part one of this tutorial, our maze was 12×12 blocks, which just fitted nicely into the game area. Now we have a 30×30 maze which, if we draw it all, will go off the sides of the game window. We need to be able to scroll the map around the screen as the player moves through the maze. The way we can do this is to keep the bouncing ball in the centre of the game area and, as the player moves, scroll the map. So, in effect, what we are saying is that we are going to draw the map relative to the player rather than relative to the game window.

11 It's all relative

To draw our map, we are going to use the same basic loops (x and y) as before, but we will start drawing our map based on the coordinates we calculated for the player screen x and y coordinates. With that starting screen position, we loop in a range that is either side of the player in both directions. See **Figure 2** for a visual explanation of what we are doing in these loops. In simple terms, what we are saying is: 'Start drawing the map from coordinates that will make the player appear in the centre of the window. Then only draw the blocks that are visible in the window.' See **figure3.py** for how we have changed the `drawMap()` function to do this.

12 Extra functions

You will see in **figure3.py** that we have a couple of new functions that we have not defined yet. The first is `onMap()`, which we pass an x and a y coordinate to. These are block locations which we test to make sure that the coordinates we are asking for are actually on our map, otherwise we will get an error. If the x or y are less than 0 or greater than the width (or height) of the map, then we can ignore it. The other function is `findData()`. This function finds the data associated with a tile of a given id. Look at **figure4.py** (overleaf) to see how these functions are written.

figure3.py

```python
def drawMap():
    psx = OFFSETX
    psy = OFFSETY-32
    mx = psx - player["sx"]
    my = psy - player["sy"]+32

    for x in range(player["x"]-12, player["x"]+16):
        for y in range(player["y"]-12, player["y"]+16):
            if onMap(x,y):
                b = mapData["data"][y][x]
                td = findData(mapData["tiles"], "id", b)
                block = td["image"]
                bheight =  td["imageheight"]-34
                bx = (x*32)-(y*32) + mx
                by = (y*16)+(x*16) + my
                if -32 <= bx < 800 and 100 <= by < 620:
                    screen.blit(block, (bx, by - bheight))
                if x == player["x"] and y == player["y"]:
                    screen.blit("ball"+str(player["frame"]),
(psx, psy))
```

▲ The updated drawMap() function

map3d.py

▶ Language: **Python**

```python
# 3dmap module for AmazeBalls
import json
import os

def loadmap(mp):
    with open(mp) as json_data:
        d = json.load(json_data)
    mapdata = {"width":d["width"], "height":d["height"]}
    rawdata = d["layers"][0]["data"]
    mapdata["data"] = []
    for x in range(0, mapdata["width"]):
        st = x*mapdata["width"]
        mapdata["data"].append(rawdata[st:st+mapdata["height"]])

    tileset = "maps/" + d["tilesets"][0]["source"].replace(
".tsx",".json")
    with open(tileset) as json_data:
        t = json.load(json_data)

    mapdata["tiles"] = t["tiles"]
    for tile in range(0,len(mapdata["tiles"])):
        path = mapdata["tiles"][tile]["image"]
        mapdata["tiles"][tile]["image"] = os.path.basename(path)
        mapdata["tiles"][tile]["id"] = mapdata["tiles"][tile]["id"]+1
    return mapdata
```

figure4.py

```
def onMap(x,y):
    if 0 <= x < mapData["width"] and 0 <= y < mapData["height"]:
        return True
    return False

def findData(lst, key, value):
    for i, dic in enumerate(lst):
        if dic[key] == value:
            return dic
    return -1
```

The functions to test if a coordinate is inside the map area – `onMap()` – and `findData()`, which finds tile data for map drawing

figure5.py

```
def doMove(p, x, y):
    if onMap(p["x"]+x, p["y"]+y):
        mt =
mapData["data"][p["y"]+y][p["x"]+x]
        if mt == 1:
            p.update({"queueX":x,
"queueY":y, "moveDone":False})
            if mt == 3:
                mazeSolved = True
```

The updated `doMove()` function

13 Masking the edges

If we draw our map now, we have lost that nice diamond shape map. And if we move the player down the map, we get a jagged edge at the top and blocks popping in and out of view as our `drawMap()` function decides which ones to draw in the range. We can tidy this up by overlaying a frame that obscures the edges of the printed area. We do this by having an image which covers the whole window but has a transparent cut-out area where we want the map blocks to be shown. We blit this frame graphic after we have called `drawMap()` in our `draw()` function.

14 I can't move!

Now that we have our map drawing, if you are adding in code following on from part one, you may notice that we can't move the bouncing ball any more. That's because the data we have loaded is a slightly different format and has floor blocks as id 1 and walls as id 2, so at the moment our `doMove()` function is thinking we are surrounded by walls (which were id 1 in the last part). We need to change our `doMove()` function to accommodate the new data format. Have a look at **figure5.py** to see what we need to write.

Top Tip

Drawing order

Remember that in the `draw()` function, things are drawn in the order you call them, so always draw the things you want on top last.

15 Finding the exit

Now that we have tidied up our display and got our ball moving again, we'll need to change a few of the default values that we start with. In the last part, we had the player starting at x = 0 and y = 3. We will need to change those values to 3 and 3 in the player data at the top of the code to be suitable for this map. We'll also want to change the `OFFSETY` constant to 300 to move the map a little further down the screen. We should now be able to guide the bouncing ball around the maze towards the bottom of the screen, where we should find the finish tile.

16 Over the finish line

If we try to move onto the finish block, we won't be able to, as our `doMove()` function is only detecting blocks we can move to as id 1. So we need to add another condition. Instead of only testing `mt` for 1, let's make that line if `mt == 1` or `mt == 3`: (because the id of the finish tile is 3). We can then add a variable to set if the player moves onto the finish, by adding inside this condition: `if mt == 3: mazeSolved = True`. We'll also need to declare `mazeSolved` as global inside `doMove()` and set its initial value to `False` at the top of our program.

17 Time is running out

Let's add a timer to the game. When the player reaches the finish (`mazeSolved is True`), we can stop the timer and display a message. So, first we make a timer variable at the top of our program with `timer = 0` and then, right at the end of the code, just before `pgzrun.go()`, we can use the Pygame Zero clock function called `schedule_interval()`. If we write `clock.schedule_interval(timerTick, 1.0)`, then the function `timerTick()` will be called once every second.

18 The clock struck one

So, all we need to do now is define the `timerTick()` function. We'll need to check if `mazeSolved` is `False` and add 1 to our `timer` variable if it is. Then we can add a `screen.draw.text` line to our `draw()` function to display the timer value and if `mazeSolved` is `True`, we can add some more text to say the maze has been solved and how many seconds it took. See the full program listing for how to write the code for those bits.

In the next instalment, we are going to add some baddies to contend with and, just for good measure, we can throw in some dynamite!

amazeballs2.py

DOWNLOAD THE FULL CODE:

magpi.cc/fPBrhM

> Language: **Python**

```python
import pgzrun
import map3d

player = {"x":3, "y":3, "frame":0, "sx":0, "sy":96,
          "moveX":0, "moveY":0, "queueX":0, "queueY":0,
          "moveDone":True, "movingNow":False, "animCounter":0}
OFFSETX = 368
OFFSETY = 300
timer = 0
mazeSolved = False

mapData = map3d.loadmap("maps/map1.json")

def draw(): # Pygame Zero draw function
    screen.fill((0, 0, 0))
    drawMap()
    screen.blit(‘title’, (0, 0))
    screen.draw.text("TIME: "+str(timer) , topleft=(20, 80), owidth=0.5, ocolor=(255,255,0), color=(255,0,0) , fontsize=60)
    if mazeSolved:
        screen.draw.text("MAZE SOLVED in " + str(timer) + " seconds!" , center=(400, 450), owidth=0.5, ocolor=(0,0,0), color=(0,255,0) , fontsize=60)


def update(): # Pygame Zero update function
    global player, timer
    if player["moveDone"] == True:
        if keyboard.left: doMove(player, -1, 0)
        if keyboard.right: doMove(player, 1, 0)
        if keyboard.up: doMove(player, 0, -1)
        if keyboard.down: doMove(player, 0, 1)
    updateBall(player)

def timerTick():
    global timer
    if not mazeSolved:
        timer += 1

def drawMap():
    psx = OFFSETX
    psy = OFFSETY-32
    mx = psx - player["sx"]
    my = psy - player["sy"]+32

    for x in range(player["x"]-12, player["x"]+16):
        for y in range(player["y"]-12, player["y"]+16):
            if onMap(x,y):
                b = mapData["data"][y][x]
                td = findData(mapData["tiles"], "id", b)
                block = td["image"]
                bheight =  td["imageheight"]-34
                bx = (x*32)-(y*32) + mx
                by = (y*16)+(x*16) + my
                if -32 <= bx < 800 and 100 <= by < 620:
                    screen.blit(block, (bx, by - bheight))
                if x == player["x"] and y == player["y"]:
                    screen.blit("ball"+str(player["frame"]), (psx, psy))

def findData(lst, key, value):
    for i, dic in enumerate(lst):
        if dic[key] == value:
            return dic
    return -1

def onMap(x,y):
    if 0 <= x < mapData["width"] and 0 <= y < mapData["height"]:
        return True
    return False

def doMove(p, x, y):
    global mazeSolved
    if onMap(p["x"]+x, p["y"]+y):
        mt = mapData["data"][p["y"]+y][p["x"]+x]
        if mt == 1 or mt == 3:
            p.update({"queueX":x, "queueY":y, "moveDone":False})
            if mt == 3:
                mazeSolved = True

def updateBall(p):
    if p["movingNow"]:
        if p["moveX"] == -1: moveP(p,-1,-0.5)
        if p["moveX"] == 1: moveP(p,1,0.5)
        if p["moveY"] == -1: moveP(p,1,-0.5)
        if p["moveY"] == 1: moveP(p,-1,0.5)
    p["animCounter"] += 1
    if p["animCounter"] == 4:
        p["animCounter"] = 0
        p["frame"] += 1
        if p["frame"] > 7:
            p["frame"] = 0
        if p["frame"] == 4:
            if p["moveDone"] == False:
                if p["queueX"] != 0 or p["queueY"] !=0:
                    p.update({"moveX":p["queueX"], "moveY":p["queueY"], "queueX":0, "queueY":0, "movingNow": True})
            else:
                p.update({"moveDone":True, "moveX":0, "moveY":0, "movingNow":False})

        if p["frame"] == 7 and p["moveDone"] == False and p["movingNow"] == True:
            p["x"] += p["moveX"]
            p["y"] += p["moveY"]
            p["moveDone"] = True

def moveP(p,x,y):
    p["sx"] += x
    p["sy"] += y

clock.schedule_interval(timerTick, 1.0)
pgzrun.go()
```

Part 03

Pi Bakery: In a spin

Enhance your Pi by adding a VGA output and a retro 8-bit sound chip emulator using the eight-core Propeller chip. This month, generating video

MAKER

Mike Cook

Veteran magazine author from the old days, writer of the Body Build series, plus co-author of *Raspberry Pi for Dummies*, *Raspberry Pi Projects*, and *Raspberry Pi Projects for Dummies*.

magpi.cc/TPaUfT

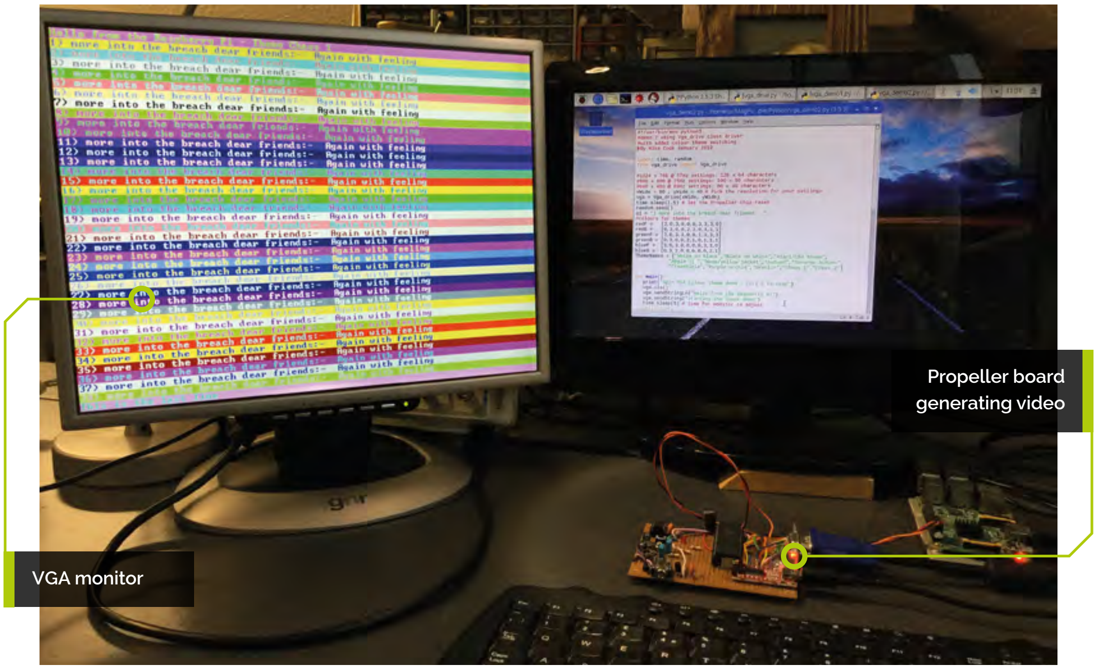
This month, we will see how this board can be made to generate VGA video. Most of the examples you'll see for this chip are in a standalone context, but we want to use video in conjunction with our Raspberry Pi. Therefore, we have to take a different approach to normal. Instead of interfacing an old PS/2 keyboard and mouse, we want to use a serial connection between Pi and Propeller, like we looked at last month. The best way is to do this is to modify what is already in the field; however, this leads to some interesting problems that need to be solved.

01 The VGA signal

To drive a VGA monitor, we need to send it five signals: two synchronisation pulses and three analogue video signals. **Figure 1** shows the connection between the Propeller chip and the VGA socket, and shows the eight digital signals used. The HS, or horizontal sync pulse, tells the monitor when to start scanning from left to right, and the VS, or vertical sync pulse, starts the top-to-bottom scan. The video is generated by two digital lines and resistors in a simple D/A (digital to analogue) converter, giving four different intensities of each colour component: red, green, and blue.

02 The video signal

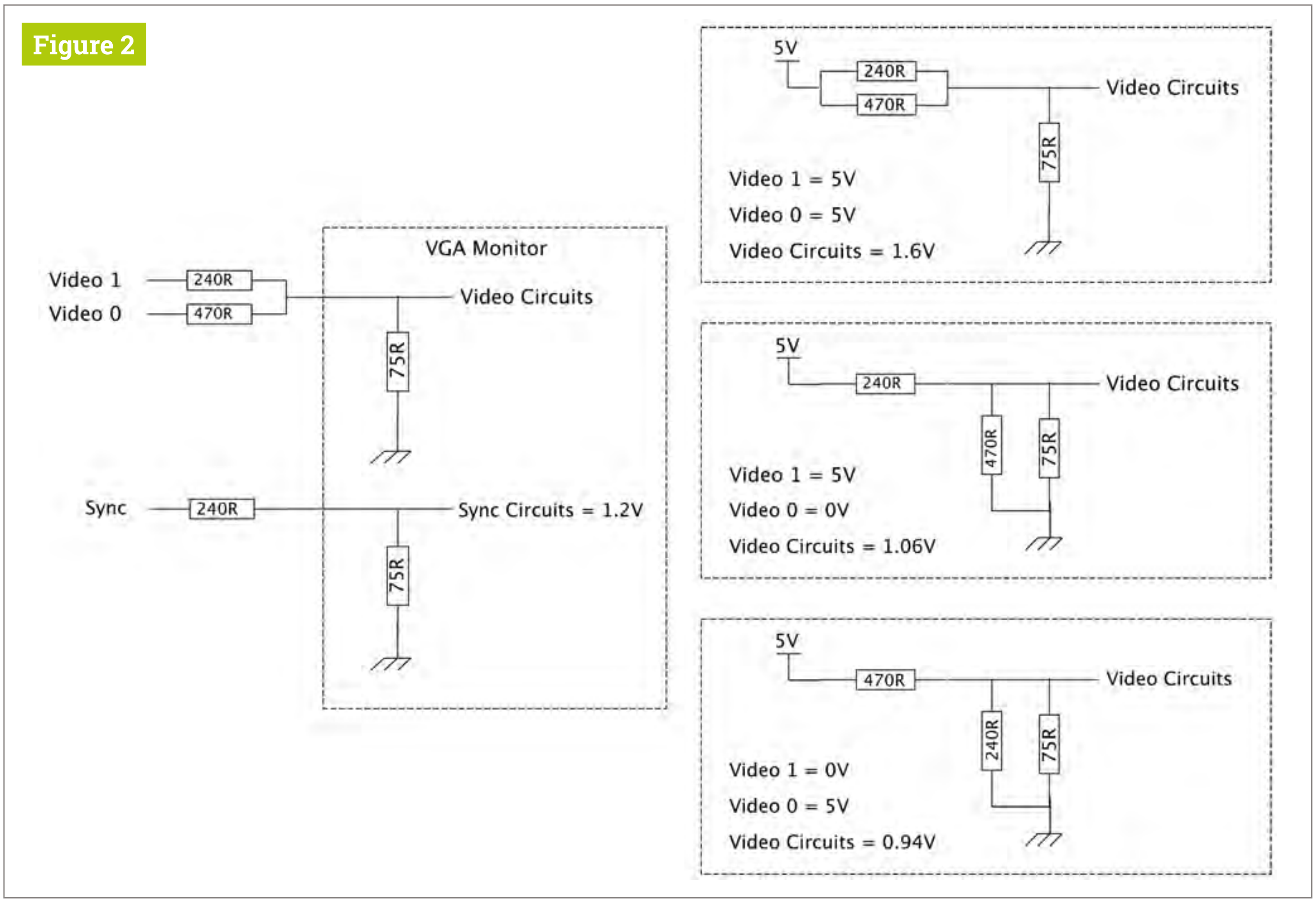
Inside the VGA monitor is a terminating resistor of 75 Ω. This, together with the series resistors, forms a voltage divider to cut down the 5 V signal from the Propeller chip. **Figure 2** shows



You'll Need

- Propeller development board from The MagPi #77 **magpi.cc/77**
- VGA monitor

Figure 2



how the voltages reaching the VGA monitor's circuits are determined by which resistors are connected to a logic one, or logic zero. These are calculated from the resistors in parallel formula, and the potential divider formula, but they are ultimately derived from Ohm's law. Note that these voltage values are not equally spaced but are 'gamma corrected' so the eye perceives them as equal brightness steps.

03 Getting started

The official Propeller Package is called AN004-GUI-StartVGA-Code-v1.0 by André LaMothe. It's now quite hard to find this code, so we have put it on our GitHub page. Out of the box, you can compile and load into RAM the **WMF_HelloWorld_010.spin** code. It's not too exciting – it just scrolls a simple message continuously – but it'll prove your VGA monitor is working. If it doesn't work, then you could try altering the VGA resolution to another value.

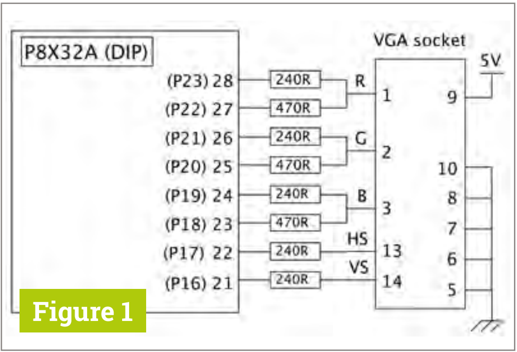


Figure 1

04 Altering the video resolution

At the start of **VGA_HiRes_Text_010.spin**, there are three tables that set up the resolution of the display. By default, 800×600 @ 75Hz video is enabled, with 100×50 characters displayed. Use the curly braces to surround any section of code to disable it, and remove the braces surrounding the resolution you want to use. These definitions are shown in the **Mods_in_VGA_HiRes.spin**

▲ **Figure 2** Converting digital signals to VGA voltages

◀ **Figure 1** Schematic for the VGA output of the propeller board

Mods_in_VGA_HiRes.spin

› Language: **Spin**

```
CON

{' 1024 x 768 @ 57Hz settings: 128 x 64 characters

  hp = 1024     'horizontal pixels
  vp = 768      'vertical pixels
  hf = 16       'horizontal front porch pixels
  hs = 96       'horizontal sync pixels
  hb = 176      'horizontal back porch pixels
  vf = 1        'vertical front porch lines
  vs = 3        'vertical sync lines
  vb = 28       'vertical back porch lines
  hn = 1        'horizontal normal sync state (0|1)
  vn = 1        'vertical normal sync state (0|1)
  pr = 60       'pixel rate in MHz at 80MHz system
clock (5MHz granularity)
}

' 800 x 600 @ 75Hz settings: 100 x 50 characters

  hp = 800      'horizontal pixels
  vp = 600      'vertical pixels
  hf = 40       'horizontal front porch pixels
  hs = 128      'horizontal sync pixels
  hb = 88       'horizontal back porch pixels
  vf = 1        'vertical front porch lines
  vs = 4        'vertical sync lines
  vb = 23       'vertical back porch lines
  hn = 0        'horizontal normal sync state (0|1)
  vn = 0        'vertical normal sync state (0|1)
  pr = 50       'pixel rate in MHz at 80MHz system
clock (5MHz granularity)

' 640 x 480 @ 69Hz settings: 80 x 40 characters
{
  hp = 640      'horizontal pixels
  vp = 480      'vertical pixels
  hf = 24       'horizontal front porch pixels
  hs = 40       'horizontal sync pixels
  hb = 128      'horizontal back porch pixels
  vf = 9        'vertical front porch lines
  vs = 3        'vertical sync lines
  vb = 28       'vertical back porch lines
  hn = 1        'horizontal normal sync state (0|1)
  vn = 1        'vertical normal sync state (0|1)
  pr = 30       'pixel rate in MHz at 80MHz system
clock (5MHz granularity)
}
```

Top Tip

Startup time

Each time you open the serial port, the Propeller processor resets and the VGA code is loaded in. This takes a little time, so we must add a small delay to allow the VGA signals to start up.

listing. Your monitor might respond to a faster or slower clock rate better, or you might want to pack more information onto the display. Remember, after changing the file, go back to the HelloWorld tab to load it into the Propeller board.

05 Sorting out the files

Make a folder for your VGA generator and copy the **VGA_HiRes_Text_010.spin** and the **WMF_Terminal_Services_010.spin** files into it from the download. You need to add the **FullDuplexSerialPlus.spin** file to the folder; it can be found on our GitHub repository or from **magpi.cc/MAYjtg**. Finally, create a folder called **Python** for our Raspberry Pi code. In the Propeller IDE, open up **WMF_Terminal_Services_010.spin** and change the `OutTerm` function to that shown in the **Change_to_Terminal_Services.spin** listing overleaf). Then create a new file and enter the code from the **HelloPi.spin** listing. Download this into the RAM and you should see a message on the VDU.

06 Why those changes?

We are going to communicate with the display over a serial port, using strings, and this necessitates having a number indicating that the string is finished. This is normally the carriage return number 13. Also, the Propeller uses null-terminated strings, so we can't send a zero as part of a string either. Therefore we have to devise a strategy that avoids using those two numbers but is still be able to send those values to the display. What we have chosen to do is to move the control codes to different values and add an offset to the positioning values.

07 Colour information

In a similar vein, colour values are six bytes long, and could be mistaken for control signals. So we set bit 6 in every colour value, and remove it when we receive it – see **Figure 3** (page 62). To make sure we don't mistake normal data for colour data, we precede colour data with the number 12. Finally, if we send data too

HelloPi.spin

Language: Spin

```
'' ==========================================================================
'' File: HelloPi.Spin
'' Mike Cook
'' ==========================================================================
CON
' CONSTANTS, DEFINES, MACROS, ETC.
  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000
  RxPin = 31      'default boot loader RX
  TxPin = 30      'default boot loader TX
  BaudRate = 115200 'this is a good baud rate for 80MHz

  ' import some constants from the Propeller Window Manager
  VGACOLS = WMF#VGACOLS
  VGAROWS = WMF#VGAROWS

  ' set these constants based on the Propeller device you are using
  VGA_BASE_PIN      = 16         'VGA pins 16-23

OBJ
  ' Propeller Windows GUI object(s)
  SerIO : "FullDuplexSerialPlus" 'object that implements serial I/O for the Propeller.
  WMF   : "WMF_Terminal_Services_010" ' include the terminal services driver which includes the VGA driver itself

VAR
' DECLARED VARIABLES, ARRAYS

  byte  gVgaRows, gVgaCols ' convenient globals to store number of columns and rows
  byte  gStrBuff1[64]      ' a string buffers
  byte  gTextCursX, gTextCursY, gTextCursMode ' text cursor 0 [x0,y0,mode0]
  long  gVideoBufferPtr ' holds the address of the video buffer passed back from the VGA driver

CON
' MAIN ENTRY POINT
PUB Start | S1, N1, CF, CB

  ' create the GUI itself
  CreateAppGUI
  SerIO.Start(RxPin, TxPin, %0000, BaudRate) ' initialise SerIO object

  ' MAIN EVENT LOOP
  WMF.StringTerm(string("Hello World! from the Propeller "))
  repeat
      S1 := SerIO.getstr(gStrBuff1)
      N1 := byte[gStrBuff1][0]
      if N1 == $0C
        CF := byte[gStrBuff1][1] & $3F
        CB := byte[gStrBuff1][2] & $3F
        WMF.SetLineColor(WMF.GetRowTerm , CF, CB )
      else
        WMF.StringTerm(S1)
      SerIO.tx (2) ' send an ACK to say that has beed processed

' end PUB -------------------------------------------------------------------

PUB CreateAppGUI | retVal
' This functions creates the entire user interface for the application
  ' start the VGA driver and terminal services
  retVal := WMF.Init(VGA_BASE_PIN, @gTextCursX )

  ' VGA buffer encoded in upper 16-bits of return value
  gVideoBufferPtr := retVal >> 16

  'setup screen colors - see Terminal Services themes
  WMF.ClearScreen( WMF#CTHEME_ATARI_C64_FG, WMF#CTHEME_ATARI_C64_BG )
  'WMF.ClearScreen( WMF#CTHEME_AUTUMN_FG, WMF#CTHEME_AUTUMN_BG )
  return
```

vga_drive.py

▶ Language: **Python**

```python
#!/usr/bin/env python3
# Class for driving Propeller VDU
# By Mike Cook Jan 2019

import serial

class Vga_drive():

    def __init__(self, chrX,chrY): # initialisation
        self.spinProcessor = serial.Serial("/dev/ttyUSB0",115200, timeout = 1)
        self.spinProcessor.flushInput()
        self.spinProcessor.flushOutput()
        self.xWidth = chrX -1 # make first line zero
        self.yWidth = chrY -1

    def sendString(self, send): # send a string to the VDU
        buf = 27 # maximum string size for Pi buffers
        self.spinProcessor.flushInput()
        ln = len(send)
        i = 0
        while i < ln:
            self.spinProcessor.write(send[i:i+buf].encode()+ b'\x0D')
            self.waitAck()
            i += buf

    def sendStringLn(self, send): # send a string no CR
        self.sendString(send)
        self.sendNl()

    def sendNl(self): # send a new line command
        self.spinProcessor.flushInput()
        self.spinProcessor.write(b'\x0E\x0D') # send new line
        self.waitAck()

    def setX(self,pos): # set X cursor
        pos += 14 # can't send zero or 13 so add 14
        if pos < 14 or pos > self.xWidth + 14: # outside limits
            print("ERROR ",pos,"IS OUT OF RANGE")
            return
        self.spinProcessor.flushInput()
        s = b'\x0A' + str(chr(pos)).encode() + b'\x0D'
        self.spinProcessor.write(s)
        self.waitAck()

    def setY(self,pos): # set Y cursor
        pos += 14 # can't send zero or 13 so add 14
        if pos < 14 or pos > self.yWidth + 14: # outside limits
            print("ERROR ",pos,"IS OUT OF RANGE")
            return
        self.spinProcessor.flushInput()
        s = b'\x0B' + str(chr(pos)).encode() + b'\x0D'
        self.spinProcessor.write(s)
        self.waitAck()

    def erase(self,line,start,length): # erase "line" from "start" for "length"
        if start + length > self.xWidth:
            length = self.xWidth - start
        s = ""
        for i in range(0,length):
            s +=" "
        if line > -1: # set line to -1 to use current line
            self.setY(line)
        self.setX(start)
        self.sendString(s)
        self.setX(start)
        if line > -1:
            self.setY(line)

    def waitAck(self): #wait for acknowledgement
        ack = self.spinProcessor.read()

    # set colour for current line
    def setColour(self,rf,gf,bf,rb,gb,bb): # r,g,b foreground & r,g,b background
        colF = ((rf & 0x3) << 4) | ((gf & 0x3) << 2) | (bf & 0x3) | 0x40
        colB = ((rb & 0x3) << 4) | ((gb & 0x3) << 2) | (bb & 0x3) | 0x40
        s = b'\x0C' + str(chr(colF)).encode() +str(chr(colB)).encode()+ b'\x0D'
        self.spinProcessor.write(s)
        self.waitAck()

    def cls(self): # clear screen and set to home
        self.spinProcessor.write(b'\x01\x0D') # clear screen
        self.waitAck()

    def home(self): # go to top left
        self.spinProcessor.write(b'\x02\x0D') # home
        self.waitAck()
```

Change_to_Terminal_Services.spin

› Language: **Spin**

```
PUB OutTerm( pChar )
{{
DESCRIPTION: Output a character to terminal, this is the primary interface from the client to the driver in "terminal mode"
direct mode access uses the register model and controls the engine from the low level

PARMS: pChar - character to print with the following extra controls:


    $00 = Null - can't be represented by a string
    $01 = clear screen
    $02 = home
    $08 = backspace
    $09 = tab (8 spaces per)
    $0A = set X position (X + 14 follows)
    $0B = set Y position (Y + 14 follows)
    $0C = set color (color follows)
    $0D = return - but can't be passed in a string from serial
    $0E = return
    others = prints to the screen

    +128 to any other printable character, draw in inverse video

RETURNS: Nothing.
}}

  case gTermFlag
    $00: case pChar
        $00..$01: bytefill( gVideoBufferPtr, ASCII_SPACE, gTermNumCols * gTermNumRows)
                gTermCol := gTermRow := 0

        $02: gTermCol := gTermRow := 0


        $08: if gTermCol
              gTermCol--

        $09: repeat
              PrintTerm(" ")
            while gTermCol & 7

        $0A..$0C: gTermFlag := pChar
                  return

        '$0D: NewlineTerm

        $0E: NewlineTerm

        other: PrintTerm( pChar )

    $0A: gTermCol := (pChar-14) // gTermNumCols
    $0B: gTermRow := (pChar-14)// gTermNumRows
    $0C: gTermFlag := pChar & 7

  gTermFlag := 0

' end PUB ---------------------------------------------------------------------------
```

fast, the buffers will be overrun – but if we put in a delay, the screen update is slower. To solve this problem, we have made the display send an acknowledgment back to the Pi when the VDU is ready for new data.

08 Talking to the display

We want to communicate with the display using Python, and for ease of use we have implemented a helper class containing functions to make driving the display from any Python program easy – this is shown in the **vga_drive.py** listing. Pyserial sends and receives only byte array data, and this class does the conversions for you. The `sendString` method can cope with any length of string; it does this by splitting up the string into chunks of 27 bytes or less and sending each separately. Look at each of the methods to see what they do; this is your toolkit.

09 Using your toolkit

The **vga_demo1.py** listing (overleaf) shows a basic program to send numbers and strings to the display. First the screen is cleared and we send a message to the top line. Note how we send a variable as a string using the `str()` function, and also note the use of the `erase()` method to blank out part of a line with spaces. If we send the line number as -1 it will work on the current line, but you can use a specific line number as well. The

Top Tip

Colours

Once you get a theme you like, you could hack around at the bottom of the HelloPi code and set them permanently. Look for the variable names in the **WMF_Terminal_ Services** file.

vga_demo1.py

Language: **Python**

```python
#!/usr/bin/env python3
#demo 1 using Vga_drive class driver
#By Mike Cook January 2019

import time

from vga_drive import Vga_drive

#1024 x 768 @ 57Hz settings: 128 x 64 characters
#800 x 600 @ 75Hz settings: 100 x 50 characters
#640 x 480 @ 69Hz settings: 80 x 40 characters
xWide = 100 ; yWide = 50 # Pick the resolution for your VDU settings
vga = Vga_drive(xWide, yWide)
time.sleep(1.5) # let the Propeller chip reset
s1 = ") Now is the time to come to the aide of the party "
s2 = " Again with feeling "

def main():
    print("Spin VGA driver demo1 - Ctrl C to stop")
    vga.cls()
    speed = 0.05 # time between each change
    while True:
        vga.sendStringLn(" Hello from the Raspberry Pi") # send with CR
        for i in range(1,yWide+20): # allow scrolling at the end
            vga.sendString(str(i)+s1) # send number and string
            time.sleep(speed*4)
            vga.sendString(s2)
            time.sleep(speed * 8)
            vga.erase(-1, len(s1)+len(str(i)), len(s1)) # remove last string
            time.sleep(speed)
            vga.sendNl() # new line
        vga.cls()

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

program will write more lines than you can fit on the screen, so you can see the screen scrolling.

10 Colour themes

In the VGA drivers there is only bit of memory per pixel; however, this bit can be displayed as two of any of the 64 available colours. The colours to be displayed can be set, but only for one line at a time. The foreground and background colours can be set independently, but to see them you need them to be different. The program in the **vga_demo2.py** listing displays ten different 'recommended' colour themes, as well as two fun chaotic or random themes. Note that the top line can be made to have the inverse of the rest of the window.

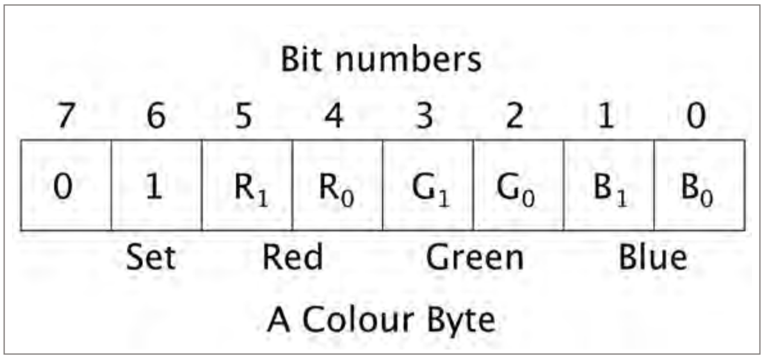


Figure 3 The composition of a colour byte

11 In conclusion

Well, that about wraps it up from us for this project, but you still have plenty to explore. Look at the examples that have multiple bits per pixel, giving you more colours and graphic capabilities – see our GitHub page for a starting example. We also have not broken out all the processor GPIO pins; you can add a header strip for ground and the first eight pins very easily. All the other unused pins are up for grabs as well. A Propeller processor must surely be the most capable and flexible of all things you can add to your Raspberry Pi.

vga_demo2.py

▶ Language: **Python**

DOWNLOAD THE FULL CODE:

magpi.cc/XCwzVh

```python
#!/usr/bin/env python3
#demo 2 using Vga_drive class driver
#with added colour theme switching
#By Mike Cook January 2019

import time, random
from vga_drive import Vga_drive

#1024 x 768 @ 57Hz settings: 128 x 64 characters
#800 x 600 @ 75Hz settings: 100 x 50 characters
#640 x 480 @ 69Hz settings: 80 x 40 characters
xWide = 100 ; yWide = 50 # Pick the resolution for your settings
vga = Vga_drive(xWide, yWide)
time.sleep(1.5) # let the Propeller chip reset
random.seed()
s1 = ") more into the breach dear friends:- "
#colours for themes
redF =   [3,0,3,0,0,0,3,3,3,0]
redB =   [0,3,0,0,2,3,0,3,1,1]
greenF = [3,0,3,3,0,0,1,3,3,3]
greenB = [0,3,0,0,2,1,0,1,1,1]
blueF =  [3,0,3,0,0,0,0,3,3,0]
blueB =  [0,3,2,0,0,0,0,0,2,1]
ThemeNames = ["White on black","Black on white","Atari/C64 theme",
            "Apple ][ ","Wasp/yellow jacket","Autumn","Inverse Autumn",
            "Creamsicle","Purple orchid","Gremlin","Chaos 1","Chaos 2"]

def main():
    print("Spin VGA colour theme demo - Ctrl C to stop")
    vga.cls()
    vga.sendStringLn("Hello from the Raspberry Pi")
    vga.sendString("starting the theme demo")
    time.sleep(5) # time for monitor to adjust
    while True:
        for j in range(0,12):
            if j >= 10:
                chaos(j & 1)
            else :
                changeTheme(j,1) # use 0 for no special top line
            vga.sendStringLn(
            "Hello from the Raspberry Pi - Theme "+ThemeNames[j])
            for i in range(1,yWide-1):
                vga.sendString(str(i)+s1) # send number and string
                vga.sendStringLn(" Again with feeling ")
            vga.sendString("This is the last line")
            time.sleep(2)
            vga.cls()

def changeTheme(theme, top): # top = 1 if inverted
    start = top & 1
    if start != 0:
        vga.setY(0)
        vga.setColour(redB[theme],greenB[theme],blueB[theme],
                    redF[theme],greenF[theme],blueF[theme])

    for i in range(start, yWide):
        vga.setY(i) # choose line
        vga.setColour(redF[theme],greenF[theme],blueF[theme],
                    redB[theme],greenB[theme],blueB[theme])
    vga.home()

def chaos(t): #just a bit of fun
    for i in range(0,yWide):
        vga.setY(i) # choose line
        c = [random.randint(0,3) for j in range(0,6)]
        if t == 1 :
            vga.setColour(c[0],c[1],c[2],c[3],c[4],c[5])
        else:
            vga.setColour(c[0],c[1],c[2],c[0]^3,c[1]^3,c[2]^3) # inverse back
    vga.home()


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

Develop an Android app

Can you create an app with just a Raspberry Pi and install it on your Android phone? Of course you can, and here's how...

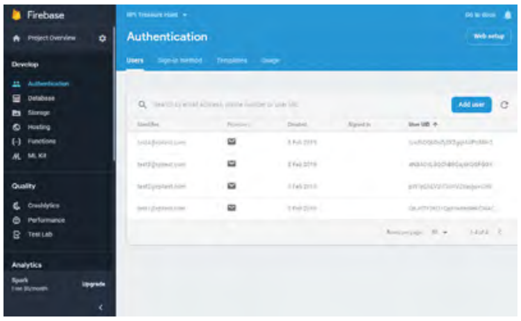
MAKER

Mark Vanstone

Educational software author from the nineties, author of the ArcVenture series, disappeared into the corporate software wasteland. Rescued by the Raspberry Pi!

magpi.cc/YiZnxL

The Raspberry Pi is generally considered to be a great platform for learning how to program and control electronics projects, but can you create programs to be used on other platforms like PCs or even mobile phones? Well, considering the cost of buying a Raspberry Pi, you may think it could be a bit limited for real-world development, but this article will explain how you can use your Pi to create a real-world, data-driven, useful mobile app that you can install on a phone and use to communicate in real-time with your Raspberry Pi.



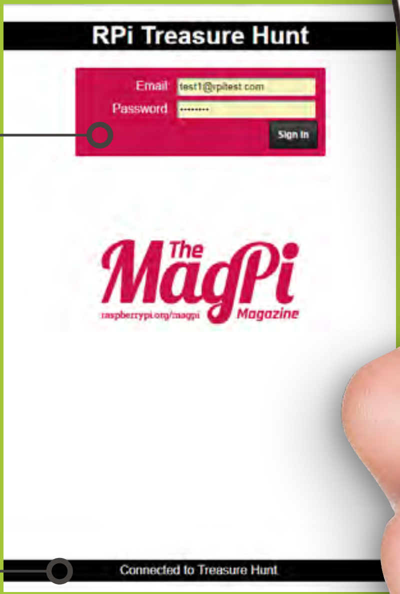
In the Firebase Console, under Authentication, you can set up users for your app

There are several methods of making mobile apps. Generally, the way to produce an app is to write it using a development system which compiles your code into a package for a specific platform. These are called 'native' apps and you would need to compile different versions for Android phones and iPhones or Windows phones. There are other apps which have a native app wrapper but then display HTML5 pages inside the wrapper or maybe even just work inside a web browser. These are called 'web' apps.

The problem with web apps is that they rely on having a connection to the internet to view the HTML pages, and any data handling will generally require calls to server-side scripts and databases, as with normal webpages. Now, there is a solution to this problem and it is called 'PWA'. That stands for 'progressive web app' and has elements of the way a native app works, but can be programmed using webpage coding.

A PWA can keep a copy of the app screens and even a local copy of database data so that when there is no internet connection it still works. It can also be installed onto a mobile phone, with its

You can allocate sign-in details for your users in the Firebase Console


RPi Treasure Hunt

Email: test1@rpitest.com

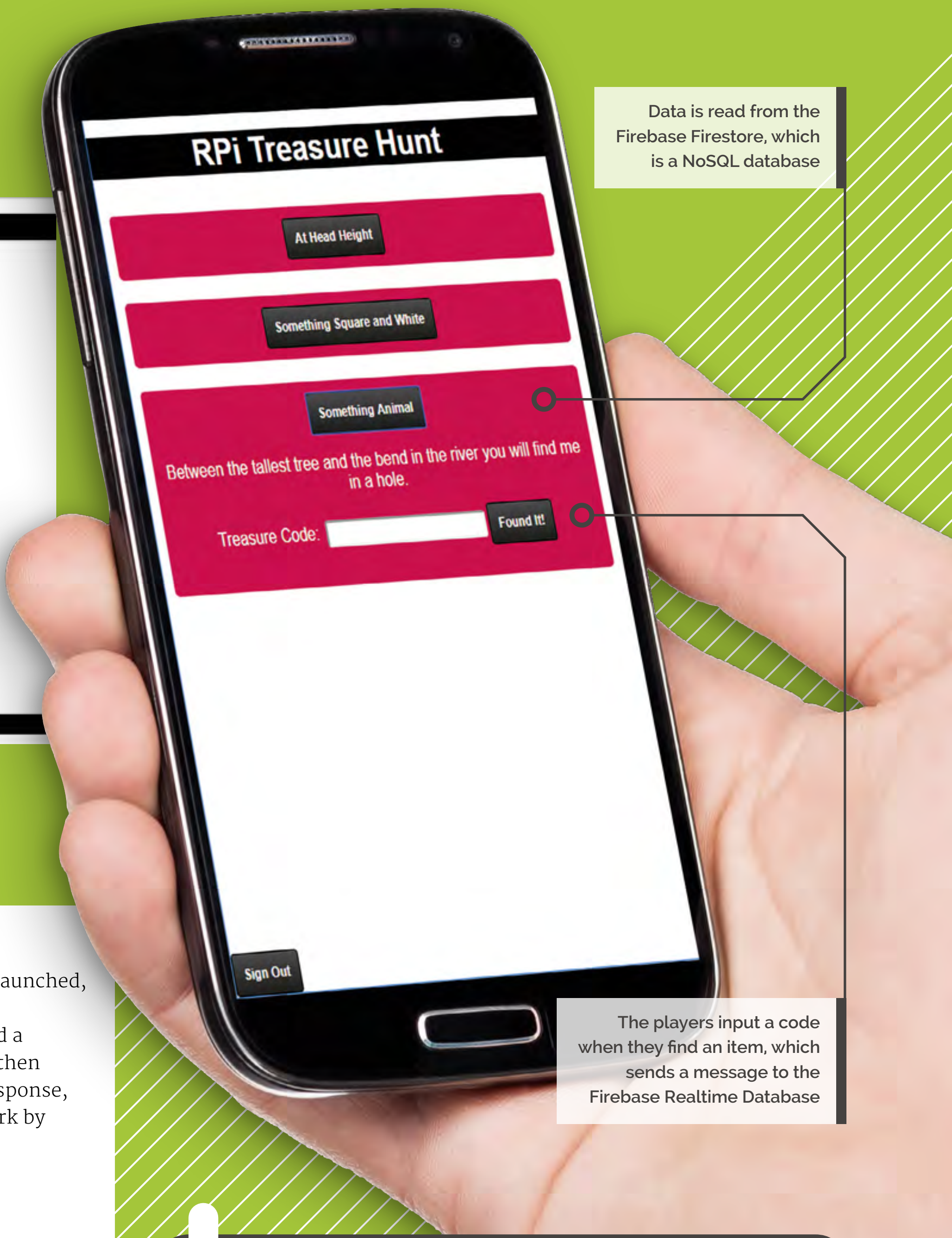
Password: ········

Sign In

Connected to Treasure Hunt

When the app launches, it checks that it has started the Firebase framework

Data is read from the Firebase Firestore, which is a NoSQL database


RPi Treasure Hunt

At Head Height

Something Square and White

Something Animal

Between the tallest tree and the bend in the river you will find me in a hole.

Treasure Code:

Found It!

Sign Out

The players input a code when they find an item, which sends a message to the Firebase Realtime Database

own icon on the home screen and, when launched, looks and works like a native app.

To write a PWA on the Raspberry Pi, and a system to capture data from the app and then make external electronics do things in response, we are going to get the help of a framework by Google called Firebase. .

“We will make a treasure hunt app for mobile phones”

So here’s the plan: using Raspbian, we will install the necessary modules on the Raspberry Pi and set up a new Firebase project. We will make a treasure hunt app for mobile phones which will talk to a Python program on the Raspberry Pi. That program will use the GPIO pins to light up coloured lights for each team as they find hidden treasure or complete the treasure hunt. All of this can be developed and deployed from the Raspberry Pi!

WHO IS SUPPORTING **PWA?**

This is a very simple example and the scope of what can be done with progressive web apps is much greater. For a lot of mobile app projects, this framework will be quite sufficient and perhaps it needs to be asked if it is necessary to submit apps to the App Store or Play Store (and the cost incurred in that) when a simple link can be sent to a user who can install a functional app that can look and feel like a native app. There is extensive documentation on the Firebase site covering all the elements of the framework. Google is committed to supporting PWA technology, and Microsoft has said that it will support it too. Although Apple is less vocal about its support, a PWA shortcut does install an app on an iPhone and the PWA acts to the user like a native app. To see a list of what is available through a PWA, view the chart at **whatwebcando.today**. Excepting a few hardware-specific functions of a mobile device, the list provides a great range of functions which will no doubt expand as more PWAs are made and used.

Making the app

A step-by-step guide to building your mobile app

You'll Need

- Raspbian (latest version)
- Node.js **magpi.cc/ogOWTs**
- firebase-tools **firebase.google.com**
- Pyrebase Python module **magpi.cc/OPQUtg**
- Android phone
- LEDs, resistors, jumper wires, and breadboard

01 Getting the right version of Node.js

The first thing to do is get Node.js installed. Unfortunately, currently the version that is available via the apt-get command is not up-to-date enough, so we need to do a manual install. Head over to **magpi.cc/ogOWTs** and download the ARM package. To find out which version you need, type `uname -m` into a Terminal window. Most Pi models will need the v7 package. You will need to unpack the file to a suitable directory (home directory is fine) and then, in the Terminal window, go to the unpacked directory with, for example, `cd node-v10.15.1-linux-armv71`. Now type `sudo cp -R * /usr/local/` to copy the files over.

02 Get Firebase tools

Firebase needs to use Node.js for its toolset and once you have the Node files in place, you can check that they are ready to use by typing `node -v`, which should reply with the version you have just installed (10.15.1 in our case). You can also check the Node Package Manager by typing `npm -v`. We will need npm to install the Firebase tools. We do that by typing `sudo npm install -g firebase-tools` into the Terminal window. The install will take a little time; when it's finished, run the same command again, to update one last package.

manifest.json

Language: **JSON**

```
{
    "short_name": "RPiTreasure",
    "name": "RPi Treasure Hunt",
    "icons": [
      {
        "src": "/images/rpit192.png",
        "type": "image/png",
        "sizes": "192x192"
      },
      {
        "src": "/images/rpit512.png",
        "type": "image/png",
        "sizes": "512x512"
      }
    ],
    "start_url": "/",
    "background_color": "#fff",
    "display": "standalone",
    "scope": "/",
    "theme_color": "#fff"
  }
```

03 Get a Firebase account

Google very kindly provides basic access to its Firebase service for free. All you need to do is register for an account at **firebase.google.com** and you'll be able to do all the things in this tutorial. You can do this by signing in with your existing Google account, if you have one. Google also provides a sliding scale payment system if you need to expand your requirements. When you have your account, you'll be able to access the Firebase console at **console.firebase.google.com**. On the console front page, you'll see an option to add a new project.

04 The new project

Select the 'Add Project' button and you can set up your Firebase project. You'll need to type in a project name and accept the terms and conditions, then select 'Create project'. When the project has been created, you'll be taken to the Firebase console where you'll see a range of tools down the left-hand side. We will be using the 'Develop' tools which should be shown here. First, let's look at Authentication. You'll need to set a sign-in method, for which 'Email/Password' can be enabled. Then, in the 'Users' tab, you can Add a user. Also at this stage, go into the Database section and create a Firestore database.

05 Storage

Firebase offers three categories of data storage. The first is database storage from which we can use either Firestore or the Realtime Database (more on this later). The second is labelled as 'Storage' in the menu, which provides

treasure.py

▶ Language: **Python**

DOWNLOAD THE FULL CODE:

magpi.cc/OdnyDP

```python
import pyrebase
import time
from gpiozero import LED

config = {
    "apiKey": "Your apiKey goes here",
    "authDomain": "Your hosting domain goes here",
    "databaseURL": "Your hosting URL goes here",
    "projectId": "Your project id",
    "storageBucket": "Your storage domain",
    "messagingSenderId": "Your sender id"
}

firebase = pyrebase.initialize_app(config)
numberOfTreasure = 3
led = {}
teams = {}
teams[0] = {"email":"test1@rpitest.com","led":17,"treasure":[]}
teams[1] = {"email":"test2@rpitest.com","led":18,"treasure":[]}
teams[2] = {"email":"test3@rpitest.com","led":22,"treasure":[]}
teams[3] = {"email":"test4@rpitest.com","led":23,"treasure":[]}

for f in range(len(teams)):
    led[f] = LED(teams[f]["led"])
    led[f].off()

db = firebase.database()

def processMessage(d):
    if(d != None):
        for v in d.values():
            updateTeam(v["email"], v["item"])

def ledFlash(t):
    for f in range(5):
        led[t].on()
        time.sleep(.2)
        led[t].off()
        time.sleep(.2)

def ledOn(t):
    led[t].on()

def updateTeam(t,i):
    for td in teams:
        if teams[td]["email"] == t:
            if i not in teams[td]["treasure"]:
                teams[td]["treasure"].append(i)
                if len(teams[td]["treasure"]) >= numberOfTreasure:
                    print(t+" complete!")
                    ledOn(td)
                else:
                    ledFlash(td)

def streamHandler(message):
    if message["event"] == "put" or message["event"] == "patch":
        processMessage(message["data"])

myStream = db.child("msg").stream(streamHandler)

while 1:
    time.sleep(.1)
```

an area to store files generated by an app. The third, which we will look at now, is 'Hosting'. If we go into the Hosting section, we'll see instructions about installing firebase-tools, which we did previously. Then there are instructions about how to set up the project on your Raspberry Pi – so let's follow along with them.

06 Local projects

First, in our Terminal window, make a directory for our app with `mkdir appname`. Then `cd appname` to go into that directory. From the Firebase instructions, type `firebase login`. A browser window will appear and you'll be asked to log in to your Google/Firebase account. With that done, go back to the Terminal window and type `firebase init`. You'll be asked which features you'd like to use, which could be all of them for the moment. It will ask you to select your project, then it'll ask you a string of questions about your project. Just select the default option (press **ENTER**) for all of the questions.

07 Setup done

Now Firebase has set up a default project for us, which has everything we need to build our app. Go back to the Firebase Console and select 'Finish' (we'll deploy a bit later). Inside our app folder we will find another folder called **public**. Inside this we have our **index.html** file, which is where our app

sw.js

> Language: **JavaScript**

```
var cacheName = 'rpitreasure';
var filesToCache = [
  '/',
  '/index.html',
  '/images/logo.png',
];

self.addEventListener('install', function(e) {
  console.log('[ServiceWorker] Install');
  e.waitUntil(
    caches.open(cacheName).then(function(cache) {
      console.log(
'[ServiceWorker] Caching app shell');
      return cache.addAll(filesToCache);
    })
  );
});

self.addEventListener('activate',  event => {
  event.waitUntil(self.clients.claim());
});

self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request,
{ignoreSearch:true}).then(response => {
      return response || fetch(event.request);
    })
  );
});
```

will be built. Let's have a look at that file: open your favourite programming editor (you could try Geany if you are undecided) and load the **index.html** file. Now go to the Firebase console, select the cog next to 'Project Overview' and select 'Project Settings'.

08 Add some Firebase

In the Project Settings page, near the bottom you'll see a panel saying that there are no apps in your project. Select the round web icon (</>) and you'll get a pop-up with some JavaScript code. Copy that code and go back to editing your **index.html** file. Now replace the code that starts `<!-- update the version number` and ends `/init.js></script>` with the code you have just copied from the Firebase console. This bit of code will connect our web app with the Firebase services. We can now deploy our test app by going back to our Terminal window and typing `firebase deploy`. After uploading the necessary files, the app will be ready for testing in a browser.

09 Testing testing

We can test on the Raspberry Pi Chromium browser or on a mobile device, but make sure that you are using either Chrome or Chromium. Other browsers do support PWAs, but let's keep it simple for now. After the project has been deployed, we'll be given a web address of the hosting URL in our Terminal window. Go to this address in a browser and you should see confirmation that the app is working and has connected. You can also test locally if you set up the Chrome Web Server extension. If you don't see a message saying 'Firebase SDK loaded with auth, database, messaging, storage', then go back over the previous steps

10 Let's make an app

The first thing to code will be the app that goes on the phone. Have a look at the listing of **index.html**. What this script does is to present a

IS THIS ANY GOOD FOR **REAL APPS?**

Of course, you will get nay-sayers proclaiming that if it is not a native app then it's not a real app, and the restrictions of the App Store and Play Store currently mean that it is difficult to adapt a PWA for distribution that way. The fact of the matter is that in industry, the cost of any project is key to it happening and while native app development is a very costly activity, PWAs can be coded in a fraction of the time and have no restrictions of the App Store or Play Store. That's not to say that PWAs cannot be published through those portals, it's just that they have to be wrapped in a framework like Cordova to make them into native apps.

For the maker/coder community, the ability to publish apps without these restrictions is no doubt a benefit and possibly a technology that will supersede the platform-dependent stores. If nothing else, PWA technology gives us a great opportunity to create useful apps with just a Raspberry Pi.

index.html

▶ Language: **HTML**

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="manifest" href="/manifest.json">
    <title>Welcome to The RPi Treasure Hunt</title>
<script src="https://www.gstatic.com/firebasejs/5.8.1/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "Your apiKey goes here",
    authDomain: "Your hosting domain goes here",
    databaseURL: "Your database URL goes here",
    projectId: "Your project id",
    storageBucket: "Your storage domain",
    messagingSenderId: "Your sender id"
  };
  firebase.initializeApp(config);
</script>
<script>
  if('serviceWorker' in navigator) {
    navigator.serviceWorker.register('/sw.js')
      .then(function() {
            console.log('Service Worker Registered');
      });
  }
</script>
    <style media="screen">
      body { background: #fff; color: #000; font-family: Helvetica, Arial, sans-serif; margin: 0; padding: 0; text-align: center; background: url(images/logo.png) no-repeat 50% 300px fixed; background-size: 50% }
      #title{ background: #000;color:#fff}
      #signin{ margin:10px; padding:10px;}
      #signout{ position: fixed; bottom: 0px;text-align: center}
      .clueholder{ background:#ca0d4c; color:#fff; border:10px solid #fff; margin:0px; padding:10px;-webkit-border-radius: 15px; -moz-border-radius: 15px;border-radius: 15px;}
      .theClue{display:none}
      #loginform{background: #ca0d4c;color:#fff;padding:10px;text-align: right;width:300px;margin: auto;-webkit-border-radius: 3px; -moz-border-radius: 3px;border-radius: 3px;}
      #load {
        display: block; text-align: center; background: #000; text-decoration: none; color: white;
        position: fixed; bottom: 0;padding-top: 5px;padding-bottom: 5px;
        width: 100%; height:18px;
      }
      input{ -webkit-border-radius: 3px; -moz-border-radius: 3px;border-radius: 3px;margin:3px;padding:2px}
      button {
        border:1px solid #000; -webkit-border-radius: 3px; -moz-border-radius: 3px;border-radius: 3px;font-size:12px;font-family:arial, helvetica, sans-serif; padding: 10px 10px 10px 10px; text-decoration:none; display:inline-block;font-weight:bold; color: #fff;
        background-image: -webkit-gradient(linear, left top, left bottom, from(rgb(77, 77, 77)), to(rgb(29, 29, 27)));;
      }
    </style>
  </head>
  <body>
    <div id="title">
      <h1>RPi Treasure Hunt</h1>
    </div>
    <div id="content">
      Loading RPi Treasure Hunt App.
    </div>
    <p id="load">Connecting ...</p>
    <script>
      clues = null;
      email = "";
      document.addEventListener('DOMContentLoaded', function() {
        try {
          let app = firebase.app();
          let features = ['auth', 'database', 'messaging', 'storage'].filter(feature => typeof app[feature] === 'function');
          document.getElementById('load').innerHTML = `Connected to Treasure Hunt.`;
          firebase.firestore().enablePersistence();
        } catch (e) {
          console.error(e);
          document.getElementById('load').innerHTML = 'Error connecting to the Treasure Hunt.';
        }
      });
      firebase.auth().onAuthStateChanged(function(user) {
        window.user = user; // user is undefined if no user signed in
        console.info("user changed - is now "+user);
    if (user == null){
      setLoginPage();
    }else{
      document.getElementById(‘load’).style = "display:none"
      getClueData();
    }
   });
   function signinUser(){
     email = document.getElementById("email").value
     password = document.getElementById("pass").value
     firebase.auth().signInWithEmailAndPassword(email, password)
     .catch(function(err) {
       console.error(err);
     });
   }

   function getClueData(){
     var db = firebase.firestore();
     db.collection("Clues").get().then(function(querySnapshot) {
       setCluesPage(querySnapshot);
     });
   }

   function signoutUser(){
     firebase.auth().signOut()
     .catch(function (err) {
       console.error(err);
     });
   }

   function openThisClue(o){
     cluelist = document.getElementsByClassName(‘theClue’)
     for(c=0;c<cluelist.length;c++){
         cluelist[c].style = "display:none";
     }
     theclue = document.getElementById(o);
     theclue.style = "display:block";
   }

   function foundItem(i){
     code = document.getElementById("clueCode_"+i).value;
     clues.forEach(function(doc){
       if(doc.id == i && code == doc.data().code){
         console.log("we have a winner");
         var newMsgKey = firebase.database().ref().child(‘msg’).push().key;
         var postData = {
           email: email,
           item: i
         };
         var updates = {};
         updates[‘/msg/’ + newMsgKey] = postData;
         firebase.database().ref().update(updates);
         document.getElementById("clue_"+i).style = "display:none"
       }
     });
   }

   function setLoginPage(){
     document.getElementById(‘content’).innerHTML = "<div id=’loginform’>Email: <input id=’email’ type=’text’><br>Password: <input id=’pass’ type=’password’><br><button onclick=’signinUser();’>Sign In</button></div>";
   }

   function getClueDisplay(cluelist){
     out = "";
     clues = cluelist;
     clues.forEach(function(doc){
       out += `<div class="clueholder" id="clue_`+doc.id+`"><button onclick="openThisClue(‘clueDetail_`+doc.id+`’)">`+doc.data().name+`</button><div id="clueDetail_`+doc.id+`" class="theClue"><p>`+doc.data().clue+`</p><p>Treasure Code: <input id="clueCode_`+doc.id+`" type="text"><button onclick="foundItem(‘`+doc.id+`’)">Found It!</button></p></div></div>`;
     });
     return out;
   }

   function setCluesPage(clues){
     out = getClueDisplay(clues);
     out += "<div id=’signout’><button onclick=’signoutUser()’>Sign Out</button></div>";
     document.getElementById(‘content’).innerHTML = out;
   }
  </script>

 </body>
</html>
```

sign-in page to allow users (defined in our Firebase Authentication) to sign in and then it gets a list of clues from the Firebase Firestore database and displays them. When a user finds an item of treasure, there will be a code number (we will set this in our Firestore) which, if they enter it correctly, will remove that clue from the list and send a message to the Firebase Realtime Database. There are two types of database that Firestore supports: the Firestore that our treasure hunt data is in, and the Realtime Database.

11 Realtime Database

The Realtime Database will trigger an event if another program is listening when data changes. That's how we'll transfer data back to our Raspberry Pi. We need to make an adjustment to the security rules to make this easy. Note that this is not a good idea for production purposes, but we can change the security so that we can read and write to the database without authentication. We do this by going to the Realtime Database in the Firebase console, selecting the 'Rules' tab, and changing the '.read' value to `true` – and the same with the '.write' value. Normally we would want to have a bit more security around a database, but for this example we'll keep it simple.

12 Make the app act like an app

There are a couple of other things we need to add to make our app installable on a mobile phone. The first is the **sw.js** file. This is a service worker file and enables us to cache files so that they don't need an internet connection to load. You'll see the script to register this file in **index.html**. The other is a manifest file that allows the app to be installed and will produce a message asking the user if they want to install it. The manifest file is linked near the top of **index.html**.

13 The Python connection

We need to gather the data that is being created by our app – this could be done with a simple webpage on our Raspberry Pi, but that would be boring. Why not have a score indicator with flashing lights and things like that? To make this, we'll need to write a Python program. Have a look at **treasure.py** and see how we do this… but wait: we need to have access to a module called pyrebase, which is a wrapper for the Firebase functions and makes it really easy for us. To install it, just type `pip3 install pyrebase` into a Terminal window.

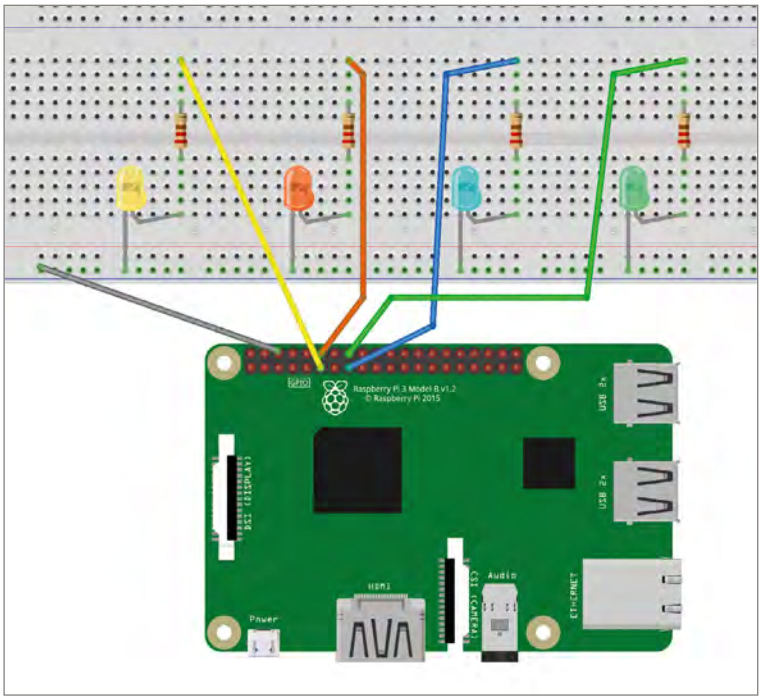
14 Wiring up

To be honest, you could have any sort of grand scheme for a scoreboard, but we can wire up an LED for each team or player in our treasure hunt. See the wiring diagram (**Figure 1**) we are using, which will flash the team's LED when they discover treasure and keep the LED lit when they have finished. If you have lots of players, you may want to use LED strips or even a much larger build.

15 What does this mean?

What we have done is to create a real-world app for mobile devices with just a £35 Raspberry Pi and a free Google account. This is quite a new development and, even as this article was being written, the Firestore service went from beta to live. This means that things may change quite quickly, so you may need to refer back to the Firebase website for updates.

Figure 1 How to connect four LEDs. You can flash the LEDs when data is received using GPIO connections



ArduinoPixed

SPECS

PORTS:
3 × USB 2.0

FEATURES:
1 × LED

1 × Thermistor (heat sensor)

1 × Push-button

ASSEMBLY:
No soldering required

▶ Reference Designer, Inc ▶ **magpi.cc/FQsHFX** ▶ From £18 / $23

Is this Arduino-on-Pi made easy? **Rob Zwetsloot** digs out a Zero to give it, and its USB hub functionality, a try

Getting Arduino hardware working with a Raspberry Pi is a common goal in the wider maker community. Right now, Arduino microcontrollers can do very specific stuff better than the Pi, and vice versa, so combining the two can be an amazing way to truly power up a project.

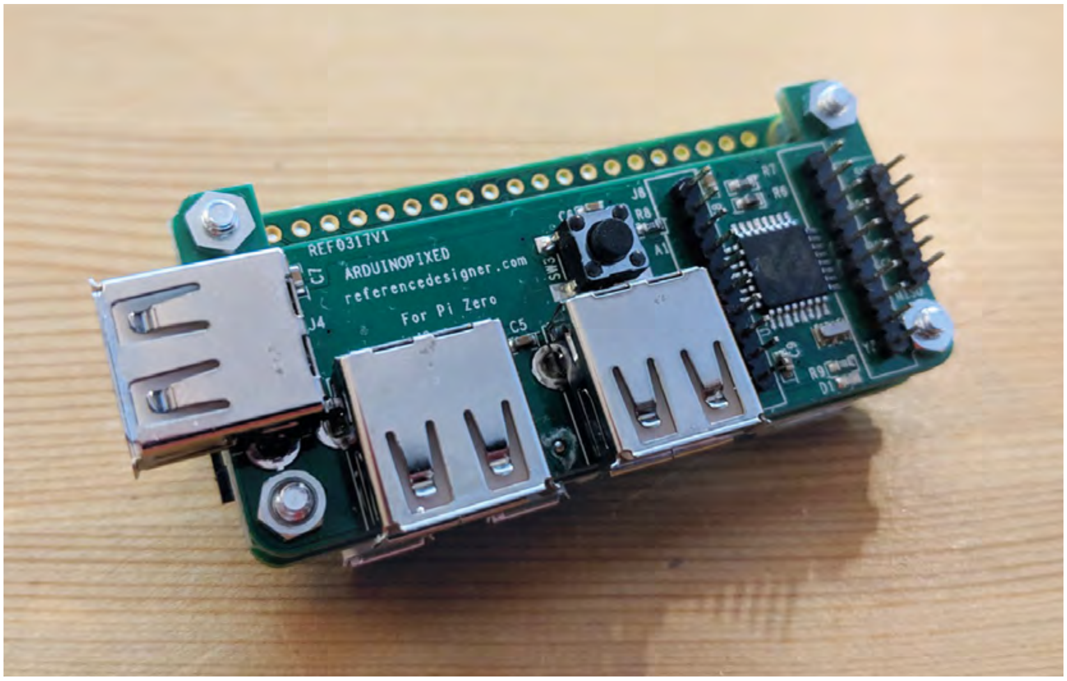
This is where the ArduinoPixed comes in. It's a relatively inexpensive add-on for the Pi Zero (yes, it's twice as much as a Pi Zero, but that is already extremely cheap) that sits comfortably underneath the latter, connecting via pogo pins and a bit of friction from the supplied screws. This instantly gives you access to three USB 2.0 type-A sockets, but the real magic begins once you install Arduino software on the Pi.

From here, you can start doing simple things: program the LED to blink, have it interact with the button, etc. The on-board chip is an ATmega328, and with the series of A/D extension pins on the board, you can start controlling a little more stuff.

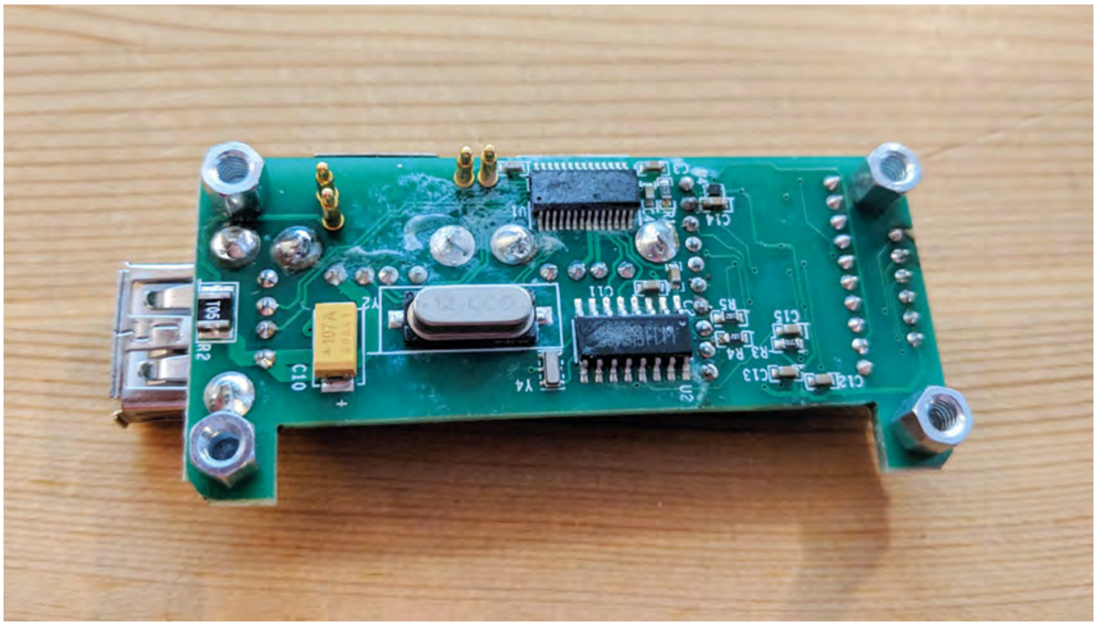
Perfectly sized

The standard Arduino IDE controls the Arduino side of the ArduinoPixed, meaning you don't need to learn anything particularly new to get it working – unless this is your first time using an Arduino system, that is.

There are some great example programs to follow along with, to get you used to how the

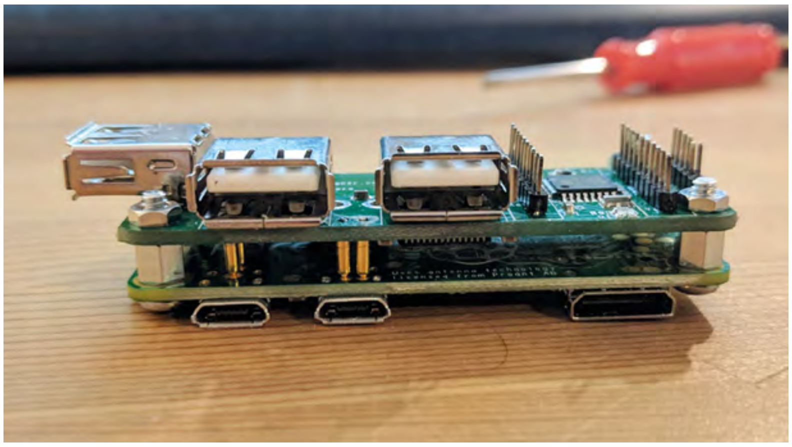


▶ It sits perfectly underneath the Pi Zero, with very little in the way of overhang



The three USB ports it adds to the Pi Zero are very welcome

It does add a lot of depth to a Pi Zero's footprint, so your regular case won't really work



> One of the best solutions we've seen for marrying Raspberry Pi and Arduino – with minimal construction and setup time

Pixed works on the Pi, allowing for a jumping-off point to control other bits and pieces with the Arduino side of the pair.

The whole board fits very neatly over the Pi Zero, keeping the GPIO pins clear so you can interact with them, with only part of a USB port overhanging the usual form factor. It does add a lot of depth to the Pi Zero's footprint – while unavoidable, the jump from around 6 mm to 18 mm is significant.

Joined together

With a few Pi/Arduino combos we've seen, the Arduino part can often be used separately from the Pi. Due to the way the ArduinoPixed is powered, you could technically do this, but it's not as easy as more dedicated Arduino boards with a dedicated power input. Still, none of those interacts with a Pi in the same way.

Speaking of Pi interaction, it's worth noting that the Pixed does not fit onto a standard-sized Raspberry Pi model. As well as the mounting points only properly working on a Pi Zero, the pogo connectors specifically require the layout on the underside of the Zero. It's not something a little solder can't fix if you're really determined, but out of the box is a different matter.

Either way, it's one of the best solutions we've seen for marrying Raspberry Pi and Arduino – and it works very well to boot, with minimal construction and setup time. We look forward to the next big project that requires it...

Verdict

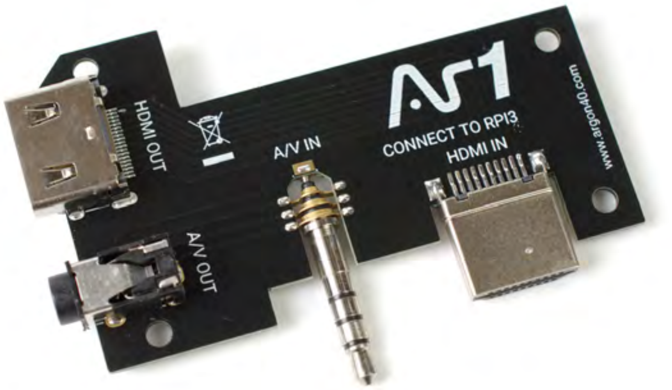
A quick build, even quicker software setup, and you're away with a fantastic marriage of Arduino and Raspberry Pi.

9/10

Argon One

▶ Argon Forty ▶ magpi.cc/oeEidS ▶ £23 / $25

A robust metal case with a hidden GPIO port. By **Gareth Halfacree**



▲ A clever daughterboard moves the Pi's HDMI and AV ports to the rear, tidying the cabling

It's the weight of the Argon One that first surprises. Whereas most Raspberry Pi cases are made out of plastic, the bulk of the Argon One is solid aluminium – and it's immediately noticeable, even before it's taken out of the box.

Inside the hefty shell is a board which connects to the Raspberry Pi's GPIO header, giving it a smart power button – found at the rear of the case – which comes with software to allow the Pi to be safely shut down with a single press. There's also a fan, and a daughterboard which snaps into a Raspberry Pi 3B or 3B+ to shift its HDMI and analogue audio-visual (AV) ports to the rear.

Assembly is quick and easy: click the daughterboard home, add a bundled thermal pad to the Pi's processor, slide everything onto the GPIO header, then screw it home before adding the plastic base.

Lower temperatures

Using the large aluminium case as a heatsink makes a real difference to temperatures: an uncased Pi 3B+ quickly reaches 65°C running a CPU-heavy benchmark, but peaks at 40.8°C in the Argon One. Sadly, the fan makes little difference, but it can be turned off in the bundled software.

There's only one real flaw in the Argon One: it draws too much power, causing the Pi to flash the 'undervoltage' lightning symbol and throttle its performance. It's a flaw that can only be fixed by purchasing Argon Forty's recommended power supply at an extra £11 / $12.50.

▼ Sleek and heavy, the Argon One adds cooling and a smart power button to a Pi 3B or 3B+

SPECS

DIMENSIONS: 106x95x34 mm

WEIGHT: 180 g (excluding Pi)

MATERIALS: Aluminium upper, plastic lower

BOARDS: Argon Forty Power Board, Argon Forty HDMI/AV Daughterboard

Verdict

The Argon One is attractive, and hiding the GPIO header under a magnetic cover is clever. You should factor in the cost of the recommended power supply. The poor fan performance is the only real issue.

7/10

Keybow

▶ Pimoroni ▶ **magpi.cc/MytktH** ▶ £50 / $54

A mini mechanical keyboard with light-up keys. By **Phil King**

With customisable key mapping, this DIY keypad could be used for a wide range of purposes, such as a games controller, hotkey pad for applications, or to insert text/code snippets with a single key press.

Solderless assembly takes around 20 minutes. After attaching the supplied Pi Zero WH to an acrylic baseplate and shim, you push-fit the twelve switches into a key plate, slot that into the PCB board, then mount it on the Pi's GPIO header.

You do need to supply your own microSD card to hold the tiny (26.6MB) Keybow OS. Upon connecting the Keybow to a computer via the USB OTG cable, the OS is loaded into a RAM disk and it boots up in a mere 10–15 seconds.

One problem we encountered was getting the USB cable to insert fully into the Pi's USB OTG port, as it was obstructed by a thin bar on the acrylic shim – breaking that off solved the issue.

▼ You have the choice of clicky or linear (quiet) keys. The switches fit easily into a key plate

▲ The keys can be lit up in static or animated patterns for dazzling displays

Custom keys

Once connected, the keys light up in an animated rainbow pattern (determined by a PNG file). The default key mapping is a numerical pad.

> The OS is loaded into a RAM disk and it boots up in a mere 10–15 seconds

To change lighting and key mapping, you need to slot the microSD card back into your computer – there's no way to switch instantly between them. Editing a Lua file lets you select a sample key layout or your own custom one. An online tutorial explains how: **magpi.cc/WdAWZc**.

The real power of the Keybow lies in creating advanced macros to trigger a whole series of key combinations with just a single key press – ideal for repeated key-based tasks in applications.

SPECS

KEY SWITCHES: 12 × Kailh Speed

LIGHTING: 12 × APA102 RGB LEDs

BREAKOUT: I²C 4-pin

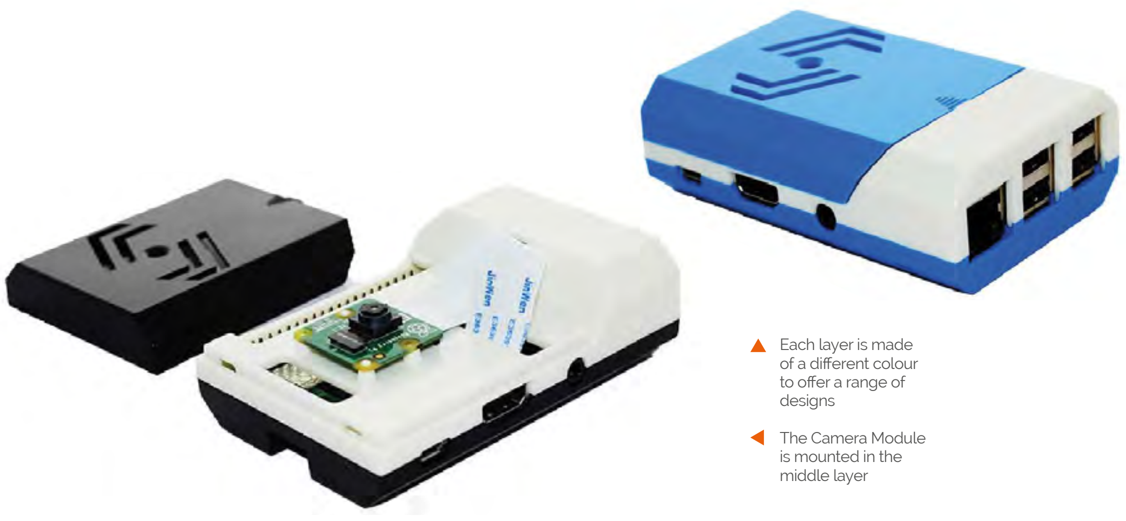
DIMENSIONS: 85×56.5×38 mm

Verdict

A little expensive and we had problems with the USB connection, but it looks cool when lit up, the key action is good, and it could prove handy for instant key sequences.

8/10

▶ SB Components ▶ **magpi.cc/MDAIHB** ▶ From £5 / $7



▲ Each layer is made of a different colour to offer a range of designs

◀ The Camera Module is mounted in the middle layer

SPECS

MATERIALS: ABS plastic with dual-shot injection moulding

MOUNTS: Camera and wall mounting

COOLING: Three vents double as a heatsink

PiShell

This three-piece case locks a Raspberry Pi and Camera Module together. **Lucy Hattersley** takes a look

The PiShell from SB Components is a compelling case option that might just fit your needs.

Made of high-quality materials, the PiShell offers a couple of features that lift it above the raft of Raspberry Pi cases.

The case is made of three parts: a 'base-bumper' that holds the Raspberry Pi board; the 'body-cover' that covers the board (with access holes for the GPIO pins, camera connector, and DSI display connector; and the 'camera-cover', which seals the whole device.

> A very good-looking case, available in a range of colour combinations

Camera Module integration is one of the key features that raise PiShell above the rest. The Camera Module board screws on to the middle board (the 'body-bumper') via two supplied screws. Once assembled, the camera points out through a hole in the case.

With the case flat on a surface, the camera points straight up, which we found handy for testing out with the Teachable Machine project (see page 40).

Wall mountable

On the reverse of the device are two holes for wall-mounting the device. We can see it neatly turning into a smart security camera or camera doorbell.

The case is made of ABS (acrylonitrile butadiene styrene) plastic. So it's got a nice feel, with good protection for your Raspberry Pi. There are vents on both the front and back.

The PiShell is a very good-looking case, available in a range of colour combinations. You can also buy a pack of five for £20 – good value for schools or clubs.

We like the PiShell case. It's not quite as sleek as the Official Case, but it's sturdy; that and the Camera Module integration make it a handy surround for your Raspberry Pi board.

Verdict

A solid case with a clever design that can house a Camera Module and be wall-mounted. We think the PiShell is a great option.

8/10

Can I start a CoderDojo club in my local area?

CoderDojo is a global network of free, volunteer-led, project-based programming clubs for children aged 7–17. Dojos are championed by individuals all around the world who are passionate about giving young people the opportunity to learn to code.

Starting a Dojo is a fun and incredibly rewarding experience

You don't need to possess technical skills to start a Dojo. The most important attribute is that you can bring people together for a shared goal.

We're ready to support you by providing:

- **Learning resources and guides**
- **A free event management system**
- **Certificate templates, digital badges, and giveaways**

"I started a Dojo to give my kids a place to meet other children also interested in programming and making games. I get to see them making new friends, learning from one other, and they loved it. Realising how I had created such a wonderful place for children has ignited a spark in me."

- Maroes, CoderDojo NL

Start your own club. Join us at CoderDojo.com

10 Best: Home automation add-ons

Get your Pi controlling your home with these IoT gadgets

The dream of home automation is forever present among makers and other tech types, and the Raspberry Pi has helped many people turn their Enterprise fantasies into a reality. Here are some kits, add-ons, and other gadgets that can help.

SparkFun ESP32 Thing

WiFi smart home

ESP32 is a standard that lets you use wireless LAN to communicate with the various IoT/home automation projects you've set up. This one also has Bluetooth as part of it!

▶ £21 / $23
▶ **magpi.cc/UchWDj**

Automation HAT

All-in-one automation

If you have big home automation plans, or already have some serious IoT around your house, you'll want to look into the Automation HAT: you can plug a lot of devices into it.

▶ £29 / $31
▶ **magpi.cc/fgMGmr**

Google AIY Voice Kit

Vocal commands

We had this as a freebie with the magazine once – the AIY Voice Kit allows you to add powerful voice control to your Raspberry Pi, and any connected IoT or home automation systems in the process.

▶ £25 / $20
▶ **magpi.cc/xDJDPp**

Energenie Pi-mote

Remote-control plug

A quick and easy way to do some home automation is to remotely control a mains socket with your Pi and some code! The Energenie Pi-mote allows you to do just that.

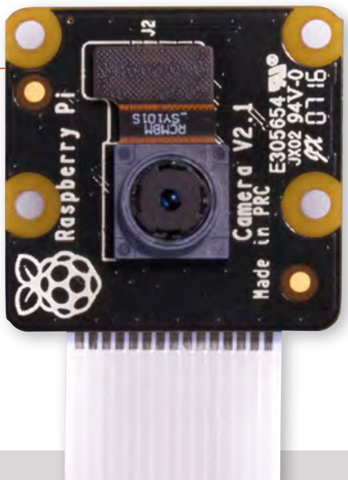
▶ £17 / $18
▶ **magpi.cc/FepLDV**

Pi NoIR Camera V2

See in the dark

If you're setting up a CCTV network in your home, or want a front-door camera that works 24 hours a day, then the IR version of the excellent Pi Camera Module is for you.

▶ £25 / $25
▶ **magpi.cc/ircamera**



Gravity Light Sensor

Ambient light control

Ambient light sensors are very common (you probably have one in your phone) and can be a good way to slowly bring up lights, as it gets dark outside, in a more natural way.

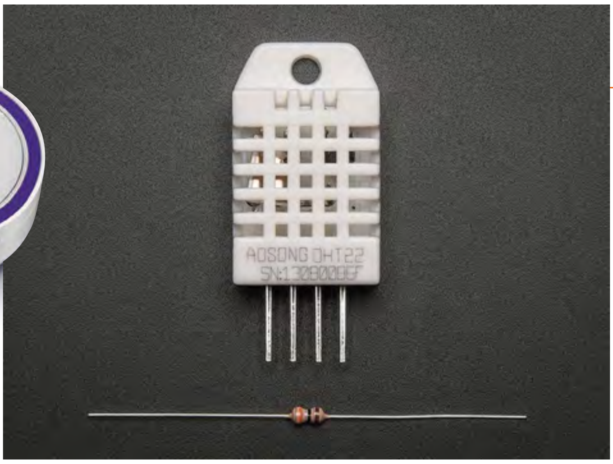
- £7 / $9
- **magpi.cc/ibSTjk**

DHT22 temperature-humidity sensor

Thermostats beware

With stuff like Nest around, it shouldn't be too surprising how many people use a Raspberry Pi and a temperature sensor such as this as a thermostat for their house. As an Astro Pi project demonstrated, you can use the humidity and temperature parts together to detect a person.

- £10 / $13
- **magpi.cc/PZrpfv**



Amazon Dash Button

Press for anything

A favourite among hackers is the Amazon Dash Button – you can easily use it as a remotely connected button that does what you'd like it to do. And pretty cheaply as well!

- £5 / $5
- **magpi.cc/SXmgpe**

SparkFun OpenPIR

Motion sensing

Want to trigger a camera recording? Or lights turning on? Or literally anything involving seeing if something has moved in a location? A PIR sensor like this will help.

- £15 / $16
- **magpi.cc/rcdvXL**

Philips Hue Lights

Controllable bulbs

An excellent solution for controlling your lights is to have remote-controllable light bulbs like the Hue! You can find our tutorial on how to control them with a Raspberry Pi in issue 61 (**magpi.cc/61**).

- Various
- **meethue.com**

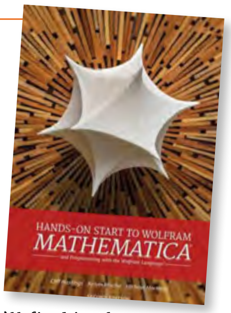
AUTOMATION SOFTWARE

While you can do a lot using Raspbian, there are dedicated home automation operating systems for the Pi that are already preconfigured. We like openHAB, which you can find here: **openhab.org**

Everything you need to learn Wolfram Language & Mathematica

Experiment with mathematics on a Raspberry Pi using Wolfram tech. By **Lucy Hattersley**

Hands-On Start to Wolfram Mathematica and Programming with the Wolfram Language



AUTHOR **Cliff Hastings, Kelvin Mischo, Michael Morrison**

Price:
£32 / $40

magpi.cc/oFxHFL

Make sure you start here. This hands-on book, written by three Wolfram Language authors, is packed with detailed examples for all the impressive things you can do with Mathematica.

It starts with simple input and output examples, but quickly moves on to visualising data with 2D and 3D graphics and performing advanced calculus, probability, and statistics. It even works its way up to parallel and GPU programs.

Now in its eleventh edition, the book is fully updated. Each chapter has detailed instructions, with examples for interactive learning and end-of-chapter exercises. It's packed with tips and advice on how to get the most from Mathematica.

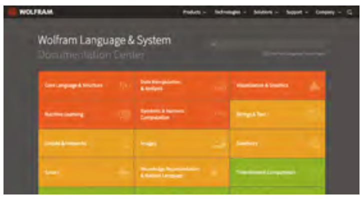
For a sneak peek, Chapter 7 is available from the Wolfram website (**magpi.cc/dsMnRO**). Titled 'Creating Interactive Models with a Single Command', this is a good example of the kind of content you'll find in the book. It outlines how to build plots with interactive sliders, buttons, and other controls.

Cliff Hastings also has a companion video that walks you through some of the concepts talked about in the book (**magpi.cc/XHPafc**).

This book is where most Mathematica users start out. While we don't think it's enough on its own, it should be your first point of call.

Websites

Make sure you bookmark these resources



WOLFRAM LANGUAGE FAST TUTORIAL

This tutorial aims to give you "what you need to read and understand almost any Wolfram Language code". It's a useful resource to have bookmarked.

▶ magpi.cc/fXisFe

WOLFRAM PROGRAMMING LAB

The Wolfram Programming lab has an incredible range of technical demonstrations and quirky uses for the language. It's a fun way to learn new ideas and discover new uses for Wolfram Language.

▶ lab.wolframcloud.com

DOCUMENTATION CENTER

The Wolfram Language & Systems Documentation Center has detailed outlines and examples of all the commands used..

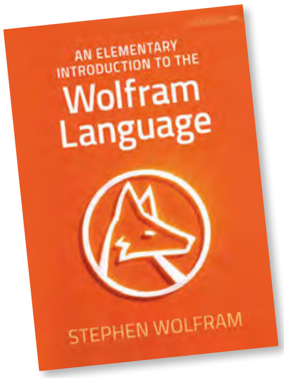
▶ reference.wolfram.com

An Elementary Introduction to the Wolfram Language

AUTHOR
Stephen Wolfram

Price:
£15 / $20

magpi.cc/gJxGni



Wolfram Language is very different from programming languages such as Python and C, and learning it can be challenging. So this book, written by Stephen Wolfram (the founder and CEO of Wolfram Research), proves very helpful.

It works more like a reference guide than a series of tutorials, with each couple of pages outlining a concept or feature of Wolfram Language. The whole programming language is too vast to be covered in a single book, so it doesn't feature everything. Instead, it's more like edited highlights of concepts you really need to know.

Like most Wolfram books, it starts out with elementary mathematics before moving on to functions lists and working with data. It also covers more advanced elements, such as natural language processing and machine learning – in not enough detail for you to get a complete understanding, but certainly enough to help you get started with each element.

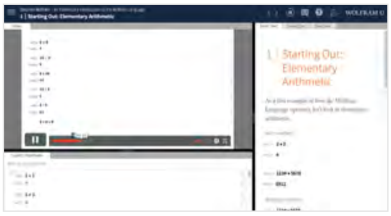
There are even some chapters about debugging and advanced coding concepts, which is far beyond what you get in most other Mathematica resources. Best of all, it's available online for free (**magpi.cc/RdGsQV**).

Wolfram U

AUTHOR
Wolfram

Price:
Free

magpi.cc/YanBnX



Reading books is all well and good, but in order to get a full understanding of Mathematica and Wolfram Language, it's better to follow a course.

Wolfram U is packed with training lectures and interactive courses that cover the language. Your first stop should be the full interactive course for 'An Elementary Introduction to the Wolfram Language' that accompanies Stephen Wolfram's book (**magpi.cc/YdMomr**).

You get to watch a video demonstration of each skill, while the book text is displayed by the side and a Scratch Notebook enables you to work on code as the course material is delivered.

There's a huge range of courses, covering a wide range of mathematical areas, such as Data Science & Statistics, Image & Signal Processing, Modelling & Simulation, and Finance. Some courses are more detailed than others, and many are just video demonstrations. But they all offer an interesting take on using Wolfram Language and Mathematica, and help you move further along than just reading books. There are also webinars and live events where you can meet up with other Wolfram learners.

Community

Get help from other Wolfram users in the community

WOLFRAM COMMUNITY

The official Wolfram forum has official staff picks and announcements. Plus a vibrant community that's on hand to answer questions.

▶ community.wolfram.com

STEPHEN WOLFRAM'S LIVESTREAMS

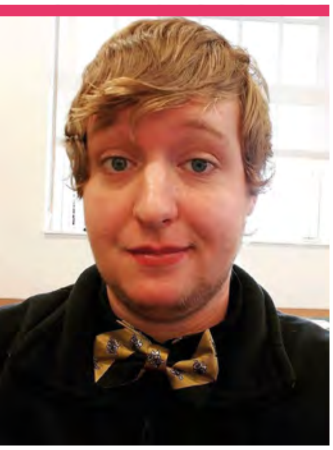
The man himself frequently chats about various Wolfram endeavours on his live-stream blog. And there's a whole back catalogue of historical video streams.

▶ magpi.cc/QhyXuf

MATHEMATICA STACKEXCHANGE

Because Wolfram is so heavily invested in developing training resources, almost all of your time is spent using official resources. So don't forget to bookmark the StackExchange so you have at least one community resource elsewhere. It's a great community, too.

▶ magpi.cc/NvKfac



Brett White

In the state of West Virginia, Project 76 is trying to improve computer science education, thanks to its organiser Brett White

> Category **Educator** | > Day job **Engineer**

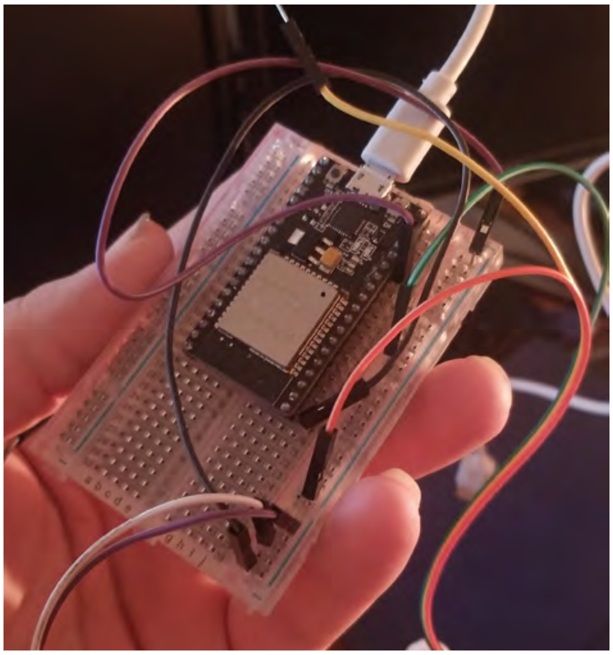
Whenever a Picademy is put on in America, people from all around the US come to attend the course. One of these new Raspberry Pi Certified Educators, Brett White, is now trying to use his new skills to help at home.

"Computer Science in my home state of West Virginia is pretty lacklustre," Brett tells us. "My high school didn't start a computer programming class until my senior year of high school (which was in 2014). While that was a first for my high school, they did not get more advanced than that, and that class I was in taught Java, which in my opinion is not a great way to get started. Middle school courses for easy coding barely exist … and there are few tech jobs in the area as a result, and the demand for it is definitely not where it should be."

Brett created Project 76 to help change this.

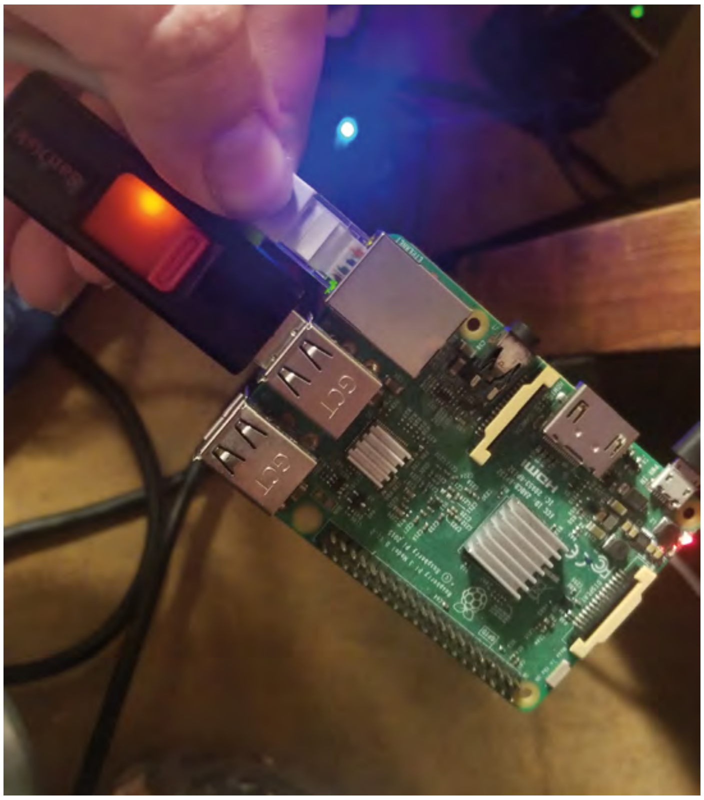
What is Project 76?
Project 76 (named after Bethesda's Fallout 76, which is primarily based in West Virginia) is my goal to:

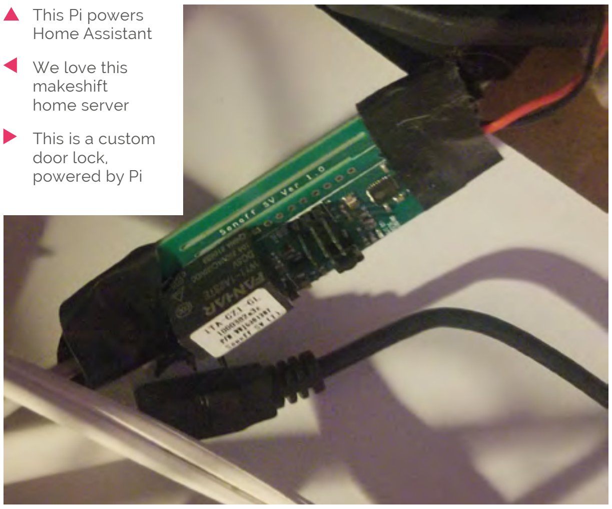
- Get more CS Classes introduced in high schools, with beginner courses starting in middle schools in West Virginia.
- Start Code Clubs in West Virginia High Schools so that students can have a chance to do extra-curricular projects,



▲ One of many ESP8266-based devices running via MQTT

◄ This is a multi-sensory node for Brett's house





▲ This Pi powers Home Assistant

◀ We love this makeshift home server

▶ This is a custom door lock, powered by Pi

"Middle school courses for easy coding barely exist, and there are few tech jobs in the area as a result"

and to bring these projects to my Statewide Raspberry Jams.

- Start area CoderDojos in West Virginia for anybody interested in projects (whether in school or not) to come down and learn how to do programming, and have CoderDojos adopted possibly by Universities in our area to support in-state education.
- Have teachers, librarians, etc. interested in getting Raspberry Pi Certifications, so we can have more certified teachers around the state to teach classes and do Code Clubs.
- Have access to Raspberry Pi and Arduino in middle, high schools, and higher education institutions, so students can have access to technology.
- Eventually to [encourage] interested tech companies to set up bases in West Virginia, because we have a lot of locations open, and it would really help our state to even evolve it, such as having internet access to many rural areas in our state.
- To build a makerspace in main cities (such as Charleston, Fairmont, Parkersburg), so anyone can use our facilities and hold CoderDojo classes and mini Raspberry Jams there as well.

What have you made yourself with a Raspberry Pi?

My personal projects involve mostly smart home. I have three Raspberry Pis in my house, running everything, such as a Raspberry Pi running Home Assistant, with additional add-ons for an MQTT broker, Node-RED, and various controller add-ons. I've also got a Raspberry Pi 3B+ running DietPi, running services such as a VPN, Plex Media Server, BitTorrent server, web front-end, and my motionEye security camera software. There's also [another] Raspberry Pi 3 running DietPi, running other services like OctoPrint (for my 3D printer), ownCloud, UrBackup, and Resilio Sync, with a desktop interface powered by Raspbian PIXEL.

Picademy USA

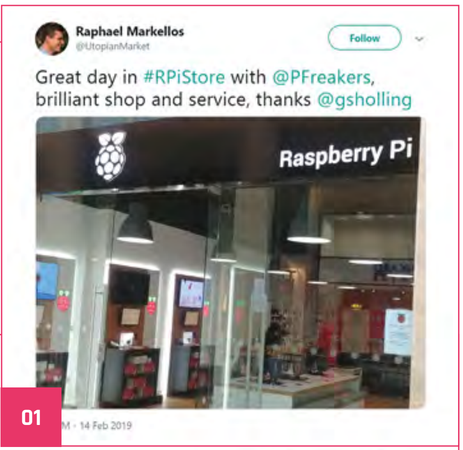
Want to become a Raspberry Pi Certified Educator in North America? There are several events already planned for this year, and you can find out more about them here: **magpi.cc/CNKtBm**

This Month in **Raspberry Pi**

Raspberry Pi **Store!**

Members of the community have been visiting the new store. Here are some pics!

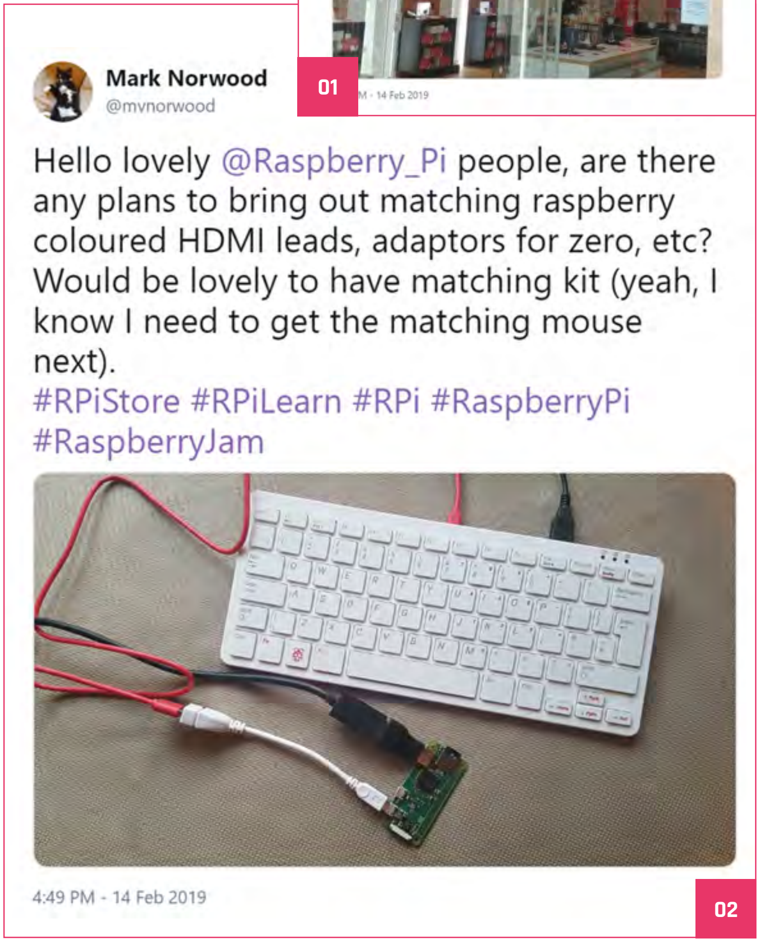
If you didn't already see it on page 6, there is now a Raspberry Pi Store in Cambridge! As we send this mag to the printers, it's only been open for a week or so now, but that hasn't stopped loads of community members from making the journey down to it, as you can see from their amazing photos…


Raphael Markellos
@UtopianMarket
Follow

Great day in #RPiStore with @PFreakers, brilliant shop and service, thanks @gsholling

M - 14 Feb 2019

01


Mark Norwood
@mvnorwood

Hello lovely @Raspberry_Pi people, are there any plans to bring out matching raspberry coloured HDMI leads, adaptors for zero, etc? Would be lovely to have matching kit (yeah, I know I need to get the matching mouse next).
#RPiStore #RPiLearn #RPi #RaspberryPi #RaspberryJam

4:49 PM - 14 Feb 2019

02

01. The façade is very minimalist

02. The official keyboard and mouse have been a huge hit at the store

03. You can buy books and *The MagPi*!

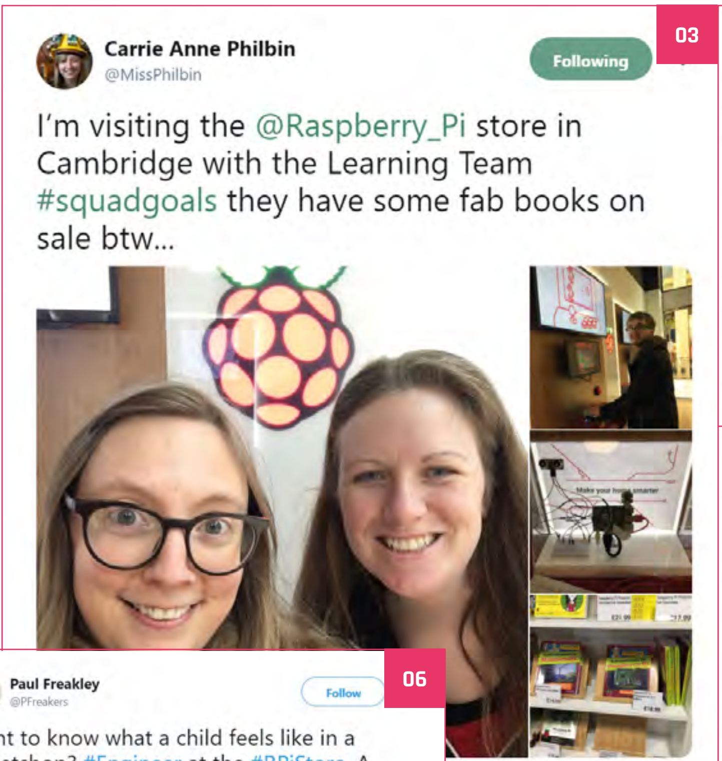
04. The Babbage bears are very popular

05. Roger, Principal Hardware Engineer at Raspberry Pi, gets a look around the store

06. Kid in a candy store!

07. Gordon, Director of Software Engineering at Raspberry Pi, has been at the store a lot

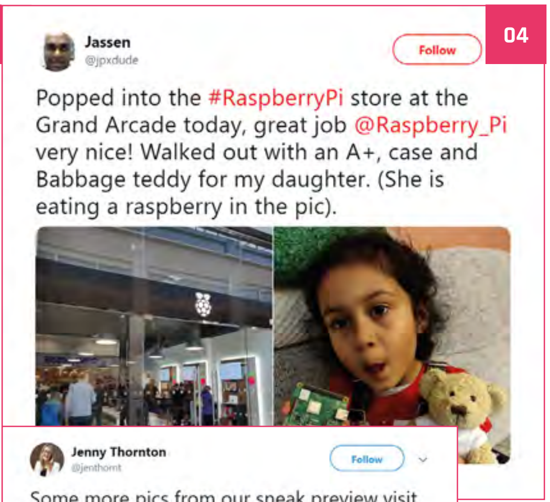
08. Meet members of the Raspberry Pi team



03

Carrie Anne Philbin @MissPhilbin Following

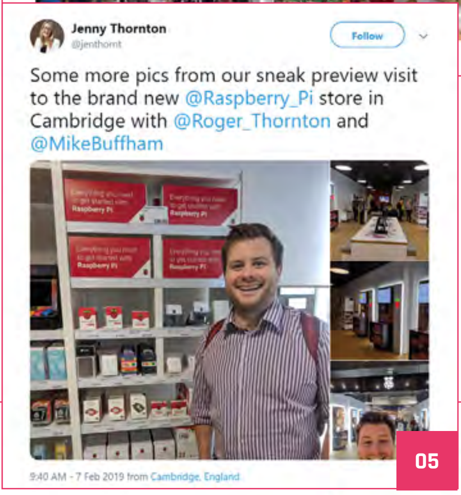
I'm visiting the @Raspberry_Pi store in Cambridge with the Learning Team #squadgoals they have some fab books on sale btw…



04

Jassen @jpxdude Follow

Popped into the #RaspberryPi store at the Grand Arcade today, great job @Raspberry_Pi very nice! Walked out with an A+, case and Babbage teddy for my daughter. (She is eating a raspberry in the pic).

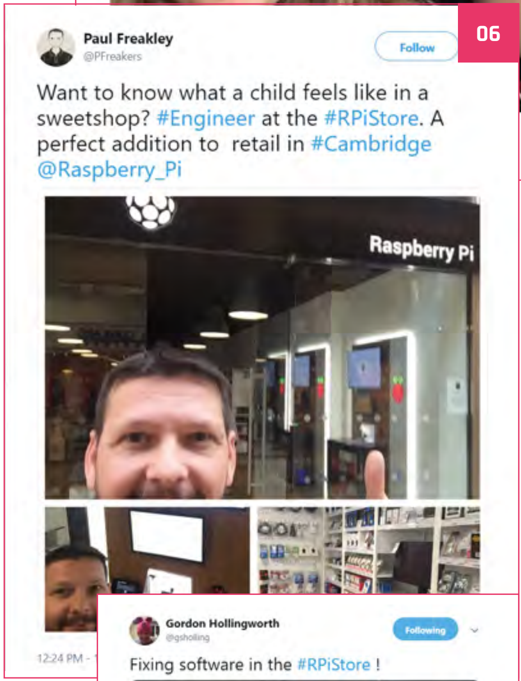


05

Jenny Thornton @jenthornt Follow

Some more pics from our sneak preview visit to the brand new @Raspberry_Pi store in Cambridge with @Roger_Thornton and @MikeBuffham

9:40 AM - 7 Feb 2019 from Cambridge, England

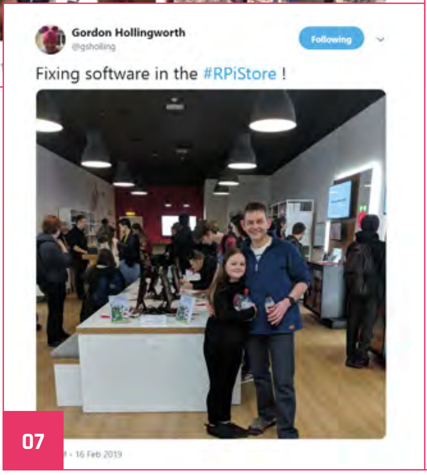


06

Paul Freakley @PFreakers Follow

Want to know what a child feels like in a sweetshop? #Engineer at the #RPiStore. A perfect addition to retail in #Cambridge @Raspberry_Pi

12:24 PM -

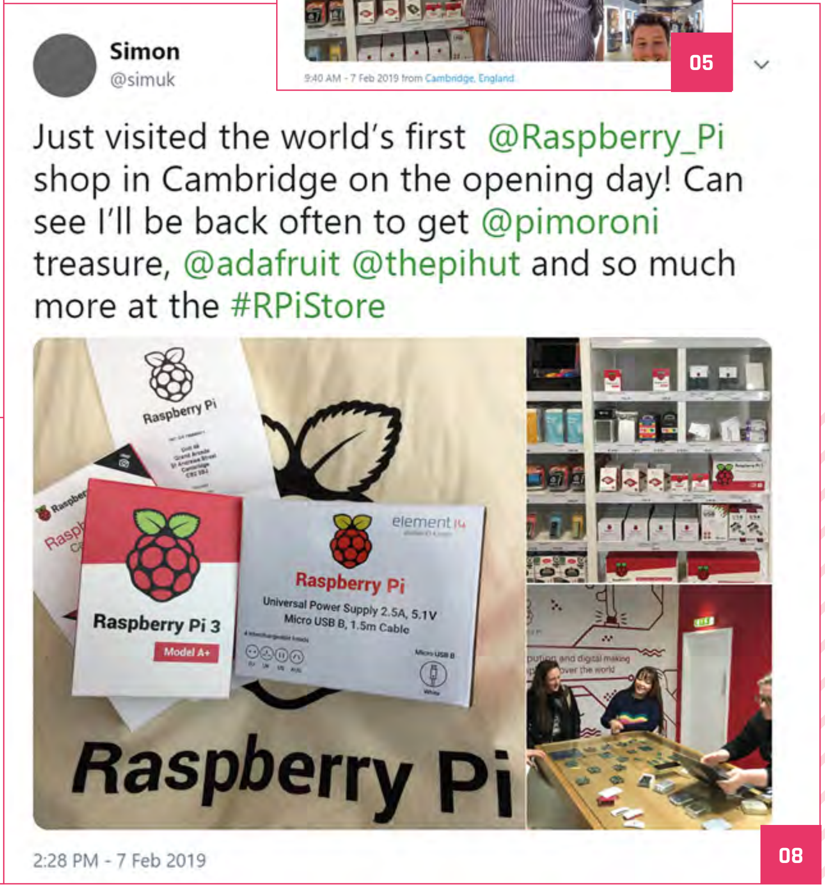


07

Gordon Hollingworth @gsholling Following

Fixing software in the #RPiStore !

- 16 Feb 2019



08

Simon @simuk

Just visited the world's first @Raspberry_Pi shop in Cambridge on the opening day! Can see I'll be back often to get @pimoroni treasure, @adafruit @thepihut and so much more at the #RPiStore

2:28 PM - 7 Feb 2019

Pi videos!

Videos from us and Raspberry Pi that you may have missed

Did you know you can find us on YouTube? It's true, we put out fun tutorial and preview videos of Pi kits and projects. Find us here: **magpi.cc/youtube**. Raspberry Pi itself also has a channel full of excellent videos, which you can find here: **magpi.cc/DXRHRQ**.

Both channels are packed with great videos – why don't you check them out?

01. Meet the Raspberry Pi 3A+

Still unsure of what the Raspberry Pi 3A+ is all about? Let us show you in this video.
magpi.cc/rrFQBu

02. Raspberry Pi Store

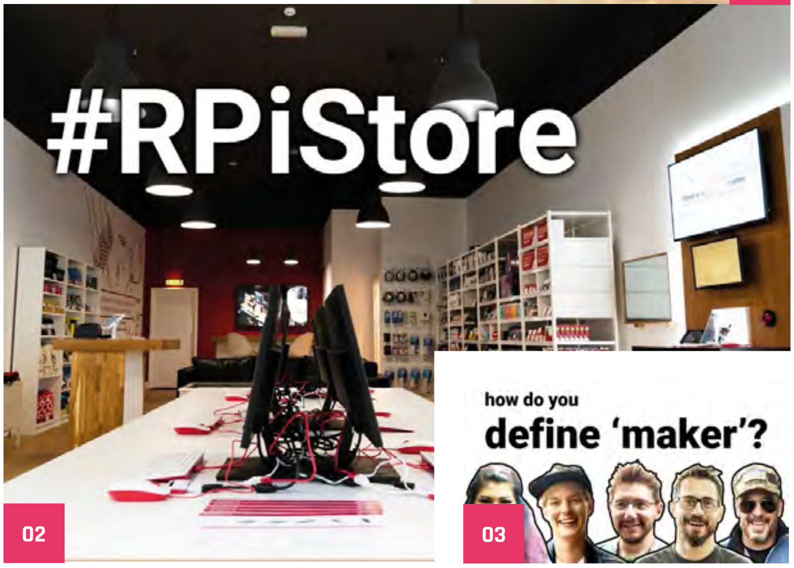
Get a look around the Raspberry Pi store from the comfort of your own home. Be warned, though: it might make you want to make the trek to Cambridge to visit it.
magpi.cc/uidcEB

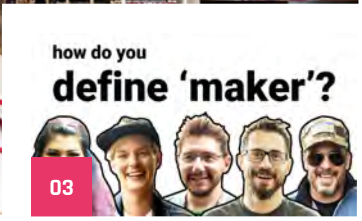
03. Defining 'maker'

We get caught up in labels and gatekeeping a lot in modern society, so Raspberry Pi talks to some makers about how they became 'makers', and what that means to them.
magpi.cc/jDiZdP

04. Jenni Sidey inspires young women in science

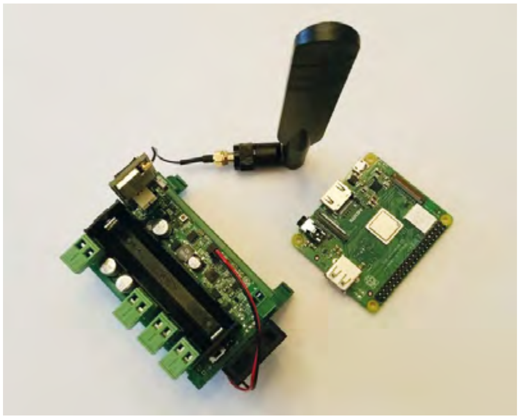
To celebrate International Day of Women and Girls in Science, Raspberry Pi interviewed Canadian astronaut Jenni Sidey, and asked why she thinks STEAM skills are important.
magpi.cc/pDNgER





Crowdfund this!

Raspberry Pi projects you can crowdfund this month



Pi18650 Smart UPS

This add-on started life as a simple UPS/battery pack for the Pi, but has since evolved far past that to include different kinds of environmental sensors, a microcontroller, LEDs, pinouts for other things to add to it, and many other features. Give it a look!

▶ **kck.st/2Wqkg06**

Picoh

The Picoh is a newer, smaller version of the Ohbot we featured in the magazine way back in issue 66 (**magpi.cc/66**). Instead of a fully mechanical face, the Picoh has a digital face that is a little more friendly than the original.

▶ **kck.st/2R1WcwN**

CROWDFUNDING A PI PROJECT?

If you've launched an irresistible Pi-related project, let us know!

magpi@raspberrypi.org

Best of the rest!

Here are some other great things we saw this month

CYBERPUNK LAPTOP

2019 seems to be the year of cyberpunk aesthetics, and we're here for it. We love this laptop design that is camouflaged in an old mah-jong box, with colourful keys and exposed PCBs. It's wonderful.

▶ **magpi.cc/CCJppK**

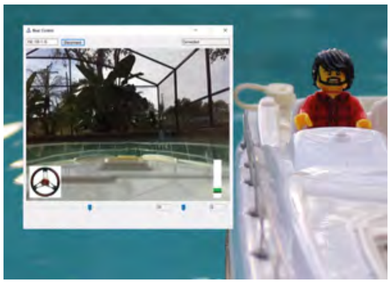
SELF-DRIVING CAR

We're not here to answer the question of whether self-driving cars are the future, but we do love seeing Pi-robot-sized automatons that are programmed like self-driving cars. Head to the link for a fun little video of this guy tootling around the track.

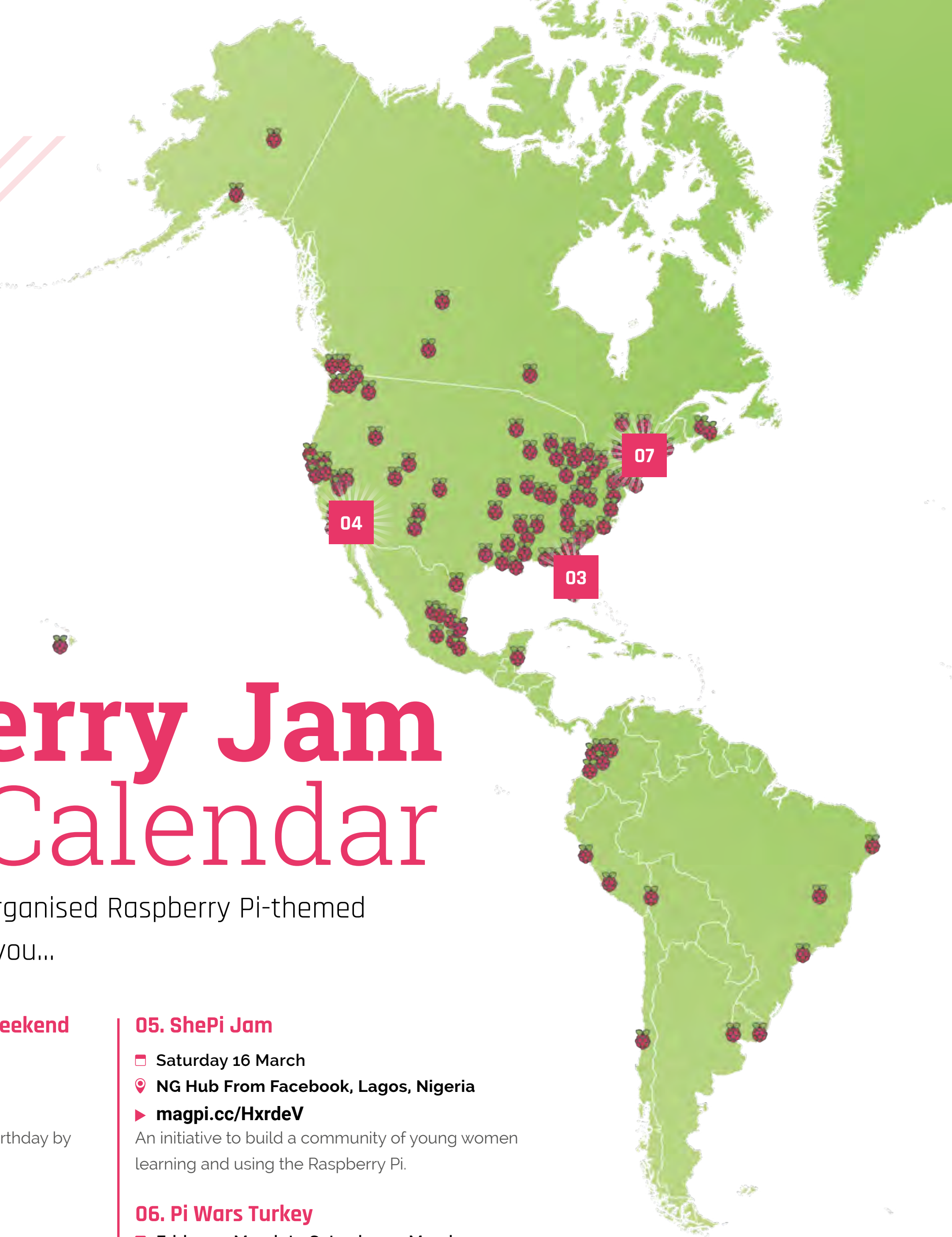
▶ **magpi.cc/vjDimi**

PI ON THE WATER

A remote, Pi-controlled boat with camera! We love this, although we assume the range on something like this is dependent on wireless LAN signal strength and/or the length of a cable. Still, an amazing project.



▶ **magpi.cc/ndipGc**



Raspberry Jam Event Calendar

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

01. Raspberry Jam Big Birthday Weekend

Saturday 2 March to Sunday 3 March
Earth, Solar System, Milky Way
▶ **magpi.cc/CrMUru**

Get ready to celebrate the Raspberry Pi's birthday by going to your local Jam!

02. Inverell Raspberry Jam

Monday 4 March
TAFE NSW – Inverell Campus, Inverell, Australia
▶ **magpi.cc/inLLWD**

A new weekly Raspberry Jam in Australia, where you can learn about Pi or get help with yours!

03. Satellite Beach Library Pi Jam

Saturday 9 March
Satellite Beach Library, Satellite Beach, FL, USA
▶ **magpi.cc/mrZtTT**

Join this family-friendly, STEAM-powered celebration of making, coding, creativity, play, and more!

04. Fresno Ideaworks' Raspberry Jam

Thursday 14 March
Fresno Ideaworks, Fresno, CA, USA
▶ **magpi.cc/vMCURi**

Projects will be on display from community members, and there are also beginner workshops.

05. ShePi Jam

Saturday 16 March
NG Hub From Facebook, Lagos, Nigeria
▶ **magpi.cc/HxrdeV**

An initiative to build a community of young women learning and using the Raspberry Pi.

06. Pi Wars Turkey

Friday 22 March to Saturday 23 March
Hisar School, Istanbul, Turkey
▶ **piwars.hisarcs.com**

Pi Wars will soon be here in the UK, but first you can head over to Istanbul for their robot challenges.

07. Patriot Pi Raspberry Jam

Saturday 30 March
Cherry Valley-Springfield Central School, NY, USA
▶ **magpi.cc/ZicaQc**

Learn and share about the Raspberry Pi, including the huge variety of projects and learning opportunities.

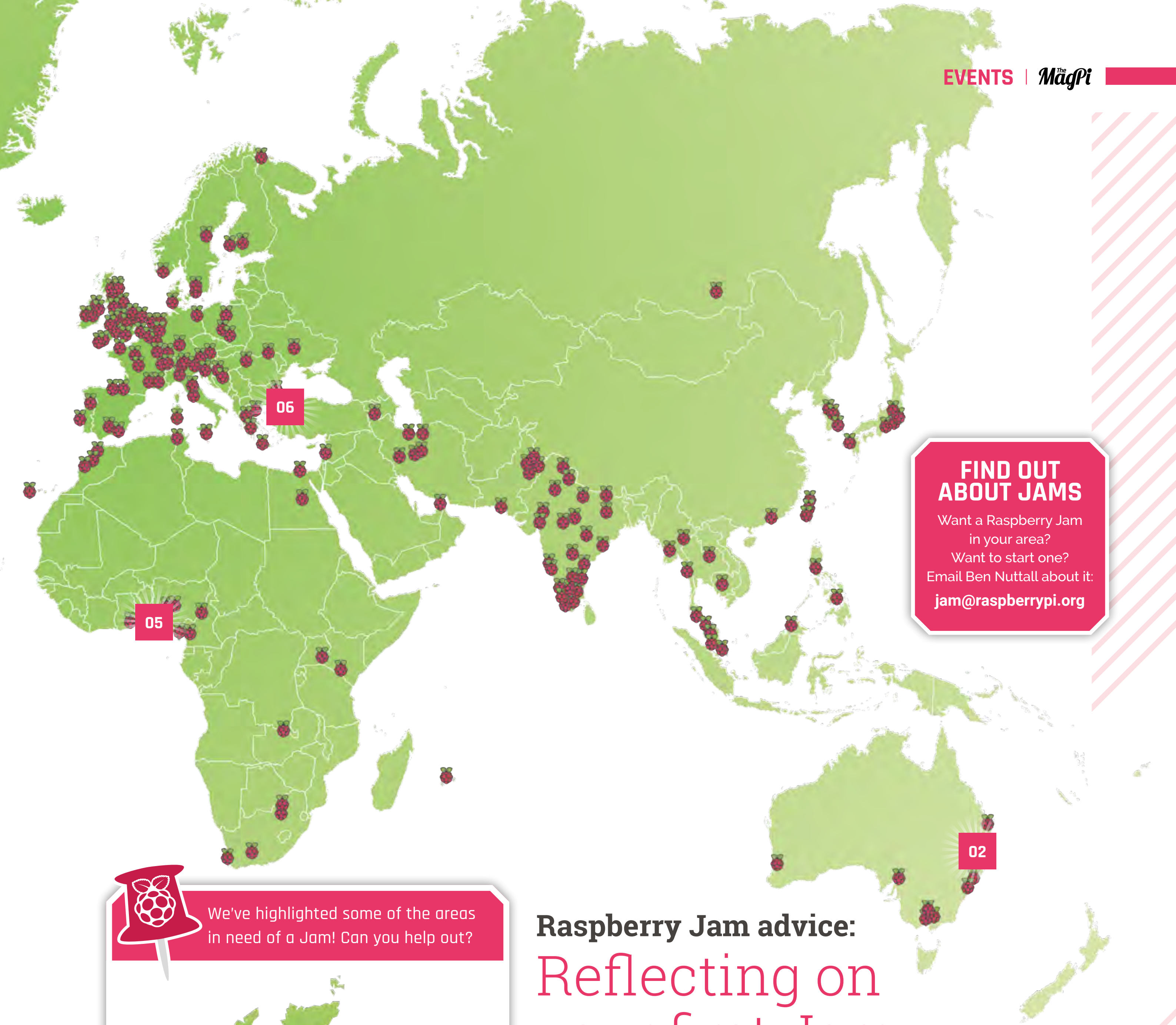
08. Stafford Raspberry Jam

Tuesday 12 March
Stafford Library, Stafford, UK
▶ **magpi.cc/iVXQyw**

A big meet-up where people of all ages get together to share ideas, help each other, and have fun!

FULL CALENDAR

Get a full list of upcoming events for March and beyond here:
rpf.io/jam



FIND OUT ABOUT JAMS

Want a Raspberry Jam in your area? Want to start one? Email Ben Nuttall about it:

jam@raspberrypi.org

We've highlighted some of the areas in need of a Jam! Can you help out?

Raspberry Jam advice: Reflecting on your first Jam

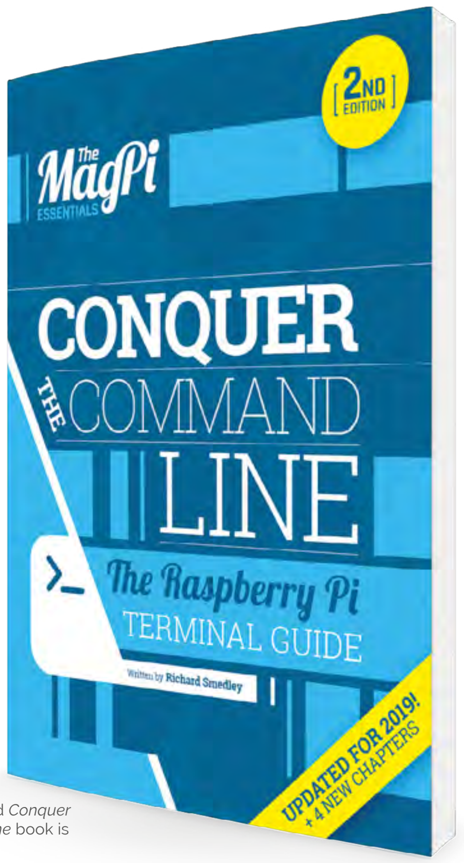
"When I ran my first Jam, I thought of it as being a meet-up for techie adults, but I decided to run the first one on a Saturday. When kids and families turned up, I realised this was a better format than I had had in mind. It was great to see people of all ages enthused by technology, so we kept the free-form practical format, and it worked really well."

Ben Nuttall – Manchester Raspberry Jam

Every Raspberry Jam is entitled to apply for a Jam starter kit, which includes magazine issues, printed worksheets, stickers, flyers, and more. Get the book here: **magpi.cc/2q9DHfQ**

Your Letters

1st


The MagPi ESSENTIALS
2ND EDITION
CONQUER THE COMMAND LINE
The Raspberry Pi TERMINAL GUIDE
Written by Richard Smedley
UPDATED FOR 2019! + 4 NEW CHAPTERS

Our fully updated *Conquer the Command Line* book is now available!

Command line and conquer

I've been looking at getting a print version of the *Conquer the Command Line* Essentials book. However, it looks like it's been out of stock on your store for a little while now. I know I can grab the PDF, and I have been using that, but I would really like to get the physical version as well. Any idea when this will be back in stock?

Mark N via Facebook

The reason the book was out of stock for a little while was because we were putting the final touches to our updated second edition of *Conquer the Command Line*, which we can happily announce is out now! You can buy it today from the Raspberry Pi Press store – find it here: **magpi.cc/gMkZAn**.

Contact us!

- Twitter **@TheMagPi**
- Facebook **magpi.cc/facebook**
- Email **magpi@raspberrypi.org**
- Online **raspberrypi.org/forums**

Download issues

Just letting you know that I get an error when trying to download *The MagPi* #64 via the Download button on the issue page. I've tried over months, and with Firefox and Edge browsers on Windows 10, and with the same error of a secure connection failure.

Also, *The MagPi* #16 fails to complete downloading. It gets to about 20% then fails. Again, this is on the issue page on your site.

I've not had any issues with any other download links that I've used on your website.

Robert G via email

We do sometimes get readers like yourself emailing about issue download problems. In this case, in all our tests the PDF downloads were working fine at our end. If this is happening to you, using a different browser or clearing your cache will usually sort it out and let you download it.

In Robert's case, we were not able to find a quick solution – they were fine our end and it could be any number of problems on his. So, we emailed him the PDFs. If you're also having some download issues, don't hesitate to contact us, and we'll try to figure it out!

You can download free PDFs of *The MagPi* at **magpi.cc/issues**

Older articles

I've been looking at the 'Program an Arduino Uno with your Raspberry Pi' article (magpi.cc/qvvCUY).
I tried using Chromium, downloaded, unpacked, and installed for a part of it, but that failed. So I tried:

```
cd Downloads/
tar -xf arduino-1.8.8-linuxarm.tar.xz
sudo mv arduino-1.8.8 /opt
sudo /opt/arduino-1.8.8/install.sh
```

But that also failed, saying:

```
  Adding desktop shortcut, menu item and file associations for Arduino IDE…touch: cannot
touch '/root/.local/share/applications/mimeapps.list': No such file or directory
  /usr/bin/xdg-mime: 803: /usr/bin/xdg-mime: cannot create /root/.local/share/
applications/mimeapps.list.new: Directory nonexistent
```

How do i install version 1.8.8 on my Raspberry Pi Model 3B+?

David H via email

It looks like you've run into one of the unfortunate realities of tech tutorials: sometimes they go out of date. As technologies change and software gets updated, eventually something that did work might not.

For this specific tutorial, you can achieve some success by creating the applications folder so the install can continue. What we recommend, though, in cases like this (which are thankfully rare in our tutorials!) is to visit the Raspberry Pi forums (**rpf.io/forums**) and the friendly folks on there can give better help on sorting it out.

Pop down the Pi shop

What is this I hear about a new Raspberry Pi Store? Where is it? When can I go to it? Are you opening more? I have so many questions – I just would really love to go to an actual Raspberry Pi store!

Charley N via Twitter

There is indeed a brand new Raspberry Pi Store, and you can read more about it on page 6 (and see more pictures in This Month in Raspberry Pi on page 86). It was opened in Cambridge, the home town of the Raspberry Pi, and is currently the only one in the world! We have absolutely zero idea if or where more will open, but for the moment it's a real-life store in Cambridge that is open seven days a week.

Visit the first ever Raspberry Pi store in the Grand Arcade, Cambridge

WIN One of Ten Official Raspberry Pi mice and keyboards

They're finally here! The Official Raspberry Pi Keyboard & Hub and the Official Raspberry Pi Mouse are amazing bits of kit that really help you with your computing learning.

Head here to enter: magpi.cc/win | **Learn more:** On page 11

Terms & Conditions

Competition opens on **27 February 2019** and closes on **27 March 2019**. Prize is offered to participants worldwide aged 13 or over, except employees of the Raspberry Pi Foundation, the prize supplier, their families or friends. Winners will be notified by email no more than 30 days after the competition closes. 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. 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. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram or Facebook.

HAM RADIO PROJECTS

Power up amateur radio projects with Raspberry Pi

THE MAGPI #80
ON SALE 28 MAR

Plus!

- Control a Lego Boost kit
- Build a Comic Book Creator
- Share files with Samba
- The best Raspberry Pi tips
- Turn lights on with a 'Clapper'

DON'T MISS OUT!
magpi.cc/subscribe

TWITTER	@TheMagPi
FACEBOOK	fb.com/MagPiMagazine
EMAIL	magpi@raspberrypi.org

EDITORIAL
Editor
Lucy Hattersley
lucy@raspberrypi.org

Features Editor
Rob Zwetsloot
rob.zwetsloot@raspberrypi.org

Sub Editors
Phil King and Nicola King

DESIGN
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Mike Kay, Harriet Knight

Illustrator
Sam Alder

CONTRIBUTORS
Mike Cook, David Crookes, PJ Evans, Gareth Halfacree, Rosie Hattersley, Nicola King, Laura Sach, Mark Vanstone

PUBLISHING
Publishing Director
Russell Barnes
russell@raspberrypi.org
+44 (0)7904 766523

Director of Communications
Liz Upton

CEO
Eben Upton

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

SUBSCRIPTIONS
Raspberry Pi Press
Mann Enterprises, Unit E,
Brocks Business Centre,
Haverhill, CB9 8QP

To subscribe
magpi.cc/subscribe

To get help:
rpipresshelp@raspberrypi.org

Helping the next generation

Laura Sach on guizero, and why you should always work with children

Creating a Google map pinpointing locations where dog poo has been found around Nottingham is not something I ever thought I would do professionally. However, I have the good fortune of working for the Raspberry Pi Foundation and, so far, my day job has legitimately involved persuading colleagues to run around Cambridge avoiding virtual zombies with a GPS-enabled smartphone, giving myself a JavaScript-enabled quiff, and modding my glasses to include a Raspberry Pi Zero time-lapse camera. Standard.

Writing educational resources might seem straightforward – if you already know how to do the thing, then just write down the steps, right? I mean, sure, anyone can write down some instructions, but it's very easy to accidentally make your third instruction (to paraphrase a popular meme) "draw the rest of the friggin' owl".

Even if you think you've got everything just right, the first time you test your lovingly created instructions on some real kids, you'll realise that, far from the careful following of the instructions that you had in mind, your game graphics will be replaced with bowls of cheesy puffs, you'll have opera singers screeching every five seconds, and the kids will be collapsed in the corner in a fit of giggles. And all the better for it – kids are amazingly good at cheering you up on even the worst of days, and their imagination and creativity will always astound me.

Anyone who runs a Raspberry Jam or otherwise works with kids informally, I salute you. I've got a piece of paper from a university which says I can do this stuff, but lots of you just wing it because you love what you do and want to share it.

> "Anyone who runs a Raspberry Jam or otherwise works with kids informally, I salute you"

Sharing knowledge

The Raspberry Pi community is very welcoming to learners of all ages. I enjoy sharing knowledge in ways it can be accessed and consumed freely, and I love that I can do this whilst also earning myself a living. The project I am most proud of is the guizero Python library, which simplifies Tkinter, so kids can easily create graphical user interfaces (GUIs). This was not straightforward – in fact, it is the first Python library I have ever written – but it has had more than 100,000 downloads.

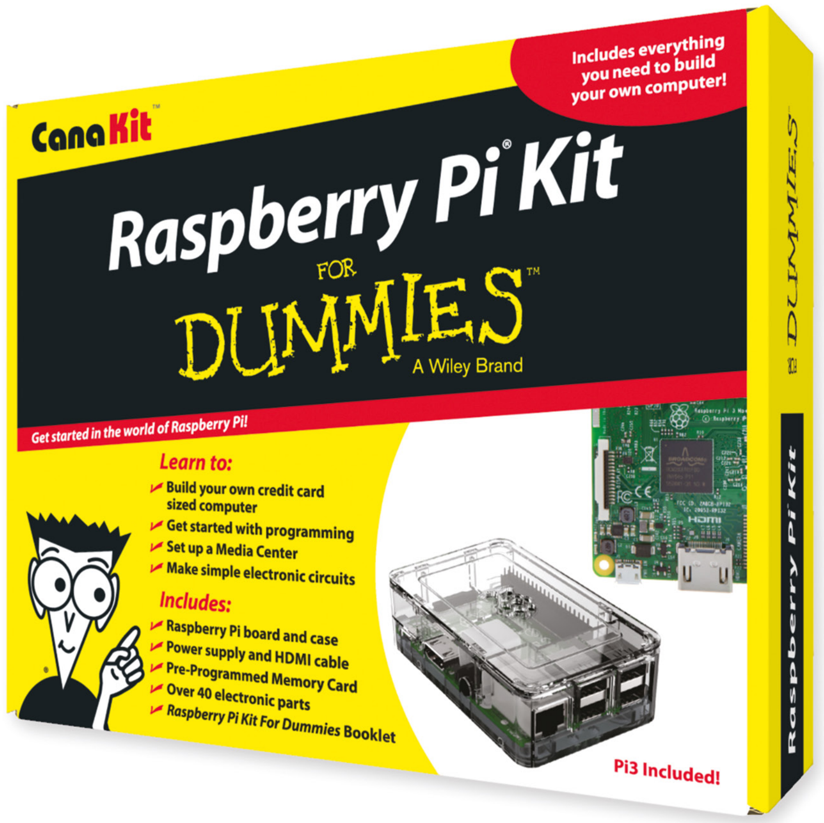
One surprise for me was that it doesn't seem to just be kids who use guizero. I originally envisaged it being useful for young people, but I've seen all kinds of uses, a popular one being basic GUIs controlling LEDs and other physical components.

Having never written a library before, it was a bit nerve-racking to put up my code in public and shout "hey everyone, come and have a look". I was a bit worried someone might come and tell me how awful it was, but actually lots of community members turned up and made helpful comments, contributed code, and now Martin O'Hanlon does more work for the library than I do!

There are many other -zero libraries, so if you've been considering simplifying some software for the community, I'd say go for it – you'll learn a lot and have fun on the way.

AUTHOR
Laura Sach
Raspberry Pi Senior Learning Manager, creator of guizero, and casual Python wrangler.
lawsie.github.io/guizero

CanaKit™


Includes everything you need to build your own computer!
CanaKit
Raspberry Pi® Kit
FOR DUMMIES™
A Wiley Brand
Get started in the world of Raspberry Pi!
Learn to:
Build your own credit card sized computer
Get started with programming
Set up a Media Center
Make simple electronic circuits
Includes:
Raspberry Pi board and case
Power supply and HDMI cable
Pre-Programmed Memory Card
Over 40 electronic parts
Raspberry Pi Kit For Dummies Booklet
Pi3 Included!
Raspberry Pi Kit

Kit Includes:

- *Raspberry Pi For Dummies* Booklet
- Raspberry Pi 3 Board
- Memory Card
- Plastic Case
- 2.5A Power Supply
- HDMI Cable
- Resistors
- LEDs
- Push Button Switches
- Prototyping Breadboard
- Jumper Wires
- Heat Sinks

FOR DUMMIES® A Wiley Brand

Available for worldwide shipping at:

WWW.CANAKIT.COM

Available in Europe through RS Components

RS

$89.99 US DOLLARS **£69.99** EXCLUDING VAT

ELECTRONIC KITS • ELECTRONIC PARTS • RASPBERRY PI • ARDUINO